DEPARTMENT OF MECHANICAL & AEROSPACE ENGINEERING

University of Strathclyde Engineering

Model-Based Optimal Control of An Underground Heating System

Erin Brown **201919578**
Kaloyan Domuschiev **201937542**
Malcolm Irving-Robertson **201927466**

Supervisor: Eero Immonen

A thesis submitted in partial fulfilment of the requirements of the MEng Aero-Mechanical Engineering Course

8th January 2024

Word Count: 21,494

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Executive Summary

Introduction and Background

Countries with colder climates experience snow cover for several months out of the year. Underground heating is a robust solution allowing football and other sports pitches to maintain adequate playing conditions year-round. However, energy costs are at an all-time high, so it is more important than ever to be energy efficient, without compromising the results of snowmelt.

This report outlines the work and findings of the Aero-Mechanical Engineering 5th Year Master's Group Project titled "Model-Based Optimal Control of An Underground Heating System". The work undertaken further refined an existing dynamical snow melt model first developed in the 2023 Computational Engineering and Analysis (COMEA) paper, "A computational model for underground heating control of outdoor sports fields in winter conditions", [1], with a more detailed description of the physics of snow and the heat and mass transfer as it melts. It aimed not only to refine the dynamical snow melting model based on the COMEA approach, but also to establish a computational methodology for optimising the operation of an underfloor heating system for a 7-day weather forecast.

Methodology

Refinement of Computational Model

To improve the heating system computational model's ability to accurately predict snow depth, a more detailed description of the physics of snow, and the heat and mass transfer processes was developed. Consideration of the effects that underground moisture transfer has on the system took place through an extensive fluid study. 4 iterations of moisture simulations were created with increasingly realistic attributes, using Ansys Fluent. At each iteration, changes to geometry, boundary conditions or patching conditions took the simulation a step closer to a realistic depiction of moisture transport.

The time delay of the water-propyleneglycol fluid within the underground pipe network was studied, and the effect on the overall heat transfer model was determined. Ambient temperature, mass flow rate of the solution through the network, and temperature difference induced by the pump were all investigated as part of this study and were found to have significant implications on the imbalance in temperature distribution, with sections furthest away from the inlet taking substantially longer to reach the desired temperature for melting of the accumulated snow.

A detailed comparison of the 2007 Liu et. al paper [2] was included, which discussed the differences between both computational models. Some differences discussed were incorporated into the newly improved model, such as the inclusion of the effects of evaporation and condensation within the snowpack.

Sensitivity Analysis

Sensitivity analyses were conducted, first studying the impact different variables had on the model's prediction of snow depth. These were selected on the merit of being more obscure parameters, which are often difficult to measure in real life, such as radiation from surface emissivity. This allowed the importance of each variable to the prediction to be reviewed.

Secondly, alternative numerical schemes such as implicit and predictor-corrector were trialled, to determine if they would produce more accurate predictions of snow depth compared to the existing explicit scheme.

Lastly, the effect of different time step sizes was tested for improving the accuracy of the model whilst still retaining acceptable computational times. Several cases were considered with values both bigger and smaller than the original time step of 60 seconds. The results were compared to determine the most suitable configuration.

Design of Experiments

To begin the process of establishing optimal system control conditions for a given weather forecast, a Design-of-Experiments was conducted through Latin Hypercube Sampling (LHS). From this, randomly generated weather data and control inputs were created. The computational model ran for a forecast of 7 days, and the final snow depth and energy use for each scenario were recorded, assuming the weather and control inputs were constant over this time. This process was repeated for N number of weeks which allowed for large samples of weather data to be created which were used to replicate real life weather trends. The results the LHS produced were used in a regression decision tree model to determine the most suitable control inputs for different weather situations and trends. The purpose of conducting this was to achieve optimal playing conditions of snow depth ≤ 1mm whilst minimising energy consumption. LHS was again used to generate a set of weather and control inputs for the complicated machine learning algorithm development.

Decision Tree

The development of the decision tree algorithm was carried out in2 stages. Initially, the algorithm was trained on the data produced in the Design-of-Experiments and used to predict whether the randomly sampled control inputs would achieve playing conditions after 1 week of operation. This was followed by the inclusion of a more complicated machine learning approach, which involved training a new decision tree with only scenarios that would generate optimal conditions. It was then tasked with predicting the control inputs required for a new set of weather scenarios. This progressed the accuracy of the optimal controls to more refined and specific weather conditions beyond the threshold of the initial decision tree.

Results and Discussion

Refinement of Computational Model

Snow depth was predicted from the computational model and compared with real life weather data provided by the Finnish Meteorological Institute. The aim was to refine and improve the computational model in a way which predicted the snow depth with high accuracy and high precision.

The results of the moisture study found that from the final iteration of simulations, the movement of moisture in soil under the pitch was insignificant and therefore the effect on thermal properties could be deemed negligible. The results of the final iteration were recorded and an example of the area-weighted average temperature [K] under the pitch is shown below in Figure 1.

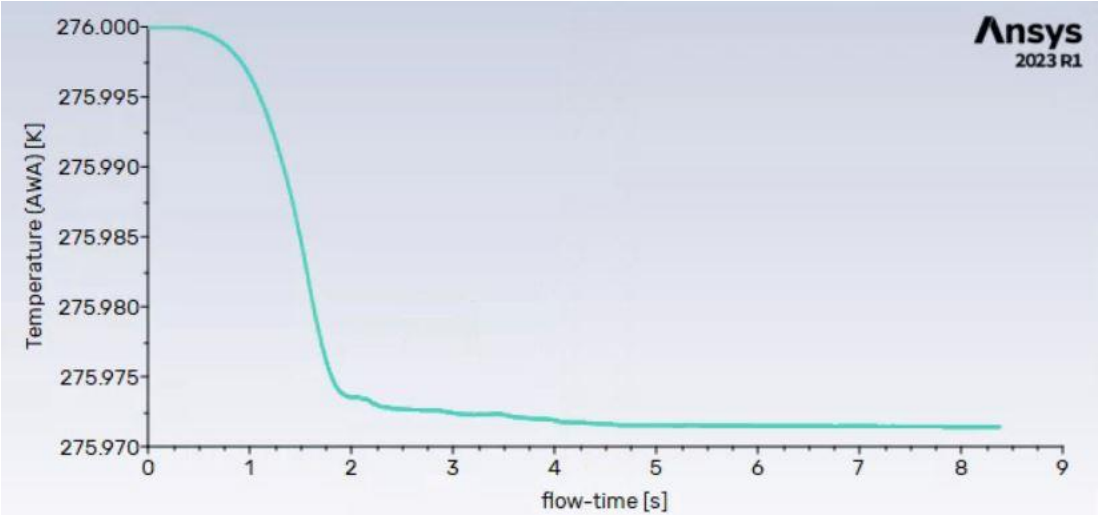


Figure 1: Moisture Transport Final Iteration Temperature Results Example – Increasing Porosity

Assuming that the initial area-weighted average temperature of the soil cross section beneath the pitch was an ambient temperature of 276K, and that water entering this zone horizontally was 275.5K (as rain is often colder than ambient temperatures), the soil average temperature settled at a constant of 0.028K lower than when dry. This observation alongside the pitch size and cross-sectional averaging of pitch properties, resulted in the decision to deem moisture transport impact as negligible.

The results from the inclusion of time delay imbalance found that although small, the effect was positive and improved the overall accuracy of the prediction of snow depth.

Finally, it was also found that the effect that the incorporation of features from [2], while also small, were again beneficial to the overall accuracy of the snow depth prediction.

Sensitivity Analysis

The results of the variable study sensitivity analysis found that snow thermal conductivity had the greatest impact on snow depth prediction and air density had the least impact. The variables were ranked by impact on the predictive

power of the python model, which was visualised using hex-graphs. When altered, none of the included variables produced a situation where no change was visible in the results. For this reason, all variables remained in the model after the Sensitivity Analysis.

Results of the comparison between different numerical schemes found that both implicit and predictor-corrector schemes failed to show meaningful improvements in the current computational model and therefore an explicit scheme was retained for the improved model.

A study conducted on the effect of time step size found that reducing the step size from 60 minutes to 20 minutes significantly improved the computational accuracy whilst still retaining an acceptable computational time. Further reductions in the step size showed some incremental improvements but were ultimately deemed unnecessary due to the large increase in computational time.

The snow depth predictions compared to the snow depth measured, in meters, that the original model produced are shown in Figure 2. The improvements made to the model predictions can be clearly shown in Figure 3, which includes additional refinements such as consideration of the time-delay within the underground heated pipes, the inclusion of features from the pavement paper and a reduced time step size of 20 minutes.

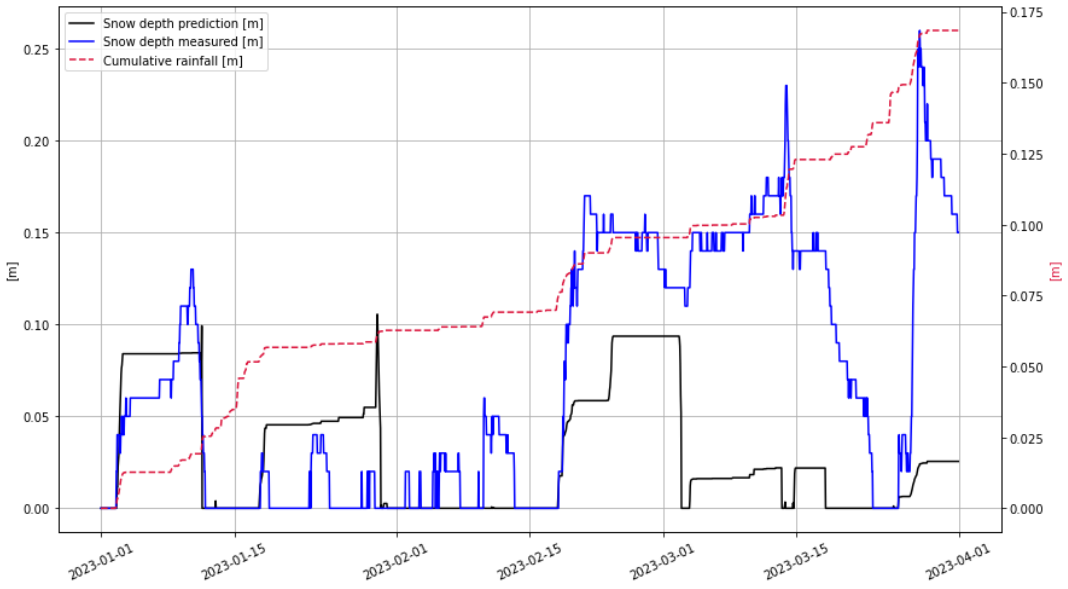


Figure 2: Original Model Snow Depth Predictions

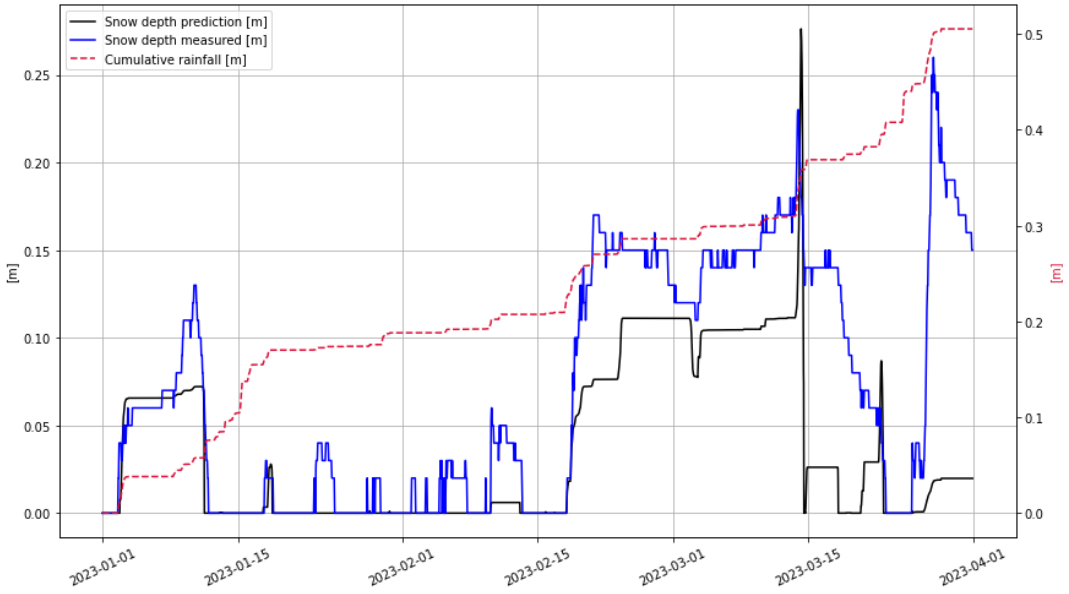


Figure 3: Improved Model Snow Depth Predictions

Optimal Control Guidance Document

The development of the LHS weather data and decision tree model allowed for a guidance document to be created. This document would be used by a human non-specialist operator who would be responsible for setting the control values of the underground heating system. The decision tree was trained to determine optimal control inputs for different weather scenarios. These weather trends and control values were simplified and presented in the form of the 'rule of thumb' document, which allowed the operator to set approximate optimal control inputs based on current weather conditions such as air temperature and snow depth.

During the analysis of the decision tree output, it was concluded that air temperature and precipitation amount were the 2 most important factors in determining whether snowfall would occur and were therefore taken as the leading choices in the guidance document. Every element was connected by a Yes/No choice to the rest of the branches, to make using the document as simple as possible. At the beginning of every branch, the air temperature is assessed. Once a suitable range has been selected, the next leaf in the branch can be followed. This was most commonly Precipitation Amount; however, other cases were included to allow situations where no snowfall is currently occurring, but snow cover is still present to be captured. For simplicity, the guidance document was limited to 2 weather parameter choices, based on which a suitable range of control inputs was recommended.

The complicated machine learning approach aimed to further the accuracy of the optimal control input variables for all possible weather situations. A more detailed optimal control guidance document was created. This could be built on in future work by producing a full and extensive operative guidance document, far beyond the scope of the guidance required within this project.

Cost Analysis

A review of the potential savings that could be made using full optimal control, over an example 3-month period, gave a value of €2222.50. This was calculated using the average cost of energy in Finland at that time of year, comparing the energy consumption for an optimal case to a standard operation case.

Conclusions and Recommendations

The aim of this report was to implement improvements to a computational model for the forecasting of snowfall and to predict the required control inputs to achieve playing conditions with minimum energy consumption. Recommendations for possible future research aims based on the work undertaken are discussed below.

While the effects of wind speed were negated in the calculations carried out as part of this report, based on previous studies conducted for building rooftops, further work could be carried out in designing a large-scale simulation representing the football field and surrounding area to confirm this negation and assess the true impact of interference by the nearby buildings and trees.

Simplifications were also made to the snowpack classification, including only the 4 major categories. In reality, a snowpack could consist of a number of different snow types, therefore the inclusion of multi-phase snowpacks in the heat and mass balance equations could contribute to further improvements in the accuracy of the melting model.

Moisture Transport under the pitch was ultimately found to have minimal effect on the heat and mass balance calculations, however the study could be adapted to other situations, for example where the surface is not impermeable. This, alongside simulating the transport of mass in the form of particles in the soil or comparing results between different turbulence models in place of laminar flow could pave the way for greater understanding of the behaviour of moisture in different soil types.

The implementation of a pandapipes model provided good estimations of the time-delay in transportation of the fluid within the network, however the limitations of the model did not allow for all the properties of the surrounding soil to be included. A large-scale model in software such as ANSYS, while highly computationally intensive, could allow for very refined calculations and better visualisation of the fluid movement in the pipes once the initial parameters have been determined using the pandapipes network.

Simple ‘Rule-of-Thumb’ guidance was developed for setting the temperature difference and liquid solution flow rate based on a range of weather parameters. This was presented in the form of a document which could be used by a non-specialist operator to set the appropriate range. Due to time constraints and being out with the scope of the project, precise optimal control guidance was only developed for 30 nodes. This provided the exact setting which should be used in a particular case rather than a range. This could be expanded, for example using other machine learning methods used in conjunction with Deep Learning to process the large decision tree output

and group the corresponding branches for a much wider selection of weather conditions.

For the Design-of-Experiments, the random sampling process was simplified by fixing the values of the weather parameters and control inputs for the full 1-week duration of the simulation, at the end of which the values of snow depth and energy consumption could be retrieved. Further improvements could be made to the model though the inclusion of non-constant weather data to reflect the real-world conditions more accurately.

The model utilised as part of this project was designed considering the specific geometry and location of the field in Naantali, Finland. In this case, the optimisation recommended was estimated to contribute to savings of up to €2222.50 over the 3 month period simulated. Simple modifications to the model could make it relevant to other locations.

Applications also exist beyond the realm of sports, and the technology and model could be utilised in the maintenance of other public spaces and infrastructure. Further work could be undertaken in adapting the model for applications in those areas, reducing the need for polluting snow clearing vehicles and other de-icing methods.

To conclude, the work undertaken to refine the dynamical snow melting model improved the overall accuracy of the prediction of snow depth greatly when compared with real life weather data for a 3-month period. The operation of the heating system was optimised for 1-week periods such that guidance could be drafted to recommend control inputs which ensure the field is kept at playing conditions, with minimum energy expenditure.

Acknowledgements

Throughout the completion of this project, and the writing of this report our team could not have achieved what we have without the support and assistance of so many.

Firstly, we would like to express our thanks and appreciation to our project supervisor Eero Immonen for his support and guidance throughout this entire project. His insights were invaluable in many aspects of our research and project development.

Secondly, we would like to thank the research associates of the Computational Engineering and Analysis (COMEA) research team at Turku University of Applied Sciences. Specifically, we would like to give appreciation to Fatemeh Ardaneh, Ashvin Chaudhari and Sajad Shahsavari. We are extremely grateful for the support and ongoing feedback throughout our project.

Lastly, we are thankful to all at Turku University of Applied Sciences for welcoming us into their university and allowing our exchange experience to be extremely rewarding and an experience to remember. The welcoming environment and range of facilities at the university allowed for our team to achieve and produce work at this high level.

TURKU AMK

Abstract

The purpose of this paper was to improve upon modelling work, began in the 2023 Computational Engineering and Analysis paper, “A computational model for underground heating control of outdoor sports fields in winter conditions”, [1], with a more detailed description of the physics of snow and the heat and mass transfer as it melts. Following from the refinement of the dynamical snow melting model based on the COMEA approach, the aim was to establish a computational methodology for optimising the operation of the underfloor heating system for 7 days ahead of the predicted weather forecast.

This report includes a breakdown of the thermodynamic processes and heat and mass transfer occurring in a snowpack on the surface of a heated football pitch, for a 3-month period. Potential improvements to the previous computational model were investigated through the incorporation of the transportation of moisture in soil under the ground, the time delay imbalance of the water-propyleneglycol transport in the heated pipes and the incorporation and comparison of additional features from a similar model.

It was found that the movement of moisture in soil under the ground was insignificant and therefore could be deemed negligible. However, it was found that although small, the effect that the inclusion of time delay imbalance and incorporation of features from [2] were positive and improved the overall accuracy of the prediction of snow depth.

Several sensitivity analyses took place, first showcasing the impact of different variables on the final prediction, with Snow Thermal Conductivity having the greatest impact. The effect of time step size was also investigated to find an appropriate value which could improve the accuracy of the model while still retaining acceptable computational times. The use of different numerical schemes, namely implicit and predictor-corrector was also explored. Results found that both schemes failed to improve on the current computational model so the explicit scheme was retained in the improved model.

To establish an optimal heating system for a given weather forecast a Design-of-Experiments was conducted through Latin Hypercube Sampling (LHS). The results the LHS algorithm produced were used within a decision tree to determine optimal control inputs for different weather situations and trends. The purpose of conducting this was to develop an operational methodology that achieves optimal playing conditions of snow depth ≤ 1mm whilst minimising energy consumption.

The development of the decision trees was followed by the inclusion of a more complicated machine learning approach which progressed the accuracy of the optimal controls to more refined and specific weather conditions beyond the threshold of the decision tree.

The real-life applications of the research discussed in this report are greater than just artificial grass football pitches. Heavy snow creates challenges in many aspects of life in a cold climate, therefore the system has applicability

in infrastructure projects such as heated pavements, train tracks and airport runways.

Contents

Table of Figures

Table of Tables

Nomenclature

Symbol	Description	Units
A_{xy}	Area of Football Pitch	$[m^2]$
L_{eff}	Characteristic Length for a Rectangular Field Surface Section	$[m]$
C	Cloudiness Ratio	
h	Convection Heat Transfer Coefficient	$[W/m^2C^o]$
Q	Darcy Flow Rate	$[cm^3/s]$
ρ	Density	$[kg/m^3]$
z	Depth Underground	$[m]$
μ	Dynamic Viscosity of Fluid in Pipe	$[Pa\ s]$
ER	Energy Ratio	
Eu	Euler Number	
ε	Ground Surface Emissivity	
$\dot{Q}$	Heat Transfer	$[W]$

w	Humidity Ratio, $kg\ (vapor)\ /\ kg\ (dry\ air)$	
U	Internal Energy of Snowpack	$[kJm^{-2}]$
h_{LV}	Latent Heat of Vaporization of Water	$[kJ/kg]$
$\dot{m}$	Mass Flow Rate	$[kg/s]$
ν	Momentum Diffusivity (Kinematic Viscosity)	$[m^2/s]$
Nu	Nusselt Number	
$\dot{q}$	Per-area Heat Flux	$[W/m^2]$
K	Permeability of the Medium	$[\mu m^2]$
Pr	Prandtl Number	
Ps	Precipitation Amount	$[m^3/s]$
$\widehat{y_i}$	Predicted Value of the $i^{th}\ Sample$	
P	Pressure	$[Pa]$
Re	Reynolds Number	
n	Sample Size	
W	Snow Water Equivalent	$[m]$
C_p	Specific Heat Capacity	$[J/kg\ K]$
σ	Stefan-Boltzmann Constant	$[W/m^2K^4]$
T	Temperature	$[C^o]$or [K]
ΔT	Temperature Difference Between Inlet and Outlet	$[C^o]$or [K]
k	Thermal Conductivity Coefficient	$[W/m\ K]$
α	Thermal Diffusivity	$[m^2/s]$
L	Total Length of Piping Travelled	$[m]$
y	True Value of the $i^{th}\ Sample$	
v	Velocity of Flow Over Flat Plate	$[m/s]$
u	Velocity of Fluid in Pipe	$[m/s]$

Subscripts and Superscripts

$\boldsymbol{amb}$	Ambient
$\boldsymbol{Cond}$	Conductive Heat Transfer
$\boldsymbol{Conv}$	Convective Heat Transfer
$\boldsymbol{Evap/Cond}$	Evaporation/Condensation
$\boldsymbol{f}$	Fluid within Underground Pipes
$\boldsymbol{Rad}$	Radiation Heat Loss
$\boldsymbol{Sens}$	Sensible Heat
$\boldsymbol{Solar}$	Solar Heating Power
$\boldsymbol{Sun}$	Sun Average Heating Power
$\boldsymbol{Sur}$	Surface
$\boldsymbol{Heat}$	Thermal Control Input

Abbreviations:

Abbreviation:	Definition:
COMEA	Computational Engineering and Analysis

DoE	Design of Experiments
ECON	Electrically Conductive Concrete
LHS	Latin Hypercube Simulation
ML	Machine Learning
MAE	Mean Absolute Error
MC	Monte Carlo
UEB	Utah Energy Balance

1.0 Introduction

1.1 Background

In Finland and other countries with cold climates, it is a normal experience to face multiple months of snow cover annually. Although this is expected by the locals, it still comes with significant challenges. Up to 3 months of thick ice and snow covering the ground causes difficulties for all members of the community. For example, slowdown in public transport and the aviation industry has many knock-on effects. Frozen streets can be difficult and dangerous to walk, cycle and drive on, especially for the elderly and those with physical disabilities. Frozen parks and sports grounds restrict the activities which can be enjoyed there, affecting the mental and physical health of millions annually. The removal of ice through manual labour and other means such gritting the surfaces is technically an option for small areas of interest but there are more effective long-term solutions.

Underground heating on football pitches is not a new development and has been around since 1958 when Everton FC introduced electric underground heating at Goodison Park [3]. Sixty-five years on, it is now a necessity, under Section K.22 of the Premier League requirements [4]. Undersoil heating is installed under all pitches to reduce the loss of revenue that would occur by cancelled matches due to extreme weather.

The 2022 energy crisis brought extreme pressure to maintain affordable and efficient systems, when the price of energy went as high as 500€/MWh [5]. The energy use of current heating systems has been deemed unreasonable and so the need to optimise them to reduce costs and minimise energy waste is vital [6]. It is now more important than ever to be energy conscious and only use the minimum consumption required without waste, but also to ensure that optimal playing conditions are met to allow for games to continue year-round.

Minimising energy consumption whilst retaining adequate playing conditions is an optimal control problem, which can be solved using computational methods provided that an accurate model of the snow melting above an underground heating system is available, which includes all relevant thermodynamic processes including heat and mass transfers.

The 2023 Computational Engineering and Analysis (COMEA) paper [1] introduced the work by modelling and simulating the underground heating system of an artificial football pitch located in Karvetti, Naantali, Finland. It described a relatively accurate physics-based dynamical heat and mass balance model which would allow for the operating costs of the underground heating system to be reduced whilst maintaining the same level of ground conditions.

1.2 Objectives

This report presents the work and findings of the University of Strathclyde, Aero-Mechanical Engineering 5th Year Masters Group Project titled “Model-Based Optimal Control of An Underground Heating System”, research

conducted whilst on international exchange at Turku University of Applied Sciences in Finland.

The purpose of the project was to improve upon the modelling work started in the COMEA paper [1] with a more detailed description of the physics of snow and the heat and mass transfer as it melts. This paper aims to build a model of an underground heating system for snow and ice removal on an outdoor football pitch, with minimum energy expenditure and cost, by refining the dynamical snow melting model based on the COMEA approach and establishing a computational methodology for optimising the operation of the underfloor heating system for 7 days ahead of the predicted weather forecast.

The objectives defined for this project were as follows:

- To gain a clear understanding of the prior work undertaken by the COMEA research team. This includes familiarisation with the current heat and mass transfer model, and the relevant physics and thermodynamics. In addition, the group should spend time to allow for a clear understanding of the current python code and key input parameters.
- To refine the dynamical snow melting model, the heat-mass transfer interactions between different layers of soil, and the interactions between the snow surface and the outside atmospheric conditions. To consider multiple layers of snow and the interactions between them through the generation of snowmelt water and the subsequent transportation of moisture into the ground.
- To account for the imbalance of snow melting across the length of the pitch, due to the water-propyleneglycol solution circulating through the ductwork creating a temperature time delay.
- To produce a detailed comparison to the heated pavement model paper. The benefits and drawbacks of this model will be analysed and the opportunities for implementing their modelling techniques into the current system will be considered.
- To perform a Sensitivity Analysis on the model, examine the output effects, and improve model accuracy.
- Through Design-of-Experiments, simulate weather conditions and operational parameters over a wide range, to study the interactions between variables.
- To utilise Machine Learning methods in establishing a simple framework for the optimal control scenario in different weather conditions. From this, write a simplified or “Rules of thumb” guidance document on the control of the system as it will be controlled by non-specialist human operators.
- To establish a computational methodology for optimal operation of the heating system for a linearised weather forecast 1 week in advance. This could be achieved through further use of Machine Learning methods.

1.3 Report Structure

The refinement of the original snow melting model is conducted in section 3.2 and considers multiple layers of snow and the interactions between them through the generation of snowmelt water and the subsequent transportation of moisture into the ground. This is investigated through an extensive fluid study, detailed from section 3.2.1 to 3.2.5. The time delay of the water-propyleneglycol fluid within the underground pipes is studied in section 3.2.6 and the effect on the overall heat transfer model should be determined. Furthermore, a detailed comparison and incorporation of features such as the inclusion of evaporation and condensation from the 2007 Liu et. al paper [2] is investigated in section 3.2.7.

The impact different variables had on the model's prediction of snow depth, the use of alternative numerical schemes such as implicit and predictor-corrector and the effect of different time step sizes for improving the accuracy of the model whilst still retaining acceptable computational times is determined through 3 distinct sensitivity analyses in Sections 3.3.1, 3.3.2 and 3.3.3, respectively.

To establish an optimal heating system for a given weather forecast a Design-of-Experiments is conducted through Latin Hypercube Sampling (LHS) in section 3.4.1, which is the preliminary work of decision trees to determine optimal control inputs for different weather situations and trends continued in section 3.4.2. The purpose of this is to achieve optimal playing conditions of snow depth ≤ 1mm whilst minimising energy consumption.

The results for all methodologies discussed in Section 3.0 are presented, analysed, and discussed within Section 4.0. Recommendations for future research are proposed in Section 4.10.

Appendix A1 – Reflective Report documents the management and group work throughout this project. It contains personal reflections from each member of the group documenting their entire experience of exchange and this project overall.

1.4 Project Scope

The scope of this project was determined through defining clear deliverables and outcomes, outlined in Section 1.2. The main deliverable of this project was an improved computational underground heating system model in the form of Python code which can accurately predict the snow depth on the artificial football pitch compared with the measured snow depth from weather data. Furthermore, a 'Rules of Thumb' guideline document was also required to be used by a non-specialist, human operator to determine the optimal control parameters, ΔT and Liquid Flow Rate, for a given weather forecast.

2.0 Literature Review

2.1 Original Model

The work within this report aimed to improve the dynamical model of the thermodynamic processes within an underfloor heating system for the melting of snow on an artificial grass football pitch, first introduced in the COMEA paper [1]. Throughout this report this model is referred to as the 'Original Model'.

Computational modelling of any underground system is based on the thermodynamic processes of heat and mass transfer. A simple concept becomes difficult when trying to understand the changing properties of snow in various weather conditions. Differences in depth, liquid fraction and density are factors to consider when snow falls and melts at different rates [1].

Many of the heating system models currently in operation are limited as they are based on simple steady state calculations and are not representative of a realistic environment, which would have many factors affecting heat transfer. Differing weather conditions, soil types and interference from external factors such as wind mean that it can be difficult to replicate a true picture of what is really happening [2].

The original model [1] introduced a computational model for energy efficient underground heating control of an outdoor sports field in winter conditions. The proposal of this nonlinear lumped-parameter model states that different parameters which would have real life effects within the system are considered, such as air temperature, wind, precipitation, cloud cover, and sky radiation heat losses. The model does not allow for weather forecasts to determine predictive control. It also does not consider feedback control due to varying field temperatures. The control objective was the energy efficient melting of the snow, which would accumulate on the pitch surface.

The arrangement of the underground heating system is shown in Figure 1 and consists of 4 distinct layers. A brief description of each layer is provided in Table 1:

Table 1: Description of Model Layers

L0:	The top snow layer which interfaces with ambient air conditions such as (wind, cloudiness, precipitation, solar heating, and sky radiation heat losses.)
L1:	The artificial grass layer which is exposed to changing weather conditions.
L2:	A layer of mechanically stabilised earth or MSE, typically composed of gravel or sand, which is artificially reinforced soil.
L3:	The heated layer. The 3 red circles within L3 represent the heating element, a serpentine pipe which runs the lengths of the sports ground, and through which flows a water-propyleneglycol solution, which causes the overall system temperature to increase.

Figure 1 represents a schematic of an overview of the artificial grass football pitch with an area $A_{xy}\ [m^2]$ and a depth of $z\ [m]$. The distinct layers of the model L_0, L_1, L_2 and L_3 are all incorporated into the layered heat and mass balance each with lumped parameter averaging across the horizontal dimensions.

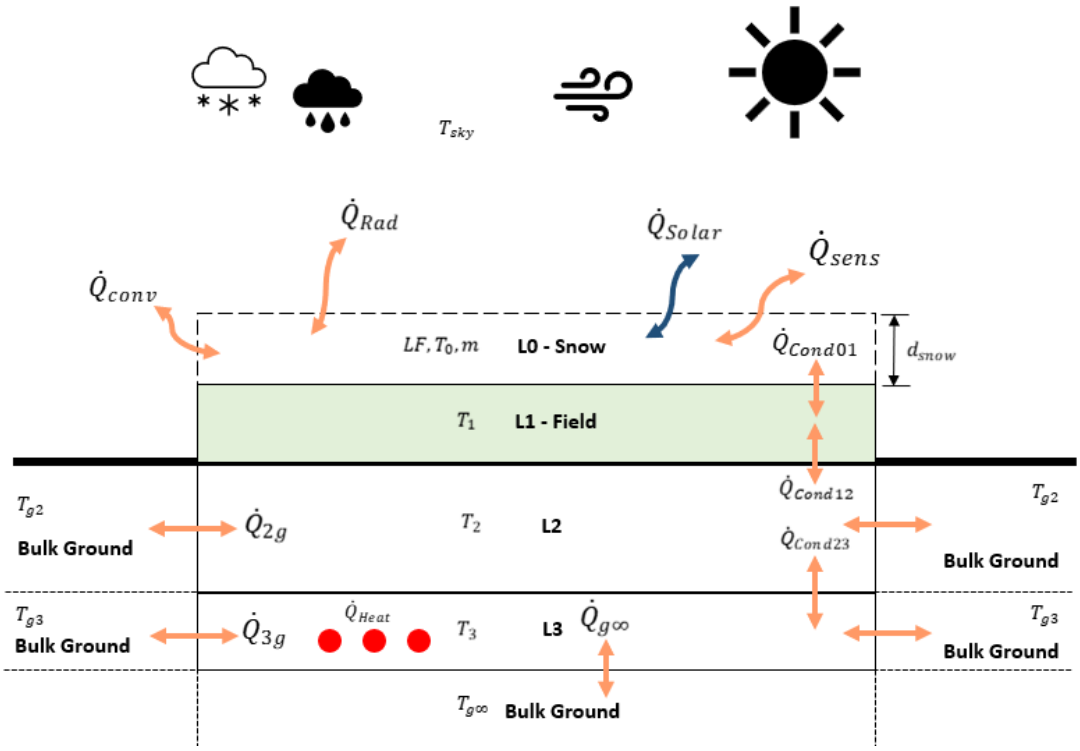


Figure 1: Schematic Overview of the Heating System

The original model does not consider the effects of moisture transport by water absorption and therefore assumes that the field surface is impermeable, and the composition of the soil is constant. The work presented in this paper addresses additional constraints and parameters such as the effects of ground moisture transport by water absorption and evaporation. The new model will also consider the delays in thermal convection between the heat exchanger and the soil.

Forced snow melt occurs due to heated pipes in L3. A heated water-propyleneglycol solution is circulated through a system of pipes in a serpentine configuration underneath the football field. The solution is heated utilising waste heat from a nearby power plant. This type of system has a wide range of benefits in many applications beyond the one considered in this paper, such as maintenance of public spaces and other infrastructure such as roads and pavements, where the accumulation of snow and ice could be an issue. A significant advantage of this implementation over the use of salt and grit for de-icing is the environmental impact. Salt and sand could both accelerate corrosion in the road surface and damage cars passing over it, which would require additional maintenance and result in higher financial costs for both governments and individuals. They also suffer from temperature limitations, leading to further restriction in their useful capacity. [7] Organisations such as airports and other transport hubs could be

significantly impacted during heavy snowfall, causing delays, and inducing high costs in maintaining the road surface. Studies have concluded that mechanical-based methods for snow melting and de-icing could cost as much as $800 per ton ($1436.96 when converted to today's equivalent). [8], increasing the burden of public spending. Other benefits of implementing this method of de-icing are previously described in 1.0, however, different solutions have been developed to help tackle this issue [9, 10].

2.2 Alternative Computational Models

2.2.1 Utah Model

The 2013 model described in 'Modeling the snow surface temperature with a 1-layer energy balance snowmelt model' [11] is the basis from which the original model was formed. It estimates melting characteristics, using simple mass and energy balances to determine heat fluxes. The Utah Energy Balance (UEB) model assumes that the surface of the field is impermeable, and the soil is therefore unchanged by temperature. Using dynamical and real-life physics it models the intersection between snow and ground in numerous weather conditions. The UEB method models using one layer of snow to simplify computations, but allows snow surface temperature to vary from the snow average temperature by using equilibrium gradient parameterisation, based on surface energy balance.

The variables used in the UEB are snow water equivalent, W $[m]$ and internal energy of the snowpack U $[kJm^{-2}]$ and the model is strongly based on conservation of mass and energy. The UEB takes into consideration the large thermal insulating properties of snow and therefore assumes the thermal gradients in the snowpack to be large and non-linear.

The UEB paper evaluates 3 approaches: equilibrium gradient, force-restore and modified force-restore as part of a heat and mass balance snowmelt model, as well as evaluating the theory of meltwater refreezing. These additions allowed for more accurate and truer 1-layer snowpacks to be included in the computational energy balance.

The Utah model is not the primary basis for the original model, or the model described in this paper, as it does not incorporate a heat source which can be controlled and is designed for snowpacks or much larger surface areas.

2.2.2 Comparison of the Heated Pavement Model

The work of the original model [1], although similar, still has significant differences to the example presented in the 2007 paper, "Modelling Snow Melting on Heated Pavement Surfaces. Part I: Model Development" [2], which describes the snow melting process on the surface of a heated pavement.

The pavement model considers snow melting on a hydronically-heated pavement. Hydronically-heated pavements are environmentally conscience alternatives to traditional de-icing methods. They work by circulating fluids such as brine, oils or glycol-water through pipes embedded below the target surface [12]. Hydronically-heated pavements work year-round and not just during the colder seasons. During warmer summer months when the surface

temperature is high, the temperature of the fluid increases, and the energy from the heated fluid is saved in thermal energy storages, used throughout colder months for de-icing pavements.

The main difference between the original model and the pavement model is the consideration of snow as a multiphase model, subject to forced conductive heat flux from below. Both models defined boundary conditions and developed heat and mass transfers for their individual conditions. The pavement model's heat and mass transfers allowed for the inclusion of a variety of weather conditions and surface types. The pavement model, however, can predict the transient surface conditions, snow cover and surface temperatures from heat fluxes and weather forecasts.

The difficulties with modelling underground heating systems come due to the varying properties of snow in real life weather conditions, which change throughout individual days. The depth of snow and the value of the liquid fraction becomes inconsistent [1]. Models of snow melting should be as close to real-time control as possible and therefore be relatively lightweight. As the pavement model attempts to include all associated physics it would require large computational power and therefore would not be considered as effective in these terms as the original model.

The improved model developed in this report will incorporate some of the features explored in the pavement model such as the inclusion of $\dot{Q}_{Evap/Cond}$ and the consideration of snow as a multiphase model based on the 7 snow types classified in [13], discussed further in Section 2.7.

2.3 Heat and Mass Transfer Equations

As described in Section 2.1 the ground under the football pitch is separated into layers. The model developed within this report keeps the same physical architecture of the football pitch from [1] and therefore the heat and mass transfers between L_1, L_2 and L_3 remain the same.

Heat is transferred upwards from the heated pipes located in L3 towards L0 where it melts the snowpack present.

The heat balance equations through these layers are described in the following section.

Firstly, $\dot{Q}_{Heat}$ is the thermal control input from the underfloor pipes in L3 and is calculated using:

$$\dot{Q}_{Heat} = \dot{m}_f C_{p,f} \Delta T_f \quad [1]$$

Where ΔT_f is the difference between the inlet and outlet temperature of the fluid.

Heat is transferred between adjacent layers by conduction and the conduction heat transfer rate for layers i and j is given by:

$$\dot{Q}_{Cond,ij} = \frac{k_{ij}}{\Delta z_{ij}} A_{xy}(T_i - T_j)$$

[2]

Where Δz_{ij} is the vertical distance between layers.

The balance equations for each subsequent layer are given.

For the heated bottom layer L3:

$$m_3 C_{p3} \frac{T_3^{t+\Delta t} - T_3^t}{\Delta t} = \dot{Q}_{Heat} + \dot{Q}_{Cond,23} - \dot{Q}_{3g}$$

[3]

For the mid layer L2:

$$m_2 C_{p2} \frac{T_2^{t+\Delta t} - T_2^t}{\Delta t} = -\dot{Q}_{Cond,23} + \dot{Q}_{Cond,12} - \dot{Q}_{2g}$$

[4]

For the turf layer L1 and the snowpack L0:

$$m_1 C_{p1} \frac{T_1^{t+\Delta t} - T_1^t}{\Delta t} = -\dot{Q}_{Cond,12} - (1-\gamma)\dot{Q}_{Cond,01} + \gamma\left(\dot{Q}_{Conv} + \dot{Q}_{Solar} + \dot{Q}_{Rad} + \dot{Q}_{sens}\right)$$

[5]

If the depth of snow, d_{Snow} is $> 10^{-4}$ m then $\gamma = 0$ and if $d_{Snow} < 10^{-4}$ m then $\gamma = 1$.

Equation [5] represents the balance equation for the turf layer L1 and if present, the snowpack layer L0.

The snowpack is exposed to external factors and ambient air conditions such as heat transfer by convection, radiation heat losses, solar heating, and precipitation heat flux.

The convective heat transfer rate from the surface of the snowpack is given by:

$$\dot{Q}_{Conv} = hA_{xy}(T_{air} - T)$$

[6]

Where h is the convection heat transfer coefficient obtained from flat plate boundary layer flow theory.

$$h = 5.74 \cdot v^{0.8} \cdot L_{eff}^{-0.2}$$

[7]

Where $L_{eff} = 4A_{xy}/(2\Delta x + 2\Delta y)\ [m]$ is the characteristic length for a rectangular field surface section.

The radiation heat loss is given by:

$$\dot{Q}_{Rad} = \varepsilon\sigma({T_{snow\_surface}}^4 - {T_{sky}}^4)$$

[8]

Where $\sigma = 5.67 \times 10^{-8}\ W/m^2K^4$ is the Stefan-Boltzmann Constant.

The effect of clouds in solar heating power is given by:

$$\dot{Q}_{Solar} = \dot{Q}_{Sun}A_{xy}$$

[9]

Where $\dot{Q}_{Sun}$ is the sun average heating power and is given by:

$$\dot{Q}_{Sun} = ((1 - C) + 0.5)\dot{q}_{solar}$$

[10]

Where C is the cloudiness ratio (with $C = 0$ indicating clear skies and $C = 1$ indicating full cloud cover).

The sensible heat flux associated with precipitation is given by:

$$\dot{Q}_{sens} = \dot{m}_{rain}C_{p,water}(T_{air} - T_1)$$

[11]

Where T_1 is the turf temperature as shown in Figure 1.

The conduction heat transfer rate at the top surface of a dry snow layer is given by:

$$\dot{Q}_{Cond} = \frac{k}{z}A_{xy}(T_{air} - T)$$

[12]

The equations described above are the basis of the original model's balance equations and subsequent python script. The script simulates snowpack over a 3-month period, which is exposed to real life physics and ambient air conditions. It was able to predict snow depth to a reasonable level of accuracy, through the use of parameter identification which inputted different weather conditions into the model. The results of this are shown in Figure 2.

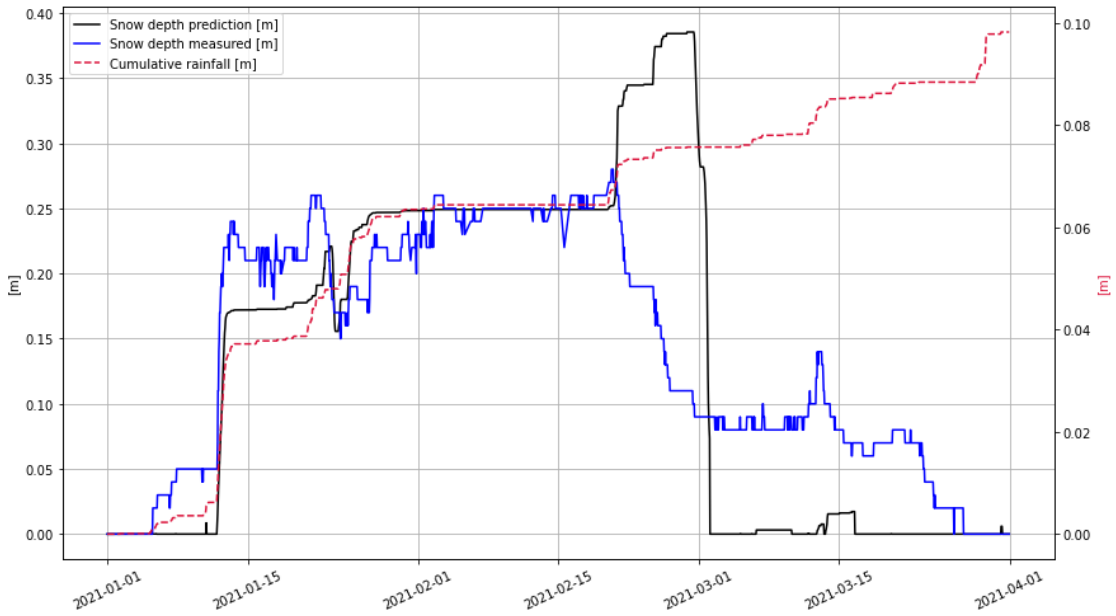


Figure 2: Measured and predicted snow depths (solid lines) and measured rainfall (dashed line).

2.4 Underground Heating Using Electrodes

The original model looks at an example of an underground heating system which uses pipes of heated water-propyleneglycol solution to melt the accumulated snow on the surface. This is not the only technology used for underfloor heating.

One implementation, which has been tested at the Iowa Department of Transport headquarters, is an ECON (Electrically Conductive Concrete) Heated Pavement System [14]. This employs the use of electrodes for heating, rather than a fluid circulating through tubing. In this case, temperature sensors are utilized to monitor the surface temperature and activate if this falls below 5°C. This helps reduce energy consumption as the system will not activate if the road surface is sufficiently warm. Testing of different ECON mix designs was performed to optimize heat transfer to the surface layer. Test sections were also designed using different numbers of electrodes, as well as varying the shape of the electrodes – a smooth circular bar, a hollow circular bar and a flat bar were all considered. This investigation found that the flat bar produced the highest power density while the solid bar produced the lowest power density.

This could have some potential advantages over the current system implemented at the football field. The key advantage is accommodating for the imbalance that exists in the current system with regards to heat distribution across different sides of the field. The time it takes for the hot fluid to be circulated through the entire system will result in uneven melting of the snow in some parts of the pitch when compared to others.

2.5 Time Delay Imbalance

One disadvantage of underground heating system of the original model is the time delay between activation of the system and heat reaching the surface layer. [15] This can be visualised in Figure 3, showing the large temperature differences between the inlet and outlet sections of the pipe. [16]

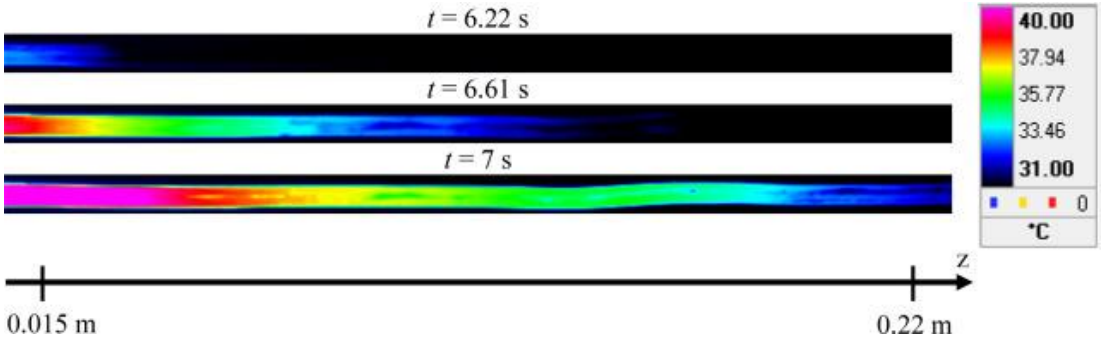


Figure 3: Temperature-Time Delay in a Pipe Section

There are several factors which impact the performance, including the depth at which the tubing is placed, and outside ambient conditions. In their paper "CFD-Based Sensitivity-Analysis and Performance Investigation of a Hydronic Road-Heating System" [7], A. Ahmed et al. performed tests to understand the time taken for a road heating system in the southern part of Germany to reach and maintain a constant temperature of 3°C on the asphalt surface. The tests were carried out in December, aiming to capture the most demanding time of the year for these systems. Initial studies considered 3 weather scenarios - when the temperature 3 different conditions - when the initial temperature of the asphalt is 3°C, 0°C, and −3°C with snowfall. Results showed that changing the initialization temperature had an impact on the time taken to reach and maintain a constant temperature on the asphalt surface. Higher initialisation temperature allows for shorter periods of operation, reducing energy demand and costs. Nevertheless, to maintain a temperature of 3°C the system required around 12 – 15 hours of preheating (dependent on the initialization temperature) to avoid the accumulation of snow and ice [7]. This demonstrates the high lead times required for heating the surface in harsh weather conditions and the need for accurate weather models to help with predicting the impact on the road surfaces.

Further improvements were made by employing a predictive controller based on live weather forecasts, instead of a simple one working at a constant heating value continuously. Instances where road surface temperature was below 0°C and another scenario where surface temperature was both below 0°C and the dew-point temperature were considered. The temperature threshold was defined between +0.1°C and +1.6°C as both the freezing temperature and the dew-point temperature for pre-heating the road surface. The second approach proved capable of saving 10 times the annual required energy for hydronic road heating [7, 17].

In this project, restrictions exist on modifications that can be made to the physical setup. There are no plans to perform maintenance on the field at the time of writing, meaning that the existing system must be utilised, with its

flaws and limitations. This means that the distance between the tubing and the top layer of soil cannot be changed to optimise for better heat transfer, as this would require costly refitting and would result in the pitch being unusable for a significant duration of time. Installing insulation is another consideration which could significantly improve the thermal properties of the existing setup. Nevertheless, completely overhauling the underlying technology is beyond the scope of this project. The focus is instead on improving the existing technology through the implementation of computational analysis methods to better predict and react to different weather conditions.

Firstly, the flow rate of the solution must be considered. The main way in which this can be impacted is through increasing the velocity at which the solution enters the piping. This will induce a change in the Reynolds Number, according to the relationship:

$$Re = \frac{\rho u L}{\mu}$$

[13]

Where ρ is the density of the solution, μ is the dynamic viscosity of the fluid, u is the velocity and L is the length of piping travelled.

The Prandtl number is a measure of the ratio between momentum diffusivity and thermal diffusivity, described by the equation:

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{C_p \mu}{k}$$

[14]

Where C_p is the specific heat capacity of the fluid and k is the thermal conductivity.

The Nusselt Number is the ratio of conductive to convective heat transfer, which describes the thermal energy transferred. This could then be found from the following equation:

$$Nu = 0.023 Re^{0.8} Pr^{0.4}$$

[15]

This equation is valid for the assumptions $0.6 < \mathrm{Pr} < 160$ and $Re > 10{,}000$ [18]. The Euler Number is the ratio of pressure forces to mechanical forces:

$$Eu = \frac{\Delta P}{\rho v^2}$$

[16]

Using the above equations for the Nusselt Number and the Euler Number, the energy ratio between the thermal energy transferred and the mechanical energy required can be found.

$$ER = \frac{Nu}{Eu}$$

[17]

One of the main observations from the equations above is that the energy ratio could be directly impacted by altering the velocity of the solution. For example, an increase in the velocity will induce an increase in the Reynolds Number. The Prandtl Number is a function of the properties of the fluid so it will not be impacted by this change. This means that as long as the validity conditions are met, the Nusselt number, which represents the transfer of thermal energy, will also be increased. An increase in velocity will also generally result in a decrease in the Euler Number, due to the v^2 condition in the Euler equation. Finally, this will result in an increase in the energy ratio, ER.

The other condition which hugely impacts the heat transfer performance is the temperature difference between the inlet and the outlet. The inlet temperature can be impacted by the heating method, which in this case is waste heat from a nearby power plant. Once the fluid is inside the pipes, there are several factors which impact thermal performance. The most important ones will be the physical properties of the solution; however, the diameter of the pipes has also been found to have a significant role in heat transfer [18]. In smaller diameter pipes, there are much smaller velocity and temperature gradients between the bulk fluid and the near-wall region, which is beneficial to the heat transfer. Smaller diameter pipes also tend to be less impacted by what is referred to as secondary flows, where a boundary layer develops, impeding the motion of parts of the incoming flow. The length of the pipes is another key condition, due to the length of time it takes to circulate the fluid. This delay could have significant adverse effects on the performance of the system, for example if it is controlled by a sensor located at a different point from the actuator [19]. “Effect of delay in thermal systems with long ducts” explores the underlying equations used to model the heat transfer in long ducts and a number of specific cases, such as closed loop heating, similar to the network currently in operation at the football field in Naantali. This includes energy balances at the inlet and outlet sections, providing important insight into the techniques used in the modelling of such phenomena [19]. Different models could be used for this purpose, detailed by J. Duquette et al [20].

2.6 Moisture Transport

Water can be introduced to the pitch soil through several mechanisms, such as rainfall, runoff, and flood water. As liquid water enters the system, gravity pushes it into the porous soil, and horizontal diffusion of moisture may affect the heating system, despite the impermeable turf layer above. There are several ways in which the introduction of moisture could impact the operating of a heating system, with heat loss to the ground presenting a major challenge [21]. The chosen properties with potential for variation, were the soil temperature, thermal conductivity, heat capacity and density [22, 23].

Porosity is a unitless measurement of the volume of soil which is comprised of air pockets compared to the overall volume. It is a key factor affecting the motion of liquid water through a porous medium.

The primary relationship to represent fluid flow through a porous medium is known as 'Darcy's Law' [24]. This function determines the flow rate of a fluid through the medium, and is made up of several terms:

$$Q = \frac{K}{\mu} A \frac{\Delta P}{L}$$

[18]

K represents the permeability of the medium, in units of Darcy, or μm^2, and is a measure of the difficulty of passage for fluids. The remaining terms represent viscosity μ in $[mPa\ s]$, flow cross-sectional area A in $[cm^3]$, pressure difference ΔP in $[10^{-1}\ MPa]$ and the length between cross-sections studied L in $[cm]$. For this study, the cross sections were the soil surface to the lower soil boundary at a chosen depth.

In addition, when outside ambient temperatures in soil are below zero, fluid within the soil becomes frozen, and transport does not occur.

The method of incorporating the moisture transport effect had to consider temperature, porosity and Darcy's Law for an accurate representation of reality [25, 26].

2.7 Snow Classifications

Snow as a substance cannot be assumed to be consistent or have constant properties. Several external factors can cause significant differences in snow properties. 7 classifications, shown in Table 2, were identified by Rees et al. [13] and discussed in [2].

Table 2: Classifications of Snow

Surface Condition	***Definition***
Dry	No precipitation present. Surface can be above or below freezing temperature but is free of all liquid and ice.
Wet	Precipitation is present. Surface temperature is above freezing and has liquid water present but no ice. The water can be a result of rainfall, condensed vapour, or melted snow.
Hoarfrost	Surface temperature is below freezing. Frost covers the surface due to the sublimation of water vapour in the ambient air on a cold surface.
Dry Snow	Surface temperature is below freezing so no melting will occur. The surface is covered with dry snow which does not contain any liquid and can be considered a porous matrix of ice.

Slush Only	Surface temperature is at freezing point. The surface contains ice crystals that are fully saturated with water. The water penetrates the porous matrix of ice upwards from surface level.
Snow and Slush	Surface temperature is at freezing point. The snow on the surface is part melted. The upper surface is dry snow, and the lower snow is saturated with water.
Solid Ice	Surface temperature is below freezing and the ice on the surface is solid not porous like snow.

It is important to understand the different classifications of snow to allow for a realistic multi-layer model of the snow on a heated surface to be described. An ideal model would achieve the ability to provide accurate control guidance, for any occurring weather pattern, minimising cost, and energy use.

2.8 Latin Hypercube Sampling

LHS is a type of stratified Monte Carlo (MC) method. MC sampling methods are computational algorithms which return numerical results by randomly sampling a range of data points. The purpose of MC is to obtain solutions to problems which are established in principle through randomness.

From [27] it was determined that LHS is more efficient and requires less computational time than MC.

LHS is based on the 2-dimensional Latin Square Design which has each variable (A, B, C and D) once in each row and once in each collum. A 4x4 Latin Square Design is shown in Figure 4. LHS is 3 or more Latin Square Designs arranged in a cube of multiple dimensions.

		Columns			
		1	2	3	4
Rows	1	A	B	C	D
	2	B	C	D	A
	3	C	D	A	B
	4	D	A	B	C

Figure 4: 4x4 Latin Square Design

First developed by McKay in 1979 [28] for computational experiments of partial differential equations, numerical methods and simulations. LHS allow for the Design of Experiments (DoE) with 1-dimensional balance projection property [29].

