Thermodynamics modeling & prediction in a Big Added Area Manufacturing process

Industrial Engineering & Management

Discrete Technology and Production Automation research group

Engineering and Technology Institute Groningen

First supervisor:
prof. dr. ir. B. Jayawardhana

Second supervisor:
prof. dr. ir. J.M.A. Scherpen

Author:
T.H.W. (Twan) Huijbers
(S3476111)

July 16, 2021

Contents

List of Figures

List of Tables

Acknowledgements

I would first like to express my appreciation to my first supervisor prof. dr. ir. B. Jayawardhana, whose expertise gave me many insights in how to pursue my research. Our meetings helped me to construct the mathematical model. Furthermore, I learned much from the feedback you gave me on the mathematical formulation. I enjoyed listening to your feedback and explanations. I would also like to thank prof. dr. ir. J.M.A Scherpen for taking the time to assist me as second supervisor. In addition, I would like to thank Ir. Mark van der Staay and Ing. Damy Többen from NHL Stenden for assisting me during my research. The discussions we had weekly helped me to structure my research and put the models created in this research into practice. Finally, I would like to thank my family and friends who supported me during this research.

Abstract

The field of High Tech Systems and Materials (HTSM) in the Netherlands is developing rapidly. NHL Stenden, the University of Groningen and several stakeholders have started the Center of Expertise Smart Sustainable Manufacturing. Currently, a Smart Polymer Granulate 3D Welding System is being developed at the Centre of Expertise, which is located at NHL Stenden in Emmen. This system is a large 3D-printing robot, which can print geometries of large sizes. The process is called Big Additive Area Manufacturing (BAAM). A prototype has been built, in order to research the benefits and possibilities of BAAM. NHL Stenden faces challenges in predicting the thermal behaviour across the printed layers, which results in poor strength of the printed products and divergent geometries. This thesis focuses on *building a validated model of the heat transfer process in the smart granulate 3D welding system that predicts the thermal behaviour across the printed layers, in order to increase the strength and decrease the geometry deviation of the printed products.* The literature that is available is used as a basis to gain insight in how to construct a model of the heat transfer process. A 1-dimensional and 2-dimensional model is proposed that predicts the thermal behaviour of the heat transfer process across the printed layers. Both proposed models have been constructed with the Finite Difference Method and simulated with Discrete Event Simulation. Several experiments have been conducted to test the accuracy of the proposed models.

Chapter 1: Introduction

1.1 Problem context

Currently the field of High Tech Systems and Materials (HTSM) in the Netherlands is developing rapidly. The HTSM top sector produces various high-quality end products, components, semi-finished products, materials and services for a wide range of customers around the world. These products are efficient (mechatronics), precise (nano electronics, high precision manufacturing) and intelligent (embedded systems, software, sensors) [1]. The HTSM top sector has the ambition to generate 68.3 billion euros, provide employment for $460,000$ employees and export 74.6 billion euros of goods in 2020 [2]. This makes the HTSM top sector a significant large player in top sectors across the Netherlands. To make this top sector strive even more, the government, industry and universities are working together. Currently, several projects are rising in the Northern Netherlands: Regions of Smart Factories (RoSF), smart shared facilities and the sensor technology project ID3AS. The first smart shared facility and field lab called Technology Added is recently started in Emmen. For enabling knowledge creation and development, a center of expertise is setup in Emmen that works together in these innovations.

1.1.1 Center of Expertise Smart Sustainable Manufacturing

The Center of Expertise Smart Sustainable Manufacturing is established for creating and sharing knowledge in a network of different partners and is located at the NHL-Stenden university of applied sciences in Emmen. One of these partners is the Rijksuniversiteit Groningen (RUG). RUG focuses on the academic research that needs to be performed in the Centre of Expertise. A research and development program is started in 2018 at the Center of Expertise Smart Sustainable Manufacturing in the domains of Added Manufacturing and Robotics and End of Arm Tooling.

1.1.2 Big Area Additive Manufacturing

Additive manufacturing (AM) has been around for several years and has much potential to change the process of how components are currently fabricated. Reasons for this are the design flexibility, rapid prototyping and low initial production costs. Most AM technologies have in common that all products are sliced into discretized layers, which are predefined by the designer. These layers are then deposited on to each other from the ground. This results in a component that is built layer-by-layer. Every new deposited layer must bond to the previous deposited layer, in order to make the component a solid. Big Area Additive Manufacturing (BAAM) is making an entry to the AM market, since BAAM has the same advantages as conventional AM technologies. However, BAAM can be used to create significant larger products, for example the body work of a car. The disadvantage of BAAM is that it is much harder to control the environmental factors, such as ambient air, due to the larger area which needs to be controlled. Furthermore, due to the large-scale of BAAM, higher temperature gradients occur between the printed layers [3] and this can impact the internal stresses and geometry of the component significantly.

1.2 Defining the problem

NHL-Stenden Hogeschool has developed Smart Polymer Granulate 3D Welding System, which is a system that is based on the techniques used in BAAM. However, this system is still in its infancy, since the system is far from optimal and there are many challenges to overcome. Products are fabricated by depositing the polymer melt layer-by-layer (BAAM), which is shown in Figure 1.1. Currently, NHL-Stenden Hogeschool faces challenges in predicting and controlling the thermal behaviour inside the printed layers, which causes inequalities in the layer thickness and results in divergent product geometries. Furthermore, the strength of the printed part is depending on successful interdiffusion and re-entanglement of the polymer melt across the printed layers [4]. Changing the layer-time of a deposited layer can change the geometry and strength of the product drastically. A large layer-time results in poor strength and cracking. On the contrary, a small layer-time results in a collapse of the product, which can be seen in Figure 1.2. The reason for this is that the viscosity is too high, due to higher temperatures in the deposited layers. Therefore, it is important to predict the thermal behaviour over time of the polymer melt across the printed layers, such that a correct layer-time can be used. However, it is still unknown how to predict the thermal behaviour of the printed layers. Therefore, research must be conducted to gain insight in modelling the thermal behaviour of the printed layers.

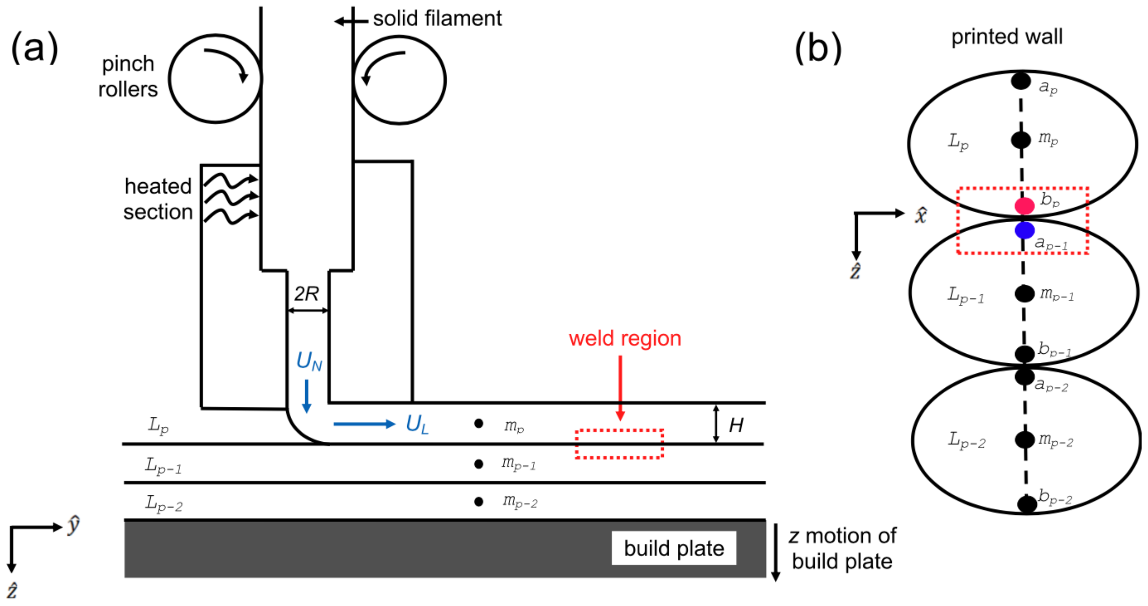


Figure 1.1: Schematic figure of layer-by-layer melt deposition, where (a) shows the nozzle with the melt that flows out of the nozzle at speed U_N and the printed layer is denoted as L_p. The center of the layer is denoted as m_p. (b) shows a section view of the wall geometry with the top layer a_p and bottom layer b_p. The figure is reproduced from [4].

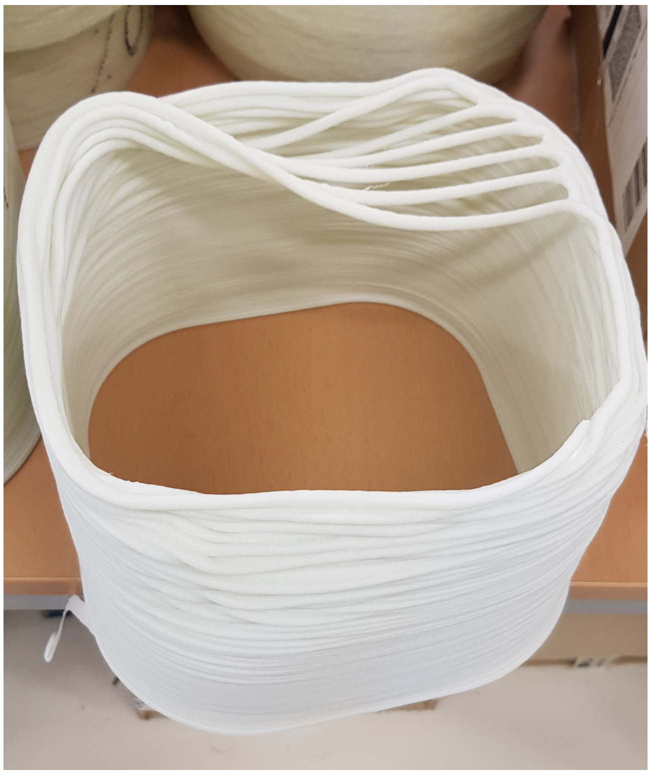


Figure 1.2: Small layer-time results in a collapse

Currently, the Robot Polymer 3D-Welding process is automated at the Centre of Expertise. NHL-Stenden has done several investments in equipment for this program. A Smart Polymer Granulate 3D Welding System is developed, with a Kuka KR16-2 manipulator and Xtrution granulate extruder that is shown in Figure 1.3. The purpose of these investments is to explore the possibilities of BAAM in a research area and gain information about BAAM for their business partners, since BAAM could potentially revolutionize the production process of their partners.

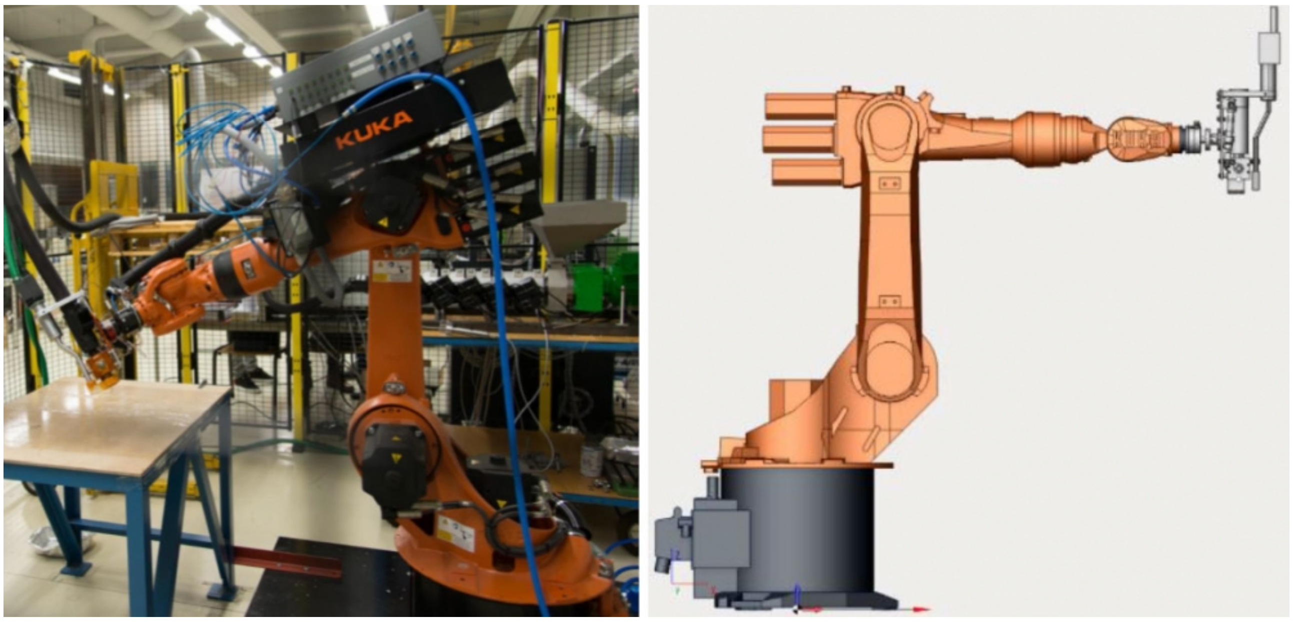


Figure 1.3: Smart Polymer Granulate 3D Welding System with a Kuka KR16-2 manipulator and Xtrution granulate extruder

1.2.1 System description

BAAM processes are still in its infancy and much research still needs to be conducted, in order to take BAAM to the next level where the quality and complexity of printed parts increase significantly. Therefore, the Center of Expertise is conducting academic research, guided by the University of Groningen. The development of BAAM and the research that is conducted by the Center of Expertise is considered to be the higher order system (see Figure 1.4). The Smart Polymer Granulate 3D Welding System is used to gather knowledge of BAAM processes through empirical research. Literature and empirical research are conducted for the development of this system. The knowledge of these developments is used by the Center of Expertise to give a scientific contribution to the development of BAAM.

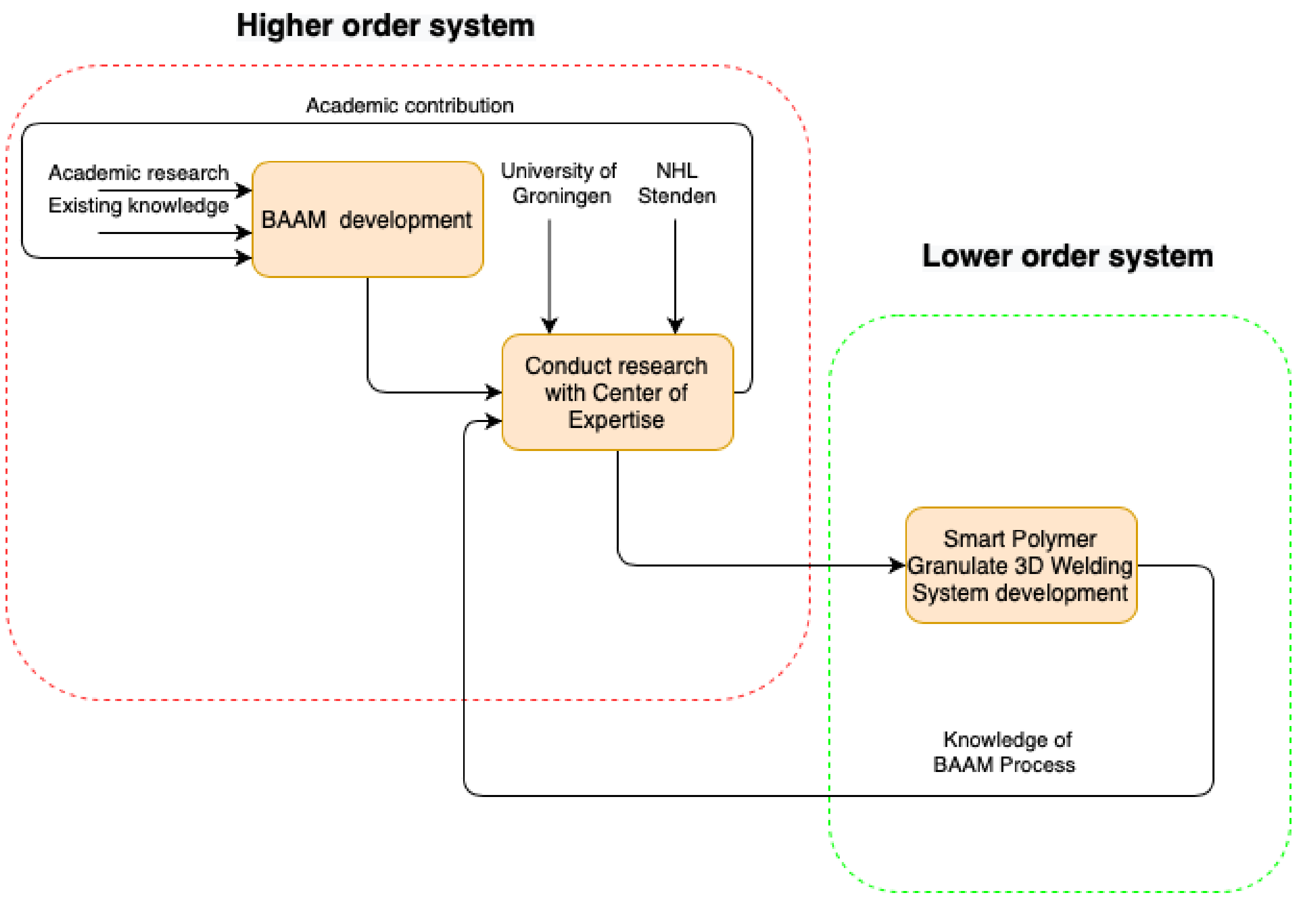


Figure 1.4: Higher and lower order system and the contribution of the lower order system to the higher order system

1.2.2 Lower order system

The lower order system is the development of the Smart Polymer 3D Welding system and is located at the Center of Expertise in Emmen. Several experiments can be conducted at the Centre of Expertise and the system architecture is explained below.

Computer Aided Manufacturing environment

A Computer Aided Manufacturing (CAM) environment is needed to convert 3-dimensional Computer Aided Design (CAD) files into a process plan that visualizes the fused filament deposition process. The program that is used for this, is called Sprutcam. Several parameters can be adjusted to optimize the quality of the printed part, such as the layer time and material feed rate.

Post-processor
Sprutcam generates a process plan of the fused filament deposition process. However, this process plan needs to be converted into Kuka Robot Language (KRL), such that the Kuka KR16-2 manipulator can carry out the process plan that is created in Sprutcam.

Xtrution granulate extruder
An Xtrution granulate extruder melts the granulate and injects the melted granulate under high pressure bar into a special insulated hose. This hose is connected to the end effector of the Kuka KR16-2 manipulator.

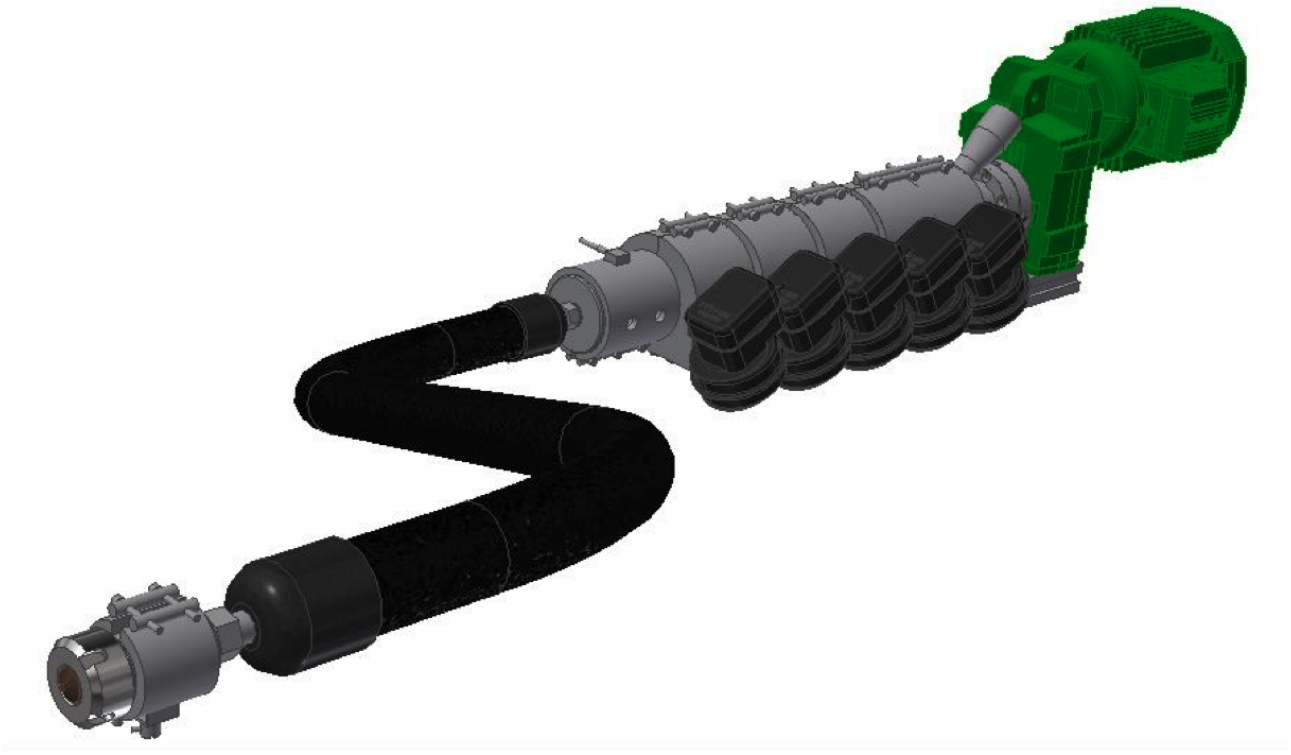


Figure 1.5: 3D CAD model of the Xtrution granulate extruder

Kuka KR16-2 manipulator
As mentioned above, NHL Stenden has invested in a Kuka KR16-2 manipulator. The manipulator is equipped with a hot extrusion nozzle at the end effector (see Figure 1.3), which deposits the melted granulate on the previous layer in the printed object.

Fused filament deposition process
The fused filament deposition process has the melted granulate and the manipulator movements as inputs (see Figure 1.1). Furthermore, environmental factors and process parameters that can affect the deposition process are considered as possible control variables. The output of the lower level system is a printed part that can be used to obtain knowledge about the BAAM process.

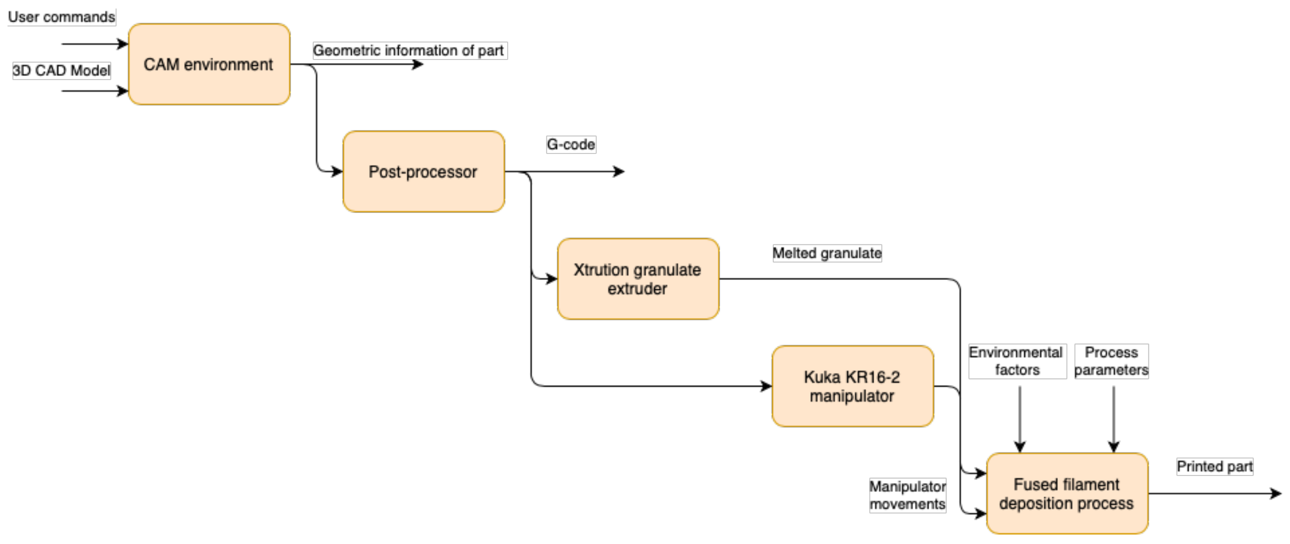


Figure 1.6: System architecture of lower order system

1.2.3 Problem statement

The problem statement for this research is stated as follows:
The thermal behaviour across the printed layers need to be predicted, in order to increase the strength and decrease the geometry deviation of the printed products.

1.2.4 Stakeholder analysis

Research is executed in cooperation with the lectorships Sustainable Polymers (PRE) and Circular Plastics and knowledge partners Wavin, NedCAM, HB3D Haarlem, Stevens Engineering Emmen (System Integrator), GeTech Westerbork (Mechanical Manufacturing). For the lower level system, which is the process control of the BAAM process at the Centre of Expertise, the stakeholders are the University of Groningen and the NHL-Stenden Hogeschool. Therefore, they are considered as primary stakeholders.

University of Groningen

The University of Groningen, specifically the Discrete Technology and Production Automation research group, guides the research performed in this thesis. The supervisors are part of the Discrete Technology and Production Automation research group and the results of this research contribute to the Production Automation sector.

NHL-Stenden Hogeschool

NHL-Stenden Hogeschool is improving the Smart Polymer 3D Welding system. This thesis contributes to the development of the Smart Polymer 3D Welding system, since the research performed in this thesis is focused on the thermal behaviour of the printed layers. Predicting the thermal behaviour of the printed layers can give insight in how to optimize the process parameters, such that an optimal temperature is achieved at a specific point in time.

As for the higher level problem, which is the development of a Big Area Added Manufacturing process as a whole, the stakeholders are Wavin, NedCAM, HB3D Haarlem, Stevens Engineering Emmen and Getech Westerbork. Since these stakeholders are mainly involved in the higher order problem, they are considered as secondary stakeholders.

Wavin

Wavin in Hardenberg delivers efficient water facilities for their customers, better sanitation and hygiene, climate proof cities and better building performances [5]. Wavin is interested in the knowledge that is obtained by developing the Smart Polymer 3D Welding system, since most of their products are made of polymers. Their products are produced with an injection molding technique, which are known for their high start-up costs, since the molds to produce products need to be developed first. This technique is not profitable for customized products with small batch sizes. Therefore, Wavin is interested in how to produce these customized products in a profitable way and BAAM could be a potential technique to achieve this.

NedCAM
Nedcam in Heerenveen produces production molds and milled products for the realization of 3D shapes [6]. Currently, Nedcam is about to invest in the development of their own BAAM Smart Polymer 3D Welding system. Therefore, all the knowledge that is gained at the Centre of Expertise is of significant interest for Nedcam.

HB3D Haarlem
HB3D in Haarlem is a company that is specialized in large scale 3D printing, also known as BAAM. HB3D has the knowledge within the area of plastic extrusion. HB3D claims that their expertise is to be a central hub of cooperation with multiple stakeholders that are specialized in BAAM processes [7]. This makes HB3D also a stakeholder for the Center of Expertise, since HB3D is interested in the knowledge that is obtained with the Smart Polymer 3D Welding system.

Stevens Engineering Emmen
Stevens Engineering is an independent Engineering Consultant and is active in the Northern region of the Netherlands. Stevens Engineering is specialized in several sectors, such as administration & Management, process & energy and manufacturing & machinery [8]. Stevens Engineering is the system integrator of the Centre of Expertise, which means that they are responsible for connecting all the components in the Smart Polymer 3D Welding system. Stevens Engineering has an interest in the knowledge that is obtained by combining all components in the system together.

Getech Westerbork
Getech Westerbork is specialized in the production of pipe fixtures for the automotive industry, especially the fixtures that check if the tolerances of a product are satisfied [9]. The advantage of Getech is that the Engineering, Production and Quality assessment is all performed without the involvement of third parties, which results in very short overall production lead times. Currently, Getech is exploring the opportunities of the Additive Manufacturing (AM), since a significant number of components can be made with AM. For the larger components that Getech is producing, BAAM could be a promising technique. Therefore, Getech has a significant stake in the development of the Smart Polymer 3D Welding system.

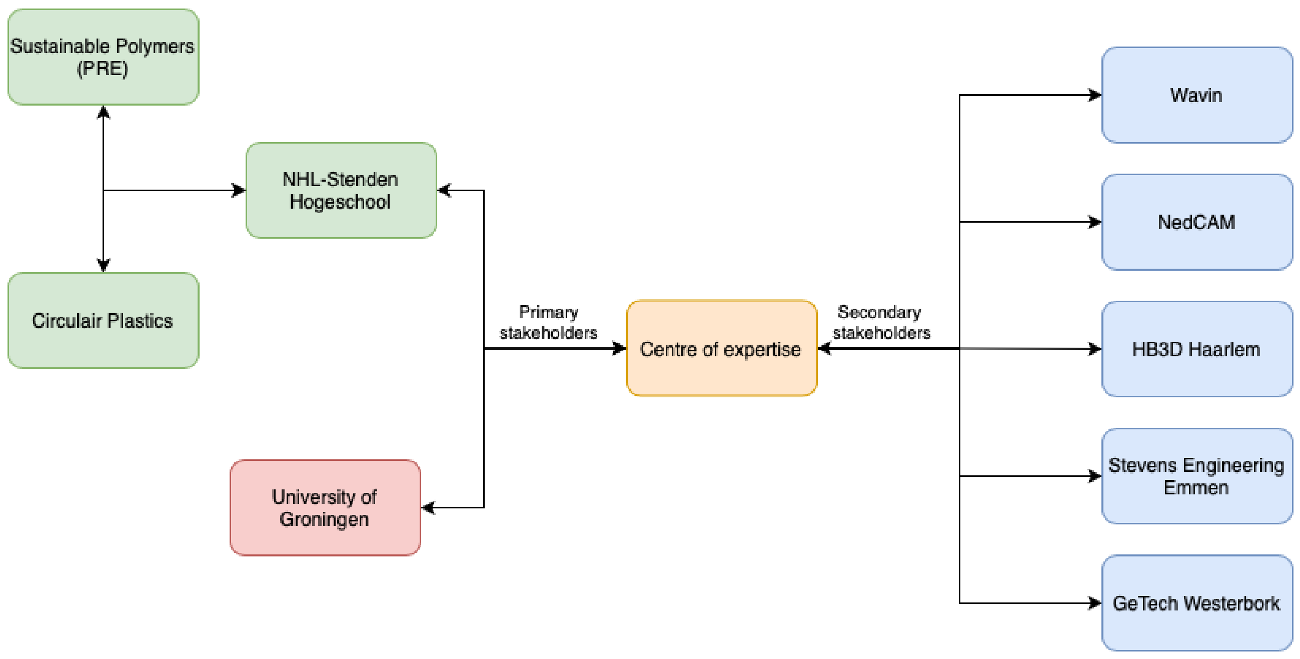


Figure 1.7: Schematic figure of stakeholder analysis

1.2.5 Research goal

The main research goal is to build a validated model of the heat transfer process in the smart granulate 3d welding system within a 21 week time frame that predicts the thermal behaviour across the printed layers, in order to increase the strength and decrease the geometry deviation of the printed products.

1.2.6 Scope

This research focuses on the numerical modeling of the thermal behaviour in the deposition process, in order to make a prediction about the thermal behaviour in the printed part. Therefore, research does not focus on the feedback control of the deposition process. Furthermore, the numerical model that will be constructed focuses on 1 and 2-dimensional heat transfer, since constructing a thermal model with 3-dimensional heat transfer is not feasible in the 21 week time frame. Furthermore, NHL-Stenden Hogeschool is currently only focusing on the improvement of symmetrical geometries. Therefore, a 3-dimensional heat transfer model is not necessarily needed, since layer times are equal for every layer, due to the symmetry of the printed geometries.

1.3 Methodology

1.3.1 Cycles

This research follows the Engineering cycle (Design cycle) that is proposed by Wieringa [10] and the Empirical cycle of van Aken [11]. First, the problem investigation stage is stated and includes the stakeholder analysis, goal, conceptual problem and causes. Second, the design that could treat the problem should be created. Third, the design needs to be validated, in order to see if the proposed design treats the problem. Thereafter, the problem is treated with the proposed design. At last, the implemented design is evaluated, in order to see if the problem is solved with the proposed design.

The empirical cycle of van Aken [11] is also used in this research, since the problem investigation phase and the validation phase involves empirical research. To gain insight in the deposition process of the layers, several experiments must be conducted. Furthermore, the model needs to be validated with experiments and hypotheses after the model is developed. The empirical research follows the empirical cycle, which is shown in Figure 8.2.

1.3.2 Research questions

To predict the thermal behaviour across the printed layers, two main research questions are drawn up, to gain insight in the deposition process of the printed layer. The sub questions help to answer the main research questions.

1. How to build a validated thermodynamics model that describes the deposition process?
   - (a) Which parameters can be adjusted to influence the thermal behaviour in the printed layers?
   - (b) Which environmental factors are important to decrease the inequalities and increase the strength of the product geometry?
   - (c) How to validate the thermodynamics model that is developed?
2. How to use the validated thermodynamics model to understand how the deposition process behaves?
   - (a) How can the validated thermodynamics model be implemented into the Smart Polymer Granulate 3D Welding System?

Chapter 2: Literature study

In this chapter, the results of the literature research will be reviewed. The influence of several environmental factors and machine parameters are discussed. This knowledge is important to create a realistic model. With this information one should be able to explain why a print is showing a certain non-desirable thermal behaviour and what measures should be taken to steer the thermal behaviour into a desired state. Furthermore, the theories that are used to model the deposition process are discussed.

2.1 Related work

Several discretized finite difference models have been developed to predict the thermal behaviour of a printed layer in the fused filament deposition process. Thomas and Rodriguez [12] developed a 2-dimensional model that had several nodes per layer. The results showed that temperature gradients become almost immediately negligible over the height and width of the printed layer. Compton et al. [3], presented a 1-dimensional thermal model that predicts the temperature behaviour of the deposited layers over time. Temperature gradients inside the layers were neglected in this thermal model. Bellehumeur et al. [13] modeled the bond formation between printed layers and assumed uniform temperature distribution along the height and width of the printed layer. Both Compton [3] and Bellehumeur [13] used constant heat transfer and convection coefficients for the thermal models. Furthermore, in these models, infinite layer lengths are assumed and both models are simplified to 1-dimensional heat transfer. Zhang et al. [14] developed an 3-dimensional thermal model based on voxelized elements with a predefined toolpath, in order to model the printing process of a small cube. However, the model did only include conduction between the elements and did not consider that the boundaries of the layers exchange heat through convection and radiation with the environment. Zhang et al. [14] and Bhandari et al. [15] developed a 3-dimensional thermal model that uses the toolpath input of a generated Gcode. The toolpath is discretized into small elements, in order to calculate the temperature distribution over time on that specific toolpath. However, as much as this approach looks promising, it is not necessarily needed to model the thermal behaviour of the printed parts which NHL Stenden is currently printing. Commercial Finite Element Analysis (FEA) software packages, such as ABAQUS ©, can also be used to model the thermal behaviour of a fused filament deposition process. The disadvantage of these software packages is that they are computationally heavy programs and take a significant amount of time to calculate, which increases even more when it should be used for a BAAM process.

2.2 Process parameters

According to Zhang et al. [14], the following process parameters can be controlled in the deposition process. printing nozzle temperature, environment temperature (T_∞, layer thickness Δz, layer width $\mathrm{W}^{\mathrm{int}}$, building orientation, geometry and printing speed (mm/s) e.g. layer time.

Furthermore, the top layer temperature ($\mathrm{T}^{\mathrm{top}}$) is also important. If $\mathrm{T}^{\mathrm{top}}$ is below a certain value when a new layer is deposited, cracking or warping can occur [3]. So, $\mathrm{T}^{\mathrm{top}} > \mathrm{T}^{\mathrm{m}}$ where T^{m} stands for the crystalline melting temperature of the material. Furthermore, the wall thickness ($\mathrm{W}^{\mathrm{int}}$) is according to [3] correlated to the influence of the layer time. Increasing the wall thickness causes a smaller cooling rate of the wall, which results in a larger time before $\mathrm{T}^{\mathrm{top}} < \mathrm{T}^{\mathrm{m}}$. Hence, the layer time can be increased before warping or cracking occurs.

