Hygrothermal performance of embedded wooden beams in solid brick masonry with interior capillary active insulation

R.T.G. Trip
Master Thesis Report
13-01-2025

Hygrothermal performance of embedded wooden beams in solid brick masonry with interior capillary active insulation

Report P5
Master Thesis
January 13, 2025

R.T.G. Trip
5421195

Msc Architecture, Urbanism & Building Sciences
Master track Building Technology

Mentors
Dr. ir. Willem van der Spoel
Environmental & Climate Design

Dr. Alessandra Luna Navarro
Façade Design & Engineering

External Mentor
Ir. Paressa Loussos
ABT

Delegate of the Board of Examiners
Dr. D.F.J. Schraven
Real Estate Management

Delft University of Technology
Faculty of Architecture & the Built Environment
Department of Building Technology

Julianalaan 134
2628 BL Delft
The Netherlands

ABT B.V.
Location Delft

Delftechpark 12
2628 XH Delft
The Netherlands

Abstract

The climate crisis is widely regarded as one of the biggest challenges of our century, driving the European Union to adopt ambitious climate action policies targeting rapid transitions across all sectors, particularly resource-intensive ones like the building industry. The building sector accounts for 40% of Europe's energy consumption and 36% of its greenhouse gas emissions, making it a critical sector for achieving the EU's goal of net-zero emissions by 2050. Despite its potential for improvement, significant amounts of European buildings remain energy inefficient, and only 1% undergo renovations each year. These inefficient buildings consume over 270 kWh/m² annually, suggesting that effective retrofitting could significantly reduce energy use. Insulating external walls alone has the potential to achieve energy savings of up to 60%.

Historic buildings, however, present unique challenges due to the need to preserve their cultural, historical, and aesthetic significance. Internal insulation is often the only viable solution for such structures but brings with it complex hygrothermal risks. Insulating solid masonry walls internally reduces heat flow, lowers the wall's temperature, and increases the likelihood of interstitial condensation and consequently frost damage, wood decay and mould growth. Moreover, the use of vapour-permeable, capillary-active insulation materials introduces additional complexities, as their ability to absorb and redistribute moisture can result in both benefits and risks depending on the interior and exterior boundary conditions.

The aim of this research is to gain insight into the most influential factors affecting the hygrothermal safety of embedded wooden beams in solid brick masonry with interior capillary active insulation. Through literature review, background information on the subject is provided. Creating an understanding of the hygrothermal changes to the solid brick masonry after the application of interior capillary active insulation and associated hygrothermal risks of the wall assembly, with regard to the biological decay of embedded wooden beams. To find the most influential factors on hygrothermal performance, methods to simulate this performance are assessed and compared and possible high influential material properties, such as temperature and moisture, further researched.

The influence of thermal conductivity, water vapour diffusion resistance and moisture retention of the insulation, as well as the water vapour diffusion resistance and moisture retention of brick on the hygrothermal performance is assessed with a Sobol sensitivity analysis. This sensitivity analysis indicates the most influential material properties, and is based on 1536 simulations using Delphin hygrothermal simulation software. Python scripting is used to create the simulation and material files, control the Delphin software and analyse and assess the generated results. To create a better understanding of these results and assess the hygrothermal safety, wood decay potential is calculated and visualised using the post processor of Delphin.

From this research, it can be concluded that the exterior boundary conditions, such as rain load, solar radiation and orientation, are more dominant on the hygrothermal performance of the geometry in comparison with the material properties. When solely analysing material properties, the sensitivity analysis concludes that the moisture retention of brick is the primary influence on hygrothermal performance. The moisture retention of insulation is of secondary importance within the research boundary conditions. Furthermore, the impact of capillary active insulation is studied by using wood decay assessments. Revealing positive contributes to the prevention of wood decay. However, incapable of reducing moisture content to levels that prevent wood decay completely.

Acknowledgement

After nine months of hard work, this master thesis marks the end of my studies at Delft University of Technology. Named: '*Hygrothermal performance of embedded wooden beams in solid brick masonry with interior capillary active insulation'*, it is the final product of my master Architecture, Urbanism & Building Sciences, track Building Technology. Many individuals have contributed to the development of this final graduation project. Therefore, I would like to express my gratitude to them.

First of all, Willem and Alessandra. Your guidance has been an invaluable contribution to the successful completion of this master thesis. The meetings, discussions, explanations and feedback really supported me to keep improving this thesis to a higher level. Willem, thank you for patiently answering all my questions, being willing to spend hours explaining everything there is to know on the complex field of hygrothermal behaviour and reviewing my work to such detail. Alessandra, thank you for navigating me through the all challenges in scientific research. For keeping me on track and critical thoughts throughout the process.

I would like to thank ABT for the opportunity to conduct my research as graduation intern. Paressa, thank you for your guidance over the past few months, assistance where needed and provided feedback. Special thanks to ABT Wassenaar. Martijn and Ronald, thank you for introducing me to the company and subsequently bringing me into contact with ABT in Delft.

Additionally, I would like to thank everyone who has been part of this journey over the last few years. Where it all started at the faculty of Civil Engineering. Followed with my time at architecture firm cepezed. Thank you, Wiebe, Ronald and Jan. For letting me start with almost no experience in the first place, answering all my questions and fuelling my enthusiasm for architecture and building technology with your one of a kind view on the creation of buildings.

I would like to thank the guys from SkiMachine. Hans, Rob and Michiel, this 'side job' once a week has been a day where I looked forward to every time. Creating this awesome product, assisting in the workshop, servicing machines for clients and the almost weekly transport of freshly powder-coated steel are certainly memorable.

I would like to thank my friends from both the master and pre-master. Britt, Nikki, Sander and Sarah, for our weekly meetings at the Bouwpub. Talking about courses, discussing exams, celebrating achieved goals or just some simple banter. I really enjoyed those weekly Tuesday moments. Of course, starting at 17:00 sharp.

I would like to thank my friends from cycling association WTOS. The weekly training sessions, commissions, activities and vacations contributed to an amazing time beside studies. These hours and hours of riding around on the bike where an outstanding moment to relax, laugh and recharge.

I would like thank my friends from 'thuis thuis'. Although the distance to the 'east' is a little too far for frequent visits, I have always been welcomed with open arms. I therefore really appreciate your continued interest in studies, new projects and life in general.

I specially would like thank current and old roommates from 'Huise J2'. During my time in Delft, I had the privilege of living with you guys. I am truly grateful for all the activities, holidays, talks, movie nights and much more. Saving me a plate when I was late for dinner, not complaining about the extensive bicycle parking in the hallway and creating a place where I really enjoyed coming home to for the past four and a half years.

And last but not least. Pap, mam and Inge. Your continued interest, encouragement and motivating words when needed, but most of all, your unlimited support and belief, contributed immensely to the point where I am right now. I sometime take this for granted, but is appreciated tremendously.

Table of Contents

1 Introduction

The climate crisis is now considered the defining challenge of our century, motivating the EU to implement an ambitious climate action policy that demands rapid transitions across all sectors. Particularly resource-intensive ones like the building sector (Posani et al., 2021). The building sector is responsible for 40% of Europe's energy consumption and 36% of its greenhouse gas emissions (Tzeiranak et al., 2020), offering significant potential for reducing energy use and meeting the European Commission's goal of net-zero emissions by 2050.

Currently, 75% of European buildings are energy inefficient, and only 1% are renovated annually (Posani et al., 2021). These buildings consume over 270 kWh/m² per year, suggesting that effective renovations could greatly reduce overall energy consumption (Economidou et al., 2011). Insulating external walls can achieve energy savings of around 60% (Harrestrup & Svendsen, 2016). However, historic buildings often require interior post-insulation to preserve their cultural, historical and aesthetic value (Vereecken et al., 2015).

From a building physics perspective, internal post-insulation of solid masonry walls is considered problematic because it reduces the heat flow to the existing wall, lowering the temperature gradient and making the wall colder. This situation increases the risk of interstitial condensation and moisture-induced damage, such as frost damage and mould growth (Jensen et al., 2023). Additionally, there is growing interest in the use of biobased building materials. Many biobased insulation materials are highly permeable to vapour diffusion, with no need for a vapour barrier. This allows indoor moisture to diffuse through the insulation and potentially condense on the cold exterior wall, creating risks of biodegradation and mould growth at high relative humidity levels (Jensen et al., 2023).

In building insulation, three different systems can be identified: vapour-closed, vapour-open, and capillary-active insulation (Stappers, 2019). The vapour-closed system is commonly used in modern building envelopes, employing vapour barriers and materials with high water vapour diffusion resistance to minimize water vapour transport due to differences in vapour pressure and absolute humidity levels on both sides of the wall (Hutkai & Katunský, 2021). Older systems rely on vapour-open construction, allowing vapour to travel through materials. This design, aided by solar radiation and the warm indoor environment, keeps the wall relatively warm, reducing risks like frost damage, mould growth due to interstitial condensation (Jensen et al., 2023).

New biobased materials are both vapour-open and capillary-active. Capillary activity refers to a material's ability to absorb liquid water due to capillary suction and redistribute it through the material. This property allows moisture transfer from the insulation back towards the interior environment if the inner surface of the existing wall becomes wet due to condensation. The liquid water then evaporates on the inside surface, maintaining low relative humidity levels within the insulation package (Hutkai & Katunský, 2021).

Multi-story monumental buildings often have solid masonry brick facades with wooden floors, where floor beams are embedded in the solid masonry. Interior insulation can lead to internal condensation and decay of the wooden beam ends. The thermal bridge created by the interruption of the insulation can worsen this problem. Improper sealing can allow warm moist air to flow alongside the construction, condensing around the beam ends. But when applied correctly, capillary active insulation can absorb the liquid water surrounding the beam ends. And transfer it back to the interior side of the new insulation where is evaporates again, maintaining acceptable relative humidity levels around the beam heads (Harrestrup & Svendsen, 2016).

In conclusion, capillary-active interior insulation systems hold significant potential. Insulating old buildings can reduce energy consumption, allowing these buildings to be used more effectively and comfortably, reducing the need for new constructions and, consequently, CO2 emissions. While still preserving the cultural, historical, and aesthetic value.

However, there are risks that require thorough research. Previous tests have identified potential problems due to high humidity levels caused by interstitial condensation and the vapour permeability of the materials. This report aims to gain more insights into the key parameters affecting hygrothermal safety of embedded wooden beams in solid brick masonry when interior capillary active insulation is applied, contributing to the restoration of monumental buildings and the reduction of total energy use and greenhouse gas emissions in line with the 2050 European goals.

1.1 Problem statement

The main problem of this research report:

There is an increased need for accurate prediction of hygrothermal safety of embedded wooden beams in solid brick masonry when interior, capillary active insulation is applied.

1.2 Objectives

The main objective of this research report:

To provide insight into the key parameters effecting the hygrothermal safety of embedded wooden beams in solid brick masonry when interior capillary active insulation is applied

Within the research scope, multiple sub-objectives are set to achieve the main objective. These sub-objectives are:

1. Providing insight into the hygrothermal effect of interior capillary active insulation on wooden beam ends embedded in solid brick masonry.

2. Identify and evaluate simulation methods to calculate the performance of complex hygrothermal building components.

3. Provide the key input parameters that affect the hygrothermal performance of embedded wooden beams in solid brick masonry with internal capillary active insulation.

4. Visualise hygrothermal performance to better understand and convey hygrothermal safety.

The final product of this report are recommendations and methodologies to predict hygrothermal safety of building components with the use of hygrothermal simulation software. Additionally, these results are visualised to create a better understanding of hygrothermal performance.

An insight is given into the key factors, the uncertainties and limitations that effect these calculations. Altogether ensuring the preferred internal insulation of monumental solid masonry walls with embedded wooden beams to minimized energy consumption and restore building in a way that has the lowest possible environmental impact.

1.3 Research questions

The objectives will be reached by answering the main research question of this research report:

What influences the hygrothermal safety of embedded wooden beam in solid brick masonry with interior capillary active insulation most and how to mitigate the risk of wood decay?

The main research question is answered by answering sub-questions and associated background questions. These questions are:

1. How does interior capillary active insulation impact the hygrothermal performance of embedded wooden beams in solid brick masonry?

1.1 How is the solid brick masonry impacted by capillary active insulation?

1.2 What are the hygrothermal risks associated with the interior insulation of solid brick masonry when focussing on embedded wooden beams?

1.3 What key factors effect calculating the hygrothermal performance of building components?

2. What methods are available to calculate the performance of complex hygrothermal building components and what simulations can be used to find the most comprehensive, quick and accurate results?

2.1 What methods can be identified and used for complex calculations?

2.2 What are uncertainties, limitations and missing parts in the simulation methods used?

3. What are the key parameters that affect the hygrothermal simulation of embedded wooden beams in solid brick masonry with interior capillary active insulation most and how can these be used to designate preferred insulation measures?

3.1. What influences the duration and accuracy of hygrothermal performance calculations?

3.2. What are the key parameters affecting hygrothermal simulations and corresponding safety and risk assessment?

3.3 What is the optimal simulation method for the most complete, accurate and quick calculations on hygrothermal performance and risk assessment?

4. How can the hygrothermal performance of a building component be visualised to create a better understanding of the generated results and hygrothermal risks?

4.1 What generated data will create the best visual understanding of the results from hygrothermal simulations?

4.2 How can the data be visualised in a simple but comprehensive way to assess hygrothermal safety.

1.4 Outline

This research consists out of eight chapters and additional references and appendices. All providing answers to the (sub-)research questions set in this research. The first chapter gives an introduction to the subject of this master thesis. Where the importance of this research is stated and research questions are defined and objectives are set.

The second chapter contains a comprehensive literature research with background information on the hygrothermal performance of solid brick masonry walls with embedded wooden beams after the application of interior, capillary active insulation. Firstly, the basics of hygrothermal performance are explained. The effects of moisture and temperature loads on the building envelope is discussed and the corresponding material properties that influence hygrothermal performance of walls are explained. Historic construction methods, both for solid brick masonry and the wooden beams that are embedded in the walls, are summarised. Building insulation materials give insight into the current standards in building insulation and new, both biobased and capillary active, possibilities in insulation are provided with additional (dis)advantages.

Sub-paragraph four is an in-depth elaboration on the hygrothermal properties of materials. Given their effect on the total amount of moisture in the wall, more knowledge is gathered to understand how temperature and especially moisture is stored and transported through materials. Including the understanding of driving forces, and various states of moisture and materials.

With the reduced wall temperature due to the interior insulation there is a higher possibility of interstitial condensation between the wall layers, resulting in higher moisture loads. The substantial amounts of moisture have a negative impact on parts of the wall assembly. The associated risks, such as mould growth in insulation and wood decay of the wooden beams is provided in subchapter five. Subsequently the simulation of hygrothermal performance is discussed. The available software options are summed and the chosen software program, Delphin, that is used in this research to simulate the performance of the wooden embedded beams in the solid brick masonry walls is further elaborated on.

The final sub-paragraphs of the literature research, seven and eight, focus on current building standards and the sensitivity analysis. For the building standards, information is gathered for both Dutch and German guidelines. More about the Sobol sensitivity analysis process and the theory behind finding the important key parameters is discussed.

Chapter 3 focusses on the input for the material study that is performed. Arranged in the order that is needed to run the simulations for the sensitivity analysis and subsequently visualise wood decay. Starting with the methodology of the research, that is divided into two steps. The first step being the creation of the simulation variations with corresponding template files, material properties that are varied, the sensitivity analysis sample generator and the eventual creation of simulation jobs. The second step focusses on the running of simulations and calculating the result dose values, being able to assess wood decay. Chapter 3 additionally encompasses the choices made for input of the simulations. With for example: geometry dimensions, indoor boundary conditions, locations, outdoor boundary conditions, settings for the sensitivity analysis and materials properties. Both varied materials to research sensitivity and the materials that remain constant within the simulations.

The next chapter, four, gives the results of the material property study. Giving answers to what are the most influential material properties on hygrothermal simulation of interior capillary active insulation on the geometry simulated. Dose visualisations contribute to the findings in the second paragraph of the fourth chapter and visualise these results for better understanding. Additional visualisations give an indication of the differences in material and geometry choices with respect to the expected moisture content in the wall and corresponding wood decay of the embedded wooden beams.

The research concludes with the conclusion and discussion on the current work completed and recommendations for future research.

2 Literature research

This chapter gives background information on the theoretical knowledge that is needed for this master thesis. It is used to create a general, wide but basic theoretical understanding into the different topics that are part of this research.

This research starts with the basics of hygrothermal behaviour, what is and influences this behaviour. More on thermal and moisture loads and material properties is discussed in the first paragraph. Secondly historic construction methods are discussed, followed by possible insulation material that can be used. Creating an understanding of the geometry researched.

With an in-depth explanation of heat and moisture properties of materials more knowledge is gathered on the possible causes of associated risks of indoor insulation. Additionally the calculation of hygrothermal performance is researched, simulation methods are compared and building regulations explained. With possible (dis)advantages and eventual choice of the simulation program used for the performance calculations.

This literature research is finalised with a detailed elaboration on different sensitivity analysis strategies and additional information is gathered on the method chosen.

2.1 Basics of hygrothermal behaviour

This first paragraph gives an overview of what hygrothermal behaviour is. It gives an introduction to the topic and the effect of hygro and thermal loads on building envelopes is discussed. A short introduction to the consequences of interior insulation to the hygrothermal performance of a building façade is given and material properties that are used when assessing hygrothermal performance is further explained.

2.1.1 Definition of hygrothermal behaviour

One part of building physics is the study of the interaction between moisture (hygro), heat (thermal) and air movement between indoor and outdoor environments. This interaction influences the indoor comfort significantly, for example insulation to minimize temperature extremes or by creating breathable wall assemblies to stabilize humidity levels in buildings.

Hall & Casey (2012) give the following definition of hygrothermal behaviour: “The hygrothermal behaviour of a material can be defined as the change in a materials physical properties as a result of the simultaneous absorption, storage and release of both heat and moisture (liquid and/or vapour phase).”, and in addition: “In a building envelope these loads may be thermal or hygric and are experienced on both internal and external aspects.”

Important is the fact that all these changes can occur simultaneously and influences each other. This study focusses on the hygrothermal behaviour of building envelop, and even more specific: the facades consisting of solid masonry brick.

2.1.2 Hygrothermal loads

The research from Künzel & Karagiozis (2010) and Hall & Casey (2012) highlights that the primary function of a building’s façade is the protection of the indoor environment from the exterior environment. With large fluctuation in ambient temperature, relative humidity and partial vapour pressure between both sides, the façade acts as a barrier to minimize the transmittance of these fluctuations. Where the outdoor conditions are highly dependent on the climate and local environment the indoor climate conditions depend on the purpose and operational use of the building. These different indoor and outdoor conditions can be seen as hygrothermal loads on the building envelope.

For the indoor these conditions are mostly set by automatic HVAC (Heating, Ventilation, Air Condition) systems, but occupant behaviour can influences these loads. For example, one household evaporates up to 10 litres of water every day. This additional load must be removed by the ventilation system in order to create a comfortable indoor condition. Often, the use of mechanical equipment cause air pressure differences and thus the flow of air to spread through the building components, changing thermal and moisture conditions. Another load that is important is the initial moisture load. This is the moisture that is trapped during the construction of the building, for example during rainy periods where the building is not yet wind and weather proof, or caused by the use of water for the construction itself. An example is in-situ cast concrete (Künzel & Karagiozis 2010).

Outdoor loads could be precipitation or thermal loads such as solar radiation. Both short and long wave solar radiation, being the direct solar radiation from the sun and re-emitted solar radiation from the earth respectively. In short, the hygrothermal loads on the building façade are summarised by Künzel & Karagiozis (2010) as:

- Short/Long wave radiation loads
- Thermal convection loads

- Thermal conduction loads
- Drying / Evaporation loads
- Vapour diffusion loads

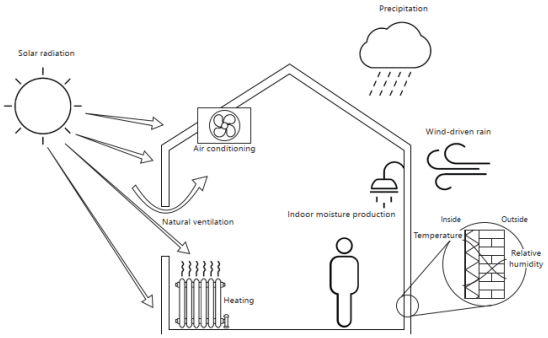


Figure 1: Hygrothermal loads on a building (Hall & Casey, 2012)

All these fluctuations in heat and moisture, act as loads on the entire building envelope, trying to reach equilibrium between the sides. During the day, or even during entire seasons the heat and moisture fluxes due to different temperature and humidity levels change directions. For example, with different flux directions between summer and winter or between day and night. The hygrothermal behaviour of a material is the result of multiple fluxes within the material, being:

- Heat storage within the material
- Heat transfer to and from the material
- Moisture storage within the material
- Moisture transfer to and from the material

Each of the hygrothermal fluxes are interdependent and occur simultaneously (Hall & Casey, 2012). The fluxes and alternating directions across a wall are visualised in Figure 2.

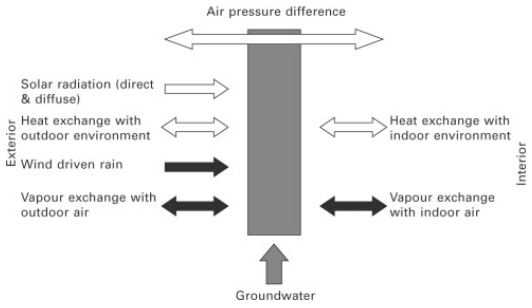


Figure 2: Hygrothermal fluxes on the building envelope (Kunzel, 2005)

Furthermore, the research from Künzel & Karagiozis (2010) mentions the effect of vapour pressure may cause moisture accumulation and drying at different locations. Air pressure differences can assist in the transport of moisture, which in its place can both lead to the accumulation or removal of moisture.

For a wall assembly, Stappers (2021) has created an overview of multiple possible situations, including the use of interior insulation and the effect on the total moisture content in the wall.

In Figure 3 the top view illustrates the application of standard, non-capillary active insulation. When a vapour open insulation layer with a high vapour permeability is applied on the interior side of a solid masonry wall, there is a risk on interstitial condensation. Warm and moist interior air can penetrate the insulation layer due to the fact there is no vapour barrier installed. When reaching the solid brick masonry, that is colder down due to the interior insulation, the air cools down causing the moisture in the air to condense. The condensed water will be absorbed by the masonry and excess condensation will run off. Moisture sensitive building components, such as wooden beam head or wall anchors, are affected with high moisture risks such as wood decay, corrosion or mould growth.

With the application of capillary active insulation, as can be found in the lower part of Figure 3, the condensed water between the insulation and solid masonry is absorbed by the insulation due to the high moisture storage properties of the material. The small pore structure causes the moisture to be transported through the material back to the interior side. When the temperature is higher, relative moisture vapour concentration is low and the stored moisture can evaporate. Reducing moisture in the wall assembly. Interstitial condensation doesn't reach the moisture sensitive building components and the risks on damage is minimized, in theory.

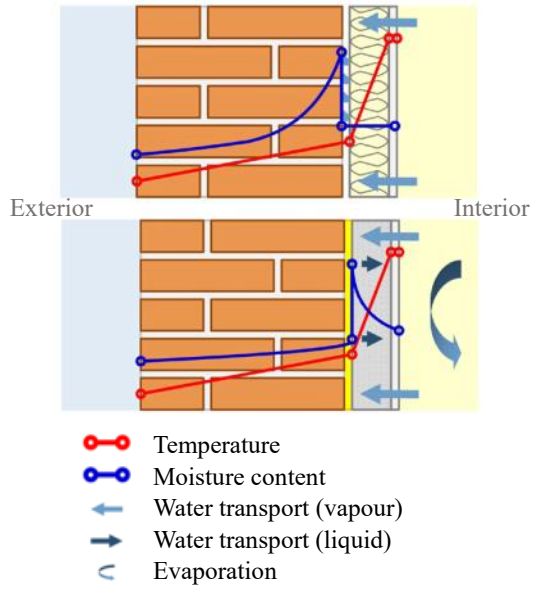


Figure 3: Hygrothermal behaviour of wall assemblies with internal insulation. Top: Standard, bottom: Capillary active insulation. (Stappers, 2021)

2.2 Historic construction methods

This thesis focusses on the hygrothermal performance of buildings constructed with solid brick masonry walls with embedded wooden beams. This paragraph gives an insight into the construction methods used when these buildings were constructed.

Divided into solid brick masonry and the embedded wooden beam, elaborating further on typical wall assemblies, brick sizes, wooden construction in buildings, the connection between brick / masonry and the hygrothermal risks that can be associated with these construction techniques. Hubregtse (2023) extensively summarizes these subjects, the following is based on this reference.

2.2.1 Solid brick masonry

2.2.1.1 Bricks

European historic buildings are commonly constructed out of brick masonry. But, due to different construction techniques, are often quite different from each other. Caused by factors like pattern bonds, sizes of bricks, material choice and construction time. According to Van Hunen (2013) and Stenvert (2013), a distinction can be made between two types of brick masonry: the solid masonry walls and the walls with cavity. Starting with solid masonry walls and changing to cavity walls from 1950. In this report only the solid brick masonry walls are studied and the literature is therefore restricted to this type of masonry wall.

As mentioned previously, brick masonry has changed significantly over time. Starting in the twelfth century with the production of the first bricks and where masonry walls often contained brick and peat stones. Evolving to larger bricks in the thirteenth century and reducing back in size to the fourteenth. Around this century, a difference between brick sizes and geographical use became clear. Where some kept using the larger bricks, other reduced in size. The advantage of these smaller sizes would be the ease in the production process, where smaller bricks could be moulded easier and faster and the baking and drying of bricks took less time. This change is size eventually also resulted in the ideal brick size ratio: 4:2:1 (length * width * height) (Stenvert & Tussenbroek, 2007) (Hunen, 2012)

A distinction can be made between the different brick by using four characteristics: origin, size, hardness and moulding process. The place of origin, that also changes the type of clay and the colour of the brick, is the first characteristic. Size is the second distinction, due to the fact there is no general brick size. Examples of commonly used bricks in the Netherlands are: Waalformaat (210 * 100 * 50 mm) or the Vechtformaat (210 * 100 * 40 mm).

The third characteristic is the hardness of the material. Different baking temperatures have an effect on the hardness of the brick, and uneven temperature levels change the colour and exact properties of the different bricks in one baking process. Typical final temperatures that determine the hardness of the brick:

- 800 – 900 °C: soft red brick
- 900 – 1080 °C: tougher greyer brick
- 1080 – 1125 °C: hard clinker quality

To conclude the last characteristic, changes between hand moulding, machine moulding and strand press moulding (moulding of one long ‘strands’ before it is pressed into individual models). (Stenvert & Tussenbroek, 2007) (Sirag & Wiedijk, 1950)

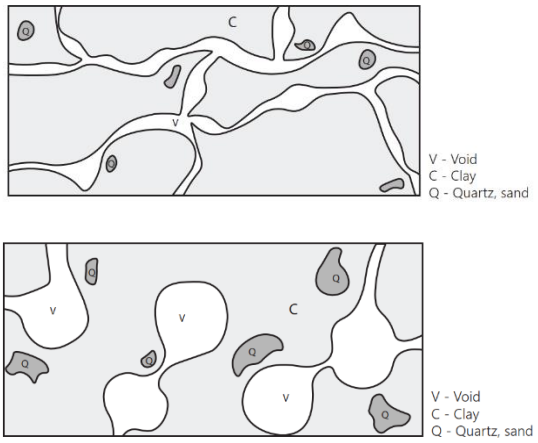


Figure 4: Different pore structures in brick (Groot & Gunneweg, 2007)

All these characteristics are important for the hygric performance of these masonry structures and caused primarily by the firing process that is used in the production of the brick. Examples for these are: total porosity, pore size diameter distribution, open and closed porosity, capillary contacts between the pores and tortuosity (complexity) of the pores. With the relation between the porosity / hardness as a result of the baking temperature. At the start of the baking process the total porosity of the clay increases (size of pores, not amount). The end of the process, that is different due to different final temperatures, the small pores merge into bigger pores. The level of this merging of pores is therefore dependent on the final temperature and effects the porosity. With a low temperature, there are more smaller pores, as a result the brick reduces in hardness. A high temperature

increases the merging of pores and creates a denser type of brick (Van Hunen, 2012)

The pore structure illustrated in Figure 4 also influences moisture storage and capillary activity of the brick. Where the top image of the brick has smaller pores and a more connected structure. Moisture infiltrates the brick further and more pore space is filled due to the connection between the pores. Additionally the small size of the pores increase capillary pressure, making moisture absorption faster. A disadvantage of this structure would be the drying of the brick. It takes longer and moisture induced damages, like frost damage (the expansion of water when freezing damages the brick), are more likely to occur. (Groot & Gunneweg, 2007)

2.2.1.2 Mortar

To connect al the bricks together, forming a masonry wall, mortar is used. It consists of an inorganic binder, aggregates, water and optionally additives.

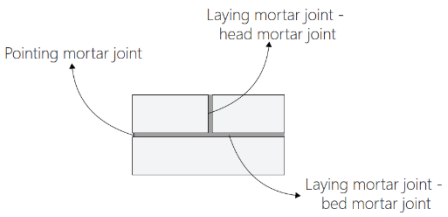


Figure 5: Brick mortar joints (Hubregtse, 2023)

Multiple binders have been used when creating mortar: early mortars used lime, from the late nineteenth cement was used as binder and current construction uses Portland cement. To create more volume, prevent cracks and make the mortar simply easier to process an aggregate is added. Sand is the standard and most common aggregate, with the corresponding material properties additionally strengthen and harden the mortar. (Van Hunen, 2012)

Similar to the bricks, mortar also has a certain porosity. This can also influence hygrothermal properties, but due to the relative small amount of material and thus the effect on the hygrothermal performance of the structure the bricks and mortar lines are not added to geometries that are simulated.

2.2.1.3 Pattern bonds

In order to create strength in the masonry walls, the bricks are ordered in a certain layout, called the pattern bonds. Again, an huge variety of pattern bonds are used but a few are very common. For examples: the Stretcher bond and Cross bond. These bonds influence the front view of the wall. Examples of these bonds can be found in Figure 6.

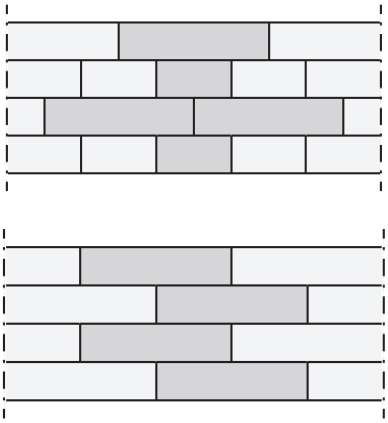


Figure 6: Front view layout of the Cross (top) and Stretcher (bottom) bond (Stenvert, 2012)

Besides the front view, that creates strength in the plane that goes through the centre line of the brick wall, the other direction can also be strengthened. This can be done by creating different layouts in the top view.

If for example the Cross bond (that has alternating layers) of one brick thick is used, the bricks can be rotated 90 degrees to create this strength in the other plane: perpendicular on the wall. In addition, if walls what are more than one brick thick are constructed, there are several options for layouts. An example of both one brick and one-and-a-half brick thick walls is visualised in Figure 7.

Besides strength, pattern bonds are due to the relative small size of the bricks and endless possibilities in orientation also used as decoration for building facades.

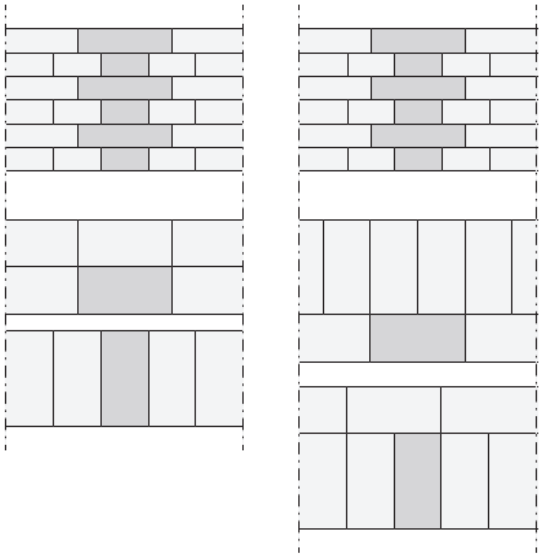


Figure 7: Top view layout of one (left) and one-and-a-half stone (right) (Stenvert, 2012))

2.2.2 Embedded wooden beam ends

2.2.2.1 Standard geometries

Wooden beams have commonly been used to span rooms and support roof an floor structures. These beams are used to hold these floor and roofs up and carry the additional loads. This can be rain and snow loads on roofs but also people and furniture used on the buildings different floors. The use of beam constructions dates back to around 1300 where wood was introduced as structural element (Van Hemert, 2013).

Starting around the nineteenth century wooden structural elements are used as single elements. Instead of complex roof structures, single elements (such as a floor beams) are implemented in building construction. Van Hemert (2013) made a visualisation of these single element beams, as can be found in Figure 8.

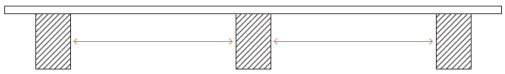


Figure 8: Single element wood constructions (Van Hemert (2013))

According to Zwiers (1920), the floor beams are commonly placed in parallel configurations, with consistent spacing between each beam. The beam ends are embedded in the masonry wall where the loads, both permanent and temporary, are diverted to walls and subsequent foundations of buildings.

Due to high loads singular beams could be additionally supported by using a triangular construction technique (Figure 10). These beams are placed in an inclination from the beam to a lower part of the wall. This extra beam reduces the loads on single points of the wall by adding an extra support point. An added benefit is the smaller beam span, minimizing bending of the beam. Large stones are placed in the wall to be used as a support point for the additional wooden beams, these stones could also be added on singular beam structures to minimize the risk of damage to the masonry wall due to the high loads. (Van Hemert, 2013).

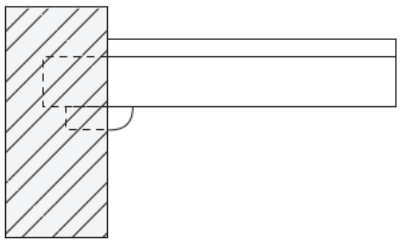


Figure 9: Corbelling stone (Van Hemert, 2013)

These corbels, spread the weight over a bigger part of the masonry. An example of corbels used in singular timber beam structures can be found in Figure 9.

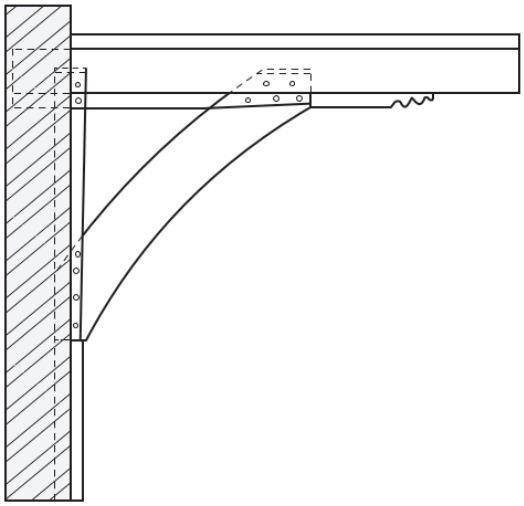


Figure 10: Example of triangular wood construction technique (Van Hemert, 2013)

The placement of the wooden beam within the wall should be sufficient enough to transfer the load correctly, therefor at least half a brick thickness should be used as guidelines. A decrease in placement depth could cause the bricks to crack. An increase in placement depth could also lead to damage, the beam can push against the masonry due to deflection caused by the load. (Wattjes, 1934)

In some case, when the solid masonry has a thickness of one brick, the floor beam could have been cut in an angle. When wider masonry wall are used, this is unnecessary due to the fact that the pattern in the bricks could be used to fully enclose the wooden beam.

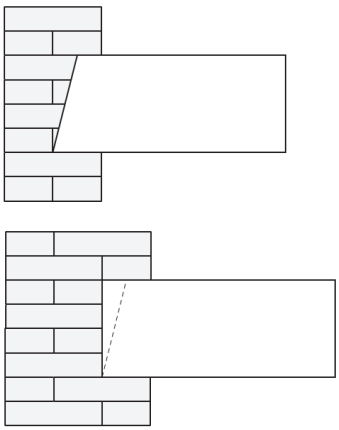


Figure 11: Wall depth in one and one and a half stone mansonry walls (Van Hemert, 2013)

2.2.2.2 Hygrothermal safety and risks

These historic construction techniques had already taken into account the possible weaknesses of the construction. Even before the application of interior insulation the beam ends were already the weakest link due to moisture induced damages. Special tiles and mortars where used to lift the beams of the bricks in order to minimize moisture contact. Lead strips in the masonry wall were used to minimize the amount of moisture that could reach the beams and stain was used to protect the wood if moisture would reach the beam.

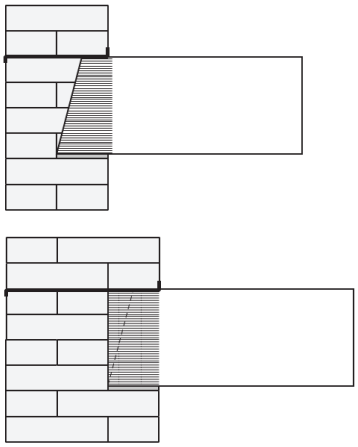


Figure 12: Protective measures against moisture induced damages (Van Hemert, 2013)

According to Van Hemert (2013), starting from the thirteenth century. The placement of wall anchors became mandatory with the use of wooden beams in residential buildings. These anchors were made out of forged steel and used to attach the wooden beams to the wall. An example can be found in Figure 13.

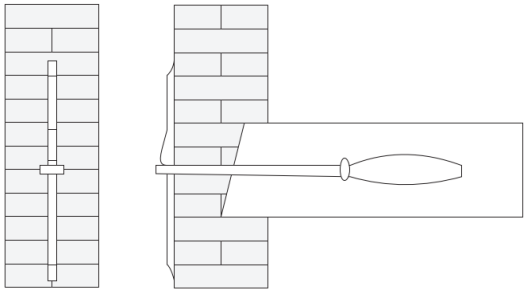


Figure 13: Forged steel beam-wall anchors (Van Hemert, 2013)

Unfortunately these anchors, despite their positive contribution to the structural characteristics, have a negative contribution on the hygrothermal performance of the wall. Firstly, the steel has a high thermal conductivity, outdoor environmental temperature conditions are therefore easily transferred to the indoor environment, creating thermal bridges. With the addition of interior insulation these thermal bridges are still present due to the fact they run through the insulated line. Additionally, the steel has different extraction and contraction rates than the brick, possibly causing the brick to crack due to dynamic weather conditions. Besides the anchors, the interior insulation increases the amount of moisture in the wall. With wood decay, mould growth and frost damages to the ends of the wooden beams as possible consequences (Figure 14).

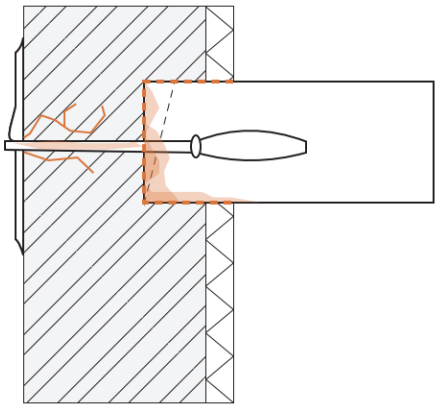


Figure 14: Possible damages to wooden beam ends and wall anchors (Van Hemert, 2013)

Capillary active insulation materials that are internally used are known for their possible, positive contribution to the amount of moisture in the wall. The high moisture transport and storage capabilities can be used to transfer more moisture that potentially condensates between the layers (interstitial condensation) in comparison with standard insulation materials with vapour barriers. With the added fact that these vapour barriers are often hard to install due to all the complex geometrical shapes that need to be sealed in old buildings and mechanical damages caused in construction and occupation stages. This is because no air gaps can be present in the total wall that can facilitate air flow.

The application of capillary active materials, that are potentially also biobased, is very new. Additional research is needed to determine if the application of capillary active insulation materials can reduce this moisture load in the structure, consequently increasing the lifespan of the beams and minimizing the risks to structural failure. Eventually being able to restore and transform more old, energy inefficient buildings. Contributing to environmental goals to minimize energy use, CO2 emissions and the use of finite materials.

2.3 Building insulation materials

Insulating a building is a common practice to reduce the loss of heat in cold periods and reduce warming of the indoor environment in warm periods. In recent years, the insulation of buildings became increasingly important due to high energy costs and the subsequent need to reduce energy losses.

Besides the standard insulation materials, new developments in insulation materials are made due to the increased interest in the building sector. Both standard and new materials are further elaborated in this paragraph. All to create a basic understanding of the current insulation possibilities with corresponding (dis)advantages.

A significant tool for the improvement of a building's energy behaviour is the addition of insulation to the skin of the building. Reducing energy consumption for heating and cooling the building results directly in a lower electricity or gas use and positively influences the building's impact on the environment. Adding interior insulation changes the hygrothermal behaviour of a façade, and thus influences the properties of the total geometry. Especially in the heating season, when the temperature of the masonry wall is lower due to the decrease in solar radiation, there is a high chance of interstitial condensation.

According to Papdopoulos (2015), insulation materials in Europe are mainly characterised by the two different groups. The insulation made out of inorganic materials and organic materials. Organic materials have carbon-hydrogen bonds, are commonly found in living matter or derived from living matter and have biological nature, in contrary to inorganic materials. With similar performance in thermal properties but significantly different other properties. Additionally there are hybrid forms and other new material technologies used in building insulation as can be found in Figure 15.

Within organic materials there are biobased materials, with the prefix "bio" meaning "life" or "living organisms". It refers to the raw product, for example plants, made out of biological or natural materials and thus not relate to the produced product.

Organic, but not biobased products can be made with synthetic raw materials such as natural gas or crude oil (Cordis, 2019). This paper researches insulation in three other forms: conventional, biobased and capillary active materials. With conventional being organic and inorganic forms and considered to have a negative impact on the environment, biobased being plant based and environmentally friendly and capillary active as a specific sort of insulation. This capillary activity can be found in both the conventional and biobased materials but has the ability to capture, store and redistribute the moisture through the material.

2.3.1 Conventional insulation

Rock and other mineral products, and therefor inorganic materials, are used as raw material to produce mineral wool. Commonly used in buildings and industrial insulation and made by spinning or drawing molten minerals at high temperatures. This thermal insulation is produced from volcanic rock, typically basalt or dolomite. (Kosny & Yarbrough, 2022).

Research from Kumar et al. (2020) mentioned that besides volcanic rock, iron slag can be used to create mineral wool. Another insulation made by melting and spinning raw materials is fiberglass. With blowing high pressure air streams over molten glass, fibres are created that can be used for thermal insulation.
Besides fibrous materials, there are inorganic cellular insulation materials. For example, calcium silicate, foam glass, perlite and vermiculite. With raw materials such as sand, chalk and magnesium-aluminium silicate.

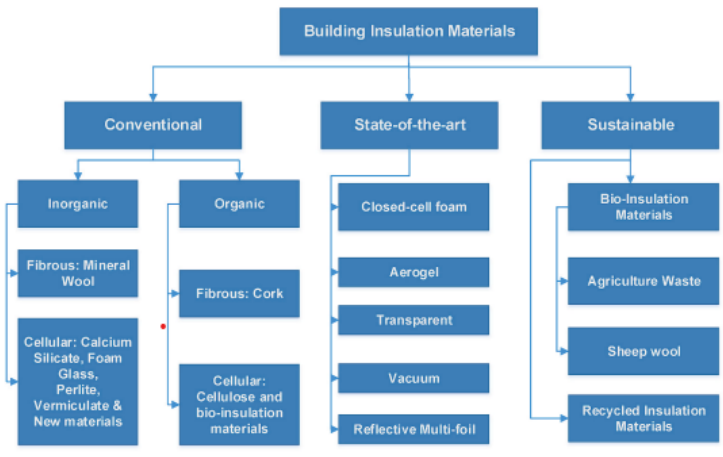


Fig. 1. Classification of building insulation materials.

Figure 15: Classification of building insulation materials (Kumar et al, 2020)

Other widely used insulation materials are organic based polystyrene, available in expanded and extruded polystyrene. All available in panels that are formed by using oil based products and additives that expand the material creating small cavities that result in low thermal conductivity properties (Kumar et al., 2020).

2.3.2 Biobased insulation

Recent developments in building insulation are mainly based on the need to reduce the impact on the environment. Not only to improve the building's energy efficiency by creating higher heat resistance of building facades, but also the type of materials used. Biobased insulation materials are emerging as a promising alternative. Compared to the conventional materials, biobased materials are more renewable, have a low embodied energy and are CO_2 neutral or even negative. (Palumbo, 2017)

Additionally, the research of Palumbo (2017) mentions that biobased materials have a completely different hygrothermal performance. The natural fibre structure of these materials cause the sorption and desorption of water vapour from the environment in their porous structure. Constantly trying to reach an equilibrium in dynamic environmental conditions. Especially high extremes in indoor humidity can be reduced, causing the use of biobased materials often having a positive effect on the users of the building, creating more comfortable indoor conditions in comparison with non-biobased materials.

Most of the biobased materials have the benefit that they are more locally available. Creating shorter loops that benefit a circular economy. Also the fact that these materials can be recycled and reused more easily contributes to a positive environmental impact. For insulation, natural fibres such as wood, hemp, kenaf, cotton, flax or sheep's wool are found as replacements for conventional alternatives.

2.3.3 Vapour closed, vapour open and capillary active

The transport of water(vapour) is an important part in building insulation, especially with new materials such as biobased and capillary active materials being used more often. Vapour closed, vapour open and capillary active are terms often used, but further elaboration is important to fully understand the effect on hygrothermal properties of a specific material.

Vapour closed is often used when building facades are built in a more traditional method. For the inner most layers of the wall the vapour resistance is highest, reducing in resistance towards the outside part of the wall. This ensures that vapour from the indoor environment can't flow through the wall, and if moisture gets in the wall, it is transported to the outside. It is therefore easier for moisture to get out, than enter the wall, resulting in less internal moisture. The aim for these system is to prevent interstitial condensation. Commonly foils are used as the vapour break. These foils have an extremely high water vapour diffusion resistance number and are therefore effective barriers against moisture diffusion. This is immediately also the disadvantage of these system, when the foil has a small gap, air with water vapour will diffuse through. Multiple type of insulation, such as PU, XPS or mineral wool are often seen in vapour-closed structures.

According to Stappers (2021), vapour open materials are materials where there is a small resistance to water vapour diffusion. Examples for these materials are wood fibre boards, mineral wool and cellulose. These materials are especially suitable for application where moisture balance is critical, such as in breathable layers for retrofitted buildings. Vapour open systems support the drying potential of assemblies and reduce the moisture content in the wall, but insulation performance can be compromised when there is a significant exposure to moisture.

