THESIS

PRACTICAL TRANSIENT THERMAL MODELLING FOR HARDNESS PREDICTION IN WIRE AND ARC ADDITIVE MANUFACTURED STEEL BLOCKS

Dennis van Kleinwee
4394410

ABSTRACT

Qualification of the mechanical properties of Wire Arc Additive Manufacturing (WAAM) products is an essential step for a successful introduction of this process in industry. To realise this, the relationship between processing conditions, the thermal history and the resulting mechanical properties needs to be established. Hardness is an easy-to-check property that correlates well with such tensile properties. This thesis describes the prediction of the relationship between thermal cycling during deposition of a WAAM produced multilayer block and the resulting hardness. A 3D transient heat model was built to investigate the effect of the thermal behaviour. The weld process was simulated using temperature boundary conditions instead of a direct heat source to calculate the thermal field. The material deposition was simulated using the element birth-death method. The calculated thermal data was used for hardness estimation based on the t8/5 time, i.e. the time to cool from 800 to 500 °C, to determine the hardness distribution throughout the deposited blocks. Temperature was measured during the experiments with thermocouples to validate the thermal model, while hardness was measured to verify the hardness results. The thermal boundary condition based model was found to be in good agreement with experimental results. The modelled results show a correct prediction of the decrease in hardness with increased layer height.

ACKNOWLEDGEMENTS

I would like to thank my supervisors of this thesis, Bin Hu and Marcel Hermans. Bin Hu for providing the project, as well as his assistance and feedback on my work and progress. Marcel Hermans for his support and insight, as well as his scientific guidance and suggestions. His endless support to finish this thesis was invaluable, and I would like to thank him for his time and energy. I would like to thank Zhaoxiang Chen for his help with robot welding, as well as general assistance in the office. I would also like to thank Allseas Engineering for proving me the opportunity to work on this project.

I would like to thank the TU Delft for its support and facilities. In particular, Jurriaan van Slingerland for his assistance and insight with the use of thermocouples and the measuring of weld current. For the assistance with optical microscopy and hardness measurements, I would like to thank Sander van Asperen. For the tensile experiments, I would like to thank Elise Reinhart for the assistance and use of the tensile test machines and Wim Velt for letting me use the workshop to produce these tensile samples.

Lastly, I would like to thank Jan Niemeijer (Allseas) for the moral support each lunchbreak, while teaching me to be better at Ping Pong. The daily lunchtime matches together with Bin and Zhaoxiang gave leisure and relaxation, as well as necessarily exercise.

1. INTRODUCTION

Additive Manufacturing, sometimes called 3D printing, is a processing technology that involves depositing materials layer-by-layer to manufacture a desired product. This layer-based deposition technique allows new freedoms in design and production, and enables both optimizing existing products and the creation of entirely new ones. Allseas is involved in not only the offshore pipelaying business but also adjacent businesses. This field of operation typically involves large structures for which certain additive manufacturing technologies could be attractive. One interesting type of 3D printing for such a purpose is Wire and Arc Additive Manufacturing (WAAM), which adapts existing welding processes for additive manufacturing. The WAAM process allows higher deposition rates and larger products to be built at the cost of resolution. Where other Additive Manufacturing techniques can deposit layers in the order of tens of microns, WAAM deposits layers in the order of millimetres.

WAAM uses welding techniques as a basis to deposit a sequence of weld beads, building up a product layer-by-layer. An electric arc between an electrode and the base material acts as a heat source for local melting. Feedstock material ("filler material") is added during this process, melting and mixing with the underlying material to form a solidified bead. These beads form a layer, and the entire set of layers form the product. Each bead undergoes a cycle of heating and cooling during deposition, but from subsequent welds also a sequence of reheating and cooling. The properties of the weld metal are dependent on this thermal history. Therefore, understanding the effect of this thermal cycling on the evolution of the microstructure is essential to ultimately be able to predict the performance of the resulting component. For enforcing safety margins, a sufficient level of strength needs to be guaranteed, while fatigue performance demands sufficient ductility. In this study steel is used, as steel is a cheap and common engineering material, but also the correlation between hardness and strength is quite well established. This allows to use hardness testing as a simple test for exploring strength of the 3D printed coupon.

WAAM can be used to make all sorts of structures. Large, complicated components that are challenging to cast or intricate geometries not readily made from stock material are typical situations where WAAM can show its advantages. The WAAMpeller (Figure 1-1) is an example of the former. Propellers are typically cast, which poses challenges with the large geometry. The initial cast has to undergo considerable machining to achieve its final shape. This is a time consuming and expensive operation. A propeller printed by WAAM can be made close to the final shape, incurring lower waste, less machining time and thus lower cost.

Figure 1-1: WAAMpeller, designed and produced by RAMLAB in collaboration with DAMEN [1]. The propeller is made of 298-layers of nickel-aluminium-bronze alloy.

Another example is the MX3D printed bridge. This is a complicated design with many aesthetic features, complicated shapes and yet a need for structural integrity. The waving, organic-looking design with thickness variations makes it hard to produce with traditional methods. WAAM allowed the bridge to be printed, creating a single-piece bridge optimized for strength. Complex hollow structural members and joints are integrated into a single structure. Testing, certification and permits are essential to ensure such a construction is safe. Combining complex structural and aesthetic features requires a good understanding of the building of such a bridge, and thus requires novel tools to iterate and validate such a design.

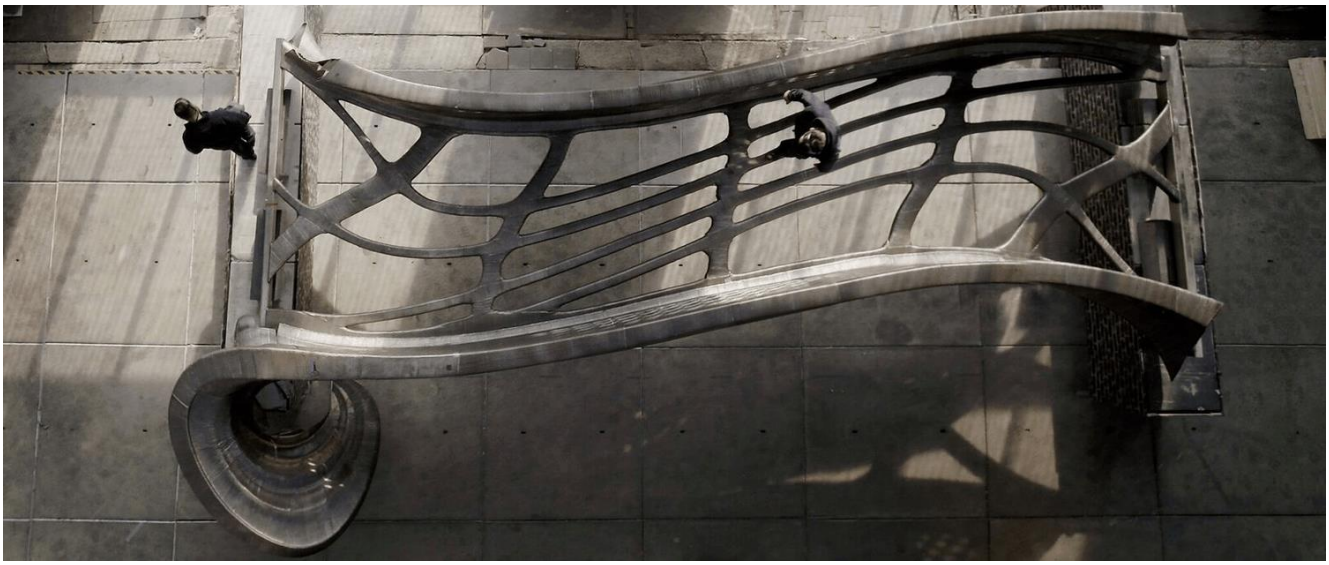


Figure 1-2: The 12m long MX3D bridge, made entirely of stainless steel [2]. A great combination of a particular aesthetic and function, as well as incorporating complicated shapes..

The production of especially large structures can take a long time. This means that ideally, detailed information about the structure should be available as much as possible before starting printing. Evaluation of the design could indicate potential weak spots and issues in the design, avoiding

extensive time on prototyping. Figure 1-3 illustrates the correlation between material, processing and properties. Not just the material, but also the processing of said material is essential to reach the desired material properties. Strength and toughness are two important mechanical properties for structures. Hardness correlates to both properties and is relatively easy to measure. A thermal model combined with a hardness model could help predict these mechanical properties, reducing design errors and ensuring the required product performance. Finite Element Analysis (FEA) can offer insight in the thermal history of the deposited material, and thus offers insight into component building strategies and product performance. However, models need to be sufficiently fast to make it useful for practical purposes. The calculation time should not vastly exceed the time it would take to build and test the material directly. Physics-based models exist but are very time consuming. In this study a model is constructed which approximates the thermal field with prescribed temperature and boundary conditions instead, reducing on the amount of equations needed for solving, while sacrificing some degree of accuracy. In combination with a hardness model mechanical properties of printed parts can be obtained. To achieve this goal the following research questions will be addressed:

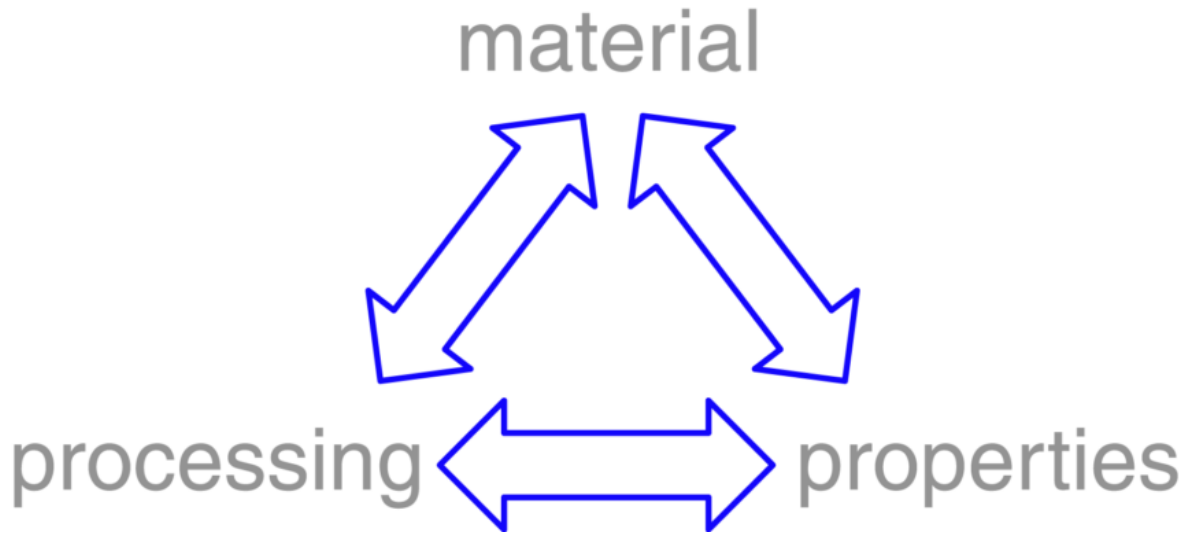


Figure 1-3: The interaction triangle of material, processing and properties

Can thermal-microstructural-mechanical models based on an imposed heat field in combination with an element birth/death material addition approach predict the mechanical properties in wire and arc based additive manufacturing with sufficient accuracy and short computational time.

A number of sub-research questions need to be addressed.
Is the thermal field in a 3D printed block modeled by an imposed heat field sufficiently accurate and what are the most sensitive parameters affecting prediction of the thermal field?

Is the material addition approach describing the shape of the printed part correctly and how should the parameters involved be selected?

Can the hardness of a steel construct be predicted with sufficient accuracy?

Thesis outline

The thesis comprises of 5 chapters and is structured as follows. After this introduction (Chapter 1), in Chapter 2 of this thesis the necessary background information will be provided to understand the concepts involved, such as additive manufacturing, in particular Wire and Arc Additive Manufacturing, welding and weld metallurgy, thermal and hardness modelling aspects.

In Chapter 3 the methodology is introduced, the details of the experiments approach and the constructed models are described.

Chapter 4 provides a detailed overview of the results obtained and also the discussion of these results.

In Chapter 5 the conclusions based on the research are stated, the formulated research questions are addressed and finally recommendations for further research are presented.

2. BACKGROUND

This chapter will present the necessary background information for the topic of this thesis. The topics to be discussed are Additive Manufacturing, WAAM, and the GMAW weld process.

2.1. Additive Manufacturing

Additive manufacturing of metallic components involves, as the name suggests, the creation of products by adding material layer-by-layer to reach a near-final product. A heat source locally melts the substrate and feedstock material, creating a metal structure. As heat source either arc or laser processes can be used, whereas the feedstock is either powder or wire [3] [4] [5].
The steps involved in AM are shown in Figure 2-1. A 3D model is made of the desired product. This model is then sliced into layers and each layer has a tool path generated for the material deposition. This tool path describes how the part should be build [6]. A machine-specific code is then generated from this toolpath and is submitted to the AM hardware. Once the part is completed several post processing stages may be needed based on the requirements of the product. In general, the part eventually needs to be removed from the substrate and cleaned. In addition, post-processing like heat treatments, to adjust mechanical properties, and milling to achieve final tolerances should be considered. Finally, the part is ready for its intended application.

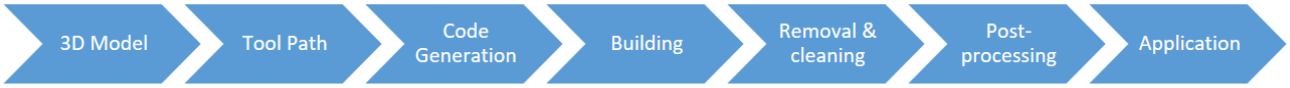


Figure 2-1: The Additive Manufacturing path from 3D model to application.

Additive manufacturing can be subdivided into many forms, based on various criteria such as materials or deposition method. In the following sections the main technologies, powder bed deposition and wire based deposition will be briefly introduced. This is followed by the effects of the thermal cycling on the evolution of the microstructure in steel.

2.1.1. Powder Bed Method

Figure 2-2 shows a schematic of the powder bed method. Powder is taken from a feed system and placed as a layer over a build platform by a roller. A laser beam scans a given tool path and locally melts the material such that the powder binds together. The build platform then lowers and a new layer of powder is added. Layer-by-layer the part is created and upon completion the part is removed from the build platform and cleaned.
The laser focus area is in the order of 70 μm^2 [7] and the powders used are relatively fine, in the range of 10 to 60 µm [8]. The layer thickness that can be realised is approximately 20-100 µm [9].

This technology allows to achieve very fine and detailed structures, depending on the layer thickness. The powder bed acts as a structural support during building, although often support structures are needed, which need to be removed [9]

A limitation is that the powder method does not produce full-density parts and may require treatment for densification. The density achieved as produced is up to approximately 99% [10]. Further densification can be achieved by for instance hot isostatic pressing [11], increasing the density to full density and removing lack-of-fusion porosity. Furthermore, the size of the part is limited by the size of the powder bed chamber. This limitation, in combination with the low deposition rate, make that the production of large structures in the order of meters cannot be realised by the powder bed technique.

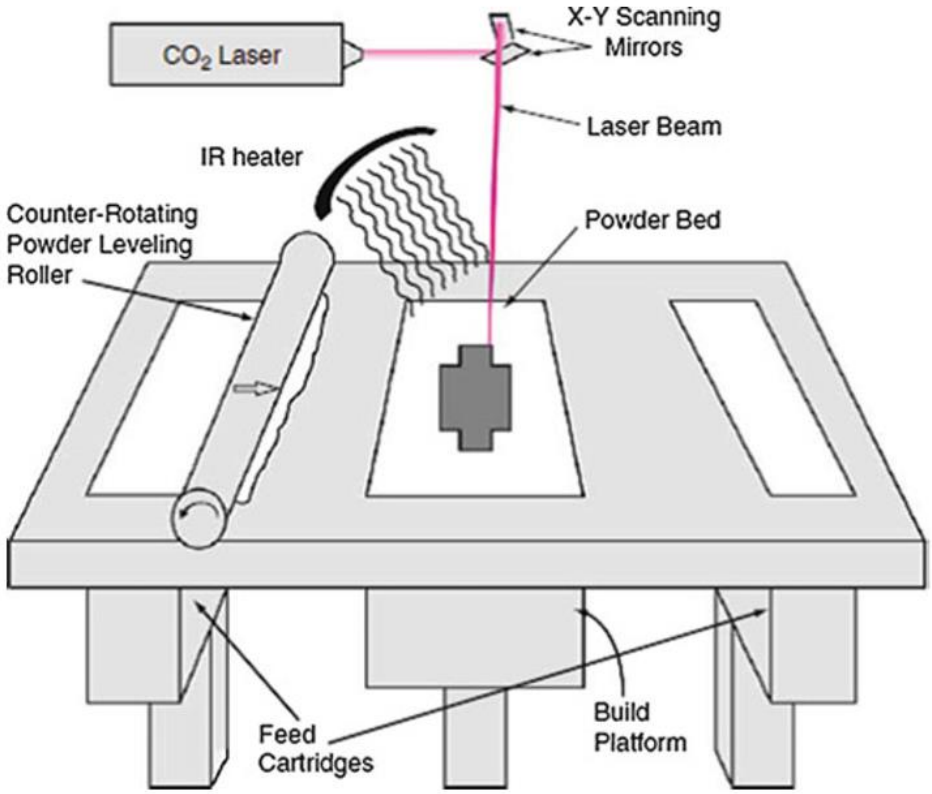


Figure 2-2: Schematic of Selective Laser Melting from [12]

2.1.2. Direct Energy Deposition methods

Direct energy deposition (DED) methods make use of a heating source and supplies the material to be deposited as a powder or wire directly to the product. The feed material and substrate material melt and mix. By repeating this process over a specific path a product is formed. This group of processes can be characterised by a much higher deposition rate compared to the powder bed techniques, see Figure 2-3. However, the details in the construct that can be realised are coarser.

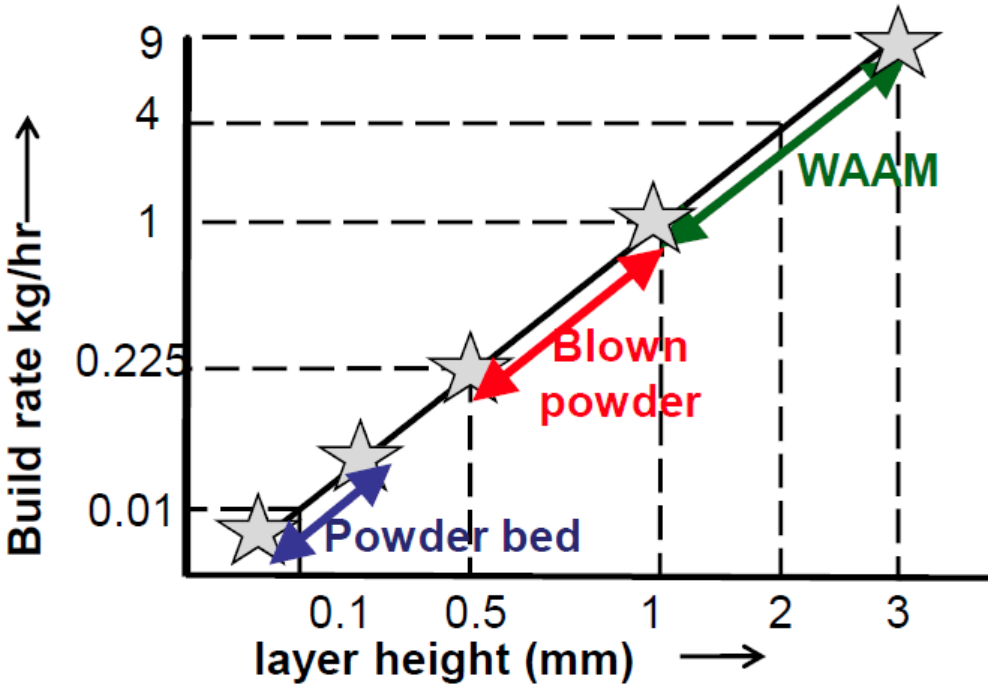


Figure 2-3: A comparison between layer height and build rate. [13]

As the process applied in this study combines an arc heat source with a wire based feedstock, i.e. Wire and Arc Additive Manufacturing (WAAM), this process is described in more detail in this section.

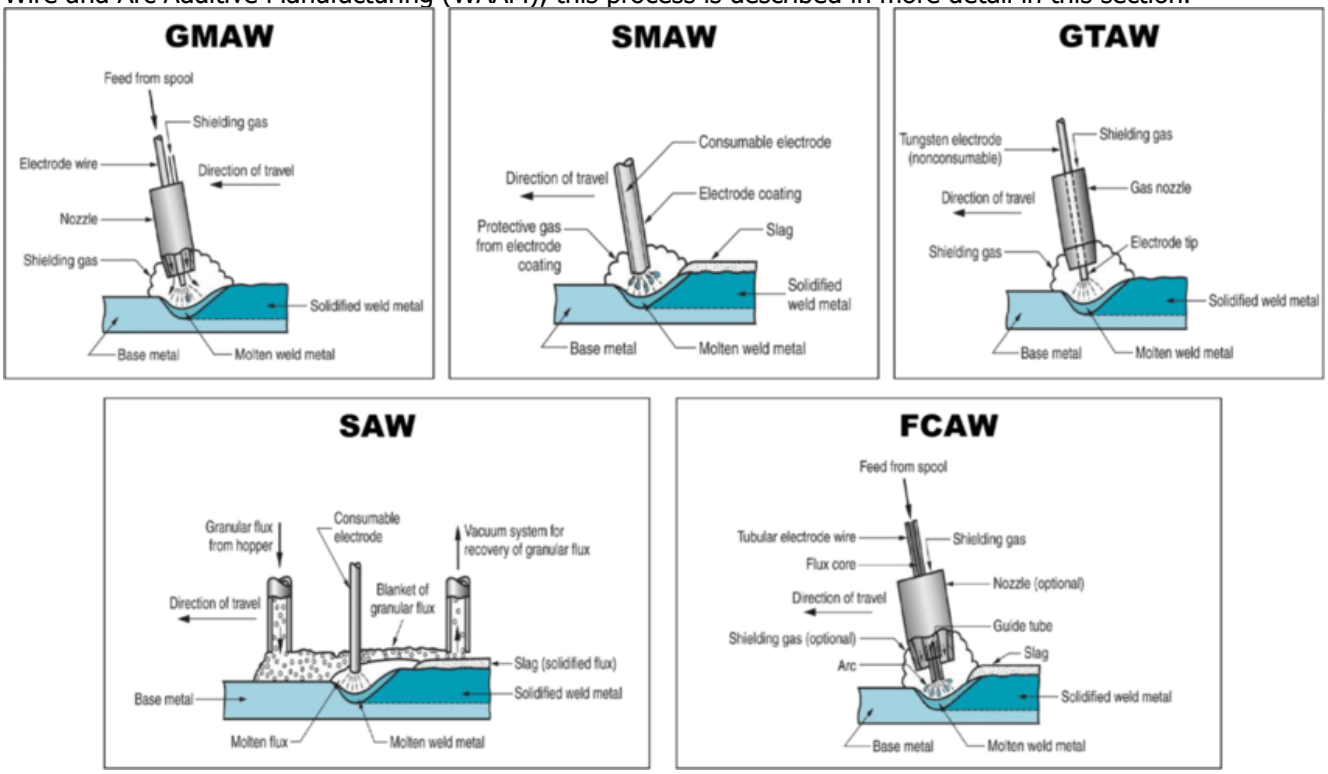


Figure 2-4: A rough overview of different arc welding processes. Although in theory any of these could be used for WAAM, the most conventional are GMAW and GTAW. From Jenney CL, O'Brien A (2001) Welding handbook, Vol. 1 [14]

In arc welding processes an electric arc is used as heat source to melt the material. The arc is ignited between a non-consumable or a consumable electrode and the work piece. A schematic of the main arc welding processes is provided in Figure 2-4. In Gas Tungsten Arc Welding (GTAW) a non-consumable

tungsten electrode is used and the wire is independently added to the melt pool. In Gas Metal Arc Welding (GMAW) the electrode also functions as filler material and is thus a consumable. The welding torch is moved along the required tool path. A weld is deposited, and multiple welds form layers and eventually structures.

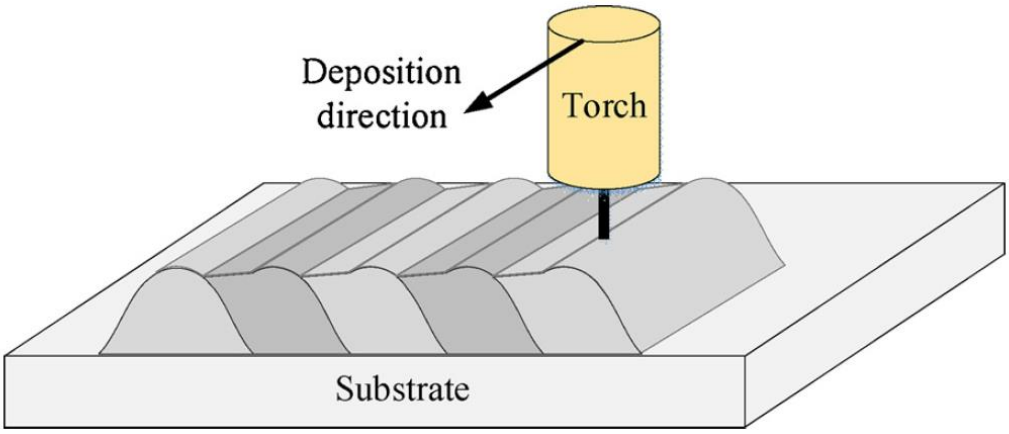


Figure 2-5: Wire and Arc Additive manufacturing in the flat position. Taken from Yongzhe Li et al. [15].

Figure 2-5 shows the basic principle of WAAM in the flat (1G) position. By spacing the depositions, a layer of welded material can be built up. This spacing is dependent on bead geometry, which in turn is dependent on the viscosity of the molten metal and the direction of gravity. If the spacing is too close, the bead will form on top of the previous bead. If the layer spacing is too wide, individual beads will be formed leading to irregularities and eventual defects in the layer. An optimal spacing ensures sufficient overlap to form a layer but with minimal variation of height across the layer.

By repeatedly welding layers upon layers, structures can be formed. Deposition strategies can vary endlessly, but some common patterns are shown in Figure 2-6. A raster pattern, for example, can be further broken depending on the orientation of individual beads and layers. The choice of a certain pattern will affect the thermal history of the entire structure. It can be expected that for an inward spiral pattern heat is accumulating in the centre of the plate, while for an outward spiral pattern the heat will build up at the edges. This also influences the quality of the layer, since the conditions of the arc are not the same at the start and end of the weld. The raster pattern leaves these start/stop regions at the edges where they can be easily machined away, but spiral patterns would have to account for possible defects. The thermal history, in particular the peak temperatures reached and the cooling rates, influence the evolution of the microstructure, but also has an effect on the development of stresses in the product.

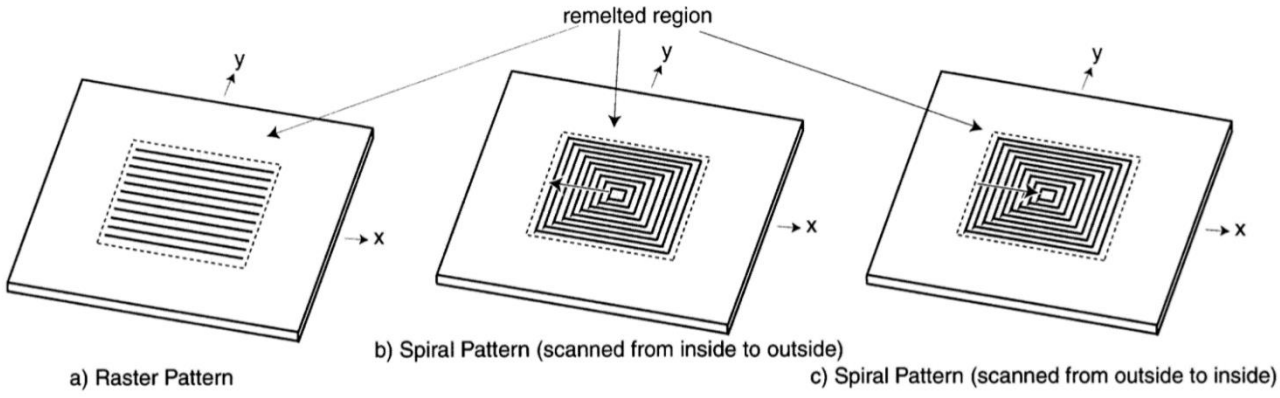


Figure 2-6: Deposition patterns for a flat plate. Figure taken from Nickel et al. [16].

Figure 2-7 shows a rough comparison of additive manufacturing methods according to various criteria. WAAM scores particularly well on build rate, part size, platform flexibility, mechanical properties, material utilization and the resulting cost savings. This makes WAAM suitable for larger parts, with the trade-off of less complex and accurate parts.

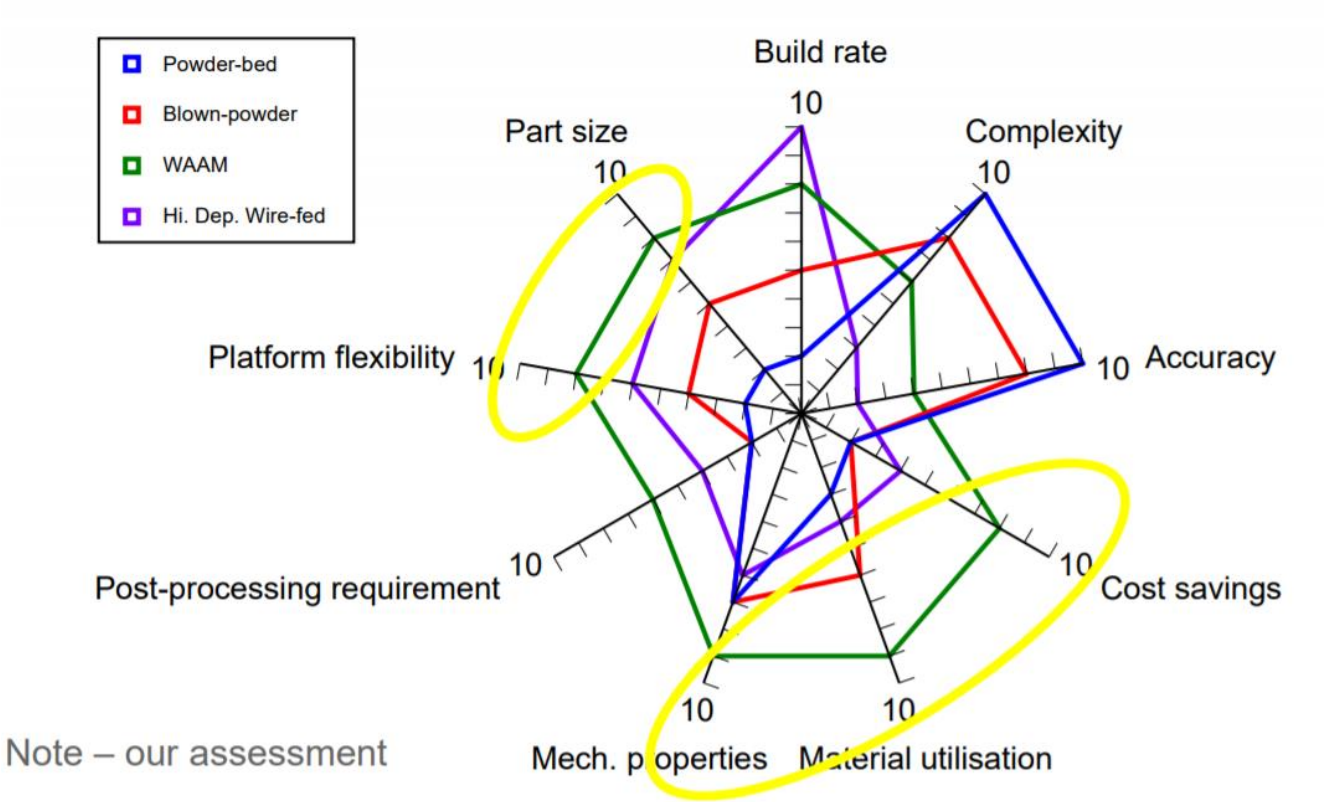


Figure 2-7: An assessment from WAAMMAT comparing WAAM to other common Additive Manufacturing methods [17]. NB: Hi.Dep. Wire-fed refers to Cladding or Hardfacing, a process where molten metal is sprayed on a surface.

2.2. The WAAM process

WAAM can be utilised with a variety of arc welding processes, each with different trade-offs. Gas Metal Arc Welding (GMAW) and Gas Tungsten Arc Welding (GTAW) are generally applied. Figure 2-8 shows a schematic of GMAW. An electric arc is struck between the work piece and a continuously fed consumable electrode wire. The tip of the electrode partially melts and the liquid metal is transferred to the workpiece in in the form of droplets (open arc welding) or via short circuits, creating a weld bead. The arc also locally melts the base material, resulting in good fusion between deposit and substrate. A shielding gas protects the weld from the surrounding atmosphere, preventing gas pore formation and unwanted oxidation. In GTAW an arc is struck between the substrate and a non-consumable tungsten electrode. Deposition is realised be feeding an additional filler wire. GTAW is a very stable process, but the travel speed and deposition rate are lower compared to GMAW.

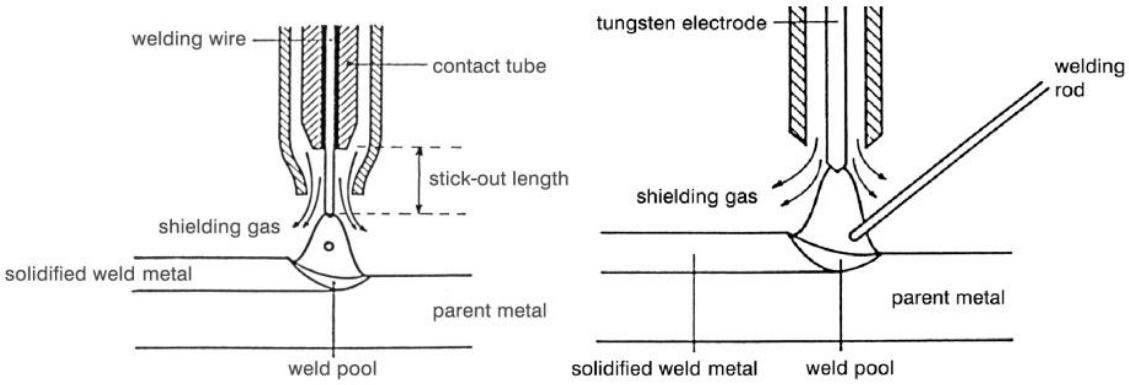


Figure 2-8: a) Schematic of the Gas Metal Arc Welding (GMAW) process, b) Schematic of the Gas Tungsten Arc Welding (GTAW) process. An arc is drawn between the electrode and the work piece. The electrode locally melts and deposits on the work piece, creating a weld bead. Shielding gas protects the weld from the surrounding atmosphere [4]. Pictures taken from [18].

During welding, heat is introduced in the work piece. If the deposited construct is too warm, the rate of solidification will be reduced, allowing the deposited liquid metal to flow from the designated deposition location and create defects. In open arc metal transfer modes the voltage and current are relatively high, which increases the chance of such defect formation. The weld process has seen many refinements to reduce this heat input, from short-circuit transfer to pulsed welding. Figure 2-9 shows the areas for each mode of transfer.

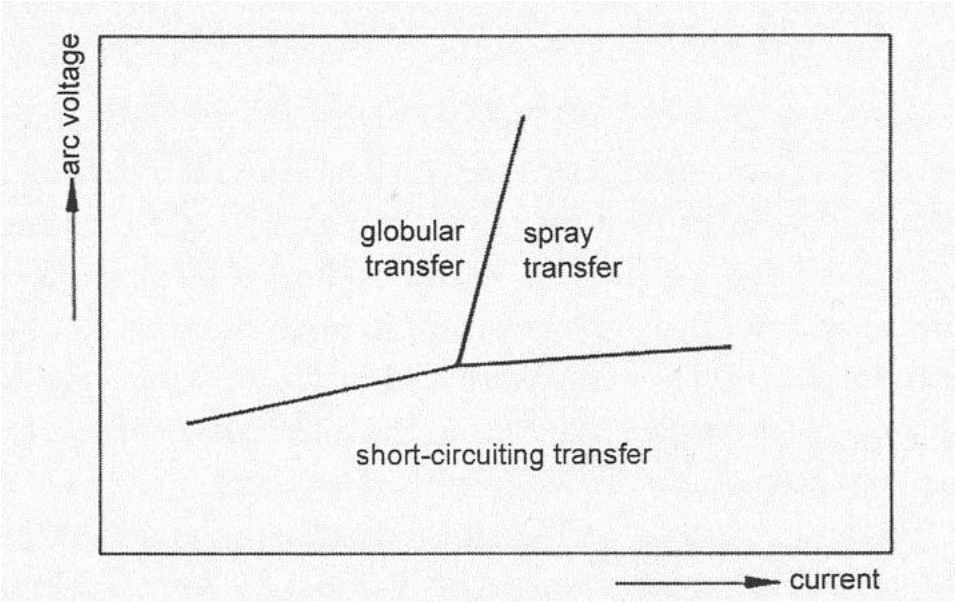


Figure 2-9: Operating areas, i.e. metal transfer modes in GMAW [18].

The particular weld technique used in this project is Cold Metal Transfer (CMT), which is a trademark of a modified short circuiting arc welding technique. As the name suggests, this technique results in welding with a low heat input [19]. General short circuiting reduces the heat input by having the wire make contact with the weld pool. This extinguishes the arc and causes the wire to drastically heat up and melt by Joule heating before explosively rupturing and re-establishing the arc. The temporary extinguishing of the arc reduces heat input. It is this rupturing of the liquid bridge that causes spatter formation to take place, which is undesirable.

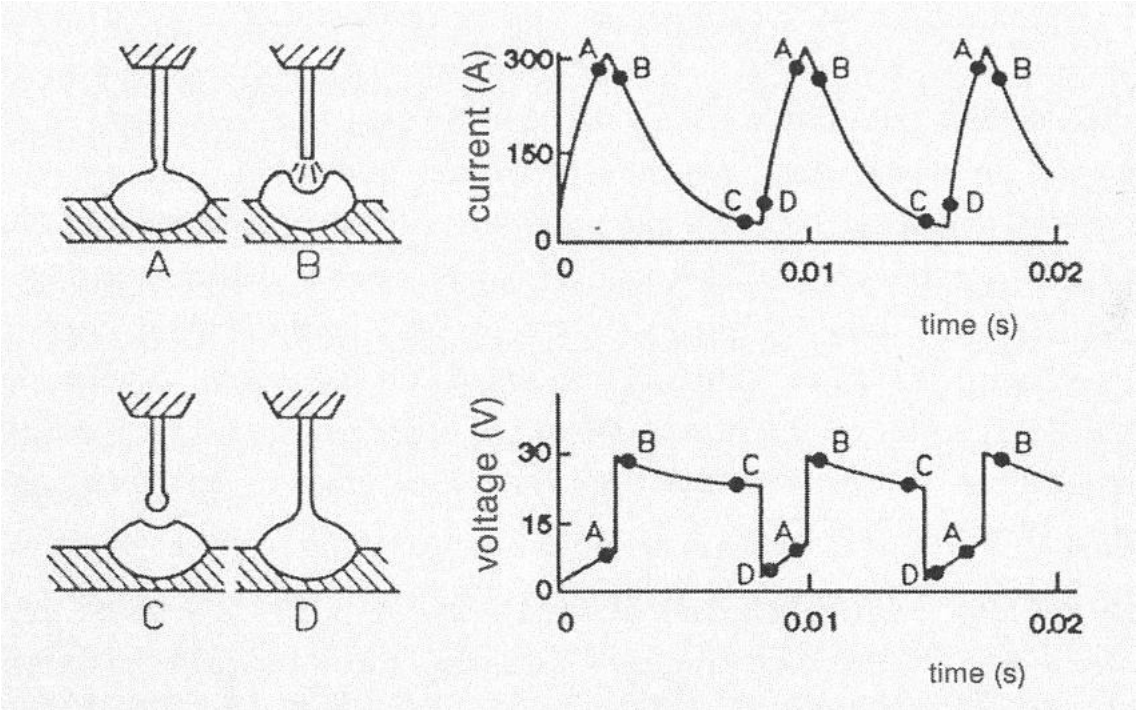


Figure 2-10: Schematic representation of conventional short circuiting arc welding indication the current and voltage as a function of time [18].

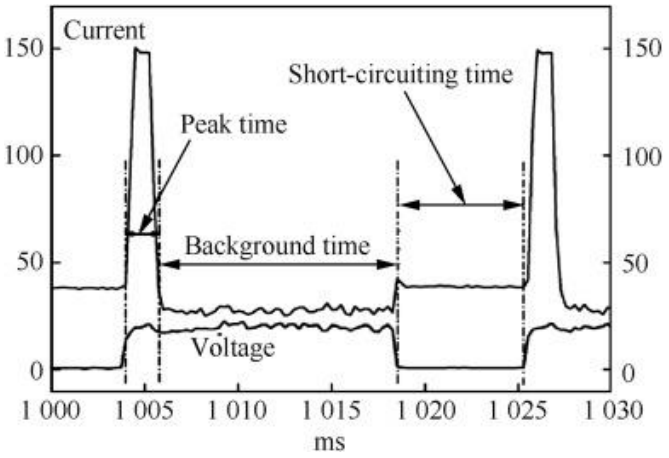


Figure 2-11: Cold Metal Transfer waveform [20].

CMT modifies this process by reducing the current as the short-circuit occurs, which reduces the electromagnetic forces and the occurrence of spattering. The wire feed speed is dynamically controlled, allowing the wire to be temporarily retracted to further enhance metal transfer from the wire to the weld pool. Figure 2-11 shows a CTM waveform, with three clear phases of the process. The first phase is the Peak Current Phase, where a high current establishes the arc and heats the wire. The second phases is the Background Current Phase where the current is reduced to prevent globular transfer and lasts until the short circuit occurs. The Short Circuit Phase is the last phase, where the arc is extinguished and voltage drops significantly. The short circuit is detected and signals retraction of the wire, which assists fracture of the liquid bridge and thus metal transfer from wire to weld pool. Once this cycle is completed, it starts again with a peak current to re-establish the arc. The arc extinguishing and background phase reduce the heat input compared to globular transfer, and the wire retraction reduces spatter.
CMT is often used in thin-plate welding, but the comparatively low heat input and reduced spatter makes it attractive for WAAM as well.