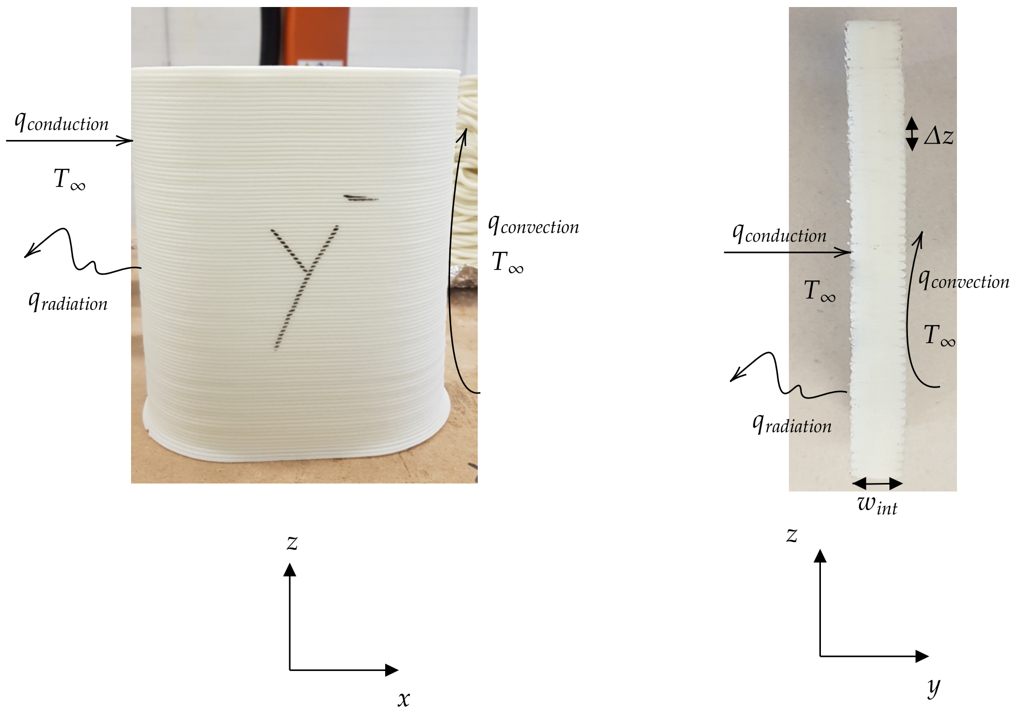


Figure 2.1: Presentation of printed product with process parameters and environmental factors Thermal radiation, Convection, Conduction. On the left a whole printed product is shown and on the right a cross-section of the printed product is depicted

2.3 Environmental factors

Temperature management is an important factor that can influence the inter-layer bonding strength [12],[16], polymer crystallinity [17], rheological behaviour [18],[19], deformation of the printed part [20],[21]. Poor temperature management indirectly influences the mechanical properties of the printed part, such as surface quality and printability [14]. Three natural phenomena act as environmental factors on the printed layers, which are Thermal radiation, Convection and Conduction between the layers. The explanation and computation of the environmental factors are stated below. Furthermore, to illustrate some process parameters and environmental factors, a presentation of a printed product that shows some process parameters and environmental factors is also stated below.

2.3.1 Thermal radiation

Heat transfer that is occurring between two objects that are not directly connected to each other (air between the two objects), is considered as thermal radiation. According to Ganji et al. [22], radiation differs from convection and conduction, since heat transfer by thermal radiation can also transfer energy through a vacuum. Therefore, it does not need a material to transfer energy through. For this research, only thermal radiation that is transferring energy through air is considered.

The law of radiation for an object that is exchanging heat with air is given by,

$$\mathrm{Q}_{\mathrm{radiation}} = \sigma e A(T_\infty^4 - T_f^4) \tag{2.1}$$

where, $\mathrm{Q}_{\mathrm{radiation}}$ = amount of heat transferred through radiation, σ = Stefan-Boltzmann constant, e = emissivity, A = surface area, T_f = absolute temperature of the object (K) and T_∞ is the environment temperature (K).

Emissivity
Emissivity is a number between zero and one, which represents the ability of an object or material to emit infrared energy. The energy that is emitted by the object or material indicates the temperature at a certain time. A low emissivity number represents an object or material that has a reflective surface. So, the more reflective the lower the emissivity number will become. For example, a mirror has an emissivity value of 0, since it is highly reflective. On the contrary, most organic objects, such as the human body have emissivity values close to 0.95.

Stefan-Boltzmann constant
The Stefan-Boltzmann constant, which is defined by σ, is a physical constant and is the indicated by the total intensity that is radiated over wavelengths. The intensity that is radiated increases when the temperature is increasing. The Stefan-Boltzmann constant has been evaluated by Blevin et al. [23] with an radiometer and the obtained value is $(5.6644 \pm 0.0075) \times 10^{-8}$ W m^{-2} K^{-4}.

2.3.2 Convection

Convective heat transfer occurs between an object and the surrounding air around that object. To calculate the rate of convection in an object, Newton's Law of Cooling is used.

Newton's law of convection is given by,

$$\mathrm{Q}_{\text{convection}} = hA\,(T_\infty - T_f) \tag{2.2}$$

where, $\mathrm{Q}_{\text{convection}}$ = amount of heat transferred through convection, A = area of object in m^2, h = heat transfer coefficient in $W/(m^2K)$, T_∞ = surface temperature of object in ℃, T_f = fluid temperature of the object in ℃,

According to Kosky et al. [24], there are two types of convective heat transfer. Namely, natural convection and forced convection, where natural convection occurs when an object is transferring heat to the surrounding area T_∞, which is caused by density differences in the fluid that are produced by temperature differences. For example, an object that is cooling down by transferring heat to the environment. Heat transfer that is created with a fan or pump is called forced convection and is used in numerous applications such as, air conditioning systems and heating systems.

Prandtl number
The Prandtl number is the ratio between the momentum of diffusivity and the thermal diffusivity in a fluid. The Prandtl number is approximately constant for most gases and is calculated by

$$\mathrm{Pr} = \frac{v}{\alpha_f} \tag{2.3}$$

where, Pr is the Prandtl number, v is the kinematic viscosity of the fluid and α_f is the thermal diffusivity of the fluid.

Grashof number
The Grashof number is the ratio of buoyant to viscous force acting on a fluid and is calculated by

$$\mathrm{Gr} = \frac{g\beta(T_s - T_\infty)L^3}{v^2} \tag{2.4}$$

where, Gr is the Grashof number, g is the gravitational constant, β, which is approximately $(\frac{1}{T_\infty})$, is the thermal expansion coefficient, T_s is the surface temperature of the fluid, T_∞ is the environment temperature, L is the vertical length and v is the kinematic viscosity.

Rayleigh number
The Rayleigh number is a dimensionless number that is associated with buoyancy driven flow, which is also known as natural convection. The term is needed for the calculation of natural convection and can be calculated by

$$\mathrm{Ra} = \mathrm{Gr}\mathrm{Pr} \tag{2.5}$$

where, Ra is the Rayleigh number, Gr is the Grashof number and Pr is the Prandtl number.

Nusselt number
The Nusselt number is the ratio between convective to conductive heat transfer across a boundary. According to Churchill et al. [25], the Nusselt number can be estimated for a flat vertical plate by

$$\overline{\mathrm{Nu}}^{1/2} = 0.825 + \frac{0.387\mathrm{Ra}^{1/6}}{[1 + (0.492/\mathrm{Pr})^{9/16}]^{8/27}} \tag{2.6}$$

where, $\overline{\mathrm{Nu}}$, Ra is the Rayleigh number and Pr is the Prandtl number.

Convection heat transfer coefficient
As explained above, there are two types of convection. To calculate the natural convection that is acting on a certain object, one should calculate the natural convection coefficient first. The natural convection coefficient can be calculated as

$$\bar{h} = \frac{\overline{\mathrm{Nu}}k_f}{L} \tag{2.7}$$

where, $\bar{h}$ is the convection heat transfer coefficient, k_f is the thermal conductivity coefficient of the fluid, $\overline{\mathrm{Nu}}$ is the Nusselt number and L is the wall height.

Forced convection coefficient
According to Bahrami [26], the forced convection coefficient at a local point in a laminar flow over a flat plate can be calculated with the Nusselt number, which is given by

$$\mathrm{Nu_x} = \frac{hx}{k} = 0.332\mathrm{Re}_x^{1/2}\,\mathrm{Pr}^{1/3} \quad \mathrm{Pr} \geq 0.6 \tag{2.8}$$

where, $N\,\mathrm{u_x}$ is the local Nusselt number, x is the distant from the top of the plate, Re is the Reynolds number and Pr is the Prandtl number. After obtaining the Nusselt number for forced convection, the forced convection is calculated in the same manner as 2.3.2.

$$h = \frac{\mathrm{Nu_x}k_f}{L} \tag{2.9}$$

where, k_f is the thermal conductivity coefficient of the fluid, $N\,\mathrm{u_x}$ is the local Nusselt number and x is the height from the top of the plate.

2.3.3 Conduction

Conduction is the transfer of heat through an object, which is causes by temperature differences between particles of that object. According to Shahidian et al. [27] and Ghassemi et al. [28], conduction is the transfer of energy from particles to other particles with less energy, which is caused by the interaction between those certain particles. Which means that conduction is only occurring between materials and does not exchange heat to the environment. Fourier's law can be used to calculate the heat transferred through conduction.

Fourier's law of conduction is given by,

$$\mathrm{Q}_{\mathrm{conduction}} = -kA\frac{\Delta T}{\Delta z} \tag{2.10}$$

where $\mathrm{Q}_{\mathrm{conduction}}$ = amount of heat transferred through conduction, k = thermal conductivity coefficient, Δz = distance between the two layers, A = surface that is conducting heat, ΔT = temperature difference between objects.

Thermal conductivity coefficient

According to Zhang [29], the thermal conductivity coefficient is a parameter that is depending on the physical properties of a material, temperature, water content and pressure on the specific material. The coefficient is represented by k and is measured in (W/mK) or $(W/m$°C). A material with a large k can be considered as a good heat conductor and a material with a small k is considered to be a good thermal insulator.

Specific heat capacity

According to Feidt [30], the specific heat capacity is the amount of heat that is put in or taken out of a material to heat or cool down a material by one degree. The specific heat capacity can be measured in $(kJkg^{-1}K^{-1})$ and is material dependent. For example, metals have a significantly lower specific heat capacity than water, which means that metals need less energy to heat or cool down by one degree.

Energy balance equation

To calculate the conduction for an elemental volume, the energy balance equation can be used to calculate the conduction in multiple dimensions. This can be significantly helpful for determining the heat transfer caused by conduction over multiple layers in the deposition process. According to Meseguer et al. [31], the rate at which energy is generated per volume of an object can be calculated with

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \dot{Q}_v = \rho c_p \frac{\partial T}{\partial t} \tag{2.11}$$

where, k is the thermal conductivity coefficient, ρ is the density of the material, c_p is the specific heat capacity, $\dot{Q}_v$ is the rate of heat transfer that is generated inside the elemental volume, $\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)$, $\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)$ and $\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)$ represent the conductive heat transfer in the x, y and z dimension respectively.

2.3.4 Biot number

The Biot Number is a dimensionless number and gives an indication of the interaction between conduction in an object and the convection at the surface. It can be used to determine the temperature history an object that is being heated or cooled down by convection at the surface. Zhang et al. [32] stated that objects with a Biot number smaller than 0.1 imply that the heat conduction inside the object is faster than the heat convection that flows out of the surface. Therefore, thermal gradients inside the object can be neglected. The Biot number can be calculated as

$$\text{Bi} = \frac{h}{k}L \tag{2.12}$$

where, k is the thermal conductivity of the object $(W/(mK))$, h is the convective heat transfer coefficient $(W/(m^2K))$ and L is the length of the object (m).

2.4 Modelling methods

There are two methods that are used together to model the thermal behaviour of the deposition process. Both methods discussed in the next two subsections.

2.4.1 Discrete event simulation

Discrete event simulation (DES) models move forward in time at discrete intervals which represent time. In each iteration actions are performed until the specified number of iterations is reached. The advantage of discrete event simulation is that for every event an exclusive action can be specified, which makes it a very powerful method for simulating complex processes. DES can be used for many applications, from Engineering applications to health care solutions.

According to Recktenwald [33], the Fourier stability criteria for a uniform grid is used to ensure the stability of a DES model and is given by

$$\Delta t < \frac{(\Delta x)^2}{2K} \tag{2.13}$$

where, $K = \frac{\lambda}{\rho c}$, Δx = grid spacing and Δt = time interval.

2.4.2 Finite difference method

According to Recktenwald [33], the finite difference method can be used to obtain several numerical solutions. This research will focus on applying the finite difference method to the energy balance equation, which is given in 2.11. To obtain the numerical solution, the energy balance differential equation is discretized by taking the Taylor Series. After that, the discretized equation can be solved numerically or with the help of DES. The result is a set of numerical solutions that are known at a finite number of grid points in the physical domain. The finite difference method uses a set of grid points (mesh), which represent the nodal locations of the nodes that are solved and is shown in the figure below. The number of those points can be changed, in order to increase the accuracy of the finite difference method. The main purpose of applying the finite difference method to the energy balance equation is to replace the continuous derivatives with discrete values that are related to the grid positions in the physical domain.

For example, to calculate the temperature in node (n, m) in Figure 2.2. The finite difference method can be applied to discretize the energy balance equation, which is calculated as follows. This example covers 2-dimensional (x, z) heat transfer and heat generation is not used in this case.

$$k\frac{\partial T}{\partial y^2} + k\frac{\partial T}{\partial z^2} = \rho c_p \frac{\partial T}{\partial t} \tag{2.14}$$

Taylor series expansion for y-dimension:

$$f(y + \Delta y) = f(y) + \Delta y f'(y) + \frac{\Delta y^2}{2} f''(y) + \sigma\left(\Delta y^3\right)$$
$$f(y - \Delta y) = f(y) - \Delta y f'(y) + \frac{\Delta y^2}{2} f''(y) - \sigma\left(\Delta y^3\right)$$

where, Δy , for example, denotes the nodal spacing of nodes between node m and $m-1$
Taylor series expansion for z-dimension:

$$f(z+\Delta z) = f(z) + \Delta z f'(z) + \frac{\Delta z^2}{2} f''(z) + \sigma\left(\Delta z^3\right)$$
$$f(z-\Delta z) = f(z) - \Delta z f'(z) + \frac{\Delta z^2}{2} f''(z) - \sigma\left(\Delta z^3\right)$$

where, Δz , for example, denotes the nodal spacing of nodes between node n and $n-1$

To determine the conduction of the interior nodes (m,n) (higher order terms cancel out),

$$f(y+\Delta y) + f(y-\Delta y) = 2f(y) + \Delta y^2 f''(y)$$

$$\frac{\partial^2 T(y,z,t)}{\partial y^2} = f''(y) = \frac{f(y-\Delta y) - 2f(y) + f(y+\Delta y)}{\Delta y^2}$$

If one sets $f(y-\Delta y) = T_{m-1,n},\ f(y) = T_{m,n},\ f(y+\Delta y) = T_{m+1,n}$ then,

$$\frac{T(y,z,t)_{m-1,n} - 2T(y,z,t)_{m,n} + T(y,z,t)_{m+1,n}}{\Delta y^2}$$

For Δz direction,

$$f(z+\Delta z) + f(z-\Delta z) = 2f(z) + \Delta z^2 f''(z)$$
$$\frac{\partial^2 T(y,z,t)}{\partial z^2} = f''(z) = \frac{f(z-\Delta z) - 2f(z) + f(z+\Delta z)}{\Delta z^2}$$

If one sets $f(z-\Delta z) = T_{m,n-1},\ f(z) = T_{m,n},\ f(z+\Delta z) = T_{m,n+1}$ then,

$$\frac{T_{m,n+1}(t) - 2T_{m,n}(t) + T(t)_{m,n+1}(t)}{\Delta z^2}$$

Substituting this in the energy balance heat equation gives,

$$\frac{k}{\Delta y^2}(T_{m+1,n}(t) - 2T_{m,n}(t) + T_{m,n-1}(t))$$
$$+\frac{k}{\Delta z^2}\left(T_{n+1,m}(t) - 2T_{m,n}(t) + T_{m-1,n}(t)\right) = \rho c_p \frac{\partial T}{\partial t}$$

If $\Delta z = \Delta y$, one obtains

$$k\left(T_{m+1,n}(t) - 4T_{m,n}(t) + T_{m,n-1}(t) + T_{n+1,m}(t) + T_{m-1,n}(t)\right) = \rho c_p \frac{\partial T}{\partial t}$$

For simplicity, steady state is assumed and therefore $\rho c_p \frac{\partial T}{\partial t} = 0$, so that

$$T_{m,n} = \frac{(T_{m+1,n} + T_{m,n-1} + T_{n+1,m} + T_{m-1,n})}{4}$$

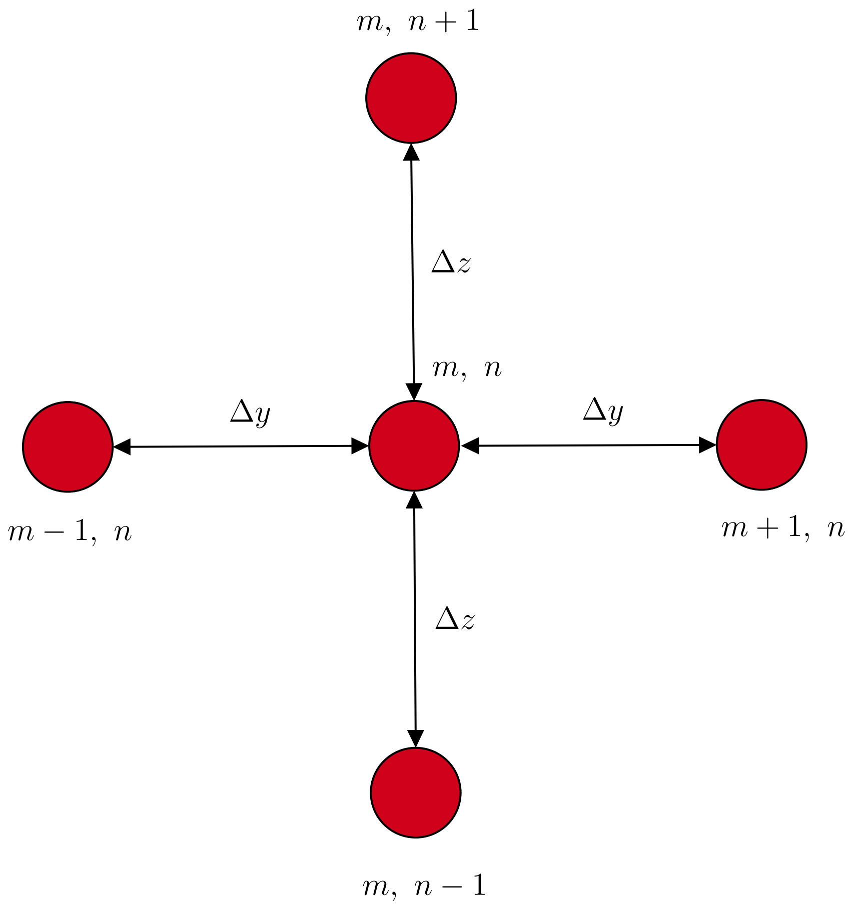


Figure 2.2: Schematic representation of nodal spacing $(\Delta y, \Delta z)$ *between nodes* (m, n)

Chapter 3: Numerical model

This chapter covers the two numerical thermal models that are used to simulate the thermal behaviour of the deposited layers over time. The models are based on the discrete event simulation theory and finite difference method, which are explained in the previous chapter. First, the assumptions that are made are stated. Second, the 1-dimensional numerical thermal model is explained. Thereafter, the 2-dimensional numerical thermal model is explained. At last, the formulas to simulate the forced convection that is created by the air-gun are stated.

3.1 Assumptions

To realize the thermal model, several assumptions are made. The first assumption is that the temperature is uniform distributed for each cross-section of a layer, due to the small height and width of the layer. Therefore, only one node per layer is used for temperature calculation. Second, the emissivity of the printing material Hostacom G3 N01 is determined experimentally. Third, a fixed value for the thermal conductivity (k) is used for the thermal model, since the thermal conductivity of non-metals only changes a very small percentage when the temperature of the material changes. Fourth, the friction between the layers is neglected. Fifth, a fixed ambient temperature is assumed, since modelling the fluid goes beyond the scope of this research. At last, the radiation of the heated nozzle on the printed layers is neglected.

3.2 1-dimensional numerical thermal model

This section covers the derivation of the 1-dimensional numerical thermal model, which is constructed with the Finite Difference Method and simulated by Discrete Event Simulation. Python is used for the simulation of the 1-dimensional numerical thermal model. The 1-dimensional model, which is illustrated in paragraph 3.2.2, is a cross-section of the printed layers. In each cross-section of a layer, a node is placed where the temperature in ℃ is calculated. First, the process of the numerical model is explained through a flowchart. After this, there is elaborated on the derivation of the formulas.

3.2.1 Process of the 1-dimensional numerical thermal model

The flowchart below gives a schematic global representation of the 1-dimensional numerical thermal model. The model uses a for loop to run the simulation several times, until the specified number of layers n_l is achieved. The input parameters define the results of the simulation. The input parameters that can be changed are stated in the table below.

Input parameters
Density of material, ρ $[kg/m^3]$
Crystalline melting temperature, T_m [°C]
Emissivity, ε
Thermal conductivity, $k[W/m.K]$
Specific heat capacity, c_p $[J/kg.K]$
Deposition temperature, T_{dep}[°C]
Ambient temperature, T_∞ [°C]
Δz, $[mm]$
$\mathrm{w_{int}}$, $[mm]$
Temperature air-gun, $\mathrm{T_{airgun}}$[°C]
Number of layers, n_L
Time step, Δt
Layer time, t_p $[mm]$
Printing speed $\mathrm{v_{airgun}}$ $[mm/s]$

Table 3.1: Input parameters of 1-dimensional numerical thermal model

After the input parameters are set to the correct values, the simulation is started. The model is running over a specified number of layers. The first condition determines if the current layer (n) is > 4, if this is the case, then the air-gun is turned on. Therefore, the parameters for the forced convection must be calculated, otherwise the simulation can be started immediately. The calculation of the parameters is discussed in section 3.4. For each layer, the simulation iterates through a specified number of iterations, which is defined by

$$n_t = t_p/\Delta t \tag{3.1}$$

where $n_t \in \mathbb{Z}$

During the deposition process an air-gun, which is mounted on the printing nozzle, is moving over the layers for a specified number of iterations. When the air-gun is moving over the simulated area, forced convection must be used on top of natural convection. Therefore, an if statement is used to ensure that this happens during a specified period. After this, there is checked if all the layers are deposited, in mathematical terms if $n = n_l$. When this is true, the cooldown period is activated, which means that the simulation is iterating for a specified number of iterations, without the deposition of new layers. Finally, the information (graphs, tables) are exported to a PDF-file.

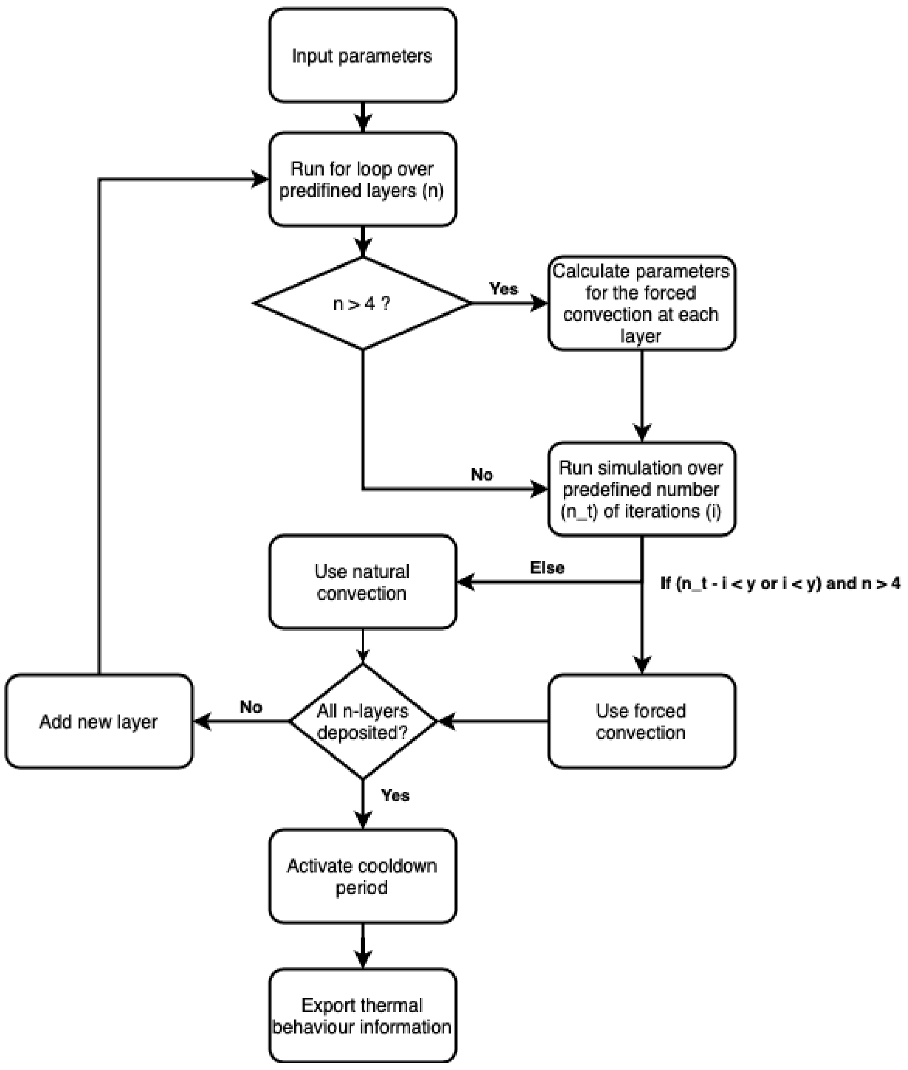

Input parameters
Run for loop over predifined layers (n)
n > 4 ?
Yes
Calculate parameters for the forced convection at each layer
No
Run simulation over predefined number (n_t) of iterations (i)
Else
If (n_t - i < y or i < y) and n > 4
Use natural convection
Use forced convection
All n-layers deposited?
No
Add new layer
Yes
Activate cooldown period
Export thermal behaviour information

Figure 3.1: Flowchart of the 1-dimensional numerical thermal model

3.2.2 Derivation of 1-dimensional numerical thermal model

The 1-dimensional numerical thermal model is derived from the energy balance equation. During the printing process, three natural phenomena (Thermal radiation, Convection, Conduction) occur that cause the dissipation of heat from the layers to the surrounding air and neighbour layers. The theory of these three phenomena are discussed in section 2.3. Therefore, they are not discussed in this section again. For the derivation an example is used, which is depicted in the figure below, in the derivation there is solved for node T_n.

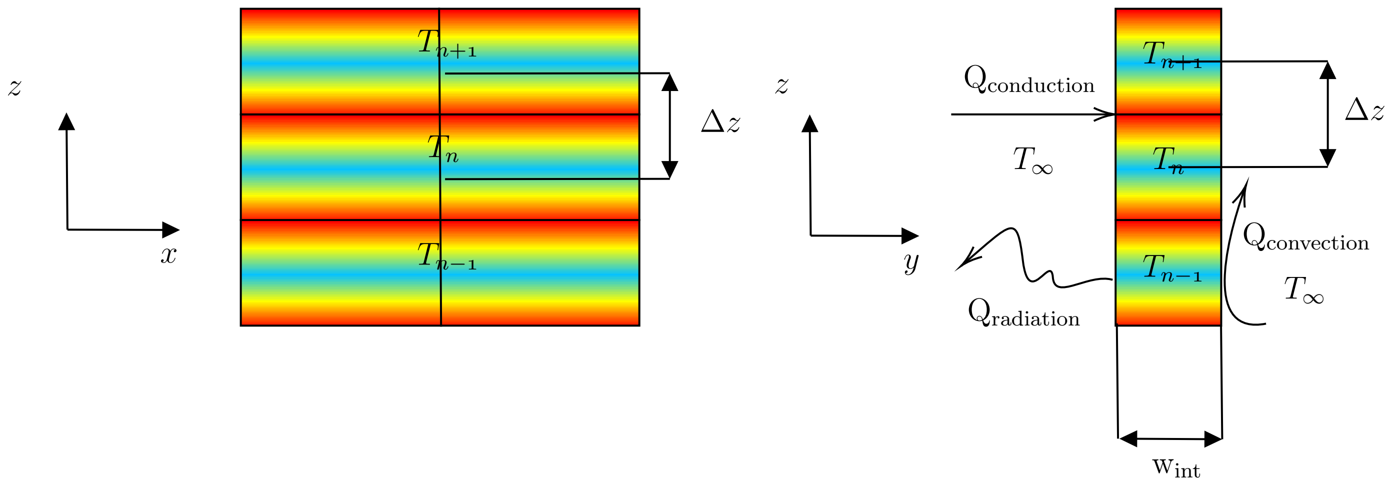


Figure 3.2: 1-dimensional model with three layers

First, the energy balance equation is simplified to a 1-dimensional state by

$$\rho c_p \frac{\partial T(x,y,z,t)}{\partial t} = k\left(\frac{\partial^2 T(x,y,z,t)}{\partial x^2} + \frac{\partial T(x,y,z,t)^2}{\partial y^2} + \frac{\partial T(x,y,z,t)^2}{\partial z^2}\right) \tag{3.2}$$

this results in

$$\rho c_p \frac{\partial T(z,t)}{\partial t} = k\left(\frac{\partial^2 T(z,t)}{\partial z^2}\right) \tag{3.3}$$

where, $T(z,t)$ is the temperature on location z at time t, k is the thermal conductivity coefficient, ρ is the density of the material, c_p is the specific heat capacity.

The dissipation of heat is caused by the three natural phenomena Conduction ($\mathrm{Q}_{\text{conduction}}$), Convection ($\mathrm{Q}_{\text{convection}}$) and Thermal radiation ($\mathrm{Q}_{\text{radiation}}$). These natural phenomena are substituted into 3.2.2. First, we will use the following first order approximation and we set $T(z,t)$ to $T_n(t)$, where n denotes the layer number

$$\rho c_p \frac{\partial T(z,t)}{\partial t} \approx \rho c_p \frac{T_n(t+1) - T_n(t)}{\Delta t} \tag{3.4}$$

$$\mathrm{Q}_{\text{conduction}}(z,t) \approx \frac{k w_{\text{int}}}{\Delta z}\left(T_{n-1}(t) - T_n(t)\right) + \frac{k w_{\text{int}}}{\Delta z}\left(T_{n+1}(t) - T_n(t)\right) \tag{3.5}$$

$$\mathrm{Q}_{\text{convection}}(z,t) \approx 2h\Delta z\left(T_\infty - T_n(t)\right) \tag{3.6}$$

$$\mathrm{Q}_{\text{radiation}}(z,t) \approx \varepsilon\sigma\Delta z\left(\left(K + T_\infty\right)^4 - \left(K + T_n(t)\right)^4\right) \tag{3.7}$$

Substituting the three natural phenomena this can be rewritten

$$\rho c_p w_{int}\Delta z \frac{T_n(t+1) - T_n(t)}{\Delta t} = \mathrm{Q}_{\text{conduction}}(z,t) + \mathrm{Q}_{\text{convection}}(z,t) + \mathrm{Q}_{\text{radiation}}(z,t) \tag{3.8}$$

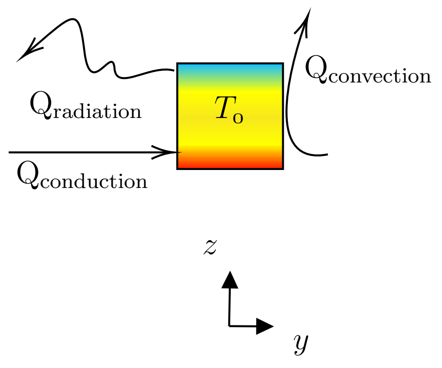
$\mathrm{Q}_{\text{conduction}}$, $\mathrm{Q}_{\text{convection}}$ and $\mathrm{Q}_{\text{radiation}}$ are obtained from the literature review and represent Fourier's law of conduction Newton's law of cooling and Thermal radiation respectively.

Below a schematic overview of the deposition process is depicted. To calculate the current temperature in each layer, 3.2.2 is rewritten into

$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_n}}(t) + \mathrm{Q}_{\mathrm{conv_n}}(t) + \mathrm{Q}_{\mathrm{rad_n}}(t)\}_2\, \Delta t}{\rho c_p w_{int} \Delta z} + T_n(t) \tag{3.9}$$

where, $T_n(t+1)$ is the current temperature and $\mathrm{Q}_{\mathrm{conduction}}(z,t)$, $\mathrm{Q}_{\mathrm{convection}}(z,t)$, $\mathrm{Q}_{\mathrm{radiation}}(z,t)$ are set to $\mathrm{Q}_{\mathrm{cond_n}}(t)$, $\mathrm{Q}_{\mathrm{conv_n}}(t)$, $\mathrm{Q}_{\mathrm{rad_n}}(t)$ respectively, such that the terms represent the current layer (n) at a certain point in time. The notations for Conduction Convection and Radiation are also shortened, such that the formulas do not take up much space.

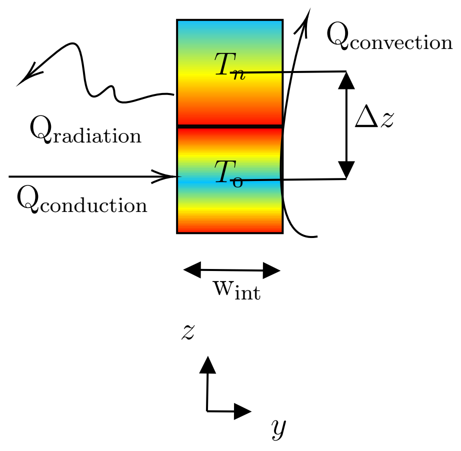
We denote the node of the first layer (n) as 0, which is the layer placed directly on the printing bed. The temperature inside this node can be calculated by


$\mathrm{Q}_{\mathrm{radiation}}$
T_0
$\mathrm{Q}_{\mathrm{convection}}$
$\mathrm{Q}_{\mathrm{conduction}}$
z
y

$$T_0(t) = \frac{(\mathrm{Q}_{\mathrm{cond_0}}(t) + \mathrm{Q}_{\mathrm{conv_0}}(t) + \mathrm{Q}_{\mathrm{rad_0}}(t)\}2\Delta t}{\rho c_p \mathrm{w_{int}} \Delta z} + T_0(t)$$

$$\begin{aligned}
&\textbf{If } \mathbf{n = 0} \\
&\mathrm{Q}_{\mathrm{rad_0}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w_{int}}\right)\left((K + T_\infty)^4 - (K + T_0(t))^4\right) \\
&\mathrm{Q}_{\mathrm{cond\ 0}}(t) = \frac{2k w_{\mathrm{int}}}{\Delta z}\left(T_b - T_0(t)\right) \\
&\mathrm{Q}_{\mathrm{conv\ 0}}(t) = h\left(\Delta z + \mathrm{w_{int}}\ \right)\left(T_\infty - T_0(t)\right)
\end{aligned} \tag{3.10}$$

After a new layer is deposited, then the first layer conducts heat with the newly deposited layer. Therefore, the conditions of the first layer change, which can be seen in the derivations of 3.2.2. The temperature inside the nodes is calculated by

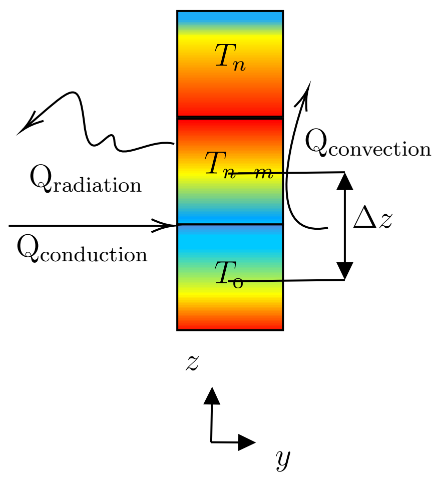

$\mathrm{Q}_{\mathrm{convection}}$
T_n
Δz
$\mathrm{Q}_{\mathrm{radiation}}$
T_0
$\mathrm{Q}_{\mathrm{conduction}}$
$\mathrm{w_{int}}$
z
y

$$T_n = \frac{(\mathrm{Q}_{\mathrm{cond_n}}(t) + \mathrm{Q}_{\mathrm{conv_n}}(t) + \mathrm{Q}_{\mathrm{rad_n}}(t)\}2\Delta t}{\rho c_p \mathrm{w_{int}} \Delta z} + T_n(t)$$

$$T_0(t) = \frac{(\mathrm{Q}_{\mathrm{cond_0}}(t) + \mathrm{Q}_{\mathrm{conv_0}}(t) + \mathrm{Q}_{\mathrm{rad_0}}(t)\}2\Delta t}{\rho c_p \mathrm{w_{int}} \Delta z} + T_n(t)$$

$$
\begin{aligned}
&\text{If } \mathbf{n} = 1\\
&\mathrm{Q}_{\mathrm{rad}_1}(t) = \varepsilon\sigma(\Delta z + \mathrm{w}_{\mathrm{int}})\left((K + T_\infty)^4 - (K + T_1(t))^4\right)\\
&\mathrm{Q}_{\mathrm{cond}_1}(t) = \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}(T_0(t) - T_1(t))\\
&\mathrm{Q}_{\mathrm{conv}_1}(t) = h(\Delta z + \mathrm{w}_{\mathrm{int}})(T_\infty - T_1(t))\\
&\mathrm{Q}_{\mathrm{rad}_0}(t) = \varepsilon\sigma\Delta z\left((K + T_\infty)^4 - (K + T_0(t))^4\right)\\
&\mathrm{Q}_{\mathrm{cond}_0}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}(T_b - T_0(t)) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}(T_1(t) - T_0(t))\\
&\mathrm{Q}_{\mathrm{conv}_0}(t) = \Delta z h(T_\infty - T_0(t))
\end{aligned}
\tag{3.11}
$$

After another new layer is deposited, the middle T_{n-m} layer conducts heat with both the top layer T_n and the bottom first layer T_0. Therefore, the conditions of the middle node change, which can be seen in the derivations of 3.2.2. In the numerical thermal model a for loop iterates through the middle nodes in the range of $[1, m]$, where m = n-1. After this, all conditions for the nodes (top, middle and bottom) are included in the model. Therefore, the schematic process stops here. The temperatures can be calculated by



T_n
T_{n-m}
T_0
$\mathrm{Q}_{\mathrm{convection}}$
$\mathrm{Q}_{\mathrm{radiation}}$
$\mathrm{Q}_{\mathrm{conduction}}$
Δz
z
y

$$T_n(t) = \frac{(\mathrm{Q}_{\mathrm{cond}_\mathrm{n}}(t) + \mathrm{Q}_{\mathrm{conv}_\mathrm{n}}(t) + \mathrm{Q}_{\mathrm{rad}_\mathrm{n}}(t)\}2\varDelta t}{\rho c_p \mathrm{w}_{\mathrm{int}}\varDelta z} + T_n(t)$$

$$T_{n-m}(t) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-m}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-m}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-m}}}(t)\}\varDelta t}{\rho c_p \mathrm{w}_{\mathrm{int}}\varDelta z} + T_{n-m}(t)$$

$$T_0(t) = \frac{(\mathrm{Q}_{\mathrm{cond}_0}(t) + \mathrm{Q}_{\mathrm{conv}_0}(t) + \mathrm{Q}_{\mathrm{rad}_0}(t)\}2\varDelta t}{\rho c_p \mathrm{w}_{\mathrm{int}}\varDelta z} + T_0(t)$$

$$
\begin{aligned}
&\text{If } \mathbf{n} > \mathbf{1}\\
&\mathrm{Q}_{\mathrm{rad}_\mathrm{n}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K + T_\infty)^4 - (K + T_n(t))^4\right)\\
&\mathrm{Q}_{\mathrm{cond}_\mathrm{n}}(t) = \frac{\mathrm{kw}_{\mathrm{int}}}{\Delta z}\left(T_{n-1}(t) - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{conv}_n}(t) = h\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n-m}}}(t) = \varepsilon\sigma(\Delta z)\left((K + T_\infty)^4 - (K + T_{n-m}(t))^4\right)\\
&\mathrm{Q}_{\mathrm{cond\ n-m}}(t) = \frac{\mathrm{kw}_{\mathrm{int}}}{\Delta z}\left(T_{n-m-1}(t) - T_{n-m}(t)\right) + \frac{\mathrm{kw}_{\mathrm{int}}}{\Delta z}\left(T_{n-m+1}(t) - T_{n-m}(t)\right)\\
&Q_{\mathrm{conv}_{\mathrm{n-m}}}(t) = 2\Delta z h\left(T_\infty - T_{n-m}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_0}(t) = \varepsilon\sigma\Delta z\left((K + T_\infty)^4 - (K + T_0(t))^4\right)\\
&\mathrm{Q}_{\mathrm{cond}_0}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_b - T_0(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_1(t) - T_0(t)\right)\\
&\mathrm{Q}_{\mathrm{conv}_0}(t) = \Delta z h\left(T_\infty - T_0(t)\right)
\end{aligned}
\tag{3.12}
$$

After this, the derivation stops, since the conditions of the top, middle and bottom nodes do not change anymore. To recall, temperatures at locations $T(z,t)$ are rewritten to $T_n(t)$, in order to represent the layer number.

