Computational Analysis Of Warpage Of Composite Laminates

Towards digital twins of laminates fabricated from an automation line

Computational Analysis Of Warpage Of Composite Laminates

Towards digital twins of laminates fabricated from an automation line

Proefschrift

om het masterdiploma te behalen

aan de Technische Universiteit Delft,

op gezag van de dr. ir. Marcel Sluiter,

in het openbaar te verdedigen op woensdag 9 october 2019 om 14:00 uur

door

Haralambos Alkiviades

Physicist,
Aristotelio University of Thessaloniki,

geboren te Limassol, Cyprus.

Dit proefschrift is goedgekeurd door de

promotor: Dr. M. Sluiter

Samenstelling promotiecommissie:

Voorzitter,	
Dr. M. Sluiter,	Technische Universiteit Delft
leidinggevende,	
Dr. M.A. Bessa,	Technische Universiteit Delft
Onafhankelijke leden:	
Dr. P. Dey,	Technische Universiteit Delft
bedrijfsafgevaardigde:	
Mr. M. Muilwijk,	Airborne Composite Automation

Dr. M.A. Bessa heeft in belangrijke mate aan de totstandkoming van het proefschrift bijgedragen.

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

SUMMARY

Haralambos Alkiviades

Airborne Composites Automation recently installed an automation product line that creates flat thermoplastic laminates intended for consumer electronics industry. However, due to the composite nature of the material and the process parameters, the final product is deformed, as a result of internal residual stresses. The purpose of this thesis is to use Finite Element Analysis methods in order to build up digital twins of the composite laminates undergoing the manufacturing process composed by several pressing cycles in order to predict their final warpage. Hence, this work aims at characterizing and predicting the real cause for warpage, offering an opportunity to minimizing it. Since the parameterization of the simulations is automated, this work is a first step towards the use of data-driven methods that enable analysis and design of future laminates under different process parameters.

Contents

1

INTRODUCTION

IN recent years, the use of fibre reinforced polymer composites in high performance structural applications has increased significantly due to improvements in processing technology that enables the production of parts of very high quality. With increasing production rates of structures, there is a demand for time- and cost-efficient manufacturing processes for large scale production. Traditional manufacturing techniques for composite materials are generally slow as a typical production cycle involves labour-intensive ply collation by either hand lay-up or the use of preforms followed by long curing cycles [1]. On the other hand, Airborne Composites Automation (ACA) was able to construct an automated manufacturing technique where robotic engineering and science replace time consuming and expensive manual layup processes by laying the thermoplastic composite into the process line. To complete consolidation, several pressing cycles are used which combine mechanical and thermal loads simultaneously with the help of pressing plates and robotic hands.

The purpose of this work is to create the whole process using Finite Element (FE) modeling techniques in order to find what processing parameters cause the deformation and how to optimize it. The complexity of the modeling and process simulation is attributed to the multi-physics and multi-scale nature of the composites. Moreover, worth mentioning is that, the specific process does not exist from previous authors, so a general digital computational twin for this new product line is important for the next generation of composite automation.

The automated product line is used to manufacture laminates that will be applied to laptop cases, among other possible applications. Therefore, the flatness of the laminates is particularly important for their functionality as well as visual appeal. However, this project has an important practical constraint: the material properties are not experimentally determined because the purpose of this thesis is to focus on simulating the process and to automate the input of the parameters involved. Moreover, it will provide working material for future work, concerning press consolidation, thermoforming, stamping and in general forming techniques. Therefore, this research has limited experimental input beyond simple characterizations of the final shape of the laminate (the output of this research). Therefore, the research goal is to create a general computational strategy that can be adopted for predicting the residual stress build-up and subsequent warping of any composite laminates manufactured by the automation line.

The manufacturing process is called the "Falcon Line" with the purpose of producing thermoplastic laminates at a fast pace without any human involvement. The whole process is very fast (a matter of minutes) where in each full run, 4 flat laminates are produced. Moreover, the process is labour free, which allows for 24/7 production, reducing vastly the manufacturing cost of the composite parts.

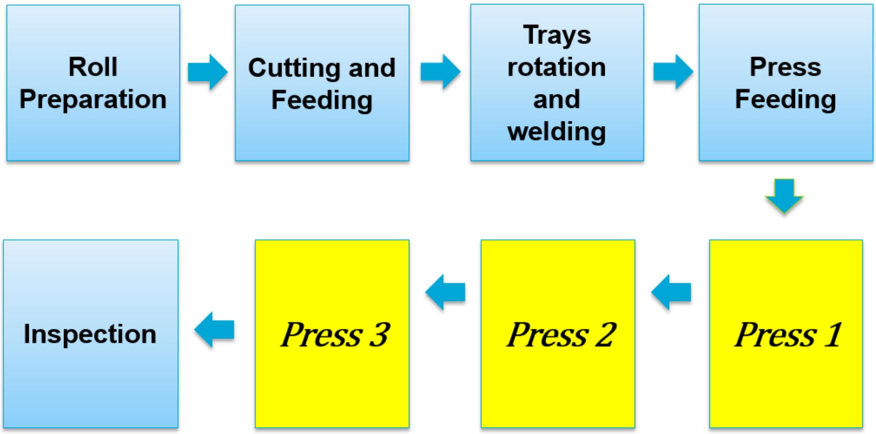


Figure 1.1: Schematic Representation of the Falcon Line

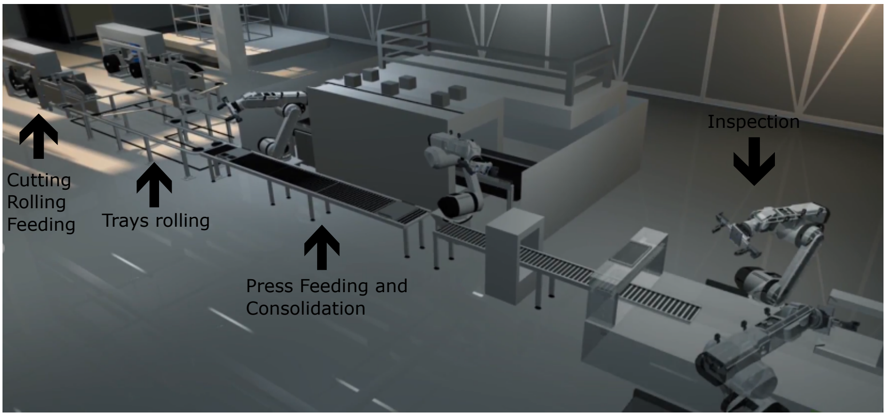


Figure 1.2: Falcon Line

The line is a combination of several steps (Figure 1.1). Also, Figure 1.2 shows the line from a digital movie that represents the real product line. Two rollers feed the system using trays that are moving into trailing lines. With correct cutting and placement, the two rollers provide laminates with different orientations and the laminate sequence is controlled automatically by choosing it before the start of feed process. At the moment, only two directions can be chosen (0 and 90 Degrees) creating cross-ply laminates. After feeding, the laminates are welded with ultrasonic welding at two spots for the robot to easily grab them and place them into the consolidation pressing machine. The system handles four composite laminates at a time (Figure 4.17), where steel plates and a compatibility layer sheet are also used separately. With the help of robotic hands, the structure is moving through the press (plates, compatibility layer and laminates). The pressing machine consists of 3 presses, each one applying different temperature and pressure values, which is the main focus of this thesis. Moreover, the time that the loads are applied is important and also the time of the transportation from each press to the next, as it controls the heating and cooling profile of the laminates. After consolidation, the last step is the inspection of the laminate's quality, which measures the deviation of the laminates from flatness. From the stages, many variables affect the outcome of the material's quality.

The first laminates fabricated by this process showed non-negligible warping. Warpage is characterized by a distortion of the composite laminate that leads to a different final shape after the manufacturing process is finished, as compared to the intended design [2]. In our case, warpage stands for the deviation from the flatness of initially flat laminates (the laminates must be as flat as possible in order to meet the design requirements) [3] – see Figure 1.3 exemplifying a laminate at the end of the Falcon line. Moreover, the laminates were extracted having concave shape (edges touching the compatibility layer). Then, the user turned over the laminates (convex shape) and measured the distance from all the laminate's corners to the flat reference plate. The final warpage is defined as the average of the 4 measured distances for each laminate individually. Typically, residual stresses generated during the production steps are the culprit for this defect. In general, there are multiple factors associated to composite warpage that can have two origins:

1. **Intrinsic sources**: arising from the material and the part itself. For example, material anisotropy, heterogeneity, thermo-mechanical properties, stacking sequence and part shape can strongly affect warpage of composite laminates. Non-symmetric laminate sequences, fibre misalignments, non-homogeneous distribution of fibers or defects (e.g. matrix voids), and moisture absorption are typical factors that lead to warpage [4].

2. **Extrinsic sources**: process related issues, such as kinetics of the forming process, and thermal gradient profiles that depend on mechanical tool-part interaction [5].

Chapter 2 provides a literature review about this subject and elaborates on the sources of composite warpage. Chapter 3 shows the influence of a temperature gradient on the development of stresses and also of the deformation of the material. Both computational and also experimental efforts will be compared and analyzed. Chapter 4 illustrates the mechanical interaction from a tool to the laminate that will deform the material, which again, both computational and experimental work will be illustrated. Chapter 5 will only provide a computational technique to simulate viscoelastic material, which may open doors for future work either computationally or experimentally. In the Appendix, all the coding will be explained in detail that was used for the simulations and also, coding for future work will be provided.

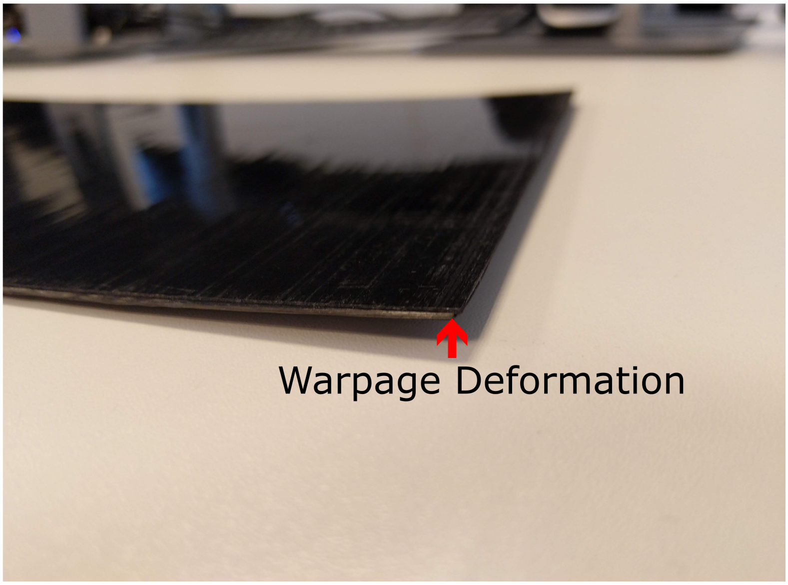


Figure 1.3: Warped laminate from Falcon Line

2

LITERATURE REVIEW

IN this chapter the origin of residual stresses and their role on the material's behavior is provided, as well as analytical models describing this behavior. A brief introduction about thermoplastic polymers composites and the relevant manufacturing process in this thesis is also included.

2.1. BASICS OF THERMOPLASTICS

Thermoplastics are high density polymers where the interaction between polymer chains usually occurs via van der Waals forces [6] that weakly attract neutral molecules to each other. Unlike thermosets that are rigidly crosslinked by permanent bonds between chains, thermoplastics can be reheated and molded into a wide range of shapes multiple times [7], which makes them recyclable. Thermoplastics do not undergo a curing process (no permanent cross-links), so the manufacturing process is significantly faster than for thermosets.