To develop a LHS of size n, with the inputs $X = (x_1, x_2 \dots, x_d)$ denote the n runs by $X_1, \dots, X_n$. The k^{th} component of X_j is represented by x_{jk} for $k = 1, \dots, d$. Input variables get sized to have domain [0,1]. $\Pi = (\pi_{jk})$ is a matrix sized $n \times d$ with columns of values $\{1,2, \dots, n\}$ arranges randomly [29].

$$\Pi = \begin{pmatrix} 1 & 5 & 3 \\ 2 & 1 & 4 \\ 3 & 3 & 1 \\ 4 & 6 & 2 \\ 5 & 2 & 6 \\ 6 & 4 & 5 \end{pmatrix}$$

Figure 5: Latin Hypercube Design with 3 Factors and 6 Design Points

Knowing Π a Latin Hypercube design can be developed using:

$$x_{jk} = \frac{\pi_{jk} - 0.5}{n}, \qquad j = 1, \dots, n; \quad k = 1, \dots, d$$

[19]

2.9 Digital Twins

A concept commonly employed in engineering is the Digital Twin. Digital Twins can exist in a range of different applications, from Healthcare to Education to Manufacturing. The term has even been mentioned in NASA's interplanetary development programme [30]. The definition is very flexible; however, it generally involves creating a contextualised virtual model of a real-life situation, meaning that it allows for the behaviour of the object in real scenarios to be accurately simulated. The degree to which the physical processes are mirrored in the computer model will determine the precision of the results [31]. If the model can provide an accurate representation of the operation of the object and its environment, it can be an extremely useful tool in making predictions and can greatly reduce the need for physical testing. A large variety of operational conditions can be tested, including extremes, providing key information about the limitations of the materials and design [32]. Thus, being able to test virtually makes a project much more flexible, allowing for small adaptations and updates without material cost [33]. With increasing use of Artificial Intelligence and Machine Learning in engineering, having a Digital Twin makes implementation of learning tools easy [34].

It is impossible for a digital twin to perfectly replicate reality, and while near-perfect simulations can be achieved, it is often highly computationally intensive. Usually, compromises are made when developing a Digital Twin,

with the intention of modelling only factors which are relevant to the desired results of the study [33]. This could lead to reductions in the precision of the model; therefore, a balance has to be found between efficiency and accuracy. The 3 main features of the architecture of a Digital Twin are described in “Digital Twins Architecture” [35] as:

1. Element: Described as a virtual model, faithful to the physical system.

2. Behavior: Representing physics through computational models (such as movements and flows).

3. Integration: Connecting the virtual and physical system to allow synchronism between them.

In the scope of this project, data gathered from the real system was used for the creation of the digital model: both for the pipe network under the surface of the pitch (the element) and weather data to accurately represent the real-world behaviour. While not a full digital twin by the definition of the term, as data is not exchanged automatically between the 2 systems (integration), the virtual model used in this thesis is an accurate depiction which could be used for the purposes of optimising the real system.

2.10 Machine Learning

Machine Learning (ML) continues to cement its importance in modern engineering. Complex algorithms can help to make useful predictions from available data in many real-world engineering problems. Data is often specifically modified for use in ML algorithms, through processes such as data cleaning and scaling. The accuracy of an ML model can be determined by many calculated metrics, and these can be selected based on suitability. There are 2 categories in which ML methods fall: Supervised and Unsupervised. Within this study, Supervised learning is used exclusively. This is when data input and output are known and labelled. Unsupervised learning involves finding patterns in unlabelled data entries, such as linking samples with similar properties [36].

2.10.1 Regression and Classification

Regression and classification are two common forms of supervised learning. The difference between the methods lies in their prediction output. Regression predicts a numerical value, based on the values given in data columns for each data sample. This means that in regression, all column values must be numerical, and string data must be converted. Classification instead allocates each data entry to a category, and there is no requirement for all values to be numerical [36].

2.10.2 Training and Testing

Before data is used to influence decision-making in a Machine Learning algorithm, it must be correctly divided into training and testing data. Training data is used to inform the algorithm of the relationships between data columns (‘Features’) and their corresponding target values. Target values for the testing data are then predicted. Data rows (‘Samples’) are often split 80% 20%, with the majority being allocated to training. The more samples that are

included in the training data, the more accurate an algorithm's predictions become. A common way to separate data as required, is through the use of scitkit-learn's '$train\_test\_split()$' tool. The tool uses a '$random\_state$' parameter to identify set sample shuffle orders, when selecting which samples will be used for training and which will be kept for testing [37].

2.10.3 Machine Learning Models

There are many different forms of automated learning mechanisms for predictive data analysis, each with their own strengths depending on the desired outcome. The simplest form of solver is the linear model, where the required target, either category or number, is expected to have a linear relationship to the features, and a line of best fit is used for prediction [38].

The k-Nearest Neighbours model is an algorithm which can be used for both classification and regression. Unlike other methods which learn from training data, k-Nearest Neighbours memorises the training data and estimates the conditional probability for each class j, shown by:

$$P_j = \frac{1}{k}\sum_{N_0} I(y^* = j)$$

[20]

Where k is the training observations closest to the predictor value x_0, represented by the neighbor region, N_0, y^* is the class label and I is the indicator [39]. The main advantage of the k-Nearest Neighbours model is that it does not require a training phase unlike other algorithms, and it is relatively simple to understand and implement.

Naïve Bayes models utilise graphical solution methods, based on Bayes' Theorem of probability [36]. With an input of x, the classification process gives a maximised posterior probability as an output, shown by:

$$P(c|x) = \frac{P(X|C)P(c)}{P(x)}$$

[21]

Where $P(c|x)$ is the posterior probability, $P(c)$ is the prior probability of a class, $P(x)$ is the prior probability of the predictor and $P(X|C)$ is the e probability of the predictor for the class c [39]. The Naïve Bayes method is effective at working with smaller datasets and is quick, easy and simple to implement. However, its performance is not as high when compared to more advanced models, so it is primarily used as a comparison to validate these methods.

Support vector machines (SVM) can be used for regression or classification. They function by creating 'hyperplanes', which can separate data samples for prediction [40]. In a 2-dimensional problem where a solver must classify 2 groups of data from a dataset, the 'hyperplane' can be visualised as a simple straight line on a scatter plot. SVMs are capable of solving Regression and Classification problems that have many more dimensions, achieved by the

hyperplane operating in additional dimensions alongside the data. Mapping a higher dimensional space allows non-linear relationships to be understood. For regression problems, this solver has an additional 'epsilon' parameter that determines the error tolerance of the hyperplane function [41].

Ensemble methods are a group of machine learning algorithms which combine several solvers to improve prediction power [42]. Bagging and boosting are applied to the base estimator to reduce variance or bias. Bagging is a simple method for manipulating the given dataset. During each iteration the learning algorithm produces a bootstrap replicate training set which contains approximately 63.2% of the original data set and the remainder consists of subsets of randomly selected data samples [43]. Bagging is done to improve the overall performance accuracy of the selected machine learning method. An example of an ensemble method is Random Forest.

2.10.4 Decision Trees

Decision trees are Machine Learning algorithms which can be visualised similarly to flow-charts. They can help to identify trends and patterns in large datasets. They allow the user to observe all possible outcomes of a specified decision and can be used for both regression and classification problems. They are a supervised learning algorithm which are suitable for categorical and continuous output variables, without the need for feature scaling or normalisation [44]. Scaling within machine learning is defined as changing the data range to fit within a desired scale, for example between 0 and 100. Machine learning methods such as Support Vector Machines and K-Nearest Neighbours require scaling.

Unlike scaling (which doesn't affect the overall shape of the data), normalisation directly affects the distribution shape. It does this to form the data so that it follows a 'bell curve' and can be described as a normal distribution or Gaussian distribution. Approximately half of the observations then lie below the mean average and half the observations lie above, with the majority of data sitting close the mean value. Machine learning methods such as linear discriminant analysis and Gaussian naive Bayes require normalisation [45].

One advantage of decision trees over alternative machine learning models is the effective way in which the output can be graphically visualised. As the name suggests, decision tree modelling results in a 'tree' being produced. This tree is very similar visually to a flow chart and can be interpreted the same way. Within a tree, data is organised by several characteristics into small groups, creating the tree-like structure. Each final decision (category or numerical target) for a sample is called a 'leaf' [36]. Branches and nodes of conditions and results are generating during training, until a final end node is present with no other possible condition to branch off to. Figure 6 is an example of a decision tree which can determine the smallest of 3 numbers named a, b and c. The trees produced are easily interpreted and understood but as still extremely effective ways of displaying the required data as they

break down the decision-making process into many smaller straightforward and logical decisions [46].

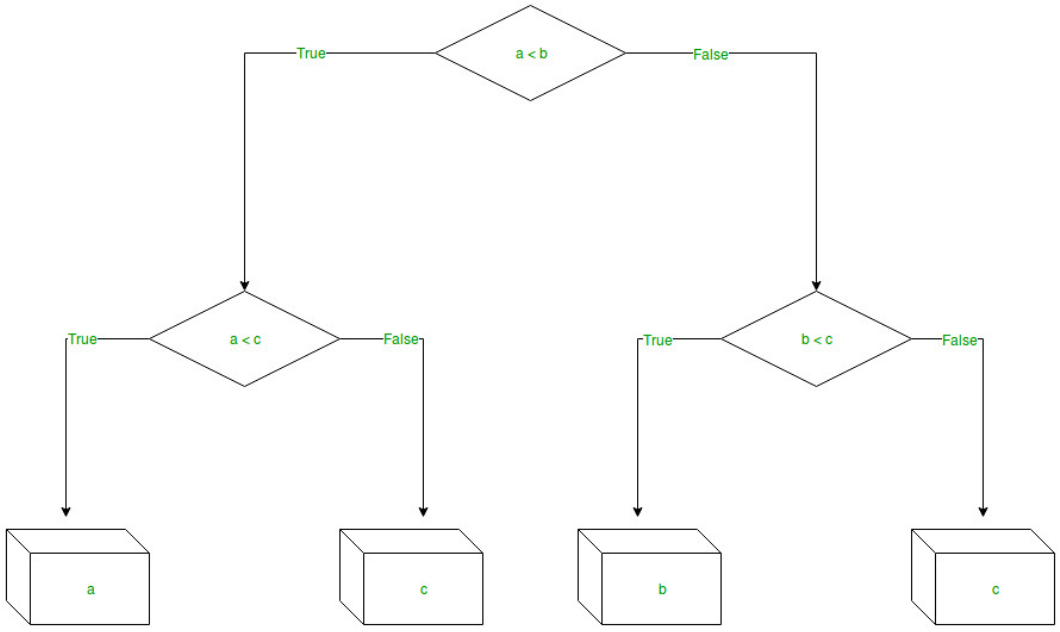


Figure 6: Decision Tree of 3 Numbers

Decision trees are initially developed by fitting the training data, x_train. This fitting considers the possibility of variations within the training data which are anomalies within the entire sample. Target values can then be found by introducing x_test, an independent sample dataset which evaluates the decisions through predictions [47].

Pruning is a process within the decision tree which reduces the depth of the output tree and replaces decision rules within with leaf nodes. This lowers the complexity of the tree and makes it easier to understand and implement. The decisions which are often removed are the ones that overfit and reduce the predictive accuracy.

Another advantage of decision trees over alternative machine learning models is the lack of data preprocessing required. This means that unlike other methods, input data is in a suitable form for training without initial analysis reducing the computational time of the overall modelling process [48].

Feature selection within the decision tree method is done primarily by taking the features with the most significant impact and splitting them. This can reduce the risk of overfitting as it limits decision splits from less impactful features, therefore improving the accuracy. Ensuring the decision tree splits the most optimal features can be achieved using scitkit-learn's '$sklearn.tree.DecisionTreeRegressor()$' [49]. The default setting for the splitter used is '$best$', but it can be changed to '$random$'. *Splitter* is the

strategy used to split decisions at each node so by using '$best$' it is ensured to pick the most optimal path.

Decision trees can also be used as the base estimator for ensemble methods such as Random Forests, Gradient Boost, and Adaptive Booster.

One limitation of decision trees is that they can be very sensitive to variations within data. If there are ever any changes, big or small, with a node located near the root of the tree which splits the results, the knock-on effect can result in a decision tree being produced with an entirely different shape and structure [46]. Another limitation is associated with the accuracy of decision tree modelling, which is not as strong in predictive accuracy as other models. The 2004 paper [47], "An Introduction to Decision Tree Modeling" evaluated the accuracy of a classifier decision tree and found that 9 out of 34 test samples were wrongly predicted partly due to overfitting of the training set.

Although there are many advantages of using ensemble methods such as excellent performance accuracy for complex and large problems, there are still some advantages for choosing a single decision tree. The outputs of single decision trees are far simpler to interpret and visualise and are less likely to overfit. For these reasons, the machine learning method chosen to establish optimal control guidance within this report is decision trees, as the scope of this project only requires "rule of thumb" guidance so does not require the level of depth produced by ensemble methods such as Random Forest.

3.0 Methodology

3.1 Thermodynamic Processes

Precipitation in the form of snow gathers on surfaces when temperatures are low and compresses into solid ice due to its own weight. When this ice reaches sufficiently high temperatures either due to solar heating or forced from underground heating systems as with this case the snow will melt. For a phase change from solid ice to liquid water to occur, a specific amount of energy must be available. This is known as the latent heat of fusion. During this process the temperature of the substance is constant until the phase change is complete. Figure 7 shows a simplified schematic of this phase change within the heated football pitch.

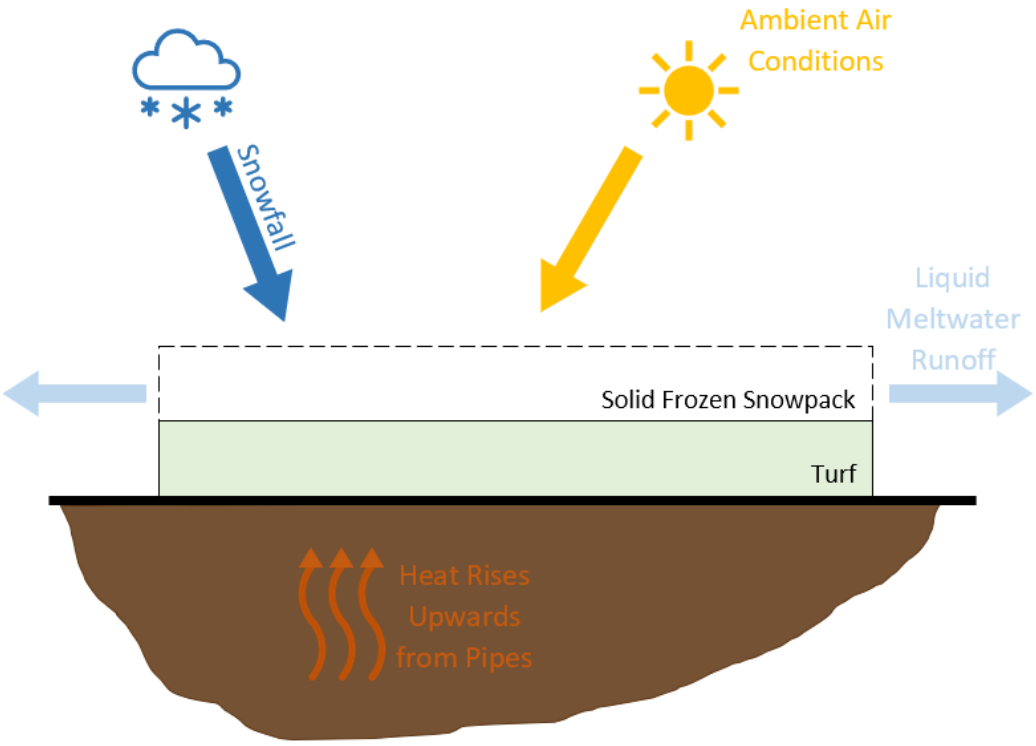


Figure 7: Simplified Phase Diagram of System

The pitch of interest from the original model (located in Naantali, Finland) is covered with impermeable artificial turf. This means that the meltwater from the snowpack cannot seep downwards directly through the soil and instead must run off the pitch entirely before penetrating. The pitch is designed in such a way that any surface water will run off to either side instead of gathering in puddles. The geometry of the pitch can be seen in Figure 8. It is sloped down the length of the pitch symmetrically through the central line at a gradient of 1%. Figure 8 is not drawn to scale and dimensions are exaggerated for clarity. Due to the small angle and comparatively large size of the football pitch the slope cannot be noticed by the players, but it is effective enough in aiding runoff precipitation in the desired direction [50].

Precipitation in the form of rain or melted snow can fall anywhere on the impermeable pitch and flow from the sides towards permeable ground where

it is absorbed. The effects of moisture under the pitch are considered and discussed in Section 3.2.1.

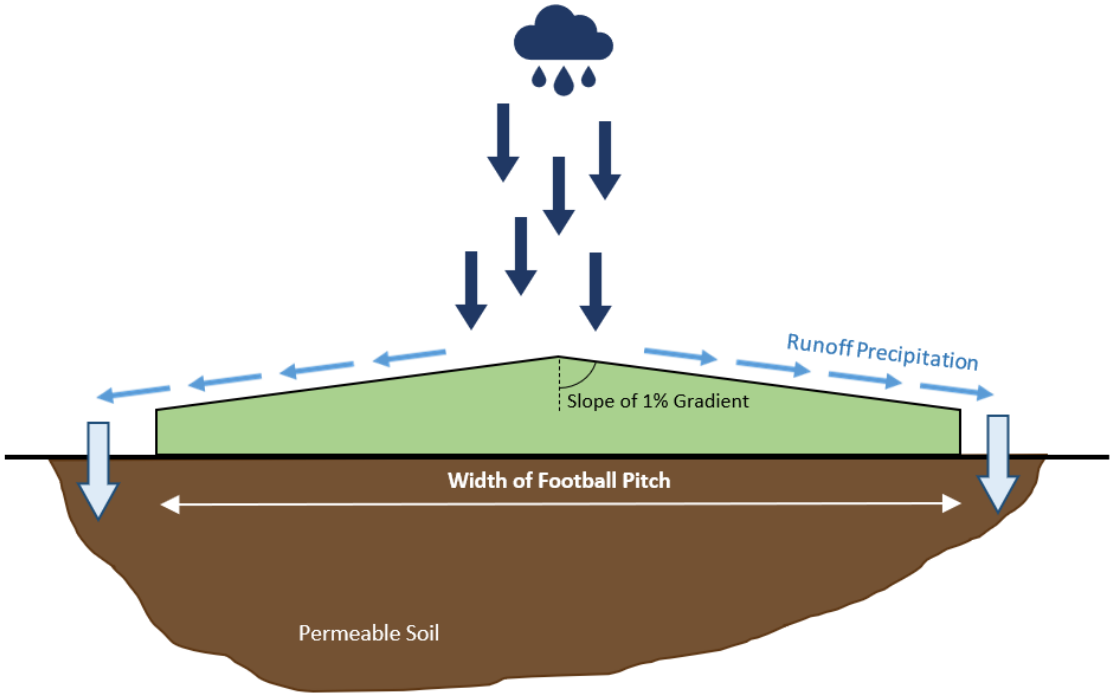


Figure 8: Side Profile View of Slope of Pitch (Not to Scale)

3.2 Melting Model Refinement

To improve the accuracy of the predicted values of snow depth by the computational model, refinements were made to the model's representation of real physics. These improvements included refining the heat-mass transfer interactions between different layers of soil, and the interactions between the snow surface and the outside atmospheric conditions. For this purpose, multiple layers of snow and the interactions between them through the generation of snowmelt water and the subsequent transportation of moisture into the ground were considered. Additionally, the imbalance of snow melting across the length of the pitch, created by the time delay in circulation, was addressed. Finally, a detailed comparison of the heated pavement model paper and implication of some attributes to the current model was created.

3.2.1 Introduction to Moisture Transport Simulation

The effect the movement of moisture had on the underground heating system's heat transfer properties, was investigated by modelling the transport of fluid within the computational model. Ansys Fluent was chosen as the best method for accurately modelling water movement. Ansys had a Volume of Fluids option, which allowed the simulation of 2 different types of fluid in a domain, modelling their interaction. For this study, the water-air interaction was in the form of surface tension, set to be constant at 0.072 N/m.

Properties were horizontally averaged across the pitch in the original model, therefore only half of the horizontal cross-section was modelled in Design

Modeller as the geometry could be assumed to be symmetrical. An outer soil domain was also modelled which represented the soil not located directly beneath the pitch. Unlike the soil directly below the pitch this outer soil did not have an impermeable layer overhead. The 2 soil domains were set as porous zones and given a porosity of 0.4 to match a typical porosity value for a sandy soil which ranges between 0.33 and 0.47 [51]. The soil was also set as a laminar zone, to reduce the complexity of the solution and improve simulation time. The solver type chosen was transient, with an adaptive timestep. With this type, complex fluid motion could be properly simulated, and in instances where fluid motion became mostly constant, the simulation would adjust to increase the timestep.

It was important to model the soil zones with a sufficient width and depth, to avoid negative effects associated with the pressure outlet boundary condition. Varying depths of 7m and 2m were analysed to determine the effect of the lower pressure outlet on accuracy, each with identical parameters. The mesh was adapted between cases to maintain resolution. To begin the test, the outer soil area was patched to be fully saturated with water. The drainage of water through the lower pressure outlet boundaries, of the inner and outer soil zones was captured using reports of area-weighted average water content.

The setup of the moisture simulation was a complicated process that spanned a large portion of the project. Several drafts of the setup were created with different criteria. Each attempt aimed to find the most accurate model of underground moisture movement to draw results from. This matched the nature of the project, completed in collaboration with members of the COMEA research group, who often suggested directions to take when attempting to increase accuracy. Finding results in a research-based project does not always follow a direct path, and this was reflected in the moisture transport subtask.

There were 4 main drafts of the setup, each taking several weeks to develop.

3.2.2 Moisture Transport Iteration 1

The first version involved a highly detailed recreation of rainfall above the pitch and outer soil. This was achieved by modelling a water tank zone, and a sky zone. The interface between these zones was set up with several small wall boundaries, causing water to fall in a similar way to raindrops. The main issue with this setup was the computational time required to run simulations. The large sky zone and high level of randomness in the water ultimately led to the requirement of a redrafted setup. Result recordings did not reach a suitable stage of flow time to be viable and were therefore omitted from consideration.

3.2.3 Moisture Transport Iteration 2

The second iteration of the setup sought to simplify the domain heavily, first by removing the sky and water tank zones. The new geometry was then finalised, shown in Figure 9.

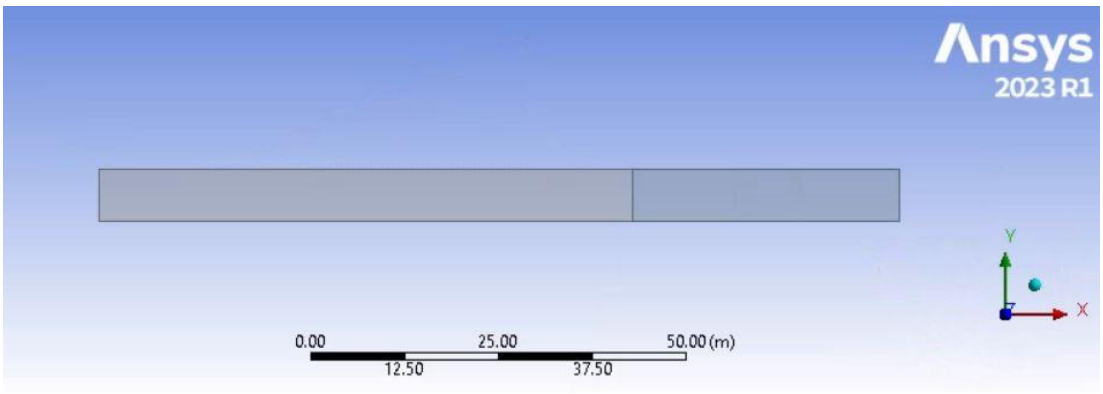


Figure 9: Geometry of 2nd Iteration Simulation

The updated geometry struck a balance between accuracy and computational time. A velocity inlet was added to the soil surface, to feed water directly. The vertical boundary beneath the pitch was set as type symmetry, keeping the other half of the pitch in consideration. To account for water flow quantity, 4 different velocity inlet cases were chosen: 0.5mm/h, 2.5mm/h, 5mm/h and 25mm/h. 0.5mm/h represented simple melt outflow, 2.5mm/h was light rain, 5mm/h was moderate rain, and 25mm/h was heavy rain or floodwater. For further representation of reality, the state of the soil under the system was included, increasing the number of cases. At first,4 cases were considered, 3 in which the gauge pressure at the lower pressure outlet boundaries was increased (0Pa, 50Pa, 100Pa). The last case assumed a solid wall with no liquid transfer across the lower boundaries. This resulted in 16 total scenarios. All 16 scenarios were calculated for a flow time of 1 hour, with monitors set to record plots of important parameters.

Another adaptation that took place before reflecting the results in the python model, was setting the velocity inlet to 0m/s. This was to prevent any artificial forcing of the water as it moved through the soil. The 0Pa case and Solid wall case were reconfigured to begin fully saturated in the outer soil zone, and slowly drain as flow time increased. The geometry with the fully saturated condition is shown in Figure 10:

Figure 10: Contour of Water Volume Fraction for a Fully Saturated Case at Flow Time 0s

To include the effects in the python model, a series of 'if' statements were added to the '$CalculateSnowAndRainMassFluxes$' function. Adding the thermal property updates to this function made it easy to define conditions, based on the liquid precipitation amount, variable 'Pr'. The user of the model would be required to set a value that indicates if the lower boundary is open or if it acts as a wall. This condition would then decide which case to draw

values from (0Pa or Solid Wall). As the 0Pa case resulted in negligible change for some parameters, the total number of 'if' statements required was less than that of the Solid Wall case.

3.2.4 Moisture Transport Iteration 3

In the third moisture model, to ensure that the movement of water would only be a function of gravity and soil properties, referred to as Unlimited flow [52], a water tank was modelled above the outer soil zone. Symmetry boundary conditions were chosen on either side of the zone, extending the water flow coverage. Switching from a velocity inlet to a water tank prevented any external forcing of the water, which affected the accuracy of results. New gauge pressure values were chosen for the second and third case within this iteration, of 5000Pa and 10000Pa. As a replacement for the constant inflow of the velocity inlet, a new strategy had to be developed. Maintaining an inflow from the water tank for a long flow time would have required a very large tank area. This in turn would have caused unwanted forcing of the water into the soil, by the tank's large body of water. To resolve these issues, it was decided that patching was the best solution. Using a macro, the water tank zone was patched to be fully saturated regularly, at every 0.1 seconds of flow time. Alongside this, the temperature was patched at 275.5K, 0.5K lower than the surrounding zones. As rain is often colder than the air it falls through, this patch made the simulation more realistic. Lastly, the velocity of the fluid in the water tank zone was patched to zero in both the vertical and horizontal direction, to minimise the impact of a large quantity of water, and any instability that may cause water to be forced under the soil. The third iteration's results indicated that pressure outlet gauge pressure variation did not provide a suitable range of simulated scenarios.

3.2.5 Moisture Transport Iteration 4

A final fourth moisture model was developed, focusing this time on the porosity of the soil. 3 new cases were devised. The first varying porosity case began with a surface layer of soil with 0.4 porosity, a deeper central layer with 0.3 porosity, and a bottom layer of 0.2 porosity. This was labelled the 'Decreasing Porosity' case. The other varying porosity case began with 0.2 porosity at the surface layer, 0.3 porosity at the central layer, and 0.4 porosity at the bottom layer. This was labelled the 'Increasing Porosity' case. The remaining case was labelled the 'Constant Porosity' case, with 0.4 porosity across all of the soil. Again, the cases were patched with a macro command, although this time a specific cell region was selected. A 40cm layer above the surface of the soil was highlighted, again bringing the water simulation closer to reality.

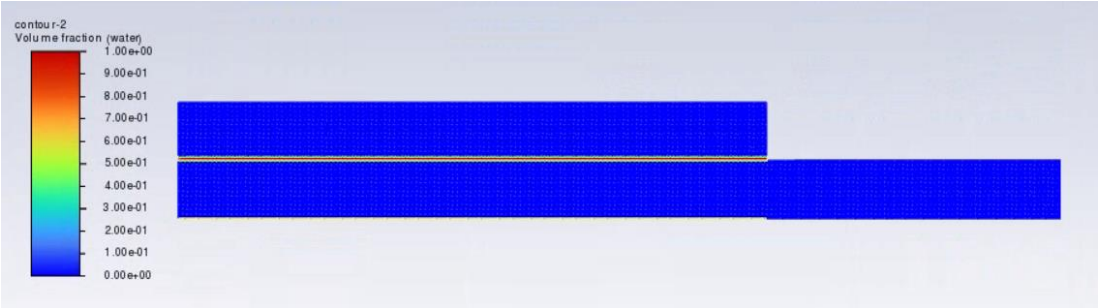


Figure 11: Geometry of 4th Iteration Simulation

The results of iteration found are presented and discussed in Section 4.1.

3.2.6 Time Delay

To understand the implications of time-delay, the geometry of the network of pipes had to be considered. The system consisted of loops of tubing $\sim 142m$ (71m each way) in length running across the width of the field. There were 220 loops in total, each connected to a main inlet and outlet line spanning the length of the pitch ($\sim 107m$). The tubing was also buried at 20cm depth.

An imbalance was created between melting of the snow on different sides of the field, as it took longer for sections further away from the inlet side to reach the desired temperature. As part of the original paper, the COMEA research team had developed a model of the pipe network in ANSYS. This contained the geometry and dimensions. However, the large scale of the model meant that it was incredibly computationally intensive and time consuming for the required level of accuracy, necessitating the use of the supercomputer facilities available to the university. Simple changes to the geometry or any of the other simulation parameters would have required a lengthy process of requesting the use of the supercomputer and waiting for results. Therefore, part of the project involved developing a simple and accurate way to test changes to the system inputs and track their impact on a much shorter timeframe and locally, meaning complicated simulations could be carried out once a final set of conditions had been decided or their use could be avoided altogether.

Different tools are capable of carrying out those simulations [53], however the $pandapipes$ package for python was selected for this purpose. After an empty network was created, the type and properties of the circulating fluid could be modified to those of the propylene-glycol solution used in the real-world system. The elements of the network were created starting with junctions, which behaved as connections between the rest of the components. The number of junctions were defined alongside the initial values of pressure and temperature for the calculation. A geodata parameter was included to provide coordinates for plotting the network and guiding the connections which were added later. The pipe elements were then added, joining at the already existing junctions. Pipe dimensions and additional considerations such as pipe roughness, pressure loss coefficient as the fluid travels around bends and ambient conditions were also defined.

The network consisted of an inlet and an outlet section which were larger in diameter, connected by a series of smaller diameter loops. A pump

connecting the inlet and the outlet lines induced a fixed pressure and temperature at the outlet side while requesting a specific flowrate at the inlet side. The flow could then be simulated for a defined number of time steps, and the resulting temperature and pressure at the junctions were recorded.

Figure 12 is a schematic of the network. For simplicity and clarity, a system consisting only the first 5 loops is presented:

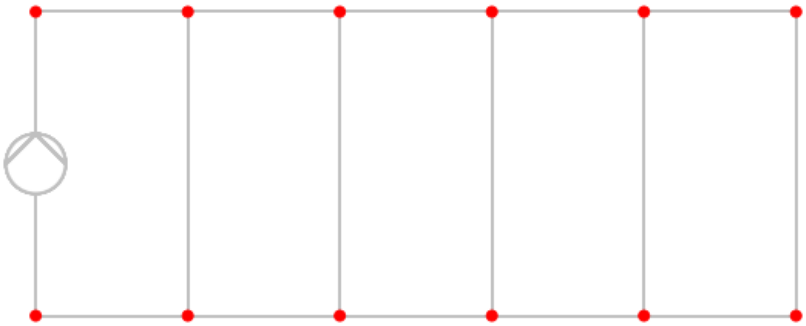


Figure 12: Simple Schematic of Pipe Network System

As hot fluid is supplied through the network, convective heat transfer occurs between the fluid and the pipes. In turn, conductive heat transfer takes place with the surrounding soil, increasing the temperature until the surface level has reached a sufficient temperature to melt the accumulated snow. The aim of the project is to optimise this process so that the field can be kept in playing conditions year-round while minimizing the cost of running. Therefore, the time taken between inducing a temperature difference, ΔT_f, at the inlet and the whole system reaching this state could have significant implications on energy consumption [54].

Another condition impacting the performance of such systems will be frictional losses in the piping, such as due to the introduction of bends as well as imperfections on the surface of the pipes. This was approximated by the addition of a pressure loss coefficient to the calculation, as shown in Figure 13 from "*Steam, its generation and use"* by The Babcock & Wilcox Company [55].

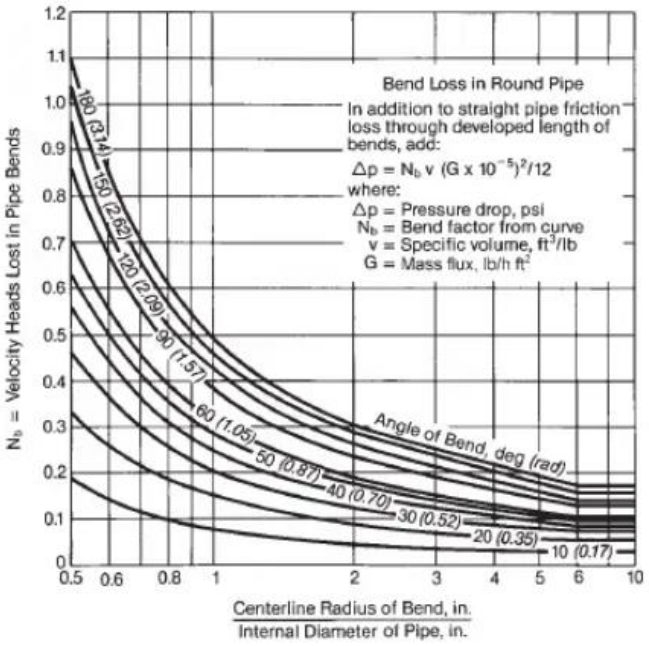


Figure 13: Pressure Losses in Pipe Bends

Using Figure 13 and the dimensions of the pipe, extracted from an ANSYS model developed by the COMEA Research Group, the pressure loss coefficient was estimated using the following method:

The internal diameter of pipe, as in the model provided was taken as: $d_i = 22mm = 0.866\ in$.

The centreline radius of bend was approximated from pipe segments with similar dimensions as: $r_c = 36mm = 1.41\ in$, from [56] [57], and assuming the angle of bend as $180°$ the final calculation could be performed:

$$\frac{r_c}{d_i} = \frac{36mm}{22mm} = 1.36 \Rightarrow N_b = 0.36$$

[22]

Another key variable was the heat transfer coefficient between the pipes and the surrounding soil. This depends on a large number of factors, including the material characteristics of the pipes, the composition of the soil and the heat conducting medium. Taking those issues into consideration, the heat transfer coefficient was estimated to be around $\alpha = 10\ W/m^2K$ [58]. Both parameters were included in the definition of the pipe elements within the network, giving significant control over the behaviour they exhibited.

The rest of the input parameters to the simulation are summarised in Table 3:

Table 3: Input Parameters for Time-Delay Simulation

Parameter	Value
Ambient Temperature, T_{amb}	$0°C$
Pump Pressure, P_{bar}	$3.6\ bar$
Temperature Difference Induced by Pump, ΔT_f	$15°C$
Mass Flow Rate of Solution, $\dot{m}$	$1kg/s$

The simulation was carried out for a number of different input conditions to include variations in ambient temperature, mass flow rate and the temperature difference between the inlet and the outlet of the network. The results of this study were recorded in Section 4.2.

3.2.7 Incorporation of the Heated Pavement Model

The 2007 paper, [2], which introduced a computational heat and mass transfer model for an underfloor heating system on a pavement, previously discussed in 2.2.2, included many features and considerations which were thought to be relevant in the creation of the improved model developed in this report.

Both the original model and the pavement model followed a similar structure which included an underground heating element at a specified depth with the purpose of providing heat to the external surface, either pavement or artificial grass to melt the acclimated snow which gathers on the surface.

There were several external factors which contributed to the heat and mass balance equations of both individual models, such as ambient air conditions, cloudiness, precipitation, solar heating, and sky radiation heat losses. Heat conduction from the surface layers also contributed to the balance equations.

The first difference between both models which was incorporated was the consideration of heat transfer of evaporating water or condensed water vapor which is given by:

$$\dot{Q}_{Evap/Cond} = h_{LV} \cdot \frac{\dot{m}}{A_{xy}} (w_{air} - w_{sur})$$

[23]

Where w_{air} and w_{sur} are the humidity ratios of the ambient air and the saturated air at the slush surface, respectively.

$\dot{m}$ is the Mass Flow Rate of Liquid into the Surface $[kg/s]$ given by:

$$\dot{m} = \mathrm{Ps} * \rho_{water}$$

[24]

Where ρ_{water} is the density of water at $1000\ kg/m^3$.

When the calculated $\dot{Q}_{Evap/Cond}$ value is positive the model can be assumed to be subject to evaporation and when negative, condensation.

Equation [23] was included in the Python code within the heat balance equations, shown in Figure 14. This allowed for $\dot{Q}_{Evap/Cond}$ to be included within the computational model as part of Equation [5], and the effect of such is presented and discussed within 4.3. The inclusion of $\dot{Q}_{Evap/Cond}$ within the Python code can be seen in Figure 15.

```
Qevap_cond = WaterHeatOfVaporization * ( m_liquid_in /A_xy) * (w_air - w_pv)
```

Figure 14: Python code excerpt of Equation [20]

```python
# Layer 2 - Gravel fill
Q23 = (k_23/dx_23)*A_xy*(T2s-T3s)
eq2 = m_2*cp_2*(T2e-T2s)/dt - Q12 + Q23 + (k_2/np.sqrt(np.pi*alpha_2*dt))*(T2s-T_g_next_2)*A_2_sides # -/+ Convection of moisture

# Layer 3 - Turf
if (SnowDs < 0.0001) and (Precipitation<0.000000001): # No snow, no rain - no need for internal iterations

    Q34 = 0
    Qcond = 0
    Qrad = -epsi*StefanBoltzmann*((T3s+273.15)**4-(Tsky+273.15)**4)*A_xy
    Qconv = h*A_xy*(AirTemperature-T3s) # < 0 if T_air < T3e
    QsolarAvg = Qsun*A_xy # > 0

    Qevap_cond = 0

    eq3 = m_3*cp_3*(T3e-T3s)/dt - Q23 + Q34 - Qconv - Qrad - QsolarAvg  - Qcond- Qevap_cond
    eq4 = SnowWaterMass_new - 0
    eq5 = Snow_Depth_new - 0
    eq6 = Liquid_Fraction_new - 0
    eq7 = TSe - 0

    return [eq1, eq2, eq3, eq4, eq5, eq6, eq7]
```

Figure 15: Python code excerpt containing $\dot{Q}_{Evap/Cond}$ in the heat balance.

Different snow classifications within the current model were also investigated.

Considering the properties of 4 different snowpack classifications: Dry Snow, Slush Only, Wet and Dry. A counter was included within the Python code in the form of if statements which allowed for the total number of times snow is a particular classification across the total time period to be graphed. This section of code is shown in Figure 18. Figure 16 and Figure 17 show the classifications of snow before and after any melting due to external factors and the underfloor heating system had occurred.

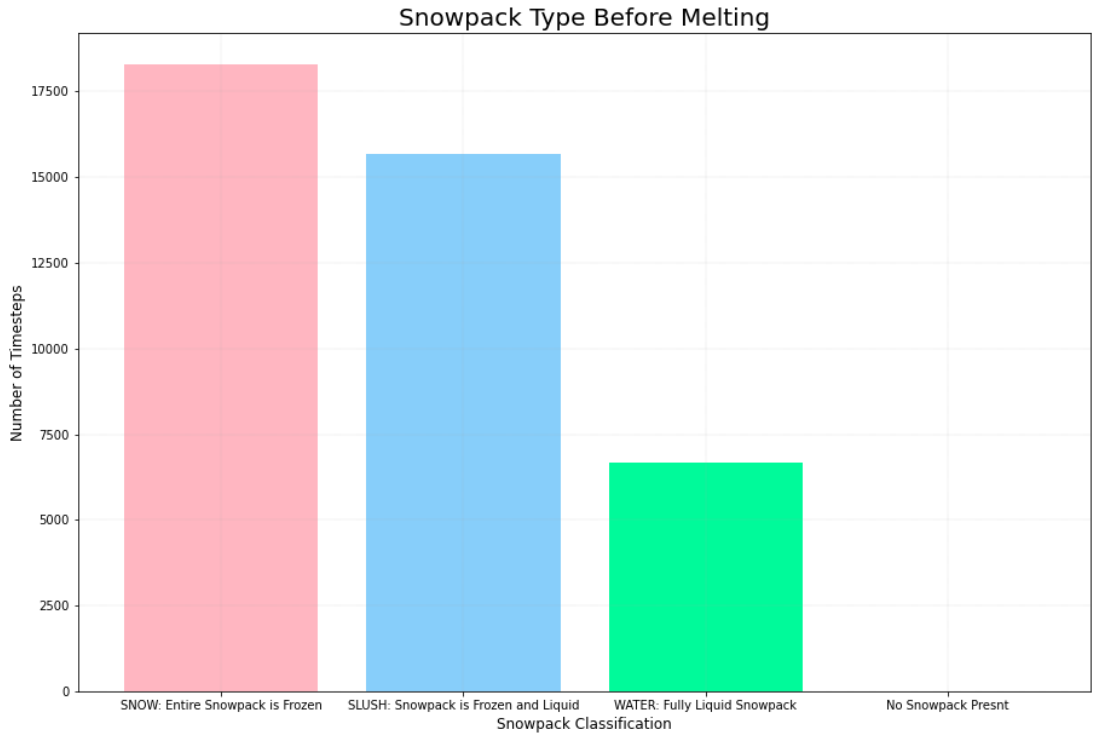


Figure 16: Snowpack Classification Before Melting

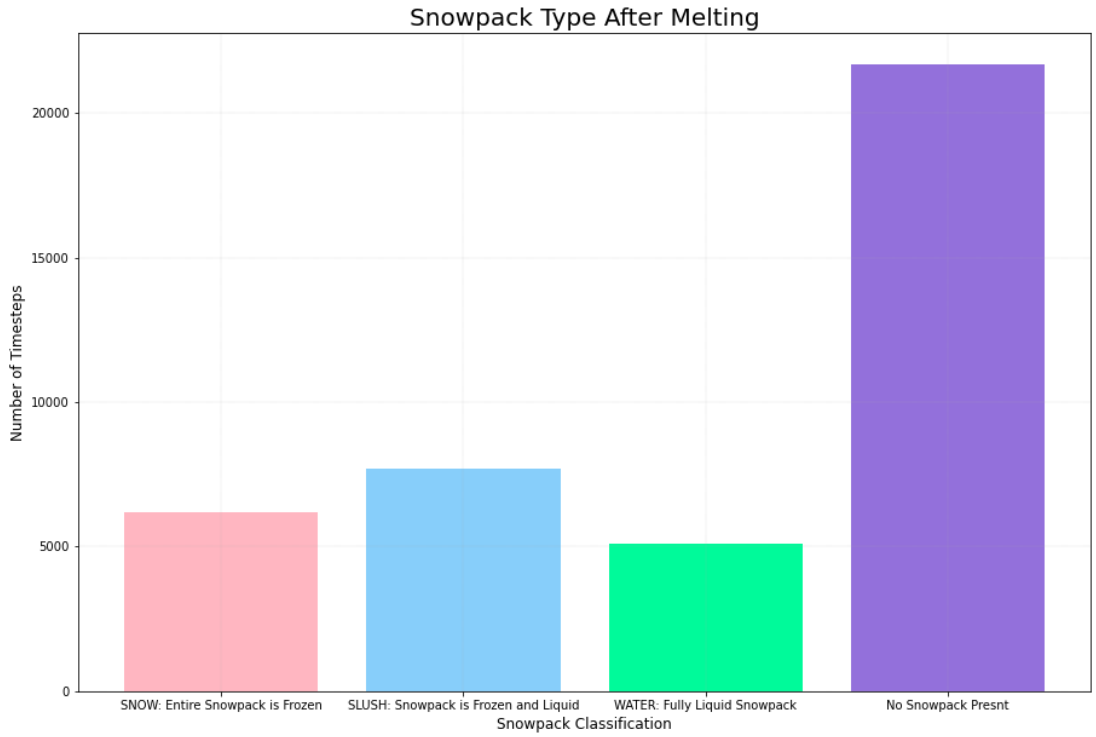


Figure 17: Snowpack Classification After Melting

It could be observed that prior to melting, the majority of the snowpacks fell within the Dry Snow or Slush only classification. After melting had occurred only a small percentage of the snowpack remained un-melted, and the majority of the snowpack was no longer present. This validated the effectiveness of the overall underground heating system at being able to melt any snow present on the surface to a playable condition.

```
Frozen_Aft = Frozen_Initial_Aft
Some_Frozen_Aft = Some_Frozen_Initial_Aft
Some_molten_Aft = Some_Molten_Initial_Aft
Molten_Aft =  Molten_Initial_Aft
other_Aft = other_in_Aft
Total_Aft =  Total_in_Aft
T_a = 3
U_melt = mSnowNew*WaterHeatOfFusion*1000
U_freeze = mWaterCurr*WaterHeatOfFusion*1000

if AirTemperature >= T_a:
    Frozen_Aft = Frozen_Aft + 1
    Frozen_Initial_Aft = Frozen_Aft

elif (U_new >= -U_freeze) and (U_new <= 0):
    Some_Frozen_Aft = Some_Frozen_Aft + 1
    Some_Frozen_Initial_Aft = Some_Frozen_Aft


elif (U_new > 0) and (U_new <= U_melt):
    Some_Frozen_Aft = Some_Frozen_Aft + 1
    Some_Frozen_Initial_Aft = Some_Frozen_Aft

elif (U_new > 0) and (U_new <= U_melt): #SOLID AND LIQUID MIXTURE # Some but not all of snow is molten
    Some_molten_Aft = Some_molten_Aft + 1
    Some_Molten_Initial_Aft = Some_molten_Aft

elif (U_new >= U_melt): #LIQUID # Snowpack fully molten - T > 0, LiquidFraction = 1
    Molten_Aft = Molten_Aft + 1
    Molten_Initial_Aft = Molten_Aft

else:
    other_Aft = other_Aft + 1 #DRY CONDITIONS NO SNOWPACK
    other_in_Aft = other_Aft

Total_Aft = Total_Aft + 1
Total_in_Aft= Total_Aft
```

Figure 18: Python code excerpt with Counters for Snow Type

3.2.8 Negation of Snow Distribution

The effect wind had on falling snow was considered within this report. When snow falls in climates with high wind speeds it is prone to drifting. This means gathering in mounds at areas and resulting in variation on the depth of the snowfall, specifically on flat surfaces such as football pitches [59]. "Numerical simulation and wind tunnel test for redistribution of snow on a flat roof" discusses and tests the distribution of snow on a flat roof within a wind tunnel which can simulate various wind speeds [60]. The flat roof within the tests discussed above can be considered as an equivalent to a flat football pitch, although the need to acquire this information differs. Structural engineers require to know the location and volume of snowfall on specific areas of a flat roof to allow for adequate support for the roof to withstand any pressures. In this report it was important to understand the distribution of snow on a football pitch to accurately describe a computational snow model which is realistic to the various types of snow previously discussed. For example, a thicker layer of snow will be heavier and therefore compress the air at a greater rate resulting in chunks of ice forming, the behaviour of the snow will be taken into consideration throughout.

The effect the wind has on the snow distribution across the football pitch will be negated throughout this report. The minimum wind speed considered in [61] is 7m/s, which has very little effect on the movement of snow. The average wind speed at the Naantali football pitch was assumed the same as that of the Turku Artukainen weather observation station, as that was the nearest observation station located around 10km away. Between December

1st 2022 and March 1st, 2023, the average wind speed was 2.86 m/s, [62]. Appendix B – Average Wind Speed and Appendix C – URL for Finnish Meteorological Institute Website (Download Observations) show the inputs and URL for the Finnish Meteorological Institute: Download Observations webpage from which the wind speed data was taken.

The wind's effect on the distribution of snow was negated within this report due to 2 reasons. Firstly, due to the average wind speed being significantly lower than that which was tested within the wind tunnel flat roof example and the effects being minimal at this higher speed and secondly, considering the altitude at which wind speed was measured compared to ground level where snow lies and the interference of surround buildings and ground elevations it could be assumed that the rate of wind speed at ground level was smaller and less significant.

3.3 Sensitivity Analysis

3.3.1 Variable Study

The sensitivity analysis allowed for variables which have the biggest effect on results to be identified, and those which have little, or no impact could be neglected. Eleven variables were included in this study and for each an array of 5 values were implemented, shown in Figure 19. The variables considered were Snow Thermal Conductivity, Turf Surface Emissivity, Snow Liquid Holding Capacity, Bed Total Depth, Snow Surface Emissivity, Wind Factor Correction, Snow Saturated Hydraulic Conductivity, Damping Parameter, Snow Liquid Fraction Initial, Field Midpoint Temperature and Air Density. The variables were chosen on the basis of exploring the impact of parameters which are more obscure and therefore much more difficult to measure in the real-world, rather than ones which have been analysed extensively. This gave a more complete overview of how the calculations were impacted by the various data inputs.

Using a loop, the code would run for each value in the variable array and hex-graphs were plotted for each variable against snow depth prediction. The plots and results are presented and discussed in Section 4.5.

```
# {Sensitivity Analysis Values}
SnowLiquidHoldingCapacity_values = [0.05, 0.1, 0.2, 0.3, 0.4]
SnowLiquidFractionInitial_values = [0, 0.25, 0.5, 0.75, 1]
SnowThermalConductivity_values = [0.024, 0.2, 0.4, 0.6, 0.8]
TurfSectionDensity_values = [900, 1034.3, 1100, 1200, 1300]
FieldMidpointTemperature_values = [-1, 0, 1, 2, 3]
AirDensity_values = [1.192, 1.202, 1.212, 1.222, 1.232]
meltinG_exp_values = [0.05, 0.1, 0.15, 0.2, 0.25]
SnowSaturatedHydraulicConductivity_values = [0.005555556, 0.0069444, 0.008333333, 0.009722222, 0.011111111]
BedTotalDepth_values = [1, 2, 3, 4, 5]
WindFactorCorrection_values = [0.8, 0.9, 1, 1.1, 1.2]
SnowSurfaceEmissivity_values = [0.1, 0.2, 0.3, 0.4, 0.5]
TurfSurfaceEmissivity_values = [0.2, 0.4, 0.6, 0.8, 1.0]
```

Figure 19: Python code excerpt of Sensitivity Analysis Values

```
# SENSITIVITY ANALYSIS NO. 1
SAoutputs1.append((SnowLiquidHoldingCapacity, D_snow_curr))
Hexvalues1.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 2
SAoutputs2.append((SnowLiquidFractionInitial, D_snow_curr))
Hexvalues2.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 3
SAoutputs3.append((SnowThermalConductivity, D_snow_curr))
Hexvalues3.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 4
SAoutputs4.append((TurfSurfaceEmissivity, D_snow_curr))
Hexvalues4.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 5
SAoutputs5.append((FieldMidpointTemperature, D_snow_curr))
Hexvalues5.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 6
SAoutputs6.append((AirDensity, D_snow_curr))
Hexvalues6.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 7
SAoutputs7.append((meltinG_exp, D_snow_curr))
Hexvalues7.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 8
SAoutputs8.append((SnowSaturatedHydraulicConductivity, D_snow_curr))
Hexvalues8.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 9
SAoutputs9.append((BedTotalDepth, D_snow_curr))
Hexvalues9.append(D_snow_curr)

# SENSITIVITY ANALYSIS NO. 10
SAoutputs10.append((WindFactorCorrection, D_snow_curr))
Hexvalues10.append(D_snow_curr)
```

Figure 20: Python code excerpt of Sensitivity Analysis Results and Plotting

3.3.2 Implicit and Predictor-Corrector Method

The second stage of the sensitivity analysis involved changing the solver method inside several functions. Within the python code the calculation of 4 different variables were changed from the original explicit solver to both an implicit and a predictor-corrector solver. The changed variable methods were for the calculation of S_star, $mSnowCurr$, $mWaterCurr$ and $mTotNew$.

The solver was also changed to Modified-Euler Method, a predictor-corrector method. Predictor-corrector solvers use a combination of explicit and implicit techniques to obtain the desired solutions.

Consider the ODE:

$$\frac{dy}{dx} = f(x, y)$$

[25]

The y value is predicted for the next step using:

$$y_{t+1,p} = y_t + h \cdot f(x_t, y_t)$$

[26]

Where h is the step size.

Using the predicted result, a corrected value is determined.

$$y_{t+1,c} = y_t + h \cdot \frac{f(x_t, y_t) + f(x_{t+1},\ y_{t+1,p})}{2}$$

[27]

Figure 21 shows a section of the python code updated for predictor-corrector, the remainder of the code follows the same format as Appendix G – Python Code: Original. The overall results from the snow melting model and the weather predictions were not as close to the real weather data as they were with the original explicit model.

The solver was also changed to the Backward Euler Method, an implicit numerical method capable of solving ordinary differential equations. Figure 22 shows a section of the python code updated for implicit, the remainder of the code again follows the same format as Appendix G – Python Code: Original.

```
def MeltOutFlowFromSnowPack(LiquidFraction, Area): #Y
    #  Melt outflow is a function of the liquid fraction, using Darcy's law
    #  Utah Energy Balance Snow Accumulation and Melt Model (UEB) page 20/64
    #  Ksat is the snow saturated hydraulic conductivity
    #  S* is the relative saturation in excess of water retained by capillary force, Male and Gray (1981, p. 400, eqn 9.45).
    global Sstar_interm
    K_sat = SnowSaturatedHydraulicConductivity # [m/s]
    Lc = SnowLiquidHoldingCapacity # [-]
    Lf = LiquidFraction # [-]
    rho_i = IceDensity
    rho_s = SnowDensity
    rho_w = WaterDensity
    if (Lf > Lc) & (Lf < 0.99):
        S_star_curr = ((Lf/(1-Lf))-Lc)/((rho_w/rho_s)-(rho_w/rho_i)-Lc)
    else:
        S_star_curr = 0 # water is retained/embedded/diffused in snow

    S_star_y = meltinG_exp*Sstar_prev
    S_star_x = (1-meltinG_exp)*S_star_curr

    ts=0.08

    S_star_P = S_star_y + ts*(S_star_x + S_star_y)
    S_star_C = S_star_y + ts*(1/2)*((S_star_x + S_star_y)+((S_star_x + ts) + S_star_P))
    S_star = S_star_C

    #M_r = K_sat*S_star^3 # [m/s]
    M_r = K_sat*S_star*S_star*S_star # [m/s]
    V_out = M_r*Area # [m^3/s] volume flow rate of liquid water out of snow pack, T = 0C
    m_out = rho_w*V_out # [kg/s]
    Q_out_kW = WaterHeatOfFusion*m_out # [kJ/s] = kW
    Q_out = 1000*Q_out_kW # [W]

    Sstar_interm = S_star

    return m_out, Q_out
```

Figure 21: Python code excerpt of a Predictor Corrector function

The backward Euler methods computes the y-value for the next step using:

$$y_{t+1} = y_t + h \cdot f(x_{t+1}, y_{t+1})$$

[28]

The results of implementing alternative numerical methods are presented and discussed in 4.5.2.

```python
def MeltOutFlowFromSnowPack(LiquidFraction, Area):
    #  Melt outflow is a function of the liquid fraction, using Darcy's law
    #  Utah Energy Balance Snow Accumulation and Melt Model (UEB) page 20/64
    #  Ksat is the snow saturated hydraulic conductivity
    #  S* is the relative saturation in excess of water retained by capillary force, Male and Gray (1981, p. 400, eqn 9.45).
    global Sstar_interm

    K_sat = SnowSaturatedHydraulicConductivity # [m/s]
    Lc = SnowLiquidHoldingCapacity # [-]
    Lf = LiquidFraction # [-]
    rho_i = IceDensity
    rho_s = SnowDensity
    rho_w = WaterDensity
    if (Lf > Lc) & (Lf < 0.99):
        S_star_curr = ((Lf/(1-Lf))-Lc)/((rho_w/rho_s)-(rho_w/rho_i)-Lc)
    else:
        S_star_curr = 0 # water is retained/embedded/diffused in snow

    S_star = meltinG_exp*Sstar_prev+(1-meltinG_exp)*S_star_curr
    S_star_y = meltinG_exp*Sstar_prev
    S_star_x = (1-meltinG_exp)*S_star_curr
    # # h=0.001
    h=0.1
    S_star = Sstar_prev*meltinG_exp + h*(S_star+(S_star_x + h))

    M_r = K_sat*S_star*S_star*S_star # [m/s]
    V_out = M_r*Area # [m^3/s] volume flow rate of liquid water out of snow pack, T = 0C
    m_out = rho_w*V_out # [kg/s]
    Q_out_kW = WaterHeatOfFusion*m_out # [kJ/s] = kW
    Q_out = 1000*Q_out_kW # [W]
    Sstar_interm = S_star
    return m_out, Q_out
```

Figure 22: Python code excerpt of an Implicit function

3.3.3 Variation of Time Step Size

A sensitivity analysis was performed to determine the optimal time step size to acquire predicted snow depth results which compare closer to real world weather data and the current snow melt model. Smaller step sizes give better accuracy and stability but require greater computational time.

The code was run for 11 different time step sizes, including the original time step size of 60 minutes. For each different time step size, the code was set to run 3 times and the computational time for each was recorded. Graphs documenting the predicted snow depth compared to the recorded snow depths were also produced. The computational times for each size were averaged and along with the graphs produced were analysed to determine the time step size which gives the best accuracy whist maintaining an appropriate computational time. The results from this are shown and discussed in 4.5.3.

3.4 Optimisation of Heating System Operating Profile

3.4.1 Design of Experiments

The purpose of carrying out the Design of Experiments (DoE) was to create a set of randomised weather scenarios which could be used in the Decision Trees Machine Learning Model, to then produce the optimal control inputs for temperature and liquid flow rate. This would allow the testing to capture a wide range of conditions including extremes, which are unlikely to occur in the real world, and monitor the stability of the system response.

LHS was selected as the method to carry out the DoE, the underlying process behind which is described in detail in Section 2.8.