Finally, there are capillary active materials. By definition capillary active insulation systems are vapour open. But additionally they can store and redistribute the water, due to a high water absorption coefficient. It is important to note that not all vapour open materials are capillary active. Examples for these insulation materials are calcium silicate or aerated autoclaved concrete. Capillary active materials can be both biobased and non-biobased. In capillary active systems interstitial condensation due to the high moisture vapour permeability is unavoidable, but not necessary problematic. The inner pores of the material can absorb this moisture and transport it back to the warm interior side of the wall where it can evaporate again and consequently reduce moisture levels in the wall.

2.4 Hygrothermal properties of building materials

The properties of materials, especially heat and moisture, have a tremendous effect on the amount of moisture in the wall. Therefore, these two properties are studied in depth in this paragraph.

First, the basics of heat and moisture are discussed, including a section on the anisotropic behaviour of materials. This is known as the material having different properties in different directions of the material. This basic explanation is followed by a more in depth elaboration on moisture storage and transport in materials.

2.4.1 Basics of hygrothermal properties

Heat transfer in materials occurs in three mechanisms: conduction, convection and radiation. With conduction transferring heat through molecule contact, convection refers to the transfer of heat through the movement of liquids and gasses and radiation using electromagnetic waves for heat transfer.

Simultaneously, moisture transport in materials is driven by flow due to pressure differences, advection and diffusion. Where pressure differences can be caused by wind pressure, gravity and capillary suction, advection is caused by movement of water molecules by another flowing medium and diffusion uses equilibrium principles where molecules try to level out concentration differences.

In building components, the transfer of heat, moisture and air occurs simultaneously and are therefore influenced by each other, e.g. air flows cause heat and moisture transfer, temperature differences cause differences in relative humidity that in turn cause moisture transfer and evaporation and condensation depend on temperature and influence the heat balance. (De Wit, 2009).

Both Stappers (2021) and Hall & Casey (2012) describe basic heat and moisture properties in detail within their research. The following is based on this reference.

2.4.1.1 Heat

To create energy efficient buildings we need to reduce heat transfer through the building envelope, among others. The transfer of heat (and thus energy) through a material can be described by the thermal conductivity. Thermal conductivity defines the ability of a material to conduct heat through its mass due to a temperature gradient between the sides of the wall. Materials with low thermal conductivity values are consequently used for insulation purposes. Importantly, the thermal conductivity of a porous material increases significantly with the degree of saturation, as the presence of moisture enhances heat transfer due to water's high thermal conductivity relative to air.

A certain amount of energy is needed to raise the temperature of a specific amount of mass of a material. The raise in temperature by one Kelvin of one unit of mass at constant pressure is called the specific heat capacity (c_p) When this is multiplied with the materials density, this value can be converted into volumetric heat capacity, commonly used for construction purposes. Similarly to thermal conductivity, water in porous materials substantially increases the heat capacity. Making the capacity for thermal storage higher.

The relation between material thickness and thermal conductivity gives an indication of the thermal resistance (R). Thermal resistance quantifies a material's ability to resist heat flow, expresses as:

$$R = d/\lambda$$

where,

R = heat resistance of a layer, [m^{2}K/W];
d = thickness of a layer, [m];
λ = thermal conductivity coefficient, [W/mK].

The thermal resistance of a construction is the sum of the thermal resistances of all individual construction layers. Resulting:

$$R_c = R_1 + \ldots + R_n$$

where,

R_c = heat resistance of total, [m^{2}K/W];
R_1 = heat resistance of first layer, [m^{2}K/W];
R_n = heat resistance of last layer, [m^{2}K/W].

Another property used to quantify thermal performance is the heat transfer coefficient (U), often used in glass construction. This gives the amount of energy transferred per square meter of surface per degree of temperature difference. The relation between U and R_c in standard energy calculations for walls is given by:

$$U = \frac{1}{0{,}04 + R_c + 0{,}13}$$

where,

U = heat transfer coefficient, [m^{2}K/W];
R_c = heat resistance of total, [m^{2}K/W].

2.4.1.2 Water(vapour)

Water vapour is the gaseous form of water. Vapour pressure represents the tension of water vapour in the air, it can be calculated using the equation:

$$p = c * R_w * T$$

where,

p = vapour pressure, [Pa];
c = water vapour concentration, [kg/m^3];
R_w = gas constant of water, [J/kgK];
T = temperature, [K].

The amount (in mass) of water vapour present in a cubic meter of air is referred to as absolute humidity. When the ratio between the actual moisture content in the air to the maximum amount it can hold is made, it is called relative humidity. Factors like temperature and air pressure can influence this humidity. Values between 40 and 70% are seen as comfortable relative humidity levels. Calculated with:

$$\varphi = \frac{p_v}{p_{sat}}$$

where,

φ = relative humidity, [%];
p_v = vapor pressure, [Pa];
p_{sat} = saturation vapor pressure, [Pa].

Moisture transport occurs via vapour diffusion. The movement of water vapour to reach equilibrium, occurs when differences in vapour pressure exist across a material. Vapour flows through the air-filled pores within the material, driven by the pressure gradient. The rate of vapour flow (g) can be expressed as:

$$g = \frac{\Delta p}{R}$$

where,

g = vapour flow rate, [kg/m^{2}s];
Δp = vapour pressure difference, [Pa];
R = vapour diffusion resistance, [m/s].

Vapour diffusion can be represented by two properties. The vapour diffusion coefficient and the water vapour diffusion resistance factor. The vapour diffusion coefficient quantifies a material's capacity to permit water vapour transport. Higher coefficients indicate materials that allow greater vapour flow, whereas lower coefficients denote materials with higher resistance to vapour diffusion.

The water vapour diffusion resistance factor (μ) represents the ratio of the vapour diffusion resistance of air to that of a material. Materials with high μ values are less permeable to vapour diffusion, effectively acting as barriers to moisture transport. In the contrary, materials with lower μ values allow vapour to pass through more easily.

Besides vapour transport there is also storage. Pores in the material can store the moisture that flows through the material. This is quantified for example with the water absorption coefficient (A_w). It quantifies the rate at which a material absorbs water via capillary action over time. It is expressed as the mass of water absorbed per unit area of the material.

The storage of moisture in a material can be expressed in a function. The moisture storage function describes the relationship between the moisture content of a porous material and the relative humidity of the surrounding air. This function, commonly referred to as the sorption isotherm, provides insight into a material's ability to adsorb and retain moisture under varying humidity conditions. The sorption isotherm typically displays a nonlinear relationship. In some cases, the desorption (release of moisture) isotherms is also visualised in the same graph.

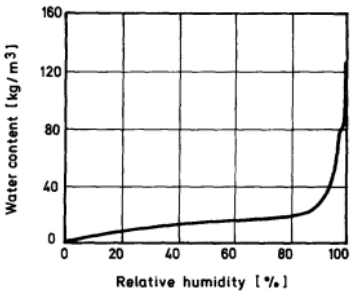


Figure 16: Sorption isotherm (sandstone) (Kunzel, 1996)

Condensation occurs when water vapour transitions to liquid form due to changes in temperature and relative humidity. As air cools, its capacity to hold moisture decreases, causing excess vapour to condense into liquid water. This phenomenon can lead to increased moisture levels within materials, potentially causing structural damage or mould growth. Proper management of vapour diffusion and material selection is therefore essential to mitigate the risks associated with condensation.

2.4.1.3 *(An)isotropic material behaviour*

Properties of materials, for example heat and moisture, are not always similar in all directions of the materials. Isotropic and anisotropic are terms used to describe these differences. Where isotropic materials have properties that are similar in all directions. Anisotropic materials properties vary in different material directions. These terms originate form the Greek *an* (no), *isos* (equal) and *tropos* (way)

These differences are important since these properties are often used in all kinds of material calculations, for example physical, mechanical or in this case hygrothermal calculations. Important to note is the fact that a material can be isotropic for one property, yet anisotropic for another. An example is glass: mechanical properties of glass are isotropic but anisotropic for light infiltration properties.

Examples of materials that are commonly isotropic are: polymers (plastic), metals (aluminium) or cubic crystals (diamonds). And anisotropic materials: wood, ice, crystals or reinforce concrete. (Helmenstine, 2022)

This research focusses on wood anisotropic behaviour in particular. The wooden embedded beams are known for their directional difference in moisture transport. Both for transport and the resistance of moisture flow through wood. The three directions are longitudinal, radial and tangential. Longitudinal is the direction of height of the tree, in the direction of the grain. Where moisture transport is highest, and consequently resistance lowest. In the 2D geometry simulated the other direction is radial, perpendicular to the growth rings of the tree (from centre point to the outside of the tree). The other direction, with lowest moisture transport, is tangential (perpendicular to the radial plane). These directions are visualised in Figure 17.

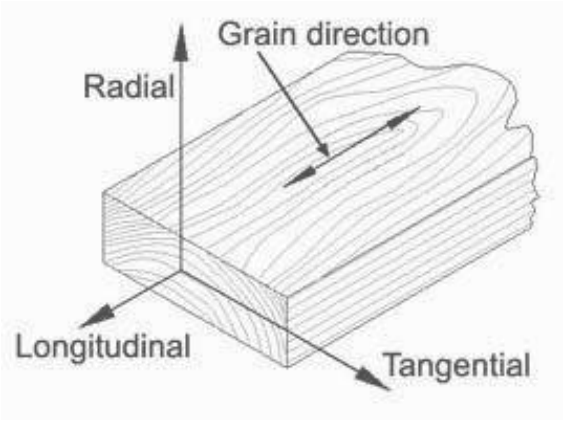


Figure 17: Wood anisotropic directions (Ravenshorst, 2015)

2.4.2 Moisture in materials

The following paragraphs focus on the complexity of moisture in materials. Divided into storage and transport of moisture in building materials.

The amount of moisture in materials is dependent on environmental conditions, can occur simultaneously in different phases and change constantly. The total amount of moisture in the materials can be found with the **mass balance**, as storage and transport influence the amount of moisture in the materials.

$$-\nabla * g = \frac{\partial w}{\partial t}$$

where,

∇ = amount of flow change in a point
g = density of moisture flow rate, [kg/m^{2}s'];
w = moisture content, [kg/m^3];
t = time in hours, [h].

(De Wit, 2009)

Where the left side of the mass balance is the amount of moisture transfer and the right side the amount of moisture storage.

In building construction, the objective is to prevent water from infiltrating building components or, at the very least, to reduce their moisture content to a level where its harmful effects are minimized. There are four primary mechanisms through which moisture enters building components. These are: wind driven rain, interstitial condensation, rising water and initial moisture in building materials (Künzel, 1995). Illustrated in Figure 18.

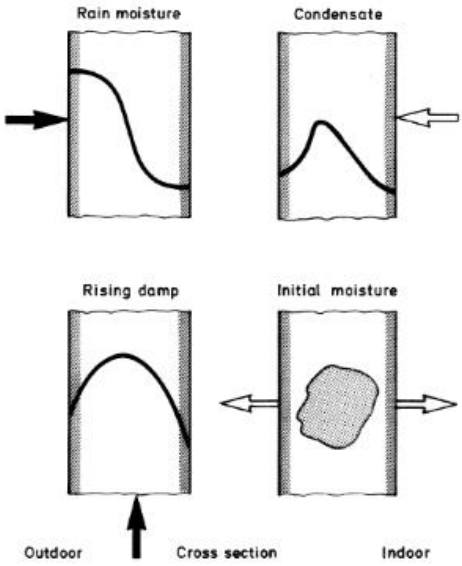


Figure 18: Primary moisture sources building components (Künzel, 1995)

Two types of building materials can be distinguished when exposed to moist air. Hygroscopic materials, that absorb water molecules at the inner surface of their pore system until they reach a moisture content in equilibrium with the ambient humidity. And non-hygroscopic, materials that remain dry when exposed to moist air. A building materials is considered capillary active if the materials absorbs moisture through capillary suction when in contact with water. When it does not, it is termed hydrophobic.

Capillary active material absorbs liquid water until it reaches a state of saturation known as free water saturation or capillary saturation. In normal situations, this is the maximum absorption level of the material. Higher moisture levels, up to pore saturation or maximum water saturation, can only be achieved when pressure is applied or in the form of water vapour diffusion caused by a gradient in temperature (Künzel, 1995).

2.4.3 Moisture storage in materials

Moisture in materials can exist in different phases, these are solid, liquid and vapour phases. These physical states are not always clearly identified, for example in the micropores of the materials. Within the building materials, these phases continuously change, therefor it is practically easier to calculate their total sum (Künzel, 1995). As described previously, a moisture balance of porous building materials, is determined by the moisture storage characteristics and the different ways of moisture transport. Both in liquid and vapour phases (Krus, 1996).

In a theoretical way, building materials can absorb moisture until all the pores of the material are filled with water. But, practically, ambient conditions affect the absorption of moisture and it is therefore important to find the effect of these ambient conditions on the moisture storage capacity. (Künzel, 1995).

The amount of moisture in materials is distinguished by three different regions, the moisture regions. Where the total amount of moisture in a material increases for every region. These regions can be found in Figure 19

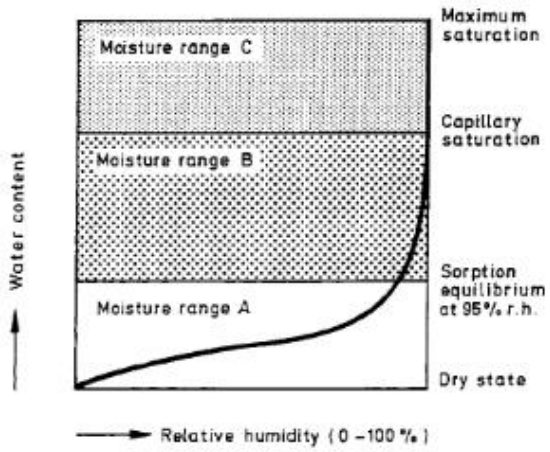


Figure 19: Schematic diagram of the moisture storage function of a hygroscopic capillary-active building material (Künzel, 1995)

2.4.3.1 Region A – Sorption moisture region

The first region of the hygroscopic building materials is the sorption moisture region, up to the sorption equilibrium at 95% relative humidity. When in contact with moist air, hygroscopic building materials become subject to equilibrium moisture which is determined by the ambient relative humidity, temperature can be disregarded in these cases.

The hygroscopic equilibrium moisture values, between water content in the material and relative humidity, of materials can be visualised in sorption isotherms. A difference can be made between absorption and desorption of moisture. But, in most cases, the absorption isotherm is enough to characterise the possible moisture storage of a material, because of the hysteresis (influence on each other) is not significant. Absorption isotherms are therefore sufficient to use in moisture transport calculations.

Water content in a material can be clearly calculated with sorption measurements that are based on the relation between relative humidity and moisture content. In Figure 20 some absorption and desorption isotherms of four building stones: Lime silica brick, cellular concrete, purnice concrete and expanded clay concrete.

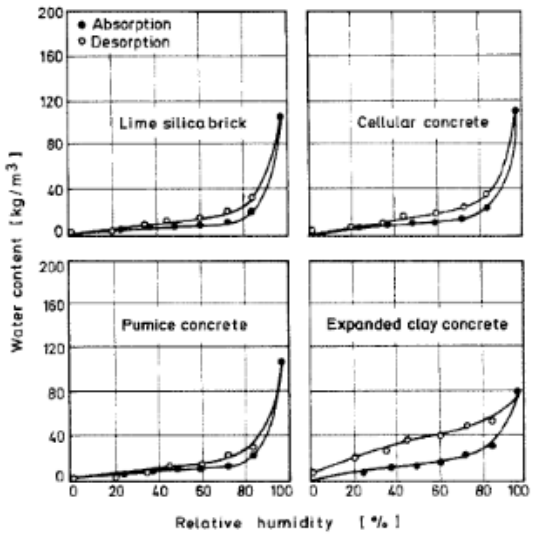


Figure 20: Sorption Isotherms of four building materials (Künzel, 1995)

2.4.3.2 Region B – Capillary water region

The second moisture region is the capillary water region. This region is bounded by the sorption equilibrium and the maximum capillary saturation, also known as the free water saturation.

When two capillary active materials are in contact with liquid water, they absorb water until there similar sized pores are equally filled with water, they are in equilibrium. This is caused by the bigger suction force of smaller pores in comparison with smaller suction in bigger pores. When in equilibrium, the building materials can have a different total water content due to material pore properties.

The amount of capillary suction forces in materials is mostly calculated with the cylinder capillary model. This **capillary pressure**, that is also called suction stress can be calculated with:

$$p_c = -\frac{2\sigma \cos\theta}{r}$$

where,

p_c = capillary pressure, [Pa];
σ = surface tension of water, [N/m];
r = capillary radius, [m];
θ = contact angle, [°].

(Künzel, 1995)

In Figure 21 capillary effect can be found. With lower pressure occurring higher up in the cylindrical tube due to the higher adhesion forces of liquid molecules to the molecules of the surface of the tube instead of the liquid molecules itself, creating capillary suction.

A example of such a liquid is water. Where the cohesion (the sticking of the water molecules to each other) is smaller than adhesions (the sticking of water molecules to another surface). Creating a concave meniscus in the cylinder. Where, for example, mercury has a convex meniscus. The mercury molecules have higher cohesion forces than adhesion forces (Mitropoulos, 2009).

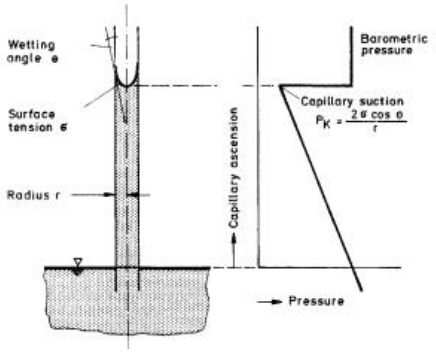


Figure 21: Schematic illustration of cylindrical capillary and pressure condition (Künzel, 1995)

With smaller radiuses, or pore sizes, a relatively bigger part of the cross section surface is affected by the pore walls in relation with the non-affected surface. Therefor creating more suction pressure in smaller pores in comparison with bigger pores.

Kelvin's formula, based on thermodynamic equilibrium (thermal, dynamic and chemical equilibrium), can be used to give the **relation between capillary suction and vapour pressure**:

$$\frac{p_v}{p_{sat}} = \varphi = \exp\left(\frac{p_c}{\rho_w R_v T}\right)$$

where,

p_v = partial vapour pressure, [Pa];
p_{sat} = partial saturation vapor pressure, [Pa];
φ = relative humidity, [-].
p_c = capillary pressure, [Pa];
ρ_w = density of water, [kg/m^3];
R_v = gas constant for water vapour, [J/kgK];
T = absolute temperature, [K].

(De Wit, 2009)

The relation of relative humidity, capillary suction and the radius of filled pores of porous building materials can therefore be calculated and is illustrated in Figure 22.

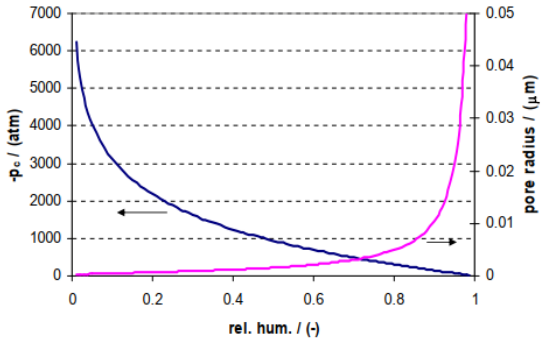


Figure 22: Relation between relative humidity, capillary pressure and radius of filled pores

2.4.3.3 Region C – Supersaturated region

The last region is the supersaturated region. In this region the relative humidity is always 100% and can never be reached within normal suction conditions. Only external pressure can cause capillary saturation to exceed its limit. Examples for possible external pressures are: forced condensation by pushing below the condensation point or the application of a vacuum to remove trapped air in the pores (Krus, 1996).

Mathematically, the relation between the potential amount of water in the material and the actual amount of water in the material can't be calculated. The **moisture storage capacity** is calculated with:

$$\Delta w_{ü} = w_{max} - w_f$$

where,

$\Delta w_{ü}$ = moisture storage capacity, [kg/m^3];
w_{max} = maximum water saturation, [kg/m^3];
w_f = free water saturation, [kg/m^3].

(Künzel, 1995)

2.4.4 Moisture storage functions

For building components with direct layer to layer contact, it is important to find the most accurate moisture storage function.

The capillary connection between the layers plays an important role, e.g. plaster or stucco over masonry walls (Künzel, 1995). The sorption isotherm is determined by the weight of materials at different relative humidity levels. The measurements are time-consuming, due to the fact that equilibrium has to be reached, without the use of external pressures that could affect the results. Especially above the 95% level of the hygroscopic region the curve increases extremely, also increasing the need for precise relative humidity measurement tools that are currently not available. Instead moisture content can be measured for different capillary pressures with pressure plates, resulting in more precise results. In Figure 23 the moisture storage in pores of building materials can be found, with the white dots illustrating humidity level, and black dots the more precise pressure plate measurements.

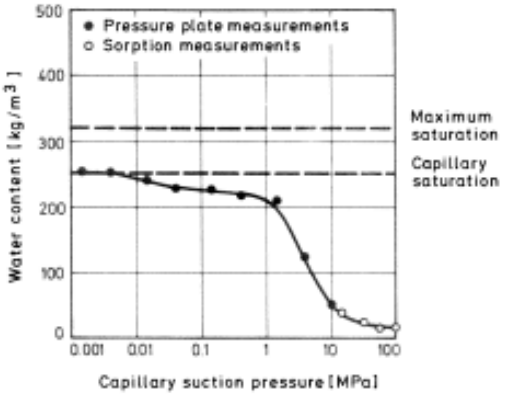


Figure 23: Moisture storage of a building brick as a function of the capillary suction in pore water (Künzel, 1995)

2.4.4.1 Theoretical Van Genuchten function

Instead of determining the moisture storage with measurements, the function can also be approximated theoretically. An approximation of the moisture storage function that is often used, is the Van Genuchten moisture storage function. The function:

$$\theta(\psi) = \theta_r \frac{\theta_s - \theta_r}{[1 + (a|\psi|)^n]^{1-1/n}}$$

The inverted Van Genuchten function is used to calculate the suction pressure at a given moisture content. The inverted function:

$$\psi = \frac{1}{\alpha * n}\left(\left(\frac{\theta_s - \theta_r}{\theta - \theta_r}\right)^{\frac{1}{1-\frac{1}{n}}} - 1\right)$$

where:

ψ = suction pressure, [Pa];
θ = volumetric water content, [m^3/m^3];
θ_s = saturated water content, [m^3/m^3];
θ_r = residual water content, [m^3/m^3];
a = factor for mean pore size, [-];
n = factor for pore size distribution, [-].

(Van Genuchten, 1980)

2.4.5 Moisture transport in materials

Not all moisture transport mechanisms are relevant for building physics. In Figure 24 an overview can be found of all moisture transport mechanisms.

In both liquid and gaseous phases, the transport of moisture due to hydraulic and gas flow is negligible. This is due to the fact these don't influence moisture transport under the normal conditions of pressure differences, convective hydraulic and gas flows are influenced by high temperature or density levels causing significantly high pressure differences and are therefore not taken into account in building physics calculations. Additionally, insignificant for capillary moisture transport calculations are influence of electrokinesis and thermodiffusion. (Krus, 1996)

The moisture transport mechanisms relevant to calculations in building physics are vapour diffusion and effusion for gaseous moisture and capillary conductivity and surface diffusion for liquid forms of moisture (Künzel, 1995). Highlighted grey in Figure 24.

2.4.5.1 Vapour transport

Within vapour transport, there are two types: vapour diffusion and effusion. Both transport mechanisms are dependent on the pore size of the material and are driven by the partial pressure. This partial pressure is influenced by the pressure on both sides of the building component. In winter conditions, temperature and vapour pressure are higher on the inside of the building. Resulting in a vapour flux from the inside to the outside of the building. (Künzel, 1996)

Both gaseous moisture transport driven by the difference in pressure trying to reach equilibrium, but differ when looking at pore size of the materials. With the **Knudsen factor** the relation between pore size and mean free path for vapour transport can be calculated:

$$K_n = \frac{L}{2r}$$

where,

K_n = Knudsen factor, [-];
r = pore radius, [m];
L = mean free path, [m].

If the mean free path length is greater than the pore diameter, the transport mechanism is called effusion. In this case the molecules of the water collide with the molecules of the pore wall. If the molecules of the water collide with each other, vapour diffusion occurs.

Both effusion and vapour diffusion can be separately calculated but are practically almost impossible to separate. Therefor one formula, also known as Fick's law for vapour diffusion, to calculate **vapour diffusion** is used:

$$g_v = -\frac{D_D}{\mu R_D T}\nabla p_v$$

where,

g_v = vapour diffusion flux density, [kg/m^{2}s];
D_D = vapour diffusion coefficient in air, [m^2/s]
μ = water vapour diffusion res. Factor, [-]
R_D = gas constant for water vapour, [J/kgK];
T = absolute temperature, [K];
p_v = partial pressure of water vapour, [Pa]

(Krus, 1996)

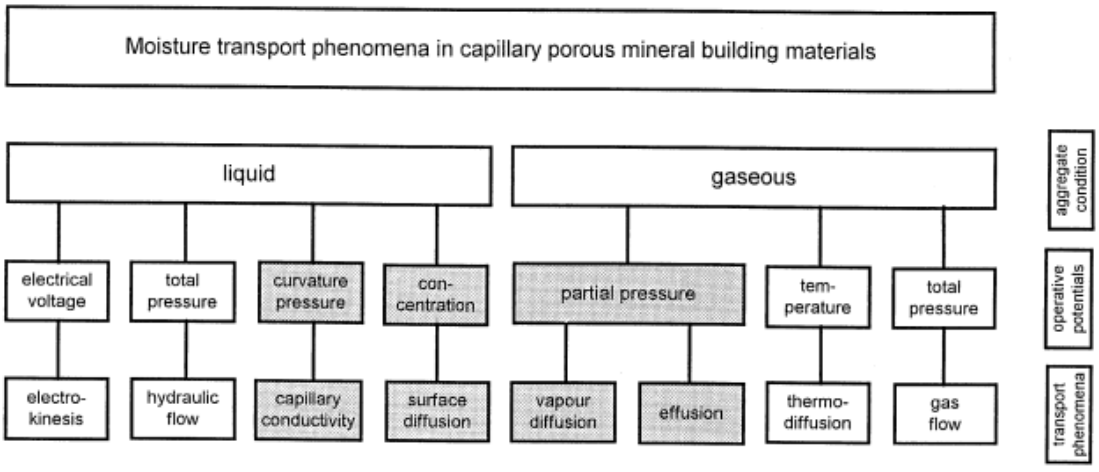


Figure 24: Moisture transport mechanismns in building materials (Krus, 1996)

When vapour flows through pores of building materials, resistance is created. This is caused by the pore structure, both increasing and decreasing in amount of area in comparison with the total cross section area and by pore ducts varying in cross section sizes. These phenomena were experimentally tested which resulted in a formula for the water vapour diffusion resistance factor. The resistance factor expresses the difference in resistance of the material with a layer of air of equal thickness. With the included factor for resistance the total **vapour diffusion** in porous materials can also be calculated.

Vapour diffusion resistance factor:

$$g_v = -\frac{\delta_a}{\mu}\nabla p$$

Without the vapour diffusion resistance factor the **total vapour diffusion**:

$$g_v = -\delta_p \nabla p$$

where,

g_v = water vapour diffusion flux density, [kg/m^2];
δ_a = vapour permeability of air, [kg/msPa];
δ_p = vapour permeability of a porous material [kg/msPa];
p_v = partial vapour pressure, [Pa];
μ = vapour diffusion resistance factor, [-].

(Krus, 1996)

2.4.5.2 *Liquid transport*

Where pore properties of materials influence the type of moisture vapour transport, the moisture content regions influence the type of liquid transport. Surface diffusion takes place in region A, the sorption moisture region. Capillary conductivity in region B, the capillary water region.

According to Krus (1996), surface diffusion occurs on the wall surface of the pores in the materials. With high relative humidity, the number of molecules sticking to the pore wall surface are higher in comparison with low humidity. Causing a thicker film of liquid water on the high relative humidity side of the pore wall and a thinner film on the relative humidity low side. The total water layer is called the sorptive film. With this difference in thickness, a mass transport of liquid water takes place, due to the water molecules seeking equilibrium alongside the wall. The assumption can be made that the surface diffusion is proportional to the layer thickness gradient and therefor the concentration gradient. In this case, Fick's law on vapour diffusion can be used (with slight adjustment). **Surface diffusion**:

$$g_{OD} = -D_{OD}\nabla w$$

where,

g_{OD} = surface diffusion flux density, [kg/m^{2}s];
D_{OD} = surface diffusion coefficient, [m^2/s];
w = water content, [kg/m^3].

Similarly to vapour transport, capillary conduction and surface diffusion occur simultaneously in liquid moisture transport and therefor a combined formula is used. The diffusion equations for **capillary conduction**:

$$g_w = -D_w(w)\nabla w$$

where,

g_w = liquid flux density, [kg/m^{2}s];
D_w = capillary transport coefficient, [m^2/s];
w = water content, [kg/m^3].

When practically, instead of theoretically looking at the liquid moisture transport in building materials, the liquid conduction coefficient and relative humidity highly effect the total transport. Simplified the previous equations in a **total liquid moisture transport** equation:

$$g_w = -D_\varphi \nabla \varphi$$

where,

g_w = liquid flux density, [kg/m^{2}s];
D_φ = liquid conduction coefficient, [kg/ms];
φ = relative humidity, [-].

2.4.6 Moisture balance

The mass balance, involving moisture storage, the total vapour diffusion and total liquid moisture transport equations result in the differential equation for moisture transport:

$$\frac{\partial w}{\partial t} = \nabla * (\delta_p \nabla p_v + D_\varphi \nabla \varphi)$$

where,

w = water content, [kg/m^3];
t = time in hours, [h];
δ_p = vapour permeability of a porous material [kg/msPa];
p_v = partial vapour pressure, [Pa];
D_φ = liquid conduction coefficient, [kg/ms];
φ = relative humidity, [-].

(Künzel, 1995)

2.5 Associated risks of indoor insulation

Indoor insulation causes the external wall to reduce in temperature. With this reduction the moisture loads in the wall increase and degradation risks to the building materials arise.

Examples of this degradation include mould growth or wood decay to the (bio)based insulation or wooden beams. These two material degradation types are explained in this paragraph. Wood decay is subsequently research in more detail. Focussing on the favourable growth conditions and corresponding decay ratings for different species of wood.

2.5.1 Mould

Mould growth can result in a degradation of building materials and can negatively influence the occupant's health. Numerous mould prediction models are available to assess the mould risk on building materials. Mostly originated from mould assessment models for wood but nowadays used in hygrothermal software to evaluate the total building component. Some mould prediction models that are frequently applied in the building physics are the VTT model, Sedlbauers's isopleths and the biohygrothermal model (Vereecken, Vanoirbeek, et al., 2015).

Both in WUFI and DELPHIN the VTT mould prediction model of the Technical Research Centre of Finland (VTT) in collaboration with the Tampere University of Technology (Fraunhofer IBP, z.d.) is available (DELPHIN, z.d.).

According to Vittanen (2008), the important starting point to understand mould in building components is the fact that mould spores are everywhere in our surroundings. Mould growth will always occur, unless the building exterior envelope is treated with fungicides and/or cleaned frequently. Mould growth can have a negative effect on the durability of the building component but also on the indoor environment.

Microbial growth requires a certain duration of suitable exposure conditions to initiate. The focus is on the response time or response duration under varying humidity and temperature conditions for mould growth (Viitanen, 1996; Hukka and Viitanen, 1999). The lowest relative humidity level necessary for mould growth is approximately 75-80%. Illustrated in Figure 25. The favourable temperature range for mould growth is 0-50 °C (Finnish mould growth model, z.d.)

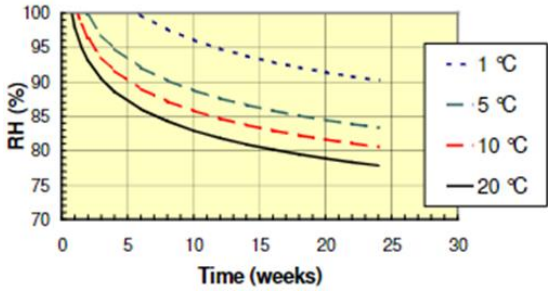


Figure 25: Critical humidity (RH %), time (weeks) and temperature to start mould growth on pine sapwood (Vittanen et al, 2000).

Mould growth is highly effected by time. The total exposure time for mould fungi to response is affected by the time period of the different conditions. Like the time period of high and low humidity conditions and same humidity and temperature levels. Research of Vittanen and Bjurman (1995) state that a short, high humidity condition will not directly cause fungal growth, especially when the low humidity time periods are long enough. When the time period of high humidity levels is longer than 24 hours, the expected mould growth will be higher. When there are long dry periods, low or even neglected fungal growth are likely.

Another big effect is seen in boundary conditions. In mould growth simulation, it is essential to identify the minimum boundary conditions where fungal growth becomes possible in various materials. Specific minimum and maximum levels for moisture content or water activity and temperature exist within which fungi can thrive in. With favourable conditions, fungal growth may start and progress at different rates and is influenced by humidity, temperature, mould organism type and properties of the building material used. Response times are short (from a few days to a few weeks) in pine sapwood under conditions favourable to micro-organism growth, but grow significantly (from a few months to a year) slower under conditions near the minimum and maximum moisture or temperature levels. An example of these growth rates is found in Figure 26, where temperature difference is illustrated between the two graphs. It is important to find these boundary conditions and the correlation to accurately predict mould growth in simulations (Viittanen, 2008).

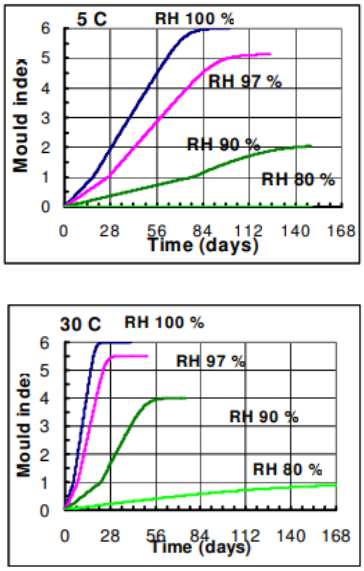


Figure 26: The critical humidity, temperature, and exposure time periods at varied constant conditions for initiation of mould growth on pine sapwood based on VTT mathematical mould growth model (Viitanen et al, 2000)

The Finnish mould growth model is currently the most advanced tool for assessing mould growth and is based on the original wood mould growth prediction model of VTT and PhD Hannu Viitanen. The Finnish model extend to also predict different materials in different temperature and humidity levels.

The mould index ranges between 0 and 6. Where 0 is no mould and 6 is the maximum amount of mould on the surface. The risk for fungal growth is calculated with hourly temperature and relative humidity levels. With the mould prediction model, the evaluations of risk of a specific mould can be executed. This is caused by the model focussing on all mould types, and not specific ones. In Figure 27 the mould index (M) of the Finnish mould growth model is shown. (Finnish Mould Growth Model, z.d.)

Mould index M	Found mould growth	Notes
0	No growth	Clean surface
1	Growth seen with a microscope	Growth is beginning in places
2	Clear growth seen with a microscope	Mould growth covers 10 % of area studied (microscope). Several mould colonies have formed
3	Growth seen with the naked eye Clear growth seen with a microscope	Growth covers less than 10 % of area studied (naked eye) Growth covers less than 50 % of area studied (microscope) New spores are starting to form
4	Clear growth seen with the naked eye Rich growth seen with a microscope	Growth covers more than 10 % of area studied (naked eye) Growth covers more than 50 % of area studied (microscope)
5	Rich growth seen with the naked eye	Growth covers more than 50 % of area studied (naked eye)
6	Very rich growth	Growth covers almost 100 % of area studied, dense mould growth

Figure 27: The Finnish mould index (Finnish Mould Growth Model, z.d.)

Alongside the mould index there are mould sensitivity and decline classes for materials. Usually the statement can be made that the faster the mould growth starts, the faster the decline is when unfavourable conditions occur. In Figure 28 below the mould sensitivity class and common building materials are shown.

Mould sensitivity class		Construction materials
MSC1	Very sensitive	Roughly sawn and dimensioned plank (pine, spruce and hardwood), planed pine, birch plywood, untreated porous wood fibre board, cardboard-surfaced gypsum board
MSC2	Sensitive	Planed spruce, paper-based bituminized/treated products and films, wood-based glued boards, pine or spruce plywood, bituminized/treated porous fibre board
MSC3	Medium resistant	Mineral wools, plastic-based materials, autoclaved aerated concrete, expanded-clay lightweight concrete, carbonated old concrete, cement-based products, bricks, cement bonded fibreboard, fibreglass-surfaced gypsum board
MSC4	Resistant	Glass and metals, alcaline new concrete, products containing mould-preventive additives

Figure 28: Mould sensitivity class for building materials (Finnish Mould Growth Model, z.d.)

2.5.2 Wood decay

The temperature (red) and moisture (blue) gradient are illustrated in Figure 29 for a non-capillary active insulation applied on the interior side of the solid masonry brick. With the interstitial condensation occurring in between the masonry and insulation layer (Stappers, 2021). The embedded wooden floor beams in the masonry walls are affected by the condensation occurring between the façade layers when high humidity levels and liquid moisture is present around the wooden beam end due to the lower temperatures.

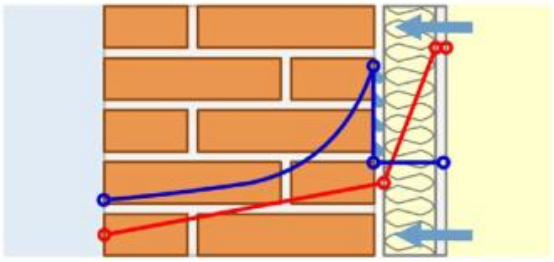


Figure 29: Temperature and moisture gradient with interstitial condensation between layers (standard insulation) (Stappers, 2012)

Another significant cause of moisture in between the façade layers, and thus damage in wooden beam ends, is the result of unexpected incidents occurring many years after the building was constructed, such as the displacement of roof tiles, blocked gutters or other sources of leaks (Fraunhofer IBP, z.d.). Despite the big possible impact, damage mechanism due to water leaks is not taken into account in this work.

Figure 30: Replacement of rotten wooden beam affected by moisture (Frunhofer IBP, z.d.)

2.5.3 Dose response models

Wood has, in the contrary to metals, minerals and other inorganic materials, an extra susceptibility to the colonization and degradation by organisms. Therefore, included in the calculation of service life of wooden components are the biological processes. According to Isaksson et al. (2012), decay fungi and bacteria are the most relevant groups of wood destroying organisms where both temperature and moisture are seen as key factors influencing the wood degradation.

A variation of moisture [u] and temperature [T] in time impacts the potential decay of wood. This is described by Isaksson et al. (2012) as the dose-response relation. This relationship model can be used to analyse the effect of weather and moisture loads or the effect of the geometry itself (e.g. materials, design detailing, etc.). It uses the daily average moisture content in the D_u component and the daily average temperature as component D_t. With an n-number of days the total dose is given by:

$$D(n) = \sum_{i}^{n} (f(D_t(T_i), (D_u(u_i)))$$

Where T_i and u_i are the average temperature and moisture values for day i. Dose-response relationship models are derived from material climatic data and corresponding decay development in field tests. Two types of models can be used: the logistic dose-response model and the two-step dose response model, respectively focussing on describing the relation between exposure and decay rating of moisture traps with long period of high moisture contents and predicting the behaviour or a wider, more simplified, sense where periods or high moisture content are interrupted by periods of drier and/or colder climates. According to (Niklewski, 2024) for this application (with the ability to calculate full moisture distribution by the use of Delphin) the use of the logistic dose-response model is the most suitable.

2.5.4 Logistic dose-response model

This model is strictly based on the physiological base parameters of wood decay fungi in combination with the corresponding exposure conditions, which refers to the biological characteristics and known growth and survival rates of these fungi in specific conditions. The model is therefore developed to reach a logistic relationship (growth rate decreases as quantity increases) between the moisture, temperature and wood decay. Resulting in two dose function for moisture component D_u and temperature component D_t and a total dose function.

Boundary conditions:

$$D_u(u)\begin{cases} 0 \; if \; u < 25\% \\ D_u(u)^* \; if \; u \geq 25\% \end{cases}$$

$$D_t(T)\begin{cases} 0 \; if \; T_{min} < 0 \; or \; if \; T_{max} > 40\,°C \\ D_t(T)^* \; if \; T_{min} \geq 0\,°C \; or \; if \; T_{max} < 40\,°C \end{cases}$$

The boundary conditions set are part of the biological conditions under which the decay fungi are able to grow. Visualised in Figure 31.

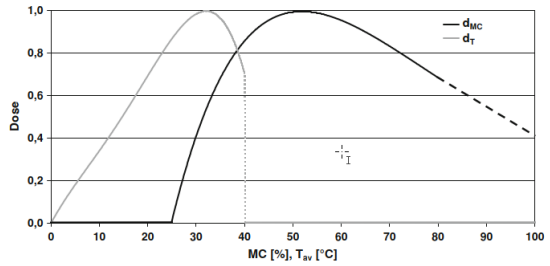


Figure 31: Function Du and DT (Brischke, 2015)

When boundary conditions are met, components D_u and D_T are calculated with:

$$D_u(u)^* = e * u^5 - f * u^4 + g * u^3 - h * u^2 + i * u - j$$

$$D_T(T)^* = k * T^4 + l * T^3 - m * T^2 + n * T$$

Total dose is calculated when both components are above zero (the suitable conditions for fungi growth). With an additional temperature weighing factor a:

$$\text{if } D_u > 0 \text{ and } D_t > 0$$
$$D = (a * D_T[T] + D_u[u]) * (a + 1)^{-1}$$

where:

D_u = moisture dose component, [d];
D_T = temperature dose component, [d];
D = dose, [d];
u = moisture content, [%];
T = temperature, [°C];

variables*:

a =	3.2;	j =	4.98;
e =	6.75 * 10^{-10};	k =	-1.8 * 10^{-6};
f =	3.50 * 10^{-7};	l =	9.57 * 10^{-5};
g =	7.18 * 10^{-5};	m =	1.55 * 10^{-3};
h =	7.22 * 10^{-3};	n =	4.17 * 10^{-2}
i =	0.34;		

Figure 32 shows the daily dose as a function of temperature and moisture content. Clearly making the dominance of the temperature component visible.

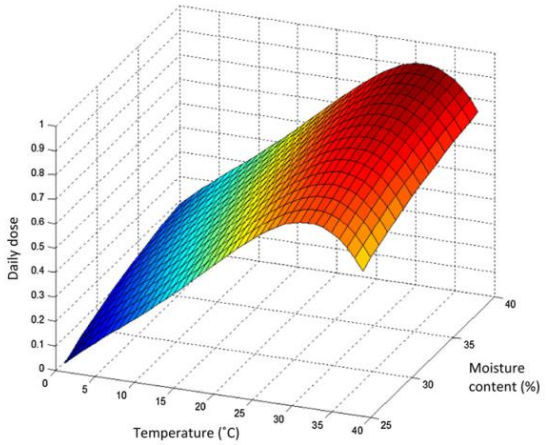


Figure 32: Relationship between daily dose, temperature and moisture content

Finally, decay rating can be calculated, this is used to predict service life according to the EN 252 and is divided into five steps: 0 (sound), 1 (slight attack), 2 (moderate attack), 3 (severe attack, or 4 (failure). Decay in non-shaded wooden-elements is initiated sooner than shaded elements and is different for multiple types of rot. The relationship between exposure and decay rating for shaded and non-shaded brown rot and white/soft rot is given by (Brischke & Meyer-Veltrup, 2015) and shown in Figure 33. Softwood species (Douglas fir hardwood, Scots pine sapwood, Larch hardwood, Larch sapwood and Norway Spruce) are used as wood types for these functions.

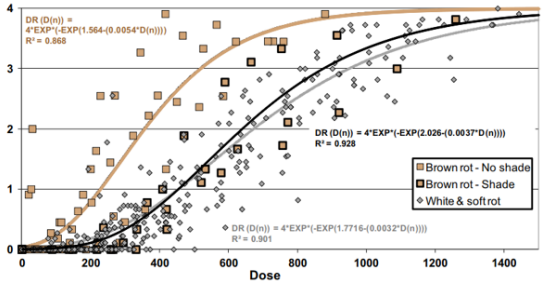


Figure 33: Relationship between exposure and decay rating for different softwood species (Brischke & Meyer-Veltrup, 2015)

Wood decay caused by brown rot is used as the most conservative model, due to the fact that brown rot is initiated earlier and proceeds faster. Therefore the Dose Response (DR) models of brown rot are used.

*Published variables for k and n are sometimes incorrect. These correct values are obtained through personal communication with the author of those papers: Jonas Niklewski.

Brown rot, non-shaded DR relation:

$$DR\big(D(n)\big) = 4 * \exp\left(-\exp\Big(1.564 - \big(0.0054 * D(n)\big)\Big)\right)$$

Brown rot, shaded DR relation:

$$DR\big(D(n)\big) = 4 * \exp\left(-\exp\Big(2.026 - \big(0.0037 * D(n)\big)\Big)\right)$$

where:

$D(n)$ = dose for n days of exposure, [d].

These decay calculations are suitable for every type of wood, deviations due to wood species-specific properties are taken into account in the following part on the resistance dose. The calculated dose from the DR functions is called the exposure dose (D_{ed}). For brown rot at decay rating 1 (where decay starts to occur), we find critical dose (D_{crit}):

D_{crit} = 225 d for non-shaded wood
D_{crit} = 460 d for shaded wood

2.5.4.1 Inherent material resistance

Meyer-Veltrup et al. (2017) has included the effect of wood type properties, for example the decay resistance in different species of wood, into the assessment of wood decay. This resistance can be natural, modified or treated and is defined as the resistance dose (D_{rd}). In general, the evaluated geometry is accepted when:

$$D_{ed} < D_{rd}$$

where:

D_{ed} = exposure dose, [d];
D_{rd} = resistance dose, [d].

Exposure dose is the result from the logistic dose-response model of Isaksson et al. (2012) and the resistance dose is calculated with:

$$D_{rd} = D_{crit} * k_{inh} * k_{wa}$$

where:

D_{rd} = resistance dose, [d];
D_{crit} = critical dose, [d];
k_{inh} = factor for inherent durability, [-];
k_{wa} = factor for wetting ability, [-].

The resistance of a wooden element is influenced by microclimate at the wood surface (relative humidity, temperature, presence of water, etc) and the inherent moisture dynamics (permeability, surface properties, etc.) Where the microclimate can be seen as boundary conditions and the moisture dynamics as the moisture transport into the bulk of the wood. Both relative to the Norway Spruce, that is used as the reference wood type.