Airborne's Falcon Line currently uses a composite supplied by Sabic [8] that has a polycarbonate thermoplastic matrix. This polymer has an amorphous structure [9], and undergoes different temperature and pressure cycles. In amorphous polymers, the material transitions from a liquid/fluid state into a glassy/solid one once it reaches the glass transition temperature. This temperature is important because it affects the mechanical properties of the material significantly. The glass transition temperature T_g should not be confused with the melting temperature T_m, since the latter is the "solid-liquid" transition in one step and can only occur for crystalline materials. Figure 2.1 shows the difference between the two temperatures, T_g and T_m, comparing an amorphous material, a semi-crystalline and a crystalline configuration. The glass transition presents features of a 2^{nd} order transition since thermal studies often indicate that the molar Gibbs energies, molar enthalpies, and the molar volumes of the two phases, i.e., the melt and the glass, are equal, while the heat capacity and the expansivity are discontinuous (but for other properties such as Elastic Modulus vs Temperature, the glass transition is of 1^{st} order [11, 55]). However, the glass transition is generally not regarded as a thermodynamic transition in view of the inherent difficulty in reaching equilibrium in a polymer glass or in a polymer melt at temperatures close to the glass-transition temperature. The figure highlights the variation of volume as a function of temperature [10].

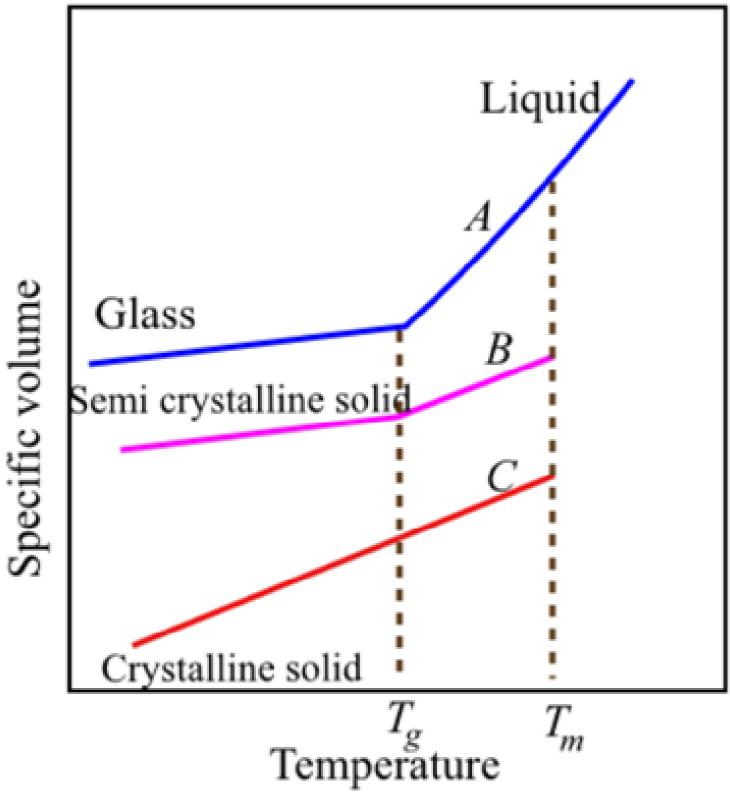


Figure 2.1: Volume Vs Temperature for solids [10].

Figure 2.2 shows how the Young's Modulus of a polyetheretherketone (PEEK) thermoplastic polymer changes as temperature increases, where it is clear the significant decrease of the Modulu's value after T_g is surpassed. As temperature rises above T_g, material volume increases and facilitates the movement of molecular chains which affects thermal and elastic properties [11].

Several factors affect the transition into a glassy state:

1. Mobility of the polymer chains as the temperature increases.
2. Presence of plasticizers, which tend to separate chains from each other and increase the free volume. In this case, chains can slide past each other more easily, lowering T_g and making the polymer more pliable [11].
3. External pressure which tends to increase the glass transition temperature (smaller free volume).

The effect of pressure has been investigated by different authors, in a technique called

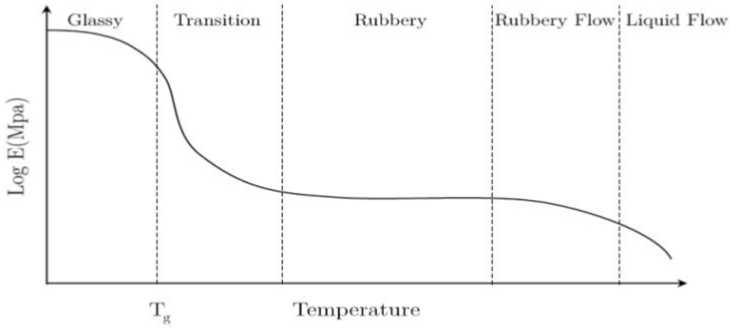


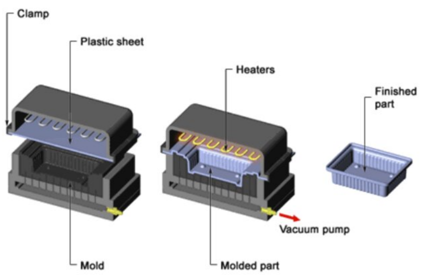
Figure 2.2: Young's Modulus vs Temperature for PEEK [12].

"Compression Induced Solidification" (CIS) [13, 14]. Using this technique, the user can manipulate the glass transition temperature concerning their manufacturing applications.

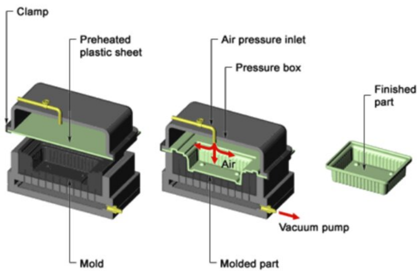
2.1.1. THERMOFORMING PROCESS

Thermoplastic parts can be manufactured by a myriad of techniques [15, 16]. However, this thesis focuses on the thermoforming or stamping process which is similar but not the same as Airborne's Falcon line. Falcon's Line best description is **press consolidation**, but it has some similarities with different forming techniques (so, the description of theoretical techniques is for the reader to get an understanding of the several existing forming techniques of thermoplastic products). Thermoforming is a manufacturing process where the thermoplastic polymer is heated until it is easily pliable so that it can be introduced into a mold by applying pressure such that the product is formed into the desirable shape after consolidation. The process is mainly used for smaller scale products but it is suitable for high production rates [17]. Moreover, a difference between regular thermoforming techniques and Falcon Line is that some thermoforming techniques use an already consolidated part due to the nature of thermoplastics to be remolded [18]. As shown in Figure 2.3, there are many different types of forming techniques (e.g. vacuum forming, pressure forming and mechanical forming) with their own advantages and limitations (cost, intended shape for the product, application, etc.). After consolidation, the excess material is then trimmed away and the formed part is released. Excess material can be reground, mixed with unused plastic, and reformed again into new thermoplastic sheets.

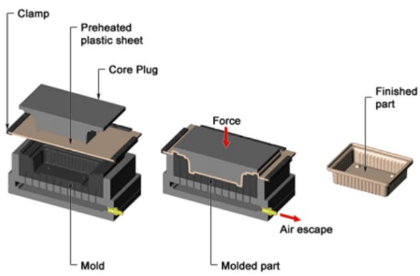
Thermoplastics and especially thermoplastic composites can have significant residual stresses after being manufactured. Residual stresses are the stresses that remain in the material after the originally induced stresses have been removed [19]. This kind of stresses can be desirable or undesirable, depending on the application. This thesis aims at creating an automated finite element analysis process to simulate the effects of residual stresses on thermoplastic composite parts manufactured by thermoforming. Therefore, understanding the origin of residual stresses is of key importance.



(a) Vacuum Forming



(b) Pressure Forming



(c) Mechanical Forming

Figure 2.3: Thermoforming techiques

2.2. ORIGIN OF RESIDUAL STRESSES

The residual stresses are generated during cooling of the material from its final hold temperature. Since these stresses are generated inside the material, they are typically separated into three main categories:

1. *Micro-mechanical or constituent level* where the mismatch between coefficients of thermal expansion (CTE) [20] is the most important factor. After heating the material to a temperature higher than T_g or T_m, during cool down, the thermoplastic matrix tends to shrink volumetrically at a different rate than the fibres [21]. For example, Figure 2.4 illustrates the effect of the small negative CTE of carbon fibers that induces longitudinal compressive stresses in the matrix when heating the composite, as opposed to the tensile stresses that originate from the thermoplastic matrix on the transverse direction. A typical value for the CTE of the carbon fibers is $a_C \approx -0.1 \cdot 10^{-6} K^{-1}$ while a polycarbonate matrix would have $a_m \approx 65 \cdot 10^{-6} K^{-1}$. Evidently, higher CTE mismatches cause larger residual stresses. Note that the matrix's CTE is temperature dependent [22].

2. *Meso-mechanical or lamination level* concerns ply to ply interactions due to the stacking sequence. Figure 2.5 summarizes this phenomenon, where interlaminar shear stresses arise between layers. If the laminate is unbalanced, for example considering a stacking sequence of [0,90,90] (which is 3 layers of the specific orientation of unidirectional laminates), then it will bend even when subjected to an axial force – see Figure 2.8 for a finite element simulation conducted by the author of

this thesis. Ply thickness or the presence of multiple plies with the same fiber direction (“block effect”) also affect the residual stresses that arise, as the "block" has high elastic modulus that causes high shear stresses to the adjacent layer, eventually causing bending [21].

3. *Macro-mechanical or Global level* pertains to boundary effects that cause, for example, thermal gradients in the material. Different thermal distributions through the thickness of the material can introduce compressive residual stresses at surface plies and tensile stresses in the centre plies, as illustrated in Figure 2.6. Unbalanced cooling and thick laminates can also affect material phase formation in different locations, for example surface plies could have different phases than centre plies (important for semi-crystalline polymers [23]). This, will create constraints in the material where the surfaces solidify quicker than the centre [21].

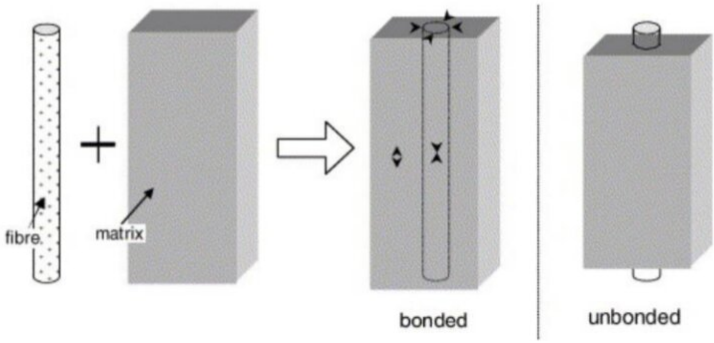


Figure 2.4: *Micro-level*: Compressive stresses from the fibres and tensile stresses from the matrix [21].

Cooling strongly affects the solidification/liquefaction of the composite through the thickness, and can cause different phase transformations [24] (in this thesis, only change in properties and not phase transformations are applied and used). The temperature in which the material does not have internal stresses is called stress-free temperature (SFT), and corresponds to a state where polymer chains have enough kinetic energy to avoid entanglements. Close to T_g, both elastic and viscous properties of the material become important, so stresses start to build up as the temperature decreases. Figure 2.9 shows a PVT diagram with the variation of specific volume when cooling in an isobaric environment for amorphous thermoplastic materials [13]. Comparing the 3 pressure configurations p_0, p_1 and p_2, where $p_0 > p_1 > p_2$, increasing the pressure, the glass transition temperature will decrease respectively $T_{g0} < T_{g1} < T_{g2}$. In this diagram, the previously mentioned effect of pressure on the T_g is evident, as well as the variation of free volume with both pressure and temperature.

