MVE385 Project course in mathematical and statistical modelling, 2013-12-13

Mathematical modelling of moisture transport in wood

Kristina Berndtsson, Simon Larson and Simon Sigurdhsson

Email kristina.berndtsson@gmail.com
larson.simon@gmail.com
sigurdhsson@gmail.com

Abstract When the moisture in wooden artefacts varies over time there is a risk that the artefact might be damaged. The Institute of Conservation at Gothenburg University researches how environmental factors affect the preservation of such artefacts. This project is an attempt to simulate and mathematically predict how the moisture distribution in a piece of wood changes under different circumstances.
To predict this a PDE model of moisture diffusion and heat conduction model was implemented in MATLAB. The parameters of this model where fitted to data using a Particle swarm algorithm. The obtained parameter values where used to evaluate the accuracy of data from two different measurement methods (the PH and MSR methods). From these simulations it was concluded that the data measured with the MSR method was more reliable then that measured using the PH method.

Partner: Charlotta Bylund Melin, Dept. of Conservation, Göteborg University

Contents

1 Introduction 2

2 Background 3

2.1 Theory . . . 3

2.2 Parameter search . . . 6

2.3 Measurement methods . . . 7

The PH method [7] The MSR method [8]

3 Implementation 8

4 Results 9

4.1 Parameter search . . . 10

4.2 MATLAB Simulation . . . 10

4.3 Parameter analysis . . . 14

5 Discussion 14

5.1 Analysis of the one-dimensional model . . . 15

5.2 Comparison of 1D and 3D models . . . 15

5.3 Parameter search . . . 17

5.4 The different mesurement methods . . . 21

The PH method [21] The MSR method [21]

5.5 Topics of further investigation . . . 22

6 Conclusions 23

7 References 24

A Nomenclature 25

B Varying parameters 26

C Code 32

1 Introduction

This project is a collaboration between the departement for conservation at the University of Gothenburg and the department of Mathematics at Chalmers University of Technology. The major focus of the research at the department of conservation is how the indoor climate and environment affect materials and cultural heritage property. The *Save and Preserve* research program, in collaboration with Gotland University, has research projects that address how built-in and loose furnishings are influenced by indoor climate and heating in which the conditions and agents that cause biodeterioration in buildings and collections are studied.

The research project is intended to provide the groundwork for developing climate criteria for building conservation. Field studies are carried out in different castles and churches that have varying degrees of climate control. Climate regulation can be a very costly process, especially for old buildings and thus the question if preservation could be done differently arises. To more accurately analyse the effect that variations of the ambient conditions have on the heat and moisture distributions in the wooden artefacts is of great interest in answering that question.

In this collaboration, the intent is to use mathematical modelling and computations to study the transport of moisture in wood under different environmental conditions. The models will be compared to measurement data, and will also be used to evaluate two different measurement techniques.

In order to evaluate measurement data from different measurement methods and to accurately predict the distribution of moisture in a wood sample a MATLAB model was implemented. Measurement data was used to estimate the sample dependent parameters in the model. The different measurement methods are evalueted by comparing measurement data to simulation data with similar environmental parameters. Comsol Multiphysics was used to create a 3D model of the sample with the estimated parameters, and this model was used to get a general idea of the behaviour of heat and moisture transfer in a three-dimensional body differs from the one-dimensional

case.

2 Background

This section details the problem background including the theory of the PDE model, a brief description of a parameter search method, and descriptions of two measurement methods used to obtain real-world data for comparison. The PDE model is presented in a form suitable for use in numerical PDE solvers.

2.1 Theory

The theory of heat and mass transfer is described by the so called Luikov equations [3]. These equations vary depending on the properties of the material where the transport takes place. According to Luikov and Mikhilov [3] the governing equations of the heat and moisture transfer in a capillary porous medium may be written as

$$\rho c_q \frac{\partial T}{\partial t} = \nabla \left(\left(k_q + \varepsilon \lambda k_m \delta \right) \nabla T + \varepsilon \lambda k_m \nabla U \right), \tag{1a}$$

$$\rho c_m \frac{\partial U}{\partial t} = k_m \delta \nabla T + k_m \nabla U, \tag{1b}$$

where T denotes the temperature and U the moisture potential (concentration of moisture). The other parameters[1] depend on the material and the surroundings in this model and in this model they are assumed to be both space and time independent.

The parameter subscripts m and q indicate whether the parameter is related to the moisture or heat transfer. The parameters c_m and c_q are the moisture and heat

[1] See the nomenclature in appendix A.

capacity respectively while k_m and k_q describe the materials' moisture and heat conductivity. The three parameters δ, ε and λ represent the so called thermographic coefficient, the ratio of the vapor diffusion coefficient to coefficient of total moisture diffusion and the heat of phase change for water.

The following boundary conditions describe the transport through a boundary that is in contact with a homogeneous gas of temperature T_a and moisture potential U_a, where both of these may be time dependent.

$$k_q \frac{\partial T}{\partial \boldsymbol{n}} = -\alpha_q (T - T_a) - (1 - \varepsilon)\lambda \alpha_m (U - U_a), \tag{2a}$$

$$k_m \frac{\partial U}{\partial \boldsymbol{n}} = -k_m \delta \frac{\partial T}{\partial \boldsymbol{n}} - \alpha_m (U - U_a). \tag{2b}$$

In these equations, $\boldsymbol{n}$ denotes the outward unit normal. For a complete description of how these equations are derived the reader is referred to Luikov and Mikhilov [3]. The parameters α_m and α_q denote the convective transfer coefficient of moisture and heat, respectively.

The mositure potential of the sourounding gas i calculated using approximations of equilibrium moisture content at the current temperature and relative humidity. The moisture equilibrium is the moisture content that the wood sample will strive to reach given the current conditions. According to Glass and Zelinka [1] the equilibrium mositure content can be approximated by

$$M_{eq} = \frac{1800}{W}\left[\frac{Kh}{1 - Kh} + \frac{K_1 K h + 2K_1 K_2 K^2 h^2}{1 + K_1 K h + K_1 K_2 K^2 h^2}\right],$$

where

$$\begin{aligned} W &= 349 + 1{,}29T + 0{,}0135T^2, \\ K &= 0{,}805 + 7{,}36 \cdot 10^{-4} T - 2{,}73 \cdot 10^{-6} T^2, \\ K_1 &= 6{,}27 - 9{,}38 \cdot 10^{-3} T - 3{,}03 \cdot 10^{-4} T^2, \\ K_2 &= 1{,}91 + 4{,}07 \cdot 10^{-2} T - 2{,}93 \cdot 10^{-4} T^2. \end{aligned}$$

Note that T is in degrees celcius. To obtain U_a one simply converts M_{eq} to molar concentration.

To simplify the solving of this coupled system of differential equations one often considers the one-dimensional case, where things are slightly simpler. According to Liu and Cheng [2], the governing equations in the one-dimensional case[2] simplify to

$$\rho c_q \frac{\partial T}{\partial t} = (k_q + \varepsilon\lambda k_m \delta)\frac{\partial^2 T}{\partial x^2} + \varepsilon\lambda k_m \frac{\partial^2 U}{\partial x^2}, \tag{3a}$$

$$\rho c_m \frac{\partial U}{\partial t} = k_m \delta \frac{\partial^2 T}{\partial x^2} + k_m \frac{\partial^2 U}{\partial x^2}, \tag{3b}$$

where $x \in [0, l]$ and $t > 0$. The boundary conditions (at $x = l$) become

$$k_q \frac{\partial T}{\partial x} = -\alpha_q (T - T_a) - (1 - \varepsilon)\lambda\alpha_m (U - U_a), \tag{4a}$$

$$k_m \frac{\partial U}{\partial x} = -k_m \delta \frac{\partial T}{\partial x} - \alpha_m (U - U_a) \tag{4b}$$

and at $x = 0$ no-flux boundary conditions are introduced. Initial values U_0 and T_0 determine the intitial distribution of heat and moisture.

Solving this simplified system analytically is possible but not a trivial task. In one dimension an exact solution can be obtained in the form of series expansions — such a solution along with its derivation can be found in Liu and Cheng [2]. Since analytical solutions are not necessary for the purposes of this project, they will not be discussed in further detail.

Most numerical solvers require a slightly different formulation of the system of PDEs than that provided above. The formulation used in most solvers, including MATLAB and Comsol which have been used in this project, are written in matrix form, with boundary conditions expressed in terms of the the flux matrix of the

[2] Assuming an infinite plate of thickness $2l$.