3.3 2-dimensional numerical thermal model

This section explains the process and derivation of the 2-dimensional numerical thermal model. This model is also simulated in Python. First, a flowchart is stated. After that, the derivation of the model is explained with an example.

3.3.1 Process of the 2-dimensional numerical thermal model

The same input parameters have been used as in table 3.1. Except in the 2-dimensional model a new input parameter has been added. Namely, Δy, such that the extra dimension is created. This parameter has been set equal to $\mathrm{w}_{\mathrm{int}}$, which means that the nodal spacing is equal to the width of the layer. Furthermore, the same structure for programming has been used, which is shown in the flowchart. The difference is that the condition for the air-gun activation is shifted to $n > 9$, since the 2-dimensional model consists over 2 layers in the y-direction, which can be seen in Figure 3.4.

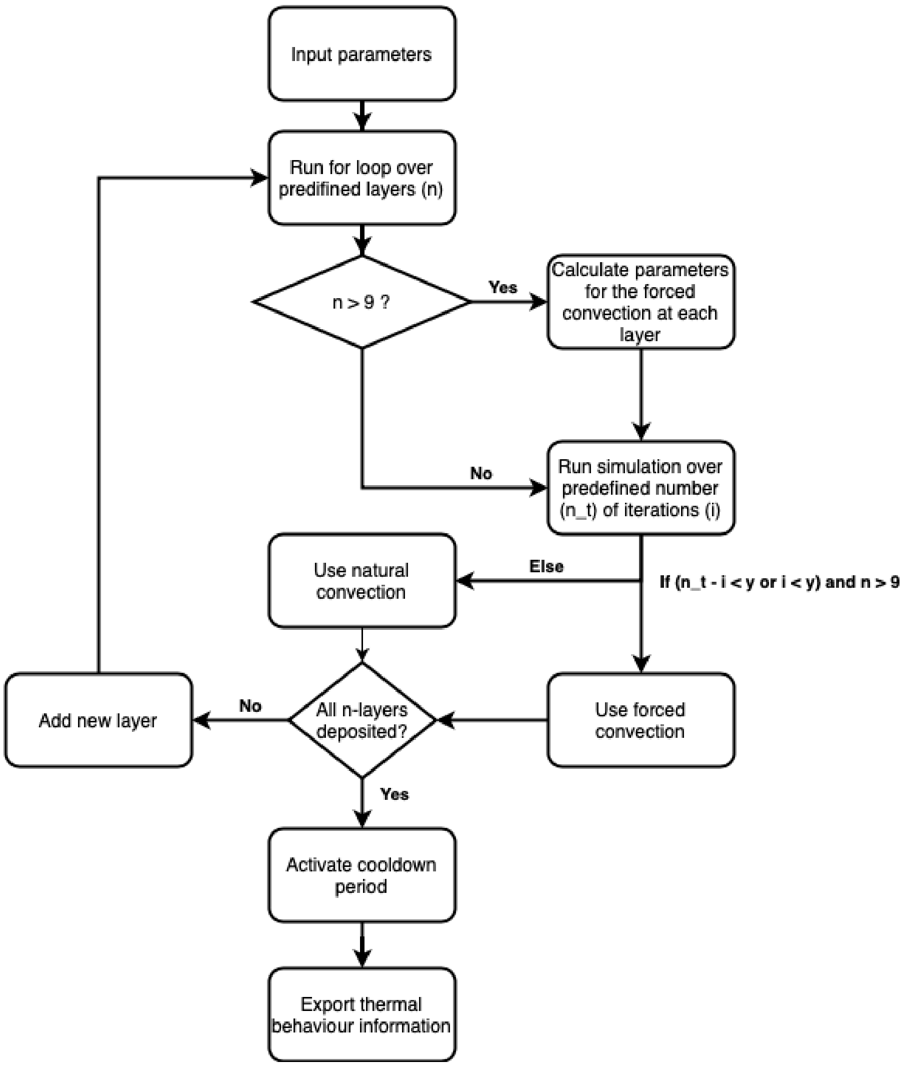

Input parameters
Run for loop over predifined layers (n)
n > 9 ?
Yes
Calculate parameters for the forced convection at each layer
No
Run simulation over predefined number (n_t) of iterations (i)
Else
If (n_t - i < y or i < y) and n > 9
Use natural convection
Use forced convection
All n-layers deposited?
No
Add new layer
Yes
Activate cooldown period
Export thermal behaviour information

Figure 3.3: Flowchart of the 2-dimensional numerical thermal model

3.3.2 Derivation of 2-dimensional numerical thermal model

The 2-dimensional model is derived in the same manner as the 1-dimensional model, which is based on the energy balance equation and considers Thermal radiation, Convection and

Conduction. However, the 2-dimensional model simulates the thermal behaviour of two deposited layers in the y-direction, instead of one deposited layer in the y-direction. For the derivation an example is used, which is depicted in the figure below, in the derivation there is solved for node T_n.

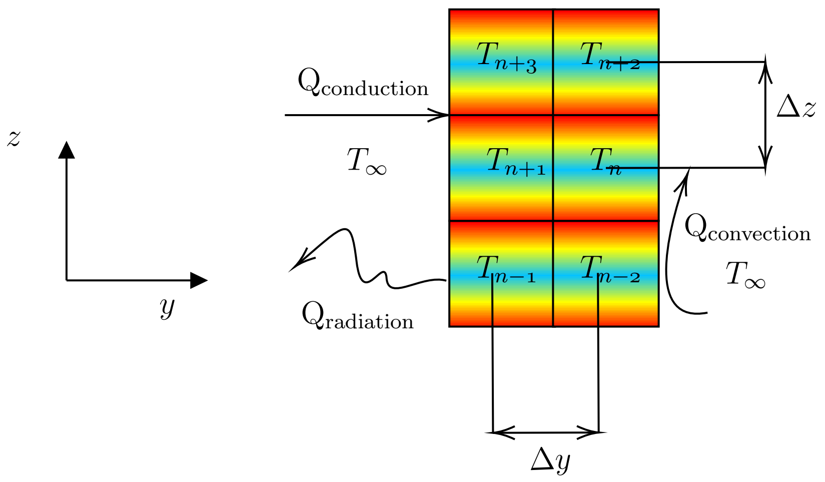


Figure 3.4: 2-dimensional model with 3 layers in the z-direction and two layers in the y direction

First, the energy balance equation is simplified to a 2-dimensional state by rewriting

$$\rho c_p \frac{\partial T(x,y,z,t)}{\partial t} = k\left(\frac{\partial^2 T(x,y,z,t)}{\partial x^2} + \frac{\partial T(x,y,z,t)^2}{\partial y^2} + \frac{\partial T(x,y,z,t)^2}{\partial z^2}\right) \tag{3.13}$$

this results in

$$\rho c_p \frac{\partial T(y,z,t)}{\partial t} = k\left(\frac{\partial^2 T(y,z,t)}{\partial y^2} + \frac{\partial^2 T(y,z,t)}{\partial z^2}\right) \tag{3.14}$$

where, $T(y,z,t)$ is the temperature on location y,z at time t, k is the thermal conductivity coefficient, ρ is the density of the material, c_p is the specific heat capacity.

The dissipation of heat is caused by the three natural phenomena Conduction (Q$_{\text{conduction}}$), Convection (Q$_{\text{convection}}$) and Thermal radiation Q$_{\text{radiation}}$. These natural phenomena are substituted into 3.3.2. First, we will use the following first order approximation and we set $T(y,z,t)$ to $T_n(t)$, where n denotes the layer number

$$\rho c_p \frac{\partial T(y,z,t)}{\partial t} \approx \rho c_p \frac{T_n(t+1) - T_n(t)}{\Delta t} \tag{3.15}$$

The three natural phenomena are stated below

$$\text{Q}_{\text{conduction}}(y,z,t) = \frac{k\text{w}_{\text{int}}}{\Delta z}\left(T_{n-2}(t) - T_n(t)\right) + \frac{k\text{w}_{\text{int}}}{\Delta z}\left(T_{n+2}(t) - T_n(t)\right) + \frac{k\text{w}_{\text{int}}}{\Delta y}\left(T_{n-1}(t) - T_n(t)\right) \tag{3.16}$$

$$\text{Q}_{\text{convection}}(y,z,t) = h\Delta z\left(T_\infty - T_n(t)\right) \tag{3.17}$$

$$\text{Q}_{\text{radiation}}(y,z,t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_n(t))^4\right) \tag{3.18}$$

Substituting the three natural phenomena for each layer (n) this can be rewritten into

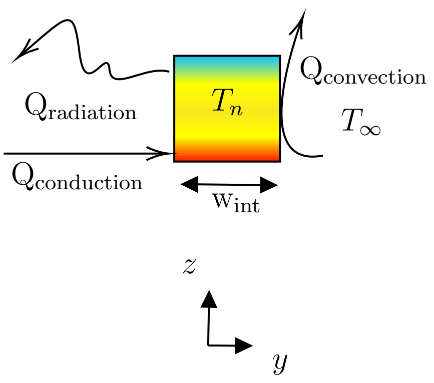
$$\rho c_p \Delta y \Delta z \frac{T_n(t+1) - T_n(t)}{\Delta t} = \mathrm{Q}_{\text{conduction}}\,(y,z,t) + \mathrm{Q}_{\text{convection}}\,(y,z,t) + \mathrm{Q}_{\text{radiation}}\,(y,z,t) \quad (3.19)$$

Below a schematic overview of the deposition process is depicted. To calculate the current temperature in each layer, 3.3.2 is rewritten into

$$T_n(t+1) = \frac{\mathrm{Q}_{\text{cond}_\text{n}}(\mathrm{t}) + \mathrm{Q}_{\text{conv}_\text{n}}(\mathrm{t}) + \mathrm{Q}_{\text{rad}_\text{n}}(\mathrm{t})\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t) \quad (3.20)$$

where, $T_n(t+1)$ is the current temperature and $\mathrm{Q}_{\text{conduction}}$(y,z,t), $\mathrm{Q}_{\text{convection}}$(y,z,t), $\mathrm{Q}_{\text{radiation}}$(y,z,t) are set to $\mathrm{Q}_{\text{cond}_\text{n}}$, $\mathrm{Q}_{\text{conv}_\text{n}}$, $\mathrm{Q}_{\text{rad}_\text{n}}$ respectively, such that the terms represent the current layer (n) at a certain point in time. The notations for Conduction Convection and Radiation are also shortened, such that the formulas do not take up much space.

The first layer (n) in the 2-dimensional model is also denoted as 0, so n starts from 0. The derivations can be found in 3.3.2 and the temperature is calculated by

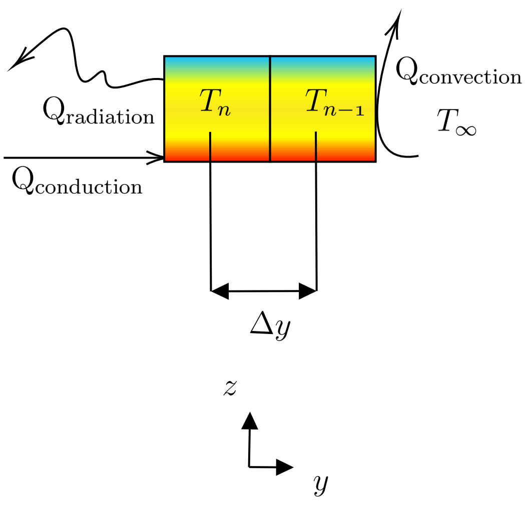


$\mathrm{Q}_{\text{radiation}}$
$\mathrm{Q}_{\text{conduction}}$
T_n
$\mathrm{Q}_{\text{convection}}$
T_∞
w_{int}
z
y

$$T_n(t+1) = \frac{(\mathrm{Q}_{\text{cond}_\text{n}}(t) + \mathrm{Q}_{\text{conv}_\text{n}}(t) + \mathrm{Q}_{\text{rad}_\text{n}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

$$\begin{aligned}
&\text{If } n = 0 \\
&\mathrm{Q}_{\text{rad}_\text{n}}(t) = \varepsilon\sigma\,(2\Delta z + \mathrm{w}_{\text{int}})\left((K + T_\infty)^4 - (K + T_0(t))^4\right) \\
&\mathrm{Q}_{\text{cond}_\text{n}}(t) = \frac{2k\mathrm{w}_{\text{int}}}{\Delta z}\,(T_b - T_0(t)) \\
&\mathrm{Q}_{\text{conv}_\text{n}}(t) = h\,(2\Delta z + \mathrm{w}_{\text{int}})\,(T_\infty - T_0(t))
\end{aligned} \quad (3.21)$$

The second layer (n) is denoted as 1. However, in this model the second layer is deposited left from the first layer $(n = 0)$, which results in heat conduction between the two deposited layers. Therefore, the conditions of heat transfer change for the first layer, which can be seen in the derivations of 3.3.2. The nodal temperatures can be calculated by

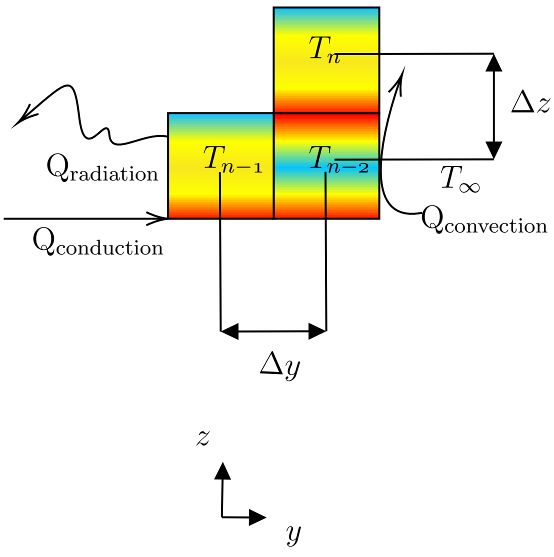


$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_n}}(t)+\mathrm{Q}_{\mathrm{conv_n}}(t)+\mathrm{Q}_{\mathrm{rad_n}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

$$T_{n-1}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-1}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-1}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-1}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-1}(t)$$

$$\begin{aligned}
&\text{If } n = 1\\
&\mathrm{Q}_{\mathrm{rad_n}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w_{int}}\right)\left((K+T_\infty)^4 - (K+T_n(t))^4\right)\\
&Q_{\mathrm{cond}_n}(t) = \frac{2k\mathrm{w_{int}}}{\Delta z}\left(T_b - T_n(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_{n-1}(t) - T_n(t)\right)\\
&Q_{\mathrm{conv}_n}(t) = h\left(\Delta z + \mathrm{w_{int}}\right)\left(T_\infty - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{rad_{n-1}}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-1}(t))^4\right)\\
&Q_{cond_{n-1}}(t) = \frac{2k\mathrm{w_{int}}}{\Delta z}\left(T_b - T_0(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_n(t) - T_{n-1}(t)\right)\\
&Q_{\mathrm{con}\,v_{n-1}}(t) = h\left(\Delta z + \mathrm{w_{int}}\right)\left(T_\infty - T_{n-1}(t)\right)
\end{aligned} \tag{3.22}$$

The third layer (n) is denoted as 2. However, in this model the third layer is deposited on top of the first layer ($n = 0$), which results in heat conduction between the two deposited layers. Therefore, the conditions of heat transfer again change for the first layer, which can be seen in the derivations of 3.3.2. The nodal temperatures can be calculated by



$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_n}}(t)+\mathrm{Q}_{\mathrm{conv_n}}(t)+\mathrm{Q}_{\mathrm{rad_n}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

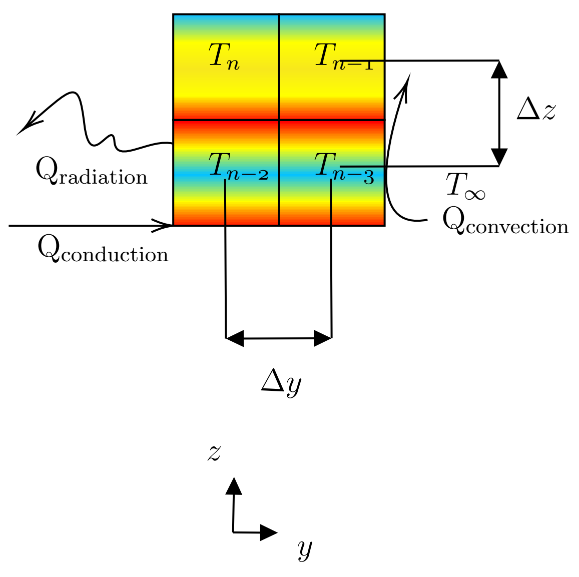
$$T_{n-1}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-1}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-1}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-1}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-1}(t)$$

$$T_{n-2}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-2}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-2}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-2}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-2}(t)$$

If $n = 2$

$$\begin{aligned}
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n}}}(t) = \varepsilon\sigma\left(2\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K + T_\infty)^4 - (K + T_n(t))^4\right)\\
&Q_{\mathrm{cond}_n}(t) = \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-2}(t) - T_n(t)\right)\\
&Q_{\mathrm{conv}_n}(t) = h\left(2\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n-1}}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K + T_\infty)^4 - (K + T_{n-1}(t))^4\right)\\
&Q_{\mathrm{cond}\ n-1}(t) = \frac{2kw_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-1}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta y}\left(T_{n-2}(t) - T_{n-1}(t)\right)\\
&Q_{\mathrm{conv}_{n-1}}(t) = h\left(\overline{\Delta z} + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_{n-1}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n-2}}}(t) = \varepsilon\sigma\Delta z\left((K + T_\infty)^4 - (K + T_{n-2}(t))^4\right)\\
&Q_{cond_{n-2}}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_b - T_0(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_n(t) - T_{n-2}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-1}(t) - T_{n-2}(t)\right)\\
&Q_{\mathrm{conv}_{n-2}}(t) = \Delta z h\left(T_\infty - T_{n-2}(t)\right)
\end{aligned} \tag{3.23}$$

The fourth layer (n) is denoted as 3. However, in this model the fourth layer is deposited on top of the second layer ($n = 1$) and next to the third layer ($n = 2$), which results in heat conduction between the three deposited layers. Therefore, the conditions of heat transfer again change for the second and third layer, which can be seen in the derivations of 3.3.2. The nodal temperatures can be calculated by



T_n
T_{n-1}
T_{n-2}
T_{n-3}
Δz
Δy
$\mathrm{Q}_{\mathrm{radiation}}$
$\mathrm{Q}_{\mathrm{conduction}}$
T_∞
$\mathrm{Q}_{\mathrm{convection}}$
z
y

$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

$$T_{n-1}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-1}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-1}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-1}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-1}(t)$$

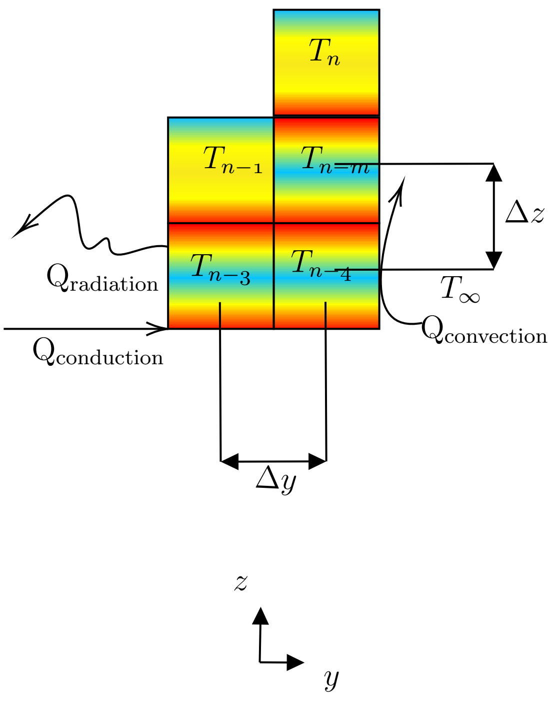
$$T_{n-2}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-2}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-2}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-2}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-2}(t)$$

$$T_{n-3}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-3}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-3}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-3}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-3}(t)$$

If $n = 3$

$$
\begin{aligned}
&\mathrm{Q}_{\mathrm{rad}_\mathrm{n}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K+T_\infty)^4 - (K+T_n(t))^4\right)\\
&Q_{\mathrm{cond}_n}(t) = \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_{n-2}(t) - T_n(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-1}(t) - T_n(t)\right)\\
&Q_{\mathrm{conv}_n}(t) = h\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n-1}}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K+T_\infty)^4 - (K+T_{n-1}(t))^4\right)\\
&Q_{\mathrm{cond}_{n-1}}(t) = \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_{n-3}(t) - T_{n-1}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_n(t) - T_{n-1}(t)\right)\\
&Q_{\mathrm{con}\,v_{n-1}}(t) = h\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_{n-1}(t)\right)\\
&Q_{\mathrm{rad}_{n-2}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-2}(t))^4\right)\\
&Q_{\mathrm{cond}_{n-2}}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-2}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_n(t) - T_{n-2}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-3}(t) - T_{n-2}(t)\right)\\
&Q_{\mathrm{con}\,v_{n-2}}(t) = \Delta z h\left(T_\infty - T_{n-2}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad}_{\mathrm{n-3}}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-3}(t))^4\right)\\
&Q_{\mathrm{cond}_{n-3}}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-3}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_{n-1}(t) - T_{n-3}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-2}(t) - T_{n-3}(t)\right)\\
&Q_{\mathrm{conv}_{n-3}}(t) = \Delta z h\left(T_\infty - T_{n-3}(t)\right)
\end{aligned}
\tag{3.24}
$$

The fifth layer (n) is denoted as 4. However, in this model the fifth layer is deposited on top of the third layer ($n = 2$), which results in heat conduction between the two deposited layers. Therefore, the conditions of heat transfer again change for the third layer, which can be seen in the derivations of 3.3.2. Furthermore, in the model the middle nodes are calculated also by a for loop in the range of $[2, m]$. The nodal temperatures can be calculated by


T_n
T_{n-1}
T_{n-m}
Δz
T_{n-3}
T_{n-4}
$\mathrm{Q}_{\mathrm{radiation}}$
T_∞
$\mathrm{Q}_{\mathrm{conduction}}$
$\mathrm{Q}_{\mathrm{convection}}$
Δy
z
y

$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_\mathrm{n}}(t) + \mathrm{Q}_{\mathrm{conv}_\mathrm{n}}(t) + \mathrm{Q}_{\mathrm{rad}_\mathrm{n}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

$$T_{n-1}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-1}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-1}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-1}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-1}(t)$$

$$T_{n-m}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-m}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-m}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-m}}}(t)\}\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-m}(t)$$

$$T_{n-3}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-3}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-3}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-3}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-3}(t)$$

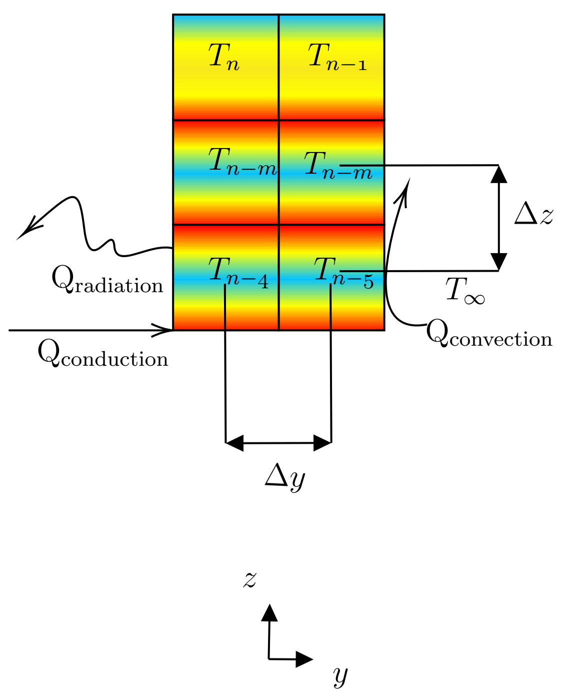
$$T_{n-4}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond}_{\mathrm{n-2}}}(t) + \mathrm{Q}_{\mathrm{conv}_{\mathrm{n-2}}}(t) + \mathrm{Q}_{\mathrm{rad}_{\mathrm{n-2}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-4}(t)$$

$$where\ m\ =\ [2, n]$$

If $n = 4$

$$
\begin{aligned}
&\mathrm{Q}_{\mathrm{rad_n}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K + T_\infty)^4 - (K + T_n(t))^4\right)\\
&Q_{\mathrm{cond}_n}(t) = \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_{n-m}(t) - T_n(t)\right)\\
&Q_{\mathrm{conv}_n}(t) = h\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_n(t)\right)\\
&\mathrm{Q}_{\mathrm{rad_{n-1}}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left((K + T_\infty)^4 - (K + T_{n-1}(t))^4\right)\\
&Q_{\mathrm{cond}_{n-1}}(t) = \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-3}(t) - T_{n-1}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta y}\left(T_{n-m}(t) - T_{n-1}(t)\right)\\
&Q_{\mathrm{conv}_{n-1}}(t) = h\left(\Delta z + \mathrm{w}_{\mathrm{int}}\right)\left(T_\infty - T_{n-1}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad_{n-m}}}(t) = \varepsilon\sigma 2\Delta z\left((K + T_\infty)^4 - (K + T_{n-m}(t))^4\right)\\
&Q_{\mathrm{cond}_{n-m}}(t) = \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_{n-4}(t) - T_{n-m}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-1}(t) - T_{n-m}(t)\right)\\
&Q_{\mathrm{conv}_{n-m}}(t) = 2h\Delta z\left(T_\infty - T_{n-m}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad_{n-3}}}(t) = \varepsilon\sigma\Delta z\left((K + T_\infty)^4 - (K + T_{n-3}(t))^4\right)\\
&Q_{cond_{n-3}}(t) = \frac{2k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-3}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta z}\left(T_n(t) - T_{n-3}(t)\right) + \frac{k\mathrm{w}_{\mathrm{int}}}{\Delta y}\left(T_{n-4}(t) - T_{n-3}(t)\right)\\
&Q_{\mathrm{conv}_{n-3}}(t) = \Delta z h\left(T_\infty - T_{n-3}(t)\right)\\
&\mathrm{Q}_{\mathrm{rad_{n-4}}}(t) = \varepsilon\sigma\Delta z\left((K + T_\infty)^4 - (K + T_{n-4}(t))^4\right)\\
&Q_{\mathrm{cond}\ _{n-3}}(t) = \frac{2kw_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-4}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-m}(t) - T_{n-4}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta y}\left(T_{n-3}(t) - T_{n-4}(t)\right)\\
&Q_{\mathrm{conv}_{n-3}}(t) = \Delta z h\left(T_\infty - T_{n-4}(t)\right)
\end{aligned}
\tag{3.25}
$$

The sixth layer (n) is denoted as 5. However, in this model the sixth layer is deposited on top of the fourth layer ($n = 2$) and next to the fifth layer ($n = 4$), which results in heat conduction between the three deposited layers. Therefore, the conditions of heat transfer again change for the fourth and fifth layer, which can be seen in the derivations of 3.3.2. Furthermore, in the model the middle nodes are calculated also by a for loop in the range of $[2, m]$. The nodal temperatures can be calculated by



$$T_n(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_n}}(t)+\mathrm{Q}_{\mathrm{conv_n}}(t)+\mathrm{Q}_{\mathrm{rad_n}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_n(t)$$

$$T_{n-1}(t+1) = \frac{(Q_{cond_{n-1}}(t)+\mathrm{Q}_{\mathrm{conv_{n-1}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-1}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-1}(t)$$

$$T_{n-m}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-m}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-m}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-m}}}(t)\}\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-m}(t)$$

$$T_{n-4}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-4}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-4}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-4}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-4}(t)$$

$$T_{n-5}(t+1) = \frac{(\mathrm{Q}_{\mathrm{cond_{n-5}}}(t)+\mathrm{Q}_{\mathrm{conv_{n-5}}}(t)+\mathrm{Q}_{\mathrm{rad_{n-5}}}(t)\}2\Delta t}{\rho c_p \Delta y \Delta z} + T_{n-5}(t)$$

$$Where\ m \;=\; [2, n]$$

$$
\begin{aligned}
&\text{If } n > 4 \\
&\mathrm{Q}_{\mathrm{rad_n}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w_{int}}\right)\left((K+T_\infty)^4 - (K+T_n(t))^4\right) \\
&Q_{\mathrm{cond}\ n}(t) = \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-m}(t) - T_n(t)\right) \\
&Q_{\mathrm{conv}_n}(t) = h\left(\Delta z + \mathrm{w_{int}}\right)\left(T_\infty - T_n(t)\right) \\
&\mathrm{Q}_{\mathrm{rad_{n-1}}}(t) = \varepsilon\sigma\left(\Delta z + \mathrm{w_{int}}\right)\left((K+T_\infty)^4 - (K+T_{n-1}(t))^4\right) \\
&Q_{\mathrm{cond}_{n-1}}(t) = \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-m}(t) - T_{n-1}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta y}\left(T_n(t) - T_{n-1}(t)\right) \\
&Q_{\mathrm{conv}\ n-1}(t) = h\left(\Delta z + w_{\mathrm{int}}\ \right)\left(T_\infty - T_{n-1}(t)\right) \\
&\text{If } m = \text{ equal number} \\
&\mathrm{Q}_{\mathrm{rad_{n-m}}}(t) = \varepsilon\sigma 2\Delta z\left((K+T_\infty)^4 - (K+T_{n-m}(t))^4\right) \\
&Q_{\mathrm{cond}_{n-m}}(t) = \frac{k\mathrm{w_{int}}}{\Delta z}\left(T_{n-4}(t) - T_{n-m}(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_{n-m-1}(t) - T_{n-m}(t)\right) \\
&Q_{\mathrm{conv}_{n-m}}(t) = 2h\Delta z\left(T_\infty - T_{n-m}(t)\right) \\
&\text{If } m = \text{ not equal number} \\
&\mathrm{Q}_{\mathrm{rad_{n-m}}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-m}(t))^4\right) \\
&Q_{\mathrm{cond}_{n-m}}(t) = \frac{k\mathrm{w_{int}}}{\Delta z}\left(T_{n-5}(t) - T_{n-m}(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_{n-m-1}(t) - T_{n-m}(t)\right) \\
&Q_{\mathrm{conv}_{n-\infty}}(t) = h(\Delta z)\left(T_\infty - T_{n-m}(t)\right) \\
&\mathrm{Q}_{\mathrm{rad_{n-4}}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-4}(t))^4\right) \\
&Q_{cond_{n-4}}(t) = \frac{2k\mathrm{w_{int}}}{\Delta z}\left(T_b - T_{n-4}(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta z}\left(T_{n-m}(t) - T_{n-4}(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_{n-5}(t) - T_{n-4}(t)\right) \\
&Q_{\mathrm{conv}_{n-4}}(t) = \Delta z h\left(T_\infty - T_{n-3}(t)\right) \\
&\mathrm{Q}_{\mathrm{rad_{n-5}}}(t) = \varepsilon\sigma\Delta z\left((K+T_\infty)^4 - (K+T_{n-5}(t))^4\right) \\
&Q_{\mathrm{cond}_{n-5}}(t) = \frac{2kw_{\mathrm{int}}}{\Delta z}\left(T_b - T_{n-5}(t)\right) + \frac{kw_{\mathrm{int}}}{\Delta z}\left(T_{n-m}(t) - T_{n-5}(t)\right) + \frac{k\mathrm{w_{int}}}{\Delta y}\left(T_{n-4}(t) - T_{n-5}(t)\right) \\
&Q_{\mathrm{conv}_{n-5}}(t) = \Delta z h\left(T_\infty - T_{n-5}(t)\right)
\end{aligned}
\tag{3.26}
$$

After $n > 4$ the conditions of the model do not change anymore, since the bottom nodes, middle nodes and the top nodes are all included. Therefore, the derivation stops after $n > 4$, since a for loop updates the temperature of the middle nodes, which are denoted by $n - m$. Where m can be in the interval $[2, n]$. Furthermore, the derivation shows that the conditions change when n is equal or unequal. In the code of the model this is programmed with an if statement. To recall, temperatures at locations $T(y, z, t)$ are rewritten to $T_n(t)$, in order to represent the layer number.

3.4 Forced convection of airgun

Forced convection is taken into account in both numerical thermal models. According to Bahrami [26], a thermal boundary is occurring when air at a specific temperature is flowing over a flat plate. When the Prandtl number P_r of the material < 1 the thermal boundary can be illustrated as.

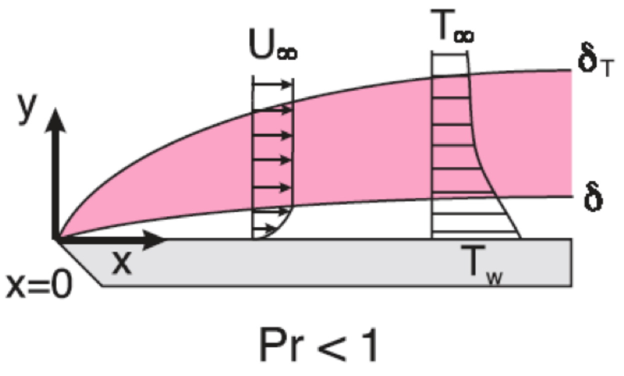


Figure 3.5: Thermal boundary layer over a flat plate, where U_∞ is the velocity of the flow (m/s), T_∞ is the temperature of the flow (°C) and at x is 0 is the air-gun in this case. Retrieved from [26]

According to Bahrami [26], the critical value of the Reynolds number for a laminar flow over a flat plate must be < 500000. The printed products have an average height of $L = 0.3m$. The Reynolds number is calculated in subsection 5.2, since the velocity of the airflow is determined experimentally.

According to Bahrami [26], the local Nusselt number in a laminar flow at a location x, or in this case a node (n), can be derived by

$$\mathrm{Nu_n} = \frac{h\Delta z}{k} = 0.332\,\mathrm{Re}_n^{1/2}\,\mathrm{Pr}^{1/3} \quad \mathrm{Pr} \geq 0.6 \tag{3.27}$$

As stated above, forced convection is occurring in the 1-dimensional model after the fifth layer is deposited, since the air-gun is not used earlier. For the 2-dimensional model, the forced convection is occurring after the tenth layer is deposited, since the 2-dimensional model consists of 2 layers in the horizontal direction. Furthermore, the air-gun is only blowing air over the layers for a specific time period. Therefore, two constraints are used, which are stated below

$$n_t - i < y \quad and \quad i < y \tag{3.28}$$

where n_t total iterations, i current iteration and y is the constraint, which is derived by

$$y = ceil(\frac{dist}{\Delta t}) \quad \text{and } dist = \frac{r}{v_{airgun}} \tag{3.29}$$

where Δt is the time step, r is the radius of the air-gun (mm) and v_{airgun} is the printing speed (mm/s).