The **cooling rate** is another parameter with significant importance. As the cooling rate increases, there is less time for residual stresses to relax and unwanted deformations occur in the final material. Figure 2.10 shows how the glass transition temperature T_g is affected by the cooling rate. The cooling rate also affects adhesion between the two constituents, as increasing the cooling rate leads to interfacial shear stress [21] which can lead to fibre

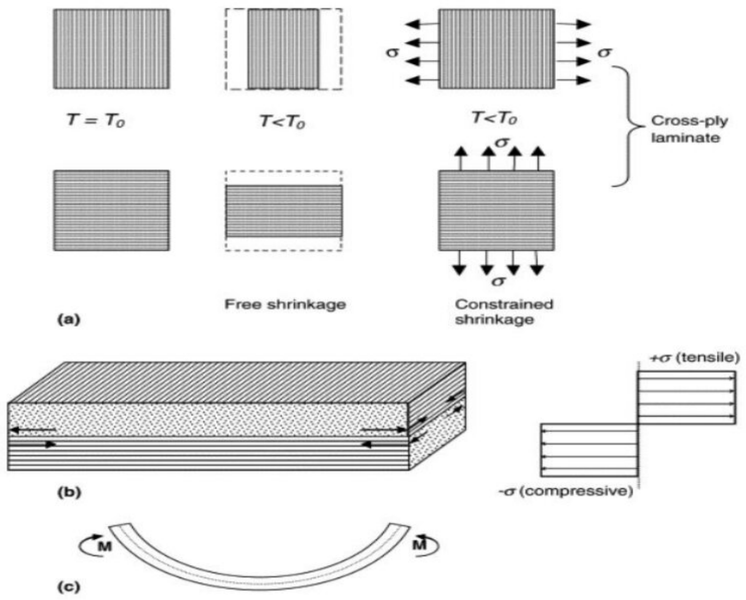


Figure 2.5: *Meso-level*: Interlaminar interaction concerning the layup sequence and block effect [21].

debonding and interfacial crack growth between the laminates.

The effect of cooling rate on composite residual stresses has been investigated by different authors. Guo et al. [25] created a micro-scale finite element model that predicts the response of a glass fibre/epoxy composite subjected to thermal stresses. Teixeira-Dias et.al [26] investigated similar effects for metal matrix composites. The accuracy of these and other investigations hinges on the quality of the material laws needed in the finite element analyses. They can include linear elasticity, viscoelasticity and friction models that affect the generation of residual stresses. Since residual stresses are sensitive to the drop of temperature between processing and working temperature, the higher the ΔT, the higher the final value of the residual stresses and also the higher the strain in the respective material. To calculate that, the simple equation:

$$\varepsilon_{thermal} = \alpha \cdot \Delta T \tag{2.1}$$

can be used, where $\varepsilon_{thermal}$ are the strains in each direction that were developed only from the temperature drop, and α is the material's CTE. For linear elasticity, the strains are independent of the cooling rates, depending uniquely on ΔT. If the effect of moisture is also important, the equation $\varepsilon_{moisture} = \beta \cdot \Delta C$ can be used where β is the respective hygroscopic expansion coefficient and ΔC is the difference between a dry composite and a composite with moisture

$$C = \frac{Moisture_{mass}}{Dry_{mass}} \cdot 100$$

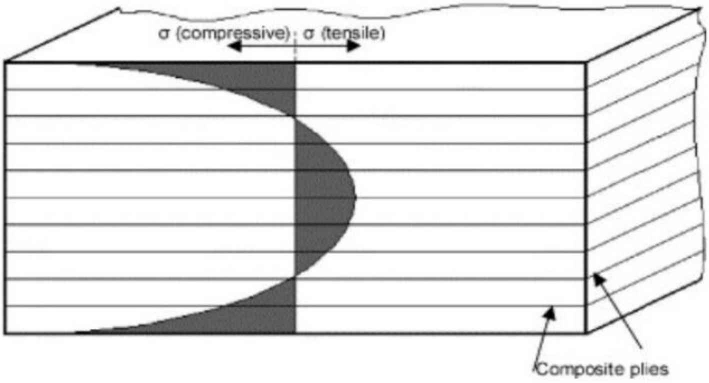


Figure 2.6: *Macro-level*: Stresses gradient through the thickness of the material [21].

Figure 2.7: Schematic illustration of the boundary conditions for Figure 2.8.

[4].

Also, phase changes can affect the strains in the material, so the most appropriate equation [20] is

$$\varepsilon_{total} = \varepsilon_{thermal} + \varepsilon_{moisture} + \varepsilon_{PhaseChanges} \tag{2.2}$$

The above assumptions do not include any applied stress but only a temperature difference.

Tool-part interaction can also lead to the development of residual stresses [5, 21], as introduced in section 2.3.4, because the presence of tool affects the transfer of heat and can introduce friction. Concerning heat transfer, if there is a tool on only one side of the part, then this will cause a temperature gradient due to different heat transfer properties. This temperature gradient can affect the material microstructure in different locations, eventually leading to residual stresses and warping (*Macro-mechanism*). In composite laminates, plies that are closer to the metal part tend to cool down/heat up more quickly (due to high thermal conductivity of the metal tool) and solidify faster, while plies from the other side that are closer to the mold (compatibility layer) remain at higher temperature and experience a phase change later. Figure 2.11 shows a heat transfer model, where the top plies tend to heat up quicker than the bottom due to the addition of extra mate-

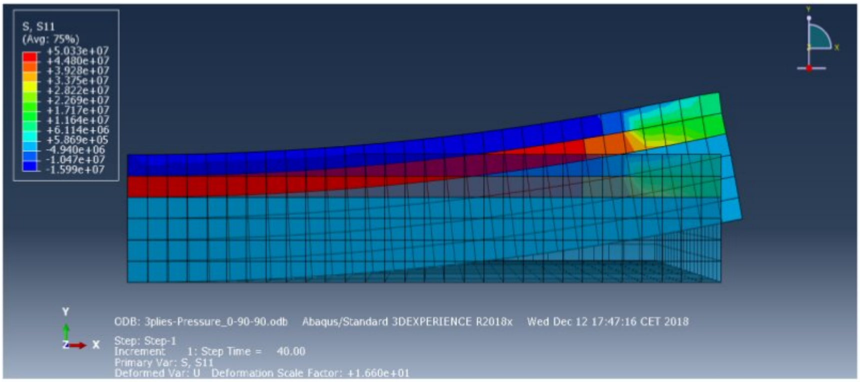


Figure 2.8: Abaqus CAE simulation of [0,90,90] with a tensile stress $\sigma = 10MPa$.

rial (compatibility layer sheet - used for simulation purposes) on the bottom of the laminate, introducing a temperature delay. The specific model was built up for the sake of visualization of the temperature gradient in the material (the dimensional scaling is not correct). Concerning the friction interaction between the interface of the tool and the adjacent composite ply [27], this phenomenon arises from different CTE of the tool (usually higher) and the composite which causes the bottom layers to deform as they will follow the expansion of the tool. This creates different morphology on both sides, which will eventually introduce a bending moment and warpage [5]. Figure 2.12 shows the deformation when the composite is subjected to pressure.

The next sections introduce analytical models describing *heat transfer* in the press, *calculating residual stresses with linear elasticity and viscoelasticity* and also *tool-part interaction* model. Note that there are many more parameters that can cause residual stresses that are not analyzed herein, such as **fiber misalignment**, **imperfections/voids** in the material, **fiber volume fraction**, **lamination sequence**, etc. [20]. Given the lack of knowledge of the microstructural details of the material in this thesis, these effects are neglected henceforth.

2.3. PREDICTING RESIDUAL STRESSES IN THERMOFORMING

Thermoforming involves heat transfer, mechanical deformation at high pressure and tool-part interactions. Computational predictions of the process need to address these three aspects. In this work, the multi-scale nature of composite laminates is neglected, and only continuum-level modeling strategies are discussed.

2.3.1. HEAT TRANSFER ANALYSIS

The link between temperature gradients and residual stresses implies a need to predict how heat transfers during the manufacturing process of composites [28]. The fast manufacturing processes for thermoplastics mean that there is short consolidation time, therefore the heat transfer processes are expected to be **transient** (as opposed to steady state). Analytical models provide simple closed-form solutions to practical heat transfer problems such as the one of interest in this thesis. Since the material is introduced in the press

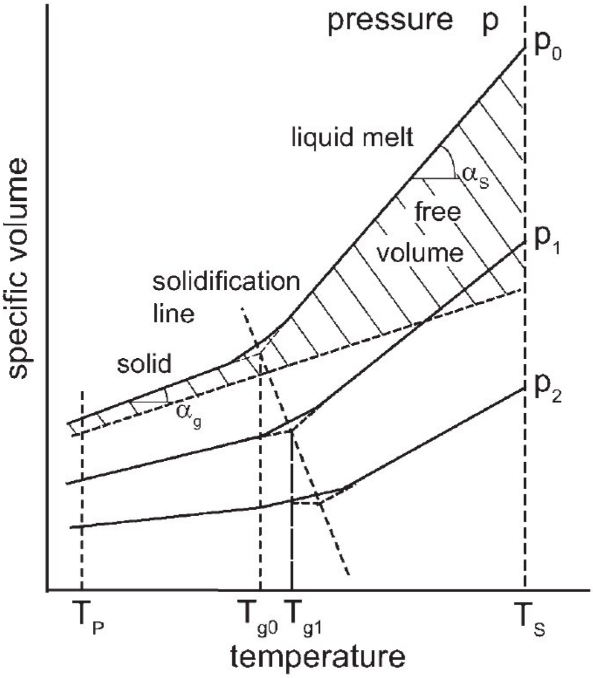


Figure 2.9: Temperature dependence of the specific volume of an amorphous thermoplastic for various pressures (isobaric cooling) [13].

and heat is transferred from both sides, the analytical model presented in [29] is selected herein, which is written as a function of the Dimensionless Fourier number:

$$\frac{T_m - T_o}{T_i - T_o} = F_o = \frac{kt}{\rho \cdot (C_\rho) \cdot (x_m)^2} \tag{2.3}$$

where T_m = desirable temperature at the mid-plane (^oC), T_o = imposed surface Temperature (^oC), T_i = initial material's Temperature (^oC), F_o = Fourier number, k = thermal conductivity ($\frac{W}{m \cdot {}^oC}$), ρ = density ($\frac{kg}{m^3}$), C_ρ = specific heat ($\frac{J}{kg \cdot {}^oC}$), x_m = distance from the surface to the center (m), t = time needed for the center to be heated at T_m (sec). Figure 2.13 can be used for practical predictions.

These classical results are useful to establish a baseline comparison with finite element models. When first learning a commercial finite element software such as Abaqus, I ensured that the predictions were correct by comparing with this analytical model using two different homogeneous materials (Copper and respective values for Composite CFRP). The simulation results are shown in Figure 2.14, and correspond to the input properties shown in Table 2.1 chosen just for illustration purposes as an average from the literature [30], and for the boundary conditions described in Table 2.2. The simulations agree with the analytical model and predict that the time needed to heat the middle of the material

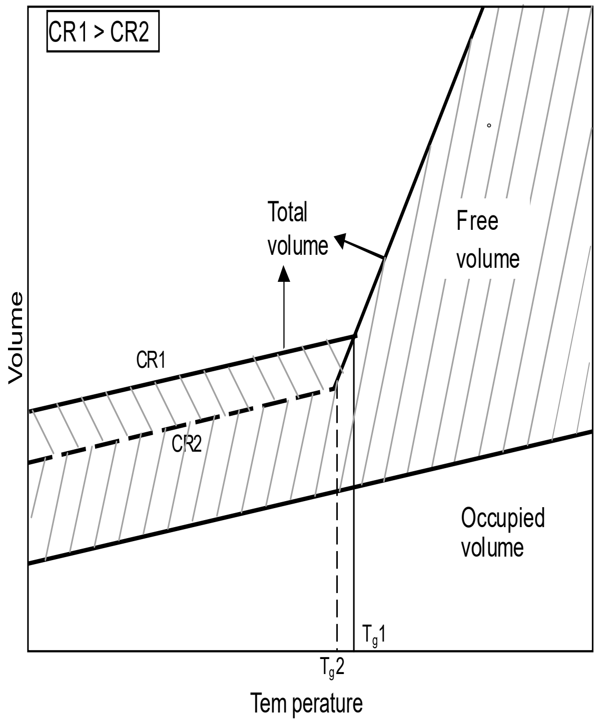


Figure 2.10: Effects of the cooling rate (CR) on the free and occupied volume of the material [21].

is $t = 35s$ for the Copper and $t = 381s$ for the Composite.