PDE system. In this form the model becomes

$$\begin{bmatrix} \rho c_q & 0 \\ 0 & \rho c_m \end{bmatrix} \begin{bmatrix} \frac{\partial T}{\partial t} \\ \frac{\partial U}{\partial t} \end{bmatrix} = \nabla \left(\begin{bmatrix} k_q + \varepsilon \lambda k_m \delta & \varepsilon \lambda k_m \\ k_m \delta & k_m \end{bmatrix} \begin{bmatrix} \nabla T \\ \nabla U \end{bmatrix} \right), \quad (5)$$

with boundary conditions written as

$$\begin{bmatrix} k_q + \varepsilon \lambda k_m \delta & \varepsilon \lambda k_m \\ k_m \delta & k_m \end{bmatrix} \begin{bmatrix} \frac{\partial T}{\partial \boldsymbol{n}} \\ \frac{\partial U}{\partial \boldsymbol{n}} \end{bmatrix} = \begin{bmatrix} \alpha_q (Ta - T) + \alpha_m \lambda (Ua - U) \\ \alpha_m (Ua - U) \end{bmatrix}. \quad (6)$$

2.2 Parameter search

The parameters in the model described above were obtained through stochastic optimisation of the unknown parameters k_m, k_q, c_m, c_q, α_m and α_q (as explained by Wahde [4], specifically using the Particle Swarm Optimisation method (PSO) described in chapter 5). This method was chosen due to the unknown properties of the objective function and the relatively large cost of function evaluation. The values of the remaining parameters are either given by literature [5, 2] or dependent on the measurement setup. These parameters were set as shown in table 1.

The particle swarm optimization algorithm could be described as generating a set of random points ('particles') in $\mathbb{R}^n$, where n is the number of parameters being estimated, and letting these move in $\mathbb{R}^n$ to find an optimal value of the objective function. The veolcities of these particles are affected by the local (particle-wise) and global best value. The position and velocity of each particle is updated each

TABLE 1: *Parameters from literature*

Parameter	l (m)	ρ (kg/m^3)	λ (J/kg)	ε (-)	δ (kg/(kg K))
Value	0.02	510	$2.5 \cdot 10^6$	0.3	2

iteration, after obtaining the local and global best value. The algorithm is discussed in further detail by Wahde [4], which also contains an algorithm outline [4, p. 123]. The implementation used here also includes inertia weights [4, p. 128].

The objective function used in the optimisation algorithm was defined as the root-mean-square difference between measurement data and corresponding (*i.e.*, equivalent initial and boundary conditions) data from the model evaluation[3]. The resulting optimal set of parameters given the measurement data is shown in table 2.

2.3 Measurement methods

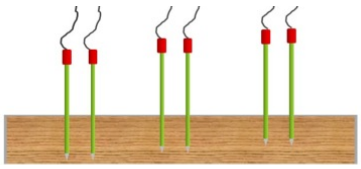
The data received from the departement of conservation is measured using two different methods. All wood samples are placed in a controlled environment where temperature and air humidity is controlled and varied over the measurement period. All wooden samples were covered by aluminium foil on five of the six sides to prevent moisture transport through these sides.

2.3.1 The PH method

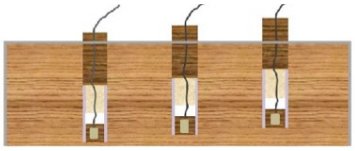
The first method measures the moisture content of a wooden sample by measuring the conductivity between two electrodes in the sample. The electrodes are isolated except for their tips. Thus by placing the electrodes at similar depths one measures the conductivity of the wood at a specific depth[4], and thus also the moisture content at that depth. This method is illustrated by fig. 1a.

[3] Note that the objective function only considers the moisture potential U, due to a lack of temperature measurement data.

[4] This may be a simplification — see discussion.



(A) *The PH method. Electrodes at different depths in a wood sample.*



(B) *The MSR method. Humidity sensors placed at different depths in a wood sample.*

FIGURE 1: *Physical appearance of the two measuring methods. Depth is measured from the bottom of the wood block in both cases. Images courtesy of Charlotta Bylund-Melin.*

2.3.2 The MSR method

The second method is similar, but here humidity sensors that measure the relative humidity in air are placed in small air pockets at different depths in the sample. Thus there is a further obstacle in the transport of moisture, where in addition to transportation inside the sample it is also transported between the air pocket and the wood surface surrounding the pocket. These humidity sensors also measure the temperature inside the pocket. To prevent moisture from entering the pocket from several sides, it is isolated using a small plastic tube and silicon washers. Figure 1b shows this setup in further detail.

3 Implementation

The model was implemented in MATLAB using a built-in PDE solver (`pdepe`) to solve the coupled one-dimensional system of partial differential equations. To prevent problems with integration tolerance of the `pdepe` solver it was necessary to apply some smoothing of the varying ambient humidity, preventing it from changing to fast (and hence yielding too large derivatives). The data used in this

project has humidity measurements once per hour, which allows for a large change between two time steps. In converting the data, time steps were converted to seconds and the data was linearly interpolated, obtaining data with which the solver had no problems.

To evaluate the difference between 1D simulations and simulations in the corresponding isotropic 3D case a model was implemented in Comsol. This model simulated the moisture transport in a slab where transport through all sides was permitted[5]. The Comsol model was governed by the equations given in eqs. (5) to (6), and the coefficient PDE mode of Comsol was used to implement the model.

As with MATLAB it is possible to implement models in Comsol where the boundary or ambient conditions vary over time. However, since the purpose of the Comsol was to compare the three-dimensional case with the corresponding one-dimensional approximation in MATLAB, such boundary conditions were not considered.

The particle swarm algorithm was also implemented in MATLAB. The initial swarm was randomly selected from $[0, 1]^6$, and weights were imposed in the objective funtion to force the parameter values into reasonable scales. A swarm size of 10 was used, which was sufficient to obtain convergence within a reasonable number of iterations.

4 Results

This section presents the results of performed simulations and compares these to measurement data. Parameters obtained from the parameter search are also presented, along with a sensitivity analysis of these parameters.

[5]Note that this differs from the conditions imposed on the measurements, discussed on page 7.

TABLE 2: *Parameters obtained using parameter search, as used by the model in fig. 2.*

Parameter	k_m	k_q	c_m	c_q	α_m	α_q
Value (PH)	$2.28 \cdot 10^{-8}$	0.709	0.639	1204	$9.87 \cdot 10^{-4}$	9.52
Value (MSR)	$1.32 \cdot 10^{-8}$	0.583	0.704	6753	$3.35 \cdot 10^{-4}$	4.54

4.1 Parameter search

The parameter search was done with respect to the data retrieved from both methods. The obtained parameters are presented in table 2.

The simulated results using the obtained optimal parameters is compared to the data from each method in figs. 2 and 3 — the ambient conditions (temperature and relative humidity) change over time throughout the measurement period, as shown by fig. 4. Ambient conditions used in simulations match these measured ambient values. As a initial distribution for the model a second degree polynomial fit was computed from the first data values (three values at different depths), assuming depth-wise symmetry in the wood sample.

4.2 MATLAB Simulation

The temperature change in the sample was modeled using the parameters found in the parameter search, and the results modeled over depth and time is shown in fig. 5. In this simulation the sample is 'moved' from room temperature (22 °C) to 15 °C and the absolute humidity is set to $4{,}5$ g/kg, with initial moistute content of the wood sample set to 15 %. The simulated moisture content is shown in fig. 6. One can see that the temperature stabilizes a lot quicker than moisture content does. To notice any significant change in moisture content, both T and U must be modeled with t between 1 h and 10 days.

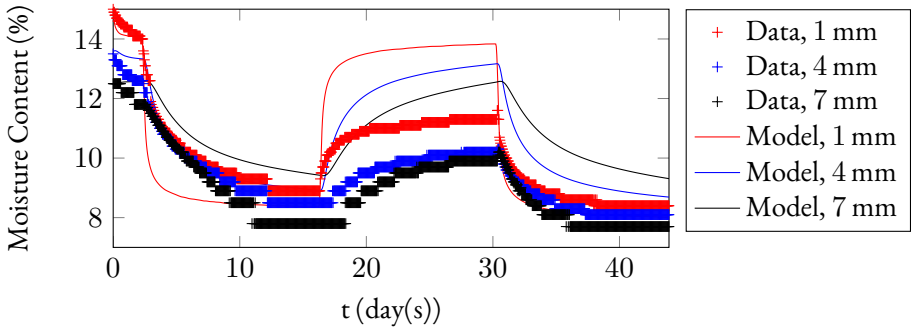


FIGURE 2: *The model, with parameters found using parameter search, compared to measurement data obtained using the PH method.*