In this case, the radius of the air-gun is $22.5mm$, when the printing speed and time step are set to $25mm/s$ and $0.5s$ respectively. The number of iterations where the air-gun is active is

$$dist = \frac{r}{v_{airgun}} = \frac{22.5}{25} \quad \text{and } y = ceil(\frac{dist}{\Delta t} = ceil(\frac{0.9}{0.5}) = 2 \tag{3.30}$$

which means that the air-gun is active in the first two and the last two iterations of the simulation. The graphical representation of this can be seen in the figure below. The constraints are also stated in this figure.

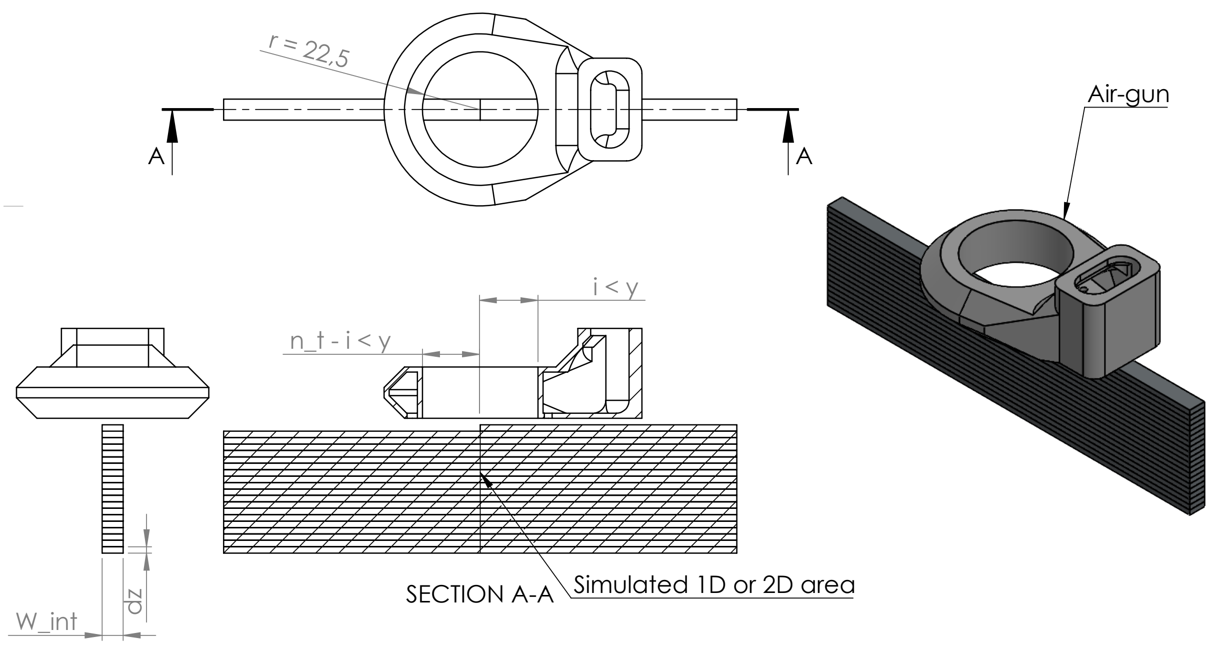


Figure 3.6: Graphical representation of air-gun that is moving over the layers during the printing process

The forced convection coefficient is modelled as

$$\mathrm{Re_n} = \frac{v_n \Delta z}{\nu} \tag{3.31}$$

where, $\mathrm{Re_n}$ is the Reynolds number, v_n is the velocity of the airflow (m/s), Δz is the nodal spacing and ν is the kinematic Viscosity of air (mm/s^2).
The forced convection coefficient h is derived by

$$h_n = \frac{\mathrm{Nu_n} k}{\Delta z} \tag{3.32}$$

where, $\mathrm{Nu_n}$ is the local Nusselt number at the node (n), k is the thermal conductivity of air $[W/m.K]$ and Δz is the nodal spacing (mm).

Chapter 4: Experiments

Experiments must be performed to validate the thermal model. First, the method of the measurements to determine the airflow velocity of the air-gun is explained. Second, the determination of the input parameters is stated. Thereafter, the method and set-up for the thermal measurements is explained. Finally, the method for the tensile tests is explained. A hypothesis is stated in order to test if the temperatures predicted by the thermal model converges to the measured temperatures.

The thermal model can be validated with the experiment that is set up and can be used for the process to optimize the quality of the printed part.

4.1 Equipment air-gun measurements

An air-gun, which is depicted in Figure 4.5 and 4.6, is mounted on the printing nozzle of the robot to avoid a collapse of the printed layers and increase the part geometry. However, this causes forced convection in a certain time interval when the nozzle moves over the layers. To determine the forced convection coefficient h_f, which is explained in the literature review 2.3.2, the temperature and airspeed that flows out of the air-gun must be determined experimentally.

To obtain the temperature of the air, a thermocouple sensor is used. The airspeed velocity is measured with a small Pitot-Static tube, which is connected to an MPXV7002DP sensor with tubes. The MPXV7002DP is connected to an Arduino UNO microprocessor. The connection diagram is depicted in Figure 4.3.

4.1.1 Pitot-Static tube

A Pitot-Static tube consists of two chambers, which are separated by two pipes. On each pipe a tube is connected, in order to measure the pressure inside the chamber. The pressures are measured by the MPXV7002DP sensor. In Figure 4.1, a schematic overview of a Pitot-Static tube is depicted. The inner Pitot tube gives the pressure at the front of the inlet. The outer static tube gives the static pressure that flows away from the inlet.

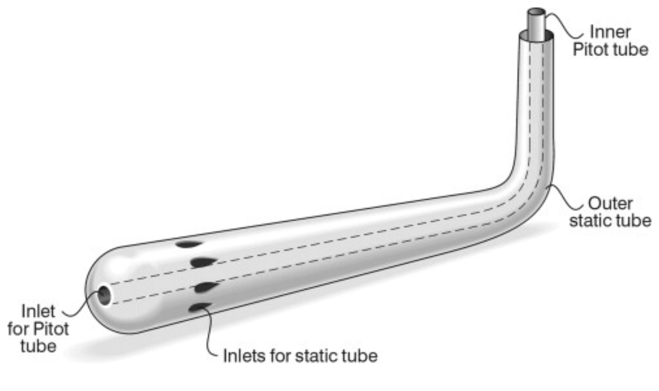
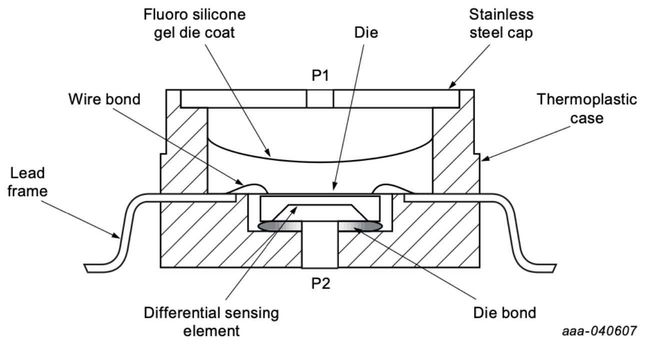


Figure 4.1: Schematic figure of a Pitot-Static tube obtained from [34]

4.1.2 MPXV7002DP

The MPXV700DP piezoresistive transducer is a monolithic silicon pressure sensor [35]. The sensor has a membrane that measures the pressure difference between the two inputs and the sensor is depicted in Figure 4.2. The pressure difference is converted to a voltage signal and the conversion rate graph can be found in the appendix in 8.1.



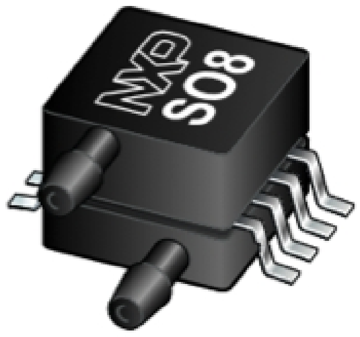


Figure 4.2: MPXV7002DP obtained from NXP [35]

4.1.3 Arduino UNO R3

The Arduino UNO R3 is used to convert the signal of the MPXV7002DP sensor. A script which was obtained from [36] was used as a base and this script was adjusted in an earlier study, in order to calibrate the MPXV7002DP sensor at low wind speeds. The sensor was calibrated with the script using reference air speeds in a wind tunnel. The script can be found in 8.4. It uses Bernoulli's principle to convert the measured pressures into airspeed velocity and is given by

$$\frac{1}{2}\rho_1 v_1^2 + P_1 + \rho_1 g h_1 = \frac{1}{2}\rho_2 v_2^2 + P_2 + \rho_2 g h_2 \tag{4.1}$$

where, ρ, v, P, g, h are the density of air, airspeed in meters per second (m/s), pressure in Pascal (Pa), the gravitational constant in m/s^2 and height in meters respectively. The subscripts 1,2 represent the two different chambers, with static and total pressure, inside the Pitot-Static tube and the MPXV7002DP sensor. The gravitational constant g is assumed to have no impact and is left out of the equation and the static pressure chamber has a velocity v_1 of $0m/s$, which results in

$$P_1 = P_2 + \frac{1}{2}\rho_2 v_2^2 \tag{4.2}$$

To calculate v the formula can be rewritten to

$$v = \sqrt{\frac{2\left(P_2 - P_1\right)}{\rho}} \tag{4.3}$$

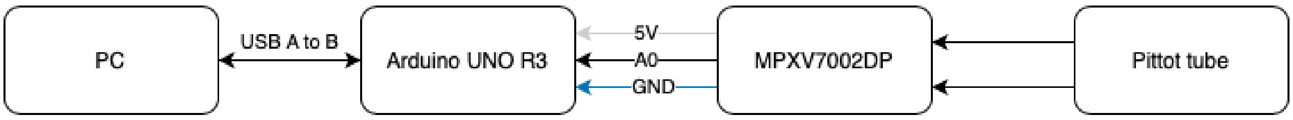


Figure 4.3: Connection diagram Pitot-Static tube

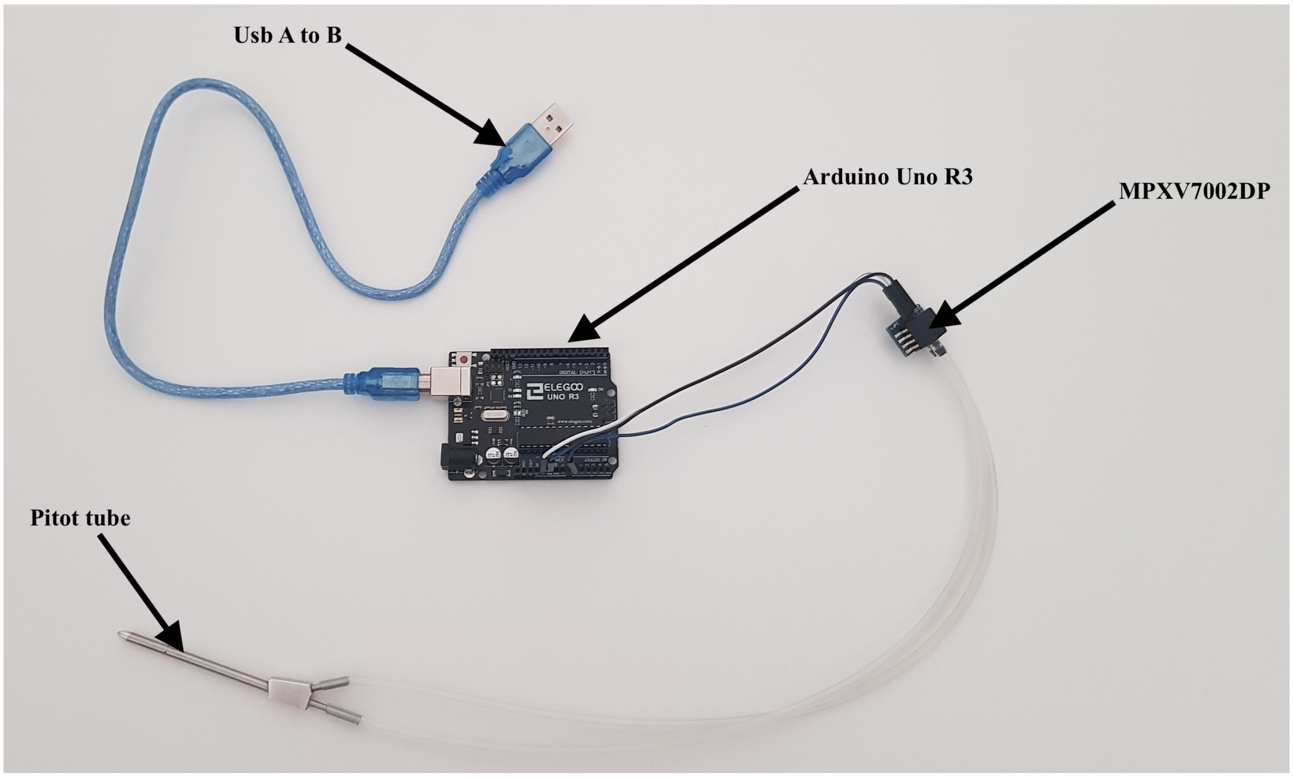
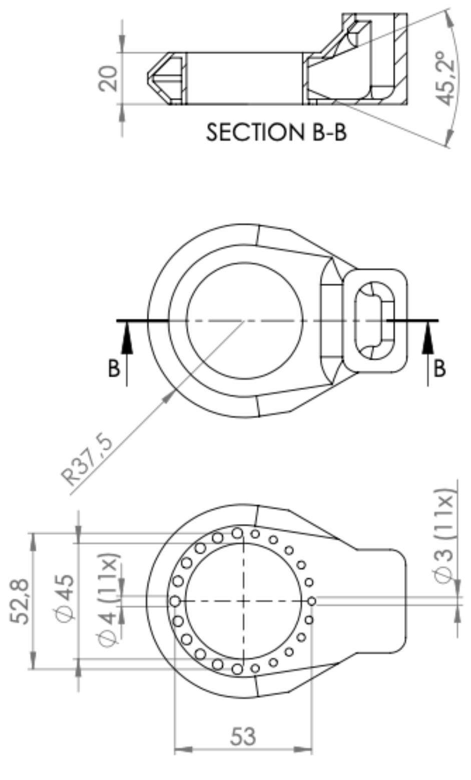


Figure 4.4: Set up of the Pitot-Static tube connected with the MPXV7002DP sensor and the Arduino UNO R3



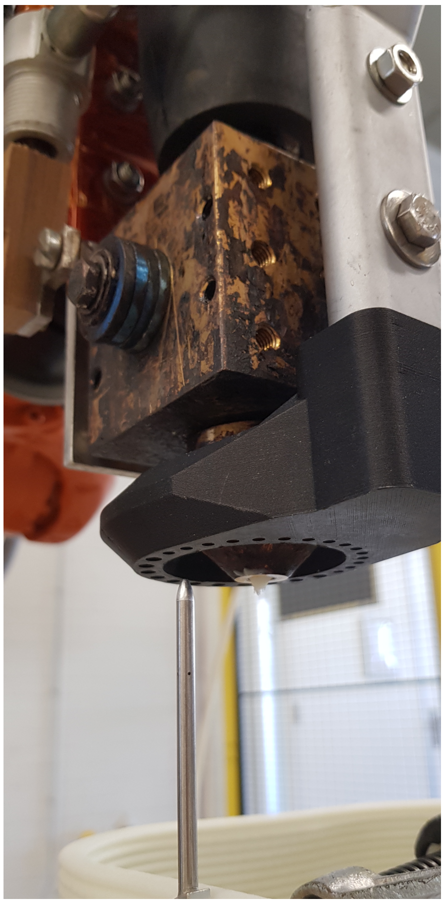


Figure 4.5: 2-dimensional drawing of the air gun

Figure 4.6: Pitot-Static tube set up for airspeed measurement

4.2 Equipment thermal measurements

The thermal behaviour of each deposited layer is analyzed with the information from the thermal image, which is being created with a thermal camera. Furthermore, the parts will be printed with the Smart Polymer Granulate 3D Welding System. The equipment and material which are needed to perform the experiment are stated below. The first two bullet points will not be explained further, since this has been stated earlier in 1.2.2. The setup of the thermal measurements with the FLIR A35 camera is depicted in the figure below.

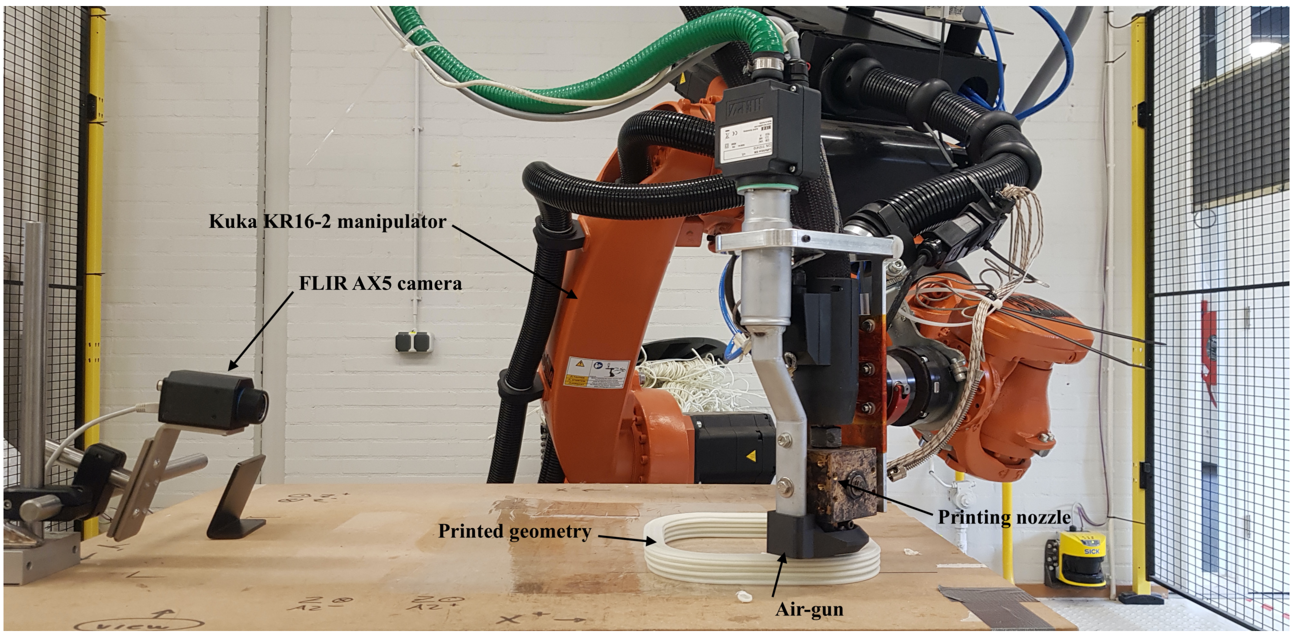


Figure 4.7: Set up thermal measurements with FLIR A35 thermal camera

1. Kuka KR16-2 manipulator
2. Xtrution granulate extruder
3. FLIR A35 thermal camera
4. Thermocouple sensor
5. Material: Hostacom G3 N01

4.2.1 Thermal imaging with FLIR A35

Thermal videos will be created with the FLIR A35 thermal camera. The videos will be analyzed with the FLIR Tools+ software. The specifications of the camera are stated in 8.1. In the FLIR Tools+ software, several measuring points can be set. These points can be recorded, which results in the temperature distribution over time for that certain measured point.

Figure 4.8: FLIR A35 infrared camera, obtained from FLIR [37]

4.2.2 Thermocouple sensor

During the process, four thermocouples are positioned on the locations of the nodes that are predicted by the model. To check if the thermocouples were accurate, they were placed in boiling water, which gave the result of 100 ℃. This is an additional measurement, in order to compare the measurements of the FLIR A35 camera. The results of this measurement are used to check if the results of the FLIR A35 camera are valid. Furthermore, this sensor is also used to determine the emissivity of the material.

4.3 Input parameters & material properties

Hostacom G3 N01 is used for the experiment. According to Basell [38], it is a Glass reinforced polypropylene (PP), which consists out of 70 % PP and 30 % Glass. This material has been chosen for this experiment, since it has good printing properties and only this material has been tested on the Smart Polymer Granulate 3D Welding System currently. The properties of Hostacom G3 N01 are depicted below. The material specific properties of Hostacom G3 N01 are obtained from Maier [39] and Basell [38].

4.3.1 Emissivity determination

The emissivity of the material is determined empirically, since this parameter is unknown. An experiment is set up with the thermocouple sensor, the FLIR A35 camera and an industrial oven. An printed product of Hostacom G3 N01 is heated to a fixed temperature and the temperatures of the thermocouple sensor and the FLIR A35 camera are compared. The emissivity parameter of the FLIR A35 is fitted to comply to the results of the thermocouple sensor. The experiment is performed in the following order.

1. Pre-set oven to 100℃
2. Drill holes for thermocouple wires
3. Set up FLIR A35 thermal camera
4. Place printed product in industrial oven
5. Take geometry out of oven
6. Put wires in heated geometry
7. Compare results of FLIR A35 with thermocouple sensor
8. Fit emissivity parameter of FLIR A35, such that the results comply

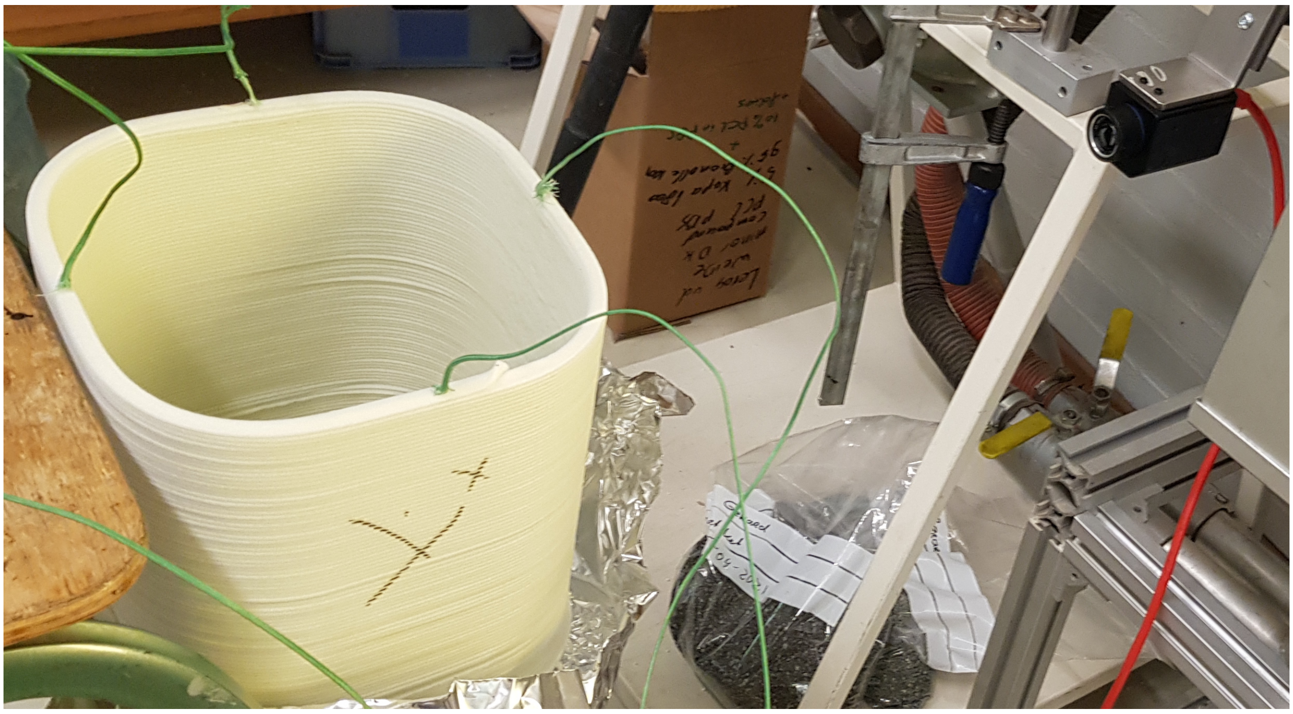


Figure 4.9: Experiment set-up for determining the emissivity of the material

Inputs	**Values**
Material	Hostacom G3 N01
Density, $[kg/m^3]$	1150
Crystalline melting temperature, T_m [°C]	120 to 130
Emissivity, ε	0.94
Thermal conductivity, $k[W/m.K]$	0.3
Specific heat capacity, c_p $[J/kg.K]$	2200
Deposition temperature, $\mathrm{T}_{\mathrm{dep}}$[°C]	175
Ambient temperature, T_∞ [°C]	26
Δz $[mm]$	2.5
$\mathrm{w}_{\mathrm{int}}$ $[mm]$	9

Table 4.1: Input parameters

4.4 Temperature Measurements

Temperatures are measured with the FLIR A35 thermal camera and a thermocouple sensor at several nodes, which are defined according to the input parameters of the thermal model. Δz represents the nodal spacing of the nodes where the temperature is measured. These temperatures are measured over the total process time, which is in line with the time of running time of the thermal model. Several measurements are done, in order to see the effect of changing the layer time on the thermal behaviour. The results of the temperature measurements are discussed in 5. Figure 4.10 shows a schematic representation of how and at which locations the temperatures are measured and Figure 4.11 shows the temperature measurements with the FLIR A35 camera.

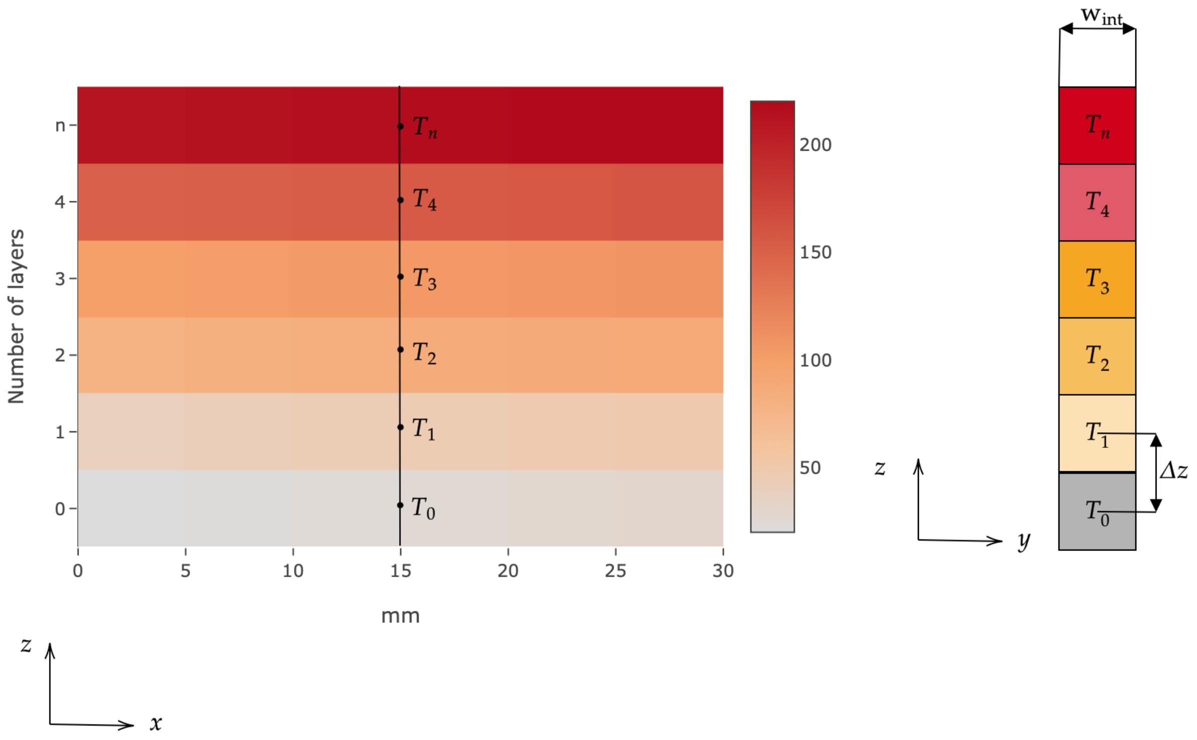


Figure 4.10: Schematic figure of temperature measurements, where the x-axis denotes the position in space and the y-axis denotes the layer number. On the right a cross section of the layers is depicted.

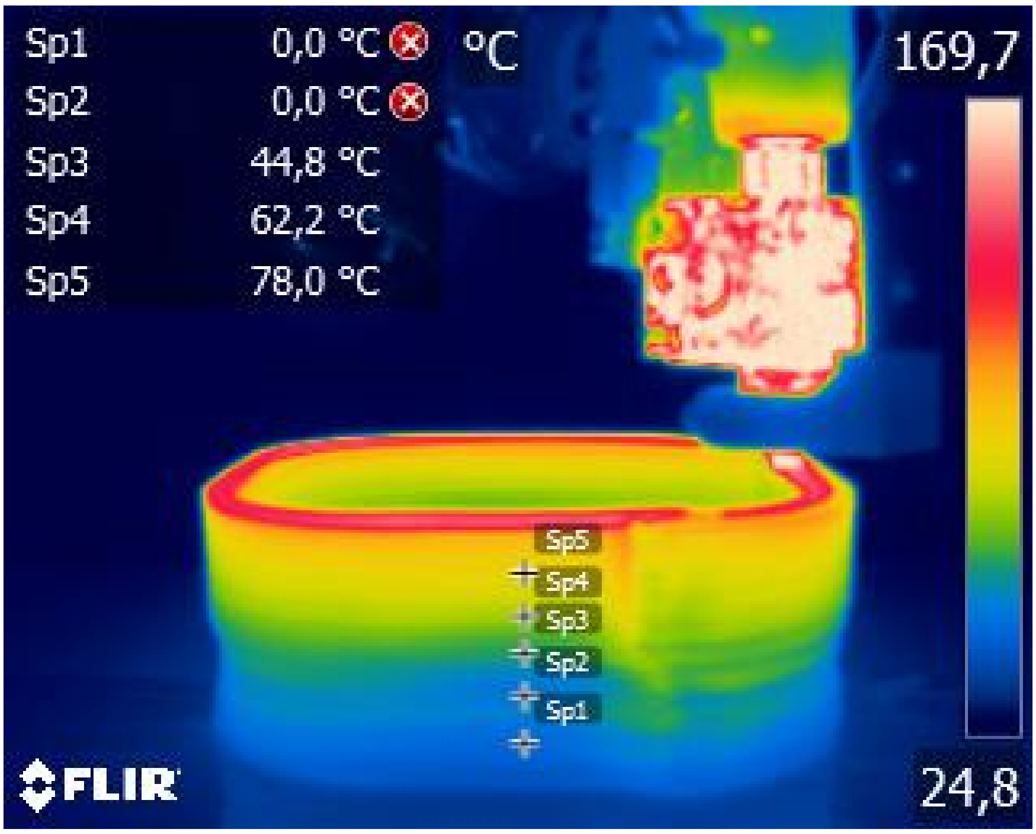


Figure 4.11: Temperature measurements with FLIR A35 camera, where Sp1, Sp2, Sp3, Sp4 *and* Sp5 *represent the data points that are being measured.*

During the printing process the thermocouple sensor is used to see if the measurements of the FLIR A35 camera are in line with the measurements of the thermocouple sensor, in order to obtain more certainty about the measurements of the FLIR A35 camera. Four times a thermocouple is placed by hand at the former top layer L_{p-1}.This is done just before a new layer L_p is deposited. The thermocouples will be placed on the same layer as the measurements of the FLIR A35 camera. The four thermocouples are placed between L_5 & L_6, L_{10} & L_{11}, L_{15} & L_{16} and L_{25} & L_{26} respectively.

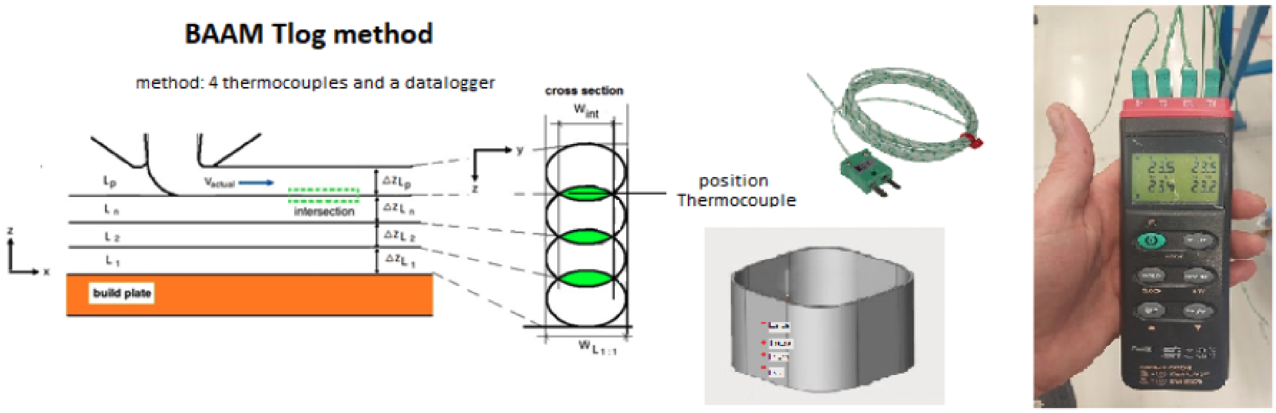


Figure 4.12: Thermocouple log method for temperature measurements, obtained from NHL Stenden

4.5 Equipment for tensile tests

First a cross-section is cut from a printed geometry. After that, the cross-section is placed inside the machine. The machine applies a pulling force, until the cross-section breaks, which can be seen in 4.13. Two cross-sections are made, which are printed with different layer times, in order to test the following hypothesis. The tensile tests are performed according to the input parameters stated in the table below.

If the temperature of the previous layer T_{top} is below the crystalline melting temperature T_m before the current layer is deposited, then the strength of the material is decreased, due to poor inter layer bonding

Inputs	**Values**
Material	Hostacom G3 N01
Ambient temperature, T_∞ [°C]	23
Pre-load, MPa	0.2
Speed, tensile modulus, mm/min	1
Test speed, mm/min	5
Machine data	Zwick UPM 14740 ZMART.PRO Zwick BZ1-EXZW003
Layer time T1, seconds	30.39
Layer time T4, seconds	22.79

Table 4.2: Input parameters tensile strength test

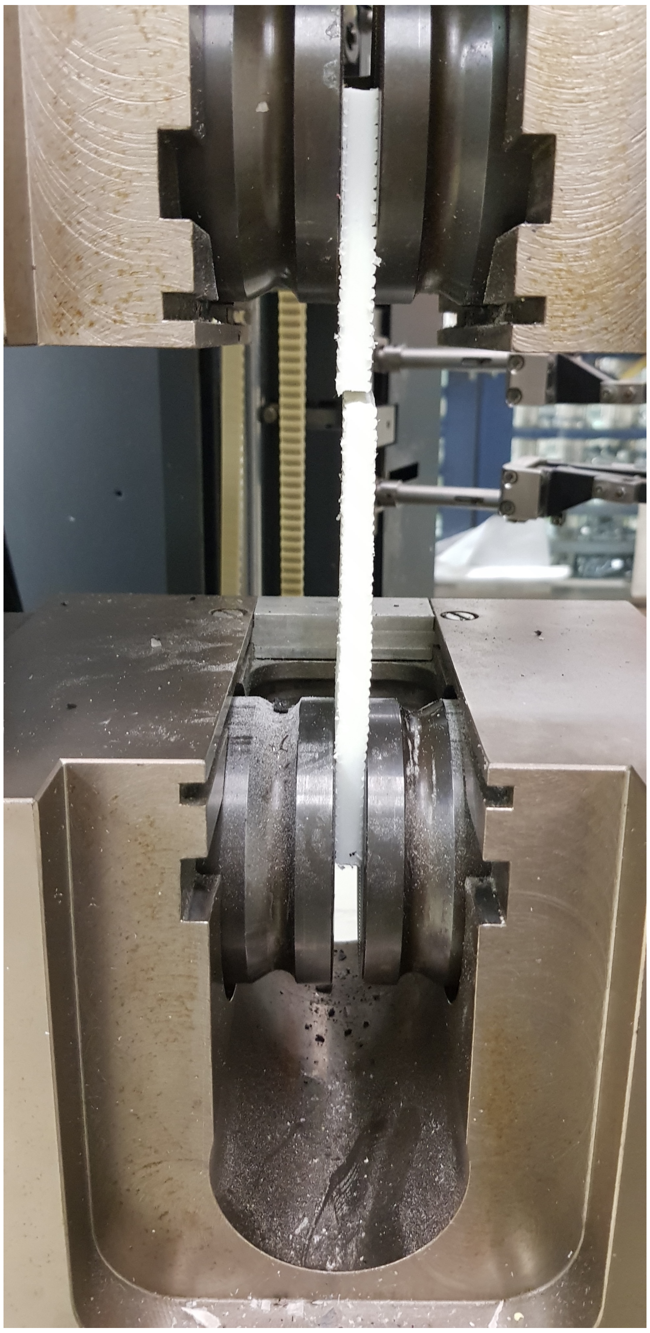


Figure 4.13: Tensile strength test setup

Chapter 5: Results

5.1 Emissivity determination

The emissivity is determined according to the previous described method, which is explained in 4.3.1. After the measurements were performed, the emissivity parameter of the FLIR A35 camera was shifted to fit the data of the thermocouple sensor measurements. The emissivity was determined by dividing the temperatures of the FLIR A35 camera with the emissivity parameter. This resulted in 0.94 for the value of the emissivity. Therefore, this value is used for the thermal measurements with the FLIR A35 thermal camera. The value is also used in both the numerical thermal models.