2.3.2. CONSTITUTIVE MODELING

A key aspect in accurately predicting the thermoforming process is the quality of the constitutive models used in the analysis. In the literature there are investigations that simply assume linear elastic behavior of the composite, while others consider their viscoelastic properties. One work that analyzes and compares both behaviors is Ghayoor's et. al [12, 31]. In the work, an automated placement laminate process was used for analyzing the residual stresses that are developed in the laminates during placement. The corresponding process is similar to thermoforming processes, as it is also using temperature to make the laminates pliable and also pressure for consolidation. Also, the consolidation times are small for the same reason that the material used is a thermoplastic. Moreover, the metal plate that the laminates were positioned was assumed to be the tool of this specific process, introducing residual stresses through tool-part interaction (as a thermal gradient or mechanical). Both elastic and viscoelastic behaviors were highlighted, comparing how the internal stresses were developed through the thickness of the material for each material model. Next sections will introduce the mathematical models for each constitutive model separately.

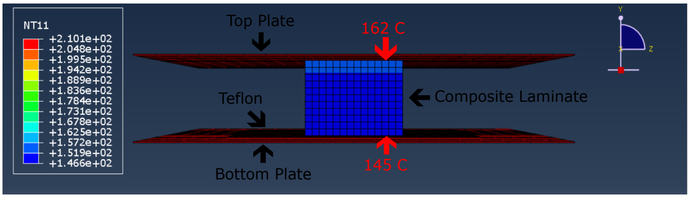


Figure 2.11: Abaqus CAE Heat transfer simulation.

Table 2.1: Reference Properties

Properties	Copper	CFRP
$k \sim \frac{W}{m \cdot {}^oC}$	380	10
$C_\rho \sim \frac{J}{kg \cdot {}^oC}$	385	919
$\rho \sim \frac{kg}{m^3}$	8940	1390
$x_m \sim m$	0.075	0.075

LINEAR ELASTICITY

From the previous section, the thermal strains due to the temperature difference can be calculated from equation 2.2. Assuming that the material has an amorphous structure, that moisture absorption is negligible, and that the composite only experiences small strains as a result of the applied pressure on the top and bottom surfaces, the effects of elasticity can be predicted at two different levels: micro-mechanical and meso-mechanical level [4, 19, 32].

Calculating these stresses at the **micro-mechanical level** follows the linear elastic law [27]:

$$\sigma_{residual} = S \cdot \varepsilon_{thermal} \tag{2.4}$$

where S is the material's stiffness matrix and the deformation is uniquely associated to thermal strains. However, in composite materials the stiffness matrix in equation 2.4 depends on the stiffness of the fibers and of the matrix, so that equation can be rewritten as [27]:

Table 2.2: Reference Temperature

Parameters	Values
$T_o({}^oC)$	200
$T_i({}^oC)$	20
$T_m({}^oC)$	150

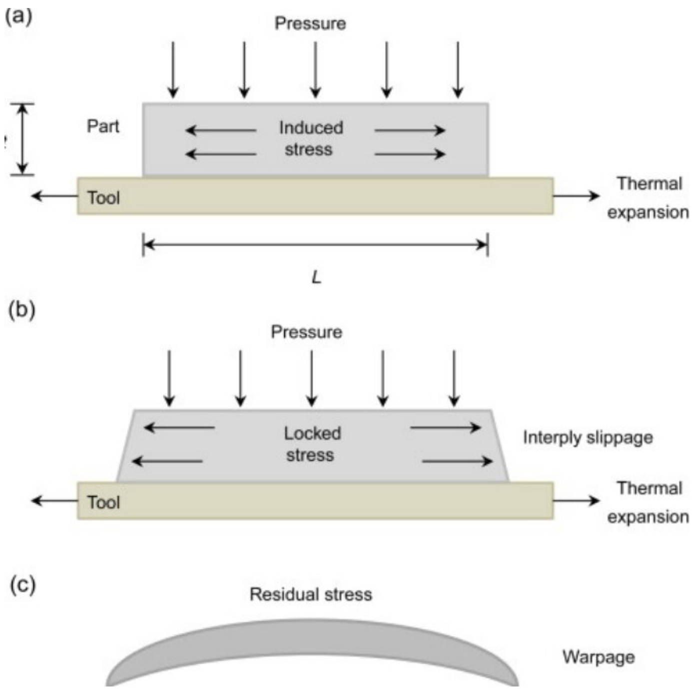


Figure 2.12: The generation of the concave shape due to the interaction between tool-part [20].

$$\sigma_{residual} = \frac{V_f E_f E_m \varepsilon_{thermal}}{E} \tag{2.5}$$

where V_f is the fiber's volume fraction, E_f is the fiber's Young's modulus, E_m is the matrix's Young's modulus and E is the total modulus [27]. This is an elementary result when calculating residual stresses from simplified models at the **Micro-mechanical level** (confront section 2.2).

At the **Meso-Mechanical level**, the simplest model to predict residual stresses follows from Classical Lamination Theory (CLT) [27, 33]. This follows from the calculation of the ABD matrix as explained in introductory books on the subject [4]:

$$A_{ij} = \sum_{k=1}^{n} (Q_{ij}^{(k)})(z_k - z_{k-1}) \tag{2.6}$$

$$B_{ij} = \frac{1}{2} \sum_{k=1}^{k} (Q_{ij}^{(n)})(z_k^2 - z_{k-1}^2) \tag{2.7}$$

$$D_{ij} = \frac{1}{3} \sum_{k=1}^{k} (Q_{ij}^{(n)})(z_k^3 - z_{k-1}^3) \tag{2.8}$$

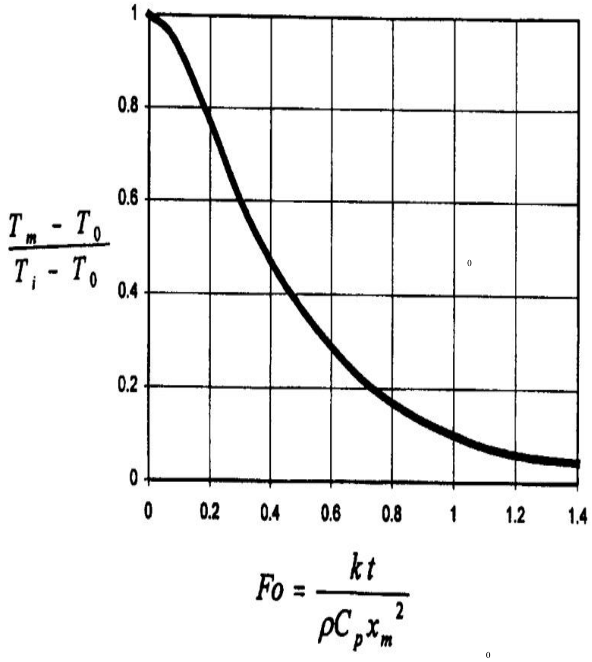


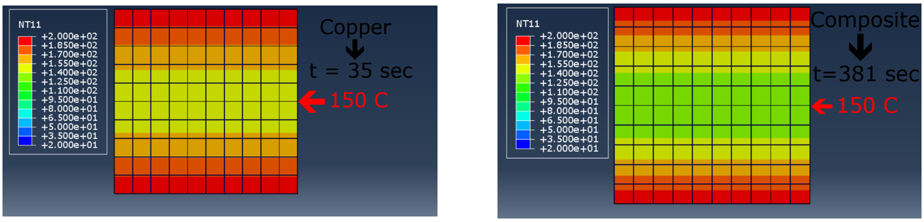
Figure 2.13: Plot for calculating the temperature T_m at the midplane of a plate as a function of time after the two surfaces are suddenly raised to T_o [29].

where $Q_{ij}^{(k)}$ is the stiffness matrix of ply k and z_k is the distance of the k-ply from the center of the laminate. For additional details on CLT, the reader is referred to Isaac et.al [4]. The assembly of the ABD matrix enables to predict strains and curvatures/warpage of the composite laminate under mechanical or thermal loads. In the case of thermal deformation, the result becomes:

$$\begin{bmatrix} N_{thermal} \\ M_{thermal} \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} \begin{bmatrix} \varepsilon_{thermal} \\ k_{thermal} \end{bmatrix}$$

where $N_{thermal}$ and $M_{thermal}$ are the force and moment per unit length, and $\varepsilon_{thermal}$ and $k_{thermal}$ are the strains and curvatures of the midplane of the laminate.

Linear elastic predictions for small strains are trivial predictions, but often they are unsatisfactory because polymers are viscoelastic which cause the composite to behave viscoelastically as well.



(a) Copper

(b) Composite

Figure 2.14: Heat transfer simulations of two materials to predict the time needed to heat their center up to 150 ^{o}C

2.3.3. VISCOELASTICITY

Most polymers exhibit viscoelastic behaviour under mechanical and thermal loads, instead of being purely linear elastic. Not accounting for viscoelastic effects can severely affect the prediction of residual stresses originated after thermoforming [12, 34–36].

Viscoelasticity is when the material exhibits both viscous and elastic characteristics under stress and deformation. These kind of properties can be found in almost all polymers, due to chain relaxation after an applied stress is removed which imposes a time-dependency to the mechanical response that is a function of the applied strain rate [37]. Viscoelasticity implies energy dissipation, unlike elastic deformation [38]. Since viscosity is the resistance to thermally activated plastic deformation, a viscous material will lose energy after a loading cycle. Moreover, when stress is applied to a viscoelastic polymer, parts of the long polymer chain change positions. This movement or rearrangement is called creep. Polymers remain a solid material even when their chains re-arrange themselves in order to accompany the stresses, and during this accommodation, it creates a back stress in the material [37].

As mentioned, viscoelastic materials experience rate-dependent behavior, i.e. their internal stresses depend of strain rate and time. Viscoelasticity can be linear or non-linear. The simplest theories assume linear elasticity and small strains [34]. If a model is only a function of the degree of cure (only valid for thermosets) and temperature it is labeled pseudo-viscoelastic and can be written as:

$$\sigma(t) = \int_{o}^{t} E(T,a)\frac{d\varepsilon}{dt}dt \tag{2.9}$$

where E is the Young's modulus as a function of temperature and a is the degree of curing (which is relevant only for thermosets).

Models that take into account the contribution of time-dependency are written as:

$$\sigma(t) = \int_{o}^{t} E(t-\tau,T,a)\frac{d\varepsilon}{dt}dt \tag{2.10}$$

Figure 2.15 shows a computational comparison between these two types of linear viscoelastic models [39], comparing computational speed and accuracy when capturing residual stresses.

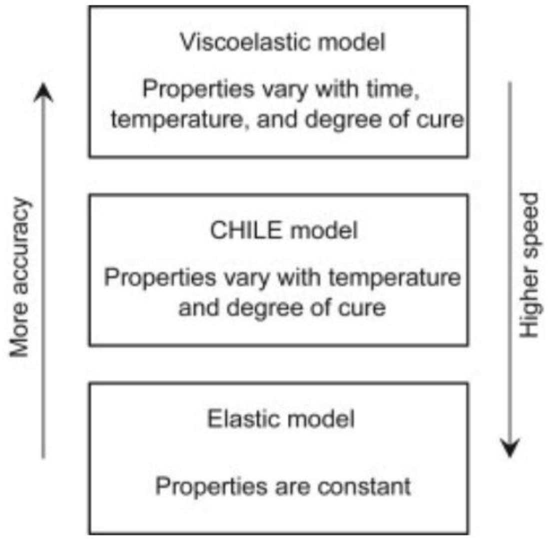


Figure 2.15: Different material constitutive models to predict residual stresses in composites [39].

One important factor about viscoelasticity is the glass transition temperature T_g. This temperature can discriminate the behaviour of the material from rubbery to glassy, although there is a transformation regime that can be called leathery or viscoelastic (Figure 2.16).

At temperatures well below T_g, only elastic bonds can be deformed, so polymers exhibit high modulus and can be assumed as glassy. When the temperature increases, the stiffness of the material will drop dramatically due to the movement of polymer chains when energy is obtained (see Figure 2.17). For thermoset polymers, stresses can be calculated from the crosslink density. If the material is not crosslinked such as thermoplastic composites, the stiffness exhibits a short plateau due to the ability of molecular entanglements to act as network junctions, but in the end the stiffness modulus will decrease to zero, as the material will eventually disassemble and melt.