To briefly describe how the DoE was carried out, the weather data was imported into Python as a csv file. From this, the maximum and minimum values for each weather parameter were extracted and set as the limits for the LHS method. The parameters chosen for this were: Cloud Amount (1/8),

Precipitation Amount (mm), Precipitation Intensity (mm/h), Snow Depth (cm), Air Temperature (℃) and Wind Speed (m/s). The control inputs, ΔT (℃) and Liquid Flow Rate (l/s) were also included in the random sampling as this provided additional considerations to the decision tree model.

The sampling was performed for 168 iterations, generating a range equivalent to 1 week of data. Due to the random nature of the sampling, some additional constraints had to be introduced to accommodate for the creation of unrealistic weather scenarios, such as heavy rainfall when Cloud Amount = 0, or decreasing the control inputs when snow cover is high. Precipitation Amount and Precipitation Intensity had to also be maintained within a small range of each other as Precipitation Intensity is simply the average of the Precipitation Amount over 1 hour.

To measure the efficiency of the randomly selected control inputs for the random weather samples, both inputs were made constant for the 1-week duration, and the new data was used in the code for the original simulation to extract the final values for the energy consumption and snow depth. The aim of the DoE was to minimise the energy consumption while also minimising the snow depth, therefore it was important to identify where the limits for those parameters should be set. The threshold value for the depth of snow at which the field could still be considered in playing conditions was identified as $0.1\,cm$. For energy consumption, the required value was selected as the average of the random samples. Cases where the control inputs produced results which were below the threshold for both parameters were assigned a value of 1, and the rest of the cases were assigned 0. This True/False designation was implemented to help significantly simplify the machine learning training process.

This method was carried out in 2 different stages. The first stage involved running this setup to generate 500 data points, each containing the constant weather and control inputs alongside final energy consumption and snow depth after a 1-week simulation. The ranking system described above was then implemented, creating the input required for training the decision tree model. After the training was carried out and the results were validated, as described in Section 3.4.2, the Design of Experiments was expanded to 3000 samples and the training process was repeated with the aim of improving the accuracy of the model.

3.4.2 Decision Trees

Using an automated learning mechanism in the form of a regression decision tree, trends in weather conditions against ΔT [℃] and Liquid Flow Rate $[kg/s]$ of the fluid in the pipes were identified.

The simulated weather data and control inputs were imported into the python code in the form of the output csv file from the Design of Experiments. This provided an array of values which, with the array of True/False designations assigned based on the corresponding values of final snow depth and energy consumption, were split into random train and test subsets as part of the $'sklearn.model\_selection.train\_test\_split'$ tool for python. The weather data and control inputs contained the features (x), and the ranking made up the

labels (y). Each was then split again into x_train, x_test, y_train, and y_test. The train dataset accounted for 80% of the data and was used for training the model to predict. The test dataset accounted for the remaining 20% of the data and was used to test if the predictions were returning correct values.

The regression model was initialised using a $'random\_state'$ parameter, which was used to control the random number generator. Working similarly to the way it was used for '$train\_test\_split()$', this parameter was important in ensuring reproducibility of the results. A value of $None$ for the random state would have produced a new set of decisions each time the script was executed. Using an integer instead allowed results to be reproducible over multiple simulations. Before the results of the large dataset could be analysed for the creation of the guidance document, suitable validation had to be carried out to confirm the accuracy of the predictions. For this purpose, a smaller set of only 10 data points was created using the DoE method described in Section 3.4.1. The algorithm, only trained on the large dataset, was then used to predict whether the control inputs would produce the required level of snowmelt for energy consumption below the average value. The new predictions, for a dataset which the algorithm was not trained on, were found to be at least 80% accurate, and even higher in some of the test cases. Due to the complicated nature of the relationship between the weather parameters and control inputs this level of accuracy was deemed acceptable, as small differences in any of the conditions could yield significant changes in the results.

The output of the script was in the form of a flowchart, which could be plotted using $webgraphviz$, a software package for the visual representation of decision trees. However, the resulting flowchart had to be arranged and simplified so it could be presented in the form of a guidance document which can be used by a non-specialist human operator to select the most optimal control actions for any given weather conditions. This was done manually, by following the different branches of the decision tree until a predicted value of 1 was reached and tracing back the weather parameters and control inputs which were involved in reaching that outcome. Once all the cases were collected, they could be grouped by weather parameters for the final document.

```python
# 1) Import the Required Libraries.
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import export_graphviz
from sklearn.model_selection import train_test_split

#2) Import/Format and Print the Weather Data.
dataframe = pd.read_csv('Decision_Tree_Inputs_3000.csv')#, index_col=0)
weatherdata = dataframe.to_numpy()
Weather_data = weatherdata [:, 1:9]
FinalSnowDepth_EnergyCon = weatherdata[:, 11:13]
X_train, X_test, y_train, y_test = train_test_split(Weather_data, FinalSnowDepth_EnergyCon, test_size=0.2, random_state=1)

#3)
# Initializing the Decision Tree Regression Model
Rg = DecisionTreeRegressor (random_state=0)

# Fitting the Decision Tree Regression model to the data
Rg.fit(X_train, y_train)

# Predicting the target values of the test set
y_pred = Rg.predict(X_test)
dataframe2 = pd.read_csv('Validate_Inputs_3000.csv')#, index_col=0)
weatherdata2 = dataframe2.to_numpy()
XWeather = weatherdata2 [:, 6:14]
Y2= Rg.predict(XWeather)

#4) Results
export_graphviz(Rg, out_file ='TreeResults_3000.dot',
          feature_names =['Cloud Amount', 'Precipitation Amount', 'Precipitation Intensity', 'Snow Depth', 'Air Temperature'
#To generate tree  flowchart. Open file ^ in word. Copy text and paste into website:http://www.webgraphviz.com/
```

Figure 23: Python code excerpt of Decision Tree Training

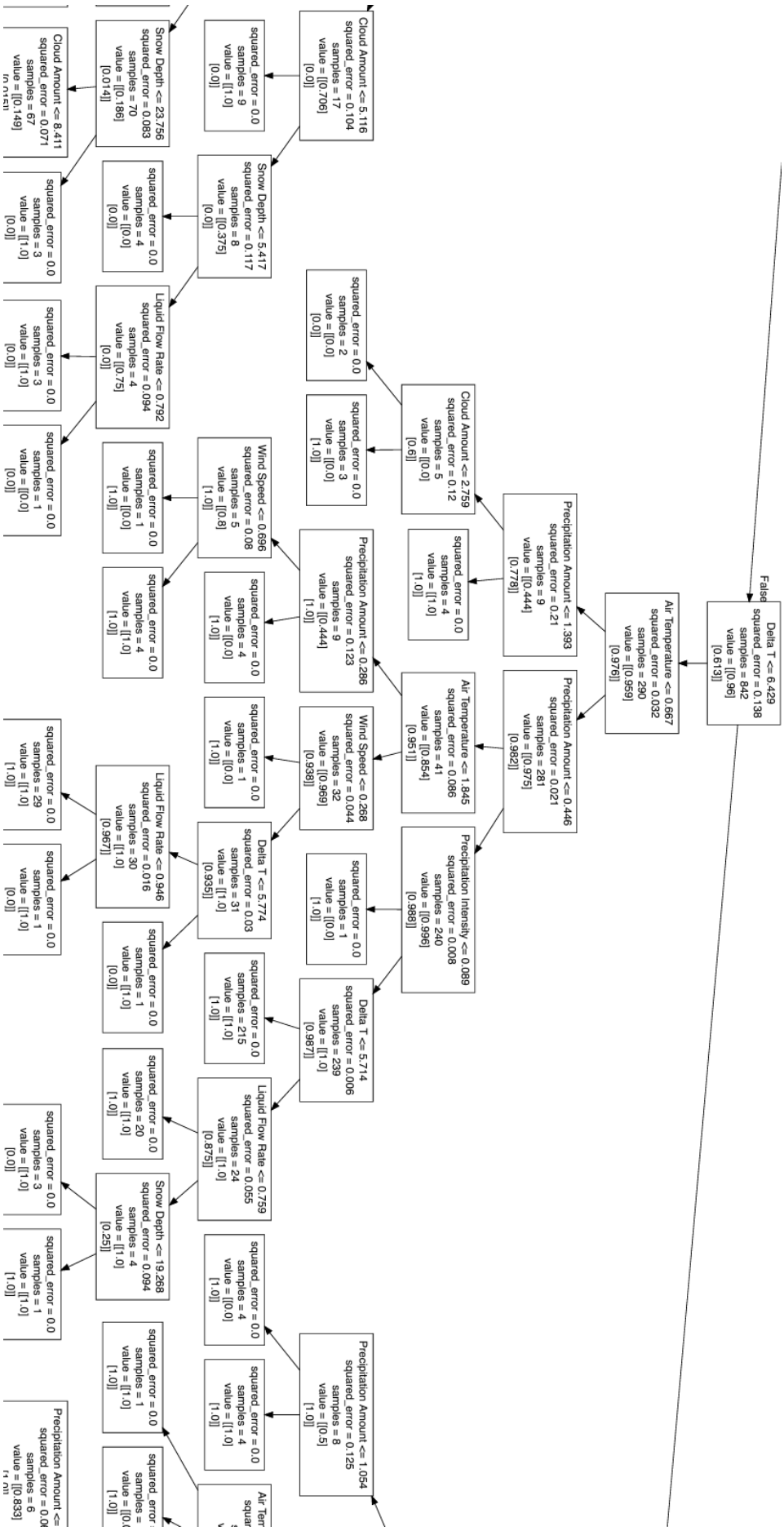


Figure 24: Example of a Decision Tree Flow Chart

3.4.3 Finding True Optimal Results

The processes described in 3.4.1 and 3.4.2 which led to the creation of a guidance document which could be used by a non-specialist human operator to determine optimal control inputs, ΔT and Liquid Flow Rate, allowed for the football pitch to be heated to playable conditions with minimal energy expenditure. The optimal controls determined through this process were limited and approximated for large ranges of weather conditions and therefore the guidance document produced for the operator was very general.

To progress the results produced by the decision tree a complicated machine learning process was undertaken, which used the results of the DoE as inputs to maintain consistency within the study.

The complicated machine learning approach did not require knowledge of specific weather and control scenarios and if the final snow depth of the football pitch and energy consumption were optimal. Instead, it sought to predict what control parameters should be used in an optimal scenario for every given weather parameter, for a minimum of a week in advance.

This was achieved by selecting every case generated by LHS for which it was possible to find an acceptable final snow depth with an acceptable energy consumption, cases which returned values equal to 1 for both final snow depth and energy consumption from the decision tree. From the LHS case of 3000 iterations, 1211 optimal iterations were used within the complicated machine learning training data.

Following a similar approach to that of the decision tree which used the $'sklearn.model\_selection.train\_test\_split'$ tool for python with the weather data and control variables as (x), and the 1 and 0 ranking values as (y). Instead, the weather data was set as the input (x) and the control parameters, as the target (y). Using the regression model, described in 3.4.2 control parameter predictions were made for generated weather scenarios, based on the optimal situations the solver was trained with. Figure 25 contains the python code used to train and determine optimal control parameters.

```
#2) Complicated ML section

dataframe = pd.read_csv('Complicated ML section input.csv')  # , index_col=0)
weatherdata3 = dataframe.to_numpy()

Weather_data3 = weatherdata3[:, 1:7]

DeltaTLiquidFlow = weatherdata3[:, 7:9]
X_train, X_test, y_train, y_test = train_test_split(
    Weather_data3, DeltaTLiquidFlow, test_size=0.2, random_state=1)

# 3)
# Initializing the Decision Tree Regression Model
Rg3 = DecisionTreeRegressor(random_state=0, max_leaf_nodes=50)

# Fitting the Decision Tree Regression model to the data
Rg3.fit(X_train, y_train)

# Predicting the target values of the test set
y_pred = Rg3.predict(X_test)
x_test_df = pd.DataFrame(X_test)
x_test_df.to_csv('Complicated ML X_Test Values.csv')
y_pred_df = pd.DataFrame(y_pred)
y_pred_df.to_csv('Complicated ML Control Predictions.csv')
print(y_pred)
export_graphviz(Rg3, out_file='ComplicatedResults_3000.dot',
                feature_names=['Cloud Amount', 'Precipitation Amount', 'Precipitation Intensity', 'Snow Depth', 'Air Temperature'
```

Figure 25: Python code excerpt for the True Optimal Control Decision Tree

4.0 Results and Discussion

4.1 Moisture Transport

4.1.1 Choosing a Suitable Depth of Soil

The results for the moisture transportation study introduced in 2.6 are presented and discussed in the following section.

Below, Figure 26 and Figure 27 present the results generated during the initial analysis of ideal soil depth. The graphs produced represent the Area-Weighted Average of Water against Flow time, in seconds for a depth of 2m and 7m respectively.

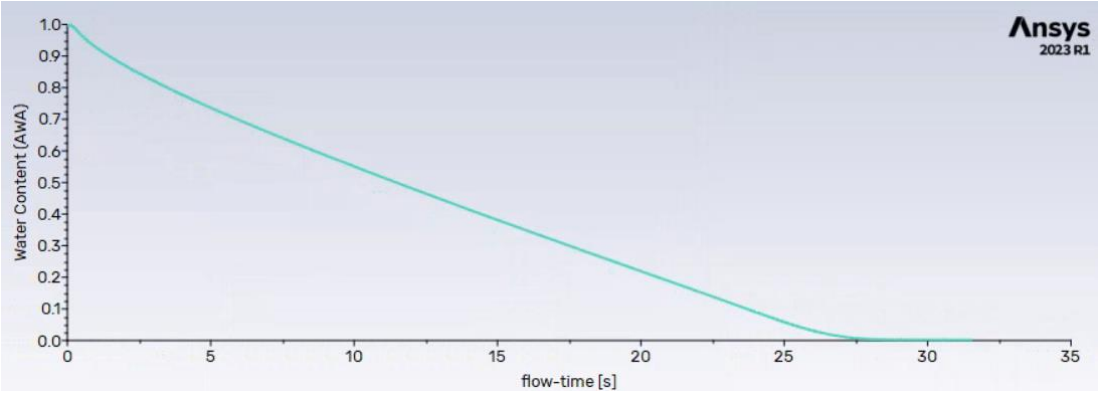


Figure 26: Outer Soil Water Fraction Report for 2m Depth

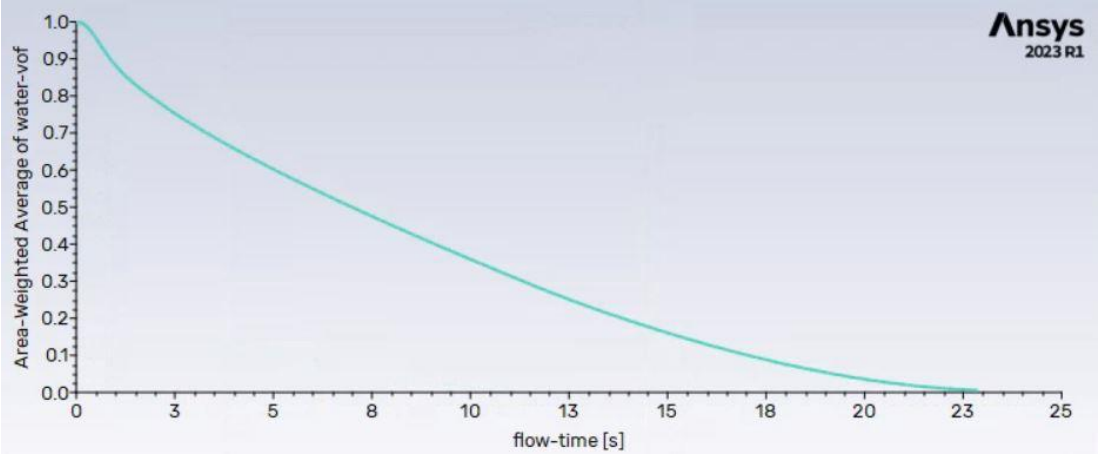


Figure 27: Outer Soil Water Fraction Report for 7m Depth

It can be observed from the results produced that water took longer to drain from the outer soil in the 2m case, despite having a smaller depth, suggesting that the pressure outlet for 2m was having an impact on accuracy. Therefore, the 7m soil depth was chosen going forward.

4.1.2 Iteration 2 Results

The results of one example case of the second iteration of the moisture study are shown in Figure 28, Figure 29 and Figure 30. This case had lower pressure outlets of 0Pa with a velocity inlet flow of 2.5mm/h.

Figure 28 shows the area-weighted average water content of the inner soil zone against flow time, in seconds. Figure 29 and Figure 30 show the area-

weighted average of thermal conductivity, (W/mK) and temperature, (K) against flow time, respectively.

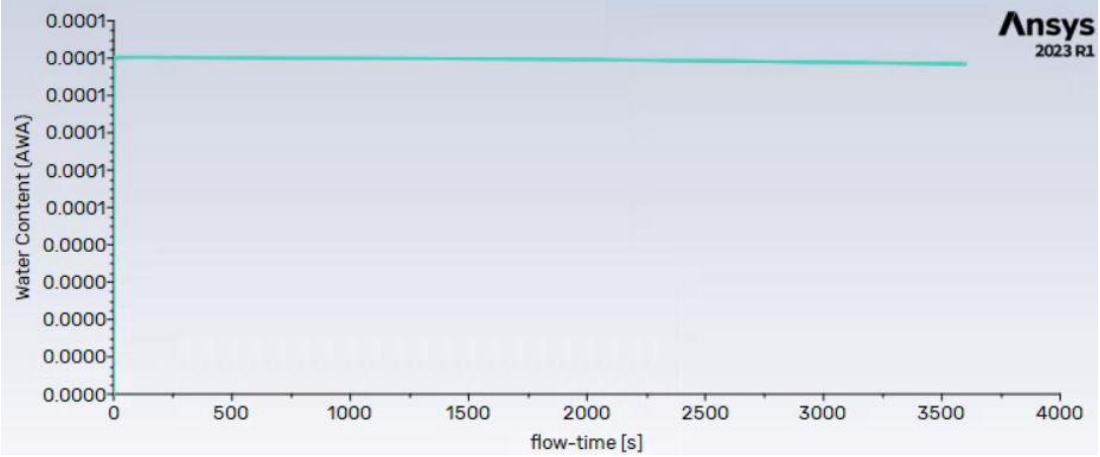


Figure 28: Area Weighted Average Water Content of the Inner Soil Zone at 0Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

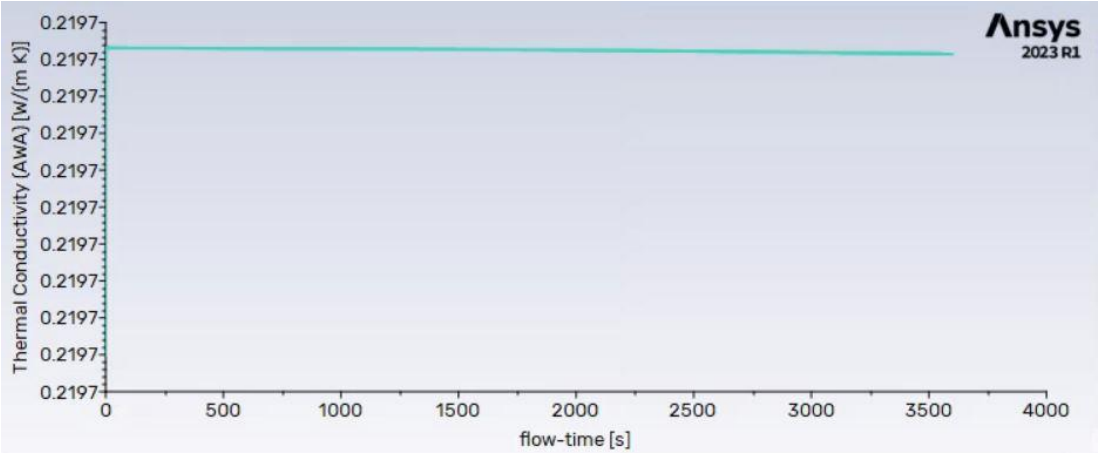


Figure 29: Area Weighted Average Thermal Conductivity of the Inner Soil Zone at 0Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

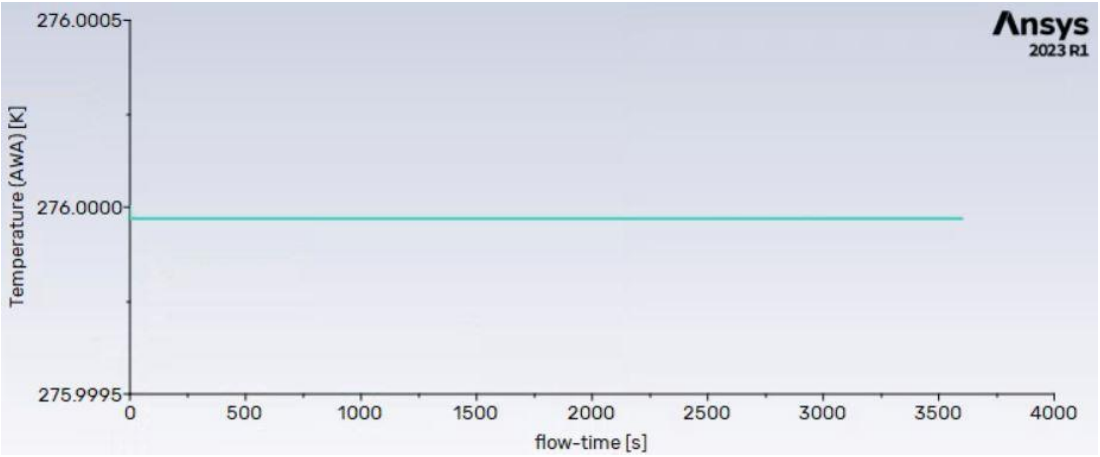


Figure 30: Area Weighted Average Temperature of the Inner Soil Zone at 0Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

Within these 3 reports very little was observed. This is due to the minute quantity of water entering through the velocity inlet. The second iteration was

repeated for the remaining 15 cases, with precipitation amount spanning 0.5mm/h to 25mm/h, and gauge pressures set as 0Pa, 50Pa, 100Pa. 4 of these cases also used a solid wall in place of the boundary, as discussed in Section 3.2.3. Another example case with lower pressure outlets at a gauge pressure of 50Pa and 2.5mm/h precipitation amount is shown below. Figure 31 shows the area-weighted average water content of the inner soil zone against flow time, in seconds. Figure 32 and Figure 33 show the area-weighted average of thermal conductivity, (W/mK) and temperature, (K) against flow time, respectively.

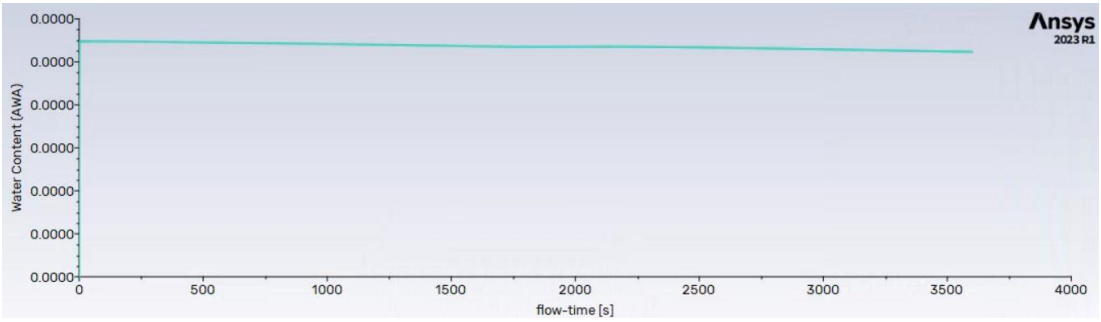


Figure 31: Area Weighted Average Water Content of the Inner Soil Zone at 50Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

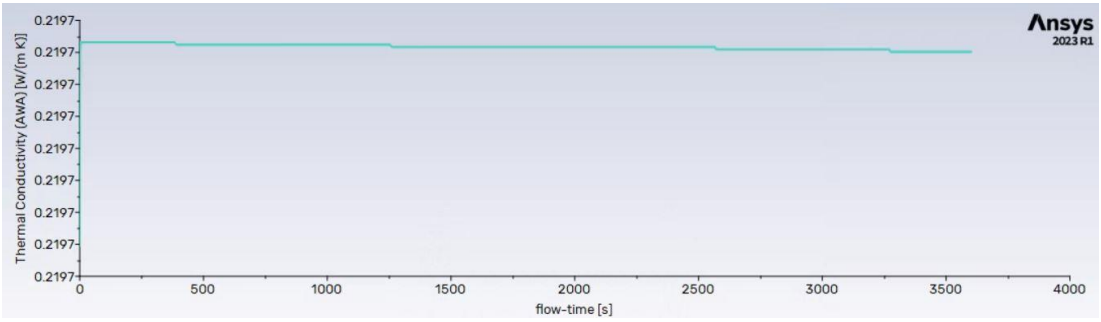


Figure 32: Area Weighted Average Thermal Conductivity of the Inner Soil Zone at 50Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

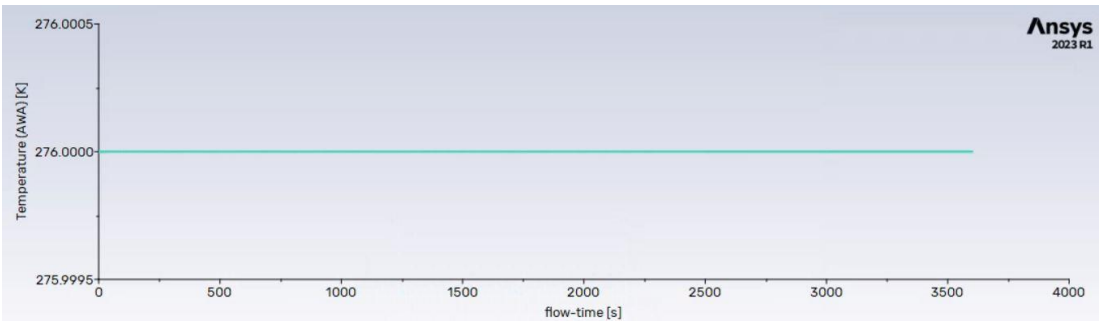


Figure 33: Area Weighted Average Temperature of the Inner Soil Zone at 50Pa Gauge Pressure and a 2.5mm/h Velocity Inlet (2nd Iteration)

It can be concluded from the above results that an increase in gauge pressure at the lower pressure outlet boundaries to 50Pa, did not produce a significant change in the movement of water. This case and the 100Pa gauge pressure case were omitted when processing the results of the second moisture model.

While the use of a velocity inlet was a more suitable situation than a complex rain simulation, it still resulted in artificial forcing of the water, as discussed in Section 3.2.3. Continuing with the second iteration, the outer domain was patched to be fully saturated with water, and the inner soil volume over time was recorded, and plotted in Figure 34 and Figure 35, which show the 0Pa and the Solid Wall case.

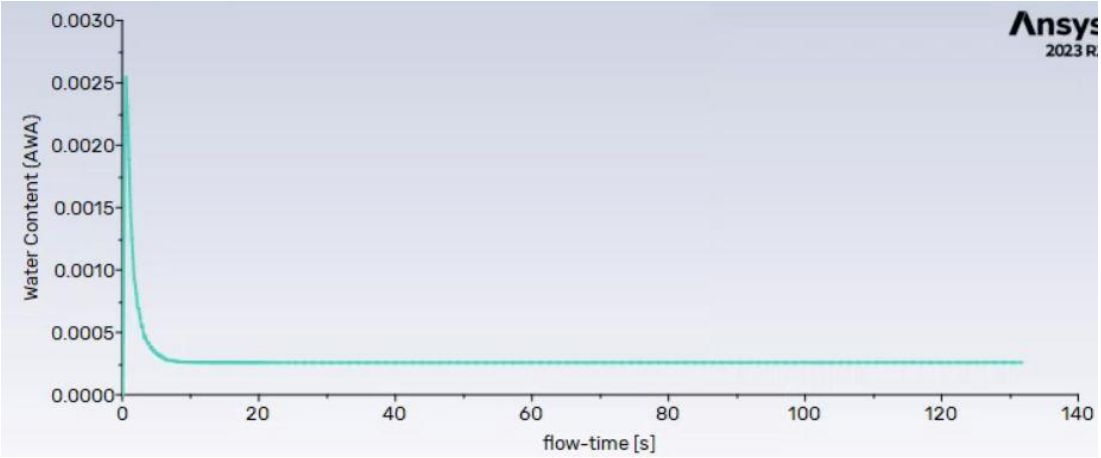


Figure 34: Inner Soil Volume Report for 0Pa Case

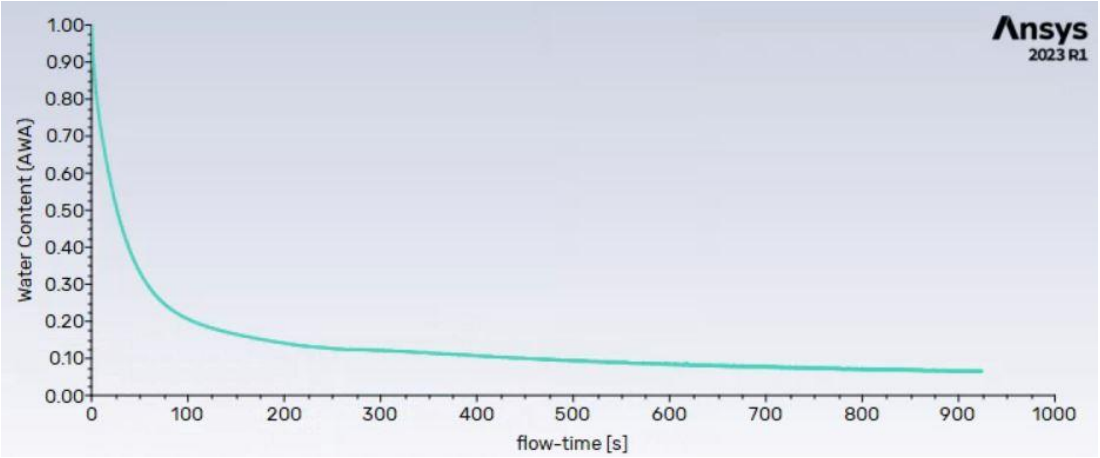


Figure 35: Inner Soil Volume Report for Solid Wall Case

Average water content over 1 hour was calculated from the reports shown in Figure 34 and Figure 35. For the 0Pa case, the peak water content divided by the time to drain was multiplied with the remaining water content divided by remaining time in 1 hour. For the Solid Wall case, the average was found using the water content at 100 second intervals over an hour. Based on these results, and the difference between pure air and pure water properties, the new soil properties were calculated. These are shown in Table 4:

Table 4: Water Volume Fraction by Case, with Corresponding Property Changes, over 1 hour

Case	1	2
Case Type	0Pa	Solid Wall
Maximum inner water content (fraction)	0.0025	0.3327
Time to drain (inner) $[s]$	5	~ 2100
Average water vol 1hr (fraction)	$3.47222e^{-6}$	0.054
Average Thermal Conductivity (mix) $[W/mK]$	0.219680008	0.219804373
Average Density (mixture) $[kg/m^3]$	1.2250346	1.7633665
Average Cp (mixture) $[J/kgK]$	1006.43011	1008.144808
Average Temperature (mixture) $[K]$	275.9999999	275.99892

To convert these new fluid mixture values into values that could be added to the python code, further processing took place.

The water content entering the inner soil was first scaled according to more realistic rainfall amounts. This was achieved by dividing by a fraction of rainfall depth after 1 hour and the depth of the lower soil boundary. These results were recorded in Table 5 and Table 6 for the 0Pa and Solid Wall case, respectively. This allowed the actual amount of water relative to the saturated condition to be reflected properly.

Table 5: Scaled Water Volume Fraction and Property Changes for 0Pa

0Pa Scaled	0.5mm/h	2.5mm/h	5mm/h	25mm/h
Water content	(0.0005/7) x water content (and effect) = $2.48 \cdot 10^{-11}$	(0.0025/7) x water content (and effect) = $1.24 \cdot 10^{-10}$	(0.005/7) x water content (and effect) = $2.48 \cdot 10^{-10}$	(0.025/7) x water content (and effect) = $1.24 \cdot 10^{-9}$
Thermal Conductivity Change (mix)	Negligible	Negligible	Negligible	Negligible
Density Change (mix)	1.225000025	1.225000124	1.225000247	1.225001236
Cp Change (mix)	Negligible	Negligible	1006.430001	1006.430004
Temperature Change (mix)	Negligible	Negligible	Negligible	Negligible

Table 6: Scaled Water Volume Fraction and Property Changes for Solid Wall

Wall Scaled	0.5mm/h	2.5mm/h	5mm/h	25mm/h
Water content	(0.0005/7) x water content (and effect) = $3.857 \cdot 10^{-6}$	(0.0025/7) x water content (and effect) = $1.928 \cdot 10^{-5}$	(0.005/7) x water content (and effect) = $3.857 \cdot 10^{-5}$	(0.025/7) x water content (and effect) = $1.928 \cdot 10^{-4}$
Thermal Conductivity Change (mix)	0.219680888	0.219684441	0.219688884	0.219724406
Density Change (mix)	1.228845475	1.244221678	1.26345475	1.41721678
Cp Change (mix)	1006.442249	1006.491225	1006.552486	1007.04225
Temperature Change (mix)	275.9999923	275.9999614	275.9999229	275.9996144

For the implementation of these results, each property to be updated was defined as a global variable within the 'CalculateSnowAndRainMassFluxes' function, and then reset to default values at each timestep, to simplify the constant update process. The porosity of 0.4 was necessary to consider, as the ratio of soil properties to fluid mixture properties needed to be accounted for. In the python code, it was assumed that each soil layer was composed of 60% soil and 40% air. This meant that estimated thermal properties were an average between the soil and the fluid mixture. Before creating conditional statements in the python code, the expected change to average thermal properties was found by inserting the updated fluid mixture value of each property into the calculation.

Initial implementation steps are shown in Figure 36:

```
##################################################################################################
  # {Section Below Updates Thermal Properties of the Soil based on Average Water Content Over One Hour}
  #Set updating variables as global#
    global TurfSectionThermalConductivity
    global BlastFurnaceSlagThermalConductivity
    global GravelThermalConductivity
    global BlastFurnaceSlagDensity
    global GravelSectionDensity
    global TurfSectionDensity
    global BlastFurnaceSlagCp
    global GravelSectionCp
    global TurfSectionCp
    global T_1_curr
    global T_2_curr
    global T_3_curr

    #Redefine variables at the start of each timestep to match water outflow#
    TurfSectionThermalConductivity = 0.4548
    BlastFurnaceSlagThermalConductivity = 0.35
    GravelThermalConductivity = 0.36
    BlastFurnaceSlagDensity = 1500
    GravelSectionDensity = 1690
    TurfSectionDensity = 1034.3
    BlastFurnaceSlagCp =  840
    GravelSectionCp = 840
    TurfSectionCp = 1289

    #Create variables for the density, Cp and Thermal Conductivity of the soil portion alone)
    TurfSectionMaxSoilDensity = (TurfSectionDensity - 0.4*1.225)/0.6 # Porosity of 0.4 in simulations
    GravelSectionMaxSoilDensity = (GravelSectionDensity - 0.4*1.225)/0.6
    BlastSectionMaxSoilDensity = (BlastFurnaceSlagDensity - 0.4*1.225)/0.6
    TurfSectionMaxSoilCp = (TurfSectionCp - 0.4*1006.43)/0.6
    GravelSectionMaxSoilCp = (GravelSectionCp - 0.4*1006.43)/0.6
    BlastSectionMaxSoilCp = (BlastFurnaceSlagCp - 0.4*1006.43)/0.6
    TurfSectionMaxSoilThermalConductivity = (TurfSectionThermalConductivity - 0.4*0.21968)/0.6
    GravelSectionMaxSoilThermalConductivity = (GravelThermalConductivity - 0.4*0.21968)/0.6
    BlastSectionMaxSoilThermalConductivity = (BlastFurnaceSlagThermalConductivity - 0.4*0.21968)/0.6
```

Figure 36: Python code excerpt of Moisture Transport Thermal Property Updates (Part 1)

'If' statements were then added, to automatically alter thermal properties across soil domains at each timestep, based on moisture content. The initial properties were acquired and validated from a variety of sources researched as part of the literature review process [63-68].

```
########################################
#0Pa Case

if FlowRestriction == 0:
  #WaterVolFractionPitch = 0.00000347222
      if Pr >= 0.0005 and TurfSectionDensity < 1434.4 and T_3_curr > 0:
      # Calculate moisture transport changes to thermal properties
      # Remaining properties including ground temp have negligible change
          TurfSectionDensity = 0.6*TurfSectionMaxSoilDensity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.00000347222)*(998.2-1.225)+1.225)
      if Pr >= 0.0005 and GravelSectionDensity < 2090.2 and T_3_curr > 0:
          GravelSectionDensity = 0.6*GravelSectionMaxSoilDensity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.00000347222)*(998.2-1.225)+1.225)
      if Pr >= 0.0005 and BlastFurnaceSlagDensity < 1900 and T_3_curr > 0:
          BlastFurnaceSlagDensity = 0.6*BlastSectionMaxSoilDensity + 0.4*(((0.0005/Outlet_RockDepthChosen)*0.00000347222)*(998.2-1.225)+1.225)


      if Pr >= 0.005 and BlastFurnaceSlagCp < 2110.22796 and T_3_curr > 0:
          BlastFurnaceSlagCp = 0.6*BlastSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.00000347222)*(4182-1006.43)+1006.43)
      if Pr >= 0.005 and GravelSectionCp < 2110.22796 and T_3_curr > 0:
          GravelSectionCp = 0.6*GravelSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.00000347222)*(4182-1006.43)+1006.43)
      if Pr >= 0.005 and TurfSectionCp < 2559.42798 and T_3_curr > 0:
          TurfSectionCp = 0.6*TurfSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.00000347222)*(4182-1006.43)+1006.43)
```

Figure 37: Python code excerpt of Moisture Transport Thermal Property Updates (Part 2)

```
if FlowRestriction == 1:
      #  WaterVolFractionPitch = 0.1
          if Pr >= 0.0005 and Pr < 0.0025 and T_3_curr > 0:
                T_3_curr = 0.6*T_3_curr + 0.4*(T_3_curr - 0.0000077)
          if Pr >= 0.0005 and Pr < 0.0025 and T_2_curr > 0:
                T_2_curr = 0.6*T_2_curr + 0.4*(T_2_curr - 0.0000077)
          if Pr >= 0.0005 and Pr < 0.0025 and T_1_curr > 0:
                T_1_curr = 0.6*T_1_curr + 0.4*(T_1_curr - 0.0000077)
          if Pr >= 0.0025 and Pr < 0.005 and T_3_curr > 0:
                T_3_curr = 0.6*T_3_curr + 0.4*(T_3_curr - 0.0000386)
          if Pr >= 0.0025 and Pr < 0.005 and T_2_curr > 0:
                T_2_curr = 0.6*T_2_curr + 0.4*(T_2_curr - 0.0000386)
          if Pr >= 0.0025 and Pr < 0.005 and T_1_curr > 0:
                T_1_curr = 0.6*T_1_curr + 0.4*(T_1_curr - 0.0000386)
          if Pr >= 0.005 and Pr < 0.025 and T_3_curr > 0:
                T_3_curr = 0.6*T_3_curr + 0.4*(T_3_curr - 0.0000771)
          if Pr >= 0.005 and Pr < 0.025 and T_2_curr > 0:
                T_2_curr = 0.6*T_2_curr + 0.4*(T_2_curr - 0.0000771)
          if Pr >= 0.005 and Pr < 0.025 and T_1_curr > 0:
                T_1_curr = 0.6*T_1_curr + 0.4*(T_1_curr - 0.0000771)
          if Pr >= 0.025 and T_3_curr > 0:
                T_3_curr = 0.6*T_3_curr + 0.4*(T_3_curr - 0.0003856)
          if Pr >= 0.025 and T_2_curr > 0:
                T_2_curr = 0.6*T_2_curr + 0.4*(T_2_curr - 0.0003856)
          if Pr >= 0.025 and T_1_curr > 0:
                T_1_curr = 0.6*T_1_curr + 0.4*(T_1_curr - 0.0003856)
```

Figure 38: Python code excerpt for Moisture Transport Thermal Property Updates (Part 3)

```
if Pr >= 0.0005 and TurfSectionThermalConductivity < 0.54693 and T_3_curr > 0:
        TurfSectionThermalConductivity = 0.6*TurfSectionMaxSoilThermalConductivity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(0.45-0.21968)+0.21968)
if Pr >= 0.0005 and GravelThermalConductivity < 0.4521 and T_3_curr > 0:
        GravelThermalConductivity = 0.6*GravelSectionMaxSoilThermalConductivity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(0.45-0.21968)+0.21968)
if Pr >= 0.0005 and BlastFurnaceSlagThermalConductivity < 0.442128 and T_3_curr > 0:
        BlastFurnaceSlagThermalConductivity = 0.6*BlastSectionMaxSoilThermalConductivity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(0.45-0.21968)+0.21968)
if Pr >= 0.0005 and TurfSectionDensity < 1434.4 and T_3_curr > 0:
    TurfSectionDensity = 0.6*TurfSectionMaxSoilDensity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(998.2-1.225)+1.225)
if Pr >= 0.0005 and GravelSectionDensity < 2090.2 and T_3_curr > 0:
    GravelSectionDensity = 0.6*GravelSectionMaxSoilDensity + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(998.2-1.225)+1.225)
if Pr >= 0.0005 and BlastFurnaceSlagDensity < 1900 and T_3_curr > 0:
    BlastFurnaceSlagDensity = 0.6*BlastSectionMaxSoilDensity + 0.4*(((0.0005/Outlet_RockDepthChosen)*0.054)*(998.2-1.225)+1.225)
if Pr >= 0.0005 and BlastFurnaceSlagCp < 2110.22796 and T_3_curr > 0:
    BlastFurnaceSlagCp = 0.6*BlastSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(4182-1006.43)+1006.43)
if Pr >= 0.0005 and GravelSectionCp < 2110.22796 and T_3_curr > 0:
    GravelSectionCp = 0.6*GravelSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(4182-1006.43)+1006.43)
if Pr >= 0.0005 and TurfSectionCp < 2559.42798 and T_3_curr > 0:
    TurfSectionCp = 0.6*TurfSectionMaxSoilCp + 0.4*(((Pr/Outlet_RockDepthChosen)*0.054)*(4182-1006.43)+1006.43)
```

Figure 39: Python code excerpt for Moisture Transport Thermal Property Updates (Part 4)

As mentioned in Section 3.2.3, the number of statements required for the 0Pa case shown in Figure 37 was less than in the Solid Wall case shown in Figure 38 and Figure 39, due to some properties seeing negligible change. The statements also permitted the user to set a depth value for the soil, which would change the scaling applied to each property update. This way, when simulating smaller soil depths, the average water content would be higher and thermal properties would subsequently increase by larger amounts on average.

After the inclusion of conditional statements, a small improvement in model prediction of snow depth was visible. Despite the removal of the velocity inlet and more natural flow of water, this iteration lacked the ability to model continuous realistic precipitation. The third iteration setup, discussed in Section 3.2.4 would be a combination of the 2 previous iterations.

4.1.3 Iteration 3 Results

The results of the third iteration of underground moisture transport are presented below. Figure 40, Figure 41, Figure 42 and Figure 43 show the water content against flow time, in seconds, for the 0Pa, 5000Pa, 10000Pa and Solid Wall case respectively.

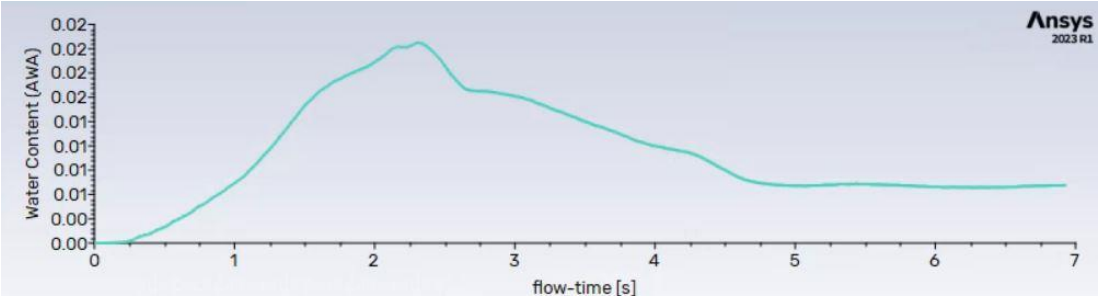


Figure 40: Water Content 0Pa Case

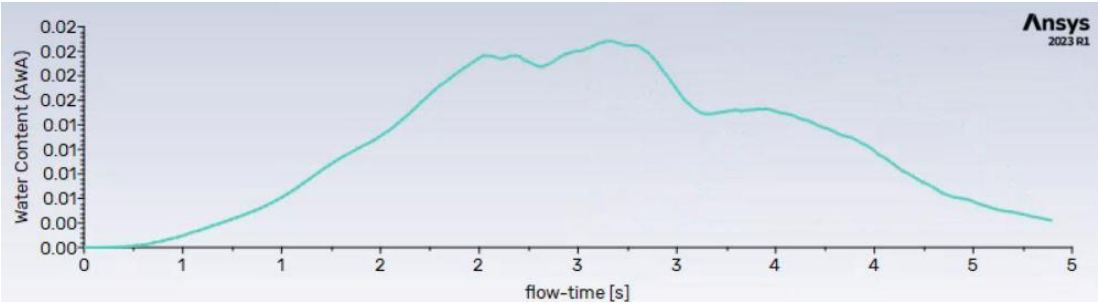


Figure 41: Water Content 5000Pa Case

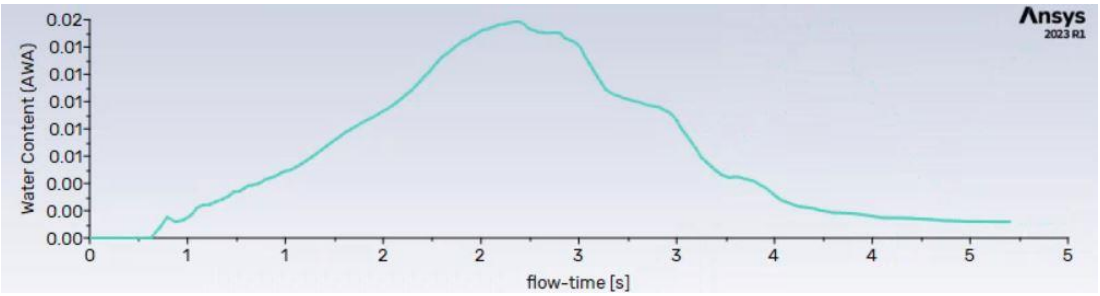


Figure 42: Water Content 10000Pa Case

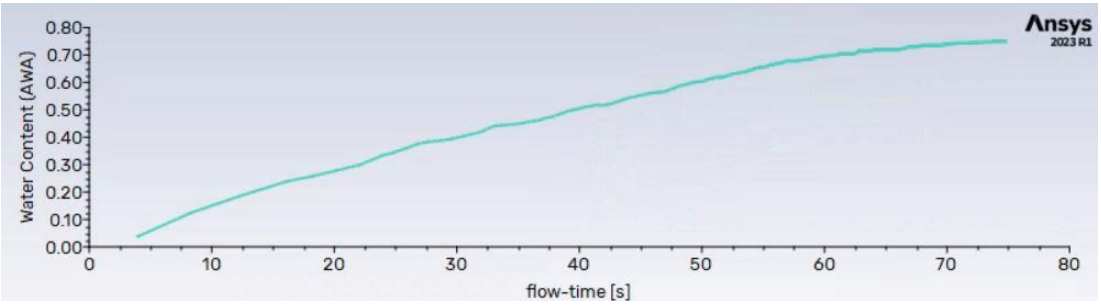


Figure 43: Water Content Solid Wall Case

Each pressure outlet variation showed a similar peak of 2% water content in the inner soil, followed by a drop, which remained stable after 5 seconds. The Solid Wall case displayed an expected gradual increase in water content below the pitch, levelling off at around 80%. The peak represented the initial instability of water motion as it first entered the porous medium. After this initial period, the water settled, following a consistent path that did not penetrate as far horizontally. Again, the similarity between gauge pressure outlet cases, even with high values chosen, suggested that an increased outlet gauge pressure had little effect on water motion. For the final iteration, a change in the varying parameter was therefore deemed important.

This iteration also had room to be developed, as the quantity of water being regularly patched above the soil was still quite large.

4.1.4 Iteration 4 Results

The results of the fourth iteration of moisture simulation are shown below. Figure 44 and Figure 45 show the area-weighted average water content and the area-weighted average temperature, for decreasing porosity, against flow time, in seconds. Figure 46 and Figure 47 show for constant porosity and Figure 48 and Figure 49 show the results for increasing porosity.

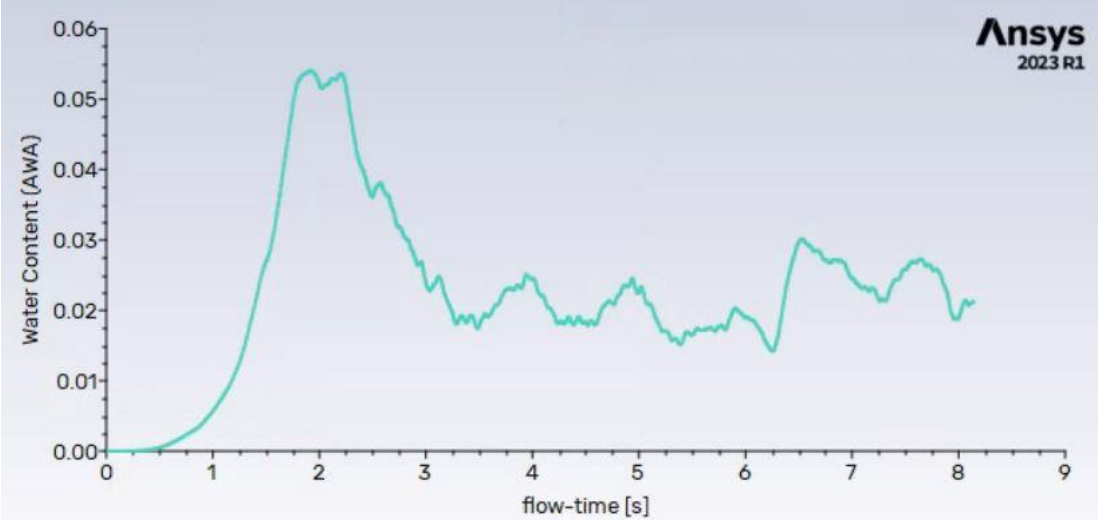


Figure 44: Area Weighted Average Water Content for Decreasing Porosity

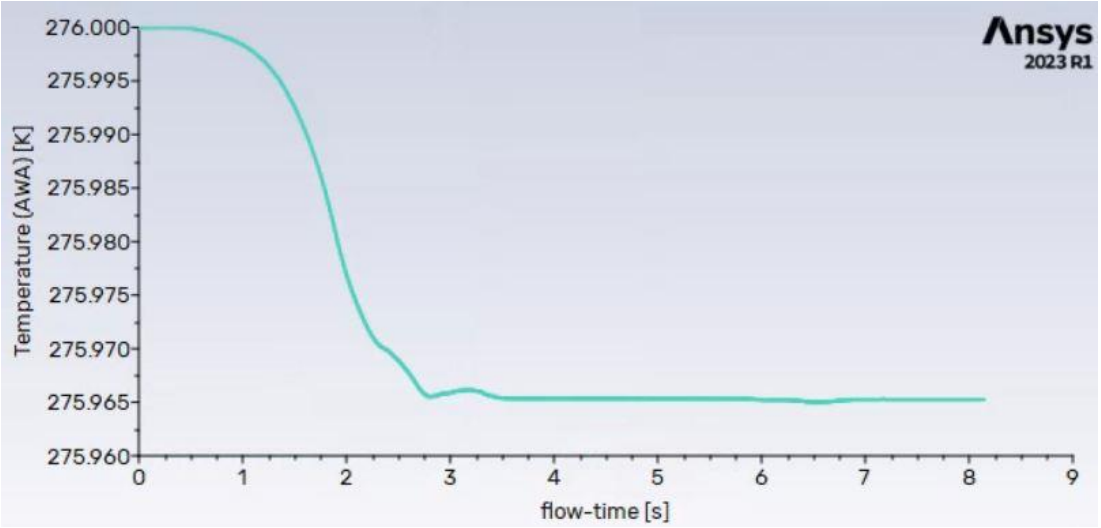


Figure 45: Area Weighted Average Temperature for Decreasing Porosity

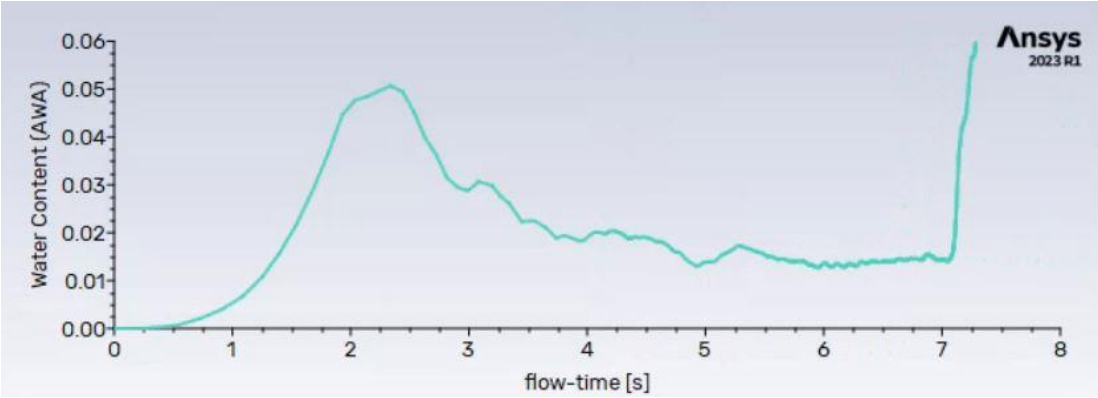


Figure 46: Area Weighted Average Water for Constant Porosity

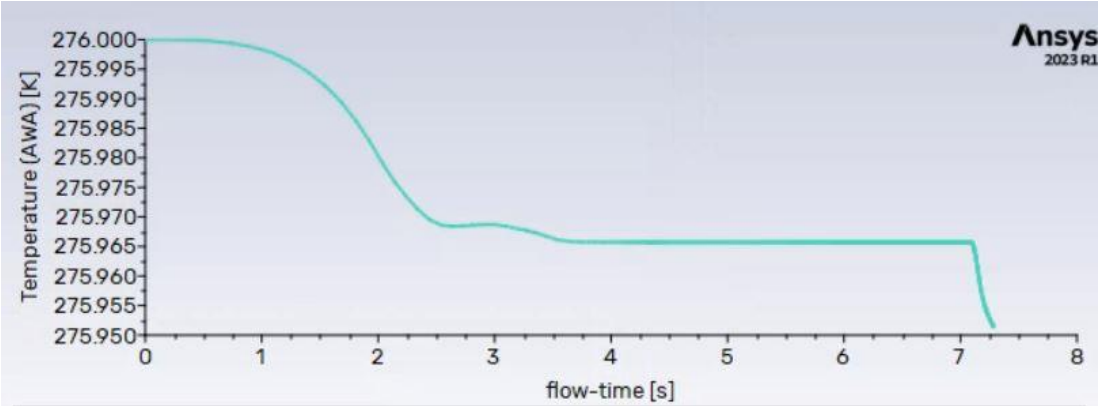


Figure 47: Area Weighted Average Temperature for Constant Porosity

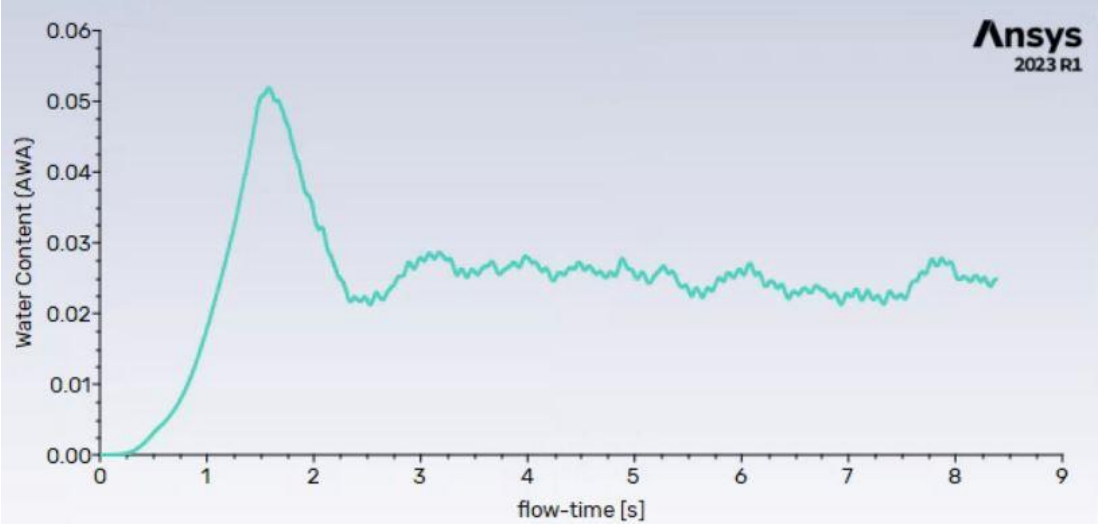


Figure 48: Area Weighted Average Water Content for Increasing Porosity

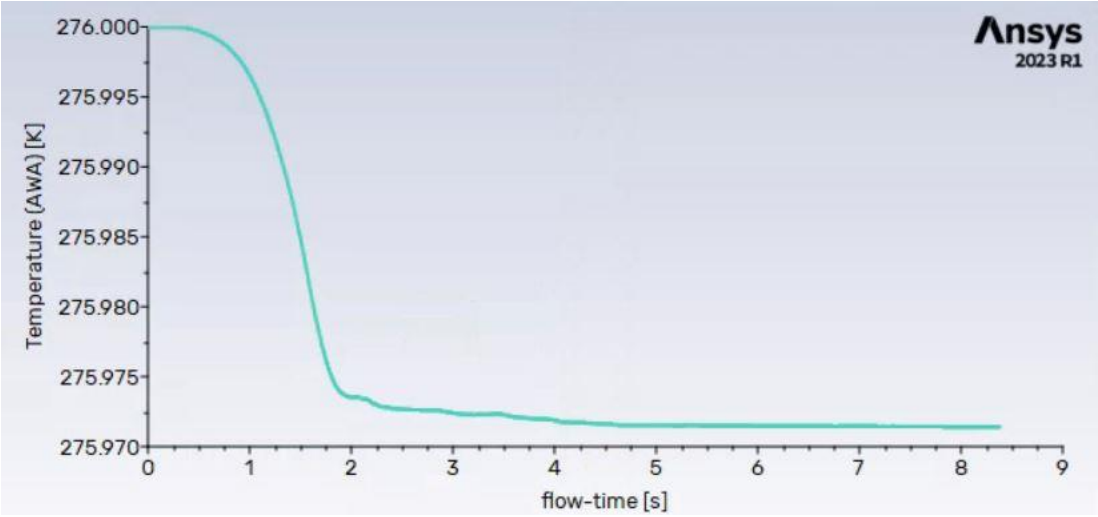


Figure 49: Area Weighted Average Temperature for Increasing Porosity

It could be seen from the water content results, that when a reduced porosity is partially modelled in the soil domain, the peak water content in the inner soil increases. An increase of around 0.2% is visible in the increasing porosity case, and around 0.4% in the decreasing case. The constant porosity case experiences an increase in inner soil water content, and a decrease in temperature at around 7 seconds, because of numerical error in the simulation. This could be rectified by further decreasing the mesh size. The effect of changing the porosity is still minimal, despite using the full ordinary range of soil porosities in this scenario (0.2-0.4).