Meyer-Veltrup et al. (2017), has compiled values for k_{wa} and k_{inh} after laboratory durability tests and field trials in Norway, Germany and Sweden that can be used in international standards to calculate the resistance dose of a specific wood type. Visualised in Figure 34.

Table 1 Moisture content after exposure to different moistening regimes, water release during drying and capillary water uptake

Wood species	$W24_{submersion}$		$W24_{100\%\ RH}$		$W24_{0\%\ RH}$		CWU	
	Mean (%)	k_{wa} (–)	Mean (%)	k_{wa} (–)	Mean (%)	k_{wa} (–)	Mean (g/cm²)	k_{wa} (–)
Hardwoods								
Norway maple	59.0	0.94	15.1	1.18	15.9	0.99	0.26	0.37
Lime	60.1	0.92	12.4	1.43	13.3	0.83	0.25	0.38
Aspen	60.9	0.91	15.7	1.13	17.4	1.09	0.15	0.65
Birch	66.0	0.84	15.3	1.16	17.4	1.09	0.26	0.36
Alder	63.0	0.88	14.4	1.23	19.5	1.22	0.28	0.35
Rowan	61.2	0.90	15.1	1.18	16.3	1.02	0.25	0.39
Goat willow	47.4	1.16	13.1	1.36	13.6	0.85	0.16	0.60
European oak	46.3	1.19	12.4	1.44	13.0	0.82	0.14	0.67
Ash	56.4	0.98	14.3	1.25	16.3	1.02	0.24	0.39
Wych elm	48.6	1.14	13.6	1.31	14.6	0.91	0.20	0.49
Beech	60.1	0.92	15.5	1.15	14.6	0.92	0.31	0.31
Cherry	82.6	0.67	15.8	1.13	17.0	1.06	0.39	0.24
Teak	23.2	2.38	6.10	2.90	14.1	0.88	0.08	1.28
Merbau	26.0	2.12	10.9	1.63	10.2	0.64	0.02	4.62
Softwoods								
Sitka spruce	57.2	0.96	16.7	1.07	16.6	1.04	0.13	0.76
Norway spruce 6 mm	55.2	1.00	17.8	1.00	16.0	1.00	0.10	1.00
Silver fir	56.7	0.97	17.5	1.01	14.6	0.92	0.15	0.66
Scots pine 3 mm	66.5	0.83	16.2	1.10	14.7	0.92	0.08	1.19
Scots pine sap	93.2	0.59	17.2	1.03	16.1	1.01	0.34	0.28
WRC (NA)	49.8	1.11	11.1	1.61	11.6	0.72	0.36	0.27
WRC (NO)	69.4	0.80	15.6	1.14	13.4	0.84	0.32	0.30
Juniper	42.6	1.30	12.4	1.43	12.3	0.77	0.08	1.20
Siberian larch	52.9	1.04	13.7	1.30	14.6	0.91	0.21	0.46
European larch	33.0	1.67	9.7	1.83	15.1	0.94	0.13	0.72
Douglas fir	34.4	1.61	11.3	1.57	13.9	0.87	0.18	0.54

Factors accounting for the wetting ability (k_{wa}) of the various materials tested. (W24 = 24 h water uptake and release tests; *CWU* capillary water uptake tests)

Table 2 Mass loss (ML) of wood samples after 16-week incubation according to EN 113 (1996), decay rate v (a^{-1}), and factors accounting for the inherent protective material properties (k_{inh}) of the various materials exposed for 11 years in ground (EN 252) at the Sørkedalen test field, Norway

Wood species	EN 113						EN 252	
	T. versicolor		*C. puteana*		*P. placenta*		v_{mean} (a^{-1})	k_{inh} (–)
	ML (%)	k_{inh} (–)	ML (%)	k_{inh} (–)	ML (%)	k_{inh} (–)		
Hardwoods								
Norway maple	33.0	1.10	40.3	0.65	–	–	1.20	1.56
Lime	34.9	1.04	46.2	0.57	–	–	1.35	1.39
Aspen	36.9	0.99	42.4	0.62	–	–	1.15	1.62
Birch	39.3	0.93	50.4	0.52	–	–	1.43	1.30
Alder	34.9	1.04	40.6	0.64	–	–	1.68	1.11
Rowan	41.4	0.88	36.8	0.71	–	–	1.04	1.79
Goat willow	31.4	1.16	26.4	0.99	–	–	0.76	2.47
European oak	2.5	5.00	0.0	5.00	–	–	0.20	5.00
Ash	32.5	1.12	2.1	5.00	–	–	0.77	2.43
Wych elm	32.6	1.12	2.5	5.00	–	–	0.80	2.32
Beech	30.0	1.21	32.1	0.81	–	–	1.41	1.32
Cherry	17.3	2.11	11.2	2.32	–	–	–	–
Teak	−3.2	5.00	−4.2	5.00	–	–	0.10	5.00
Merbau	2.8	5.00	0.9	5.00	–	–	0.10	5.00
Softwoods								
Sitka spruce	31.6	1.15	16.0	1.63	26.4	0.95	1.67	1.12
Norway spruce 6 mm	36.4	1.00	26.1	1.00	25.0	1.00	1.87	1.00
Norway spruce 3 mm	38.2	0.95	27.3	0.96	23.2	1.08	1.73	1.08
Norway spruce 1 mm	33.2	1.10	27.8	0.94	25.3	0.99	0.92	2.02
Silver fir	29.9	1.22	15.5	1.67	18.4	1.36	1.19	1.56
Scots pine 3 mm	15.1	2.42	0.5	5.00	21.9	1.14	0.79	2.36
Scots pine 1 mm	13.9	2.62	3.4	5.00	16.8	1.49	–	–
Scots pine sap	33.6	1.08	41.8	0.62	35.5	0.70	1.06	1.76
WRC (NA)	0.8	5.00	0.0	5.00	0.0	5.00	0.95	1.97
WRC (NO)	8.7	4.19	0.0	5.00	0.0	5.00	1.43	1.31
Juniper	2.8	5.00	0.4	5.00	0.4	5.00	0.26	5.00
Siberian larch	23.6	1.55	3.4	5.00	24.7	1.01	0.31	5.00
European larch	3.7	5.00	3.1	5.00	10.5	2.38	0.15	5.00
Douglas fir	3.8	5.00	0.7	5.00	16.8	2.59	0.17	5.00

Figure 34: K_{wa} and K_{inh} values for different wood species (Meyer-Veltrup et al., 2017)

2.6 Simulation of hygrothermal performance

Hygrothermal, or Heat, Air, Moisture (HAM) simulation software plays a big role in assessing the response of wall assemblies to climate loads, providing insights into potential deterioration risks such as mould growth, wood rot, and other degradation. As highlighted by Defo et al. (2021), numerous numerical simulation packages are available, dedicated wholly or partially to heat, air, and moisture transfer simulations for building materials and wall assemblies.

These simulation tools enable detailed numerical analyses to calculate the moisture response of materials and components under varying boundary conditions (Defo et al., 2021). By predicting moisture and temperature conditions within building envelope assemblies over time, hygrothermal simulation enhances understanding of how the envelope interacts with the interior and exterior environment, aiding in the early identification of potential moisture-related issues (Glass et al., 2013).

Given that moisture problems in buildings are linked to health problems and are a leading cause of construction litigation claims, finding possible problems during the design phase is essential. Hygrothermal analysis methods range from simplistic steady-state models to advanced computer simulations, with the advanced simulations offering detailed insights into heat, vapour, liquid, and air transfer. (Glass et al., 2013).

2.6.1 Different types of hygrothermal simulation software

Among the available models, as can be found in the research of Delgado et al. (2010), a select few are found as promising choice for hygrothermal software. With WUFI and DELPHIN standing out as commonly used platforms by building practitioners, designers, academia, and researchers worldwide (Defo et al., 2021).

To assess the frequency and popularity of these software tools in academic studies, an extensive search was conducted across multiple scholarly databases by Yildiz (2021). The search included databases as Google Scholar, Web of Science, Scopus, and ScienceDirect, focusing on articles published in peer-reviewed academic journals and written in English. This study excluded conference papers, periodicals, books, theses, and other non-academic publications from consideration.

The results of this research revealed that the WUFI program emerged as the most frequently utilized software in academic studies compared to other simulation tools. According to Yildiz (2021), DELPHIN follows closely as the second most widely employed simulation tool after WUFI. WUFI and DELPHIN were encountered 1035 and 669 times, respectively, across the different databases, significantly more than the third most commonly found program, which appeared 234 times.

Software	Type
1D-HAM	1D heat-air-moisture
BSim2000	1D heat-moisture
DELPHIN 5	1/2D heat-air-moisture-pollutant-salt
HAM	1D heat-air-moisture
HAMLab	1/2/3D heat-air-moisture
hygIRC-1D	1D heat-air-moisture
hygIRC-2D	2D heat-air-moisture
LATENITE	2D heat-moisture
MATCH	1D heat-air-moisture
MOISTURE-EXPERT	1/2D heat-air-moisture
WUFI-2D	2D heat-moisture
WUFI-ORNL	1D heat-moisture
WUFI-Plus	1D heat-moisture
WUFI-Pro	1D heat-moisture

Figure 35: A number of hygrothermal programs (Delgado et al., 2010)

2.6.2 Input in simulation software

Correct input in hygrothermal software is crucial for finding reproducible simulation results. Different input parameters can be set in the software:

- Assembly of building component
- Material properties
- Simulation properties
- Interior / Exterior conditions
- Performance criteria

In the initial phase of a hygrothermal analysis, defining the assembly, its orientation, and boundaries is important. Typically, simplification into a one-dimensional representation is sufficient, although two dimensions or three dimensions may be required for complex areas like corners or intersections. For the materials, different properties are specified, including thickness, bulk density, specific heat, thermal conductivity, moisture storage characteristics, vapour permeability, liquid water diffusivity, and possibly porosity, capillary saturation, maximum saturation, and airflow permeability. Some of the software programs have a material database included. But (new) materials can also be added. (Glass et al., 2013).

While most models exclude airflow, exceptions exist, with some incorporating cladding ventilation effects. These can be crucial for drying potential in the construction. Air leakage through insulated envelope assemblies is recognized as a significant moisture transfer mechanism, and has high influence. Certain building materials, such as

concrete, wet-spray cellulose insulation, and wood, may contain moisture when installed, high initial moisture loads can be added to the simulation program to represent these loads.

Interior conditions, like temperature and humidity, are important especially in cold climate design analyses. Similarly, exterior conditions, encompassing wind, rain, temperature, humidity, and solar radiation, are important climatic parameters.

To evaluate design analysis results effectively, performance criteria are used. For wood-based or biobased components, potential concerns include wood decay, mould growth, corrosion of metal fasteners, expansion/contraction damage, and loss of structural capacity. These performance criteria are used as base for the assessment of building envelope designs on these hygrothermal risks. (Glass et al., 2013).

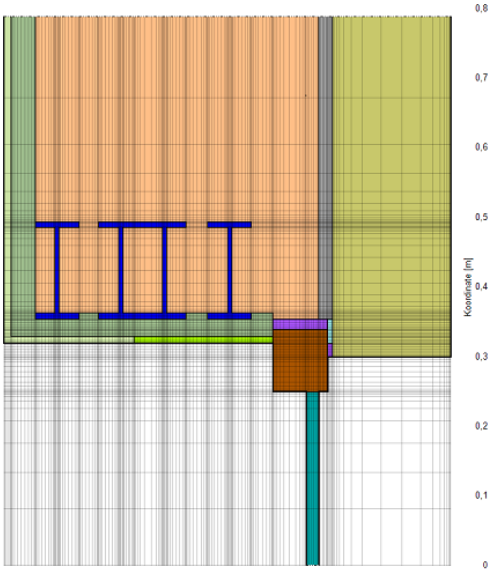


Figure 36: Building component assembly in DELPHIN 6 (DELPHIN, z.d.)

2.6.3 Calculation types

Hygrothermal simulation models differ significantly in their mathematical calculation types, which depends on factors such as the moisture transfer dimension, type of flow (steady-state, quasi-static, or dynamic), quality and availability of information, and the variable data. For example material properties, weather conditions and construction quality. (Straube et al., 1991)

Hygrothermal simulation tools, among them DELPHIN and WUFI, typically rely on numerical methods for making a detailed analysis of heat and moisture transfer within building assemblies. These are used to solve mathematical models of physical systems, mostly differential equations. The most common numerical methods used in these simulations include (Delgado et al., 2013):

- Finite Difference Method (FDM)
- Finite Control Volume (FCV)
- Finite Element Method (FEM)

DELPHIN and WUFI both use the Finite Control Volume, where the model is divided in small control volumes. Differential equations approximated heat and moisture in these smaller volumes numerically (Defo et al., 2021).

2.6.4 Output of simulation software

Output options include temperature and relative humidity profiles, moisture content, heat flux, vapour pressure, liquid water transport, mould growth potential, structural integrity, and drying/wetting cycles.

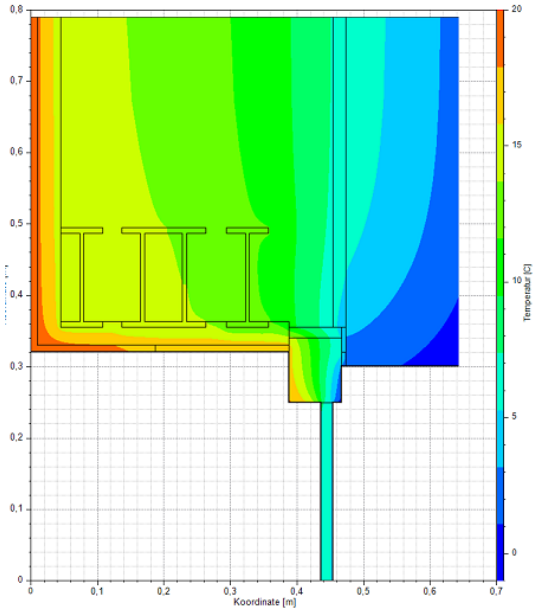


Figure 37: Building component temperature visualisation in DELPHIN 6 (DELPHIN, z.d.)

2.6.5 Validation of results

Both DELPHIN and WUFI are validated using test setups and other methods. Additionally, A number of studies have compared the two WUFI and DELPHIN software programs. Delgado et al. (2013) used two hygrothermal models for verification purposes. DELPHIN was used to verify the WUFI model. The results for the temperature were good but there was some disagreement for humidity. Carbonez et al. (2015) used the two programs to simulate a water leak in a wood frame wall. The results were validated by experimental tests and it was concluded that DELPHIN gives more realistic results for moisture (Hejazi et al., 2020).

2.6.6 Differences DELPHIN and WUFI

Both DELPHIN and WUFI are commonly used HAM simulation software, but they use different driving potentials as well as storage and transport equations. Simplifications and approximations when implementing materials properties and boundary conditions cause variations. Consequently changing the results of the simulations. WUFI is relatively simpler, requiring a limited number of standard material properties data while the equations in DELPHIN are more complicated (Dang et al, 2023). The heat and moisture balance equations in WUFI PRO 6 and DELPHIN 6 are shown in the following balance equations:

WUFI heat balance equation:

$$\frac{\partial H(T)}{\partial t} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + L_v \frac{\partial}{\partial x}\left(\frac{\delta_a}{\mu}\frac{\partial p_v}{\partial x}\right)$$

WUFI moisture balance equation:

$$\frac{\partial \omega(\varphi, T)}{\partial t} = \frac{\partial}{\partial x}\left(D_\omega \frac{\partial \omega}{\partial \varphi}\frac{\partial \varphi}{\partial x}\right) + \frac{\partial}{\partial x}\left(\frac{\delta_a}{\mu}\frac{\partial p_v}{\partial x}\right)$$

DELPHIN heat balance equation:

$$\frac{\partial H(T, p_c)}{\partial t} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + (L_v + c_v T)\frac{\partial}{\partial x}\left(K_v \frac{\partial p_v}{\partial x}\right) + c_l T \frac{\partial}{\partial x}\left(K_v \frac{\partial p_c}{\partial x}\right)$$

DELPHIN moisture balance equation:

$$\frac{\partial \omega(p_c)}{\partial t} = \frac{\partial}{\partial x}\left(K_l \frac{\partial p_c}{\partial x}\right) + \frac{\partial}{\partial x}\left(K_v \frac{\partial p_v}{\partial x}\right)$$

where:

H = heat density, [J/m^3];
T = absolute temperature, [K];
t = time in hours, [s];
λ = thermal conductivity, [W/mK];
L_v = latent heat of evaporation, [J/kg];
δ_a = vapour permeability of air, [kg/msPa];
μ = vapour diffusion resistance factor, [-];
p_v = partial vapour pressure, [Pa];
p_c = capillary pressure, [Pa];
ω = water content, [kg/m^3];
D_ω = moisture diffusivity, [m^2/s];
φ = relative humidity, [-].
c_v = vapour specific heat capacity, [J/kgK];
c_l = liquid specific heat capacity, [J/kgK];
K_v = vapour permeability, [kg/msPa];
K_l = liquid permeability, [kg/msPa].

(Dang et al, 2023)

When solely looking at moisture transport in the simulation software, there are a few differences between the two HAM simulation programs: WUFI uses relative humidity and vapour pressure and DELPHIN uses vapour pressure and capillary pressure as moisture transport calculations. These differences in material properties can be transformed into one another:

Liquid transport:

$$D_\omega(\omega)\frac{\partial \omega}{\partial p_c}\frac{\partial p_c}{\partial x} = K_l(\omega)\frac{\partial p_c}{\partial x}$$

Vapour transport:

$$\frac{\delta_a}{\mu(\varphi)}\frac{\partial p_v}{\partial x} = K_v(\omega)\frac{\partial p_v}{\partial x}$$

(Dang et al, 2023)

For liquid transport, the diffusivity is translated into the permeability and for vapour transport the vapor permeability is translated into the vapor resistance factor.

2.7 Building standards

2.7.1 Dutch NEN-EN-ISO 13788

The Dutch guideline focusses only on moisture transfer by vapour diffusion, and provides simplified calculation methods when other sources of moisture, such as rain penetration, convection and moisture transport by air flow, do not have to be considered.

The NEN-EN-ISO 13788 gives calculation methods for: (1.) The internal surface temperature of a building component or building element below which mould growth is likely, given the internal temperature and relative humidity. The method can also be used to assess the risk of other internal surface condensation problems. (2.) The assessment of the risk of interstitial condensation due to water vapour diffusion. The method used does not take account of a number of important physical phenomena including:

- The variation of material properties with moisture content;
- Capillary suction and liquid moisture transfer within materials;
- Air movement from within the building into the component through gaps or within air space;
- The hygroscopic moisture capacity of materials

Therefore, this method can only be used when the effects of these phenomena are negligible.

And (3.) the time taken for water to dry out and the risk of interstitial condensation occurring elsewhere in the component during the dying process.

With **Error! Reference source not found.** and Figure 39 the user can specify the humidity class based on the internal humidity load. When unsure the highest class (dotted line) should be use. With Δv and Δp, respectively internal moisture excess and indoor-outdoor pressure difference, plotted on the vertical and temperature on the horizontal axis (Nederlands Normalisatie Instituut, 2013).

Humidity class	Building
1	Unoccupied building, storage of dry goods
2	Offices, dwellings with normal occupancy and ventilation
3	Buildings with unknown occupancy
4	Sports halls, kitchens, canteens
5	Special buildings, e.g. laundry, brewery, swimming pools

Figure 38: Internal humidity classes (Nederlands Normalisatie Instituut, 2013)

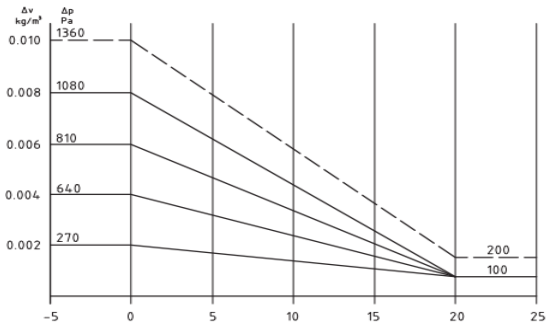


Figure 39: Variation of internal humidity classes with external temperature (Nederlands Normalisatie Instituut, 2013)

2.7.2 German WTA Guideline 6-2

The German WTA Guideline 6-2 also focusses on the internal climate of the building. With occupants and HVAC systems substantially impacting the humidity levels indoors, e.g. moisture production, manual ventilations, mechanical ventilation or air-conditioning systems.

Similarly to the Dutch guidelines internal air temperature and humidity levels can be derived from external air temperature. Additionally, the indoor moisture loads are divided in four levels: high, normal +5%, normal and low moisture levels and be used for assessing moisture loads on building components. (Fraunhofer IRB, 2014)

Moisture load level	Application
High	Buildings with an extraordinary high occupancy
Normal +5%	Houses, normal situations, but with desired additional safeguards
Normal	Houses, normal situations. Including kitchens and bathrooms
Low	Buildings with rooms such as offices, classrooms, sales premises. Approach does not include any safeguard and limits future changes of use

Figure 40: Indoor climate classes (Fraunhofer IRB, 2014)

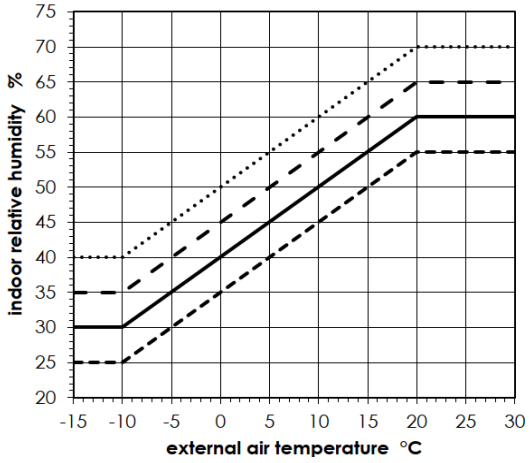


Figure 41: Derivation of internal relative humidity depending on external air temperature (Fraunhofer IRB, 2014)

2.8 Sensitivity analysis on hygrothermal simulation software

This paragraph gives an overview of the possible types of Sensitivity Analysis (SA) options that can be used for hygrothermal simulations and dives deeper into the pros and cons of methods. With this overview, a choice for type of SA is made and more information about this SA is further elaborated.

Simulation software, like Delphin, can simulate heat, air and moisture transport and storage in building components. With the use of this software long term real life test can be avoided. Giving answers to questions that normally need multiple year to be answered within minutes.

But complexity of the software and the lack of knowledge on used building materials, for example when renovating buildings, can cause significant uncertainties in the results of the simulation. Consequently, the mitigation of these uncertainties is important. The techniques of sensitivity analysis are used to gain reliability and confidence in the results of these dynamic hygrothermal simulations. It is used to understand how the variation in the output of a model can be linked to the different variation in the input. It identifies the most influential factors in the model, giving insights into the correct use of the model and therefore generating reliable and correct results.

2.8.1 Types of sensitivity analysis

The type of SA depends on the purpose of the research because it is determined by the choice of the sensitivity index, the estimation method and the sampling strategy. Being the metric that shows how sensitive the output is to change in each input, the way of calculating the index and the way the input values are selected. Similarly, the choice of variation ranges and type of probability distribution that is needed for the research have influence on the type of SA. There are many methods that have been developed but generally follow the same steps. Being:

1. Definition of the target of the research: used to choose the specific type of SA;
2. Selection of input parameters that will be part of the study: the uncertain inputs that will be varied;
3. Set associated uncertainty range and probability distribution for every uncertain input parameter;
4. Application of the sampling-based method: generating the N-number of simulation variations;
5. Execution of the simulation variants: running the model with input variations;
6. Analysis of simulation results: to quantify the effect of uncertain inputs on the model response.

In general, there are three main types of SA: the local sensitivity analysis, the global sensitivity analysis and the screening method. (Goffart & Woloszyn, 2021)

2.8.1.1 Local sensitivity

Local sensitivity analysis, which analyses the impact of small changes in individual input values, is generally not suitable for hygrothermal simulation due to the non-linear nature of these models. This method only analyses sensitivity at a single point and the effect between points is not analysed. For example, the effect on indoor temperature by just varying the insulation value of a wall is approximately linear because of the direct relationship between input and output. While this works for linear models, where sensitivity is a simple calculation, it's problematic for non-linear models. These local sensitivity method don't account for interactions between inputs, leaving a large part of change in output due to input interaction unanalysed. For hygrothermal simulations, this can miss important insights, especially since these interactions in non-linear models have significant impact on the results. (Goffart & Woloszyn, 2021)

2.8.1.2 Global sensitivity

On the contrary to local sensitivity, there is the global sensitivity. It examines how uncertainty in each input variable, over the whole range, affects the output of the model. And can thus be used for hygrothermal simulations due to the effect that variables have on each other is also tested and analysed. According to Goffart and Woloszyn (2021) a commonly used sensitivity analysis used for these problems is the Sobol method.

The Sobol method, developed by Ilya Sobol, is a quantitative method that gives the percentage of total output variance that each parameter accounts for. The method is a variance based method to quantify the impact of uncertainties in variables on the uncertainty in the model output. This method is quite computationally expensive but gives accurate results. The Sobol method is a variance-based method that used and estimates first-order sensitivity indices, higher–order indices and total indices.

The first-order term represents the variance in model output due to the effect of one individual variable. The higher-order terms analyses the interaction between two or more variables to the output and the total-order indices relates to all direct and indirect output variance from one input parameter (Nielsen, 2012).

2.8.1.3 *Screening method*

The screening method is used to provide a temporary insight into the influence of the inputs on the output results. This is useful when there are many potential input variables but the importance and influence of the input parameters is unknown. With many input parameters a fast and simple analysis can be completed and the number of variable input parameters can be limited. This limited number can later be used for a more rigorous analysis. A commonly used screening technique is the Morris method.

The Morris method is a screening method to find parameters that are important or negligible. The method varies one parameter at a time. Each parameter is divided into a number of values that are chosen within the range of variation. The method calculates two sensitivity measures for each parameter: Morris mu and Morris sigma. A high Morris mu indicates a parameter with an important overall influence on the output. A high Morris sigma indicates either interaction with other parameters or a parameter that is non-linear. If both the mu and the sigma are low then this parameter is negligible. With this information a ranking of input parameters in order of importance can be made. (Nielsen, 2012)

2.8.2 Sobol method

Without the need for a screening method to find the most influential input parameters and the local sensitivity analysis being insufficient for the goal that needs to be reached with this research it was chosen to work with the Sobal SA on the hygrothermal performance of the indoor insulated solid brick masonry wall structure.

2.8.2.1 *Sobol calculation*

The Sobol SA is based on a quantitative method. A significant number of simulation variants is created to calculate the first-, higher- and total order indices. Zhang et al. (2015), clearly summarizes the Sobol theory. This can be found in the following part and is explained with the use of the flow chart typically used in Sobol analysis. See, Figure 42.

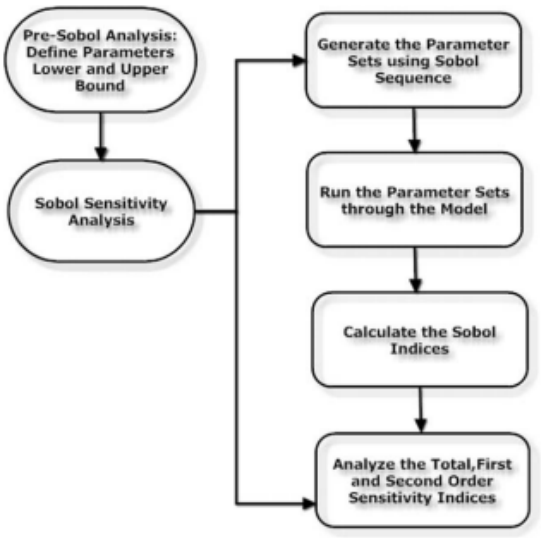


Figure 42: Sobol flow chart (Zhang et al. 2015)

Before starting the Sobol SA, the number of parameters and lower and upper bounds are set. To explain we use $x = (x_1, x_2, \ldots, x_n)$ as input parameters. These are given with finite interval ranges, it has set min and max endpoints and after rescaling a variable can be seen as uniformly distributed between [0, 1]. The model that is checked on sensitivity is given as a function of x, being f(x). Within the function output being an average model output (when all input parameters are set on their mean value) that is set as f(0) and the variation set as (D) where this variation represents the deviation between f(x) and f(0) and thus the variation of one parameter on the total output. Variance and mean value are set as integrals with bounds [0, 1].

$$f(0) = \int f(x)dx$$

$$D = \int f(x)^2 dx - {f_0}^2$$

The basis of the Sobol method is the decomposition of the variance in output (D) into contributions that are effected by every single input parameter. This can be written as:

$$f(x) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n}\sum_{j\neq i} f_{ij}(x_i, x_j) + \cdots + f_{i\ldots n}(x_i, x_j, \ldots, x_n)$$

Where f(0) is the mean output value, f_i (x_i) representing the first order indices and f_{ij} (x_i,x_j) representing the second order indices.

The total output variance (D) is calculated and can be written as an summation of the first order, second order and higher order variance:

$$D = \sum_{i=1}^{k} D_i + \sum_{i<j} D_{ij} + \cdots + D_{1,2,\ldots,k}$$

where:

$$k = i_1, i_2, \ldots, i_n$$

The final Sobol sensitivity indices are calculated. Within this research only first order (S1), second order (S2) and total order (ST) indices are calculated, this is done as following:

General:

$$S_{i,j,\ldots,n} = \frac{D_{i,j,\ldots,n}}{D}$$

First, second and total order:

$$S_i = \frac{D_i}{D} \; and \; S_{ij} = \frac{D_{ij}}{D} \; and \; S_T = S_i + S_{ij}$$

The total sum of all sensitivity indices should be equal to 1, indicating the full variance in model output can be linked to the individual and combined effects from input parameters. If it's less than 1, this can indicate that some (important) variables have been missed.

2.8.2.2 Sobol analysis

Additional to the calculation of indices the assessment of the indices and the reliability of the Sobol SA is important to understand. The assessment of indices need to be reliable and this reliability is directly related to the sample size created for the SA. With the sample size being the number of parameter sets generated, being the same as the number of model simulations to complete. One simulation for every parameter set. This size is dependent on two main factors; (1) the complexity of the model and (2) the number of parameters that are evaluated. (Zhang et al., 2015)

The sampling technique used for Sobol SA is often Monte Carlo or quasi-random sampling. With Monte Carlo a random selection of values is generated. In a quasi-random technique there is some sample that is independent to another extra relation between the random samples. There is more uniform coverage and thus a more precise result with the same number of samples as the Monte Carlo sampling. Sobol has its own Python library that can be used to create the samples. This is a quasi-random based sampling called Sobol sequences. (Lemieux & Lemieux, 2009)

The numer of samples can be calculated with the formula:

$$N = n * (p + 2)$$

where:

N = sample size, [st];
n = number of samples per parameter, [st];
p = number of parameters, [st].

This number of samples per parameter influences the computational time significantly. Commonly a number to the power of 2 is used. Starting with 32 or 64 for simple models and 512 or 1024 for complex models. (Salib, z.d.) When performing the Sobol method an optimal point needs to be found between computational time and precision in results when calculating sensitivity indices.

The higher the sensitivity indices value, the more influence the input parameter has on the total output. Zhang et al. (2015), mentions a number of 0.05 as a frequently accepted distinction value between important and unimportant parameters. This number is primarily used for complex models and may be not sufficient enough for simple models.

A final visualisation is used in forms of tables, scatterplots or column chart to enhance the interpretation of calculated parameter indices and the influences on the model output. In Figure 43 and Figure 44 examples of these visualisations are given.

S_i		Sobol' value	Bootstrap average	95% CI percentile meth.		S_{Ti}
1	CN2	0.252	0.251	0.205	0.297	1
2	CH_N	0.069	0.069	0.035	0.106	2
3	GWQMN	0.067	0.066	0.047	0.085	3
4	RCHRG_DP	0.025	0.026	0.012	0.040	4
5	SLSUBBSN	0.019	0.019	0.010	0.028	5
6	SOL_K	0.014	0.014	0.006	0.021	6
7	CH_K2	0.011	0.011	−0.007	0.033	7
8	SURLAG	0.010	0.010	−0.002	0.022	8
9	SOL_AWC	0.007	0.007	−0.002	0.016	9
10	GW_REVAP	0.007	0.007	0.003	0.011	10
...	...	...	...	...	...	...
13	SMTMP	0.001	0.001	0.000	0.003	17
...	...	...	...	...	...	...
26	ALPHA_BF	0.000[a]	0.000[a]	−0.026	0.017	26
	SUM	0.482				

Figure 43: Table indices visualisation (Nossent et al., 2011)

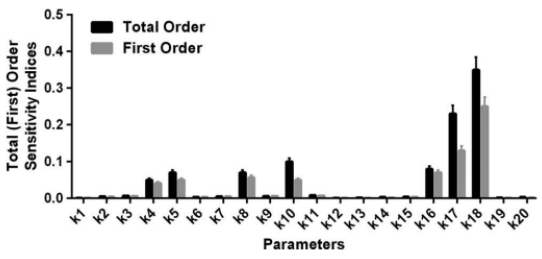


Figure 44: Column chart indices visualisation (Zhang et al., 2015)

3 Material parameter study

Delphin hygrothermal simulation software is used to calculate the wood decay in the beams. Several factors, like weather, geometry of the construction, and materials affect this wood decay significantly. This chapter is written to give an comprehensive overview of the input and output data used and corresponding reasoning behind the choices that are made in the wood decay simulations. For example: program settings, boundary conditions, materials used and output files.

The chapter consist of multiple paragraphs, arranged in order of use when running the simulations. Prior to the simulations parts there is a paragraph on the full methodology of the study. Giving an overview of the layout of the study, where use of multiple programs, exchange of in- and output of data and Python integration is described.

3.1 Methodology

This paragraph gives an overview of the steps taken in order to find the most influential material parameters that effect wood decay. Multiple programs are used to do the simulations.

Running the simulations is divided into two parts: Creating the material and simulation variations and running and calculating the results. A schematic overview of the methodology is found on the next page and a more extensive methodology and the used Python scripts are found in the appendix.

3.1.1 Part 1: Creating variations

To find the most influential material parameters affecting the risk on wood decay, a sensitivity analysis is executed. This analysis creates a number of simulations with different material parameters and analyses the simulating results. This gives insights in the effect of the material changes to the total result but also the affect to other material properties.

Therefor new material and simulation files need to be created. These can afterwards be used in Delphin to calculate the moisture and temperature in the geometry. The steps in short:

1. Within Delphin a template model is made, in this model the geometry is created and all program settings are set correctly

2. In the next step the simulation and material template files are made. The new parameters resulting from the Sobol samples, are places in the template files specific placeholders, creating similar simulation and material files.

3. In Python, with additional libraries, the generation of Sobol samples is executed. These values are stored in a dataset and saved to a .csv file.

4. A second Python script imports the dataset and replaces al the values to the correct material placeholders. Additionally the script calculates and replaces the new moisture storage functions by using the Van Genuchten moisture retention curve function.

5. For every Sobol sample a simulation job is written. All job lines are written and stored in a .csv file. Python writes and uses these jobs to run the Delphin software .

3.1.2 Part 2: Running and calculating results

This part imports and calculates the dose and exports the result files to be used in visualisations. The steps in short:

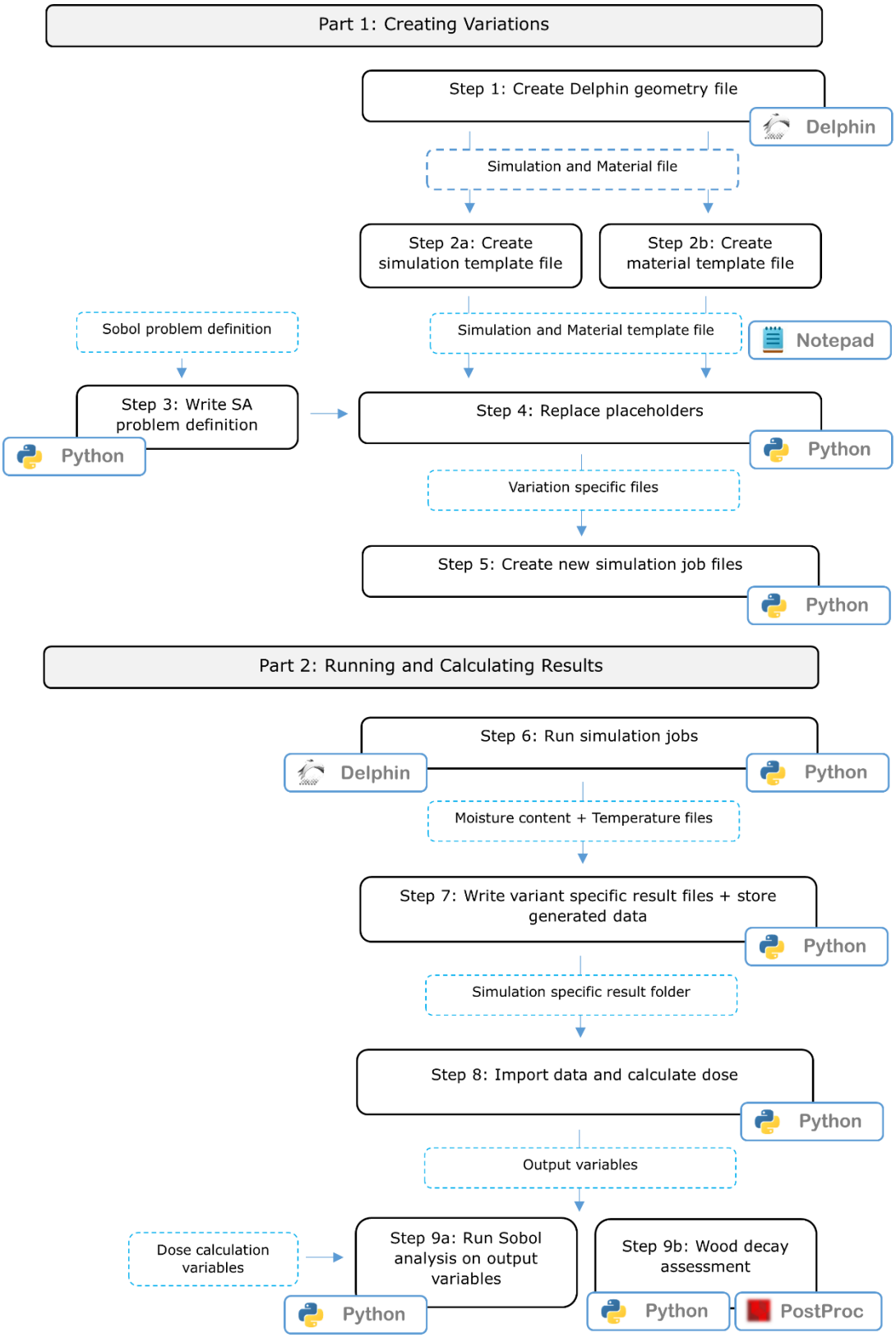
6. The second Python script additionally imports the .csv file with the Delphin executable jobs. The JobRunner.py is used to run multiple jobs in parallel.

7. Python script three imports these calculated results for every simulation and stores this data in multiple Python lists. For every simulation one moisture list and one temperature list. The next step is the calculation of the daily dose. This is calculated for all coordinates on all days and uses the corresponding temperature and moisture data of that specific coordinate of the specific day.

8. The values that are analysed by the Sobol analyser are calculated with script three and saved to a .csv file. The final Python script (four) imports the Sobol values and calculates the First, Second and Total order indices for all the four sensitivity analysis.

With the script the Sobol values are plotted in graphs. Contributing to the analysis of the results indices.

9. The final part is the visualisation of the calculated dose. The script imports the dose template file, calculates the dose values from the moisture, temperature and coordinate data and export the dose result file to be visualised in Delphin PostProc:


Part 1: Creating Variations
Step 1: Create Delphin geometry file
Delphin
Simulation and Material file
Step 2a: Create simulation template file
Step 2b: Create material template file
Sobol problem definition
Simulation and Material template file
Notepad
Step 3: Write SA problem definition
Python
Step 4: Replace placeholders
Python
Variation specific files
Step 5: Create new simulation job files
Python
Part 2: Running and Calculating Results
Step 6: Run simulation jobs
Delphin
Python
Moisture content + Temperature files
Step 7: Write variant specific result files + store generated data
Python
Simulation specific result folder
Step 8: Import data and calculate dose
Python
Output variables
Dose calculation variables
Step 9a: Run Sobol analysis on output variables
Python
Step 9b: Wood decay assessment
Python
PostProc

3.2 Input

The input required for the hygrothermal simulations in Delphin is clarified below. The input settings are divided in: geometry, location, outdoor boundary conditions, indoor boundary conditions, discretization grid and Sobol settings. Input with regard to material parameters, is elaborated in paragraph 3.3.

3.2.1 Geometry

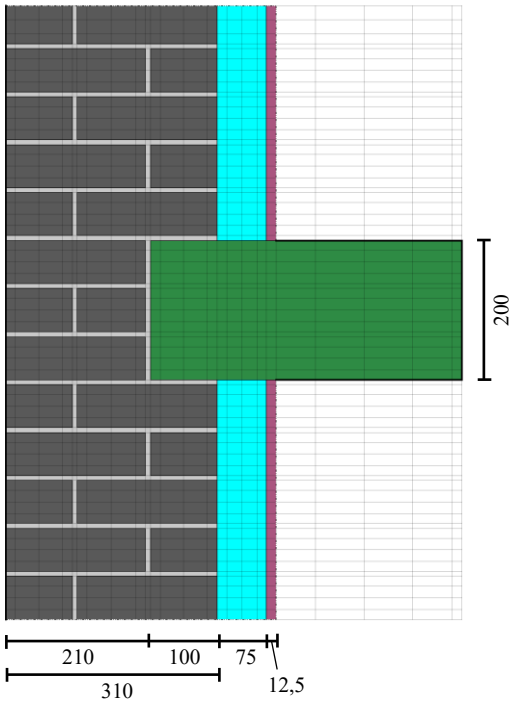


Figure 45: Simulated Delphin geometry

In Figure 45 the 2D simulated geometry is illustrated. It consists of a standard solid one and a half thick masonry wall. Interior insulation of 75 mm and a gypsum board of 12.5 mm as finishing layer. The wooden beam embedded in the masonry wall is 200 mm high. Total geometry is 1000 mm high and 700 mm wide. The moisture and temperature values gradually shift to a 1D profile on the top and bottom of the total geometry. Initial conditions for the geometry are:

Initial temperature	20	[°C]
Initial relative humidity	80	[%]

From Ruisinger en Kautsch (2020) it has become clear that three dimensional simulation increase computational time enormously but results are almost similar to two dimensional simulations. Additionally adjusting the simulation files with python script become much more difficult. Consequently a 2D simulation (cartesian grid) is used for the simulations.

3.2.2 Location

Within the Delphin climate database multiple climate files are available for different locations. It is possible to add additional climate data files into the Delphin program. For this research, the location chosen for the simulation is Essen (Longitude: 6.97 deg, Latitude: 51.40 deg, Elevation: 152 m), Germany. No sensitivity analysis on the location is conducted. The measured data is from 2011.

Figure 46: Simulation location: Essen, Germany (Google Maps, 2024)

3.2.3 Outdoor boundary conditions

For Essen, the climate data on temperature [°C], shortwave solar radiation [W/m^2], relative humidity [%], sky radiation [W/m^2] and rain [l/m^2h] is illustrated in Figure 47. The simulated wall is orientated 270 deg (west) and has a 90 deg (vertical) angle. This is the orientation with the highest rain and wind load and therefor the risk on wood decay is the highest.