VISCOELASTICITY MODELS FOR COMPOSITE MATERIALS

Thermoplastic composites are in the viscoelastic state between T_g and a lower temperature which is different for each polymer ($\approx 60-80^oC$) [12]. This state implies that stresses that were induced due to thermal shrinkage of the composite can be relaxed as a function of time. This is possible because molecular chains of the polymer matrix have enough energy to re-entangle and move, causing relaxation of the loads in order to adapt with the stresses. Moreover, viscoelasticity is a time- and temperature- dependent parameter, so cooling rate is the most important factor that affect the generation of stresses. As an example, lower cooling rates give more time to the chains to move and relax and lead to smaller warpage. For cross ply laminates, the residual stresses between the 0° and 90° are discontinuous, which can cause delamination and fracture in the material.

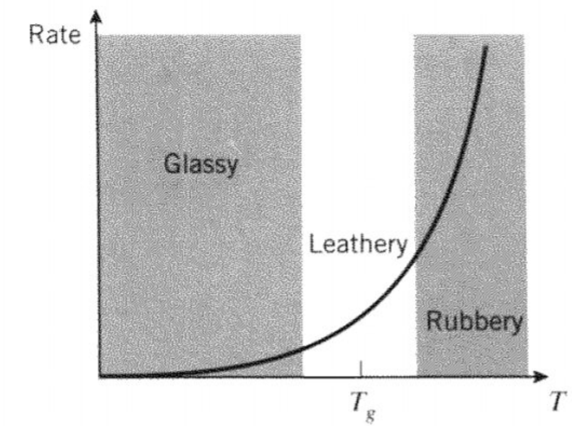


Figure 2.16: Temperature dependence of rate [40].

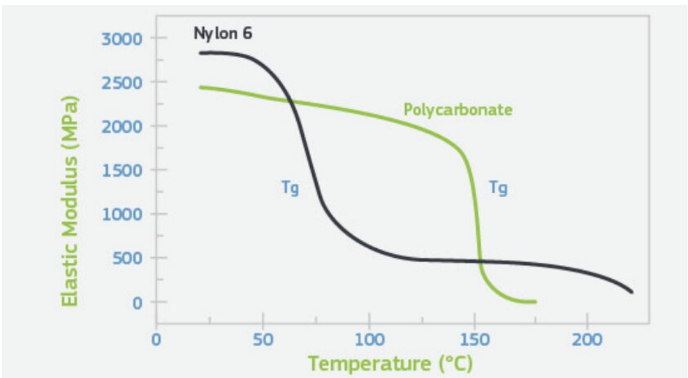


Figure 2.17: Temperature dependence of Young's modulus of amorphous PC and semi-crystalline Nylon [41].

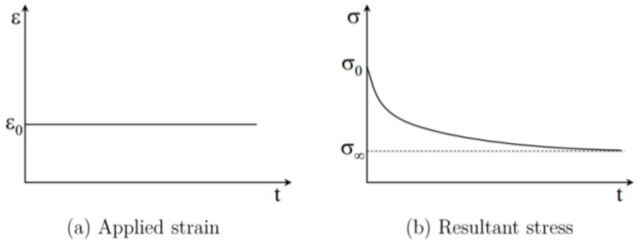


Figure 2.18: Applied strain and the stress relaxation through time [12].

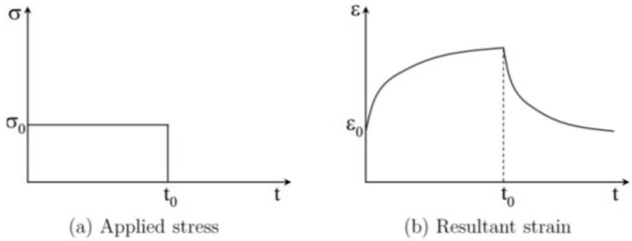


Figure 2.19: Applied constant stress and the resultant strain, especially at time t_o when the step stress is zero (creep recovery) [12].

There are three concepts that are important to consider in viscoelasticity: hysteresis, which reflects the dependency of the system on its loading and unloading history [37]; stress relaxation, occurring when the material is under a constant applied strain ε_o (Figure 2.18); and creep, which is the reverse case, i.e. a constant stress σ_o is applied to the material up to time t_o as the material deforms.

For stress relaxation, the Young's modulus can be calculated as:

$$E(t) = \frac{\sigma(t)}{\varepsilon_o} \tag{2.11}$$

In the case of creep, the strain starts increasing but after the removal of stress it tends to exponentially decrease (Figure 2.19). As with stress relaxation, the creep compliance is:

$$D(t) = \frac{\varepsilon(t)}{\sigma_o} \tag{2.12}$$

The two elementary mechanical models describing the viscoelastic response of polymers (Figure 2.20 are the **Maxwell model** and the **Kelvin model** [12]). The spring represents the elastic behaviour of the material (instantaneous bond deformation [40]) which is the Young's modulus E and the dashpot shows how the material behaves under viscous conditions where μ represents the viscosity. For the viscous part, the stress can be calculated by the following equation:

$$\sigma = \mu \frac{d\varepsilon}{dt} \tag{2.13}$$

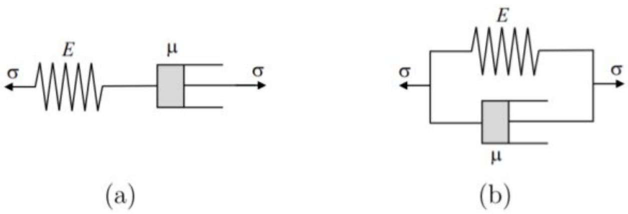


Figure 2.20: a) Maxwell model, b) Kelvin model [12].

Each model captures different behaviours. The Maxwell model is more adequate to capture stress relaxation as opposed to creep due the constraints that the dashpot apply to the spring during deformation.

Maxwell model assumes that:

$$\sigma = \sigma_s = \sigma_d$$

and

$$\varepsilon = \varepsilon_s + \varepsilon_d$$

where the subscript s refers to spring and d to dashpot.

Therefore, the basic relation for the Maxwell model is:

$$E\frac{d\varepsilon}{dt} = \frac{d\sigma}{dt} + (\frac{1}{t}) \cdot \sigma \tag{2.14}$$

Due to the time derivatives, it is difficult to calculate the respective quantities, so experimental data is important. Expanding more, the **relaxed** Young's modulus can be calculated ([12, 40]) as:

$$E_{rel}(t) = kexp(\frac{-t}{\tau}) \tag{2.15}$$

where $\tau = \frac{\mu}{E}$ is the relaxation time, an important parameter to consider in order to perfectly calculate the respective parameters [40].

The **Kelvin-Voigt** model is exactly the opposite. Due to the assumption that the spring and the dashpot are in parallel mode the strains can be assumed equal:

$$\varepsilon = \varepsilon_s = \varepsilon_d$$

and

$$\sigma = \sigma_s + \sigma_d$$

After calculations, the basic equation is:

$$\sigma = E\varepsilon + \mu\frac{d\varepsilon}{dt} \tag{2.16}$$

This model is adequate to capture creep since the strains are assumed equal. The model is governed by the spring and not by the dashpot which gives no stress relaxation.

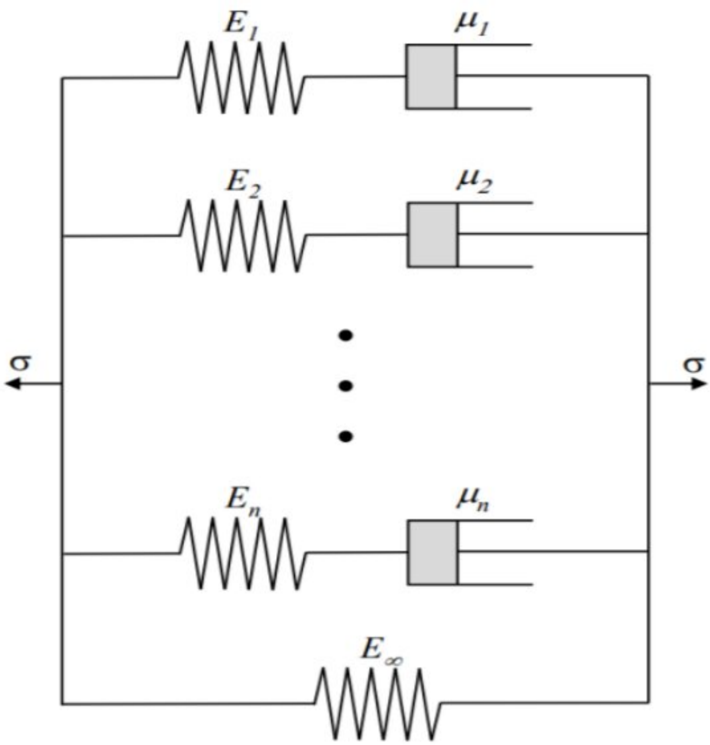


Figure 16: Weichert model [6].

Figure 2.21: Weichert model [12].

The creep compliance (time-dependent creep strain) can be calculated as:

$$J(t) = \frac{1}{E}(1 - exp(\frac{t}{\tau})) \tag{2.17}$$

where τ is the retardation time for creep strains [42].

More complicated models are needed due to the complexity of the polymer chains. Models such as the **Three elements model** which capture the phenomenon using a combination of one elementary model (Maxwell or Kelvin) and one additional spring or dashpot. These models are called Standard models and they are separated in those suitable for solids and those suitable for fluids (an example is given later) [42].

A more generalized model about viscoelasticity is the **Weichert model**, which is a combination of many parallel Maxwell models. This model is good for stress relaxation calculations and can be used when the highlight of the project is to capture the stresses on a material with time dependency (Figure 2.21).

The constitutive equation now is:

$$\sigma(t) = \varepsilon_o(\sum_{i=1}^{n} E_i exp(\frac{-t}{\tau}) + E_\infty) \tag{2.18}$$

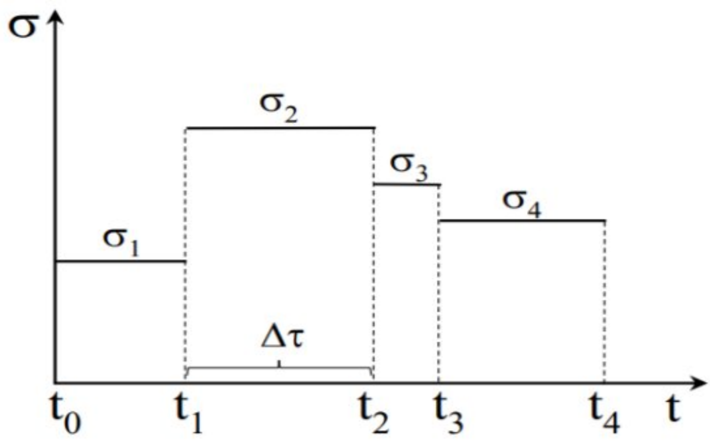


Figure 2.22: Stress vs Time diagram [12].

where i is for each spring and damper set up in the model. Moreover, E_∞ is the equilibrium modulus which explain the minimum stress after relaxation [12].

In practical implementations of viscoelastic models in finite element simulations, considering the stress variation with time is important, i.e. where every cycle has different applied load. For that, the concept of Boltzmann superposition should be introduced (Figure 2.22). In that case, the constitutive equation becomes:

$$\sigma(t) = \int_0^t (\sum_{i=1}^{n} E_i exp(\frac{-t}{\tau}) + E_\infty) \frac{d\varepsilon(t)}{dt} dt \tag{2.19}$$

This equation can be used in finite element analysis simulations, provided that appropriate experimental data is available. Reference [43] details a three-dimensional implementation using tensorial notation. Lastly the Young's modulus as a function of time is written as:

$$E(t) = \sum_{i=1}^{n} E_i exp(\frac{-t}{\tau}) + E_\infty \tag{2.20}$$

This equation is called the **Prony Series**, and represents a basic input of most viscoelastic models used in finite element simulations.