2.3. Research into WAAM

Additive manufacturing can to some extent be regarded as multi-pass welding. The construct is build be deposition layer-by-layer. This means that the deposited material experiences multiple thermal cycles. In the first cycle the material is deposited and will solidify. During the deposition of subsequent passes the prior passes will experience a heating, eventually re-melting and cooling. This repeated thermal cycle will affect the microstructure of the metal and thus the properties.
WAAM research often involves the building of simple single walls [4] [21] [22] [23] [24]. Yang et al. researched the thermal field during wall deposition and during cooling by means of an infrared camera, see Figure 2-12. The large difference in temperature between the weld pool and the surrounding material is an artefact of the drastically different emissivity of the weld pool compared to the base material.

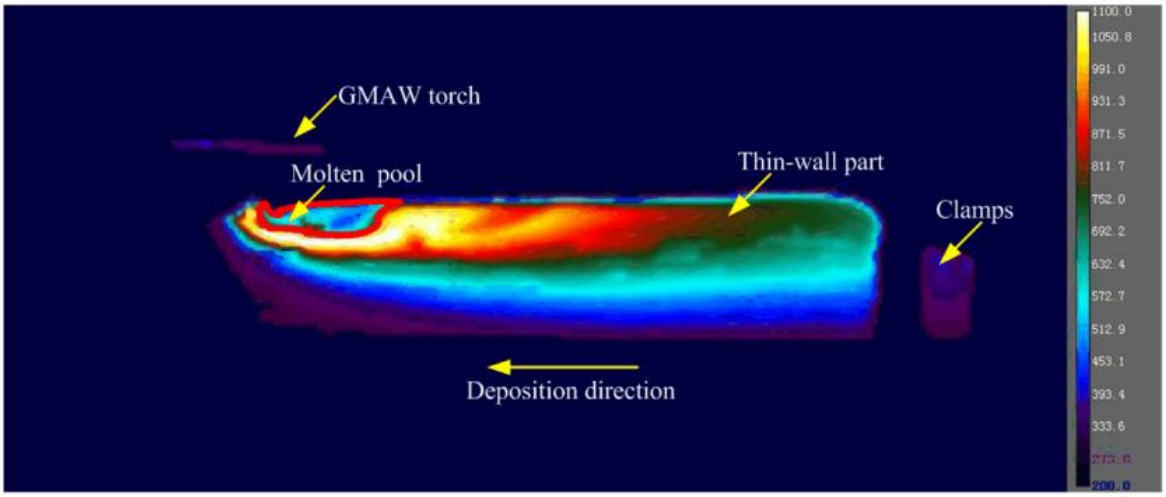


Figure 2-12: Temperature field distribution (Celsius) of a thin-wall part by IR camera during deposition. Image taken from [11].

As the wall is built up layer-by-layer, the temperature field remains remarkably similar. Only the first few layers show a large difference, though this is likely due to the different shape of the substrate. Once the wall is several layers high, the geometry remains constant for subsequent layers. The flow of heat from the weld pool into the material and further is not notably different for 5 or 20 layers. The entire layer remains hot (above 200 ^{o}C) until the end of the deposition process.

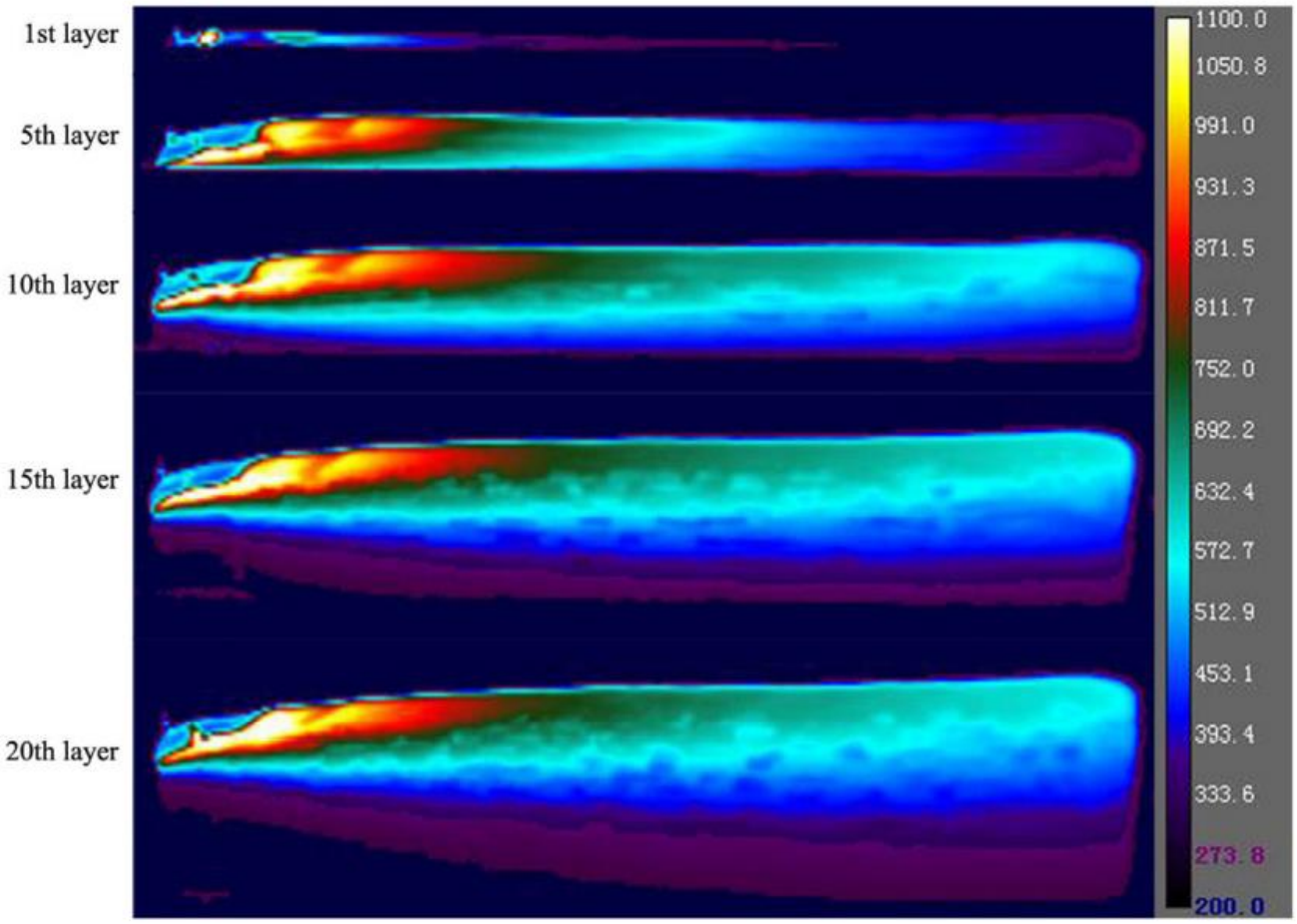


Figure 2-13: Temperature field (Celsius) at the end of welding for increasing layer height of a single-wall structure. Note the similar temperature field for the different layers. The weld was deposited from right to left. Image taken from [11].

Li et al. investigated reducing the heat input and grain anisotropy by using pre-heated welding wires. This led to a reduction in wall width and an increase in layer height. In addition, the grain morphology changed gradually from coarse columnar to equiaxed, with an associated reduction of tensile strength anisotropy.
Wu et al. [25] used low-frequency pulsing TIG to reduce the heat input allowing to produce greater overhang structures without support, but were not able to obtain the desired grain refinement.

Haden et al. [26] studied wear, hardness and tensile properties of stainless steel type 304. After building a single-bead width wall, samples were taken at the middle of the wall, at the start, middle and end of this bead. This corresponds to the labels MB, MM, ME. Hardness analysis of the three samples reveals values similar to the wrought stainless steel equivalent (Figure 2-14). By interpolating the hardness for the indents of the MB sample , a contour plot can be made. Indents are spaced 0.25 mm in the X direction and 0.2mm in the Y direction. Weld layers are deposited in the X direction, with an average layer height of 1.8mm. No weld-bead pattern is visible in the hardness results.

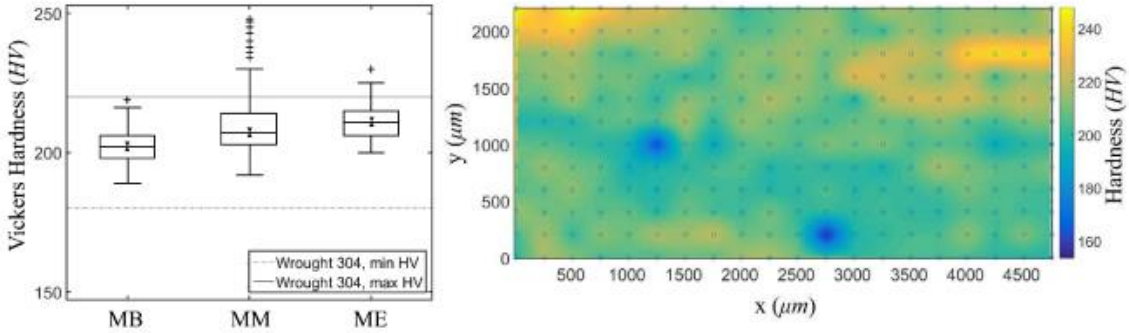


Figure 2-14: Average hardness per sample (left). Interpolated hardness contour plot of the MB sample.

Xiong et al. [22] studied the effect of preheating on temperature gradients in the weld metal and microstructure for structural steels. A circular thin-wall structure was build using GMAW and simulated by applying a Goldak heat source. They found that preheat temperatures up to 400-600 °C would reduce the thermal gradients in the material. Figure 2-15 shows the thermal cycle for four different substrate preheats, and also indicates the melting point and the interpass temperatures in the thermal cycle. Each peak represents one pass of the welding arc. Note that for preheating to 400°C and 600°C, the interpass temperatures in the cycle remains approximately similar for subsequent layers, whereas it increases for the room-temperature and 200°C preheating cases. The figure clearly shows that the material experiences thermal cycling, with repeated re-heating and re-melting.

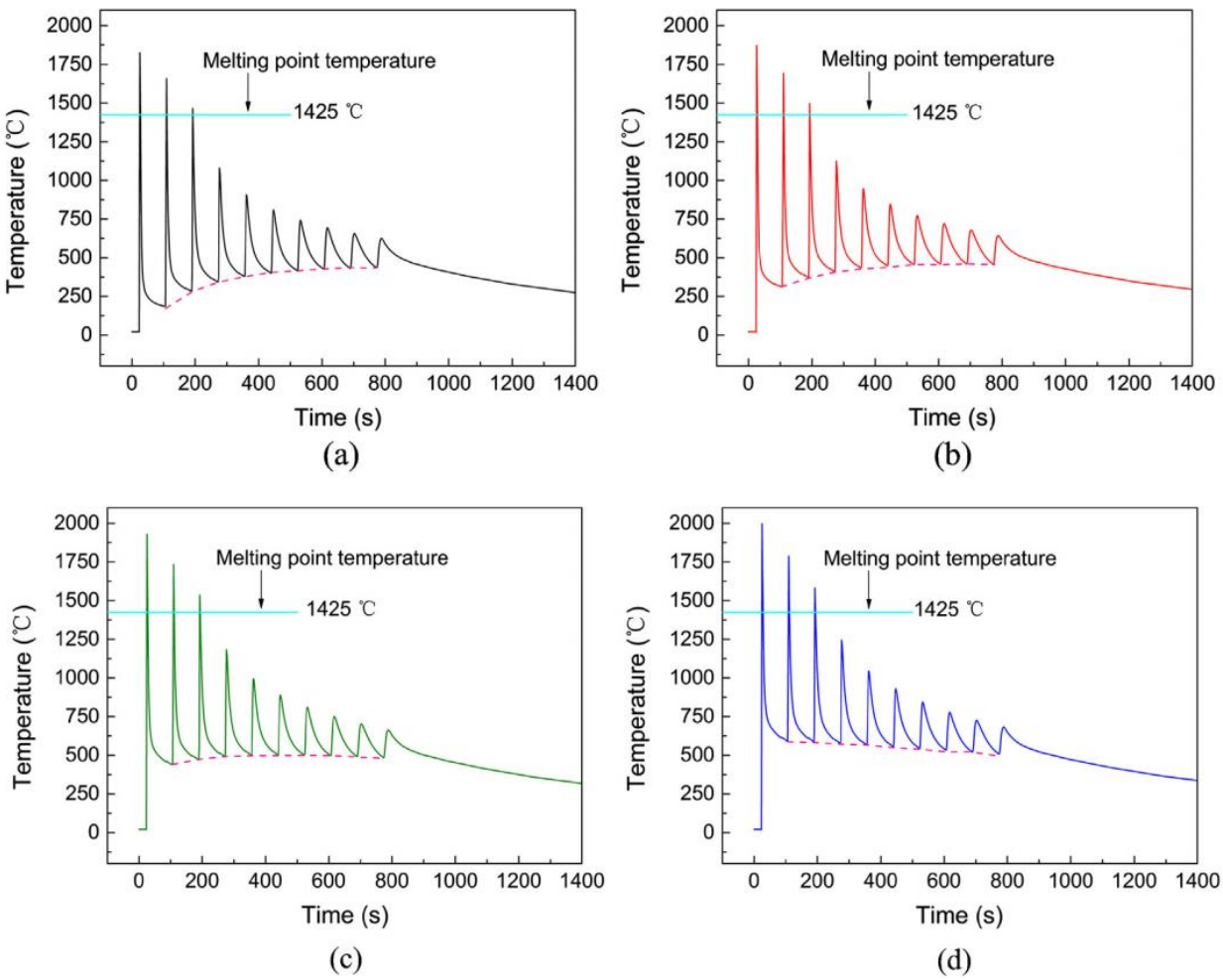


Figure 2-15: Temperature history of the centre point of the first layer for different substrate preheat temperatures. (a) Room temperature, (b) 200°C, (c) 400°C (d) 600°C. From Xiong et al. [22].

Lastly, Aldalur et al. [27] studied the effect of a torch weaving versus overlapping method. A weaving motion means the torch follows a sinusoidal path. The overlapping method involves several adjacent beads per wall layer. Figure 2-16 shows the temperature measurements taken during the deposition of a wall using each method. This shows a considerable advantage for weaving due to being able to weld the entire width of the wall in one move, whereas the overlapping method requires multiple passes where each pass needs to wait for the temperature to reach the accepted interpass temperature level.

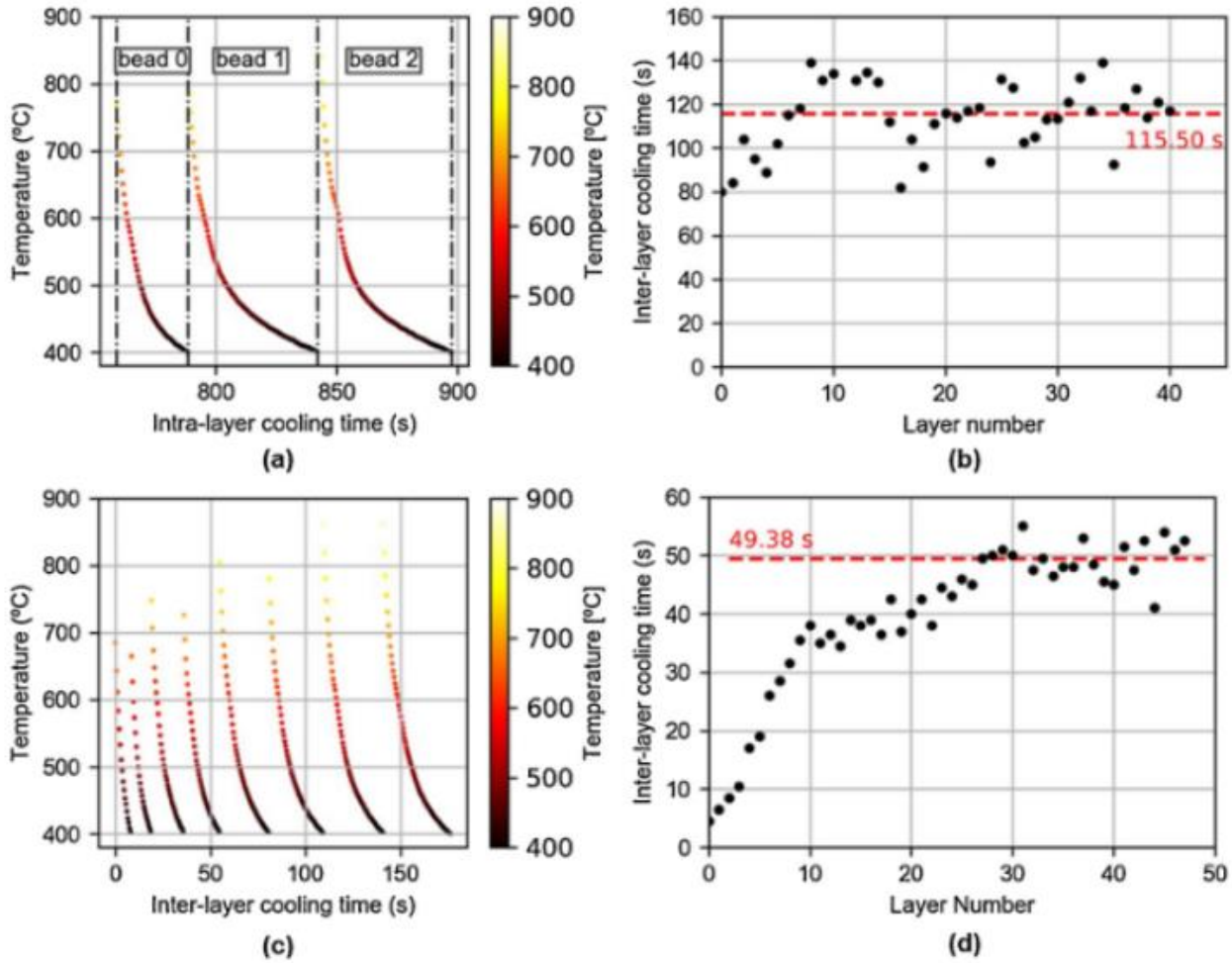


Figure 2-16: (a) Temperature evolution during the intra-layer cooling time in the eighth layer of the overlapped wall; (b) Inter-layer cooling time evolution along the overlapped wall; (c) Temperature evolution during the inter-layer cooling time in the first eight layers in the oscillated wall; (d) Inter-layer cooling time evolution along the oscillated wall. Taken from Aldalur et al. [27].

Figure 2-17 shows stress-strain curves of multiple samples, in both the weld direction (horizontal) and layer direction (vertical). The oscillation strategy has a lower tensile strength of around 480MPa compared to the 500MPa of the overlap strategy. The yield strength is also lower, about 340MPa and 380MPa respectively. With the oscillation strategy the local temperature remains much higher, leading to longer cooling times and also a longer tempering of the underlying structure. Still, the significantly improved deposition rate suggests this is a worthwhile trade-off.

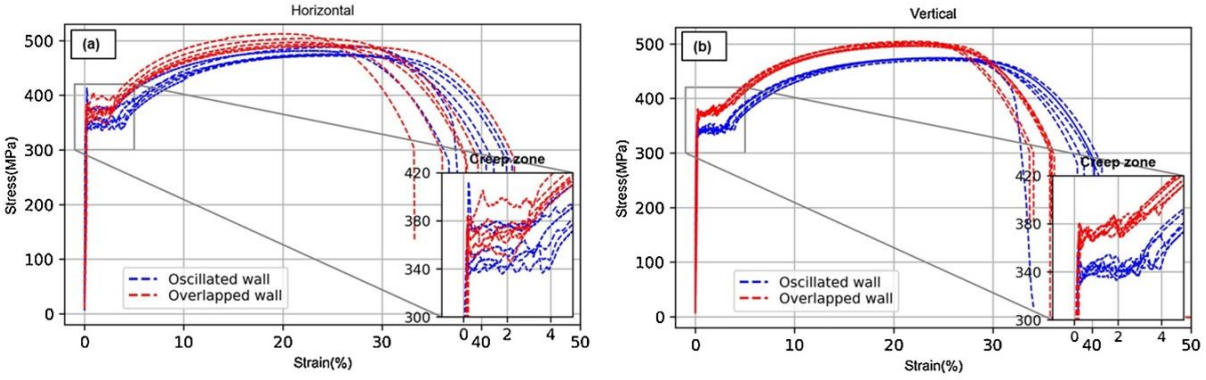


Figure 2-17:Stress-strain curves of several samples of the wall for each method. Horizontal samples are plotted left, vertical samples are plotted right. The oscillated wall shows a slightly lower strength (around 40 MPa yield strength and tensile strength) and higher strain at fracture for the vertical samples. Taken from Aldalur et al. [27].

To numerically obtain the thermal history, Seleš et al. [28] compared the Goldak double-ellipsoid heat source model with a prescribed temperature method for a mild-steel welded by Metal Active Gas (MAG) welding. Figure 2-18 shows the calculated results between the Goldak method (CHF) and the prescribed temperature approach (PWT). Notably, for a large prescribed temperature range between 1900°C and 1550°C, the results are very comparable, showing that the Prescribed Weld Temperature(PWT) method is fairly robust. The Goldak approach leads to significantly higher peak temperatures of 3000°C compared to the prescribed temperatures. Comparing the temperatures after 60s, the prescribed weld temperature of 1725°C is the closest to the Goldak result, indicating that this is the closest fit. As will be discussed later, in this work the 800-500°C is considered the most interesting temperature region regarding phase transformation in steel taking place, meaning the prescribed temperature approach will provide a good and valid approximation of the cooling time.

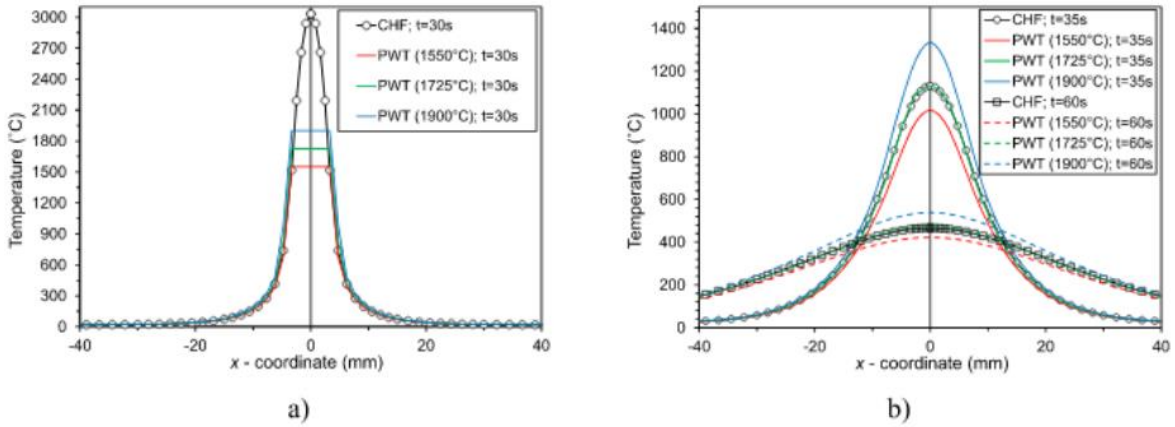


Figure 2-18: A comparison between the Goldak Double Ellipsoid method (CHF) and the prescribed temperature method (PWT) for 30s and 60s after welding a fillet weld. [28]

These studies are process related, as the selection of different process parameters has a clear impact on the resulting structure, anisotropy, defects and overall quality of the end product. The resulting microstructure and the residual stress level determine the mechanical properties. They show the importance of temperature management and the right choice of deposition strategy to obtain the required microstructure and mechanical properties.

2.4. Microstructure in a multi-bead weld

To understand the structure to be expected in welded blocks, first the microstructure development in a regular steel weld will be discussed. The weld pool is a region where the base material is molten and is mixed with the filler material. A schematic is shown in Figure 2-19a, where the weld process moves from left to right. The heat generated by the arc (q) is introduced into the base material and will cause melting of both the substrate and the filler material added ('L'). As the welding arc has passed, the material cools (heat losses due to convection, radiation and conduction) and starts to solidify ('S'). The direction of solidification is parallel to the maximum temperature gradient, i.e. perpendicular to the solidification front. In alloys, due to compositional changes at the solid-liquid interface, compositional supercooling may occur related to the solidification rate and the cooling rate, which results in different morphologies of the solidified material.
This solidification front in the weld pool is shown with a thick red arrow in the weld pool. G is the actual thermal gradient in the melt pool. A steep temperature gradient will lead to a planar microstructure (no compositional supercooling), but as the gradient decreases, the molten material cools into differently sub-structured microstructures (Figure 2-19b).

G versus solidification velocity maps are highly dependent on the chemical composition, making prediction of the final grain morphology difficult. Grain growth occurs along the maximum thermal gradient, shown here to be close to the X-direction in Figure 2-19a.

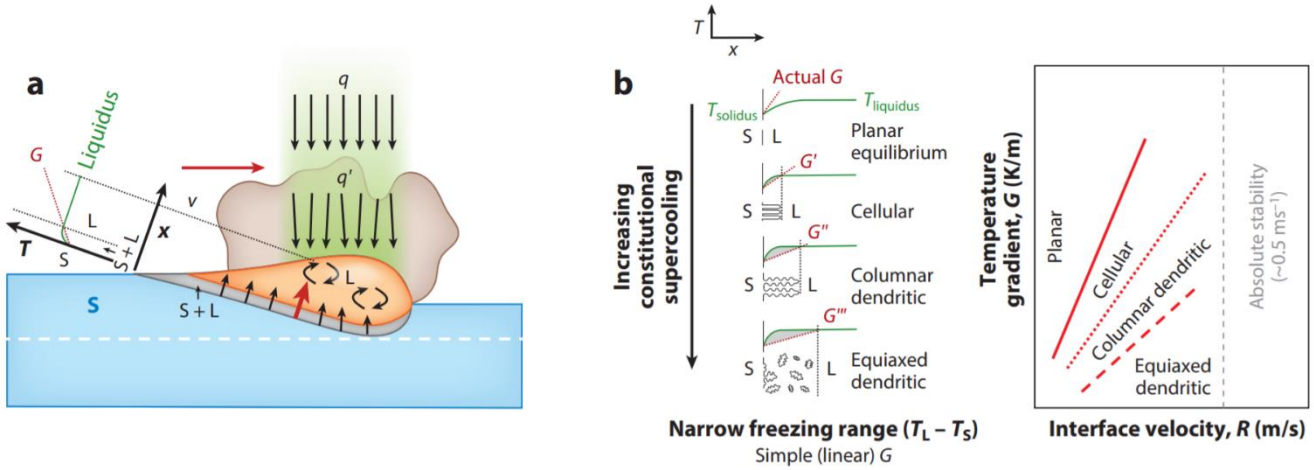


Figure 2-19: Schematic of a Weld pool (a)) and constitutional undercooling (b), from [29]

Upon solidification the resulting microstructure will experience solid state phase transformations; the austenite will transform. The final constituents depend on the cooling rate and can be predicted based on continuous cooling transformation (CCT) diagrams.

2.4.1. Heat Affected Zone development

Adjacent to the deposited bead is the coarse-grained heat affect zone, a region where the base material has been heated to 1100-1550°C for long enough that the material transforms to austenite and grain coarsening takes place. With increasing distance from the fusion line the peak temperature decreases and grain growth is less apparent. The fine-grained heat affect zone is heated to 850-1100°C, just above the A3 temperature, allowing the ferrite to austenite transformation and vice versa to take place. The solid state phase transformations result in grain refinement and the relatively low temperature avoids excessive grain growth. The intercritical zone is the region where the base material is only partially transformed to austenite and experiences temperatures high enough (750-850°C) for tempering effects. Finally, a subcritical zone can be distinguished, which is heated (500-750°C) but not sufficient for the austenite transformation. Thus, the material in this region experiences only tempering. Figure 2-20 shows the zones in a schematic form.

2.4.2. Multi pass weld

Figure 2-21 shows the heat-affected zones in a multi-bead weld. The heat-affected zones are re-heated by the new subsequent weld, affecting the original heat affected zones. The coarse-grained zone is fully re-austenitized and grain growth takes place to reduce interface energy. and so has the same microstructure as previously. The coarse-grained heat-affect zone (CG HAZ) that is reheated between 850-1100°C, is defined as the supercritically reheated zone. The CGHAZ reheated to 500-750°C is the intercritical reheated CGHAZ and finally the tempered region (500-750°C) is the subcritical CGHAZ.

In the case of a WAAM block, welds are also stacked on prior welds, creating a complicated multi-bead structure where heat-affected zones can be re-heated multiple times depending on the deposition strategy. Because of this, simulating the hardness over a volume can be problematic in terms of the required resolution.

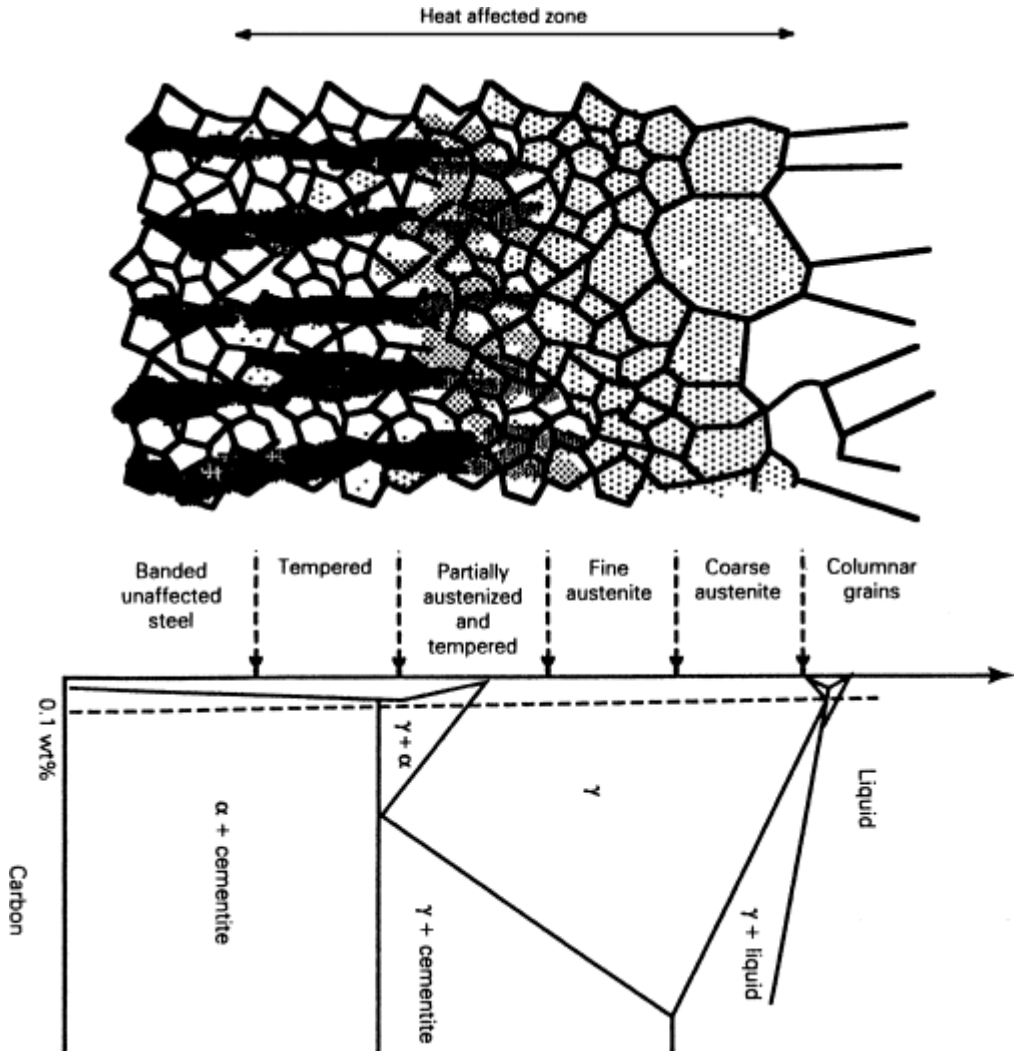


Figure 2-20: Schematic illustration of the microstructural variation to be expected in the HAZ of steel welds. The microstructural zones do not coincide with the equilibrium transformation temperature since heating and cooling rates during welding do not approach equilibrium rates [30].

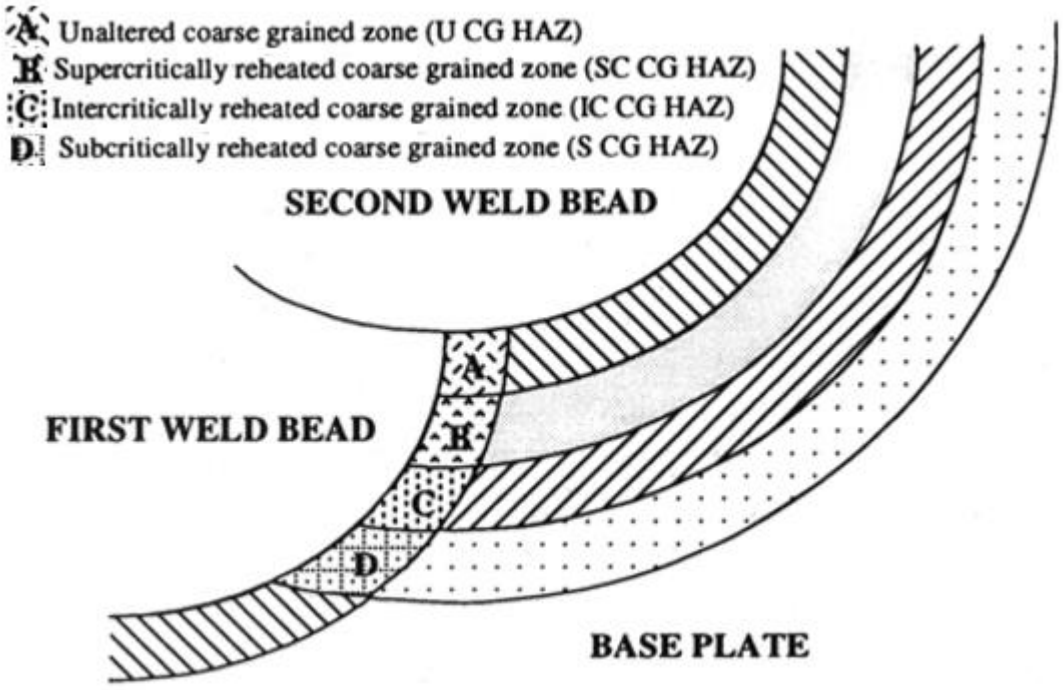


Figure 2-21: Schematic diagram of the HAZ of a multi-pass weld. Picture taken from [31]

Continuous Cooling Transformation Diagrams

The iron-carbon phase diagram can be used to determine the thermal equilibrium phases of steel at any temperature and composition. The thermal equilibrium means that time is not a factor. However, in real welding practice, cooling rate is an important factor. To consider cooling rates, Continuous Cooling Transformation (CCT) diagrams can be constructed. By subjecting a sample to a constant cooling rate and measuring the dilation, it is possible to track phase transformations. Subsequent metallurgical examination is used to determine the microstructures that occur as well as the hardness. By repeating this procedure for different cooling rates, a CCT diagram can be obtained.

Figure 2-22 shows an example of a CCT diagram for a 34Cr4 steel. Samples are austenitized at 850°C with a holding period of 8 minutes and then continuously cooled. The cooling rates are shown as lines, decrease from left to right, starting at around 800 °C in the diagram. Phases are shown as areas with a letter marking the phase. The numbers where cooling lines cross the phase field area boundaries are phase percentages. Two lines will be followed, one with a low cooling rate ending in the bubble 180 and one ending in 56. At low cooling rates, the austenite transforms to 50% ferrite at around 750 °C, due to the low undercooling giving the carbon around 10^4 seconds to diffuse. The remaining austenite is now oversaturated with carbon, and undercooled even more. The carbon diffuses to form regions of low and high carbon, forming a coarse ferrite-cementite structure named pearlite.

At an extremely high cooling rate, the undercooling is large and so the driving force for transformation is large. The martensitic phase transformation takes place at around 380 °C.

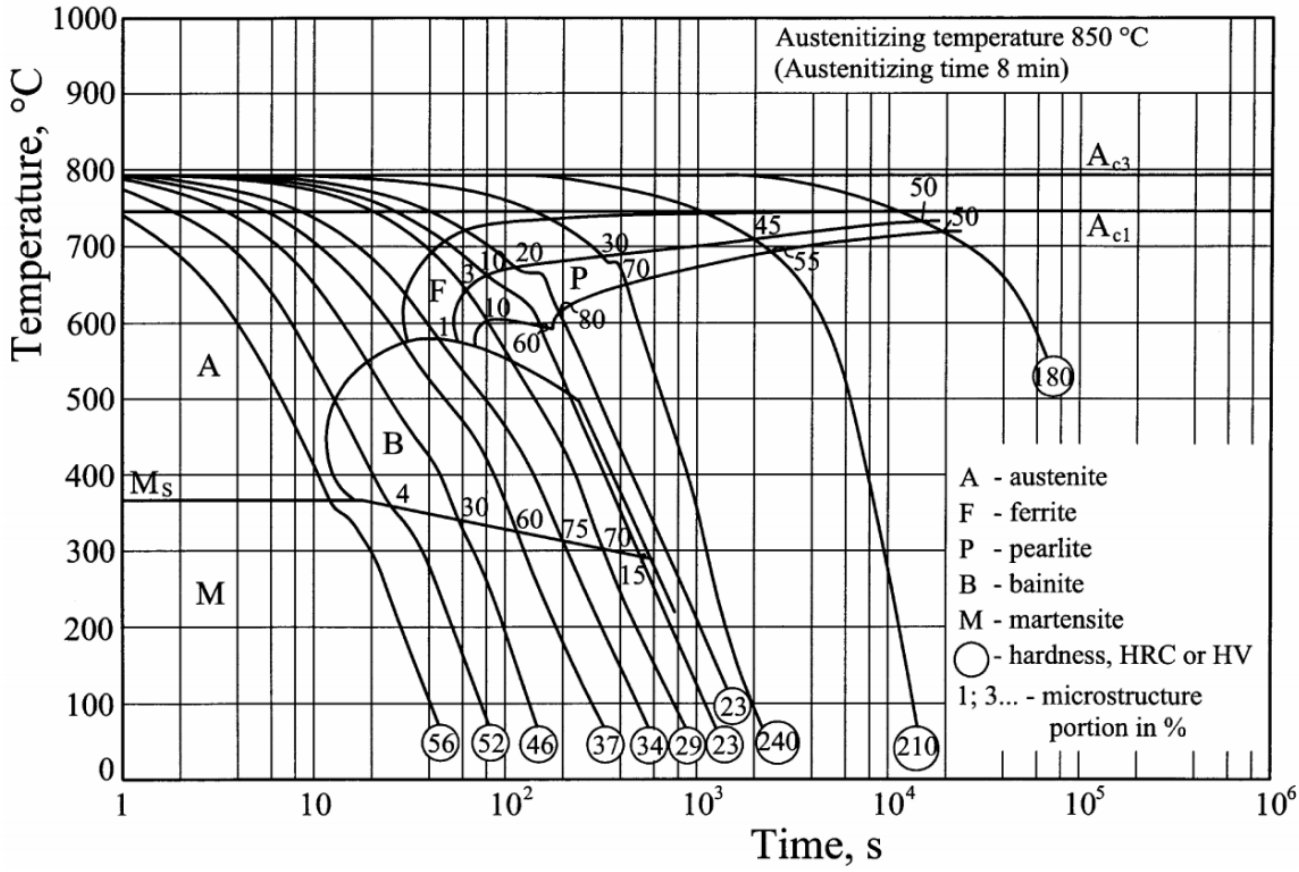


Figure 2-22: Example Continuous Cooling Transformation (CCT) diagram of 34Cr4 steel [32]

Such a diagram allows estimation of the phases present for a certain cooling rate, as well as the prediction of the hardness of a material.

For structural purposes, it is important to know that the mechanical properties of the built block match a certain specification, to ensure the structure can withstand designed loads. For toughness or hot cracking requirements, a maximum hardness is usually prescribed. The properties of a weld depend on the thermal history of the deposited material, and thus on the welding process conditions, geometry of the constructed part and the weld order. Hardness is a property that is indicative of tensile and yield strength [33], and is easily measured across the block. This makes it interesting as an indication of the property distribution of the welded block: harder material is expected to have a higher yield and tensile strength. The aim of this thesis is thus to simulate the thermal history of a block, extract the local cooling rate and predict the hardness based on data from CTT diagrams.

2.5. Finite Element Analysis.

Finite Element Analysis or Finite Element Method (FEA/FEM) divides a continuum material into discrete elements to calculate a numerical solution to a problem. Each element contains connected points called nodes. A collection of elements creates the mesh of the considered part.

2.5.1. Heat Transfer

To calculate the thermal field, considering heat transfer, C3D8 elements can be used, indicated by Figure 2-10. This is a 3-dimensional, 8-node element. In heat transfer models, the degree of freedom for each node is the temperature. The temperatures are thus calculated only at the nodes, temperatures at other locations are linearly interpolated.

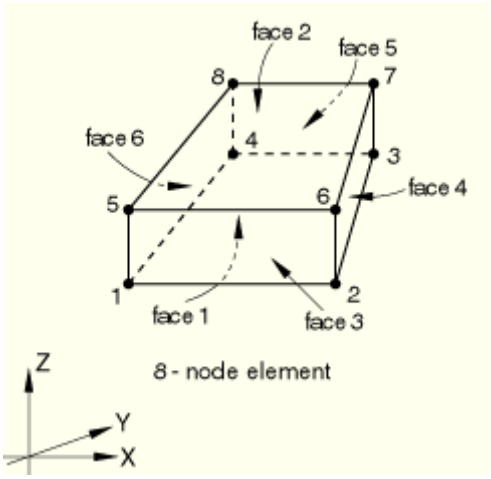


Figure 2-23: Abaqus User Manual 22.1.4. 8 node brick element [34]

For heat transfer models, equation 1 can be applied.

$$c\dot{\boldsymbol{T}} + k\boldsymbol{T} = \boldsymbol{f_T} \qquad Eq.(1)$$

In this equation, $\boldsymbol{c}$ represents the temperature-dependent heat capacity matrix, $\boldsymbol{k}$ is the temperature-dependent conductivity matrix, $\boldsymbol{f_T}$ is the temperature field vector, and $\boldsymbol{T}$ and $\dot{\boldsymbol{T}}$ are the nodal temperature and its time-derived matrix, respectively. Calculating with a heat flux input $\boldsymbol{f_T}$ is computationally time consuming in the weld region due to the high temperature gradients, which are the result of temperature differences between high peak temperatures and the surroundings. This requires finer time increments, driving up the computational time [28].

The convection Q_{conv} and radiation Q_{rad} are handled via boundary conditions expressed as equations 2 and 3:

$$Q_{conv} = h_c * A_c * (T - T_{amb}) \qquad Eq.(2)$$

where h_c is the heat transfer coefficient, A_c is the area on which the convection acts, and T_{amb} is the ambient temperature, and

$$Q_{rad} = k_b * A_{rad} * (T^4 - T_{amb}^4) \qquad Eq.(3)$$

where k_b is the Stephan-Boltzmann constant, A_{rad} is the area on which the radiation acts, and T_{amb} is the ambient temperature.