Focusing on Figure 49, and assuming both that the initial area-weighted average temperature of the soil cross section beneath the pitch was an ambient temperature of 276K, and that water entering this zone horizontally was 275.5K (as rain is often colder than ambient temperatures), the soil average temperature settled at a constant of 0.028K lower than when dry. In Figure 48, water content beneath the pitch settled at 2% to 3% of the total area.

Factoring in the potential rainfall amounts mentioned previously, the soil would receive even less water than a 40cm zone patched every 0.1s. For these reasons, the conclusion was that moisture transport ultimately has a negligible impact on the thermal properties of the soil beneath the pitch and was therefore not included in refinement of the python model. Existing conditional statements added to the python code during the second iteration of the moisture study were removed.

One of the most significant challenges which arose when modelling rainfall, was the time required to complete each simulation. Mesh refinement helped to find a good compromise between accuracy and simulation time, however the model was still quite complicated. Adapting the geometry further to only include important areas of study reduced the computational time of simulations. For the final case, we split the task between group members, with each person running one of the 3 cases. This collaborative effort further sped up the process.

4.2 Time Delay Imbalance

The model described in Section 3.2.6 was used to simulate a variety of cases, modifying different input parameters and assessing their impact on the time delay. Running the simulation as detailed in the original case (Table 3), the following results were obtained:

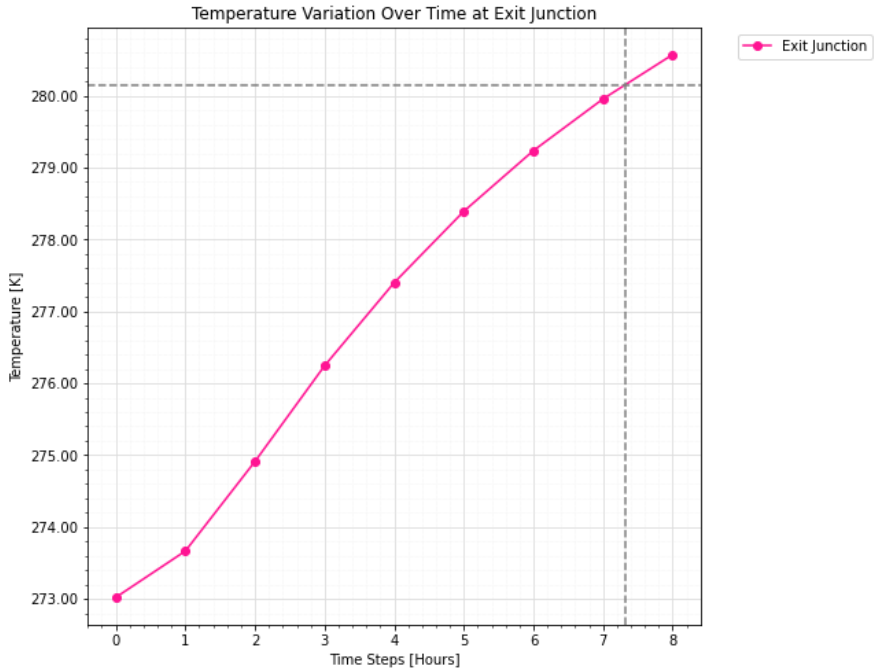


Figure 50: Temperature Variation at Exit Junction

In the original paper published by the COMEA research group [1], a temperature increase, $\Delta T = 7°C$ resulted in 92.9% of snow-free time. This was taken as the increase in temperature required at all junctions to consider the field in playing conditions. Figure 50 shows the temperature variation over time at the junction furthest away from the pump, just before the fluid is returned to the outlet line and eventually to the pump to be reheated. The dashed grey lines show the time taken to reach the required value, which in this case was just over 7 hours.

The presence of ΔT and Liquid Flow Rate as variables in the Mass and Energy Balance Equations meant that adapting the existing code to include the time delay in distribution of temperature could help improve the predicted values of snow depth.

Table 7 further showcases the variation in time-delay for various ambient temperatures:

Table 7: Impact of Ambient Temperature on Time-Delay

Ambient Temperature, T_{amb} $[°C]$	Time Taken, $[Hours]$
0	7.32
−5	11.38
−10	15.4
−15	19.39

Based on these results, there is a significant increase in time duration for lower ambient temperatures. Prior knowledge of the weather conditions based on forecast data can help minimise the impact of this as the system could be activated earlier to accommodate for any delay.

A similar study was conducted on the impact of mass flow rate of the solution and the temperature difference between the inlet and outlet temperatures, the results of which are summarised in Table 8 and Table 9, respectively. The remaining inputs were kept constant, as described in the original case.

Table 8: Impact of Mass Flow Rate on Time-Delay

Mass Flow Rate of Solution, $\dot{m}$ $[ks/s]$	**Time Taken, $[Hours]$**
1	7.32
0.8	9.39
0.6	12.84
0.4	19.75

Table 9: Impact of Temperature Difference Between Inlet and Outlet on Time-Delay

Temperature Difference Induced by Pump, ΔT_f [℃]	**Time Taken, $[Hours]$**
20	4.97
16	6.65
15	7.32
12	10.9
10	17.43

In all 3 cases, there was a significant increase in the time taken for the full network to reach the required temperature as the input parameters were reduced. However, while changes in the ambient temperature cause a steady increase, the liquid flow rate and temperature difference appeared to have a more substantial effect at their respective lower values. In both cases, decreasing the inputs further showed that it would not be possible to reach the threshold temperature, so those values were discarded. A significant contribution to this result was made by the fact that while one parameter was being tested, all other inputs remained the same. In both cases of changing control inputs, the calculation assumed that the ambient temperature remains constant at its original value of 0 ℃ the entire time, something which would not be reflected in the real world. This showed the limitations of this type of analysis. The development of a more complicated model which updates the ambient temperature automatically within the calculation could make up future work on the subject. On the other hand, when simulating for lower ambient conditions, the controls would not be kept constant but increased as the temperature decreased, once again reducing the time it would take for the real system to reach the required temperature.

It is also important to note that some of the variables used in the simulation, such as the pressure loss coefficient and the heat transfer coefficient of the

pipes were estimations rather than exact known values. This introduced some level of uncertainty, which although small, could have had an impact on the final result. A more robust approach would be to use the large scale Ansys model developed by the COMEA research team for the specific geometry of the field to acquire those parameters. However, the requirements of running such a complicated simulation were found to far outweigh the benefits, as similar results could be produced using the model developed as part of this project for a fraction of the computational time and requirements.

Taking the above into account and modifying the original code to incorporate time delay, it was found that there were negligible changes in the snow depth prediction when the system was activated sufficiently early. However, the more accurate representation of the transportation of fluid allowed for more precise calculation of the energy consumption at each time step, as this is directly dependent on the values of ΔT and the liquid flow rate. This could become an important factor in the comparison between the energy expenditure of a fixed heating system and one based on the true optimal control developed as part of this project. Sections 4.8 and 4.9 describe how this issue was tackled and the results of the evaluation.

4.3 Evaporation and Condensation

The effect the addition $\dot{Q}_{Evap/Cond}$ had on the snow melting model, previously discussed in Section 3.2.7 can be observed in Figure 51 to Figure 54 inclusive.

Figure 51 and Figure 52 show the measured snow depth compared to the predicted snow depth for computational models with and without the consideration of evaporation and condensation. Both graphs display similar levels of accuracy, with the case without evaporation or condensation ranking just slightly higher. Accuracy was determined from these graphs by comparing the predicted snow depth (black line) to the measured snow depth (blue line).

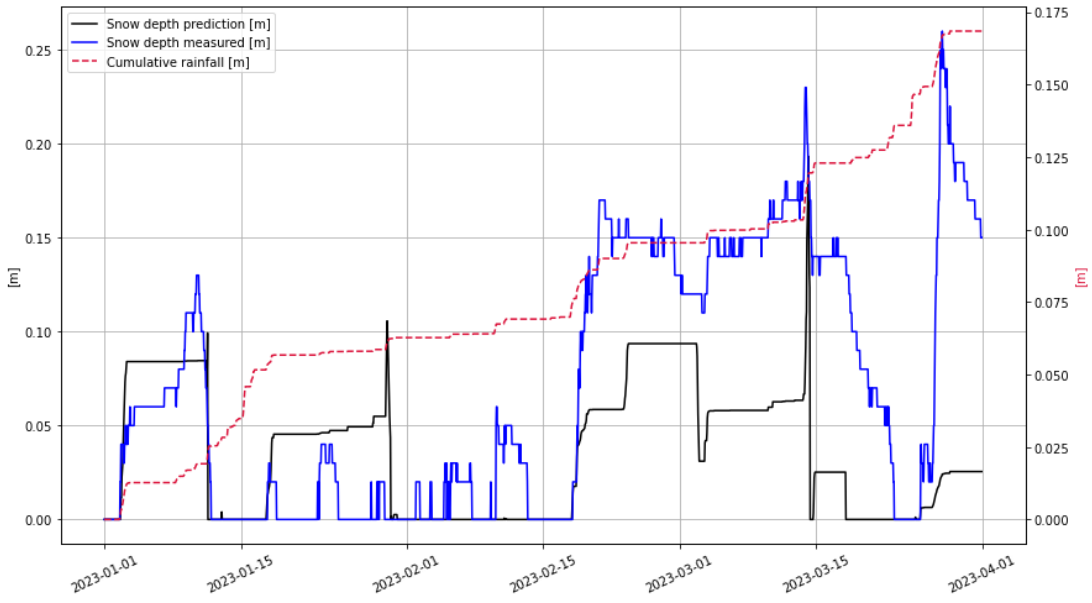


Figure 51: Measured and Predicted Snow Depth [Without $\dot{Q}_{Evap/Cond}$]

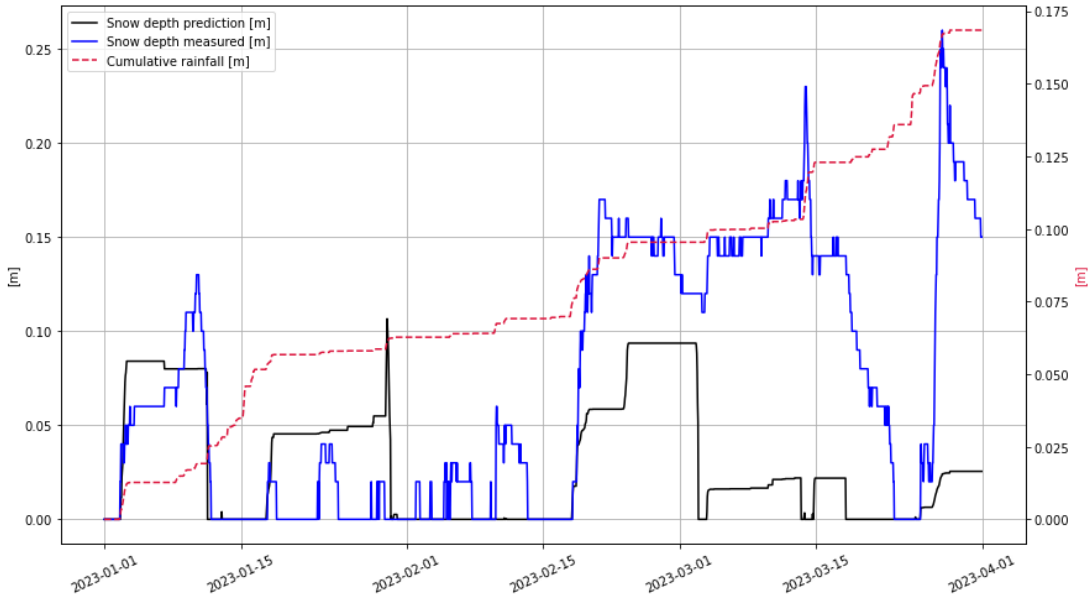


Figure 52: Measured and Predicted Snow Depth [With $\dot{Q}_{Evap/Cond}$]

The addition of $\dot{Q}_{Evap/Cond}$ resulted in very minimal changes but still decreased the overall accuracy. This can be further shown by comparing Figure 53 and Figure 54 which show little to no difference in the measured and predicted temperatures with the inclusion of the effects of evaporation and condensation. The calculated value of $\dot{Q}_{Evap/Cond}$ was negative for each time step which confirmed that it was condensation of the water vapor within the snowpack that was occurring, however small that amount may have been. This reduction in precision could be attributed to errors in observational data [69], leading to exploration of other techniques for calculating evaporation [70, 71], but ultimately no definitive solution was found.

It could therefore be concluded that the amount of surface precipitation which was subject to condensation was minimal due to the ambient air conditions and water content present within the snowpack.

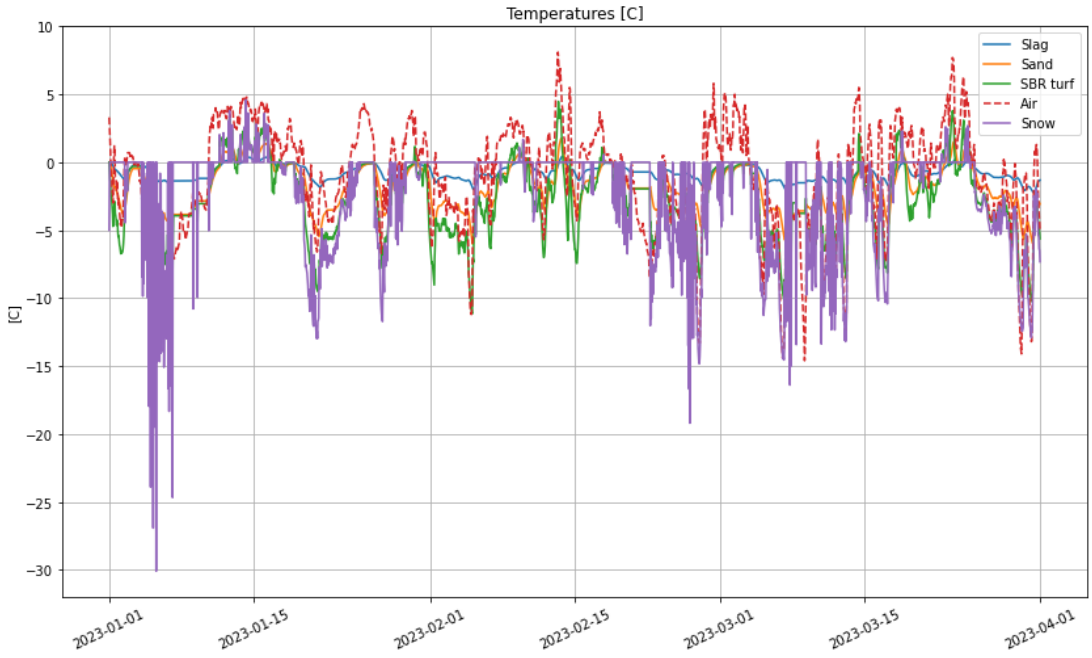


Figure 53: Measured and Predicted Temperature [Without $\dot{Q}_{Evap/Cond}$]

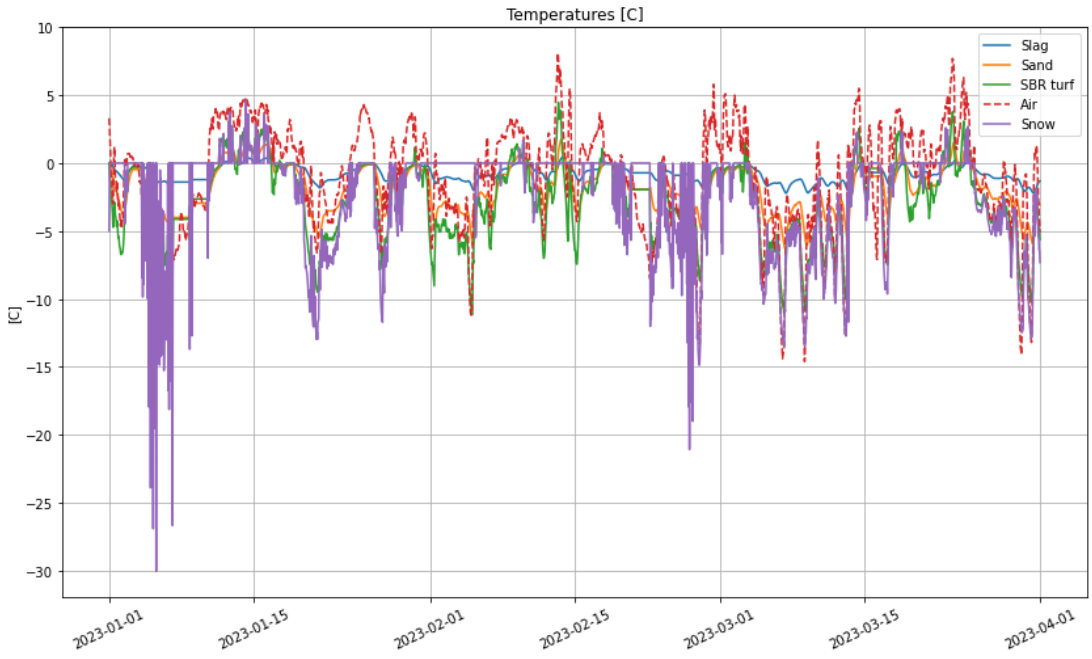


Figure 54: Measured and Predicted Temperature [With $\dot{Q}_{Evap/Cond}$]

Although the effect that $\dot{Q}_{Evap/Cond}$ had on the overall heat and mass balance model was minimal for the current weather data, it will remain present within the Python code and computational model to support any future time period where weather conditions may cause it to have a larger impact.

The small effect the inclusion of $\dot{Q}_{Evap/Cond}$ had within the current computational model suggested that the snowpack was not subject to

extreme cases of evaporating water or condensing water vapor, for the specific timeframe the study observed. The refinement of the computational heat and mass transfer model only looked at weather data from January 1st, 2023, to March 31st, 2023. Evaporation is more noticeable with snow in weather conditions where the air temperature is above frozen, but not too high that the snowpack has melted. By considering longer or different periods where the weather conditions meet the evaporation requirements would show a much greater effect $\dot{Q}_{Evap/Cond}$ has on the overall accuracy.

4.4 Comparison To Existing Code

Additional validation had to be carried out for the modified version of the code so the impact of the changes could be suitably evaluated in comparison to the original code. For a time-step size of 60 minutes, the simulation results were compared for 2 sets of weather data: 1st January 2021 – 31st March 2021 and for the same period in 2023. This allowed for direct assessment of the accuracy between the 2 versions of the script and identification of further areas of improvement. Figure 55 and Figure 56 show the first set of results:

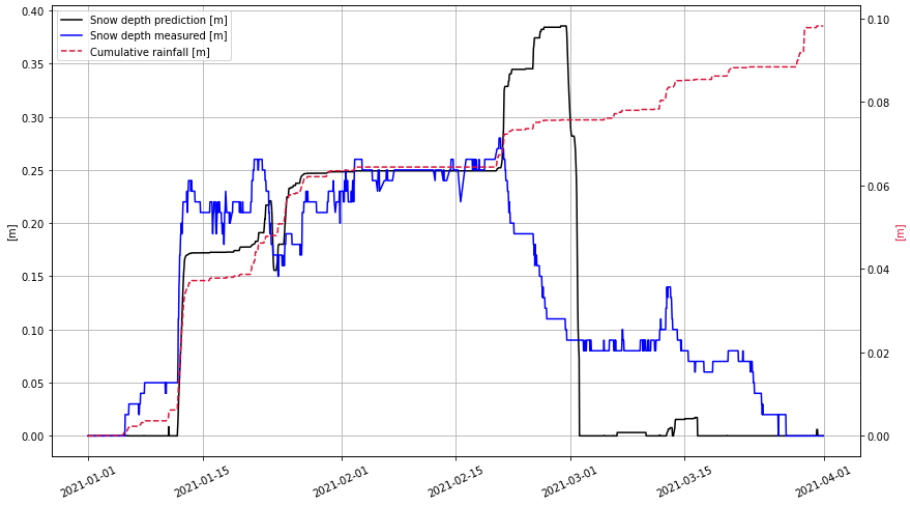


Figure 55: Original Code Predictions for 2021 Data

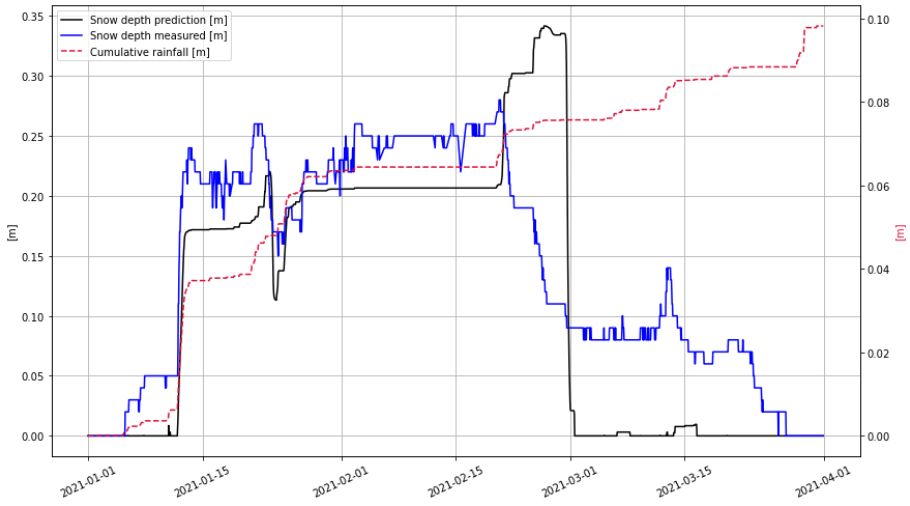


Figure 56: Modified Code Predictions for 2021 Data

From the comparison carried out it was found that the incorporation of the additional features into the snow melting model significantly reduced

overprediction, which took place in the data around the second half of February, by around $5\,cm$. In addition, while no noticeable improvements were made in the prediction for March, the trends in weather in January were captured more accurately, once again reducing the initial overprediction which happened in the original code. It is important to note that such overpredictions could have a notable impact on the energy consumption, as the model would predict much higher heating requirements. Even marginal reductions could mean large savings in energy costs when measured over the full duration of the simulation. Moreover, establishing the limits of operation was one of the objectives of this report, as there would inevitably be cases where the heating system would not be able to melt all of the snow, therefore any time spent running the system in those cases would lead to significant energy expenditure but no improvement in the playing conditions. Inaccurately predicting such cases could lead to longer periods where the field has to be taken out of service as the snow accumulated in reality could have been melted. The same period was also simulated for 2023 to test the stability of the predictions, shown in Figure 57 and Figure 58:

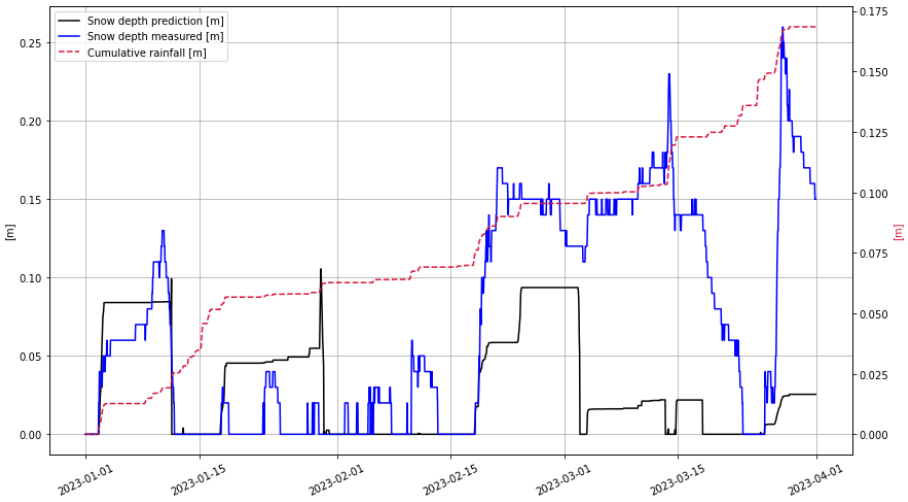


Figure 57: Original Code Predictions for 2023 Data

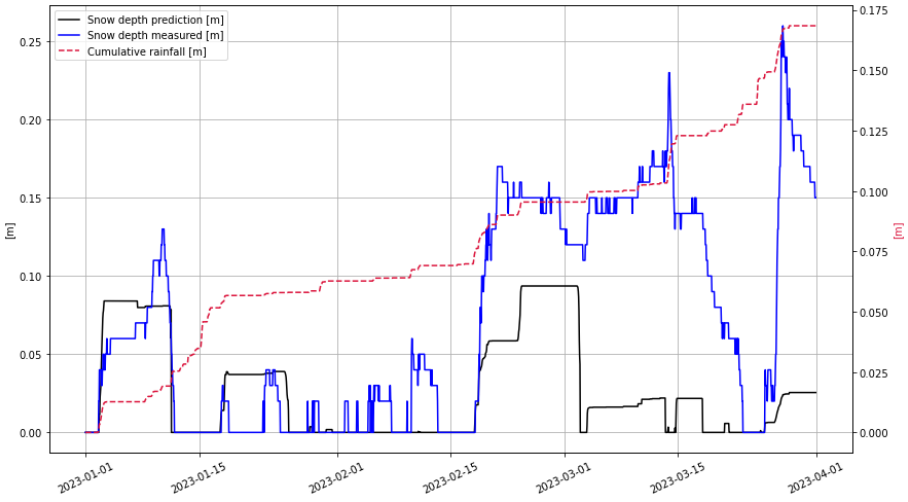


Figure 58: Modified Code Predictions for 2023 Data

The same trends were observed in the new results, with significantly different weather conditions. Analogous to the plot of 2021 data, overpredictions of

snow depth were significantly reduced when using the modified version of the code, as observed in the January period. In this case, both versions of the code failed to predict the significant increase in snowfall in the second half of February and March period. This could have perhaps been attributed to unexpected shifts in weather patterns in recent years due to the effects of climate change. Some of the functions used in the calculation contained variables which were very sensitive to weather phenomena and may have not been accurately reflected in the conditions for the chosen study period. In addition, as discussed in Section 3.3.1, the simulation depended on many obscure parameters which are often difficult or almost impossible to measure in the real world but nevertheless carry significant implications on accuracy. Further improvements could be made through developing methods of more accurately representing those values, replacing the existing ones based on ranges or estimations.

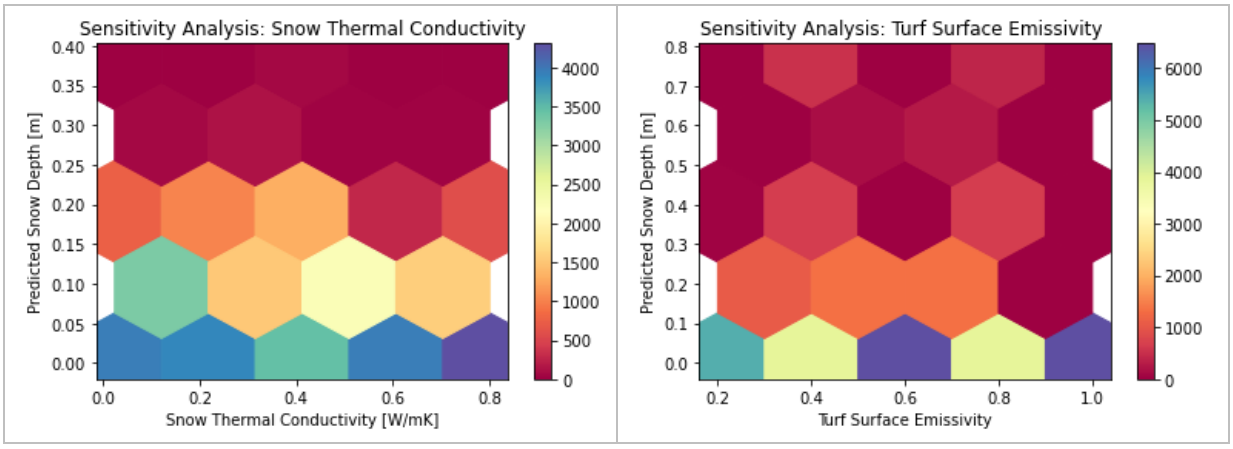
4.5 Sensitivity Analysis

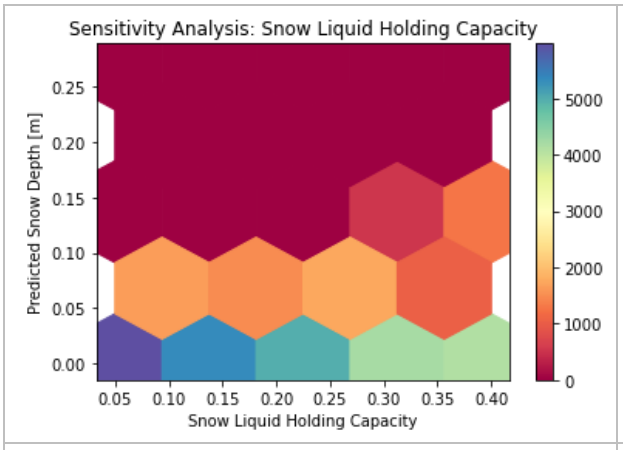
4.5.1 Variable Study

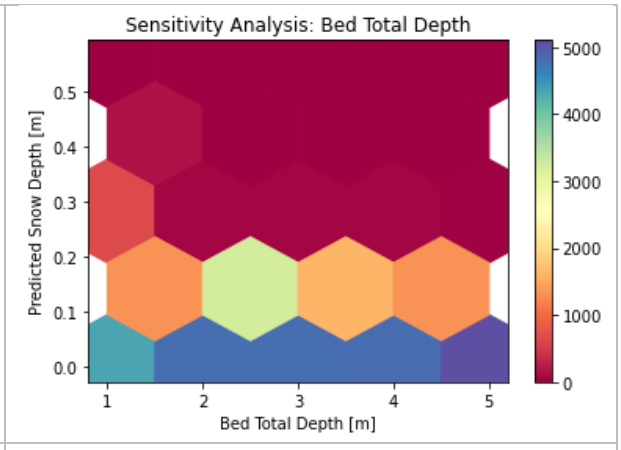
This following section presents and discusses the results of the variable study introduced in 3.3.1. The hex-graphs produced for each variable are presented and ranked by order of sensitivity, in Table 10 and Table 11.

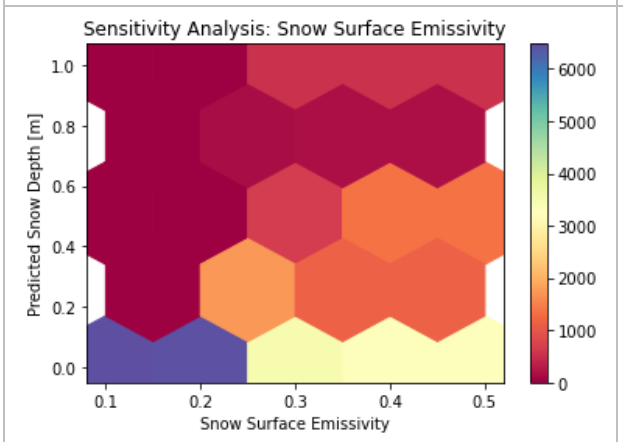
Reading each graph in columns from left to right, the distribution of samples for various snow depth predictions was mapped. For each new value of x, the variation from the same point at the previous x-value determined the sensitivity of the solution to the parameter. For example, Snow Thermal Conductivity was deemed to be more sensitive than Turf Surface Emissivity, due to the varying composition of colours moving across x values. Similarly, the plot of Air Density had more consistency moving left to right and was therefore deemed to have contributed less significantly to the overall results compared to Snow Thermal Conductivity.

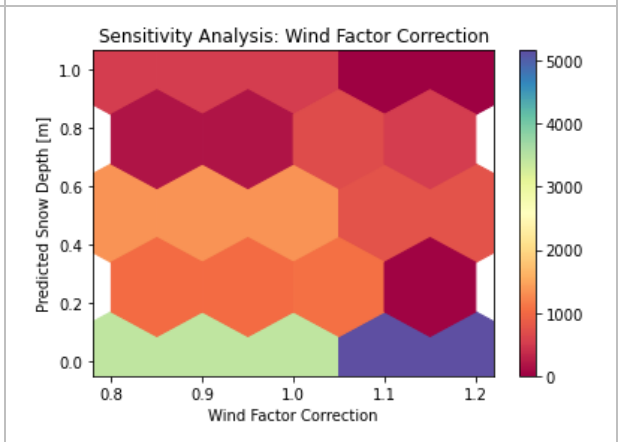
Table 10: Result Plots of Sensitivity Analysis

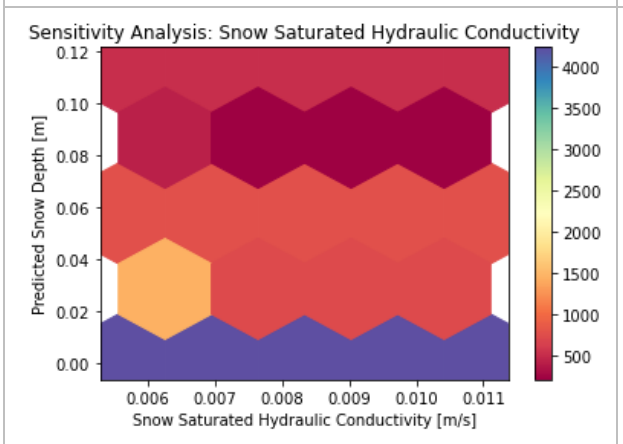


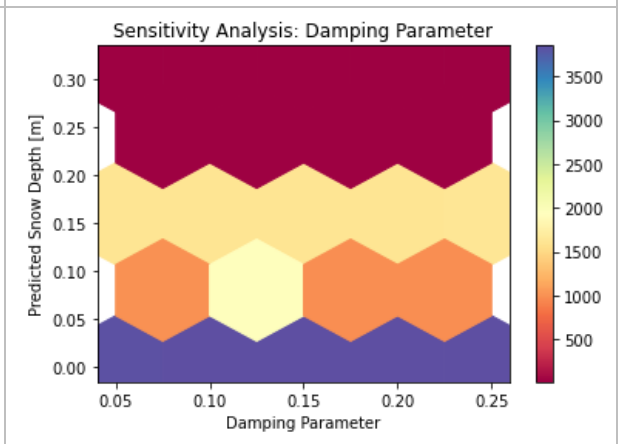

Sensitivity Analysis: Snow Liquid Holding Capacity
Predicted Snow Depth [m]
Snow Liquid Holding Capacity

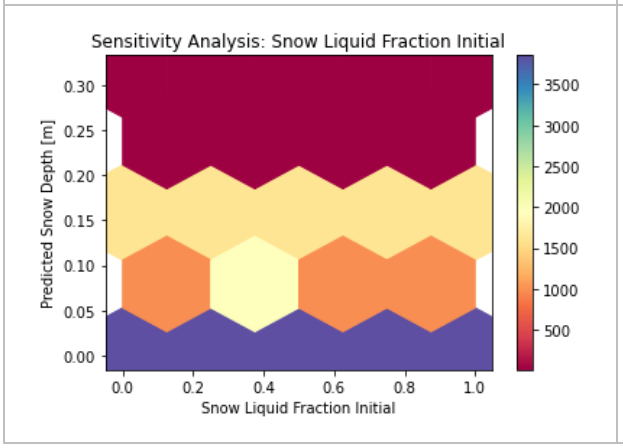

Sensitivity Analysis: Bed Total Depth
Predicted Snow Depth [m]
Bed Total Depth [m]

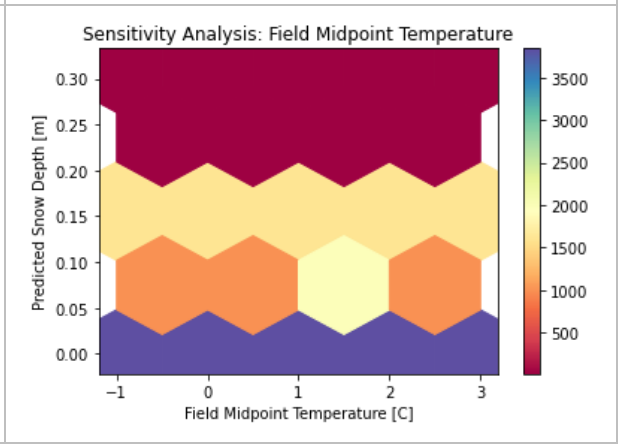

Sensitivity Analysis: Snow Surface Emissivity
Predicted Snow Depth [m]
Snow Surface Emissivity


Sensitivity Analysis: Wind Factor Correction
Predicted Snow Depth [m]
Wind Factor Correction


Sensitivity Analysis: Snow Saturated Hydraulic Conductivity
Predicted Snow Depth [m]
Snow Saturated Hydraulic Conductivity [m/s]


Sensitivity Analysis: Damping Parameter
Predicted Snow Depth [m]
Damping Parameter


Sensitivity Analysis: Snow Liquid Fraction Initial
Predicted Snow Depth [m]
Snow Liquid Fraction Initial


Sensitivity Analysis: Field Midpoint Temperature
Predicted Snow Depth [m]
Field Midpoint Temperature [C]

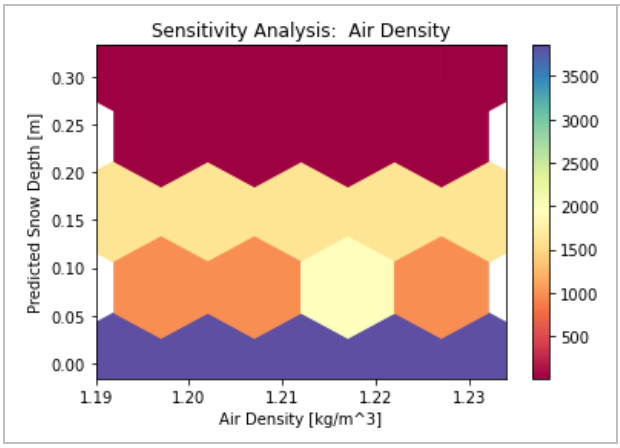


Table 11 presents all the results of the variable study. Each variable included in the study was ranked from most to least sensitive using the hex-graphs above.

Table 11: Results of Variables Study Ranked from Most to Least Sensitive

Ranked Position	Variable	Units
1	Snow Thermal Conductivity	W/mK
2	Turf Surface Emissivity	-
3	Snow Liquid Holding Capacity	-
4	Bed Total Depth	m
5	Snow Surface Emissivity	-
6	Wind Factor Correction	-
7	Snow Saturated Hydraulic Conductivity	m/s
8	Snow Liquid Fraction Initial	-
8	Damping Parameter	-
9	Field Midpoint Temperature	℃
9	Air Density	kg/m^3

It is important to note that the higher ranked variables are often estimations from graphs or ranges of possible values, such as snow thermal conductivity, turf surface emissivity and snow liquid holding capacity. As the code is most sensitive to these variables and the effect on the results is therefore greater, caution must be taken when selecting appropriate values. Those variables are also often interconnected by complicated relationships, for example surface emissivity will change depending on the solar radiation on a particular day. This means that the values would have to be updated continuously at every time-step in the simulation, rather than retaining the constant value assigned at the beginning. Future work could focus on updating the script to replace some of the more sensitive variables with new estimations which will result in more accurate snow depth predictions.

It was noted that even low-ranking variables still visibly impacted the snow depth solution in this analysis. For this reason, removal of variables to reduce computational time was not considered as an option.

4.5.2 Implicit and Predictor-Corrector Method

Alternative numerical methods, previously discussed in 3.3.2 were implemented to compute the snow balance model. Accuracy of a method was determined by the predicted snow depth graphs. An ideal and most accurate graph would have the predicted snow depth (black line) identical to the measured snow depth (blue line). Figure 59 shows the results produced by the original explicit method model.

Figure 60 shows the results produced by the Implicit method, and although more accurate and truer to the real-life data than the results obtained from the predictor-corrector method, Figure 61, the accuracy of the implicit method is not as precise as the original explicit method. In both cases the trends are captured differently, as slopes instead of straight lines, and further improvements could be made by introducing additional levels of complexity to the solvers [72]. Nevertheless, both approaches fail to improve on the original method.

For this reason, a decision was made to not progress any further with the alternative solvers, but to instead begin to improve upon the computing time and accuracy within the explicit solver through a sensitivity analysis studying the variation of time step size discussed in 4.5.3.

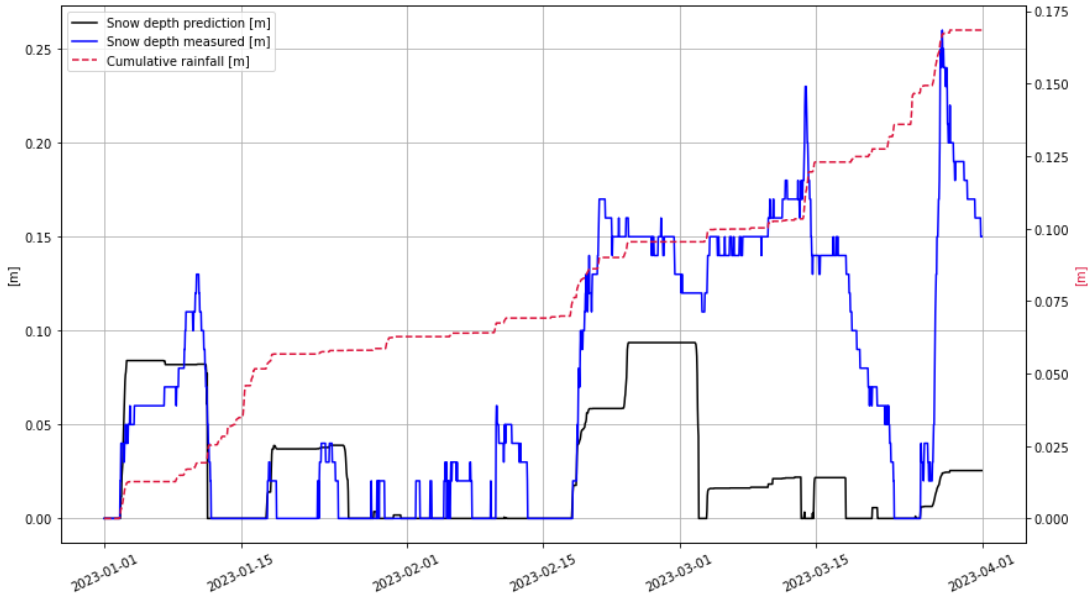


Figure 59: Explicit Method

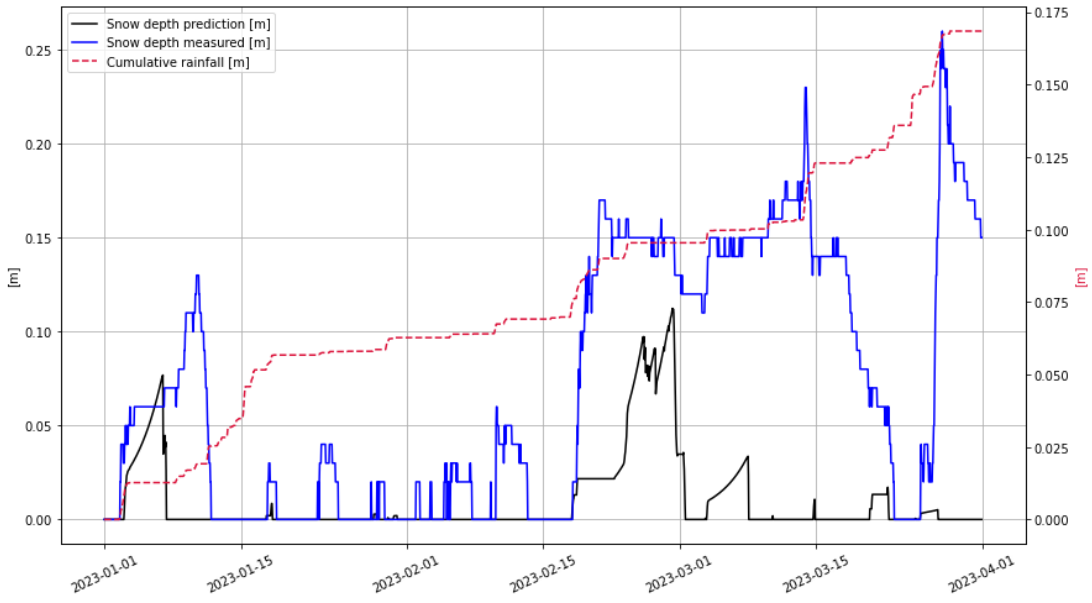


Figure 60: Implicit Method

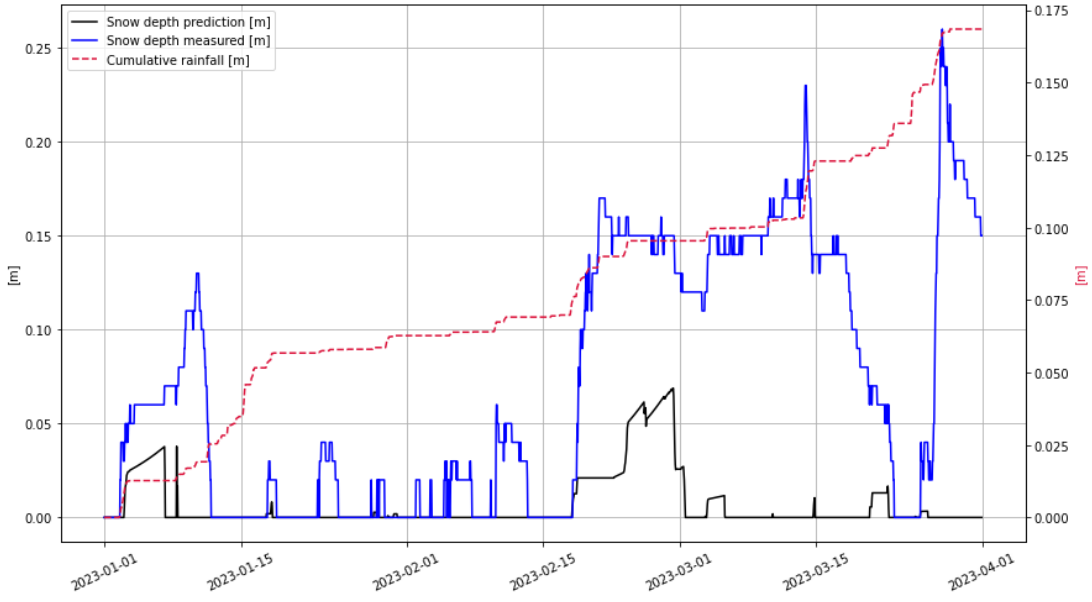


Figure 61: Predictor-Corrector Method

4.5.3 Variation of Time Step Size

Table 12 shows the variation of computational time of the python code with varying time step sizes, the methodology of which was discussed in 3.3.3. For each time step size, the code was executed 3 times, and an average of the recorded computational times was calculated.

Table 12: Variation of Time Step Size with Computational Run Time

Time Step Size (min)	Time 1	Time 2	Time 3	Avg. Time [sec]	Accuracy of Results
180	9.8987696	10.2833476	10.4838072	10.2219748	UNSTABLE
170	10.2275422	10.1487798	10.0650485	10.1471235	UNSTABLE
125	11.0272375	11.5079219	12.4770695	11.67074297	UNSTABLE
120	10.0628477	11.061905	11.3324518	10.81906817	Similar
90	11.276005	11.4905825	11.3867072	11.38443157	Similar
60	14.4811299	14.1855643	13.8547635	14.17381923	
30	18.9880579	18.6461951	18.5381286	18.7241272	More Accurate
20	26.7890872	25.2854188	24.6379858	25.5708306	More Accurate
15	36.4005352	34.970359	35.4149082	35.59526747	More Accurate
10	45.5436176	39.3475056	45.8171928	43.56943867	More Accurate
8	57.6534594	56.9973805	56.6516055	57.10081513	More Accurate

The last column of Table 12 describes the accuracy of the results at each time step compared with the results at the original time step size of 60 minutes, shown in Figure 62. Accuracy was determined by comparing the graphs produced. An ideal and most accurate graph would have the predicted snow depth (black line) identical to the measured snow depth (blue line). Figure 62 to Figure 65 inclusive present the graphs produced for each time step size investigated. As expected, all time steps sized greater than 60 minutes produced results on par or to lesser quality. At time step sizes greater than 120 minutes, the simulation became unstable and did not produce adequate results.

The most stable time step size which also produced the most accurate results was 8 minutes, shown in Figure 65. Compared with the original time step size of 60 minutes, in Figure 62 the differences in results when the time step size was decreased can be seen.

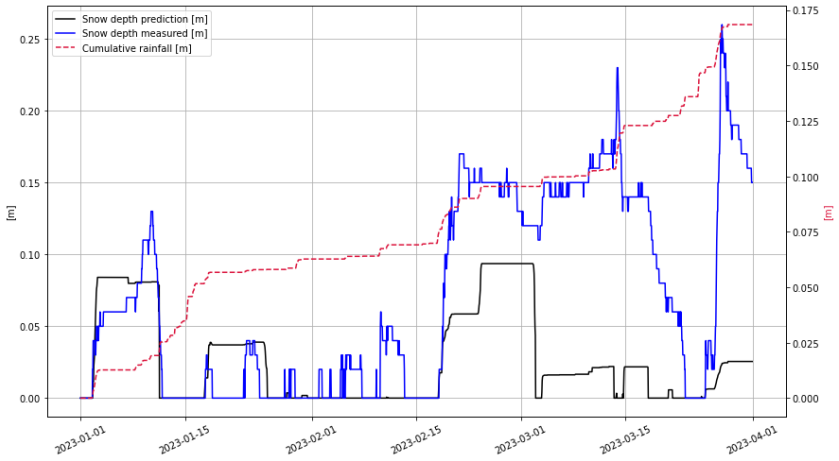


Figure 62: Measured and Predicted Snow Depths for a Time Step = 60 minutes

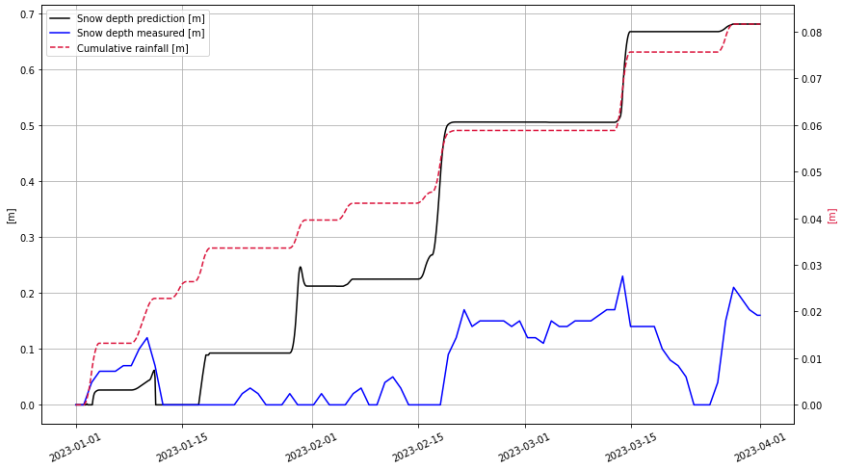


Figure 63: Measured and Predicted Snow Depths for a Time Step = 125 minutes

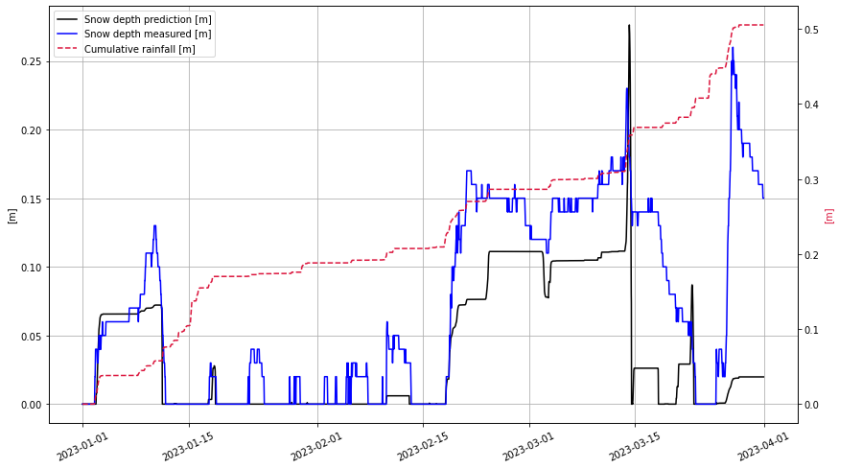


Figure 64: Measured and Predicted Snow Depths for a Time Step = 20 minutes

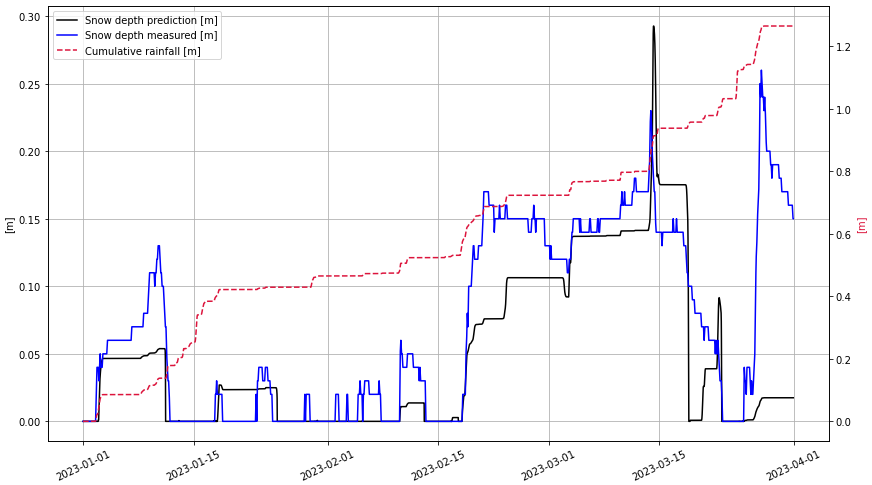


Figure 65: Measured and Predicted Snow Depths for a Time Step = 8 minutes

Although a time step size of 8 minutes came with the improved accuracy, it also came with a greater computational run time, 57 seconds compared to 14 seconds for the original time step size.

Although less than 1 minute was a reasonable computational time when the code was only running one time, when developing the code to reflect the improved snow and mass balance model, the code was required to be run multiple times for each addition to be implemented and verified.

An alternative compromise would be using a slightly greater runtime than the original 14 seconds which still produces improved results, such as a time step size of 20 minutes which takes less than 26 seconds to run. The results shown in Figure 64 show a greater accuracy within the model and therefore were what was used moving forward for the remainder of the computational model refinement.

4.6 Latin Hypercube Sampling

The processes of the LHS within the DoE described in 3.4.1 allowed for csv files of weather data to be generated randomly based on an input consisting of real life weather data taken from the Turku Artukainen weather observation station between December 1st 2022 and March 1st, 2023. This process was undertaken and validated for 500 and 3000 iterations. Each iteration represented a week.

Table 13 shows the results of 10 iterations or weeks which the LHS process was followed with the addition of the Final Snow Depth (m) and Energy Consumption (J) columns which were calculated by using the weather data generated within the improved snow melting model.