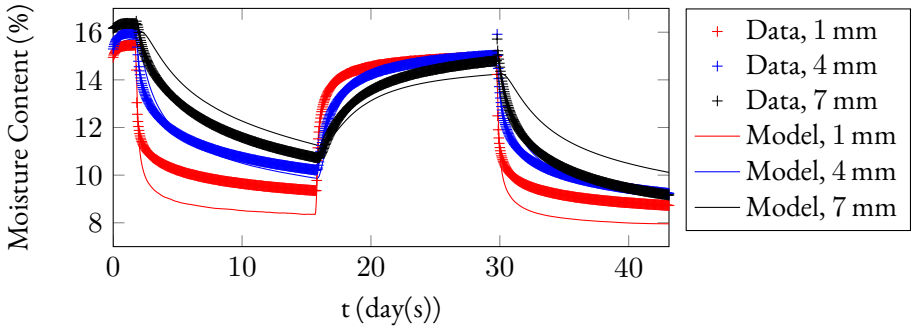
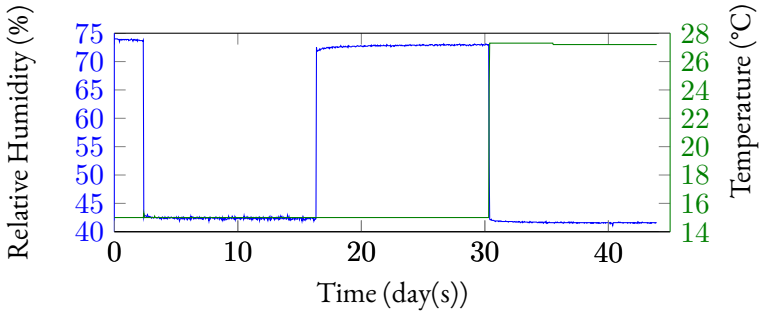
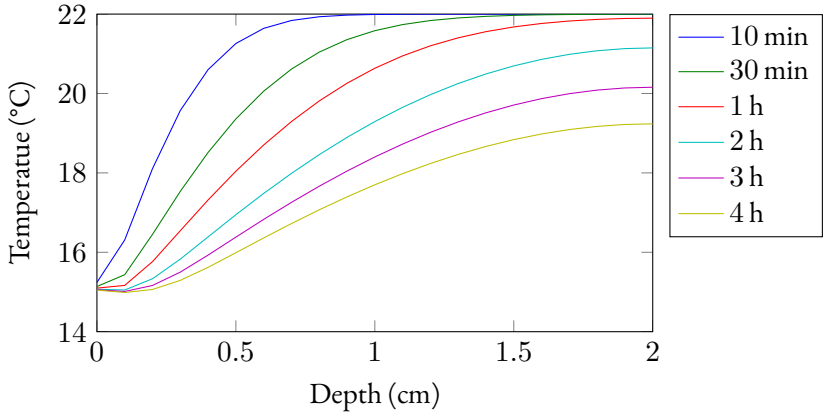


FIGURE 3: *The model, with parameters found using parameter search, compared to measurement data obtained using the MSR method.*

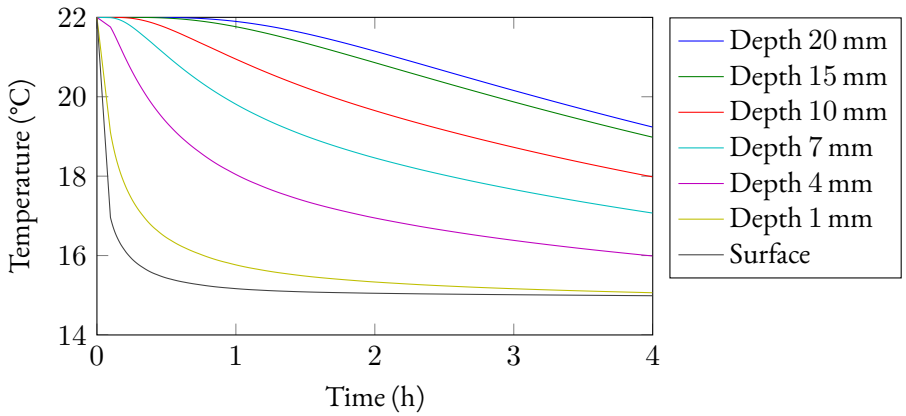


II

FIGURE 4: *Ambient conditions corresponding to measurement data, also used as ambient conditions in simulations.*

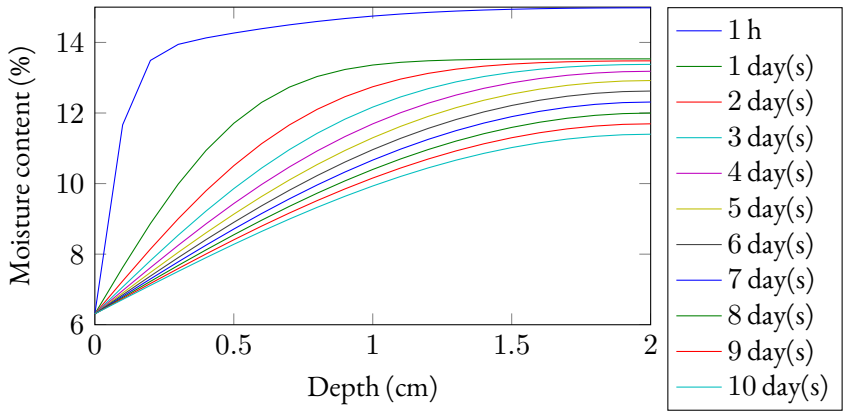


(A) *Temperature over depth at different times of the simulation*

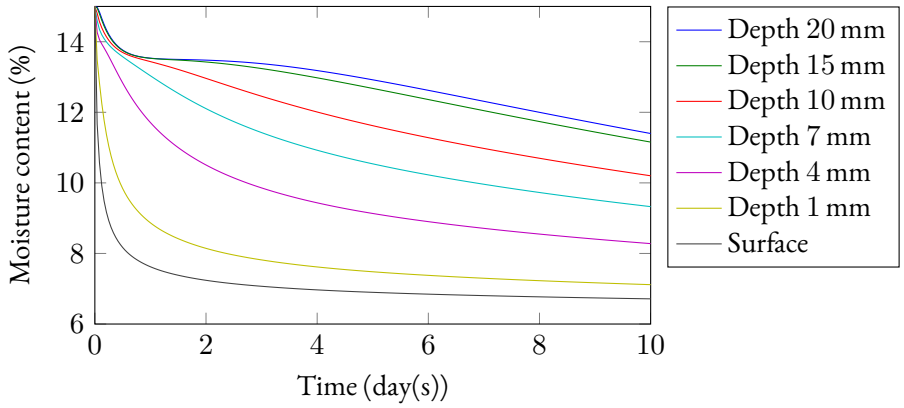


(B) *Temperature over time at different depths of the sample*

FIGURE 5: *Temperature as modeled in MATLAB*



(A) *Moisture content over depth at different times of the simulation*



(B) *Moisture content over time at different depths of the sample*

FIGURE 6: *Moisture content as modeled in MATLAB*

TABLE 3: *Values of* $\frac{\partial \log(U)}{\partial p}$ *and* $\frac{\partial \log(T)}{\partial p}$*, where* p *is an estimated parameter, after* 1 h.

Parameter	p	$\frac{\partial \log(U)}{\partial p}$	$\frac{\partial \log(T)}{\partial p}$
k_m	$1.93 \cdot 10^8$	$7.24 \cdot 10^4$	$-1.06 \cdot 10^7$
k_q	0.04	0.09	-3.55
c_m	0.09	0.02	-0.29
c_q	4075.2	$1.93 \cdot 10^{-7}$	$3.88 \cdot 10^{-5}$
α_m	$7.47 \cdot 10^{-4}$	4.08	$7.76 \cdot 10^{-8}$
α_q	235	$3.42 \cdot 10^{-6}$	$-1.02 \cdot 10^{-5}$

4.3 Parameter analysis

Appendix B shows how the results of the simulation vary when the parameters are changed. In the appendix, parameters obtained by parameter search are varied one by one, with ambient conditions unchanged (see page 10 above).

Table 3 summarizes the results of the parameter analysis by logarithmic derivatives of both U and T. In accordance with figs. 10 to 15 in appendix B, most parameters are insensitive to change when modeling temperature, while all parameters except k_m (in particular, c_q and α_q which describe heat transfer) are fairly insensitive when measuring moisture potential.

5 Discussion

In this section the results presented earlier are discussed and several shortcomings of the models are analysed. Further suggestions on how the work in this area could proceed, in order to construct a more accurate and complete model, are also discussed.

5.1 Analysis of the one-dimensional model

The model used to fit the parameters from the data has several shortcomings that could potentially render the simulations inaccurate.

The moisture transport through the wood sample is approximated as moisture transport through an infinitely large wooden plate with similar thickness, which is a significant assumption. In addition, wood is a highly heterogeneous material — yet our model assumes the opposite. Both these assumptions will undoubtedly affect the accuracy of the model. These approximations may be corrected by using a more advanced 3D anisotropic model which may be implemented in Comsol using the coefficient PDE mode.

The difficulty in implementing a more complete three-dimensional model lies in that the parameters for this kind of model are unknown. To perform a parameter search one needs to perform a large amount of simulations. Thus, with such an advanced model and an increased number of parameters the optimisation would increase the computational requirements drastically. Also, the measurement equipment required to gather data for this has to accurately measure the differences between the moisture transport in different directions. The data provided for this project was obtained from measurements where the moisture was transported predominantly in the radial direction of the wood. Therefore, fitting parameters to such a model is presently not possible.