Rather than using a prescribed flux, the welding can also be simulated with a prescribed temperature approach [28]. The welding arc is simulated as an area at high temperature. By changing which area experiences the high temperature over time, weld arc movement is simulated. This thermal boundary condition is applied over a short amount of time, consistent with how long the torch would be above the specific element as it moves. By sequentially applying new boundary conditions and removing the old, the movement of the active weld torch is simulated.

2.5.2. Material addition: Quiet element and inactive element methods.

In addition to heat, the welding process also introduces new material. There are generally two ways to achieve this in numerical simulations, called the quiet element method or the inactive element (element-birth-death) method [35] [36] [37].

In the quiet element method, the elements that represent the deposited material are present from the start of the analysis. These elements are given artificial properties to ensure they do not affect the analysis. The thermal conductivity and heat capacity are set to low values to ensure low heat transfer into the quiet elements. The advantages are the easy implementation in commercial finite element codes via user subroutines, and the unchanging number of elements means no equation renumbering and solver initialization is needed. The disadvantage is that insufficiently low thermal conductivity and heat capacity results in errors. The implementation can also be time-consuming due to additive manufacturing model domains largely consisting of quiet elements.

With the inactive method or element birth-death method, the elements that represent the deposited regions are removed from the analysis at the start, and re-activated as needed. Nodal degrees of freedom for inactive elements are ignored. The advantages are that there are no errors from insufficiently low artificial properties, and only the active elements are considered leading to smaller algebraic systems. The downside is that the equation numbering and solver initiation has to be repeated every element activation, increasing calculation time. Additionally, nodes shared between an active and inactive element may not have the correct temperature upon activation, which results in artificial heat transfer. Figure 2-24 illustrates this. Element one is prescribed a high temperature T1. Element two is prescribed a low temperature T3. Depending on how the data is processed, the temperature of Node 2 can either be T1 or T3. In the program used for this thesis, the process led to T2=T3, rather than the intended T2=T1. The temperature profile thus followed the thick line, rather than the desired dotted line. Depending on the FEA code, the reverse may also be true. In this thesis, the problem is solved by prescribing a boundary condition on Node 2.

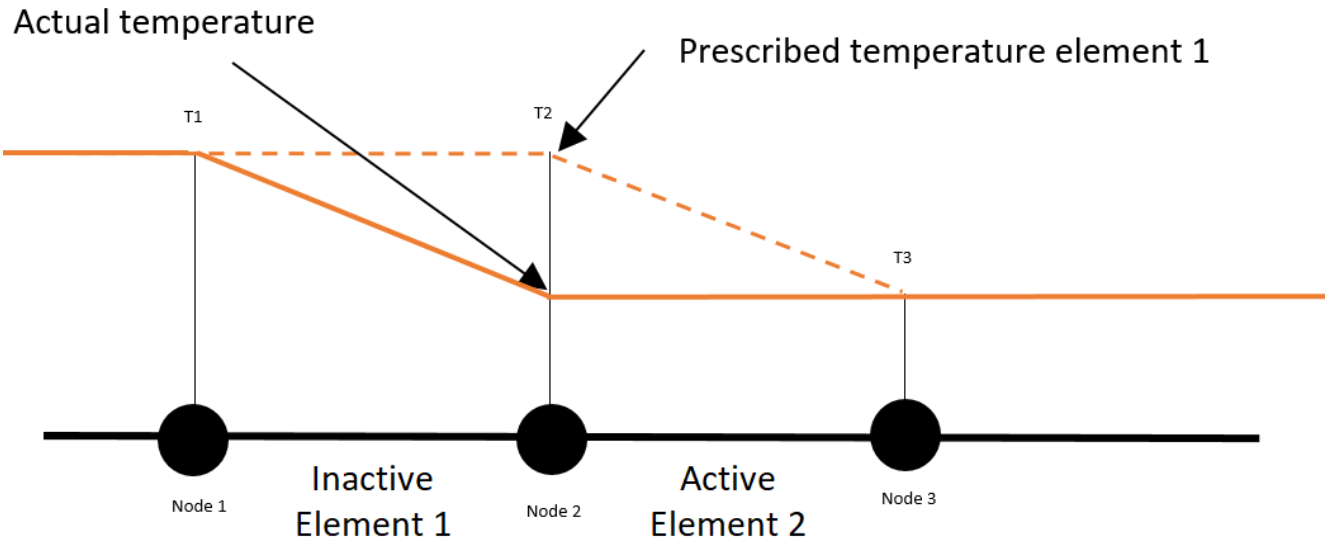


Figure 2-24: Influence of element activation on nodal temperature.

3. CHAPTER 3: METHODS

This chapter will describe the implementation of the thermal and hardness model, as well as the setup of the validation experiments.

3.1. Experimental details

This chapter will discuss the details of the experiments performed and the construction of a modelling framework to obtain the thermal history and predictions of the resulting hardness profiles.

3.1.1. Experimental set-up

Coupons were printed making use of a robotic system consisting of a GMAW power source (Fronius TSP 5000 CMT) connected to a robotic system (Fanuc M710iC) displayed in Figure 3-1. The robot is a 6-axis robot, meaning it has six degrees of freedom in which it can move: three direction of translation and three direction of rotation. This offers freedom in the way the weld can be deposited and so even complicated shapes can be made with relative ease. This means the size of the product that can be made this way is limited only by the reach of the robot. During welding current and voltage were measured by means of a Yokogawa DL716 scope. In addition the temperature was measured at several locations during welding by K-type thermocouples.

Figure 3-1: The welding setup is a FANUC M710iC 6-axis robotic arm with Fronius welding torch. Photo was taken at the TU Delft welding labs.

A welded specimen, Block A, with dimensions of 200 x 92 x 83 mm^3 (length x width x height) and a building angle of 72° was printed on a pipeline steel base plate (350 x 150 x 37 mm^3) consisting of 29 deposited layers, see Figure 3-2. The surface is slightly curved as the plate was taken from a standard 46 inch pipe, with a 1168.4 mm diameter. Layers 1-8 were deposited in the morning, followed by a break. Another 8 layers were deposited in the afternoon. Layers 17-24 were deposited the next morning, with the remainder of the block deposited in the afternoon. Each layer was deposited in a series of welds in a raster pattern, with the solid orange lines representing the weld direction and the dotted orange lines the welding order. After completion of the layer, the start position for the next layer was checked, the welding torch inspected, the layer brushed with a steel brush to remove spatter, and the next set of instructions loaded into the robot. This preparation took roughly 5 minutes on average, allowing the freshly deposited layer some time to cool. After each major break the block was entirely at room temperature.

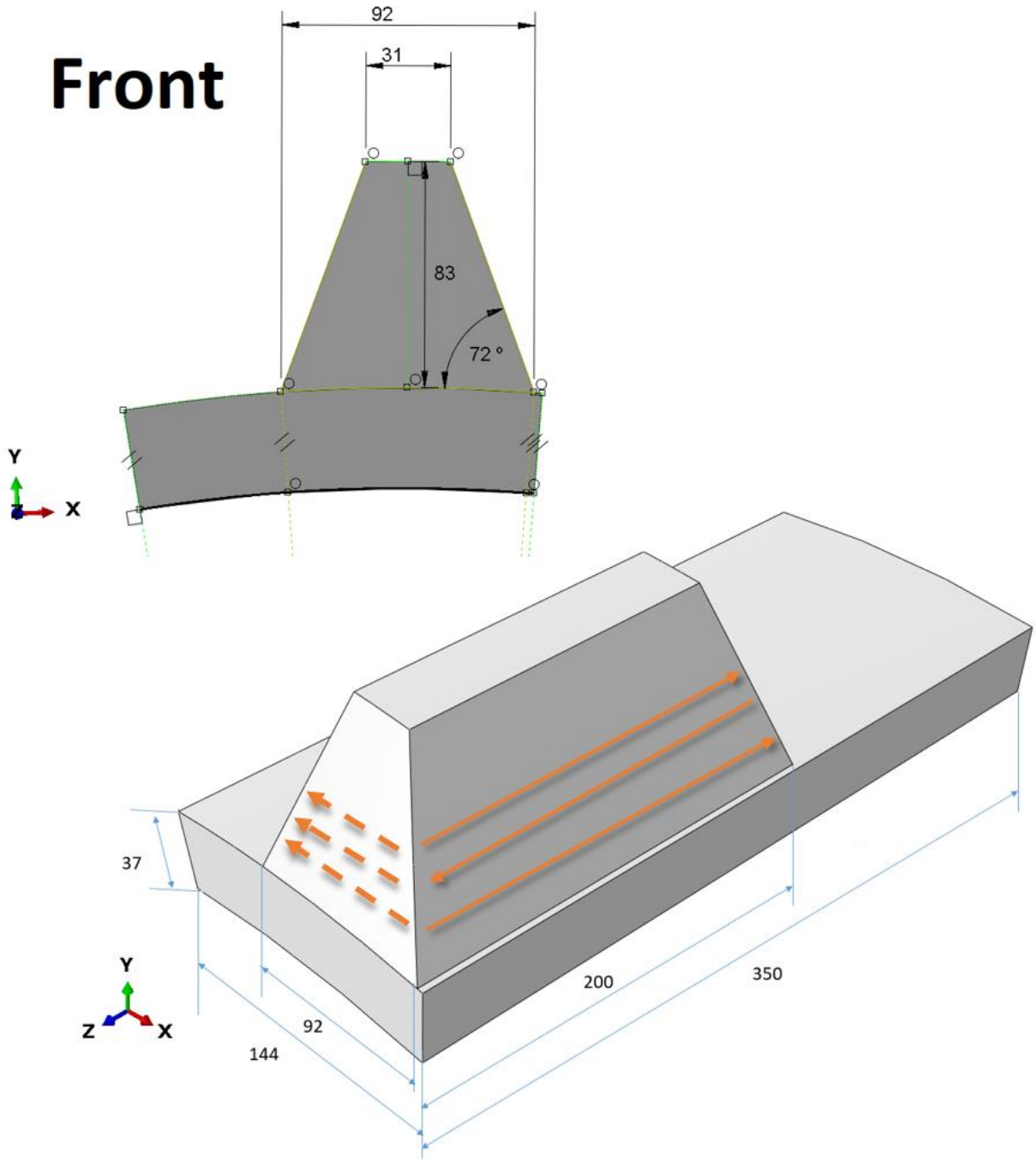


Figure 3-2: Block A, built on a section of pipe. All dimensions in millimetres. The base of the printed block is 200 mm long and 92 mm wide. The height of the block is 83mm with a 72° slope. The solid orange arrows indicate welding direction, which alternates every layer. The dashed orange lines indicate the direction the beads that are subsequently deposited, which is the same direction every layer.

Block A was designed specifically for mechanical testing, leaving an insufficient amount of material for metallurgical evaluation. To obtain this additional information, block B was made according to the same procedure of alternate layer direction welding. Since this block did not need to be as large to accommodate suitable test samples, the dimensions were changed to facilitate easier handling. The dimensions of the printed part were reduced to 120 x 60 x 30 mm^3.

Block C was printed with 6 layers in one direction, 3 layers in the opposite direction, and the final in the original direction.

Block A was fully instrumented and temperature measurements were carried out to validate the thermal models, whereas block B and C were used for hardness measurements. Due to initial difficulty with stacking layers, the walls were tapered for better layering for Block A. Subsequent mastery of the process led to smaller blocks with vertical walls for Blocks B and C.

3.1.2. The process parameter

The welding parameters used for all three blocks are listed in Table 1. The welding was performed in the 1G (PA) position. The weld strategy for block A and B was similar. Parallel beads were deposited, using the same direction throughout the layer. Each welding bead was deposited along the longest dimension of the block, minimizing the number of welds needed. Once a layer is deposited, the start positions of each bead become the end position of the next layer and vice versa. The current and voltage data is the average value provided by the power source, as specified by the operating program. These are average values that the welding machine attempts to maintain, but on a millisecond scale the voltage and current varies considerably, as discussed in Chapter 2. The contact tip distance is the distance between the tip of the retracted filler wire and the surface of the substrate. Although the contact tip distance is usually measured from the contact tube, for this equipment the used measuring strategy was easier to execute.

Table 1: Process Parameters GMAW:

Weld Technique:	CMT
Deposition current:	150 A
Arc Voltage	20 V
Torch Travel speed	10 mm/s
Shielding gas	80%Ar + 20% CO2
Gas flow rate	18 L/min
Contact tip distance	10 mm

3.1.3. Materials

The base material is an X65 pipe line steel commonly used for gas or oil pipes.
The composition of the pipe material and the welding wires is listed in Table 2. The first, OK Aristorod 12.50, is a common S355-rated filler wire and was used to build block A. The other two wires are of an increased strength grade to S420 and S460, respectively. The OK Aristorod 12.63 wire was used for building blocks B and C. The third wire (OK Aristorod 13.26) was unused in this thesis, but the properties mentioned by the manufacturer were used for simulation purposes.

Table 2: Chemical composition weld in wt% per manufacturers datasheet.

Material:	Type:	C	Mn	Si	Cr	Ni	V	Nb	Ti	S	P	Fe
X65	Base	0.16	1.65	0.45	-	-	0.07	0.05	0.04	0.010	0.020	
OK Aristorod 12.50	Block A	0.08	1.46	0.85	-	-	-	-		0.012	0.013	remainder
OK Aristorod 12.63	Block B, C	0.074	1.68	0.95	-	-	-	-	-	0.012	0.013	remainder
OK Aristorod 13.26	-	0.095	1.35	0.8	0.12	0.84	-	-	-	0.012	0.013	remainder

3.1.4. Temperature measurements

Temperatures were measured during welding of Block A by six K-type thermocouples. Once layer 24 was reached, the thermocouples were attached to layer 24 and temperature was measured starting from the welding of layer 25. Block A along with the thermocouple locations is shown in Figure 3-3, after completion of layer 25. The numbers added in the figure refer to the recording channels and will be used to indicate the location in Chapter 4, when the temperature results are provided.

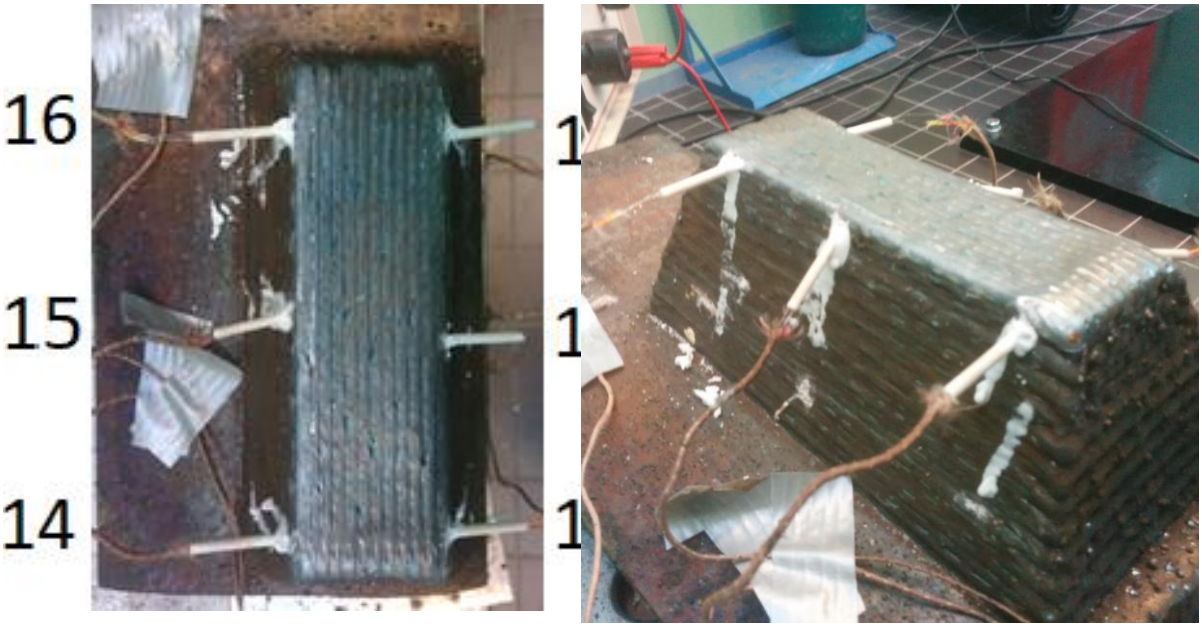


Figure 3-3: Diagram showing thermocouple locations and numbering for the pyramid block A, after deposition of layer 25 (total:29 layers)

Figure 3-4: Block B consisting of 10 layers, deposited in a back and forth deposition strategy. (shows only 8 here)

Figure 3-5: Block C consisting of 10 layers, deposited in a unidirectional deposition strategy.

3.1.5. Microstructure evaluation

Optical microscopy was used to study the effect of the thermal cycling on the microstructural evolution of the deposited steel. Samples were prepared according to standard procedures (grinding and polishing up to 3 μm) and etched with 2% Nital (2% nitric acid in ethanol). A Keyence 5000 digital microscope was used for microstructure pictures.

3.1.6. Mechanical testing

3.1.6.1. Hardness measurements

The constructs were cross sectioned to obtain the hardness distribution over the building direction. The block was sectioned at 50 mm from the starting position of welding, to ensure that stable welding condition were achieved. The hardness was measured with a Struers Duramin A300 Hardness Tester. The indents were made with a load of 10 kgf starting from the base plate towards the final layer with an inter-distance of 1 mm.

3.1.6.2. Tensile testing

From Block A tensile test specimens were extracted. Figure 4-6 shows a schematic of the block with tensile specimen numbering. First a band saw was used for rough cuts due to the size of the block and the handling difficulty. Once square blocks were cut, a lathe was used to produce round tensile test specimens. These specimens were produced according to ISO EN 6892-1:2016. Afterwards these were tested according to displacement controlled tensile testing with a displacement rate of 0.0343 mm/s.

3.2. Finite element model description

In this section, first the thermal model will be described, including heat input and heat removal. The welding process is simulated using the commercial Finite Element Analysis package ABAQUS [38]. As described in chapter 2, a finite element method approximates a real case by subdividing a geometry into elements and calculating the effects of the application of a heat source to the temperature of these discrete elements. All elements combined form a mesh. The type of elements used, the size of these elements, the shape of the mesh can all have influence on the results. Secondly, material deposition during welding, is incorporated in the model. From the combined model, the cooling rate and resulting hardness can be calculated. In summation, four models will be described:

1. A thermal model
2. A material addition model
3. A cooling model
4. A hardness model.

3.2.1. Thermal Model

Figure 3-6 shows the various heat flows in welding. At the top is the heat input due to the welding arc. The introduced heat has a Gaussian distribution with the peak at the location of the electric arc. This arc heat flows into the weld pool though convection and conduction towards the surrounding metal. After the heat source has passed, the hot molten pool will lose heat to the environment due to radiation and convection. The heat from the arc is used to melt both the added metal and the substrate, allowing them to mix and solidify.

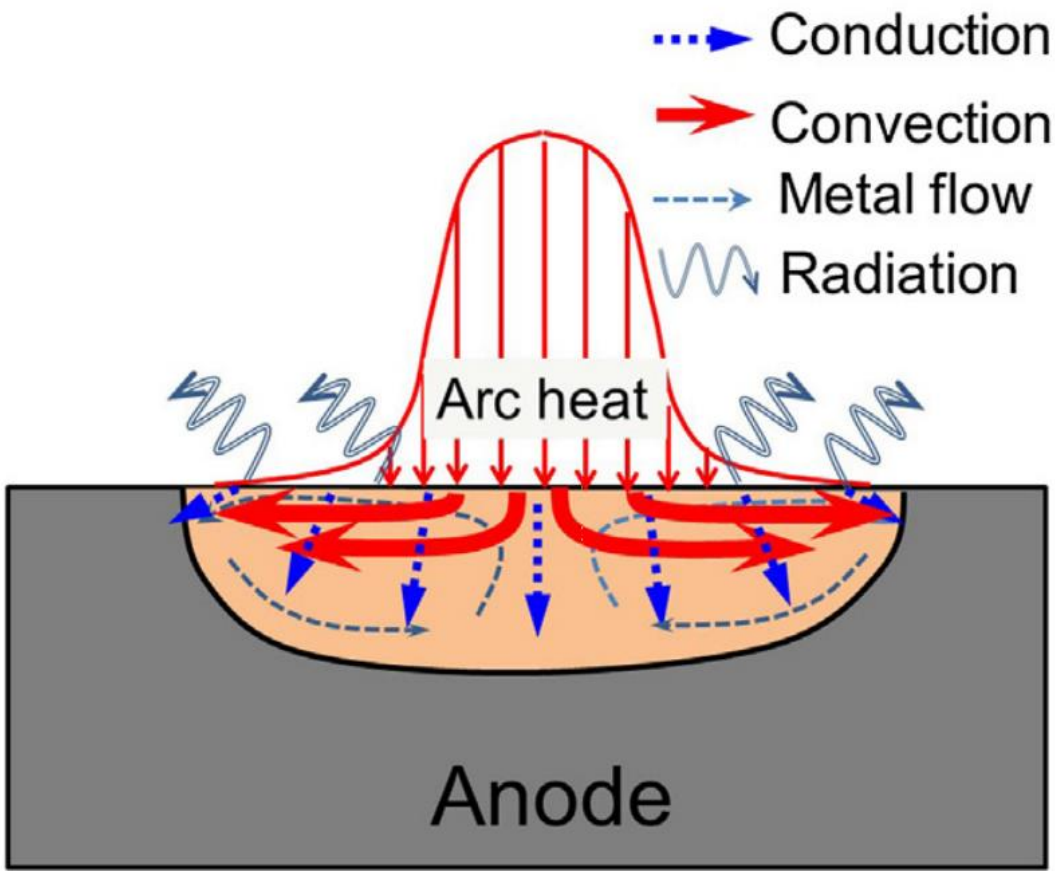


Figure 3-6: Heat flow in a weld. [39]

Modelling the various phenomena in the weld process through physics models is complicated and time-consuming, which is why alternative approaches are chosen A commonly used alternative to describe the heat introduced to the workpiece is the Goldak method. Rather than simulating the individual thermal effects, the energy added by the process is instead distributed according to a Goldak double ellipsoid thermal flux. This method has a number of parameters to describe the shape of the flux, allowing great control over the shape of the flux at the cost of a need for fitting these parameters. This double-ellipsoid flux then moves across the substrate to simulate welding [40].

A further simplification is to not consider the flux, but rather the temperature of the added material. In this approach, newly deposited material is added in a very hot state (above melting temperature), and the affected substrate is also heated to above this hot state through a prescribed thermal boundary condition [28]. As discussed in chapter 2.3, the Goldak method approximates the various thermal flows with a double-ellipsoid distribution of the energy flux; the prescribed temperature method shows a good agreement with this method outside the weld pool. This suggests that, despite its simplicity, the prescribed temperature method can produce useful results.

The prescribed temperature approach is applied in this study. To simulate the heat input of the arc, a temperature boundary condition is applied before the element insertion step. This boundary condition is applied for a certain amount of time to simulate the weld arc passing. The elements that represent the deposited weld are then re-activated in a short (10^-7) time step. This time step is short to minimize thermal transfer during the reactivation step. This results in a temperature field similar to the actual weld process.

3.2.2. Material Addition Model

The model also requires addition of material during welding. To capture this, elements are initially deactivated as the mesh is defined and reactivated during the weld simulation as needed. This is the so called element birth/death method as described in chapter 2.5.2. During meshing, the substrate volume and weld volume are marked. The substrate volume consists of elements that are active from the beginning of the weld. For the elements that are part of the weld, a deposition order is determined. These weld elements are marked as deactivated before the start of the simulation to remove them from thermal computations and re-activated as the weld process simulation continues [41] [42].

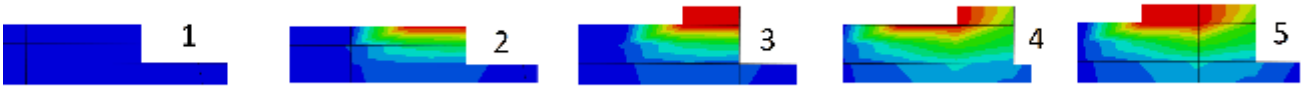


Figure 3-7: Element Birth-death method visualized. (1): Initial state, weld elements deactivated. (2): Boundary condition applied. (3): Element inserted. (4): Boundary condition applied. (5): Element Inserted. This process is then repeated.

Figure 3-7 shows how the element birth-death model works in more detail. Initially all the elements of the weld volume are deactivated. In step 2, a thermal boundary condition is applied to simulate the arc passing, i.e. the arc heat input. In step 3, which follows after the boundary condition has been applied for some time to simulate the passing of the arc, the addition of filler metal is simulated by adding an element at a prescribed temperature. This sequence is then repeated as shown in step 4, showing the application of the boundary condition, followed by step 5 showing a newly inserted element. Additional time steps without added elements or thermal boundary conditions can be applied to simulate the robot moving to a new position without welding to start a new welding sequence.

With MIG/MAG welding the deposition of material is linked to the wire feed speed and the input energy. A higher wire feed speed leads to more deposited material and more energy required to melt that material. The following balance should be established:

$$\left[\frac{Energy}{Volume}\right]_{Filler} = \left[\frac{Energy}{Volume}\right]_{Weld\ bead} \qquad Eq.(4)$$

The energy per unit volume is then

$$\left[\frac{Energy}{Volume}\right]_{Filler} = \frac{P_{weld}}{A_{wire} * WFS} = \frac{\rho * C_p * T_{ele}}{V_{ele}} + \frac{E_{BC}}{V_{BC}} \qquad Eq.(5)$$

where P_{weld} is the power deposited by the welding process, A_{wire} is the area of the wire cross-section, WFS is wire feed speed, V_{ele} is the cross-section of the modelled weld, and E_{BC} and V_{BC} are the energy from the boundary condition and the affected volume, respectively. The model allows flow of heat during the application of the boundary condition, this implicates that the energy added is dependent on the duration of this step. The boundary condition is linearly ramped up over the time step, both to simulate heating from the arc as well as allowing the solution algorithm to converge faster. The FEM

implementation determines its time increment dynamically based on the remaining error of the thermal calculations. Large instantaneous thermal differences cause large initial remaining errors, which result in long calculation times. A more gradual ramp-up of the boundary condition ensures that the solution converges more quickly.

In addition, Figure 3-8 shows that the thermal boundary condition ensures a correct behaviour of the model. Because the thermal calculations are applied to the nodes and not the elements, the thermal transition between the weld and substrate exists purely as a linear interpolation within the element. Upon deactivation of the weld nodes, the model applies the prescribed substrate temperature to the nodes within the weld/substate boundary. Upon element reactivation, the weld has a much lower temperature than intended. By adding the boundary condition step, this effect is mitigated.

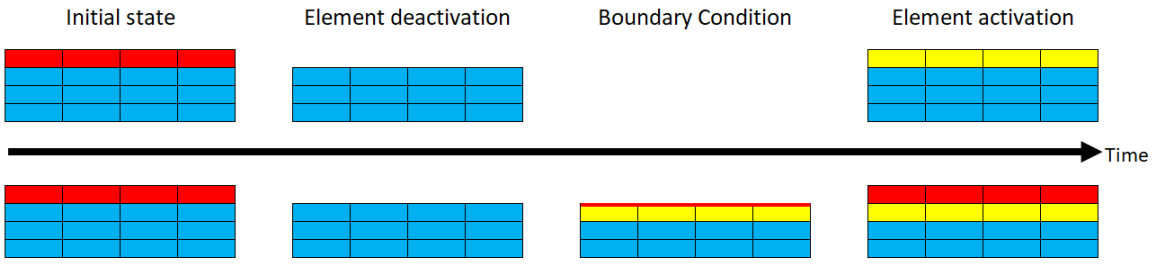


Figure 3-8: Influence of boundary condition on the temperature field. Starting left, an element layer is designated as high temperature (red) and the other elements low (blue). At the start the weld, elements are deactivated, meaning that the nodes shared by both the " hot" and " cold" elements are now only "cold". When the element is activated again, the modelled temperature is much lower than it was intended to be. By adding a boundary condition step, the shape of the thermal field is corrected.

The energy added by the boundary condition consists out of two components: a) the energy needed to raise the temperature to the boundary condition temperature, and b) the heat that is allowed to flow during the step.

The other material properties, i.e. thermal conductivity and specific heat are given as a function of time in Figure 3-9, calculated using JMAT Pro [43]. The thermal conductivity is artificially increased above the melting point to simulate the convection in the weld pool [44]

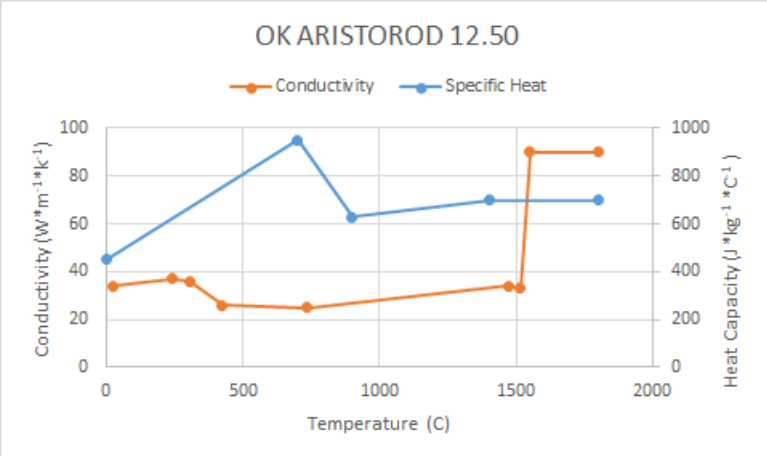


Figure 3-9: Thermal properties of the used filler material [37].

3.2.3. Cooling rate and Hardness modelling

Rather than predicting a highly precise local hardness, the interest is in describing the hardness distribution in an entire welded block, in a way that is fairly fast and easy to model. Welds are taken as a single entity with an average hardness, instead of a complicated distribution in weld metal and heat-affected zones.

JMat Pro is used to calculate a CCT diagram for hardness determination, as well as for the temperature-dependent material properties. The parameter used to calculate hardness is the time spent between 800 °C and 500 °C, which is commonly used as an indicator of hardness [45]. By modelling the thermal field and then calculating for each node how long it spends between 800 °C and 500 °C, a hardness can be predicted. To link calculated times and the CCT diagram, the T800-500 time is divided by the temperature difference to obtain a T800-500 cooling rate.

Figure 3-10 shows the temperature history of a node, along with indicators added by the hardness algorithm. The thermal history of a node is loaded, then checks for peaks ('x' in figure 3-11, T>800 °C) and valleys (diamonds, T < 500 °C). Note that segment three (Figure 3-10, red triangle labelled 3) has its peak below 800 °C and thus is ignored. With time and temperature at the high and low points, the cooling rate *dT/dt* can be simply calculated by:

$$dT/dt = \left(T_{high} - T_{low}\right)/\left(t_{low} - t_{high}\right) \qquad Eq.(6)$$

This cooling rate is calculated for each segment for each node. For each node, the model then determines the last peak and compares it with the CCT diagram to obtain the hardness.

The model assumes that when the temperature reaches aforementioned 800 °C, the previous temperature history is ignored and thus the material is essentially considered to be fully re-austenitized. Since the hardness is estimated solely by considering the CCT diagram, the effect of tempering is not taken into account.

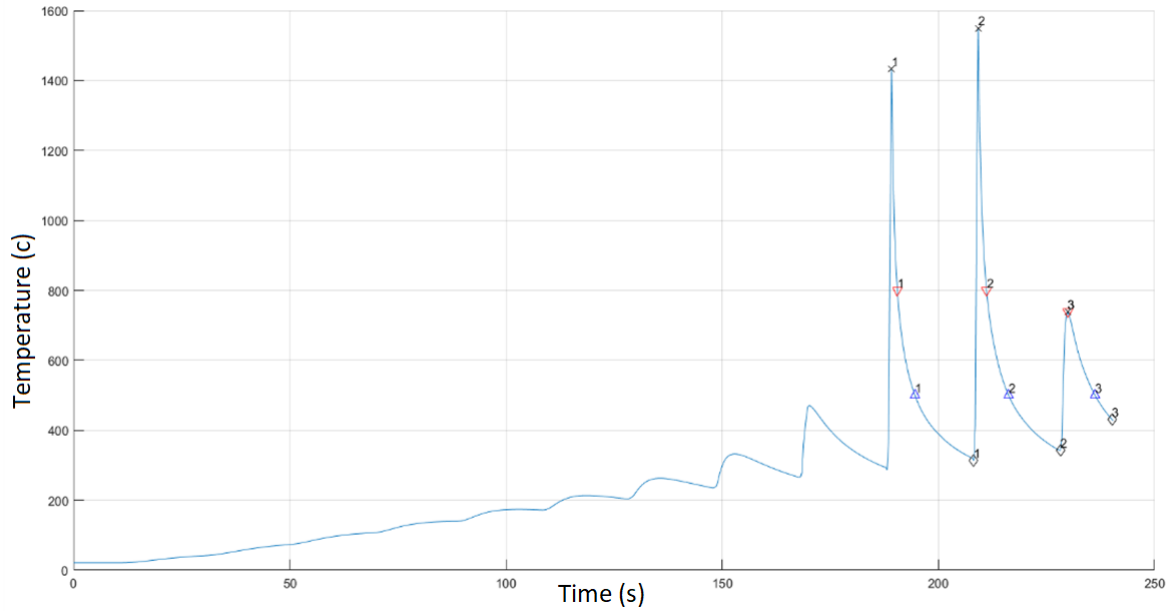


Figure 3-10: Temperature history of a node. Black cross: Start of the segment. Black diamond: end of the segment. Red triangle: T800 or equivalent. Blue triangle: T500 or equivalent.4

JMAT PRO was used to calculate a CCT diagram for the S355 filler material, shown in Figure 3-11. The bold values at the bottom of the cooling lines are the expected hardness in Vickers. Note that after cool-to-room temperature times of about 100 s, the hardness stabilizes to 157 HV. By matching cooling lines from the CCT diagram to cooling rates calculated in the transient model, a hardness distribution can be calculated. A CCT diagram is sensitive to prior austenite grain size, as these

austenite grain boundaries serve as nucleation sites for phase transformations, and thus peak temperature [46]. The hardness estimation model could be expanded to include grain-size effects, but this was not considered in the scope of this thesis.

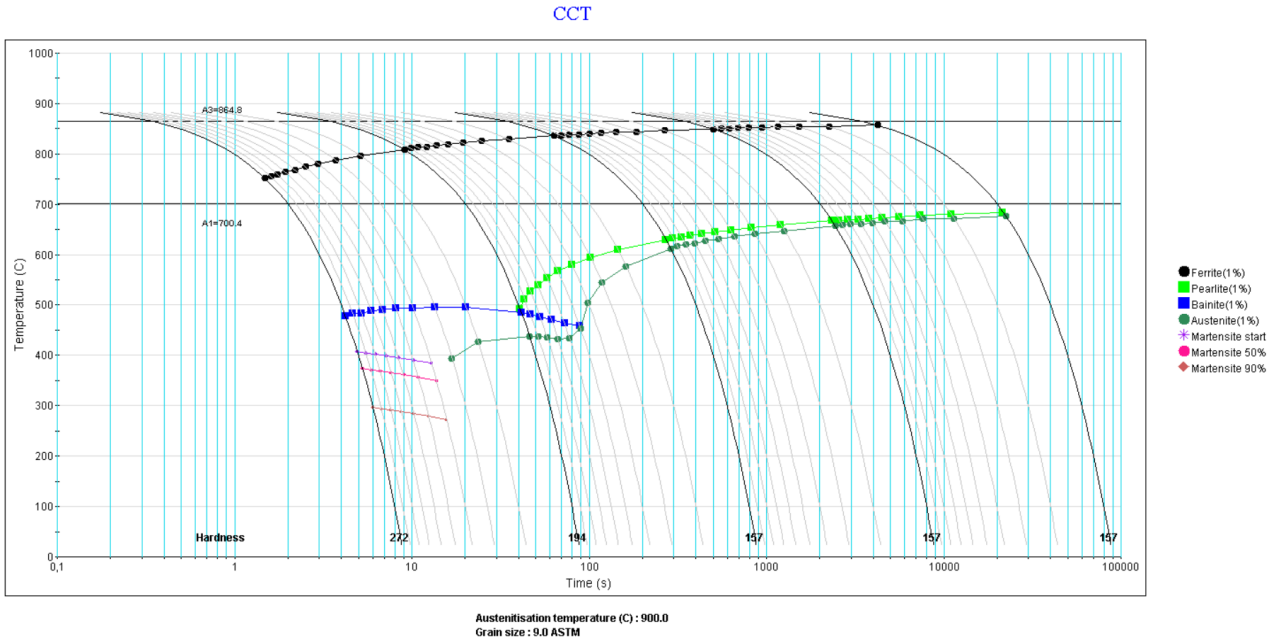


Figure 3-11: JMAT PRO calculated CCT diagram of OK ARISTOROD 12.50, the filler material used for S355 equivalent.

3.2.4. Meshing

For meshing, it is necessary to divide the welded segment into elements with similar dimensions to ensure consistent calculation results. The thermal model activates elements to simulate the addition of material, and since the welding is carried out in a controlled and consistent manner, the mesh in the weld region needs to be consistent too. Thus, the mesh size was set equal to the average weld width and average weld height. In the deposition direction the mesh size selected was equal to the weld width. To simulate a deposited block, a designed geometry is sliced into layers and each layer is converted into a tool path for the torch to follow. Therefore, a difference between the designed geometry and the actual geometry will result (see Figure 3-12). This is called the stair-stepping effect. To avoid a too complicated mesh, the shape of the deposited blocks is simply considered smooth "as-designed" rather than rough "as-deposited".

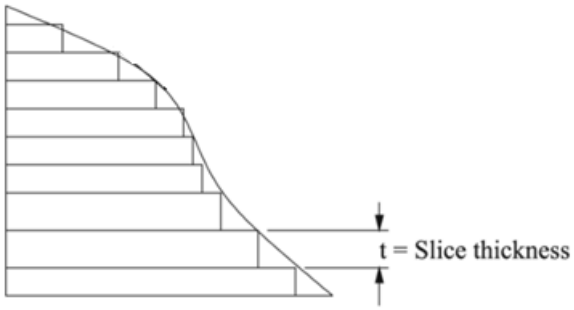


Figure 3-12: A schematic representation of the stair-stepping effect [6]

For the thermal model validation, thermocouples were welded onto layer 24 to measure the deposition thermal cycle of layer 25. To reflect this in the model, the mesh size in layer 24 was set to a height of 1mm to ensure that the actual location of the thermocouples could be accurately extracted from the model. The smooth-wall model and the stair-step-shaped experiment will have a small difference in the exact thermocouple location.

3.3. Model Sensitivity

3.3.1. Material properties

The hardness is calculated based on CCT diagrams. Figure 3-13 shows the relation between cooling rate and hardness for the three filler wires of interest, along with the hardness calculated by equation 7.

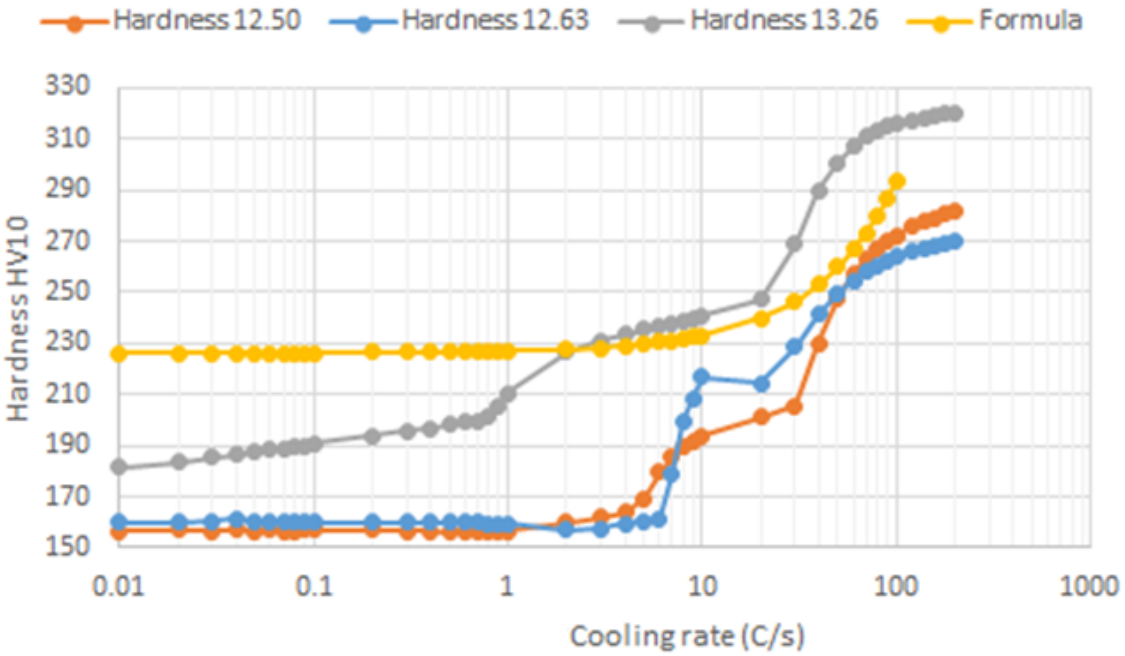


Figure 3-13: Cooling rate versus hardness, comparing the sensitivity of the three fillers to a hardness calculated with Equation 7 [47].

As expected, higher cooling rates lead to a higher hardness. For the 12.50 wire and 12.63 wire, below a certain cooling rate the hardness reaches a plateau, indicating that these cooling rates will not result in non-equilibrium phases. The 13.26 wire shows a different behaviour. Even at low cooling rates the hardness increases. Based on the chemical composition of this materials, phase fields in the CCT diagram shift to lower times, which makes this material more sensitive to hardening.

Lastly, equations (Eq.7 and 8) to obtain the hardness for the intercritical zones are shown below [48]. These can be used to calculate the maximum hardness in a multi-pass weld. The equation (for HV) is based on the chemical composition Pcm* with *CR* the cooling rate. For wire OK Aristorod 12.50 this will result in equation (9).

$$Pcm^* = C + \frac{Si}{30} + \frac{Mn + Cu + Cr}{20} + \frac{12 * Ni}{60} + \frac{Mo}{15} + \frac{V}{10} + \frac{Nb}{5} + 5B \qquad Eq.(7)$$

$$HV = (11.5 * Pcm^* - 1.57) * CR + 289 * Pcm^* + 170 \qquad Eq.(8)$$

$$HV_{12.50} = 0.6725 * CR + 226.355 \qquad Eq.(9)$$

The equation is a linear function which does not capture the Bainite formation at 10 °C/s. However, in cases where the behaviour of the material is already well-established with known formula, these formula can also be used.