Table 13: LHS Results for 10 Weeks

Cloud Amount (1/8)	Precipitation Amount (mm)	Precipitation Intensity (mm/hr)	Snow Depth (cm)	Air Temperature (°C)	Wind Speed (m/s)	ΔT (°C)	Liquid Flow Rate (m/s)	Final Snow Depth (m)	Energy Consumption (J)
3	2.5	0.4	2.4	-9.1	2.4	8.0	1.0	0.1	36295803
3	0.3	4.1	8.4	4.9	4.2	18.6	0.0	0.0	3908021
2	2.3	4.8	9.5	-5.8	0.8	18.3	0.8	0.1	69762325
1	0.8	2.6	2.6	-0.4	1.8	13.2	0.2	0.1	10485458
6	1.0	2.3	7.0	4.0	7.2	4.6	0.4	0.0	9421671
3	1.2	0.3	0.9	3.5	6.9	17.0	0.9	0.0	69033129
5	2.9	3.2	3.6	-13.8	2.2	18.6	0.8	0.2	72688946
4	0.8	4.5	2.7	7.1	3.2	14.6	0.1	0.0	5506162

The results from the LHS simulated weather data for N number of weeks, which were comparable to real life weather observations downloadable from the Finnish Meteorological Institute or any other weather source. It was used as the input data for training the decision tree and complicated machine learning and therefore had to be as close to replicating real life weather trends as possible.

When developing this method some constraints were applied. Each iteration represented 1 week of constant weather data. This is not how weather behaves and therefore it did not fully represent weather trends accurately. Future work could develop a Design of Experiments which allows for weather to vary over time, days, or weeks.

Another limitation within the LHS came when determining values for final snow depth and energy consumption by running the new generated weather data through the developed original model code. For each week, iteration, the initial conditions of snow depth were updated from 0 mm snow depth to whatever value is produced by the LHS. However, other parameters such as temperatures within the layers of the field and within the snowpack itself remained at constant values of $0℃$ and $-5℃$, respectively, for each week.

Improvements to the LHS results overall could be made by directly addressing these limitations. Using similar methods to the code which the initial snow depth used to update for each week, once determined, temperatures of the ground and snowpack could also be automatically updated to represent initial conditions accurately. As the LHS output was the data which the decision trees used to learn, the more accurate a representation it was to real life weather data, the better the decision tree could train and therefore the accuracy of the optimal inputs produced would increase.

4.7 Optimal Control Guidance Document

This section includes the final guidance document which provides the non-specialist human operator of the football pitch optimal control inputs. These control inputs, ΔT and Liquid Flow Rate were selected through machine learning methods to allow a final snow depth suitable for playing on, ≤ 1mm, with minimal energy consumption.

This is a basic, ‘rule of thumb’ guidance which can be used by the operator to get approximately optimal control values.

The operator should follow 3 basic steps:

1. **Check the Weather Forecast**

   This can be from a number of sources, but it is recommended for the region of Finland in which the football pitch is located to use the Finnish Meteorological Institute website [73]. This website allows the reader to access an upcoming forecast which includes daily values for predicted Air Temperature ($℃$), Wind Speed (m/s), Precipitation Amount (mm), Cloud Cover (1/8) and Snow Depth (cm).

2. Follow Guidance Document and Identify Optimal Control Parameters

Figure 68, Figure 69 and Figure 70 make up the 3 pages of the guidance document which will be provided to the operator.

To follow this guidance firstly the operator will have to determine the Air Temperature. If air temperature is greater than 0.3℃ then the operator should follow page 1, Figure 68. If the air temperature is less than or equal to than 0.3℃ then the operator should follow page 2, Figure 69 which leads into page 3, Figure 70.

Regardless of what page the operator starts on the process for determining optimal control input is the same. Starting on the yellow rhombus as shown in Figure 66, follow the flow chart downwards.

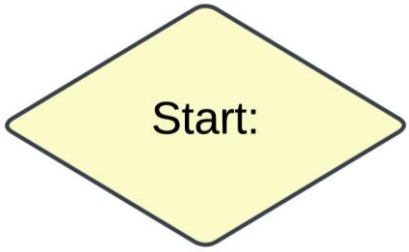


Figure 66: Yellow Rhombus, Start Point

The decision on what path to take, either yes or no, green, or red, depends on the answer to each subsequent decision rhombus based on the weather data from 1.

Note: Page 2 leads into page 3 through 2 possible decision arrows. The flow chart continues and should follow the same path as it moves between pages.

All end nodes of the flow chart are pink rounded rectangles, Figure 67. These nodes contain values and/or ranges of values for the control inputs.

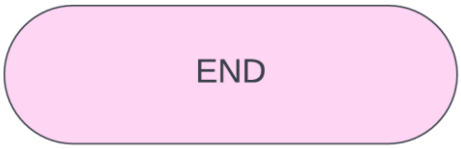


Figure 67: Pink Rounded Rectangle, End Point

3. Set ΔT and Liquid Flow Rate to Optimal Values

Based on the control values determined by 2, the operator should set the physical controls to acceptable values within the ranges.

These 3 steps are all that are required from the operator to ensure that the football pitch is returned to adequate playing conditions with a snow depth of zero achieved through minimum energy consumption.

The operator will still have the task of determining specific control values within the ranges determined from the flowcharts below. This allows for some room for human error to occur. The optimal control guidance document focused in on near to optimal control input values but did not take into consideration all weather conditions experienced at any time.

This guidance document was a foundational stage for the underground heating system considered within this report. Prior to completing this stage of the project, the control inputs were selected by a specialist operator on the basis of previous experience, rather than following a pre-defined set of conditions. Therefore, the underfloor heating system was not using the most optimal control input parameters, which means it was not using the minimum possible energy consumption whilst still allowing for adequate playing conditions. The current method for determining ΔT and Liquid Flow Rate values priorities snow melting on the pitch without considering the energy consumption at all. Therefore, even although this wasn't the most accurate and most optimal guidance, it was still a significant improvement on the current method, simplifying the process and no longer requiring a specialist operator, and meaning that adjustments could be easily made if the weather forecast changes throughout the day. The guidance document itself was designed to be user-friendly, presenting branching information in an organised way. Making sure that the information was easy to process for a non-specialist operator, simply by reading upcoming weather forecasts, was another key step in ensuring that the system would be operated correctly, and subsequently in an energy-efficient manner.

The applications for optimal control guidance documents of this sort goes far beyond controlling the underground heating of an amateur football pitch in a small city in Southern Finland. With sports grounds across the northern hemisphere and beyond, all experiencing periods of cold weather and frozen pitches, resulting in cancelled or postponed games. Detrimental knock-on effects to the mental and physical wellbeing of millions annually. With energy prices increasing dramatically, individuals and businesses are looking to reduce costs in all areas so are not likely to install or implement an underground heating system without significant personal benefit. The optimal control guidance document could be developed and adapted for different conditions to provide the most optimal control parameters for underfloor heating systems across the world which can determine the minimal energy expenditure whilst still maintaining playable conditions. The money spent on powering the underground heating system could be significantly less than the loss in revenue if games are cancelled, particularly within professional sports teams.

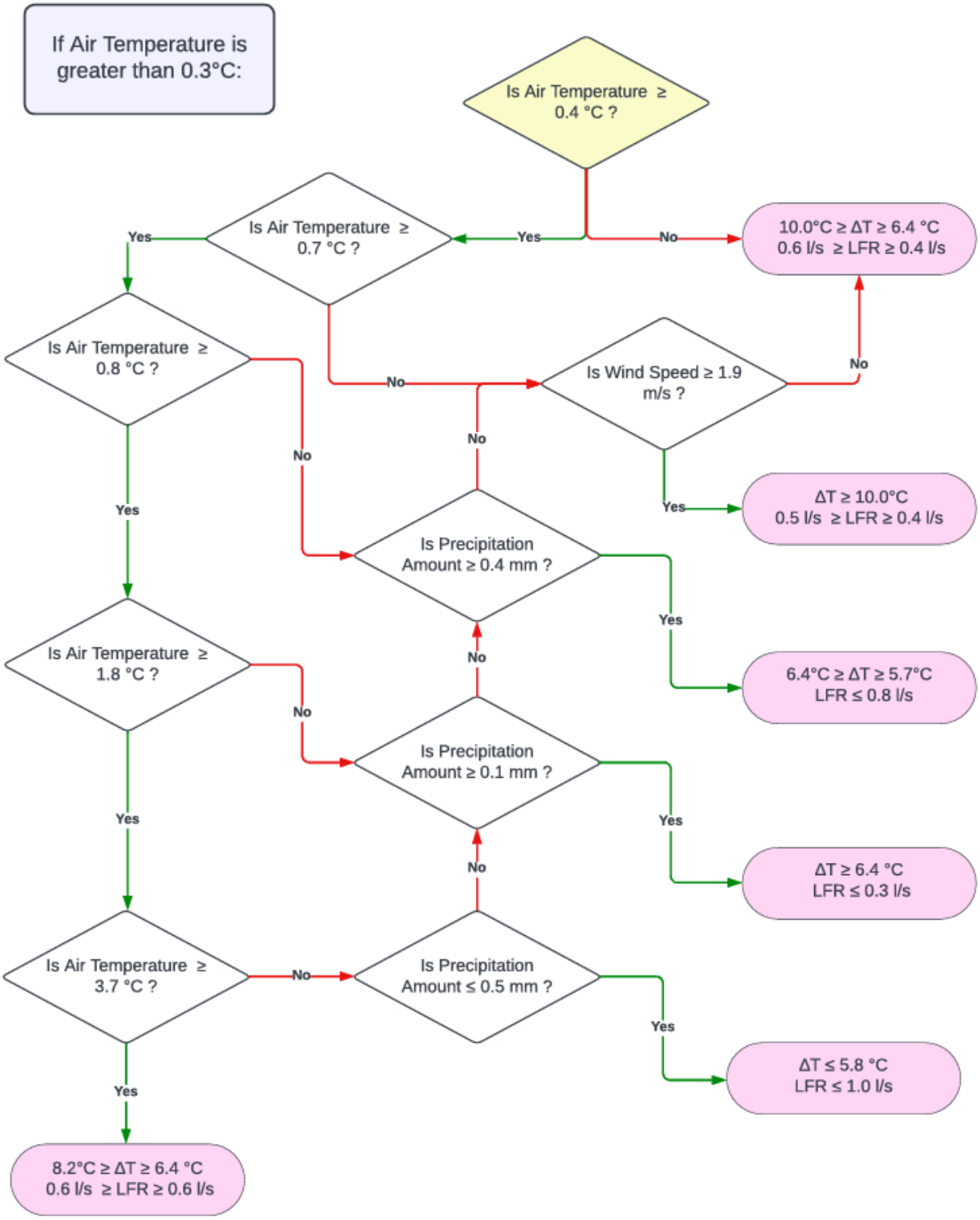


Figure 68: Optimal Control Guidance Document Page 1

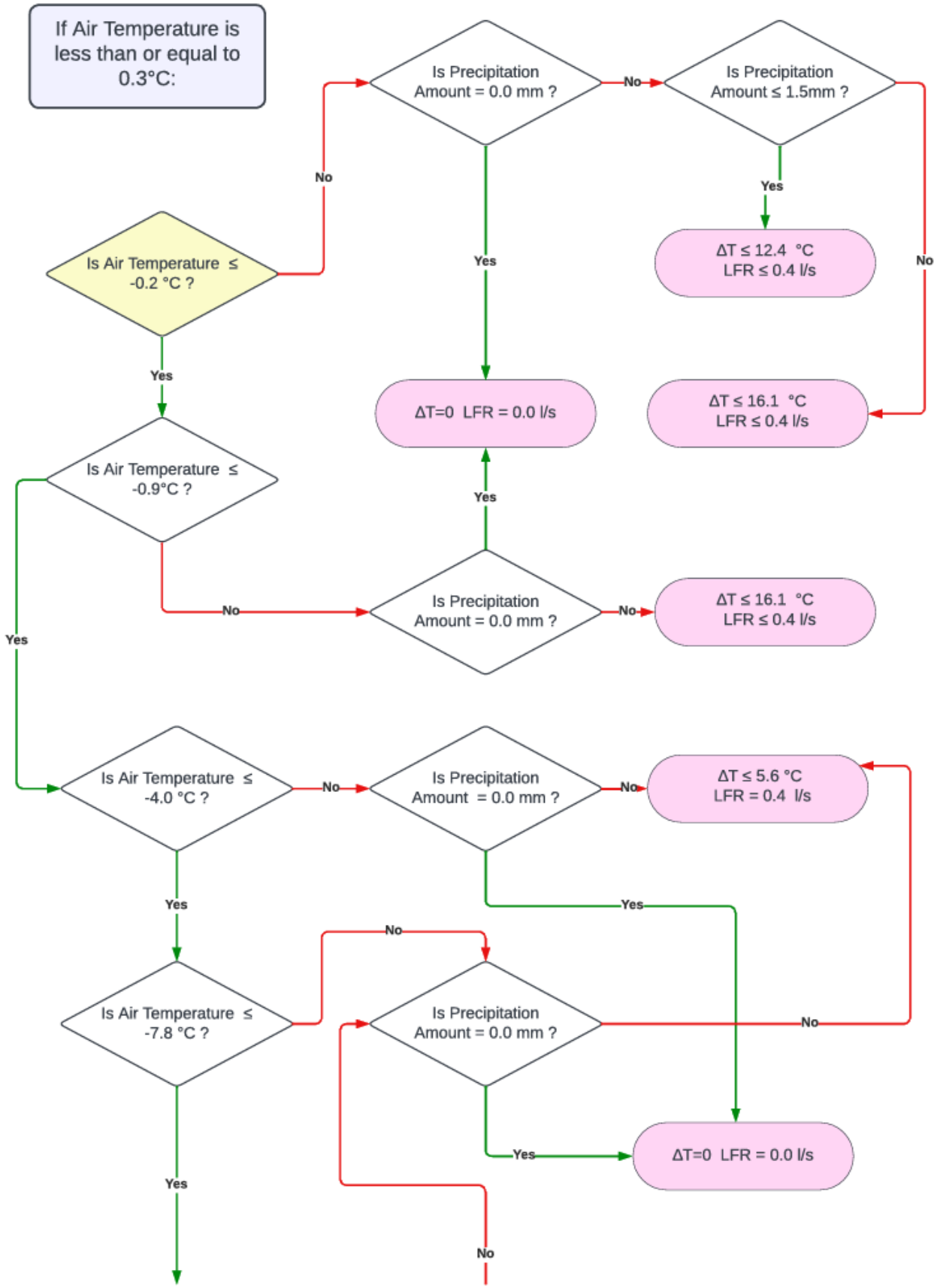


Figure 69: Optimal Control Guidance Document Page 2

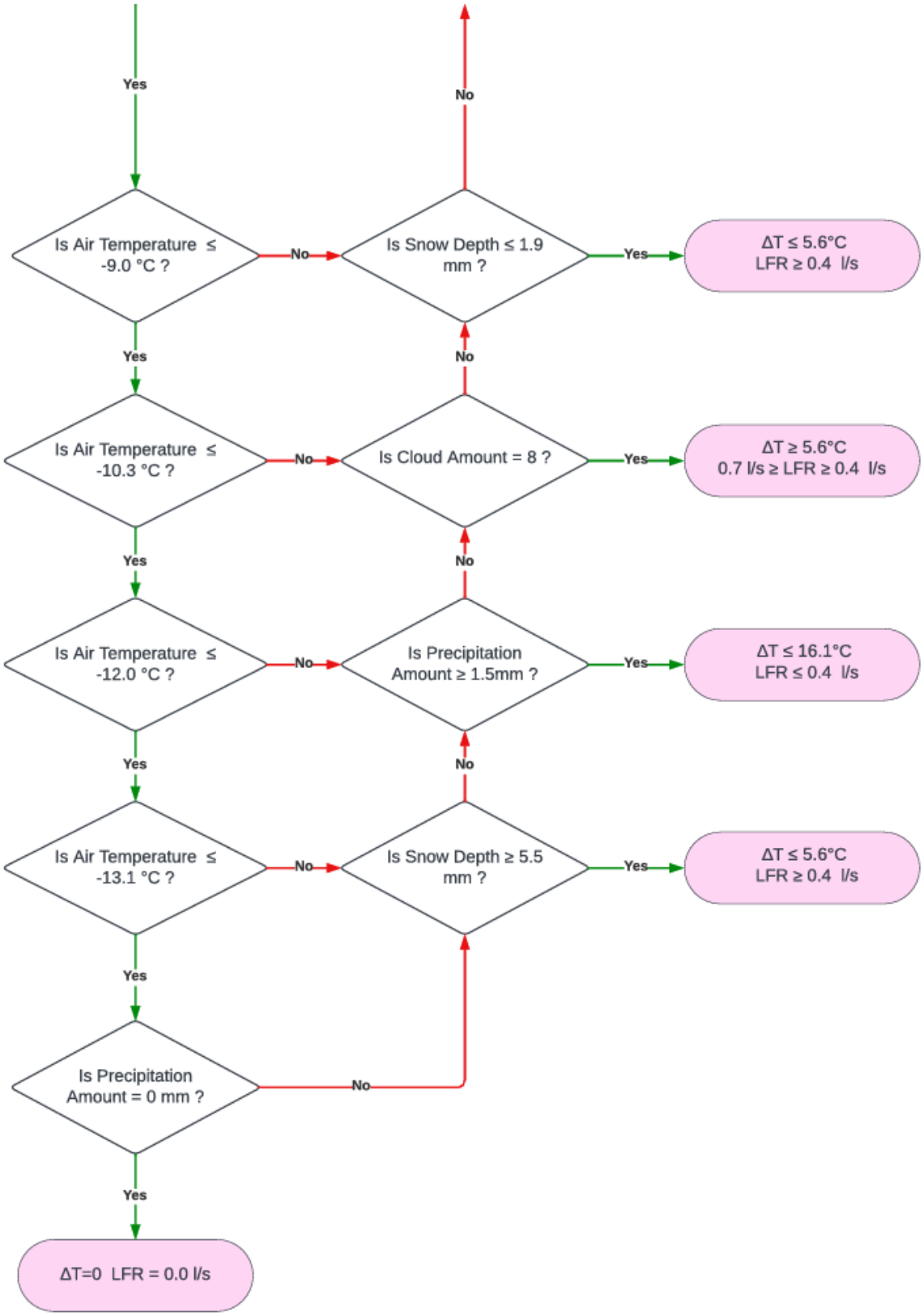


Figure 70: Optimal Control Guidance Document Page 3

After the data was organised for the creation of the guidance document, it became apparent that some weather parameters had a much more important role in the decision-making process of the regression model. As it was expected, air temperature and precipitation amount appeared the most and were found to have the largest ranges of values, as those were the key parameters determining whether snowfall would occur. Therefore, those values were taken as the leading choices in the guidance document. Every branch begins by assessing the value of air temperature. Once the range of temperature is identified, the next leaf in the branch can be followed. Precipitation amount was the second most common variable, but other parameters were also included, allowing conditions where no snowfall is currently occurring, but snow cover is still present, to be captured. Based on the 2 conditions, a suitable range was identified for the control inputs to meet the conditions described above. This also served as an additional validation process for the algorithm, proving the correct factors were chosen as part of the prediction. Others, such as wind speed, were entirely omitted, supporting the conclusion reached as part of Section 3.2.8 that the impact of wind on the snow distribution could be neglected.

One of the features of the simple guidance document was that it provided ranges of values, rather than a set point. This allowed for some flexibility, where lower values could be used if the pitch did not have to be in use within a certain timeframe. This allowed for the lowest energy solution. Higher values in the range could be picked when there are time constraints, to minimise the downtime. However, this could also become a limitation of the model, as while more accurate than the current system, leaves room for wrong interpretation. One of the most important factors when considering energy consumption was not only determining the cases where the conditions were favourable, but also situations where conditions were adverse. These could be described as situations where regardless of the values of ΔT and liquid flow rate, not enough snow will be melted to achieve playing conditions. This could occur in cases of extreme snowfall for a continuous period of time, or if the system has been out of use and a large amount of snow cover has accumulated. Having the ability to identify such cases based on the forecast could further improve the energy efficiency of the system as it will simply not be used if it is not capable of maintaining the necessary surface conditions. This was partially investigated as part of the decision tree validation, and a snow depth of around 9 cm was established as a near-critical value for the melting. While this was the case, it left a lot to be desired in terms of the specific conditions which led to this result, due to the prevalence of air temperature and precipitation amount in the decision tree output. Further investigation will be required to establish the exact relationship between those values and to validate the exact snow depth at which the heating network would no longer be effective.

4.8 True Optimal Control

As discussed in Section 4.7 above, the limitations of the simple guidance model led to the development of the complicated machine learning algorithm, which would predict the exact values of the required control inputs.

The metric of Mean Absolute Error (MAE) was calculated to determine the accuracy of predictions made by the Decision Tree, for optimal control parameters. The random state value of the train test split was changed for 4 different cases to validate the mean absolute error for different versions of the trained decision tree.

Table 14: Mean Absolute Error Values for Optimal Control Decision Tree

Random State	ΔT Mean Absolute Error (℃)	Liquid Flow Rate Mean Absolute Error (l/s)
0	6.703561611570248	0.3153286033057851
1	6.999704797520661	0.3384494380165289
2	6.751770942148760	0.3412534338842976
3	6.179161735537190	0.3355962561983471
Average Values:	**6.66**	**0.33**

Table 14 shows the MAE value for both ΔT and the Liquid Flow Rate for 4 random state values. The average Mean Absolute Error for ΔT was 6.66℃, with an operational range of 0℃ to 20℃ giving an error of 33.3% and the average Mean Absolute Error for Liquid Flow Rate is $0.33\,l/s$ with an operational range of $0.01\,l/s$ to $1\,l/s$ giving an error of 33%. Both MAE values were within an acceptable range and confirm the accuracy of the complicated machine learning model.

The complicated machine learning model ran for 3000 iterations, Table 15 represents an example of the types of outputs produced for the first 10 iterations. It shows that for each iteration with specific and unique weather parameters optimal control input variables, ΔT and the Liquid Flow Rate values, had been determined.

Table 15: True Optimal Control Values for 10 Iterations

Cloud Amount (1/8)	Precipitation Amount (mm)	Precipitation Intensity (mm/hr)	Snow Depth (mm)	Air Temp (°C)	Wind Speed (m/s)	Δ T (°C)	Liquid Flow Rate (l/s)
2	0.2	0.1	9.7	6.6	6.4	19.8	0.8
2	1.5	0.4	23.8	0.7	4.0	13.8	0.8
1	2.3	4.6	4.9	6.8	4.8	9.9	0.5
7	1.8	1.9	25.0	4.8	2.4	8.6	0.3
5	0.2	0.7	9.4	7.9	8.6	2.0	1.0
3	2.6	3.8	1.8	5.8	1.1	2.0	0.7
0	0.3	2.6	4.3	-3.7	8.0	5.7	0.9
1	0.4	1.9	10.9	1.9	7.6	0.4	0.3
2	0.9	5.0	14.8	7.4	5.8	5.2	0.1
0	2.0	1.7	11.8	0.5	0.7	16.8	1.0
4	0.1	1.9	18.2	5.2	5.8	14.2	0.2

The above results show that through further refined training and testing, this approach for determining control inputs, ΔT and the Liquid Flow Rate, shows that it is possible to determine optimal control inputs for any combination of weather scenarios.

Figure 71, Figure 72 and Figure 73 make up 3 pages of a summarised complicated guidance document. This particular guidance document was created using the 3000-iteration trained and tested data but had been simplified significantly to 30 possible control inputs based on trends in the weather data. The complicated guidance is a foundation for a full and extensive guidance document which could be developed for a non-specialist human operator in charge of the control of the underground heating system. The complicated optimal control guidance document should be read and implemented the same way as the general guidance document described in 4.7, with page 1 leading into page 2 and page 2 into page 3.

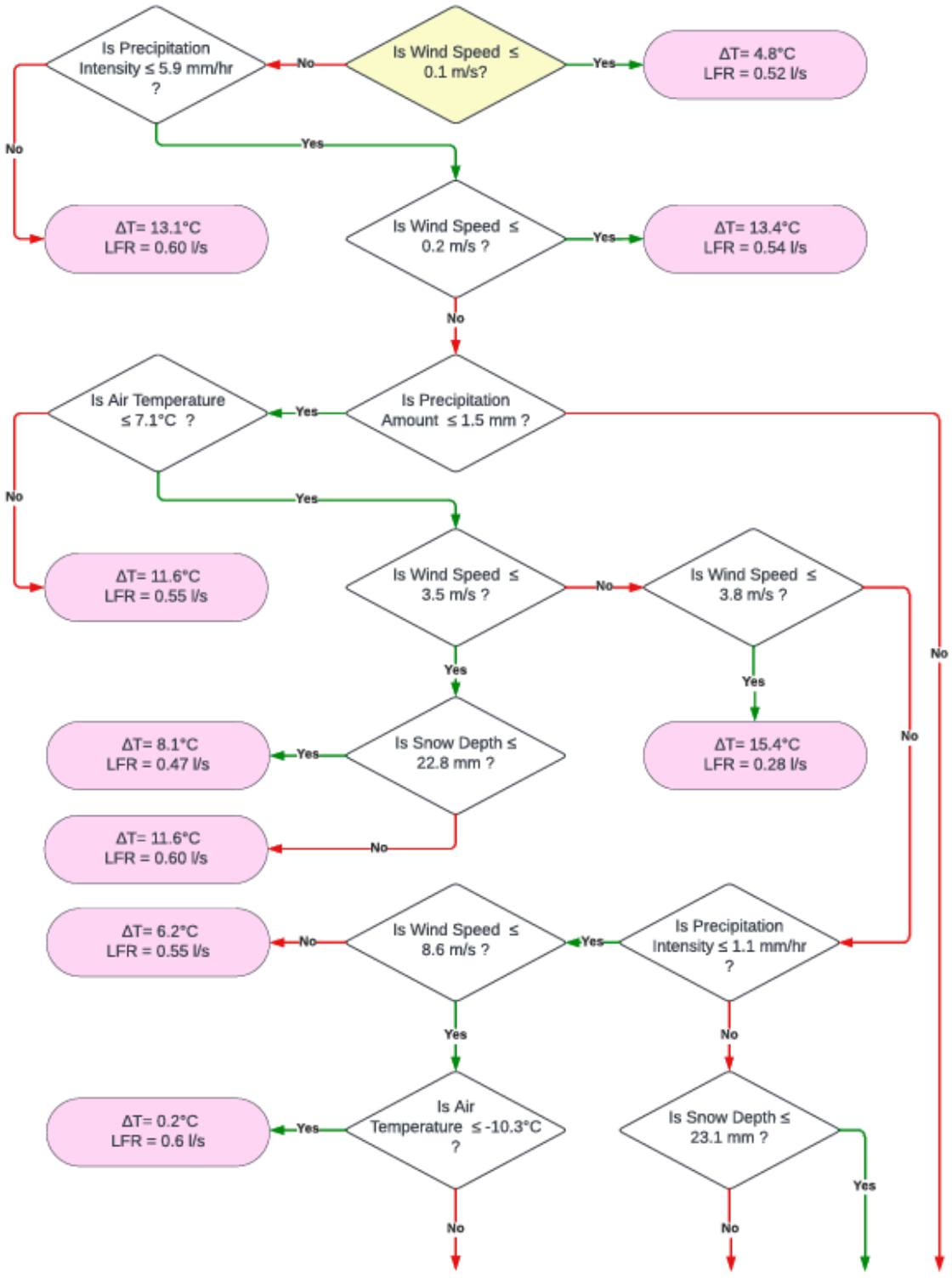


Figure 71: Complicated Optimal Control Guidance Page 1

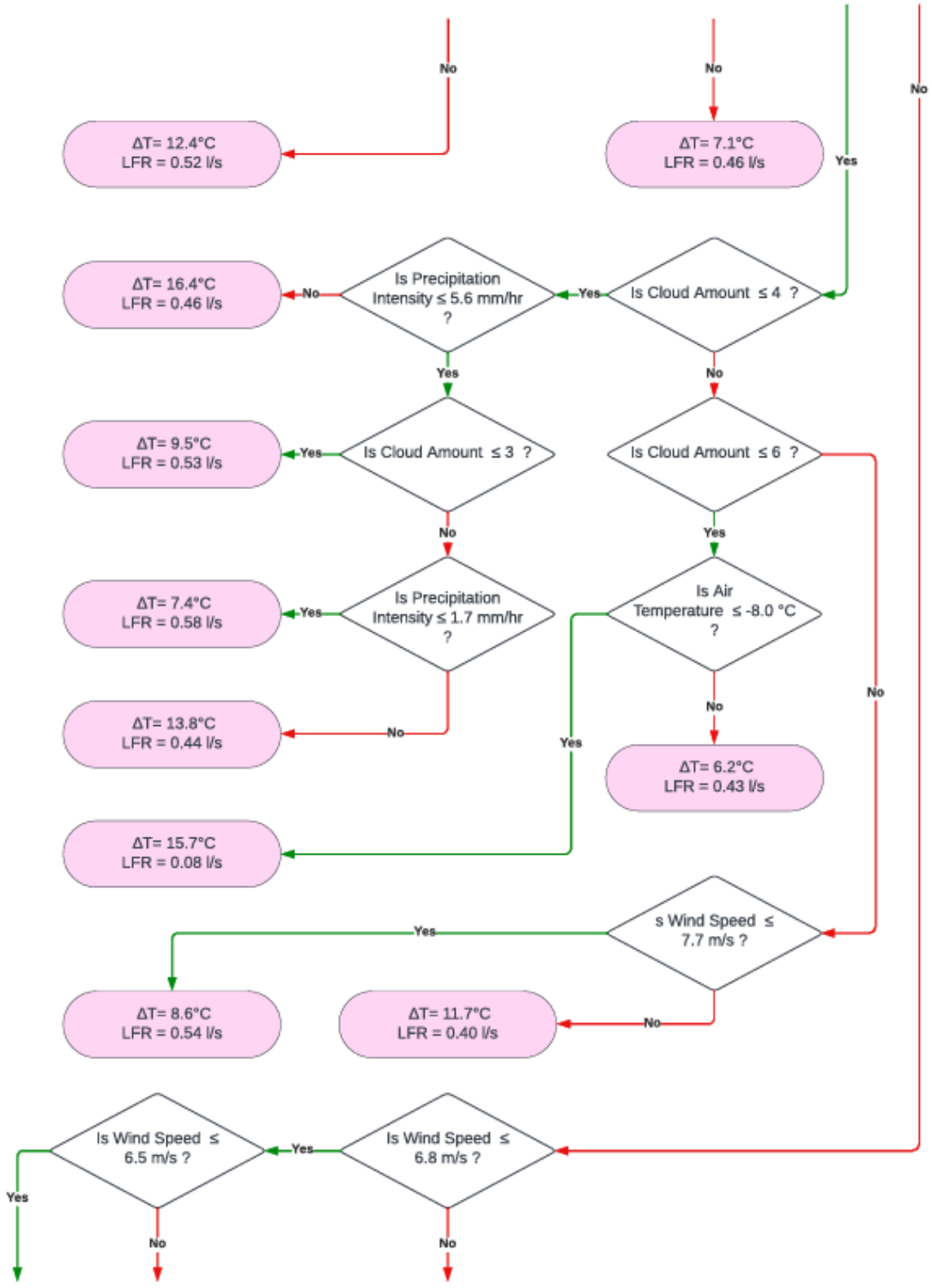


Figure 72: Complicated Optimal Control Guidance Page 2

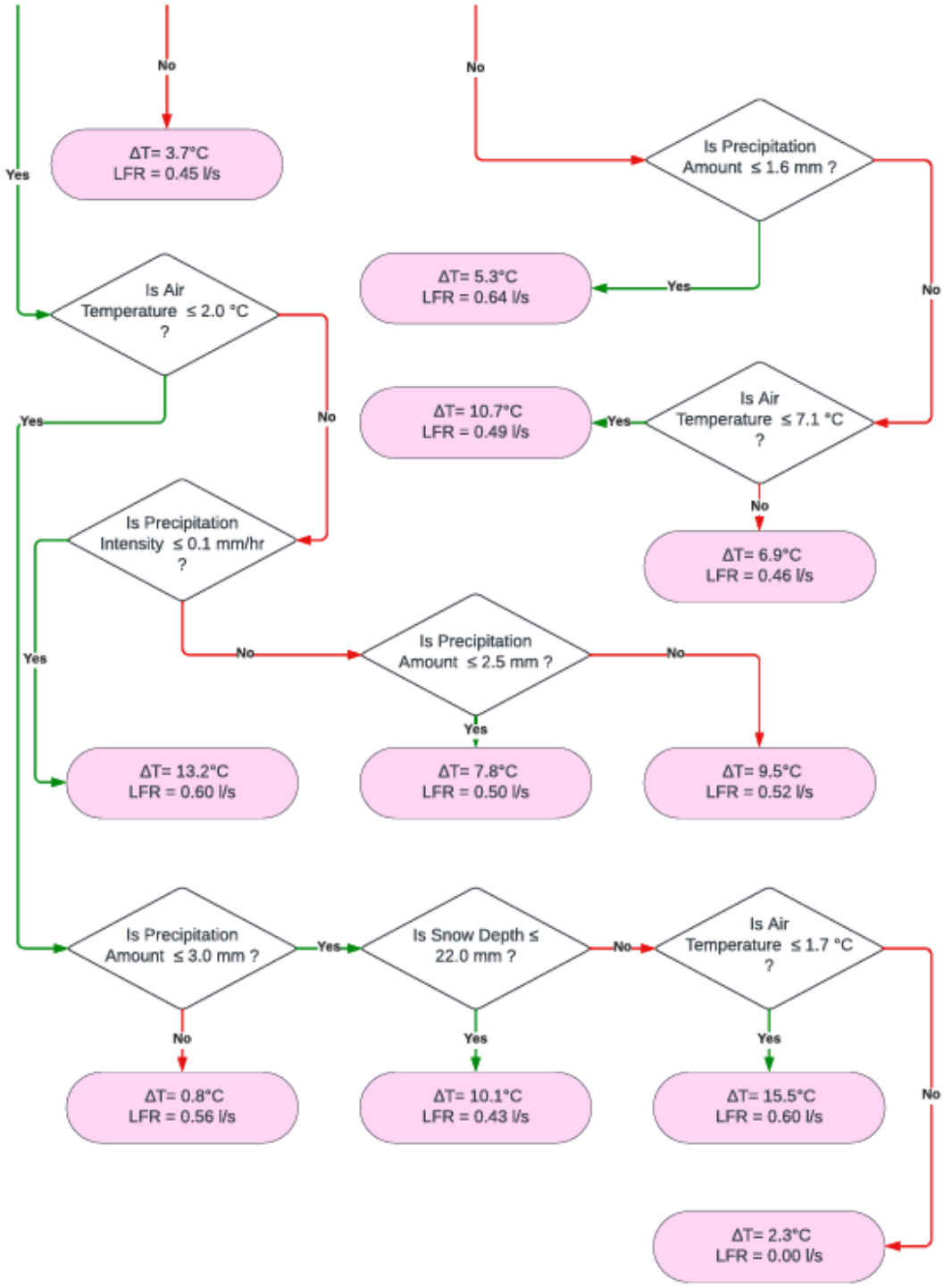


Figure 73: Complicated Optimal Control Guidance Page 3

The complicated optimal control guidance document differed from the generalised optimal control guidance document of 4.7 as it considered more than 2 weather conditions for each control input suggested.

Another way that the complicated optimal control guidance document differed from the generalised optimal control guidance was that it provided an exact value for ΔT and the Liquid Flow Rate and not a range. It therefore presented a more reliable and accurate method of training and testing weather

conditions as it provided the exact value that the human operator should set the input controls to over the course of a week, leaving less room for human error compared to having to select values within the ranges provided.

The increased specificity of the solution also carried its own set of problems. As the number of specific cases of control increased, the amount of data required to produce those values also increased significantly. This risked creating a set of instructions which was too narrow and complicated to be used by a non-specialist operator. A balance was required in selecting the depth to which the guidance should be developed, to still allow for weather parameters to be grouped for simplification.

In addition, the training set used during the creation of this model only considered cases where the snow would be sufficiently melted with a lower-than-average energy expenditure. This excluded a subset of conditions where the snow would not be fully melted or the energy expenditure would be too high, which could result in a small sample of extreme cases where the algorithm would recommend controls which would not work. Including this data would have required manually identifying those cases in the training data and assigning them zero values for both control inputs. This was once again beyond the scope of the project, which only requested a simple guidance, and time constraints did not allow this to be fully explored, however it provides an interesting avenue for future research on the topic. This also strongly relies on first developing the knowledge of which weather conditions would lead to insufficient melting, previously mentioned in Section 4.7.

A final and most optimal guidance would be trained and tested with a larger number of iterations than 3000 and would present optimal control values, ΔT and the Liquid Flow Rate, for all 6 combination of weather conditions included within the training. Weather conditions included currently are Cloud Amount (1/8), Precipitation Amount (mm), Precipitation Intensity (mm/h), Snow Depth (cm), Air Temperature (℃) and Wind Speed (m/s). The difficulties with creating a guidance document of this scale are the large range of possible combinations of conditions which can occur to replicate real life weather scenarios, this would require a larger time frame of concentrated focus to develop written guidance in some format which could easily be understood. This task goes far beyond the scope of this project currently but is something that the COMEA research group hope to progress with in the future.

Although different in nature, both control approaches were valid ways to tackle the issue of determining the optimal control inputs for different weather scenarios. The simple guidance could be utilised with an extensive dataset in the creation of a general set of ranges for conditions and control. Once those have been determined, the complicated machine learning algorithm could be used for determining the precise inputs within this range required to maintain playing conditions. This would both make the process of creating a new guidance document for a new set of conditions or location easier, by simplifying some of the very specific solutions into more general categories, but also allow less room for wrong interpretation as those ranges can be

broken down into more refined categories, leading to more efficient maintenance and larger energy savings.

4.9 Cost Analysis

To validate outputs produced from the true optimal control decision tree, a cost analysis was carried out to compare how the dynamically adapted values recommended by the algorithm compared to the original 'fixed' operation method.

Notably, as optimal control was calculated using a Machine Learning algorithm and the system only requires human input to increase operational efficiency, no material or implementation costs were involved. This meant that to determine the savings that the optimal control algorithm could provide, money saved during operation for a 3-month period could be directly compared between optimal and standard control.

Firstly, the 3-month period between January – March 2023 was chosen for the weather data, to be kept constant between the cases. One case maintained standard control parameters of constant $0.4\ l/s$ Liquid Mass Flow Rate (chosen as the lowest flow rate that could melt enough snow to maintain the field in playing conditions), and an Inlet/Outlet Temperature Difference following existing time delay control conditions, as described in Section 4.2. The second case took the same weather data and used the algorithm developed as part of Section 4.8 to predict the exact values of the control inputs which should be implemented. The new control inputs were then introduced to the original calculation.

After running both examples, savings of around 80GJ or 22,225 kWh were observed for the second case, confirming that the optimal control algorithm allows for significant energy savings. Assuming that the heating system was powered by the grid as opposed to power station excess heat, and using an average market price for district heating from January to March of €100/MWh [74], the money saved in this situation was around €2222.50 over 3 months.

The original COMEA paper [1] shared an example of a similar undersoil heating system in Sweden, estimating that the total cost of heating the system for 1 year was around €85,000. Assuming that a football pitch in Finland would cost the same, and that €2222.50 is saved every 3 months, use of true optimal control predictions allows for a saving of €8890 annually.

4.10 Recommendations For Future Work

This project reviewed and advanced the work published in [1]. The following section proposes recommendations for future research advancements based on the work undertaken within this report.

Firstly, Section 3.2.8 briefly described the negation of snow distribution due to the effects of wind speed. This assumption was based on previous studies which concluded that wind speeds up to $7\ m/s$ had little effect on the snow cover, which far exceeded the values measured in the case of the football pitch considered as part of this project. These results were determined for building rooftops, which are usually free of obstruction or surrounding

interference. Future work could confirm these negations through the creation of a large-scale model to simulate the field and surrounding areas, which would include interference by nearby buildings and trees.

As discussed in Section 3.2.7, the snowpack was classified into 4 categories for the purposes of the calculation, but in reality, a snowpack can consist of any number of different snow types within one snowpack at any one time. Therefore, future research could investigate the implementation of multi-phase snowpacks within the heat and mass balance computational model, further refining the existing model to capture the true conditions more accurately.

Moisture Transport was a section that could continue to be studied in future work. Firstly, the moisture simulation itself could be expanded. An example of this could be simulating the transport of mass in the form of particles in the soil or comparing results between different turbulence models in place of laminar flow. When applying the techniques used in the study to another project, Ansys could be used again to model the physical processes occurring in and around the system.

While the Python model of the pipe network provided good estimations of the time delay, further improvements could be made to the model by performing a large scale, more complicated simulation in software such as Ansys Fluent. The script and results of the study conducted as part of this project could be used to determine the necessary input parameters for the complex simulation. This could then be used to create a visual representation of the circulation of the fluid in the network. While this is highly computationally intensive and not possible to run locally, it could allow for the properties of the soil surrounding the pipes and other parameters to be incorporated. This would provide the most accurate representation of the heat transfer mechanisms occurring within the pipes.

Physical modifications to the existing system were not possible at the time of writing, therefore the work carried out on this project focused on the development of the computational model for optimal control. In the future, further improvements could be achieved through the use of new technologies such as inserts in the pipes to improve heat transfer capabilities [75].

Furthermore, the scope of this project did not require for a full and extensive operator guidance document to be produced beyond the ‘rule of thumb’ guidance from the simplified decision tree. However, the results generated from the complicated machine learning approach described in Section 3.4.3 and the shortened 30 node complicated optimal control guidance could be used as a foundation for determining optimal control values for a much wider range of weather conditions. Trends could be identified for very specific cases, grouped in the form of a guidance document which could be understood by non-specialist human operators. This process could be accelerated through the use of other machine learning models. For example, an unsupervised learning model could understand the large decision tree output and group the corresponding branches for optimal control. Remaining in the area of supervised learning, ensemble methods could be applied to

increase prediction accuracy, such as a Random Forest algorithm, or a combination of Machine Learning algorithms and Deep Learning. Also, increasing the number of data samples in the dataset often vastly improves the prediction power of a Machine Learning algorithm. In future we could further extend the number of generated weather conditions to achieve this increase in accuracy.

In the Design of Experiments, the values for the snow depth and energy consumption at the end of the simulation were acquired by fixing the weather parameters for the full duration. Using more complicated methods, such as incorporation of non-constant weather could further improve the accuracy of the machine learning model as this would reflect real world conditions more accurately, improving the quality of the training data used.

Finally, this model was designed for the very specific purpose of being implemented for the football field in Naantali. As discussed in Section 4.9, the optimisation recommended as part of this project could contribute to savings of up to €2222.50 over the 3-month period simulated. This could have significant implications in the wider sports industry, where a small model, modified to account for the geometry of the specific location could be used to optimise energy usage in the heating network.

Many other applications also exist beyond sports, in the maintenance of public infrastructure and roads. High traffic areas such as city centres could benefit greatly from the technology, reducing emissions from snow-clearing vehicles and the associated pollution from the use of salt and sand for de-icing, the costs of which are discussed in Section 2.1. Wider adoption could also contribute to reducing delays in public transport and minimising the impact of snow and ice accumulation in public spaces.

5.0 Conclusions

In conclusion, the use of computational modelling in the prediction of snow accumulation and subsequent operation of an underground heating system could help greatly reduce energy expenditure while also avoiding the high cost of refitting the existing infrastructure. This project aimed at improving a model developed for a sports field located in Finland, however it has wide ranging applications beyond current uses, including in the maintenance of public spaces and transport hubs.

The first main outcome was an improved computational model for prediction of snow accumulation. This was achieved through the implementation of additional features into the existing code developed by the COMEA Research Group at Turku University of Applied Sciences. This took into consideration the moisture transport of meltwater, time-delay in the distribution network and evaporation and condensation.

The final conclusion of the Moisture Transport simulation subtask was that the movement of liquid water beneath the football pitch did not affect the thermal properties of the soil in a significant way. In the 4th iteration of moisture simulation, the quantity of water entering the inner soil through horizontal motion was never more than 3% of the simulated cross-sectional area. With a precipitation amount that rarely goes far beyond $1\,mm/h$ in reality, the quantity of water entering a real system would be even lower. While the impact of moisture could be considered negligible in this case, if the heating system were to be used in other situations in the future, for example where the pitch layer cannot be considered impermeable, the moisture transport may have more of an effect. For this reason, the moisture transport subtask was effective in functioning as both a research task, and an adaptable technique that could be used in future work.

Studying the time delay and the subsequent imbalance created in the melting of snow on the pitch surface showed that depending on the ambient conditions, there could be a difference of around 7 hours on average between the network being initiated and all sections of the system reaching the required temperature. This would significantly increase as the initial temperature decreased, therefore activating the system at an appropriate time could prove key to tackling this issue. Furthermore, the liquid flow rate and temperature of the solution proved to have a similar effect, meaning those parameters had to be chosen carefully to avoid creating significant differences in the melting.

Following a detailed comparison of features from the 2007 paper [2], which introduced a computational heat and mass transfer model for an underfloor heating system on a pavement against the original model, incorporation of these features came in the form of the inclusion of evaporation and condensation in the overall heat transfer model. The effect $\dot{Q}_{Evap/Cond}$ had on the overall model was minimal and made only a slight improvement to the accuracy of Snow Depth prediction. It was however kept within the improved computational model for future usage which will use weather data for different time periods or different observation locations.

The results were then compared to the Original Code for 2 sets of test data – one for the January-March 2021 period and the same period in 2023. This analysis was carried out to validate the implementation of additional code, showing that the computational model showed improved results and behaved as expected for multiple sets of data.

Moreover, sensitivity analysis was performed on the model with regards to both the variables utilised in the calculations and the numerical scheme implemented. For the former, 11 parameters were chosen for the study, on the basis of exploring the importance of parameters which were more obscure and therefore very difficult to measure in the real-world, for example radiation such as turf and snow surface emissivities. Each variable was assigned 5 values within a range and the results were plotted using hex-graphs and ranked in order of importance. Snow Thermal Conductivity was found to have the greatest impact on the calculation, while Air Density was the least significant. 2 additional numerical schemes were tested in place of the existing 'explicit' scheme: implicit and predictor-corrector. Several functions within the script were modified to adopt the new solver methods, Backward Euler and Modified Euler respectively, however in both cases they were found to be less accurate than the existing implementation. Therefore, the explicit scheme was deemed the most accurate as the predictions it produced most closely resembled the real-world measurements.

In addition, the impact of wind speed was investigated as another possible inclusion in the computational model. The behaviour of the snowpack has important implications in the accuracy of the calculations, and therefore a greater understanding of the effect of snow drifting caused by high wind speeds was essential. Similar studies conducted for rooftops in buildings discovered that wind speeds as high as $7\,m/s$ had minimal impact on the distribution of the snowpack. This was taken as the minimum boundary, and weather data from a weather station located around $10\,km$ from the football pitch spanning several months was analysed. The 2 main findings were that the wind speed at that location remained significantly below the threshold for the period analysed and that measurements were carried out at altitude, where little interference is present. The additional considerations of the presence of buildings in the area surrounding the field and differences in elevation meant that the wind speed would be even lower than the measured values. It was therefore concluded that the effects of wind speed could be negated throughout the report.

The improved computational model was then utilised in a Design of Experiments, where Latin Hypercube Sampling was used to produce a large set of random samples of weather data and control inputs. The new data was introduced into the original calculation and simulated for the duration of a week, generating results for the snow melting and energy consumption in various scenarios. These results could then be analysed and ranked according to the principle of minimising both snow cover and energy usage. To simplify the training process, thresholds were identified for which the field could be considered in playing conditions. For snow depth, this was selected as $0.1\,cm$, while the energy consumption had to be below the average value

for a set of 3000 simulations. When a simulation result met these conditions, it was assigned a value of 1, otherwise it was allocated a 0. The true/false designation was useful in presenting the algorithm with a clear target variable to aim for.

The outcome of the Design of Experiments was used as the basis for the creation of a small machine learning model. The ranked scenarios were implemented, with a large segment making up the training data, and a regression model was fitted to predict whether the criteria would be met for the remainder, or test data. The decision-making process was further investigated and validated using a set of parameters the model was unfamiliar with. The validation process found that the model could predict the outcome correctly with over 80% accuracy, before being used in the subsequent creation of a guidance document for the control of the system by a non-specialist operator. This included the necessary control inputs given a certain set of weather parameters, ensuring that the field remained with minimal snow cover while also reducing the energy consumption.

This guidance was simplified and could be considered ‘Rules-of-Thumb’, and so it included broad ranges of values for both the weather and control inputs and was mostly based on Air Temperature. Therefore, a complicated machine learning model was also developed, which only took into account conditions where enough snowmelt had occurred and energy consumption was below the average value, meaning only scenarios which would receive a ‘True’ designator were included. This model was used to predict the exact values of the ΔT and Liquid Flow Rate which would be required to achieve those conditions, resulting in a much more complicated decision tree. Due to the time constraints of the project, it was not possible to process this amount of data in its entirety for this project, however future work could tackle the process of developing more efficient ways of doing this, perhaps through the use of other machine learning models. In the scope of the project, the decision tree was limited to allow for some processing. This resulted in an additional set of guidelines which could be used in much more specific cases, rather than the simple rulebook created initially.

To validate the decision-making process of the algorithm, the control output values were tested in the original simulation. The energy consumption could then be compared directly to the values achieved if only constant controls were applied. It was determined that using the optimal controls recommended could save 80 GJ in the 3-month test case, which was estimated to be around €2222.50, or the equivalent of €8890 annually. A value of €8890 is a rough estimation, as weather conditions will result in much less snow during summer months, which would reduce how much the system is used. This savings estimation was proven in just one test case, and further use and testing under different conditions could lead to even larger savings. This also proves the viability of such systems in applications outside of the one considered in this project, such as in the maintenance of public spaces, pavements and roads. Looking beyond the financial savings, this would provide significant reductions in the downtime of such infrastructure as periods of heavy snowfall could be anticipated and the

system could be adapted for those conditions. This would also reduce the need for de-icing and gritting, which is more time consuming and less effective.

This report detailed the development of a refined computational model for the optimal control of an underground heating system. The success of this project can be assessed through the ability to achieve the aims and objectives previously defined in Section 1.2.

Firstly, a clear understanding of the prior work undertaken by the COMEA research team was achieved by allowing dedicated time and focus to allow familiarisation of the original heat and mass transfer. The first 2 weeks of the project became this dedicated time, where we developed our understanding of the initial work in tandem with a first draft of the literature review. This allowed for all relevant physics and thermodynamics to be accurately represented within the improved upon computational model.

Secondly, the refinement of the dynamical snow melting model, the heat-mass transfer interactions between different layers of soil, and the interactions between the snow surface and the outside atmospheric conditions were achieved through multiple additional considerations, such as the moisture transport within the soil. Additionally, accounting for the imbalance of the snow melt rate across the length of the pitch due to the water-propyleneglycol solution circulating through the ductwork and creating a temperature time delay further refined the model.

Furthermore, the objective to include a detailed comparison of the original model to the heated pavement paper's model was met through the comparison and incorporation of features such as the consideration of the effects of evaporation and condensation on the snowpack.

Improvements to the model accuracy were also accomplished by performing a sensitivity analysis on the model, through the inclusion of 3 distinct approaches. Firstly, through a parameter study, then a study on the impact of time-step size and finally by considering different numerical schemes.

The objective for simulating accurate weather conditions and therefore operational parameters for a specific time was achieved through LHS methods.

Finally, machine learning methods were used to meet the final 2 defined objectives to establish frameworks for the optimal control of the system in different weather conditions and for establishing a computational methodology for optimal operation of the heating system for a linearised weather forecast 1 week in advance. Regression decision trees allowed for simplified or "Rule of Thumb" guidance document on the control of the system as it will be controlled by non-specialist human operators to be written.

All objectives outlined within this report have been fully discussed and the work undertaken to achieve these within this project have been described in detail throughout.

To conclude, this report outlined the work and findings of the Aero-Mechanical Engineering 5th Year Masters Group Project titled “Model-Based Optimal Control of An Underground Heating System”, which improved upon the modelling work introduced in the 2023 Computational Engineering and Analysis paper, “A computational model for underground heating control of outdoor sports fields in winter conditions”, [1].

6.0 References

[1] E. P. Immonen, Tommi; Ardaneh, Fatemeh; Chaudhari, Ashvinkumar, "A Computational Model For Underground Heating Control of Outdoor Sports Fields in Winter Conditions," 2023.

[2] X. Liu, S. J. Rees, and J. D. Spitler, "Modeling snow melting on heated pavement surfaces. Part I: Model development," *Applied Thermal Engineering,* vol. 27, no. 5, pp. 1115-1124, 2007/04/01/ 2007, doi: https://doi.org/10.1016/j.applthermaleng.2006.06.017.

[3] "The History Of Undersoil Heating." Football Stadiums. https://www.football-stadiums.co.uk/articles/undersoil-heating-in-football/ (accessed 11/09/23, 2023).

[4] W. Morris. "Football pitch works and Premier League promotion: Key considerations." Walker Morris. https://www.walkermorris.co.uk/in-brief/football-pitch-works-and-premier-league-promotion/ (accessed 11/09/23, 2023).

[5] S. N. Cevik, Keitaro "Chasing the Sun and Catching the Wind: Energy Transition and Electricity Prices in Europe." [Online]. Available: file:///C:/Users/erinb/Downloads/wpiea2022220-print-pdf.pdf

[6] J. Good, V. Ugursal, and A. Fung, "Simulation strategy and sensitivity analysis of an in-floor radiant heating model," *IBPSA 2005 - International Building Performance Simulation Association 2005,* 01/01 2005.

[7] A. Ahmed, F. Conti, M. Schießl-Widera, and M. Goldbrunner, "CFD-Based Sensitivity-Analysis and Performance Investigation of a Hydronic Road-Heating System," *Energies*, vol. 16, no. 5*,* doi: 10.3390/en16052173.

[8] D. F. Vitaliano, "An Economic Assessment of the Social Costs of Highway Salting and the Efficiency of Substituting a New Deicing Material," *Journal of Policy Analysis and Management,* vol. 11, no. 3, pp. 397-418, 1992, doi: 10.2307/3325069.

[9] G. Zhou, M. Cui, J. Wan, and S. Zhang, "A Review on Snowmelt Models: Progress and Prospect," *Sustainability*, vol. 13, no. 20*,* doi: 10.3390/su132011485.

[10] M. Zukowski, "A NUMERICAL ANALYSIS OF HEAT AND MASS TRANSFER IN A ROOM WITH THE AIR-UNDERFLOOR HEATING," 08/11 2003.

[11] J. You, D. G. Tarboton, and C. H. Luce, "Modeling the snow surface temperature with a one-layer energy balance snowmelt model," *Hydrol. Earth Syst. Sci.,* vol. 18, no. 12, pp. 5061-5076, 2014, doi: 10.5194/hess-18-5061-2014.

[12] B. Adl-Zarrabi, R. Mirzanamadi, and J. Johnsson, "Hydronic Pavement Heating for Sustainable Ice-free Roads," *Transportation Research Procedia,* vol. 14, pp. 704-713, 2016/01/01/ 2016, doi: https://doi.org/10.1016/j.trpro.2016.05.336.

[13] S. J. Rees, J. D. Spitler, and X. Xia, "Transient analysis of snow-melting system performance," (in English), *ASHRAE Transactions,* vol. 108, p. 406, 2002

2023-07-26 2002. [Online]. Available: https://www.proquest.com/scholarly-journals/transient-analysis-snow-melting-system/docview/192534330/se-2?accountid=14116

https://suprimo.lib.strath.ac.uk/openurl/SU/SUVU01?genre=proceeding&atitle=Transient+analysis+of+snow-melting+system+performance&author=Rees%2C+Simon+J%3BSpitler%2C+Jeffrey+D%3BXia%2C+Xiao&volume=108&issue=&spage=406&date=2002-01-01&rft.btitle=&rft.jtitle=ASHRAE+Transactions&issn=0001-2505&isbn=&sid=ProQ%3Asciencejournals_.

[14] A. Malakooti *et al.*, "Design and Full-scale Implementation of the Largest Operational Electrically Conductive Concrete Heated Pavement System," *Construction and Building Materials,* vol. 255, p. 119229, 2020/09/20/ 2020, doi: https://doi.org/10.1016/j.conbuildmat.2020.119229.

[15] J. Wurm, S. Bachler, and F. Woittennek, "On delay partial differential and delay differential thermal models for variable pipe flow," *International Journal of Heat and Mass Transfer,* vol. 152, p. 119403, 2020/05/01/ 2020, doi: https://doi.org/10.1016/j.ijheatmasstransfer.2020.119403.

[16] L. Pagliarini, L. Cattani, M. Mameli, S. Filippeschi, and F. Bozzoli, "Heat transfer delay method for the fluid velocity evaluation in a multi-turn pulsating heat pipe," *International Journal of Thermofluids,* vol. 17, p. 100278, 2023/02/01/ 2023, doi: https://doi.org/10.1016/j.ijft.2022.100278.

[17] H. Liu, P. Maghoul, and H. M. Holländer, "Sensitivity analysis and optimum design of a hydronic snow melting system during snowfall," *Physics and Chemistry of the Earth, Parts A/B/C,* vol. 113, pp. 31-42, 2019/10/01/ 2019, doi: https://doi.org/10.1016/j.pce.2019.01.009.

[18] H. Fatahian, H. Taherian, H. Salarian, and E. Fatahian, "Thermal performance of a ground heat exchanger in a cold climate: a CFD study," *The European Physical Journal Plus,* vol. 136, no. 3, p. 265, 2021/03/01 2021, doi: 10.1140/epjp/s13360-021-01265-7.