5.2 Comparison of 1D and 3D models

To evaluate the accuracy of our one-dimensional approximation we simulated the same situation in both MATLAB and Comsol. The simulated conditions are the same as previously stated: a wooden sample is moved from temperature $22\,^{\circ}\mathrm{C}$ to $15\,^{\circ}\mathrm{C}$, with initial moisture content $15\,\%$ and ambient absolute humidity $4{,}5\,\mathrm{g/kg}$. In Comsol our slab which is set to have the size $10\,\mathrm{cm} \times 10\,\mathrm{cm} \times 4\,\mathrm{cm}$. Thus the simulation in MATLAB approximates the transport through one of the larger faces

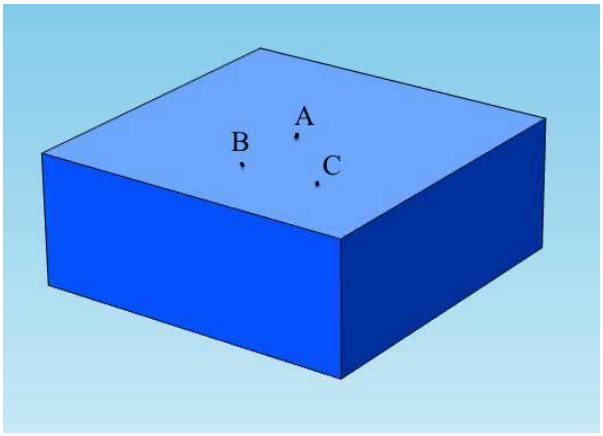


FIGURE 7: *The three different measurement points (A, B, C) for the Comsol model.*

and along a line normal to this surface. We chose three different such lines, one in the center of the slab (A), one that lies halfway from the center towards one of the sides (B) and one halfway between one of the slabs corners and the center (C). This is illustrated in fig. 7 where the three measurement points are marked A, B and C.

In figs. 8a to 8c one can see the resulting change in moisture distribution at different points in time. In these figures the parameter arc length represents the depth in cm, as in the graphs from the MATLAB simulation. As one can see from the graphs these do not differ significantly — on close inspection, there are definitely differences but in comparison with the one-dimensional approximation (a similar graph from this simulation can be found in fig. 6), which due to symmetry should be very close to the measurements at point A, it is evident that the one-dimensional model is a highly accurate approximation.

However, fig. 9 shows that the moisture distribution is not homogeneous at a specific depth from the surface. Thus, if one performs measurements close to the edge

of a sample there might be significant differences between measurements and the simulated data.

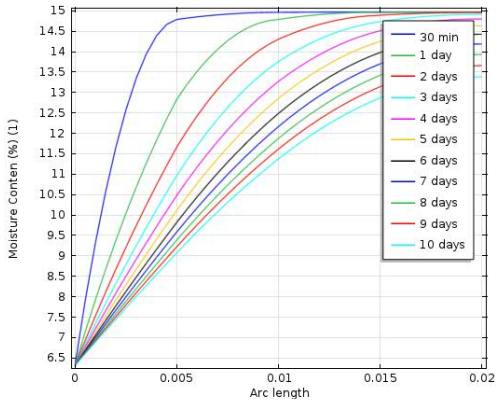
On the other hand one should note that both these models assume that the material is isotropic and that the transport of moisture is not directionally dependent. This is not the case for wood, where the fiber structure and capillary structure of the material makes transport in certain directions easier then in others.

To perform more accurate simulations of the measurement techniques used to gather the data used in this project it would be a good idea to use a 3D model where one boundary is given the boundary conditions above (in eqs. (2a) to (2b)), while the others where given boundary conditions emulating the covering of the sides by aluminium foil. This would probably resemble the condition for heat given above (aluminium foil does not inhibit temperature diffusion), but with the transport of moisture across the boundary set to zero.

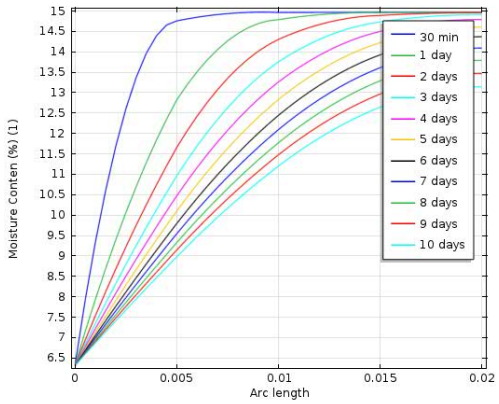
5.3 Parameter search

The chosen algorithm for parameter fitting is well equipped to deal with this problem. Due to the large computational requirements of each function evaluation, the relatively quick convergence of the PSO algorithm (at the cost of not having an optimality guarantee) that was used is a great advantage. The problem itself is not easily solved due to the instability of the solutions with respect to the different parameters. Since the variation in many places is very small it is difficult to find a direction in which to search for a minimum. This makes the PSO algorithm a well suited algorithm for the current optimisation problem.

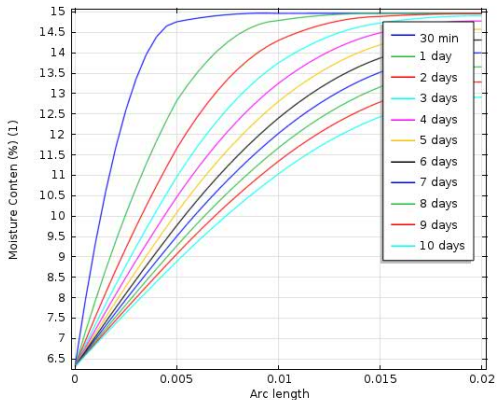
In the first part of fig. 2, the modeled moisture content does not seem to behave like the measured data. The model suggests that the wood dries faster closer to the surface, but the measured data does not indicate that this is the case. This is possibly a flaw of the measurements, and the behaviour is unexpected due to the fact that drying and absorbtion of moisture should react more rapidly close to the surface. This hypothesis (that measurement data of the PH method is flawed) is somewhat confirmed by the fact that the measurement data from the MSR method fits better



(A) *Point A*



(B) *Point B*



(C) *Point C*

FIGURE 8: *Moisture content simulated in the Comsol 3D model at different times, in the three points shown by fig. 7*

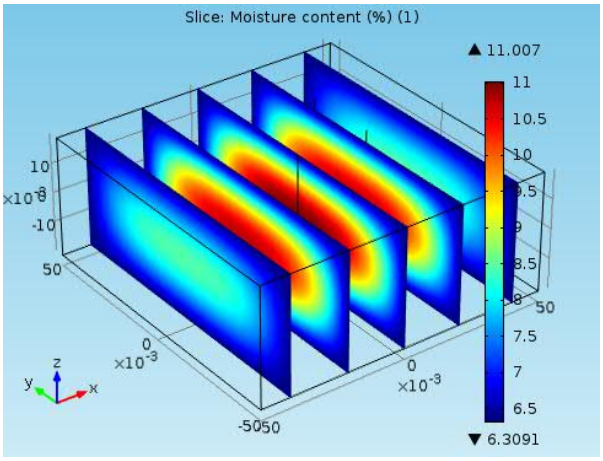


FIGURE 9: *The moisture distribution in our slab after 10 days, as simulated in Comsol*

with the expected behaviour of the model even if the parameters have been optimized to fit with the data from the PH measurement method.

If one compares the obtained parameters with parameters found for spruce in the article by Younsi, Kocaefe and Kocaefe [5], one finds high agreement. In table 4 we see that the parameters are highly similar with exception for the patamer α_q. One explanation for why the parameter α_q is further from its counterpart given by Younsi, Kocaefe and Kocaefe [5] is that this parameter mainly concerns the heat transfer through the material. Since the data used to find the parameters only consists of moisture measurements the optimization algorithm has the freedom to change these parameters quite a lot without affecting the fit of the data significantly.

As one can see in the figures in appendix B, the effect of changing the parameters concerning heat transfer does not to any large extent affect the moisture distribution. Thus, by approximating the heat parameters either by values from litterature or by comparing it to measurement data, one could obtain much more accurate estimations of these parameters.

TABLE 4: *Parameters from the optimization and litterature parameters for spruce.*

Parameter	k_m	k_q	c_m	c_q	α_m	α_q
Value (PH)	$2.28 \cdot 10^{-8}$	0.709	0.639	1204	$9.87 \cdot 10^{-4}$	9.52
Value (MSR)	$1.32 \cdot 10^{-8}$	0.583	0.704	6753	$3.35 \cdot 10^{-4}$	4.54
Value (spruce)	$2.20 \cdot 10^{-8}$	0.577	1.000	4201	$1.00 \cdot 10^{-4}$	25

To obtain a better fit it would have been favorable to also measure the temperatures at the different depths in the wood, and include these in the objective function. This is possible with the MSR method — but not using the PH method — which again provides an argument for the use of this measurement method instead.