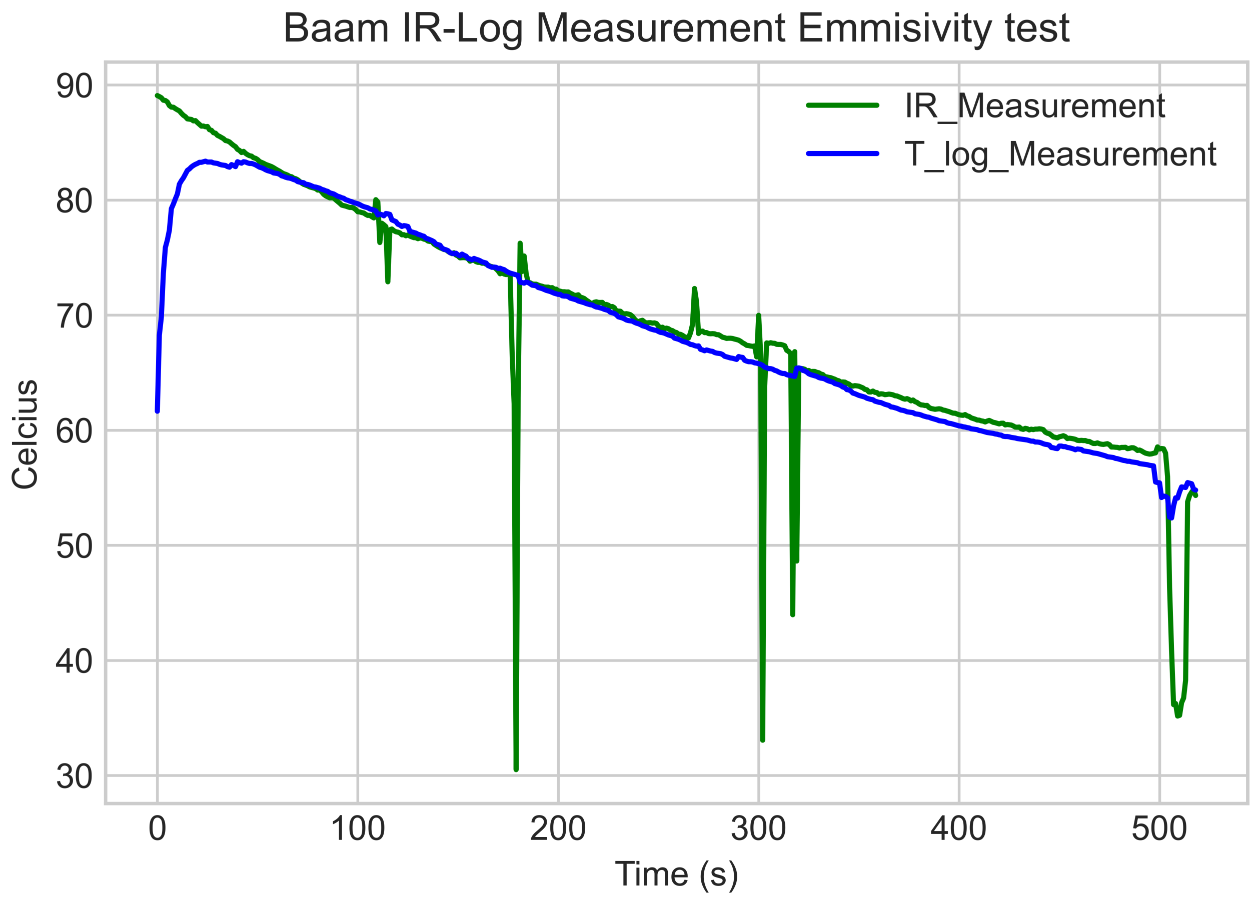


Figure 5.1: Temperature measurements of Emissivity determination

5.2 Air-gun measurements

The measurements that are performed with the Pitot-Static tube are shown in 5.2. In total 342 measurements are obtained per measured point and the average is plotted in 5.2. The error of the measurements is plotted in 5.3. From this data a trend line is obtained, which is given by the quadratic function

$$v = 1.8247^{-5}x^2 + -0.0246212x + 11.7115 \tag{5.1}$$

where, v is the velocity (m/s) and x is the Displacement (mm).

A trend line was needed to approximate the airspeed at 0mm, 25mm, 50mm and 75mm, since these values could not be measured by the Pitot-Static tube. This happened because the air was not flowing in front of the Pitot-Static tube, which created higher values for P_1 than P_2 and this resulted in a negative airspeed. Therefore, a trend line is used to approximate these values. Furthermore, the quadratic equation is used to calculate the local forced convection coefficient, which is explained in 3.4. The local forced convection coefficient per layer of the 1-dimensional model is depicted in 5.4. For the 2-dimensional model, the local forced convection coefficient at the different layers is the same. Therefore, only one graph is depicted. The graph shows that the local forced convection coefficient increases with the layer number, which is logical, since the velocity of the air that is flowing out of the air-gun is higher at the higher layer numbers. This is due to the fact that the air-gun is closer to the higher layers, which are deposited later.

This gives a Reynolds number of

$$\mathrm{Re} = \frac{vL}{\nu} \tag{5.2}$$

where v is the velocity of the flow (m/s), L is the length (m), ν is the kinematic viscosity of the fluid (m/s^2).

If the parameters of the printed product are filled in, the Reynolds number becomes

$$\mathrm{Re} = \frac{11.7086(0.12)}{1.493e-05} \quad \longrightarrow 94107.8 \tag{5.3}$$

where, $L = n(\Delta z) = 48(0.0025)$ and the Reynolds number is below the critical value of 500000. Therefore according to Bahrami [26], it can be considered as a laminar flow.

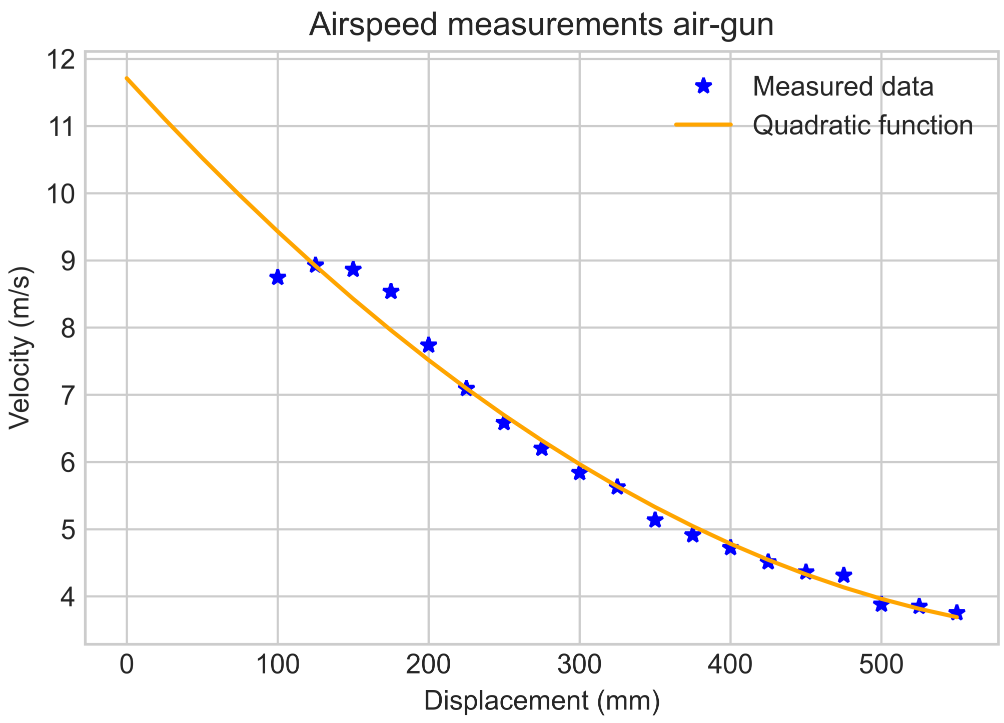


Figure 5.2: Airspeed measurements with Pitot-Static tube

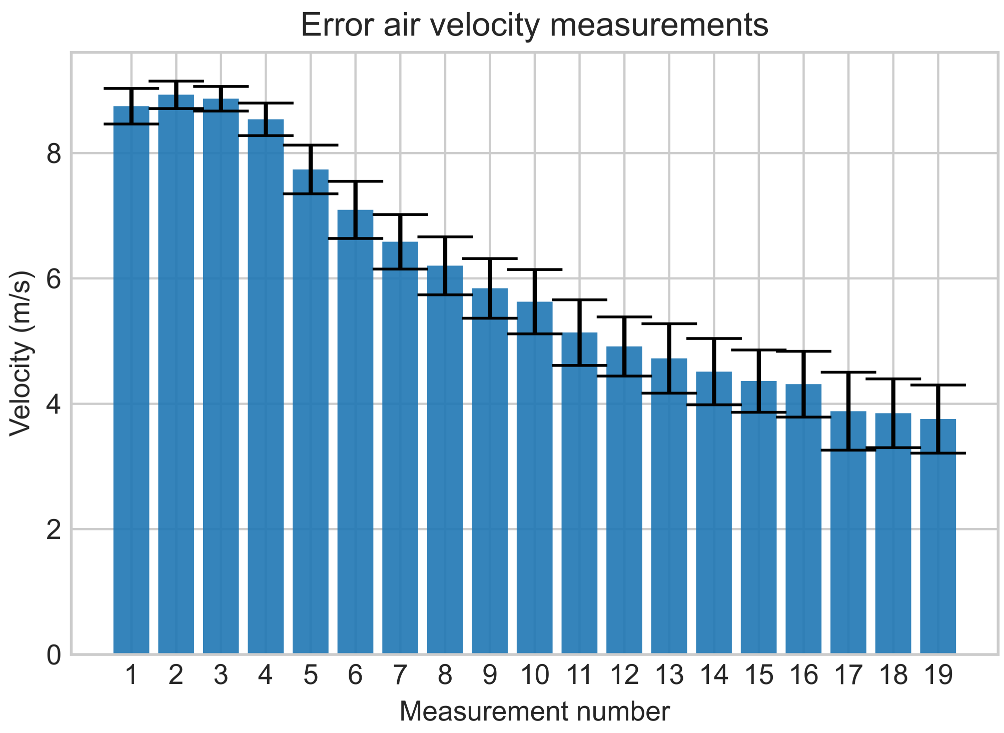


Figure 5.3: Error bar of 342 airspeed measurements with Pitot-Static tube

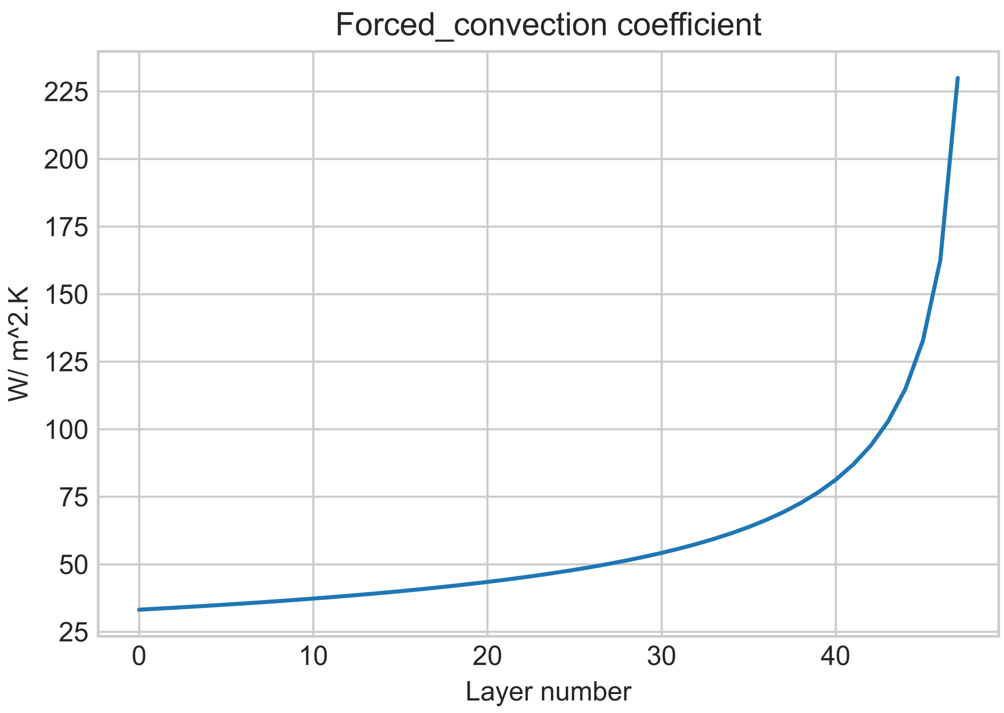


Figure 5.4: Forced convection coefficient per layer in the 1-dimensional model

5.3 Results 1-dimensional numerical thermal model

The 1-dimensional numerical thermal model predicts the thermal behaviour over time for thin walls. For the simulation the same input parameters have been used as in the experiments. The input parameters are stated below. The nodal temperature predictions over time for the first 15 layers of the product are shown in Figure 5.6.

Inputs	**Values**
Material	Hostacom G3 N01
Density, $[kg/m^3]$	1150
Melt temperature, T_m [°C]	[125,130]
Emissivity, ε	0.94
Thermal conductivity, $k[W/m.K]$	0.3
Specific heat capacity, c_p $[J/kg.K]$	2200
Deposition temperature, $\mathrm{T_{dep}}$[°C]	180
Ambient temperature, T_∞ [°C]	34
Δz $[mm]$	2.5
w_{int} $[mm]$	6
Temperature air-gun, $\mathrm{T_{airgun}}$[°C]	40
Number of layers, n_L	48
Layer time, t_p $[mm]$	30
Printing speed $\mathrm{v_{airgun}}$ $[mm/s]$	25

Table 5.1: Input parameters 1-dimensional model

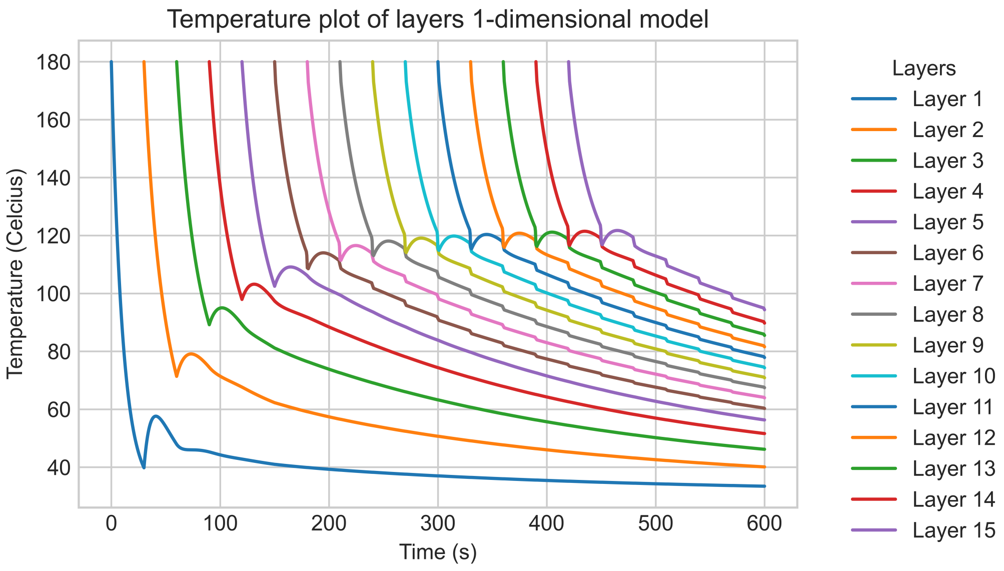


Figure 5.5: Nodal temperature prediction over time for the first 15 layers of the product with $\Delta t = 1/30$ seconds

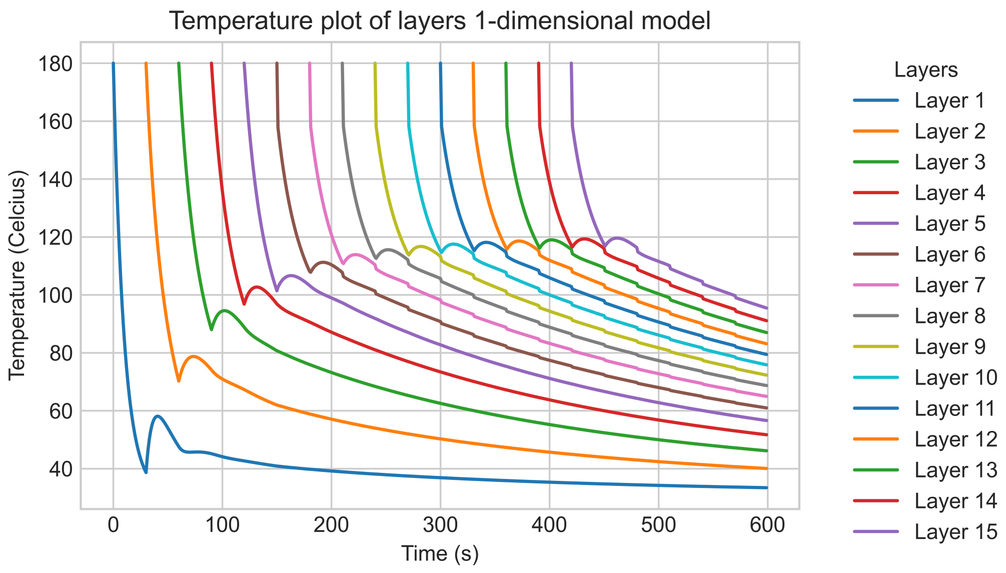


Figure 5.6: Nodal temperature prediction over time for the first 15 layers of the product with $\Delta t = 1$

The plots show that the temperature at each node starts at around 180 °C. Figures 5.5 and 5.6 show that over time the thermal behaviour across the layers reach a steady state. Figure 5.5 has a much smaller time step $t = 0.0333$ seconds, whereas Figure 5.6 has a time step of 1 second. There can be seen that a smaller time step does not increase the accuracy of the thermal model significantly, only the computation time increases. The accuracy of the number of iterations where the forced convection is active increases slightly. However, the FLIR A35 camera measures with 30 or 60 frames per second and the thermocouple measures with 1 data point per second. Therefore, these figures have different time steps. As stated earlier in the report, the strength and geometry deviation is affected if the previous layer reaches a temperature below the melt temperature (T_m) when a new layer is deposited on the previous layer ($\mathrm{T_{top}}$). In this case, $\mathrm{T_{top}}$ reaches a steady-state value of around 115 °C. Decreasing the layer time or increasing the ambient temperature could result in a higher value for $\mathrm{T_{top}}$. However, the process is situated in an uncontrolled area, which makes it hard to increase the ambient temperature.

5.3.1 Experimental results of 01-06-21

On 01-06-21 several experiments were performed. A product was printed with the input parameters that are stated in 5.3. For the simulation a time step (Δt) of 1 second was used for the thermocouple and a time step (Δt) of $1/30$ second was used for the camera. In Figures 5.7, 5.8, 5.9 and 5.10 the results of the measured data points with the thermocouple sensor are plotted against the predicted temperature data points. At higher temperatures, there can be seen that the predicted temperature is lower at higher temperatures. Over time, the predicted and measured temperatures converge to each other. Furthermore, Figures 5.11, 5.12,5.13 and 5.14 show the results of the FLIR A35 thermal camera measurements and the predicted temperatures for that specific layer. There can be seen that the measured results of the camera have much deviation in the beginning, this is caused by the radiation of the printing nozzle. The camera is overexposed by radiation when the printing nozzle is passing by, which is decreasing over time, since the printing nozzle is moving further away from the camera. Nevertheless, a clear contour can be seen from the graphs. The measured results are in line with the data which is measured with the thermocouple sensor. The predicted temperatures cool down faster at higher temperatures than the measured temperatures of the FLIR A35 thermal camera. Two reasons could cause this behaviour, which is the assumption that the radiation of the printing nozzle on the printed layers is neglected. The second reason could be that the thermal conduction coefficient (k) of the material is divergent of what [38] claims. However, this must be a very high deviation, since the coefficient was increased and decreased within a range of 0.02 to 4.0 and the predicted results still did not comply with the measured data at higher temperatures.

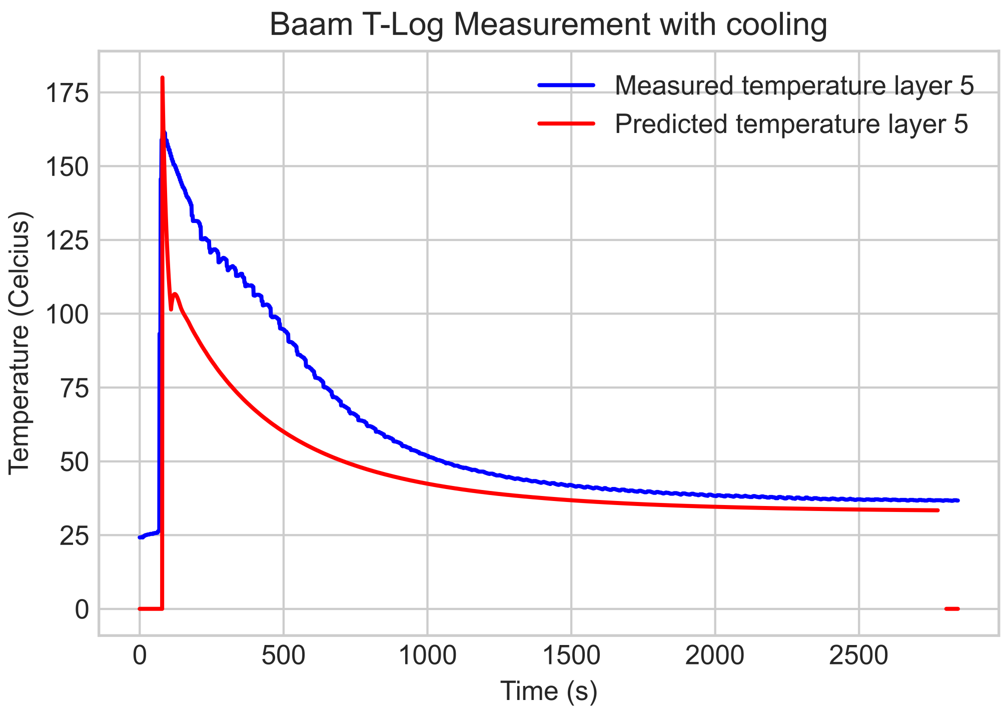


Figure 5.7: Measured temperature with thermocouple sensor and predicted temperature of layer 5

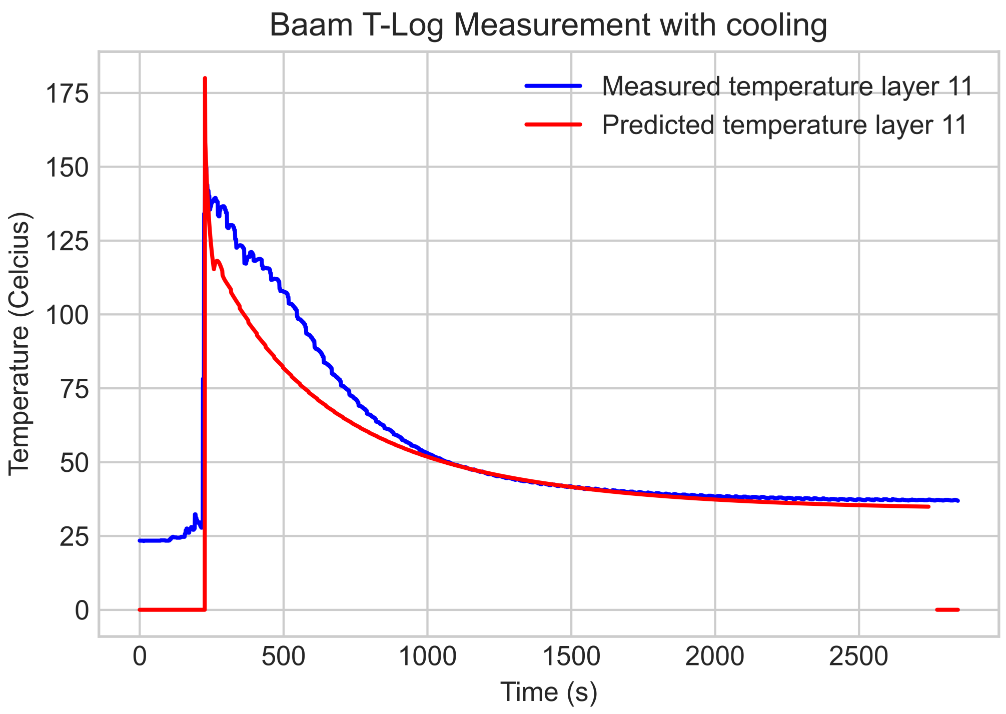


Figure 5.8: Measured temperature with thermocouple sensor and predicted temperature of layer 11

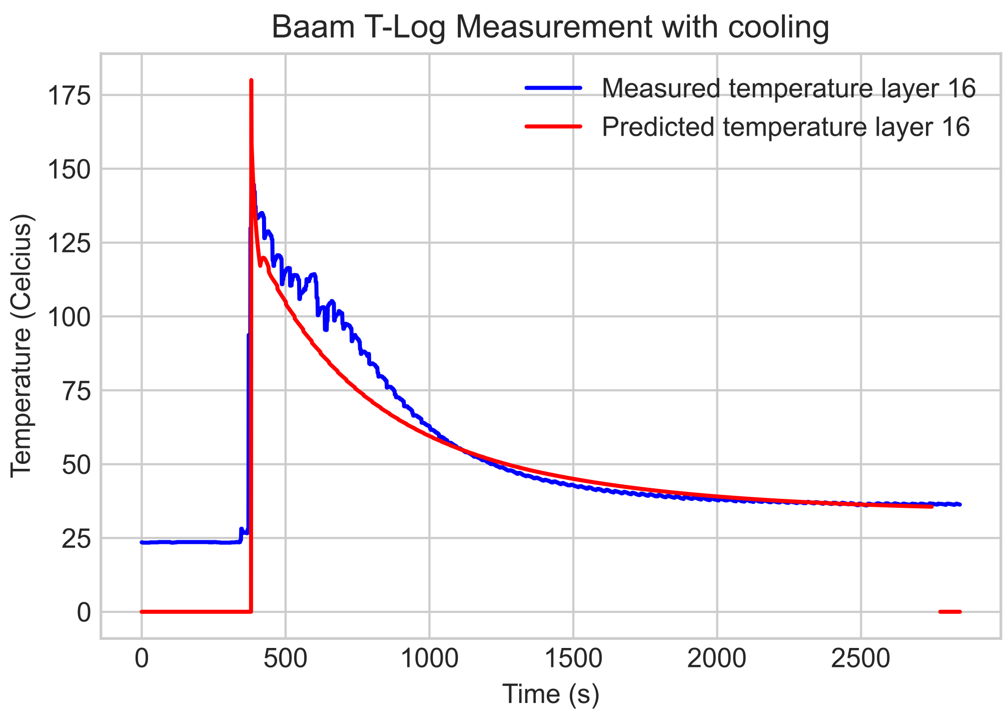


Figure 5.9: Measured temperature with thermocouple sensor and predicted temperature of layer 16

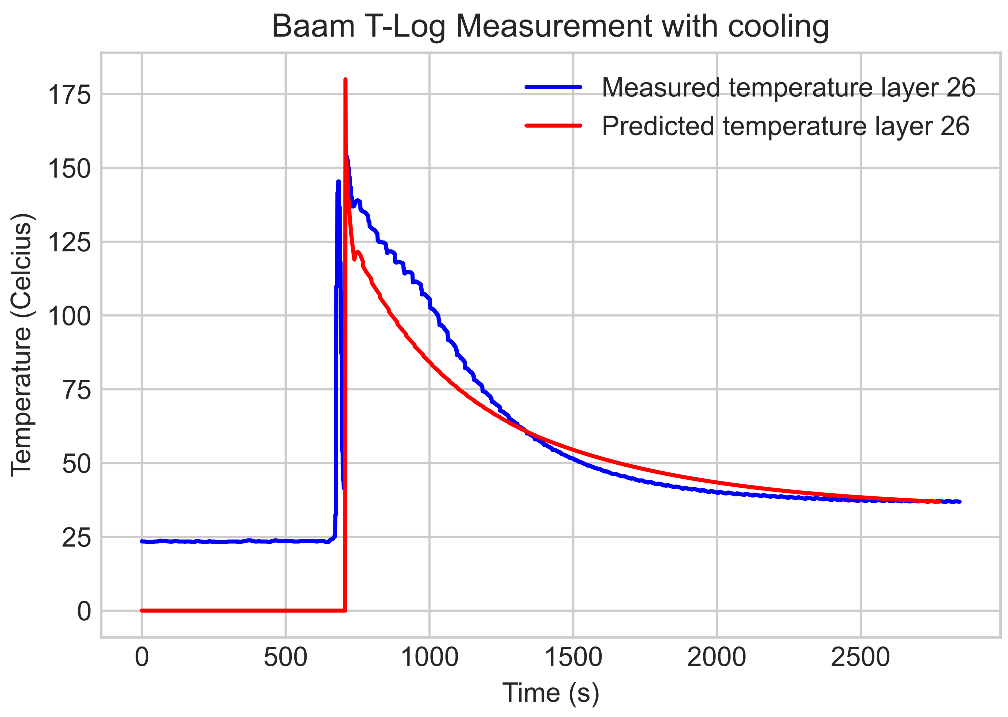


Figure 5.10: Measured temperature with thermocouple sensor and predicted temperature of layer 26

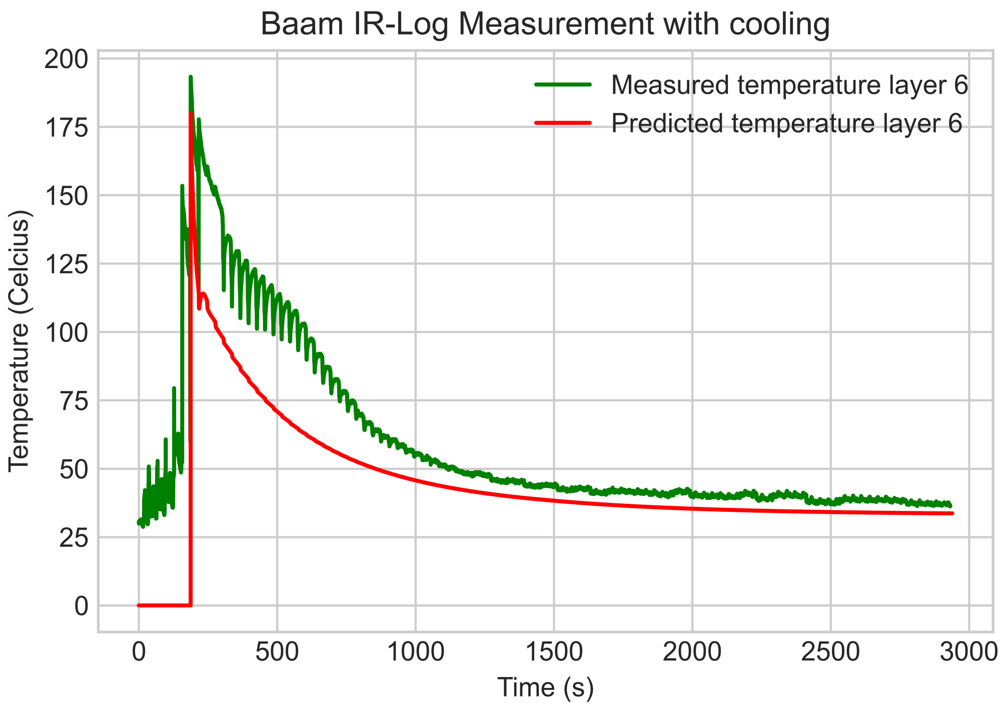


Figure 5.11: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 5

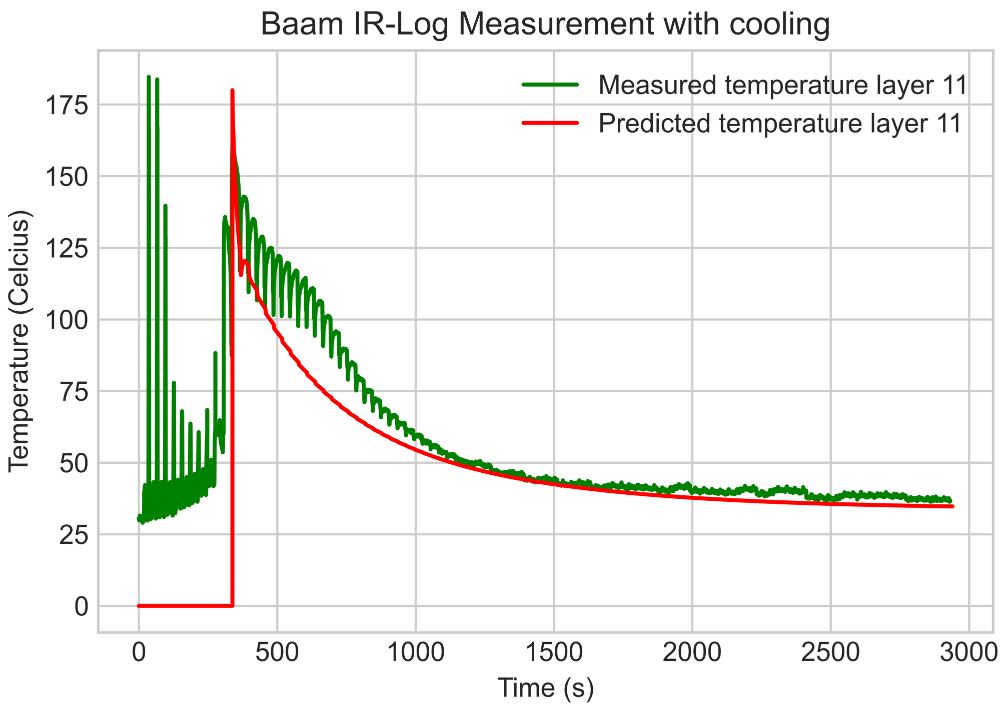


Figure 5.12: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 11

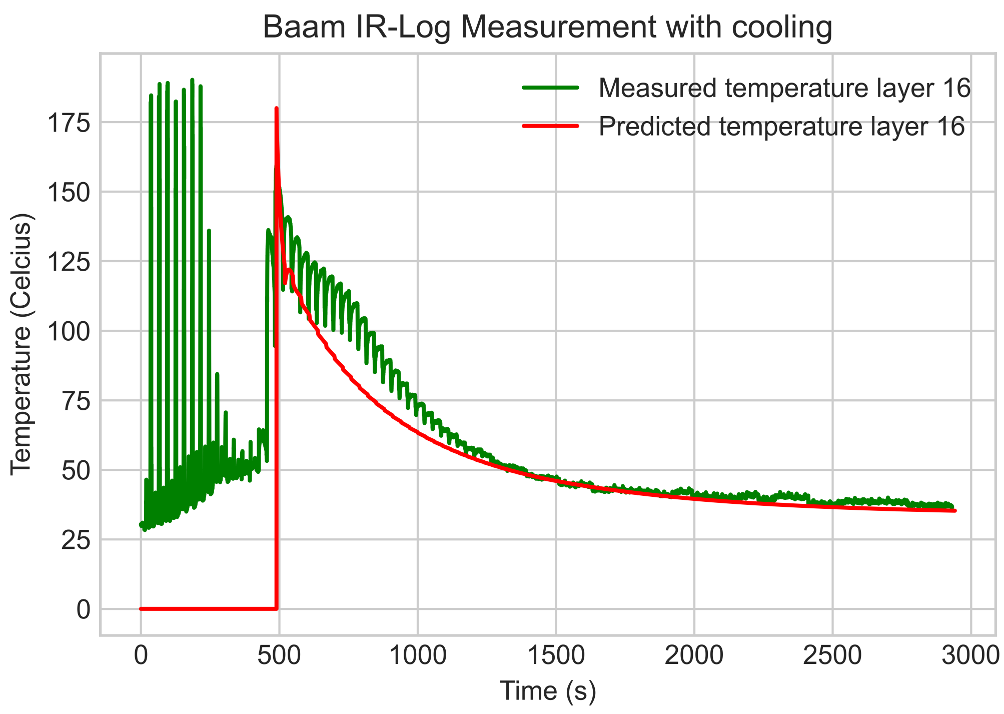


Figure 5.13: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 16

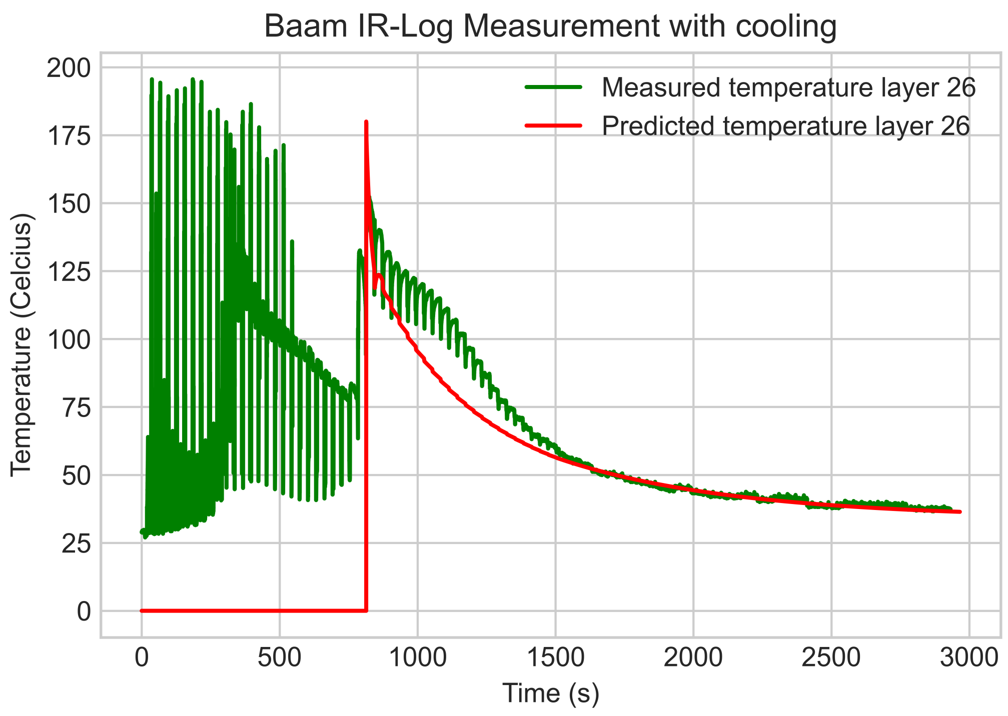


Figure 5.14: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 26

5.3.2 Experimental results of 08-06-21

Another measurement with the FLIR A35 camera at a different date was performed, in order to ensure that the measured data of the camera was reliable, since the measurements of the camera showed much noise caused by radiation of the printing nozzle. This measurement had also different input parameters. The input parameters that changed are stated in the table below. The other input parameters did not change and can be found in the table in 5.3. Figures 5.15, 5.16, 5.17, 5.18 and 5.19 show the same thermal behaviour of the measured data with the FLIR A35 camera, which means that the measured data of the FLIR A35 camera can be seen as reliable. The temperatures of the model still cool down faster at higher temperatures than the measured temperatures.