3.3.2. Sensitivity check of prescribed temperature boundary conditions.

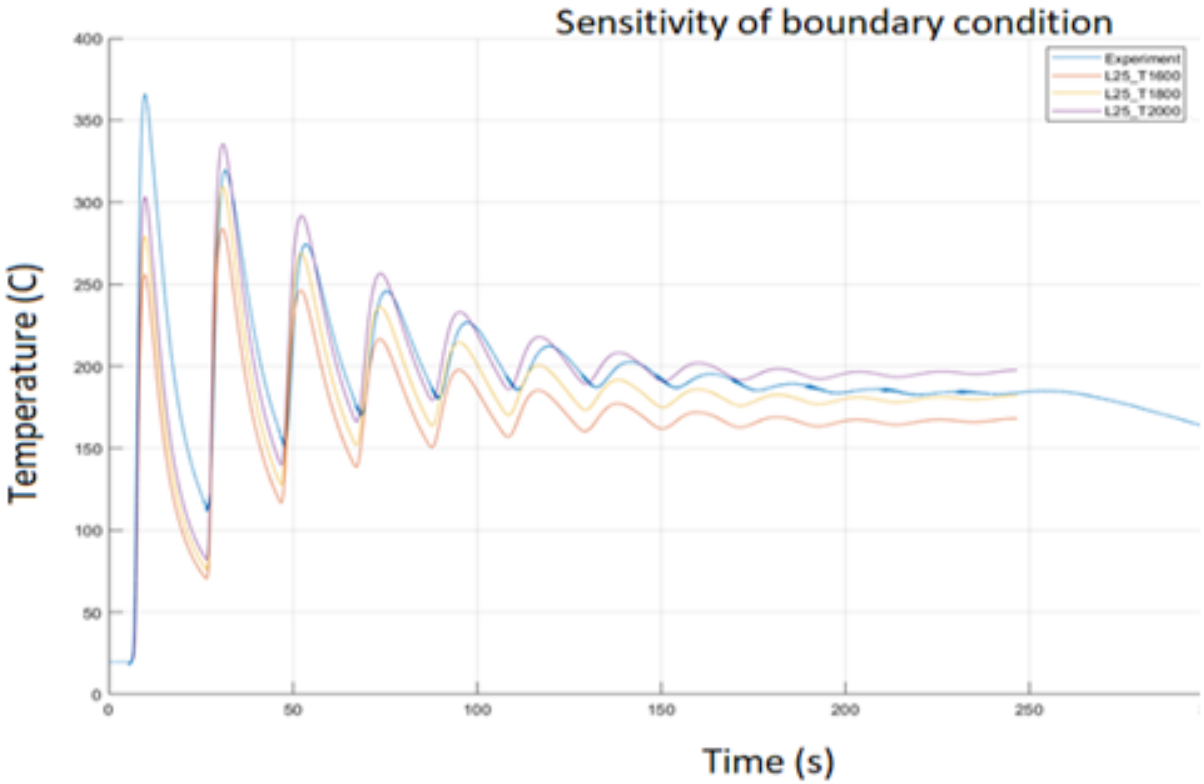


Figure 3-14: Temperature versus time for different prescribed temperature boundary conditions to analyse the sensitivity regarding the selected boundary conditions, comparing an experimental (blue) temperature history to three modelling results. The orange line represents a prescribed temperature of 1600°C, yellow 1800°C and purple 2000°C.

Figure 3-15 shows the experimentally obtained thermal history of the 24th layer when layer 25 is deposited together with the numerical results for the three deposition temperatures. It appears that a deposition temperature of 1800 °C matches the experimental values best. The reason for the choice of 1800 °C as the boundary condition temperature, where an experimental value (blue) is compared to three model values. The boundary condition temperatures are 1600 °C (red), 1800 °C (yellow) and 2000 °C (purple). As can be seen in the figure, the 1800 °C imposed field works best in this case.

3.3.3. Deposited length

FEM divides a continuous domain into discrete elements in order to solve equations. The finer the discretization, the more closely the results will match the continuous domain. The smaller the discretization, the longer the computing time. A balance has to be sought between fast and accurate. For this, we have to consider the deposited length.

However, as the mesh size also to some extend dictates the volume of deposited material, this thus raises the question what the optimum volume of deposited material should be. The standard mesh had an equally long and wide size, resembling the actual width of the deposited material. The sensitivity to deposited volume was studied by using deposition characteristics of block A, while the deposited length was changed. Simulations were run in which the deposited length size was twice and triple the width of the deposited length. Figure 3-16 shows the standard mesh strategy and Table 3-1 shows parameters to evaluate the effect of deposition length, also indicating the time steps.

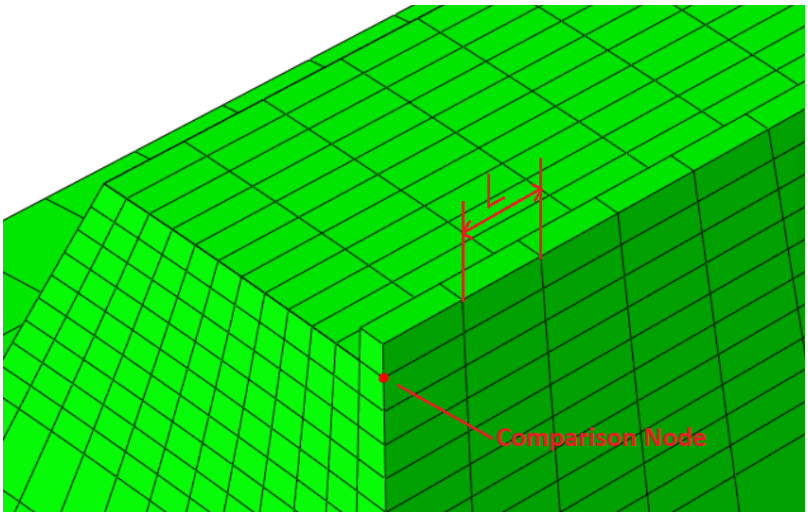


Figure 3-15: The width and height of the nodes is unchanged, but the length and thus deposition is made bigger. Also indicated is the node for which temperatures are compared.

Table 3: Dimensions and timestep of the deposition for each mesh.

	Normal (L=W)	Coarser (L=2W)	Coarsest (L=3W)
Element Height	2.8 mm	2.8 mm	2.8 mm
Element Width	3.2 mm	3.2 mm	3.2 mm
Element Length	3.2 mm	6.4 mm	9.6 mm
Time step	0.32 s	0.64 s	0.96 s

Figure 3-17 shows the node temperature profile for the three meshes, the node location is indicated in figure 3-16.
By looking at the temperature history of the highlighted node in the corner for each of the three models, and calculating the T800/500 time, we can see whether a larger deposition length influences on the calculated results. This is shown in Figure 3-16.

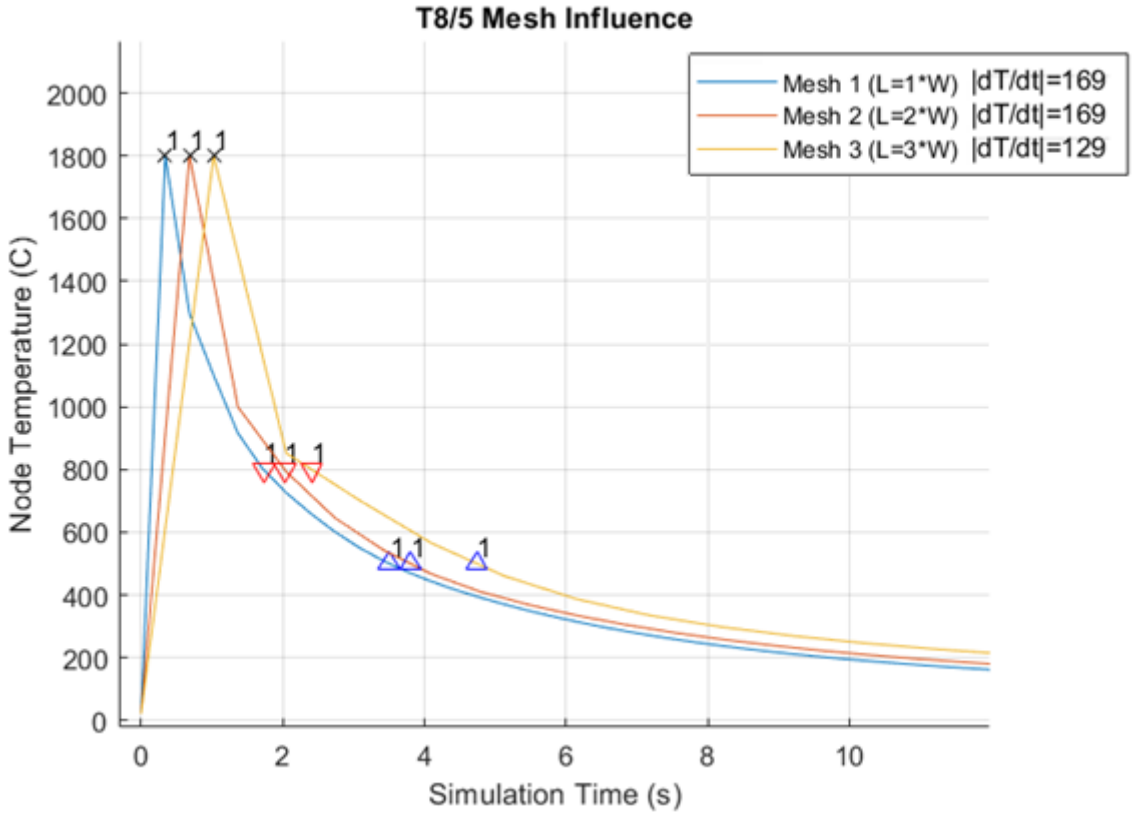


*Figure 3-16: Node temperature in °C versus simulation time. Note that for a coarser meshes, the heating-up period is extended resulting in peak shifts. The large linear cooling segment between 1800 and around 900 °C is due to the output time steps. For the two finest setting no notable cooling rate difference exists, but for the 3*W deposition length, the cooling rate is underestimated.*

To limit the size of the output results, during the simulation of the weld the output is saved at the end of each step. At 0.32s per step for the finest setting, this clearly creates no issues due to this being well out of the critical temperature range. **Fout! Verwijzingsbron niet gevonden.** also shows this is not an issue for nodes further away. As the elements become bigger, this does become more of an issue. The cooling rate becomes lower at coarser meshes because a larger area is kept hot at the same time, making it harder for heat to move away.

3.3.1. Sensitivity of thermocouple location

As mentioned before, the precise location of the thermocouple in relation to the weld could have significant effects on the measured temperature. In order to get a sense of this sensitivity, for thermocouple 11 the distance to the weld was varied by 1 mm. Figure 3-17 shows the calculated results. At 0 mm from the weld, the node is part of the molten metal and the maximum temperature is 1800 °C, i.e. the temperature prescribed by the boundary condition. At a distance of 1 mm from the fusion line, a peak temperature of about 1500 °C is calculated. At 2mm from the fusion line the temperature drops to 900 °C, at 3mm this is 600 °C, at 4mm the temperature reaches 460 °C and at 5mm this is 400°C. Finally, at 6 mm the maximum temperature is approximately 380 °C. The location of the thermocouple was measured to be around 5 mm from the top of layer 24. Taking the difference between the 4 and 5 millimeter result, the thermal gradient is 60 degrees over 1 millimeter. For the 5 and 6 millimeter result, the thermal gradient is 20 degrees over 1 millimeter. This results in an average thermal gradient of around 40 °C mm^{-1} at the 5mm location and thus the thermocouple location. The size of the weld bead made by spot-welding the thermocouple was around 0.5 mm, and the distance of the thermocouple was determined from the top of the layer to the bead middle. This suggests an average uncertainty in the thermocouple location of around 0.5 mm, and therefore a temperature uncertainty due to thermocouple location of 20 °C.

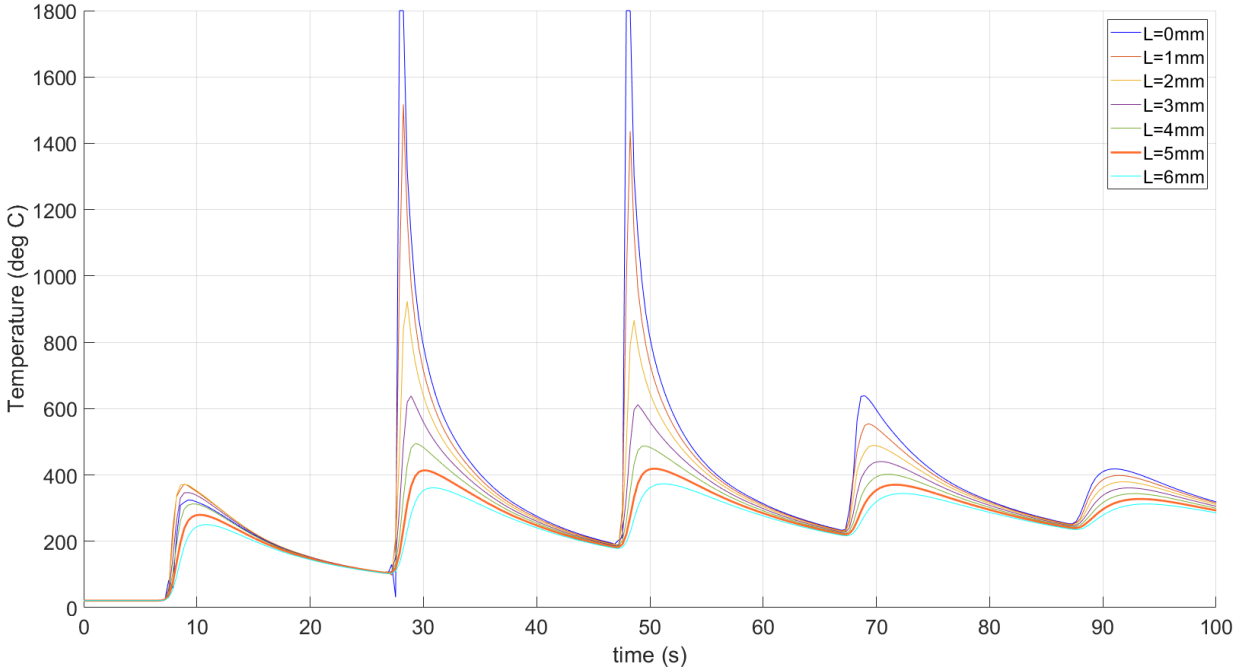


Figure 3-17: Modelled temperature at the location of Thermocouple 11 for various distances from the weld. 0 mm means the edge of the weld (fusion line). The actual location of the thermocouple is 5 mm from the fusion line.

4. RESULTS AND DISCUSSION

This chapter will show the results of the evaluation of the 3D-printed blocks as described in CH3, as well as the modelling results.
The pyramid block A is used for model validation. In this respect the thermal measurements and the obtained hardness values are used. Based on these results prediction on hardness distribution are obtained for the blocks B and C.

4.1. Block A: temperature measurements

Figure 4-1 shows the thermal measurements and welding current measurement during the welding of the Block A, as was shown in Figure 4-1. The thermocouples are situated at both sides of the top part of the sloped block at layer 24. The measurements are made during the deposition of the 26th layer as described in Chapter 3. Twelve welds are made in the same direction. The total weld time is roughly 240 s. The time between welds is 5 s, which is the time the torch needs to move from the end of a beat to the start of the next bead. The measured current data is used to ensure the robot current readout is correct, but also to time the welding simulation correctly.

Thermocouples 11, 12 and 13 are situated on the starting side of the layer. At the start of the new layer the block is at room temperature (20 °C). The current data is used to indicate the timing of the weld. As the torch approaches the thermocouple, noise becomes visible in the signal. The noise is however limited to lower-temperature regions and only affects the signal by a few degrees. The thermocouples CH11, 12 and 13 indicate an increase in temperature when the arc approaches and a slower temperature decrease when the arc has passed. The delay in reaching the peak temperature is obviously related to position of the thermocouple along the weld. The maximum peak temperature slightly varies for these three locations, related to the exact positioning of the thermocouple.

As the deposition of a layer progresses, the peak temperature measured by these thermocouples gradually decreases until they reach a constant value around 200 °C. The torch is closest to the thermocouples on the first bead and gradually becomes more distant when the layer reaches the opposite side of the block, decreasing the temperatures measured. Thermocouple 12 shows consistently the highest temperature of the three. One potential reason is that due to geometrical effects, the middle of the weld cools slower than the sides. Another reason is a potential mismatch in thermocouple location, meaning that the thermocouple was closer to the weld than the other two.

Thermocouples 14, 15 and 16 are situated at the opposite edge of the block, far from the start of the first deposit bead of a layer. For the first several beads, the temperature increases gradually as the heat of the welding process conducts evenly throughout the block. The thermocouple closest to the start measures the lowest temperature, followed by the middle thermocouple, while the highest peak temperatures are measured by thermocouple 16. For the last seven beads, peaks and valleys become more pronounced with differences of roughly 5 to 10 °C.

After approximately 240 s the layer is completed. The heat from the last beads flows into the block and the temperature registered by all thermocouples reach around 200°C at 300 s. From this point in time the block cools homogeneously.

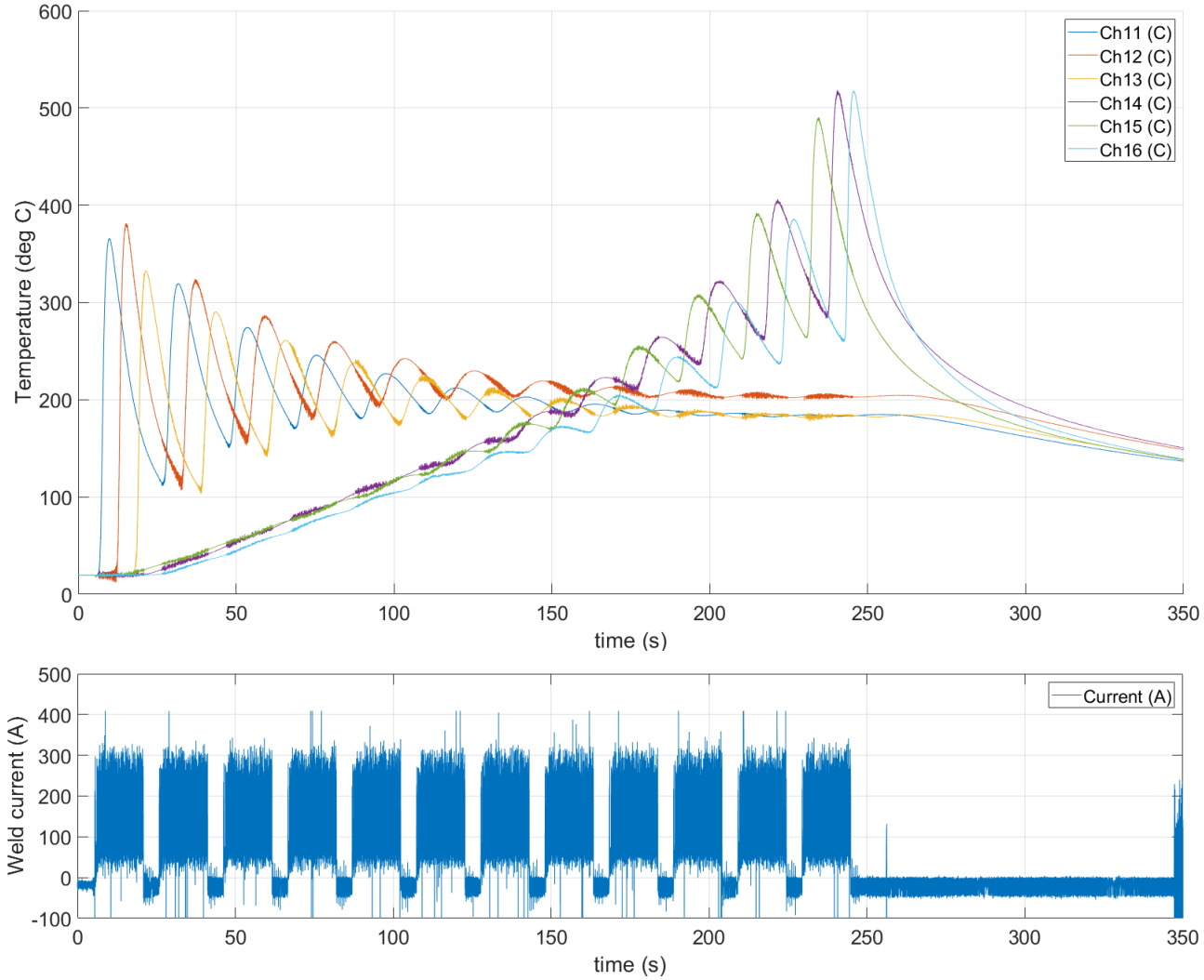


Figure 4-1: Temperature (top) and current (bottom) measured for block A, during deposition of layer 25. Channel designations according to Figure 3-3.

More information on the thermal cycles was also obtained for the deposition of layer 26, plotted in Figure 4-2. In this layer after depositing three welds according to the printing strategy, arc start failure occurred at the 4th weld bead. The reason for this failure is the surface condition of the workpiece or the filler wire condition, as a results of for instance spatter formation. After local cleaning, welding was restarted. Start failure was avoided in further deposits by the application of anti-spatter spray.

Note that in layer 26 the deposition direction is reversed. While the different thermocouples are named and coloured the same, the order of the activation of the thermocouples is also reversed. Thermocouple 13 is now situated closest to the start position of the deposition process. The same differences in temperature are still seen as was in layer 25. During the pause after weld 3 at approximately 60 s, a similar cooling profile as after deposition of layer 25 can be seen. A more or less homogeneous temperature is reached. After restarting the printing process, the first beads on the starting side of the block showed a slowly increasing peak temperature to around 230°C, which remains constant when the deposition process moves towards the opposite side of the block. At completion of layer 26 the process is closer to thermocouples 14-16 which show the typical thermal cycling behaviour. Peak temperatures are slightly lower compared to the deposition of layer 25 as the distance from the deposit is increased by the height of one layer.

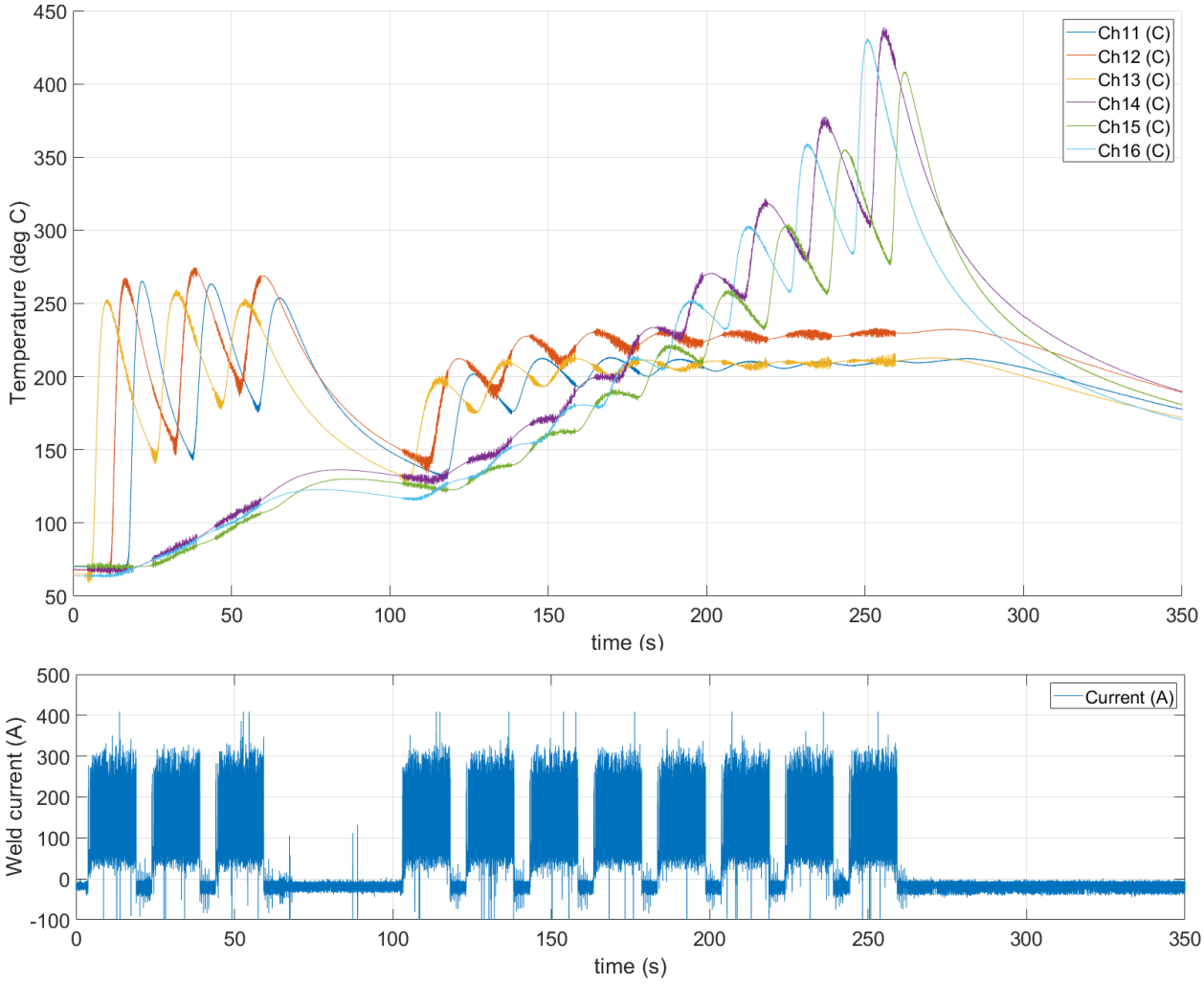


Figure 4-2: Temperature (top) and current (bottom) of Block A, during deposition of layer 26. Note the gap after the third weld, which occurred due to arc start failure.

4.2. Block A: Validation of the thermal model

In chapter 3.2 the thermal model is described including all boundary conditions and assumptions. The thermal data described in section 4.1 is used to validate the constructed model. Figure 4-3 shows the comparison between the experimentally measured temperatures in block A (at the points shown in Figure 3-3) and the results from the numerical simulation. A first impression reveals that there is a relatively good agreement, however at certain points the simulations deviate from the experimental results.

The following general trend can be seen. First the starting side is discussed, with thermocouples CH11-13 located on the 24th bead. The simulated temperature at the thermocouple locations, during deposition of the first bead of layer 25, is lower than the measured temperature. For the second thermal cycle, during the deposition of the second bead, the simulated and measured peak temperatures are very similar, while for the subsequent deposits the modelled temperature peaks are all slightly higher than the experimentally obtained data by 20-30 °C.

At the other side of the block , which contains thermocouples CH14-16, a similar situation can be observed. The measure temperature of the last bead is higher than simulated. Also for the thermocouples CH14-16, it is observed that the positions of peaks and valleys agree very well for experiment and simulation. The overall calculated temperature is approximately 40 °C higher compared to the measurements.

The other peaks and valleys appear to match well. The higher calculated temperatures at the end of the layer welding suggests that either too much energy is added to the block or the assumptions about cooling are too low. However, the temperature slopes at the end of the weld show a similar thermal loss compared to the experiment, suggesting a correct value for the thermal losses via radiation, convention and conduction.

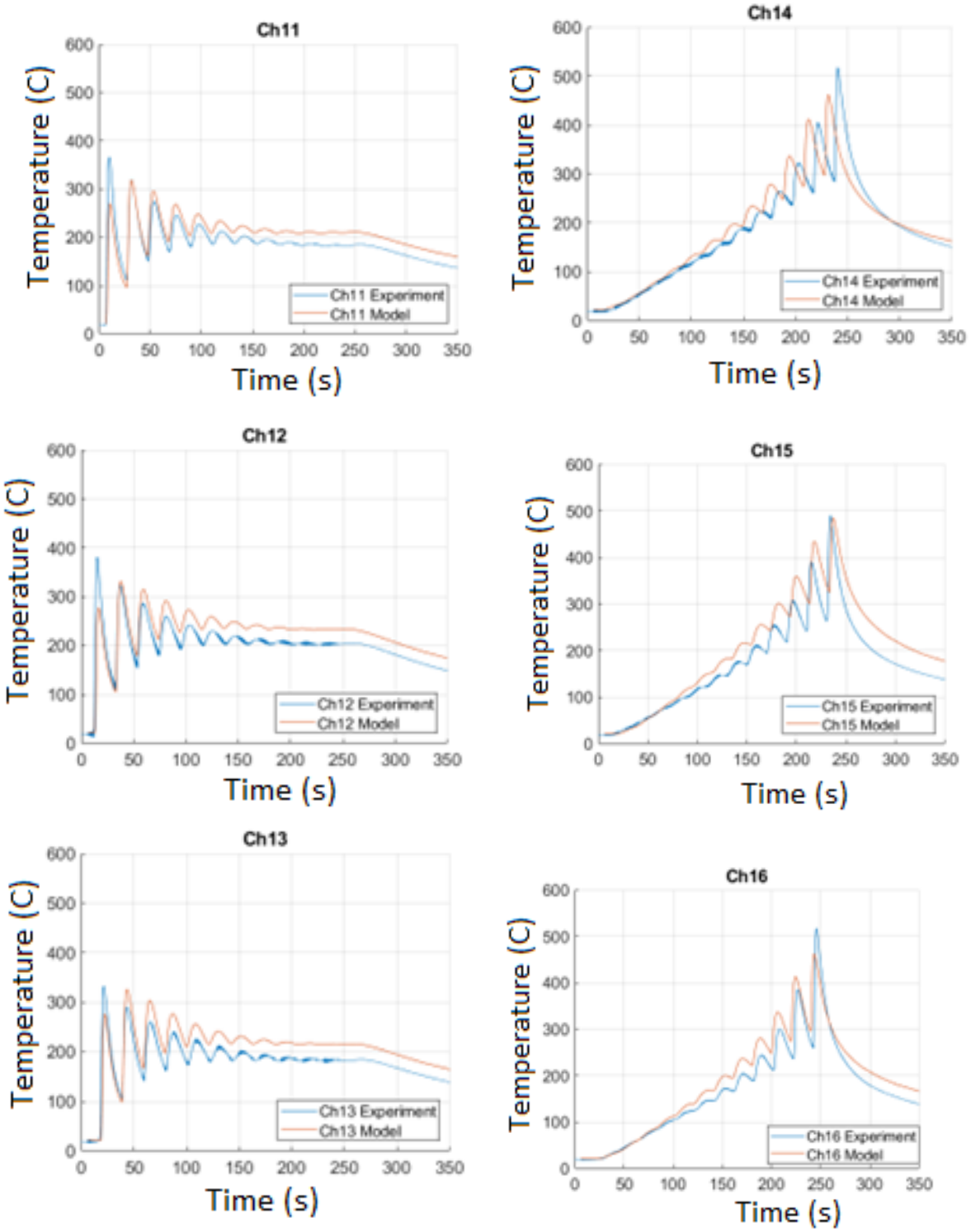


Figure 4-3: Measured temperature (blue) and simulated temperature (orange) as a function of time for each thermocouple location (see figure 3.3).

Several factors could be the reason for the mismatch between simulation and experimental results:

1. Heat accumulation of the block,
2. Heat input mismatch due to geometry,
3. Non-constant robot motion and post-weld gas,
4. Plasma and shielding gas flow,
5. Material property mismatch.
6. Thermocouple location mismatch

These factors are addressed below:

1. Heat accumulation

The block experiences a heat input during the deposition of each bead. During the entire production process of the block, heat is lost via radiation, convection and conduction. In the production stage the heat input is in excess of the heat losses. The block will gradually increase in temperature. The temperature in the block is not homogeneous. At location where materials are deposited, i.e. in the neighbourhood of the heat source, the temperature increases. Further away from the heat source heat input and heat losses are coming into balance and the temperature becomes relatively constant until the deposition process is terminated and the block cools homogeneously. This trend is observed in both simulations and experiments.

The deposition of the first bead takes place when the workpiece is at room temperature, for each subsequent bead the temperature is well above room temperature. As can be seen in Figure 4-3, the second bead starts at 100 °C. For the first bead deposited it appears that the actual peak temperature is higher than the simulated result. The model has a constant temperature of the deposited material, which implements that energy is dissipated to heat up the block, i.e. a certain fraction of the simulated heat input diffuses to the location of the thermocouple, resulting in a lower peak temperature. In addition, in reality the melt pool is highly superheated, much hotter than the 1800 °C assumed as deposition temperature in the model. This means that the model probably underestimates the increase in temperature. However, at the other end of the block, for thermocouples 14-16, the last bead model results also show a lower peak temperature. This rules out heat accumulation as a viable explanation.

2. Geometrical features

Another factor that may explain the differences in the experimental and calculated results for the peak temperature of the first deposited bead is related to geometrical features of the block.
The thermal model approximates each weld as being equally wide and high. This is not entirely in accordance with reality, since the first weld is deposited on a flat surface, while subsequent welds are made next to a prior weld, see Figure 4-4. This means that the first bead in reality is wider than subsequent beads, and is also slightly lower in height. Because the energy input in the boundary condition model is dependent on the area the boundary condition is applied to, an underestimation of the first bead's connecting area means that the heat input is also lower than reality. This means that the first bead would have a lower heat input and thus lower measured peak than subsequent welds. This would match the observed phenomenon. Furthermore, the first weld is deposited close to the edge where the heat losses are both due to the thermal conduction within the block and radiation and convection from the free surface. Within the model the assumptions of thermal heat losses to the environment could be too high. During the deposition of subsequent passes the simulation predicts a higher temperature.

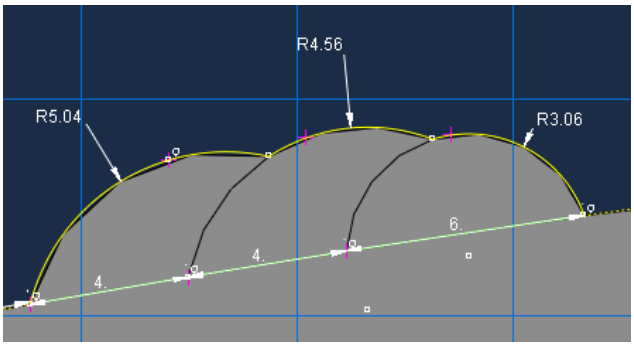


Figure 4-4: Schematic of three adjacent welds. The first weld (right) is deposited on the flat plate, and the following two beads are deposited in a manner that the space between beads is filled and the surface remains relatively even.

3. Non-constant robot motion and post-weld gas,

The third possible explanation is related to processing condition during deposition. It was mentioned that the model assumes a uniform, constant robot motion. In the actual experimental conditions, however, the robot arm must accelerate from zero to the constant speed at the start of the process. At the end of the weld, the robot arm has to decelerate from constant speed to zero. In addition, at the end of the weld, the metal is still extremely hot (>800 °C and glowing mild-orange) and post-weld shielding gas is applied to protect the hot metal from oxidation. Both factors combined means that at the start and end of the weld, a small region would be colder than expected. The robot is designed for far higher speeds than needed during welding, up to 2m/s, meaning that robot acceleration at the low weld speeds is negligible. For the majority of the bead deposition, the robot arm is moving at a constant velocity with a stable arc and gas flow. If the robot acceleration has a major impact, it would mean that there would be significant differences between the start and end sections of the weld as compared to the middle segment. This was not observed during welding. Therefore it is expected that start and stop condition will not have a significant effect on the discrepancy between simulation and experiment.

4. Plasma and shielding gas flow

Thermocouples are more exposed to radiation and convection of the heat source during the deposition of the first deposit of a layer. The thermocouples are not shielded from the presence of the arc by subsequent beads. Hot plasma and shielding gas moves past the thermocouple due to the welding position being near the edge of the block. However, as this was observed during welding, it should be kept in mind that the passing of the heat source only lasted for a few seconds. In addition, the thermocouples were encased in ceramic thermal protection sleeves to minimise the direct influence on the measurement. It remains a possibility that plasma and shielding gas flow had an effect on the results.

5. Material properties

This difference in heat build-up also reflects in the material properties. As discussed in 3.3.1, the material properties in the model are temperature-dependent. Ferrite has a higher thermal conductivity than cementite or bainite [49], so local differences in composition could lead to locally higher or lower thermal conductivity. However, while the austenite starts to transform at 800 °C, the effects of these different phases on the thermal field do not become significant until below around 500 °C. Since the entire measurement is in this region, these local differences should have effects for every bead and not just the first.

6. Thermocouple location mismatch

As mentioned in 3.3.1, there is some uncertainty in the location of the thermocouple that could influence the result. The thermocouples were manually spotwelded to the beads of layer 24. As the

geometry of the bead is simplified in the thermal model, and slight placement errors could occur during spotwelding, the thermocouples may not be precisely in the location assumed in the model. The thermocouple locations were measured using a digital calliper, making misreading these results unlikely. The thermocouple locations do have uncertainty in location, as the bead is finite in size in the experiment yet an infinitesimal point in the model. This can only account for some of the deviation as this was previously estimated at 20 °C difference. Although the bead geometry does not precisely match between experiment and model, the distance between fusion line and thermocouple is across one bead, and thus a mismatch in the order of millimeters would have been measured. While thermocouple location mismatch cannot be ruled out, it is reasonable to suggest it cannot fully explain a 40 °C mismatch.

Taken together, the likely factors to explain the 40 °C mismatch are a combination of:

- geometrical simplification in the model leading to a colder initial weld and a slight mismatch in thermocouple location
- potential plasma and gas flow leading to additional thermal input at the thermocouple location, which is not accounted for

4.3. Block A: Validation of Cooling rate

Following the thermal model is a cooling rate model. To validate the results, cooling rates are calculated from the thermocouple results of the previous section.

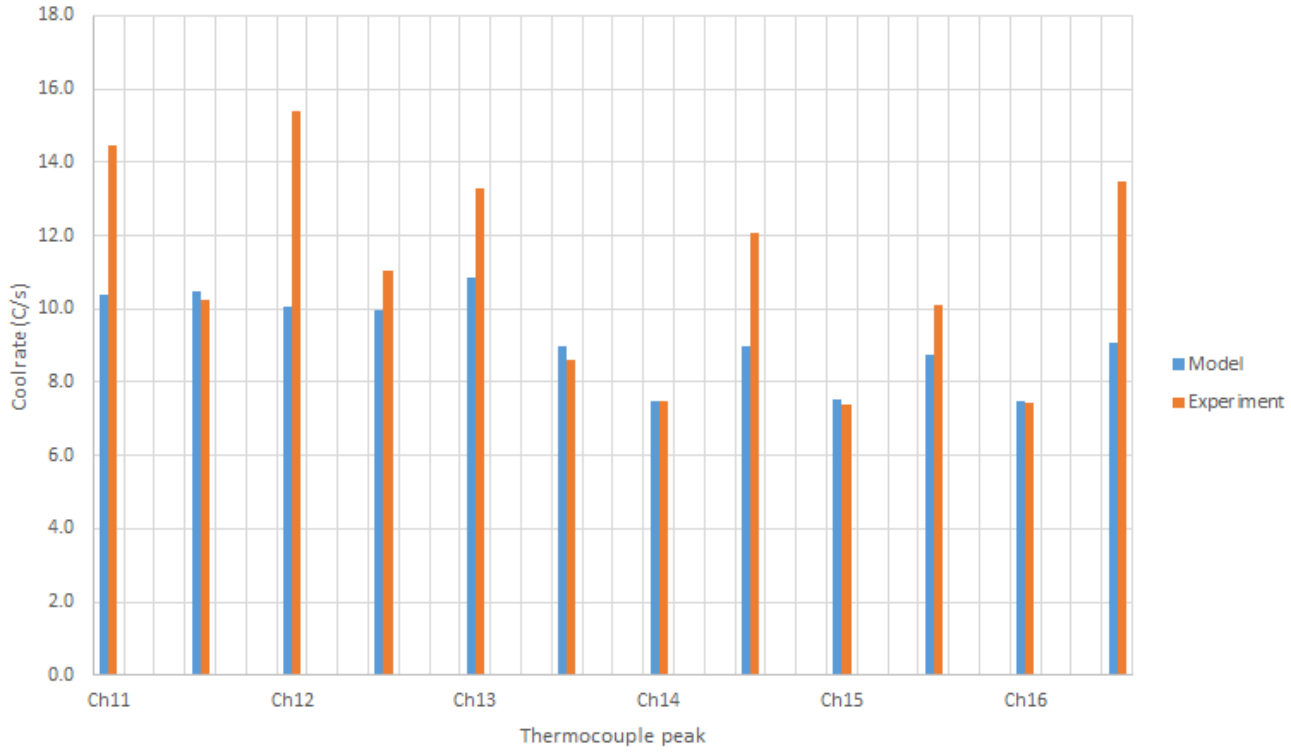


Figure 4-5: Measured and calculated cooling rate comparisons for all thermocouple locations for the closest two beads. For Ch11-12-13 these are the first two beads. For Ch14-15-16 these are the last two beads.

Figure 4-5 shows the calculated cooling rates, together with the experimentally obtained cooling rate, at the location of the thermocouples. The cooling rate for the experiment is calculated by taking the temperature difference between the local minimum and maximum divided by the difference in time. This method deviates from the method established earlier as the temperatures do not reach above 800°C, but would still be instructive to the validity of the cooling rate model.

For thermocouples 11 to 13 the cooling rates are extracted from the measured temperatures for the first and second bead of layer 25. The thermal cycle of the first bead of the layer show that the thermocouples 11-13 have a the higher cooling rate compared to the second bead. On the other edge of the layer, where thermocouples 14 to 16 are situated, the last bead has higher cooling rates.

Following the same procedure the cooling rates are calculated from the model results. There is no substantial difference between the first and second bead either at the start of the layer nor at the final beads in the layer. However, at the starting side the cooling rates are predicted to be somewhat higher compared to the finishing edge of the block.

4.4. Block A: Hardness & tensile strength

This section will discuss the mechanical properties of Block A as experimentally determined.

4.4.1. Block A: hardness

Hardness was measured on the tensile test specimens cut from the pyramid blocks.
The cylindrical proportional tensile specimens were made according to ISO 6892-1. For specimens 1 and 2 the diameter of the parallel length was 10 mm. For specimens 3,4 and 5, the diameter of the parallel length was 8 mm to maintain proportionality despite size limitations and manufacturing difficulties. In particular, the size and unwieldiness of the welded block made precision cutting harder than previously estimated, and so requiring scaling the horizontal test pieces in accordance with the standard.