Researching more accurate approximations of the moisture equilibrium, either from literature or by experimental means, could result in an improved parameter estimation and better model-measurement fit for the PH method. For the MSR method it should also be noted that the measurement data is retreived as relative humidity and temperature, requiring the same approximation of moisture equilibrium as when evaluating the ambient conditions. Therefore the fit is dependent on identical approximations both when calculating ambient conditions and when deriving moisture content from the measured data. This could potentially make the model appear to fit better than it would if such approximations were not made.

Performing the parameter search against a three dimensional model with more appropriate boundary conditions to improve accuracy could be a favourable idea. But since the computational time for solving the system of PDEs in three dimensions is substantially larger this would result in a much larger requirement of computational power. Also in the three dimensional case an anisotropic model could be implemented but here the number of parameters needed to be optimized would increase aswell and resulting in additional computational complexity.

5.4 The different mesurement methods

5.4.1 The PH method

Measuring the moisture content of the sample with the PH method may be potentially erroneous. The method does not in fact measure at an exact depth, but rather (due to the current distributing itself) measures a kind of average around a certain depth. The possible paths for the current differ at different depths in the wood, shifting the center of momentum of this average. At 1 mm the current is more restricted than at 7 mm, which could lead to differences not only depending on the depth of the measurement (since the measurement may in fact be one at a depth of more than 1 mm. Measuring a system that is not at equilibrium with this method is also a potential error source, as the anisotropic nature of wood suggests that the moisture diffusion is not entirely uniform in the normal direction.

When inspecting the data measured with the PH method (see fig. 2) it seems like the drying occurs faster for the measuremet made at 7 mm than the mesurment at 1 mm. This is not in agreement with either the model or intuition. This might be a problem that is confined to these specific measurments, but new measurments with this method would be needed to ensure that this is not a general problem with the measurment technique.

5.4.2 The MSR method

In the MSR method several things complicate the comparison with simulated values. Since this method measures the relative humidity inside a small air pocket in the wood, there is an additional obstacle of transport from wood to air. This extra transport might result in delays in the course of the moisture transport. This could lead to problems with the parameter search since the parameters determine exactly the speed at which the moisture is transported.

Additionally, since the moisture transport with high probability is direction de-

pendent the fact that the MSR loggers are isolated from the sides might increase the inacuracy of the measurments. Attempting to perform these measurments without the isolation would result in not being able to distinguish between different depths of the sample, since the size of the sensor is relatively large compared to the differences in depth.

However, as displayed in fig. 3, the data from the MSR sensors behaves more like the model prediction than the data from the PH method. Thus it is not surprising that one obtains a significantly better fit when comparing the model to the data.

5.5 Topics of further investigation

Beyond the scope of this investigation there are some topics that may be interesting for future studies. One such topic would be to use the model implemented in this paper to see the effects on the wood sample when the ambient conditions vary periodically over time — both with shorter intervals, as the temperature and humidity varies in a room over a day, and longer intervals, where the temperature and humidity difference over *e.g.* a year. This might show which effects are more harmful for the wood: the small but frequent changes that occur over short time spans, or the larger ones over longer time.

This work has only focused on modelling heat and moisture transport in one direction in a wood sample. Since wood is a highly anisotropic material measurements in several directions have to be obtained to formulate a more complete model. One way to do this, in terms of fitting the theoretical model to the measured data, would be to perform the measurements at different depths in the same sample where the sample is not insulated in any direction.

6 Conclusions

The model used in this project is accurate in modeling the temperature and moisture transfer in wood [5]. In the one-dimensional case MATLAB was used to solve the PDE and fit the parameters. With estimated parameters, MATLAB was also used to efficiently study the diffrent behaviors of temperature and moisture transfer with respect to ambient conditions and parameter stability.

The three-dimensional model studied in Comsol left room for improvment. To get a more accurate result from the three-dimensional model, the boundary conditions have to be defined to more accurately describe the conditions of the relevant measurement method. Another aspect of the Comsol implementation is the fact that computations in three dimensions require a considerable amount of computational power compared to the one-dimensional case, which may not be justified by the supposedly greater accuracy. Also if one was to implement this model with anisotropic behaviour the parameter set grows in size, increasing the computational time required to estimate all parameters significantly.

The moisture-related parameters found by the parameter search are in good agreement with literature, while the temperature-related parameters are less accurate. This is likely due to the fact that the objective function used in the parameter search disregards difference in temperature, only taking the moisture content into account. Temperature was omitted from the parameter search since no data describing temperature inside the wood samples was available.

The comparison between the two models indicates that the MSR measurment technique is the better of the two measurement methods. The MSR method corresponds better to the teoretical model and agrees more with the intuitive interpretation of how moisture behaves in wood. Why this is the case is left as an open question, but it may be due to a general lack of accuracy in the PH method (as discussed earlier) or possibly the result of a corrupt data set.

7 References

[1] S. V. Glass and S. L. Zelinka. *Physical Properties and Moisture Relations of Wood*. 1st ed. Madison, WI: U.S. Dept. of Agriculture, Forest Service, Forest Products Laboratory, 2010.

[2] Jen Y. Liu and Shun Cheng. 'Solutions of Luikov equations of heat and mass transfer in capillary-porous bodies'. In: *International Journal of Heat and Mass Transfer* 34 (7 1991), pp. 1747–1754.

[3] A. V. Luikov and Yu. A. Mikhilov. *Theory of energy and mass transfer*. English edition. Oxford: Pergamon Press, 1965. ISBN: 978-0080101279.

[4] M. Wahde. *Biologically Inspired Optimization Methods. An Introduction*. 1st ed. Southampton, Boston: WIT Press, 2008. ISBN: 978-1-84564-148-1.

[5] R. Younsi, D. Kocaefe and Y. Kocaefe. 'Three-dimensional simulation of heat and moisture transfer in wood'. In: *Applied Thermal Engineering* 26 (11–12 2006), pp. 1274–1285.

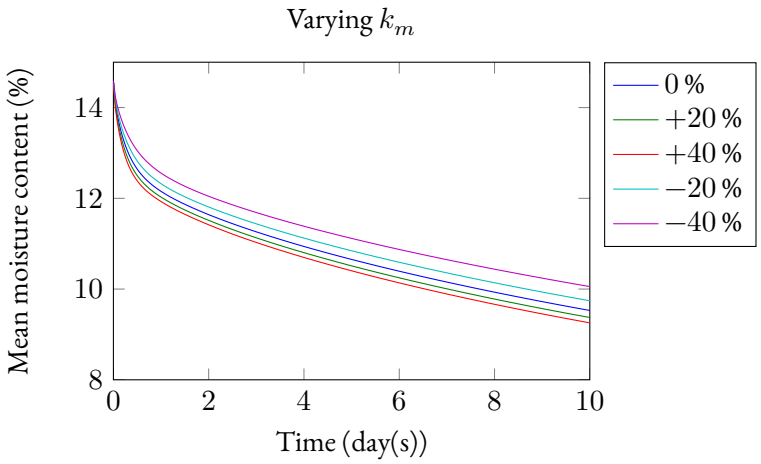
A Nomenclature

Variable	Unit	Description
C	$\mathrm{kg_{moisture}/kg_{wood}}$ (%)	moisture content
c_m	$\mathrm{kg_{moisture}/(kg_{wood}\ M)}$	moisture capacity
c_q	J/(kg K)	heat capacity
k_m	$\mathrm{kg_{moisture}/(m\ s\ M)}$	moisture conductivity coefficient
k_q	W/(m K)	heat conductivity coefficient
l	m	thickness of wood specimen
T	K	temperature
T_a	K	ambient temperature
T_0	K	initial temperature
t	s	time
U	M	moisture potential
U_a	M	ambient moisture potential
U_0	M	initial moisture potential

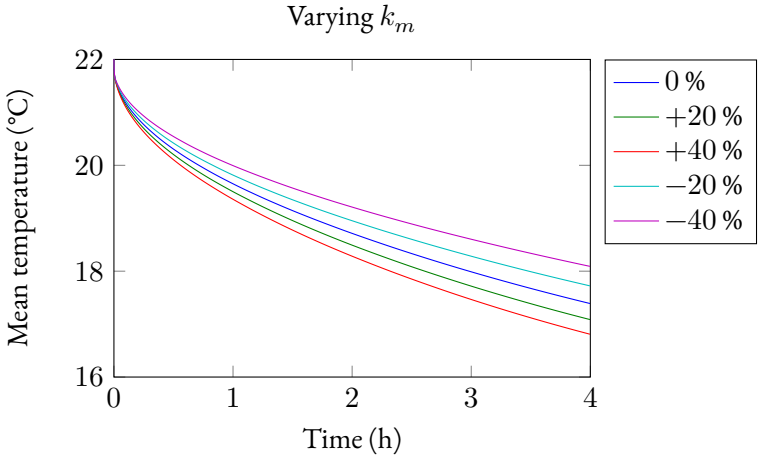
Greek symbols

Symbol	Unit	Description
α_m	$\mathrm{kg_{moisture}/(m\ s^2\ M)}$	convective mass transfer coefficient
α_q	$\mathrm{W/(m\ K^2)}$	convective heat transfer coefficient
δ	$\mathrm{kg_{moisture}/(kg\ K)}$	thermographic coefficient
ε	—	ratio of vapor diffusion coefficient to coefficient of total moisture diffusion
λ	J/kg	heat of phase change
ρ	$\mathrm{kg/m^3}$	dry body density