In Delphin multiple climate conditions can be turned on/off, depending on the situation that needs to be simulated. The sensitivity analysis run with heat conduction, vapor diffusion, short-wave solar radiation, long-wave solar radiation and wind driven rain (DIN EN ISO 15927-3) switched on. Corresponding values:

Conv. heat coefficient	12	[W/m²K]
Eff. heat coefficient	12	[W/m²K]
Vapor diff. coefficient	7.5e-08	[s/m]
Solar abs. coefficient	0.7	[-]
Long wave emissivity	0.9	[-]
Splash reduc. coefficient	0.7	[-]

The same one year climate data files are used three times in order to simulate the full three years. Outdoor climate data used in the simulations:

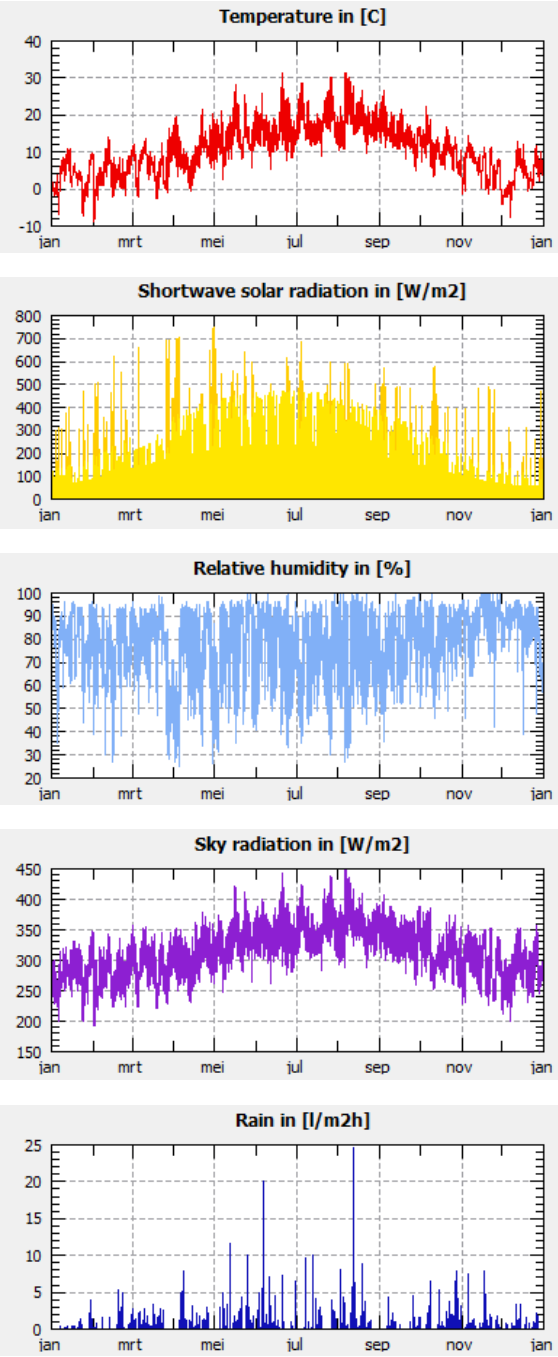


Figure 47: Outdoor temperature [°C], shortwave solar radiation [W/m^2], relative humidity [%], sky radiation [W/m^2] and rain [l/m^2 h] (Delphin, 2024)

3.2.4 Indoor boundary conditions

Similarly to the outdoor conditions, the indoor conditions are dynamic. Using the standard German WTA adaptive indoor climate model (DIN EN 15026) with additional WTA 6.2 increased moisture load of 5%. Indoor condition values are set to:

Upper temperature	25	[°C]
Lower temperature	20	[°C]
Upper relative humidity	65	[%]
Lower relative humidity	35	[%]
Surf. heat transfer coef.	8	[W/m²K]
Surf. vapor diff. coef.	2.5e-08	[s/m]

Indoor climate data used in the simulations:

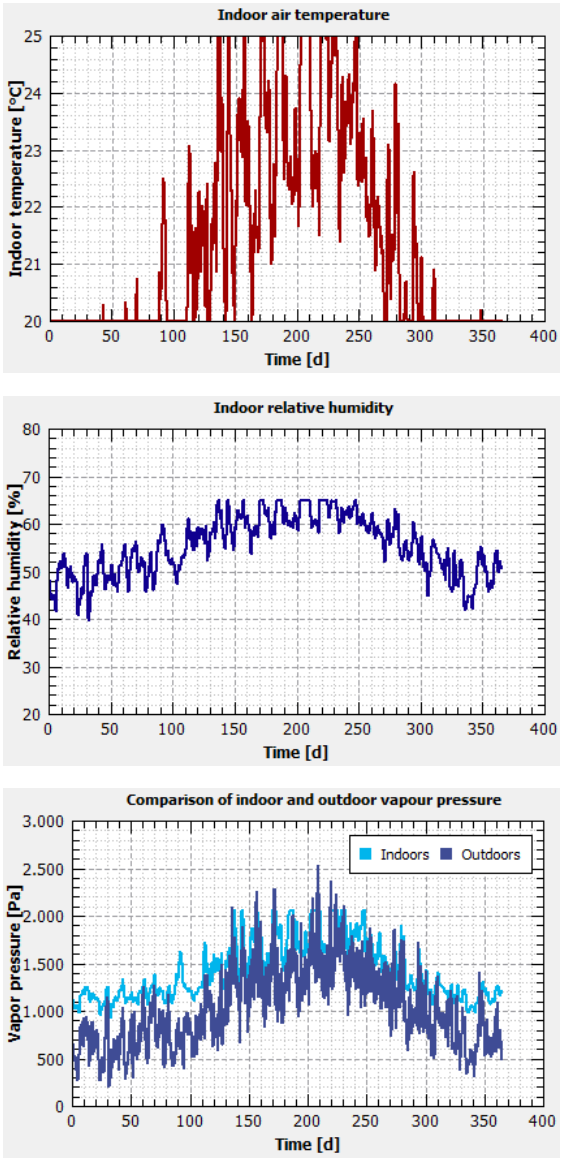


Figure 48: Indoor temperature [°C], relative humidity [%], and comparison of indoor and outdoor vapour pressure [Pa] (Delphin, 2024)

3.2.5 Discretization grid

The discretization grid is used to specify the grid points where the moisture and temperature values are calculated. The number of grid points highly influence the simulation time. For these simulations the following values are set:

Minimum element size	1	[mm]
Maximum element size	50	[mm]
Stretch factor	4000	[-]

This results in the following discretization grid:

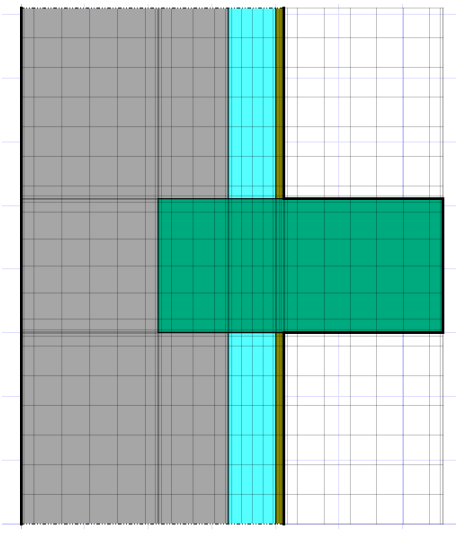


Figure 49: Geometry discretization grid

With grid statistics:

Grid elements (total)	1176	[pcs]
Grid elements (used)	978	[pcs]

3.2.6 Sobol settings

The first Python script creates the Sobol samples. Due to the amount of data storage and computational time needed the n-value of the Sobol sampler is set to 128. To improve the reliability of the result this number can be set higher. This n-number is always a power of 2, ranging from 32 for simple models to 1024 for complex models.

The option: second order indices is set to 'True', the sampler will create the number of samples that can also calculate the second order indices. The number of samples refers to the number of simulations needed. Every sample (row) has x varying values, depending on the number of parameters (columns) that need to be varied.

Sobol sampler statistics:

Number of parameters	5	[pcs]
Number of samples	1536	[pcs]
Number of values	7680	[pcs]

3.3 Material parameters

The materials properties effect moisture storage and transport and thus the amount of moisture in the geometry at a certain time and place. The material properties are the only variables in the sensitivity analysis. Due to the change in material properties, additional changes are made to storage functions and other parameters. These changes are required for Delphin to run the simulation without errors. This is further explained in the next paragraphs.

There are four types of material in the geometry. Two remain unchanged and two vary between simulations. The fact that only two materials are varied is mostly depending on computational time and expected influence on the results. With less parameters, more simulations can be conducted within the same simulation time.

The gypsum and wood are set to be constant materials, whereas the insulation and brick properties are varied. This choice is based on the expectation that brick and insulation properties are likely to have the most effect on the wood decay.

3.3.1 Gypsum board and wood

The interior of the wall is cladded with gypsum board, a frequently used finishing board in construction. The gypsum is placed against the insulation and surrounds the wooden beam. Important to note is the fact that the geometry is detailed without any air gaps between the layers, both for the gypsum board and for the other materials. For gypsum board the material with ID 81 from the Delphin database is used, which has the following material properties:

Gypsum properties:

Density	850	$[kg/m^3]$
Specific heat capacity	850	[J/kgK]
Thermal conductivity	0.2	[W/mK]
Vapour diff. rest. factor	10	[-]
Effective saturation	551.0	$[kg/m^3]$
Hygroscopic SV at rh80	7.2	$[kg/m^3]$
Water uptake coef.	0.28	$[kg/m^3/s^{0,5}]$

With corresponding moisture storage function:

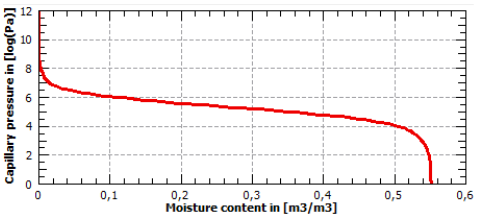


Figure 50: Moisture storage function gypsum board, Delphin database ID 81 (Bauklimatic Dresden, 2024)

As outlined in the literature study, within the wood decay calculation of Isaksson et al. (2012), Meyer-Veltrup et al. (2017) and Brischke en Meyer-Veltrup (2015) one specific wood type is used. The wood in the geometry is set to be (Norway) Spruce. It is a low durability wood type but frequently used in wooden construction all over Europe.

The anisotropic behaviour of wood is integrated into the hygrothermal simulation. The material ID 711 and 712 from the Delphin material database are used, respectively for the longitudinal (U/X) and radial (V/Y) material properties of Spruce. The properties of the wood and moisture storage functions:

Wood properties (longitudinal: U, radial: V):

Density	393.7	[kg/m^3]
Specific heat capacity	1843	[J/kgK]
Thermal conductivity U	0.151	[W/mK]
Thermal conductivity V	0.112	[W/mK]
Vapour diff. rest. fact. U	4.6	[-]
Vapour diff. rest. fact. V	186.1	[-]
Effective saturation	728.1	[kg/m^3]
Hygroscopic SV at rh80	59.8	[kg/m^3]
Water uptake coefficient	0.012	[kg/m^3/s^{0,5}]

With corresponding moisture storage function:

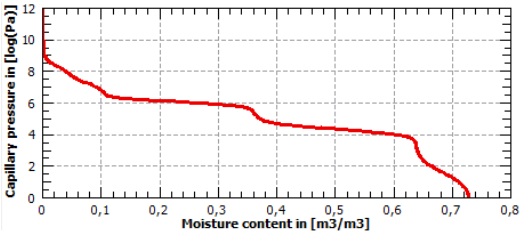


Figure 51: Moisture storage function spruce (radial), Delphin database ID 712 (Bauklimatic Dresden, 2024)

3.3.2 Insulation and brick

Some of the brick and insulation properties are varied. This range of variation is set in the Sobol sampler and is called the bound of the parameter. These bounds are used as minimum and maximum values where the sampler can take values from. The chosen parameters are parameters that are expected to have a significant effect on the moisture and temperature properties of the geometry.

For insulation a standard capillary active material is used, with ID 1780 in the material database from Delphin.

In terms of temperature in the geometry the insulation is highly influential, and temperature is subsequently influential on wood decay risk. The thermal conductivity of the insulation is therefore one of the material properties that is analysed in this study. The bounds are set to be ranging from a R_C value of 1 m^{2}K/W as minimum to a value of 4 m^{2}K/W as maximum. This corresponds to a thermal conductivity of 0.01875 to 0.075 W/mK for an insulation thickness of 75 mm.

Capillary active insulation is able to absorb 'condensed' moisture and temporary store this water during cold periods. In subsequent warmer periods, this water can be transport back to the inside where it can evaporate again.

For this principle to work the insulation should have a high moisture storage capacity and a high moisture transport diffusivity. In the hygroscopic range, this is referred to by the vapour diffusion resistance factor that gives an indication of the resistance the water vapor encounters when being transported through the material by water vapour concentration gradients. When this factor is equal to 1, the material diffuses water with the same resistance as air. An increase in this factor means that there is more resistance. To research the effect of the water vapour diffusion resistance factor for both standard and capillary active insulation. The bounds is set between 1 and 100 and is the second property to be varied.

For brick the Delphin database material with ID 1824 is used, and vapor diffusion resistance factor varied between 5 and 50. Representative for most common brick types.

The moisture storage is represented by the moisture retention curve. As used in Delphin, this curve shows the amount of moisture w [kg/m^3] as a function of the capillary suction pressure. To vary the moisture storage capacity in the sensitivity analysis, a different moisture retention curve for each variant is made.

To do so, different retention curves are generated using the Van Genuchten parametrisation (Van Genuchten, 1980). The parameterisation take the form:

$$\theta(\psi) = \theta_r \frac{\theta_s - \theta_r}{[1 + (a|\psi|)^n]^{1-1/n}}$$

where:

ψ = suction pressure, $[Pa]$;
θ = volumetric water content, $[m^3/m^3]$;
θ_s = saturated water content, $[m^3/m^3]$;
θ_r = residual water content, $[m^3/m^3]$;
α = factor for mean pore size, [-];
n = factor for pore size distribution, [-].

For both insulation and brick the values for θ_s, θ_r and α are not varied. The only value that is varied is n. This factor relates to the pore size distribution. The factor n changes the moment of capillary activity of the material and the factor α the general amount of moisture storage.

Both α and n were fitted (by eye) to the measured data. This resulted in the following values: n = 1.4 and $\alpha = 2 \cdot 10^{-5}$ for brick and n = 2.1 and $\alpha = 2 \cdot 10^{-6}$ for insulation. Bounds for n in the sensitivity analysis are 1.2 and 2 for brick and 1.5 and 2.75 for insulation. For every variant in the simulation, a different moisture retention curve is thus made.

In the appendix, the fitted n-value to the measured moisture retention curve, from the Delphin database for brick and insulation can be found. The fitted curve of brick can additionally be found in Figure 52. Plots to show the range of n-value used for the sensitivity analysis are also placed in the appendix.

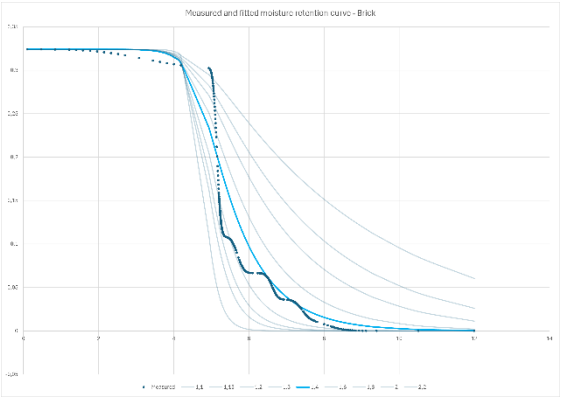


Figure 52: Measured and fitted moisture retention curve of brick

To prevent an error in the simulation, theta-80 and theta-cap are calculated for every specific moisture retention curve.

Using the inverted Van Genuchten function:

$$\psi = \frac{1}{\alpha * n}\left(\left(\frac{\theta_s - \theta_r}{\theta - \theta_r}\right)^{\frac{1}{1-\frac{1}{n}}} - 1\right)$$

where,

Density of water	1000	[kg/m^3]
Gas const. of water vapor	462	[J/kgK]
Absolute temperature	298	[K]

Resulting in the following material properties, with varied properties (*VAR*) and corresponding calculated properties (*CAL*):

<u>Insulation properties:</u>

Density	186.9	[kg/m^3]
Specific heat capacity	1100	[J/kgK]
Thermal conductivity	*VAR*	[W/mK]
Vapour diff. rest. fact.	*VAR*	[-]
Open porosity	929.49	[kg/m^3]
Effective saturation	928.8	[kg/m^3]
Hygroscopic SV at rh80	*CAL*	[kg/m^3]
Water uptake coef.	0.85	[kg/m^3/s^{0,5}]
Factor for mean pore size	2E-6	[-]
Factor for pore size dist.	*VAR*	[-]

Insulation bounds:

Thermal conductivity	[0.01875 – 0.075]
Vapour diff. rest. factor	[1 - 100]
Factor for pore size dist.	[1.5 – 2]

<u>Brick properties:</u>

Density	1698.3	[kg/m^3]
Specific heat capacity	929	[J/kgK]
Thermal conductivity	0.599	[W/mK]
Vapour diff. rest. factor	*VAR*	[-]
Open porosity	359.13	[kg/m^3]
Effective saturation	324.05	[kg/m^3]
Hygroscopic SV at rh80	*CAL*	[kg/m^3]
Water uptake coef.	0.097	[kg/m^3/s^{0,5}]
Factor for mean pore size	2E-5	[-]
Factor for pore size dist.	*VAR*	[-]

Brick bounds:

Vapour diff. rest. factor	[5 - 50]
Factor for pore size dist.	[1.2 – 2]

3.4 Output

This paragraph gives an overview of the output from the simulation software. What is needed to calculate wood decay, output settings and how this data is extracted from Delphin.

Moisture content and temperature are needed as result files to calculate daily dose. For temperature [°C] there is only one option, in the quantity selection 'Temperature' as a state variable or related quantity is chosen. Within the calculation tab, the options for (1.) Individual values of each selected element or side [Single] and (2.) Write values as calculated at output times [None] are selected. Time interval is set to 1 day, with field/profiles as output file frequency. Resulting in a .d6o temperature profile file.

For the moisture content [kg/kg] the quantity selection is set to 'MoistureMassByMass', this quantity gives the total mass of moisture per mass of the REV (Representative Elementary Volume). Similarly to temperature the calculation options set to (1.) Individual values of each selected element or side [Single] and (2.) Write values as calculated at output times [None], giving the same amount of values in both grid points and total days. These makes calculation of day and coordinate specific dose possible. An example of both the temperature and moisture profile files can be found in the appendix.

Besides data files, files with simulation information are needed to check simulation progress and save simulation data such as the coordinate and material information. These are: screenlog.txt (log folder) containing solver information and the .g6a file (result folder) containing geometry information.

4 Results

This chapter is divided into parts: (1.) results from the material sensitivity analysis giving the effect of key parameter impacting moisture content and dose and (2.) the visualisations of hygrothermal safety by using examples.

To find the most influential material property on wood decay, multiple sensitivity analyses are executed. The effect on maximum and average moisture content and maximum and average dose in the wooden beam is studied.

4.1 Sensitivity analysis

This study focusses on four different output variables, all based on the same data from the simulations. This ensures that the Sobol indices of the analysis can be compared, using different parts of data to calculate different result values but using the same dataset.

Multiple analyses are primarily executed to compare the sensitivity of a material property between different result values. Such as moisture and dose. Second, the calculated values that are used by the Sobol analysis can be compared and checked for discrepancies.

The sensitivity analysis is used to calculate the effect of the material properties on potential wood decay. Due to the fact that dose is calculated from the generated data with boundary conditions, it is interesting to add both a sensitivity on moisture and on dose to see if differences are present between the two. The comparison between calculated values is done to ensure correct calculation of the values analysed by Sobol. With four different Sobol results, deviations can be spotted and reliability of correct indices calculation is improved.

Additionally, a comparison is made between average and maximum values in the wooden beam. The averages give insights in the overall hygrothermal performance of the beam while maximum values are used to study extremes in the wooden beam. The maximum value for both moisture content and dose are found at the corners of the beam.

Temperature is, although it is known to have a high influence on wood decay calculations, is not included as output variable. Only as material property in the Sobol parameters (I_TC).

The four output variables studied:

SA 1	Maximum moisture content in the beam
SA 2	Average moisture content in the beam
SA 3	Maximum dose value in the beam
SA 4	Average dose value in the beam

4.1.1 Clarification of results

The results of the sensitivity analysis are plotted in graphs, in which the following abbreviations for the material properties are used. These are:

I_TC	Insulation – Thermal conductivity
I_WVDRF	Insulation – Water vapour diffusion resistance factor
I_NV	Insulation – Moisture retention (n-value)
B_WVDRF	Brick – Water vapour diffusion resistance factor
B_NV	Brick – Moisture retention (n-value)

A selection of the geometry, and corresponding data, is used for the sensitivity analysis. Only data from the coordinates of the wooden beam is used. These coordinates are highlighted and surrounded by a red dotted line in Figure 53.

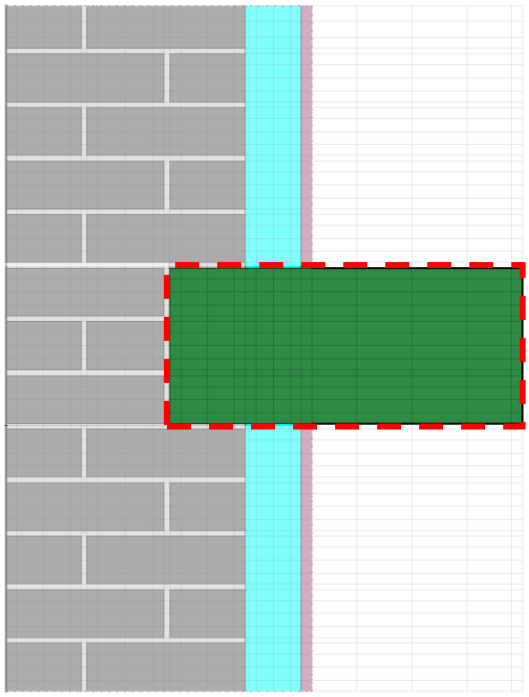


Figure 53: Selection of geometry used for sensitivity analysis

4.1.2 SA 1: Maximum moisture content in beam

The first sensitivity analysis conducted is the analysis on the maximum moisture content in the beam. For every simulation, the average is calculated over all daily maximum moisture content in the beam. These maximum are typically found in the corners of the beam. This results in one average per simulation, of maximum daily values and thus one value for every simulation that can be analysed using Sobol.

From Figure 54 can be concluded that properties I_TC, I_WVDRF and B_WVDRF have no direct influence on the output variance, excluding interaction between the material properties. In contrast, B_NV significantly contributes to the output variance, accounting for 78.1% of the direct variance. Unusual is the slightly negative Sobol index of I_NV, due to the fact the indices should range between 0 and 1. This could indicate an error but the negative value is so small (-1.0%) that it is counted as non-influential.

Interaction from the second order indices (Figure 55) is present between I_NV and B_WVDRF and contributes for approximately 6.1% to the output variance, while the interaction between I_NV and B_NV account for a more significant 20.2%. Other interaction are negligible.

The total Sobol indices, plotted in the Figure 56, include both the direct effects and interactions. Parameter I_TC, I_WVDRF and B_WVDRF show again no influence with total indices of zero. B_NV dictates the total contribution with 91,8%, highlighting the importance of the moisture retention. And I_NV, despite the low direct influence, contributes moderately to the total output variance with 16.9%.

From the findings in this first sensitivity analysis on the maximum moisture content in the beam can be concluded that moisture retention of brick is the biggest influence. With an added moderate influence from the moisture retention of insulation. Indicating that the source of moisture in brick, probably being wind driven rain, causes the high extreme moisture loads in the wooden beam. Additional interaction between the two enhance the influence on model output variance.

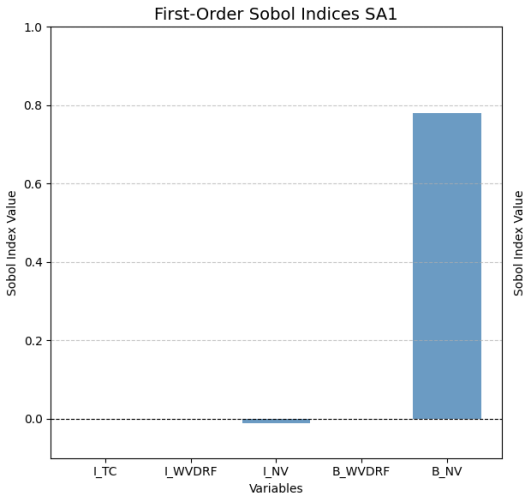


Figure 54: Sensitivity analysis 1: First order indices

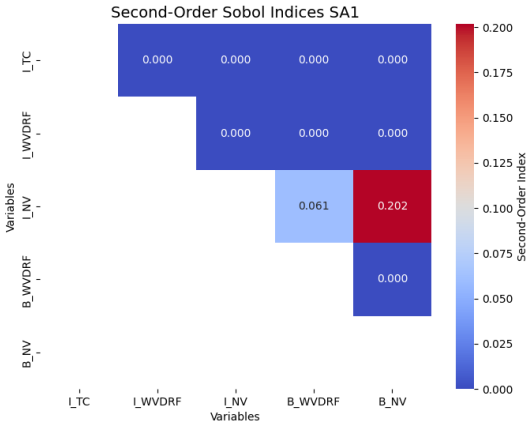


Figure 55: Sensitivity analysis 1: Second order indices

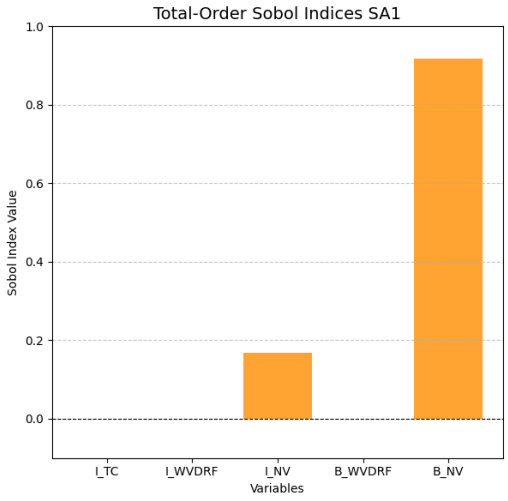


Figure 56: Sensitivity analyses 1: Total order indices

4.1.3 SA 2: Average moisture content in beam

The second sensitivity analysis focusses on the average moisture content in the wooden beam.

Similarly to the first sensitivity analysis, the average moisture content in the beam is, when solely focussing on direct output variance, dominated by material property B_NV. With 97% of the changes in model output as a result of the moisture retention of brick. Other properties are non-influential.

Interactions between material properties are low. The second order indices indicate a 2.8% contribution between I_NV and B_WVDRF and 3.1% interaction between I_NV and B_NV.

Due to low interactions, first and total order indices are comparable. Material property B_NV is the biggest influence of moisture levels in the entire beam with 97% of variance. A small effect of 2% is linked to I_NV. I_TC, I_WVDRF and B_WVDRF can be excluded from the study without compromising results on average moisture content in the beam.

Notable is the difference between SA 1 (max moisture content) and SA 2 (average moisture content), where the I_NV is more influential in the maximum moisture levels in the beam in comparison with the average levels.

This result indicates that the moisture retention of brick has more influence on the long term moisture content in the beam. And consequently moisture retention of insulation influencing the moisture in de beam with moisture load extremes. Suggesting the ability of insulation to reduce these extremes.

With knowledge from literature, the non-influential indices values of I_TC, I_WVDRF and B_WVDRF are unexpected. The question arises if the influence of the moisture retention is so extreme that influence from other parameters seem zero. Additional research should be conducted to determine if the values are zero or that sensitivity of the Sobol analysis is affected. This could be executed by changing only the individual parameters, and analysing the result values if the other parameters remain constant.

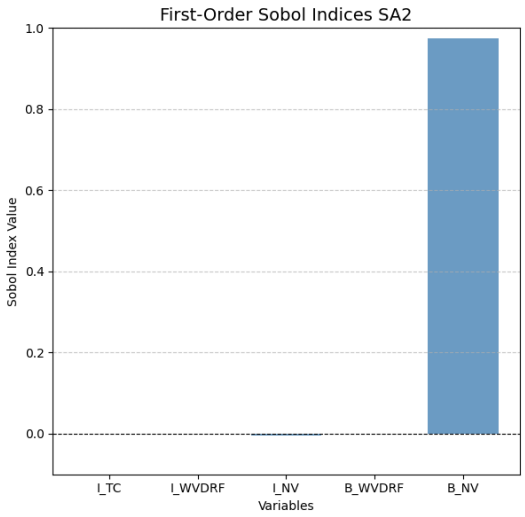


Figure 57: Sensitivity analysis 2: First order indices

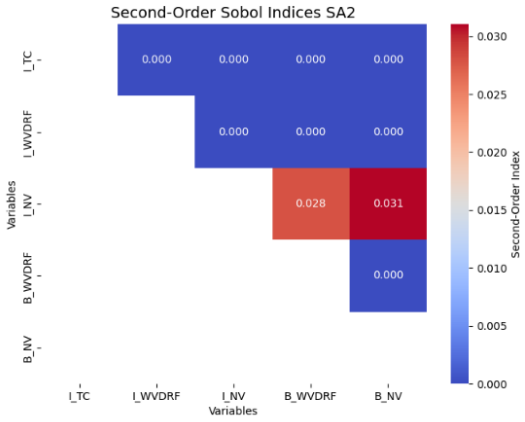


Figure 58: Sensitivity analysis 2: Second order indices

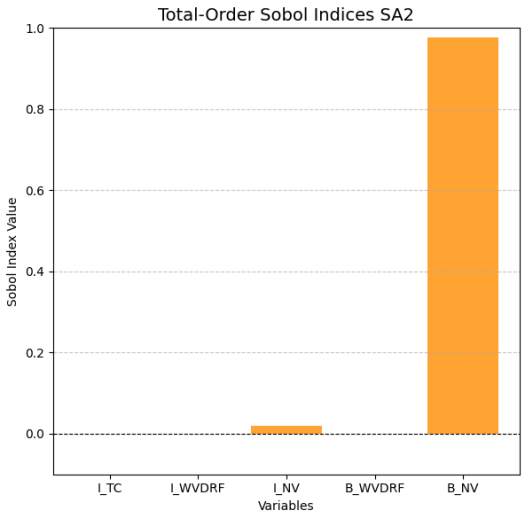


Figure 59: Sensitivity analysis 2: Total order indices

4.1.4 SA 3: Maximum dose in beam

The third sensitivity analysis conducted is the analysis on the maximum dose in the beam. For all the simulations, the daily maximum dose from all grid volumes is found to calculate an average daily maximum dose. Again, this maximum value is found in the corners of the beam. Directly giving the sensitivity of the material parameters on the highest wood decay parts of the beam.

The direct indices give, similarly to sensitivity analysis 1 and 2, that material properties I_TC, I_WVDRF and B_WVDRF have no direct influence on the output variance. B_NV contributes with 35.1% and I_NV with 43.1%, both significantly with the moisture retention of insulation being slightly higher to the direct output variance.

Second order, and thus indirect influence, is only substantial for the interaction between B_NV and I_NV with 19.8%. Other interactions are negligible.

For the maximum dose in the beam. Both the moisture retention of brick and insulation are significant. With B_NV highest at 54.5%, giving differences in output variance by both direct and indirect effect and I_NV having a similar role in the total output variance with 52.4%.

Interesting is the comparison between SA1 (maximum moisture content) and SA3 (maximum dose value). With all other input settings set similar, and only the calculation of the dose value being different, the difference in influence can be linked to the dose calculation boundary conditions or temperature influence of insulation.

If the temperature would influence dose, the I_TC should also result non-zero values (have influence on model output). Resulting to the fact that the difference can be accounted to the dose calculation boundary conditions.

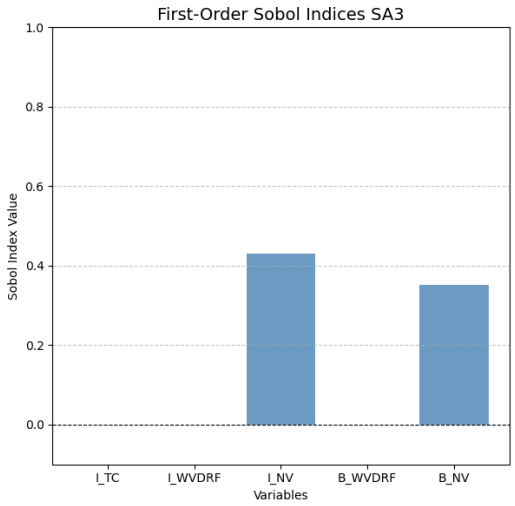


Figure 60: Sensitivity analysis 3: First order indices

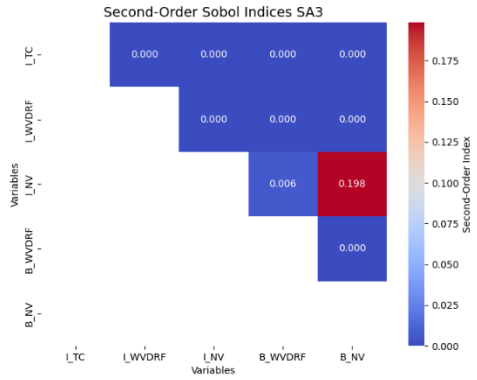


Figure 61: Sensitivity analysis 3: Second order indices

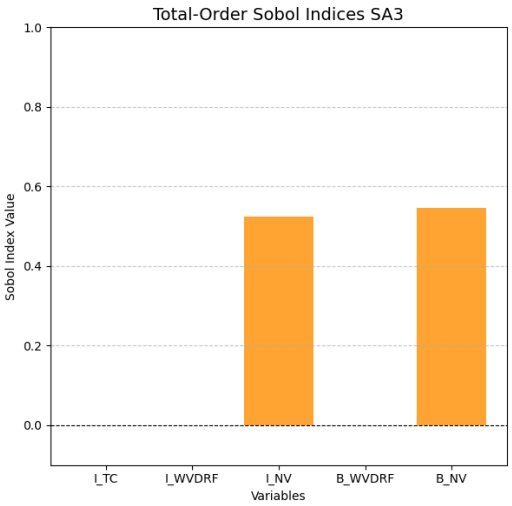


Figure 62: Sensitivity analysis 3: Total order indices

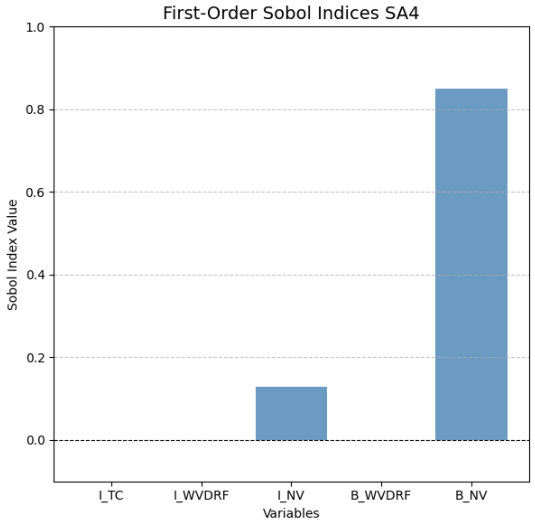
4.1.5 SA 4: Average dose in beam

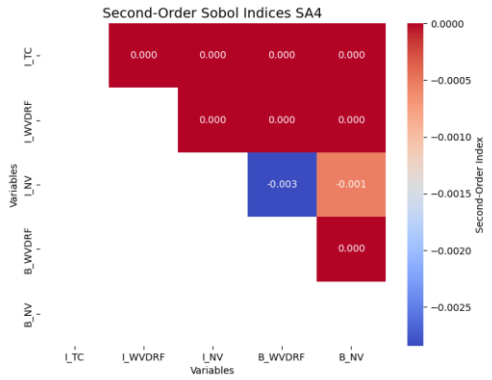
The last sensitivity analysis, calculates the effect of the material properties to the average dose values of the beam. Notably different from SA2, B_NV is the dominant property looking at direct output variance with 85.0%. Also influential is I_NV with 13.0% as direct effect. I_TC, I_WVDRF and B_WVDRF do not contribute to first order effects.

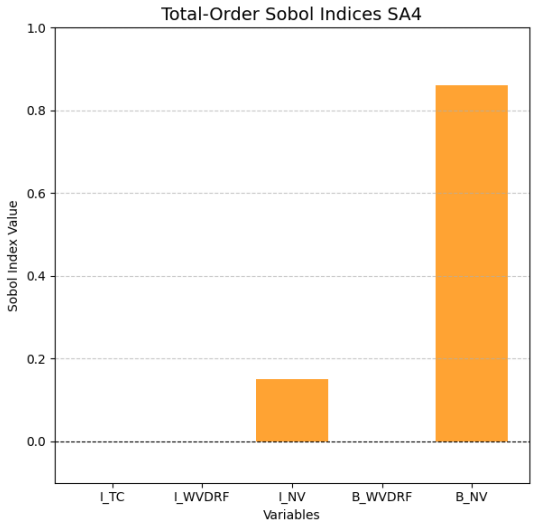
The second-order Sobol indices, which account for interactions between parameter pairs, show negligible interaction effects. Most of the second-order indices are close to zero. Negative values are unusual and might indicate numerical instability or errors in the estimation process. However, their overall contribution to the output variance remains minimal and are therefore assumed to be non-influential

The total-order Sobol indices, which combines both direct contributions and interactions, confirm the dominant influence of B_NV at 86%. I_NV plays a secondary role with 15.2%, highlighting additional effect from the insulation material properties.

Similar to the comparison between maximum and average moisture content, the comparison between maximum and average dose values shows higher I_NV influence in the maximum sensitivity analysis. Implying dose extremes to be enhanced or reduced by the insulation.







4.2.1 Conclusion

The results of these sensitivity analyses are summarised in this paragraph. Including direct conclusions, interpretations and discussion as a result of the material property study.

Conclusion of individual parameters

From all sensitivity analyses, being maximum and average moisture content and maximum and average dose value, can be concluded that I_TC (thermal conductivity insulation), I_WVDRF (water vapour diffusion resistance factor insulation) and B_WVDRF (water vapour diffusion resistance factor brick) have no influence on the model output variance. All parameters result in zero values for the first and total order indices.

Only for the second order indices, B_WVDRF shows minor interaction with parameter I_NV (moisture retention of insulation) in both the maximum and average moisture content. With an 6.1% and 2.8% influence respectively.

However, the non-influential indices values of I_TC, I_WVDRF and B_WVDRF are unexpected. The question arises if the influence of the moisture retention is so extreme that influence from other parameters seem/are calculated as zero. Additional research should be conducted to determine if the values are zero or that sensitivity of the Sobol analysis is affected by extremes of other parameters. This could be executed by changing only the individual parameters, and analyse the result values with other parameters kept constant.

Both I_NV and B_NV (moisture retention of brick) have influence on model output variance in all the executed sensitivity analyses. With total influences of 16.9, 2.0, 52.4 and 15.2% for I_NV and 91.8, 97.0, 54.5 and 86.0% for B_NV in SA1, SA2, SA3 and SA4 respectively.

From these results can be concluded that B_NV consistently has a higher influence to the output variance and I_NV additionally influences the variance but in lesser extents. Additionally, all analyses show interaction between the two materials.

Conclusion of comparison between SA's

Multiple sensitivity analyses are executed to research difference between the analysed result values and check for discrepancies in result value calculation.

All indices give similar results, indication no calculation errors of result values and increase confidence in the calculated Sobol indices. If differences are visible, such as the difference between the maximum moisture content and maximum dose value, they can be linked to viable reasons.

This difference between the two analyses on maximum values is likely caused by the dose calculation boundary conditions. Where dose values are more influenced by insulation than compared to moisture content. Wood decay assessment in the next paragraphs can clarify if this influence is positive of negative. In order words: if insulation properties are able to reduce the risk on wood decay.

Furthermore, there is a notable difference between the average and maximum value. Where maximum analyses show higher I_NV influence in comparison with B_NV. Implying that moisture and dose extremes are probably reduced by the insulation.

4.3 Wood decay assessment

After running the simulations the dose is calculated based on moisture and temperature data. This dose can be seen as a wood decay potential and therefor gives an indication of the hygrothermal safety of the construction.

The first sub-paragraph will elaborate on the wood decay assessments over time. Being the first possible assessment of the hygrothermal safety of the embedded wooden beams without changing the simulation or simulating multiple situations. Subsequently, the changes to the simulation settings, what could be interior or exterior boundary condition are visualised. Making, among others, the assessment of brick hydrophobization possible. Finally, geometry and material changes are illustrated. Being able to compare and assess geometry changes and material choices. For example, the performance difference between standard or capillary active insulation.

The wood decay starts at the beam ends. Surrounding the entire beam end and progressing into the beam itself. In all simulations Spruce is used as wood type. With the decay rating formulas the dose values are calculated for the five step decay rating of the EN 252 (brown rot, non-shaded):

$$DR\big(D(n)\big) = 4 * \exp\left(-\exp\Big(1.564 - \big(0.0054 * D(n)\big)\Big)\right)$$

With the inverse to calculate the number of days for the specific service life of spruce:

$$D(n) = \frac{1.564 - \ln\left(-\ln\left(\frac{DR}{4}\right)\right)}{0.0054}$$

gives:

DR 1 – slight attack	229.1 days
DR 2 – moderate attack	357.5 days
DR 3 – severe attack	520.4 days
DR 4 – failure	1398.9 days

4.3.1 Development over time

Whereas moisture and temperature are dynamic over time, dose and wood decay are cumulative and can only increase over time. Three different visualisation show the progress of wood decay in the wooden beam over time.

In Figure 63 the maximum dose is 118.1 days. Indicating that until the end of the simulation the dose values in the beam haven't succeeded decay rating 1 (slight attack), and thus no wood decay is present at this point. However, dose values are calculated, meaning that wood decay growth conditions are satisfied and will result in failure at a certain point in time due to the similar yearly weather file used in the simulation.

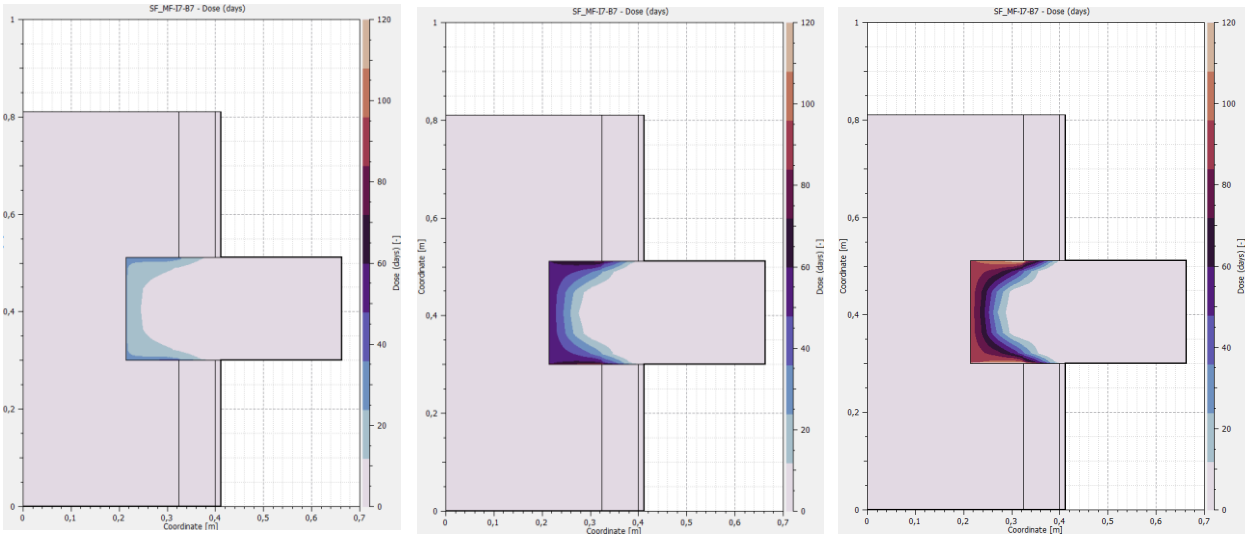


Figure 63: Dose visualisation over time. Simulation SF_B7-I7, left: day 365, middle: day 730, rights: day 1095.

4.3.2 Changes in boundary conditions (wind driven rain)

From literature is known, that wind driven rain has a substantial amount of influence on the amount of moisture in the geometry and therefore on the risk of wood decay in the beam. Rain penetration in bricks can be decreased by adding a hydrophobization to the outside surface of the façade. These changes can be simulated, by changing the simulation settings of Delphin, and thus can be used to assess the effect of these hydrophobization products.

In this case, the simulation is run with the similar geometry but over a simulation time of ten years, visualised at the end of year 4, 7 and 10. From the sensitivity analysis is concluded that the brick moisture retention properties have a significant role in the amount of moisture in the beam, causing exterior moisture loads to reach the beam. This is also clearly visible in the dose visualisation where the result of wind driven rain on dose is illustrated.

With wind driven rain turned off, the dose values for all grid volumes is zero after 10 years. Highly indicating no wood decay to the beam for the rest of its service life due to the yearly weather files. Figure 65, where wind driven rain is turned on, shows high dose values up to 1800. This indicates decay rating 4, where service life is known as failure. The wooden end of the beam will decay within 10 years.

In future simulations, when real life situations are assessed, more research and time should be invested into the boundary conditions. These simulations use yearly weather files, but a more, future proof weather files need to be included. For example weather files with more annual rain loads or higher average and maximum temperatures. These will negatively influence wood decay of the beam and result positively on simulation accuracy.

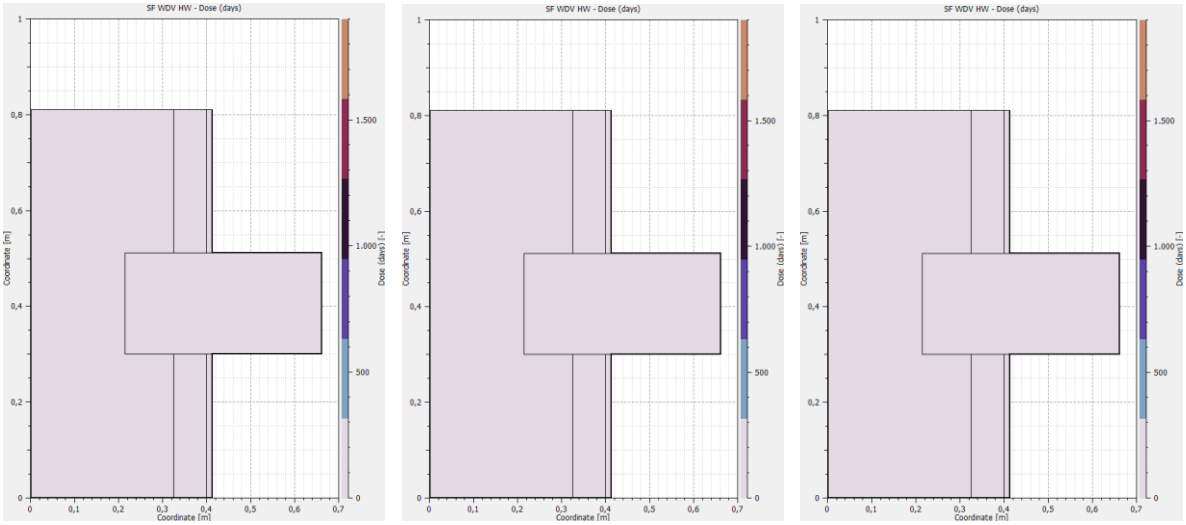


Figure 64: Dose visualisation. Without wind driven rain.
Left: day 1460, middle: day 2555, right: day 3650.

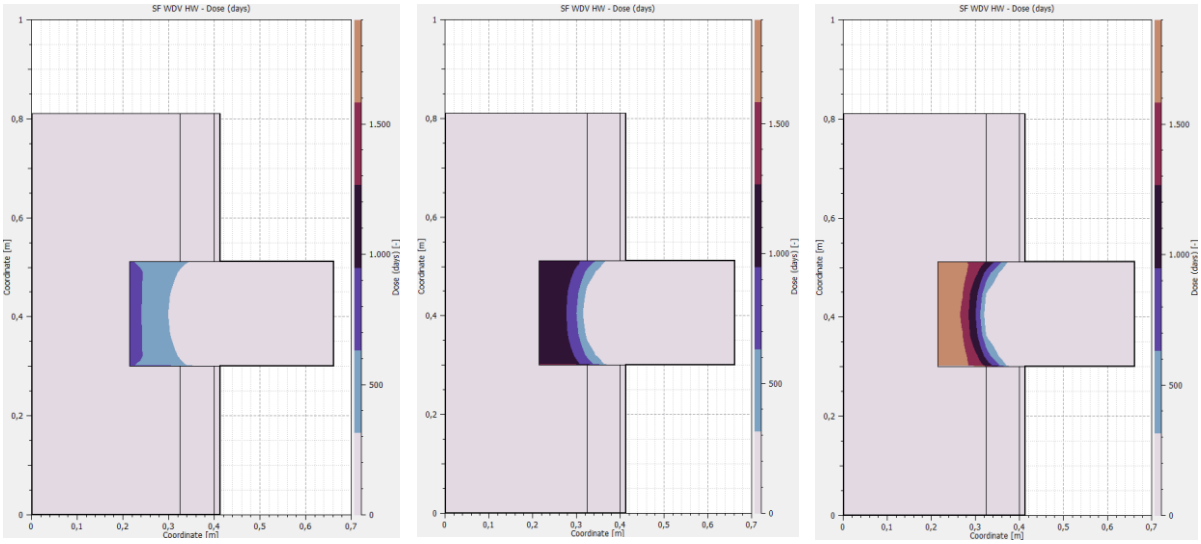


Figure 65: Dose visualisation. With wind driven rain.
Left: day 1460, middle: day 2555, right: day 3650.