However, composite materials are not isotropic, which means that it is not sufficient to express one Young's modulus as a function of time. Current commercial finite element software do not have implementations for orthotropic viscoelastic models, which introduces practical difficulties in analyzing composite viscoelasticity in a practical setting. The interested reader in more advanced viscoelastic models is referred to references [42, 44]. In this thesis, the focus is on finding practical solutions that are sufficiently simple to quickly implement and use in a practical (industrial) setting. In a recent work, Martynenko [44] proposed a new method in which he merges two finite element models with independent meshes where one is an isotropic viscoelastic model and the other an orthotropic elastic one (Figure 2.23).

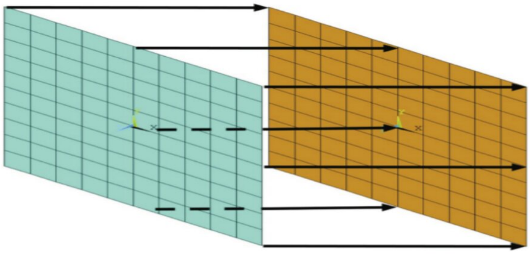


Figure 2.23: Nodes merging [44]

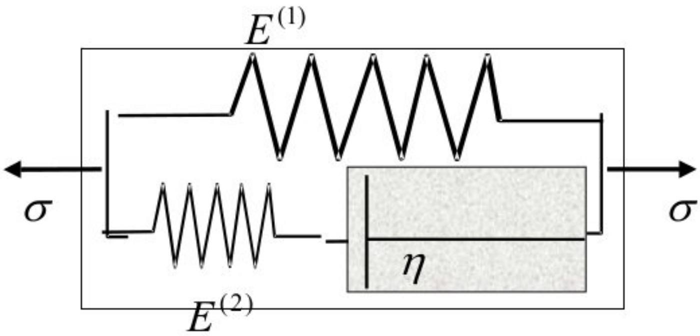


Figure 2.24: Standart Solid II [34].

This modeling strategy of merging the nodes of two material models provides a simple pathway to exhibit both elastic and viscoelastic properties [44]. If the viscoelastic model is the standard Maxwel model, then this modeling strategy is equivalent to the Standard Solid II model [34] shown in Figure 2.24 where the additional spring ($E^{(1)}$) represents a solid with orthotropic elasticity, due to the superposition principle.

The stress can be calculated as:

$$\sigma + \frac{\eta}{E_2}\dot{\sigma} = E_1\epsilon + \frac{\eta(E_1 + E_2)}{E_2}\dot{\epsilon} \tag{2.21}$$

where $\dot{\sigma}$ and $\dot{\epsilon}$ indicate the stress and strain rate that the material undergo and η is the viscosity. This modeling strategy is adopted in this thesis, and additional details are provided in section 5.2.

A final note is included herein concerning the modeling of temperature-dependent viscoelastic properties. A simple strategy that is widely used [12, 45] is to define a master curve. In essence many measurements at different temperatures are considered, and then the shifting of the responses is captuted via the Williams-Landel-Ferry (WLF) [46, 47]:

$$log(a_T) = \frac{-C_1(T - T_o)}{C_2 + (T - T_o)} \tag{2.22}$$

2

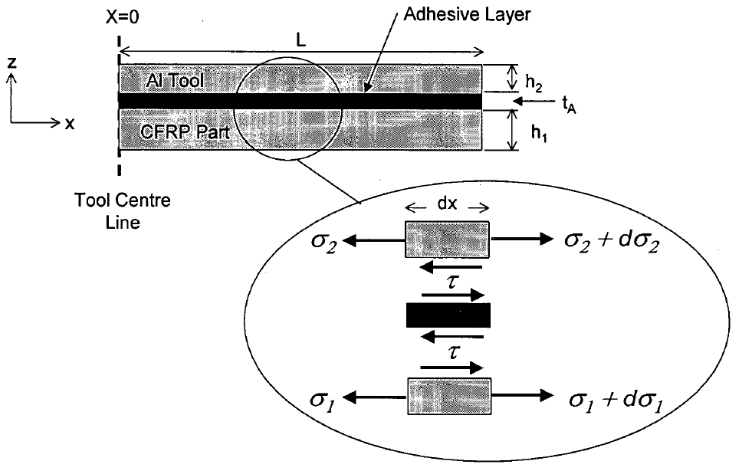


Figure 2.25: Interply/Interface stresses due to slippage [48].

where a_T is the time-based shift factor, T is the measured temperature, T_o is the reference temperature and C_1, C_2 are material parameters.

2.3.4. FRICTION MODEL

Completing the essential models for simulating the thermoforming of composites is the selection of an appropriate friction model. Interaction between the composite and the tool is important because it imposes transversal deformation to the composite (see Figure 2.12). For example, if a metallic tool imposes pressure on the top surface of the composite laminate and the material does not allow slippage, it will create shear stresses at the top and induces bending moments that create a stress gradient between the plies, which can cause undesirable deformation and damage. Twigg et.al [5] proposed analytical models and conducted experiments to confirm the influence of friction. In his master thesis [48], he extensively analyzed all the effects of friction with different parameters such as the **applied pressure**, **material dimensions** and **different tooling**. He assumed that the stress gradient is only developed between the first layer that is in touch with the tool, instead of through the entire thickness of the material (Figure 2.25). With this assumption, the stress can be obtained as [5]:

$$\sigma = \frac{\tau_{Net}(L-x)}{t_{ply}} \tag{2.23}$$

where $\tau_{Net} = \tau_{interface} - \tau_{interply}$, with $\tau_{interface}$ being the shear stress between the first and the second ply. The other parameters can be seen in Figure 2.25.

The interested reader is referred to the original reference for better understanding about the analytical model proposed. As will be investigated in Chapter 4, the interaction between the laminates and the compatibility layer is found to be very important due to the no-slip conditions that occur in practice.

3

INFLUENCE OF THE THERMAL GRADIENT

AUTOMATING the computational analysis of laminates manufactured by the thermo-forming process (Falcon production line) is achieved by creating parametric finite element models for the thermal and mechanical interactions that the laminates undergo. The codes for the simulations of this Chapter are provided in Appendix A.

In this chapter, the focus is solely on the thermal analysis and corresponding influence on laminate warpage, in an attempt to isolate different effects. The chapter starts with section 3.1 defining the various material properties required for these simulations, section 3.2.1 presents the the different consolidation cycles to be simulated via a commercial finite element software (Abaqus), section 3.2.2 includes experimental results of warpage that can be used to establish a baseline comparison with the predictions, and section 3.3 concerns the predictions of warpage by finite element analysis.

3.1. MATERIAL PROPERTIES

The composite laminates are pressed under high temperature. In a first stage, understanding the influence of the heat transfer process without considering mechanical deformation provides important information on whether the laminates experience a significant temperature gradient through their thickness which would cause warpage. The essential parameters for these simulations are the **thermal conductivity** of the constituents and the **thermal conductance** (GAPCON). The latter explains how the thermal energy is conducted between materials that are in contact [49]. Contact between two materials or systems is not ideal, instead there are microgaps of air called asperities that lower the thermal conductivity (or equivalently, the thermal conductance, which is the thermal conductivity per unit thickness multiplied by the contact area). High thermal conductance indicates quick transfer of the thermal energy from one material to the other, which implies that more "surface points" are in contact. Simulations and experiments from many authors have been conducted to establish qualitatively the thermal conductance [50].

Table 3.1: Carbon Fibers: Type AS4 (High strength) [4]

Parameters	Values
Longitudinal Young's Modulus E_{f1} (GPa)	230
Transverse Young's Modulus E_{f2} (GPa)	15
Poison Ratio	0.2
Flexural Modulus G_{12f} (GPa)	27
Linear Thermal Expansion coefficient (longitudinal) a_{1f} $(\frac{1}{K})$	$-0.5x10^{-6}$
Linear Thermal Expansion coefficient (transverse) a_{2f} $(\frac{1}{K})$	$15x10^{-6}$

3

Table 3.2: Sabic's Polycarbonate: Type ALS01 (medium-low flow rate) [8]

Parameters	Values
Young's Modulus E (GPa)	2.35
Poison Ratio	0.37
Flexural Modulus G_m (GPa)	2.3
Linear Thermal Expansion coefficient a_m $(\frac{1}{K})$	$70x10^{-6}$
Thermal Conductivity k_m $(\frac{W}{mK})$	0.2

Due to practical constraints and confidentiality issues, determining the properties of the specific composite laminates of interest is not possible. Instead, this thesis focuses on automating the simulation process and the input material properties were estimated with appropriate literature, as referenced throughout [51–59]. Unquestionably, not measuring the properties directly invalidates a rigorous validation of the simulations since there is significant scatter in the literature for the different properties. Table 3.1 provides the properties of AS4 carbon fibers (high strength fibers) obtained by Isaac et. al [4], while Table 3.2 includes the properties for polycarbonate ALS01 provided by **Sabic** [8]. Moreover, due to the lack of some important properties, articles were used to fill the missing values (from the main references provided above).

3.1.1. COMPOSITE PROPERTIES

Composite's **Density** is calculated by the rule of mixtures assuming a fiber volume fraction of $V_f = 55\%$:

$$p_{composite} = p_{matrix} V_m + p_{fibers} V_f = 1390(\frac{kg}{m^3}) \tag{3.1}$$

The composite elastic properties are estimated from elementary micromechanical models thoroughly explained in appropriate textbooks, e.g. [4]. More accurate models can be considered, without loss of generality of the automated finite element analysis. Con-

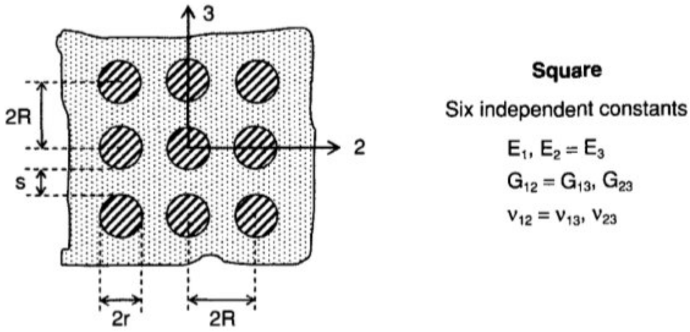


Figure 3.1: Square Properties of a composite laminate [4].

sidering the laminate transversely isotropic, which means that there are 6 independent elastic constants as shown in Figure 3.1, the elementary micromechanical models predict the following properties [4, 60, 61]:

$$E_1 = V_f E_f + V_m E_m \tag{3.2}$$

$$E_2 = E_3 = \frac{E_{2f}}{V_f + V_m \frac{E_{2f}}{E_m}} = \frac{E_{2f} E_m}{V_f E'_m + E_{2f} V_m} \tag{3.3}$$

$$G_{12} = G_{13} = \frac{1}{\frac{V_f}{G_{12f}} + \frac{V_m}{G_m}} = \frac{G_{12f} G_m}{V_f G_m + V_m G_{12f}} \tag{3.4}$$

$$G_{23} = \frac{E_f}{2(1+\mu_{23})} \tag{3.5}$$

$$\mu_{12} = V_f \mu_{12f} + V_m \mu_m \tag{3.6}$$

$$\mu_{23} = 1 - \mu_{21} - \frac{E_2}{3K} = 1 - \frac{E_2}{2K} - 2\mu_{12}^2 \frac{E_2}{E_1} \tag{3.7}$$

where V_f and V_m are the fiber and matrix volume fractions, respectively, $G_{12f} = \frac{E_f}{2(1+\mu_f)}$, $G_m = \frac{E_m}{2(1+\mu_m)}$, $\mu_{21} = \frac{E_2}{E_1}\mu_{12}$ and $K = (\frac{V_f}{K_f} + \frac{V_m}{K_m})^{-1}$ which K being the bulk modulus. The numbering of 1,2 and 3 correspond to the local ply coordinates (1 is the fiber orientation and 2,3 are the transverse directions for a Unidirectional laminate).

Note that the elastic properties of the material are strongly dependent on temperature. As previously discussed (Figure 2.17), the elastic modulus of polycarbonate decreases with the increase of temperature, especially above its glass transition temperature $T_g = 150^oC$.