B Varying parameters

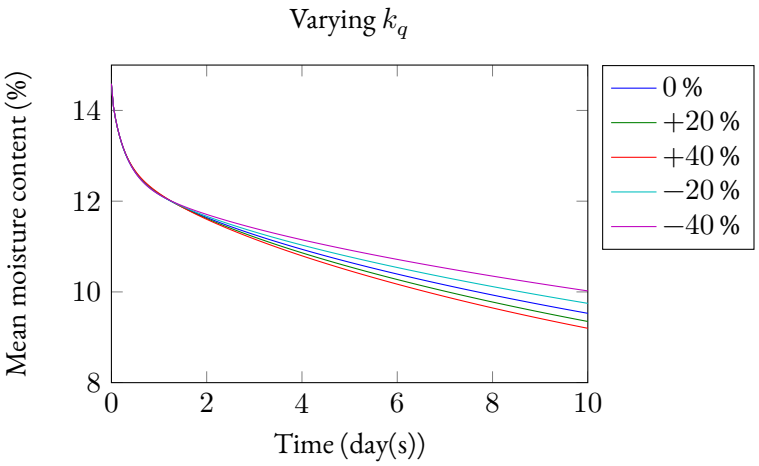


(A) *Moisture content*

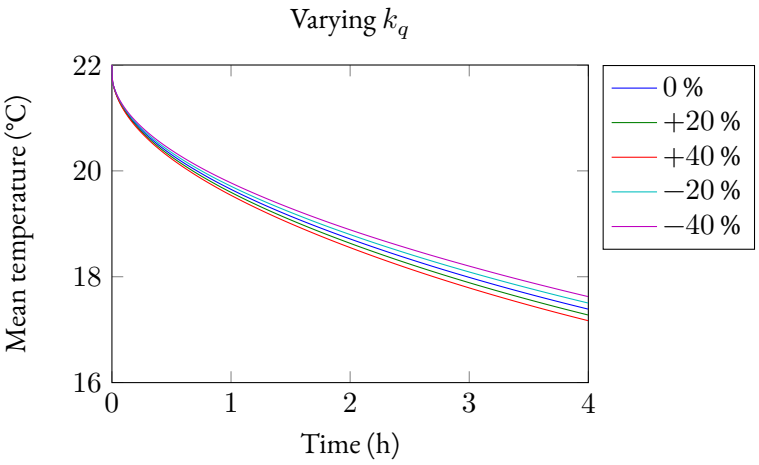


(B) *Temperature*

FIGURE 10: *The mean moisture content and temperature changing over time for different values of* k_m

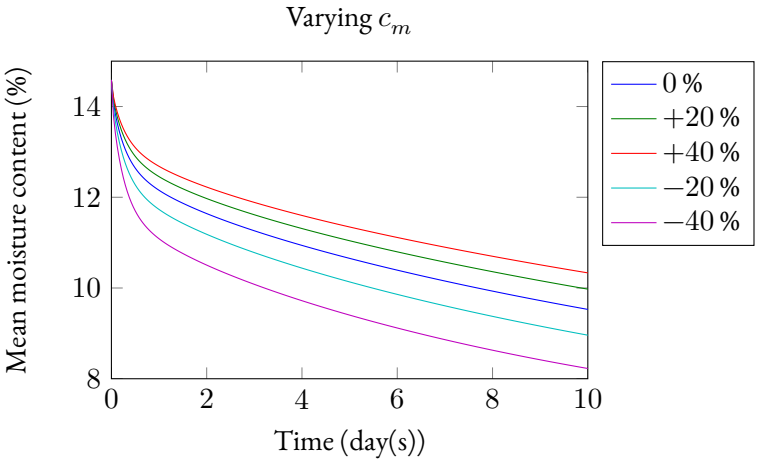


(A) *Moisture content*

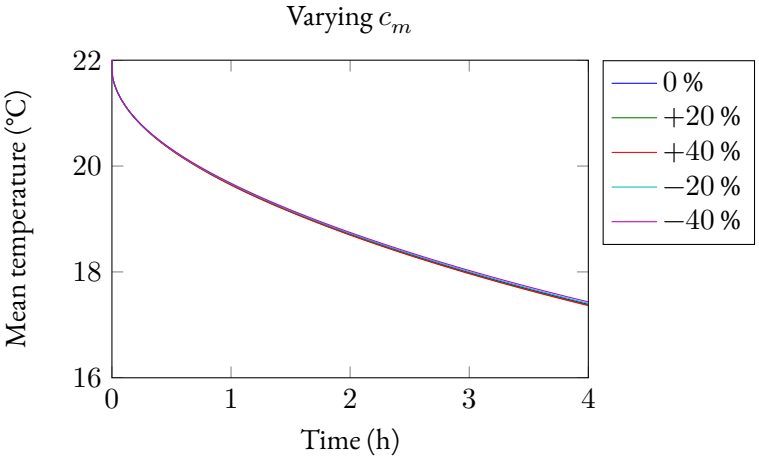


(B) *Temperature*

FIGURE II: *The mean moisture content and temperature changing over time for different values of* k_q

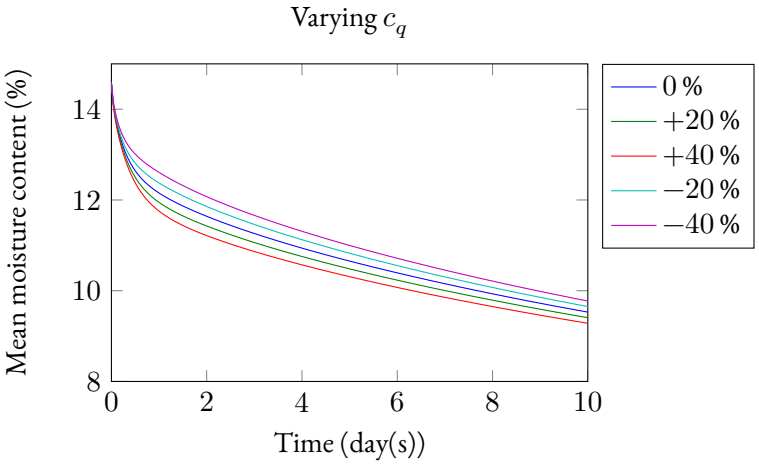


(A) *Moisture content*

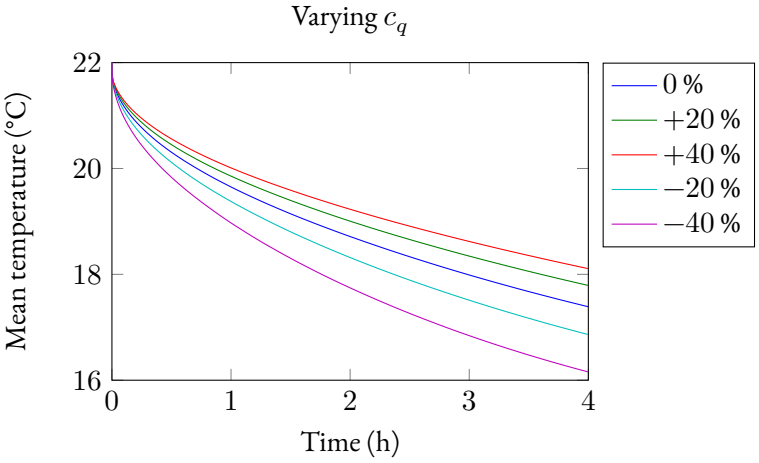


(B) *Temperature*

FIGURE 12: *The mean moisture content and temperature changing over time for different values of* c_m