Inputs	**Values**
Ambient temperature, T_∞ [°C]	29
$\mathrm{w}_{\mathrm{int}}$ $[mm]$	8.2

Table 5.2: Change input parameters of 1-dimensional model

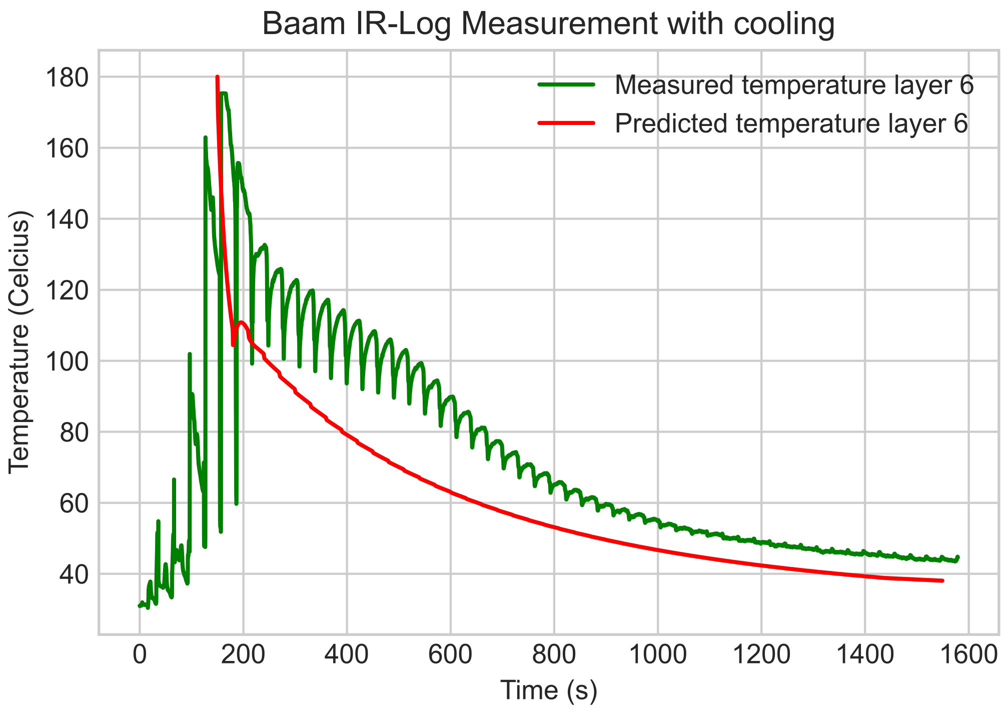


Figure 5.15: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 5

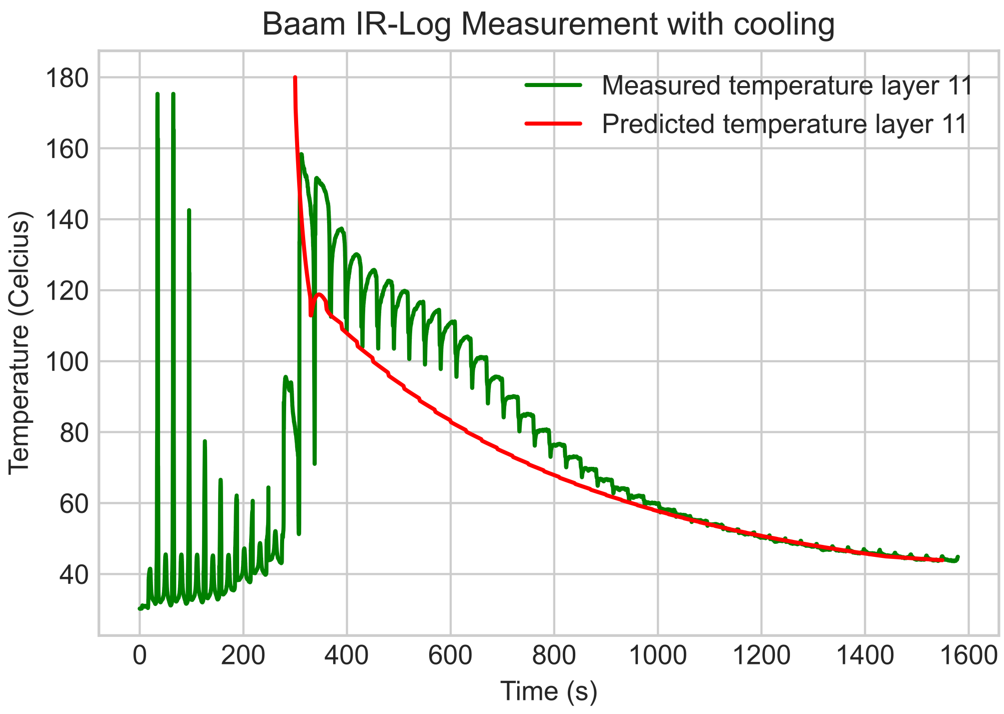


Figure 5.16: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 11

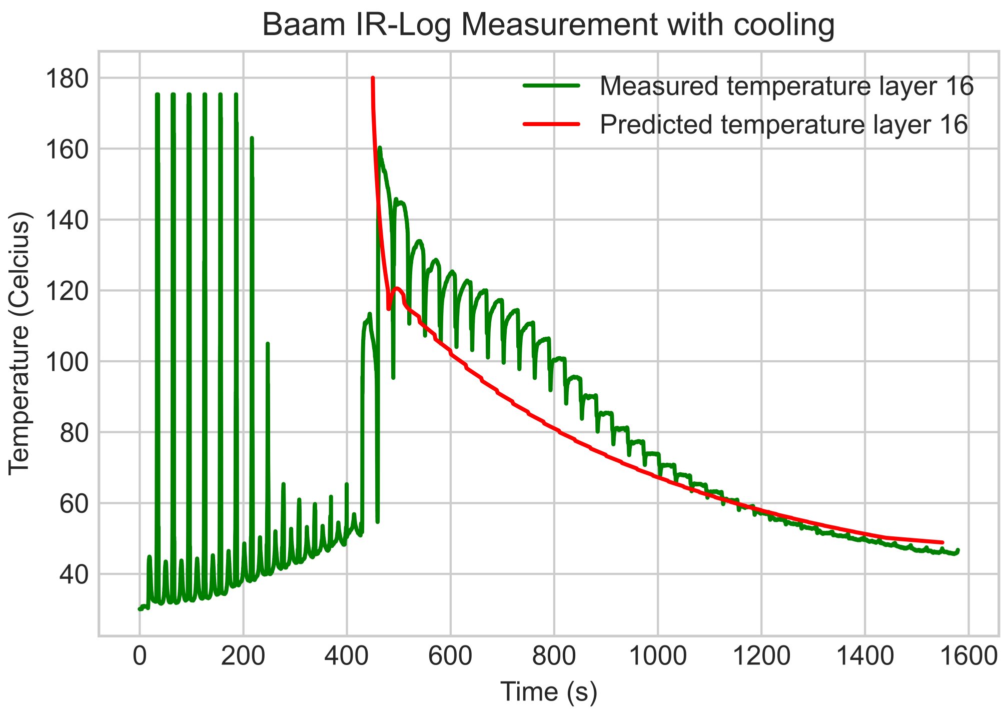


Figure 5.17: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 16

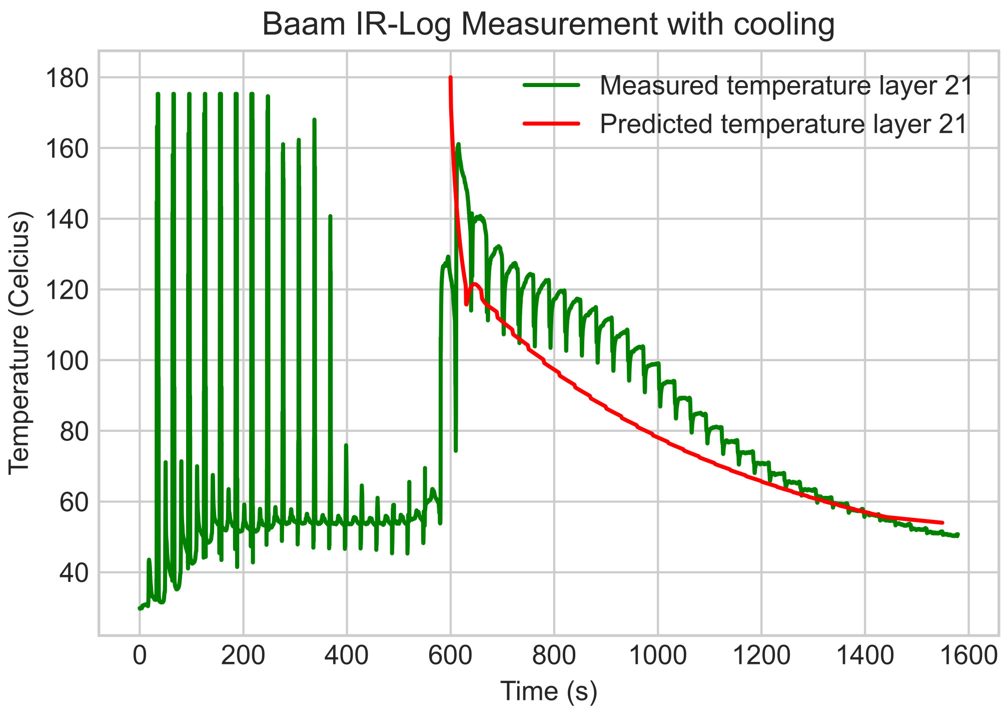


Figure 5.18: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 21

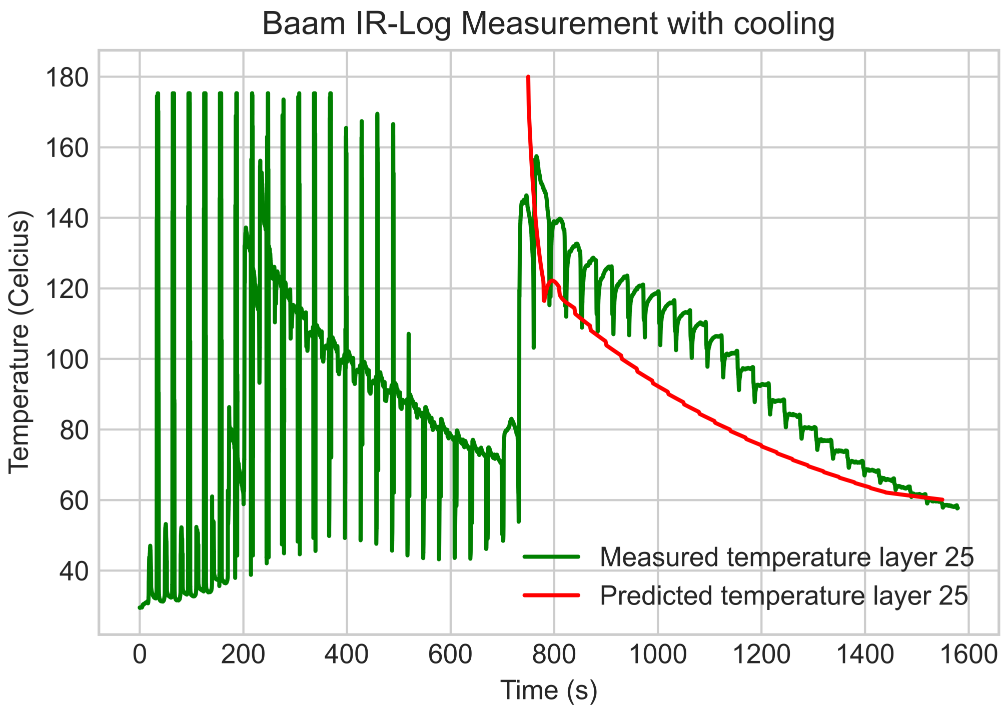


Figure 5.19: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 26

5.4 Results 2-dimensional numerical thermal model

In this section the results of the 2-dimensional numerical thermal model are discussed. The 2-dimensional model predicts the thermal behaviour across the layers over time within a thin wall with two layers next to each other. The deposition plan is depicted in subsection 3.3.2. Almost the same input parameters have been used for the print as for the 1-dimensional model. The input parameters which are changed are stated below.

Inputs	**Values**
Deposition temperature, T_{dep}[°C]	170
Ambient temperature, T_∞ [°C]	24
Temperature air-gun, T_{airgun}[°C]	33
Number of layers, n_L	56
Nodal spacing y-direction, Δy[mm]	6

Table 5.3: Input parameters 2-dimensional model

The temperature prediction plot of the first 15 layers is depicted below. There can be seen that the layers temperature increase more often when a new neighbour layer is deposited. This is also logical, since a neighbour layer is not only deposited on top in the z-direction, but also in the y-direction.

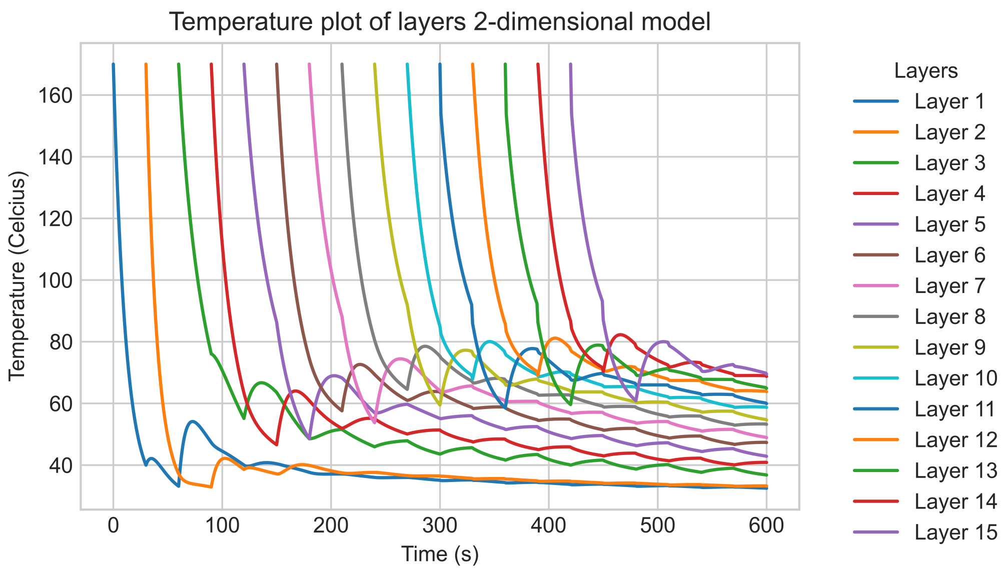


Figure 5.20: Nodal temperature prediction over time for the first 15 layers of the product with Δt = 1/30

5.4.1 Experimental results of 24-06-2021

On 24-06-2021 an experiment was performed, in order to test the accuracy of the 2-dimensional thermal numerical model. For the simulation, a time step ($\Delta t = 1/30$) was used, since the measurements of the FLIR A35 camera were taken with 30 frames per second. Unfortunately, only two measurements could be used to compare the measured data with the simulation results, since the first three measurements gave a temperature of 0. Therefore, these measurements could not be used. Figures 5.21, 5.22 show the measured temperatures and predicted layers at number 35 and 45 respectively. The measured data show a strange behaviour of the temperature, since the temperature at the deposition time is not around 170 °C. Instead the temperature is at 80 °C, which could mean that this data is also not correct or disturbed by noise of the environment or a software error. However, the contour of the predicted temperature plot for layer 35 and 45 is following the contour of the measured data.

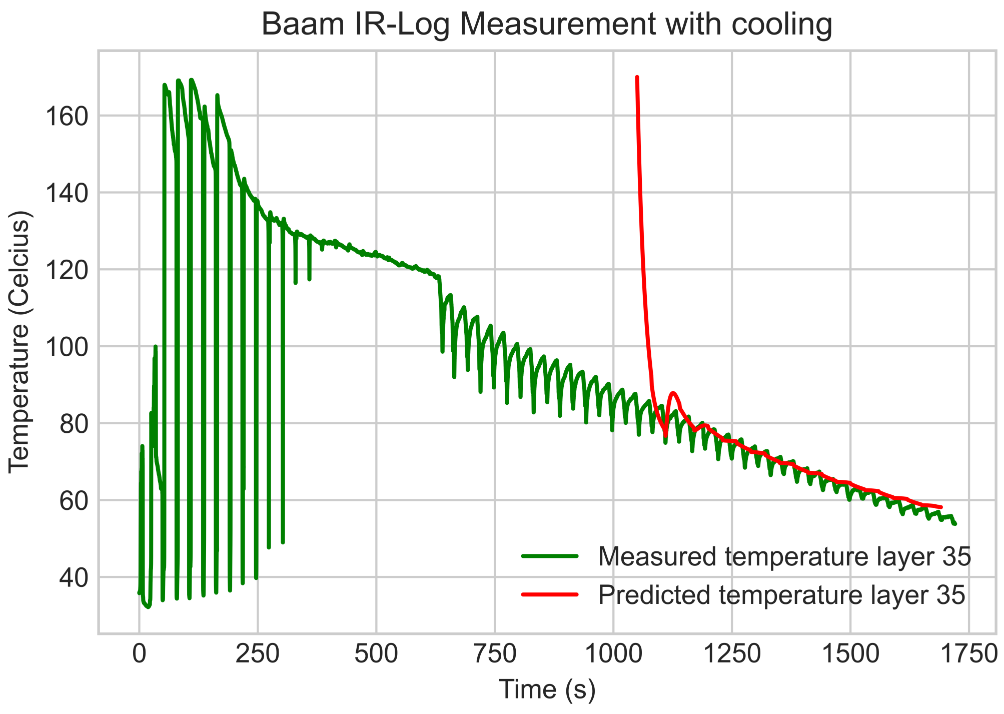


Figure 5.21: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 35

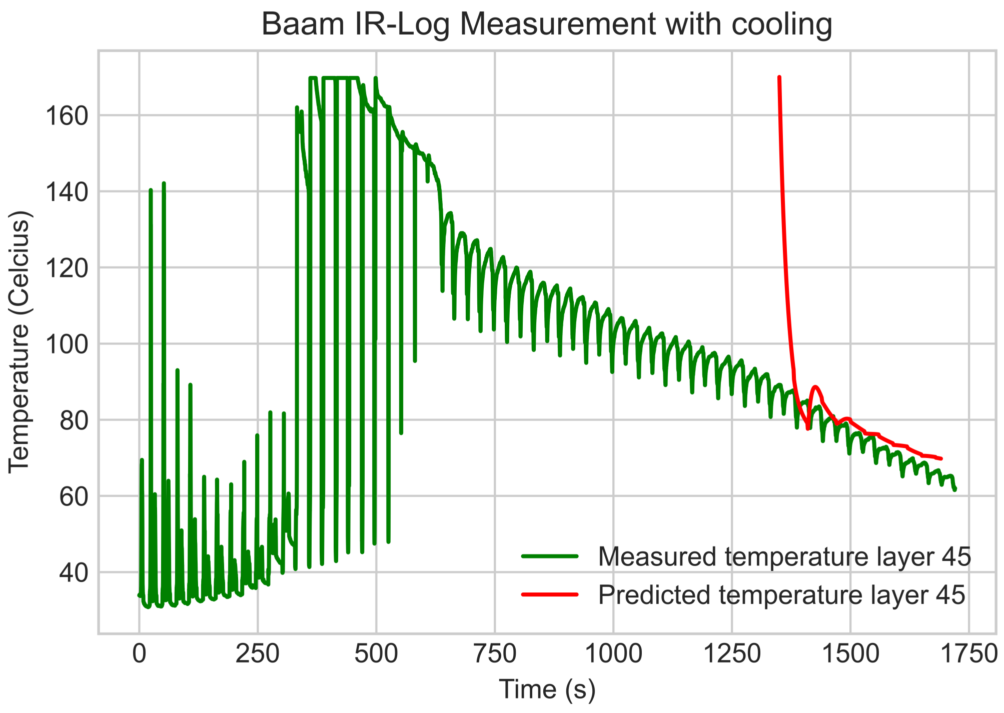


Figure 5.22: Measured temperature with FLIR A35 thermal camera and predicted temperature of layer 45

5.5 Results tensile tests

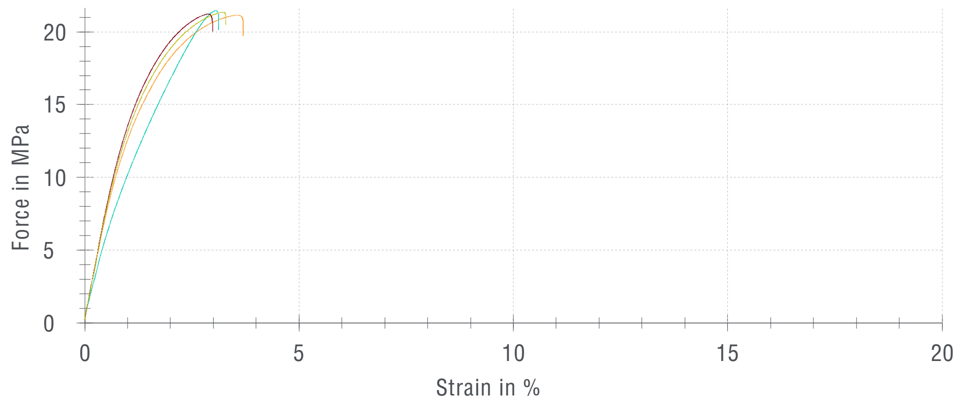
First, product T1 is simulated in the 1-dimensional model with the input parameters below. The material specifications are left out the table, since the material did not change.

Inputs	Values
Deposition temperature, T_{dep}[°C]	180
Ambient temperature, T_∞ [°C]	25
Δz $[mm]$	2.5
w_{int} $[mm]$	6.958
Temperature air-gun, T_{airgun}[°C]	40
Number of layers, n_L	48
Layer time, t_p $[mm]$	30.39
Printing speed v_{airgun} $[mm/s]$	25

Table 5.4: Input parameters 1-dimensional model for tensile test

According to the simulation the temperature of the previous layer T_{top} before a new layer is deposited is around 117, which is according to Basell [38] lower than the melt temperature (between 120 and 130). Product T4 is also simulated in the 1-dimensional model. Only the layer time is changed to 22.79 seconds, such that T_{top} increases. The simulation results showed that T_{top} is around 127 . The results of the tensile tests show that product T1 has a fracture around 21 MPa and T4 has a fracture around 22.5 Mpa, which shows that product T4 has a higher strength. However, T_{top} of product T4 is just above the crystalline melt temperature, decreasing the layer time could increase the strength even more. Nevertheless, decreasing the layer time too much could result in a collapse of the printed product, since to much heat is brought in the product in a small time period.

Series graph:

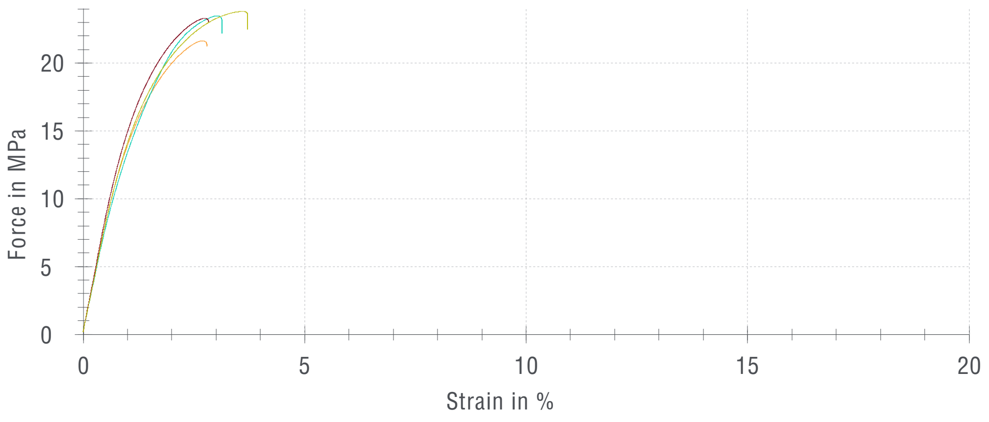


Test results:

Legend	No.	Specimen designation	E_t MPa	$\overline{\sigma 0.2}$ MPa	σ_Y MPa	ε_Y %	σ_M MPa	ε_M %	σ_B MPa	ε_B %	h mm
🟧	1	X+	1460	13,29	-	-	21,2	3,6	19,7	3,7	8,37
🟦	2	Y+	1150	11,20	-	-	21,5	3,1	20,1	3,1	8,34
🟥	3	X-	1510	14,88	-	-	21,2	2,9	20,0	3,0	8,32
🟨	4	Y-	1490	14,22	-	-	21,3	3,2	20,5	3,3	8,47

Figure 5.23: Tensile test report of product T1

Series graph:



Test results:

		Specimen designation	E_t	$\overline{\sigma 0.2}$	σ_Y	ε_Y	σ_M	ε_M	σ_B	ε_B	h
Legend	No.		MPa	MPa	MPa	%	MPa	%	MPa	%	mm
🟧	1	X+	1580	15,07	-	-	21,6	2,7	21,3	2,8	6,96
🟦	2	Y+	1520	14,87	-	-	23,5	3,0	22,2	3,1	6,95
🟥	3	X-	1650	16,77	-	-	23,3	2,7	23,0	2,8	6,94
🟨	4	Y-	1550	15,98	-	-	23,8	3,6	22,5	3,7	6,98

Figure 5.24: Tensile test report of product T4

Chapter 6: Conclusion

During this research, the questions *How to build a validated thermodynamics model that describes the deposition process?* and *How can the validated thermodynamics model be implemented into the Smart Polymer Granulate 3D Welding System?* were researched. To answer these questions two models (1 and 2-dimensional) are proposed which describe the deposition process.

The results of the temperature measurements of the 1-dimensional model show that the predicted temperature of the 1-dimensional comply with the measured data at lower temperatures. Nevertheless, one should not forget the fact that at higher temperatures $75 < \mathrm{T} < 125$ ℃ the predicted temperatures deviate slightly. A possible reason for this could be the radiation of the printing nozzle or that the thermal conductivity coefficient (k) deviates in practice from the theory. However, this must be a very high deviation, since the coefficient was increased and decreased over the interval $0.02 < \mathrm{k} < 4.0$ and the predicted results still did not comply with the measured data at higher temperatures.

The results of the temperature measurements of the 2-dimensional model are deviating from the measurements of the 1-dimensional model. A possible reason for this could be that the measurements are disturbed by noise from the environment or a software error. To ensure this, another test should be performed in a further research.

The results of the tensile tests show that the strength of the printed products increase if $\mathrm{T_{top}} > T_m$. This means that the hypothesis: *If the temperature of the previous layer* $\mathrm{T_{top}}$ *is below the crystalline melting temperature* T_m *before the current layer is deposited, then the strength of the material is decreased, due to poor inter layer bonding* can be accepted.

In this research, several important environmental factors and process parameters are stated. A higher ambient temperature and a higher deposition temperature increases the time before $\mathrm{T_{top}} < T_m$. Furthermore, the literature review and the observations of the models and experiments show that the forced convection has a significant impact on the temperature degradation of the layer.

The proposed models can be implemented into the Smart Polymer Granulate 3D Welding System by predicting the critical layer time $\mathrm{T_{top}} < T_m$. The proposed models can be used to gain insight in the thermal behaviour across the printed layers. Furthermore, NHL Stenden can use the proposed models to experiment by adjusting several input parameters with an interface that is developed.

This research has shown that the proposed models can be used to increase the quality and strength of the printed geometries. The models help to indicate the critical layer time before $\mathrm{T_{top}} < T_m$. From the experiments there can be concluded that the proposed 1-dimensional model is valid and can be used to optimize the quality and strength of the printed part. Therefore, the hypothesis *The thermal model can be validated with the experiment that is set up and can be used for the process to optimize the quality of the printed part* can be accepted. For the 2-dimensional model, the measured results are different from the measured results obtained in the 1-dimensional model. However, the same material properties and thermal laws have been used in this model. Therefore, it is expected that this model is also valid. Nevertheless, further research should be conducted in some deviating temperatures of the 1-dimensional model and the deviating measurements of the 2-dimensional model.

Chapter 7: Discussion

The proposed models give a good insight in the thermal behaviour across the printed layers during the deposition process. The error of the predicted temperatures and the measured temperatures are relatively small for lower temperatures in the 1-dimensional model. However, at higher temperatures the error increases. A possible reason for this could be that the printing nozzle is radiating heat to the printed layers or that the thermal conduction coefficient is deviating significantly from the theory. Furthermore, the forced convection of the air-gun plays a significant role, since it decreases the temperature of the printed layers. Two important factors for the strength of the printed products are the ambient air temperature and the layer time. A large layer time results in poor strength, due to bad inter layer bonding of the material. The proposed models help to determine the critical layer time and the effect of changing several parameters can be simulated with the proposed models.

7.1 Scientific contribution

This research contributes to the existing knowledge of thermal analysis in BAAM processes. In the literature, several models are proposed that predict the thermal behaviour of fused filament deposition processes. However, only a small number of researches focus on the thermal behaviour prediction of BAAM processes. Most of the researches focus on predicting the thermal behaviour in AM processes, which are different in a way that these processes print much smaller products. This research differs from the existing researches for analyzing the thermal behaviour in BAAM processes, since this research also includes forced convection in the models. In this research Hostacom G3 N01 is used for printing, which also has never been mentioned in the existing literature. Furthermore, in this research also a 2-dimensional thermal numerical model is proposed for a BAAM process. The literature that was researched did not mention a 2-dimensional thermal numerical model that was used for a BAAM process. Therefore, this research contributes to the literature, since it shows new insights in predicting the thermal behaviour for BAAM processes with forced convection and with double layers in the y-direction.

7.2 Limitations

The limitations of the proposed models are that these models can only predict the thermal behaviour of simple symmetrical products with the same layer time for each layer. NHL Stenden is currently not printing complex geometries. Therefore, it is not a limitation yet. Nevertheless, in the future NHL Stenden might want to print more complex geometries.

7.3 Recommendations

This research gives insight in the thermal behaviour across the printed layers during the printing process. A 1-dimensional and 2-dimensional numerical thermal model are constructed during this research. Nevertheless, much research should be conducted in how to make the models more applicable for complex geometries.

7.3.1 3-dimensional model

To increase the design flexibility of the printed products in the future, a 3-dimensional numerical thermal model could be a solution. The 3-dimensional model would also be based on the energy balance equation and could be simulated with the help of discrete event simulation. However, in this case an extra dimension is added, which is shown below. Furthermore, a flowchart for this model is depicted below. The input for this model could be the Cartesian X,Y,Z coordinates of a discretized toolpath. The model would then simulate the thermal behaviour across the printed layers according to the predefined toolpath. The simulation could go on until the top layer temperature (T_{top}) is higher than the crystalline melt temperature of the material (T_m).

$$\rho c_p \frac{\partial T(x,y,z,t)}{\partial t} = k\left(\frac{\partial^2 T(x,y,z,t)}{\partial x^2} + \frac{\partial^2 T(x,y,z,t)}{\partial y^2} + \frac{\partial^2 T(x,y,z,t)}{\partial z^2}\right) \tag{7.1}$$

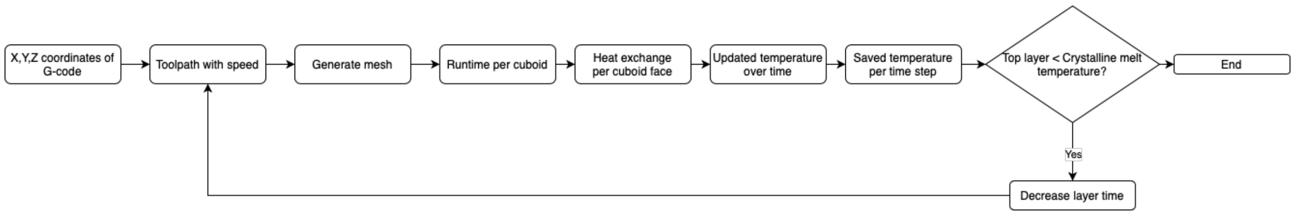

X,Y,Z coordinates of G-code
Toolpath with speed
Generate mesh
Runtime per cuboid
Heat exchange per cuboid face
Updated temperature over time
Saved temperature per time step
Top layer < Crystalline melt temperature?
End
Yes
Decrease layer time

Figure 7.1: Flowchart potential 3-dimensional model

7.3.2 Control process with model

A control loop with the FLIR A35 camera could also be a possible solution to prevent bad inter layer bonding or collapses of the printed products. The camera could detect when to little or to much heat is in the printed product. This detection signal could be sended to the Kuka KR16-2 manipulator and Xtrution granulate extruder to put more or less heat in the printed product.

Chapter 8: Appendix

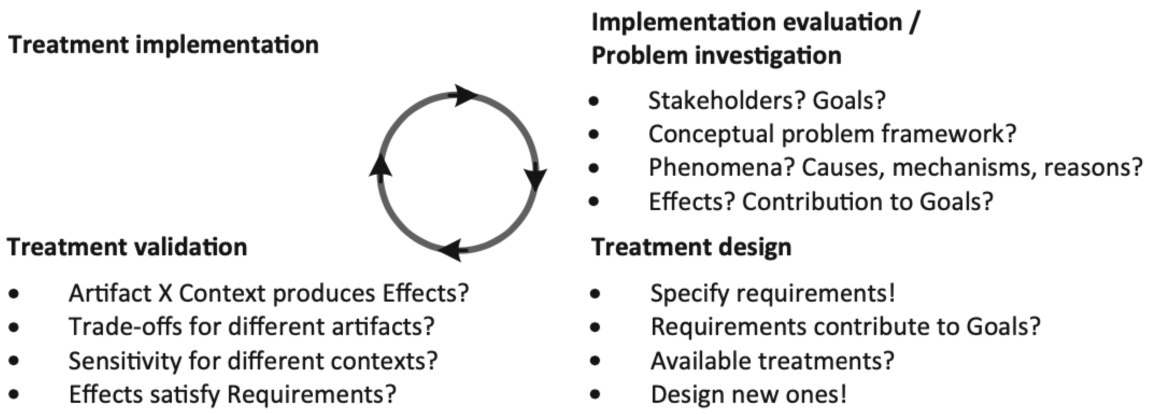

Treatment implementation
Implementation evaluation / Problem investigation
• Stakeholders? Goals?
• Conceptual problem framework?
• Phenomena? Causes, mechanisms, reasons?
• Effects? Contribution to Goals?
Treatment validation
• Artifact X Context produces Effects?
• Trade-offs for different artifacts?
• Sensitivity for different contexts?
• Effects satisfy Requirements?
Treatment design
• Specify requirements!
• Requirements contribute to Goals?
• Available treatments?
• Design new ones!