[19] G. Díaz, M. Sen, and K. T. Yang, "Effect of delay in thermal systems with long ducts," *International Journal of Thermal Sciences,* vol. 43, no. 3, pp. 249-254, 2004/03/01/ 2004, doi: https://doi.org/10.1016/j.ijthermalsci.2003.07.004.

[20] J. Duquette, A. Rowe, and P. Wild, "Thermal performance of a steady state physical pipe model for simulating district heating grids with variable flow," *Applied Energy,* vol. 178, pp. 383-393, 2016/09/15/ 2016, doi: https://doi.org/10.1016/j.apenergy.2016.06.092.

[21] H. Janssen, J. Carmeliet, and H. Hens, "The influence of soil moisture transfer on building heat loss via the ground," *Building and Environment,* vol. 39, no. 7, pp. 825-836, 2004/07/01/ 2004, doi: https://doi.org/10.1016/j.buildenv.2004.01.004.

[22] A. S. Association, *A guide to the use of iron blast furnace slag in cement and concrete*: Australasian Slag Association, 1990, p. 37. [Online]. Available: https://www.asa-inc.org.au/uploads/default/files/asa_guide_to_the_use_of_iron_blast_furnace_slag_in_cement_and_concrete.pdf. Accessed on: 18th of September 2023.

[23] O. K. B.C., "Geo-engineering properties of granulated blast furnace slag," in *International Conference on Geotechnical Engineering*, Tunis, Tunisia, 2008, no. 24th–26th March, pp. pp249 - 257. [Online]. Available:

http://www.tara.tcd.ie/bitstream/handle/2262/67138/2008_Geo-engineering%20properties%20of%20granulated%20blast%20furnace%20slag.pdf?sequence=1&isAllowed=y. [Online]. Available: http://www.tara.tcd.ie/bitstream/handle/2262/67138/2008_Geo-engineering%20properties%20of%20granulated%20blast%20furnace%20slag.pdf?sequence=1&isAllowed=y

[24] X. G. Liang Xue, Hao Chen, "Fluid Flow in Porous Media," 2020, ch. 2.

[25] H. J. Steeman, M. Van Belleghem, A. Janssens, and M. De Paepe, "Coupled simulation of heat and moisture transport in air and porous materials for the assessment of moisture related damage," *Building and Environment,* vol. 44, no. 10, pp. 2176-2184, 2009/10/01/ 2009, doi: https://doi.org/10.1016/j.buildenv.2009.03.016.

[26] M. Van Belleghem, H.-J. Steeman, M. Steeman, A. Janssens, and M. De Paepe, "Sensitivity analysis of CFD coupled non-isothermal heat and moisture modelling," *Building and Environment,* vol. 45, no. 11, pp. 2485-2496, 2010/11/01/ 2010, doi: https://doi.org/10.1016/j.buildenv.2010.05.011.

[27] A. Atangana, "Chapter 2 - Principle of Groundwater Flow," in *Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology*, A. Atangana Ed.: Academic Press, 2018, pp. 15-47.

[28] M. McKay and W. Conover, "RJ Beckman A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics,* vol. 21, pp. 239-245, 1979.

[29] Y. Hung, "Optimal Experiment Design, Latin Hypercube," in *Encyclopedia of Systems Biology*, W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota Eds. New York, NY: Springer New York, 2013, pp. 1583-1585.

[30] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," *IEEE Access,* vol. 8, pp. 108952-108971, 2020, doi: 10.1109/ACCESS.2020.2998358.

[31] M. Batty, "Digital twins," *Environment and Planning B: Urban Analytics and City Science,* vol. 45, no. 5, pp. 817-820, 2018/09/01 2018, doi: 10.1177/2399808318796416.

[32] N. Crespi, A. T. Drobot, and R. Minerva, "The Digital Twin: What and Why?," in *The Digital Twin*, N. Crespi, A. T. Drobot, and R. Minerva Eds. Cham: Springer International Publishing, 2023, pp. 3-20.

[33] M. Company. "What is digital-twin technology?" McKinsey & Company. https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-digital-twin-technology (accessed December, 2023).

[34] Claudia Muñiz, Wenfeng Hu, and Y. N. Molina. "Digital twins: what are they and how are they enabling future networks?" Ericsson. https://www.ericsson.com/en/blog/2022/3/what-are-digital-twins-three-real-world-examples (accessed December, 2023).

[35] C. H. dos Santos and J. A. B. Montevechi, "Digital Twins Architecture," in *Digital Twins: Basics and Applications*, Z. Lv and E. Fersman Eds. Cham: Springer International Publishing, 2022, pp. 1-12.

[36] C. Ntakolia, A. Anagnostis, S. Moustakidis, and N. Karcanias, "Machine learning applied on the district heating and cooling sector: a

review," *Energy Systems,* vol. 13, no. 1, pp. 1-30, 2022/02/01 2022, doi: 10.1007/s12667-020-00405-9.

[37] "sklearn.model_selection.train_test_split¶." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (accessed 14/12/2023, 2023).

[38] s.-l. developers. "1.1. Linear Models." scikit-learn. https://scikit-learn.org/stable/modules/linear_model.html (accessed 11th of September, 2023).

[39] Y. Aldossary, S. Alhaddad, M. Ebrahim, and A. M. Zeki, "Comparing K-Nearest Neighbors, Random Forest and Naïve Bayes Models to Classify Fetal Health Using Resampling Methods," in *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, 25-26 Oct. 2022 2022, pp. 250-254, doi: 10.1109/ICDABI56818.2022.10041489.

[40] s.-l. developers. "1.4. Support Vector Machines." scikit-learn. https://scikit-learn.org/stable/modules/svm.html (accessed 11th of September, 2023).

[41] Viswa. "Support Vector Regression: Unleashing the Power of Non-Linear Predictive Modeling." Medium. https://medium.com/@vk.viswa/support-vector-regression-unleashing-the-power-of-non-linear-predictive-modeling-d4495836884 (accessed 2023).

[42] s.-l. developers. "1.11. Ensemble methods." scikit-learn. https://scikit-learn.org/stable/modules/ensemble.html (accessed 11th of September, 2023).

[43] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, Berlin, Heidelberg, 2000// 2000: Springer Berlin Heidelberg, pp. 1-15.

[44] AnkanDas22. "Python | Decision Tree Regression using sklearn." GeeksForGeeks. https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/ (accessed 13, 2023).

[45] A. Cook. "Scaling and Normalization." https://www.kaggle.com/code/alexisbcook/scaling-and-normalization (accessed 14/12/2023, 2023).

[46] L. A. Jimenez-Roa, T. Heskes, and M. Stoelinga, "Fault Trees, Decision Trees, And Binary Decision Diagrams: A Systematic Comparison," *arXiv preprint arXiv:2310.04448,* 2023.

[47] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics,* vol. 18, no. 6, pp. 275-285, 2004/06/01 2004, doi: https://doi.org/10.1002/cem.873.

[48] D. Varghese. "Comparative Study on Classic Machine learning Algorithms." https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222 (accessed 14/12/2023, 2023).

[49] "sklearn.tree.DecisionTreeRegressor." https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html (accessed 14/12/2023, 2023).

[50] P. E. Bryan P. Strohman. "Drainage of Artificial Turf Systems." https://cdn.ymaws.com/sportsbuilders.org/resource/resmgr/tm_presentations/2019/5c_%E2%80%93_drainage_of_artificial_.pdf (accessed 18th of September, 2023).

[51] C. H. Bruand A, G. Lesturgez, "Physical properties of tropical sandy soils: A large range of behaviours," presented at the Management of Tropical Sandy Soils for Sustainable Agriculture, Khon Kaen, Thailand, 2005. [Online]. Available: https://horizon.documentation.ird.fr/exl-doc/pleins_textes/divers16-03/010066548.pdf.

[52] D. M. Gray, B. Toth, L. Zhao, J. W. Pomeroy, and R. J. Granger, "Estimating areal snowmelt infiltration into frozen soils," *Hydrological Processes,* vol. 15, no. 16, pp. 3095-3111, 2001/11/01 2001, doi: https://doi.org/10.1002/hyp.320.

[53] MathWorks. "Simulating Systems with Variable Transport Delay Phenomena." MathWorks. https://uk.mathworks.com/help/simulink/slref/simulating-systems-with-variable-transport-delay-phenomena.html#d126e746625 (accessed September, 2023).

[54] V. Bansal, R. Misra, G. D. Agrawal, and J. Mathur, "Performance analysis of earth–pipe–air heat exchanger for winter heating," *Energy and Buildings,* vol. 41, no. 11, pp. 1151-1154, 2009/11/01/ 2009, doi: https://doi.org/10.1016/j.enbuild.2009.05.010.

[55] J. B. Kitto, S. C. Stultz, Babcock, and C. Wilcox, *Steam, its generation and use*, 41st . ed. (Steam 41). Barberton, Ohio: Barberton, Ohio : Babcock & Wilcox, 2005.

[56] "180° Elbow Long and Short Radius." Wellgrow Industries Corp. https://www.pipefittingweb.com/images/product/04_but-tweld-fittings/pdf/180-elbow-long-radius.pdf (accessed 2023).

[57] "180 Degree Bend Diagram." Tulsa Tube Bending Company. https://ttb.com/recent-news/diagrams/ (accessed 2023).

[58] "Heat Transfer Coefficients in Heat Exchanger Surface Combinations." The Engineering Toolbox. https://www.engineeringtoolbox.com/overall-heat-transfer-coefficients-d_284.html (accessed 2023).

[59] L. Kang, X. Zhou, T. van Hooff, B. Blocken, and M. Gu, "CFD simulation of snow transport over flat, uniformly rough, open terrain: Impact of physical and computational parameters," *Journal of Wind Engineering and Industrial Aerodynamics,* vol. 177, pp. 213-226, 2018/06/01/ 2018, doi: https://doi.org/10.1016/j.jweia.2018.04.014.

[60] X. Zhou, L. Kang, M. Gu, L. Qiu, and J. Hu, "Numerical simulation and wind tunnel test for redistribution of snow on a flat roof," *Journal of Wind Engineering and Industrial Aerodynamics,* vol. 153, pp. 92-105, 2016/06/01/ 2016, doi: https://doi.org/10.1016/j.jweia.2016.03.008.

[61] X. Zhou, L. Kang, X. Yuan, and M. Gu, "Wind tunnel test of snow redistribution on flat roofs," *Cold Regions Science and Technology,* vol. 127, pp. 49-56, 2016/07/01/ 2016, doi: https://doi.org/10.1016/j.coldregions.2016.04.006.

[62] "Download observations." Finnish Meteorological Institute. https://en.ilmatieteenlaitos.fi/download-observations (accessed 11/09/23, 2023).

[63] A. K. Bhardwaj, D. Goldstein, A. Azenkot, and G. J. Levy, "Irrigation with treated wastewater under different irrigation methods: Effects on hydraulic conductivity of a clay soil," *Geoderma,* vol. 140, no. 1, pp. 199-206, 2007/06/15/ 2007, doi: https://doi.org/10.1016/j.geoderma.2007.04.003.

[64] P. S. S. eLibraryPRO. Soils - Part 2: Physical Properties of Soil and Soil Water [Online] Available: http://passel-test.unl.edu/beta/pages/informationmodule.php?idinformationmodule=1130447039&topicorder=10&maxto=10&minto=1

[65] A. D. W. Nuijten and K. V. Høyland, "Modelling the thermal conductivity of a melting snow layer on a heated pavement," *Cold Regions Science and Technology,* vol. 140, pp. 20-29, 2017/08/01/ 2017, doi: https://doi.org/10.1016/j.coldregions.2017.04.008.

[66] M. S. Roxy, V. B. Sumithranand, and G. Renuka, "Estimation of soil moisture and its effect on soil thermal characteristics at Astronomical Observatory, Thiruvananthapuram, south Kerala," *Journal of Earth System Science,* vol. 123, no. 8, pp. 1793-1807, 2014/12/01 2014, doi: 10.1007/s12040-014-0509-x.

[67] J. T. F. Wong, K. L. Chow, X. W. Chen, C. W. W. Ng, and M. H. Wong, "Effects of biochar on soil water retention curves of compacted clay during wetting and drying," *Biochar,* vol. 4, no. 1, p. 4, 2022/01/20 2022, doi: 10.1007/s42773-021-00125-y.

[68] L. Zhao and D. M. Gray, "Estimating snowmelt infiltration into frozen soils," *Hydrological Processes,* vol. 13, no. 12-13, pp. 1827-1842, 1999/09/01 1999, doi: https://doi.org/10.1002/(SICI)1099-1085(199909)13:12/13<1827::AID-HYP896>3.0.CO;2-D.

[69] V. P. SINGH and C.-Y. XU, "Sensitivity of mass transfer-based evaporation equations to errors in daily and monthly input data," *Hydrological Processes,* vol. 11, no. 11, pp. 1465-1473, 1997, doi: https://doi.org/10.1002/(SICI)1099-1085(199709)11:11<1465::AID-HYP452>3.0.CO;2-X.

[70] J. G. EARL HARBECK, "A Practical Field Technique For Measuring Reservoir Evaporation Utilizing Mass-Transfer Theory," 1962. [Online]. Available: https://pubs.usgs.gov/pp/0272e/report.pdf.

[71] Z. Wang, H. Chen, and S. Weng, "“Partial pressures” of humid air in wide pressure and temperature ranges," *Frontiers in Energy,* vol. 7, no. 4, pp. 511-517, 2013/12/01 2013, doi: 10.1007/s11708-013-0281-7.

[72] Y. Nawaz, M. S. Arif, and W. Shatanawi, "A New Fourth-Order Predictor-Corrector Numerical Scheme for Heat Transfer by Darcy-Forchheimer Flow of Micropolar Fluid with Homogeneous-Heterogeneous Reactions," *Applied Sciences*, vol. 12, no. 12*,* doi: 10.3390/app12126072.

[73] "Finnish Meteorlogical Institute." https://en.ilmatieteenlaitos.fi/ (accessed 07/12, 2023).

[74] "District Heating Prices." Helen Ltd. https://www.helen.fi/en/companies/heating-for-companies/district-heating-for-companies/prices (accessed 2023).

[75] M. Khoshvaght-Aliabadi, H. Shabanpour, A. Alizadeh, and O. Sartipzadeh, "Experimental assessment of different inserts inside straight tubes: Nanofluid as working media," *Chemical Engineering and Processing: Process Intensification,* vol. 97, pp. 1-11, 2015/11/01/ 2015, doi: https://doi.org/10.1016/j.cep.2015.08.009.

7.0 Appendices

7.1 Appendix A1 – Reflective Report

Introduction

This reflective report outlines the overview of the project and team structure. It includes details on the original objectives and of the teams working plan. Finally, this report includes personal summaries of the reflections each group member made throughout their exchange experience.

Project Overview and Team Structure

To allow for the greatest success with the group project early within the process the team agreed on a coordinated working structure which allowed for the group to be successful and make a genuine contribution to Engineering.

Communications began in the early weeks of the project through discussions with the project supervisor, Eero Immonen and the research assistants, Fatemeh Ardaneh, Ashvin Chaudhari and Sajad Shahsavari who all work within the Computational Engineering and Analysis (COMEA) research group at Turku University of Applied Sciences.

The project titled “Model-Based Optimal Control of An Underground Heating System” was proposed by Eero and develops work previously published by COMEA in early 2023. Expectations and objectives were established and agreed between the group and the supervisor. An initial plan was developed in the form of a Gantt Chart and Statement of Purpose which was submitted to Strathclyde for final approval.

Throughout the project each member of the group contributed and collaborated with enthusiasm throughout the project. All group members agreed on a 3-way split with no one member taking charge, however Malcolm Irving-Robertson formally took on the role of project lead. It was his responsibility for communicating with the project supervisors via email and arranging meeting times and locations that suited every party, however all team members actively contributed within bi-weekly supervisor meetings.

Objectives and Deliverables

In early September the group created an initial Gantt Chart which outlined a plan for a rough timeframe to meet project objectives and deliverables. The Gantt Chart included specific references to all the tasks required to complete the objectives and deliverables shown in Appendix A2 – Gantt Chart. The timeframe proposed within the initial Gantt Chart was not set in stone and changed as the project developed. A formal meeting the group held every Monday at 9 a.m. acted as an opportunity to discuss the plan for the upcoming week, update the Gantt Chart following the progress of the previous weeks and discuss any other issues which may present themselves.

A deliverable required from Turku AMK was to produce general guidance documents for the COMEA team to allow them to continue our work and access our files once we return to Scotland. These guidance documents can be found in Appendix H – General Guidance to Project Code and Appendix I – Simulation Full Worked Example.

Changes to Timeframe

As the project progressed plans for the timeframe, objectives and final deliverables shifted. This following section describes some key changed the group faced during the Semester.

Firstly, after receiving feedback from the Project Supervisor on the Statement of Purpose, the group shifted focus from their initial delegated tasks to work on reviewing the project plan to be as efficient as possible. This updated plan accommodated for the possibility of delays with different aspects of the timeframe and the setbacks which could occur. Although this took longer than initially expected to plan out, the group agreed it was beneficial to allow for the project to go as smoothly as possible.

Another significant change to the project plan came when refining the computational heat and mass balance model. The time frame initially proposed was not an accurate estimation for how long the tasks took, particularly with researching the effect of moisture transportation. Although slightly disheartening to the team to be behind schedule the proposed timeframe allowed for tasks to be pushed back whilst still managing all the key deliverables. Tasks such as the sensitivity analyses and machine learning however took less time that originally set out. This allowed the team sufficient time from early December to organise the results, write the report and complete any remaining tasks.

Time Management and Working Plan

The group agreed on a plan and timeframe for the project during the first week and discussed expectations. It was agreed that the best way to run the project was as a Monday to Friday, 9 a.m. to 5 p.m. This was a good way to allow for a healthy work-life balance as the team is taking at least one day away from the project each week to have a break from the work.

The group work took place at least 4 days a week in person on the university campus and the other day was done working from home. This change of scenery and more relaxed environment allowed for good progress to be made continuously throughout. Although the group set out these working hours there was flexibility and trust within the team, which meant that an individual could adapt their hours to suit their schedule. This worked very well during times when the group were at weekday events, had family visiting or were travelling to visit new cities.

Part of the agreed upon plan required that a record of all work was kept within a collaborative logbook which got updated daily. This acted as a form of communication keeping a dated record of what everyone had been working on and achieved on specific days. The group also made great use of

different collaborative cloud based working software tools such as OneDrive, for any Word Documents, Excels or PowerPoints created, and GitHub saving all Python code, results and ANSYS projects.

The group attended supervisor meetings which took place bi-weekly on Tuesdays from 12-2pm and allowed for the group to delve into the technical aspects of the project, report on findings and seek advice on any problems encountered. This meeting was the primary opportunity to discuss anything project related with Eero and the COMEA team. However, additionally, communication was possible outside of these formal meetings, via email or Microsoft Teams.

Personal Reflections Overview

Throughout the entirety of the project each of the three group members kept a personal reflective log which included details and reflections of all aspects of the project including progress made, difficulties encountered and experience working as a team. The sections below include these reflections.

Overall, the group worked together extremely successfully, and the positive results and progress achieved has reflected on this directly. All group members continuously contributed to the project work, aiming to bring the same level of enthusiasm each day. We were successful in cultivating an environment that boosted individual morale on days when group members may have felt discouraged. There were no cases of an individual feeling as though not everyone is pulling their own weight, and the equal 3-way distribution of tasks has made working on the project an enjoyable experience.

The group consisted of 3 peers who prior to exchange only knew one another from university lectures but quickly became great friends. This was very beneficial to the group dynamic as it allowed for open communications to happen to address the successes or challenges which may come up. The strong communication within the group allowed for easy adaptability and flexibility with tasks and timeframes.

When difficulties arose throughout this project, such as with the refinement of the snow melting model, individual team members did not lose any motivation despite the challenges faced being bigger than expected. All team members were willing to help with all problems throughout despite them not being specifically assigned tasks to them. On reflection this was an extremely successful mindset to have with any group project as the entire group worked as a team and supported each other in whatever way they could.

The entirety of the group project and study abroad experience was an excellent learning experience for all. All group members have previous experience working in teams as part of their time at university, however nothing compared to the scale of this project.

On top of the significant challenge of producing a piece of academic work to the scale of this report there were also many external factors which influenced the members of the group throughout the process. Most

significantly were the effects of moving abroad to a new country. Difficulties with culture shock and adapting to new lifestyles came hand in hand as the work developed.

The group understands the importance of having a healthy work life balance which allowed for all to make the most out of their entire exchange and throw themselves headfirst into all experiences that were given to them.

Erin's Personal Reflections

My experience working as part of a group within this project has been extremely positive overall and very beneficial to my future professional experience as a graduate engineer. The skills I have gained and the lessons I have learned throughout the entirety of my exchange will stay with me far beyond my time at university.

Working on a group project of this scale has allowed me to develop my communication skills within a team environment. Working daily for over 4 months with a small group of peers has made me realise how vital it is to develop strong relationships with one another in order for open and healthy communications to take place. I believe that our group worked very well at discussing all aspects of the project from planning what our objectives and timeframe were to developing our technical content and progressing with the formal writeups. We openly discussed and debated the highs and lows of our projects process in a healthy and professional manner, allowing every member of the group to have their voice heard and actively listening to what everyone had to say.

My experience working on this project has allowed me to massively improve my time management and organisational skills. Prior to this experience I often had difficulties with separating my work life balance. In previous years I was guilty of working minimal hours some days and then working excessively others, this would often result in a burnout and resentment to whatever task I was working towards. However, the effective way our group has planned our schedule this past semester has allowed for clear goals and expectations to be achieved within a reasonable timeframe. The way in which we have structured our time throughout this project and the techniques we used to do so such as developing and regularly updating the Gantt chart and agreeing on a weekly working schedule which suits all team members, are techniques which will stay with me beyond this project.

My programming experience prior to starting this project was minimal. Beyond what basic MATLAB was included as part of my first- and second-year classes I would say I had practically no experience or confidence with anything of the sorts. This however is not the case anymore as my experience working for months developing and improving the computational model on python has reinforced the foundational skills which I had and allowed me to develop a greater understanding for Python and similar programming languages. I am now far more confident and competent when doing any such task which requires an understanding of Python, so much so that I have amended my curriculum for the second semester to take classes

such as 16565: Engineering Composites which require the student to be able to work on any programming language. Although exclusively working on Python this semester the skills are transferable.

This project has introduced me to new areas of interest in the form on machine learning and decision trees. An area of engineering which I had no previous experience with I have learned throughout this project the basis of it, and I am keen to continue to expand my interest beyond the scope of this project.

This project and the entirety of my exchange experience has forced me to step out with my comfort zone and form professional relationships with people from a range of countries. Our bi-weekly meetings with the COMEA team were an excellent opportunity to discuss a wide variety of questions, concerns, and comments about our project with specialist researchers.

Our group also took part in a student and company brokerage event at Turku University of Applied Science in mid-November. This event consisted of a series of lectures and seminars with the general themes of sports and exercise. Both Strathclyde groups on exchange in Turku have projects related to these themes, our group with underfloor heating of a football pitch and the other team with a project about optimising the perfect race for a runner and an eRallycross car. Our group set up a stand, created posters and a PowerPoint and had the opportunity to talk with the participants of the event during their coffee break Figure 74 and Figure 75 is an example of one of the posters we produced and our group at the event. The audience came from a range of backgrounds with varying technical knowledge, so event was an excellent opportunity to talk about our project in several different ways allowing our audience to have a clear understanding of the work we have done and our final goals for the end of our project.

Figure 74: Poster from Turku AMK Event

Figure 75: Group Presentation at Turku AMK Event

Although my overall experience working in this group was positive and my entire exchange is one which I will cherish forever, it is important to address both the ups and downs and determine what could be improved for similar ventures in the future. Working in any group comes with it owns challenges and our group was no exception. There were occasions which each of us within the group would lose motivation and be disheartened when the task we were working on would not progress the way we were hoping. These feelings could last for several days at a time but thankfully not any longer due to the encouraging nature of our group. It is important to stay positive when difficulties arise with academic work and remember that you will get there in the end.

Towards the beginning of this project our group incorrectly estimated the time some tasks would take such as with the refining of the snow melting model. We were advised from our academic supervisor to allow ourselves more time to complete said tasks, but even with the additional time given we were still behind schedule. This was not an issue with completing the entire project on time as we and our supervisor were confident that all would work out however some group members became obsessed with if we were on track or not which would stress everyone out far more than needed to be. I tried to stay optimistic throughout and encouraging for all team members. The challenge of working on this project has taught me so much about time management and meeting deadlines. It is important to realistically plan the timeframe of tasks and to understand that setbacks happen but not to be unmotivated by them. This is an important lesson which I have learned, and I will keep with me throughout the remainder of my university experience and throughout my professional career.

Kaloyan's Personal Reflections

Working on this project over the course of my exchange period has been an incredibly delightful experience. Our group was able to quickly settle into a dynamic where discussions could easily take place about the work and the progression of different aspects of the project, and everyone was able to contribute equally to the completion of the various objectives.

The decisions that we made at the beginning of the semester with regards to how work should take place, mostly collaborative on campus, really contributed to creating an encouraging environment and all members of the group were always readily available to discuss and help with any issues faced in individual tasks. This helped me develop my collaborative and team working skills greatly, as working on the project full-time, and experiencing all the positives and negatives that come with that, are common occurrences in the work of any graduate student. Learning how to effectively manage those relationships is an important skill which is transferrable to any future undertaking.

This was further underlined in our work with our supervisor, and the COMEA research team. From the very beginning, they were incredibly active and provided us with a lot of support in understanding the work that they had already carried out, and what the objectives were for the project that they had set out for us. They were also extremely helpful with any technical challenges we faced and were keen on understanding the new developments we brought to the existing code. Building that relationship continuously was something that I felt required me to learn to effectively communicate these issues to someone who might not be directly involved with the project but had an area of expertise which was relevant to the task that I was working on.

Moreover, the nature of the project allowed me to massively improve my skills and confidence working with programming languages, an important skill which is transferable to many fields of engineering. Python was the environment we utilised during the project, on the recommendation of our supervisor, and only one member of our group had any previous experience working outside of MATLAB. This meant that we had to quickly learn to adapt to a new coding environment so that we could tackle the technical aspects of the project effectively. Fortunately, these types of programming languages share many similarities so the foundational knowledge of how to operate them was relatively easy to establish for Python. The work also went beyond anything I had previously done, stretching into areas I found inaccessible before starting this project, such as optimisation and machine learning. Learning the complicated inner workings of these processes has allowed me to gain invaluable experience which could be implemented in projects well beyond my time at university.

Another big takeaway from this experience has been learning to better manage my own workload. Maintaining a good balance between work and finding the time to rest and do other things I enjoy has been something I have struggled with previously. I found this to be the case again at some points over the course of the semester, leading to days where I felt I was contributing very little to the task I was working on. This could sometimes be

exacerbated by the fact that a big part of research projects in general is that they do not have a clear path or an intended result. Exploring different avenues is part of the process and progress is being made even when it appears that an unsuitable result has been reached. As the semester progressed, I found it easier to set more realistic expectations for myself, in large part through collaboration and discussion with the other members of my group. The way in which we organised our work accommodated for those issues and still allowed us to stay on track to finish the project on time and has been an important learning experience.

The close relationships we were able to build within our group were very important to the success of the project. We had the opportunity to attend events outside of our work at university together, allowing us spend time with each other in a less professional setting while also learning more about Finnish culture. Getting to know the people we were working with on a more personal level made communication easier and helped significantly in maintaining a good work-life balance, avoiding the pitfalls of constantly focusing on which tasks remained to be done.

In addition to the Student Brokerage event, we were given an additional opportunity to present our work at the Capstone competition taking place on the university campus. For this, we developed a 3-minute PowerPoint presentation which detailed the technical work we completed over the course of the project and presented in front of a panel of staff members and other parties interested in the projects. While our projects were not allowed to participate in the competition due to the level of technical complexity involved, this was invaluable experience in learning to adapt our approach to communicating the content of our project to people of different academic backgrounds and for networking, allowing us to learn more about the various projects that Turku University was undertaking in cooperation with industry partners.

Malcolm's Personal Reflections

I have had a thoroughly enjoyable time working on this project with my other group members. Our communication quality was consistently high, and we did not have any issues with lack of input or an uneven workload. I was very impressed with their level of motivation and felt energised while collaborating as a group.

In the early weeks we faced an initial challenge after our Statement of Purpose review meeting with our supervisor. Once the feedback was received, we postponed our current delegated tasks and returned to redraft the statement of purpose. It was quickly altered to bring the supervisor comments on board and was a good example of the flexibility we aimed to maintain during the project. During supervisor meetings in future weeks, we were all engaged and motivated to ask important questions that would allow us to progress. We were also good at encouraging group members to request further help from supervisors if they needed it.

Being able to book rooms at the university was very helpful in providing a quiet space for us to focus and discuss key issues in our project. As a group have been good at supporting each other with individual difficulties. We have shown understanding and patience when explaining issues in our delegated tasks.

In terms of my own development, I feel that this project has been an excellent opportunity to build on my existing experience with Python. It has been enjoyable to discover how Python code can be harnessed in the field of engineering optimisation, and I feel that this experience will be very useful in my future career. Equally, the project was a chance to expand my knowledge of Ansys Fluent. Throughout my time at university, I had opportunities to use the software through projects, however I had never used the software to freely create a CFD study. I feel like I have gained a new level of understanding that will be very useful when using the software again in the workplace.

Before this semester, I had never participated in a research project, in which we were expected to continually update conclusions, and had an element of uncertainty on the results. I think this experience again translates well to the work that I will conduct in future, preparing me for the element of unpredictability that exists in many engineering positions.

I met my two group project partners for the first time at the beginning of this exchange, and since then I feel that we all become close friends. We get along very well with each other, and we will make sure to meet again next semester. The friendship formed with Erin and Kaloyan made my exchange experience all the more worthwhile and on top of working together on this project we also made sure to get the most out of our cultural exchange by attending events within Finland. We were even able to travel to new countries together, a picture of our group on a hike in Bergen, Norway is shown in Figure 76.

Figure 76: Group Turku 1 in Norway

As mentioned in the interim report, we aimed for a good level of flexibility, and I was very happy with how we managed this throughout the project. When parents visited or we were required to attend residence permit appointments, we were able to adjust timescales accordingly without disruption. Occasionally working from home to continue tasks was very beneficial, providing a change of atmosphere and renewed focus.

Reflecting on my time as project manager, I believe that the leadership experience I have gained has been invaluable, and it has provided me with areas to improve for future leadership roles. Firstly, I thoroughly enjoyed supporting the development of my group members, while they grew their skills in Python and in Machine Learning. I enjoyed helping to explain certain procedures, breaking down key points to aid understanding. I also enjoyed leading weekly progress meetings, as I found that in addition to making other group member's issues with subtasks known, it helped me to solidify my weekly goals for my own subtasks. When submitting documents such as the Interim Report, I felt proud of what the group had accomplished. For the next leadership opportunity, I would like to look at new ways to review progress and improve morale. Based on our reflections of project management and of the process as a whole, there were a few general points that we noted as areas we would improve if undertaking the project again. The table below lays out and ranks the potential benefit of techniques for future projects:

Table 16: Possible Future Actions

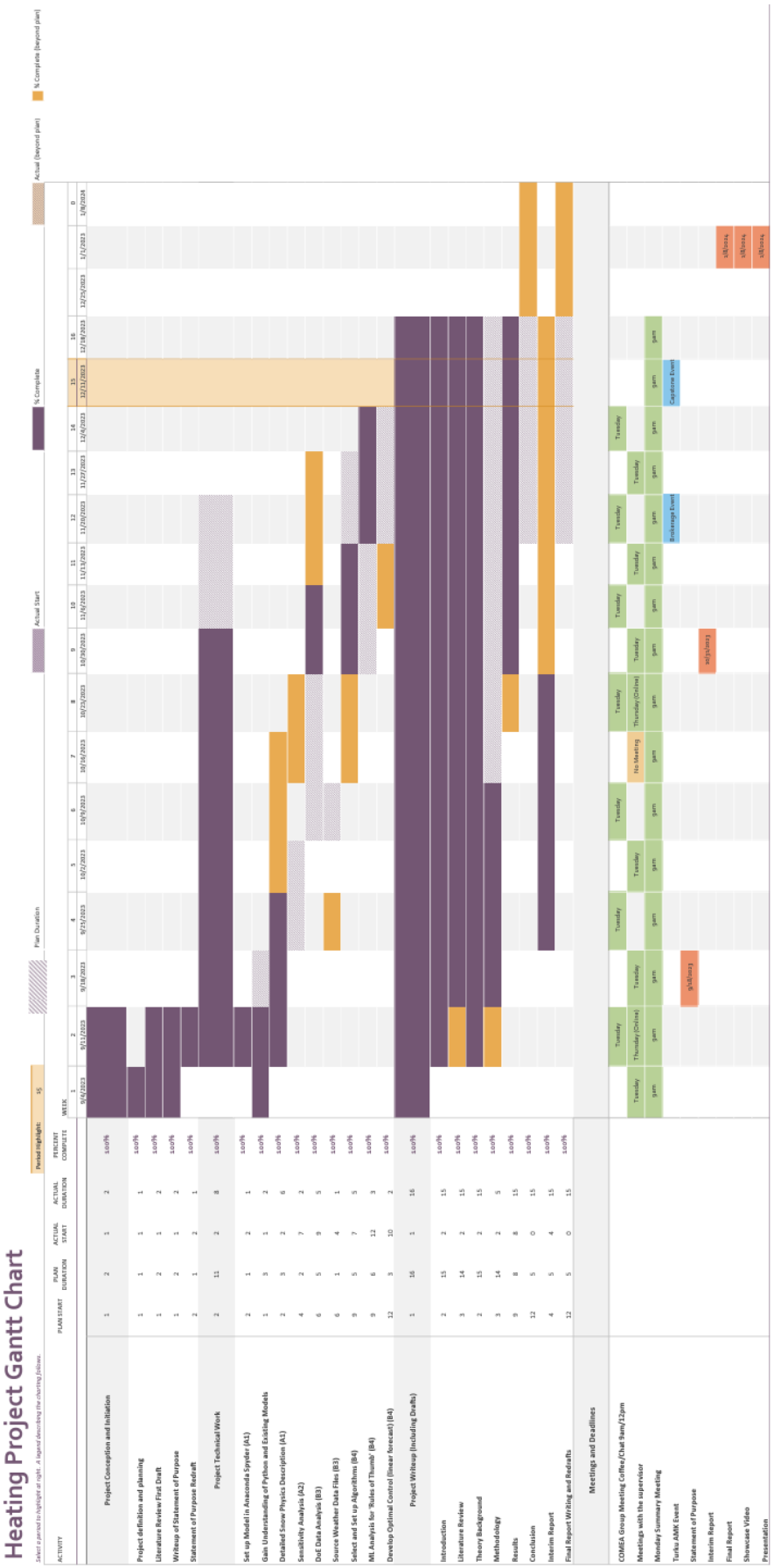
Technique	Description	Benefits	Potential Ranking
Data Graphic	Creating a data-focused graphic for each weekly meeting, or on a bi-weekly basis.	Group members would be able to see how far we have come, and what needs to be done to ensure success.	**Medium** (Possibility for increased morale, and a better understanding of project progress)
Work Style Experimentation	Designating time for group members to experiment with a range of different work styles.	Group members could discover a more comfortable or efficient way of completing a task.	**High** (Possibility for increased efficiency, motivation and work quality)
Risk Diagnosis	It would be interesting to develop a risk diagnosis system. This could be done manually with a diagram, or even in Python, with a simple algorithm to classify a problem.	We could use categories to group problems, and automatically reach relevant solutions. It would also allow us to rank issues and instantly know how to best prioritise it among other tasks.	**Medium** (Potential to save time and make risk prevention simpler)
Deepen knowledge of GitHub, OneDrive and other tools	If conducting the project again, we would aim to gain further experience in using collaborative software.	We would be able to use every feature available to us, and ensure maximum functionality.	**Low** (Existing knowledge is of a good level, but there may be room to discover time-saving techniques)

Turku University of Applied Sciences was a fantastic setting for the project, with brand new facilities for working. The location was very comfortable, and I think it has helped us to put forward our best work. I am very grateful to have been a part of this exchange.

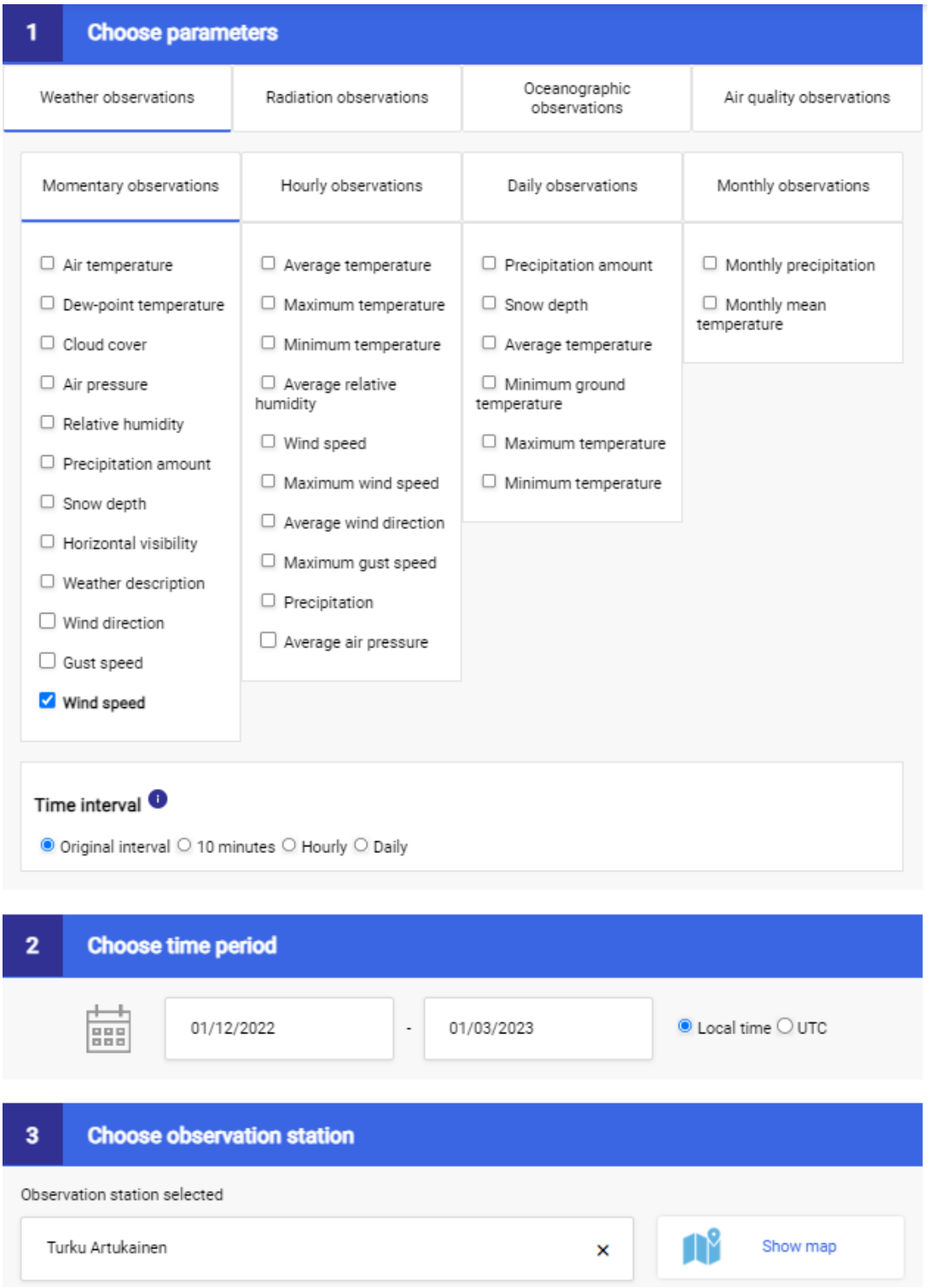
Conclusions

This reflective report summarises the entirety of this group's experience on exchange at Turku University of Applied Science. Details of project management and progress are discussed and reflected upon. Each group member reported on their personal experience of the project and exchange. Overall, it can be concluded that all members of the team reflect very positively on their time in Finland and working together on this group project.