4.3.3 Changes in boundary conditions (orientation)

Another example that shows the different effects to hygrothermal performance of the wooden beam is the orientation or location of the simulated wall. Delphin can be used to assess hygrothermal performance of walls in multiple locations. Subsequently, orientation of the wall can be defined to the exact degree (north is 0) and inclination of the wall (vertical is 90).

In all simulations, Essen is used for its exterior boundary conditions, generating conservative results due to the high rain loads compared to other geographical locations. To show the effect to wood decay, Figure 67 and Figure 66 illustrates the difference between a wall orientated east (90 deg rotation, vertical) and a wall orientated west. (270 deg rotation, vertical). Indicating full failure of the wooden beam up to the insulation layer in the west oriented wall and no decay in the east oriented wall at all.

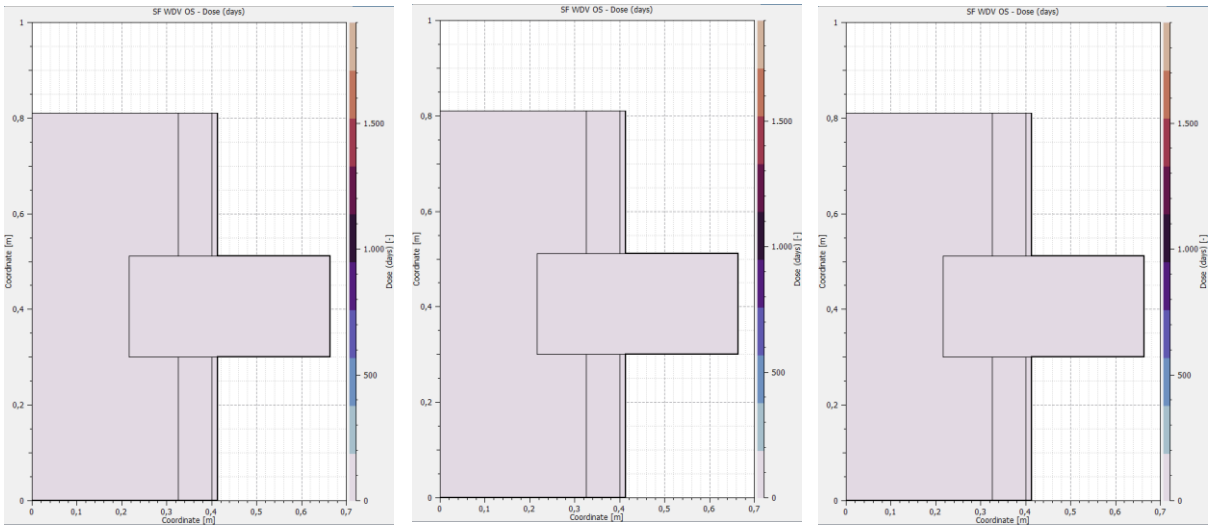


Figure 67: Dose visualisation. Orientation: East
Left: day 1460, middle: day 2555, right: day 3650.

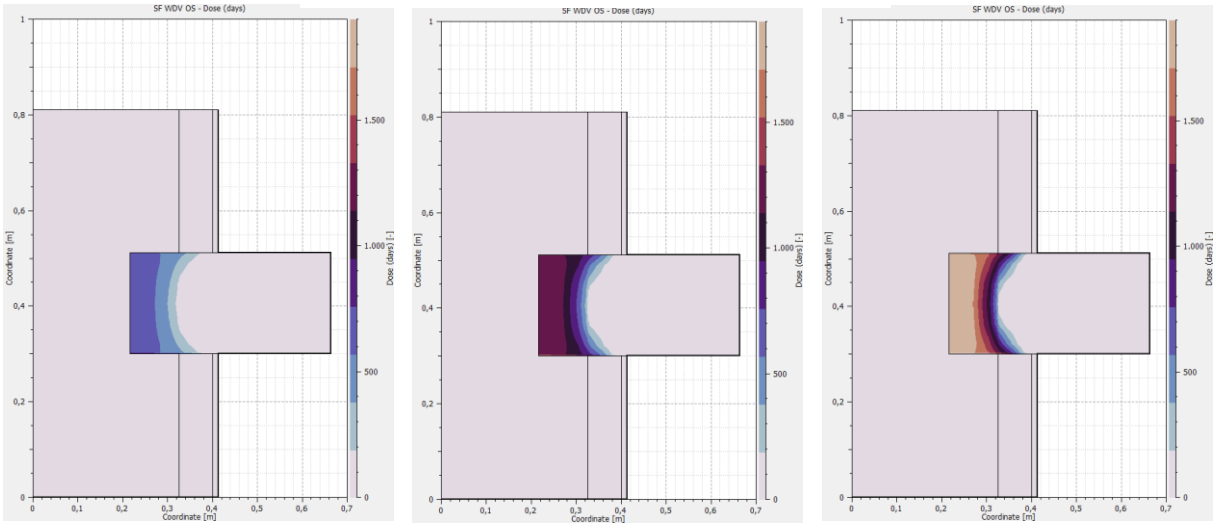


Figure 66: Dose visualisation. Orientation: West
Left: day 1460, middle: day 2555, right: day 3650.

4.3.4 Changes in geometry (pre and post insulation)

The previous comparisons visualise changes to simulation settings, and therefor compare changes in exterior conditions. Additionally, changes to the simulated geometry itself can be visualised with this tool. Contributing to the assessment of hygrothermal safety between optional design choices when renovating buildings with solid brick masonry walls and embedded wooden beams. The examples visualised are: pre and post insulation, the exclusion of insulation surrounding the beam and the difference between standard and capillary active insulation.

This first example shows the impact of the application of capillary active interior insulation in comparison with the original situation.

In Figure 68 and Figure 69 the top visualisation are the moisture content, temperature and dose values pre insulation after 10 years of simulation. Bottom visuals are post insulation situation.

This comparison states the importance of temperature on wood decay. Temperature increase in the wall due to interior conditions decreases moisture content significantly. Mainly due to the fact there is one less moisture source, being interstitial condensation between the insulation and masonry wall and the transport and evaporation of moisture to the interior of the building.

However, dose values are calculated in both situations, indicating wood decay within years. With approximately 1100 and 1800 days for pre and post insulation respectively, the wood of the latter situation will deteriorate faster but beams in non-insulated walls will also decay at some point in time within these set boundary conditions.

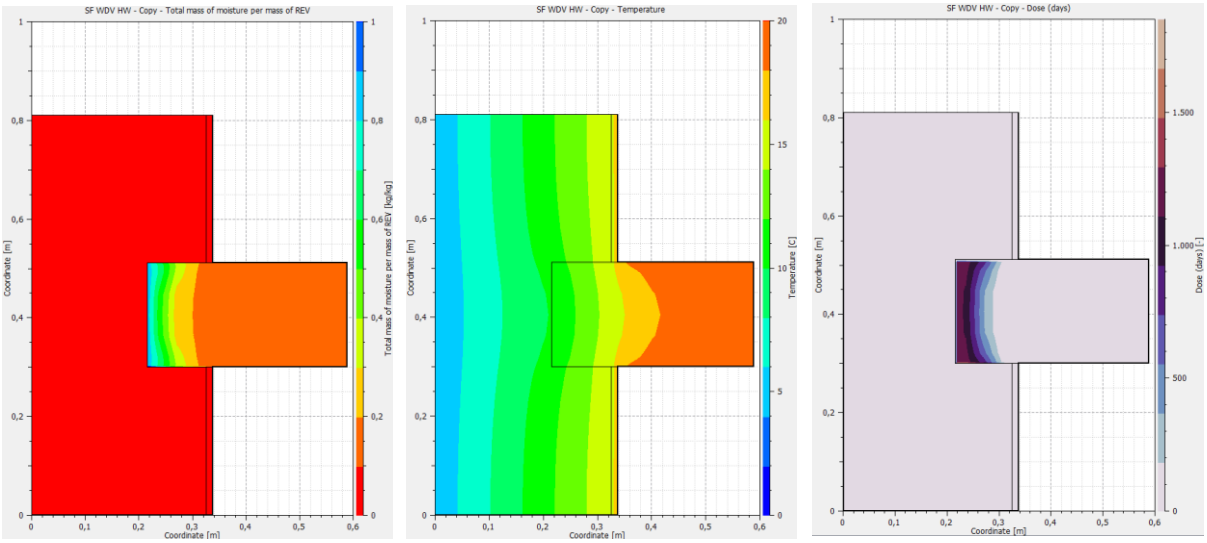


Figure 68: Before capillary active insulation. Day 3650
Left: moisture content. Middle: temperature. Right: dose

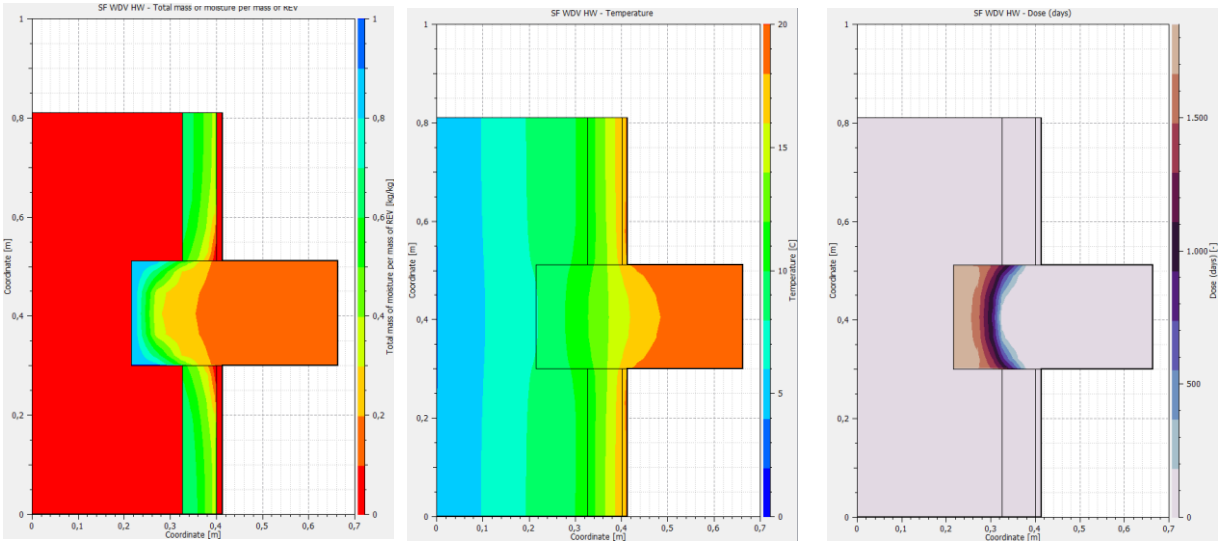


Figure 69: After capillary active insulation. Day 3650
Left: moisture content. Middle: temperature. Right: dose

4.3.5 Changes in geometry (insulation exclusion surrounding the beam)

Not fully encapsulating the embedded wooden beam with interior insulation is a well-known measure to prevent wood decay in the current monumental renovation standards. Theoretically, temperature in the beam and surrounding masonry is higher caused by the interior heating. This prevents moisture condensation around the wooden beam and increases moisture evaporation.

The confirmation of this theory can be found when using Delphin to simulate the differences in the geometry. In Figure 70 is visible that temperature surrounding the beam is significantly higher than in comparison with the temperature in Figure 71.

It causes moisture content to be lower, resulting subsequently in lower dose values in the beam. Important to note is that full failure dose values are visible in the beam but the infected area is lower. Again confirming that changes to geometry only to a small extent prevent wood decay, but external factors like weather and orientation have a substantially higher influence.

No only the prevention of interstitial condensation but also the prevention of contact between materials could be research with these visualisation. With the prevention of contact, for example by leaving a 5 or 10 mm gap between the wooden beam and the brick, capillary suction is prevented. Minimizing the effect from the saturated bricks to the wooden beam. In its place reducing wood decay.

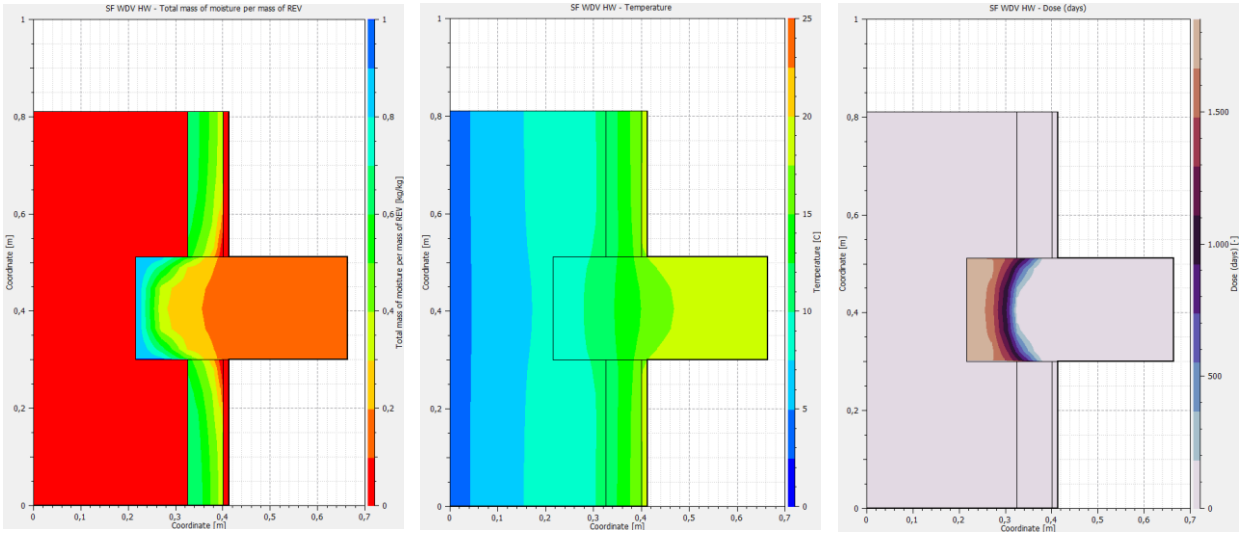


Figure 71: No insulation gap around beam. Day 3650
Left: moisture content. Middle: temperature. Right: dose

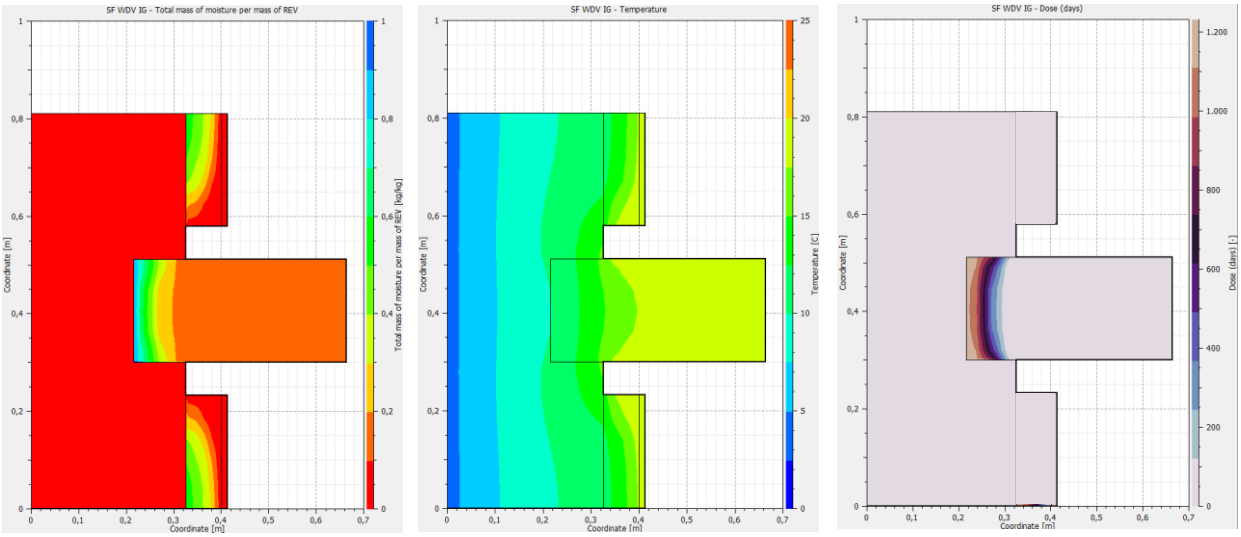


Figure 70: Insulation gap around beam. Day 3650.
Left: moisture content. Middle: temperature. Right: dose

4.3.6 Changes in material types

In this visualisation the effects on hygrothermal performance due to changes in material types is shown. A comparison is made between capillary active insulation and a standard vapour-tight insulation (XPS).

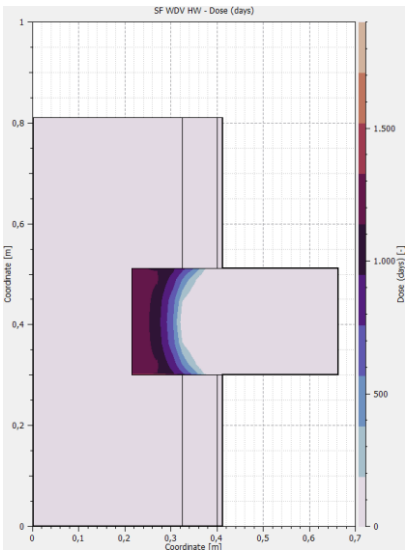
In Figure 72 the results for the capillary active insulation is shown. Dose values of >1700 days indicate failure of the end of the wooden beam due to decay. Similar to capillary active insulation, with standard insulation the wood is expected to fail after 10 years, but the affected surface area is noticeably larger.

Additional to the dose visualisation the moisture content in the geometry is shown in Figure 74 and Figure 75 for capillary active insulation and standard insulation respectively.

In these simulations the thermal conductivity values are both set to equal values. To be able to assess solely the moisture characteristics and effect on wood decay.

The capillary effect of the insulation is clearly visible. It distributes the moisture throughout the capillary active insulation, partly back to the interior side. Contrary to the standard insulation, where extremely high moisture loads are present between the insulation layer and solid brick masonry wall, resulting in higher moisture levels in the wooden beam.

From these decay and moisture visualisation can be concluded that capillary active insulation positively contributes to the prevention of wood decay, but is not capable of reducing moisture content to levels that prevent decay completely.



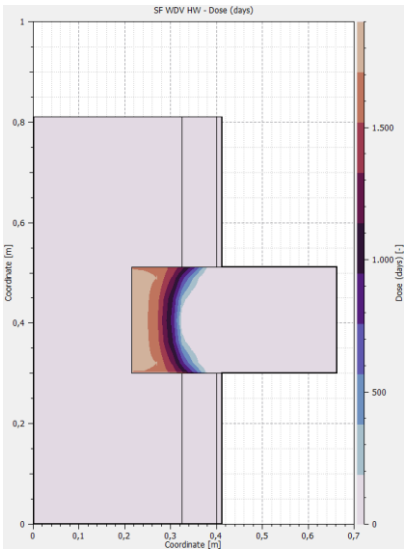


Figure 72: Dose visualisation. Capillary active insulation. Left: day 2555. Right: day 3650.

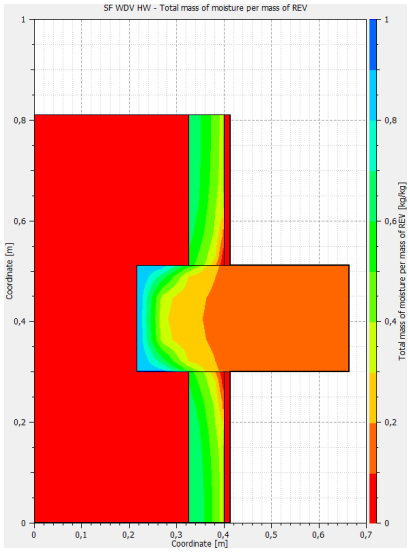
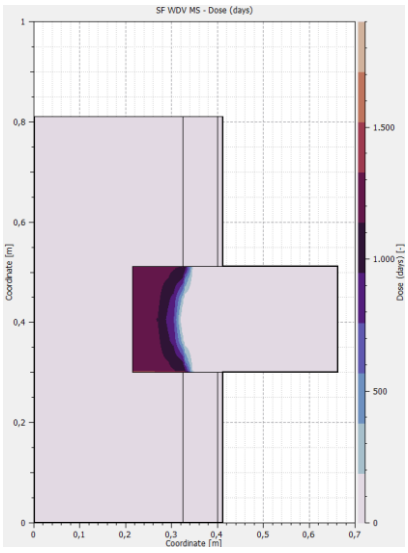


Figure 74: Moisture content. Capillary active insulation. Day 3650



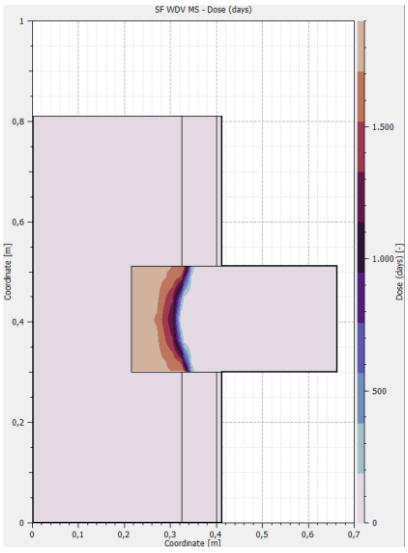


Figure 73: Dose visualisation. Standard XPS insulation. Left: day 2555. Right: day 3650.

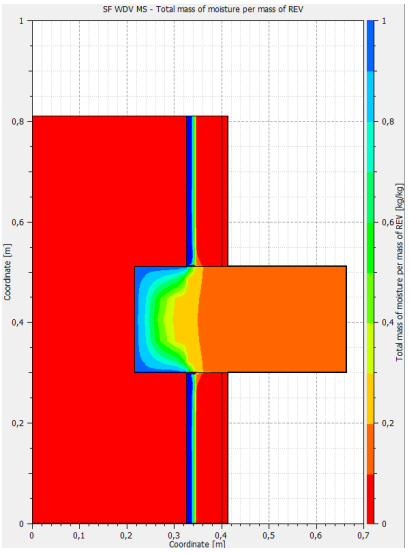


Figure 75: Moisture content. Standard insulation. Day 3650

4.3.7 Conclusion

The goal of these visualisations is to improve understanding of generated results from the simulations and assessment of hygrothermal performance of the wooden beams that are embedded in the solid brick masonry wall.

With these visualisation, created with Python scripts, the understanding of simulation results is improved. These visualisations can show wood decay potential over time in 2D geometry grid visualisation by using the same post processor software as the original result files from Delphin. Dose values are shown with specific, consistent colour maps to indicate service life prediction according to current building regulations. Significantly improving understanding of wood decay risk to the beam and positively impacting the assessment of the hygrothermal performance of the geometry.

Besides the actual visualisation of the dose to improve wood decay assessment. The impact on hygrothermal performance when changing the simulated geometry is researched. For example effects to hygrothermal performance when changing the location or boundary conditions of the simulated wall. Contributing to the overall goal: to find what influences the hygrothermal safety of solid brick masonry with embedded wooden beams and interior capillary active insulation most.

Supported by literature research, from these visualisation can be concluded that external influences on the geometry affect wood decay most. Influences like wind driven rain, annual rain load, solar radiation, location and wall orientation have significant influence on the moisture content in the wall and can completely result in full decay of the wooden beam within years.

When focussing on the moisture reducing potential of capillary active insulation, visualisation show that there is a positive contribution to the prevention of wood decay. Moisture content visualisation show that moisture is divided within the insulation layer, reducing the moisture content in the beam. However, the insulation is not able to reduce the moisture content in the beam to levels that prevent wood decay completely.

5 Discussion

This chapter interprets the findings presented in the previous chapters, aligning them with the research objectives and existing literature. Furthermore, the discussion addresses unexpected outcomes, acknowledges the limitations of this study, and suggests next steps for future research.

5.1 Simulation software

From literature, Delphin is known to be an complex but comprehensive hygrothermal simulation software. Being, among others, the reason for choosing this software for this study. However, this complexity can both negatively and positively influence results from the simulations. By using existing material from the database and by not altering values from where effect on results is not completely known, these possible negative influences of the complex software are kept to a minimum. Similar studies in other hygrothermal simulation software, for example: Wufi or Comsol, could improve reliability of results.

5.2 Boundary conditions

For this research, Essen is chosen as simulation locations. With a quite extreme rain load in comparison with other location and low solar radiation the model is conservative in relation with other simulation locations. Additionally the west orientated wall and the 5% increased interior moisture load cause high moisture contents in the wall. Other situations are likely to have lower moisture contents. Reducing dose value levels of the embedded wooden beam. Next steps could include the study of location and orientation influence on the wood decay potential of the beam.

5.3 Material selection

Four materials: brick, insulation, wood and gypsum with Delphin ID 1824, 1780, 711-712 (multiple for an-isotropic behaviour of wood) and 81 respectively, are used as material for the simulations. These materials are chosen from the Delphin material database and selected as most representative for most situations.

However, this selection with corresponding material properties could have an impact on the results of the simulations. Where other, similar type materials with slightly different material properties could result in other moisture and temperature data. These differences are expected to have minor influence on simulation results but further studies could focus on these differences.

5.4 Sobol settings

Sobol is used to perform the sensitivity analysis on the output variables. This analysis uses a number of simulations to calculate the influence of the chosen variable parameters on the model output. In order to limit computational time, a representative number of values is generated by the sample generator. This representative value set is used to prevents the need for simulating all the possible variations. This sample generator uses a factor, the n-value, to determine the number of representative values. Higher n-values generate more values and therefore increase accuracy of the Sobol analysis. But consequently, increase computational time significantly.

This thesis uses a n-value of 128. For simple geometries a number of 32 is sufficient but more complex models use 512 or 1024 as n-value. Preferably a higher n-values was used for this study, but due to limited available time, 128 as n-value is used. Still generating 1536 simulations, each running approximately six to eight minutes resulting in a full week of constant simulations.

5.5 Sobol bounds

This range of variation is set in the Sobol sampler and is called the bound of the parameter. These bounds are used as minimum and maximum values where the sampler can take values from. Bounds are chosen to represent minimum and maximum values of material types. This selection can influence results, for example when certain bounds are not chosen correctly and possible property values of material are not within that specific range. This risk is minimized by checking all values of similar materials in the Delphin database and by consulting and discussing the bounds with mentors.

Furthermore, the bounds of the n-value (fitted by eye) have a significant influence on the simulation results. The first chosen bounds and week of simulation runs resulted in 130 out of 1500 simulation errors. Caused by the n-values bounds of both insulation and brick. Adjusted bounds reduced the simulation errors to zero but curve fitting by eye seems therefore not the best method and more research should be conducted into its effects.

5.6 Moisture storage functions

Delphin used extensive material data for moisture storage and transport. For the sensitivity analysis this study needed template files where to Sobol generated value set could be placed into. Both for insulation and brick, certain materials where chosen and changed into the template files.

While running the simulations, the model indicated an error in the moisture storage functions due to the Sobol changes in moisture storage properties. Therefor a theoretical function for moisture storage is integrated into the calculation of moisture storage properties for every simulation and placed in the template file, solving the error. However, precise effects of the theoretical function in comparison with the actual material data is unknown. Assumption are made to approximate the theoretical function as close to the actual material data as possible but more research is needed to determine actual influence.

5.7 Sensitivity analysis results

From the sensitivity analysis can be concluded that the moisture retention of brick has a dominant influence on the hygrothermal performance of the embedded wooden beam. Secondly, the moisture retention of insulation is influential on performance but in far lesser extent. No influence is calculated for the thermal conductivity of insulation, the water vapour diffusion resistance of insulation and the water vapour diffusion resistance of brick.

Despite the suggestion of accurate results, with for example the sum of total indices being approximately 100%, indicating that full influence of output variance is dedicated to the varied parameters, the non-influential results seem inaccurate. From literature is known that temperature has a high influence on moisture content in the wall, supported by the wood decay assessments in paragraph 4.3.

The question arises if the analysis is to sensitivity to the high moisture and dose values, probably caused by external boundary conditions and that non-influential parameters are calculated as zero but are calculated influential if these exterior boundary conditions were turned off. Additional research is needed to study the influence of the boundary conditions. For example, by running the sensitivity analysis again but in locations with less rain load or with wind driven rain turned off in the simulation. Solely studying material property influence.

5.8 Influence of parameter indices

The sensitivity analysis calculates first-, second- and total order indices as result value to indicate parameter influence to the model output variance. These influences are calculated in percentage, dividing the total variance between parameters.

This gathered knowledge contributes to finding the key parameters effecting hygrothermal performance. However, positive or negative influence of parameters is not included in this analysis, for example: a certain parameter can affect moisture decrease in a geometry and another parameter could cause an increase in similar amount of moisture, resulting in a similar influence percentage. Dose visualisations can support risk assessment by comparing situations and provide this needed extra information.

6 Conclusion

This thesis focusses on what influences hygrothermal safety of embedded wooden beams in solid brick masonry with interior capillary active insulation most and mitigate the risk of wood decay. Sub-research questions are formulated to contribute to answer the main research question. These sub-research questions are answered below:

1. How does interior capillary active insulation impact the hygrothermal performance of embedded wooden beams in solid brick masonry?

Applying insulation to the interior side of the wall changes the hygrothermal performance significantly, both when applying standard and capillary active insulation. In Figure 77 the application of standard, non-capillary active insulation is illustrated. When a vapour open insulation layer with a high vapour permeability is applied on the interior side of a solid masonry wall, there is a risk on interstitial condensation. Warm, moist interior air diffuses through the insulation layer. When reaching the solid brick masonry, that is colder due to the interior insulation, the air cools down causing the moisture in the air to condense. The 'condensed' water will be absorbed by the masonry and excess condensation will run off.

With the application of capillary active insulation, as can be found in Figure 78, the condensed water between the insulation and solid masonry is absorbed by the insulation due to the high moisture storage properties of the material. The small pore structure causes the moisture to be transported through the material back to the interior side. When the temperature is higher, relative moisture vapour concentration is low and the stored moisture can evaporate. Reducing moisture in the wall assembly. Interstitial condensation does not reach the moisture sensitive building components and the risks on damage is decreased.

Besides interstitial condensation due to the interior insulation, hygrothermal performance is additionally influenced by other factors. Examples are wind driven rain, interior and exterior boundary conditions, prevailing wind directions, solar radiation, location and wall orientation. All potentially able to change moisture content in the wall, and thus reduce or increase the risk of moisture induced damages. This thesis focusses on the decay of wooden beams. Especially the beam ends, embedded in the wall, are susceptible to wood decay.

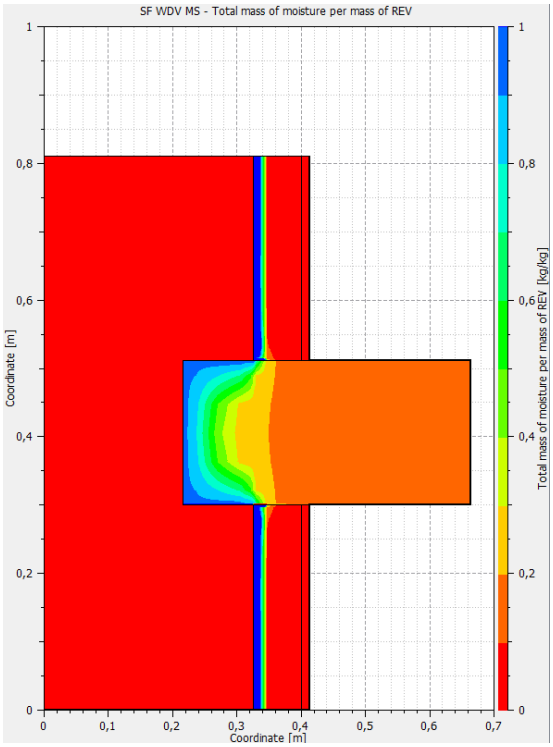


Figure 77: Moisture content visualisation of standard, non-capillary active interior insulation

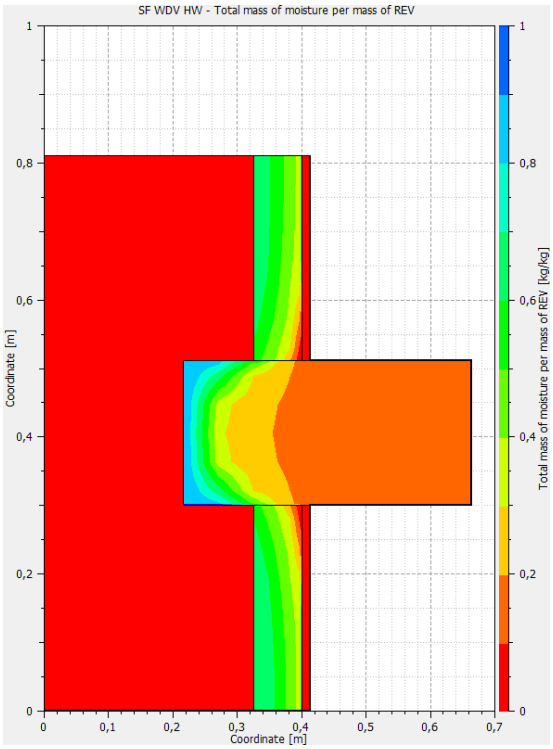


Figure 78: Moisture content visualisation of capillary active, interior insulation in Delphin

2. What methods are available to calculate the performance of complex hygrothermal building components and what simulations can be used to find the most comprehensive, quick and accurate results?

Hygrothermal, or Heat, Air, Moisture (HAM) simulation software plays a big role in assessing the response of wall assemblies to climate loads, providing insights into potential deterioration risks such as mould growth, wood rot, and other degradation.

These simulation tools enable detailed numerical analyses to calculate the moisture response of materials and components under varying boundary conditions. By predicting moisture and temperature conditions within building envelope assemblies over time, hygrothermal simulation enhances understanding of how the wall interacts with the interior and exterior environment, aiding in the early identification of potential moisture-related issues.

With Wufi and Delphin standing out as commonly used software by building practitioners, designers, academia, and researchers worldwide. Wufi is relatively simpler, requiring a limited number input settings and material properties. Delphin is known to be more comprehensive and complex in use. With the additional availability of Python control scripts the choice is made to use Delphin for this thesis.

Interior conditions, like temperature and humidity, are important, especially in cold climate design analyses. Similarly, exterior conditions, encompassing wind, rain, temperature, humidity, and solar radiation, are important climatic parameters. Output options include temperature and relative humidity profiles, moisture content, heat flux, vapour pressure and liquid water transport.

Extensive databases are available with validated material data from laboratory test to use in the simulations. New materials with, for example measured material data from the simulated building can be added to increase result accuracy.

3. What are the key parameters that affect the hygrothermal simulation of embedded wooden beams in solid brick masonry with interior capillary active insulation most and how can these be used to designate preferred insulation measures?

Multi-year simulation can be execute within relative short periods of time, but simulation settings and computing properties of the device used for the simulations have significant influence on the usability of the software. Settings such as the discretization grid in Delphin, that influences the number of grid volumes where output values are calculated increase simulation time. With programming of simulation jobs, Delphin can run multiple jobs at the same time and options to run the simulation on a remote solver are available.

From the sensitivity analysis can be concluded that the moisture retention of brick and insulation have the most effect on the hygrothermal safety of the embedded wooden beam out of the five studied material properties. With moisture retention of brick being dominant and moisture retention of insulation having a significant smaller effect.

No influence is calculated for the thermal conductivity of insulation, the water vapour diffusion resistance of insulation and the water vapour diffusion resistance of brick. But, non-influential results seem inaccurate for similar geometries. From literature is known that temperature has a high influence on moisture content in the wall and therefore subsequently should impact dose calculation on the wooden beam. This result can be affected by the chosen exterior boundary conditions such as location weather data or wall orientation. Potentially causing the Sobol analysis to be highly responsive to output variations and indicating influential parameters to be non-influential.

Future studies should research the effects of these exterior boundary conditions more and other sensitivity analyses need to be executed on researching material properties solely.

Wood decay assessments support these findings. Comparisons between pre and post insulation situations and exclusion of insulation surrounding wooden beams indicate that, especially temperature changes due to interior insulation, affect dose calculations. Where an increase in temperature reduce the level of dose calculation and subsequently wood decay potential.

Furthermore, the impact of capillary active insulation is studied by using the wood decay assessments. Again revealing positively contributions to the prevention of wood decay. However, not capable of reducing moisture content to levels that prevent wood decay completely.

Exterior boundary conditions, such as rain load, solar radiation and orientation, are in simulations more dominant on the hygrothermal performance of the geometry. Exterior moisture is transported through the brick and influences the hygrothermal safety of the embedded wooden beams most.

Concluding, the most accurate results when simulating hygrothermal performance can be obtained by using location specific weather data and correct material data from the simulated wall. Subsequently, wood decay assessment can be used to assess potential design options and further reduce hygrothermal risks to the embedded wooden beam.

4. How can the hygrothermal performance of a building component be visualised to create a better understanding of the generated results and hygrothermal risks?

Delphin hygrothermal simulation software is used to simulate hygrothermal performance of a geometry. In this case the hygrothermal safety of embedded wooden beams in solid brick masonry with interior capillary active insulation. Visualised in Figure 79.

After running the simulation, two result files (moisture content and temperature) are processed by a Python script to calculate the dose. This dose is an number of days that indicates wood decay potential of the wooden beam. This research focuses on Spruce, but species dependent dose calculations are possible.

Hygrothermal safety of the beam is best assessed by visualising this dose value. Showing wood decay potential over time in 2D geometry grid visualisation in the already available Delphin post processor PostProc. Python scripts process the data automatically, creating similar result files as the moisture and temperature files. Separate dose specific colour maps indicate service life prediction according to current building regulations and wood species. Significantly improving understanding of wood decay risk to the beam and positively impacting the assessment of the hygrothermal performance of the simulated geometry.

An example of the dose visualisations can be found in Figure 80.

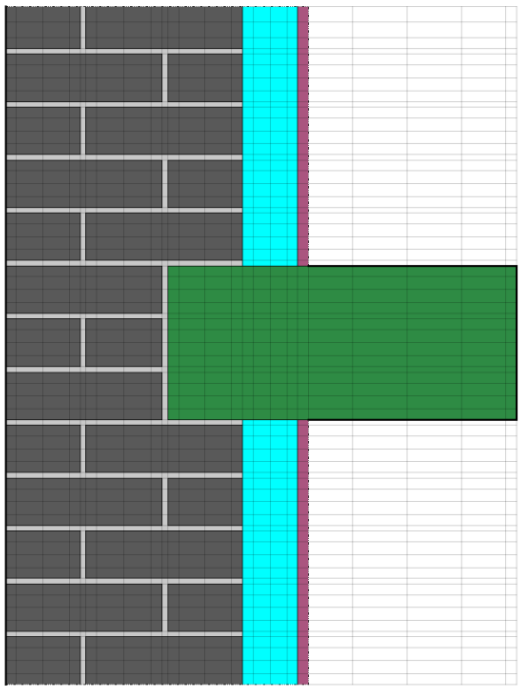


Figure 79: Geometry simulated: solid brick masonry with embedded wooden beams and interior capillary active insulation

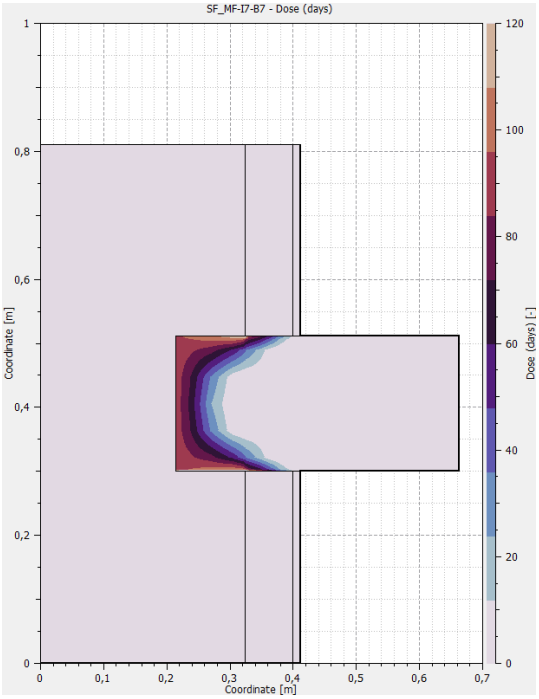


Figure 80: Example of dose visualisation, enhancing wood decay assessment

7 Recommendations

This chapter contains research work that is coherent with the research conducted. It could be an addition to this master thesis or could be a new research part on the simulation of hygrothermal performance to assess the hygrothermal safety of embedded wooden beams.

7.1 Geometry changes

This study researches only one specific geometry made out of a one and a half brick thick solid masonry wall with 75 mm of capillary active insulation and a gypsum finishing board. The effect of the thickness of the wall, placement depth of the wooden beam and insulation thickness is not studied. Next steps in research can include these changes.

7.2 Location and weather data

More knowledge is needed on the effects of location, orientation and corresponding weather data. Exterior boundary conditions are highly influential and therefor important to understand. Further research should include other climates, amounts of rain load and changes in solar radiation or orientation. All able to influence the moisture load on the wall and subsequently the risk on wood decay.

7.3 Sensitivity analysis on temperature

Next steps in finding key parameters affecting hygrothermal performance of the simulated geometries should include more research into the influence of temperature properties. These influences can develop additional understanding on the effect of capillary active insulation and the prevention of wood decay.

7.4 Wood disintegration

Due to wood and brick contact, moisture transport due to capillary suctions is increased. Additional research might study the effects of the changes by adding air gaps between the two materials. These gap can be added on purpose, for example during building renovations or due to the degradation of wood caused by wood decay.

7.5 Wood species

The effect of different wood species is a next research step. Distinct material properties, for example the wetting ability or inherent durability of wood species or coherent, the treatment of embedded wooden beam ends on the hygrothermal safety would be an interesting next research step.

7.6 Improve wood decay assessment

Wood decay assessments are currently only partly automated with the use of python scripts. The dose calculations are automatically generated but other settings and changes have to be made manually adjusted. Improvements to wood decay assessment can be made by the automatic generation of wood material coordinates where dose is calculated in every specific geometry, the automated creation of dose template files and inclusion of wood species specific service life class calculations.

Secondly, the visualisation of dose is in current work quite basic. Large differences are easily visible but alterations could be done to create more insight into small changes between compared wood decay assessments. For example, the addition of minimal and maximal values of wood decay and more detailed colour ranges and colormaps.

8 Reflection

This master thesis aimed to optimize the assessment of hygrothermal safety of wooden beam ends that are embedded in solid brick masonry and internal capillary active insulation is applied. Primarily focussing on the key input material properties that have the highest effect on decay of the wooden beam and the visualisation of those results in order to create a better visual understanding of the effect of input changes on hygrothermal safety. However, initially the scope of the project and objective I wanted to reach was broader. Starting from P2 it was my goal to create a comprehensive, quick and accurate method to simulate the hygrothermal performance of the interior insulation around the wooden beam ends. But this quickly seemed to be a step to far. Because, before creating the best method for simulating and assessing the simulation, results when varying the simulation programs input should be research. How does the geometry and possible changes in settings, both material and program, affect the hygrothermal performance of a construction. And thus, finding what is important to get right in order to receive accurate results from the simulation, eventually resulting in being able to give a more accurate advise to clients.

In order to reach this objective I had to gain more theoretical knowledge. Dive deeper into the settings of Delphin, find what input was used in the calculations and make decisions how I could research the input effects on hygrothermal performance. I started with simple simulations, changing settings as wind driven rain, solar radiation and detail of geometry but quickly realized that a comprehensive and accurate analysis of the software needed a more complete method. With the knowledge from the first simulations: the fact that the application of WDR and SR have significant influence on the simulation results and should thus be included in future simulations, highest moisture loads were found on south and west orientated facades and the detail of the geometry does not have a big influence on the simulation results, I explored other possible high influential settings that could influence the results. From the literature study and during talks with my mentors it became clear that material properties would have big effects on the simulation results.

I determined at P3 that 10 material properties would be research on sensitivity on the simulation results with the Sobol sensitivity analysis and the moisture dose component as result value used in the sensitivity analysis. For this I needed to learn Python programming from the basics, and how to control the Delphin software with the programmed scripts. Tasks sometimes seemingly impossible but in the end being moments where you just have to divide the problem in to multiple smaller steps in order to reach these bigger end goals. With the guidance of mentors, and colleagues at ABT I eventually solved the problems, always taking longer than anticipated. While gaining more theoretical knowledge the number of variable material parameters decreased to 5 and became more complex. It made myself familiar with new hygrothermal functions and integrated these in the scripts. All in order to find more accurate results.

Looking back at the plans I had for this master thesis at P2 and P3, I hoped to be a bit further in my research at this point. Possibly it's the feeling that I've put in a lot of research and programming time, but results are not really as I would have expected. Both in quality and quantity. On the contrary, the thoroughness of the research conducted is increased significantly, maybe even more important than result alone.

Evaluating at P4, I had an enormous list of possible additions to the research. For example: researching wood an-isotropic behaviour, location, orientation, and weather influences. Adding more variables, creating better visualisations, changing geometries and improve the analysis of result. Available time was, when looking back, not sufficient for all these changes and new research opportunities. But eventually at P5, I implemented all these changes and have succeeded in making a master thesis where I'm extremely satisfied about.

I'm proud that I've learned so much. Theorical knowledge about all topics that can be associated with hygrothermal performance of building components, and learned a lot of new skills. For example, software skills but also skill to do literature research more efficiently. I still find it hard to motivate myself and work efficiently every day, but think I did a quite adequate job during this master thesis. I've found that discussing problems, really helped solving them. New insights, opinions and knowledge really helped to reach a higher level, and people really would like to help if you ask them. All beneficial to reaching this master thesis objective.

1. What is the relation between your graduation project topic, your master track (A, U, BT, LA, MBE), and your master programme (MSc AUBS)?

The total building sector is going through an enormous transformation towards sustainability. Architecture, construction and consultancy companies are all searching for ways to reduce the use of finite materials and energy and to reuse materials and energy in order to contribute to Dutch, European and global environmental goals. This thesis, where the use of internal, capillary active insulation around wooden beam ends in solid brick masonry is researched, contributes to those challenges.

With the reuse of buildings, there is a decrease in need for new buildings and thus the need for new materials and corresponding CO2 emissions. Buildings are more energy efficient and can been given other functions. Biobased materials minimize CO2 emissions, while the application of indoor insulation preserves their aesthetical, historical and cultural value. Additionally, new innovative materials, like capillary active insulation, are researched and tested and integrated into current building standards. Bridging the gap between the architectural design and engineering and improving the implementation of innovations. Both can be seen as principles of the master track Building Technology and the master of Architecture, Urbanism and Building Sciences. Where we have set ourself the goal to integrate new knowledge and skills into design practices to create sustainable development.

2. How did your research influence your design/recommendations and how did the design/recommendations influence your research?