Figure 8.1: Engineering cycle from Wieringa [10]

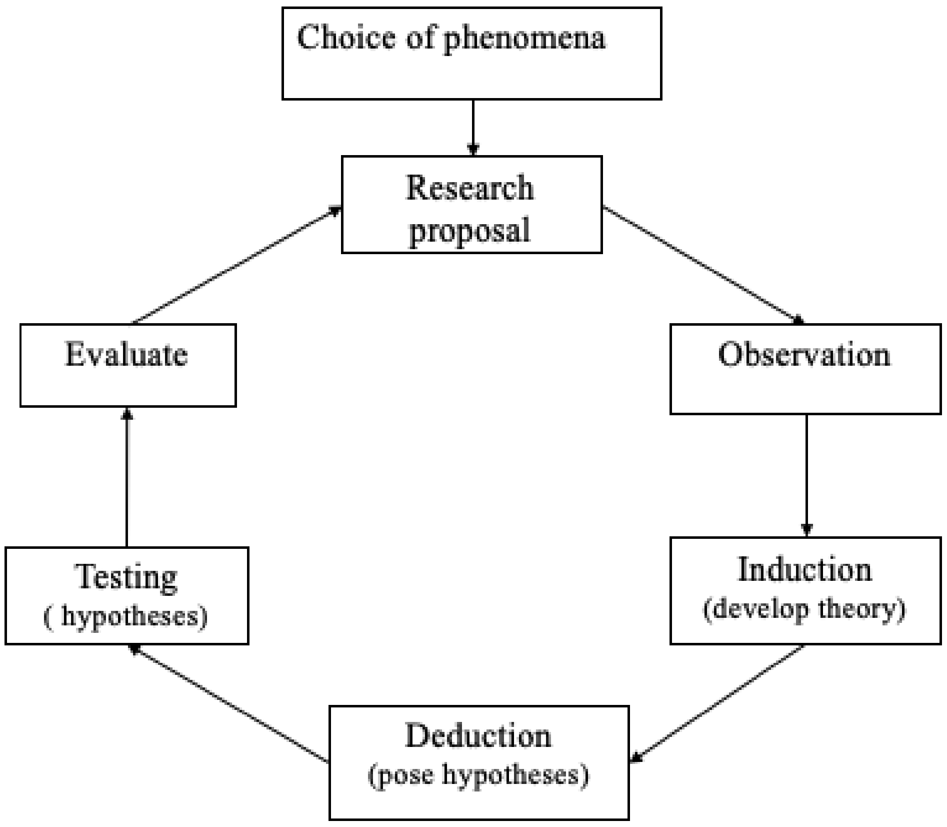


Figure 8.2: Empirical cycle of van Aken [11]

8.1 Specifications of measuring equipment

According to FLIR [37], the camera has the following imaging and data processing specifications.

Imaging and optical data	
IR resolution	320×256 pixels
Thermal sensitivity/NETD	$< 0.05°C@ + 30°C\,(+86°F)\,/50mK$
Field of view (FOV)	$45° \times 35°$
Minimum focus distance	2.5 cm(0.98in.)
Focal length	7.5 mm(0.30in.)
Spatial resolution (IFOV)	2.267mrad
F-number	1.4
Image frequency	60 Hz
Focus	Fixed
Detector data	
Detector type	Focal plane array (FPA), uncooled VOX microbolometer
Spectral range	$7.5 - 13\mu m$
Detector pitch	$17\mu m$
Detector time constant	Typical 12 ms
Measurement	
Object temperature range	$-25to + 100°C\,(-13to212°F)$ $-40to + 550°C\,(-40to + 1022°F)$
Accuracy	$\pm 5°C\,(\pm 9°F)$ or $\pm 5\%$ of reading

Characteristic	Symbol	Min	Typ	Max	Unit
Pressure rRange [1]	P_{OP}	-2.0	–	2.0	kPa
Supply voltage [2]	V_S	4.75	5.0	5.25	Vdc
Supply current	I_0	–	–	10	mAdc
Pressure offset [3] (10°C to 60°C) @ $V_s = 5.0$ Volts	V_{off}	0.25	0.5	0.75	Vdc
Full scale output [4] (10°C to 60°C) @ $V_s = 5.0$ Volts	V_{FSO}	4.25	4.5	4.75	Vdc
Full Scale Span[5] (10°C to 60°C) @ $V_S = 5.0$ Volts	V_{FSS}	3.5	4.0	4.5 V	Vdc
Accuracy [6] (10°C to 60°C)	–	–	±2.5[7]	±6.25	% V_{FSS}
Sensitivity	V/P	–	1.0	–	V/kPa
Response time [8]	t_R	–	1.0	–	ms
Output source current at full scale output	l_{O+}	–	0.1	–	mAdc
Warm-up time [9]	–	–	20	–	ms

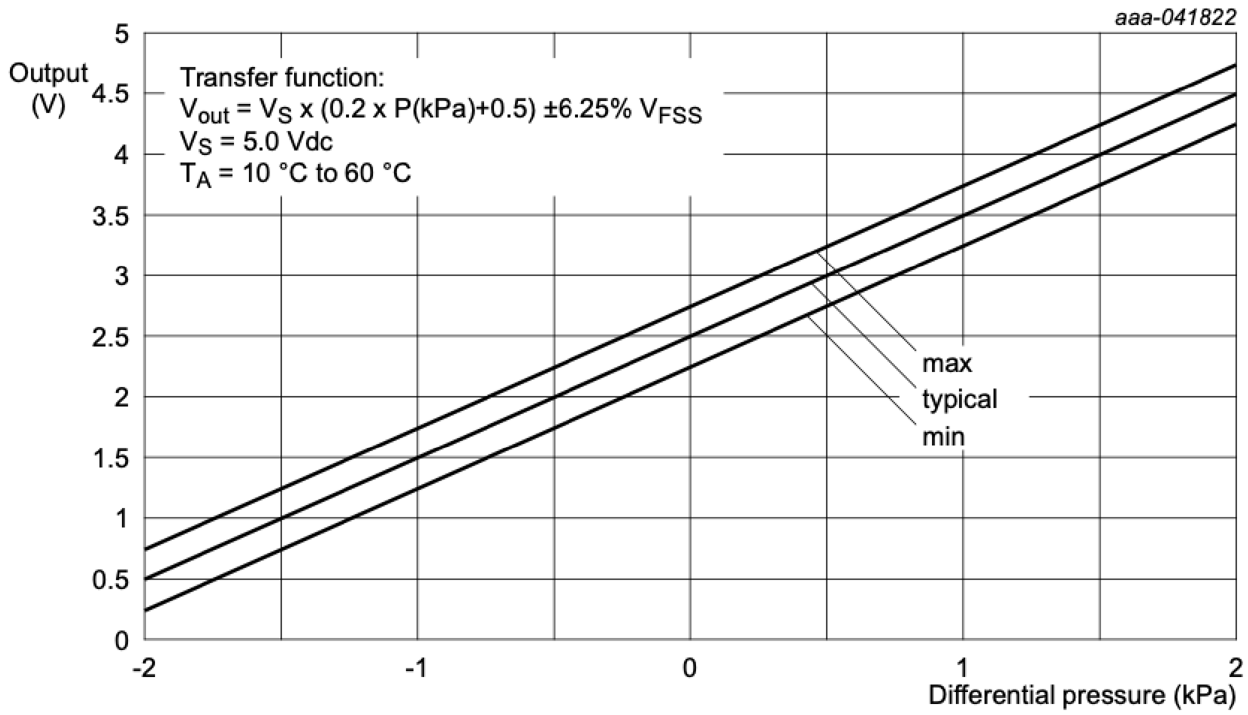


Figure 8.3: Voltage output vs Pressure difference, obtained from NXP [35]

8.2 Python code 1D-model

Listing 8.1: 2-dimensional numerical thermal model of BAAM process

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  6 11:31:12 2021

@author: twan
"""
# Inputs
import numpy as np
from numpy.linalg import inv
from matplotlib import pyplot as plt
import pandas as pd
from matplotlib.backends.backend_pdf import PdfPages
from datetime import datetime
#from pytexit import py2tex
plt.style.use('seaborn-whitegrid')

n_l = 8;
t_p = 30;         # seconds
dt = 0.03;            # seconds
v = 1500/60;      # mm/s
dz = 0.0025;              # meters
b = 0.00695;      # meters
Tdep = 180;       #Celcius
T_inf = 25;       #Celcius
```

```
Tb = 50;          #Celcius
T_airgun = 40;  #Celcius
Cooldown_period = 200    #Seconds




#Material properties
k = 0.3;
rho = 1150;
c_p = 2200;
e = 0.94;

# Other constants
K = 273;
d = 0.0025;
Bolt = 5.6700e-08;

n_t = round(t_p/dt);

shapeT = (1+n_t,n_l+2)
shapeTnew = (n_t*(n_l+1),n_l+2)
T = np.zeros(shapeT);
Tnew = np.zeros(shapeTnew);
h_s = np.zeros(n_t*(n_l+1));
T[0,0] = Tdep;

h_store = np.zeros(n_t)
hradm = e*Bolt*(Tdep+T_inf)*(Tdep^2+T_inf^2);
qcond= np.zeros((n_t,n_l+1));
qconv= np.zeros((n_t,n_l+1));
hrad = np.zeros((n_t,n_l));
iterations = 0
iterationstotal = 0
iterations =  np.arange(n_t*(n_l+1))*dt;

#Airgun
r = 22.5;
dist = (r/v);
dist_hole = (3/v)
y_hole = int(np.ceil(dist_hole/dt))
y = int(np.ceil(dist/dt))
g = 9.81; #m/s^2
# Define parameters for air
Beta = 3.44e-03 #1/T_inf;
Viscosity = 1.493e-05; #m^2/s
Diffusivity = 2.074e-05; #m^2/s
Thermal_cond = 0.02514; # W/m*K
Pr = Viscosity/Diffusivity;
i_top_begin = np.arange(int(y/2),y_hole)
i_top_last =  np.arange(n_t-int(y/2),n_t)
```

```
top_air = np.append(i_top_begin,i_top_last)



Vel = pd.read_csv ('/Users/twan/Documents/Velocity/Velocity over
   distance/V_distance_log.csv', sep=';')
Pitot_tube = Vel.mean(axis = 0)
Displacement = np.arange(4,23)*25


coeff = np.polyfit(Displacement, Pitot_tube,2)
m_pitot = coeff[0]
b_pitot = coeff[1]
Vtrend = np.linspace(Displacement[0],Displacement[-1],100)
Ttrend = np.polyval(coeff, Vtrend)

Displacement_total = np.arange(0,23)*25
f = coeff[0]*pow(Displacement_total,2) + coeff[1]*Displacement_total
    + coeff[2]


T_cooldown = np.zeros((1+Cooldown_period,n_l+2))
hradc = np.zeros((Cooldown_period,n_l));
qcondc= np.zeros((Cooldown_period,n_l+1));
qconvc= np.zeros((Cooldown_period,n_l+1));
shapeT_total = (n_t*(n_l+1)+Cooldown_period,n_l+2)
T_total = np.zeros(shapeT_total);
shapeh_total = (n_t*(n_l+1)+Cooldown_period)
h_total = np.zeros(shapeh_total);
iterationstotal =  np.arange(n_t*(n_l+1)+Cooldown_period)*dt;
Velocity_air = np.zeros((n_l))
Re = np.zeros((n_l))
Nusselt_airgun = np.zeros((n_l))

h_store_side = np.zeros(n_l)
h_store_top = np.zeros(n_t)
h_side = np.zeros(n_l)
h_cooldown = np.zeros(Cooldown_period)



# Number of layers
for n in range(0,n_l):
    #qdep = (b*d*rho*c_p*Tdep)/(t_p);
    L = (n+1)*dz;

     if n > 4:
         L = (n+1)*dz;
         #forced convection
         Velocity_air[n] =  coeff[2] #coeff[0]*pow(dz*(n-o),2) +
            coeff[1]*dz*(n-o) + coeff[2]
         Re[n] = (Velocity_air[n]*dz)/Viscosity;

         Nusselt_airgun[n] = pow(0.332*Re[n], 1/2)*pow(Pr, 1/3);
         #Nusselt_airgun = (pow(0.037*Re, 4/5)-871)*pow(Pr, 1/3);

         #Nusselt_airgun[n] = pow(0.102*Re[n], 0.675)*pow(Pr, 1/3);
         h_side[n] = (Nusselt_airgun[n]*Thermal_cond)/dz;
         for o in range(1,n+1):
             L_b = (o+1)*dz;
             #forced convection
             Velocity_air[n-o] =  coeff[0]*pow(dz*(o),2) + coeff[1]*
                dz*(o) + coeff[2]

             Re[n-o] = (Velocity_air[n-o]*L_b)/Viscosity;

             Nusselt_airgun[n-o] = pow(0.332*Re[n-o], 1/2)*pow(Pr,
                1/3);
             #Nusselt_airgun = (pow(0.037*Re, 4/5)-871)*pow(Pr, 1/3);

             #Nusselt_airgun[n] = pow(0.102*Re[n], 0.675)*pow(Pr,
                1/3);
             h_side[n-o] = (Nusselt_airgun[n-o]*Thermal_cond)/L_b;
     elif n == 0:

         #forced convection
         Velocity_air[n] =  coeff[0]*pow(dz*(n),2) + coeff[1]*dz*(n)
            + coeff[2]

         Re[n] = (Velocity_air[n]*L)/Viscosity;

         Nusselt_airgun[n] = pow(0.332*Re[n], 1/2)*pow(Pr, 1/3);
         #Nusselt_airgun = (pow(0.037*Re, 4/5)-871)*pow(Pr, 1/3);

         #Nusselt_airgun[n] = pow(0.102*Re[n], 0.675)*pow(Pr, 1/3);
         h_side[n] = (Nusselt_airgun[n]*Thermal_cond)/L;


     for i in range(0,n_t):
         #Natural convection
         T_last = T[i,0:n+1];
         Ts = np.mean(T_last, axis = 0);

         Gr = (g*Beta*(Ts -T_inf)*pow(L,3))/(pow(Viscosity,2));
         Ra = Gr*Pr;

         Nu_L1 = 0.825+(0.387*pow(Ra,1/6))/(pow((1+pow(0.492/Pr,9/16)
            ),8/27))
         Nu_L = pow(Nu_L1,2)
         h = (Nu_L*Thermal_cond)/L
```

```
        h_store[i] = h
        #if n > 5:


        #else:
           # h_side[n] = h
        if i in top_air and n > 4:
            Velocity_air_top = coeff[0]*pow(dz,2) + coeff[1]*dz +
               coeff[2]
            Re_top = (Velocity_air_top*dz)/Viscosity;
            Nusselt_airgun_top = pow(0.102*Re_top, 0.675)*pow(Pr,
               1/3);
            #if n > 5:
            h_top = (Nusselt_airgun_top*Thermal_cond)/dz;
            h_store_top[i] = h_top
            #else:
                #h_top = h

        else :
            h_top = h

        if i < y and n > 4:
            ## 1 layer condition
            if n == 0 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))*(pow(K+
                   T[i,0],2)+pow(K+T_airgun,2));
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]);
                qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(T_airgun -
                   T[i,0])+b*(h_top+hrad[i,0])*(T_airgun-T[i,0]);
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                   c_p*b*dz)))+T[i,0]; # bottom node

## 2 layers condition
            elif n == 1 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))*(pow(K+
                   T[i,0],2)+pow(K+T_airgun,2));
                hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_airgun))*(pow(K+
                   T[i,1],2)+pow(K+T_airgun,2));

                qcond[i,1] = ((k*b)/dz)*(T[i,0]-T[i,1]);
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
                   T[i,1]-T[i,0]);



                qconv[i,1] = 2*dz*(h_side[1]+hrad[i,1])*(T_airgun -
                   T[i,1])+b*(h_top+hrad[i,1])*(T_airgun-T[i,1]);
                qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(T_airgun -
                   T[i,0]);

                T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
                c_p*b*dz)))+T[i,1]; #Top node
            T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                c_p*b*dz)))+T[i,0]; # bottom node

## 3 or more layers condition
        elif n > 1 :
            hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_airgun))*(pow(K+
                T[i,n],2)+pow(K+T_airgun,2));
            qcond[i,n] = ((k*b)/dz)*(T[i,n-1]-T[i,n]);
            qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_airgun - T[
                i,n])+b*(h_top+hrad[i,n])*(T_airgun-T[i,n]);

            T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(rho*
                c_p*b*dz)))+T[i,n]; #Top node


            for m in range(1,n):
                if n-m > 4:
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_airgun))*(pow(K+T[i,n-m],2)+pow(K+
                        T_airgun,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-1]-T[i,n-
                        m]) + ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m]);
                    qconv[i,n-m] = 2*dz*(h_side[n-m]+hrad[i,n-m
                        ])*(T_airgun - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n-m])
                        *((dt/(rho*c_p*b*dz))))+T[i,n-m]; # mid
                        node

                    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))
                        *(pow(K+T[i,0],2)+pow(K+T_airgun,2));
                    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*
                        b)/dz)*(T[i,1]-T[i,0]);
                    qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(
                        T_airgun - T[i,0]);
                    T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt
                        /(rho*c_p*b*dz)))+T[i,0]; # bottom node
                else:
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_airgun))*(pow(K+T[i,n-m],2)+pow(K+
                        T_airgun,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-1]-T[i,n-
                        m]) + ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m]);
                    qconv[i,n-m] = 2*dz*(h+hrad[i,n-m])*(
                        T_airgun - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n-m])
                        *((dt/(rho*c_p*b*dz))))+T[i,n-m]; # mid
                        node

                    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))
```

```
                        *(pow(K+T[i,0],2)+pow(K+T_airgun,2));
                    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*
                        b)/dz)*(T[i,1]-T[i,0]);
                    qconv[i,0] = dz*(h+hrad[i,0])*(T_airgun - T[
                        i,0]);
                    T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt
                        /(rho*c_p*b*dz)))+T[i,0]; # bottom node

        elif (n_t- i < y and n > 4):
            ## 1 layer condition
            if n == 0 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))*(pow(K+
                    T[i,0],2)+pow(K+T_airgun,2));
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]);
                qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(T_airgun -
                    T[i,0])+b*(h_top+hrad[i,0])*(T_airgun-T[i,0]);
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,0]; # bottom node

 ## 2 layers condition
            elif n == 1 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))*(pow(K+
                    T[i,0],2)+pow(K+T_airgun,2));
                hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_airgun))*(pow(K+
                    T[i,1],2)+pow(K+T_airgun,2));

                qcond[i,1] = ((k*b)/dz)*(T[i,0]-T[i,1]);
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
                    T[i,1]-T[i,0]);



                qconv[i,1] = 2*dz*(h_side[1]+hrad[i,1])*(T_airgun -
                    T[i,1])+b*(h_top+hrad[i,1])*(T_airgun-T[i,1]);
                qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(T_airgun -
                    T[i,0]);

                T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,1]; #Top node
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,0]; # bottom node

## 3 or more layers condition
            elif n > 1 :
                hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_airgun))*(pow(K+
                    T[i,n],2)+pow(K+T_airgun,2));
                qcond[i,n] = ((k*b)/dz)*(T[i,n-1]-T[i,n]);
                qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_airgun - T[
                    i,n])+b*(h_top+hrad[i,n])*(T_airgun-T[i,n]);

                T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,n]; #Top node


                for m in range(1,n):
                    if n-m > 4:
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_airgun))*(pow(K+T[i,n-m],2)+pow(K+
                        T_airgun,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-1]-T[i,n-
                        m]) + ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m]);
                    qconv[i,n-m] = 2*dz*(h_side[n-m]+hrad[i,n-m
                        ])*(T_airgun - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n-m])
                        *((dt/(rho*c_p*b*dz))))+T[i,n-m]; # mid
                        node

                    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))
                        *(pow(K+T[i,0],2)+pow(K+T_airgun,2));
                    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*
                        b)/dz)*(T[i,1]-T[i,0]);
                    qconv[i,0] = 2*dz*(h_side[0]+hrad[i,0])*(
                        T_airgun - T[i,0]);
                    T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt
                        /(rho*c_p*b*dz)))+T[i,0]; # bottom node
                    else:
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_airgun))*(pow(K+T[i,n-m],2)+pow(K+
                        T_airgun,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-1]-T[i,n-
                        m]) + ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m]);
                    qconv[i,n-m] = 2*dz*(h+hrad[i,n-m])*(
                        T_airgun - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n-m])
                        *((dt/(rho*c_p*b*dz))))+T[i,n-m]; # mid
                        node

                    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_airgun))
                        *(pow(K+T[i,0],2)+pow(K+T_airgun,2));
                    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*
                        b)/dz)*(T[i,1]-T[i,0]);
                    qconv[i,0] = dz*(h+hrad[i,0])*(T_airgun - T[
                        i,0]);
                    T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt
                        /(rho*c_p*b*dz)))+T[i,0]; # bottom node



        else:
## 1 layer condition
            if n == 0 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
                    ,0],2)+pow(K+T_inf,2));
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]);
                qconv[i,0] = 2*dz*(h+hrad[i,0])*(T_inf - T[i,0])+b*(
                    h+hrad[i,0])*(T_inf-T[i,0]);
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,0]; # bottom node

 ## 2 layers condition
            elif n == 1 :
                hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
                    ,0],2)+pow(K+T_inf,2));
                hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
                    ,1],2)+pow(K+T_inf,2));

                qcond[i,1] = ((k*b)/dz)*(T[i,0]-T[i,1]);
                qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
                    T[i,1]-T[i,0]);


                qconv[i,1] = 2*dz*(h+hrad[i,1])*(T_inf - T[i,1])+b*(
                    h+hrad[i,1])*(T_inf-T[i,1]);
                qconv[i,0] = 2*dz*(h+hrad[i,0])*(T_inf - T[i,0]);

                T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,1]; #Top node
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,0]; # bottom node

## 3 or more layers condition
            elif n > 1 :
                hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K+T[i
                    ,n],2)+pow(K+T_inf,2));
                qcond[i,n] = ((k*b)/dz)*(T[i,n-1]-T[i,n]);
                qconv[i,n] = dz*(h+hrad[i,n])*(T_inf - T[i,n])+b*(h+
                    hrad[i,n])*(T_inf-T[i,n]);

                T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(rho*
                    c_p*b*dz)))+T[i,n]; #Top node



                for m in range(1,n):
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+T_inf))*(
                        pow(K+T[i,n-m],2)+pow(K+T_inf,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m])
                        + ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m]);
                    qconv[i,n-m] = 2*dz*(h+hrad[i,n-m])*(T_inf - T[i
                        ,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n-m])*((
                        dt/(rho*c_p*b*dz))))+T[i,n-m]; # mid node
                    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K
                        +T[i,0],2)+pow(K+T_inf,2));
                    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/
                        dz)*(T[i,1]-T[i,0]);
                    qconv[i,0] = dz*(h+hrad[i,0])*(T_inf - T[i,0]);
                    T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(
                        rho*c_p*b*dz)))+T[i,0]; # bottom node




        ## Temperature update

        #Tnew[n_t*(n)+n:(n_t)*(n+1)+1+n,:] = T;
        Tnew[n_t*(n):(n_t)*(n+1)+1,:] = T;
        h_s[n_t*(n):(n_t)*(n+1)]= h_store
        for r   in range(0,n+1):
            T[  0,r] = T[n_t,r];

    ## Deposition of new layer
        T[0,n+1] = 180;
        T_cooldown[0,:]= T[n_t,:]
    #Cooldown period

    for it in range(0,Cooldown_period):
        T_last = T_cooldown[it,0:n+1];
        Ts = np.mean(T_last, axis = 0);
        L = (n+1)*dz;
        Gr = (g*Beta*(Ts -T_inf)*pow(L,3))/(pow(Viscosity,2));
        Pr = Viscosity/Diffusivity;
        Ra = Gr*Pr;

        Nu_L1 = 0.825+(0.387*pow(Ra,1/6))/(pow((1+pow(0.492/Pr,9/16))
            ,8/27))
        Nu_L = pow(Nu_L1,2)
        h = (Nu_L*Thermal_cond)/L
        h_cooldown[it] = h
        hradc[it,n] = e*Bolt*((K+T_cooldown[it,n])+(K+T_inf))*(pow(K+
            T_cooldown[it,n],2)+pow(K+T_inf,2));
        qcondc[it,n] = ((k*b)/dz)*(T_cooldown[it,n-1]-T_cooldown[it,n]);
        qconvc[it,n] = 2*dz*(h+hradc[it,n])*(T_inf - T_cooldown[it,n])+b
            *(h+hradc[it,n])*(T_inf-T_cooldown[it,n]);

        T_cooldown[it+1,n] = (( qcondc[it,n] + qconvc[it,n])*(2*dt/(rho*
            c_p*b*dz)))+T_cooldown[it,n]; #Top node


        for m in range(1,n):
```

```
        hradc[it,n-m] = e*Bolt*((K+T_cooldown[it,n-m])+(K+T_inf))*(
            pow(K+T_cooldown[it,n-m],2)+pow(K+T_inf,2));
        qcondc[it,n-m] = ((k*b)/dz)*(T_cooldown[it,n-m-1]-T_cooldown
            [it,n-m]) + ((k*b)/dz)*(T_cooldown[it,n-m+1]-T_cooldown[
            it,n-m]);
        qconvc[it,n-m] = 2*dz*(h+hradc[it,n-m])*(T_inf - T_cooldown[
            it,n-m]);
        T_cooldown[it+1,n-m] = (( qcondc[it,n-m] + qconvc[it,n-m])
            *((dt/(rho*c_p*b*dz))))+T_cooldown[it,n-m]; # mid node

        hradc[it,0] = e*Bolt*((K+T_cooldown[it,0])+(K+T_inf))*(pow(K
            +T_cooldown[it,0],2)+pow(K+T_inf,2));
        qcondc[it,0] = ((2*k*b)/dz)*(Tb-T_cooldown[it,0]) + ((k*b)/
            dz)*(T_cooldown[it,1]-T_cooldown[it,0]);
        qconvc[it,0] = 2*dz*(h+hradc[it,0])*(T_inf - T_cooldown[it
            ,0]);
        T_cooldown[it+1,0] = (( qcondc[it,0] + qconvc[it,0])*(2*dt/(
            rho*c_p*b*dz)))+T_cooldown[it,0]; # bottom node


#for t in range(0,i):
#Tnew2[n_t*(n):(n_t)*(n+1)+1,:] = T_cooldown;
#h_s[n_t*(n):(n_t)*(n+1)]= h_store
#for r  in range(0,n+1):
 #  T[  0,r] = T[n_t,r];

## remove all zeros from temperature-matrix
def zero_to_nan(Tnew):
    """Replace every 0 with 'nan' and return a copy."""
for j in range(0,n+1):
  Tnew[:,j] = [np.nan if x==0 else x for x in Tnew[:,j]]



T_total[0:(i+1)*(n_l),:] = Tnew[0:(i+1)*(n_l),:]
T_total[(i+1)*n_l:(i+1)*n_l+1+Cooldown_period,:] = T_cooldown[:,:]
h_total[0:(i+1)*(n_l)] = h_s[0:(i+1)*(n_l)]
h_total[(i+1)*n_l:(i+1)*n_l+Cooldown_period] = h_cooldown

def zero_to_nan(T_total):
    """Replace every 0 with 'nan' and return a copy."""
for j in range(0,n+3):
  T_total[:,j] = [np.nan if x==0 else x for x in T_total[:,j]]


fig1 = plt.figure(dpi=200)
plt.title("Temperature plot without cooldown period")
plt.xlabel("Time (s)")
plt.ylabel("Temperature (Celcius)")
plt.plot(iterations, Tnew[:,0:n+1])
plt.show()


fig2 = plt.figure(dpi=200)
plt.title("Temperature plot with cooldown period ")
plt.xlabel("Time (s)")
plt.ylabel("Temperature (Celcius)")
plt.plot( iterationstotal,T_total[:,0:n+1])
plt.show()

fig3 = plt.figure(dpi=200)
plt.title("Convection coefficient")
plt.xlabel("Time (s)")
plt.ylabel("W/ m^2.K")
plt.plot(iterationstotal[0:n_t*n_l+Cooldown_period], h_total[0:n_t*
    n_l+Cooldown_period],)
plt.show()