7.2 Appendix A2 – Gantt Chart



7.3 Appendix B – Average Wind Speed



7.4 Appendix C – URL for Finnish Meteorological Institute Website (Download Observations)

https://en.ilmatieteenlaitos.fi/download-observations

7.5 Appendix D – Graphviz Data for Optimal Control Decision Tree

```
digraph Tree {
node [shape=box, fontname="helvetica"] ;
edge [fontname="helvetica"] ;
0 [label="Air Temperature <= 0.274\nsquared_error = 0.24\nsamples =
2400\nvalue = [[0.402]\n[0.605]]"] ;
1 [label="Liquid Flow Rate <= 0.393\nsquared_error = 0.165\nsamples
= 1558\nvalue = [[0.101]\n[0.6]]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="Precipitation Amount <= 0.018\nsquared_error =
0.079\nsamples = 629\nvalue = [[0.105]\n[0.93]]"] ;
1 -> 2 ;
3 [label="Cloud Amount <= 2.357\nsquared_error = 0.049\nsamples =
9\nvalue = [[1.0]\n[0.889]]"] ;
2 -> 3 ;
4 [label="squared_error = 0.0\nsamples = 1\nvalue = [[1.0]\n[0.0]]"]
;
3 -> 4 ;
5 [label="squared_error = 0.0\nsamples = 8\nvalue = [[1.0]\n[1.0]]"]
;
3 -> 5 ;
6 [label="Delta T <= 16.131\nsquared_error = 0.074\nsamples =
620\nvalue = [[0.092]\n[0.931]]"] ;
2 -> 6 ;
7 [label="Wind Speed <= 7.714\nsquared_error = 0.049\nsamples =
500\nvalue = [[0.09]\n[0.984]]"] ;
6 -> 7 ;
8 [label="Air Temperature <= -0.119\nsquared_error = 0.029\nsamples
= 428\nvalue = [[0.042]\n[0.981]]"] ;
7 -> 8 ;
9 [label="Liquid Flow Rate <= 0.006\nsquared_error = 0.019\nsamples
= 411\nvalue = [[0.022]\n[0.983]]"] ;
8 -> 9 ;
10 [label="Delta T <= 9.94\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.667]\n[1.0]]"] ;
9 -> 10 ;
11 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
10 -> 11 ;
12 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
10 -> 12 ;
13 [label="Delta T <= 13.81\nsquared_error = 0.017\nsamples =
408\nvalue = [[0.017]\n[0.983]]"] ;
9 -> 13 ;
14 [label="Wind Speed <= 6.268\nsquared_error = 0.01\nsamples =
345\nvalue = [[0.02]\n[1.0]]"] ;
13 -> 14 ;
15 [label="Air Temperature <= -0.25\nsquared_error = 0.002\nsamples
= 283\nvalue = [[0.004]\n[1.0]]"] ;
14 -> 15 ;
16 [label="squared_error = 0.0\nsamples = 281\nvalue =
[[0.0]\n[1.0]]"] ;
15 -> 16 ;
17 [label="Snow Depth <= 9.363\nsquared_error = 0.125\nsamples =
2\nvalue = [[0.5]\n[1.0]]"] ;
15 -> 17 ;
18 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
17 -> 18 ;
19 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
```

```
17 -> 19 ;
20 [label="Air Temperature <= -13.149\nsquared_error =
0.044\nsamples = 62\nvalue = [[0.097]\n[1.0]]"] ;
14 -> 20 ;
21 [label="Precipitation Intensity <= 5.5\nsquared_error =
0.111\nsamples = 3\nvalue = [[0.667]\n[1.0]]"] ;
20 -> 21 ;
22 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
21 -> 22 ;
23 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
21 -> 23 ;
24 [label="Delta T <= 0.238\nsquared_error = 0.032\nsamples =
59\nvalue = [[0.068]\n[1.0]]"] ;
20 -> 24 ;
25 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
24 -> 25 ;
26 [label="Precipitation Amount <= 1.17\nsquared_error =
0.025\nsamples = 58\nvalue = [[0.052]\n[1.0]]"] ;
24 -> 26 ;
27 [label="Snow Depth <= 10.524\nsquared_error = 0.064\nsamples =
20\nvalue = [[0.15]\n[1.0]]"] ;
26 -> 27 ;
28 [label="Liquid Flow Rate <= 0.062\nsquared_error = 0.125\nsamples
= 6\nvalue = [[0.5]\n[1.0]]"] ;
27 -> 28 ;
29 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
28 -> 29 ;
30 [label="Delta T <= 11.905\nsquared_error = 0.094\nsamples =
4\nvalue = [[0.75]\n[1.0]]"] ;
28 -> 30 ;
31 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
30 -> 31 ;
32 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
30 -> 32 ;
33 [label="squared_error = 0.0\nsamples = 14\nvalue =
[[0.0]\n[1.0]]"] ;
27 -> 33 ;
34 [label="squared_error = 0.0\nsamples = 38\nvalue =
[[0.0]\n[1.0]]"] ;
26 -> 34 ;
35 [label="Liquid Flow Rate <= 0.318\nsquared_error = 0.049\nsamples
= 63\nvalue = [[0.0]\n[0.889]]"] ;
13 -> 35 ;
36 [label="squared_error = 0.0\nsamples = 55\nvalue =
[[0.0]\n[1.0]]"] ;
35 -> 36 ;
37 [label="Precipitation Amount <= 0.643\nsquared_error =
0.055\nsamples = 8\nvalue = [[0.0]\n[0.125]]"] ;
35 -> 37 ;
38 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
37 -> 38 ;
39 [label="squared_error = 0.0\nsamples = 7\nvalue =
[[0.0]\n[0.0]]"] ;
37 -> 39 ;
40 [label="Precipitation Intensity <= 3.696\nsquared_error =
0.152\nsamples = 17\nvalue = [[0.529]\n[0.941]]"] ;
8 -> 40 ;
```

```
41 [label="squared_error = 0.0\nsamples = 7\nvalue =
[[1.0]\n[1.0]]"] ;
40 -> 41 ;
42 [label="Precipitation Amount <= 1.455\nsquared_error =
0.125\nsamples = 10\nvalue = [[0.2]\n[0.9]]"] ;
40 -> 42 ;
43 [label="Wind Speed <= 1.232\nsquared_error = 0.061\nsamples =
7\nvalue = [[0.0]\n[0.857]]"] ;
42 -> 43 ;
44 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
43 -> 44 ;
45 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
43 -> 45 ;
46 [label="Snow Depth <= 3.405\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.667]\n[1.0]]"] ;
42 -> 46 ;
47 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
46 -> 47 ;
48 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
46 -> 48 ;
49 [label="Precipitation Amount <= 1.482\nsquared_error =
0.117\nsamples = 72\nvalue = [[0.375]\n[1.0]]"] ;
7 -> 49 ;
50 [label="Air Temperature <= -4.048\nsquared_error = 0.124\nsamples
= 30\nvalue = [[0.533]\n[1.0]]"] ;
49 -> 50 ;
51 [label="Delta T <= 1.369\nsquared_error = 0.114\nsamples =
17\nvalue = [[0.353]\n[1.0]]"] ;
50 -> 51 ;
52 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
51 -> 52 ;
53 [label="Delta T <= 12.381\nsquared_error = 0.098\nsamples =
15\nvalue = [[0.267]\n[1.0]]"] ;
51 -> 53 ;
54 [label="Wind Speed <= 7.768\nsquared_error = 0.041\nsamples =
11\nvalue = [[0.091]\n[1.0]]"] ;
53 -> 54 ;
55 [label="Cloud Amount <= 4.152\nsquared_error = 0.125\nsamples =
2\nvalue = [[0.5]\n[1.0]]"] ;
54 -> 55 ;
56 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
55 -> 56 ;
57 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
55 -> 57 ;
58 [label="squared_error = 0.0\nsamples = 9\nvalue =
[[0.0]\n[1.0]]"] ;
54 -> 58 ;
59 [label="Cloud Amount <= 4.929\nsquared_error = 0.094\nsamples =
4\nvalue = [[0.75]\n[1.0]]"] ;
53 -> 59 ;
60 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
59 -> 60 ;
61 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
59 -> 61 ;
```

```
62 [label="Cloud Amount <= 6.75\nsquared_error = 0.089\nsamples =
13\nvalue = [[0.769]\n[1.0]]"] ;
50 -> 62 ;
63 [label="squared_error = 0.0\nsamples = 10\nvalue =
[[1.0]\n[1.0]]"] ;
62 -> 63 ;
64 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
62 -> 64 ;
65 [label="Precipitation Intensity <= 2.375\nsquared_error =
0.097\nsamples = 42\nvalue = [[0.262]\n[1.0]]"] ;
49 -> 65 ;
66 [label="Liquid Flow Rate <= 0.11\nsquared_error = 0.124\nsamples
= 15\nvalue = [[0.533]\n[1.0]]"] ;
65 -> 66 ;
67 [label="Cloud Amount <= 6.67\nsquared_error = 0.086\nsamples =
9\nvalue = [[0.222]\n[1.0]]"] ;
66 -> 67 ;
68 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
67 -> 68 ;
69 [label="Precipitation Intensity <= 0.357\nsquared_error =
0.111\nsamples = 3\nvalue = [[0.667]\n[1.0]]"] ;
67 -> 69 ;
70 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
69 -> 70 ;
71 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
69 -> 71 ;
72 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[1.0]\n[1.0]]"] ;
66 -> 72 ;
73 [label="Air Temperature <= -10.268\nsquared_error =
0.049\nsamples = 27\nvalue = [[0.111]\n[1.0]]"] ;
65 -> 73 ;
74 [label="Air Temperature <= -10.988\nsquared_error =
0.125\nsamples = 4\nvalue = [[0.5]\n[1.0]]"] ;
73 -> 74 ;
75 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
74 -> 75 ;
76 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
74 -> 76 ;
77 [label="Liquid Flow Rate <= 0.018\nsquared_error = 0.021\nsamples
= 23\nvalue = [[0.043]\n[1.0]]"] ;
73 -> 77 ;
78 [label="Precipitation Intensity <= 4.125\nsquared_error =
0.125\nsamples = 2\nvalue = [[0.5]\n[1.0]]"] ;
77 -> 78 ;
79 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
78 -> 79 ;
80 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
78 -> 80 ;
81 [label="squared_error = 0.0\nsamples = 21\nvalue =
[[0.0]\n[1.0]]"] ;
77 -> 81 ;
82 [label="Liquid Flow Rate <= 0.262\nsquared_error = 0.148\nsamples
= 120\nvalue = [[0.1]\n[0.708]]"] ;
6 -> 82 ;
```

```
83 [label="Wind Speed <= 8.759\nsquared_error = 0.045\nsamples =
81\nvalue = [[0.099]\n[1.0]]"] ;
82 -> 83 ;
84 [label="Air Temperature <= -0.119\nsquared_error = 0.03\nsamples
= 77\nvalue = [[0.065]\n[1.0]]"] ;
83 -> 84 ;
85 [label="Wind Speed <= 6.964\nsquared_error = 0.019\nsamples =
75\nvalue = [[0.04]\n[1.0]]"] ;
84 -> 85 ;
86 [label="squared_error = 0.0\nsamples = 57\nvalue =
[[0.0]\n[1.0]]"] ;
85 -> 86 ;
87 [label="Cloud Amount <= 2.625\nsquared_error = 0.069\nsamples =
18\nvalue = [[0.167]\n[1.0]]"] ;
85 -> 87 ;
88 [label="Air Temperature <= -8.958\nsquared_error = 0.122\nsamples
= 7\nvalue = [[0.429]\n[1.0]]"] ;
87 -> 88 ;
89 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
88 -> 89 ;
90 [label="Snow Depth <= 14.161\nsquared_error = 0.094\nsamples =
4\nvalue = [[0.75]\n[1.0]]"] ;
88 -> 90 ;
91 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
90 -> 91 ;
92 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
90 -> 92 ;
93 [label="squared_error = 0.0\nsamples = 11\nvalue =
[[0.0]\n[1.0]]"] ;
87 -> 93 ;
94 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
84 -> 94 ;
95 [label="Cloud Amount <= 5.571\nsquared_error = 0.094\nsamples =
4\nvalue = [[0.75]\n[1.0]]"] ;
83 -> 95 ;
96 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
95 -> 96 ;
97 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
95 -> 97 ;
98 [label="Liquid Flow Rate <= 0.286\nsquared_error = 0.092\nsamples
= 39\nvalue = [[0.103]\n[0.103]]"] ;
82 -> 98 ;
99 [label="Delta T <= 17.202\nsquared_error = 0.12\nsamples =
10\nvalue = [[0.0]\n[0.4]]"] ;
98 -> 99 ;
100 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
99 -> 100 ;
101 [label="Cloud Amount <= 3.482\nsquared_error = 0.094\nsamples =
8\nvalue = [[0.0]\n[0.25]]"] ;
99 -> 101 ;
102 [label="Wind Speed <= 8.116\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.0]\n[0.667]]"] ;
101 -> 102 ;
103 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
102 -> 103 ;
```

```
104 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
102 -> 104 ;
105 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[0.0]\n[0.0]]"] ;
101 -> 105 ;
106 [label="Precipitation Amount <= 0.143\nsquared_error =
0.059\nsamples = 29\nvalue = [[0.138]\n[0.0]]"] ;
98 -> 106 ;
107 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
106 -> 107 ;
108 [label="Cloud Amount <= 0.562\nsquared_error = 0.048\nsamples =
28\nvalue = [[0.107]\n[0.0]]"] ;
106 -> 108 ;
109 [label="Precipitation Intensity <= 3.625\nsquared_error =
0.125\nsamples = 4\nvalue = [[0.5]\n[0.0]]"] ;
108 -> 109 ;
110 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
109 -> 110 ;
111 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
109 -> 111 ;
112 [label="Liquid Flow Rate <= 0.381\nsquared_error = 0.02\nsamples
= 24\nvalue = [[0.042]\n[0.0]]"] ;
108 -> 112 ;
113 [label="squared_error = 0.0\nsamples = 21\nvalue =
[[0.0]\n[0.0]]"] ;
112 -> 113 ;
114 [label="Snow Depth <= 20.274\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.333]\n[0.0]]"] ;
112 -> 114 ;
115 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
114 -> 115 ;
116 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
114 -> 116 ;
117 [label="Delta T <= 6.786\nsquared_error = 0.162\nsamples =
929\nvalue = [[0.098]\n[0.377]]"] ;
1 -> 117 ;
118 [label="Delta T <= 5.595\nsquared_error = 0.082\nsamples =
329\nvalue = [[0.1]\n[0.921]]"] ;
117 -> 118 ;
119 [label="Wind Speed <= 8.089\nsquared_error = 0.05\nsamples =
267\nvalue = [[0.097]\n[0.989]]"] ;
118 -> 119 ;
120 [label="Air Temperature <= -0.905\nsquared_error =
0.033\nsamples = 246\nvalue = [[0.061]\n[0.992]]"] ;
119 -> 120 ;
121 [label="Precipitation Amount <= 0.018\nsquared_error =
0.021\nsamples = 226\nvalue = [[0.035]\n[0.991]]"] ;
120 -> 121 ;
122 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
121 -> 122 ;
123 [label="Wind Speed <= 6.696\nsquared_error = 0.019\nsamples =
225\nvalue = [[0.031]\n[0.991]]"] ;
121 -> 123 ;
124 [label="Liquid Flow Rate <= 0.423\nsquared_error =
0.005\nsamples = 188\nvalue = [[0.005]\n[0.995]]"] ;
123 -> 124 ;
```

```
125 [label="Precipitation Amount <= 0.411\nsquared_error =
0.08\nsamples = 5\nvalue = [[0.2]\n[1.0]]"] ;
124 -> 125 ;
126 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
125 -> 126 ;
127 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
125 -> 127 ;
128 [label="Liquid Flow Rate <= 0.97\nsquared_error = 0.003\nsamples
= 183\nvalue = [[0.0]\n[0.995]]"] ;
124 -> 128 ;
129 [label="squared_error = 0.0\nsamples = 176\nvalue =
[[0.0]\n[1.0]]"] ;
128 -> 129 ;
130 [label="Delta T <= 4.94\nsquared_error = 0.061\nsamples =
7\nvalue = [[0.0]\n[0.857]]"] ;
128 -> 130 ;
131 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
130 -> 131 ;
132 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
130 -> 132 ;
133 [label="Air Temperature <= -11.905\nsquared_error =
0.081\nsamples = 37\nvalue = [[0.162]\n[0.973]]"] ;
123 -> 133 ;
134 [label="Snow Depth <= 5.494\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.667]\n[1.0]]"] ;
133 -> 134 ;
135 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
134 -> 135 ;
136 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
134 -> 136 ;
137 [label="Liquid Flow Rate <= 0.557\nsquared_error =
0.066\nsamples = 34\nvalue = [[0.118]\n[0.971]]"] ;
133 -> 137 ;
138 [label="Precipitation Intensity <= 1.875\nsquared_error =
0.122\nsamples = 7\nvalue = [[0.429]\n[1.0]]"] ;
137 -> 138 ;
139 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
138 -> 139 ;
140 [label="Precipitation Amount <= 2.268\nsquared_error =
0.094\nsamples = 4\nvalue = [[0.75]\n[1.0]]"] ;
138 -> 140 ;
141 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
140 -> 141 ;
142 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
140 -> 142 ;
143 [label="Delta T <= 5.298\nsquared_error = 0.036\nsamples =
27\nvalue = [[0.037]\n[0.963]]"] ;
137 -> 143 ;
144 [label="Air Temperature <= -3.655\nsquared_error = 0.02\nsamples
= 24\nvalue = [[0.042]\n[1.0]]"] ;
143 -> 144 ;
145 [label="squared_error = 0.0\nsamples = 19\nvalue =
[[0.0]\n[1.0]]"] ;
144 -> 145 ;
```

```
146 [label="Cloud Amount <= 5.92\nsquared_error = 0.08\nsamples =
5\nvalue = [[0.2]\n[1.0]]"] ;
144 -> 146 ;
147 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
146 -> 147 ;
148 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
146 -> 148 ;
149 [label="Precipitation Intensity <= 1.357\nsquared_error =
0.111\nsamples = 3\nvalue = [[0.0]\n[0.667]]"] ;
143 -> 149 ;
150 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
149 -> 150 ;
151 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
149 -> 151 ;
152 [label="Cloud Amount <= 1.393\nsquared_error = 0.114\nsamples =
20\nvalue = [[0.35]\n[1.0]]"] ;
120 -> 152 ;
153 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
152 -> 153 ;
154 [label="Cloud Amount <= 7.795\nsquared_error = 0.09\nsamples =
17\nvalue = [[0.235]\n[1.0]]"] ;
152 -> 154 ;
155 [label="Snow Depth <= 0.542\nsquared_error = 0.058\nsamples =
15\nvalue = [[0.133]\n[1.0]]"] ;
154 -> 155 ;
156 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
155 -> 156 ;
157 [label="Precipitation Intensity <= 5.732\nsquared_error =
0.033\nsamples = 14\nvalue = [[0.071]\n[1.0]]"] ;
155 -> 157 ;
158 [label="squared_error = 0.0\nsamples = 12\nvalue =
[[0.0]\n[1.0]]"] ;
157 -> 158 ;
159 [label="Wind Speed <= 4.232\nsquared_error = 0.125\nsamples =
2\nvalue = [[0.5]\n[1.0]]"] ;
157 -> 159 ;
160 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
159 -> 160 ;
161 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
159 -> 161 ;
162 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
154 -> 162 ;
163 [label="Air Temperature <= -7.78\nsquared_error = 0.147\nsamples
= 21\nvalue = [[0.524]\n[0.952]]"] ;
119 -> 163 ;
164 [label="Snow Depth <= 1.857\nsquared_error = 0.109\nsamples =
8\nvalue = [[0.125]\n[0.875]]"] ;
163 -> 164 ;
165 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
164 -> 165 ;
166 [label="Liquid Flow Rate <= 0.896\nsquared_error =
0.061\nsamples = 7\nvalue = [[0.0]\n[0.857]]"] ;
164 -> 166 ;
```

```
167 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
166 -> 167 ;
168 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
166 -> 168 ;
169 [label="Snow Depth <= 16.792\nsquared_error = 0.089\nsamples =
13\nvalue = [[0.769]\n[1.0]]"] ;
163 -> 169 ;
170 [label="squared_error = 0.0\nsamples = 7\nvalue =
[[1.0]\n[1.0]]"] ;
169 -> 170 ;
171 [label="Delta T <= 2.262\nsquared_error = 0.125\nsamples =
6\nvalue = [[0.5]\n[1.0]]"] ;
169 -> 171 ;
172 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
171 -> 172 ;
173 [label="Precipitation Amount <= 1.527\nsquared_error =
0.094\nsamples = 4\nvalue = [[0.25]\n[1.0]]"] ;
171 -> 173 ;
174 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
173 -> 174 ;
175 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
173 -> 175 ;
176 [label="Liquid Flow Rate <= 0.75\nsquared_error = 0.167\nsamples
= 62\nvalue = [[0.113]\n[0.629]]"] ;
118 -> 176 ;
177 [label="Liquid Flow Rate <= 0.741\nsquared_error =
0.038\nsamples = 36\nvalue = [[0.083]\n[1.0]]"] ;
176 -> 177 ;
178 [label="Cloud Amount <= 8.384\nsquared_error = 0.027\nsamples =
35\nvalue = [[0.057]\n[1.0]]"] ;
177 -> 178 ;
179 [label="Air Temperature <= -2.083\nsquared_error =
0.015\nsamples = 33\nvalue = [[0.03]\n[1.0]]"] ;
178 -> 179 ;
180 [label="squared_error = 0.0\nsamples = 28\nvalue =
[[0.0]\n[1.0]]"] ;
179 -> 180 ;
181 [label="Air Temperature <= -1.887\nsquared_error = 0.08\nsamples
= 5\nvalue = [[0.2]\n[1.0]]"] ;
179 -> 181 ;
182 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
181 -> 182 ;
183 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
181 -> 183 ;
184 [label="Air Temperature <= -8.696\nsquared_error =
0.125\nsamples = 2\nvalue = [[0.5]\n[1.0]]"] ;
178 -> 184 ;
185 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
184 -> 185 ;
186 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
184 -> 186 ;
187 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
177 -> 187 ;
```

```
188 [label="Wind Speed <= 7.688\nsquared_error = 0.116\nsamples =
26\nvalue = [[0.154]\n[0.115]]"] ;
176 -> 188 ;
189 [label="Delta T <= 5.833\nsquared_error = 0.078\nsamples =
23\nvalue = [[0.043]\n[0.13]]"] ;
188 -> 189 ;
190 [label="Cloud Amount <= 3.589\nsquared_error = 0.194\nsamples =
6\nvalue = [[0.167]\n[0.5]]"] ;
189 -> 190 ;
191 [label="Snow Depth <= 16.637\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.333]\n[0.0]]"] ;
190 -> 191 ;
192 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
191 -> 192 ;
193 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
191 -> 193 ;
194 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
190 -> 194 ;
195 [label="squared_error = 0.0\nsamples = 17\nvalue =
[[0.0]\n[0.0]]"] ;
189 -> 195 ;
196 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
188 -> 196 ;
197 [label="Delta T <= 10.357\nsquared_error = 0.08\nsamples =
600\nvalue = [[0.097]\n[0.078]]"] ;
117 -> 197 ;
198 [label="Liquid Flow Rate <= 0.598\nsquared_error =
0.158\nsamples = 147\nvalue = [[0.136]\n[0.272]]"] ;
197 -> 198 ;
199 [label="Wind Speed <= 7.821\nsquared_error = 0.159\nsamples =
48\nvalue = [[0.188]\n[0.792]]"] ;
198 -> 199 ;
200 [label="Liquid Flow Rate <= 0.536\nsquared_error =
0.134\nsamples = 42\nvalue = [[0.095]\n[0.762]]"] ;
199 -> 200 ;
201 [label="Air Temperature <= -0.643\nsquared_error =
0.081\nsamples = 28\nvalue = [[0.107]\n[0.929]]"] ;
200 -> 201 ;
202 [label="Delta T <= 9.702\nsquared_error = 0.041\nsamples =
22\nvalue = [[0.0]\n[0.909]]"] ;
201 -> 202 ;
203 [label="squared_error = 0.0\nsamples = 16\nvalue =
[[0.0]\n[1.0]]"] ;
202 -> 203 ;
204 [label="Liquid Flow Rate <= 0.476\nsquared_error =
0.111\nsamples = 6\nvalue = [[0.0]\n[0.667]]"] ;
202 -> 204 ;
205 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
204 -> 205 ;
206 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
204 -> 206 ;
207 [label="Wind Speed <= 3.911\nsquared_error = 0.125\nsamples =
6\nvalue = [[0.5]\n[1.0]]"] ;
201 -> 207 ;
208 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
207 -> 208 ;
```

```
209 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[1.0]]"] ;
207 -> 209 ;
210 [label="Delta T <= 8.631\nsquared_error = 0.156\nsamples =
14\nvalue = [[0.071]\n[0.429]]"] ;
200 -> 210 ;
211 [label="Precipitation Intensity <= 4.857\nsquared_error =
0.061\nsamples = 7\nvalue = [[0.0]\n[0.857]]"] ;
210 -> 211 ;
212 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
211 -> 212 ;
213 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
211 -> 213 ;
214 [label="Air Temperature <= -0.446\nsquared_error =
0.061\nsamples = 7\nvalue = [[0.143]\n[0.0]]"] ;
210 -> 214 ;
215 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[0.0]]"] ;
214 -> 215 ;
216 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
214 -> 216 ;
217 [label="Cloud Amount <= 2.652\nsquared_error = 0.069\nsamples =
6\nvalue = [[0.833]\n[1.0]]"] ;
199 -> 217 ;
218 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
217 -> 218 ;
219 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[1.0]\n[1.0]]"] ;
217 -> 219 ;
220 [label="Wind Speed <= 7.018\nsquared_error = 0.059\nsamples =
99\nvalue = [[0.111]\n[0.02]]"] ;
198 -> 220 ;
221 [label="Air Temperature <= -0.25\nsquared_error = 0.013\nsamples
= 77\nvalue = [[0.026]\n[0.0]]"] ;
220 -> 221 ;
222 [label="squared_error = 0.0\nsamples = 69\nvalue =
[[0.0]\n[0.0]]"] ;
221 -> 222 ;
223 [label="Snow Depth <= 14.083\nsquared_error = 0.094\nsamples =
8\nvalue = [[0.25]\n[0.0]]"] ;
221 -> 223 ;
224 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[0.0]\n[0.0]]"] ;
223 -> 224 ;
225 [label="Wind Speed <= 2.411\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.667]\n[0.0]]"] ;
223 -> 225 ;
226 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
225 -> 226 ;
227 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
225 -> 227 ;
228 [label="Liquid Flow Rate <= 0.688\nsquared_error =
0.162\nsamples = 22\nvalue = [[0.409]\n[0.091]]"] ;
220 -> 228 ;
229 [label="Wind Speed <= 7.848\nsquared_error = 0.125\nsamples =
4\nvalue = [[0.0]\n[0.5]]"] ;
228 -> 229 ;
```

```
230 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
229 -> 230 ;
231 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
229 -> 231 ;
232 [label="Precipitation Amount <= 0.643\nsquared_error =
0.125\nsamples = 18\nvalue = [[0.5]\n[0.0]]"] ;
228 -> 232 ;
233 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[0.0]]"] ;
232 -> 233 ;
234 [label="Delta T <= 9.464\nsquared_error = 0.115\nsamples =
14\nvalue = [[0.357]\n[0.0]]"] ;
232 -> 234 ;
235 [label="Air Temperature <= -6.47\nsquared_error = 0.094\nsamples
= 12\nvalue = [[0.25]\n[0.0]]"] ;
234 -> 235 ;
236 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[0.0]]"] ;
235 -> 236 ;
237 [label="Air Temperature <= -2.935\nsquared_error =
0.125\nsamples = 6\nvalue = [[0.5]\n[0.0]]"] ;
235 -> 237 ;
238 [label="Precipitation Intensity <= 4.661\nsquared_error =
0.094\nsamples = 4\nvalue = [[0.75]\n[0.0]]"] ;
237 -> 238 ;
239 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
238 -> 239 ;
240 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
238 -> 240 ;
241 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
237 -> 241 ;
242 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
234 -> 242 ;
243 [label="Wind Speed <= 8.679\nsquared_error = 0.046\nsamples =
453\nvalue = [[0.084]\n[0.015]]"] ;
197 -> 243 ;
244 [label="Precipitation Amount <= 0.036\nsquared_error =
0.036\nsamples = 436\nvalue = [[0.06]\n[0.016]]"] ;
243 -> 244 ;
245 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[0.0]]"] ;
244 -> 245 ;
246 [label="Wind Speed <= 7.286\nsquared_error = 0.032\nsamples =
432\nvalue = [[0.051]\n[0.016]]"] ;
244 -> 246 ;
247 [label="Air Temperature <= -0.119\nsquared_error = 0.02\nsamples
= 362\nvalue = [[0.025]\n[0.017]]"] ;
246 -> 247 ;
248 [label="Liquid Flow Rate <= 0.446\nsquared_error =
0.016\nsamples = 349\nvalue = [[0.014]\n[0.017]]"] ;
247 -> 248 ;
249 [label="Delta T <= 12.083\nsquared_error = 0.07\nsamples =
44\nvalue = [[0.023]\n[0.136]]"] ;
248 -> 249 ;
250 [label="Air Temperature <= -13.083\nsquared_error =
0.094\nsamples = 8\nvalue = [[0.0]\n[0.75]]"] ;
249 -> 250 ;
```

```
251 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
250 -> 251 ;
252 [label="Cloud Amount <= 8.009\nsquared_error = 0.061\nsamples =
7\nvalue = [[0.0]\n[0.857]]"] ;
250 -> 252 ;
253 [label="squared_error = 0.0\nsamples = 6\nvalue =
[[0.0]\n[1.0]]"] ;
252 -> 253 ;
254 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
252 -> 254 ;
255 [label="Precipitation Intensity <= 5.875\nsquared_error =
0.014\nsamples = 36\nvalue = [[0.028]\n[0.0]]"] ;
249 -> 255 ;
256 [label="squared_error = 0.0\nsamples = 35\nvalue =
[[0.0]\n[0.0]]"] ;
255 -> 256 ;
257 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
255 -> 257 ;
258 [label="Air Temperature <= -1.036\nsquared_error =
0.006\nsamples = 305\nvalue = [[0.013]\n[0.0]]"] ;
248 -> 258 ;
259 [label="Liquid Flow Rate <= 0.452\nsquared_error =
0.002\nsamples = 283\nvalue = [[0.004]\n[0.0]]"] ;
258 -> 259 ;
260 [label="Wind Speed <= 6.08\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.333]\n[0.0]]"] ;
259 -> 260 ;
261 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
260 -> 261 ;
262 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
260 -> 262 ;
263 [label="squared_error = 0.0\nsamples = 280\nvalue =
[[0.0]\n[0.0]]"] ;
259 -> 263 ;
264 [label="Precipitation Intensity <= 0.929\nsquared_error =
0.059\nsamples = 22\nvalue = [[0.136]\n[0.0]]"] ;
258 -> 264 ;
265 [label="Wind Speed <= 3.402\nsquared_error = 0.12\nsamples =
5\nvalue = [[0.6]\n[0.0]]"] ;
264 -> 265 ;
266 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
265 -> 266 ;
267 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
265 -> 267 ;
268 [label="squared_error = 0.0\nsamples = 17\nvalue =
[[0.0]\n[0.0]]"] ;
264 -> 268 ;
269 [label="Liquid Flow Rate <= 0.568\nsquared_error =
0.107\nsamples = 13\nvalue = [[0.308]\n[0.0]]"] ;
247 -> 269 ;
270 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[0.0]]"] ;
269 -> 270 ;
271 [label="squared_error = 0.0\nsamples = 9\nvalue =
[[0.0]\n[0.0]]"] ;
269 -> 271 ;
```

```
272 [label="Snow Depth <= 23.756\nsquared_error = 0.083\nsamples =
70\nvalue = [[0.186]\n[0.014]]"] ;
246 -> 272 ;
273 [label="Cloud Amount <= 8.411\nsquared_error = 0.071\nsamples =
67\nvalue = [[0.149]\n[0.015]]"] ;
272 -> 273 ;
274 [label="Precipitation Intensity <= 5.875\nsquared_error =
0.053\nsamples = 60\nvalue = [[0.1]\n[0.017]]"] ;
273 -> 274 ;
275 [label="Delta T <= 14.762\nsquared_error = 0.046\nsamples =
59\nvalue = [[0.102]\n[0.0]]"] ;
274 -> 275 ;
276 [label="Snow Depth <= 2.012\nsquared_error = 0.076\nsamples =
32\nvalue = [[0.188]\n[0.0]]"] ;
275 -> 276 ;
277 [label="Liquid Flow Rate <= 0.777\nsquared_error =
0.111\nsamples = 3\nvalue = [[0.667]\n[0.0]]"] ;
276 -> 277 ;
278 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
277 -> 278 ;
279 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
277 -> 279 ;
280 [label="Delta T <= 14.583\nsquared_error = 0.059\nsamples =
29\nvalue = [[0.138]\n[0.0]]"] ;
276 -> 280 ;
281 [label="Wind Speed <= 7.446\nsquared_error = 0.048\nsamples =
28\nvalue = [[0.107]\n[0.0]]"] ;
280 -> 281 ;
282 [label="Cloud Amount <= 5.491\nsquared_error = 0.12\nsamples =
5\nvalue = [[0.4]\n[0.0]]"] ;
281 -> 282 ;
283 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[0.0]]"] ;
282 -> 283 ;
284 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
282 -> 284 ;
285 [label="Wind Speed <= 8.304\nsquared_error = 0.021\nsamples =
23\nvalue = [[0.043]\n[0.0]]"] ;
281 -> 285 ;
286 [label="squared_error = 0.0\nsamples = 18\nvalue =
[[0.0]\n[0.0]]"] ;
285 -> 286 ;
287 [label="Wind Speed <= 8.357\nsquared_error = 0.08\nsamples =
5\nvalue = [[0.2]\n[0.0]]"] ;
285 -> 287 ;
288 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
287 -> 288 ;
289 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[0.0]]"] ;
287 -> 289 ;
290 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
280 -> 290 ;
291 [label="squared_error = 0.0\nsamples = 27\nvalue =
[[0.0]\n[0.0]]"] ;
275 -> 291 ;
292 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
274 -> 292 ;
```

```
293 [label="Snow Depth <= 16.327\nsquared_error = 0.122\nsamples =
7\nvalue = [[0.571]\n[0.0]]"] ;
273 -> 293 ;
294 [label="Delta T <= 15.952\nsquared_error = 0.094\nsamples =
4\nvalue = [[0.25]\n[0.0]]"] ;
293 -> 294 ;
295 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[0.0]]"] ;
294 -> 295 ;
296 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
294 -> 296 ;
297 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
293 -> 297 ;
298 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
272 -> 298 ;
299 [label="Cloud Amount <= 5.116\nsquared_error = 0.104\nsamples =
17\nvalue = [[0.706]\n[0.0]]"] ;
243 -> 299 ;
300 [label="squared_error = 0.0\nsamples = 9\nvalue =
[[1.0]\n[0.0]]"] ;
299 -> 300 ;
301 [label="Snow Depth <= 5.417\nsquared_error = 0.117\nsamples =
8\nvalue = [[0.375]\n[0.0]]"] ;
299 -> 301 ;
302 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[0.0]]"] ;
301 -> 302 ;
303 [label="Liquid Flow Rate <= 0.792\nsquared_error =
0.094\nsamples = 4\nvalue = [[0.75]\n[0.0]]"] ;
301 -> 303 ;
304 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
303 -> 304 ;
305 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
303 -> 305 ;
306 [label="Delta T <= 6.429\nsquared_error = 0.138\nsamples =
842\nvalue = [[0.96]\n[0.613]]"] ;
0 -> 306 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
307 [label="Air Temperature <= 0.667\nsquared_error = 0.032\nsamples
= 290\nvalue = [[0.959]\n[0.976]]"] ;
306 -> 307 ;
308 [label="Precipitation Amount <= 1.393\nsquared_error =
0.21\nsamples = 9\nvalue = [[0.444]\n[0.778]]"] ;
307 -> 308 ;
309 [label="Cloud Amount <= 2.759\nsquared_error = 0.12\nsamples =
5\nvalue = [[0.0]\n[0.6]]"] ;
308 -> 309 ;
310 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
309 -> 310 ;
311 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[1.0]]"] ;
309 -> 311 ;
312 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[1.0]]"] ;
308 -> 312 ;
313 [label="Precipitation Amount <= 0.446\nsquared_error =
0.021\nsamples = 281\nvalue = [[0.975]\n[0.982]]"] ;
307 -> 313 ;
```

```
314 [label="Air Temperature <= 1.845\nsquared_error = 0.086\nsamples
= 41\nvalue = [[0.854]\n[0.951]]"] ;
313 -> 314 ;
315 [label="Precipitation Amount <= 0.286\nsquared_error =
0.123\nsamples = 9\nvalue = [[0.444]\n[1.0]]"] ;
314 -> 315 ;
316 [label="Wind Speed <= 0.696\nsquared_error = 0.08\nsamples =
5\nvalue = [[0.8]\n[1.0]]"] ;
315 -> 316 ;
317 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
316 -> 317 ;
318 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[1.0]]"] ;
316 -> 318 ;
319 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
315 -> 319 ;
320 [label="Wind Speed <= 0.268\nsquared_error = 0.044\nsamples =
32\nvalue = [[0.969]\n[0.938]]"] ;
314 -> 320 ;
321 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
320 -> 321 ;
322 [label="Delta T <= 5.774\nsquared_error = 0.03\nsamples =
31\nvalue = [[1.0]\n[0.935]]"] ;
320 -> 322 ;
323 [label="Liquid Flow Rate <= 0.946\nsquared_error =
0.016\nsamples = 30\nvalue = [[1.0]\n[0.967]]"] ;
322 -> 323 ;
324 [label="squared_error = 0.0\nsamples = 29\nvalue =
[[1.0]\n[1.0]]"] ;
323 -> 324 ;
325 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
323 -> 325 ;
326 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
322 -> 326 ;
327 [label="Precipitation Intensity <= 0.089\nsquared_error =
0.008\nsamples = 240\nvalue = [[0.996]\n[0.988]]"] ;
313 -> 327 ;
328 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
327 -> 328 ;
329 [label="Delta T <= 5.714\nsquared_error = 0.006\nsamples =
239\nvalue = [[1.0]\n[0.987]]"] ;
327 -> 329 ;
330 [label="squared_error = 0.0\nsamples = 215\nvalue =
[[1.0]\n[1.0]]"] ;
329 -> 330 ;
331 [label="Liquid Flow Rate <= 0.759\nsquared_error =
0.055\nsamples = 24\nvalue = [[1.0]\n[0.875]]"] ;
329 -> 331 ;
332 [label="squared_error = 0.0\nsamples = 20\nvalue =
[[1.0]\n[1.0]]"] ;
331 -> 332 ;
333 [label="Snow Depth <= 19.268\nsquared_error = 0.094\nsamples =
4\nvalue = [[1.0]\n[0.25]]"] ;
331 -> 333 ;
334 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
333 -> 334 ;
```

```
335 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
333 -> 335 ;
336 [label="Liquid Flow Rate <= 0.399\nsquared_error =
0.141\nsamples = 552\nvalue = [[0.96]\n[0.422]]"] ;
306 -> 336 ;
337 [label="Liquid Flow Rate <= 0.283\nsquared_error =
0.065\nsamples = 231\nvalue = [[0.97]\n[0.887]]"] ;
336 -> 337 ;
338 [label="Air Temperature <= 0.798\nsquared_error = 0.02\nsamples
= 166\nvalue = [[0.964]\n[0.994]]"] ;
337 -> 338 ;
339 [label="Precipitation Amount <= 1.054\nsquared_error =
0.125\nsamples = 8\nvalue = [[0.5]\n[1.0]]"] ;
338 -> 339 ;
340 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[1.0]]"] ;
339 -> 340 ;
341 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[1.0]]"] ;
339 -> 341 ;
342 [label="Precipitation Amount <= 0.054\nsquared_error =
0.009\nsamples = 158\nvalue = [[0.987]\n[0.994]]"] ;
338 -> 342 ;
343 [label="Air Temperature <= 1.649\nsquared_error = 0.125\nsamples
= 2\nvalue = [[0.5]\n[1.0]]"] ;
342 -> 343 ;
344 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
343 -> 344 ;
345 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
343 -> 345 ;
346 [label="Liquid Flow Rate <= 0.262\nsquared_error =
0.006\nsamples = 156\nvalue = [[0.994]\n[0.994]]"] ;
342 -> 346 ;
347 [label="Wind Speed <= 0.429\nsquared_error = 0.003\nsamples =
150\nvalue = [[0.993]\n[1.0]]"] ;
346 -> 347 ;
348 [label="Precipitation Amount <= 0.527\nsquared_error =
0.069\nsamples = 6\nvalue = [[0.833]\n[1.0]]"] ;
347 -> 348 ;
349 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
348 -> 349 ;
350 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[1.0]\n[1.0]]"] ;
348 -> 350 ;
351 [label="squared_error = 0.0\nsamples = 144\nvalue =
[[1.0]\n[1.0]]"] ;
347 -> 351 ;
352 [label="Delta T <= 17.143\nsquared_error = 0.069\nsamples =
6\nvalue = [[1.0]\n[0.833]]"] ;
346 -> 352 ;
353 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[1.0]\n[1.0]]"] ;
352 -> 353 ;
354 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
352 -> 354 ;
355 [label="Delta T <= 15.179\nsquared_error = 0.126\nsamples =
65\nvalue = [[0.985]\n[0.615]]"] ;
337 -> 355 ;
```

```
356 [label="Air Temperature <= 7.738\nsquared_error = 0.024\nsamples
= 41\nvalue = [[0.976]\n[0.976]]"] ;
355 -> 356 ;
357 [label="Delta T <= 13.988\nsquared_error = 0.012\nsamples =
40\nvalue = [[1.0]\n[0.975]]"] ;
356 -> 357 ;
358 [label="squared_error = 0.0\nsamples = 37\nvalue =
[[1.0]\n[1.0]]"] ;
357 -> 358 ;
359 [label="Air Temperature <= 6.232\nsquared_error = 0.111\nsamples
= 3\nvalue = [[1.0]\n[0.667]]"] ;
357 -> 359 ;
360 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
359 -> 360 ;
361 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
359 -> 361 ;
362 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
356 -> 362 ;
363 [label="squared_error = 0.0\nsamples = 24\nvalue =
[[1.0]\n[0.0]]"] ;
355 -> 363 ;
364 [label="Delta T <= 10.0\nsquared_error = 0.062\nsamples =
321\nvalue = [[0.953]\n[0.087]]"] ;
336 -> 364 ;
365 [label="Liquid Flow Rate <= 0.604\nsquared_error =
0.132\nsamples = 87\nvalue = [[0.943]\n[0.299]]"] ;
364 -> 365 ;
366 [label="Liquid Flow Rate <= 0.589\nsquared_error =
0.106\nsamples = 29\nvalue = [[0.897]\n[0.862]]"] ;
365 -> 366 ;
367 [label="Liquid Flow Rate <= 0.423\nsquared_error =
0.042\nsamples = 23\nvalue = [[0.957]\n[0.957]]"] ;
366 -> 367 ;
368 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[1.0]]"] ;
367 -> 368 ;
369 [label="Cloud Amount <= 8.491\nsquared_error = 0.022\nsamples =
22\nvalue = [[1.0]\n[0.955]]"] ;
367 -> 369 ;
370 [label="squared_error = 0.0\nsamples = 19\nvalue =
[[1.0]\n[1.0]]"] ;
369 -> 370 ;
371 [label="Delta T <= 9.226\nsquared_error = 0.111\nsamples =
3\nvalue = [[1.0]\n[0.667]]"] ;
369 -> 371 ;
372 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
371 -> 372 ;
373 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
371 -> 373 ;
374 [label="Delta T <= 8.214\nsquared_error = 0.236\nsamples =
6\nvalue = [[0.667]\n[0.5]]"] ;
366 -> 374 ;
375 [label="Air Temperature <= 3.679\nsquared_error = 0.111\nsamples
= 3\nvalue = [[0.333]\n[1.0]]"] ;
374 -> 375 ;
376 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[1.0]]"] ;
375 -> 376 ;
```

```
377 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
375 -> 377 ;
378 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
374 -> 378 ;
379 [label="Wind Speed <= 0.482\nsquared_error = 0.025\nsamples =
58\nvalue = [[0.966]\n[0.017]]"] ;
365 -> 379 ;
380 [label="Delta T <= 7.917\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.333]\n[0.0]]"] ;
379 -> 380 ;
381 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
380 -> 381 ;
382 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[0.0]\n[0.0]]"] ;
380 -> 382 ;
383 [label="Air Temperature <= 7.476\nsquared_error = 0.009\nsamples
= 55\nvalue = [[1.0]\n[0.018]]"] ;
379 -> 383 ;
384 [label="squared_error = 0.0\nsamples = 49\nvalue =
[[1.0]\n[0.0]]"] ;
383 -> 384 ;
385 [label="Air Temperature <= 7.673\nsquared_error = 0.069\nsamples
= 6\nvalue = [[1.0]\n[0.167]]"] ;
383 -> 385 ;
386 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[1.0]]"] ;
385 -> 386 ;
387 [label="squared_error = 0.0\nsamples = 5\nvalue =
[[1.0]\n[0.0]]"] ;
385 -> 387 ;
388 [label="Wind Speed <= 1.875\nsquared_error = 0.025\nsamples =
234\nvalue = [[0.957]\n[0.009]]"] ;
364 -> 388 ;
389 [label="Precipitation Amount <= 0.33\nsquared_error =
0.079\nsamples = 41\nvalue = [[0.805]\n[0.0]]"] ;
388 -> 389 ;
390 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[0.0]\n[0.0]]"] ;
389 -> 390 ;
391 [label="Air Temperature <= 1.583\nsquared_error = 0.048\nsamples
= 37\nvalue = [[0.892]\n[0.0]]"] ;
389 -> 391 ;
392 [label="Snow Depth <= 19.577\nsquared_error = 0.125\nsamples =
6\nvalue = [[0.5]\n[0.0]]"] ;
391 -> 392 ;
393 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[1.0]\n[0.0]]"] ;
392 -> 393 ;
394 [label="squared_error = 0.0\nsamples = 3\nvalue =
[[0.0]\n[0.0]]"] ;
392 -> 394 ;
395 [label="Precipitation Amount <= 0.384\nsquared_error =
0.016\nsamples = 31\nvalue = [[0.968]\n[0.0]]"] ;
391 -> 395 ;
396 [label="Air Temperature <= 3.679\nsquared_error = 0.125\nsamples
= 2\nvalue = [[0.5]\n[0.0]]"] ;
395 -> 396 ;
397 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[1.0]\n[0.0]]"] ;
396 -> 397 ;
```

```
398 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
396 -> 398 ;
399 [label="squared_error = 0.0\nsamples = 29\nvalue =
[[1.0]\n[0.0]]"] ;
395 -> 399 ;
400 [label="Air Temperature <= 0.405\nsquared_error = 0.01\nsamples
= 193\nvalue = [[0.99]\n[0.01]]"] ;
388 -> 400 ;
401 [label="Wind Speed <= 7.58\nsquared_error = 0.111\nsamples =
3\nvalue = [[0.667]\n[0.0]]"] ;
400 -> 401 ;
402 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[0.0]]"] ;
401 -> 402 ;
403 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
401 -> 403 ;
404 [label="Liquid Flow Rate <= 0.452\nsquared_error =
0.008\nsamples = 190\nvalue = [[0.995]\n[0.011]]"] ;
400 -> 404 ;
405 [label="Liquid Flow Rate <= 0.443\nsquared_error =
0.065\nsamples = 13\nvalue = [[1.0]\n[0.154]]"] ;
404 -> 405 ;
406 [label="squared_error = 0.0\nsamples = 11\nvalue =
[[1.0]\n[0.0]]"] ;
405 -> 406 ;
407 [label="squared_error = 0.0\nsamples = 2\nvalue =
[[1.0]\n[1.0]]"] ;
405 -> 407 ;
408 [label="Cloud Amount <= 0.321\nsquared_error = 0.003\nsamples =
177\nvalue = [[0.994]\n[0.0]]"] ;
404 -> 408 ;
409 [label="Precipitation Intensity <= 1.25\nsquared_error =
0.08\nsamples = 5\nvalue = [[0.8]\n[0.0]]"] ;
408 -> 409 ;
410 [label="squared_error = 0.0\nsamples = 1\nvalue =
[[0.0]\n[0.0]]"] ;
409 -> 410 ;
411 [label="squared_error = 0.0\nsamples = 4\nvalue =
[[1.0]\n[0.0]]"] ;
409 -> 411 ;
412 [label="squared_error = 0.0\nsamples = 172\nvalue =
[[1.0]\n[0.0]]"] ;
408 -> 412 ;
}
```

7.6 Appendix E – Python Code: *LHS_Loop.py*

```
# {Set Timer}
from timeit import default_timer as timer
start = timer()

# {Loop For N number of weeks}
i=0
while i<3000: #Choose value for N number of weeks

    # {Import Necessary Modules}
    import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    from smt.sampling_methods import LHS
    from numpy import savetxt

    # {Read CSV File and Select Variables Which Will be Used}
    df2 = pd.read_csv('Weather_Data_(Not_Formatted).csv')
    maxv = df2['Year']
    maxv2 = df2['Month']
    maxv3 = df2['Day']
    maxv4 = df2['Time']
    maxv5 = df2['CloudAmount']
    maxv6 = df2['PrecipitationAmount']
    maxv7 = df2['PrecipitationIntensity']
    maxv8 = df2['SnowDepth']
    maxv9 = df2['AirTemperature']
    maxv10 = df2['WindSpeed']

    # {Identify Boundaries}
    maxv5max = maxv5.max()          # {Cloud amount (1/8)}
    maxv5min = maxv5.min()
    maxv6max = maxv6.max()          # {Precipitation amount (mm)}
    maxv6min = maxv6.min()
    maxv7max = maxv7.max()          # {Precipitation intensity
(mm/h)}
    maxv7min = maxv7.min()
    maxv8max = maxv8.max()          # {Snow depth (cm)}
    maxv8min = maxv8.min()
    maxv9max = maxv9.max()          # {Air temperature (degC)}
    maxv9min = maxv9.min()
    maxv10max = maxv10.max()        # {Wind speed (m/s)}
    maxv10min = maxv10.min()
    maxv11max = 20                  # {DeltaT (C)}
    maxv11min = 0
    maxv12max = 1                   # {Liquid Flow Rate (l/s)}
    maxv12min = 0.01

    # {Setting Limits on Random Number Generation}
    xlimits = np.array([[int(maxv5min), int(maxv5max)],
[int(maxv6min), int(maxv6max)], [int(maxv7min), int(maxv7max)],
                        [int(maxv8min), int(maxv8max)],
[int(maxv9min), int(maxv9max)], [int(maxv10min), int(maxv10max)],
                        [int(maxv11min), int(maxv11max)],
[int(maxv12min), int(maxv12max)]])

    # {Running Latin Hyper Cube Sampling (LHS)}
    rng = np.random.RandomState(None)
    sampling = LHS(xlimits=xlimits, criterion = 'c',
random_state=rng)
    num = 168
    xnew = sampling(num)
```

```
    # {Save Outputs}
    savetxt('Intermediate1.csv', xnew, fmt ='%f', delimiter=',',
header= "Cloud Amount (1/8), Precipitation Amount (mm),
Precipitation Intensity (mm/h), Snow Depth (cm), Air Temperature
(degC), Wind Speed (m/s), DeltaT (C), LiquidFlowRate (l/s)")

    # {Convert Output to DataFrame}
    amd = pd.read_csv('Intermediate1.csv')
    print(amd)

    amd["Year"] = pd.to_numeric(df2["Year"],errors='coerce')
    amd["Month"] = pd.to_numeric(df2["Month"],errors='coerce')
    amd["Day"] = pd.to_numeric(df2["Day"],errors='coerce')
    amd["Time"] = pd.to_numeric(df2["Time"],errors='coerce')
    amd["Cloud Amount"] = pd.to_numeric(amd['# Cloud Amount
(1/8)'],errors='coerce')
    amd["Precipitation Amount"] = pd.to_numeric(amd[' Precipitation
Amount (mm)'],errors='coerce')
    amd["Precipitation Intensity"] = pd.to_numeric(amd['
Precipitation Intensity (mm/h)'],errors='coerce')
    amd["Snow Depth"] = pd.to_numeric(amd[' Snow Depth
(cm)'],errors='coerce')
    amd["Air Temperature"] = pd.to_numeric(amd[' Air Temperature
(degC)'],errors='coerce')
    amd["Wind Speed"] = pd.to_numeric(amd[' Wind Speed
(m/s)'],errors='coerce')
    amd["Delta T"] = pd.to_numeric(amd[' DeltaT
(C)'],errors='coerce')
    amd["Liquid Flow Rate"] = pd.to_numeric(amd[' LiquidFlowRate
(l/s)'],errors='coerce')

    # {Correct Generated Parameters Based on Unrealsitic Weather
Scenarios}
    amd["Precipitation Amount"] = np.where(amd["Precipitation
Intensity"] >= (amd["Precipitation Amount"]+2.9), amd["Precipitation
Intensity"], amd["Precipitation Amount"])
    amd["Precipitation Amount"] = np.where(amd["Precipitation
Intensity"] == 0, amd["Precipitation Intensity"], amd["Precipitation
Amount"])
    amd["Precipitation Intensity"] = np.where(amd["Precipitation
Amount"] == 0, amd["Precipitation Amount"], amd["Precipitation
Intensity"])
    amd["Precipitation Amount"] = np.where(amd["Cloud Amount"] == 0,
0, amd["Precipitation Amount"])
    amd["Precipitation Intensity"] = np.where(amd["Cloud Amount"] ==
0, 0, amd["Precipitation Intensity"])

    # {Set All Weather Parameters to Constant Values and Use Linear
Interpolation For Control Inputs}
    amd["Cloud Amount"] = np.where(amd["Cloud Amount"] !=
amd.iloc[0,0], amd.iloc[0,0], amd.iloc[0,0])
    amd["Precipitation Amount"] = np.where(amd["Precipitation
Amount"] != amd.iloc[0,1], amd.iloc[0,1], amd.iloc[0,1])
    amd["Precipitation Intensity"] = np.where(amd["Precipitation
Intensity"] != amd.iloc[0,2], amd.iloc[0,2], amd.iloc[0,2])
    amd["Snow Depth"] = np.where(amd["Snow Depth"] != amd.iloc[0,3],
amd.iloc[0,3], amd.iloc[0,3])
    amd["Air Temperature"] = np.where(amd["Air Temperature"] !=
amd.iloc[0,4], amd.iloc[0,4], amd.iloc[0,4])
    amd["Wind Speed"] = np.where(amd["Wind Speed"] != amd.iloc[0,5],
amd.iloc[0,5], amd.iloc[0,5])
    amd["Delta T"] = np.where(amd["Delta T"] != amd.iloc[0,6],
amd.iloc[0,6], amd.iloc[0,6])
```

```
    amd["Liquid Flow Rate"] = np.where(amd["Liquid Flow Rate"] !=
amd.iloc[0,7], amd.iloc[0,7], amd.iloc[0,7])


    # {When Snow Depth is Increasing then Delta T and Liquid Flow
Rate Will Also Increase}
    if amd.iloc[0,3] < amd.iloc[167,3] and amd.iloc[0,6] >
amd.iloc[167,6]:
        amd["Delta T"] = amd["Delta T"].values[::-1]

    if amd.iloc[0,3] < amd.iloc[167,3] and amd.iloc[0,7] >
amd.iloc[167,7]:
        amd["Liquid Flow Rate"] = amd["Liquid Flow Rate"].values[::-
1]

    # {Save Final Results to CSV file}
    LHSresults = amd.to_csv('LHS_Results.csv', sep = ',', columns=
["Year", "Month", "Day", "Time", "Cloud Amount", "Precipitation
Amount", "Precipitation Intensity",
                                                          "Snow
Depth", "Air Temperature", "Wind Speed", "Delta T", "Liquid Flow
Rate"])

    # {Formatting CSV to 1 Column}
    import csv
    # Read the original CSV file with 10 columns
    input_filename = 'LHS_Results.csv'  # Replace with your file
name
    output_filename = 'LHS_Results_Formatted.csv'  # Name for the
new CSV file

    with open(input_filename, 'r', newline='') as input_file:
        csv_reader = csv.reader(input_file)
        data = list(csv_reader)  # Read all rows

    # Concatenate values from each row into a single column
    single_column_data = []
    for row in data:
        concatenated_row = ';'.join(row)  # Join values in a row
with a comma
        single_column_data.append(concatenated_row)

    # Write the single-column data to a new CSV file
    with open(output_filename, 'w', newline='') as output_file:
        csv_writer = csv.writer(output_file)
        for item in single_column_data:
            csv_writer.writerow([item])
|
|
Code-Explicit.py
|
|
    import csv

    # Define the path for the input CSV file
    input_file = 'LHS_Results.csv'
    # Define the path for the output CSV file
    output_file = 'Intermediate2.csv'

    # Function to add a column to the CSV
    def add_column_to_csv(input_file, output_file,
new_column_header, new_column_data):
        with open(input_file, 'r') as file:
            # Read the existing CSV file
```

```
            reader = csv.reader(file)
            data = list(reader)

            # Add the new column header to the first row
            data[0].append(new_column_header)

            # Add data to the new column
            for i in range(1, len(data)):  # Start from 1 to skip
the header row
                data[i].append(new_column_data[i - 1])  # Adjust
index to match data

        with open(output_file, 'w', newline='') as file:
            # Write the modified data to a new CSV file
            writer = csv.writer(file)
            writer.writerows(data)


    add_column_to_csv(input_file, output_file, 'FinalSnowDepth',
D_snow)

    # Define the path for the input CSV file
    input_file = 'Intermediate2.csv'
    # Define the path for the output CSV file
    output_file = 'Intermediate3.csv'

    # Function to add a column to the CSV
    def add_column_to_csv(input_file, output_file,
new_column_header, new_column_data):
        with open(input_file, 'r') as file:
            # Read the existing CSV file
            reader = csv.reader(file)
            data = list(reader)

            # Add the new column header to the first row
            data[0].append(new_column_header)

            # Add data to the new column
            for i in range(1, len(data)):  # Start from 1 to skip
the header row
                data[i].append(new_column_data[i - 1])  # Adjust
index to match data

        with open(output_file, 'w', newline='') as file:
            # Write the modified data to a new CSV file
            writer = csv.writer(file)
            writer.writerows(data)

    add_column_to_csv(input_file, output_file, 'EnergyConsumption',
energyConsumption)
    df = pd.read_csv('Intermediate3.csv')

    import numpy
    last_line = df.take([166])
    dcdata= pd.DataFrame(last_line)

    dcdata.to_csv('Decision_Tree_Inputs_3000.csv', mode='a',
index=False, header=False)

    i=i+1
end = timer()
print('Runtime [sec]: ')
print(end - start)
```

7.7 Appendix F – Python Code: *Full_Scale_Network.py*

```
# {Time Delay Simulation}
# {Import Necessary Modules}
import pandapipes as pp
import pandapipes.properties.fluids as plg
import numpy as np
import pandas as pd
import pandapower.control as control
from pandapower.timeseries import DFData
from pandapower.timeseries import OutputWriter
from pandapipes.timeseries import run_timeseries
import itertools
from itertools import product
import csv

# {Create an empty network}
net = pp.create_empty_network(fluid = "water") # Test the system
with water, then correct fluid properties can be added.

# {Changing FLuid Properties to Create an Incompressible Fluid}
# {Creating Properties}
density_constant = plg.FluidPropertyConstant(1025)            #
kg/m3 - 38% 1,2-Propylenglycol C3H6(OH)2 at 40C, see
https://detector-cooling.web.cern.ch/data/Table%208-3-1.htm
viscosity_constant = plg.FluidPropertyConstant(0.00226)       #
Value taken from CERN for 38% solution (https://detector-
cooling.web.cern.ch/data/Table%208-3-1.htm)
heat_capacity_constant = plg.FluidPropertyConstant(3820)      #
Value taken from CERN for 38% solution (https://detector-
cooling.web.cern.ch/data/Table%208-3-1.htm)

# {Adding Properties to the Existing Fluid}
water_const = plg.Fluid("water_const", "liquid")
water_const.add_property(property_name = "density", prop =
density_constant, overwrite=True)
water_const.add_property(property_name = "viscosity", prop =
viscosity_constant, overwrite=True)
water_const.add_property(property_name = "heat_capacity", prop =
heat_capacity_constant, overwrite=True)

# {Add New Fluid to Net}
plg._add_fluid_to_net(net, water_const)

# {Create Elements: junctions, external grid connection, pipes,
valves, sources, heat exchangers and sinks}
# {Junctions}
inlet = pp.create_junctions(net, nr_junctions=200, pn_bar=1,
tfluid_k=275, geodata=list(itertools.product(range(0,1),
range(0,200))))
outlet = pp.create_junctions(net, nr_junctions=200, pn_bar=1,
tfluid_k=275, geodata=list(itertools.product(range(100,101),
range(200,0,-1))))

# {Pipes}
# {Inlet Side}
pipes_inlet = pp.create_pipes_from_parameters(net,
from_junctions=np.arange(0,199), to_junctions=np.arange(1,200),
length_km=4.658e-4, diameter_m=0.159, k_mm=0.0035, sections=5,
alpha_w_per_m2k=10, text_k=274)

# {Loops}
```

```
loops = pp.create_pipes_from_parameters(net,
from_junctions=np.arange(1,200), to_junctions=np.arange(398,199,-1),
length_km=0.142, diameter_m=0.022, k_mm=0.0035,
loss_coefficient=0.23, sections=5, alpha_w_per_m2k=10, text_k=274)

# {Outlet Side}
pipes_outlet = pp.create_pipes_from_parameters(net,
from_junctions=np.arange(200,399), to_junctions=np.arange(201,400),
length_km=4.658e-4, diameter_m=0.159, k_mm=0.0035, sections=5,
alpha_w_per_m2k=10, text_k=274)

results = []
time = []

tstep = 0
while tstep < 8:

# {Pump}
      pp.create_circ_pump_const_mass_flow(net, return_junction=399,
flow_junction=0, p_flow_bar=3.6, mdot_flow_kg_per_s=1, t_flow_k=289)

# {Run Simulation}
      pp.pipeflow(net, stop_condition="tol", iter=10,
ambient_temperature = 274, friction_model="colebrook", mode="all",
transient=False, nonlinear_method="automatic", tol_p=1e-4, tol_v=1e-
4)

# {Displaying Results for Different Components}
      print(' ')
      print('Time:', tstep, 'Hours')
      #print(net.res_junction)
      res= pd.DataFrame(data=net.res_junction)
      p_bar = res['p_bar']
      t_k = res['t_k']

# {Record Results over Loop Iterations}
      results.append(p_bar)
      results.append(t_k)
      #results.append(res['p_bar'].iloc[399])

# {Setting up Time-Step Calculations}
      time_steps = pd.timedelta_range(0, periods=8, freq='H') #
Select Number of Time Steps
# {Collecting Outputs from Time-Step Calculations}
      log_variables = [('res_junction', 'p_bar'), ('res_junction',
't_k')]
      ow = OutputWriter(net, time_steps, output_path=None,
log_variables=log_variables)
      run_timeseries(net, time_steps)
      time.append(tstep)
      tstep = tstep + 1

# {Create Data Frame from Results}
# results= np.transpose(results)
P_T = pd.DataFrame(results)
P_T = P_T.transpose()

# 1) {Plot of the Pipe Network}
import pandapipes.plotting as plot
plot.simple_plot(net, pipe_width=1.0, junction_size=0.5,
pump_size=0.7)

# 2a) {Interception between horizontal line y=280.15 and Junction
399 to detrmine time needed for main 'code'.}
```

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def displayPoint(self, p):
        f"({p.x}, {p.y})"
        print(f"({p.x}, {p.y})")
def lineLineIntersection(A, B, C, D):
    # {Line AB represented as a1x + b1y = c1}
    a1 = B.y - A.y
    b1 = A.x - B.x
    c1 = a1*(A.x) + b1*(A.y)
    # {Line CD represented as a2x + b2y = c2}
    a2 = D.y - C.y
    b2 = C.x - D.x
    c2 = a2*(C.x) + b2*(C.y)
    determinant = a1*b2 - a2*b1
    x = (b2*c1 - b1*c2)/determinant
    y = (a1*c2 - a2*c1)/determinant
    return Point(x, y)

# {Inputs}:
A = Point(time[0], 280.15)  # Horizontal line at temp=280.15K
B = Point(time[-1], 280.15) # Horizontal line at temp=280.15K
C = Point(time[-2], P_T['t_k'].iloc[200, -2])   # Taken from last
junction and outputs at second last time step
D = Point(time[-1], P_T['t_k'].iloc[200, -1])   # Taken from last
junction and outputs at last time step

# {Outputs}:
intersection = lineLineIntersection(A, B, C, D)
# print("The intersection of the given lines" + "is: ")
intersection.displayPoint(intersection)
xx = intersection.x  # x coordinate from output
print('')
print('Time When Last Junction Reaches Required Temperature:',
round(xx, 2), '[Hours]')

# 2b) Plot of Temperature Variation Over Time at Last Junction
import matplotlib.pyplot as plt
timesteps = time_steps
x = time_steps
y399 = P_T['t_k'].iloc[200]           # Choosing Junction To Plot
fig = plt.figure(figsize = (8,8))
plt.xlabel("Time Steps [Hours]")
plt.ylabel("Temperature [K]")
plt.title("Temperature Variation Over Time at Exit Junction")
from matplotlib.ticker import StrMethodFormatter
plt.gca().yaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
# No decimal places
plt.gca().yaxis.set_major_formatter(StrMethodFormatter('{x:,.2f}'))
# 2 decimal places
plt.plot((x/3.6e+12), y399, "-o", color='deeppink')
plt.axhline(y = 280.15, color = 'grey', linestyle = '--') # y=7
degree C
plt.axvline(x = xx, color = 'grey', label = 'axvline - full height',
linestyle = '--')
plt.legend(['Exit Junction'], bbox_to_anchor=(1.05, 1.0), loc='upper
left', prop = { "size": 10 })
plt.grid(which='major', color='#DDDDDD', linewidth=0.8)
plt.grid(which='minor', color='#EEEEEE', linestyle=':',
linewidth=0.5)
plt.minorticks_on()
```

7.8 Appendix G – Python Code: Original

```
#                          TURFHEAT - v/1.0                          #
# (C) 2023 COMPUTATIONAL ENGINEERING AND ANALYSIS RESEARCH GROUP     #
#       TURKU UNIVERSITY OF APPLIED SCIENCES, FINLAND                #
# ------------------------------------------------------------------ #
# Use this file to input variable values for calculation             #
# ------------------------------------------------------------------ #

#[MAIN DIMENSIONS]
BottomSectionThickness = 0.06 # meters -> heating is at the middle
of this section, at 19 cm depth
MidSectionThickness = 0.105 # meters
SurfaceSectionThickness = 0.055 # meters
BedTotalDepth = 2 # meters, bottom level of blast furnace + clay
layer
FieldWidth = 71 # meters, per Google Maps
FieldLength = 107 # meters, per Google Maps

#[3-LAYER (TOP, MID, BOTTOM) TURF FIELD MATERIAL PROPERTIES]
ThermalElementVoidFraction = 0.2618 # the percentage of thermal
cables relative to total volume of the bottom section (ratio of
cross section areas); the remaining (1-ThermalElementVoidFraction)
is filled with gravel
GravelSectionDensity = 1690 # kg/m3
https://en.wikipedia.org/wiki/Gravel#:~:text=The%20bulk%20density%20
of%20gravel,3%2C240%20lb%2Fcu%20yd).
GravelThermalConductivity = 0.36 # W/m/K, "Gravel"
https://help.iesve.com/ve2021/table_6_thermal_conductivity__specific
_heat_capacity_and_density.htm
GravelSectionCp = 840 # J/kg/K
https://help.iesve.com/ve2021/table_6_thermal_conductivity__specific
_heat_capacity_and_density.htm
TurfSectionDensity = 1034.3 # kg/m3 - assumption based on plastics
https://omnexus.specialchem.com/polymer-
properties/properties/density
TurfSectionCp = 1289.2 # J/kg/K, estimate (foam plastic)
https://www.engineeringtoolbox.com/specific-heat-solids-d_154.html
TurfSectionThermalConductivity = 0.4548 # W/m/K, estimate
https://ctherm.com/resources/newsroom/blog/the-thermal-conductivity-
of-unfilled-plastics/
TurfSurfaceEmissivity = 0.6 # Winter Football: "The emissivity of
the surface, which in this case is the artificial pitch made out of
polyethylene has an emissivity of 0.92
BlastFurnaceSlagDensity = 1500
BlastFurnaceSlagThermalConductivity = 0.35
BlastFurnaceSlagCp = 840 # same as gravel --- no better info!

#[HEATING FLUID PARAMETERS]
LiquidDensity = 1025 # kg/m3 - 38% 1,2-Propylenglycol C3H6(OH)2 at
40C, see https://detector-cooling.web.cern.ch/data/Table%208-3-1.htm
LiquidThermalConductivity = 0.433 # W/m/K - 38% 1,2-Propylenglycol
C3H6(OH)2 at 40 degrees centigrade; https://detector-
cooling.web.cern.ch/data/Table%208-3-1.htm
LiquidCp = 3820 # J/kg/K , ibid

#[SNOW MATERIAL PROPERTIES]
EmpiricalGroundDampingFactor = -3.5 # 1/m, "f" value between -2 ...
-7, see Rankinen, K., Karvonen, T., & Butterfield, D. (2004). A
simple model for predicting soil temperature in snow-covered and
seasonally frozen soil: model description and testing. Hydrology and
Earth System Sciences, 8(4), 706-716.
SnowThermalConductivity = 0.2 # W/m/K, Snow thermal conductivity
ranges from 0.024 (kair) to 0.8
```

```
SnowSurfaceEmissivity = 0.3 #
https://www.tandfonline.com/doi/abs/10.1080/01431169308904420 : 0.7
... 0.92
SnowDensity = 150 # kg/m3 # Utah Energy Balance model default,
should be > 150 kg/m3, see Liu, Y., et al. (2019). Exploration of
the Snow Ablation Process in the Semiarid ... Water, 11(5), 1058.
IceDensity = 917 # kg/m3 # Geotop,
http://eprints.biblio.unitn.it/551/1/geotop.pdf, p. 59
SnowSpecificHeatCapacity = 2090 # J/kg/K
https://www.engineeringtoolbox.com/specific-heat-capacity-d_391.html
WaterDensity = 1000 # kg/m3
WaterHeatOfFusion = 333.5 # kJ/kg -- energy required to melt snow,
h_f parameter in Utah Energy Balance Model
WaterSpecificHeatCapacity = 4184 # J/kg/K
SnowSaturatedHydraulicConductivity = 0.0069444 # m/s, Reference 25
m/h is 0.0069444, see Table 3 in Liu, Y., et al. (2019). Exploration
of the Snow Ablation Process in the Semiarid ... Water, 11(5), 1058.
SnowLiquidHoldingCapacity = 0.4 # Utah Energy Balance model default
0.05, see Table 3 in Liu, Y., et al. (2019). Exploration of the Snow
Ablation Process in the Semiarid ... Water, 11(5), 1058.