This research primarily focussed on providing recommendations when simulating building components in Delphin hygrothermal simulation software, especially when focussed on embedded wooden beam in solid brick masonry facades. Finding key parameters that affect wood decay. The findings of this research can be used in future hygrothermal simulations, by giving an indication of parameters that have an high influence on the simulation results and can be taken into account when giving advice to clients in similar practical situations. Wood decay assessment can be used in all kinds of situations, assessing design options and hygrothermal safety in all kinds of locations, orientations and weather types worldwide.

3. How do you assess the value of your way of working (your approach, your used methods, used methodology)?

I think, when looking back to the start of this master thesis, that I gained an substantial amount of (theoretical) knowledge. I learned from scratch how to use new software programs like Delphin hygrothermal simulation software and programming in Python and learned a lot on material properties and their behaviour, especially when looking at moisture and heat properties.

When assessing my approach, used methods and methodology from this starting point I'm really satisfied with the completed work. Continuously adaption to new insights, new skills and knowledge to lift this master thesis to a higher level. But, I'm also very aware that I could have reach a higher research level if could have started with this all knowledge that I have gathered to this point. I would have made different choices, included more important research in order to create a more complete study and could have done work so much more efficient. Knowing that this is part of learning and only that I can see this because of the fact I learned so much. I'm therefore definitely satisfied with the approach and think the used methods positively impacted the value of the final result.

4. How do you assess the academic and societal value, scope and implication of your graduation project, including ethical aspects?

With the renovation of monumental buildings, aesthetic, cultural and historical value is preserved. Internal insulation ensure that the outside of the building stays the same, keeping their character and cultural and historic features. Secondly, hygrothermal performance of building facades have a significant positive effect on the users of the building. Creating a safe and comfortable indoor environment.

But societal impact is not only made on the users or cultural aspects of the building. Building construction companies, architects and consultant companies are in need for quick, correct and more accurate ways to simulate the hygrothermal effects of interior insulation. The results of this thesis can be used in future research and to develop the best possible solution for insulation around embedded wooden beams in practice. Resulting in an increase in monumental buildings that can be renovated. Contributing to the needed increase in deep building renovations to reduce material use, conserve energy and minimise CO2 emissions eventually aiding in reaching the UN Paris Agreement goals.

Finally academic relevance and value. Standard insulation has been tested and applied, but new, biobased and capillary active materials that can reduce humidity levels in building envelopes have to be researched and developed further. More knowledge has to be gathered on the storage, transport and absorption and desorption of the capillary active materials around wooden beam ends, simultaneously verifying simulation models. Applied to transport moisture back to the indoor side of the insulation layer to minimize moisture induced damages, while still preserving their functions as insulation material. Contributing to finding design solutions that can be used within the building sector.

5. How do you assess the value of the transferability of your project results

During the entire master thesis period, I've focussed on the fact that I would like my research to be used in practice. From the start I've tried to make my research to be as transparent as possible. Within my report there is a chapter on the inputs and choices made in the material parameter study, that could be used when others want to use my results. For example in other sensitivity analysis or to extend this specific analysis with the recommendations from this study.

Besides the sensitivity analysis results, the wood decay assessments are made to be used by others. To make the wood decay risks more visual and understanding, even for people without the knowledge on the subject. With the python script that is used to make the visualisations, standardly divided into different parts and parts of text added to create a better understanding of the specific task of the scripted part. In addition to the script, extra information is written within the thesis report for more clarification and transferability of the project and corresponding results.

6. What is the practical value of the innovative capillary active materials from this master thesis?

The use of capillary active materials for interior insulation is quite new in current building construction, precise effect on building hygrothermal performance is still unknown and the application of this new material when transforming or renovating buildings is often complicated. The material and its important real life application rules, such as the need for direct contact with the old wall and the installation without air gaps, is not yet common and therefor an risk on performance when installed in projects.

This research adds to the knowledge needed for the real life application of these new innovative materials, probably used by consultancy companies in the first place but eventually leading to new, and hopefully standard building practices in the future. Making the renovation of old buildings with more (biobased) capillary active materials possible and reducing environmental impact.

7. How did the graduation internship at ABT influence your master thesis?

During this master thesis, I had the opportunity to do a graduation internship at ABT in Delft. A consultancy company part of the Oosterhof group. ABT provided me with all resources I needed to do my research, both practical and knowledge wise. When asked, I received all the help I needed, for example with Python programming questions, or missing theoretical knowledge. All crucially important for the progress of this research and definitely one of the reasons that I reached the research level to the point where I'm right now.

The fact that I had my own place to work and the structural motivation to work at least those 8 hours every day helped me a lot. I'm confident this had a tremendous positive effect on the total amount of hours and work that went into this thesis, motivating me every day to start and keep improving my work. I'm therefor very grateful for this opportunity and appreciate this immensely.

9 References

Bauklimatik dresden (n.d.). *Delphin hygrothermal software.* https://www.bauklimatik-dresden.de/delphin/index.php

Bauklimatik dresden (n.d.). *Dokumentation & Hilfe*. https://www.bauklimatik-dresden.de/delphin/documentation.php

Blumberga, A., & De Place Hansen, E. J. (2020). Robust internal thermal insulation of historic buildings. In RIBuild. European Commission.

Blumberga, A., Blumberga, D., Kamendere, E., Kamenders, A., Kass, K., Purvins, R., & Zogla. G. (2015). Report on historical building types and combinations of structural solutions. RiBuild Deliverable D1.1 RIBuild Template Deliverables (squarespace.com)

Brischke, C., & Meyer-Veltrup, L. (2015). Modelling timber decay caused by brown rot fungi. *Materials And Structures*, *49*(8), 3281–3291. https://doi.org/10.1617/s11527-015-0719-y

Brischke, C., & Meyer-Veltrup, L. (2015b). Modelling timber decay caused by brown rot fungi. *Materials And Structures*, *49*(8), 3281–3291. https://doi.org/10.1617/s11527-015-0719-y

Brischke, C., & Rapp, A. O. (2008). Dose–response relationships between wood moisture content, wood temperature and fungal decay determined for 23 European field test sites. *Wood Science And Technology*, *42*(6), 507–518. https://doi.org/10.1007/s00226-008-0191-8

Carbonez, K., Van Den Bossche, N., Ge, H., & Janssens, A. (2015). Comparison between uniform rain loads and point sources to simulate rainwater leakage with commercial HAM-models. Proceedings from ISBP2015: International Symposium on Building Patholog). Porto (Portugal), 24-27 March 2015

Council of the EU. (2023, 23 oktober). Paris Agreement: Council submits updated NDC on behalf of EU and member States. https://www.consilium.europa.eu/en/press/press-releases/2023/10/16/paris-agreement-council-submits-updated-ndc-on-behalf-of-eu-and-member-states/

Dang, X., Janssen, H., & Roels, S. (2023). Hygrothermal Modelling of one-dimensional Wall Assemblies: inter-model Validation between WUFI and DELPHIN. Journal Of Physics. Conference Series, 2654(1), 012040. https://doi.org/10.1088/1742-6596/2654/1/012040

Defo, M., Lacasse, M., & Laouadi, A. (2021). A comparison of hygrothermal simulation results derived from four simulation tools. Journal Of Building Physics, 45(4), 432–456. https://doi.org/10.1177/1744259120988760

Delgado, J. M. P. Q., Ramos, N. M., Barreira, E. and V. P. De Freitas. (2010). A critical review of hygrothermal models used in porous building materials. Journal of Porous Media, 13(3), 221-234.

Delgado, J. M., Barreira, E., Ramos, N. M., & De Freitas, V. P. (2013). Hygrothermal Numerical Simulation Tools Applied to Building Physics. In SpringerBriefs in applied sciences and technology. https://doi.org/10.1007/978-3-642-35003-0

DELPHIN. (z.d.). Bauklimatic-Dresden. Geraadpleegd op 7 juni 2024, van https://www.bauklimatik-dresden.de/delphin/index.php?aLa=en

Economidou, M., Atanasiu, B., Despret, C., Maio, J., Nolte, I., Rapf, O., Laustsen, J., Ruyssevelt, P., Staniaszek, D., Strong, D., & Zinetti, S. (2011). Europe's buildings under the microscope. A country-by-country review of the energy performance of buildings. BPIE. https://www.osti.gov/etdeweb/biblio/21514343

European Commission. (2018, 23 maart). Insulation retrofit for energy-efficient historic buildings. https://projects.research-and-innovation.ec.europa.eu/en/projects/success-stories/all/insulation-retrofit-energy-efficient-historic-buildings

European Commission. (2019). Energy performance of buildings. Retrieved from https://ec.europa.eu/energy/en/topics/energy-efficiency/energy-performance-of-buildings (accessed on Jul 19, 2019)

Finnish mould growth model. (z.d.). Tampere University - Building Physics. Geraadpleegd op 8 juni 2024, van https://research.tuni.fi/buildingphysics/finnish-mould-growth-model/

Fraunhofer IBP. (z.d.). WUFI® Mould Index VTT. WUFI. Geraadpleegd op 8 juni 2024, van https://wufi.de/en/2017/03/31/wufi-mould-index-vtt/

Fraunhofer IRB (2014). WTA Guideline 6-2. Simulation of Heat and Moisture Transfer. Fraunhofer IRB Verlag.

Glass, S. V., TenWolde, A., & Zelinka, S. L. (2013). Hygrothermal Simulation: A Tool for Building Envelope Design Analysis. In USDA.

Goffart, J., & Woloszyn, M. (2021). EASI RBD-FAST: An efficient method of global sensitivity analysis for present and future challenges in building performance simulation. *Journal Of Building Engineering*, *43*, 103129. https://doi.org/10.1016/j.jobe.2021.103129

Groot, C. & Gunneweg, J. (2007). OR1 - Historische metselwerk:Deelonderzoek: Kwaliteitseisen Restauratiebaksteen. Delft University of Technology. http://resolver.tudelft.nl/uuid:d1ac14cf-d694-4365-a78e-c31e3739a034

Hall, & Casey, S. (2012). Hygrothermal behaviour and occupant comfort in modern earth buildings. In *Elsevier eBooks* (pp. 17–40). https://doi.org/10.1533/9780857096166.1.17

Harrestrup, M., & Svendsen, S. (2016). Internal insulation applied in heritage multi-storey buildings with wooden beams embedded in solid masonry brick façades. Building And Environment, 99, 59–72. https://doi.org/10.1016/j.buildenv.2016.01.019

Hejazi, B., Sakiyama, N. R. M., Frick, J., & Garrecht, H. (2020). Hygrothermal Simulations Comparative Study: Assessment of Different Materials Using WUFI and DELPHIN Software. Building Simulation Conference Proceedings. https://doi.org/10.26868/25222708.2019.211033

Helmenstine, A. (2022, 21 maart). *Isotropic vs Anisotropic - Definition and Examples*. Science Notes And Projects. https://sciencenotes.org/isotropic-vs-anisotropic-definition-and-examples/

Hubregtse, N. (2023). *Vapour-open, non-capillary active internally insulated historic solid brick masonry. The influence of hygrothermal properties on the hygrothermal performance.*

Hukka A. & Viitanen H. (1999). A mathematical model of mould growth on wooden material. Wood Science and Technology 33(6): 475-485.

Hutkai, K., & Katunský, D. (2021). Insulation of historic buildings and case study simulation. IOP Conference Series. Materials Science And Engineering, 1209(1), 012003. https://doi.org/10.1088/1757-899x/1209/1/012003

International Energy Agency. (2020). Energy Technology Perspectives 2020. (International Energy Agency, 2020)

International Energy Agency. (2021). Net Zero by 2050: A Roadmap for the Global Energy Sector. (International Energy Agency, 2021)

Isaksson, T., Brischke, C., & Thelandersson, S. (2012). Development of decay performance models for outdoor timber structures. *Materials And Structures*, *46*(7), 1209–1225. https://doi.org/10.1617/s11527-012-9965-4

Jensen, N. F., Møller, E. B., Hansen, K. K., & Rode, C. (2023). Hygrothermal assessment of solid masonry walls internally insulated with bio-based insulation materials. AIP Conference Proceedings. https://doi.org/10.1063/5.0173596

Krus, M. (1996). Moisture Transport and Storage Coefficients of Porous Mineral Building Materials: Theoretical Principles and New Test Methods. [PhD-thesis, Fraunhofer Institute for Building Physics]

Künzel, H. M., Kiessl, K. (1996). Calculation of heat and moisture transfer in exposed building components.

Künzel, H., & Karagiozis, A. (2010). Hygrothermal behaviour and simulation in buildings. In *Elsevier eBooks* (pp. 54–76). https://doi.org/10.1533/9781845699277.1.54

Künzel, H.M. (1995). Simultaneous Heat and Moisture Transport in Building Components: One- and two-dimensional

Lemieux, C., & Lemieux, V. (2009). Monte Carlo and Quasi-Monte Carlo Sampling. In *Springer series in statistics*. https://doi.org/10.1007/978-0-387-78165-5

Meyer-Veltrup, L., Brischke, C., Alfredsen, G., Humar, M., Flæte, P., Isaksson, T., Brelid, P. L., Westin, M., & Jermer, J. (2017). The combined effect of wetting ability and durability on outdoor performance of wood: development and verification of a new prediction approach. *Wood Science And Technology*, *51*(3), 615–637. https://doi.org/10.1007/s00226-017-0893-x

Mijzen, D. T. (2024, July 8). *RC waardes Bouwbesluit | Nieuw-Bestaande bouw - HBA*. Handelbouwadvies. https://www.handelbouwadvies.nl/rc-isolatie-waarde-bouwbesluit/

Mitropoulos, A. C. (2009). Capillarity. Journal Of Engineering Science And Technology Review, 2(1), 28–32. https://doi.org/10.25103/jestr.021.06

Nederlands Normalisatie Instituut. (2013). NEN-EN-ISO - Hygrothermische prestatie van bouwcomponenten en -elementen –Binnenoppervlaktetemperatuur om kritische oppervlaktevochtigheid en inwendige condensatie te vermijden – Berekeningsmethode (ISO 13788:2012,IDT). Nederlands Normalisatie-instituut.

NOAA. (2024, 18 januari). Climate change: Global temperature. https://www.climate.gov/news-features/understanding-climate/climate-change-global-temperature

Nossent, J., Elsen, P., & Bauwens, W. (2011). Sobol' sensitivity analysis of a complex environmental model. *Environmental Modelling & Software*, *26*(12), 1515–1525. https://doi.org/10.1016/j.envsoft.2011.08.010

Posani, M., Veiga, R., & De Freitas, V. P. (2021). Retrofitting Historic Walls: Feasibility of Thermal Insulation and Suitability of Thermal Mortars. Heritage, 4(3), 2009–2022. https://doi.org/10.3390/heritage4030114

Ravenshorst, G. (2015). Species independent strength grading of structural timber. *Proefschrift*. https://doi.org/10.4233/uuid:b2243d5d-3275-4d93-9378-b39f97a39f02

Ruisinger, U., & Kautsch, P. (2020). Comparison of hygrothermal 2D- and 3D-simulation results with measurements from a test house. *E3S Web Of Conferences*, *172*, 08004. https://doi.org/10.1051/e3sconf/202017208004

SALib - Sensitivity Analysis Library in Python — SALib's documentation. (z.d.). https://salib.readthedocs.io/en/latest/index.html

Sirag, M., & Wiedijk, K. (1950). Bouwmaterialen. Van Mantgem.

Stappers, M. (2019). Na-isolatie van historische woonhuizen. Rijksdienst Voor het Cultureel Erfgoed.

Stappers, M. (2021). Binnengevelisolatie van monumenten: dampopen of capillair actief?. Rijksdienst Voor het Cultureel Ergoed.

Stenvert, R. (2012). Biografie van de baksteen, 1850-2000. (Stichting UvA Vertalers (Amsterdam), Trans.). WBooks.

Stenvert, R., & Van Tussenbroek, G. (2007). Inleiding in de bouwhistorie: opmeten en onderzoeken van oude gebouwen. Matrijs.

Tzeiranak, S. T., Daniele, P., Castellazzi, L., Ribeiro Serrenho, T., Economidou, M., & Zangheri, P. (2020). Energy Consumption and Energy Efficiency trends in the EU-28, 2000-2018. JRC Science For Policy Report. https://doi.org/10.2760/847849

United Nations Framework Convention on Climate Change. (2015). *Paris Agreement*.

Van Genuchten, M. T. (1980). A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Science Society Of America Journal*, *44*(5), 892–898. https://doi.org/10.2136/sssaj1980.03615995004400050002x

Van Hemert, R. (2013). Houtconstructies: balklagen, gebinten, kapconstructies (2e geh. herz. en uitgebr. druk, Ser. Restauratie timmerwerken, dl. 3). NRC.

Van Hunen, M. (2012). Historisch metselwerk: instandhouding, herstel en conservering. WBooks.

Vereecken, E., & Roels, S. (2019). Wooden beam ends in combination with interior insulation: An experimental study on the impact of convective moisture transport. Building And Environment, 148, 524–534. https://doi.org/10.1016/j.buildenv.2018.10.060

Vereecken, E., Van Gelder, L., Janssen, H., & Roels, S. (2015). Interior insulation for wall retrofitting – A probabilistic analysis of energy savings and hygrothermal risks. Energy And Buildings, 89, 231–244.

Vereecken, E., Vanoirbeek, K., & Roels, S. (2015). A Preliminary Evaluation of Mould Prediction Models Based on Laboratory Experiments. Energy Procedia, 78, 1407–1412. https://doi.org/10.1016/j.egypro.2015.11.162

Viitanen H. & Bjurman J. (1995). Mould growth on wood under fluctuating humidity conditions. Mat. und Org. 29(1): 27-46.

Viitanen H. (1996). Factors affecting the development of mould and brown rot decay in wooden material and wooden structures. Effect of humidity, temperature and exposure time. Doctoral thesis. Uppsala. The Swedish University of Agricultural Sciences, Department of Forest Products. 58 p.

Viitanen H., Hanhijärvi A., Hukka A. & Koskela K. (2000). Modelling mould growth and decay damages Healthy Buildings. Espoo, 6 - 10 August 2000. Vol. 3. FISIAQ, 2000, p. 341–346.

Viitanen, H., Vinha, J., Peuhkuri, R., Ojanen, K., Lähdesmäki, K., & Salminen, K. (2008). Development of an improved model for mould growth: Modelling. In C. Rode (Ed.), Proceedings of the 8th Symposium on Building Physics in the Nordic Countries NSB2008, Copenhagen, June 16-18, 2008 (pp. 927-934) http://urn.fi/URN:NBN:fi:tuni-201911256261

Vogelsang, S., Fechner, H., Nicolai, A.. (2013) Delphin 6 Material File specification, Version 6.0

Wattjes, J. G. (1934). Constructie van gebouwen (Vol. Dl. vi, vloeren, binten, kolommen, plafonds en skeletbouw). Kosmos.

Yildiz, Y. (2021). An Overview of Hygrothermal Simulation Tools. Dicle University Journal Of The Institute Of Natural And Applied Science.

Zhang, X., Trame, M., Lesko, L., & Schmidt, S. (2015). Sobol Sensitivity Analysis: A Tool to Guide the Development and Evaluation of Systems Pharmacology Models. *CPT Pharmacometrics & Systems Pharmacology*, *4*(2), 69–79. https://doi.org/10.1002/psp4.6

Zwiers, L. (1920). Handboek der burgerlijke bouwkunde (Vol. Dl. 1, houtconstructies). Van Mantgem en de Does

10 Appendix

10.1 Methodology

This paragraph gives an overview of the steps taken in order to find the most influential material parameters that effect wood decay. Multiple programs are used together to do the simulations.

Running the simulations is divided into two parts: Creating the material and simulation variations and running and calculating the results.

10.1.1 Part 1: Creating variations

To find the most influential material properties affecting the risk on wood decay, a sensitivity analysis is executed. This analysis creates a number of simulations with different material parameters and analyses the simulating results. This gives insights in the effect of the material changes to the total result but also the affect to other material properties.

Therefor new material and simulation files need to be created. These can afterwards be used in Delphin to calculate the moisture and temperature in the geometry. The steps in short:

1. Create Delphin template model
2. Create simulation and material template files
3. Create SA Sobol parameter variation values
4. Replace simulation and material placeholders
5. Write simulation jobs

1. Within Delphin a template model is made, in this model the geometry is created and all program settings are set correctly. For example: simulation time, location, boundary conditions, output files, discretization grid (dividing a geometry in discrete grid points) and indoor / outdoor interfaces.

The simulation of the model is executed once, this creates a simulation file that can later be used as template file and tests if the model runs correctly and no errors occur in the simulation job. Output:

- {Delphin Model Template.d6p}

2. In the next step the simulation and material template files are made. The new parameters resulting from the Sobol samples, are placed in the template files specific placeholders, creating similar simulation and material files except for the material variations. Output:

- {Simulation Template File.d6p}
- {Material Brick Template File.m6}
- {Material Insulation Template File.m6}

3. In Python, with additional libraries, the generation of Sobol samples is executed. The Sobol sampler uses a problem statement as input (with material parameters to vary and corresponding bounds) to create a set of representative values for a correct analysis without the need for executing al possible simulations. These values are stored in a dataset and saved to a .csv file.

4. A second Python script imports the dataset and replaces all the values to the correct material placeholders. Additionally the script calculates and replaces the new moisture storage functions by using the Van Genuchten moisture retention curve function. For every Sobol sample (simulation that needs to be executed) there is a simulation and two material files as output:

- {SF_MF-I*n*-B*n*.m6} Simulation file
- {MF-B*n*.m6} Material brick file
- {MF-I*n*.m6} Material insulation file

5. For every Sobol sample a simulation job is written. This is a line of text where the Delphin executable is called and some additional settings are defined. With this line every individual simulation file can be executed. All job lines are written and stored in a .csv file. Python writes and uses these jobs to run the Delphin software on the background of the computer. The written job:

- {[*Executable*, '-x', '—verbosity level=0', '-p=2', SA_Variants/Simulation/+ SF_MF-I*n*-B*n*.m6}

10.1.2 Part 2: Running and calculating results

This part imports and calculates the dose and exports the result files to be used in visualisations. The steps in short:

6. Run simulation jobs in Delphin
7. Import result data and calculate dose
8. Run Sobol sensitivity analysis on results
9. Visualise wood decay in Delphin PostProc

6. The second Python script additionally imports the .csv file with the Delphin executable jobs. The additional JobRunner.py is used to run multiple jobs in parallel, making use of the multi core CPU of the laptop used (in this case 4). The script additionally creates a results folder with the same simulation file name. Output files used for the daily dose calculation:

- {Moisture content profile.d60}
- {Temperature.d60}
- {Coordinate info.g6a}

7. Python script three imports these calculated results for every simulation and stores this data in multiple Python lists. For every simulation file it creates two times a list with multiple lists inside. Representing one moisture list and one temperature list, within there is for every day a list of values. This second list represents the specific value on a grid point of the geometry. Schematic overview:

Simulation 1:
- Moisture content of simulation 1:
    - [Day 1: [Coordinate 1, Coordinate 2, C*n*],
    - Day 2: [Coordinate 1, Coordinate 2, C*n*],
    - Day *m:* [Coordinate 1, Coordinate 2, C*n*]]

- Temperature of simulation 1:
    - [Day 1: [Coordinate 1, Coordinate 2, C*n*],
    - Day 2: [Coordinate 1, Coordinate 2, C*n*],
    - Day *m*: [Coordinate 1, Coordinate 2, C*n*]]

Simulation *i:*
- Moisture content of simulation *i*:
    - [Day 1: [Coordinate 1, Coordinate 2, C*n*],
    - Day 2: [Coordinate 1, Coordinate 2, C*n*],
    - Day *m*: [Coordinate 1, Coordinate 2, C*n*]]

- Temperature of simulation *i:*
    - [Day 1: [Coordinate 1, Coordinate 2, C*n*],
    - Day 2: [Coordinate 1, Coordinate 2, C*n*],
    - Day *m*: [Coordinate 1, Coordinate 2, C*n*]]

Where:

i	= the number of simulations:	1536
m	= the number of days:	1095
n	= the number of coordinates:	978

The next step is the calculation of the daily dose. This is calculated for all coordinates on all days and uses the corresponding temperature and moisture data of that specific coordinate of the specific day.

The total dose is a cumulative calculation, the script adds the daily dose of a coordinate to the total dose for that coordinate. Resulting in a result file that is similar to the Delphin output files and can thus be used to visualised in the visualisation program PostProc.

8. The values that are analysed by the Sobol analyser are calculated with script three and saved to a .csv file. These values are:

- Maximum moisture content in beam
- Average moisture content in beam
- Maximum wood decay (dose) in beam
- Average wood decay (dose) in beam

Saved to:

- {SA_simulation_results.csv}

The moisture and dose result files are used to be analysed in the Sobol analysis. The final Python script (four) imports the Sobol values and calculates the First, Second and Total order indices for all the four sensitivity analysis.

With the script the Sobol values are plotted in graphs. Contributing to the visualisation of results.

9. The final part is the visualisation of the calculated wood decay values to improve both understanding of beam decay and assessment of hygrothermal safety. With Excel the coordinates of the wooden beam are listed, to be able to only calculate dose values where the material of the geometry is wood.

Moisture and temperature files of the selected simulation to be visualised are imported in Python script five. To ensure correct geometrical data from the specific simulation, the moisture result file is manually copied to be used as template file for the calculated dose values. Within the copied file, text is adapted to correct names and unit and the placeholder for dose values is added. Named:

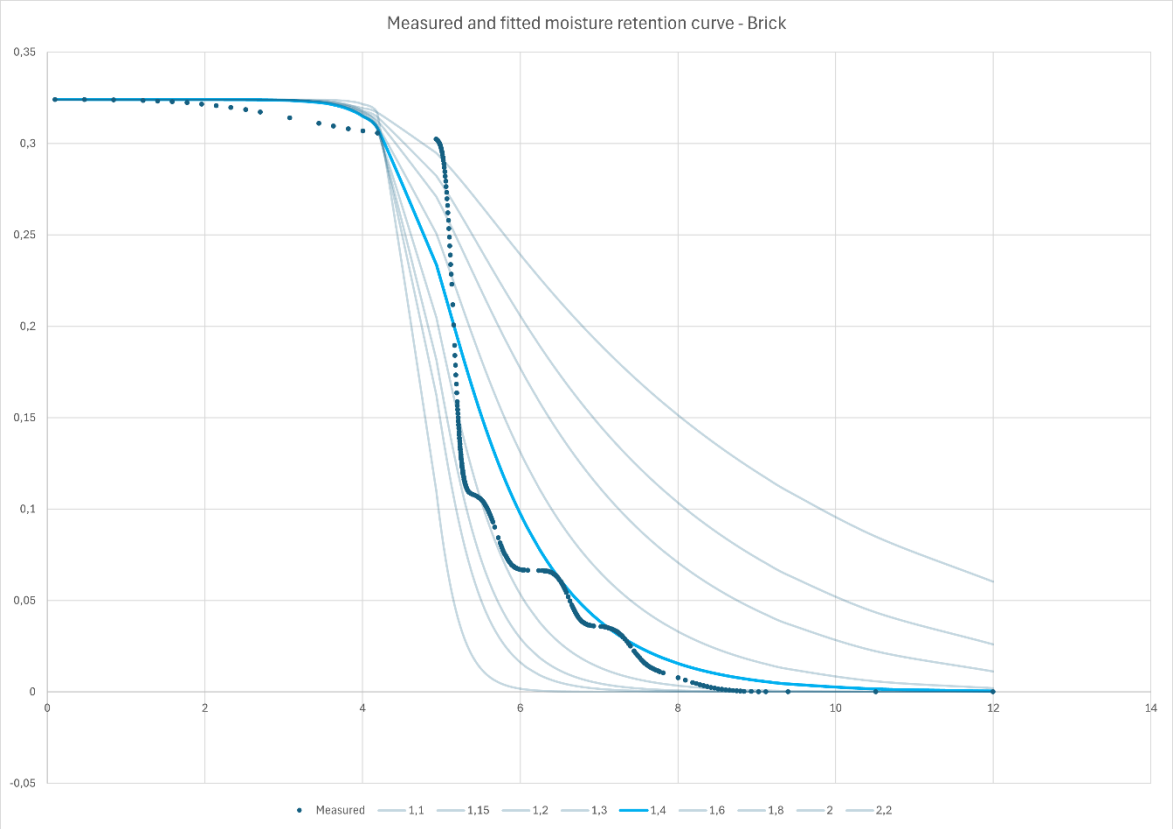
- {Dose calculation profile template.d6o}

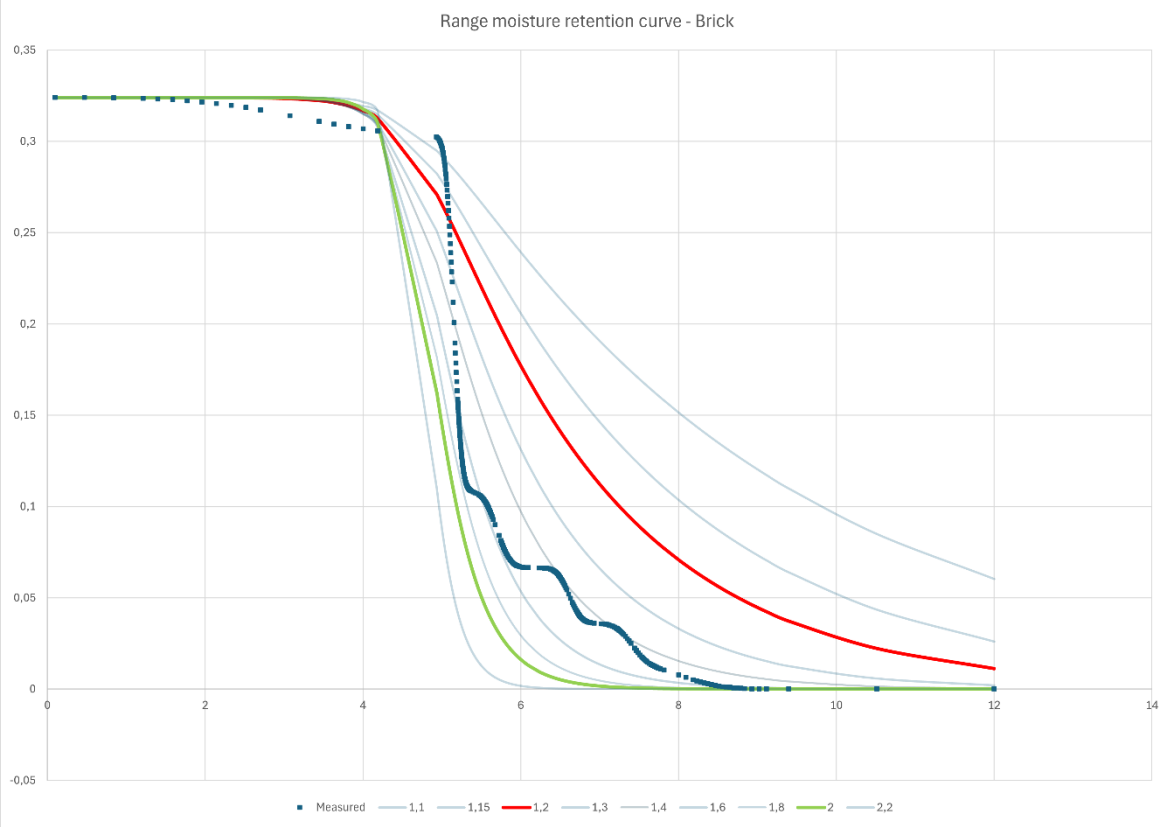
The script imports the dose template file, calculates the dose values from the moisture, temperature and coordinate data and export the dose result file to be visualised in Delphin PostProc:

- {Dose calculation profile.d6o}

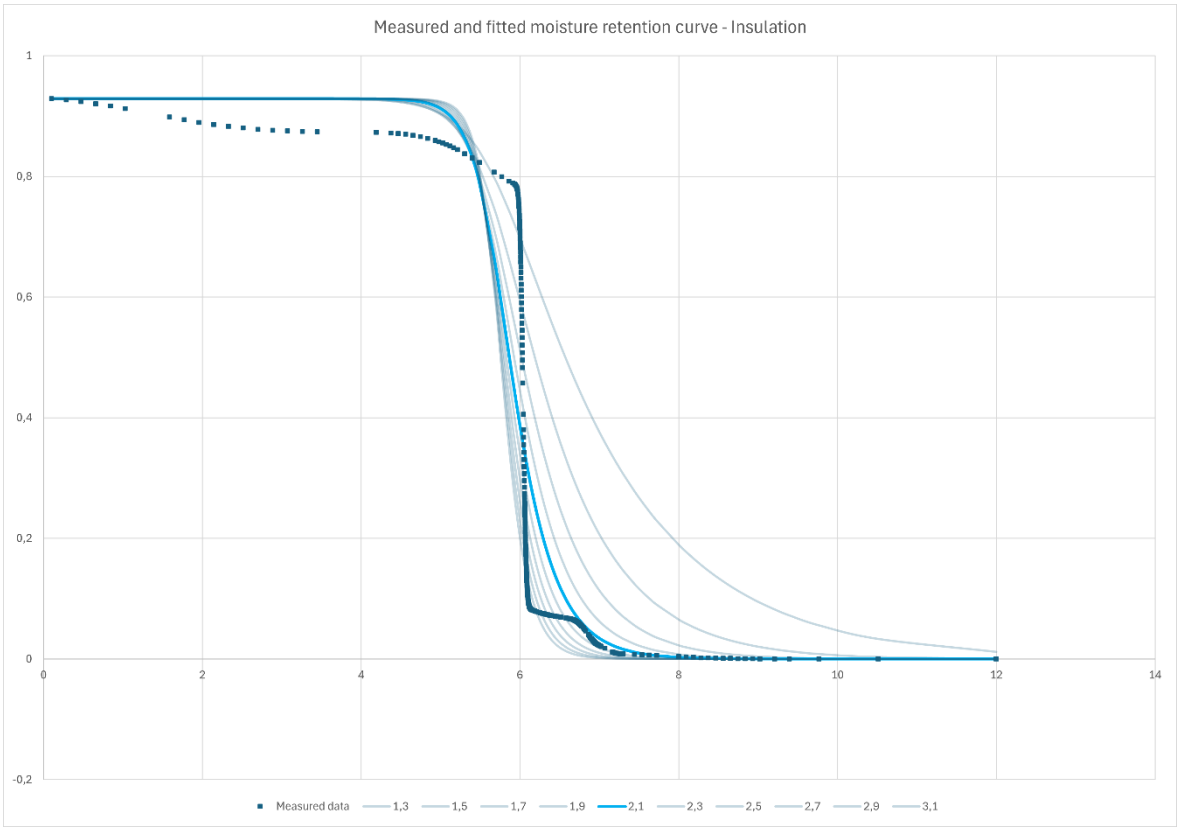
10.2 Moisture retention curves

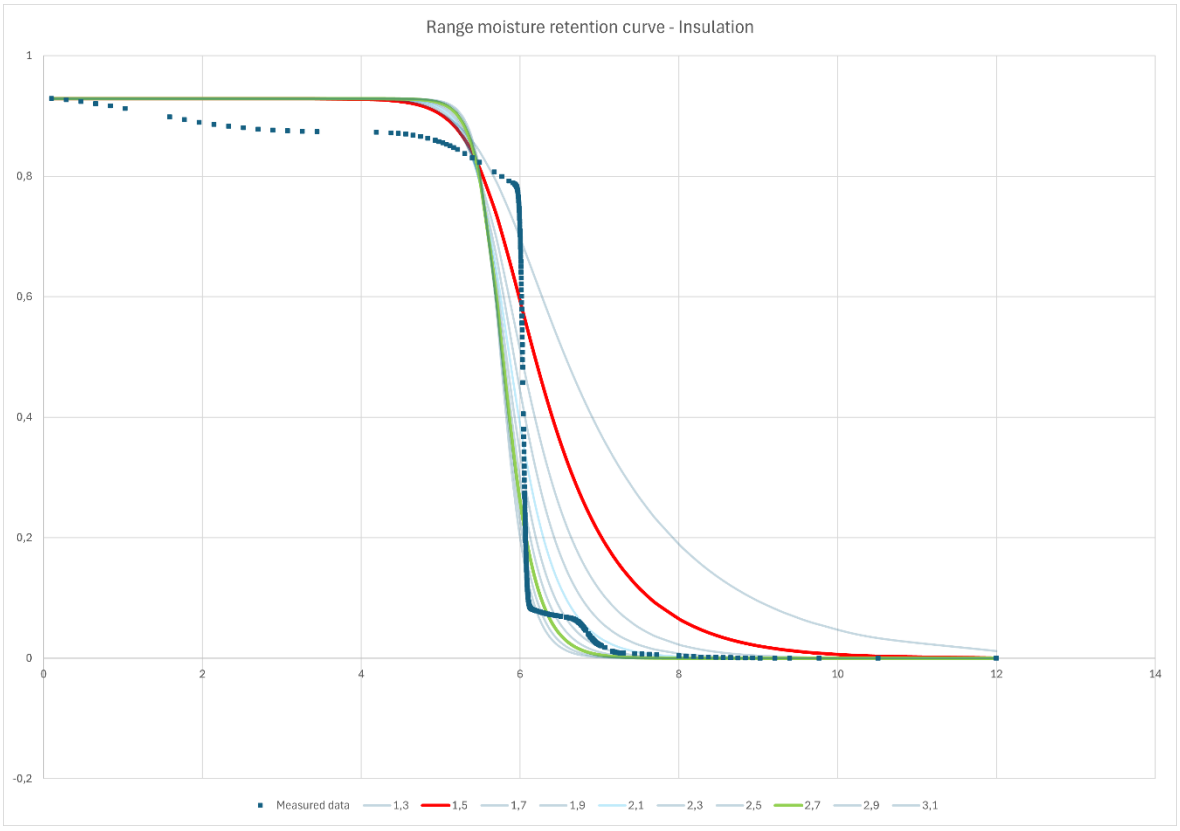
10.2.1 Brick





10.2.2 Insulation





10.3 Moisture and temperature output file

10.3.1 Moisture

```
Moisture content profile.d6o - Notepad
File  Edit  Format  View  Help
D6OARLZ! 007.000
TYPE          = FIELD
PROJECT_FILE  = SF_MF-I0-B0.d6p
CREATED       = Thu Dec 19 11:28:32 2024
GEO_FILE      = SF_MF-I0-B0_592561214.g6a
GEO_FILE_HASH = 574338514
QUANTITY      = Total mass of moisture per mass of REV
QUANTITY_KW   = MoistureMassByMass
SPACE_TYPE    = SINGLE
TIME_TYPE     = NONE
VALUE_UNIT    = kg/kg
TIME_UNIT     = h
START_YEAR    = 2020
INDICES       = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
9 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321
5 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577
1 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833

0               0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.00062020
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.00062020
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.00062020
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.152577        0.00062020
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.152577        0.152577        0.152577
085245          0.085245        0.085245        0.085245        0.085245        0.085245        0.000620206     0.000620206     0.000620206     0.000620206     0.00062020
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.152577        0.152577        0.152577        0.152577        0.152577        0.152577
152577          0.152577        0.152577        0.152577        0.152577        0.152577        0.085245        0.085245        0.085245        0.085245        0.085245
085245          0.00848389      0.00848389      0.00848389      0.00848389      0.00848389      0.000620206     0.000620206     0.000620206     0.000620206     0.00062020
000620206       0.000620206     0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577
000620206       0.000620206     0.000620206     0.000620206     0.000620206     0.000620206     0.085245        0.152577        0.152577        0.152577        0.152577
085245          0.085245        0.085245        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577
152577          0.152577        0.152577        0.152577        0.152577        0.152577        0.000620206     0.085245        0.085245        0.085245        0.085245
085245          0.085245        0.085245        0.085245        0.085245        0.00848389      0.00848389      0.00848389      0.152577        0.152577        0.152577
152577          0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.152577        0.085245
00848389        0.152577        0.152577        0.152577        0.152577        0.152577        0.00848389      0.00848389      0.152577        0.152577        0.152577
24              0.000782578     0.000783188     0.000713422     0.000784489     0.000713814     0.000568256     0.000786616     0.000714657     0.000568295     0.00061009
000614053       0.000614052     0.000614049     0.000803088     0.000723674     0.000569431     0.000609921     0.000614248     0.000613434     0.00061315      0.00061393
000619447       0.000626022     0.000617254     0.000614177     0.000614018     0.00061381      0.000613779     0.000614914     0.000634853     0.000803087     0.00072583
000723523       0.00056945      0.000609903     0.000614105     0.000613002     0.000612816     0.000624308     0.000639994     0.000639399     0.154165        0.00063044
000569405       0.000609921     0.000614237     0.00061334      0.000613122     0.000624308     0.000643467     0.000649339     0.155461        0.154622        0.152691
0959484         0.0865288       0.0845751       0.0828883       0.0731595       0.056523        0.000784489     0.000716052     0.000568765     0.000609981     0.00061428
000613602       0.000613035     0.000618617     0.000629607     0.000636534     0.154483        0.154622        0.153261        0.152376        0.15235         0.152379
152379          0.152391        0.152392        0.152587        0.153478        0.151665        0.0835435       0.0830878       0.075567        0.05692         0.0535017
0565913         0.00621525      0.00558922      0.00477915      0.00410005      0.00369045      0.00061387      0.000613365     0.000613962     0.000614047     0.00061404
Ln 1, Col 1    100%    Unix (LF)    UTF-8
```

10.3.2 Temperature

```
Temperature profile.d6o - Notepad
File  Edit  Format  View  Help
D6OARLZ! 007.000
TYPE          = FIELD
PROJECT_FILE  = SF_MF-I0-B0.d6p
CREATED       = Thu Dec 19 11:28:32 2024
GEO_FILE      = SF_MF-I0-B0_592561214.g6a
GEO_FILE_HASH = 574338514
QUANTITY      = Temperature
QUANTITY_KW   = Temperature
SPACE_TYPE    = SINGLE
TIME_TYPE     = NONE
VALUE_UNIT    = C
TIME_UNIT     = h
START_YEAR    = 2020
INDICES       = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
9 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321
5 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577
1 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833

0               20              20              20              20              20              20              20              20              20              20
                20              20              20              20              20              20              20              20              20              20
                20              20              20              20              20              20              20              20              20              20
                20              20              20              20              20              20              20              20              20              20
                20              20              20              20              20              20              20              20              20              20
                20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
        20              20              20              20              20              20              20              20              20              20              20
20              20              20              20              20              20              20              20              20              20              20
20              20              20              20              20              20              20              20              20              20              20
24              2.00233         1.9996          2.16117         1.99379         2.15828         2.79369         1.98433         2.15213         2.79015         4.6299
.4859           12.5774         12.6147         1.93049         2.0891          2.7064          4.49338         6.89072         9.01858         10.9349         12.0914
.6866           12.0403         12.2465         12.44           12.6037         12.9427         13.6187         14.274          14.5773         1.93049         2.07422
09015           2.70513         4.48231         6.82622         8.81849         10.439          11.3308         11.7724         11.986          12.1013         12.2529
7082            4.49338         6.88537         8.97514         10.6646         11.3308         11.4476         11.5006         11.6424         12.0057         12.6447
.0314           15.5554         16.3313         17.0889         17.9226         18.7152         1.99379         2.1421          2.75271         4.53091         6.90614
10091           10.7972         11.7129         11.9959         12.0526         12.0273         12.0057         12.3759         13.6506         15.06           15.7139
.8741           16.1663         16.0868         15.7716         15.6446         16.1147         16.7193         17.2451         17.8374         18.6939         18.9304
.6497           18.954          19.0467         19.142          19.2375         19.2996         9.30578         11.1613         12.1707         12.4599         12.4941
Ln 1, Col 1    100%    Unix (LF)    UTF-8
```

10.4 Use of Python in Delphin simulation

To configure and run the sensitivity analysis and calculate wood decay performance a Python script is written. This script controls the Delphin simulation software and imports, processes and exports generated data through various scripts.

This paragraph provides additional information how this combination between Delphin and Python is made in different sub-paragraphs: the general control of Delphin software with Python, changing input material and simulation files, running the Delphin executable and the processing of generated data in output files.

10.4.1 Control of Delphin software with python

Delphin software is created by the Institut für BauKlimatik (IBK), part of the Technische Universität Dresden. Besides the simple use of Delphin software they have integrated more advanced techniques to use the software, for example the use of programming scripts to change and run the simulation software.

The Delphin Solver can be used to run the program with a simple command line executable, this command line consists of plain ASCII (American Standard Code for Information Interchange) text and XML (eXtensible Markup Language) files that can be used for automatization of simulation variant studies. For example a sensitivity analysis. The IBK provides simple tutorials (Bauklimatic Dresden, z.d.) for reading/modifying/writing of Delphin project files, the regeneration/adjustment of the discretization grid, modifying material properties and climate data, running the simulation in parallel, reading simulation output files and the visualisation of generated data. These tutorials are used to create more advanced scripts in this master thesis project.

Additional to the tutorials there are control scripts provided. Unfortunately these where written in a non-compatible Python version to the current one and in order to use, change and run the scripts in upcoming years these needed to be rewritten. These Delphin control scripts are:

- __init__.py
- Delphin6GeoFile.py
- Delphin6GeoFileManager.py
- Delphin6OutputFile.py
- JobRunner.py

The rewritten scripts can be found in the attached files.

The __init__.py file gives an overview of the provided classes, in the Delphin6GeoFile.py the geometry file is created, the Delphin6Outputfile.py reads the Delphin output files in ASCII format, the Delphin6GeoFileManger.py stores all geometry files referenced to the output files so that it is not necessary to read the same geometry file multiple times and the JobRunner.py is used to run multiple jobs in parallel.

10.4.2 Material files

The paper by Vogelsang (2013) gives more insights into the use and change of material files in Delphin. These files are used to store material information needed for hygrothermal simulations, such as storing simple material parameters and advanced functions for heat and moisture transport and storage models.

The ASCII format file is designed in such a way that simple text editors, like the Notepad, can be used to create and change these inputs. For the file extension, m6 is used as standard (Example: materialfile1.m6). UTF8 is used as encoding string, commonly used as encoder to transfer text files from all languages, symbols, etc. to computational text.

Material files are structured in these sections:

- [IDENTIFICATION]
- [STORAGE_BASE_PARAMETERS]
- [TRANSPORT_BASE_PARAMETERS]
- [MECHANICAL_BASE_PARAMETERS]
- [MOISTURE_STORAGE]
- [MOISTURE_TRANSPORT]
- [HEAT_TRANSPORT]
- [AIR_TRANSPORT]

If an anisotropic material needs to be simulated the specific material properties need to be specified to material direction. In the material file they should be stored in the correct format, the following sections can be used. Where the appendix U (standard, non-visible in sections above) is radial, appendix V is tangential and W is longitudinal.

- [MOISTURE_TRANSPORT_V]
- [MOISTURE_TRANSPORT_W]
- [HEAT_TRANSPORT_V]
- [HEAT_TRANSPORT_W]
- [AIR_TRANSPORT_V]
- [AIR_TRANSPORT_W]

Every section of the material file holds specific parts of information needed to run the simulations. For this report the storage base parameters, transport base parameters, moisture storage and moisture transport are further elaborated.