plt.figure(dpi=1200)
plt.title("Forced_convection coefficient")
plt.xlabel("Layer number")
plt.ylabel("W/ m^2.K")
plt.plot(h_side)
plt.legend()
plt.show()
```

8.3 Python code 2D-model

Listing 8.2: 2-dimensional numerical thermal model of BAAM process

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr  6 11:31:12 2021

@author: twan
"""
# Inputs
import numpy as np
from numpy.linalg import inv
from matplotlib import pyplot as plt
import labellines
import pandas as pd
from matplotlib.backends.backend_pdf import PdfPages
from datetime import datetime
#from pytexit import py2tex
plt.style.use('seaborn-whitegrid')

#Input parameters
n_l = 13;
```

```
t_p = 30;          # seconds
dt = 1;            # seconds
v = 1500/60;       # mm/s
dz = 0.0025;       # meters
dx = 0.006;        # meters
b = 0.00695;       # meters
Tdep = 180;        #Celcius
T_inf = 25;        #Celcius
Tb = 50;           #Celcius
T_airgun = 40;     #Celcius
Cooldown_period = 200    #Seconds



#Material properties
k = 0.3;
rho = 1150;
c_p = 2200;
e = 0.94;

# Other constants
K = 273;
d = 0.0025;
Bolt = 5.6700e-08;

n_t = round(t_p/dt);

shapeT = (1+n_t,n_l+2)
shapeTnew = (n_t*(n_l+1),n_l+2)
T = np.zeros(shapeT);
Tnew = np.zeros(shapeTnew);
h_s = np.zeros(n_t*(n_l+1));
T[0,0] = Tdep;

h_store = np.zeros(n_t)
hradm = e*Bolt*(Tdep+T_inf)*(Tdep^2+T_inf^2);
qcond= np.zeros((n_t,n_l+1));
qconv= np.zeros((n_t,n_l+1));
hrad = np.zeros((n_t,n_l));
iterations = 0
iterationstotal = 0
iterations =  np.arange(n_t*(n_l+1))*dt;

#Airgun
r = 22.5;
dist = (r/v);
dist_hole = (3/v)
y_hole = int(np.ceil(dist_hole/dt))
y = int(np.ceil(dist/dt))
g = 9.81; #m/s^2
```

```
# Define parameters for air
Beta = 3.44e-03 #1/T_inf;
Viscosity = 1.493e-05; #m^2/s
Diffusivity = 2.074e-05; #m^2/s
Thermal_cond = 0.02514; # W/m*K
Pr = Viscosity/Diffusivity;
i_top_begin = np.arange(int(y/2),y_hole)
i_top_last =  np.arange(n_t-int(y/2),n_t)
top_air = np.append(i_top_begin,i_top_last)



Vel = pd.read_csv ('/Users/twan/Documents/Velocity/Velocity over
   distance/V_distance_log.csv', sep=';')
Pitot_tube = Vel.mean(axis = 0)
Displacement = np.arange(4,23)*25


coeff = np.polyfit(Displacement, Pitot_tube,2)
m_pitot = coeff[0]
b_pitot = coeff[1]
Vtrend = np.linspace(Displacement[0],Displacement[-1],100)
Ttrend = np.polyval(coeff, Vtrend)

Displacement_total = np.arange(0,23)*25
f = coeff[0]*pow(Displacement_total,2) + coeff[1]*Displacement_total
    + coeff[2]

#Cooldown period

T_cooldown = np.zeros((1+Cooldown_period,n_l+2))
hradc = np.zeros((Cooldown_period,n_l));
qcondc= np.zeros((Cooldown_period,n_l+1));
qconvc= np.zeros((Cooldown_period,n_l+1));
shapeT_total = (n_t*(n_l+1)+Cooldown_period,n_l+2)
T_total = np.zeros(shapeT_total);
shapeh_total = (n_t*(n_l+1)+Cooldown_period)
h_total = np.zeros(shapeh_total);
iterationstotal =  np.arange(n_t*(n_l+1)+Cooldown_period)*dt;

#Airgun inputs

Velocity_air = np.zeros((n_l))
Re = np.zeros((n_l))
Nusselt_airgun = np.zeros((n_l))

h_store_side = np.zeros(n_l)
h_store_top = np.zeros(n_t)
h_side = np.zeros(n_l)
h_cooldown = np.zeros(Cooldown_period)
```

```
# Number of layers
for n in range(0,n_l):
    #qdep = (b*d*rho*c_p*Tdep)/(t_p);
    if n % 2 == 0 :
        if n == 0:
            L = (n+1)*dz;
        else:
            L = (n)*dz;
    if n > 9:
        if n % 2 == 0 :

            #forced convection
            Velocity_air[n] =  coeff[2] #coeff[0]*pow(dz*(n-o),2) +
                coeff[1]*dz*(n-o) + coeff[2]

            Re[n] = (Velocity_air[n]*dz)/Viscosity;

            Nusselt_airgun[n] = pow(0.332*Re[n], 1/2)*pow(Pr, 1/3);
            #Nusselt_airgun = (pow(0.037*Re, 4/5)-871)*pow(Pr, 1/3);

            #Nusselt_airgun[n] = pow(0.102*Re[n], 0.675)*pow(Pr,
                1/3);
            h_side[n] = (Nusselt_airgun[n]*Thermal_cond)/dz;
            for o in range(2,n+1,2):
                L_b = (o)*dz;
                #forced convection
                Velocity_air[n-o] =  coeff[0]*pow(dz*(o),2) + coeff
                    [1]*dz*(o) + coeff[2]

                Re[n-o] = (Velocity_air[n-o]*L_b)/Viscosity;

                Nusselt_airgun[n-o] = pow(0.332*Re[n-o], 1/2)*pow(Pr
                    , 1/3);
                #Nusselt_airgun = (pow(0.037*Re, 4/5)-871)*pow(Pr,
                    1/3);

                #Nusselt_airgun[n] = pow(0.102*Re[n], 0.675)*pow(Pr,
                     1/3);
                h_side[n-o] = (Nusselt_airgun[n-o]*Thermal_cond)/L_b
                    ;
            for o in range(1,n+1,2):
                h_side[n-o] = h_side[n-o-1]

    for i in range(0,n_t):
        #Natural convection
        T_last = T[i,0:n+1];
        Ts = np.mean(T_last, axis = 0);
```

```
Gr = (g*Beta*(Ts -T_inf)*pow(L,3))/(pow(Viscosity,2));
Ra = Gr*Pr;

Nu_L1 = 0.825+(0.387*pow(Ra,1/6))/(pow((1+pow(0.492/Pr,9/16)
    ),8/27))
Nu_L = pow(Nu_L1,2)
h = (Nu_L*Thermal_cond)/L
h_store[i] = h

if i in top_air and n > 4:
     Velocity_air_top = coeff[0]*pow(dz,2) + coeff[1]*dz +
         coeff[2]
     Re_top = (Velocity_air_top*dz)/Viscosity;
     Nusselt_airgun_top = pow(0.102*Re_top, 0.675)*pow(Pr,
         1/3);
     #if n > 5:
     h_top = (Nusselt_airgun_top*Thermal_cond)/dz;
     h_store_top[i] = h_top
     #else:
          #h_top = h

else :
     h_top = h

if i < y and n > 9:
     ## 3 or more layers condition
     if n > 3 :

          if n % 2 == 0 :
               hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
                   +T[i,n],2)+pow(K+T_inf,2));
               qcond[i,n] = ((k*b)/dz)*(T[i,n-2]-T[i,n]);
               qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_inf - T
                   [i,n])+b*(h_side[n]+hrad[i,n])*(T_inf-T[i,n])
                   ;

               T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
                   rho*c_p*b*dz)))+T[i,n]; #Top node


               hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
                   pow(K+T[i,n-1],2)+pow(K+T_inf,2));
               qcond[i,n-1] = ((k*b)/dx)*(T[i,n-2]-T[i,n-1])+
                   ((k*b)/dz)*(T[i,n-3]-T[i,n-1]);
               qconv[i,n-1] = dz*(h_side[n]+hrad[i,n-1])*(T_inf
                    - T[i,n-1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n
                   -1]);

               T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
                   dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node
```

```
        for m in range(2,n):
            if m% 2 != 0 :

                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                    n-m])+ ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m
                    ]);
                qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
                    m])*(T_inf - T[i,n-m]);
                T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                    -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                     # mid node

            if m % 2 == 0:
                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                    n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
                    ]);
                qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
                    m])*(T_inf - T[i,n-m]);
                T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                    -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                     # mid node

    if n % 2 != 0 :
        hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
            +T[i,n],2)+pow(K+T_inf,2));
        qcond[i,n] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+ ((k*b
            )/dz)*(T[i,n-2]-T[i,n]);
        qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_inf - T
            [i,n])+b*(h_side[n]+hrad[i,n])*(T_inf-T[i,n])
            ;

        T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
            rho*c_p*b*dz)))+T[i,n]; #Top node

        hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
            pow(K+T[i,n-1],2)+pow(K+T_inf,2));
        qcond[i,n-1] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+((k*
            b)/dz)*(T[i,n-3]-T[i,n-1]);
        qconv[i,n-1] = dz*(h_side[n]+hrad[i,n-1])*(T_inf
             - T[i,n-1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n
            -1]);
            T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
                dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node
            for m in range(2,n):
                if m % 2 == 0 :
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                        T_inf,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                        i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                        n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
                        ]);
                    qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
                        m])*(T_inf - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                        -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                         # mid node

                if m % 2 != 0 :
                    hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                        T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                        T_inf,2));
                    qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                        i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                        n-m])+ ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m
                        ]);
                    qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
                        m])*(T_inf - T[i,n-m]);
                    T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                        -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                         # mid node


        hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
            ,1],2)+pow(K+T_inf,2));
        qcond[i,1] = ((2*k*b)/dz)*(Tb-T[i,1]) + ((k*b)/dz)*(
            T[i,0]-T[i,1]) + ((k*b)/dz)*(T[i,3]-T[i,1]);
        qconv[i,0] = dz*(h_side[1]+hrad[i,1])*(T_inf - T[i
            ,1]);
        T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
            c_p*b*dz)))+T[i,1];

        hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
            ,0],2)+pow(K+T_inf,2));
        qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
            T[i,1]-T[i,0]) + ((k*b)/dz)*(T[i,2]-T[i,0]);
        qconv[i,0] = dz*(h_side[0]+hrad[i,0])*(T_inf - T[i
            ,0]);
        T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
            c_p*b*dz)))+T[i,0]; # bottom node
elif n_t-i < y and n > 9:
```

```
## 3 or more layers condition
if n > 3 :

    if n % 2 == 0 :
        hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
            +T[i,n],2)+pow(K+T_inf,2));
        qcond[i,n] = ((k*b)/dz)*(T[i,n-2]-T[i,n]);
        qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_inf - T
            [i,n])+b*(h_side[n]+hrad[i,n])*(T_inf-T[i,n])
            ;

        T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
            rho*c_p*b*dz)))+T[i,n]; #Top node

        hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
            pow(K+T[i,n-1],2)+pow(K+T_inf,2));
        qcond[i,n-1] = ((k*b)/dx)*(T[i,n-2]-T[i,n-1])+
            ((k*b)/dz)*(T[i,n-3]-T[i,n-1]);
        qconv[i,n-1] = dz*(h_side[n]+hrad[i,n-1])*(T_inf
             - T[i,n-1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n
            -1]);

        T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
            dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node

        for m in range(2,n):



            if m% 2 != 0 :

                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                    n-m])+ ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m
                    ]);
                qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
                    m])*(T_inf - T[i,n-m]);
                T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                    -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                     # mid node

            if m % 2 == 0:
                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
                ]);
            qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
               m])*(T_inf - T[i,n-m]);
            T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
               -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                # mid node

if n % 2 != 0 :
    hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
       +T[i,n],2)+pow(K+T_inf,2));
    qcond[i,n] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+ ((k*b
       )/dz)*(T[i,n-2]-T[i,n]);
    qconv[i,n] = dz*(h_side[n]+hrad[i,n])*(T_inf - T
       [i,n])+b*(h_side[n]+hrad[i,n])*(T_inf-T[i,n])
       ;

    T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
       rho*c_p*b*dz)))+T[i,n]; #Top node

    hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
       pow(K+T[i,n-1],2)+pow(K+T_inf,2));
    qcond[i,n-1] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+((k*
       b)/dz)*(T[i,n-3]-T[i,n-1]);
    qconv[i,n-1] = dz*(h_side[n]+hrad[i,n-1])*(T_inf
        - T[i,n-1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n
       -1]);

    T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
       dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node

    for m in range(2,n):
        if m % 2 == 0 :
            hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
               T_inf))*(pow(K+T[i,n-m],2)+pow(K+
               T_inf,2));
            qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
               i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
               n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
               ]);
            qconv[i,n-m] = dz*(h_side[n-m]+hrad[i,n-
               m])*(T_inf - T[i,n-m]);
            T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
               -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                # mid node

        if m % 2 != 0 :
            hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
               T_inf))*(pow(K+T[i,n-m],2)+pow(K+
               T_inf,2));
```

```
                qcond[i,n−m] = ((k∗b)/dz)∗(T[i,n−m−2]−T[
                    i,n−m]) + ((k∗b)/dz)∗(T[i,n−m+2]−T[i,
                    n−m])+ ((k∗b)/dz)∗(T[i,n−m−1]−T[i,n−m
                    ]);
                qconv[i,n−m] = dz∗(h_side[n−m]+hrad[i,n−
                    m])∗(T_inf − T[i,n−m]);
                T[i+1,n−m] = (( qcond[i,n−m] + qconv[i,n
                    −m])∗((dt/(rho∗c_p∗b∗dz))))+T[i,n−m];
                     # mid node


        hrad[i,1] = e∗Bolt∗((K+T[i,1])+(K+T_inf))∗(pow(K+T[i
            ,1],2)+pow(K+T_inf,2));
        qcond[i,1] = ((2∗k∗b)/dz)∗(Tb−T[i,1]) + ((k∗b)/dz)∗(
            T[i,0]−T[i,1]) + ((k∗b)/dz)∗(T[i,3]−T[i,1]);
        qconv[i,0] = dz∗(h_side[1]+hrad[i,1])∗(T_inf − T[i
            ,1]);
        T[i+1,1] = (( qcond[i,1] + qconv[i,1])∗(2∗dt/(rho∗
            c_p∗b∗dz)))+T[i,1];

        hrad[i,0] = e∗Bolt∗((K+T[i,0])+(K+T_inf))∗(pow(K+T[i
            ,0],2)+pow(K+T_inf,2));
        qcond[i,0] = ((2∗k∗b)/dz)∗(Tb−T[i,0]) + ((k∗b)/dz)∗(
            T[i,1]−T[i,0]) + ((k∗b)/dz)∗(T[i,2]−T[i,0]);
        qconv[i,0] = dz∗(h_side[0]+hrad[i,0])∗(T_inf − T[i
            ,0]);
        T[i+1,0] = (( qcond[i,0] + qconv[i,0])∗(2∗dt/(rho∗
            c_p∗b∗dz)))+T[i,0]; # bottom node




    else:
## 1 layer condition
        if n == 0 :
            hrad[i,0] = e∗Bolt∗((K+T[i,0])+(K+T_inf))∗(pow(K+T[i
                ,0],2)+pow(K+T_inf,2));
            qcond[i,0] = ((2∗k∗b)/dz)∗(Tb−T[i,0]);
            qconv[i,0] = dz∗(h+hrad[i,0])∗(T_inf − T[i,0])+b∗(h+
                hrad[i,0])∗(T_inf−T[i,0]);
            T[i+1,0] = (( qcond[i,0] + qconv[i,0])∗(2∗dt/(rho∗
                c_p∗b∗dz)))+T[i,0]; # bottom node



        if n == 1 :
            hrad[i,1] = e∗Bolt∗((K+T[i,1])+(K+T_inf))∗(pow(K+T[i
                ,1],2)+pow(K+T_inf,2));
            qcond[i,1] = ((2∗k∗b)/dz)∗(Tb−T[i,1]) + ((k∗b)/dx)∗(
                T[i,0]−T[i,1]);
            qconv[i,1] = dz∗(h+hrad[i,1])∗(T_inf − T[i,1])+b∗(h+
            hrad[i,1])*(T_inf-T[i,1]);
        T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
            c_p*b*dz)))+T[i,1]; # bottom node

        hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
            ,0],2)+pow(K+T_inf,2));
        qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dx)*(
            T[i,1]-T[i,0]);
        qconv[i,0] = dz*(h+hrad[i,0])*(T_inf - T[i,0])+b*(h+
            hrad[i,0])*(T_inf-T[i,0]);
        T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
            c_p*b*dz)))+T[i,0]; # bottom node

## 2 layers condition
    if n == 2 :

        hrad[i,2] = e*Bolt*((K+T[i,2])+(K+T_inf))*(pow(K+T[i
            ,2],2)+pow(K+T_inf,2));
        hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
            ,1],2)+pow(K+T_inf,2));
        hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
            ,0],2)+pow(K+T_inf,2));


        qcond[i,2] = ((k*b)/dz)*(T[i,0]-T[i,2]);
        qcond[i,1] = ((2*k*b)/dz)*(Tb-T[i,1]) + ((k*b)/dx)*(
            T[i,0]-T[i,1]);
        qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
            T[i,2]-T[i,0]) + ((k*b)/dx)*(T[i,1]-T[i,0]);


        qconv[i,2] = dz*(h+hrad[i,1])*(T_inf - T[i,2])+b*(h+
            hrad[i,2])*(T_inf-T[i,2]);
        qconv[i,1] = dz*(h+hrad[i,1])*(T_inf - T[i,1])+b*(h+
            hrad[i,1])*(T_inf-T[i,1]);
        qconv[i,0] = dz*(h+hrad[i,0])*(T_inf - T[i,0]);

        T[i+1,2] = (( qcond[i,2] + qconv[i,2])*(2*dt/(rho*
            c_p*b*dz)))+T[i,2]; #Top node
        T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
            c_p*b*dz)))+T[i,1]; #Top node
        T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
            c_p*b*dz)))+T[i,0]; # bottom node

    if n == 3 :
        hrad[i,3] = e*Bolt*((K+T[i,3])+(K+T_inf))*(pow(K+T[i
            ,3],2)+pow(K+T_inf,2));
        hrad[i,2] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
            ,1],2)+pow(K+T_inf,2));
        hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
                ,1],2)+pow(K+T_inf,2));
            hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
                ,0],2)+pow(K+T_inf,2));

            #To Be Done
            qcond[i,3] = ((k*b)/dz)*(T[i,1]-T[i,3])+ ((k*b)/dx)
                *(T[i,2]-T[i,3]);
            qcond[i,2] = ((k*b)/dz)*(T[i,0]-T[i,2])+ ((k*b)/dx)
                *(T[i,3]-T[i,2]);
            qcond[i,1] = ((2*k*b)/dz)*(Tb-T[i,1]) + ((k*b)/dx)*(
                T[i,0]-T[i,1]) + ((k*b)/dx)*(T[i,3]-T[i,1]);
            qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
                T[i,2]-T[i,0])+ ((k*b)/dx)*(T[i,1]-T[i,0]);



            qconv[i,3] = dz*(h+hrad[i,3])*(T_inf - T[i,3])+b*(h+
                hrad[i,3])*(T_inf-T[i,3]);
            qconv[i,2] = dz*(h+hrad[i,2])*(T_inf - T[i,2])+b*(h+
                hrad[i,2])*(T_inf-T[i,2]);
            qconv[i,1] = dz*(h+hrad[i,1])*(T_inf - T[i,1]);
            qconv[i,0] = dz*(h+hrad[i,0])*(T_inf - T[i,0]);

            T[i+1,3] = (( qcond[i,3] + qconv[i,3])*(2*dt/(rho*
                c_p*b*dz)))+T[i,3]; #Top node
            T[i+1,2] = (( qcond[i,2] + qconv[i,2])*(2*dt/(rho*
                c_p*b*dz)))+T[i,2]; #Top node
            T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
                c_p*b*dz)))+T[i,1]; #Top node
            T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                c_p*b*dz)))+T[i,0]; # bottom node

## 3 or more layers condition
        if n > 3 :

            if n % 2 == 0 :
                hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
                    +T[i,n],2)+pow(K+T_inf,2));
                qcond[i,n] = ((k*b)/dz)*(T[i,n-2]-T[i,n]);
                qconv[i,n] = dz*(h+hrad[i,n])*(T_inf - T[i,n])+b
                    *(h+hrad[i,n])*(T_inf-T[i,n]);

                T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
                    rho*c_p*b*dz)))+T[i,n]; #Top node

                hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
                    pow(K+T[i,n-1],2)+pow(K+T_inf,2));
                qcond[i,n-1] = ((k*b)/dx)*(T[i,n-2]-T[i,n-1])+
                    ((k*b)/dz)*(T[i,n-3]-T[i,n-1]);
                qconv[i,n-1] = dz*(h+hrad[i,n-1])*(T_inf - T[i,n
        -1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n-1]);

    T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
        dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node

    for m in range(2,n):



        if m% 2 != 0 :

            hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                T_inf,2));
            qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                n-m])+ ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m
                ]);
            qconv[i,n-m] = dz*(h+hrad[i,n-m])*(T_inf
                 - T[i,n-m]);
            T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                 # mid node

        if m % 2 == 0:
            hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                T_inf,2));
            qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
                ]);
            qconv[i,n-m] = dz*(h+hrad[i,n-m])*(T_inf
                 - T[i,n-m]);
            T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                 # mid node

if n % 2 != 0 :
    hrad[i,n] = e*Bolt*((K+T[i,n])+(K+T_inf))*(pow(K
        +T[i,n],2)+pow(K+T_inf,2));
    qcond[i,n] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+ ((k*b
        )/dz)*(T[i,n-2]-T[i,n]);
    qconv[i,n] = dz*(h+hrad[i,n])*(T_inf - T[i,n])+b
        *(h+hrad[i,n])*(T_inf-T[i,n]);

    T[i+1,n] = (( qcond[i,n] + qconv[i,n])*(2*dt/(
        rho*c_p*b*dz)))+T[i,n]; #Top node

    hrad[i,n-1] = e*Bolt*((K+T[i,n-1])+(K+T_inf))*(
            pow(K+T[i,n-1],2)+pow(K+T_inf,2));
        qcond[i,n-1] = ((k*b)/dx)*(T[i,n-1]-T[i,n])+((k*
            b)/dz)*(T[i,n-3]-T[i,n-1]);
        qconv[i,n-1] = dz*(h+hrad[i,n-1])*(T_inf - T[i,n
            -1])+b*(h+hrad[i,n-1])*(T_inf-T[i,n-1]);

        T[i+1,n-1] = (( qcond[i,n-1] + qconv[i,n-1])*(2*
            dt/(rho*c_p*b*dz)))+T[i,n-1]; #Top node
        for m in range(2,n):
            if m % 2 == 0 :
                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                    n-m])+ ((k*b)/dz)*(T[i,n-m+1]-T[i,n-m
                    ]);
                qconv[i,n-m] = dz*(h+hrad[i,n-m])*(T_inf
                     - T[i,n-m]);
                T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                    -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                     # mid node

            if m % 2 != 0 :
                hrad[i,n-m] = e*Bolt*((K+T[i,n-m])+(K+
                    T_inf))*(pow(K+T[i,n-m],2)+pow(K+
                    T_inf,2));
                qcond[i,n-m] = ((k*b)/dz)*(T[i,n-m-2]-T[
                    i,n-m]) + ((k*b)/dz)*(T[i,n-m+2]-T[i,
                    n-m])+ ((k*b)/dz)*(T[i,n-m-1]-T[i,n-m
                    ]);
                qconv[i,n-m] = dz*(h+hrad[i,n-m])*(T_inf
                     - T[i,n-m]);
                T[i+1,n-m] = (( qcond[i,n-m] + qconv[i,n
                    -m])*((dt/(rho*c_p*b*dz))))+T[i,n-m];
                     # mid node


    hrad[i,1] = e*Bolt*((K+T[i,1])+(K+T_inf))*(pow(K+T[i
        ,1],2)+pow(K+T_inf,2));
    qcond[i,1] = ((2*k*b)/dz)*(Tb-T[i,1]) + ((k*b)/dz)*(
        T[i,0]-T[i,1]) + ((k*b)/dz)*(T[i,3]-T[i,1]);
    qconv[i,0] = dz*(h+hrad[i,1])*(T_inf - T[i,1]);
    T[i+1,1] = (( qcond[i,1] + qconv[i,1])*(2*dt/(rho*
        c_p*b*dz)))+T[i,1];

    hrad[i,0] = e*Bolt*((K+T[i,0])+(K+T_inf))*(pow(K+T[i
        ,0],2)+pow(K+T_inf,2));
    qcond[i,0] = ((2*k*b)/dz)*(Tb-T[i,0]) + ((k*b)/dz)*(
        T[i,1]-T[i,0]) + ((k*b)/dz)*(T[i,2]-T[i,0]);
```

```
                qconv[i,0] = dz*(h+hrad[i,0])*(T_inf - T[i,0]);
                T[i+1,0] = (( qcond[i,0] + qconv[i,0])*(2*dt/(rho*
                   c_p*b*dz)))+T[i,0]; # bottom node

        ## Temperature update

    #Tnew[n_t*(n)+n:(n_t)*(n+1)+1+n,:] = T;
    Tnew[n_t*(n):(n_t)*(n+1)+1,:] = T;
    h_s[n_t*(n):(n_t)*(n+1)]= h_store
    for r in range(0,n+1):
        T[0,r] = T[n_t,r];

## Deposition of new layer
    T[0,n+1] = Tdep;
    T_cooldown[0,:]= T[n_t,:]
#Cooldown period

for it in range(0,Cooldown_period):
    T_last = T_cooldown[it,0:n+1];
    Ts = np.mean(T_last, axis = 0);
    L = (n)*dz;
    Gr = (g*Beta*(Ts -T_inf)*pow(L,3))/(pow(Viscosity,2));
    Pr = Viscosity/Diffusivity;
    Ra = Gr*Pr;

    Nu_L1 = 0.825+(0.387*pow(Ra,1/6))/(pow((1+pow(0.492/Pr,9/16))
        ,8/27))
    Nu_L = pow(Nu_L1,2)
    h = (Nu_L*Thermal_cond)/L
    h_cooldown[it] = h
    if n % 2 == 0 :
        hradc[it,n] = e*Bolt*((K+T_cooldown[it,n])+(K+T_inf))*(pow(K
           +T_cooldown[it,n],2)+pow(K+T_inf,2));
        qcondc[it,n] = ((k*b)/dz)*(T_cooldown[it,n-2]-T_cooldown[it,
           n]);
        qconvc[it,n] = dz*(h+hradc[it,n])*(T_inf - T_cooldown[it,n])
           +b*(h+hradc[it,n])*(T_inf-T_cooldown[it,n]);

        T_cooldown[it+1,n] = (( qcondc[it,n] + qconvc[it,n])*(2*dt/(
           rho*c_p*b*dz)))+T_cooldown[it,n]; #Top node

        #top layer below
        hradc[it,n-1] = e*Bolt*((K+T_cooldown[it,n-1])+(K+T_inf))*(
           pow(K+T_cooldown[it,n-1],2)+pow(K+T_inf,2));
        qcondc[it,n-1] = ((k*b)/dx)*(T_cooldown[it,n-2]-T_cooldown[
           it,n-1])+ ((k*b)/dz)*(T_cooldown[it,n-3]-T_cooldown[it,n
           -1]);
        qconvc[it,n-1] = dz*(h+hradc[it,n-1])*(T_inf - T_cooldown[it
           ,n-1])+b*(h+hradc[it,n-1])*(T_inf-T_cooldown[it,n-1]);
```

```
T_cooldown[it+1,n-1] = (( qcondc[it,n-1] + qconvc[it,n-1])
    *(2*dt/(rho*c_p*b*dz)))+T_cooldown[it,n-1]; #Top node

for m in range(2,n):
    if m% 2 != 0 :
        hradc[it,n-m] = e*Bolt*((K+T_cooldown[it,n-m])+(K+
            T_inf))*(pow(K+T_cooldown[it,n-m],2)+pow(K+T_inf
            ,2));
        qcondc[it,n-m] = ((k*b)/dz)*(T_cooldown[it,n-m-2]-
            T_cooldown[it,n-m]) + ((k*b)/dz)*(T_cooldown[it,n
            -m+2]-T_cooldown[it,n-m])+ ((k*b)/dz)*(T_cooldown
            [it,n-m-1]-T_cooldown[it,n-m]);
        qconvc[it,n-m] = dz*(h+hradc[it,n-m])*(T_inf -
            T_cooldown[it,n-m]);
        T_cooldown[it+1,n-m] = (( qcondc[it,n-m] + qconvc[it
            ,n-m])*((dt/(rho*c_p*b*dz))))+T_cooldown[it,n-m];
             # mid node

    if m % 2 == 0:
        hradc[it,n-m] = e*Bolt*((K+T_cooldown[it,n-m])+(K+
            T_inf))*(pow(K+T_cooldown[it,n-m],2)+pow(K+T_inf
            ,2));
        qcondc[it,n-m] = ((k*b)/dz)*(T_cooldown[it,n-m-2]-
            T_cooldown[it,n-m]) + ((k*b)/dz)*(T_cooldown[it,n
            -m+2]-T_cooldown[it,n-m])+ ((k*b)/dz)*(T_cooldown
            [it,n-m+1]-T_cooldown[it,n-m]);
        qconvc[it,n-m] = dz*(h+hradc[it,n-m])*(T_inf -
            T_cooldown[it,n-m]);
        T_cooldown[it+1,n-m] = (( qcondc[it,n-m] + qconvc[it
            ,n-m])*((dt/(rho*c_p*b*dz))))+T_cooldown[it,n-m];
             # mid node

hradc[it,1] = e*Bolt*((K+T_cooldown[it,1])+(K+T_inf))*(pow(K
    +T_cooldown[it,1],2)+pow(K+T_inf,2));
qcondc[it,1] = ((2*k*b)/dz)*(Tb-T_cooldown[it,1]) + ((k*b)/
    dz)*(T_cooldown[it,0]-T_cooldown[it,1])+ ((k*b)/dz)*(
    T_cooldown[it,3]-T_cooldown[it,1]);
qconvc[it,1] = dz*(h+hradc[it,1])*(T_inf - T_cooldown[it,1])
    ;
T_cooldown[it+1,1] = (( qcondc[it,1] + qconvc[it,0])*(2*dt/(
    rho*c_p*b*dz)))+T_cooldown[it,1]; # bottom node

hradc[it,0] = e*Bolt*((K+T_cooldown[it,0])+(K+T_inf))*(pow(K
    +T_cooldown[it,0],2)+pow(K+T_inf,2));
qcondc[it,0] = ((2*k*b)/dz)*(Tb-T_cooldown[it,0]) + ((k*b)/
    dz)*(T_cooldown[it,1]-T_cooldown[it,0])+ ((k*b)/dz)*(
    T_cooldown[it,2]-T_cooldown[it,0]);
qconvc[it,0] = 2*dz*(h+hradc[it,0])*(T_inf - T_cooldown[it
    ,0]);
T_cooldown[it+1,0] = (( qcondc[it,0] + qconvc[it,0])*(2*dt/(
        rho*c_p*b*dz)))+T_cooldown[it,0]; # bottom node

if n % 2 != 0 :
    hradc[it,n] = e*Bolt*((K+T_cooldown[it,n])+(K+T_inf))*(pow(K
        +T_cooldown[it,n],2)+pow(K+T_inf,2));
    qcondc[it,n] = ((k*b)/dx)*(T_cooldown[it,n-1]-T_cooldown[it,
        n])+ ((k*b)/dz)*(T_cooldown[it,n-2]-T_cooldown[it,n]);
    qconvc[it,n] = dz*(h+hradc[it,n])*(T_inf - T_cooldown[it,n])
        +b*(h+hradc[it,n])*(T_inf-T_cooldown[it,n]);

    T_cooldown[it+1,n] = (( qcondc[it,n] + qconvc[it,n])*(2*dt/(
        rho*c_p*b*dz)))+T_cooldown[it,n]; #Top node

    #top-side layer
    hradc[it,n-1] = e*Bolt*((K+T_cooldown[it,n-1])+(K+T_inf))*(
        pow(K+T_cooldown[it,n-1],2)+pow(K+T_inf,2));
    qcondc[it,n-1] = ((k*b)/dx)*(T_cooldown[it,n-1]-T_cooldown[
        it,n])+((k*b)/dz)*(T_cooldown[it,n-3]-T_cooldown[it,n-1])
        ;
    qconvc[it,n-1] = dz*(h+hradc[it,n-1])*(T_inf - T_cooldown[it
        ,n-1])+b*(h+hradc[it,n-1])*(T_inf-T_cooldown[it,n-1]);

    T_cooldown[it+1,n-1] = (( qcondc[it,n-1] + qconvc[it,n-1])
        *(2*dt/(rho*c_p*b*dz)))+T_cooldown[it,n-1]; #Top node
    for m in range(2,n):
        if m % 2 == 0 :
            hradc[it,n-m] = e*Bolt*((K+T_cooldown[it,n-m])+(K+
                T_inf))*(pow(K+T_cooldown[it,n-m],2)+pow(K+T_inf
                ,2));
            qcondc[it,n-m] = ((k*b)/dz)*(T_cooldown[it,n-m-2]-
                T_cooldown[it,n-m]) + ((k*b)/dz)*(T_cooldown[it,n
                -m+2]-T_cooldown[it,n-m])+ ((k*b)/dz)*(T_cooldown
                [it,n-m+1]-T_cooldown[it,n-m]);
            qconvc[it,n-m] = dz*(h+hradc[it,n-m])*(T_inf -
                T_cooldown[it,n-m]);
            T_cooldown[it+1,n-m] = (( qcondc[it,n-m] + qconvc[it
                ,n-m])*((dt/(rho*c_p*b*dz))))+T_cooldown[it,n-m];
                 # mid node

        if m % 2 != 0 :
            hradc[it,n-m] = e*Bolt*((K+T_cooldown[it,n-m])+(K+
                T_inf))*(pow(K+T_cooldown[it,n-m],2)+pow(K+T_inf
                ,2));
            qcondc[it,n-m] = ((k*b)/dz)*(T_cooldown[it,n-m-2]-
                T_cooldown[it,n-m]) + ((k*b)/dz)*(T_cooldown[it,n
                -m+2]-T_cooldown[it,n-m])+ ((k*b)/dz)*(T_cooldown
                [it,n-m-1]-T_cooldown[it,n-m]);
            qconvc[it,n-m] = dz*(h+hradc[it,n-m])*(T_inf -
                T_cooldown[it,n-m]);
            T_cooldown[it+1,n-m] = (( qcondc[it,n-m] + qconvc[it
                    ,n-m])*((dt/(rho*c_p*b*dz))))+T_cooldown[it,n-m];
                     # mid node

            hradc[it,1] = e*Bolt*((K+T_cooldown[it,1])+(K+T_inf))*(pow(K
               +T_cooldown[it,1],2)+pow(K+T_inf,2));
            qcondc[it,1] = ((2*k*b)/dz)*(Tb-T_cooldown[it,1]) + ((k*b)/
               dz)*(T_cooldown[it,0]-T_cooldown[it,1])+ ((k*b)/dz)*(
               T_cooldown[it,3]-T_cooldown[it,1]);
            qconvc[it,1] = dz*(h+hradc[it,1])*(T_inf - T_cooldown[it,1])
               ;
            T_cooldown[it+1,1] = (( qcondc[it,1] + qconvc[it,0])*(2*dt/(
               rho*c_p*b*dz)))+T_cooldown[it,1]; # bottom node

            hradc[it,0] = e*Bolt*((K+T_cooldown[it,0])+(K+T_inf))*(pow(K
               +T_cooldown[it,0],2)+pow(K+T_inf,2));
            qcondc[it,0] = ((2*k*b)/dz)*(Tb-T_cooldown[it,0]) + ((k*b)/
               dz)*(T_cooldown[it,1]-T_cooldown[it,0])+ ((k*b)/dz)*(
               T_cooldown[it,2]-T_cooldown[it,0]);
            qconvc[it,0] = 2*dz*(h+hradc[it,0])*(T_inf - T_cooldown[it
               ,0]);
            T_cooldown[it+1,0] = (( qcondc[it,0] + qconvc[it,0])*(2*dt/(
               rho*c_p*b*dz)))+T_cooldown[it,0]; # bottom node
## remove all zeros from temperature-matrix
def zero_to_nan(Tnew):
    """Replace every 0 with 'nan' and return a copy."""
for j in range(0,n+1):
   Tnew[:,j] = [np.nan if x==0 else x for x in Tnew[:,j]]
T_total[0:(i+1)*(n_l),:] = Tnew[0:(i+1)*(n_l),:]
T_total[(i+1)*n_l:(i+1)*n_l+1+Cooldown_period,:] = T_cooldown[:,:]
h_total[0:(i+1)*(n_l)] = h_s[0:(i+1)*(n_l)]
h_total[(i+1)*n_l:(i+1)*n_l+Cooldown_period] = h_cooldown
for j in range(0,n+3):
   T_total[:,j] = [np.nan if x==0 else x for x in T_total[:,j]]
## Plot temperature matrix
fig1 = plt.figure(dpi=1200)
plt.title("Temperature plot without cooldown period")
plt.xlabel("Time (s)")
plt.ylabel("Temperature (Celcius)")
plt.plot(iterations, Tnew[:,0:n+1])
plt.show()


fig2 = plt.figure(dpi=1200)
plt.title("Temperature plot with cooldown period ")
plt.xlabel("Time (s)")
plt.ylabel("Temperature (Celcius)")
plt.plot( iterationstotal,T_total[:,0:n+1])
plt.show()
```

```
fig3 = plt.figure(dpi=1200)
plt.title("Convection coefficient")
plt.xlabel("Time (s)")
plt.ylabel("W/ m^2.K")
plt.plot(iterationstotal[0:n_t*n_l+Cooldown_period], h_total[0:n_t*
   n_l+Cooldown_period],)
plt.show()

fig4 = plt.figure(dpi=1200)
plt.title("Forced_convection coefficient")
plt.xlabel("Layer number")
plt.ylabel("W/ m^2.K")
plt.plot(h_side)
plt.legend()
plt.show()
```

8.4 Arduino code Pitot-Static tube

Listing 8.3: Arduino code Pitot-Static tube

```
//Routine for calculating the velocity from
//a pitot tube and MPXV7002DP pressure differential sensor

float V_0 = 5.0; // arduino voltage
float rho = 1.204; // density of air

// parameters for averaging and offset
float offset = 0;
int offset_size = 2000; //how many times does the offset need to be
   calculated before the script can start
int veloc_mean_size = 600;
int zero_span = 0.75;
int x = 0;

// setup and calculate offset
void setup() {
  Serial.begin(9600);

  for (int ii=0;ii<offset_size;ii++){
    offset += analogRead(A0)-(1023/2);
  }
  offset /= offset_size;
}

void loop() {

  double adc_avg = 0.0; double veloc = 0.000;

// average a few ADC readings for stability
  for (int ii=0;ii<veloc_mean_size;ii++){
    adc_avg+= analogRead(A0)-offset;
  }
  adc_avg/=veloc_mean_size;

  // make sure if the ADC reads below 511, then we equate it to a
     negative velocity
  //511 is halve of the 1022value that gets recieved from the
     arduino
  if (adc_avg>510.9000 and adc_avg<511.10){
    //*** Error bij op tellen dit kan ook in Excel gedaan worden of
       er na.//
  } else{//
    if (adc_avg<511.00000){//
      veloc = -sqrt((-10000.00000*(((adc_avg)/1022.00000)-0.5))/rho)
         ;//
    } else{//
      veloc = sqrt((10000.00000*(((adc_avg)/1022.00000)-0.5))/rho)
         ;//
    }
    if (veloc>0 and veloc<3.2){//
      veloc = veloc+0.43 ;//
    }//
    else if (veloc<0 and veloc>-3.2){
      veloc = veloc - 0.43;//
    }//
    else {
      veloc = veloc +(-0.1171*veloc+0.7844);//
    }//
  }//
  //dit kan weg gehaald worden                                  //
 x=analogRead(A0);

  Serial.println(veloc); // print velocity
 //pi- Serial.println(x); // print velocity
  delay(10); // delay for stability


}
```

Bibliography

[1] Dutchglory, *High-tech industry in the netherlands*, Dec. 2016. [Online]. Available: `https://www.dutchglory.com/high-tech-industry-in-the-netherlands/`.

[2] CBS, *Monitor bedrijvenbeleid*, `https://www.rvo.nl/sites/default/files/2015/10/monitor-bedrijvenbeleid-2015.pdf` Accessed on: 23-02-2021, 2015.

[3] B. G. Compton, B. K. Post, C. E. Duty, L. Love, and V. Kunc, "Thermal analysis of additive manufacturing of large-scale thermoplastic polymer composites," *Additive Manufacturing*, vol. 17, pp. 77–86, 2017.

[4] C. McIlroy and P. Olmsted, "Disentanglement effects on welding behaviour of polymer melts during the fused-filament-fabrication method for additive manufacturing," *Polymer*, vol. 123, pp. 376–391, 2017.

[5] Wavin, *About wavin*, `https://www.wavin.com/nl-nl/over-wavin/over-wavin-nederland` Accessed on: 25.05.2021, 2021.

[6] Nedcam, *About nedcam*, `https://nedcam.com/company-profile/` Accessed on: 25.05.2021, 2021.

[7] HB3D, *Hb3d*, `https://www.hb3d.nl` Accessed on: 25.05.2021, 2021.

[8] S. Engineering, *About stevens engineering*, `https://www.stevens-engineering.eu` Accessed on: 25.05.2021, 2021.

[9] Getech, *About getech*, `https://www.getech.nl` Accessed on: 25.05.2021, 2021.

[10] R. J. Wieringa, *Design science methodology for information systems and software engineering.* Springer, 2014.

[11] H. B. Joan Ernst van Aken, *Problem Solving in Organizations: A Methodological Handbook for Business and Management Students*, 3rd ed. Cambridge University Press, 2018, ISBN: 9781108416269. [Online]. Available: `http://gen.lib.rus.ec/book/index.php?md5=47dfde8832fcd6c5e635b4ca9f73b878`.

[12] J. Thomas and J. Rodriguez, "Modeling the fracture strength between fused-deposition extruded roads 16," in *2000 international solid freeform fabrication symposium*, 2000.

[13] C. Bellehumeur, L. Li, Q. Sun, and P. Gu, "Modeling of bond formation between polymer filaments in the fused deposition modeling process," *Journal of manufacturing processes*, vol. 6, no. 2, pp. 170–178, 2004.

[14] J. Zhang, X. Z. Wang, W. W. Yu, and Y. H. Deng, "Numerical investigation of the influence of process conditions on the temperature variation in fused deposition modeling," *Materials & Design*, vol. 130, pp. 59–68, 2017.

[15] S. Bhandari and R. A. Lopez-Anido, "Discrete-event simulation thermal model for extrusion-based additive manufacturing of pla and abs," *Materials*, vol. 13, no. 21, p. 4985, 2020.

[16] Q. Sun, G. Rizvi, C. Bellehumeur, and P. Gu, "Effect of processing conditions on the bonding quality of fdm polymer filaments," *Rapid prototyping journal*, 2008.

[17] D. Drummer, S. Cifuentes-Cuéllar, and D. Rietzel, "Suitability of pla/tcp for fused deposition modeling," *Rapid Prototyping Journal*, 2012.

[18] S. Hwang, E. I. Reyes, K.-s. Moon, R. C. Rumpf, and N. S. Kim, "Thermo-mechanical characterization of metal/polymer composite filaments and printing parameter study for fused deposition modeling in the 3d printing process," *Journal of Electronic Materials*, vol. 44, no. 3, pp. 771–777, 2015.

[19] W. W. Yu, J. Zhang, J. R. Wu, X. Z. Wang, and Y. H. Deng, "Incorporation of graphitic nano-filler and poly (lactic acid) in fused deposition modeling," *Journal of Applied Polymer Science*, vol. 134, no. 15, 2017.

[20] L. Xinhua, L. Shengpeng, L. Zhou, Z. Xianhua, C. Xiaohu, and W. Zhongbin, “An investigation on distortion of pla thin-plate part in the fdm process,” *The International Journal of Advanced Manufacturing Technology*, vol. 79, no. 5, pp. 1117–1126, 2015.

[21] Y. Zhang and Y. Chou, “Three-dimensional finite element analysis simulations of the fused deposition modelling process,” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 220, no. 10, pp. 1663–1671, 2006.

[22] D. Ganji, Y. Sabzehmeidani, and A. Sedighiamiri, *Nonlinear systems in heat transfer: Chapter 3—radiation heat transfer*, 2018.

[23] W. Blevin and W. Brown, “A precise measurement of the stefan-boltzmann constant,” *Metrologia*, vol. 7, no. 1, p. 15, 1971.

[24] P. G. Kosky, *Exploring engineering : an introduction to engineering and design*. London, United Kingdom: Academic Press is an imprint of Elsevier, 2021, ISBN: 9780128150733.

[25] S. W. Churchill and H. H. Chu, “Correlating equations for laminar and turbulent free convection from a vertical plate,” *International journal of heat and mass transfer*, vol. 18, no. 11, pp. 1323–1329, 1975.

[26] M. Bahrami, “Forced convection heat transfer,” *ENSC 388 (F09)*, pp. 1–11, 2009.

[27] A. Shahidian, M. Ghassemi, J. Mohammadi, and M. Hashemi, *Bio-Engineering Approaches to Cancer Diagnosis and Treatment*. Academic Press, 2020.

[28] M. Ghassemi and A. Shahidian, *Nano and bio heat transfer and fluid flow*. Academic Press, 2017.

[29] Z.-X. Zhang, *Rock fracture and blasting: theory and applications*. Butterworth-Heinemann, 2016.

[30] M. Feidt, *Finite Physical Dimensions Optimal Thermodynamics 1: Fundamentals*. Elsevier, 2017.

[31] J. Meseguer, I. Pérez-Grande, and A. Sanz-Andrés, *Spacecraft thermal control*. Elsevier, 2012.

[32] Y. Zhang and V. Shapiro, “Linear-time thermal simulation of as-manufactured fused deposition modeling components,” *Journal of Manufacturing Science and Engineering*, vol. 140, no. 7, 2018.

[33] G. W. Recktenwald, “Finite-difference approximations to the heat equation,” *Mechanical Engineering*, vol. 10, no. 01, 2004.

[34] R. Gentle, P. Edwards, and B. Bolton, “3 - fluid mechanics,” in *Mechanical Engineering Systems*, ser. IIE Core Textbooks Series, R. Gentle, P. Edwards, and B. Bolton, Eds., Oxford: Butterworth-Heinemann, 2001, pp. 112–168, ISBN: 978-0-7506-5213-1. DOI: `https://doi.org/10.1016/B978-075065213-1/50003-9`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780750652131500039`.

[35] NXP, *Mpxv7002*, `https://www.nxp.com/docs/en/data-sheet/MPXV7002.pdf` Accessed on: 15.06.2021, 2021.

[36] J. Hrisko, *Arduino pitot tube wind speed and airspeed indicator - theory and experiments*, `https://makersportal.com/blog/2019/02/06/arduino-pitot-tube-wind-speed-theory-and-experiment` Accessed on: 15.06.2021, February 8, 2019.

[37] FLIR, *Flir a35*, `https://www.flir.eu/products/a35/` Accessed on: 14.06.2021, 2021.

[38] basell, *Hostacom and Hifax Basell polypropylene compounds used in automotive and electrical appliance applications*. basell, 2021.

[39] C. Maier and T. Calafut, *Polypropylene: the definitive user’s guide and databook*. William Andrew, 1998.