Table 3.3: Composite's elastic properties as a function of temperature (GPa)

T (^oC)	E_{11}	E_{22}	E_{33}	G_{12}	G_{13}	G_{23}	μ_{12}	μ_{13}	μ_{23}
20	130.47	4.95	4.95	4.62	4.62	1.65	0.276	0.276	0.51
70	130.34	4.51	4.51	4.40	4.40	1.48	0.276	0.276	0.52
120	130.24	4.15	4.15	4.00	4.00	1.36	0.276	0.276	0.53
170	129.90	2.88	2.88	2.50	2.50	0.92	0.276	0.276	0.55
210	129.51	1.23	1.23	0.3	0.3	0.1	0.276	0.276	0.58

According to references [41, 62, 63], and from the previously discussed micromechanical predictions, the composite elastic properties as a function of temperature are calculated and shown in Table 3.3. In these predictions, the properties of the carbon fibers are considered temperature independent.

The composite thermal properties are predicted according to models proposed in [52, 54]. The **coefficients of thermal expansion CTE** and **Specific Heat** are estimated as,

$$a_{11} = \frac{E_f a_f V_f + E_m a_m V_m}{E_f V_f + E_m V_m} \tag{3.8}$$

$$a_{22} = a_{33} = a_{2f} V_f (1 + \mu_{12f} \frac{a_{1f}}{a_{2f}}) + a_m V_m (1 + \mu_m) - (\mu_{12f} V_f + \mu_m V_m) a_{11} \tag{3.9}$$

leading to the values in Table 3.4 based on the constituent properties in [4, 8, 54, 64]). When the material undergoes phase transformations, the specific heat exhibits a peak due to the heat absorption or extraction (so, the experimental points need to be refined around that region).

Thermal conductivity in the longitudinal and transverse directions is also estimated according to elementary micromechanical models,

$$k_{11} = V_m k_m + V_f k_f \tag{3.10}$$

$$\frac{1}{k_{22}} = \frac{1}{k_{33}} = \frac{V_m}{k_m} + \frac{V_f}{k_f} \tag{3.11}$$

from which the estimated values are summarized in Table 3.5.

3.1.2. TOOL PROPERTIES

The steel plates that were used for the simulations are stainless steels type 304 (Table 3.6).

Table 3.4: Composite's Volume Thermal Expansion and Specific Heat

T (^oC)	$a_{11}(\frac{\mu m}{mK})$	$a_{22}(\frac{\mu m}{mK})$	$a_{33}(\frac{\mu m}{mK})$	$C(\frac{J}{kgK})$
20	0.4	39	39	919
70	0.42	40	40	938.4
120	0.39	41	41	996.7
170	0.36	41.5	41.5	1035.5
210	0.33	42	42	1105

Table 3.5: Composite's Thermal Conductivity

T (^oC)	$k_{11}(\frac{W}{mK})$	$k_{22}(\frac{W}{mK})$	$k_{33}(\frac{W}{mK})$
20	20	1.2	1.2
70	21	1.5	1.5
120	22	2	2
170	24	2.5	2.5
210	25	3	3

Table 3.6: Stainless Steel Properties

Parameters	Values
Young's Modulus (GPa)	200
Poison Ratio	0.28
Yield Strength (MPa)	215
Thermal Conductivity ($\frac{W}{mK}$)	16.2
Volume Thermal Expansion ($\frac{\mu m}{mK}$)	17.55
Density ($\frac{kg}{m^3}$)	8000

Table 3.7: Compatibility layer sheet Properties for the computational work [65]

Parameters	Values
Young's Modulus (GPa)	2
Poison Ratio	0.4
Yield Strength (MPa)	30
Thermal Conductivity ($\frac{W}{mK}$)	0.25
Volume Thermal Expansion ($\frac{\mu m}{mK}$)	120
Density ($\frac{kg}{m^3}$)	2200

The bottom part of the laminate is in contact with a compatibility layer for which the properties could only be estimated roughly due to confidentiality issues. Table 3.7 summarizes the properties of the material [65]. The main reference for obtaining these properties is [65], and the material was assumed to be isotropic. In Chapter 4, this additional sheet is considered orthotropic (a composite) due to its influence on the mechanical deformation.

3.2. HEAT TRANSFER ANALYSIS MODULE

Before any experimental result was available, the heat transfer analysis were conducted to provide a qualitative assessment on whether thermal gradients could be responsible for the warpage of the composite laminate. Subsection 3.2.1 details how these simulations were defined, while subsection 3.2.2 presents experimental results (subsequent to the simulations).

3.2.1. HEAT TRANSFER SIMULATIONS

When compared to the material properties discussed in the previous section, the thermal conductance is particularly challenging to estimate, as this property changes not only with the applied pressure and temperature but it is also dependent on the different material interfaces that are in contact. For the process and materials under analysis, typical values can range from 500 to more than 3000 $JT^{-1}L^{-2}\Theta^{-1}$ (typically thermal conductance is a measure in Watts per Kelvin) [66, 67]. Recall that the process under analysis involves different material interfaces that have different thermal conductance.

Nevertheless, since the finite element simulations are parameterized (Appendix A), analyzing the thermoforming process assuming different thermal conductance values between different materials is straightforward. A commercial finite element software is used (Abaqus), and transient **heat Transfer** analysis are conducted to obtain the temperature profiles considering convection and conduction.

For these simulations, two stainless steel 304 plates, a compatibility layer sheet and the composite laminate are considered, as seen in Figure 3.2 and Figure 3.3. A complete simulations involves 6 heat transfer steps, three heating cycles and each of them followed by a cooling cycle as the material is transferred from one press to another (confront with

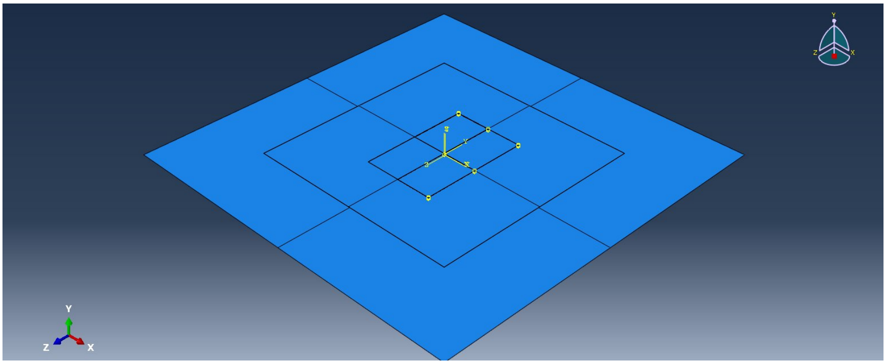


Figure 3.2: Assembly on Abaqus CAE

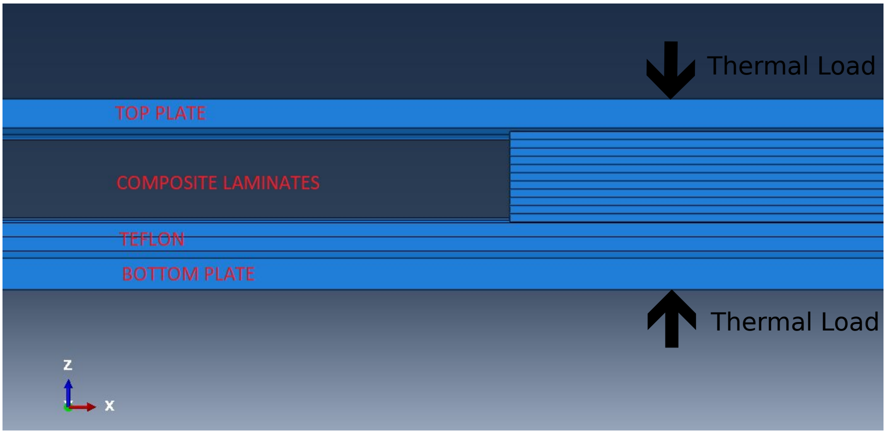


Figure 3.3: Structure layup of the Assembly

Figure 1.1). The cooling mechanism was assumed to be *Natural Convection*, which is applied on the top and bottom of the stainless steel's faces with a value of 7 $\frac{W}{m^2K}$, given past experience with the process.

Thermal conductance (GAPCON) is assumed to be higher at interfaces between stainless steel and each polymer composite (steel with compatibility layer sheet composite, or steel with composite laminate) when compared to the value between the two composites (compatibility layer sheet and laminate). This assumption follows from the smoothness and higher thermal conductivity of the steel plates compared to the compatibility layer sheet and the Laminate. Also in the literature, compatibility layer sheet and PC have GAPCON values in the range of 370-1300 $JT^{-1}L^{-2}\Theta^{-1}$ depending of the applied temperature, pressure and the materials that they are in contact with. For the stainless steel, it is on the range of 1000-2500 $JT^{-1}L^{-2}\Theta^{-1}$ [66–68].

Each simulation of the complete process using different GAPCON values leads to different predictions of the temperature profile that can be assessed at different locations of the laminate. Figures 3.4 and 3.5 assumed $GAPCON_{T-C} = 500JT^{-1}L^{-2}\Theta^{-1}$, i.e. the thermal

3

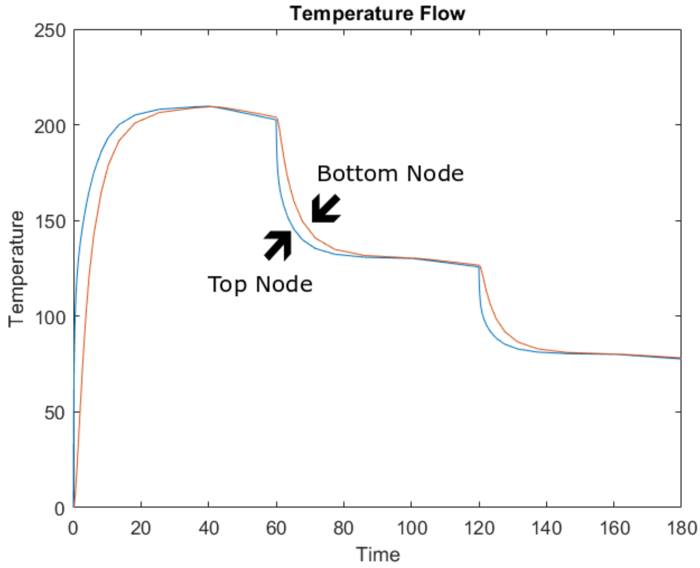


Figure 3.4: Temperature vs Time from the top and the bottom node of the laminate for $GAPCON_{T-C} = 500$ and $GAPCON_{SS-T-C} = 1500$

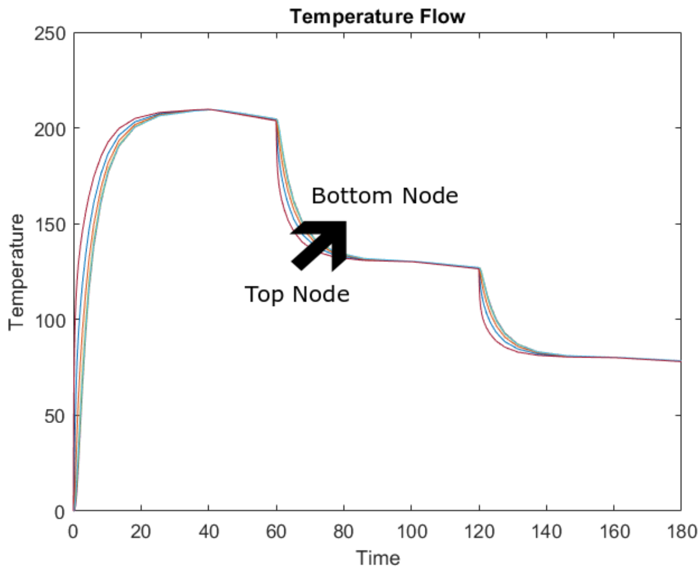


Figure 3.5: Temperature vs Time, nodes through all the laminate's thickness for $GAPCON_{T-C} = 500$ and $GAPCON_{SS-T-C} = 1500$