10.4.2.1 Material storage and transport base parameters

Within a material file the following properties can be varied for both storage and transport base parameters, including the parameter description, optional file format and unit within Delphin:

Storage base parameters

RHO (ρ) [kg/m^3]: Density of dry material including pores (bulk density).

CE (c_p) [J/kgK]: Specific heat capacity at constat pressure. Energy needed to increase the temperature of 1 kg of dry material by 1 K.

THETA_POR (θ_{por}) [m^3/m^3]: Open porosity, does not include closed pores. The open porosity must be greater or equal then the effective saturation moisture content.

THETA_EFF (θ_{cap}) [m^3/m^3]: Effective saturation moisture content. The effective saturation moisture content is a long term {maximum} saturation. It must be greater or equal than the capillary saturation moisture content.

THETA_CAP (θ_{por}) [m^3/m^3]: Capillary saturation moisture content. Mean moisture content of a sample obtained in the water uptake experiment at the end of the first water uptake period. The capillary saturation moisture content is a short term saturation. It is determined when the water front reaches the top of the specimen.

THETA_80 (θ_{80}) [m^3/m^3]: Hygroscopic moisture content at RH=80% obtained in an hygroscopic adsorption experiment. It should match the (ad)sorption isotherm at RH=80%.

THETA_LIM_HYG ($\theta_{lim,hyg}$) [m^3/m^3]: Limitation hygroscopic moisture content for those materials that must not get wet. May be used as indicator for materials that shall be subjected to hygroscopic moisture loads only.

Transport base parameters

LAMBDA (λ_{dry}) [W/mK]: Thermal conductivity for the dry material.

LAMBDA_DESIGN (λ_{design}) [W/mK]: Design value for thermal conductivity for material. These values are given from material producer and not used for calculation.

AW (A_w) [$kg/m^2\sqrt{s}$]: Water uptake coefficient.

MEW (μ) [-]: Water vapour diffusion resistance factor (dry cup).

KLEFF ($K_{l,eff}$) [s]: Liquid water conductivity at effective saturation.

DLEFF ($D_{l,eff}$) [s]: Liquid water diffusivity at effective saturation.

KG ($K_{g,dry}$) [s]: Air permeability of dry material.

SD (s_d) [m]: Equivalent air layer thickness. Relates the vapour resistance of a material (foils and coatings only) to the vapour resistance of air.

10.4.2.2 Material storage and transport functions

Storage and transport of moisture is integrated in Delphin with the use of functions. Different options are available.

Storage functions

A few functions can be specified for the storage of moisture. These functions give the moisture retention characteristics, describing the moisture content in a material versus the potential of moisture. Possible functions:

THETA_l ($\theta_l(pC)$) [m^3/m^3]: Moisture retention curve. Both desorption and adsorption.

PC ($P_C(\theta_l)$) [$\log_{10}(Pa)$]: Reversed moisture retention curve. Both desorption and adsorption.

THETA_l ($\theta_l(\varphi)$) [m^3/m^3]: Sorption isotherm. Both desorption and adsorption

Function are given in x and y values, creating datapoints for the functions. Values on x-axis should be strictly increasingly monotonic.

Transport functions

Two types of transport are used in Delphin, being: capillary liquid moisture transport and water vapour diffusion. For capillary transport two models can be used: liquid conductivity model and the moisture diffusivity model, respectively driven by liquid pressure gradient and moisture content gradient.

For water vapour transport the driving force, for both models, is the gradient of vapour pressure. Similarly to the first transport type, two models are available: One including the vapour diffusion flux density and one using the water vapour diffusion resistance factor.

Possible functions:

lgKl ($lgK_{l(}\theta_{l)}$) [$\log_{10}(s)$]: Capillary conductivity depending on moisture content.

lgDl ($lgD_{l(}\theta_{l)}$) [$\log_{10}(m^2/s)$]: Liquid water diffusivity depending on moisture content

lgKv ($lgK_{v(}\theta_{l)}$) [$\log_{10}(s)$]: Water vapour permeability depending on moisture content.

mew ($\mu(\varphi)$) [-]: Water vapour diffusion resistance factor depending on relative humidity.

10.4.3 Simulation files

Besides the material file, Delphin uses simulation files (.d6p) where information is stored to run the software. Similarly in the ASCII format with UTF8 encryption. The files use the following sections:

- [PROJECT INFO]
- [DIRECTORY PLACEHOLDERS]
- [INIT]
- [MATERIALS]
- [DISCRETIZATION]
- [CONDITIONS]
- [OUTPUTS]
- [ASSIGNMENTS]
- [ASSESSMENTS]

These files include settings for duration, grid layout, material types, indoor and outdoor conditions, types of output files and the assignment of materials to a certain grid surfaces.

Additionally, with the python script, the adjusted material files can be replaced and the simulation file is used as a job file in the Delphin executable.

10.4.4 Placeholders in files

Parameters can be changed quite simply within the material and simulation files by using placeholders that can be filled with the Python script. Example with the water uptake coefficient in tangential direction:

```
[TRANSPORT_BASE_PARAMETERS]
        LAMBDA_U        = 0.1557 W/mK
        LAMBDA_V        = 0.1863 W/mK
        AW_U            = 0.0088 kg/m2s05
        AW_V            = ${PH_AW} kg/m2s05
        MEW_U           = 648.02 -
        MEW_V           = 729.05 -
```

Here, the placeholder ${PW_AW} is used as the location of the new value. All placeholders in the templates should be filled before simulation.

10.4.5 Running Delphin executable

With Python a job command line is created that is used by the Delphin Solver (DelphinSolver.exe). This solver is called with the script and runs the simulation file, with corresponding settings and material properties.

For a SA multiply jobs commands are created that are stored in a job command list. Python can run the jobs one after another or the JobRunner.py can be used to run jobs in parallel, depending on computing power of the device used. These jobs are executed in the computer background, without opening the Delphin software itself.

File name formats are used in order to combine the correct material and simulation files. One template file with placeholders is created for every file type. Being:

- [MF_I1_B1]: Material file, insulation material 1, brick material 1
- [SF_LOC_MF1]: Simulation file, location, material file 1

Depending on the differences in settings these file name formats can be changed to create a logical file layout.

10.4.6 Output file

Result files (.d6o) are written within result folders, with the corresponding simulation file name. In order for simple and correct processing of generated data.

Output files store the generated data in large rows. Where every row contains all calculated values for every geometry points (from the discretization grid). Every row represents one output frequency, for example every day. An example: a temperature result file for 3 years and 40x50 grid points would have 1095 (3x365days) rows and every row 2000 (40x50grid points) individual values. These values are directly linked to coordinates of the geometry and can therefore be used to generate 2D visualisations.

10.4.7 Delphin PostProc

PostProc 2.5 is the visualisation program of the Delphin simulation software. The .d6o result files are read and processed by the Delphin6GeoFileManager.py and can be visualised in the software. Settings can be adjusted in the program itself. Not only Delphin generated data, but also, the with python calculated wood decay performance, result files can be visualised in the program to make the results more clear.

10.5 Python script elaboration

The use of the Python script is an important part of the total research. This paragraph focuses on the script itself, explaining the way it works and how to use it if another sensitivity analysis or wood decay calculation need to be made.

For every python script part, there is a separate subparagraph. These five scripts are used in following order. Starting with the creation of the Sobol samples and ending with the visualisation of dose and Sobol indices.

Additionally to the five scripts made for this research, there are an extra five scripts to control the Delphin simulation software. These scripts have been rewritten to Python version 3, the extended explanation of the workings of these scripts can be found in the last subparagraph.

Note: due to the fact that no initial knowledge on programming / python language and writing script was present when doing the research, the script probably could have been written in a more correct and efficient programming language. All full python scrips can be found in the appendix.

All Python scripts are based on two parts: setup (# *** SETUP ***) and main (# *** MAIN ***). Within these two part, multiple steps are defined. This separation is made to give more structure to the script. With import functions, parameters and definitions being defined in setup and the programming part of the specific script located in the second main part.

10.5.1 Part 1: Sobol sampler

The first script uses the 'pandas' and 'SALib' library to create a certain number of simulations that represent the total number of possible simulations. This use of the SALib (Sensitivity Analysis Library in python) is needed due to the fact that an enormous number of simulations would be needed if the variations would be changed one at the time. SALib contains multiple types of sensitivity analysis, Sobol is one of them. (SALib, n.d.) With the sampler a significant decrease in simulations is made while still achieving a representative result in comparison when all possible simulations would be executed. Python is used to automate this creation of Sobol samples and the execution of simulation jobs.

Defining the problem definition is step one. The number of variables 'num_vars' is set to 5 and 'names' and 'bounds' are the parameters that are varied, with corresponding range.

The prefix 'I' or 'B' is given to the parameters for insulation and brick.

{problem}	Problem definition
{num_vars}	Number of parameters
{names}	Names of parameters
{bounds}	Range of parameters

The sobol sampler uses the 'problem definition', set 'n-value' and 'calc_second_order' set to true to generate the values. The script places these values in a dataframe that is saved as .csv file.

{N:}	Number of n values
{calc_second_order}	Setting Second Order
{.to_csv(*path*, index)}	Create and save .csv file

Due to the extensive number of simulations needed, the Sobol value data frame is generated once and saved. This makes sure that when the generated simulated jobs need to be restarted, the Sobol values are not generated again. This would result in an incorrect collection of simulations, for example with the first 100 simulation using sample set x and the second 100 simulation using sample set y. The one sample set can be called multiple times, and the simulation jobs executed in parts.

10.5.2 Part 2a: Create Jobs

The two material templates and the simulation template need to be filled with the Sobol values to create all the jobs that need to be executed. The data frame from part one is imported, and two new list for the names of the material files are created. These list of names are in a later stage used to import the correct material file in the simulation files.

A loop runs through all possible material variants, similar to the length of the Sobol data frame. Number of simulations = number of material variations. Every loop imports the next row of the data frame, with i number of loops giving the correct values for every parameter (five in total) for that specific material. Written as:

{i_lambda = sobol_samples.iloc[i, 0]}
{i_mew = sobol_samples.iloc[i, 1]}
…
{b_nvalue = sobol_samples.iloc[i, 4]}

Additional to the change of the parameters of a material also the storage function of every material needs to be calculated. The definition {van_genuchten_theta} calculates every moisture content at a certain capillary pressure, this capillary pressure ranges between 0 and 12 $[log(Pa)]$ in 50 steps.

This number of steps is found iterating to find the best possible curve without to many calculations and data points. The moisture storage function is created for both insulation and brick for every simulation. Written as:

{i_mc_thpc_x}	x values for moisture storage function of insulation
…	
{b_mc_thpc_y}	y values for moisture storage function of brick

Parameter {theta_80} needs to be calculated from every moisture storage function for every single material file, otherwise Delphin returns an error. The same definition is used and capillary pressure is calculated at 80% with the Kelvin relation.

{rho_w}	1000	$[kg/m^3]$
{r_v}	462	$[J/kgK]$
{T}	298	$[K]$
{phi}	0,80	[-]

{p_c = abs(rho_w * r_v * T * np.log(phi))}

All Sobol values, moisture storage functions and parameter values as a result from the storage functions are known and can be replaced in the template files. For both materials there are eight placeholders in the template files. These placeholders are replace with:

{.replace}	Replace placeholder with value

For the simulation files a new empty name list is created, again to use later in the scripts. The placeholders in the template file are replace in a similar way to the material files. Additionally a link to the new material database (own folder instead of Delphin database) is made by replacing {${REP_MATERIAL_DATABASE}}. From this database the new material files previously created are imported to the corresponding simulation file.

To ensure the correct combining of files, all files are saved with the same name as the row of the data frame they are in. This means that every *i* number of the simulations has the same *i* number material files and simulation file. Example:

Row 46 of data frame:
{MF-Brick-46.m6}
{MF-Insulation-46.m6}
{SF_MF-I46_MF_B46.d6p}

The final executable job is created by importing the simulation file and adding it to a job list. These jobs are stored and used in the next part to be executed with the Delphin solver.

10.5.3 Part 2b: Run Jobs

Delphin is created in a way that the Delphin executable solver {DelphinSolver.exe} can run simulation jobs in the background of the computer. The second script uses this possibility to automatically run al simulations one after another. Two possible scripts are written, one that solves the simulation one at the time and a one that can run multiple jobs.

When running more jobs at the same time, the script uses the multi-core CPU of the computer. Laptop and computer with modern processors have multiple cores that can all do one job at the same time. The laptop used has an Intel® Core™ i7 processor that has four cores, and thus can simulate four jobs at the same time. Minimizing simulations of around 10 to 6/7 minutes. An additional advantage is the fact that simulation errors can occur and that one of the processor cores stops executing jobs. The remaining cores keep running until four error are found, but this is highly unlikely.

One of the five extra scripts is the {JobRunner.py}. Within the script the exact jobs that need to be executed can be defined, this specific line of script runs jobs ranging from 1185 to 1536.

{JobRunner.run(job[1185: 1536]}

10.5.4 Part 4: Calculating dose

Python script three calculates the dose for every specific coordinate of the simulated geometry. This dose calculation uses variables and formulas from literature are integrated in the script. With ‘aoc’, ‘aod’ and ‘wood_coordinates’ the number of coordinates of the geometry, number of days of the simulation and indices of coordinates with wood as materials respectively, are defined.

The third script simultaneously calculates al the output variables for the multiple sensitivity analyses. Using the following lists for calculated values:

{first_sa_list = []}
{second_sa_list = []}
{third_sa_list = []}
{fourth_sa_list = []}

All simulation files are imported in step, to calculate all the dose values for all simulations. A for loop is used to run the dose calculation for every simulation. Therefore, two lists: {moisture_content = []} and {temperature = []} are created and filled with the moisture and temperature values of that specific

simulation. Moisture output values (average moisture content in beam and maximum moisture content in the beam) are calculated with this data and stored in the designated output variable list.

In step 5 of the third script. A empty dose list is made with the same number of values as coordinates. For every coordinate the cumulative dose value is calculated. Subsequently a for loop runs through all the days of the simulation. Evaluating if the moisture and temperature component are within boundary limits and calculating the dose value where needed with:

{Du_result = evar * Du**5 - fvar * Du**4 + gvar * Du**3 - hvar * Du**2 + ivar * Du – jvar}

{Dt_result = kvar * Dt**4 + lvar * Dt**3 - mvar * Dt**2 + nvar * Dt}

{dose = (avar * float(Dt_result) + float(Du_result)) * (avar + 1)**-1}

If calculated, the dose is added to the dose of the previous day, resulting in a cumulative dose at the end of all simulated days. The output variables (average dose value and maximum dose value) are added to the output variable lists.

To finish the calculation of values that are analysed by Sobol. A dataframe with the four output variables is created and save to a .csv file:

{SA_simulation_results.csv}

10.5.5 Part 5: Sobol analysis

The result dataframe is imported and the problem definition (same as the problem definition in the sobol sampler) is defined. Variables are named:

{variables = ['I_TC', 'I_WVDRF', 'I_NV' ,'B_WVDRF' ,'B_NV']}

With {sobol_indices_sa1 = sobol.analyze (problem, result_data_sa1)} the Sobol analysis is executed on the output variables. Figures of the first-, second and total order indices for every sensitivity analysis are plotted and exported.

10.5.6 Rewritten control scripts

To control the Delphin software, five scripts made by the TU Dresden are adapted to be used in this study. These scripts are provide online to be used in these kinds of situations. Unfortunately all the scripts needed to be updated to Python 3. With the help of collueages at ABT, the five scripts are adapted to work for this study. These five scripts are:

{__init__.py}
{Delphin6GeoFile.py}
{Delphin6GeoFileManager.py}
{Delphin6OutputFile.py}
{JobRunner.py}

10.6 File / Data handling

For the scripts and programs to work specific programs, folder layout and naming system is used. This makes sure the data can be found and that the correct values are used in the calculations. Within the python script, there are multiple paths that lead to the correct folders. For example the template, variants and results folder but also the python scripts that are used in the background to run the Delphin executable.

For this research the following programs are used, with corresponding versions:

- Python	v3.13	Programming language
- PyCharm	v24.2.1	Python IDE
- Delphin	v6.1	Simulation software
- PostProc	v2.5	Visualisation software
- Notepad		Text editor

The project folder is saved on a certain directory locations. This directory is set as Python Content Root, it uses this root to find the other files and is set as a base from where the next file paths are written,

It is not needed to write down the entire path, just from this base directory. Additionally the full path to the Delphin software within the program files is added to the Content Root.

This is used to find the .exe files and climate/material files that can be found in the standard Delphin database.

- {*full project path*} Project location
- {*full program file path*} Windows program files

Within the directory where the project is saved this specific folder / file layout is used:

- {.idea} Folder
- {.venv} Folder
- {__pycache__} Folder
- {SA_Templates} Folder
    - {Material} SubFolder
    - {Simulation} SubFolder
- {SA_Variants} Folder
    - {Material} SubFolder
    - {Simulation} SubFolder
- {SA_Results} Folder
    - {Material} SubFolder
- {__init__.py} File
- {Delphin6GeoFile.py} File
- {Delphin6GeoFileManager.py} File
- {Delphin6OutputFile.py} File
- {JobRunner.py} File
- {Part 1_Sobol Sampler.py} File
- {Part 2_Create and Run Jobs.py} File
- {Part 3_Sensitivity anlaysis.py} File
- {Part 4_Sobol analysis.py} File
- {Part 5_Wood decay assessment.py} File
- {File and Folder check.py} File

10.7 Python scripts

10.7.1 Part 1_Sobol sampler.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
# Part (1/5) for Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Creation of Sobol samples dataframe
#   Run once and use part 2 to create and run the simulations

# *** SETUP ***
# 1. IMPORT
import pandas as pd
from SALib.sample import sobol

# *** MAIN ***
# 1. SETUP SENSITIVITY ANALYSIS PROBLEM DEFINITION
    # * Change *: Optionally change parameters and/or bounds of SA

problem = {
    'num_vars': 5,                                              #AMOUNT OF PARAMETERS
    'names' : ['I_Thermal Conductivity',                        #I_LAMBDA
               'I_Water Vapor Diffusion Resistance Factor',     #I_MEW
               'I_NValue',                                      #NEW_STORAGE_FUNCTIONS:
                                                                    #I_THETA_CAP
                                                                    #I_THETA_80
               'B_Vapor Diffusion Resistance Factor',           #B_MEW
               'B_NValue'],                                     #NEW_STORAGE_FUNCTIONS:
                                                                    #B_THETA_CAP
                                                                    #B_THETA_80
    'bounds' : [[0.01875, 0.075],                               #I_LAMBDA
                [1, 100],                                       #I_MEW
                [1.5, 2.75],                                    #I_N-VALUE
                [5, 50],                                        #B_MEW
                [1.2, 2]]                                       #B_N-VALUE
}

# 2. CREATE SOBOL SAMPLE COLLECTION and CREATE CSV FILE
    # * Change *: n value of accuracy of sensitivity analysis
sobol_samples_values = sobol.sample(problem, 128, calc_second_order=True)
sobol_samples_for_csv = pd.DataFrame(sobol_samples_values)
print(sobol_samples_for_csv)
    # * Change *: Path from content root to dataframe csv storage location
sobol_samples_for_csv.to_csv('SA_Variants/SA_Sobol Samples v3.csv', index=False)
print(r'Sobol sample dataframe is created and saved to directory')
```

10.7.2 Part 2_Create and Run Jobs.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
# Part (2/5) for Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Create material and simulation files and run simulation jobs
#   Import Sobol samples from dataframe (part 1) and creates data for Sobol analysis (part 3)

# *** SETUP ***
#1. IMPORT
import os
import csv
import time
import subprocess
import pandas as pd
import numpy as np
from JobRunner import JobRunner

#2. DEFINITIONS
    #READ / WRITE FILE
def read_file(fname):
    try:
        fobj = open(fname, 'r', encoding='utf-8')
        content = fobj.read()
        fobj.close()
        del fobj  # release file handle
        return content
    except IOError as e:
        print(str(e))
        raise RuntimeError("Error reading/opening file '{}'".format(fname))

def write_file(fname, contents):
    try:
        fobj = open(fname, 'w', encoding='utf-8')
        fobj.write(contents)
        fobj.close()
        del fobj  # release file handle
    except IOError as e:
        print(str(e))
        raise RuntimeError("Error opening file '{}' for writing".format(fname))

    #VAN GENUCHTEN MOISTURE STORAGE FUNCTION + inverse
def van_genuchten_theta(pa, theta_r, theta_s, alpha, n):
    theta = theta_r + (theta_s - theta_r) / (1 + (alpha * pa) ** n) ** (1 - 1 / n)
    return theta

def van genuchten pa(theta, theta r, theta s, alpha, n):
    x = (theta_s - theta_r) / (theta - theta_r)
    pa = ((x ** (n / (n - 1)) - 1) ** (1 / n)) / alpha
    return pa

#3. EXECUTABLES, DIRECTORY AND FILE LOCATION
    # * Change *: absolute path to IBK Delphin Solver in program files
DELPHIN_EXECUTABLE = r'C:\Program Files\IBK\Delphin 6.1\DelphinSolver.exe'

    # * Change *: make sure to put in both subfolders in project folder
TEMPLATE_SUBDIR = "SA_Templates"    # Subdirectory templates
VARIANTS_SUBDIR = "SA_Variants"     # Subdirectory variants

    # * Change *: path to insulation and brick material template and simulation template
project_template = read_file(TEMPLATE_SUBDIR + '/Simulation/simulation template v5.d6p')
material_template_brick = read_file(TEMPLATE_SUBDIR + '/Material/material template -
brick.m6')
material_template_insulation = read_file(TEMPLATE_SUBDIR + '/Material/material template -
insulation.m6')

#4. PARAMETERS
    #INSULATION
i_theta_r = 0           # Residual moisture content
i_theta_s = 0.929       # Saturated moisture content
i_alpha = 0.000002      # Alpha parameter (1/Pa)

    #BRICK
b_theta_r = 0           # Residual moisture content
```

```
b_theta_s = 0.324       # Saturated moisture content
b_alpha = 0.00002       # Alpha parameter (1/Pa)

    #CLIMATE FILES      # Eventually not used in Sensitivity analysis (optional)
climate_files = [
    ("DE-05-TRY-2010__Essen__Jahr_00000K0_00152m.c6b", "ESS"),
    ("DE-01-TRY-2010__Bremerhaven__Jahr_00000K0_00007m.c6b", "BRE"),
    ("DE-04-TRY-2010__Potsdam__Jahr_00000K0_00081m.c6b", "POT"),
    ("DE-15-TRY-2010__Garmisch_Partenkirchen__Jahr_00000K0_00719m.c6b", "GAR")
]

# *** MAIN ***
# 1. IMPORT SOBOL VARIANT DATA FRAME
    # * Change *: path to content root for used Sobol dataframe
sobol_samples = pd.read_csv('SA_Variants/SA_Sobol Samples v3.csv')
print(sobol samples)

# 2. CREATE INSULATION AND BRICK FILE LIST
list_material_files_insulation = []
list_material_files_brick = []

# 3. LOOP THROUGH MATERIAL VARIANTS
    # * Change *: amount of rows in dataframe (amount of simulations)
for i in range(1536):
    mc_list_brick = []
    mc_list_insulation = []

    # 3A. IMPORT FROM SOBOL DATAFRAME
    i_lambda = sobol_samples.iloc[i, 0]
    i_mew = sobol_samples.iloc[i, 1]
    i_nvalue = sobol_samples.iloc[i, 2]
    b_mew = sobol_samples.iloc[i, 3]
    b_nvalue = sobol_samples.iloc[i, 4]

    # 3B. CALCULATE STORAGE FUNCTIONS
    pa_interval = np.logspace(0, 12, 50)
    #print(pa_interval)
    pa_interval_log = np.log10(pa_interval)

    for j in range(len(pa_interval)):
        moisture_insulation = float(van_genuchten_theta(pa_interval[j], i_theta_r, i_theta_s,
i_alpha, i_nvalue))
        mc_list_insulation.append(moisture_insulation)

    i_mc_thpc_x = " ".join([str(x) for x in pa_interval_log])
    i_mc_thpc_y = " ".join([str(x) for x in mc_list_insulation])
    i_mc_pcth_x = " ".join([str(x) for x in mc_list_insulation[::-1]])
    i_mc_pcth_y = " ".join([str(x) for x in pa_interval_log[::-1]])

    for k in range(len(pa_interval)):
        moisture_brick = float(van_genuchten_theta(pa_interval[k], b_theta_r, b_theta_s,
b_alpha, b_nvalue))
        mc_list_brick.append(moisture_brick)

    b_mc_thpc_x = " ".join([str(x) for x in pa_interval_log])
    b_mc_thpc_y = " ".join([str(x) for x in mc_list_brick])
    b_mc_pcth_x = " ".join([str(x) for x in mc_list_brick[::-1]])
    b_mc_pcth_y = " ".join([str(x) for x in pa_interval_log[::-1]])

    # 3C. CALCULATE THETA_CAP AND THETA_80
    rho_w = 1000    # Density of water (kg/m^3)
    r_v = 462       # Gas constant of water vapor (J/kgK)
    T = 298         # Absolute temperature (Kelvin, example value: 298 K for ~25°C)
    phi = 0.80      # RH80%

    p_c = abs(rho_w * r_v * T * np.log(phi))

    # Output result
    i_theta_cap = 0.60992
    i_theta_80 = van_genuchten_theta(p_c, i_theta_r, i_theta_s, i_alpha, i_nvalue)

    b_theta_cap = 0.24376
    b_theta_80 = van_genuchten_theta(p_c, b_theta_r, b_theta_s, b_alpha, b_nvalue)

    # * Change *: If you change parameters, add placeholders and replace lines in script
    # 3D. REPLACE MATERIAL FILE PLACEHOLDERS
    variant_insulation = material_template_insulation.replace('${I_LAMBDA}', str(i_lambda))
```

```
    variant_insulation = variant_insulation.replace('${I_MEW}', str(i_mew))
    variant_insulation = variant_insulation.replace('${I_THETA_CAP}', str(i_theta_cap))
    variant_insulation = variant_insulation.replace('${I_THETA_80}', str(i_theta_80))

    variant_insulation = variant_insulation.replace('${MS_TH/PC_X}', str(i_mc_thpc_x))
    variant_insulation = variant_insulation.replace('${MS_TH/PC_Y}', str(i_mc_thpc_y))
    variant_insulation = variant_insulation.replace('${MS_PC/TH_X}', str(i_mc_pcth_x))
    variant_insulation = variant_insulation.replace('${MS_PC/TH_Y}', str(i_mc_pcth_y))

    material_file_insulation = "MF-I{}.m6".format(i)
    write_file(VARIANTS_SUBDIR + "/Material/" + material_file_insulation, variant_insulation)
    list_material_files_insulation.append(material_file_insulation)

    variant_brick = material_template_brick.replace('${B_MEW}', str(b_mew))
    variant_brick = variant_brick.replace('${B_THETA_CAP}', str(b_theta_cap))
    variant_brick = variant_brick.replace('${B_THETA_80}', str(b_theta_80))

    variant_brick = variant_brick.replace('${MS_TH/PC_X}', str(b_mc_thpc_x))
    variant_brick = variant_brick.replace('${MS_TH/PC_Y}', str(b_mc_thpc_y))
    variant_brick = variant_brick.replace('${MS_PC/TH_X}', str(b_mc_pcth_x))
    variant_brick = variant_brick.replace('${MS_PC/TH_Y}', str(b_mc_pcth_y))

    material_file_brick = "MF-B{}.m6".format(i)
    write_file(VARIANTS_SUBDIR + "/Material/" + material_file_brick, variant_brick)
    list_material_files_brick.append(material_file_brick)

print(list_material_files_insulation)
print(list_material_files_brick)

print(f"Creation of material files done")

# 4. CREATE LIST OF SIMULATION FILES
#create empty list with all simulation name and job files
list_simulation_files = []
jobs = []

#create simulation files for all different sobol variants
# * Change *: The first for loop can be used for a study with different locations
for j in range(1): #location options
    # * Change *: Amount of rows in the Sobol dataframe
    for l in range (1536):
        #replace placeholders in template with material and climate data and specify database
directory
        #variant_2 = project_template.replace('${REP_LOCATION}',
'{}'.format(climate_files[j][0]))
        variant_2 = project_template.replace('${REP_MATERIAL_INSULATION}',
'{}'.format(list_material_files_insulation[l]))
        variant_2 = variant_2.replace('${REP_MATERIAL_BRICK}',
'{}'.format(list_material_files_brick[l]))
        variant_2 = variant_2.replace('${REP_MATERIAL_DATABASE}', r'C:\Users\r.trip\OneDrive -
Oosterhoff Group\Documents\Graduation_Rik Trip_ABT\02. Master Thesis\3. Simulaties\02. Delphin
Analysis on Hygrothermal Performance\SA_Variants\Material')
        #variant_2 = variant_2.replace('${REP_CLIMATE_DATABASE}', r'PATH')

        #write simulation name and add to list
        simulation_file = "SF_MF-I{}-B{}.d6p".format(l, l)
        write_file(VARIANTS_SUBDIR + "/Simulation/" + simulation_file, variant_2)
        list_simulation_files.append(simulation_file[:-4])

        #write simulation job and add to list
        if not os.path.exists(simulation_file[:-4] + "/var/restart.bin"):
            jobs.append([DELPHIN_EXECUTABLE, '-x', '--verbosity-level=0', '-p=2',
'SA_Variants/Simulation/'+ simulation_file])


print(list simulation files)
print(f"Creation of simulation files done")

df_job_list = pd.DataFrame(jobs)
# * Change *: Path to content root for jobs files
df_job_list.to_csv('SA_Variants/SA_job files 2.csv', index = False)
#print(jobs)

# * Change *: Path to content root for name simulation files
df_list_simulation_files = pd.DataFrame(list_simulation_files)
df_list_simulation_files.to_csv('SA_Variants/SA_simulation_files.csv', index = False)
```

```
print("\nStarting on Simulation job list:")

# 5. SIMULATE JOBS
# * Optional*: One job at the time:
"""
for job in jobs:
    try:
        #note start time
        start time = time.time()

        #start solver
        solverProcess = subprocess.Popen(job)
        solverProcess.wait()

        #note end time and calculate duration
        end_time = time.time()
        job_duration = end_time - start_time
        print(f"Job {job} of {len(jobs)} done at {time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(end_time))}, in {job_duration:.2f} seconds \n")

    except OSError as e:
        print(str(e))
"""
# * Optional *: Run all the jobs in parallel
# * Change *: The jobrunner number is the amount of cores in your CPU. And can be used to run
multiple simulations at same time
#             Do check the amount of cores and change accordingly
jobRunner = JobRunner(4)         # 4 jobs at the same time
jobRunner.run(jobs[0:1536])      # * Change *: Range of numbers. 100 you run jobs 0 to 99.
                                 #             Stop the file when needed and put in last
unfinished file and the end file.
                                 # * Example *: [56:1536] This will start simulating does job
56 until 1536

print("\n Finished with Jobrunner list")
```

10.7.3 Part 3_Sensitivity Analysis.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
# Part (3/5) for Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Import generated data of simulations (part 2)
#   Calculate dose for wood coordinates
#   Calculate and export sensitivity output variables for analysis (part 4)


# *** SETUP ***
#1. IMPORT
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from SALib.analyze import sobol
from Delphin6GeoFileManager import Delphin6GeoFileManager
from Delphin6OutputFile import Delphin6OutputFile

#2. DEFINITIONS
    #DELPHIN MANAGERS
DelphinGeoFileManager = Delphin6GeoFileManager()
DelphinOutputFile = Delphin6OutputFile()
    # * Change *: path to content root
VARIANTS_SUBDIR = "SA_VARIANTS/Simulation/"

#3. VARIABLES
    #DOSE CALCULATION
avar = 3.2          # temperature weighing factor
evar = 6.75e-10     # variables for moisture content component: Du
fvar = 3.5e-7
gvar = 7.18e-5
hvar = 7.22e-3
ivar = 0.34
jvar = 4.98
kvar = -1.8e-6      #variables for temperature component: Dt
lvar = 9.57e-5
mvar = 1.55e-3
nvar = 4.17e-2

    #GEOMETRY INFO
    # * Change *: Add correct number of coordinates and days from the simulation.
aoc = 978                           # number of coordinates
aod = 1096                          # number of days
two_year_range = range(366, 1096)   # range of 2 years (needed for SA)

    # * Change *: Add correct wood coordinates for designated dose calculation
wood_coordinates = [200, 220, 221, 241, 242, 243, 263, 264, 265, 266, 286, 287, 288, 289, 290,
310, 311,
                    312, 313, 314, 315, 335, 336, 337, 338, 339, 340, 341, 361, 362, 363, 364,
365, 366,
                    367, 368, 388, 389, 390, 391, 392, 393, 394, 395, 396, 415, 416, 417, 418,
419, 420,
                    421, 422, 423, 424, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 471,
472, 473,
                    474, 475, 476, 477, 478, 479, 480, 499, 500, 501, 502, 503, 504, 505, 506,
507, 508,
                    526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 552, 553, 554, 555, 556,
557, 558,
                    559, 560, 561, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 601, 602,
603, 604,
                    605, 606, 607, 608, 609, 610, 624, 625, 626, 627, 628, 629, 630, 631, 632,
633, 646,
                    647, 648, 649, 650, 651, 652, 653, 654, 655, 667, 668, 669, 670, 671, 672,
673, 674,
                    675, 676, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 706, 707, 708,
709, 710,
                    711, 712, 713, 714, 715, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734,
744, 745,
                    746, 747, 748, 749, 750, 751, 752, 753, 763, 764, 765, 766, 767, 768, 769,
770, 771,
                    772, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 801, 802, 803, 804,
805, 806,
                    807, 808, 809, 810, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 839,
```

```
840, 841,
                    842, 843, 844, 845, 846, 847, 848, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867,
                    877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 895, 896, 897, 898, 899,
900, 901,
                    902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 921, 922, 923,
924, 925,
                    926, 927, 928, 935, 936, 937, 938, 939, 940, 941, 947, 948, 949, 950, 951,
952, 957,
                    958, 959, 960, 961, 965, 966, 967, 968, 971, 972, 973, 975, 976, 977]

# *** MAIN ***
#1. IMPORT SIMULATION FILE NAMES
    # * Change *: Add correct content root with SA output variables.
simulation_files = (pd.read_csv('SA_Variants/SA_simulation_files.csv')).iloc[:, 0].tolist()

first_sa_list = []      #empty list for SA result files moisture (max moisture content)
second_sa_list = []     #empty list for SA result files moisture (average moisture content)
third_sa_list = []      #empty list for SA result files dose (max dose content)
fourth_sa_list = []     #empty list for SA result files dose (average dose content)

# 2. IMPORT DATA FOR SPECIFIC SIMULATION FILE
    #FOR LOOP FOR ALL DATA (ALL DAYS / COORDINATES AND MOISTURE / TEMPERATURE) OF ONE
SIMULATION
for name in simulation_files:

    # 2A. IMPORT ALL MOISTURE AND TEMPERATURE DATA FOR EVERY DAY ('AOD')
    moisture_content = [] #WITHIN THIS LIST (FOR EVERY SIMULATION) THERE ARE X (AOD) NUMBER OF
LISTS (FOR EVERY DAY)
    moisture_content_at_index = DelphinOutputFile.read(VARIANTS_SUBDIR + str(name) +
"/results/Moisture content profile.d6o", DelphinGeoFileManager)
    for day in range(aod):
        moisture_content_at_index = DelphinOutputFile.valuesAtIndex(day)
        moisture_content.append(moisture_content_at_index)
        #print(f"At day {day}, the list of moisture = {moisture_content_at_index}")
    #print(moisture_content)

    temperature = [] #WITHIN THIS LIST (FOR EVERY SIMULATION) THERE ARE X (AOD) NUMBER OF
LISTS (FOR EVERY DAY)
    temperature_at_index = DelphinOutputFile.read(VARIANTS_SUBDIR + str(name) +
"/results/Temperature profile.d6o", DelphinGeoFileManager)
    for day in range(aod):
        temperature_at_index = DelphinOutputFile.valuesAtIndex(day)
        temperature.append(temperature_at_index)
        #print(f"At day {day}, the list of temperature = {temperature_at_index}")
    #print(temperature)

    # 3. DELETE ALL DATA FROM NON-WOOD COORDINATES IN IMPORTED MOISTURE AND TEMPERATURE LISTS
    #CREATE MOISTURE CONTENT LIST ONLY FOR WOOD COORDINATES (REST OF COORDINATES = 0)
    moisture_content_wood_coordinates = []
    temperature_wood_coordinates = []

    #iterate over all lists (days) in the imported list
    for sublist in moisture_content:
        # Create a new sublist with 0s, and replace with values from data_list where the index
is in indices_list
        result_sublist = [sublist[i] if i in wood_coordinates else 0 for i in
range(len(sublist))]
        moisture_content_wood_coordinates.append(result_sublist)
    #print(moisture_content_wood_coordinates)
    for sublist in temperature:
        # Create a new sublist with 0s, and replace with values from data_list where the index
is in indices_list
        result_sublist = [sublist[i] if i in wood_coordinates else 0 for i in
range(len(sublist))]
        temperature wood coordinates.append(result sublist)

    # 4. CALCULATE RESULT LISTS FOR SOBOL ANALYSIS 1 AND 2 (MOISTURE CONTENT)
    # list with max and avg moisture content per day
    max_moisture_content_list = [max(sublist) for i, sublist in
enumerate(moisture_content_wood_coordinates) if i in two_year_range and sublist]
    avg moisture content list = [sum(sublist) / len(sublist) for i, sublist in
enumerate(moisture_content_wood_coordinates) if i in two_year_range and sublist]

    # calculate average maximum moisture for all days
    simulation_max_moisture_average = sum(max_moisture_content_list) /
len(max_moisture_content_list)
```

```
    first_sa_list.append(round(simulation_max_moisture_average, 5))
    print(f"The average of all daily maximum moisture contents of simulation {name} is
{simulation_max_moisture_average}")

    # calculate average of average moisture for all days
    simulation_avg_moisture_average = sum(avg_moisture_content_list) /
len(avg_moisture_content_list)
    second_sa_list.append(round(simulation_avg_moisture_average, 5))
    print(f"The average of all daily average moisture contents of simulation {name} is
{simulation avg moisture average}")

    # 5. CREATE EMTPY CUMULATIVE DOSE LIST (ALL DAYS DOSE) FOR EVERY SIMULATION FILE
    cumulative_dose_list = [0] * aoc
    #print(cumulative_dose_list)

    # 6. CYCLE THROUGH INDEXES FOR ALL DAYS
    #WITHIN EVERY SIMULATION FILE WE CYCLE THROUGH ALL THE DAYS (EVERY DAY SPECIFIC LIST OF
THE SIMULATION LIST)
    for day in range(aod): #LATER: PLACE 'AOD'

        moisture_content_at_day = moisture_content_wood_coordinates[day]
        temperature_at_day = temperature_wood_coordinates[day]
        #print(f"At day {day} the amount of coordinates = {len(moisture_content_at_day)}")

        #CHECK IF MOISTURE AND TEMPERATURE FILES HAVE THE SAME SIZE (AND THUS AMOUNT OF
COORDINATES)
        if len(temperature) != len(moisture_content):
            print("Error: Temperature and moisture content lists must have the same length")

        #SAME LENGTH: CYCLE THROUGH ALL COORDINATES TO CALCULATE DOSE AT EVERY SPECIFIC
COORDINATE
        else:

            # 7. CYCLE THROUGH COORDINATES FOR ONE SPECIFIC DAY
            #CREATE EMPY DOSE LIST FOR SPECIFIC DAY
            dose_list = []

            #FIND AND PROCESS COORDINATE SPECIFIC MOISTURE AND TEMPERATURE DATA
            for j in range(aoc):
                Dt = temperature_at_day[j]
                Du = moisture_content_at_day[j] * 100
                #print(f"\nTemperature: {Dt}, Moisture content: {Du} at coördinate {j}")

                #CHECK IF VALUES ARE WITHIN THERE LIMIT AND CALCULATE DOSE + ADD TO DOSE LIST
                if Du >= 25 and 0 <= Dt <= 40:
                    Du_result = evar * Du**5 - fvar * Du**4 + gvar * Du**3 - hvar * Du**2 +
ivar * Du - jvar
                    Dt_result = kvar * Dt**4 + lvar * Dt**3 - mvar * Dt**2 + nvar * Dt
                    #print(f"\nMoisture: {Du}, m component: {Du_result}. Temperature: {Dt}, t
component: {Dt_result} at day {day} for coordinate {j}")

                    dose = (avar * float(Dt_result) + float(Du_result)) * (avar + 1)**-1
                    dose_list.append(round(dose, 1))
                    #print ("The temperature and moisture content values are within there
limits. \n"
                    #       "Calculated: Du_result= " + str(round(Du_result, 3)) + ",
Dt_result= " + str(round(Dt_result, 3)) + ", Dose= " + str(round(dose, 3)))

                # add zero to list if values are not within there limit
                else:
                    dose_list.append(0)
                    #print ("Temperature or moisture content is outside there limit\n")

            # 8. ADD LIST OF COORDINATE DOSES FOR ONE DAY TO CUMULATIVE DOSE LIST
            for k in range(aoc):
                cumulative dose list[k] += dose list[k]
            #print(f"\n Dose List at Index {index_number}: {dose_list}")

    max_dose_simulation = max(cumulative_dose_list)
    third_sa_list.append(round(max_dose_simulation, 5))
    avg_dose_simulation = sum(cumulative_dose_list) / len(cumulative_dose_list)
    fourth sa list.append(round(avg dose simulation, 5))
    print(f"With maximum dose of simulation {name}: {max_dose_simulation}")
    print(f"With average dose of simulation {name}: {avg_dose_simulation}\n")

print(f"Total result list for SA 1 (max moisture content): {first_sa_list}")
print(f"Total result list for SA 2 (avg moisture content): {second_sa_list}")
```

```
print(f"Total result list for SA 3 (max dose content): {third_sa_list}")
print(f"Total result list for SA 4 (avg dose content): {fourth_sa_list}")

all data = {
    'Max Moisture Content': first_sa_list,
    'Avg Moisture Content': second_sa_list,
    'Max Dose Content': third_sa_list,
    'Avg Dose Content': fourth_sa_list
}

simulation_result = pd.DataFrame(all_data)
simulation_result.to_csv('SA_Results/SA_simulation_results V3.csv', index=False)

print("DataFrame exported successfully!")
```

10.7.4 Part 4_Sobol Analysis.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
# Part (4/5) for Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Import generated indices of sensitivity analysis
#   Plot first-, second and total order indices of SA 1, SA 2, SA 3, SA 4

# *** SETUP ***
#1. IMPORT
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from SALib.sample import saltelli
from SALib.analyze import sobol

# *** MAIN ***
#1. IMPORT SOBOL INDICES
    # * Change *: Add path to base directory
base_directory = r"PATH"
    # * Change *: Add path to content root
sobol_values = np.loadtxt(base_directory +"/SA_Results/SA_simulation_results V3.csv",
delimiter=",", skiprows=1)
print(sobol_values)

result_data_sa1 = sobol_values[:, 0]
result_data_sa2 = sobol_values[:, 1]
result_data_sa3 = sobol_values[:, 2]
result_data_sa4 = sobol_values[:, 3]

#2. SET PROBLEM DEFINITON (same as part 1_sobol sampler)
    # * Change *: Change number of variables, parameters and bounds accordingly
problem = {
    'num_vars': 5,
    'names' : ['I_Thermal Conductivity',                          #I_LAMBDA
               'I_Water Vapor Diffusion Resistance Factor',       #I_MEW
               'I_NValue',                                        #NEW_STORAGE_FUNCTIONS
                                                                  #I_THETA_CAP
                                                                  #I_THETA_80
               'B Vapor Diffusion Resistance Factor',             #B_MEW
               'B_NValue'],                                       #NEW_STORAGE_FUNCTIONS
                                                                  #B_THETA_CAP
                                                                  #B_THETA_80
    'bounds' : [[0.01875, 0.075],
                [1, 100],
                [1.5, 2.75],
                [5, 50],
                [1.2, 2]]
}

variables = ['I TC', 'I WVDRF', 'I NV','B WVDRF','B NV']
#print(variables)

#3. SENSITIVITY ANALYSIS 1
sobol_indices_sa1 = sobol.analyze(problem, result_data_sa1)
print(result_data_sa1)
print(sobol_indices_sa1)
print(f"Values for Sensitivity Analysis 1 (max moisture content)")
print("First-order indices:", sobol_indices_sa1["S1"])
print("Second-order indices:", sobol_indices_sa1["S2"])
print("Total-order indices:", sobol_indices_sa1["ST"])
print("Total-order confidence indices:", sobol_indices_sa1["ST_conf"])

first_order_indices_sa1 = sobol_indices_sa1["S1"]
second_order_indices_sa1 = sobol_indices_sa1["S2"]
total_order_indices_sa1 = sobol_indices_sa1["ST"]

# Plotting the first-order and total-order indices
plt.figure(figsize=(12, 6))

# First-order indices bar plot
plt.subplot(1, 2, 1)
```

```
plt.bar(variables, first_order_indices_sa1, color='steelblue', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("First-Order Sobol Indices SA1", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Total-order indices bar plot
plt.subplot(1, 2, 2)
plt.bar(variables, total_order_indices_sa1, color='darkorange', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("Total-Order Sobol Indices SA1", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Plotting the second-order indices as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(second_order_indices_sa1, annot=True, fmt=".3f", cmap="coolwarm",
            cbar_kws={'label': 'Second-Order Index'}, xticklabels=variables,
yticklabels=variables)
plt.title("Second-Order Sobol Indices SA1", fontsize=14)
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.show()


#4. SENSITIVITY ANALYSIS 2
sobol_indices_sa2 = sobol.analyze(problem, result_data_sa2)
print(f"Values for Sensitivity Analysis 2 (avg moisture content)")
print("First-order indices:", sobol_indices_sa2["S1"])
print("Second-order indices:", sobol_indices_sa2["S2"])
print("Total-order indices:", sobol_indices_sa2["ST"])
print("Total-order confidence indices:", sobol_indices_sa2["ST_conf"])

first_order_indices_sa2 = sobol_indices_sa2["S1"]
second_order_indices_sa2 = sobol_indices_sa2["S2"]
total_order_indices_sa2 = sobol_indices_sa2["ST"]

# Plotting the first-order and total-order indices
plt.figure(figsize=(12, 6))

# First-order indices bar plot
plt.subplot(1, 2, 1)
plt.bar(variables, first_order_indices_sa2, color='steelblue', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("First-Order Sobol Indices SA2", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Total-order indices bar plot
plt.subplot(1, 2, 2)
plt.bar(variables, total_order_indices_sa2, color='darkorange', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("Total-Order Sobol Indices SA2", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight layout()
plt.show()

# Plotting the second-order indices as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(second_order_indices_sa2, annot=True, fmt=".3f", cmap="coolwarm",
            cbar kws={'label': 'Second-Order Index'}, xticklabels=variables,
yticklabels=variables)
plt.title("Second-Order Sobol Indices SA2", fontsize=14)
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.show()
```

```
#5. SENSITIVITY ANALYSIS 3
sobol_indices_sa3 = sobol.analyze(problem, result_data_sa3)
print(f"Values for Sensitivity Analysis 3 (max dose content)")
print("First-order indices:", sobol_indices_sa3["S1"])
print("Second-order indices:", sobol_indices_sa3["S2"])
print("Total-order indices:", sobol_indices_sa3["ST"])

first_order_indices_sa3 = sobol_indices_sa3["S1"]
second order indices sa3 = sobol indices sa3["S2"]
total_order_indices_sa3 = sobol_indices_sa3["ST"]

# Plotting the first-order and total-order indices
plt.figure(figsize=(12, 6))

# First-order indices bar plot
plt.subplot(1, 2, 1)
plt.bar(variables, first_order_indices_sa3, color='steelblue', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("First-Order Sobol Indices SA3", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Total-order indices bar plot
plt.subplot(1, 2, 2)
plt.bar(variables, total_order_indices_sa3, color='darkorange', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("Total-Order Sobol Indices SA3", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Plotting the second-order indices as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(second_order_indices_sa3, annot=True, fmt=".3f", cmap="coolwarm",
            cbar_kws={'label': 'Second-Order Index'}, xticklabels=variables,
yticklabels=variables)
plt.title("Second-Order Sobol Indices SA3", fontsize=14)
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.show()

#6. SENSITIVITY ANALYSIS 4
sobol_indices_sa4 = sobol.analyze(problem, result_data_sa4)
print(f"Values for Sensitivity Analysis 4 (avg dose content)")
print("First-order indices:", sobol_indices_sa4["S1"])
print("Second-order indices:", sobol_indices_sa4["S2"])
print("Total-order indices:", sobol_indices_sa4["ST"])

first_order_indices_sa4 = sobol_indices_sa4["S1"]
second_order_indices_sa4 = sobol_indices_sa4["S2"]
total_order_indices_sa4 = sobol_indices_sa4["ST"]

# Plotting the first-order and total-order indices
plt.figure(figsize=(12, 6))

# First-order indices bar plot
plt.subplot(1, 2, 1)
plt.bar(variables, first_order_indices_sa4, color='steelblue', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("First-Order Sobol Indices SA4", fontsize=14)
plt.ylabel("Sobol Index Value")
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Total-order indices bar plot
plt.subplot(1, 2, 2)
plt.bar(variables, total_order_indices_sa4, color='darkorange', alpha=0.8)
plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
plt.title("Total-Order Sobol Indices SA4", fontsize=14)
plt.ylabel("Sobol Index Value")
```

```
plt.xlabel("Variables")
plt.ylim(-0.1, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight layout()
plt.show()

# Plotting the second-order indices as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(second_order_indices_sa4, annot=True, fmt=".3f", cmap="coolwarm",
            cbar kws={'label': 'Second-Order Index'}, xticklabels=variables,
yticklabels=variables)
plt.title("Second-Order Sobol Indices SA4", fontsize=14)
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.show()
```