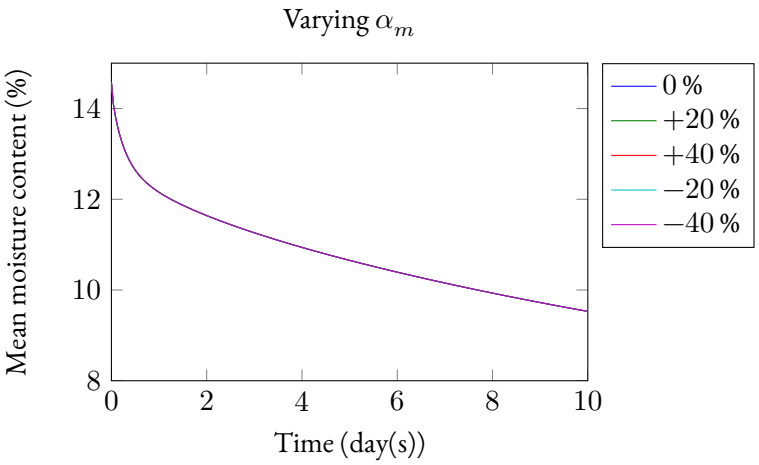


(A) *Moisture content*

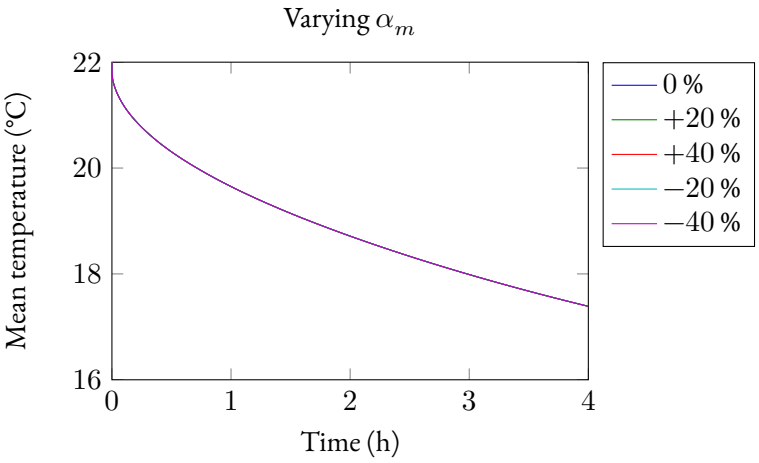


(B) *Temperature*

FIGURE 13: *The mean moisture content and temperature changing over time for different values of* c_q

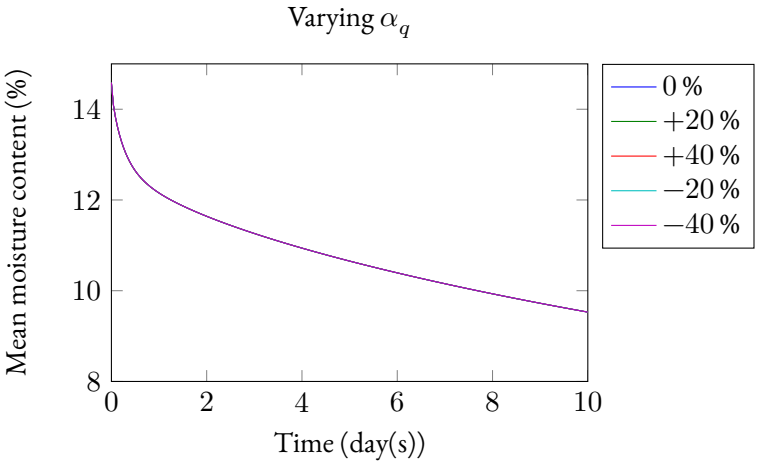


(A) *Moisture content*

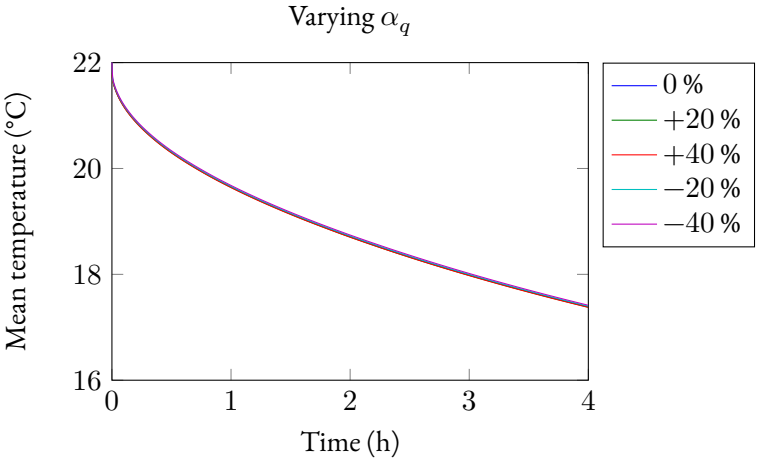


(B) *Temperature*

FIGURE 14: *The mean moisture content and temperature changing over time for different values of* α_m



(A) *Moisture content*



(B) *Temperature*

FIGURE 15: *The mean moisture content and temperature changing over time for different values of* α_q

C Code

MATLAB PDE implementation

GetResult.m

```
%% Run model and return results
function [u1, u2] = GetResult( t_in, variables )
    % Generate parameters and data
    [l, rho, lambda, epsilon, delta, k_q, k_m, c_q, c_m, h_q, h_m, T0] = Parameters( variables );
    [data_t, depths, data_U, Ua, Ta] = LoadData(c_m);

    % Definitions used in the PDE
    flux = [k_q+epsilon*lambda*k_m*delta epsilon*lambda*k_m; k_m*delta k_m];
    % Polyfit the initial distribution
    data_x=l-depths;
    p=polyfit([data_x -data_x],[data_U(1,:) data_U(1,:)],2);
    x=unique([linspace(0, l, 100) data_x(:)']);
    U0=polyval(p,x);

    % Generate t-points for the solver and save the indices required for comparison
    x = unique([linspace(0, l, 100) data_x(:)']);
    t = unique([1:60:max(data_t) data_t(:)']);
    t_indices = [];
    for i = 1:length(t)
        if sum(data_t == t(i)) > 0
            t_indices = [t_indices i];
        end
    end
    t_in_indices = [];
    for i = 1:length(t)
        if sum(t_in == t(i)) > 0
            t_in_indices = [t_in_indices i];
        end
    end

```

```
    % Define ambient conditions
    Ua=interp1(t(t_indices)',Ua',t');
    Ta=interp1(t(t_indices)',Ta',t');

    % Define PDE system
    pdefun = @(a,b,c,d)(PDEF(a, b, c, d, rho, c_m, c_q, flux));
    pdebc = @(a,b,c,d,e)(PDEBC(a, b, c, d, e, lambda, h_q, h_m, Ta, Ua, flux,t));
    pdeic = @(a)([T0; U0(find(x==a))]);

    solution = pdepe(0, pdefun, pdeic, pdebc, x, t);

    u1 = solution(t_in_indices,:,1);
    u2 = solution(t_in_indices,:,2);
end
```

PDEF.m

```
%% PDE function
function [c,f,s] = PDEF(~, ~, ~, DuDx, rho, c_m, c_q, flux)
        c = [rho*c_q
         rho*c_m];
        f = flux * DuDx;
        s = [0; 0];
end
```

PDEBC.m

```
%% PDE boundary conditions
function [pl,ql,pr,qr] = PDEBC(~, ~, ~, ur, t, lambda, h_q, h_m, Ta, Ua, flux,t_temp)
        pl = [0;0];
    ql = inv(flux);

    [~,closest]=min(abs(t-t_temp));

        pr = [h_q*(ur(1) - Ta(closest)) + h_m*lambda*(ur(2) - Ua(closest)); ...
                  h_m*(ur(2)-Ua(closest))];
          qr = [1 0; 0 1];
end
```

Parameters.m

```
%% Return parameters used in the model
function [l, rho, lambda, epsilon, delta, k_q, k_m, c_q, c_m, h_q, h_m,T0] = ...
  Parameters( searchedParameters )
        % Known parameters
        l = 0.02;
        rho = 510;
        lambda = 2.5e6;
        epsilon = 0.3;
        delta = 2;
    T0 = 22+273.15;

    % Parameters to be estimated
        k_q = searchedParameters(1);
        k_m = searchedParameters(2);
        c_q = searchedParameters(3);
        c_m = searchedParameters(4);
        h_q = searchedParameters(5);
        h_m = searchedParameters(6);
end
```