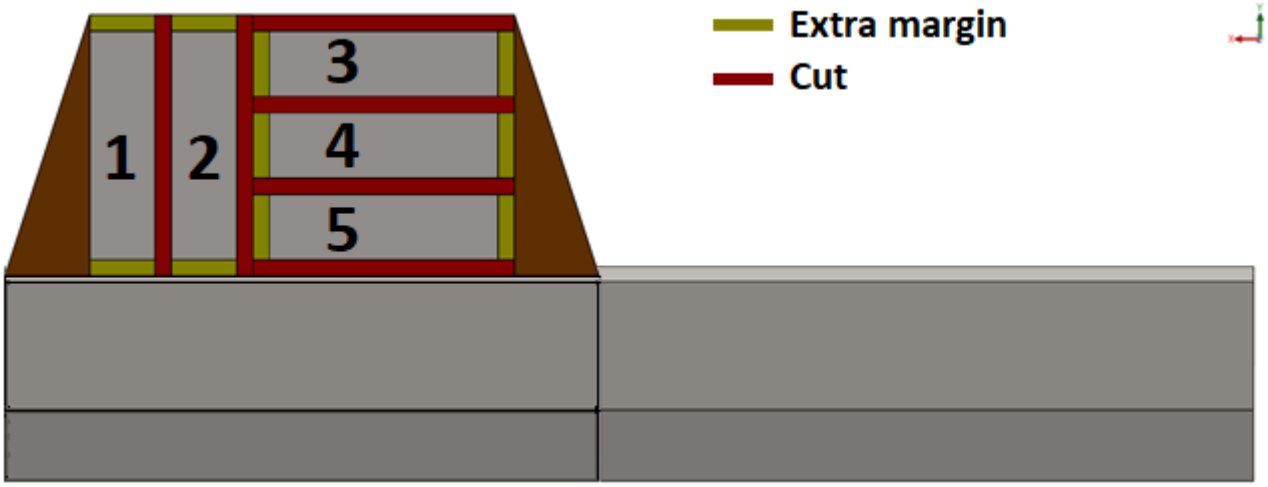
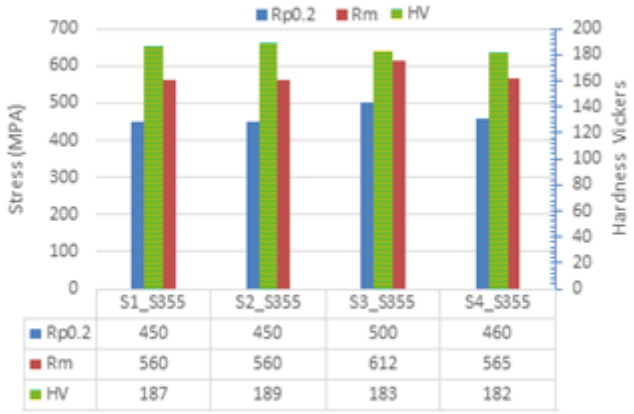


Figure 4-6: Cutting plan for block A. The samples are numbered 1-5. The red segments indicate material waste due to cutting. The yellow segments are extra margins because of cutting. The cuts are not fully draw into the angled sides or the base plate for simplicity.

Both tensile and hardness properties are plotted in Figure 4-7. The yield strength lies between 450 and 500 MPa, while the tensile strength lies between 560 and 612 MPa. This is considerably higher than the tensile properties of S355, but consistent with the "as welded" properties provided by the supplier [50].



	S1_S355	S2_S355	S3_S355	S4_S355
Rp0.2	450	450	500	460
Rm	560	560	612	565
HV	187	189	183	182

Figure 4-7: Yield strength, ultimate tensile strength and hardness of block A. Testing of Sample 5 failed and yielded no meaningful results. Each sample was 10 mm in diameter, with the hardness measured over the gauge length at two sides. Note the consistent yield strength, tensile strength and hardness values. As-welded, the block exceeds S355 qualification.

Figure 4-8 shows the hardness measurements of the two vertical samples. Two rows of measurements were taken per sample. Notable is the very consistent hardness across the sample, suggesting little hardness variation across the block.

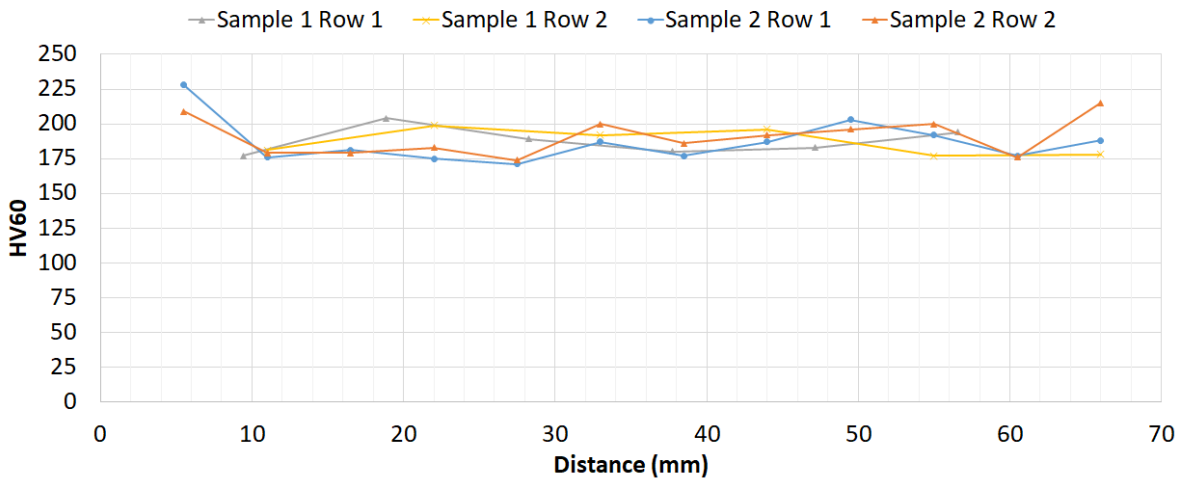


Figure 4-8: Hardness measurements of the two vertical samples (Sample 1 and 2 in figure 4-6 side view), starting from the base material.

Figure 4-9 show the hardness measurements of the horizontal samples, which follow the weld direction. Similar to the vertical direction, there is little variation across the length of the block.

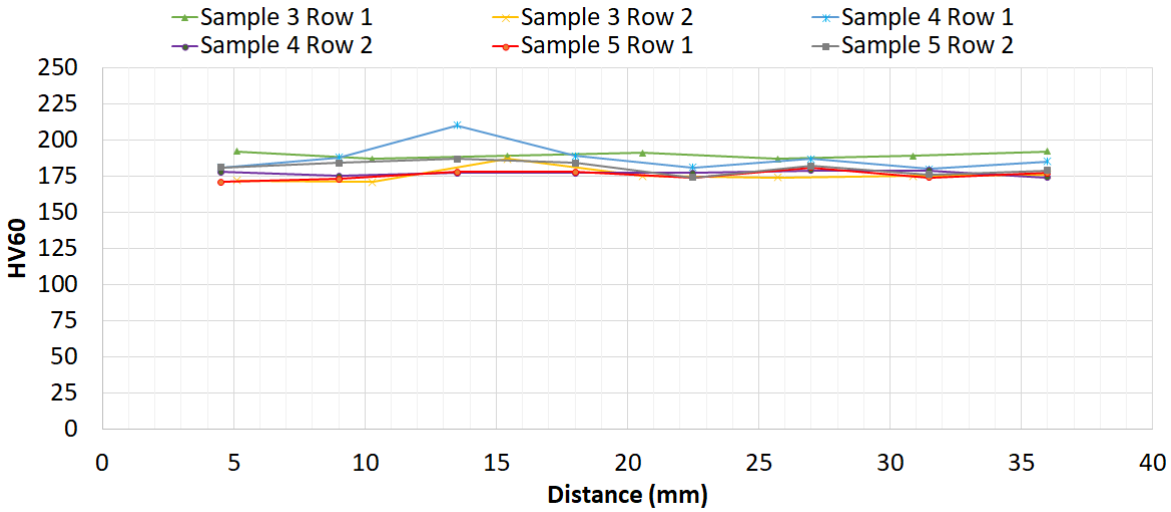


Figure 4-9: Hardness measurement of the horizontal samples in weld direction, (Sample 3, 4 and 5 in figure 4-6 side view).

Together, these results suggest a fairly homogeneous hardness distribution across the welded sample.

4.4.2. Block A: tensile strength

Figure 4-10 shows the tensile curves of the specimens. It should be mentioned that the number of tensile test specimen is very limited. Sample 2 (S2) was not tested until fracture. The result of Sample 5 is disregarded as the extensometer was not properly attached, resulting in unusable readings. The two vertical samples show very similar curves, implying that there is little mechanical variation in the vertical direction. The yield strength and tensile strength are within a few MPa. The two horizontal samples do show a greater variation, a yield and tensile strength difference of about 50 MPa.

The Sample 4 shows less work hardening and a greater strain-at-fracture, which might imply that the middle of the block is softer and more ductile than the top. The number of test results is too limited to derived hard conclusions.

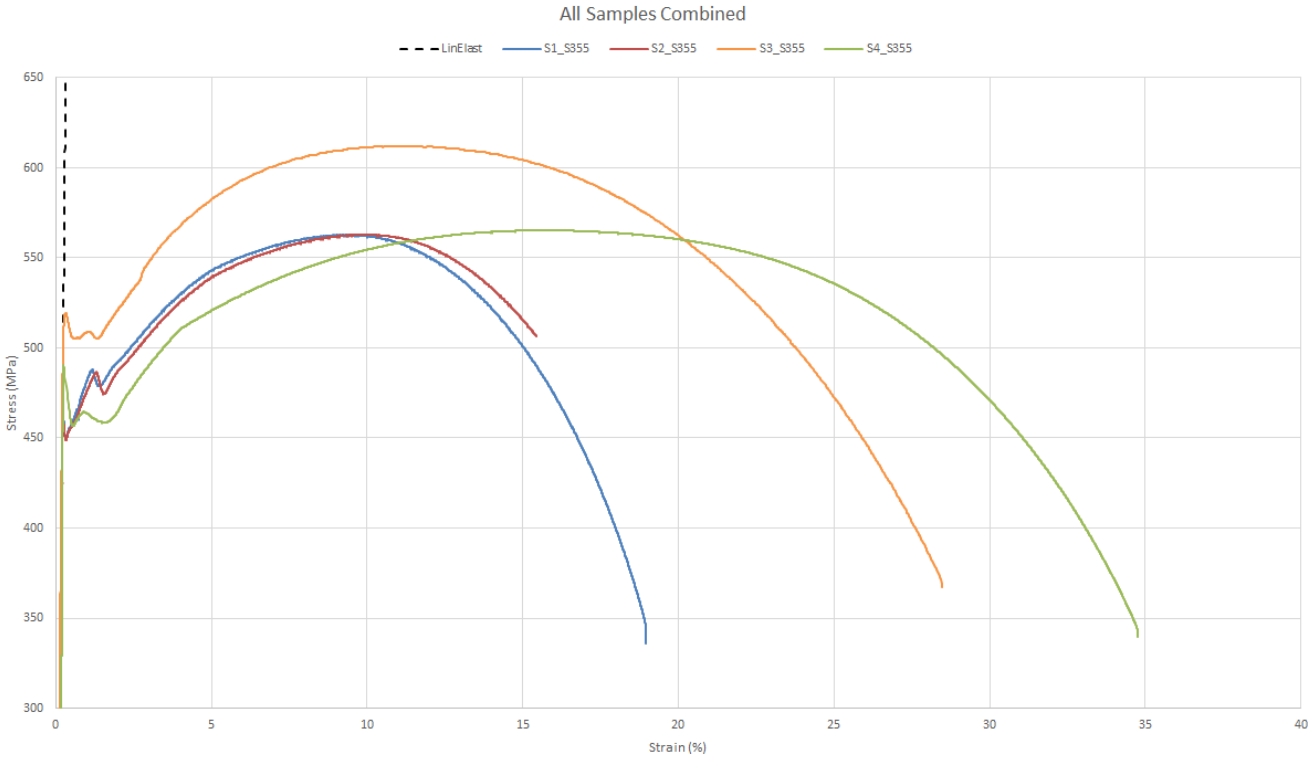


Figure 4-10: Stress-strain curves of the specimens. Note that the two vertical samples (S1, S2) are consistent but the horizontal oriented samples (S3, S4) show a considerable spread. Despite being welded metal, the samples show a strain-to-break of up to around 18%. Note that sample 2 (red curve) was stopped before fracture. The tensile results show that the material fulfils the requirement of the S355 standard

However, the tensile results show that the material fulfils the requirement of the S355 grade steel.

A correlation between hardness and strength (S) was established by Pavlina and Van Tyne [51] with the regression formula:

$$S = A_0 + A_1 H_V \qquad Eq.(10)$$

By taking the regression constants and coefficients from the "all steels" category for the yield and tensile strength, and taking 190HV as an average hardness a prediction of the tensile properties is obtained, shown in Table 4.

Table 4: Input values and calculated strength obtained from the Pavlina and van Tyne [48] strength-hardness correlation.

HV10	A_0	A_1		S (MPa) (calculated)		S (MPa) (measured)	
	-90.7	2.876		Yield:	456	Yield:	450-520
190	-99.8	3.734		Tensile:	610	Tensile:	560-610

From these results, it is clear that this correlation matches closely with the experimentally obtained values in this study. Hardness values provide a good indication of the tensile properties.

4.5. Block A: Cooling rate and Hardness Modelling

Now that the sensitivity of the thermal model has been established and the models are validated by temperature and hardness measurements, the models are applied to obtain further insight in the properties of the entire? block produced, based on calculated cooling rates. Section 4.5.1 provide the modelling results when layer 25 is deposited. In section 4.5.2 the effects of the deposition of an additional layer (layer 26) is simulated.

4.5.1. Block A layer 25

As a reminder, the model mesh is subdivided by layer height, weld width and a prior established sufficient deposition length. At layer 25, 12 welds are made, resulting in a width of each layer of 39 mm. Each bead is thus taken as 3.25 mm wide. The layer height was determined by taking the difference in height between two welding sessions and the amount of welds deposited, which resulted in an average height of approximately 3.2 mm. The deposition length in the model was set equal to the bead width, i.e. 3.2 mm. To properly locate the thermocouples, the two layers below the deposition (layer 23 and layer 24) are subdivided in an element height of 1 mm. This yields a total of 16k nodes and 14k elements in the model.

Figure 4-11 shows the resulting meshed block. The further away from the weld, the coarser the mesh is made to reduce the number of elements and thus the calculation time. The fine-meshed layers clearly show where the top of the previously deposited block is situated. The top layer (layer 25) is one element high, while layers 23 and 24 consist of three elements in the high direction. The layers below layer 23 are again 1 element high, while the remainder of the block is modelled by 1 element for 3 layers. The plate upon which the block is made is coarsely meshed as it experiences no large temperature changes during welding.

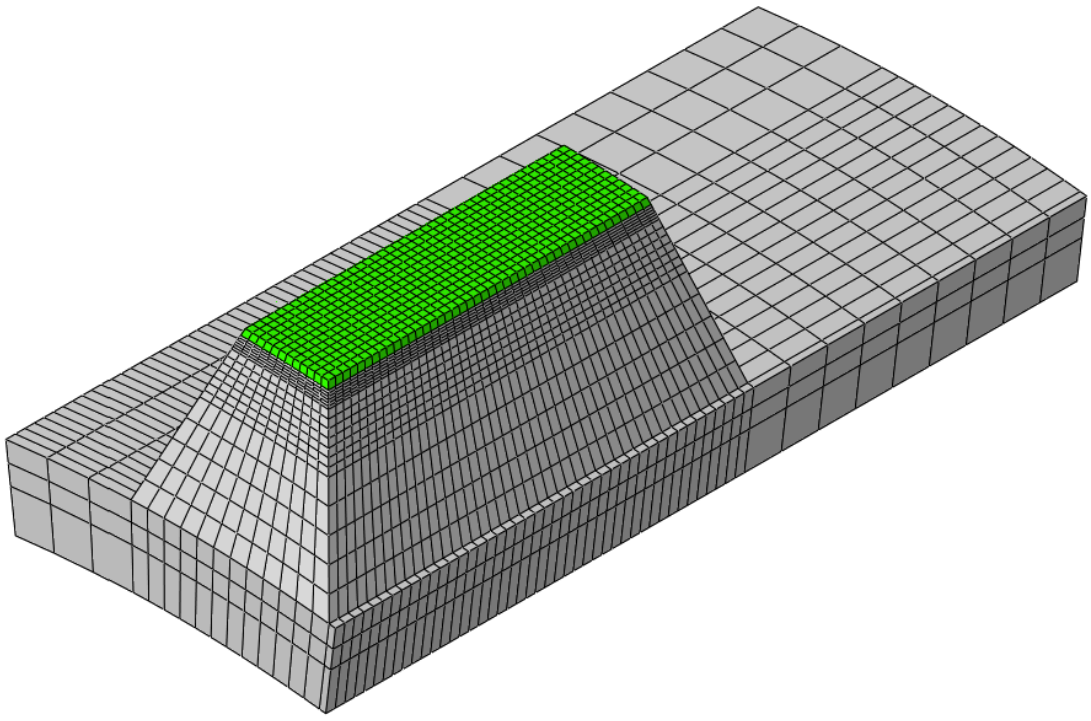


Figure 4-11: The mesh used for layer 25, with the deposited layer indicated in green. Note the finer mesh subdivision for layers 23 and 24. For material far away from the weld, the mesh is made increasingly courser to reduce the number of elements and calculation time.

Figure 4-12 shows the thermal field for several beads. The 5-second time step that is needed for the robot arm to move from the end of bead N to bead N+1, is enough to make the peak temperature of the weld cool to about 500-600 °C. For each next bead the region that remains hot grows, indicating heat-build-up in the block. As the start side of the block cools, this creates a temperature difference with the freshly-welded region, allowing heat to flow not just down towards the plate but also towards the starting edge. In addition, it can be seen that the 200 °C region becomes substantial, about half the layer width. This suggests that the influence of the heat input due to the welding of a bead remains constant after several welds have been deposited. It can also be seen that when depositing the last bead, i.e. after 4 minutes of welding, the heat from this layer has only penetrated about 4 layers for the 300 °C isotherm and around 11 layers for the 169 °C isotherm.

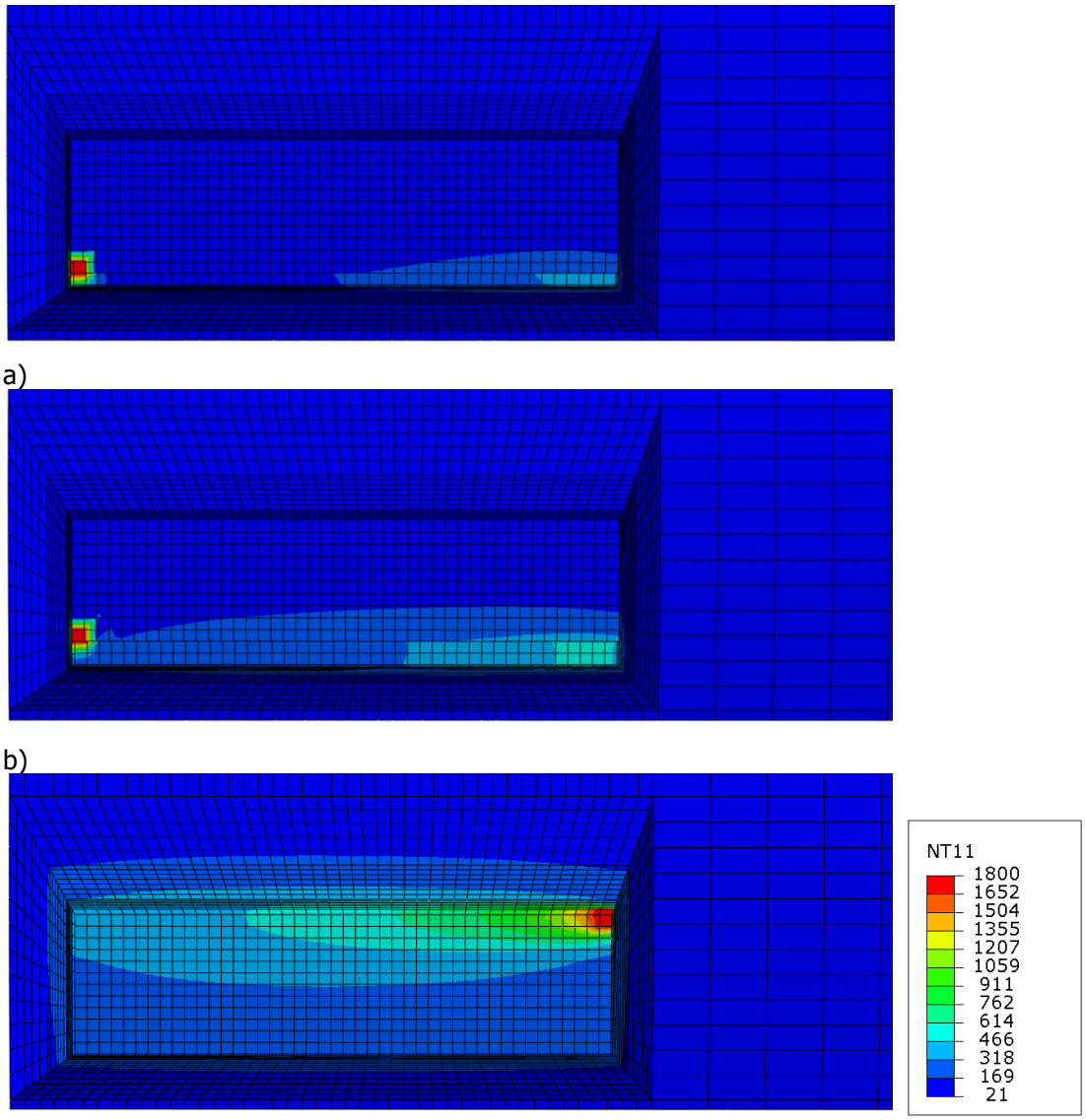


Figure 4-12: Different simulated thermal fields for the start of the second bead (a), the start of the third bead (b), and the end of the final bead (c) in layer 25. Note how the temperature of the block increases with each bead. Each bead N reheats bead N-1, causing a steady temperature build-up.

In Figure 4-13, the calculated cooling rate results are shown for block A layer 25 only. This means that the thermal history only for the simulation of depositing layer 25 is considered and not the deposition of prior layers. Values below 10 °C/s are shown as grey, values above 110 °C/s as pink. The values above 100 °C/s are cut off because these only occur in a few nodes and otherwise distort the picture too much. Pink nodes are not visible on this general overview figure, but can be observed in later cross-sections.

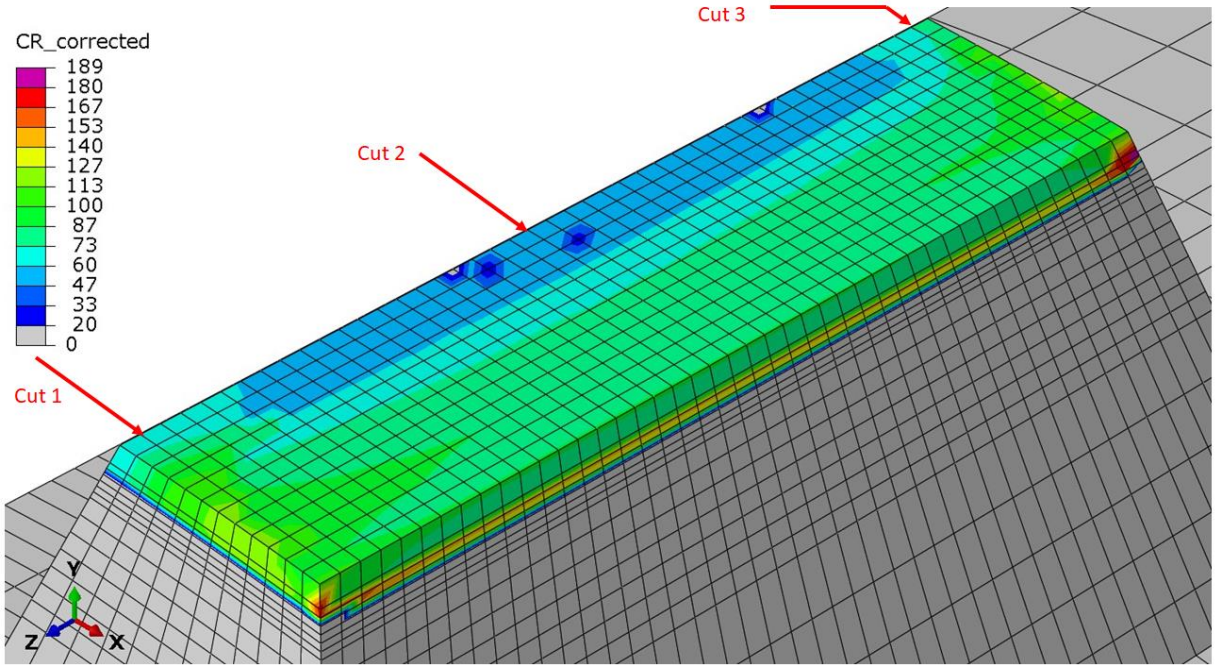


Figure 4-13: T800-T500 cooling rate for layer 25-influence only, Block A. Cooling rates are in °C/s.

The calculated results of the cooling rate shows two zones with high cooling rates at the start and end of bead 1. The cooling rate appears consistent from top to bottom. The cooling rate decreases from the first to the last bead, as the interpass temperature increases due to heat accumulation in the block. The cooling rate also has a higher value at the start and end of welds compared to the center of the block. At the start of the weld, the area is locally colder (Figure 4-12), leading to higher cooling rates. At the end of the weld, the removal of the heat source leads to a similar result. In the middle, the material in the vicinity of the weld is kept hot by the applied boundary condition, leading to lower cooling rates.

Figure 4 14 shows the thermal history of two nodes: one in bead two, and one in bead ten. Both are near the start of the deposition. The node in bead 2 is deposited in the liquid state, after solidification re-molten by bead 3, then re-heated by bead 4 to 900 °C. For the calculation of the cooling rate this is considered to be a valid re-austinitization. The fifth bead only re-heats the material to 422 °C, and therefore this thermal cycle is disregarded. For the bead close to the end of the layer, bead ten, the first two steps are similar, but the deposition of bead 12 only re-heats the material to 748 °C, what is considered an invalid temperature to obtain a cooling rate. To some degree this is a correct assumption. The layer is accumulating heat and becomes considerably hotter for the later welds, the cooling rates will be correspondingly lower. However, to some degree the boundary condition is artificial. 748 °C is still a substantially high temperature peak that might bring changes in the microstructure. Additionally, the temperature stays at around 200 °C for about 5 minutes, which will have a tempering effect on the material, resulting in a lower hardness [52].

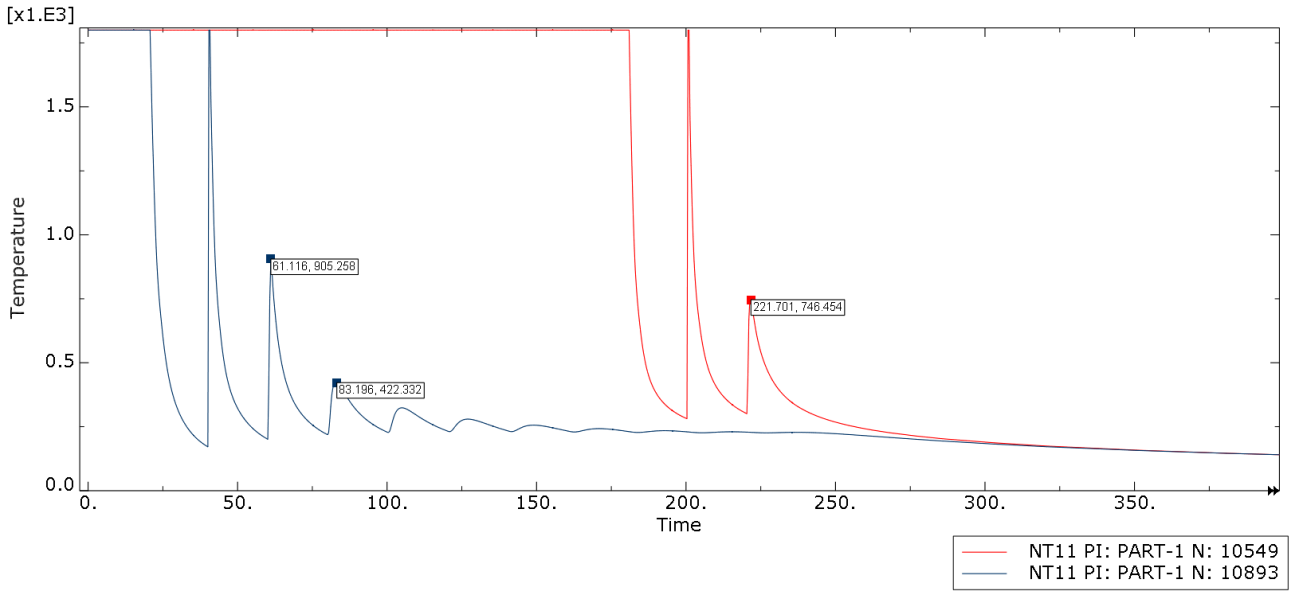
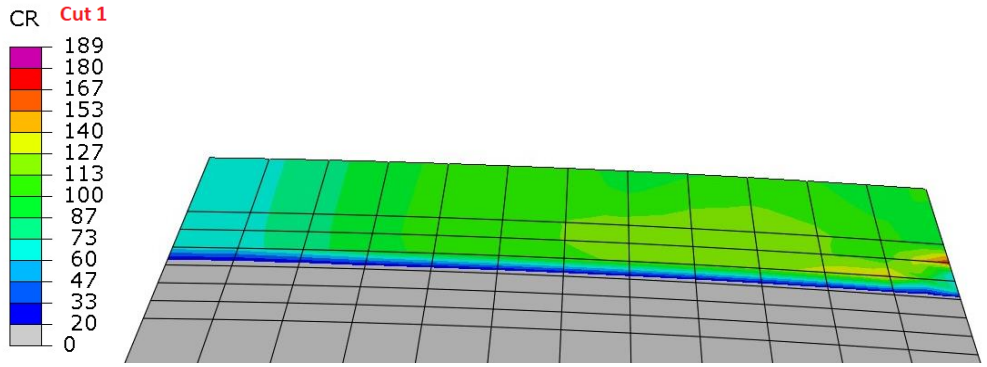
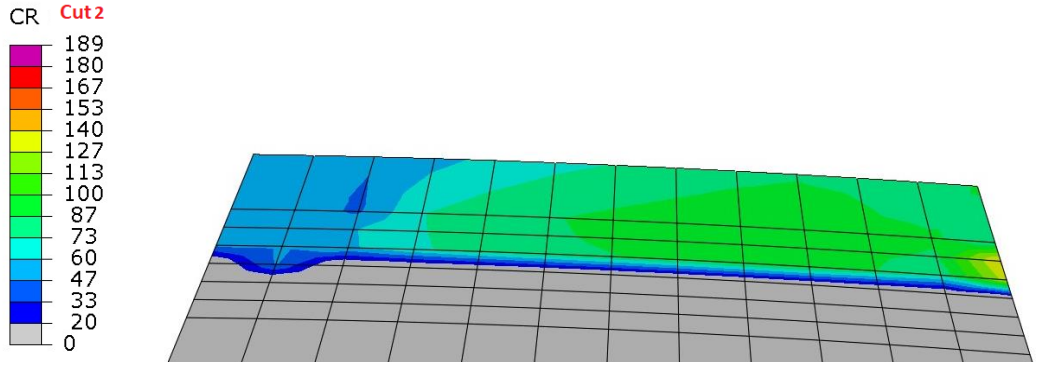


Figure 4-14: Block A layer 25. Thermal history of a node from the 2nd bead (blue curve), and a node from the 10th bead (red curve). The 2nd bead is deposited at 1800 °C, re-melted by bead 3, re-heated to 905 °C by bead 4. The 10th bead is deposited at 1800 °C, re-melted by bead 11 and reheated by bead 12 to 748 °C

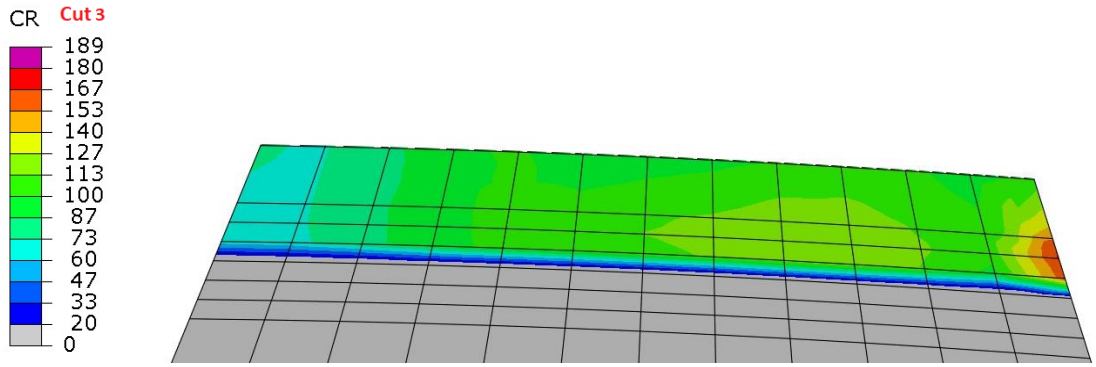
To get a better insight of the cooling rate as a function of height in the block, the cooling rate distribution of three slices are plotted, at locations specified in Figure 4-13 by the red arrows. These cross sectional cooling rate distributions are shown in Figure 4-15a-c. These figures confirm that the cooling rate is fairly consistent throughout the layer. The start and end of a weld show similar cooling rate distributions. At the deposition start side (right side), the cooling rates are consistently higher than at the deposition end side (left side). The results also show roughly 2mm penetration into the layer below, re-austinitizing the previous layer. Notable is that for the middle segment, the cooling rates on the layer end side are much lower. However, the layer below also show more cooling results. This suggests that the lower cooling rates are because of heat accumulation, leading to higher temperatures deeper in the previous layer.



a) Cross-section at cut 1



b) Cross-section at cut 2



c) Cross-section at cut 3

Figure 4-15: Block A. Cross-sections of the cooling rate at different locations as indicated in Figure 4-13.

Based on the calculated thermal profiles and the cooling rates obtained, the hardness distribution due to the deposition of layer 25 is simulated. The modelled results are shown in Figure 4-16. As would be expected, the hardness profile follows the cooling rate profile closely. High cooling rates correspond to high hardness. The edges of the block show a higher hardness than the center, due to aforementioned edge effects. The average hardness of the block is 260HV, with the yellow-orange region a low of 250HV and the red areas a high of 270. The global lowest hardnes is 163HV, on a single node. The hardness of the layer is notably higher than the average of 190HV as measured.

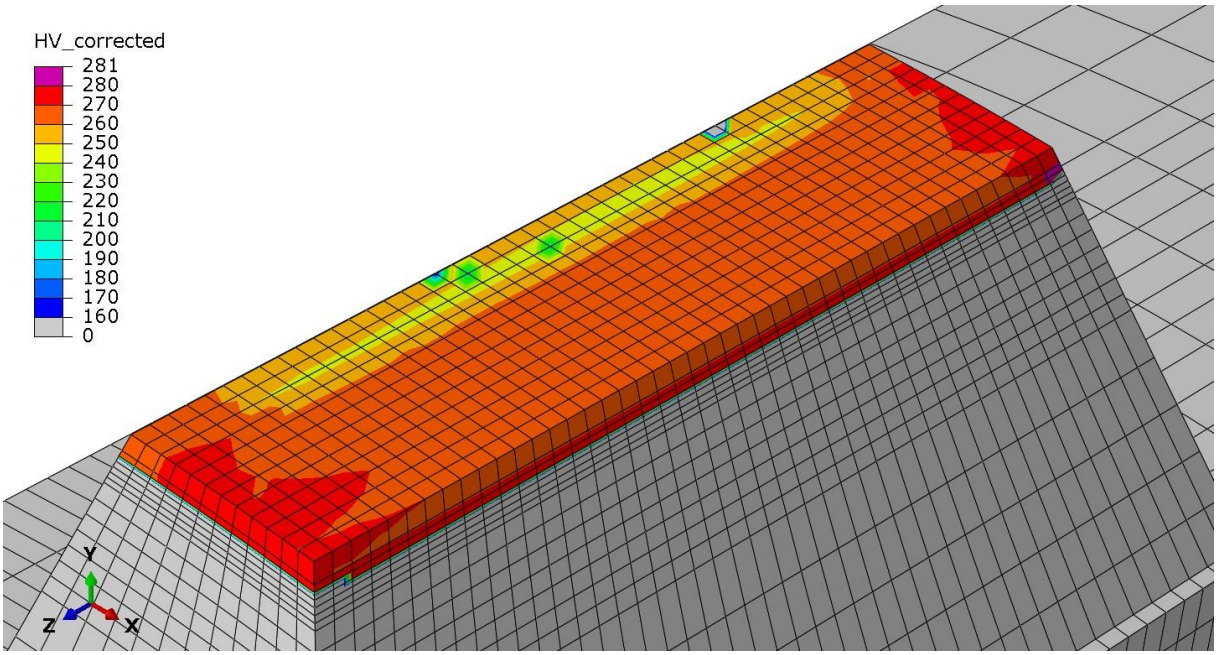


Figure 4-16: Block A layer 25: hardness corresponding to the cooling rates.

Looking at the CCT diagram in Figure 3-11, the predicted microstructure is roughly 50% martensitic by JMAT Pro estimations. This seems very unlikely, as martensite is fairly brittle yet the tensile testing showed no such behavior. The hardness of the actual block is also too low for such levels of martensite. This suggests that the model significantly overestimates the cooling rates. Working backwards, a hardness of around 190 HV suggests a cooling rate of about 20-30°C/s. Another reason could be that the effect of tempering is much stronger than previously assumed in this thesis. It appears unlikely that such a large mismatch in cooling rates would not be observed in the thermal analysis, making tempering the more likely reason.

4.5.2. Block A layer 26

As previously mentioned, depositing a layer reheats and partially remelts the previous layer. To obtain the cooling rates and hardness of layer 25, the thermal cycle of layer 26 should also be considered. This data will also supplement the previous analysis. Deposition procedure of layer 26 is implemented in the model. Please note that bi-directional deposition was employed in printing the layers of the block. Each bead is thus deposited in the reversed direction, although in the same order.. In addition, when layer 26 was deposited an arc start failure occurred as shown in Figure 4-2. Since this gives the block extra cooling time, therefore higher cooling rates are expected.

Figure 4-17 shows the simulated cooling rate result for the deposition of layer 26. A similar pattern is seen as with layer 25. The result indicates that alternating the bead direction for each layer has little impact on the general trend of the cooling rate distribution.

The arc start failure happened at the 4th bead of layer 26. At this location a higher cooling rate is calculated, as can be seen in figure 4-17. The time involved in restarting the deposition process allowed the block to cool down. The consequences for the hardness distribution can be seen in Figure 4-17.

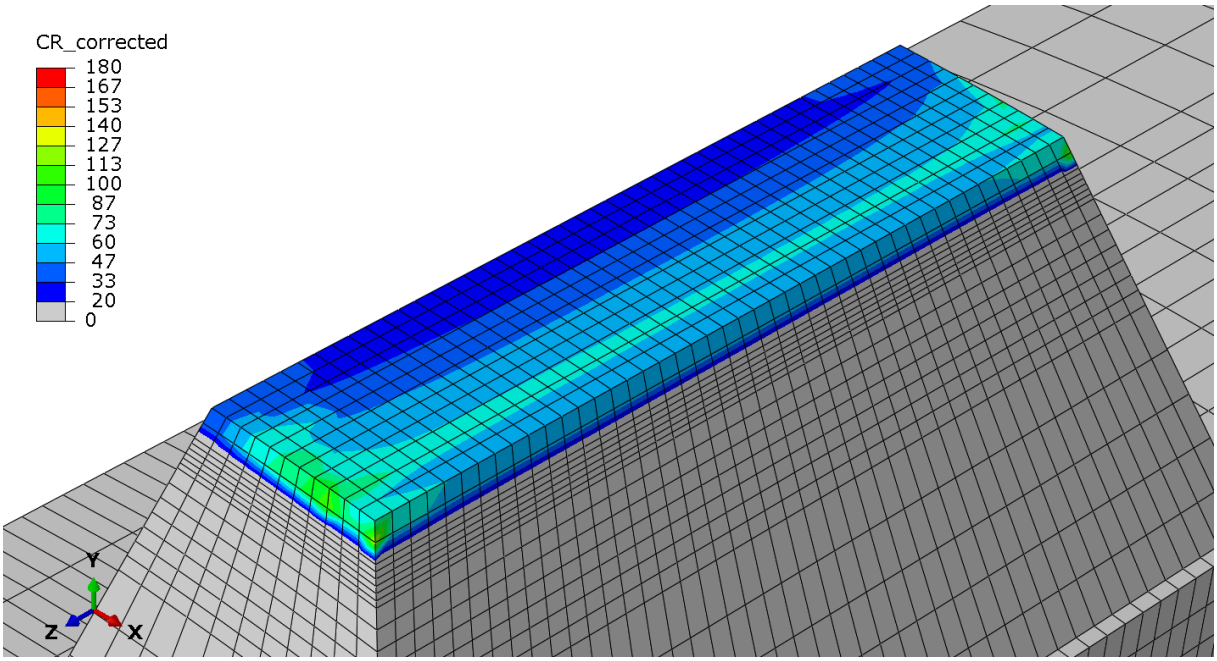


Figure 4-17: Block A layer 26, cooling rate distribution in °C/s.

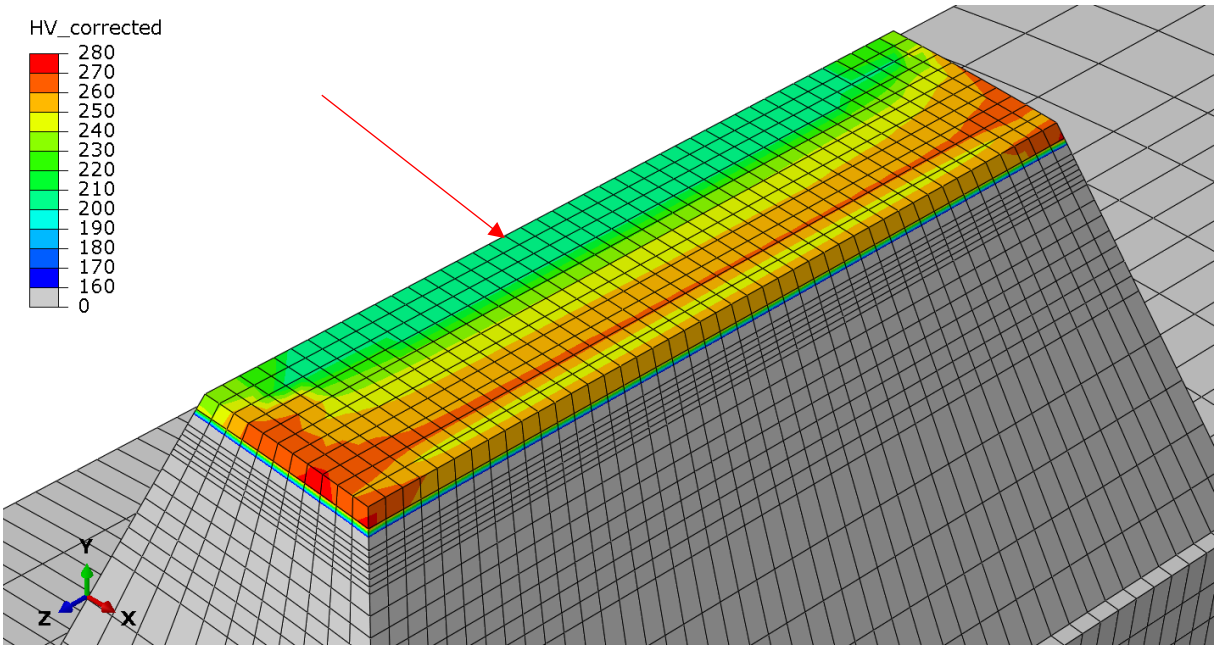


Figure 4-18: Block A layer 26 hardness. The red arrow indicates the cut plane for figure 4-21.