10.7.5 Part 5_Wood decay assessment.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
# Part (5/5) for Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Import moisture and temperature for specific simulation
#   Calculate dose values for wood coordinates
#   Export calculated dose files for PostProc visualisation

# *** SETUP ***
#1. IMPORT
import pandas as pd
import numpy as np
import shutil

from Delphin6GeoFileManager import Delphin6GeoFileManager
from Delphin6OutputFile import Delphin6OutputFile

#2. DEFINITIONS
    # DELPHIN MANAGERS
geoManager = Delphin6GeoFileManager()
d6o = Delphin6OutputFile()

    # READ / WRITE FILE
def read_file(fname):
    try:
        fobj = open(fname, 'r', encoding='utf-8')
        content = fobj.read()
        fobj.close()
        del fobj  # release file handle
        return content
    except IOError as e:
        print(str(e))
        raise RuntimeError("Error reading/opening file '{}'".format(fname))

def write_file(fname, contents):
    try:
        fobj = open(fname, 'w', encoding='utf-8')
        fobj.write(contents)
        fobj.close()
        del fobj  # release file handle
    except IOError as e:
        print(str(e))
        raise RuntimeError("Error opening file '{}' for writing".format(fname))

#3. VARIABLES
    # GEOMETRY INFORMATION
aoc = 754       #number of coördinates
aod = 3650      #amount of days (10 years)

wood_coordinates = [201, 221, 222, 242, 243, 244, 264, 265, 266, 267,
                    287, 288, 289, 290, 291, 310, 311, 312, 313, 314, 315,
                    333, 334, 335, 336, 337, 338, 339, 356, 357, 358, 359,
                    360, 361, 362, 363, 379, 380, 381, 382, 383, 384, 385,
                    386, 387, 402, 403, 404, 405, 406, 407, 408, 409, 410,
                    411, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435,
                    450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 474,
                    475, 476, 477, 478, 479, 480, 481, 482, 483, 498, 499,
                    500, 501, 502, 503, 504, 505, 506, 507, 521, 522, 523,
                    524, 525, 526, 527, 528, 529, 530, 543, 544, 545, 546,
                    547, 548, 549, 550, 551, 552, 564, 565, 566, 567, 568,
                    569, 570, 571, 572, 573, 583, 584, 585, 586, 587, 588,
                    589, 590, 591, 592, 602, 603, 604, 605, 606, 607, 608,
                    609, 610, 611, 620, 621, 622, 623, 624, 625, 626, 627,
                    628, 629, 637, 638, 639, 640, 641, 642, 643, 644, 645,
                    646, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661,
                    667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 682,
                    683, 684, 685, 686, 687, 688, 689, 690, 691, 697, 698,
                    699, 700, 701, 702, 703, 704, 705, 706, 712, 713, 714,
                    715, 716, 717, 718, 719, 720, 721, 727, 728, 729, 730,
                    731, 732, 733, 734, 735, 736, 742, 743, 744, 745, 746,
                    747, 748, 749, 750, 751, 757, 758, 759, 760, 761, 762,
                    763, 764, 765, 766, 772, 773, 774, 775, 776, 777, 778,
                    779, 780, 781, 760, 761, 762, 763, 764, 765, 767, 768,
```

```
                    769]

    # DOSE CALCULATION
avar = 3.2              #temperature weighing factor
evar = 6.75e-10         #variables for moisture content: Du
fvar = 3.5e-7
gvar = 7.18e-5
hvar = 7.22e-3
ivar = 0.34
jvar = 4.98
kvar = -1.8e-6          #variables for temperature: Dt
lvar = 9.57e-5
mvar = 1.55e-3
nvar = 4.17e-2

# *** MAIN ***
# 1. IMPORT SIMULATION FILE NAMES
    # * Change *: Multiple assessments can be executed at once. If only one is needed, do not
change name.
#simulation_files = ['SF_MF-I0-B0', 'SF_MF-I1-B1', 'SF_MF-I2-B2', 'SF_MF-I3-B3', 'SF_MF-I4-
B4','SF_MF-I5-B5']
simulation_files = ['One wood decay assessment']

total_dose_list = []
# 2. IMPORT DATA FOR SPECIFIC SIMULATION FILE
#FOR LOOP FOR ALL DATA (ALL DAYS / COORDINATES AND MOISTURE / TEMPERATURE) OF ONE SIMULATION
for name in simulation_files:

    dose_data = []
    # * Change *: Add path to content root of dose template in result directory of simulation
    # IMPORTANT: need to add template file yourself. Is NOT included in result folder after
simulation
    dose_template = read_file(f"SA_Variants/Simulation _ Wood decay/SF WDV IG/results/Dose
template.d6o")

    #IMPORT ALL MOISTURE AND TEMPERATURE DATA FOR EVERY DAY ('AOD')
    moisture_content = [] #WITHIN THIS LIST (FOR EVERY SIMULATION) THERE ARE X (AOD) AMOUNT OF
LISTS (FOR EVERY DAY)
    # * Change *: Add path to content root
    moisture_content_at_index = d6o.read("SA_Variants/Simulation _ Wood decay/SF WDV
IG/results/Moisture content.d6o", geoManager)
    for day in range(aod):
        moisture_content_at_index = d6o.valuesAtIndex(day)
        moisture_content.append(moisture_content_at_index)
        #print(f"At index {day} and day {day+1}, the list of moisture =
{moisture_content_at_index}")

    temperature = [] #WITHIN THIS LIST (FOR EVERY SIMULATION) THERE ARE X (AOD) AMOUNT OF
LISTS (FOR EVERY DAY)
    # * Change *: Add path to content root
    temperature_at_index = d6o.read("SA_Variants/Simulation _ Wood decay/SF WDV
IG/results/Temperature profile.d6o", geoManager)
    for day in range(aod):
        temperature_at_index = d6o.valuesAtIndex(day)
        temperature.append(temperature_at_index)
        #print(f"At index {day} day {day+1}, the list of temperature =
{temperature_at_index}")

    # 3. CYCLE THROUGH INDEXES FOR ALL DAYS
    #WITHIN EVERY SIMULATION FILE WE CYCLE THROUGH ALL THE DAYS (EVERY DAY SPECIFIC LIST OF
THE SIMULATION LIST)
    total_dose_data = []
    for day in range(aod):

        moisture_content_at_day = moisture_content[day]
        temperature at day = temperature[day]
        dose_at_day = []

        #CHECK IF MOISTURE AND TEMPERATURE FILES HAVE THE SAME SIZE (AND THUS AMOUNT OF
COORDINATES)
        if len(temperature) != len(moisture_content):
            print("Error: Temperature and moisture content lists must have the same length")

        #SAME LENGTH: CYCLE THROUGH ALL COORDINATES TO CALCULATE DOSE AT EVERY SPECIFIC
COORDINATE
        else:
            # 4. CYCLE THROUGH COORDINATES FOR ONE SPECIFIC DAY
```

```
            #FIND AND PROCESS COORDINATE SPECIFIC MOISTURE AND TEMPERATURE DATA
            for j in range(aoc):
                Dt = temperature_at_day[j]
                Du = moisture content at day[j] * 100
                #print(f"\nOn day {day+1}, the Temperature: {Dt}, Moisture content: {Du} at
coördinate {j+1}")

                #CHECK IF VALUES ARE WITHIN THERE LIMIT AND CALCULATE DOSE + ADD TO DOSE LIST
                if Du >= 25 and 0 <= Dt <= 40: #and j in wood_coordinates:
                    Du result = evar * Du**5 - fvar * Du**4 + gvar * Du**3 - hvar * Du**2 +
ivar * Du - jvar
                    Dt_result = kvar * Dt**4 + lvar * Dt**3 - mvar * Dt**2 + nvar * Dt
                    print(f"\nMoisture: {Du}, m component: {Du_result}. Temperature: {Dt}, t
component: {Dt_result} at day {day+1} for coordinate {j+1}")

                    dose = (avar * float(Dt result) + float(Du result)) * (avar + 1)**-1
                    dose_at_day.append(round(dose, 1))
                    print ("The temperature and moisture content values are within there
limits. \n"
                           "Calculated: Du_result= " + str(round(Du_result, 3)) + ",
Dt_result= " + str(round(Dt_result, 3)) + ", Dose= " + str(round(dose, 3)))

                # add zero to list if values are not within there limit
                else:
                    dose_at_day.append(0)
                    #print ("Temperature or moisture content is outside there limit\n")
            if day == 0:
                total_dose_data.append(dose_at_day)
            else:
            # Otherwise, accumulate doses from previous days
                cumulative_dose_at_day = [
                    total_dose_data[day - 1][i] + dose_at_day[i] for i in range(aoc)
                ]
                total_dose_data.append(cumulative_dose_at_day)
    #print(total_dose_data)

    final_dose_data = [[day * 24] + doses for day, doses in enumerate(total_dose_data)]
    output = "\n".join(" ".join(f"{value:.8f}" for value in day) for day in final_dose_data)

    #5. CREATE FILE FOR DOSE VALUES AND REPLACE PLACEHOLDER WITH CALCULATED VALUES
    content = dose_template.replace('${REPLACE_VALUES}', output)
    # * Change *: Add path to content root
    # IMPORTANT: Only the first part of content root, the 'write file' will add 'Dose
calculation profile' to folder
    write_file(f"SA_Variants/Simulation _ Wood decay/SF WDV IG/results/Dose calculation
profile.d6o", content)
```

10.7.6 File and folder check.py

```
# Created by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis
#
#For Sensitivity Analysis (SA) on material properties and wood decay assessment
#   Check if simulations is started
#   Check if simulation is finished (could also be finished with error)
#   Check if simulation result file has data size as expected (finished without error)

# *** SETUP ***
#1. IMPORT
import os
import time
import pandas as pd
import matplotlib.pyplot as plt
from JobRunner import JobRunner

#2. DIRECTORIES
    # * Change *: Path to Delphin executable (in program files)
DELPHIN_EXECUTABLE = r'C:\Program Files\IBK\Delphin 6.1\DelphinSolver.exe'
    # * Change *: Path to base directory (where files are saved)
base_directory = r"PATH"

# *** MAIN ***
#1. CREATE EMPTY LISTS
folder_check = []       # List for simulations that are started
file_check = []         # List for simulations that are finished
jobs = []               # List with job lines to be executed

#2. CHECK IF SIMULATION FOLDERS HAS BEEN MADE
for i in range(1536):
    # * Change *: Add correct simulation name
    folder_name = os.path.join(base_directory, f"SF_MF-I{i}-B{i}")
    if os.path.exists(folder_name) and os.path.isdir(folder_name):
        print(f"The folder with name: SF_MF-I{i}-B{i} exists")
    else:
        folder_check.append(i)
        # * Check *: Add correct simulation name
        if not os.path.exists('SF_MF-I{i}-B{i}' + "/var/restart.bin"):
            jobs.append([DELPHIN_EXECUTABLE, '-x', '--verbosity-level=0', '-p=2',
f'SA_Variants/Simulation again/SF_MF-I{i}-B{i}'])
        print(f"The folder with name: SF_MF-I{i}-B{i} does not exist")
print(jobs)

#2. CHECK IS SUMMARY FILE (AND THUS SIMULATION COMPLETE) HAS BEEN MADE
for j in range (1536):
    # * Change *: Add correct simulation name
    file_name = os.path.join(base_directory, f"SF_MF-I{j}-B{j}/log/summary.txt")
    if os.path.isfile(file_name):
        print(f"The file '{file_name}' exists in the directory.")
    else:
        file_check.append(j)
        print(f"The file '{file_name}' does NOT exist in the directory of simulation {j}.")

print(f"There are {len(folder_check)} simulations that didn't run, these have the number(s):
{folder_check}")
print(f"There are {len(file_check)} simulations that didn't finish, these have the number(s):
{file_check}")

#3. CHECK IF FILE HAS THE CORRECT DATA SIZE (AND THUS IF SIMULATION FINISHED WITHOUT ERRORS)
    # * Change *: The desired file size (check in completed simulations)
desired_size_bytes = 17170843
def file_size_in_kb(file_path):
    return os.path.getsize(file_path)

list_good = []
list_bad = []
error_list_number = []

for i in range(1536):
    file_name_2 = os.path.join(base_directory, f"SF_MF-I{i}-B{i}/results/Moisture content
profile.d6o")
    if os.path.isfile(file_name_2):                    # Only process files, not
subdirectories
        file_size_kb = file_size_in_kb(file_name_2)     # Get file size in KB
        if abs(file_size_kb - desired_size_bytes) < 100:  # Tolerance for matching sizes
```

```
            #print(f"File SF_MF-I{i}-B{i} has the size of 17.170 KB.")
            list_good.append(i)
        else:
            print(f"File SF MF-I{i}-B{i} does not match 17.170 KB. Size: {file_size_kb:.3f}
KB")
            list_bad.append(i)
            error_list_number.append(i)

#print(f"Amount of good results = {len(list_good)}, these are simulations: {list_good}")
#print(f"Amount of bad results = {len(list_bad)}, these are simulations: {list_bad}")
print(error_list_number)
```

10.7.7 Jobrunner.py

```
# Edited by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis

# Written by Andreas Nicolai (TU Dresden) for DELPHIN Simulation job processing.
# This file holds a class that can run simulation jobs in parallel.

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import subprocess      # import the module for calling external programs (creating
subprocesses)
import sys
import os
import shutil
import threading
import queue
import time

class JobRunner:
    """
    This class encapsulates a thread queue, a runner function and provides a convenient
    interface for running jobs in parallel.
    """
    def __init__(self, N):
        # create a queue for task scheduling
        self.jobQueue = queue.Queue()
        self.N = N

    # here we define our solver thread function
    def thread_function(self):
        """This function runs a simulation solver."""

        # run as long as we have jobs in the Queue
        while 1:
            # get a new task from the Queue
            job = self.jobQueue.get()
            start_time = time.time()
            print("Running '{}'\n".format(job[-1])) # print project file, should be last
argument
            # CREATE_NEW_CONSOLE is needed for Windows
            try:
                if sys.platform == "win32":
                    solverProcess =
subprocess.Popen(job,creationflags=subprocess.CREATE_NEW_CONSOLE)
                else:
                    solverProcess = subprocess.Popen(job)
                solverProcess.wait()
            except Exception as e:
                print(str(e))
            # tell the Queue that the task was done
            #end_time = time.time()
            #job duration = end time - start time
            #print(f"Job {job} of {len(jobs)} done at {time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(end_time))}, in {job_duration:.2f} seconds \n")

            self.jobQueue.task_done()

    def run(self, jobs):
        # now create as many threads as we need
        for i in range(self.N):
            t = threading.Thread(target=JobRunner.thread_function,args=[self])
            t.daemon = True
            t.start()
        # now add project files to the queue
        for job in jobs:
            self.jobQueue.put(job)
        # wait until all tasks are done
        self.jobQueue.join()
        return
```

10.7.8 __init__.py

```
# Edited by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis

# Written by Andreas Nicolai (TU Dresden) for DELPHIN
# IBK Python Helper Library
# Provides classes Delphin6OutputFile, Delphin6GeoFileManager, Delphin6GeoFile and JobRunner

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

__author__ = "Andreas Nicolai"

__copyright__ = "Copyright 2014-2020, IBK, TU Dresden"
__license__ = "BSD-compatible (see LICENSE.txt)"
__version__ = "1.1.0"

from JobRunner import *
from Delphin6OutputFile import *
from Delphin6GeoFile import *
from Delphin6GeoFileManager import *
```

10.7.9 Delphin6GeoFile.py

```
# Edited by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis

# Written by Andreas Nicolai (TU Dresden) for DELPHIN

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os.path
import numpy as np

class Element:
    """A class to store element index information. """
    def __init__(self):
        self.i = -1
        self.j = -1
        self.k = -1
        self.x = 0
        self.y = 0
        self.z = 0

    def __init__(self, i, j, k, x, y, z):
        self.i = i
        self.j = j
        self.k = k
        self.x = x
        self.y = y
        self.z = z

class Delphin6GeoFile:
    """
    Can read and interpret Delphin 6 geometry files (ASCII Versions).
    """

    def __init__(self):
        self.clear()

    def clear(self):
        self.headerData = dict()
        self.quantityDescription = ""
        # step sizes (element thickness) in x direction, size = x dimension
        self.xSteps = []
        # step sizes (element thickness) in y direction, size = y_dimension
        self.ySteps = []
        # step sizes (element thickness) in z direction, size = z_dimension
        self.zSteps = []
        # all elements (i,j,k indexes and x,y,z coordinates), size = nElements
        self.elements = []
        # filename of Geofile
        self.filename = ""

    def readMaterialsTable(self, lines, lastLine):
        for i in range(lastLine+1, len(lines)):
            line = lines[i].strip()
            if len(line) == 0:
                continue
            if line.startswith("TABLE") or line.startswith("INDICES") :
                return i
            # parse materials line... currently ignored
        raise RuntimeError("Incomplete Delphin 6 output file, missing data after MATERIALS
table.")

    def readGridTable(self, lines, lastLine):
        xSteps = lines[lastLine+1].split()
        self.xSteps = np.array(list(map(float, xSteps)))
        ySteps = lines[lastLine+2].split()
        self.ySteps = np.array(list(map(float, ySteps)))
        zSteps = lines[lastLine+3].split()
        self.zSteps = np.array(list(map(float, zSteps)))
        # we decide on x or y direction based on number of x values
        if self.zSteps.size != 1:
            raise RuntimeError("Invalid Delphin 6 output file, must have only one z-
coordinate!")
```

```
        for i in range(lastLine+4, len(lines)):
            line = lines[i].strip()
            if len(line) == 0:
                continue
            if line.startswith("TABLE") or line.startswith("INDICES") :
                return i
        raise RuntimeError("Incomplete Delphin 6 output file, missing data after GRID table.")

    def readElementsTable(self, lines, lastLine):
        for i in range(lastLine+1, len(lines)):
            line = lines[i].strip()
            if len(line) == 0:
                continue
            if line.startswith("TABLE") or line.startswith("INDICES") :
                return i
            # parse elements line... extract coordinates
            elementData = line.split()
            if len(elementData) == 6:
                self.elements.append( Element( int(elementData[3]), int(elementData[4]), -1,
float(elementData[1]), float(elementData[2]), 0) )
            else:
                self.elements.append( Element( int(elementData[4]), int(elementData[5]),
int(elementData[6]), float(elementData[1]), float(elementData[2]), float(elementData[3])) )
        raise RuntimeError("Incomplete Delphin 6 output file, missing data after ELEMENTS
table.")

    def readSidesTable(self, lines, lastLine):
        for i in range(lastLine+1, len(lines)):
            line = lines[i].strip()
            if len(line) == 0:
                continue
            # parse sides line... currently ignored
            # sidesData = line.split()
        # no data after sides table, so we read until the end
        return len(lines)

    def read(self, geoName):
        """Reads a geometry file.

        *geoName* -- Geometry file name.

        Returns:
            True on success, False on error (error messages are written).
        """

        try:
            geofile_obj = open(geoName, 'r')
            print ("USED: Geometry file: '{}'".format(os.path.split(geoName)[1]))
            geoLines = geofile_obj.readlines()
        except IOError:
            print ("Can't read/open geometry file.")
            return False
        geofile_obj.close()
        if (len(geoLines) == 0):
            print ('Empty geometry file.')
            return False

        # check header data
        self.d6gVersion = ""
        if geoLines[0].find('D6GARLZ! 006.') == -1:
            self.d6gVersion = "6"
        if geoLines[0].find('D6GARLZ! 007.') == -1:
            self.d6gVersion = "7"
        if self.d6gVersion == "":
            print ('Not a Delphin 6 geometry file (neither version 6 nor 7).')
            return False

        # process all lines, fork off read functions when a TABLE is found
        maxI = len(geoLines)
        i = 0
        while i < maxI:
            line = geoLines[i].strip()
            # skip empty geoLines
            if len(line) == 0:
                i = i + 1
                continue
```

```
            if line.startswith("TABLE"):
               # parse all tables
               tableNameTokens = line.split()
               if len(tableNameTokens) < 2:
                  raise RuntimeError('Incomplete/invalid Delphin 6 Output file, error in table
definition, line:\n{}'.format(keyword))
               tableName = tableNameTokens[1].strip()
               if tableName == "MATERIALS":
                  i = self.readMaterialsTable(geoLines, i)
               elif tableName == "GRID":
                  i = self.readGridTable(geoLines, i)
               elif tableName == "ELEMENT_GEOMETRY":
                  i = self.readElementsTable(geoLines, i)
               elif tableName == "SIDES_GEOMETRY":
                  i = self.readSidesTable(geoLines, i)
            else:
               i = i + 1

         # Done reading

         # TODO : add check for completeness/consistency of data
         return True

      def coordinatesInProfileCut(self, elementIndexes, direction):
         """This function obtains an array of continuous x, y or z-coordinates based on the
element indexes
         in the passed array.

         This function expects the selected elements to be part of a profile cut in either x, y,
or z direction.

         Arguments:
            elementIndexes Array of element indexes.
            direction Either 0 for x direction, 1 for y direction or 2 for z direction.
         """
         coordinates = []
         for i in elementIndexes:
            if direction == 0:
               coordinates.append(self.elements[i].x)
            elif direction == 1:
               coordinates.append(self.elements[i].y)
            else:
               coordinates.append(self.elements[i].z)
         return coordinates
```

10.7.10 Delphin6GeoFileManager.py

```
# Edited by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis

# Written by Andreas Nicolai (TU Dresden) for DELPHIN

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os.path
import Delphin6GeoFile

class Delphin6GeoFileManager:
    """Stores and provides geometry files for access by data files.

    Members:
       geoFiles -- dictionary with geometry files, keys are full file paths to
                   geometry file, values are Delphin6GeoFile objects
    """
    def __init__(self):
       self.clear()

    def clear(self):
       """Clears cached data."""
       self.geoFiles = dict()

    def geoFile(self,geoFilePath):
       """Returns the Delphin6GeoFile instance matching the file path.
       If geometry file has not been read, yet, the manager attempts to read
       the geometry file and returns an instance if successful.

       Arguments:
          geoFilePath - Full file path to geometry file

       Returns:
          Instance of Delphin6GeoFile object corresponding to geoFilePath, or
          None if reading of file failed (error messages are written out).
       """
       #if not self.geoFiles.has_key(geoFilePath):
       if geoFilePath not in self.geoFiles:
          geoFile = Delphin6GeoFile.Delphin6GeoFile()
          if geoFile.read(geoFilePath):
             self.geoFiles[geoFilePath] = geoFile
          else:
             print ("Error reading geometry file.")
             return None
       # return cached geometry file instance
       return self.geoFiles[geoFilePath]
```

10.7.11 Delphin6OutputFile.py

```
# Edited by Rik Trip, final edit on 13-01-2025
# TU Delft, Building Technology master thesis

# Written by Stefan Vogelsang based on work from Andreas Nicolai (TU Dresden) for DELPHIN

#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import os.path
import Delphin6GeoFileManager
import pandas as pd
import numpy as np
import datetime

class Delphin6OutputFile:
    """Can read and interpret Delphin output files (ASCII Versions)."""
    def __init__(self):
        self.clear()

    def clear(self):
        """Resets member variables to match an empty container."""
        self.headerData = dict()
        self.timePoints = []
        # holds time unit
        self.timeUnit = ""
        # holds all data as list of lists
        self.values = []
        # holds first column of data for convenient access to 2D data
        self.valueVector = []
        self.quantityDescription = ""
        # coordinate vector, holds either x, y or z coordinates of the profile cut
        self.coordinates = []
        # step sizes/element thickness vector, holds either x, y or z widths of the elements in
the profile cut
        self.steps = []
        # holds a string identifying the cut direction, either "X" or "Y" or "Z"
        self.direction = ""
        self.filename = ""
        # name for geofile
        self.geoName = ""

    def read(self, fname, geoFileManager):
        """Reads DataIO container file.

        Function supports DataIO containers with major version 6 and 7.

        - fname -- Full path to DataIO file.
        - geoFileManager -- Instance to geometry file manager.

        Returns:
            True if reading was successful, or False if reading failed due to an error.
            In case of errors, an error message is printed.
        """
        self.clear()
        self.filename = fname
        # open output file and parse data
        try:
            outfile_obj = open(fname, 'r')
            lines = outfile_obj.readlines()
        except IOError:
            print ("Can't read output or geo file.")
            return False
        outfile_obj.close

        if (len(lines) == 0):
            print ('Empty output file.')
            return False

        # check header data
        self.d6oVersion = ""
        if lines[0].find("D6OARLZ! 006.") != -1:
            self.d6oVersion = "6"
```

```
        if lines[0].find("D6OARLZ! 007.") != -1:
           self.d6oVersion = "7"
        if self.d6oVersion == "":
           print ('Not a Delphin 6 Output file.')
           return False

        # start parsing data file header until indices vector found
        headerEnd = 0
        for i in range(len(lines)):
           line = lines[i].strip()
           # skip empty geoLines
           if len(line) == 0:
              continue
           # check for end-of-header
           tokens = line.split('=')
           keyword = tokens[0].strip(" #\t")

           # how do we find to parse actual data ?
           if len(tokens) == 1:
              # must be a comment line without keyword-value pair, just skip this
              continue
           if keyword.startswith("INDICES") :
              headerEnd = i
              break

           # parse header types
           value = tokens[1].strip()
           self.headerData[keyword] = value


        # extract number of indices
        if headerEnd == 0:
           print ('Missing data or incomplete header.')
           return False

        try:
           numberString = lines[headerEnd]
           numbers = numberString.split("=")
           numbers = numbers[1].split()
           self.numberCount = len(numbers)

           # check if we have a TYPE Keyword
           #if not self.headerData.has_keys("TYPE"):
           if "TYPE" not in self.headerData:
              raise RuntimeError('Incomplete/invalid Delphin 6 Output file header, missing TYPE
keyword.')
           #if not self.headerData.has_key("SPACE_TYPE"):
           if "SPACE_TYPE"    not in self.headerData:
              raise RuntimeError('Incomplete/invalid Delphin 6 Output file header, missing
SPACE_TYPE keyword.')
           #if not self.headerData.has_key("TIME_UNIT"):
           if "TIME_UNIT" not in self.headerData:
              raise RuntimeError('Incomplete/invalid Delphin 6 Output file header, missing
TIME_UNT keyword.')
           self.timeUnit = self.headerData["TIME_UNIT"]

           # process selected elements and determine numbering direction and populate
           # the coordinates (cut) vector
           # we only do this for 3D output files
           if self.numberCount > 1:
              if self.headerData["SPACE_TYPE"] == "SINGLE":
                 # check if we have a FLUX file, currently sides files with multiple values are
not supported
                 if self.headerData["TYPE"] == "FLUX":
                    raise RuntimeError('Currently flux files with SPACE_TYPE = NONE are only
supported for single sides.')

                 # check if we have a TYPE Keyword FIELD file
                 if self.headerData["TYPE"] == "FIELD":

                    # we need a geometry file
                    #if not self.headerData.has key("GEO FILE"):
                    if "GEO_FILE" not in self.headerData:
                       raise RuntimeError('Missing GEO_FILE tag in header.')
                    # get relative path to geo file
                    self.geoName = self.headerData["GEO_FILE"]
                    # create absolute path
```

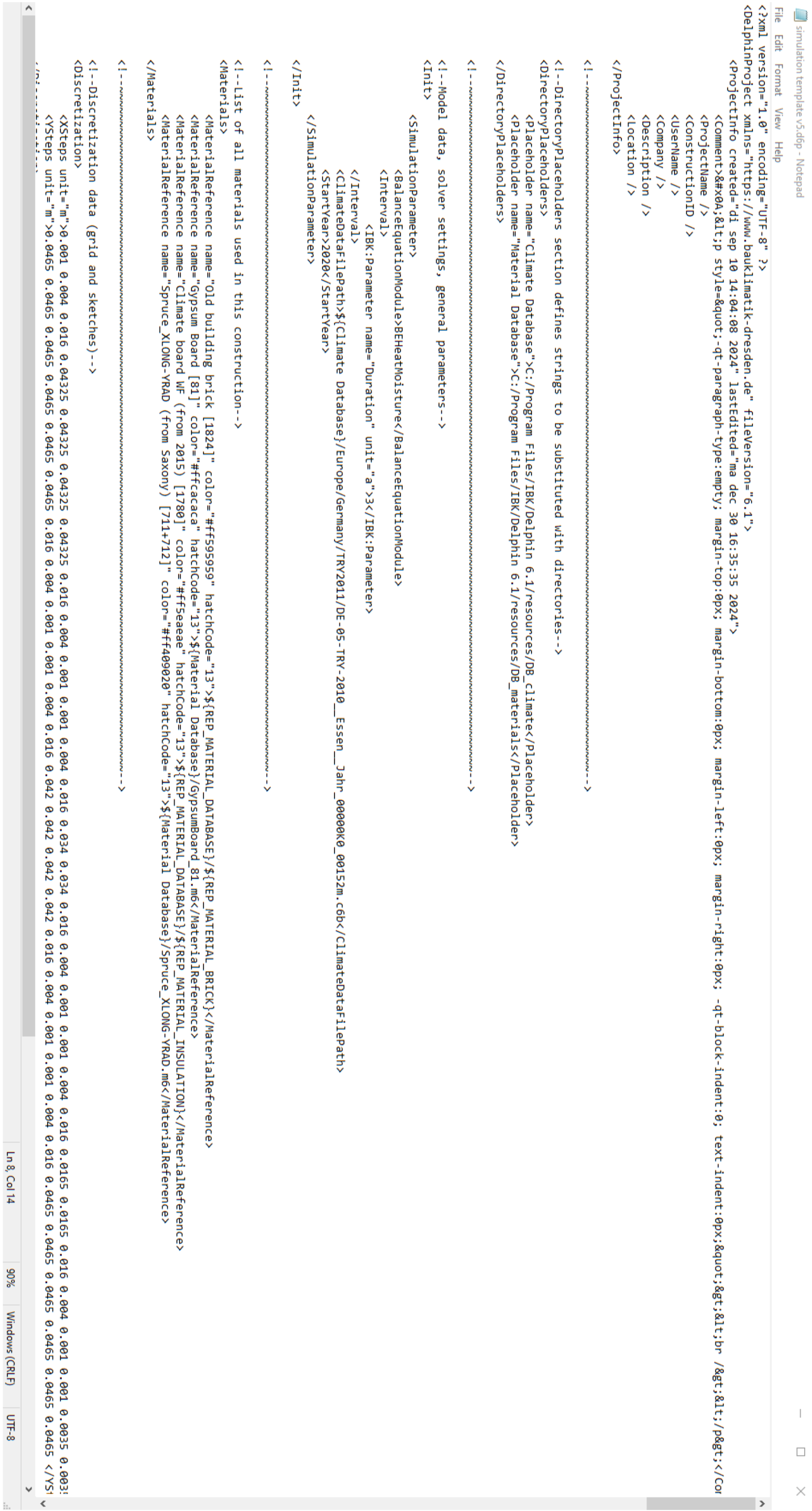
```
                    self.geoName = os.path.join( os.path.split(self.filename)[0], self.geoName)

                    geoFile = geoFileManager.geoFile(self.geoName)
                    if geoFile is None:
                       raise RuntimeError('Error reading/accessing geometry file.')

                    #iIndexSet = set()
                    #jIndexSet = set()
                    iIndexSet = []
                    jIndexSet = []
                    # collect all element indexes used in output data file
                    for i in range(self.numberCount):
                       elementIndex = int(numbers[i])
                       i = geoFile.elements[elementIndex].i
                       j = geoFile.elements[elementIndex].j
                       iIndexSet.append(i)
                       jIndexSet.append(j)
                    # different code based on coordinate direction
                    if len(jIndexSet) == 1:
                       self.direction = "X"
                       self.coordinates = geoFile.coordinatesInProfileCut(iIndexSet, 0)
                       for i in iIndexSet:
                          self.steps.append( geoFile.xSteps[ geoFile.elements[i].i ] )
                    else:
                       self.direction = "Y"
                       self.coordinates = geoFile.coordinatesInProfileCut(iIndexSet, 1)
                       for j in jIndexSet:
                          self.steps.append( geoFile.ySteps[ geoFile.elements[j].j ] )
             elif self.headerData["SPACE_TYPE"] == "MEAN" or self.headerData["SPACE_TYPE"] ==
"INTEGRAL":
                self.numberCount = 1

          headerEnd = headerEnd + 1

          self.timePoints = []
          self.values = []
          lines = lines[headerEnd:]
          for line in lines:
             # parse values
             vals = line.split()
             if len(vals) == 0:
                # skip empty lines
                pass
             else:
                self.timePoints.append(float(vals[0]))
                self.valueVector.append(float(vals[1]))
                values = []
                for i in range(1, len(vals)):
                   values.append(float(vals[i]))
                self.values.append(values)

          return (len(self.values) > 0)

       # catch raised exceptions
       except RuntimeError as err:
          print (str(err))
          return False

    def valuesAt(self, timeCutValue):
       # check if timeCutValue is present in list of time points
       timeCutIndex = -1
       for i in range(len(self.timePoints)):
          if self.timePoints[i] == timeCutValue:
             timeCutIndex = i
             break
       if timeCutIndex == -1:
          raise RuntimeError("Time cut value {0} does not match any output time
points.".format(timeCutValue) )
       valueVector = self.values[timeCutIndex]
       if self.numberCount != 1 and (len(valueVector) != len(self.coordinates)):
          raise RuntimeError("{1} values found at time cut value {0}, but only {2}
coordinates/elements "
                             "specified in elements table.".format(timeCutValue,
len(valueVector), len(self.coordinates) ) )
       return valueVector

    def valuesAtIndex(self, timeCutIndex):
```

```
        if timeCutIndex >= len(self.values):
            raise RuntimeError("Time cut value {} out of range (max index =
{}).".format(timeCutIndex, len(self.values)))
        valueVector = self.values[timeCutIndex]
        if self.numberCount != 1 and (len(valueVector) != len(self.coordinates)):
            raise RuntimeError("{1} values found at time cut index {0}, but {2}
coordinates/elements "
                               "specified in elements table.".format(timeCutIndex,
len(valueVector), len(self.coordinates) ) )
        return valueVector

    def valueVectorAt(self, index):
        """Extracts and returns a vector/column of data, most useful for reference data"""
        valueVec = []
        for i in range(len(self.values)):
            valueVec.append(self.values[i][index])
        return valueVec

    def asDataFrame(self):
        return pd.DataFrame(self.values, index=self.timePoints)

    def asDataFrameWithTimeIndex(self):
        unitConversionFactors = { 's' : 1, 'min' : 60, 'h' : 3600, 'd' : 24*3600, 'a' :
365*24*3600 }
        factor = unitConversionFactors[self.timeUnit]
        dateIndex = [datetime.datetime.fromtimestamp(i*factor) -
datetime.datetime.fromtimestamp(0) + datetime.datetime(year=2000, month=1, day=1)  for i in
self.timePoints ]
        return pd.DataFrame(self.values, index=dateIndex)
```

10.8 Template files

10.8.1 Simulation template file

Note: Not entire file is illustrated. But only edited in this section. Rest of file remains similar to original.




```
<?xml version="1.0" encoding="UTF-8" ?>
<DelphinProject xmlns="https://www.bauklimatik-dresden.de" fileVersion="6.1">
	<ProjectInfo created="di sep 10 14:04:08 2024" lastEdited="ma dec 30 16:35:35 2024">
		<Comment>&#x0A;&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;</Co
		<ProjectName />
		<ConstructionID />
		<UserName />
		<Company />
		<Description />
		<Location />
	</ProjectInfo>

	<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->

	<!--DirectoryPlaceholders section defines strings to be substituted with directories-->
	<DirectoryPlaceholders>
		<Placeholder name="Climate Database">C:/Program Files/IBK/Delphin 6.1/resources/DB_climate</Placeholder>
		<Placeholder name="Material Database">C:/Program Files/IBK/Delphin 6.1/resources/DB_materials</Placeholder>
	</DirectoryPlaceholders>

	<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->

	<!--Model data, solver settings, general parameters-->
	<Init>
		<SimulationParameter>
			<BalanceEquationModule>BEHeatMoisture</BalanceEquationModule>
			<Interval>
				<IBK:Parameter name="Duration" unit="a">3</IBK:Parameter>
			</Interval>
			<ClimateDataFilePath>${Climate Database}/Europe/Germany/TRY2011/DE-05-TRY-2010__Essen__Jahr_00000K0_00152m.c6b</ClimateDataFilePath>
			<StartYear>2020</StartYear>
		</SimulationParameter>
	</Init>

	<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->

	<!--List of all materials used in this construction-->
	<Materials>
		<MaterialReference name="Old building brick [1824]" color="#ff595959" hatchCode="13">${REP_MATERIAL_DATABASE}/${REP_MATERIAL_BRICK}</MaterialReference>
		<MaterialReference name="Gypsum Board [81]" color="#ffcacaca" hatchCode="13">${Material Database}/GypsumBoard_81.m6</MaterialReference>
		<MaterialReference name="Climate board WF (from 2015) [1780]" color="#ff5eaeae" hatchCode="13">${REP_MATERIAL_DATABASE}/${REP_MATERIAL_INSULATION}</MaterialReference>
		<MaterialReference name="Spruce_XLONG-YRAD (from Saxony) [711+712]" color="#ff409020" hatchCode="13">${Material Database}/Spruce_XLONG-YRAD.m6</MaterialReference>
	</Materials>

	<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->

	<!--Discretization data (grid and sketches)-->
	<Discretization>
		<XSteps unit="m">0.001 0.004 0.016 0.04325 0.04325 0.04325 0.04325 0.016 0.004 0.001 0.001 0.004 0.016 0.034 0.034 0.016 0.004 0.001 0.001 0.004 0.016 0.0165 0.0165 0.016 0.004 0.001 0.001 0.0035 0.003
		<YSteps unit="m">0.0465 0.0465 0.0465 0.0465 0.0465 0.0465 0.016 0.004 0.001 0.001 0.004 0.016 0.042 0.042 0.042 0.042 0.016 0.004 0.001 0.001 0.004 0.016 0.0465 0.0465 0.0465 0.0465 0.0465 0.0465 </YS
```

10.8.2 Material template file (brick)

Note: Not entire file is illustrated. But only edited in this section. Rest of file remains similar to original.


```
material template - brick.m6 - Notepad
File  Edit  Format  View  Help
D6MARL2! 006.001

[IDENTIFICATION]
    NAME                = DE: Altbeuziegel |EN: Old building brick
    AQUISITION_ID       = 1004
    INVESTIGATOR        = MaterialGenerator2 1.3.0
    LABORATORY          = DE: Institut für Bauklimatik, Technische Universität Dresden
    DATE                = 16.08.2023
    SAMPLING_DATE       = 02.05.2020
    COLOUR              = #ff400020
    FLAGS               = AIR_TIGHT
    CATEGORY            = BRICK
    COMMENTS            = DE: Altbauziegel 2 für Forschungsprojekt NAVE
    DBTYPE              = 1,7
    HATCHING            = 13


[STORAGE_BASE_PARAMETERS]
    RHO                 = 1698.32 kg/m3
    CE                  = 928.583 J/kgK
    THETA_POR           = 0.3513 m3/m3
    THETA_EFF           = 0.324047 m3/m3
    THETA_CAP           = ${B_THETA_CAP} m3/m3
    THETA_80            = ${B_THETA_80} m3/m3


[TRANSPORT_BASE_PARAMETERS]
    LAMBDA              = 0.59854 W/mK
    AW                  = 0.09695 kg/m2s05
    MEW                 = ${B_MEW} -
    KLEFF               = 9.9901e-10 s


[MOISTURE_STORAGE]
    FUNCTION =          Theta_l(pC)_de
        ${MS_TH/PC_X}
        ${MS_TH/PC_Y}
    FUNCTION =          pC(Theta_l)_de
        ${MS_PC/TH_X}
        ${MS_PC/TH_Y}


[MOISTURE_TRANSPORT]
    FUNCTION =          lgKl(Theta_l)
        0               0.000324401     0.000648803     0.000973204     0.00129761      0.00162201      0.00194641      0.00227081      0.00259521      0.00291961      0.00324401      0.00356842      0.00389
    0.0194641       0.0197885       0.0201129       0.0204373       0.0207617       0.0210861       0.0214105       0.0217349       0.0220593       0.0223837       0.0227081       0.0230325       0.0233569
    389282          0.0392526       0.039577        0.039014        0.0402258       0.0405502       0.0408746       0.041199        0.0415234       0.0418478       0.0421722       0.0424966       0.042821        0.0
    23              0.0581167       0.0584411       0.0587655       0.0590899       0.0594143       0.0597387       0.0600631       0.0603875       0.0607119       0.0610363       0.0613607       0.0616851       0.0622851       0.082073
    0.0781807       0.0785051       0.0788295       0.0791539       0.0794783       0.0798027       0.0801271       0.0804515       0.0807759       0.0811004       0.0814248       0.0817492       0.101538
0.0976448       0.0979692       0.0982936       0.098618        0.0989424       0.0992668       0.0995912       0.0999156       0.10024         0.100564        0.100889        0.101213
```

10.8.3 Material template file (insulation)

Note: Not entire file is illustrated. But only edited in this section. Rest of file remains similar to original.

material template - insulation.mtc - Notepad

File Edit Format View Help

```
D6WARLZ! 006.001

[IDENTIFICATION]
NAME                = DE: Klimaplatte WF (ab 2015) |EN: Climate board WF (from 2015)
AQUISITION_ID       = 707
PRODUCER            = DE: Calsitherm Silikatbaustoffe GmbH |EN: Calsitherm Silikatbaustoffe GmbH
PRODUCT_ID          = DE: Klimaplatte
LABORATORY          = DE: TU Dresden, File created by MaterialGenerator2 |EN: TU Dresden, File created by MaterialGenerator2
DATE                = 16.09.2021
COUNTRY             = Germany
COLOUR              = #ffeaeae
FLAGS               = AIR_TIGHT
CATEGORY            = INSULATION
COMMENTS            = DE: Calciumsilikatplatte ab 2016, diffusionsoffene und kapillaraktive Platte für Innendämmung |EN: Calcium silicate board from 2016, diffusion-open and capillary-active board for interior
DBTYPE              = 1,2
DATA_SHEET_SOURCE   = https://www.calsitherm.de/fileadmin/Downloads/Innendaemmung/Calsitherm_Technische_Hinweise_2018.pdf
HATCHING            = 13

[STORAGE_BASE_PARAMETERS]
RHO                 = 186.85 kg/m3
CE                  = 1100 J/kgK
THETA_POR           = 0.92949 m3/m3
THETA_EFF           = 0.928808 m3/m3
THETA_CAP           = ${I_THETA_CAP} m3/m3
THETA_80            = ${I_THETA_80} m3/m3

[TRANSPORT_BASE_PARAMETERS]
LAMBDA              = ${I_LAMBDA} W/mK
AW                  = 0.84657 kg/m2s05
MEW                 = ${I_MEW} -
LAMBDA_DESIGN       = 0.062 W/mK

[MOISTURE_STORAGE]
FUNCTION =          Theta_l(pC)_de
    ${MS_TH/PC_X}
    ${MS_TH/PC_Y}
FUNCTION =          pC(Theta_l)_de
    ${MS_PC/TH_X}
    ${MS_PC/TH_Y}

[MOISTURE_TRANSPORT]
FUNCTION =          lgK(Theta_l)
    0               0.000935365     0.00187073      0.0028061       0.00374146      0.00467683      0.00561219      0.00654756      0.00748292      0.00841829      0.00935365      0.010289        0.01122
```

10.8.4 Dose template file

Important: Copy moisture content file of that specific simulation, change file name to 'Dose calculation template (or to specified name in script) and change 'QUANTITY', 'QUANTITI_KW, 'VALUE_UNIT'. Remove all moisture content values and place placeholder '${REPLACE_VALUES}. Check if 'GEO_FILE' and 'GEO_FILE_HASH' are similar to simulation geographical data (.d6a file).