#[INITIAL VALUES]
FieldBottomTemperature = 0 # C, soil average temperature at 17 cm
depth, at the start of simulations
FieldMidpointTemperature = 0 # C, soil average temperature at 7 cm
depth, at the start of simulations
FieldSurfaceTemperature = 0 # C, turf surface temperature at the
start of simulations
SnowDepth = 0.0 # meters, average depth of snow at the start of
simulations
SnowTemperature = -5 # C, average
SnowLiquidFractionInitial = 0 # 0 = dry snow (no liquid), 1 = liquid
water (no ice/snow)

#[WEATHER FORECAST FILE]
WeatherFile = "training_2020_small.csv" #"weather_data_trendi.csv"
#"weather_data_vakio.csv" #"weather_data_2pts.csv"
#"weather_data_no_snow.csv" # Remember double quotations (") on both
sides; format as in Ilmatieteenlaitos
https://www.ilmatieteenlaitos.fi/havaintojen-lataus, Turku
Artukainen
TimeStepSizeMinutes = 60 # 60 = 1 hour etc; smaller gives better
accuracy and stabiilty but takes longer to execute
```

```
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
from scipy.optimize import fsolve  # required library
import sympy as sp
import scipy
import matplotlib.pyplot as plt

global Sstar_interm
global Sstar_prev


# Arrays

# Initialize solution arrays
X_sol = []
T_1 = []
```

```
T_2 = []
T_3 = []
T_S = []
T_ground_1 = [] # This is the bulk ground temperature at bottom (for
horizontal heat conduction)
T_ground_2 = [] # This is the bulk ground temperature at mid (for
horizontal heat conduction)
T_ground_3 = [] # This is the bulk ground temperature at
masuunihiekka layer
D_snow = [] #
mSnowPackTot = []
LiquidFraction = []
energyConsumption = []
Liquid_dTPrev = 0
Sstar_prev = 0
Sstar_interm = 0
meltinG_exp = 0.1
CloudFactorCorrection = 1
WindFactorCorrection = 1
# Natural constants

StefanBoltzmann = 5.6704*10**(-8) # W/m^2/K^4

def GetConfiguration():

    with open("turfheat.ini") as file:
        lines = [line.rstrip() for line in file]

    for l in lines:
        ll = l.split("#", 1)[0]
        lll = ll.split("[", 1)[0]
        if len(lll)>0:
            exec(lll,globals())

def isnumber(x):
    try:
        float(x)
        return float(x)
    except:
        return np.nan

def ReadCsv(WeatherFile):
    df = pd.read_csv(WeatherFile, sep=';', decimal='.')

    if len(df) == 2: # First and last points given -> apply linear
interpolation
        df["datestrH"] = df["Year"].astype(str) + '-' +
df["Month"].astype(str).str.zfill(2) + '-' +
df["Day"].astype(str).str.zfill(2) + ' ' + df["Time"]
        df["datestr"] = df["Year"].astype(str) + '-' +
df["Month"].astype(str).str.zfill(2) + '-' +
df["Day"].astype(str).str.zfill(2)
        df["date"] = pd.to_datetime(df['datestrH'], format='%Y-%m-%d
%H:%M')
        ddf = df.set_index('date').resample('60T').interpolate()
        ddf["datestr"] = ddf.index.strftime('%Y-%m-%d')
        ddf["datestrH"] = ddf.index.strftime('%Y-%m-%d %H:%M')
        ddf["date"] = ddf.index
        ddf["index"] = range(0,len(ddf))
        df = ddf.set_index("index")
    else:
        df["datestr"] = df["Year"].astype(str) + '-' +
df["Month"].astype(str) + '-' + df["Day"].astype(str)
```

```
        df["datestrH"] = df["Year"].astype(str) + '-' +
df["Month"].astype(str).str.zfill(2) + '-' +
df["Day"].astype(str).str.zfill(2) + ' ' + df["Time"]
        df["date"] = pd.to_datetime(df['datestrH'], format='%Y-%m-%d
%H:%M')

    alkupv = df["datestr"].iloc[0]
    loppupv = df["datestr"].iloc[-1]

    df["AirTemperature"] = pd.to_numeric(df['Air Temperature
(degC)'],errors='coerce')
    df["DeltaT"] = pd.to_numeric(df['DeltaT (C)'],errors='coerce')
    df["LiquidFlowRate"] = pd.to_numeric(df['LiquidFlowRate
(l/s)'],errors='coerce')
    df["WindVelocity"] = pd.to_numeric(df['Wind Speed
(m/s)'],errors='coerce')
    #df["SnowDepth"] = pd.to_numeric(df['Lumensyvyys
(cm)'],errors='coerce')/100 # conversion cm -> m
    #df["SnowDepth"] = [0]*len(df)
    df["SnowDepth"] = pd.to_numeric(df['Snow Depth
(cm)'],errors='coerce')
    #df["SnowDepth"][df["SnowDepth"]<0] = 0
    df.loc[df["SnowDepth"]<0,'SnowDepth'] = 0

    df["Precipitation"] = pd.to_numeric(df['Precipitation Amount
(mm)'],errors='coerce')/1000 # conversion mm -> m
    df["CloudCover"] = pd.to_numeric(df["Cloud Amount
(1/8)"],errors='coerce')/8 # conversion to [0 .. 1] , 0 = no clouds,
1 = fully clouded
    #df["RelativeHumidity"] = pd.to_numeric(df["Suhteellinen kosteus
(%)"], errors='coerce')/100 # conversion to [0 ... 1]
    df["RelativeHumidity"] = [0.5]*len(df)

    # df.groupby(['group_col1', 'group_col2'])['value_col'].mean()
    #df_daily = df.groupby(pd.Grouper(freq='D', key='date')).mean()

    df_daily = df.groupby(['datestr'])[['AirTemperature',
'WindVelocity', 'SnowDepth', 'Precipitation', 'CloudCover',
'RelativeHumidity', 'Month', 'Day', 'DeltaT',
'LiquidFlowRate']].mean()

    df = df[["date", 'AirTemperature', 'WindVelocity', 'SnowDepth',
'Precipitation', 'CloudCover', 'RelativeHumidity', 'Month', 'Day',
'DeltaT', 'LiquidFlowRate']]
    #ddf = df.set_index('date').resample('60T').interpolate()
    interpolaatio_min = str(int(TimeStepSizeMinutes)) + 'T'
    ddf =
df.set_index('date').resample(interpolaatio_min).interpolate()
    #ddf = df.set_index('date').resample('5T').interpolate()

    return ddf, df_daily, alkupv, loppupv

def CalculateGroundTemperature(depth_z, T_air, T_g_prev, k, rho, cp,
f, depth_snow, dt):
    # Rankinen, K., Karvonen, T., & Butterfield, D. (2004).
    # A simple model for predicting soil temperature in snow-covered
and seasonally frozen soil:
    # model description and testing. Hydrology and Earth System
Sciences, 8(4), 706-716.
    # Equations (12)-(13)

    kerr = (dt*k)/(rho*cp*np.power(2*depth_z,2))
    #T = (T_g_prev+kerr*(T_air-T_g_prev))*np.exp(-f*depth_snow)
    T = T_g_prev+kerr*(T_air-T_g_prev)*np.exp(f*depth_snow)
```

```
    return T

def CalculateConvection_h(Plate_x, Plate_y, v):
    # Sartori, E. (2006). Convection coefficient equations for
forced air flow over flat surfaces. Solar Energy, 80(9), 1063-1071.
    # Equation (9)
    L_eff = (4*Plate_x*Plate_y)/(2*Plate_x+2*Plate_y) #
Characteristic length for flat rectangular plate
    h = 5.74*np.power(v, 0.8)/np.power(L_eff,0.2)
    return h

def CalculateSkyTemperature(T_amb, cloudiness_ratio):
    # Oliveti, G., Arcuri, N., & Ruffolo, S. (2003). Experimental
investigation on thermal radiation exchange of horizontal outdoor
surfaces. Building and Environment, 38(1), 83-89.
    # Equation (10)
    # Temperatures in CELSIUS!!
    # k = cloudiness_index/8
    # T_sky = -29 + 1.09*T_amb - 19.9*k
    ClearSkyIndex = (1-cloudiness_ratio)
    T_sky = -29 + 1.09*T_amb -19.9*ClearSkyIndex

    # Whillier model
    #T_sky = T_amb - 6

    return T_sky

def CalculateSkyEmissivity(T_amb, cloudiness_ratio):
    # On the sky temperature models and their influence on buildings
energy performance: A critical review. Luca Evangelisti, Claudia
Guattari, Francesco Asdrubali

    Tamb = T_amb + 273.15 # To Kelvin

    # Clear sky emissivity (Idso and Jackson 1969, Ohio; ibid Table
1)
    eclearsky = 1-(0.261*np.exp(-7.77*0.0001*(273.15-Tamb)*(273.15-
Tamb)))

    # Kasten and Czeplak 1980 ε sky = ε clear-sky + 0 . 8( 1 − ε
clear-sky ) CF; ibid Table 3
    esky = eclearsky + 0.8*(1-eclearsky)*cloudiness_ratio

    return esky

def CalculateGroundEmissivity(snowDepth):
    if snowDepth < 0.001:
        e = TurfSurfaceEmissivity # no snow = dark green
    elif (snowDepth >= 0.0001) & (snowDepth < 0.01):
        scaling_ratio = (snowDepth-0.0001)/(0.01-0.0001) # = 0 if
snowDepth <= 0.001, 1 if snowDepth >= 0.005
        e = scaling_ratio*SnowSurfaceEmissivity+(1-
scaling_ratio)*TurfSurfaceEmissivity
    else:
        e = SnowSurfaceEmissivity # fully covered by white snow

    return e


def InitializeSnow(T,Area,LF):
    # Snow is initially dry at a given temperature
    m_snow = (1-LF)*Area*SnowDepth*SnowDensity
    m_water = LF*Area*SnowDepth*WaterDensity
```

```
    U_snow_latent = -WaterHeatOfFusion*m_snow*1000 # [J]
    U_water_latent = WaterHeatOfFusion*m_water*1000 # [J]
    U_latent = U_water_latent + U_snow_latent

    U_snow_T = m_snow*SnowSpecificHeatCapacity*SnowTemperature #
[J], note that this is negative as SnowTemperature < 0
    U_water_T = m_water*WaterSpecificHeatCapacity*0 # = 0 as Water
in the snowpack is at 0 Celsius
    U_cp = U_snow_T + U_water_T

    # Net
    U_SnowEnergy = U_latent + U_cp

    return U_SnowEnergy


def CalculateSnowAndRainMassFluxes(T_air, Precipitation):
    # Utah Energy Balance Model + U.S. Army Corps of Engineers, 1956
    # Tarboton, D. G., & Luce, C. H. (1996). Utah energy balance
snow accumulation and melt model (UEB) (p. 63). Utah Water Research
Laboratory.
    T_a = 3
    T_b = -1
    if T_air >= T_a:
        Pr = Precipitation # All liquid
        Ps = 0
    elif (T_air < T_a) & (T_air > T_b):
        #Pr = Precipitation*(T_a-T_b)/(T_air-T_b)
        Ps = Precipitation*(T_a-T_air)/(T_a-T_b)
        Pr = (Precipitation-Ps)
    else:
        Ps = Precipitation
        Pr = 0

    return Pr, Ps # [m3/s]

def MeltOutFlowFromSnowPack(LiquidFraction, Area):
    #  Melt outflow is a function of the liquid fraction, using
Darcy's law
    #  Utah Energy Balance Snow Accumulation and Melt Model (UEB)
page 20/64
    #  Ksat is the snow saturated hydraulic conductivity
    #  S* is the relative saturation in excess of water retained by
capillary force, Male and Gray (1981, p. 400, eqn 9.45).

    global Sstar_interm

    K_sat = SnowSaturatedHydraulicConductivity # [m/s]
    Lc = SnowLiquidHoldingCapacity # [-]
    Lf = LiquidFraction # [-]
    rho_i = IceDensity
    rho_s = SnowDensity
    rho_w = WaterDensity

    if (Lf > Lc) & (Lf < 0.99):
        S_star_curr = ((Lf/(1-Lf))-Lc)/((rho_w/rho_s)-(rho_w/rho_i)-
Lc)
    else:
        S_star_curr = 0 # water is retained/embedded/diffused in
snow

    S_star = meltinG_exp*Sstar_prev+(1-meltinG_exp)*S_star_curr

    #M_r = K_sat*S_star^3 # [m/s]
```

```
    M_r = K_sat*S_star*S_star*S_star # [m/s]

    V_out = M_r*Area # [m^3/s] volume flow rate of liquid water out
of snow pack, T = 0C
    m_out = rho_w*V_out # [kg/s]

    Q_out_kW = WaterHeatOfFusion*m_out # [kJ/s] = kW
    Q_out = 1000*Q_out_kW # [W]

    Sstar_interm = S_star

    return m_out, Q_out


def CalculateSolarHeating(MonthNumber, DayN, CloudCover):
    # Merkouriadi, I., Lepparanta, M., & Shirasawa, K. (2013).
Seasonal and annual heat budgets offshore the Hanko Peninsula, Gulf
of Finland. Boreal environment research.
    # Table lookup, Table 4. Sea surface heat balance (W m-2) in
Santala Bay in 2000.
    MonthNumber = round(MonthNumber) # just in case we're
interpolating
    ratio = 1-(1+(DayN-15)/15)/2 # in 1 ... 0 and 0.5 at middle
    cloudFactor = (1-CloudCover)+0.5 # CloudCover = 0 -> No clouds -
> cloudFactor = 1.5 = 150% average


    if MonthNumber == 1: # January
        q = 0.5*(5 + ratio*4 + (1-ratio)*9) # W/m2, assuming 5 W/m2
is on 15th January and interpolation to nearest month averages...
    elif MonthNumber == 2:
        q = 0.5*(9 + ratio*5 + (1-ratio)*53)
    elif MonthNumber == 3:
        q = 0.5*(53 + ratio*9 + (1-ratio)*116)
    elif MonthNumber == 4:
        q = 0.5*(116 + ratio*53 + (1-ratio)*163)
    elif MonthNumber == 5:
        q = 0.5*(163 + ratio*116 + (1-ratio)*153)
    elif MonthNumber == 6:
        q = 0.5*(153 + ratio*163 + (1-ratio)*114)
    elif MonthNumber == 7:
        q = 0.5*(114 + ratio*153 + (1-ratio)*95)
    elif MonthNumber == 8:
        q = 0.5*(95 + ratio*114 + (1-ratio)*58)
    elif MonthNumber == 9:
        q = 0.5*(58 + ratio*95 + (1-ratio)*16)
    elif MonthNumber == 10:
        q = 0.5*(16 + ratio*58 + (1-ratio)*3)
    elif MonthNumber == 11:
        q = 0.5*(3 + ratio*16 + (1-ratio)*4)
    elif MonthNumber == 12:
        q = 0.5*(4 + ratio*3 + (1-ratio)*5)

    return CloudFactorCorrection*q*cloudFactor/24


# ----------------------MAIN CODE : MASS AND ENERGY BALANCES ------
----------

def CalculateSnowFall(T_air, Precipitation, T_snow):
    # T_snow is the snow pack temperature from previous time step
    # Precipitation is in m3/s

    # (1) : Rainfall and snowfall
```

```
    Pr, Ps = CalculateSnowAndRainMassFluxes(T_air, Precipitation) #
[m3/s]

    # (2) : Liquid mass flux and heat flux
    m_liquid_in = Pr*WaterDensity # [kg/s]
    Q_rain = m_liquid_in*WaterSpecificHeatCapacity*(T_air-T_snow) #
[W] <<---- liquid rain TYPICALLY warms up snow, but not necessarily

    # (3) :
    m_snow_in = Ps*SnowDensity # [kg/s]
    Q_snow_latent = -m_snow_in*WaterHeatOfFusion*1000 # [W]
    Q_snow_T = m_snow_in*SnowSpecificHeatCapacity*(T_air-T_snow) #
[W]
    Q_snow = Q_snow_latent + Q_snow_T # [W]

    return m_snow_in, m_liquid_in, Q_snow, Q_rain


def SnowMeltBalance(mTot_prev, U_snow_E_prev, Q_net, T_snow_prev,
T_air, Area, dt, m_snowfall_in, m_rainfall_in, Lf_prev):

    # (1) : Mass of liquid and solid water from previous time
    mWaterPrev = mTot_prev*Lf_prev
    mSnowPrev = mTot_prev*(1-Lf_prev)

    # (2) : Current masses of liquid and solid, without energy
effect, which will have to be resolved (new liquid fraction)
    # Current mass of snow = previous mass + snowfall - snow melted
    mSnowCurr = mSnowPrev + m_snowfall_in*dt

    # Update snowpack temperature based on mass-weighted average
    T_snow_backup = T_snow_prev
    T_snow_prev =
min(0,(mSnowPrev*T_snow_prev+(m_rainfall_in+m_snowfall_in)*T_air)/(m
SnowPrev + (m_snowfall_in + m_rainfall_in)))

    U_melt = mSnowCurr*WaterHeatOfFusion*1000 # J > 0

    # The (negative) heat required to freeze all the water
equivalence at 0C [J]
    # Current mass of water is previous mass of water plus rainfall
    mWaterCurr = mWaterPrev + m_rainfall_in*dt #- m_melt_out*dt #
m_melt_out < 0 by definition

    U_freeze = mWaterCurr*WaterHeatOfFusion*1000 # J

    # Total mass of water (frozen/liquid) in the bed
    mTotCurr = mSnowCurr + mWaterCurr

    # Net change in snow pack energy (positive or negative)
    U_net_in = Q_net*dt # J, as [Q_net] = W

    # Final energy state of snow pack
    U_new = U_snow_E_prev + U_net_in # J


    # Work out new liquid fraction by U_new and incoming snow/water
masses, and the corresponding snowpack temperature

    if U_net_in < -U_freeze: # The entire snowpack is frozen
        U_diff = U_net_in-U_freeze # U_freeze is spent on freezing
the liquid water, U_diff is available for cooling the snowpack
        dT_snow = U_diff/(mTotCurr*SnowSpecificHeatCapacity)
        T_new = T_snow_prev + dT_snow
```

```
        LiquidFrac = 0

    elif (U_net_in >= -U_freeze) & (U_net_in <= 0): # Some but not
all of liquid water is frozen
        T_new = 0 # snow pack is at zero temperature since
multiphase
        m_frozen = min(mWaterCurr,
abs(U_net_in)/(WaterHeatOfFusion*1000)) # J
        mWaterCurr = mWaterCurr - m_frozen
        mSnowCurr = mSnowCurr + m_frozen
        mTotCurr = mSnowCurr + mWaterCurr
        LiquidFrac = mWaterCurr/mTotCurr

    elif (U_net_in > 0) & (U_net_in <= U_melt): # Some but not all
of snow is molten
        T_new = 0 # snow pack is at zero temperature since
multiphase
        m_molten = min(mSnowCurr,
abs(U_net_in)/(WaterHeatOfFusion*1000)) # J
        mWaterCurr = mWaterCurr + m_molten
        mSnowCurr = mSnowCurr - m_molten
        mTotCurr = mSnowCurr + mWaterCurr
        LiquidFrac = mWaterCurr/mTotCurr

    elif (U_net_in > U_melt): # Snowpack fully molten - T > 0,
LiquidFraction = 1

        U_excess = U_net_in - U_melt # Energy left for warming up
fully liquid water
        dT_snow = U_excess/(mTotCurr*SnowSpecificHeatCapacity)
        #T_new = T_snow_prev + dT_snow
        T_new = T_air # no need to update -> no snow!
        LiquidFrac = 1

    # (4) : Change of depth (only by snowfall minus melting, not
liquid inflow)
    mSnowNew = (1-LiquidFrac)*mTotCurr
    D_snow_new = (mSnowNew/SnowDensity)/Area

    return T_new, LiquidFrac, D_snow_new, U_new, mTotCurr


def MassAndEnergyBalanceEquations(x, *data):

    # ----------------- Unpack parameters (data) -------------------
---------
    #A_xy, A_1_sides, A_2_sides, A_3_sides, V_1, V_2, V_3, m_1, m_2,
m_3, rho_1, rho_2,  #12
    #rho_3, cp_1, cp_2, cp_3, k_1, k_2, k_3, dx_12, dx_23, k_12,
k_23, alpha_1, alpha_2, alpha_3, MonthN, CloudCover,  #28
    #WindVelocity, Precipitation, RelativeHumidity, AirTemperature,
U_snow_E, epsi, Tsky,  #35
    #Qsun, h, z_depth_1, T_g_next_1, z_depth_2, T_g_next_2, T1s,
T2s, T3s, TSs, SnowDs, LF_curr,  #47
    #mdotfluid, Tf_in, Tf_out = data #50
    dt, A_xy, A_1_sides, A_2_sides, A_3_sides, V_1, V_2, V_3, m_1,
m_2, m_3, rho_1, rho_2, rho_3, cp_1, cp_2, cp_3, k_1, k_2, k_3,
dx_12, dx_23, k_12, k_23, alpha_1, alpha_2, alpha_3, MonthN,
CloudCover, WindVelocity, Precipitation, RelativeHumidity,
AirTemperature, U_snow_E, epsi, Tsky, Qsun, h, z_depth_1,
T_g_next_1, z_depth_2, T_g_next_2, z_depth_3, T_g_next_3, T1s, T2s,
T3s, TSs, SnowDs, LF_curr, massSnowPack_curr, mdotfluid, Tf_delta =
data
```

```
    (T1e, T2e, T3e, SnowWaterMass_new, Snow_Depth_new,
Liquid_Fraction_new, TSe) = x # Temperatures in layers: bottom (1),
mid (2), turf (3), and

    # ----------------- EXTERNAL LOOP: BOTTOM AND MIDDLE
TEMPERATURES -------

    # Layer 1 - Heating level
    Q12 = (k_12/dx_12)*A_xy*(T1s-T2s)
    eq1 = m_1*cp_1*(T1e-T1s)/dt - mdotfluid*LiquidCp*Tf_delta + Q12
+ (k_1/np.sqrt(np.pi*alpha_1*dt))*(T1s-T_g_next_1)*A_1_sides +
(k_1/np.sqrt(np.pi*alpha_1*dt))*(T1s-T_g_next_3)*A_xy

    # Layer 2 - Gravel fill
    Q23 = (k_23/dx_23)*A_xy*(T2s-T3s)
    eq2 = m_2*cp_2*(T2e-T2s)/dt - Q12 + Q23 +
(k_2/np.sqrt(np.pi*alpha_2*dt))*(T2s-T_g_next_2)*A_2_sides

    # Layer 3 - Turf
    if (SnowDs < 0.0001) & (Precipitation<0.000000001): # No snow,
no rain - no need for internal iterations
        Q34 = 0
        Qcond = 0
        #Qrad = epsi*StefanBoltzmann*((T3s+273.15)^4-
(Tsky+273.15)^4)*A_xy # < 0
        Qrad = -epsi*StefanBoltzmann*((T3s+273.15)**4-
(Tsky+273.15)**4)*A_xy
        Qconv = h*A_xy*(AirTemperature-T3s) # < 0 if T_air < T3e
        QsolarAvg = Qsun*A_xy # > 0
        #LF_curr_target = 0

        eq3 = m_3*cp_3*(T3e-T3s)/dt - Q23 + Q34 - Qconv - Qrad -
QsolarAvg

        #eq4 = 0
        #eq5 = 0
        #eq6 = Liquid_Fraction_new-LF_curr_target
        #eq7 = 0

        eq4 = SnowWaterMass_new - 0
        eq5 = Snow_Depth_new - 0
        eq6 = Liquid_Fraction_new - 0
        eq7 = TSe - 0

        return [eq1, eq2, eq3, eq4, eq5, eq6, eq7]

    # Layer 3 - Turf with snow cover
    D_snow_curr = SnowDs
    dx_34 = (SurfaceSectionThickness+D_snow_curr)/2
    k_34 =
((SurfaceSectionThickness/2)*k_3+(D_snow_curr/2)*SnowThermalConducti
vity)/dx_34
    Q34 = (k_34/dx_34)*A_xy*(T3s-TSe)
    Qrad = 0
    Qconv = 0
    QsolarAvg = 0

    eq3 = m_3*cp_3*(T3e-T3s)/dt - Q23 + Q34 - Qconv - Qrad -
QsolarAvg

    # Layer 4 - Snowpack

    # Heat fluxes to atmosphere
    Qconv = h*A_xy*(AirTemperature-TSs) # < 0 if T_air < T_snow
```

```
    Qrad = -epsi*StefanBoltzmann*((TSs+273.15)**4-
(Tsky+273.15)**4)*A_xy # < 0
    QsolarAvg = Qsun*A_xy # > 0

    # Snowfall and rainfall fluxes
    m_snow_in, m_liquid_in, Q_snow, Q_rain =
CalculateSnowFall(AirTemperature, Precipitation, TSs)

    # Power flow [W] to/from snowpack by heating / cooling
    dQsnow = QsolarAvg + Qconv + Qrad + Q34 #+ Q_snow + Q_rain #+
Q_out

    # Work out thermal balance (new temperature, liquid fraction,
snow depth and internal energy)
    T_new, LiquidFraction_new, D_snow_new, U_snow_new, mTotCurr =
SnowMeltBalance(massSnowPack_curr, U_snow_E, dQsnow, TSs,
AirTemperature, A_xy, dt, m_snow_in, m_liquid_in, LF_curr)

    if (LiquidFraction_new > SnowLiquidHoldingCapacity) &
(D_snow_new < 0.005):
        #if (LiquidFraction_new > 99999999):
        # if rainfall is all liquid, then we assume everything is
transported out of the field during the time step (no water pools)
        mAfterMeltingTot = 0
        D_snow_new_after_melt = 0
        #mAfterMeltingTot = 0.75*SnowWaterMass_new
        #LiquidFraction_new = 0.75*LiquidFraction_new
        #D_snow_new_after_melt = max(0,Snow_Depth_new)
        #D_snow_new_after_melt = (mAfterMeltingTot/SnowDensity)/A_xy

    else:
        mAfterMeltingTot, D_snow_new_after_melt =
CalculateMelting(mTotCurr, LiquidFraction_new, A_xy, dt)

    #sol = [Q12, Q23, Q34, Qcond, Qrad, Qconv, QsolarAvg,
SnowWaterMass_new, Snow_Depth_new, m_snow_in, m_liquid_in,
LiquidFraction_new, mAfterMeltingTot, D_snow_new_after_melt]

    # Update output variables
    eq4 = SnowWaterMass_new-mAfterMeltingTot
    eq5 = Snow_Depth_new - max(0,D_snow_new_after_melt)
    eq6 = Liquid_Fraction_new-LiquidFraction_new
    eq7 = TSe-T_new


    return [eq1, eq2, eq3, eq4, eq5, eq6, eq7]

def CalculateMelting(mTotCurr, LiquidFraction, Area, dt):

    #mSnow = (1-LiquidFraction)*mTotCurr

    m_out, Q_out = MeltOutFlowFromSnowPack(LiquidFraction, Area)

    mSnowInit = (1-LiquidFraction)*mTotCurr
    mWaterInit = LiquidFraction*mTotCurr

    m_out_in_time_step = m_out*dt

    mSnowAfterMeltingTot = max(0, mSnowInit - m_out_in_time_step) #
this is less than mSnowInit

    mTotNew = mSnowAfterMeltingTot + mWaterInit

    LiquidFractionNew = mWaterInit/mTotNew
```

```
    D_snow_new_after_melt = (mTotNew/SnowDensity)/Area

    return mTotNew, D_snow_new_after_melt


# ------------------------------------------------------------------
-

if __name__ == "__main__":

    #global Sstar_interm
    #global Sstar_prev

    GetConfiguration()

    # The below values are from parameter optimization carried out
in Matlab ...
    x_in_0 = [0.21004,0.00012923,0.61776,0.20649,-
0.14678,0.010003,50.0347,992.7121,0.84647,1.4996]

    # ... and this is where the parameters are fed:
    SnowSurfaceEmissivity = x_in_0[0] # 0.28026312 #
https://www.tandfonline.com/doi/abs/10.1080/01431169308904420 : 0.7
... 0.92
    SnowSaturatedHydraulicConductivity = x_in_0[1] #0.00672378 #
m/s, Reference 25 m/h is 0.0069444, see Table 3 in Liu, Y., et al.
(2019). Exploration of the Snow Ablation Process in the Semiarid ...
Water, 11(5), 1058.
    SnowLiquidHoldingCapacity = x_in_0[2] # 0.44091809 # Utah Energy
Balance model default 0.05, see Table 3 in Liu, Y., et al. (2019).
Exploration of the Snow Ablation Process in the Semiarid ... Water,
11(5), 1058.
    TurfSurfaceEmissivity = x_in_0[3] #0.59648589 # Winter Football:
"The emissivity of the surface, which in this case is the artificial
pitch made out of polyethylene has an emissivity of 0.92
    EmpiricalGroundDampingFactor = x_in_0[4] # 1/m, "f" value
between -2 ... -7, see Rankinen, K., Karvonen, T., & Butterfield, D.
(2004). A simple model for predicting soil temperature in snow-
covered and seasonally frozen soil: model description and testing.
Hydrology and Earth System Sciences, 8(4), 706-716.
    meltinG_exp = x_in_0[5]
    SnowDensity = x_in_0[6]
    IceDensity = x_in_0[7]
    CloudFactorCorrection = x_in_0[8]
    WindFactorCorrection = x_in_0[9]

    df, df_daily, alkupv, loppupv = ReadCsv(WeatherFile) # outputs
daily averages
    N_all = len(df)
    N_days = len(df_daily)
    energyConsumption = [0]*len(df) # initialization of energy
consumption


    # ---------------------------------- TIME STEP SIZE ------------
-----
    #dt = 60*60 # time step size seconds
    #dt = 5*60 # time step size seconds
    dt = TimeStepSizeMinutes*60

    # ---------------------- INTERMEDIATE CONSTANTS ----------------
-----
```

```
    # LAYERS: bottom = 3 (gravel+heating), middle = 2 (gravel), 1 =
top surface (turf)

    # Areas
    A_xy = FieldWidth*FieldLength
    A_1_sides = BottomSectionThickness*FieldLength*2 +
BottomSectionThickness*FieldWidth*2
    A_2_sides = MidSectionThickness*FieldLength*2 +
MidSectionThickness*FieldWidth*2
    A_3_sides = SurfaceSectionThickness*FieldLength*2 +
SurfaceSectionThickness*FieldWidth*2

    dataAreas = (dt, A_xy, A_1_sides, A_2_sides, A_3_sides)

    # Volumes
    V_1 = A_xy*BottomSectionThickness
    V_2 = A_xy*MidSectionThickness
    V_3 = A_xy*SurfaceSectionThickness

    dataVolumes = (V_1, V_2, V_3)

    # Masses and densities
    m_1 = ThermalElementVoidFraction*V_1*LiquidDensity+(1-
ThermalElementVoidFraction)*V_1*GravelSectionDensity
    m_2 = V_2*GravelSectionDensity
    m_3 = V_3*TurfSectionDensity
    rho_1 = m_1/V_1
    rho_2 = GravelSectionDensity
    rho_3 = TurfSectionDensity

    dataMrho = (m_1, m_2, m_3, rho_1, rho_2, rho_3)

    # Effective specific heat capacities
    cp_1 = ThermalElementVoidFraction*LiquidCp+(1-
ThermalElementVoidFraction)*GravelSectionCp
    cp_2 = GravelSectionCp
    cp_3 = TurfSectionCp

    # Distance-weighted effective thermal conductivity between
layers
    k_1 = ThermalElementVoidFraction*LiquidThermalConductivity+(1-
ThermalElementVoidFraction)*GravelThermalConductivity
    k_2 = GravelThermalConductivity
    k_3 = TurfSectionThermalConductivity
    dx_12 = (BottomSectionThickness+MidSectionThickness)/2
    dx_23 = (MidSectionThickness+SurfaceSectionThickness)/2
    k_12 =
((BottomSectionThickness/2)*k_1+(MidSectionThickness/2)*k_2)/dx_12
    k_23 =
((MidSectionThickness/2)*k_2+(SurfaceSectionThickness/2)*k_3)/dx_23

    # Thermal diffusivities (alpha)
    alpha_1 = k_1/(rho_1*cp_1)
    alpha_2 = k_2/(rho_2*cp_2)
    alpha_3 = k_3/(rho_3*cp_3)

    dataCpKAlpha = (cp_1, cp_2, cp_3, k_1, k_2, k_3, dx_12, dx_23,
k_12, k_23, alpha_1, alpha_2, alpha_3)

    # Initialize solution arrays
    T_1 = [FieldBottomTemperature]
    T_1_curr = FieldBottomTemperature
    T_2 = [FieldMidpointTemperature]
    T_2_curr = FieldMidpointTemperature
```

```
    T_3 = [FieldSurfaceTemperature]
    T_3_curr = FieldSurfaceTemperature
    T_S = [SnowTemperature]
    T_S_curr = SnowTemperature
    T_ground_1 = [FieldBottomTemperature] # This is the bulk ground
temperature at bottom (for horizontal heat conduction)
    T_g_curr_1 = FieldBottomTemperature
    T_ground_2 = [FieldMidpointTemperature] # This is the bulk
ground temperature at mid (for horizontal heat conduction)
    T_g_curr_2 = FieldMidpointTemperature
    T_ground_3 = [FieldBottomTemperature] # This is the bulk ground
temperature at masuunihiekka layer
    T_g_curr_3 = FieldBottomTemperature
    D_snow = [SnowDepth] #
    D_snow_curr = SnowDepth
    mSnowPack_curr = A_xy*SnowDepth*SnowDensity
    mSnowPackTot = [mSnowPack_curr]
    LiquidFraction = [SnowLiquidFractionInitial]
    LF_curr = SnowLiquidFractionInitial
    U_snow_E = InitializeSnow(SnowTemperature, A_xy, LF_curr)
    dTControl = [0]

    # Iterate
    tstep = 1
    t = 0
    while tstep < N_all:

        # (1) : Get current boundary conditions from weather data
        MonthN = df["Month"].iloc[tstep]
        DayN = df["Day"].iloc[tstep]
        CloudCover = float(df['CloudCover'].iloc[tstep])
        WindVelocity = float(df['WindVelocity'].iloc[tstep]) # m/s
        PrecipitationPerm2Perdt =
float(df['Precipitation'].iloc[tstep]) # meters of rain per m^2 area
during hour (Ilmatieteenlaitos)
        Precipitation = PrecipitationPerm2Perdt*A_xy/3600 # m3/s on
the entire field area
        RelativeHumidity = float(df['RelativeHumidity'].iloc[tstep])
        AirTemperature = float(df['AirTemperature'].iloc[tstep])
        pvm = df.index[tstep].strftime("%m/%d/%Y, %H:%M:%S")

        # (1.1) : Get control inputs
        LiquidTemperatureDeltaT = df["DeltaT"].iloc[tstep] # degrees
C
        LiquidMassFlowRate =
LiquidDensity*df["LiquidFlowRate"].iloc[tstep]/1000 # [kg/m3]*[m3/s]
= [kg/s]


        dataSnowAir = (MonthN, CloudCover, WindVelocity,
Precipitation, RelativeHumidity, AirTemperature, U_snow_E)

        # (2) : Update parameters

        # (2b) : Ground emissivity
        epsi = CalculateGroundEmissivity(D_snow_curr)

        # (2c) : Sky temperature and emissivity
        Tsky = CalculateSkyTemperature(AirTemperature, CloudCover)
        #esky = CalculateSkyEmissivity(AirTemperature, CloudCover)
        #Tsky = (273.15+AirTemperature)*np.power(esky,0.25)-273.15

        # (2d) : Sun average heating power
        Qsun = CalculateSolarHeating(MonthN, DayN, CloudCover)
```

```
        # (2e) : Rainfall and snowfall (meters per unit time)

        # (2f) : Convection by wind
        h = WindFactorCorrection*CalculateConvection_h(FieldWidth,
FieldLength, WindVelocity)

        # (2h) : Ground continuum (bulk) temperature around the
field
        z_depth_1 = BottomSectionThickness/2 + MidSectionThickness +
SurfaceSectionThickness
        T_g_next_1 = CalculateGroundTemperature(z_depth_1,
AirTemperature, T_g_curr_1, k_1, rho_1, cp_1,
EmpiricalGroundDampingFactor, D_snow_curr, dt)
        z_depth_2 = MidSectionThickness/2 + SurfaceSectionThickness
        T_g_next_2 = CalculateGroundTemperature(z_depth_2,
AirTemperature, T_g_curr_2, k_2, rho_2, cp_2,
EmpiricalGroundDampingFactor, D_snow_curr, dt)
        z_depth_3 = BedTotalDepth/2
        T_g_next_3 = CalculateGroundTemperature(z_depth_3,
AirTemperature, T_g_curr_3, BlastFurnaceSlagThermalConductivity,
BlastFurnaceSlagDensity, GravelSectionCp,
EmpiricalGroundDampingFactor, D_snow_curr, dt)

        dataAmbient = (epsi, Tsky, Qsun, h, z_depth_1, T_g_next_1,
z_depth_2, T_g_next_2, z_depth_3, T_g_next_3)

        # (3) Collect current solution values as the reference to
the next iteration
        dataStart = (T_1_curr, T_2_curr, T_3_curr, T_S_curr,
D_snow_curr, LF_curr, mSnowPack_curr)

        # (4) Collect control inputs
        if (abs(MonthN-1)<0.1) & (DayN > 14):
            Liquid_dT = 15
        elif (abs(MonthN-2)<0.1):
            Liquid_dT = 10
        elif (abs(MonthN-3)<0.1) & (DayN < 16):
            Liquid_dT = 5
        else:
            Liquid_dT = 0

        if (Liquid_dT > 0) & (Precipitation > 0.0001) &
(AirTemperature < 1):
            Liquid_dT = 15

        Liquid_dT = 0
        Liquid_dTPrev = Liquid_dT
        dTControl.append(Liquid_dT)

        dataControl = (LiquidMassFlowRate, Liquid_dT) # <---- UPDATE
THESE IF TIME DEPENDENT CONTROLS!

        data = dataAreas + dataVolumes + dataMrho + dataCpKAlpha +
dataSnowAir + dataAmbient + dataStart + dataControl

        # (4) Work out energy balance and new temperatures
        x_init = (T_1_curr, T_2_curr, T_3_curr, mSnowPack_curr,
D_snow_curr, LF_curr, T_S_curr) # Temperatures in layers: bottom
(1), mid (2), turf (3), and
        amb = (AirTemperature, Qsun, epsi, Tsky, h, T_g_next_1,
T_g_next_2, T_g_next_3, Precipitation)
        X_sol.append((pvm, ) + x_init + amb)
```

```
            x1, x2, x3, x4, x5, x6, x7 =
scipy.optimize.fsolve(MassAndEnergyBalanceEquations, x_init,
args=data)

            Sstar_prev = Sstar_interm

            # (5) Update solutions
            T_g_curr_1 = T_g_next_1
            T_g_curr_2 = T_g_next_2
            T_g_curr_3 = T_g_next_3

            T_1_curr = x1
            T_2_curr = x2
            T_3_curr = x3
            mSnowPack_curr = x4
            D_snow_curr = x5
            LF_curr = x6
            T_S_curr = x7
            T_1.append(x1)
            T_2.append(x2)
            T_3.append(x3)
            mSnowPackTot.append(x4)
            D_snow.append(x5)
            LiquidFraction.append(x6)
            T_S.append(x7)

            # (5.1) Update energy consumption
            energyConsumption_curr =
LiquidMassFlowRate*LiquidCp*Liquid_dT*dt # [kg/s]*[J/kg/C]*[C]*[s] =
[J]
            energyConsumption[tstep-1] = energyConsumption_curr

            # (6) Proceed to next time step
            tstep = tstep + 1
            if (tstep % 1000) == 0:
                print(pvm + " ::: progress " +
str(round(100*tstep/N_all, 2)) + ' %')


d = np.array(D_snow)
d[d<0] = 0
df["d"] = d

fig, axz = plt.subplots(1, 1)
fig.set_size_inches((14, 8), forward=False)
#plt.plot(df["d"])
#plt.plot(df["SnowDepth"]/100)
#plt.show()
l1, = axz.plot(df.index, df["d"], '-', color='black')
l2, = axz.plot(df.index, df["SnowDepth"]/100, color='blue')
axz2 = axz.twinx()
l3, = axz2.plot(df.index, df["Precipitation"].cumsum(), '--',
color='crimson')
plt.legend([l1, l2, l3], ["Snow depth prediction [m]", "Snow depth
measured [m]", "Cumulative rainfall [m]"])

axz.grid()
axz.tick_params(axis='x', rotation=25)
plt.gca().ticklabel_format(axis='y', style='plain')
axz.tick_params(axis='y', colors='black')
axz.set_ylabel('[m]', color='k')
axz2.set_ylabel('[m]', color='crimson')
plt.savefig("Result0.png")
```

```
dff =pd.DataFrame(X_sol,
columns=['Date','T_1','T_2','T_3','SMass','Depth','LiquidF','T_Snow'
,"AirT", "Qsun", "epsi", "Tsky", "h", "T_g_1", "T_g_2", "T_g_3",
"Precipitation"])
dff.to_csv('soln.csv',index=False,sep=';',decimal=',')

sulana = len(np.where(d<0.001)[0])
prossulana = np.round(100*sulana/len(d),1)
#plt.plot(d)
#plt.show()
TotalEnergyConsumption =
np.round(float(np.sum(energyConsumption)/3.6e+9),2)
fig, axs = plt.subplots(1, 1)
fig.set_size_inches((14, 8), forward=False)

#fig.tight_layout()
#plt.xticks(rotation=45, ha='right')
#fig.suptitle(alkupv + ' -- ' + loppupv + ' energiankulutus : ' +
str(TotalEnergyConsumption) + ' MWh, sulana: ' + str(prossulana) + '
% ajasta')
axs.plot(df.index, T_1, label = 'Slag')
axs.plot(df.index, T_2, label = 'Sand')
axs.plot(df.index, T_3, label = 'SBR turf')
axs.plot(df.index, df["AirTemperature"], '--', label = 'Air')
if np.max(d) > 0.0001:
    axs.plot(df.index, T_S, label = 'Snow')
axs.set_title('Temperatures [C]')
#axs[0].legend(loc='upper left')
axs.legend()
axs.grid()
axs.tick_params(axis='x', rotation=25)
axs.set_ylabel('[C]', color='k')
plt.gca().ticklabel_format(axis='y', style='plain')

plt.savefig("Result1.png")

#figManager = plt.get_current_fig_manager()
#figManager.window.showMaximized()

#axs[0,1].set_title('Lumi ja sade')
#prec1 = df["Precipitation"]*1000
#l1, = axs[0,1].plot(df.index, prec1, '--', color='blue')
#ax2 = axs[0,1].twinx()
#l2, = ax2.plot(df.index, 100*np.round(d,4), color='black')
#plt.legend([l1, l2], ["Sademäärä [mm/h]", "Lumen syvyys [cm]"])
#axs[0,1].grid()
#axs[0,1].tick_params(axis='x', rotation=25)
#plt.gca().ticklabel_format(axis='y', style='plain')
#axs[0,1].tick_params(axis='y', colors='black')
#axs[0,1].set_ylabel('[mm/h]', color='b')
#ax2.set_ylabel('[cm]', color='k')

fig, axs = plt.subplots(1, 1)
fig.set_size_inches((14, 8), forward=False)
#axs.set_title('We')
pilvi = np.array(df['CloudCover'])
pilvi[pilvi>1] = 1
l3, = axs.plot(df.index, 100*pilvi, '-', color='crimson')
ax3 = axs.twinx()
tuuli = np.array(df['WindVelocity'])
l4, = ax3.plot(df.index, tuuli, '-', color='black')
plt.legend([l3, l4], ["Cloudiness [%]", "Wind speed [m/s]"])
axs.grid()
```

```
axs.tick_params(axis='x', rotation=25)
axs.set_ylabel('[%]', color='crimson')
ax3.set_ylabel('[m/s]', color='black')
plt.savefig("Result2.png")


#axs[1,1].set_title('Lämmityksen ohjaus')
##deltaT = np.array(df['DeltaT'])
#deltaT = np.array(dTControl)
#fr = np.array(df["LiquidFlowRate"])
#l5, = axs[1,1].plot(df.index, deltaT, color='magenta')
#ax4 = axs[1,1].twinx()
#l6, = ax4.plot(df.index, fr, '--', color='darkgreen')
#plt.legend([l5, l6], ["Lämpötilaero (C)", "Tilavuusvirta [l/s]"])
#axs[1,1].grid()
#axs[1,1].tick_params(axis='x', rotation=25)
##axs[1,1].tick_params(axis='y', colors='magenta')
#axs[1,1].set_ylabel('[C]', color='magenta')
#ax4.set_ylabel('[l/s]', color='darkgreen')

#plt.savefig("Result.png")

#plt.pause(0.5)
#figManager = plt.get_current_fig_manager()
#figManager.window.showMaximized()

#plt.show()

all_done = 1
```

7.9 Appendix H – General Guidance to Project Code

Main Code Simulation

The objective of the code is to predict the accumulation of snow on the field and calculate the subsequent melting and energy consumption.

All of the initial values, such as the main dimensions of the field, material properties of the soil and snow, and heating fluid parameters are defined in the $'turfheat'$ configuration file. This is also where the input weather file and the time step size are controlled. These are important functions when carrying out a sensitivity analysis, where the stability of the prediction for different weather data and time step sizes must be validated.

The $'Future\_Weather.py'$ file is where the main calculations are carried out. The weather parameters are extracted from the CSV file, and formatting and unit conversions are carried out to produce the desired input for the calculation. Functions are defined for the calculations of ground temperature, convection, sky temperature, sky emissivity, ground emissivity and snow and rain mass fluxes. One of the new additions to the code is a set of conditions which control the soil properties, namely density, specific heat capacity and thermal conductivity based on the average water content in the soil. Melt outflow and solar heating are also defined here, providing all the values required in the following calculations.

Snowmelt balance is then solved, providing the change in snow depth at each time step. The mass and energy balance calculations split the equations into distinct sections for the 3 layers of soil – the heating level (1), gravel fill (2), and the turf layer (3), as well the snowpack layer (4) on top. The revised version now includes the evaporation and condensation heat fluxes in the turf and snowpack layers. Some additional control for Tf_delta is also included at the heating level to accommodate for the time delay in the system reaching the required temperature. Control inputs are also present in the following section for working out energy consumption.

The $pandapipes$ package for python was used to model time delay in the network. After an empty network is created, the type and properties of the circulating fluid can be modified. The elements of the network are created starting with junctions, which act as connections between other components. The number of junctions, initial values of pressure and temperature for the calculation must be defined. The geodata parameter provides coordinates for plotting the network. The pipe elements can then be added, connecting the existing junctions. Dimensions, as well as roughness, pressure loss coefficient and ambient temperature are also defined. The network has an inlet and an outlet section which are larger in diameter, connected by a series of smaller diameter loops. A pump connecting the first and the last junction enforces a chosen pressure and temperature at the outlet side while requesting a specific flowrate at the inlet side. The flow can then be simulated for a defined number of time steps, and the resulting temperature and pressure at the junctions are recorded to a pandas dataframe, which

could be used for analysis of the data. The script can automatically generate a schematic of the network and a plot displaying the evolution in temperature at a selected junction over the length of the simulation. The line intersection tool can be used to calculate the time required for the junction to reach the requested temperature.

The final simulation plots can be shown in Figure 77 and Figure 78:

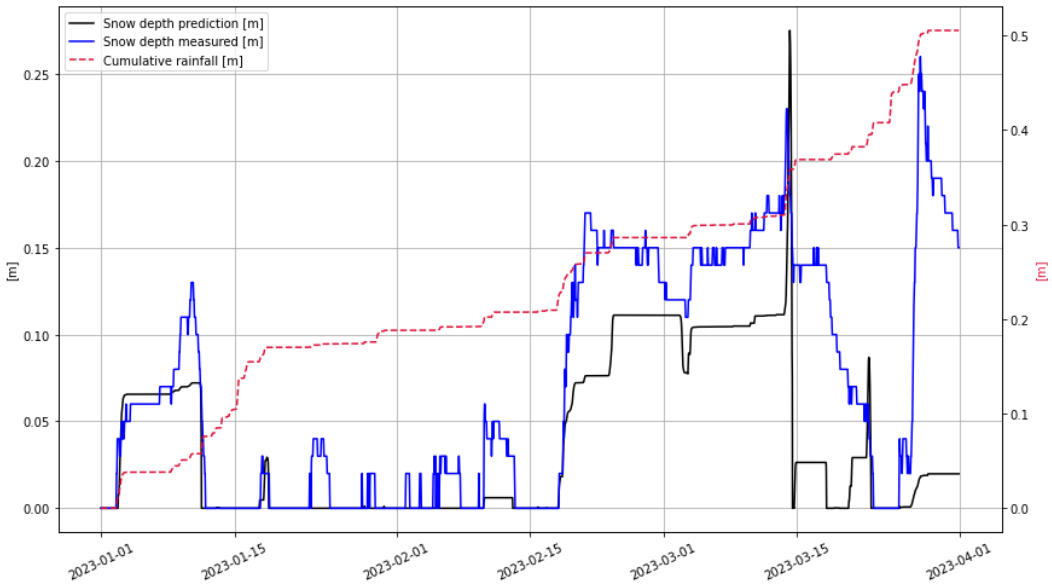


Figure 77: Predicted and Measured Snow Depth

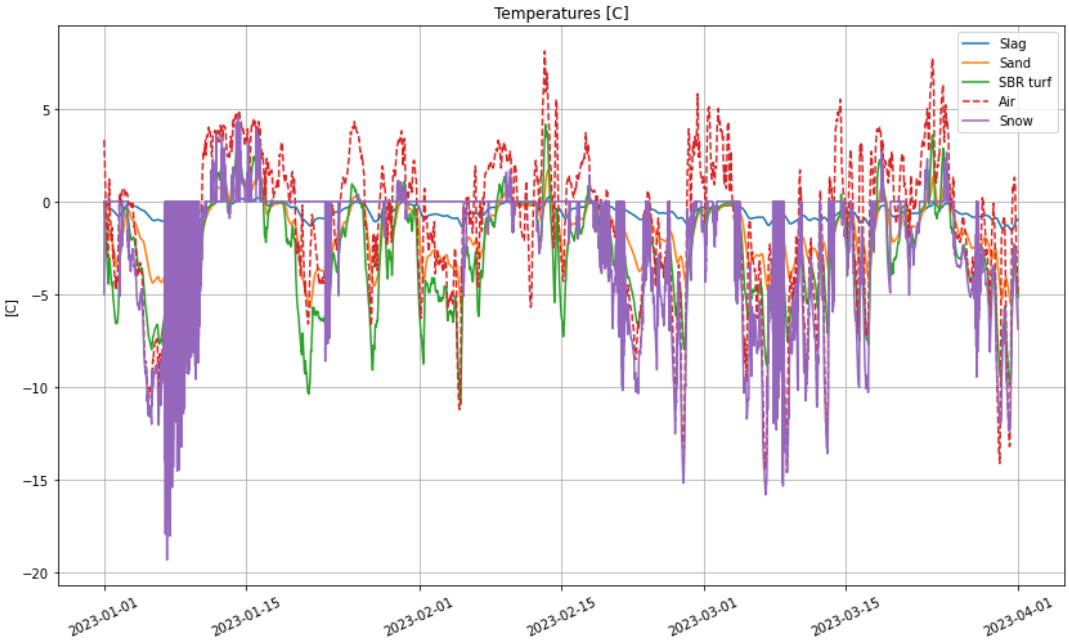


Figure 78: Temperature Variations in Different Layers

Several variables are included for carrying out a sensitivity analysis, and plots are produced showcasing the impact on the predicted snow depth. Annotations are included in the script at all stages to identify the functions of different parts of the code.

Latin Hypercube Sampling (LHS)

The '$LHS\_Loop.py$' file is used to carry out this part of the analysis. A weather file in the CSV format was imported into the code. From this, the maximum and minimum values for each weather parameter could be extracted for any length of time. These values set the limits for the random sampling. The sampling was done so the weather parameters: Cloud Amount $(1/8)$, Precipitation Amount (mm), Precipitation Intensity (mm/h), Snow Depth (cm), Air Temperature (℃) and Wind Speed (m/s) were kept constant. The control inputs: ΔT (℃) and the Liquid Flow Rate (l/s) were also fixed to assess the impact they have on the final snow depth and the energy consumption at the end of a 1-week simulation.

The sampler is initialised using a $'random\_state'$ parameter, which controls the random number generator used. This parameter is used to control the reproducibility of the results. A value of $None$ for the random state will produce a new set of results each time the script is executed. Using an integer instead will allow results to be reproducible over multiple simulations. The most common values are between 0 and 42 and each one will generate a different set of results.

Due to the random nature of the sampling, some additional constraints had to be introduced to accommodate for the creation of unrealistic weather scenarios, such as heavy rainfall when Cloud Amount = 0, or decreasing the control inputs when snow cover is high.

Depending on the type of analysis being carried out, the values generated can be used in their original state, where each cell will be assigned a value sampled randomly from the limits assigned earlier. It is also possible to modify the data, such that all the values in a particular column are constant, or linear interpolation is caried out between the first and the last value. Both options are present in the script and can be assigned to any of the weather parameters and control inputs, or removed altogether, with small changes to the code.

The results are saved to a CSV file, '$LHS\_Results\_Formatted.csv$' and formatted to 1 column so that they could be inputted into the Main Code.

```
Year;Month;Day;Time;Cloud Amount (1/8);Precipitation Amount (mm);Precipitation Intensity (mm/h);Snow Depth (cm);Air Temperature (degC);Wind Speed (m/s);DeltaT (C);LiquidFlowRate (l/s)
2023;1;1;00:00;0;0;0;0;3.3;2.3;0;0
2023;1;1;01:00;1;0;0;0;2.6;2.7;0;0
2023;1;1;02:00;4;0;0;0;1.7;1.9;0;0
2023;1;1;03:00;7;0;0;0;0.8;1.6;0;0
2023;1;1;04:00;6;0;0;0;-0.9;0.8;0;0
2023;1;1;05:00;7;0;0;0;-1;1.6;0;0
2023;1;1;06:00;7;0;0;0;-1.1;1.8;0;0
2023;1;1;07:00;7;0;0;0;-2.2;1.5;0;0
2023;1;1;08:00;4;0;0;0;-2.3;2;0;0
```

Figure 79: Formatted Weather Data

The script could be run for any number of iterations, and each time, a new set of randomly sampled weather parameters and control inputs are generated, which could be used to obtain the associated values of snow depth and energy consumption at the end of the simulation.

Decision Tree Model

The results file produced at the end of the Design of Experiments had to then be processed before it could be implemented in the Regression Decision Tree Model. The output file, taken from the '$LHS\_Loop.py$' script includes all 6 weather parameters and the two control inputs, alongside the addition of the final snow depth and energy consumption. This can be found in the $'Decision\_Tree\_Input3000..csv'$ file. The first requirement for the cases was that the snow depth was below $0.1\ cm$, which was identified as the threshold at which the field could be considered in playing conditions. The other requirement was below average energy consumption, which was averaged out over 3000 simulations, producing a varied set of results which could then be ranked. Cases where the energy consumption was below the average value were assigned a rank of 1, and the rest of the cases were assigned 0. This significantly simplified the training process as ranking the cases on a scale between 0 and 1 provided the algorithm with a clear target variable to optimise for.

Delta T	Liquid Flov	FinalSnow	EnergyConsumption	True/False
6.863773	0.648008	0.101075	63641413.59	1
4.53272	0.09408	0.100098	6086567.283	1
10.44518	0.585169	0.100714	87643337.35	0
13.5946	0.656918	0.100089	126756953.6	0
4.922655	0.273791	0.100607	19288402.6	1
7.662176	0.164474	0.100102	18154762.46	1
1.922939	0.929872	0.107471	25937259.55	1

Figure 80: Implementation of Ranking System

After this analysis was carried out, the $'Decision\_Tree.py'$ script could be run. The weather and control parameters were selected as the features (defined by x) and the column containing the ranking was selected as the target (defined by y). The data is then split up into a training and a testing set, which can be modified using the parameter $'test\_size'$ within the $'train\_test\_split'$ module. This scales between 0 and 1 and defines the amount of data which will be used for the training of the model. The $'random\_state'$ parameter, also included in the decision tree initialiser, is used to control the reproducibility of the results, as described in the Latin Hypercube Sampling section. The resulting decision tree was plotted using $webgraphviz$, and analysis was carried out to select the weather conditions for which the highest amount of melting could be achieved for the smallest energy expenditure. This was done by following the different branches of the decision tree until a predicted value of 1 was reached and tracing back the weather parameters which were selected for that branch.

7.10 Appendix I – Simulation Full Worked Example

To run the main code (which calculates the heat and mass balance and predicts snow depth).

Note: The file named 'Future_Weather.py' contains the main code, and when it runs, it calculates the heat and mass balances and predicts snow depth at each timestep. It outputs results of predicted against observed snow depth, as well as energy use at each timestep.

Main Code: *Future_Weather.py*

Parameter Inputs: *turfheat.ini*

Weather Input File: *training_2023_small.csv*

PandaPipes Input File: *Full_Scale_Network.py*

Open *Future_Weather.py (Full_Scale_Network.py, turfheat.ini* and *training_2023_small.csv* must be saved in same folder as the weather input file is named within the parameter inputs file etc.).

Run code and 5 graphs will be plotted.

Graph 1: Schematic of Pipe Network

Graph 2: Temperature Variation at Exit Junction of Underground Pipes

Graph 3: Snow Depth Measured and Predicted Over Time (Result0)

Graph 4: Various Temperatures Over Time (Result1)

Graph 5: Cloudiness % and Wind Speed Over Time (Result2)

To replicate the results of the sensitivity analysis: parameter study.

Again open *Future_Weather.py.*

On line #681 copy the commented-out variables within the square brackets [0.4].

This by default will run the variables study for Snow Liquid Holding Capacity. Which will produce the original 5 graphs described in 1 with the addition of Graph 3 and graph 4 which will be printed for each subsequent value for snow liquid holding capacity.

For the desired hex graphs to be plotted lines #1099 to #1200 should be commented in.

A variable study for all other variables named in lines #682 to #692 can be done in a similar sense.

All the lines relating to *Snow Liquid Holding Capacity must be commented out and the lines relating to the desired variable commented in.*

Firstly, between line #699 and line #730.

Secondly, between line #928 and line #970.

To produce basic decision tree.

Design of Experiments/LHS: *LHS_Loop.py*

Weather Input File: *Weather_Data_(Not_Formatted).csv*

Parameter Inputs: *turfheatDoE.ini*

Intermediate .csv files produced as part of the *LHS_Loop.py:* *(they are only there to allow the code to run successfully and format the generated weather data, so for the most part can be ignored).*

Intermediate1.csv

Intermediate2.csv

Intermediate3.csv

LHS_Results.csv

LHS_Results_Formatted.csv

LHS Output /Decision Tree Input: *Decision_Tree_Inputs_3000.csv*

This is a $.csv$ file which replicates real life weather data as a result of the Design of Experiments/LHS. The weather data can be imported into all code in the form of a $.csv$ file, given it is formatted correctly. Lines #107 to #127 of *LHS_Loop.py* format the results to the correct format.

Running a file through the $'Weather\_Format.py'$ script will also convert the data to 1 column (the correct format). Ensure also that there is no index column, and that headings are correct. The end results should be similar to:

Year;Month;Day;Time;Cloud Amount (1/8);Precipitation Amount (mm);Precipitation Intensity (mm/h);Snow Depth (cm);Air Temperature (degC);Wind Speed (m/s);DeltaT (C);LiquidFlowRate (l/s)
2023;1;1;00:00;0;0;0;0;3.3;2.3;0;0
2023;1;1;01:00;1;0;0;0;2.6;2.7;0;0
2023;1;1;02:00;4;0;0;0;1.7;1.9;0;0
2023;1;1;03:00;7;0;0;0;0.8;1.6;0;0
2023;1;1;04:00;6;0;0;0;-0.9;0.8;0;0
2023;1;1;05:00;7;0;0;0;-1;1.6;0;0
2023;1;1;06:00;7;0;0;0;-1.1;1.8;0;0
2023;1;1;07:00;7;0;0;0;-2.2;1.5;0;0
2023;1;1;08:00;4;0;0;0;-2.3;2;0;0
2023;1;1;09:00;0;0;0;0;-3;0.7;0;0

Figure 81: Weather Format End Result Example CSV

By running *Decision_Tree.py* a text file named *TreeResults_3000.dot will be produced, the contents which can be copied and pasted into the website:* http://www.webgraphviz.com/ *to generate a decision tree.*

1. **To produce decision tree with complicated machine learning.**

By running *Decision_Tree.py but by commenting in lines #52 to #77 and commenting out lines #16 to #48* a text file named

ComplicatedResults_3000.dot will be produced, the contents which can be copied and pasted into the website: http://www.webgraphviz.com/ to generate a decision tree created from complicated machine learning techniques.

Input Weather File: *Complicated ML section input.csv*

Outputs:

(X_test Values): *Complicated ML X_Test Values.csv*
(Predictions): *Complicated ML Control Predictions.csv*

Lines #80 to #89 should be commented on to replicate results of Cost Analysis.