The hardness distribution for layer 26 is comparable to that of layer 25. The red regions show a hardness of about 260HV while the green area has a hardness of roughly 200 HV. The larger green area is thus much more closely in agreement with the observed hardness values. However, this arc start issue only occurred in layer 26. Similar to the previous layer a section cut is made near the center of the block, indicated by the red arrow. Figure 4-19 shows the cross-section at this arrow.

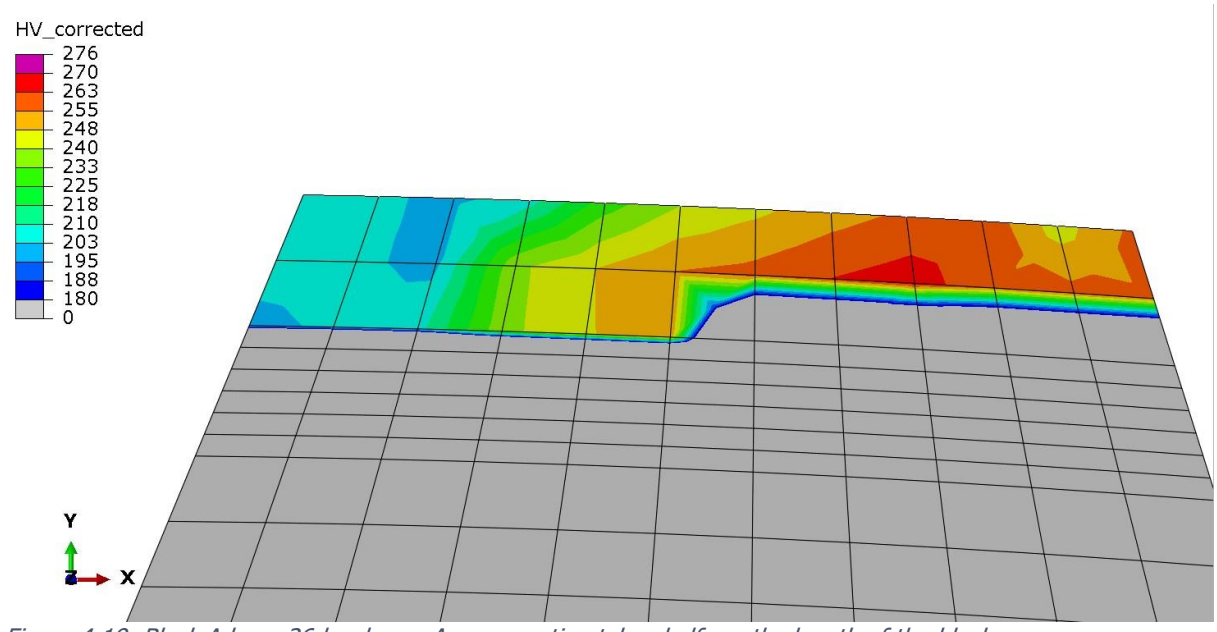


Figure 4-19: Block A layer 26 hardness. A cross section taken halfway the length of the block.

The hardness follows a similar pattern as with layer 25, although the effect of the arc start failure can still be seen. The fourth layer has the local high hardness consistent with a fast cooling rate due to a cooler block.

4.6. Block B: Case study

The constructed model is now used to simulated temperature field and hardness for the deposition of block B, which is a 120 x 60 mm^2 block built in S420. Details of the welding conditions can be found in chapter 3.1.1 and 3.1.2. In priciple the of printing the block is similar to Block A.

4.6.1. Block B: Modelled hardness

The simulation results of the first and last (10^{th}) layer will be presented to show the effect of distance from the substrate on the final hardness properties. In this simulation material data obtained from JMAT Pro [43] is used to include the effect of this higher strength steel in the computations. The mesh size was maintained the same as the previous model, where the layer height is 3 mm and the width and length of the element is 3.2 mm. In the model, the number of nodes is now 17k, whereas the number of elements is 15k. Figure 4-20 shows the mesh and the temperature field during welding. From the temperature data of both layers the cooling rates are calculated as well as the hardness distribution for two different steel grades. The results are presented in in Figure 4-21.

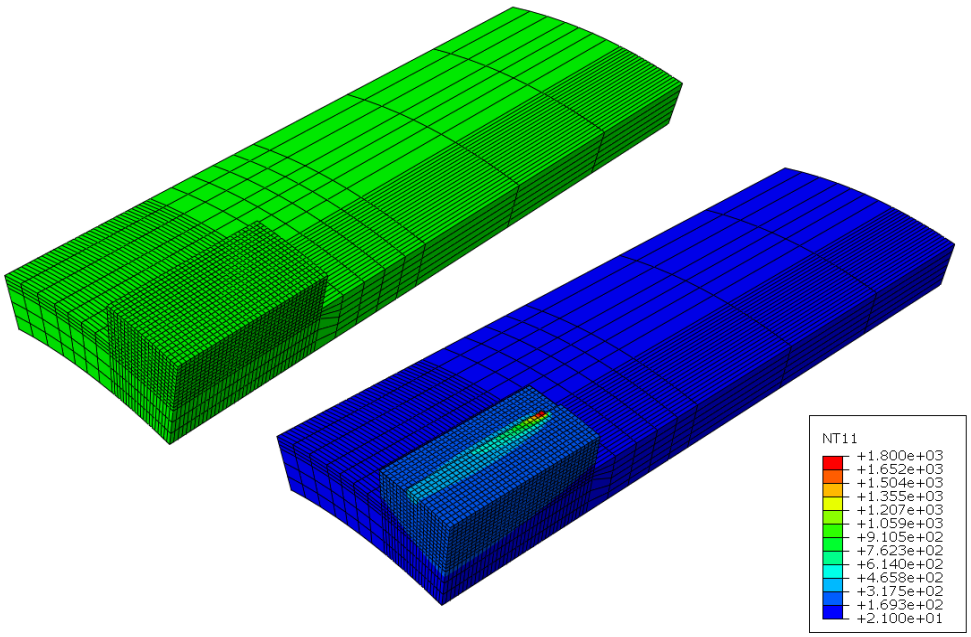


Figure 4-20: Mesh and temperature profile for layer 10 in block B. Layer 1 is simulated with the same mesh, the not yet deposited layers are deactivated.

The cooling rate is calculated for both layers. Based on the relevant calculated CCT diagrams the hardness profile is simulated for S420, a higher grade than the previous block. In this case, S420-equivalent material is expected to be more sensitive to the temperature changes. This would be shown as a bigger variation in hardness.

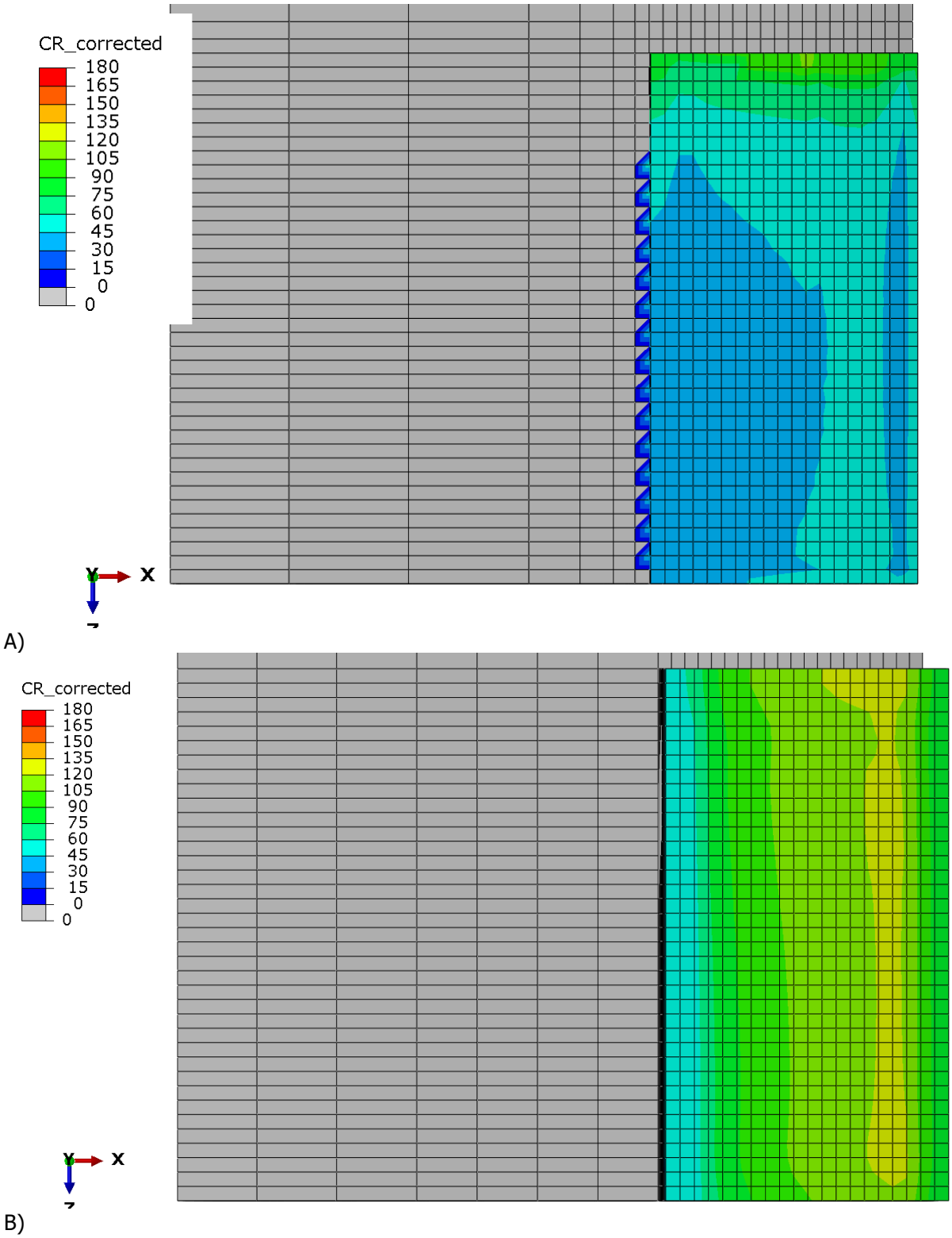


Figure 4-21: Cooling rates for layer 1 (A) and Layer 10 (B)

Figure 4-21a shows the cooling rate for layer 1. The layer shows a clear pattern where the cooling rates are highest at the end of the bead deposition. Cooling rates also decrease from the first deposited bead to the last deposited bead, similar to the previous two cases. At the end of the weld, the large amount of surrounding cold material allows for high cooling rate compared to the start. However, at the start of the bead this effect appears much less, likely due to the stronger edge effect. The sloped angles of Block A appear to somewhat offset this local effect compared to block B.

Figure 4-21b shows the cooling rate for layer 10. Compared to layer 1 the effect of bead order is much more pronounced and the effects of the start and end of the weld is much less visible. The overall coolingrate is also much more uniform, and overall higher than layer 1. These values will be compared to the experimental values in the next section.

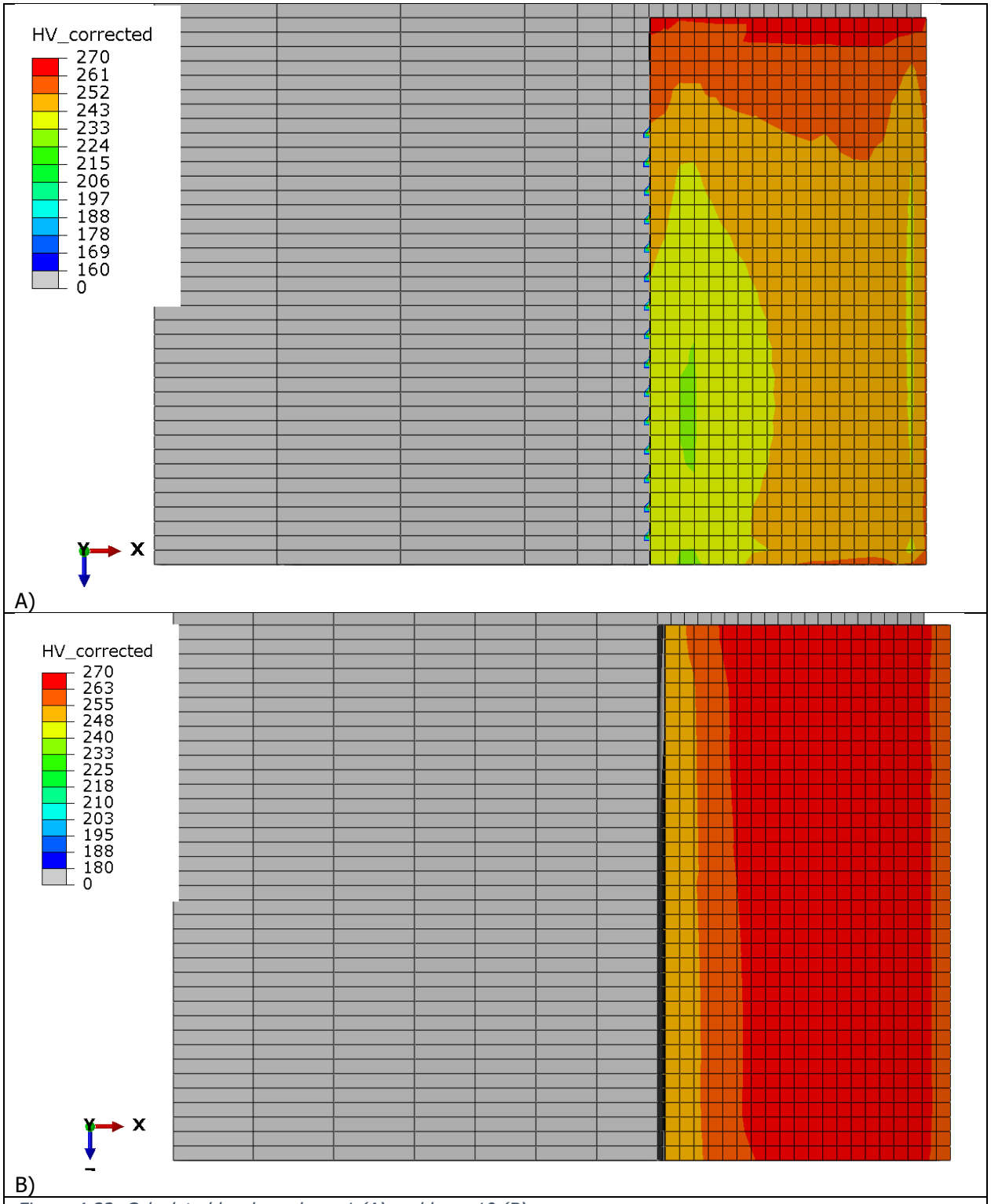


Figure 4-22: Calculated hardness layer 1 (A) and layer 10 (B)

Figure 4-22A shows the hardness of layer 1. The hardness distribution follows the cooling rate results quite closely. The average hardness is about 240HV, with a maximum of about 260 HV at the bead end region and a local low in the green region of about 230HV.

Figure 4-22B shows the hardness of layer 10. Similar to the cooling rate results, the hardness is very uniform. The average hardness is about 260HV.

4.6.2. Block B: Experimental hardness

The predicted hardness values are compared to experimentally obtained results. Block B was sectioned and a cross sections 50 mm from the edge of the block was prepared for hardness measurements (Figure 4-23). Three lines of hardness measurements were made, from the base plate towards the top layer at three locations as indicated in Figure 4-25. The results are provided in figure 4-26.

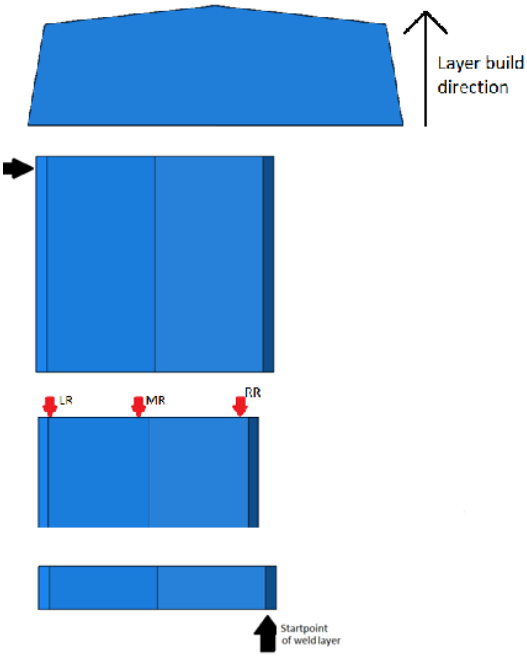


Figure 4-23: Top: cross sectional view of the deposit indicating the building direction, Bottom: top indication of starting location and finish location of a layer, all beads in a layer start at the same side, Middle: indication of the location of the hardness measurements, 50 mm from starting edge (measured on a cross section), RR right row, MR middle row, LR left row.

The hardness distribution of block B is notably quite homogeneous. The hardness deviates between 184 and 225 HV_{10}. The average hardness is around 200HV_{10}. The hardness does not change significantly over the height of the block, nor between the measurement lines. The alternation of the weld direction for each layer leads to a more even hardness throughout the printed part.

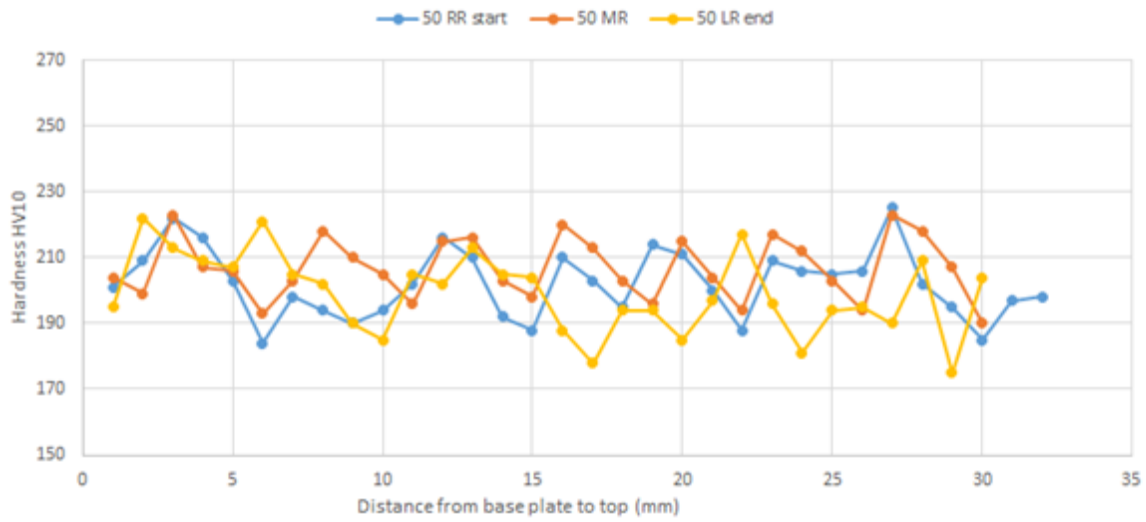


Figure 4-26: Hardness as a function of the distance from the base plate for three location, RR starting side of the layer deposition, MR middle section of the block, LR finishing side of the layer deposition.

Comparing the experimental results with the modelled values in Figure 4-22. It appears that the model calculates both a higher hardness minimum and maximum. The modelled hardness lies between 230 and 260 HV, which is approximately 30 HV higher than what is measurements. This can be explained by the tempering effect not taken into account in the model. The block stays at relatively high temperatures (200-300 °C) for a substantial duration, which results in a lower hardness. The variation in simulated hardness is around 35 HV, similar to the measured results. The difference is also notably better compared to block A, which had a difference of nearly 80HV points between model and experiment.

Fout! Verwijzingsbron niet gevonden. shows a micrograph of the 120x60 block (Block B), including indents and measured hardness. The hardness just below the fusion line is lower than that in the weld. The grain growth direction from the fusion line can be seen clearly. As seen with Figure 2-19a, the deposited material grows from the fusion line to the top. This creates the crescent shaped pattern in the microstructure. The prior austenite grain boundaries are visible, showing long cellular-like growth indicating a relatively high thermal gradient compared to the interface velocity. The prior austenite grains are decorated with grain boundary ferrite. This makes sense in the melt pool of the weld. Near the Heat Affected Zone (HAZ) the grains become larger and more equiaxed, indicating a lower thermal gradient. When the indentations line up with the welds, the wave-like pattern such as shown in **Fout! Verwijzingsbron niet gevonden.** appears. The size of the weld is around 3 by 3 mm, similar to the element size in the model. As such, these variations are averaged out (in this case, 206 HV_{10}).

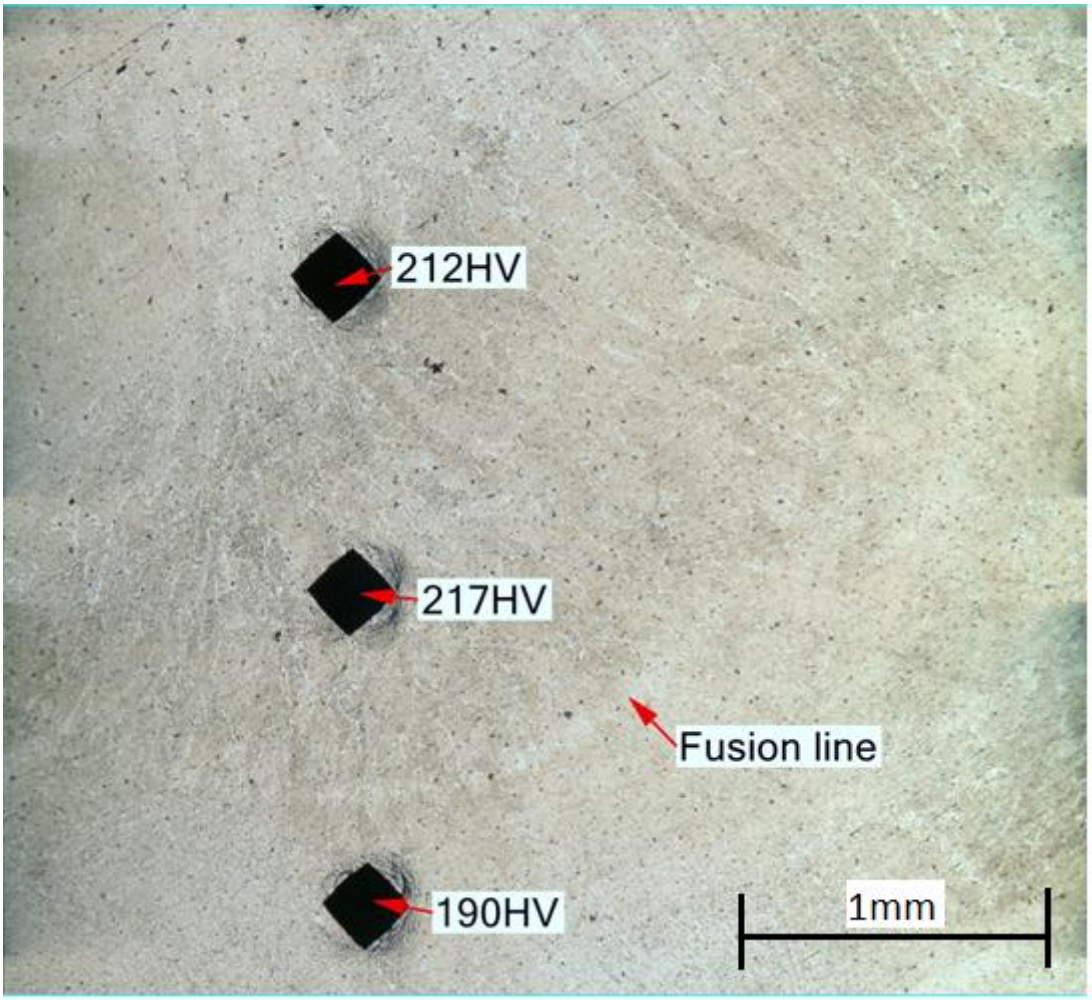
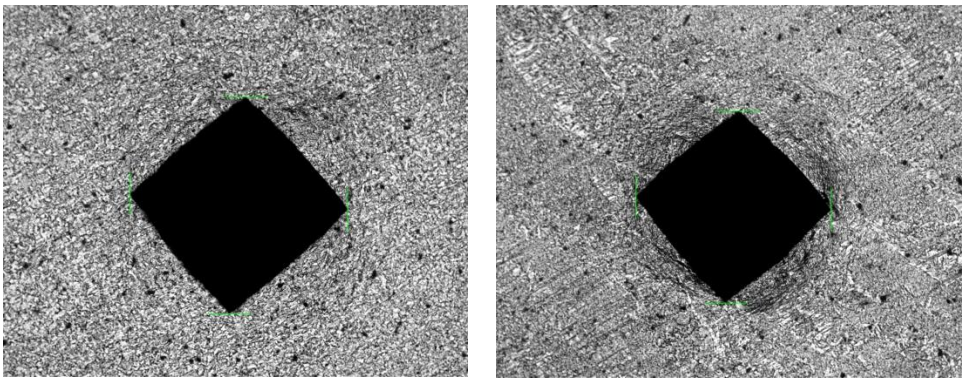


Figure 4-24: Macrograph of deposit in Block B showing hardness indentations.

Figure 4-25 shows the detail of two separate indents.



a) b)

Figure 4-25: Indent details. a) an equiaxed microstructure found in the heat affected zone, 312 µm diagonal indent (190 HV), b) a columnar structure in the middle of the weld, 277 µm diagonal indent (241 HV).

4.1. Block C: Case study 2

In the second case study, the intention is to stud the hardness distribution when the deposition strategy is changed. For this block C printed in S420 steel, 3 layers in the middle section of the block were deposited with a reversed deposition direction . First 6 layers were made with the same strategy as in block A, the subsequent 3 layers had a reversed orientation followed by a final layer similar to the first 6 layers.

Figure 4-26 shows the schematic drawing and the procedure for sectioning for hardness and microstructural evaluation, similar to Figure 4-23. In this block two slices were cut from the block, at 10 and 50 mm from the starting edge, to investigate the hardness change across the length of the block. Hardness was measured across several lines from bottom to top of the block to investigate the effect across the layers.

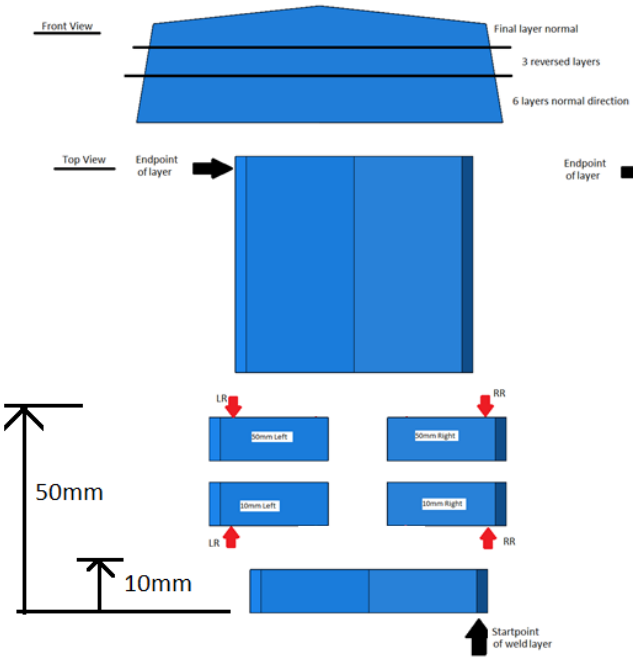


Figure 4-26: Top: cross sectional view of the deposit indicating the building direction, Bottom: top view indication of starting location and finish location of a layer, all beads in a layer start at the same side, Middle: indication of the location of the hardness measurements, 10 and 50 mm from starting edge (measured on a cross section), RR right row, LR left row.

Figure 4-27**Fout! Verwijzingsbron niet gevonden.** shows the hardness results for block C. The weld size is approximately 3 mm, while the indent distance is about 1 mm, meaning that three indents can probe one weld. When the indents are at similar positions across welds, a wave-like pattern appears. Figure 4-28 shows this in more detail.

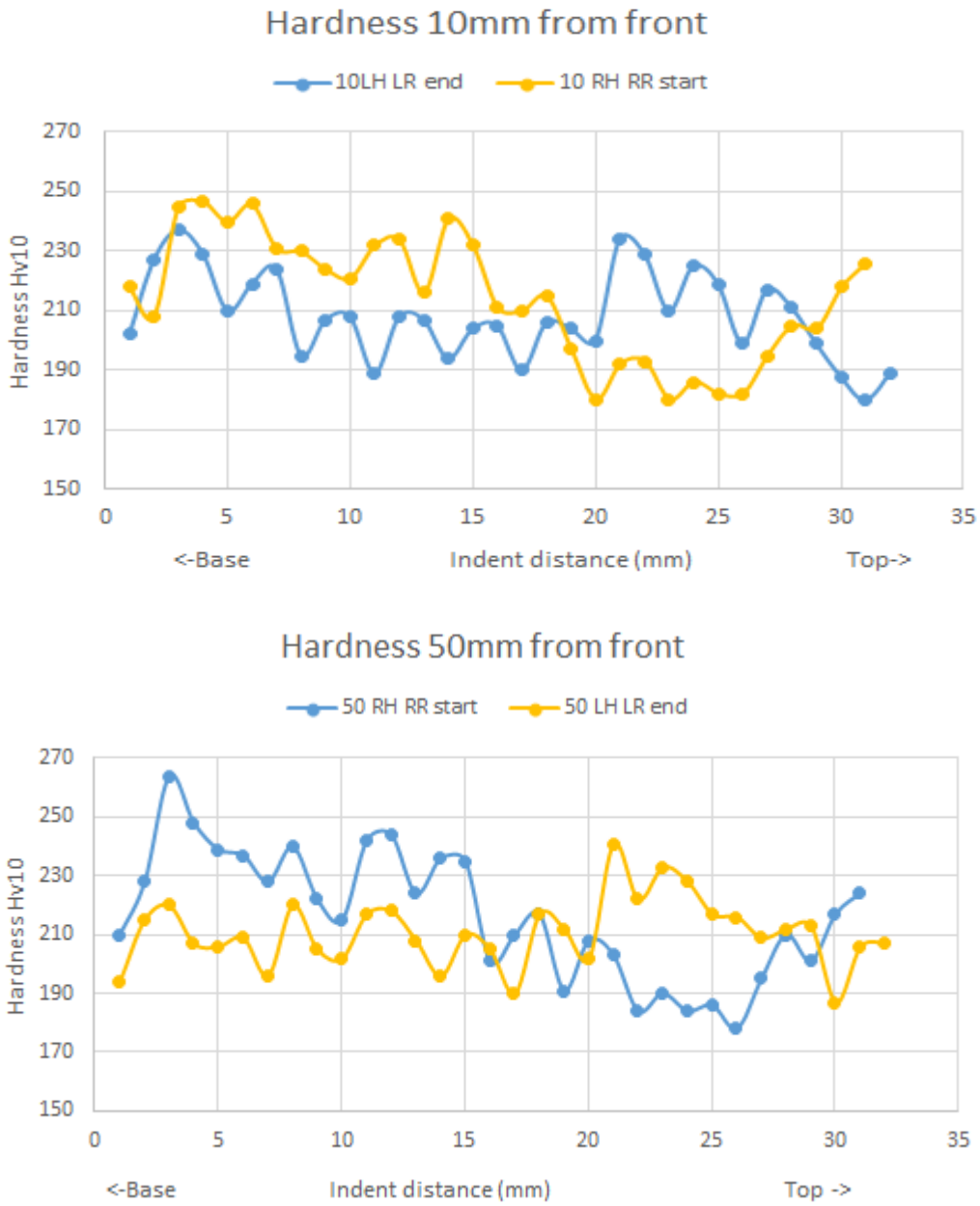


Figure 4-27: Hardness of block C. Note the switch in deposition direction at around 20mm, where the weld direction changes. After initially high values, the hardness becomes lower closer to the top. The weld is harder near the start position than at the end position.

The hardness distribution in block C shows that the hardness decreases with increasing layer height until the switch in weld direction takes place at around 20 mm. The hardness at the start of the layer is around 10-20 HV higher than the end. Lastly, the hardness does not change significantly between the two slices, indicating that the hardness is fairy homogenous in the length direction. The overall hardness is quite similar to block B and C. Since the layers were not alternated, the hardness difference across the length of the bead appears to be greater. As previously shown, each layer influences partially the previous layer by remelting or re-austinitization. This also means that the thermal differences across each layer are somewhat evened out.

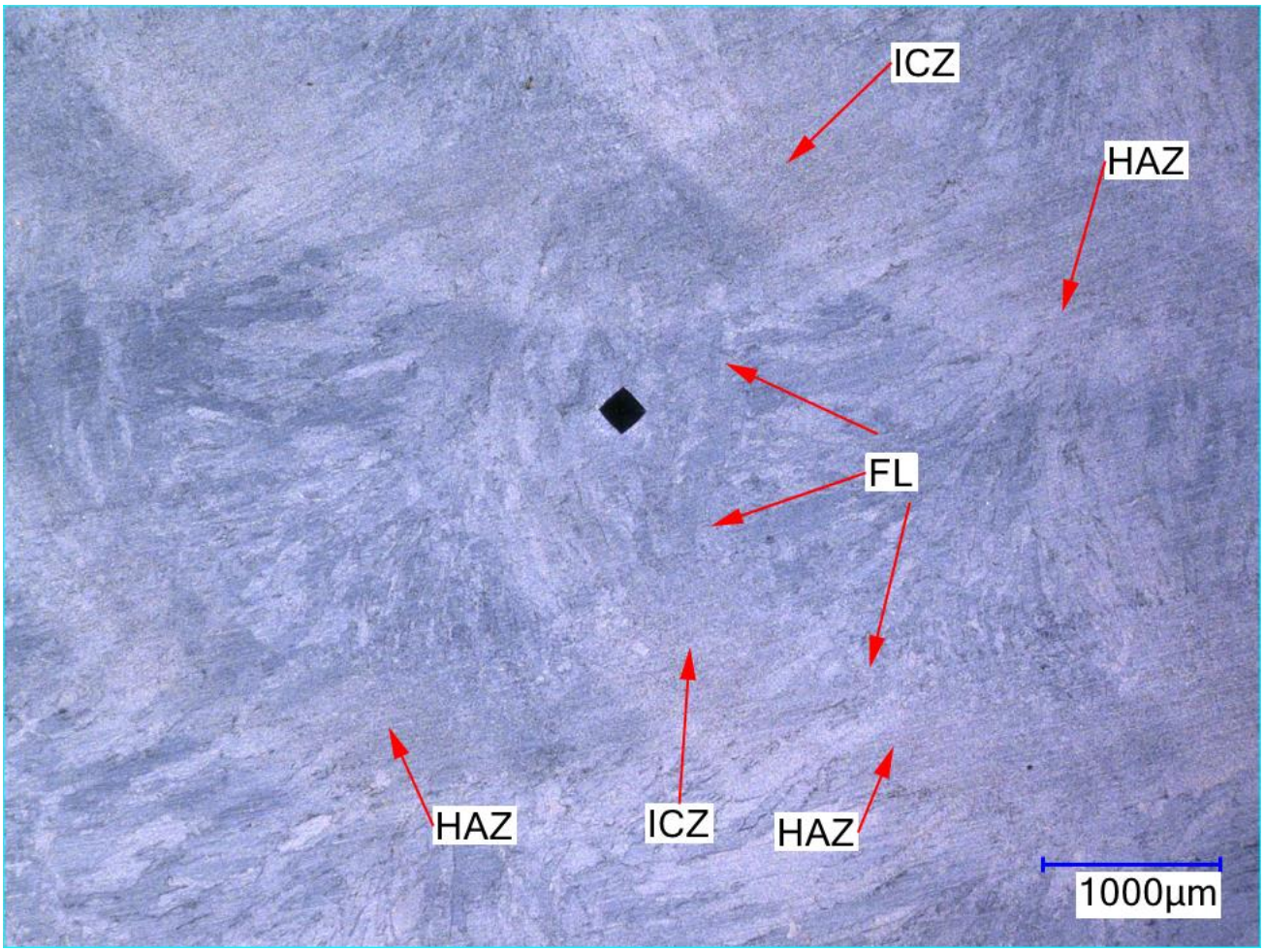


Figure 4-28: Macrograph of block C. The Heat Affected Zone (HAZ), Inter-Critical Zone (ICZ) and Fusion Line (FL) are indicated. Note the repeated structure. The horizontal stripes are the result of polishing. Grain size is fine near the fusion line and becomes coarser?

Figure 4-28 shows a macrograph of a larger part of welded material from Block C. Note the crescent-shaped heat affected zone about 1 mm into the neighbouring weld. In addition, a fusion line is indicated showing that the weld penetrates properly into the previous bead. Grain growth direction is perpendicular to the fusion line and converges to a point. Heat-affected zones overlap comparable to a multi-pass weld, resulting in complex structures. Still, a considerable amount of welded material is homogeneous, making the average-hardness approach reasonable. Taking into account the CCT diagram in Figure 3-11 and the measured hardness of around 180-220 HV, a ferrite structure with bainite and some martensite is to be expected.

This can also be seen in Figure 4-29, which shows the phase fractions calculated by JMAT Pro as part of the CCT diagram calculation. The structure is primarily expected to be ferrite-bainite. The phase fraction calculation does not take into account precipitations and complex phases in the inter-critical heat affected zone, which is appropriate for low carbon steels.

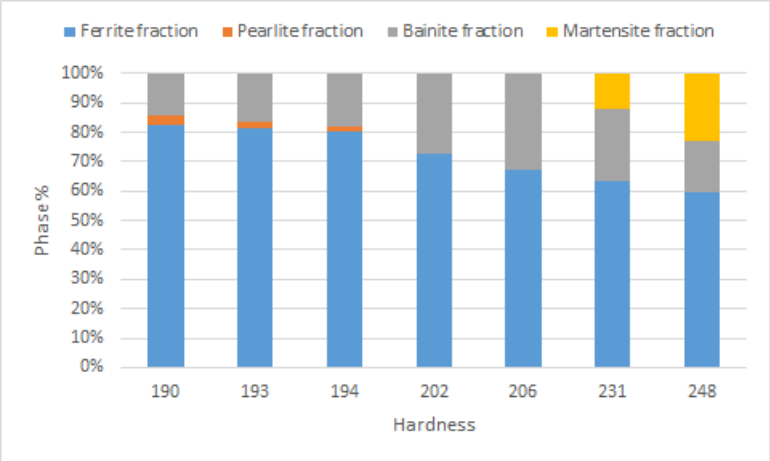


Figure 4-29: Phase fractions, calculated by JMAT Pro, for OK Aristorod 12.63. The measured hardness range is 190-220, corresponding to around 20-30% Bainite and the rest ferrite.

5. CONCLUSION

In this study, a three-dimensional finite element model has been established to investigate the distribution of mechanical properties using hardness in a large WAAM-deposited block. Experimental procedures were carried out according to the same process parameters as in the numerical model, and the numerical thermal cycles are in agreement with the simulated results. Major findings can be summed up as such:

- A boundary condition model can be used to model WAAM well.
- The hardness model exceeds the hardness of the block by around 40HV.
- Hardness trends predicted by the model are also seen in the experiments, indicating a working hardness model.
- Hardness is highest near the start of the layer deposition and lowest near the end of the deposition.
- Hardness is highest near the start and end of the weld.
- The mechanical strength of the deposited block is sufficient to match the minimum designation. The number of tensile tests is not sufficient to establish anisotropy with sufficient confidence.
- The effects of different deposition strategies can be investigated quite quickly, and have a noticeable effect on the hardness distribution of the block.
    - A single-direction weld strategy yields to a large variation in hardness between the start and end of the layer. Changing the weld direction and weld order can mitigate this.

The study was an early study into investigating the benefits of a boundary-condition based model. There are undoubtedly many aspects that could be optimized in future research. In particular:

- The effect of tempering appears to have a strong effect on the hardness of each block, as the model appears sufficiently accurate for the resulting coolrates. This leaves tempering as a major missing element.
- The effect and prediction of toughness is an essential component of grading welds, but there was not enough time to take this into account.
- The effect and prediction of residual stresses and distortion.
- Investigation into wildly different welding parameters. Welding of the investigated blocks was done with similar parameters and materials.
- Investigation into a quiet element model. In a quiet element model, the elements are present in the model but do not participate due to artificial thermal properties. This saves computational time. This was not done because it requires custom code to work while element birth-death method was a ready-to-use feature.
- A more extensive series of tensile test. Time and material limitations ensured only a limited batch could be made and tested. In particular the anisotropy of the block could be useful to ensure that the strength of the block is optimized for its intended function.
- Integrating an Infrared camera into the workflow. Thermocouples offer limited information, and the block deposition needs to be halted to place them. An IR-camera could offer in-situ information about the temperature field and thus improve the accuracy of the model.
- Improving the workflow of the modelling. A plugin was used to automate the task. Still, there is a considerable amount of repetitive tasks such as setting up wait times after the completion of a bead. Data has to be manually extracted and inserted into a separate hardness algorithm. Integrating these functions into Abaqus was not achieved due to limited time, resources and coding expertise.
- A more extensive series of tests of blocks to investigate the full effect of different deposition strategies such as a contour weld strategy. Additionally, investigating the hardness distribution of more complicated shapes that may lead to more strict material requirements.

6. REFERENCES

[1] DAMEN, "WORLD'S FIRST CLASS APPROVED 3D PRINTED SHIP'S PROPELLER UNVEILED," 30 November 2017. [Online]. Available: https://www.damen.com/en/news/2017/11/worlds_first_class_approved_3d_printed_ships_propeller_unveiled. [Accessed 2019].

[2] MX3D, "MX3D Bridge," JUNE 2015. [Online]. Available: https://mx3d.com/projects/mx3d-bridge/. [Accessed 2019].