LoadData.m

```
%% Load measurement data from file
function [data_t, data_x, data_U, Ua, data_Ta] = LoadDataMSR(c_m,rho)
        % For speed, these values are pre-calculated and saved in dataMSR.mat
%     rows = 15:1050;
%     data = xlsread('MSR.20120705_20120924.Chalmers.xlsx');
%
%     t0 = (data(rows(1),1) + data(rows(1),2))*86400;
%     data_t = (data(rows,1) + data(rows,2))*86400 - t0;
%
%     data_T = data(rows, [7 10 25])+273.15;
%     data_Ta=data(rows,4)+273.15;
%
%     data_RHa = data(rows, 3);
%     data_RH = data(rows, [6 9 24]);
%
%     TaC=data_Ta-273.15;
%     TC=data_T-273.15;
```

```
%
%     W=349+1.29.*TC+0.0135*(TC.^2);
%     k=0.805+(7.36e-4 .*TC)-(2.73e-6).*(TC.^2);
%     k_1=6.27-(9.38e-3 .*TC)-(3.03e-4).*(TC.^2);
%     k_2=1.91+(4.07e-2 .*TC)-(2.93e-4).*(TC.^2);
%
%     data_x = [1e-3 4e-3 7e-3];
%
%     RH=data_RH./100;
%     RHa=data_RHa./100;
%
%     Meq=(1800./W).*(k.*RH./(1-k.*RH)+(k_1.*k.*RH+2*k_1.*k_2.*(k.^2).*(RH.^2))./ ...
%         (1+k_1.*k.*RH+k_1.*k_2.*(k.^2).*(RH.^2)));
%
%     Wa=349+1.29.*TaC+0.0135*(TaC.^2);
%     ka=0.805+(7.36e-4 .*TaC)-(2.73e-6).*(TaC.^2);
%     ka_1=6.27-(9.38e-3 .*TaC)-(3.03e-4).*(TaC.^2);
%     ka_2=1.91+(4.07e-2 .*TaC)-(2.93e-4).*(TaC.^2);
%
%     Meqa=(1800./Wa).*(ka.*RHa./(1-ka.*RHa)+(ka_1.*ka.*RHa+2*ka_1.*ka_2.*(ka.^2).* ...
%          (RHa.^2))./(1+ka_1.*ka.*RHa+ka_1.*ka_2.*(ka.^2).*(RHa.^2)));
%
%     save('dataMSR.mat','data_t','Meq','Meqa','data_x','data_Ta');

    % Load saved data
    load('dataMSR.mat');
    Ua=smooth(Meqa.*(rho*55.55/(100*1000)),5);
    data_U=Meq.*(rho*55.55/(100*1000));
end
```

MATLAB parameter search implementation

PSO algorithm

ParameterSearch.m

```
%% Particle swarm optimization algorithm for estimating parameters

%% Variables
nParticles = 10;
nVariables = 6;
variableWeights = [1 1e-7 1e4 1e-1 1e3 1e-3];
xMin = 0;
xMax = 1;
vMax = 0.025;
inertiaWeight = 1.4;
beta = 0.99;
lowerInertiaWeight = 0.4;
alpha = 1;
deltaT = 1;
nIterations = 100;

%% Initialize positions and velocities of particles
particlePositions = xMin + rand(nParticles, nVariables) * (xMax - xMin);
particleVelocities = alpha/deltaT * ( (xMin - xMax)/2 + ...
                        rand(nParticles, nVariables) * (xMax - xMin) );

globalBestValue = Inf;
globalBestPosition = zeros(1, nVariables);
particleBestValue = Inf * ones(nParticles, 1);
particleBestPosition = Inf * ones(nParticles, nVariables);

%% Iterate
for i=1:nIterations
    % Evaluate all particles
    particleValues = EvaluateParticles(particlePositions, variableWeights);
    % Update local/global best solution
```

```
    [particleBestValue, particleBestPosition] = ...
        UpdateParticleBest(particlePositions, particleValues, particleBestPosition, particleBestValue);
    [globalBestValue, globalBestPosition] = ...
        UpdateGlobalBest(particlePositions, particleValues, globalBestPosition, globalBestValue);
    % Update velocities and positions
    particleVelocities = UpdateParticleVelocities(particleVelocities, particlePositions, ...
                                particleBestPosition, globalBestPosition, inertiaWeight, deltaT);
    particleVelocities = RestrictParticleVelocities(particleVelocities, vMax);
    particlePositions = UpdateParticlePositions(particlePositions, particleVelocities, deltaT);
    % Update inertia weight
    if inertiaWeight > lowerInertiaWeight
        inertiaWeight = beta * inertiaWeight;
    end
    fprintf('Current best (%d): f = %6.3f,  x = (%s)\n', ...
        i, globalBestValue, sprintf('%g,', globalBestPosition.*variableWeights));
end
% Pretty-print information
fprintf('Global best: f = %6.3f,  x = (%s)\n', ...
    globalBestValue, sprintf('%g,', globalBestPosition.*variableWeights));
```

EvaluateParticles.m

```
%% Evaluates all particles of the swarm
function particleValues = EvaluateParticles(particlePositions, variableWeights)
    [nParticles, ~] = size(particlePositions);
    particleValues = zeros(nParticles, 1);
    for i=1:nParticles
        if sum(particlePositions < 0) > 0
            particleValues(i) = Inf;
        else
            particleValues(i) = ObjectiveFunction(particlePositions(i, :).*variableWeights);
        end
    end
end
```

RestrictParticleVelocities.m

```
%% Restrict the velocities of all particles in the swarm
function restrictedVelocities = RestrictParticleVelocities(originalVelocities, vMax)
    [nParticles, nVariables] = size(originalVelocities);
    restrictedVelocities = originalVelocities;
```

```
    particleVelocityNorm = sqrt(sum(originalVelocities.^2, 2));
    particleNeedsRestriction = (particleVelocityNorm > vMax);
    particleVelocitiesNormalized = ...
        originalVelocities ./ repmat(particleVelocityNorm, 1, nVariables);
    restrictedVelocities(particleNeedsRestriction, :) = ...
        vMax .* particleVelocitiesNormalized(particleNeedsRestriction, :);
end
```

UpdateGlobalBest.m

```
%% Update the swarm best position
function [newBestValue, newBestPosition] = ...
  UpdateGlobalBest(particlePositions, particleValues, oldBestPosition, oldBestValue)
    newBestValue = oldBestValue;
    newBestPosition = oldBestPosition;
    if min(particleValues) < oldBestValue
        newBestValue = min(particleValues);
        newBestPosition = particlePositions(find(min(particleValues) == particleValues, 1), :);
    end
end
```

UpdateParticleBest.m

```
%% Update the personal best of all particles in the swarm
function [newBestValue, newBestPosition] = ...
  UpdateParticleBest(particlePositions, particleValues, oldBestPosition, oldBestValues)
    newBestValue = particleValues;
    newBestPosition = oldBestPosition;
    particleHasImprovedBest = (particleValues < oldBestValues);
    newBestValue(particleHasImprovedBest) = particleValues(particleHasImprovedBest);
    newBestPosition(particleHasImprovedBest, :) = particlePositions(particleHasImprovedBest, :);
end
```

UpdateParticlePositions.m

```
%% Update all particle positions in the swarm
function newPositions = UpdateParticlePositions(oldPositions, velocities, deltaT)
    newPositions = oldPositions + velocities .* deltaT;
end
```

UpdateParticleVelocities.m

```
%% Update all particle velocities in the swarm
function updatedVelocities = ...
  UpdateParticleVelocities(oldVelocities, particlePositions, particleBestPosition, ...
   globalBestPosition, inertiaWeight, deltaT)
    [nParticles, nVariables] = size(oldVelocities);
    cognitiveComponent = 2 .* rand(nParticles, nVariables) .* ...
        (particleBestPosition - particlePositions) ./ deltaT;
    socialComponent = 2 .* rand(nParticles, nVariables) .* ...
        (repmat(globalBestPosition, nParticles, 1) - particlePositions) ./ deltaT;
    updatedVelocities = inertiaWeight.*oldVelocities + cognitiveComponent + socialComponent;
end
```

Objective function

ObjectiveFunction.m

```
%% Run model and compare with measurement data
function [ value ] = ObjectiveFunction( variables )
    % Generate parameters and data
    [l, rho, lambda, epsilon, delta, k_q, k_m, c_q, c_m, h_q, h_m, T0] = Parameters( variables );
    [data_t, depths, data_U, Ua, Ta] = LoadData(c_m);

    % Definitions used in the PDE
    flux = [k_q+epsilon*lambda*k_m*delta epsilon*lambda*k_m; k_m*delta k_m];
    % Polyfit the initial distribution
    data_x=l-depths;
    p=polyfit([data_x -data_x],[data_U(1,:) data_U(1,:)],2);
    x=unique([linspace(0, l, 100) data_x(:)']);
    U0=polyval(p,x);

    % Generate t-points for the solver and save the indices required for comparison
    x = unique([linspace(0, l, 100) data_x(:)']);
    t = unique([1:60:max(data_t) data_t(:)']);
    t_indices = [];
    for i = 1:length(t)
        if sum(data_t == t(i)) > 0
            t_indices = [t_indices i];
        end
```

```
    end

    % Define ambient conditions
    Ua=interp1(t(t_indices)',Ua',t');
    Ta=interp1(t(t_indices)',Ta',t');

    % Define PDE system
    pdefun = @(a,b,c,d)(PDEF(a, b, c, d, rho, c_m, c_q, flux));
    pdebc = @(a,b,c,d,e)(PDEBC(a, b, c, d, e, lambda, h_q, h_m, Ta, Ua, flux,t));
    pdeic = @(a)([T0; U0(find(x==a))]);

    % Find indices corresponding to measurement data in x direction
    [~,ind1]=min(abs(x-0.019));
    [~,ind2]=min(abs(x-0.016));
    [~,ind3]=min(abs(x-0.013));

    % Solve PDE system and compute RMS
    try
        solution = pdepe(0, pdefun, pdeic, pdebc, x, t);
        diff_U = sum((solution(t_indices, [ind1,ind2,ind3], 2) - data_U).^2, 1);
        value = sum(diff_U);
    catch
        value = Inf
    end
end
```