[3] M. Cotteleer and J. Joyce, "3D opportunity: Additive manufacturing paths to performance, innovation, and growth," *Deloitte Review,* vol. 14, pp. 5-19, 2014.

[4] D. Ding, Z. Pan, D. Cuiuri and H. Li, "Wire-feed additive manufacturing of metal components: technologies,developments and future interests," *International Journal of Advanced Manufacturing Technology,* pp. 465-481, October 2015.

[5] S. W. Williams, F. Martina, A. C. Addison, J. Ding, G. Pardal and P. Colegrove, "Wire + Arc Additive Manufacturing," *Materials Science and Technology ,* vol. 32, no. 7, pp. 641-647, 2016.

[6] P. PM, R. NV and D. SG, "Slicing procedures in layered manufacturing: a review," *Rapid Prototyping Journal,* vol. 9, no. 5, pp. 274-288, 2003.

[7] Y. Yang, J.-b. Lu, Z.-Y. Luo and D. Wang, "Accuracy and density optimization in directly fabricating customized orthodontic production by selective laser melting," *Rapid Prototyping Journal,* vol. 18, no. 6, pp. 482-489, 2012 .

[8] S. K. Vock, A. Kirchner, T. Weißgärber and B. Bernd Kieback, "Powders for powder bed fusion: a review," *Progress in Additive Manufacturing,* vol. 4, p. 383–397, 2019.

[9] S. L. Sing, . J. An and W. Y. Yeong, "Laser and electron-beam powder-bed additive manufacturing of metallic implants: A review on processes, materials and designs," *Journal of Orthopaedic Research,* vol. 43, no. 3, pp. 369-385, 2016.

[10] J. P. Pragana, P. Pombinha and V. R. Duarte, "Influence of processing parameters on the density of 316L stainless steel parts manufactured through laser powder bed fusion," *Proc IMechE Part B:,* vol. 234, no. 9, p. 1246–1257, 2020.

[11] P. A. Kobryn and S. L. Semiatin, "Mechanical properties of laser-deposited Ti-6Al-4V.," in *2001 International Solid Freeform Fabrication Symposium*, 2001.

[12] I. Gibson, D. Rosen and B. Stucker, "Powder Bed Fusion Processes," in *Additive Manufacturing Technologies - 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing, second edition*, Springer , 2015, pp. 107-145.

[13] I. Agote, "WAAM: UNA OPORTUNIDAD PARA LOS FABRICANTES DE MÁQUINA," in *Bilbao*, 2014.

[14] C. Jenney and A. O'Brien, Welding handbook, Vol. 1: welding science and technology., Miami, Florida: American Welding Society, 2001.

[15] L. Yongzhe , H. Xing, H. Imre and . Z. Guangjun, "GMAW-based additive manufacturing of inclined multi-layer multi-bead parts with flat-position deposition," *Journal of Materials Processing Tech.,* vol. 262, p. 359–371, 2018.

[16] A. Nickel, D. Barnett and F. Prinz, "Thermal stresses and deposition patterns in layered manufacturing," *Materials Science and Engineering: A,* vol. 317, pp. 59-64, 2001.

[17] S. WILLIAMS, "WAAMMAT Documents," 13 July 2016. [Online]. Available: https://waammat.com/documents/s-williams-large-scale-metal-wire-arc-additive-manufacturing-of-structural-engineering-parts. [Accessed 2019].

[18] G. den Ouden and M. Hermans, "Welding Technology," in *Welding Technology*, Delft, VSSD, 2009, pp. 25-38.

[19] "CMT: Cold Metal Transfer," digitalweldingsolutions, [Online]. Available: http://www.digitalweldingsolutions.com/CMT.pdf. [Accessed 27 5 2019].

[20] S. Selvi, A. Vishvaksenan and R. E, "Cold metal transfer (CMT) technology - An overview," *Defense Technology,* vol. 14, no. 1, pp. 28-44, Februari 2018.

[21] D. Yang, G. Wang and G. Zhang, "Thermal analysis for single-pass multi-layer GMAW based additive manufacturing using infrared thermography," *Journal of Materials Processing Technology,* vol. 244, pp. 215-224, June 2017.

[22] J. Xiong, Y. Lei and R. Li, "Finite element analysis and experimental validation of thermal behavior for thin-walled parts in GMAW-based additive manufacturing with various substrate preheating temperatures," *Applied Thermal Engineering,* vol. 126, pp. 43-52, 2017.

[23] Z. Li, C. Liu, T. Xu, D. Wang, J. Lu, S. Ma and H. Fan, "Reducing arc heat input and obtaining equiaxed grains by hot-wire method during arc additive manufacturing titanium alloy," *Materials Science and Engineering: A,* vol. 742, pp. 287-294, 2019.

[24] L. Wang, J. Xue and Q. Wang, "Correlation between arc mode, microstructure, and mechanical properties during wire arc additive manufacturing of 316L stainless steel," *Materials Science and Engineering: A,* vol. Volume 751, pp. 183-190, 2019.

[25] Q. Wu, Z. Ma, G. Chen, C. Liu, D. Ma and S. Ma, "Obtaining fine microstructure and unsupported overhangs by low heat input pulse arc additive manufacturing," *Journal of Manufacturing Processes,* vol. 27, pp. 198-206, 2017.

[26] C. Haden, G. Zeng, C. Ruhl, F. Carter, B. Krick and D. Harlow, "Wire and arc additive manufactured steel: Tensile and wear properties," *Additive Manufacturing,* vol. 16, pp. 115-123, 2017.

[27] E. Aldalur, F. Veiga, A. Suárez, J. Bilbao and A. Lamikiz, "High deposition wire arc additive manufacturing of mild steel: Strategies and heat input effect on microstructure and mechanical propertie," *Journal of Manufacturing Processes,* vol. 58, pp. 615-626, 2020.

[28] C. Seles, M. Peric and Z. Tonkovic, "Numerical simulation of a welding process using a prescribed temperature approach," *Journal of Constructional Steel Research,* vol. 145, pp. Pages 49-57, 2018.

[29] P. Collins, D. Brice, P. Samimi, I. Ghamarian and H. F. Fraser, "Microstructural Control of Additively Manufactured Metallic Materials," *Annual Review of Materials Research,* no. 46, pp. 63-91., 2016.

[30] "Ch13 Weld Microstructures," in *Steels: Microstructure and Properties (Fourth edition)*, Butterworth-Heinemann , 2017 , pp. 377-400.

[31] C. DAVIS and J. KING, "Cleavage Initiation in the Intercritically Reheated Coarse-Grained Heat-Affected Zone: Part I. Fractographic Evidence," *Metallurgical and Materials Transactions A,* vol. 25, no. 3, p. 563–573, March 1994, Volume 25, Issue 3, pp 563–573 | Cite as 1994.

[32] A. Rose and H. Hougardy, Atlas of steels heattreatment, Düsseldorf: Verlag Stahleisen, 1972.

[33] E. Pavlina and C. V. Tyne, "Correlation of Yield Strength and Tensile Strength with Hardness for Steel," *Journal of Materials Engineering and Performance ,* pp. 888-893, December 2008.

[34] Dassault System, " Three-dimensional solid element library," 11 April 2007. [Online]. Available: http://130.149.89.49:2080/v6.7/books/usb/default.htm?startat=pt06ch22s01ael03.html. [Accessed 23 8 2023].

[35] L.-E. LINDGREN, H. RUNNEMALM and M. O. NÄSSTRÖM, "SIMULATION OF MULTIPASS WELDING OF A THICK PLATE," *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING,* vol. 44, pp. 1301-1316, 1999.

[36] A. Lundbäck and L.-E. Lindgren, "Modelling of metal deposition," *Finite Elements in Analysis and Design,* vol. 47, pp. 1169-1177, 2011.

[37] P. Michaleris, "Modeling metal deposition in heat transfer analyses of additive manufacturing processes," *Finite Elements in Analysis and Design,* vol. 86, pp. 51-60, 2014.

[38] "SIMULIA," Dassault Systèmes SIMULIA Corporation, [Online]. Available: https://www.3ds.com/products-services/simulia/. [Accessed 29 5 2019].

[39] X. Wang, "Investigation of heat transfer and fluid flow in activating TIG welding by numerical modeling," *Applied Thermal Engineering,* vol. 113, pp. 27-35, 2017.

[40] J. Goldak, A. Chakravarti and M. Bibby, "A new finite element model for welding heat sources," *Metallurgical Transactions B,* vol. 15, no. 2, p. 299–305, 1984.

[41] Shubert, M; Pandheeradi, M; Arnold, F; Habura, C;, "An Abaqus Extension for Welding Simulations," in *SIMULIA Customer Conference* , 2010.

[42] D. Hodgson, C. Gill, B. Pellerreau, P. Hurrell and A. Mark, "Simulation of multi-pass welds using abaqus weld GUI and comparison with experimental results," Materials Performance Centre, The Mill, University of Manchester, Derby, 2011.

[43] "JMAT PRO," [Online].

[44] A. Mandal and R. S. Parmar, "Numerical Modeling of Pulse MIG Welding," *ISIJ International,* vol. 47, no. 10, p. 1485–1490, 2007.

[45] B. SMOLJAN, "NUMERICAL SIMULATION OF AS-QUENCHEDHARDNESS IN A STEEL SPECIMEN OF COMPLEX FORM," *COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING,* vol. 14, pp. 277-285, 1998.

[46] M. ecker, C. Jordan, S. Lachhander, M. A. and M. Renauld, "Prediction and Measurement of Phase Transformations, Phase-Dependent Properties and Residual Stresses in Steels," KAPL (Knolls Atomic Power Laboratory), Niskayuna, NY, 2005.

[47] W. Zuidema, "Investigation of the coarse grained heat-affected zone microstructure and hardness of multipass X65 linepipe steel," Delft University of Technology, Delft, 2015.

[48] W. Zuidema, "Investigation of the coarse grained heat-affected zone microstructure and hardness of multipass welded X65 linepipe steel," Delft University of Technology, Delft, 2016.

[49] D. STEFANESCU, "Physical Properties of Cast Iron," in *Iron castings engineering handbook*, Des Plaines, American Foundry Society, 2003, p. 25.

[50] *OK ARISTOROD 12.50 sheet.*

[51] E. Pavlina and C. C.J. Van Tyne, "Correlation of Yield Strength and Tensile Strength with Hardness for Steels," *Journal of Materials Engineering and Performance,* pp. 888-893, December 2008.

[52] V. Mishra, A. Babu, R. Schreurs, K. Wu, M. Hermans and C. Ayas, "Microstructure estimation and validation of ER110S-G steel structures produced by wire and arc additive manufacturing," in *Journal of Materials Research and Technology*, Elsevier, 2023, pp. 3579-3601.

[53] d. chachra, "material hierarchies and-systems - some futures for material debbie chachra," [Online]. Available: https://optional.is/required/2020/03/04/material-hierarchies-and-systems-some-futures-for-material-debbie-chachra/. [Accessed 10 8 2020].

[54] I. S. Organization, *ISO_6507_1: Vickers Hardness Test,* 2005.

[55] P. S. Almeida and S. W. , "Innovative process model of Ti–6Al–4V additive layer manufacturing using cold metal transfer (CMT).," in *Proceedings of the twenty-first annual international solid freeform fabrication symposium, University of Texas at Austin*, Austin, TX, USA., 2010.

[56] European Committee for Iron and Steel Standardisation, *EN 10025,* 2004.

[57] H. Geng, J. Li and X. J., "Geometric Limitation and Tensile Properties of Wire and Arc Additive Manufacturing 5A06 Aluminum Alloy Parts," *Journal of Materials Engineering and Performance,* vol. Volume 26, no. Issue 2, p. 621–629, 2017.

[58] J. Ding, P. Colegrove, J. Mehnen, S. Ganguly, P. Sequeira Almeida, F. Wang and S. Williams, "Thermo-mechanical analysis of wire and arc additive layer manufacturing process on large multi-layer parts," *Comp Mater Sci,* vol. 50, no. 12, pp. 3315-3322, 2011.

[59] C. Haden, G. Zeng, F. C. III, C. Ruhl, B. Krick and D. Harlow, "Wire and arc additive manufactured steel: Tensile and wear properties," *Additive Manufacturing ,* vol. 16, p. 115–123, 2017.

[60] F. Montevecchia, G. Venturinia, A. Scippaa and G. Campatellia, "Finite element modelling of Wire-Arc-Additive-Manufacturing process," *Procedia CIRP,* vol. 55, p. 109 – 114, 2016.

[61] M. Agnani, "Wire and Arc Additive Manufacturing of Manganese Aluminium," Technical Unviersity Delft, 2017.

[62] "Finite element analysis and experimental validation of thermal behavior for thin-walled parts in GMAW-based additive manufacturing with various substrate preheating temperatures," *Applied Thermal Engineering,* vol. 126, pp. 43-52, 2017.

APPENDIX A: PYHON CODE

CRProgramNewDB (base file)

```
rom abaqusConstants import *
from abaqusGui import *
from kernelAccess import mdb, session
import os

thisPath = os.path.abspath(__file__)
thisDir = os.path.dirname(thisPath)


###########################################################################
# Class definition
###########################################################################

class CRprogramNewDB(AFXDataDialog):

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def __init__(self, form):

        # Construct the base class.
        #

        #defines main box
        AFXDataDialog.__init__(self, form, 'Coolrate App',
            self.OK|self.APPLY|self.CANCEL, DIALOG_ACTIONS_SEPARATOR)

        #defines OK button
        okBtn = self.getActionButton(self.ID_CLICKED_OK)
        okBtn.setText('OK')

        #defines apply button
        applyBtn = self.getActionButton(self.ID_CLICKED_APPLY)
        applyBtn.setText('Apply')

        #defines group box for icon
        GroupBox_01 = FXGroupBox(p=self, text='Basic Principle',
opts=FRAME_GROOVE|LAYOUT_FILL_X)
        #location of icon
        fileName = os.path.join(thisDir, 'icon.png')
        #create icon objeect
        icon = afxCreatePNGIcon(fileName)
        FXLabel(p=GroupBox_01, text='', ic=icon)
        #create tab book
        TabBook_1 = FXTabBook(p=self, tgt=None, sel=0,
            opts=TABBOOK_NORMAL,
            x=0, y=0, w=0, h=0, pl=DEFAULT_SPACING, pr=DEFAULT_SPACING,
            pt=DEFAULT_SPACING, pb=DEFAULT_SPACING)
```

```
        ##################################################
        #TAB 1
        ##################################################
        tabItem = FXTabItem(p=TabBook_1, text='Required Input', ic=None,
opts=TAB_TOP_NORMAL,
            x=0, y=0, w=0, h=0, pl=6, pr=6, pt=DEFAULT_PAD, pb=DEFAULT_PAD)
        TabItem_1 = FXVerticalFrame(p=TabBook_1,
            opts=FRAME_RAISED|FRAME_THICK|LAYOUT_FILL_X,
            x=0, y=0, w=0, h=0, pl=DEFAULT_SPACING, pr=DEFAULT_SPACING,
            pt=DEFAULT_SPACING, pb=DEFAULT_SPACING, hs=DEFAULT_SPACING,
vs=DEFAULT_SPACING)
        l = FXLabel(p=TabItem_1, text='Here you choose the ODB and node set to
perform the calculation on. ', opts=JUSTIFY_LEFT)
        GroupBox_11 = FXGroupBox(p=TabItem_1, text='ODB and nodeset used',
opts=FRAME_GROOVE|LAYOUT_FILL_X|LAYOUT_FILL_Y)
        frame = AFXVerticalAligner(GroupBox_11, 0, 0,0,0,0, 0,0,0,0)
        #
        # ODB combo box
        #
        self.RootComboBox_2 = AFXComboBox(p=frame, ncols=0, nvis=1,
text='ODB:', tgt=form.odbNameKw, sel=0)
        self.RootComboBox_2.setMaxVisible(10)
        msgCount = 595
        form.odbNameKw.setTarget(self)
        form.odbNameKw.setSelector(AFXDataDialog.ID_LAST+msgCount)
        msgHandler = str(self.__class__).split('.')[-1] + '.onNodesetsChanged'
        exec('FXMAPFUNC(self, SEL_COMMAND, AFXDataDialog.ID_LAST+%d, %s)' %
(msgCount, msgHandler) )
        #
        # Nodesets combo box
        #
        self.ComboBox_2 = AFXComboBox(p=frame, ncols=0, nvis=1, text='Node
set:', tgt=form.nodeSetNameKw, sel=0)
        self.ComboBox_2.setMaxVisible(10)

        self.form = form

        if False:
            pickHf = FXHorizontalFrame(p=GroupBox_11, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0, hs=DEFAULT_SPACING, vs=DEFAULT_SPACING)
            # Note: Set the selector to indicate that this widget should not
be
            #       colored differently from its parent when the 'Color layout
managers'
            #       button is checked in the RSG Dialog Builder dialog.
            pickHf.setSelector(99)
```

```
            label = FXLabel(p=pickHf, text='Pick Nodes' + ' (None)', ic=None,
opts=LAYOUT_CENTER_Y|JUSTIFY_LEFT)
            pickHandler = CRprogramNewDBPickHandler(form, form.PickNodeKw,
'Pick an entity', NODES, MANY, label)
            icon = afxGetIcon('select', AFX_ICON_SMALL )
            FXButton(p=pickHf, text='\tPick Items in Viewport', ic=icon,
tgt=pickHandler, sel=AFXMode.ID_ACTIVATE,
                opts=BUTTON_NORMAL|LAYOUT_CENTER_Y, x=0, y=0, w=0, h=0, pl=2,
pr=2, pt=1, pb=1)

            FXCheckButton(p=pickHf, text='Override?',
tgt=form.FlagPickOverrideKw, sel=0)

        #
        #CCT File Selector
        #

        if isinstance(self, FXHorizontalFrame):
            FXVerticalSeparator(p=TabItem_1, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        else:
            FXHorizontalSeparator(p=TabItem_1, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)


        GroupBox_12 = FXGroupBox(p=TabItem_1, text='Calculation properties',
opts=FRAME_GROOVE|LAYOUT_FILL_X|LAYOUT_FILL_Y)
        fileHandler = CRprogramNewDBFileHandler(form, 'fileNameCCT', 'Data
files(*.dat)\nAll files (*)')
        fileTextHf = FXHorizontalFrame(p=GroupBox_12, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0, hs=DEFAULT_SPACING, vs=DEFAULT_SPACING)
        # Note: Set the selector to indicate that this widget should not be
        #       colored differently from its parent when the 'Color layout
managers'
        #       button is checked in the RSG Dialog Builder dialog.
        fileTextHf.setSelector(99)
        AFXTextField(p=fileTextHf, ncols=12, labelText='CCT file
name                ', tgt=form.fileNameCCTKw, sel=0,
            opts=AFXTEXTFIELD_STRING|LAYOUT_CENTER_Y)
        icon = afxGetIcon('fileOpen', AFX_ICON_SMALL )
        FXButton(p=fileTextHf, text='   Select File\nFrom Dialog', ic=icon,
tgt=fileHandler, sel=AFXMode.ID_ACTIVATE,
            opts=BUTTON_NORMAL|LAYOUT_CENTER_Y, x=0, y=0, w=0, h=0, pl=1,
pr=1, pt=1, pb=1)

        if isinstance(self, FXHorizontalFrame):
            FXVerticalSeparator(p=TabItem_1, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
```

```
        else:
            FXHorizontalSeparator(p=TabItem_1, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)

        GroupBox_13 = FXGroupBox(p=TabItem_1, text='Options',
opts=FRAME_GROOVE|LAYOUT_FILL_X)
        #button 1
        FXCheckButton(p=GroupBox_13, text='Calculate only last peaks?
(faster)', tgt=form.FlagLastresultKw, sel=0)
        #button 2
        FXCheckButton(p=GroupBox_13, text='Save fields to ODB',
tgt=form.FlagSavefieldKw, sel=0)









        ##################################################
        #TAB  2
        ##################################################
        tabItem = FXTabItem(p=TabBook_1, text='Calculation', ic=None,
opts=TAB_TOP_NORMAL,
            x=0, y=0, w=0, h=0, pl=6, pr=6, pt=DEFAULT_PAD, pb=DEFAULT_PAD)
        TabItem_2 = FXVerticalFrame(p=TabBook_1,
            opts=FRAME_RAISED|FRAME_THICK|LAYOUT_FILL_X,
            x=0, y=0, w=0, h=0, pl=DEFAULT_SPACING, pr=DEFAULT_SPACING,
            pt=DEFAULT_SPACING, pb=DEFAULT_SPACING, hs=DEFAULT_SPACING,
vs=DEFAULT_SPACING)
        l = FXLabel(p=TabItem_2, text='Here you chose the options for the
calculation', opts=JUSTIFY_LEFT)

        #creates box to group text fields
        if isinstance(self, FXHorizontalFrame):
            FXVerticalSeparator(p=TabItem_2, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        else:
            FXHorizontalSeparator(p=TabItem_2, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        GroupBox_21 = FXGroupBox(p=TabItem_2, text='T8/5 Options',
opts=FRAME_GROOVE|LAYOUT_FILL_X)

        VAligner_5 = AFXVerticalAligner(p=GroupBox_21, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0)
```

```
        AFXTextField(p=VAligner_5, ncols=12, labelText='Peak
Temperature    (C)', tgt=form.peaktemperatureKw, sel=0)
        AFXTextField(p=VAligner_5, ncols=12, labelText='Upper
Temperature  (C)', tgt=form.uppertemperatureKw, sel=0)
        AFXTextField(p=VAligner_5, ncols=12, labelText='Lower
Temperature  (C)', tgt=form.lowertemperatureKw, sel=0)

        #creates box to group buttons
        if isinstance(self, FXHorizontalFrame):
            FXVerticalSeparator(p=TabItem_2, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        else:
            FXHorizontalSeparator(p=TabItem_2, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        GroupBox_22 = FXGroupBox(p=TabItem_2, text='Data Handling Options',
opts=FRAME_GROOVE|LAYOUT_FILL_X)
        #button 3
        FXCheckButton(p=GroupBox_22, text='Linear Interpolate CCT?',
tgt=form.FlagInterpolateKw, sel=0)
        #button 4
        FXCheckButton(p=GroupBox_22, text='Output Positive Coolrate (absolute
values)?', tgt=form.FlagAbsoluteKw, sel=0)
        #button 5
        FXCheckButton(p=GroupBox_22, text='Ignore CR=0 in HV calculation',
tgt=form.FlagHVzeroKw, sel=0)


        ##################################################
        #TAB 3
        ##################################################
        tabItem = FXTabItem(p=TabBook_1, text='Field Output', ic=None,
opts=TAB_TOP_NORMAL,
            x=0, y=0, w=0, h=0, pl=6, pr=6, pt=DEFAULT_PAD, pb=DEFAULT_PAD)
        TabItem_3 = FXVerticalFrame(p=TabBook_1,
            opts=FRAME_RAISED|FRAME_THICK|LAYOUT_FILL_X,
            x=0, y=0, w=0, h=0, pl=DEFAULT_SPACING, pr=DEFAULT_SPACING,
            pt=DEFAULT_SPACING, pb=DEFAULT_SPACING, hs=DEFAULT_SPACING,
vs=DEFAULT_SPACING)
        l = FXLabel(p=TabItem_3, text='Here you chose what the plugin should
output. ', opts=JUSTIFY_LEFT)
        GroupBox_31 = FXGroupBox(p=TabItem_3, text='Field Output Choice',
opts=FRAME_GROOVE|LAYOUT_FILL_X|LAYOUT_FILL_Y)
        #add radiobutton 1
        FXRadioButton(p=GroupBox_31, text='Output Coolrate',
tgt=form.ChoiceOutputKw1, sel=1834)
        #add radiobutton 2
        FXRadioButton(p=GroupBox_31, text='Output Hardness',
tgt=form.ChoiceOutputKw1, sel=1835)
        #add radiobutton 3
```

```
        FXRadioButton(p=GroupBox_31, text='Output Both',
tgt=form.ChoiceOutputKw1, sel=1836)

        #new groupbox with aligner
        if isinstance(self, FXHorizontalFrame):
            FXVerticalSeparator(p=TabItem_3, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        else:
            FXHorizontalSeparator(p=TabItem_3, x=0, y=0, w=0, h=0, pl=2, pr=2,
pt=2, pb=2)
        GroupBox_32= FXGroupBox(p=TabItem_3, text='Field Location Choice',
opts=FRAME_GROOVE|LAYOUT_FILL_X|LAYOUT_FILL_Y)
        VAligner_4 = AFXVerticalAligner(p=GroupBox_32, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0)
        #text fields
        AFXTextField(p=VAligner_4, ncols=12, labelText='Output CR fieldname',
tgt=form.OutputCRfieldnameKw, sel=0)
        AFXTextField(p=VAligner_4, ncols=12, labelText='Output HV fieldname',
tgt=form.OutputHVfieldnameKw, sel=0)
        AFXTextField(p=VAligner_4, ncols=12, labelText='Output Step Index',
tgt=form.OutputStepKw, sel=0)
        AFXTextField(p=VAligner_4, ncols=12, labelText='Output Frame Index',
tgt=form.OutputFrameKw, sel=0)




        ################################################
        # TAB END
        ################################################


        #creates group box for output file options
        GroupBox_02 = FXGroupBox(p=self, text='Output File Options',
opts=FRAME_GROOVE|LAYOUT_FILL_X)
        #creates label for group box
        l = FXLabel(p=GroupBox_02, text='Here you chose the text file the
output should be written to. ', opts=JUSTIFY_LEFT)




        #Creates file input and button for file search
        HFrame_1 = FXHorizontalFrame(p=GroupBox_02, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0)
        fileHandler = CRprogramNewDBFileHandler(form, 'fileNameOutput', 'Text
Files (*txt)')
```

```
        fileTextHf = FXHorizontalFrame(p=GroupBox_02, opts=0, x=0, y=0, w=0,
h=0,
            pl=0, pr=0, pt=0, pb=0, hs=DEFAULT_SPACING, vs=DEFAULT_SPACING)
        # Note: Set the selector to indicate that this widget should not be
        #       colored differently from its parent when the 'Color layout
managers'
        #       button is checked in the RSG Dialog Builder dialog.
        fileTextHf.setSelector(99)
        AFXTextField(p=fileTextHf, ncols=20, labelText='Log File:',
tgt=form.fileNameOutputKw, sel=0,
            opts=AFXTEXTFIELD_STRING|LAYOUT_CENTER_Y)
        icon = afxGetIcon('fileOpen', AFX_ICON_SMALL )
        FXButton(p=fileTextHf, text='   Select File\nFrom Dialog', ic=icon,
tgt=fileHandler, sel=AFXMode.ID_ACTIVATE,
            opts=BUTTON_NORMAL|LAYOUT_CENTER_Y, x=0, y=0, w=0, h=0, pl=1,
pr=1, pt=1, pb=1)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def show(self):

        AFXDataDialog.show(self)

        # Select the current ODB, if there is one
        #
        names = session.odbs.keys()
        names.sort()
        objectType = getCurrentContext()['objectType']
        if objectType == 'ODB':
            self.currentOdbName = getCurrentContext()['objectPath']
        elif names:
            self.currentOdbName = names[0]
        else:
            self.currentOdbName = ''

        self.form.odbNameKw.setValue(self.currentOdbName)

        session.odbs.registerQuery(self.updateRootComboBox_2Odbs)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def hide(self):

        AFXDataDialog.hide(self)

        session.odbs.unregisterQuery(self.updateRootComboBox_2Odbs)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def updateRootComboBox_2Odbs(self):

        # Update the names in the ODB combo
```

```
        #
        self.RootComboBox_2.clearItems()
        names = session.odbs.keys()
        names.sort()
        for name in names:
            self.RootComboBox_2.appendItem(name)

        if names:
            if not self.form.odbNameKw.getValue() in names:
                self.form.odbNameKw.setValue( names[0] )
        else:
            self.form.odbNameKw.setValue('')

        self.onNodesetsChanged(None, None, None)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def onNodesetsChanged(self, sender, sel, ptr):

        odbName = self.form.odbNameKw.getValue()

        # Update the names in the Nodesets combo
        #
        self.ComboBox_2.clearItems()
        if odbName:
            names = session.odbs[odbName].rootAssembly.nodeSets.keys()
            names.sort()
            for name in names:
                self.ComboBox_2.appendItem(name)

            if names:
                if not self.form.nodeSetNameKw.getValue() in names:
                    self.form.nodeSetNameKw.setValue( names[0] )
            else:
                self.form.nodeSetNameKw.setValue('')

        else:
            self.form.nodeSetNameKw.setValue('')

        self.resize( self.getDefaultWidth(), self.getDefaultHeight() )

        return 1



###########################################################################
# Class definition
###########################################################################

class CRprogramNewDBPickHandler(AFXProcedure):
```

```
        count = 0

        #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
        def __init__(self, form, keyword, prompt, entitiesToPick,
numberToPick, label):

                self.form = form
                self.keyword = keyword
                self.prompt = prompt
                self.entitiesToPick = entitiesToPick # Enum value
                self.numberToPick = numberToPick # Enum value
                self.label = label
                self.labelText = label.getText()

                AFXProcedure.__init__(self, form.getOwner())

                CRprogramNewDBPickHandler.count += 1
                self.setModeName('CRprogramNewDBPickHandler%d' %
(CRprogramNewDBPickHandler.count) )

        #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
        def getFirstStep(self):

                return  AFXPickStep(self, self.keyword, self.prompt,
                    self.entitiesToPick, self.numberToPick,
sequenceStyle=TUPLE)

        #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~
        def getNextStep(self, previousStep):

                self.label.setText( self.labelText.replace('None', 'Picked') )
                return None

        def deactivate(self):

            AFXProcedure.deactivate(self)
            if  self.numberToPick == ONE and self.keyword.getValue() and
self.keyword.getValue()[0]!='<':
                sendCommand(self.keyword.getSetupCommands() +
'\nhighlight(%s)' % self.keyword.getValue() )



###########################################################################
# Class definition
```

```
###########################################################################

class CRprogramNewDBFileHandler(FXObject):

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def __init__(self, form, keyword, patterns='*'):

        self.form = form
        self.patterns = patterns
        self.patternTgt = AFXIntTarget(0)
        exec('self.fileNameKw = form.%sKw' % keyword)
        self.readOnlyKw = AFXBoolKeyword(None, 'readOnly',
AFXBoolKeyword.TRUE_FALSE)
        FXObject.__init__(self)
        FXMAPFUNC(self, SEL_COMMAND, AFXMode.ID_ACTIVATE,
CRprogramNewDBFileHandler.activate)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def activate(self, sender, sel, ptr):

       fileDb = AFXFileSelectorDialog(getAFXApp().getAFXMainWindow(), 'Select
a File',
           self.fileNameKw, self.readOnlyKw,
           AFXSELECTFILE_ANY, self.patterns, self.patternTgt)
       fileDb.setReadOnlyPatterns('*.odb')
       fileDb.create()
       fileDb.showModal()
```

CRProgramNew_plugin (plugin file)

```
from abaqusGui import *
from abaqusConstants import ALL
import osutils, os


###########################################################################
# Class definition
###########################################################################

class CRprogramNew_plugin(AFXForm):

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def __init__(self, owner):

        # Construct the base class.
        #
        AFXForm.__init__(self, owner)
        self.radioButtonGroups = {}
```

```
        #defines what happens when you press OK.
        self.cmd = AFXGuiCommand(mode=self, method='ActionCoolRateCalcNew',
            objectName='ActionCoolRateCalcNew', registerQuery=False)


        #self . [internal name] = [datatype] ([self.command], [name of
variable passed to method],[ register], [defaultvalue])

        #object keywords:
        #default nodeset:
        pickedDefault = ''
        self.PickNodeKw = AFXObjectKeyword(self.cmd, 'PickNode', TRUE,
pickedDefault)
        #stringkeywords:
        self.odbNameKw = AFXStringKeyword(self.cmd, 'odbName', True)
        self.nodeSetNameKw = AFXStringKeyword(self.cmd, 'nodeSetName', True)
        self.fileNameCCTKw = AFXStringKeyword(self.cmd, 'fileNameCCT', True,
'')
        self.fileNameOutputKw = AFXStringKeyword(self.cmd, 'fileNameOutput',
True, 'DefaultOutput.txt')
        self.OutputCRfieldnameKw = AFXStringKeyword(self.cmd, 'CRfieldname',
True, 'CR11')
        self.OutputHVfieldnameKw = AFXStringKeyword(self.cmd, 'HVfieldname',
True, 'HV11')


        #int Keywords:
        self.peaktemperatureKw = AFXIntKeyword(self.cmd, 'peaktemperature',
True, 1800)
        self.uppertemperatureKw = AFXIntKeyword(self.cmd, 'uppertemperature',
True, 800)
        self.lowertemperatureKw = AFXIntKeyword(self.cmd, 'lowertemperature',
True, 500)
        self.OutputStepKw = AFXIntKeyword(self.cmd, 'OutputStep', True, -1)
        self.OutputFrameKw = AFXIntKeyword(self.cmd, 'OutputFrame', True, -1)


        #radiobutton:

        if not self.radioButtonGroups.has_key('ChoiceOutput'):
            self.ChoiceOutputKw1 = AFXIntKeyword(None, 'ChoiceOutputDummy',
True)
            self.ChoiceOutputKw2 = AFXStringKeyword(self.cmd, 'ChoiceOutput',
True)
            self.radioButtonGroups['ChoiceOutput'] = (self.ChoiceOutputKw1,
self.ChoiceOutputKw2, {})
        self.radioButtonGroups['ChoiceOutput'][2][1834] = 'Output Coolrate'
        if not self.radioButtonGroups.has_key('ChoiceOutput'):
```

```
            self.ChoiceOutputKw1 = AFXIntKeyword(None, 'ChoiceOutputDummy',
True)
            self.ChoiceOutputKw2 = AFXStringKeyword(self.cmd, 'ChoiceOutput',
True)
            self.radioButtonGroups['ChoiceOutput'] = (self.ChoiceOutputKw1,
self.ChoiceOutputKw2, {})
        self.radioButtonGroups['ChoiceOutput'][2][1835] = 'Output Hardness'
        if not self.radioButtonGroups.has_key('ChoiceOutput'):
            self.ChoiceOutputKw1 = AFXIntKeyword(None, 'ChoiceOutputDummy',
True)
            self.ChoiceOutputKw2 = AFXStringKeyword(self.cmd, 'ChoiceOutput',
True)
            self.radioButtonGroups['ChoiceOutput'] = (self.ChoiceOutputKw1,
self.ChoiceOutputKw2, {})
        self.radioButtonGroups['ChoiceOutput'][2][1836] = 'Output Both'
        self.ChoiceOutputKw1.setValue(1836)

        #flags:
        self.FlagLastresultKw = AFXBoolKeyword(self.cmd, 'FlagLastresult',
AFXBoolKeyword.TRUE_FALSE, True, True)
        self.FlagSavefieldKw = AFXBoolKeyword(self.cmd, 'FlagSavefield',
AFXBoolKeyword.TRUE_FALSE, True, True)
        self.FlagInterpolateKw=AFXBoolKeyword(self.cmd, 'FlagInterpolate',
AFXBoolKeyword.TRUE_FALSE, True, True)

        self.FlagAbsoluteKw=AFXBoolKeyword(self.cmd, 'FlagAbsolute',
AFXBoolKeyword.TRUE_FALSE, True, True)
        self.FlagHVzeroKw=AFXBoolKeyword(self.cmd, 'FlagHVzero',
AFXBoolKeyword.TRUE_FALSE, True, True)


        self.FlagPickOverrideKw=AFXBoolKeyword(self.cmd, 'FlagPickOverride',
AFXBoolKeyword.TRUE_FALSE, True, False)









    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def getFirstDialog(self):

        import cRprogramNewDB
        return cRprogramNewDB.CRprogramNewDB(self)

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
    def doCustomChecks(self):

        # Try to set the appropriate radio button on. If the user did
        # not specify any buttons to be on, do nothing.
        #
        for kw1,kw2,d in self.radioButtonGroups.values():
            try:
                value = d[ kw1.getValue() ]
                kw2.setValue(value)
            except:
                pass
        return True

    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def okToCancel(self):

        # No need to close the dialog when a file operation (such
        # as New or Open) or model change is executed.
        #
        return False

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Register the plug-in
#
thisPath = os.path.abspath(__file__)
thisDir = os.path.dirname(thisPath)

toolset = getAFXApp().getAFXMainWindow().getPluginToolset()
toolset.registerGuiMenuButton(
    buttonText='CRprogramNew',
    object=CRprogramNew_plugin(toolset),
    messageId=AFXMode.ID_ACTIVATE,
    icon=None,
    kernelInitString='import ActionCoolRateCalcNew',
    applicableModules=ALL,
    version='N/A',
    author='N/A',
    description='N/A',
    helpUrl='N/A'
)
```

Actioncoolratecalcnew (Main calculation program)

```
import os
from abaqus import *
from abaqusConstants import *
from odbAccess import*
import time
import sys

##############################################################################
# Class definition
# Linear Interpolation
##############################################################################

def ActionInterpolate(X0,Y0,X1,Y1,X):
    Y=(Y0)+(X-X0)*((Y1-Y0)/(X1-X0))
    return Y

##############################################################################
# Class definition
# Reads a CCT File, returns useable data
##############################################################################

def ActionReadCCTFile(filenameCR):
    print("Start read CCT")
    flag1=False
    CCTData=[]
    #retrieve file
    cpFileCR=open(filenameCR,"r")
    #convert to a list of lines
    CCTLines=cpFileCR.readlines()
    #we have no further use of the file, so close it
    cpFileCR.close()
    #check each line, find the tabs,then get the first and last data into
CCTDat
    for line in CCTLines:
        #print("Line: " + line)
        if flag1:
            #find each tab
            line1=line.splitlines()
            #print(["Line1: " , line1])
            splitLine=line1[0].split("\t")
            #print(["splitLine: " , splitLine])
            #append found data, also check if there arent any empty lines
            #print("len(splitLine" + str(len(splitLine)))
            if len(splitLine) > 1:
                #print([float(splitLine[0]),float(splitLine[-1])])
                CCTData.append([float(splitLine[0]),float(splitLine[-1])])
        #keep track of whenn CoolingRate starts
        if line.startswith("CoolingRate"):
```

```
            flag1=True
    print("CCT DATA: (CR, HV)")
    print(CCTData)
    print("End read CCT")
    return CCTData

##############################################################################
# Class definition
# Calculate harness based on table, no interpolation
##############################################################################

def ActionTableExact(Table,Datapoint):



    minval=[0]*len(Table)
    for i in range(len(Table)):
        minval[i]=(abs(Table[i][0]-Datapoint))

    idx=minval.index(min(minval))
    HV=(Table[idx][1])
    return HV

##############################################################################
# Class definition
# Calculate harness based on table, linear interpolation
##############################################################################

def ActionTableInterpolate(Table,Datapoint):

    minval=[0]*len(Table)
    for i in range(len(Table)):
        minval[i]=(abs(Table[i][0]-Datapoint))

    idx=minval.index(min(minval))

    print('len(minval)',len(minval) ,'idx',idx)

    #check for edge conditions idx=0 and idx=last first
    if idx==0:
        if Datapoint > Table[0][0]:
            HV=(ActionInterpolate(Table[idx][0],Table[idx][1],Table[idx+1][0],
Table[idx+1][1],Datapoint))
            print('X0=',Table[idx][0],'Y0=',Table[idx][1],',X1=',Table[idx+1][
0],'Y1=',Table[idx+1][1],'X=',Datapoint,'HV=',HV )
        elif Datapoint<=Table[0][0]:
            HV=Table[idx][1]
        else:
            print('Error ','idx',idx,'Datapoint',Datapoint)
```

```
            HV=-1


    elif idx==len(minval)-1:
        if Datapoint > Table[-1][0]:
            HV=Table[idx][1]
        elif Datapoint <= Table[-1][0]:
            HV=(ActionInterpolate(Table[idx-1][0],Table[idx-
1][1],Table[idx][0],Table[idx][1],Datapoint))
            print('X0=',Table[idx-1][0],'Y0=',Table[idx-
1][1],',X1=',Table[idx][0],'Y1=',Table[idx][1],'X=',Datapoint,'HV=',HV )
        else:
            print('Error ','idx',idx,'Datapoint',Datapoint)
            HV=-1

    else:
        if minval[idx-1] < minval[idx+1]:
            HV=(ActionInterpolate(Table[idx][0],Table[idx][1],Table[idx+1][0],
Table[idx+1][1],Datapoint))
            print('X0=',Table[idx][0],'Y0=',Table[idx][1],',X1=',Table[idx+1][
0],'Y1=',Table[idx+1][1],'X=',Datapoint,'HV=',HV )


        elif minval[idx-1] > minval[idx+1]:
            HV=(ActionInterpolate(Table[idx-1][0],Table[idx-
1][1],Table[idx][0],Table[idx][1],Datapoint))
            print('X0=',Table[idx-1][0],'Y0=',Table[idx-
1][1],',X1=',Table[idx][0],'Y1=',Table[idx][1],'X=',Datapoint,'HV=',HV )

        else:
            HV=((Table[idx][1]))

    return  HV



#Coolrate Program Old
def CoolrateCalcSimple(**CCSargs):
    import time
    print("Start Simple CR Calc")
    milestone(message='Coolrate Calculation Start')
    #print(CCSargs)



    chosenNS        = (CCSargs["chosenNS"])        #Chosen node Seet
    myodbKeys       = (CCSargs["myodbKeys"])       #Odb Keys
    cpFile5         = (CCSargs["cpFile5"])         #Log file object
    myInstance      = (CCSargs["myInstance"])      #Chosen Instancee
```

```
    Tupper          = (CCSargs["Tupper"])          #Upper Temperature for CR
calc
    Tlower          = (CCSargs["Tlower"])          #Loweer Temperature for CR
calc
    myodb           = (CCSargs["myodb"])           #Odb that is open
    CCTData         = (CCSargs["CCTData"])         #CCT data for HV calc
    FlagInterpolate = (CCSargs["FlagInterpolate"]) #Interpolate results?
    FlagAbsolute    = (CCSargs["FlagAbsolute"])    #Absolut values?
    FlagHVzero      = (CCSargs["FlagHVzero"])    #should CR=0 mean HV=0?

    print("Interpolate Hardness",FlagInterpolate)
    print("Absolute Values:",FlagAbsolute)

    cpFile5.write('\n'+'\n'+'Interpolate Hardness:'
+'\t'+str(FlagInterpolate)+'\n')
    cpFile5.write('Absolute Values:' +'\t'+str(FlagAbsolute)+'\n'+'\n')

    # gets the nodesets in an odb
    myNSets = myodb.rootAssembly.nodeSets
    #end adaptational change
    nNodes=session.odbData[myodbKeys[0]].nodeSets[chosenNS].nodes
    cpFile5.write('Chosen Node Set:' +'\t'+str(chosenNS)+'\n')

    #Preallocate data for list function
    numNodesChosen=len( nNodes[myInstance.name])
    cpFile5.write('Nodes in chosen set:' +'\t'+str(numNodesChosen)+'\n')

    flag1 = [0] * numNodesChosen
    nPeak = [0] * numNodesChosen
    tT800 = [0] * numNodesChosen
    TT800 = [0] * numNodesChosen
    tT500 = [0] * numNodesChosen
    TT500 = [0] * numNodesChosen
    CR    = [0] * numNodesChosen
    HV    = [0] * numNodesChosen

    #display the model in the viewport
    myVP=session.viewports.keys()
    objChosen=session.odbs[myodbKeys[0]]
    session.viewports[myVP[0]].setValues(displayedObject=objChosen)
    session.viewports[myVP[0]].odbDisplay.setPrimaryVariable(variableLabel='NT
11', outputPosition=NODAL, )
    session.viewports[myVP[0]].odbDisplay.display.setValues(plotState=CONTOURS
_ON_DEF)



    #for every step in ODB
    mysteps = myodb.steps.keys()
```

```
    cpFile5.write('Steps in chosen ODB:' +'\t'+str(len(mysteps))+'\n')
    for step in range(len(mysteps)):
        tElapsed=myodb.steps[mysteps[step]].totalTime #time at frame 0 of this
step
        myframes=myodb.steps[mysteps[step]].frames
        milestone(message='Retrieving Data', object='Step',
done=step+1,total=len(mysteps))

        landmark(text='step='+ str(step))

        #update the displayed part once per step
        session.viewports[myVP[0]].odbDisplay.setPrimaryVariable(variableLabel
='NT11', outputPosition=NODAL, )
        session.viewports[myVP[0]].odbDisplay.display.setValues(plotState=CONT
OURS_ON_DEF)
        session.viewports[myVP[0]].odbDisplay.setFrame(step=step, frame=0)


        #for every frame in step
        for frame in range(len(myframes)-1):

            #get the current time


            t1=tElapsed+myframes[frame  ].frameValue #Total time at frame 0 of
this step + frame time witin step.
            t2=tElapsed+myframes[frame+1].frameValue  #Total time at frame 0
of this step + frame time witin step.


            if t1 != t2:#if the two frames are not equal; this is needed to
avoid calculation complications

                #get the current frame

                mySubset1=myframes[frame  ].fieldOutputs['NT11'].getSubset(reg
ion=myNSets[chosenNS])
                mySubset2=myframes[frame+1].fieldOutputs['NT11'].getSubset(reg
ion=myNSets[chosenNS])

                for idxNode in range(len(mySubset1.values)):
                    #TT8Prev=tT800[idxNode]
                    #tT8Prev=TT800[idxNode]

                    T1=mySubset1.values[idxNode].data
                    T2=mySubset2.values[idxNode].data
                    dT=T2-T1
                    dt=t2-t1
```

```
                    if dT <  0: #if we go downhill
                        if T1>Tupper and T2<Tupper: #if 800deg is between
points, interpolate
                            tT800[idxNode]=t1+(dt)/(dT)*(Tupper-T1)
                            TT800[idxNode]=Tupper
                            flag1[idxNode]=1
                            nPeak[idxNode]=nPeak[idxNode]+1

                        elif T1==Tupper: #if 800deg is exactly a point, write
                            tT800[idxNode]=t1
                            TT800[idxNode]=Tupper
                            flag1[idxNode]=1
                            nPeak[idxNode]=nPeak[idxNode]+1

                        if flag1[idxNode]==1: #if after a proper peak
                            if T1>Tlower and T2<Tlower:#if 500deg is between
points, interpolate
                                tT500[idxNode]=t1+(dt)/(dT)*(Tlower-T1)
                                TT500[idxNode]=Tlower

                            elif T1==Tlower:      #if 500deg is exactly a
point, write
                                tT500[idxNode]=t1
                                TT500[idxNode]=T1



                            elif dT >= 0:#if we go uphill
                                if flag1[idxNode]==1: #if after a proper peak
                                    if T1>Tlower and T2<Tlower:#if 500deg is
between points, interpolate
                                        tT500[idxNode]=t1+(dt)/(dT)*(Tlower-
T1)
                                        TT500[idxNode]=Tlower

                                    elif T1==Tlower:      #if 500deg is
exactly a point, write
                                        tT500[idxNode]=t1
                                        TT500[idxNode]=T1
                                    elif frame == (len(myframes)-1) and
step==(len(mysteps)):#if something has been found but its the the last entry
                                        tT500[idxNode]=t1
                                        TT500[idxNode]=T1
    milestone(message='Hardness Calculation')
    cpFile5.write('\n'+'Final Coolrates are: ' )
    print("T85 finished, do CR/HV calc")
    #dummy for easy text writing. Brackets denote spaces to put values
    txt1="N= {} \t tT800= \t {} \t TT800= \t {} tT500= {} \t {} TT500= {}  \t
CR= \t {} \t HV= \t {}"
    CRabs=[]#in case absolute value is needed
```

```
    for idxN in range(len(TT800)):
        #milestone(message='Calculating Coolrate', object='Node',
done=idxN+1,total=len(TT800))

        if (TT800[idxN]-TT500[idxN])==0 or (tT800[idxN]-tT500[idxN] )==0:
            CR[idxN]=0
        #elif (TT8Prev[idxN]-TT500[idxN] ) / ( tT8Prev[idxN]-tT500[idxN] )>0 :
        #    CR[idxN]=(TT800[idxN]-TT500[idxN] ) / ( tT800[idxN]-tT500[idxN]
)
        else:
            CR[idxN]=(TT800[idxN]-TT500[idxN] ) / ( tT800[idxN]-tT500[idxN]
)
            #print('N= '+
str(myNSets[chosenNS].instances[0].nodes[idxN].label) + ', CR= ' +
str(CR[idxN])+' C/s')

        if FlagInterpolate:
            #if interpolation requested:
            if FlagHVzero and CR[idxN]==0:
                HV[idxN]= 0
            else:
                HV[idxN]= ActionTableInterpolate(CCTData,abs(CR[idxN]))

        else:
            #if interpolation  not requested:
            #if CR=0 then HV=0
            if FlagHVzero and CR[idxN]==0:
                HV[idxN]= 0
            else:
                HV[idxN]= ActionTableExact(CCTData,abs(CR[idxN]))

        if FlagAbsolute:
            #if absolute values requested, convert the CR values
            CR[idxN]=abs(CR[idxN])

        #if absolute values not requested, print as calculated
        print(txt1.format(myNSets[chosenNS].instances[0].nodes[idxN].label,CR[
idxN], HV[idxN]))
        cpFile5.write('\n'+'N='+
str(myNSets[chosenNS].instances[0].nodes[idxN].label) + '\t
tT800='+str(tT800[idxN])+'\t TT800= '+str(TT800[idxN])+ '\t
tT500='+str(tT500[idxN])+'\t TT500= '+str(TT500[idxN])+ '\t
CR='+str(CR[idxN])+'\t HV= '+str(HV[idxN]))

    print("return HV, CR")

    return [CR,HV]
```

```
#Coolrate Program New
def CoolrateCalcAdvanced(**CCAargs):
    print("Start Advanced Coolrate Calc")
    milestone(message='Coolrate Calculation Start')
    #print(CCAargs)


    chosenNS        = (CCAargs["chosenNS"])        #
    myodbKeys       = (CCAargs["myodbKeys"])       #
    cpFile5         = (CCAargs["cpFile5"])         #Log File Object
    myInstance      = (CCAargs["myInstance"])      #
    Tupper          = (CCAargs["Tupper"])          #
    Tlower          = (CCAargs["Tlower"])          #
    myodb           = (CCAargs["myodb"])           #
    CCTData         = (CCAargs["CCTData"])         #
    FlagInterpolate = (CCAargs["FlagInterpolate"]) #Interpolate?
    FlagAbsolute    = (CCAargs["FlagAbsolute"])    #Absolut values?


    print("Interpolate Hardness",FlagInterpolate)
    print("Absolute Values:",FlagAbsolute)

    cpFile5.write('\n'+'\n'+'Interpolate Hardness:'
+'\t'+str(FlagInterpolate)+'\n')
    cpFile5.write('Absolute Values:' +'\t'+str(FlagAbsolute)+'\n'+'\n')


    # gets the nodesets in an odb
    myNSets = myodb.rootAssembly.nodeSets
    #end adaptational change
    nNodes=session.odbData[myodbKeys[0]].nodeSets[chosenNS].nodes
    cpFile5.write('Chosen Node Set:' +'\t'+str(chosenNS)+'\n')

    #Preallocate data for list function
    numNodesChosen=len( nNodes[myInstance.name])
    cpFile5.write('Nodes in chosen set:' +'\t'+str(numNodesChosen)+'\n')

    #assign numerical trackers for detected peaks and number of peaks
    flag1 = [0]*numNodesChosen
    nPeak = [0]*numNodesChosen

    #preallocate data
    #this is a list of lists, IE one list contains a bunch of sub-lists,
namely one list per node
    #so first make an empty list that can be appeended
    tT800 = []
    TT800 = []
    tT500 = []
    TT500 = []
```

```
    CR    = []
    CRabs = []#in case absolute value is needed
    HV    = []



    #the empty list is now appended to the chosen length, namely one per node.
    for idx1 in range(numNodesChosen):
        tT800.append([])
        TT800.append([])
        tT500.append([])
        TT500.append([])
        CR.append([])
        CRabs.append([])
        HV.append([])

    #display the model in the viewport
    myVP=session.viewports.keys()
    objChosen=session.odbs[myodbKeys[0]]
    session.viewports[myVP[0]].setValues(displayedObject=objChosen)
    session.viewports[myVP[0]].odbDisplay.setPrimaryVariable(variableLabel='NT
11', outputPosition=NODAL, )
    session.viewports[myVP[0]].odbDisplay.display.setValues(plotState=CONTOURS
_ON_DEF)



    #for every step in ODB
    mysteps = myodb.steps.keys()
    cpFile5.write('Steps in chosen ODB:' +'\t'+str(len(mysteps))+'\n')
    milestone(message='Calculation Start')
    #txt2=" idxNode= \t {}\tT800= \t {} \t t800= \t {} \t T500= \t {}\t t500=
\t {}\t nPeak[idxNode]-1= \t {}"

    for step in range(len(mysteps)):
        tElapsed=myodb.steps[mysteps[step]].totalTime #time at frame 0 of this
step
        myframes=myodb.steps[mysteps[step]].frames
        milestone(message='Retrieving Data', object='Step',
done=step+1,total=len(mysteps))

        landmark(text='step='+ str(step))

        #update the displayed part once per step

        session.viewports[myVP[0]].odbDisplay.setPrimaryVariable(variableLabel
='NT11', outputPosition=NODAL, )
        session.viewports[myVP[0]].odbDisplay.display.setValues(plotState=CONT
OURS_ON_DEF)
```

```
        session.viewports[myVP[0]].odbDisplay.setFrame(step=step, frame=0)


        #for every frame in step
        for frame in range(len(myframes)-1):

            #get the current time


            t1=tElapsed+myframes[frame  ].frameValue #Total time at frame 0 of
this step + frame time witin step.
            t2=tElapsed+myframes[frame+1].frameValue  #Total time at frame 0
of this step + frame time witin step.

            if t1 != t2:#if the two frames are not equal; this is needed to
avoid calculation complications

                #get the current frame

                mySubset1=myframes[frame  ].fieldOutputs['NT11'].getSubset(reg
ion=myNSets[chosenNS])
                mySubset2=myframes[frame+1].fieldOutputs['NT11'].getSubset(reg
ion=myNSets[chosenNS])



                for idxNode in range(len(mySubset1.values)):
                    T1=mySubset1.values[idxNode].data
                    T2=mySubset2.values[idxNode].data
                    dT=T2-T1
                    dt=t2-t1

                    if dT <  0: #if we go downhill
                        if T1>Tupper and T2<Tupper: #if 800deg is between
points, interpolate
                            nPeak[idxNode]+=1
                            tT800[idxNode].append(t1+(dt)/(dT)*(Tupper-T1))
                            TT800[idxNode].append(Tupper)
                            flag1[idxNode]=1


                        elif T1==Tupper: #if 800deg is exactly a point, write
                            nPeak[idxNode]+=1
                            tT800[idxNode].append(t1)
                            TT800[idxNode].append(Tupper)
                            flag1[idxNode]=1

                        if flag1[idxNode]==1: #if after a proper peak
```

```
                            if T1>Tlower and T2<Tlower:#if 500deg is between
points, interpolate
                                tT500[idxNode].append(t1+(dt)/(dT)*(Tlower-
T1))
                                TT500[idxNode].append(Tlower)
                                flag1[idxNode]=0


                            elif T1==Tlower:      #if 500deg is exactly a
point, write
                                tT500[idxNode].append(t1)
                                TT500[idxNode].append(T1)
                                flag1[idxNode]=0


                            elif dT >= 0:#if we go uphill
                                if flag1[idxNode]==1: #if after a proper peak
                                    if T1>Tlower and T2<Tlower:#if 500deg is
between points, interpolate
                                        tT500[idxNode].append(t1+(dt)/(dT)*(Tl
ower-T1))
                                        TT500[idxNode].append(Tlower )
                                        flag1[idxNode]=0

                                    elif T1==Tlower:      #if 500deg is
exactly a point, write
                                        tT500[idxNode].append(t1)
                                        TT500[idxNode].append(T1)
                                        flag1[idxNode]=0

                                    elif frame == (len(myframes)-1) and
step==(len(mysteps)):#if something has been found but its the the last entry
                                        tT500[idxNode].append(
tT800[idxNode][nPeak[idxNode]-1])
                                        TT500[idxNode].append(
TT800[idxNode][nPeak[idxNode]-1])
                                        flag1[idxNode]=0
                    #print(txt2.format(idxNode,TT800[idxNode],tT800[idxNode],T
T500[idxNode],tT500[idxNode], nPeak[idxNode]))


    milestone(message='Hardness Calculation')
    #cpFile5.write('\n'+"TT800"+str(TT800) )
    #cpFile5.write('\n'+"tT800"+str(tT800) )
    #cpFile5.write('\n'+"TT500"+str(TT500) )
    #cpFile5.write('\n'+"tT500"+str(tT500) )
    cpFile5.write('\n'+"nPeak"+str(nPeak) )
```

```
    #dummy for easy text writing
    txt1="N= {} \t CR= \t {} \t HV= \t {}"
    positiveResults=[]

    for idxN in range(len(TT800)):
        #print(["len(TT800)  ",len(TT800)," idxN  ",idxN])


        #milestone(message='Calculating Coolrate', object='Node',
done=idxN+1,total=len(TT800))
        for idxM in range(max(nPeak)):

            #sanity check:
            #one of the four required pieces of info may not actually exist.
this will cause issues. So if the issue exists, there's no result
            #so first we check each part of the CR formula, then continue with
that data
            try:
                dT1=TT800[idxN][idxM]-TT500[idxN][idxM]
                dt1=tT800[idxN][idxM]-tT500[idxN][idxM]
            except IndexError:
                dT1=0
                dt1=0


            if (dT1==0) or (dt1==0):
                CR[idxN].append(0)

            else:
                CR[idxN].append((dT1) / (dt1) )


                    #this is only true in case of bad data, so you need to
know where it happens
                if  CR[idxN][idxM]>0:
                    positiveResults.append(["N=",myNSets[chosenNS].instances[0
].nodes[idxN].label,"T8",TT800[idxN][idxM],  "t8",tT800[idxN][idxM],
"T5",TT500[idxN][idxM], "t5",tT500[idxN][idxM]])




            if FlagInterpolate:
                HV[idxN].append(
ActionTableInterpolate(CCTData,abs(CR[idxN][idxM])))

            else:
```

```
                HV[idxN].append(
ActionTableExact(CCTData,abs(CR[idxN][idxM])))

            #if absolute values are asked, write them
            if FlagAbsolute:
                CR[idxN][idxM]=(abs(CR[idxN][idxM]))


        print(txt1.format(myNSets[chosenNS].instances[0].nodes[idxN].label,CR[
idxN], HV[idxN]))
        cpFile5.write('\n'+'N= '+
str(myNSets[chosenNS].instances[0].nodes[idxN].label) + '\t CR=
'+str(CR[idxN])+ '\t HV= '+str(HV[idxN]))

    cpFile5.write('\n')

    for result in  positiveResults:
        cpFile5.write('\n'+str(result))
    cpFile5.write('\n')


    return [CR,HV]


def EnergyDensity(**EDargs):
    materials        =  (EDargs["materials"])   #materials
    materialkeys     =  (EDargs["materialkeys"])   #materialkeys

    Matdensity = materials[materialkeys[-1]].density.table
    MatspecificHeat = materials[materialkeys[-1]].specificHeat.table

    rho=[]
    CP=[]
    T1=[]
    T2=[]
    print(["Matdensity",Matdensity])
    print(["MatspecificHeat",MatspecificHeat])



    for idx1 in range(len(Matdensity)):
        rho.append(Matdensity[idx1][0])
        T1.append(Matdensity[idx1][1])

    for idx2 in range(len(MatspecificHeat)):
        CP.append(MatspecificHeat[idx2][0])
        T2.append(MatspecificHeat[idx2][1])

    T3=T1+T2
```

```
    T3=list(set(T3))
    T3.sort()


    CPList=[0]*len(T3)
    RhoList=[0]*len(T3)

    #CPList[0]=CP[0]
    #RhoList[0]=rho[0]
    #CPList[-1]=CP[-1]
    #RhoList[-1]=rho[-1]

    print("rho",rho)
    print("T1",T1)
    print("CP",CP)
    print("T2",T2)
    print("T3",T3)

    Tabs=273.15


    for id1 in range(len(T1)):
        for id3 in range(len(T3)):
            if T1[id1]==T3[id3]:
                RhoList[id3]=rho[id1]

            elif id1 == 0:
                if T3[id3] < T1[id1]:
                    print(["Entry found before
list","T1",T1[id1],"T3",T3[id3]])
                    RhoList[id3]=rho[id1]
                elif T1[id1]<T3[id3] and T1[id1+1]>T3[id3]:
                    X0=T1[id1]
                    Y0=rho[id1]
                    X1=T1[id1+1]
                    Y1=rho[id1+1]
                    RhoList[id3]=ActionInterpolate(X0,Y0,X1,Y1,T3[id3])

            elif id1 == len(T1)-1:
                if T3[id3] > T1[id1]:
                    print(["Entry found After
list","T1",T1[id1],"T3",T3[id3]])
                    RhoList[id3]=rho[id1]

            else:
                if T1[id1]<T3[id3] and T1[id1+1]>T3[id3]:
                    X0=T1[id1]
                    Y0=rho[id1]
                    X1=T1[id1+1]
```

```
                    Y1=rho[id1+1]
                    RhoList[id3]=ActionInterpolate(X0,Y0,X1,Y1,T3[id3])

    for id2 in range(len(T2)):
        for id3 in range(len(T3)):
            if T2[id2]==T3[id3]:
                CPList[id3]=CP[id2]
            elif id2 == 0:
                if T3[id3] < T2[id2]:
                    CPList[id3]=CP[id2]
                    print(["Entry found before
list","T2",T2[id2],"T3",T3[id3]])
                elif T2[id2]<T3[id3] and T2[id2+1]>T3[id3]:
                    X0=T2[id2]
                    Y0=CP[id2]
                    X1=T2[id2+1]
                    Y1=CP[id2+1]
                    CPList[id3]=ActionInterpolate(X0,Y0,X1,Y1,T3[id3])

            elif id2 == len(T2)-1:
                if T3[id3] > T2[id2]:
                    CPList[id3]=CP[id2]
                    print(["Entry found after
list","T2",T2[id2],"T3",T3[id3]])

            else:
                if T2[id2]<T3[id3] and T2[id2+1]>T3[id3]:
                    X0=T2[id2]
                    Y0=CP[id2]
                    X1=T2[id2+1]
                    Y1=CP[id2+1]
                    CPList[id3]=ActionInterpolate(X0,Y0,X1,Y1,T3[id3])

    print(["CPList",CPList])
    print(["RhoList",RhoList])
    print(["T3",T3])


    EnergyDensity=[0]*len(T3)
    EnergyCumulative=[[0,0]]*len(T3)


    EnergyDensity[0]=(T3[0]+Tabs)*CPList[0]*RhoList[0]
    Cumulative=EnergyDensity[0]
    EnergyCumulative[0][0]=T3[0]+Tabs
    EnergyCumulative[0][1]=Cumulative

    for idxY in range(1,len(T3)):
        dT=T3[idxY]-T3[idxY-1]
```

```
        EnergyDensity[idxY]=(dT)*CPList[idxY]*RhoList[idxY]
        Cumulative+= EnergyDensity[idxY]
        EnergyCumulative[idxY][0]=T3[idxY]
        EnergyCumulative[idxY][1]=Cumulative




    print(["EnergyDensity",EnergyDensity])
    print(["Cumulative",Cumulative])
    print(["EnergyCumulative",EnergyCumulative])


    return EnergyCumulative


def EnergyCalc(**Args):
    EnergyCumulative=Args["EnergyCumulative"]
    myodb           = (CCAargs["myodb"])           #

    milestone(message='Calculation Start')
    #txt2=" idxNode= \t {}\tT800= \t {} \t t800= \t {} \t T500= \t {}\t t500=
\t {}\t nPeak[idxNode]-1= \t {}"
    mysteps = myodb.steps.keys()
    myElementSet=myodb.rootAssembly.elementSets[' ALL ELEMENTS']
    myNodeSet=myodb.rootAssembly.nodeSets[' ALL NODES']
    for step in range(len(mysteps)):
        tElapsed=myodb.steps[mysteps[step]].totalTime #time at frame 0 of this
step
        myframes=myodb.steps[mysteps[step]].frames
        milestone(message='Retrieving Data', object='Step',
done=step+1,total=len(mysteps))

        landmark(text='step='+ str(step))
        #for every frame in step
        for frame in range(len(myframes)-1):

            #get the current time
            t1=tElapsed+myframes[frame  ].frameValue #Total time at frame 0 of
this step + frame time witin step.

            if t1 != t2:#if the two frames are not equal; this is needed to
avoid calculation complications

                #get the current frame

                myElements=myElementSet.elements[0]
                mySubset1=myframes[frame].fieldOutputs['NT11'].getSubset(regio
n=myNodeSet)
```

```
                mySubset2=myframes[frame].fieldOutputs['EVOL'].getSubset(regio
n=myElementSet)

                for idxEl in range(len(myElements)):
                    cumEl=0

                    nodelist=myElements.elements[idxEl][0].connectivity

                    mySubVolume=mySubset2.values

                    for idxN in range(len(nodelist)):
                        cumE+=ActionTableInterpolate(EnergyCumulative,mySubset
1.values[idxN].data)

                    EEl=cumE*vEl




##############################################################################
# Class definition
# Main program
##############################################################################

#ActionCoolRateCalc(odbName,nodeSetName,PickNode,fileNameCCT,peaktemperature,u
ppertemperature,lowertemperature,ChoiceOutput,FlagLastresult,FlagSavefield,fil
eNameOutput,FlagInterpolate):
#**KWArgs makes sure that the arguments are passed as a dictionary of
arbitrary length and form. The relevant data is then processed to a more
useable form.
def ActionCoolRateCalcNew(**KWArgs):
    print(KWArgs)

    #def
ActionCoolRateCalc(odbName,nodeSetName,PickNode,KWoutputfile,KWpeaktemperature
,KWuppertemperature,KWlowertemperature,KWCCTPath):
    filename        = str  (KWArgs["fileNameOutput"])   #Log file name
    Tlower          = int  (KWArgs["lowertemperature"]) #T500 temperature
    FlagLastresult  = bool (KWArgs["FlagLastresult"])   #Save only last result
yes/no
    chosenNS        = str  (KWArgs["nodeSetName"])      #name of the node set
    Tupper          = int  (KWArgs["uppertemperature"]) #T800 temperature
    FlagInterpolate = bool (KWArgs["FlagInterpolate"])  #Should cct diagram be
interpolated?
    decisionSave    = bool (KWArgs["FlagSavefield"])    #Save to ODB yes/no
    myodbname       = str  (KWArgs["odbName"])          #name of the odb
    TPeak           = int  (KWArgs["peaktemperature"])  #peak to be found
```

```
    fileNameCCT     = str  (KWArgs["fileNameCCT"])      #file name of CCT
input
    ChoiceOutput    = str  (KWArgs["ChoiceOutput"])     #What output is
requested?
    FlagAbsolute    = bool (KWArgs["FlagAbsolute"])     #should the coolrate
be taken as absolute?
    FlagHVzero    = bool (KWArgs["FlagHVzero"])       #should CR=0 mean HV=0?
    idxOutputStep   = int  (KWArgs["OutputStep"])       #What Frame IDX for
output?
    idxOutputFrame  = int  (KWArgs["OutputFrame"])      #What Frame IDX for
output?

    HVfieldname     = str  (KWArgs["HVfieldname"])      #What is name of HV
field?
    CRfieldname     = str  (KWArgs["CRfieldname"])      #What is name of CR
field?

    FlagPickNode    = bool (KWArgs["FlagPickOverride"]) #should chosenNS be
ignored in favor of PickNode?




    #print PickNode
    #[(session.openOdb(r'D:/temp/PythonTest_DoublePeak.odb').rootAssembly.inst
ances['PART-1-1'].nodes[66],
session.openOdb(r'D:/temp/PythonTest_DoublePeak.odb').rootAssembly.instances['
PART-1-1'].nodes[67],
session.openOdb(r'D:/temp/PythonTest_DoublePeak.odb').rootAssembly.instances['
PART-1-1'].nodes[77],
session.openOdb(r'D:/temp/PythonTest_DoublePeak.odb').rootAssembly.instances['
PART-1-1'].nodes[78])]
    #print PickNode[0][0]
    #({'coordinates': array([0.0, 30.0, 2.85714292526245], 'f'),
'instanceName': 'PART-1-1', 'label': 67})
    #IE print PickNode[0][0].label = 67
    #so it gives node numbers, which could be used to get the necessary data.
    #picknode is a list of objects, need to check if it exists andd then tur
to a nodde list.
    try:
        PickNode=KWArgs["PickNode"]
        print("PickNode",PickNode)
        print("type(PickNode)",type(PickNode))

        PickNodeList=[]
        for idx in range(len(PickNode)):
            PickNodeList.append(PickNode[idx].label)
```

```
    except:
        PickNode=[]
        PickNodeList=[]


    print("PickNodeList",PickNodeList)
    print("type(PickNodeList)",type(PickNodeList))




    #Adaption need

    myodbKeys=[myodbname]

    materials = session.odbs[myodbKeys[0]].materials
    materialkeys=materials.keys()
    EnergyCumulative=EnergyDensity(materials=materials,materialkeys=materialke
ys)









    #end adaption need

    abqVersion=version #get abaqus version for compatibility
    print('Program Start')
    print(str(time.ctime()))

    milestone(message='Program Start')

    Time1=time.time()

    milestone(message='User Input Phase (1/2)')

    #if the proper extension is not given, add extension
    if ".txt" not in filename:
        filename=(filename + '.txt')

    #test if the user input works, if so accept, if not give second chance
    try:
        cpFile5 = open(filename, 'a+')
        print('success, file name= ', filename)
    except ValueError:
        print('error, give second chance')
```

```
        filename=getInput('Output Filename was incorrect, try again?:\tThe
file to which the output is written', default='regionoutput.txt')
        print('User Input:', filename)
        #test if the new user input works, if so accept, if not go to default
        try:
            cpFile5 = open(filename, 'a+')
            print('success, file name= ', filename)
        except ValueError:
            filename='fieldoutput_default.txt'
            print('error, file name= ', filename)
            cpFile5 = open(filename, 'a+')




    cpFile5.write('Program Started'+'\n')
    cpFile5.write(str(time.ctime())+'\n')

    myodb = openOdb(myodbKeys[0])

    print("Odb readonly?",myodb.isReadOnly)
    if (myodb.isReadOnly):
        myodb.close()
        myodb = openOdb(myodbKeys[0],False)
        print("Odb readonly fixed?",myodb.isReadOnly)



    mysteps = myodb.steps.keys()

    #myodbKeys=session.odbs.keys()
    #myodb = openOdb(myodbKeys[0])

    cpFile5.write('\n'+'ODB= '+ '\t'+ str(myodbKeys[0])+'\n')
    #myodb= session.odbData[myodbKeys[0]]
    part_name = myodb.parts.keys()
    cpFile5.write('\n'+'ABAQUS VERSION = '+ '\t'+ str(abqVersion)+'\n')
    if '2019' in abqVersion:
        model_name=session.odbData.keys()

    elif '2017' in abqVersion:
        model_name=session.mdbData.keys()

    cpFile5.write('Peak Temperature: = '+ '\t'+ str(TPeak)  + '\t'+' C' +'\n')
    cpFile5.write('Upper Temperature: = '+ '\t'+ str(Tupper)  + '\t'+' C'
+'\n')
    cpFile5.write('Lower Temperature: = '+ '\t'+ str(Tlower) + '\t'+' C'
+'\n')
```

```
    myfieldname = 'NT11'
    cpFile5.write('\n'+'FieldOutput= '+ '\t'+ str(myfieldname)+'\n')
    myInstanceKeys= myodb.rootAssembly.instances.keys()
    myInstance=myodb.rootAssembly.instances[myInstanceKeys[0]]
    cpFile5.write('\n'+'myInstance Name= '+ '\t'+ str(myInstance.name))
    cpFile5.write('\n'+'myInstance Node Sets= '+ '\t'+
str(myInstance.nodeSets.keys()))



    if len(PickNodeList)>0:
        NewNodeSet=myInstance.NodeSetFromNodeLabels(name='pickedNodes',nodeLab
els=(PickNodeList))
        print("NewNodeSet",NewNodeSet)


    if FlagPickNode:
        chosenNS='pickedNodes'



    # gets the nodesets in an odb
    myNSets = myodb.rootAssembly.nodeSets
    # gets the names of the node sets
    myNSetKeys = myNSets.keys()
    # gets the elsets in an odb
    myESets = myodb.rootAssembly.elementSets
    # gets the names of the elsets
    myESetKeys = myESets.keys()

    milestone(message='User Input Phase (2/2)')

    #creates a default cct diagram to use in case of errors
    #this is a 1-to-1 conversion


    CCTDefault=[[0.01,0.01],[0.02,0.02],[0.03,0.03],[0.04,0.04],[0.05,0.05],[0
.06,0.06],[0.07,0.07],[0.08,0.08],[0.09,0.09],
        [0.1,0.1],[0.2,0.2],[0.3,0.3],[0.4,0.4],[0.5,0.5],[0.6,0.6],[0.7,0.7],
[0.8,0.8],[0.9,0.9],
        [1,1],[2,2],[3,3],[4,4],[5,5],[6,6],[7,7],[8,8],[9,9],
        [10,10],[20,20],[30,30],[40,40],[50,50],[60,60],[70,70],[80,80],[90,90
],
        [100,100],[200,200]]

    print("Try reading CCT")
```

```
    try:
        CCTData=ActionReadCCTFile(fileNameCCT)
        cpFile5.write('\n'+'\n'+'CCTData = '+ '\t'+ str(CCTData))
        print("CCT Data read correctly")
    except IOError :
        CCTData=CCTDefault
        cpFile5.write("NO CCT Data Found, use default ")
        cpFile5.write('\n'+'\n'+'CCTData = '+ '\t'+ str(CCTData))
        print("NO CCT Data Found, use default ")
        print("Debug: "+str(fileNameCCT))


    idxNS=0#default index for nodeset, this usually is ALL NODES

    nNodes=session.odbData[myodbKeys[0]].nodeSets[chosenNS].nodes

    #Preallocate data for list function
    numNodesChosen=len( nNodes[myInstance.name])




    print('Elapsed time before calculation is:', str(round((time.time()-
Time1),0)),' seconds')

    #FlagLastresult: should it only show last result?
    #note: CoolrateCalcSimple does not keep track of intermediate peaks. This
makes it faster, though the effect will be stronger for bigger files.


    if FlagLastresult:
        print("Calculate only last peaks, use CoolrateCalcSimple")
        Results =CoolrateCalcSimple(chosenNS=chosenNS,  myodbKeys=myodbKeys,
cpFile5=cpFile5, myInstance=myInstance,
myodb=myodb,Tupper=Tupper,  Tlower=Tlower,CCTData=CCTData,FlagInterpolate=Flag
Interpolate,FlagAbsolute=FlagAbsolute,FlagHVzero=FlagHVzero)
        print('Elapsed time after simple calculation is:',
str(round((time.time()-Time1),0)),' seconds')
    else:
        print("Calculate all peaks, use CoolrateCalcAdvanced")
        Results =CoolrateCalcAdvanced(chosenNS=chosenNS,  myodbKeys=myodbKeys,
cpFile5=cpFile5, myInstance=myInstance,
myodb=myodb,Tupper=Tupper,  Tlower=Tlower,CCTData=CCTData,FlagInterpolate=Flag
Interpolate,FlagAbsolute=FlagAbsolute,FlagHVzero=FlagHVzero)
        print('Elapsed time after advanced calculation is:',
str(round((time.time()-Time1),0)),' seconds')
```

```
    CR=Results[0]
    HV=Results[1]

    ########################
    # postprocess
    ########################
    milestone(message='Post-Processing')
    #Coolrate Postprocess

    #Calculte last peaks, only applies in case complicated variant was chosen

    if not FlagLastresult:
        CRf=[0]*len(CR) # CR final
        HVf=[0]*len(HV) # HV final
        peakrange=range(len(CR[0]))
        peakrange.reverse()


        for idxN in range(len(CR)):
             for idxM in peakrange:
                if not( CR[idxN][idxM] == 0):
                    CRf[idxN]=(CR[idxN][idxM])
                    break
        for idxN in range(len(HV)):
            for idxM in peakrange:
                if not( HV[idxN][idxM] == 0):
                    HVf[idxN]=(HV[idxN][idxM])
                    break
        #print("CRf",CRf)
        #print("HVf",HVf)

    else:
        CRf=CR # CR final
        HVf=HV # HV final

    CRnz=[] #list of nonzero coolrates
    idxNZ=0 #number of nonzero coolrates
    CRtot=0 #total coolrate for average

    for idx2 in range(len(CRf)):
        if CRf[idx2]!=0:
            CRnz.append(CRf[idx2])
            idxNZ+=1
            CRtot+=(CRf[idx2])

    if CRnz: #true if exists, false if empty
        crMax=( max(CRnz)         )
        crAvg=( (CRtot) / (idxNZ) )
        crMin=( min(CRnz)         )
```

```
        nList=nNodes[myInstance.name]

        #find Nodes for Min andd Max
        idxMAX=-1
        idxMIN=-1

        for idx3 in range(len(nList)):
            if CRf[idx3] == crMax:
                idxMAX=idx3
            elif CRf[idx3] == crMin:
                idxMIN=idx3

        crMax=round(crMax,2)
        crAvg=round(crAvg,2)
        crMin=round(crMin,2)


        print('max coolingrate')
        cpFile5.write('\n' + 'max coolingrate'+'\n')

        if idxMAX == -1:
            cpFile5.write('\n' + 'max coolingrate not found'+'\n')
            print('max coolingrate not found')
        else:
            hvMax=round(HVf[idxMAX],2)
            print('Node: ',nList[idxMAX],' CR: ', crMax,' C/s',' HV: ', hvMax)
            cpFile5.write('Node:'+ str(nList[idxMAX]) + '\t'+str(crMax)+'\t'+'
C/s'+'\t'+str(hvMax)+'\t'+'HV'+'\n')


        print('min coolingrate')
        cpFile5.write('\n' + 'min coolingrate'+'\n')

        if idxMIN == -1:
            print('min coolingrate not found')
            cpFile5.write('\n' + 'min coolingrate not found'+'\n')

        else:
            hvMin=round(HVf[idxMIN],2)
            print('Node: ',nList[idxMIN],' CR: ', crMin,' C/s',' HV: ',hvMin)
            cpFile5.write('Node:'+ str(nList[idxMIN]) + '\t'+str(crMin)+'\t'+'
C/s'+'\t'+str(hvMin)+'\t'+'HV'+'\n')


        print('average coolingrate')
        print(crAvg,' C/s')

        cpFile5.write('\n'+'average coolingrate'+'\t'+ str(crAvg) +'\t'+'
C/s'+'\n')
```

```
    else:
        print('No Results Found')
        try:
            print("CRf",CRf)
            print("CR",CR)
        except:
            False

        cpFile5.write('\n' + 'No Results Found'+'\n')



    tTotal=round(time.time()-Time1,2)

    cpFile5.write('\n'+'Total Time Taken:' + '\t'+str(tTotal) +'\t'+
'seconds')
    print('Elapsed time is:', str(tTotal),' seconds')

    print('time per step: ',round((tTotal)/(len(mysteps)),2), ' seconds for '
, len(mysteps) ,' steps')
    print('time per node: ',round((tTotal)/(numNodesChosen),2), ' seconds for
' ,numNodesChosen, ' nodes' )


    cpFile5.write('\n'+'time per step: '+
str(round((tTotal)/(len(mysteps)),2))   + ' seconds for ' +
str(len(mysteps)  ) + ' steps')
    cpFile5.write('\n'+'time per node: '+
str(round((tTotal)/(numNodesChosen),2)) + ' seconds for ' +
str(numNodesChosen) + ' nodes' )
    cpFile5.write('\n'+'End of Program')
    cpFile5.write('\n'+str(time.ctime())+'\n'+'\n'+'\n')


    milestone(message='Close TXT  File')
    cpFile5.close()



    ########################
    # Write data as Field output
    ############################
    if decisionSave:
        milestone(message='Write Field Outputs')
```

```
        #ask for a viewport so the model is properly visible
        myViewport=session.viewports.keys()
        #first we need an existing field to get the instance for the new field
from.
        print("Gett Old Field Data")
        print(["myodbKeys[0]",myodbKeys[0]])
        print(["mysteps[-1]",mysteps[-1]])
        print(["session.odbs[myodbKeys[0]]",session.odbs[myodbKeys[0]]])
        print(["session.odbs[myodbKeys[0]].steps[mysteps[-
1]]",session.odbs[myodbKeys[0]].steps[mysteps[-1]]])
        print(["session.odbs[myodbKeys[0]].steps[mysteps[-1]].frames[-
1]",session.odbs[myodbKeys[0]].steps[mysteps[-1]].frames[-1]])




        oldFieldOutput=session.odbs[myodbKeys[0]].steps[mysteps[-1]].frames[-
1].fieldOutputs['NT11']
        print("Old Field Data retrieved")
        #get data from old field
        #idxFrameTaken=[[],[]]
        #for idxSteps in range(len(mysteps)):
        #    myframes=session.odbs[myodbKeys[0]].steps[mysteps[idxSteps]].fram
es
        #
        #    for idxFrames in range (len(myframes)):
        #        print("idxSteps",idxSteps,"idxFrames",idxFrames)
        #        try:
        #            myframes[idxFrames].fieldOutputs['CR11']
        #        except KeyError:
        #            idxFrameTaken[0].append(idxFrames)
        #            idxFrameTaken[1].append(idxSteps)


        #now we get the instancce from the old field output
        #the instance contains the node number and location data
        myInstance=oldFieldOutput.values[0].instance #get instance
        myLabels=nNodes.values() #get node labels

        #check if the user asked for coolrate output
        print("ChoiceOutput: " + str(ChoiceOutput))
        fIdx=idxOutputFrame

        myData=[] #data to write to CR field
        myData2=[] #data to write to HV field

        #in case the number of results is not the number of nodes, just fill
in zeroes
```

```
        nNodesAll=session.odbData[myodbKeys[0]].nodeSets[" ALL
NODES"].nodes[myInstanceKeys[0]]#number of nodes in entire model
        #print("nNodesAll",nNodesAll)
        print("len nNodesAll",len(nNodesAll))
        print("len(nNodes)",len(nNodes))
        print("nNodes",nNodes)
        myNodes=nNodes[myInstance.name]


        print("DO CR CaLC")
        if len(myNodes)==len(nNodesAll):
            for i in range(len(nNodesAll)):
                myData.append([CRf[i]])
                myData2.append([HV[i]])

        else:

            for i in range(len(nNodesAll)):
                if myNodes[i] in nNodesAll:
                    myData.append([CRf[i]])
                    myData2.append([HV[i]])
                else:
                    myData.append([0])
                    myData2.append([0])

        if ("Both" in ChoiceOutput) or ("Coolrate" in ChoiceOutput):

            nPeaksFound=len(CR)
            nFramesNeeded= -nPeaksFound

            #Next, we define a new empty field with the name and type we
desire.

            try:
                newFieldOutput=session.odbs[myodbKeys[0]].steps[mysteps[idxOut
putStep]].frames[fIdx].FieldOutput(name=CRfieldname,description='T800-T500
Coolrate', type=SCALAR)#create new, empty field
            except OdbError:
                try:
                    fIdx-=1
                    print("Field already exists")
                    newFieldOutput=session.odbs[myodbKeys[0]].steps[mysteps[id
xOutputStep]].frames[fIdx].FieldOutput(name=CRfieldname,description='T800-T500
Coolrate', type=SCALAR)#create new, empty field
                except OdbError:

                    try:
                        fIdx-=1
```

```
                        print("Field already exists")
                        newFieldOutput=session.odbs[myodbKeys[0]].steps[mystep
s[idxOutputStep]].frames[fIdx].FieldOutput(name=CRfieldname,description='T800-
T500 Coolrate', type=SCALAR)#create new, empty field
                    except OdbError:
                        print("Field already exists part 2")



        print("add the custom data to the field")
        #add the custom data to the field
        newFieldOutput.addData(position=NODAL,instance=myInstance,labels=myLab
els[-1],data=myData) #add data to field
        print("custom data added to the field")



        #do the same for hardness
        #check if the user asked for coolrate output
        print("ChoiceOutput: " + str(ChoiceOutput))

        if ("Both" in ChoiceOutput) or ("Hardness" in ChoiceOutput):

            try:
                newFieldOutput2=session.odbs[myodbKeys[0]].steps[mysteps[idxOu
tputStep]].frames[fIdx].FieldOutput(name=HVfieldname,description='Calculated
Hardness', type=SCALAR)#create new, empty field
            except OdbError:
                newFieldOutput2=session.odbs[myodbKeys[0]].steps[mysteps[idxOu
tputStep]].frames[fIdx].fieldOutputs[HVfieldname]#get data from old field

            newFieldOutput2.addData(position=NODAL,instance=myInstance,labels=
myLabels[-1],data=myData2) #add data to field

        #save, close and reopen the odb


        myodb.save()#save ODB
        myodb.close()
        myodb = openOdb(myodbKeys[0],False)
        print("Saved data to ODB")
        #display the odb properly
        objChosen=session.odbs[myodbKeys[0]]
        session.viewports[myViewport[0]].setValues(displayedObject=objChosen)
        session.viewports[myViewport[0]].odbDisplay.setPrimaryVariable(variabl
eLabel='NT11',outputPosition=NODAL,)
        session.viewports[myViewport[0]].odbDisplay.setFrame(step=
mysteps[fIdx], frame=fIdx)
```

```
    else:
        print("data local only")



    milestone(message='Program End')
    print('Job Done')
```