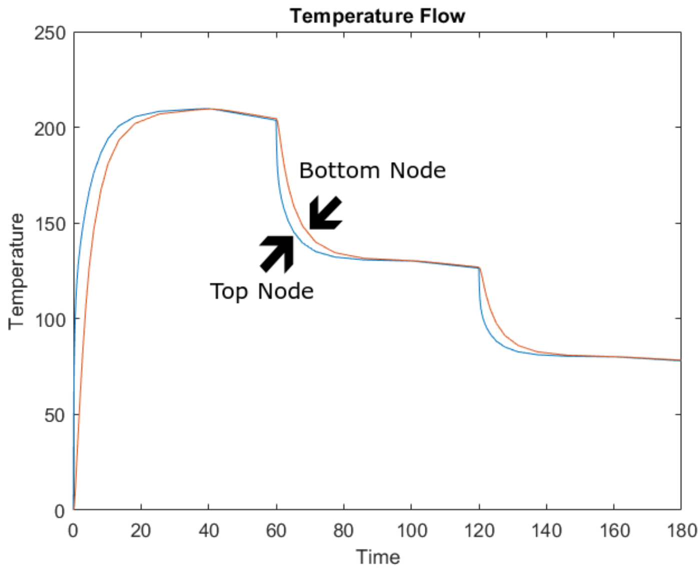


Figure 3.6: Temperature vs Time for $GAPCON_{T-C} = 1000$ and $GAPCON_{SS-T-C} = 1500$

conductance between the compatibility layer sheet and the Laminate is $500JT^{-1}L^{-2}\Theta^{-1}$, and $GAPCON_{SS-T-C} = 1500JT^{-1}L^{-2}\Theta^{-1}$ as the thermal conductance between the stainless steel plates with the compatibility layer sheet and with the Laminate (assumption that due to their polymeric nature, they have the same GAPCON with the stainless steel sides). Figure 3.4 shows the temperature values as the laminate goes through the cycles, where the blue line corresponds to the temperature at the top surface of the laminate (the assigned node is the one at the very top of the laminate which is in contact with the top plate), and the orange line is the first node of the laminate from the bottom, showing how the temperature changes from the compatibility layer sheet side. Figure 3.5 shows similar information, but including more nodes through the thickness of the laminate (a node is assigned from every two plies).

For illustrative purposes, two other simulations are shown where the influence of different thermal conductance values on the temperature profile can be seen. Figure 3.6 presents the heat flow for a different thermal conductance between the compatibility layer sheet composite sheet and the composite laminate $GAPCON_{T-C} = 1000JT^{-1}L^{-2}\Theta^{-1}$ while maintaining $GAPCON_{SS-T-C} = 1500JT^{-1}L^{-2}\Theta^{-1}$. Figure 3.7 shows the heat flow for $GAPCON_{T-C} = 500JT^{-1}L^{-2}\Theta^{-1}$ while considering different $GAPCON_{SS-T-C} = 2000JT^{-1}L^{-2}\Theta^{-1}$. These figures illustrate how the thermal conductance delays the heating and cooling of the laminate, especially when the material is close to its glass transition temperature (recall that PC's $T_g \approx 150^oC$). From Figure 3.4, the top surface cooled down from it's T_g in 63 seconds, but the bottom one only achieves the same temperature after 67 sec. The other two figures show similar heating and cooling times (62-63 sec for the top surface and 67-68 for the bottom). Therefore, the lack of precise values for the thermal conductance does not seem to be a significant issue.

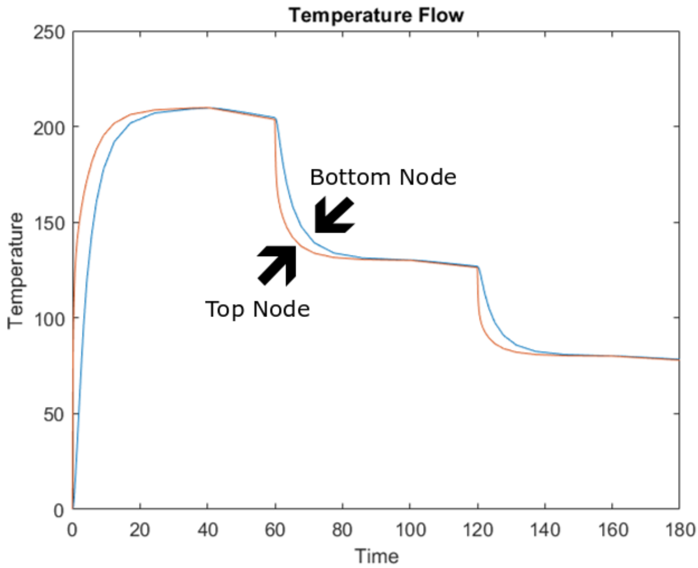


Figure 3.7: Temperature vs Time for $GAPCON_{T-C} = 500$ and $GAPCON_{SS-T-C} = 2000$

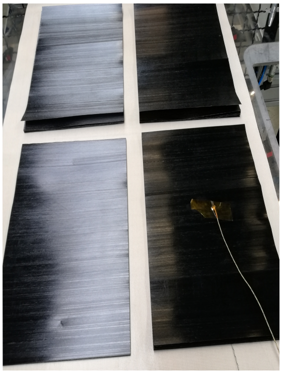
3.2.2. HEAT TRANSFER EXPERIMENTS

The qualitative computational investigation pointed towards an asymmetric temperature profile of the laminate, especially around the glass transition temperature. Therefore, an experimental investigation is conducted to determine the temperature at the top and bottom of the laminate during the thermoforming process. The laminates before and after the process are shown in Figure 3.8.

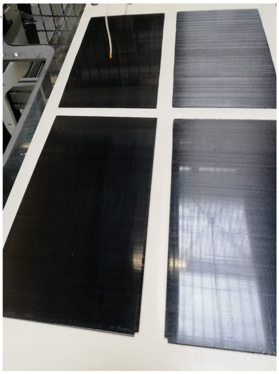
In order to measure temperatures during the process, thermocouples are placed in various locations through the laminate's thickness. Figure 3.9 shows a Temperature vs Time measurement, where the 2 thermocouples are on the top surface of the laminate (between the last laminate ply and the top stainless steel plate) and one at the bottom (between the last laminate ply and the compatibility layer). Figure 3.10 shows result for two thermocouples at other positions through the thickness of the laminate (one between ply 3 and ply 4, and one between ply 8 and 9). Figure 3.11 is similar to Figure 3.9, but where the process is conducted at lower pressure configurations.

Warpage of twelve laminates are analyzed after going through the manufacturing process. Nine laminates with the following stacking sequence $[0_2/90/0/90_2]_s$, and three consisting with a similar stacking sequence but only 7 plies with stacking sequence $[0_2/90/0]_s$. The warpage measurements were manually conducted by measuring the height for each edge (displacement in the out-of-plane dimension; z-axis). The measurements include an offset of $1.4 - 1.5mm$ due to the thickness of the laminates. Only 3 out of 4 laminates that were extracted from each run were used for measurement due to fracture conditions of one of the laminates caused by the presence of the thermocouples (can be seen from Figure 3.8). Tables 3.8, 3.9, 3.10 and 3.11 summarize the measurements.

The Temperature vs Time figures demonstrate that there is negligible temperature differ-



(a) Laminates before the Press



(b) Laminates after the press

Figure 3.8: Airborne configuration of Laminates

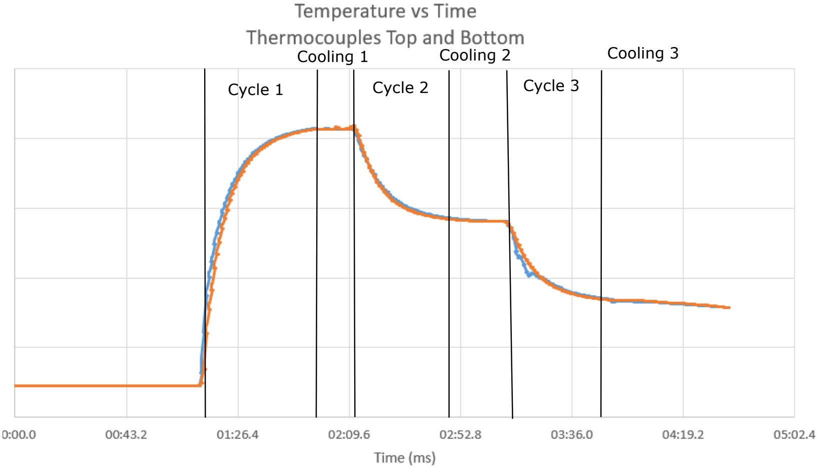


Figure 3.9: Temperature vs time obtained from thermocouples at the top and the bottom of the laminate (laminate has 11 plies).

Table 3.8: First Run with High applied Pressure and 11 plies

First Run				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.6	1.05	0.6	1.3
Laminate 2 (mm)	1.7	0.25	1.25	0.55
Laminate 3 (mm)	1.1	1.2	0.4	1

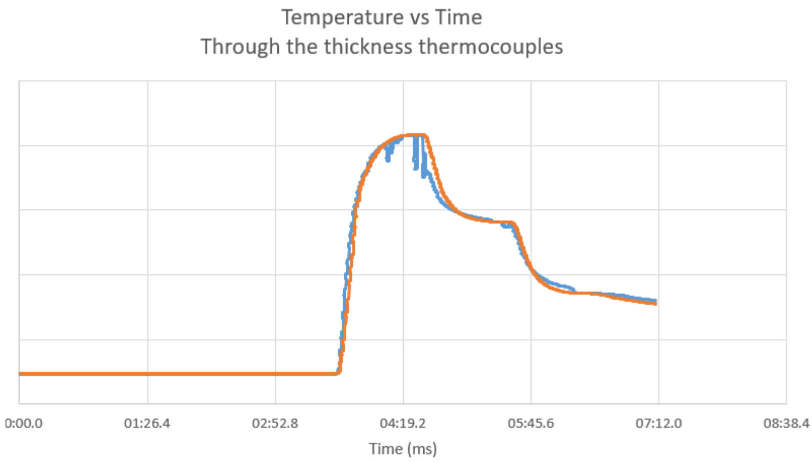


Figure 3.10: Temperature vs time obtained from thermocouples within the laminate (plies 3 and 4, and between 8 and 9).

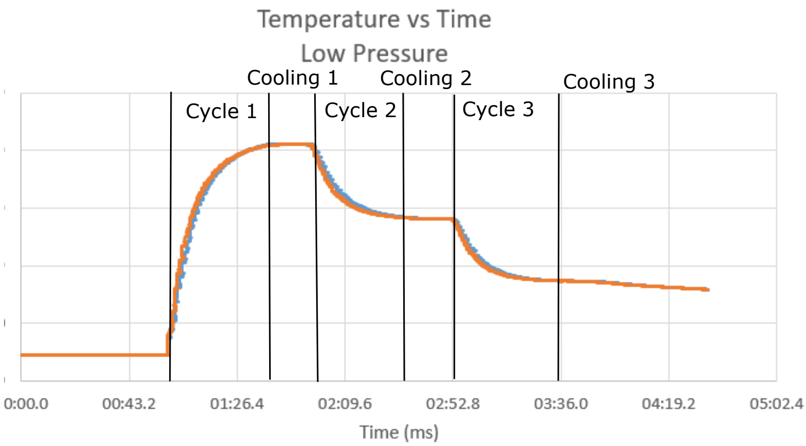


Figure 3.11: Temperature vs time obtained from thermocouples at the top and bottom of the laminate, but using lower pressure.

Table 3.9: Second Run with High applied Pressure and 11 plies

Second Run				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.9	1.7	0.9	1
Laminate 2 (mm)	0.8	1.5	0.7	1
Laminate 3 (mm)	0.9	0.5	1.6	1.4

Table 3.10: Third Run with High applied Pressure and 11 plies

Third Run				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.55	2.1	1.5	1.7
Laminate 2 (mm)	1.65	2	1.2	1.5
Laminate 3 (mm)	1.4	1.4	1.3	0.6

Table 3.11: Fourth Run with High applied Pressure and 7 plies

Fourth Run				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	0.9	0.85	0.3	0.4
Laminate 2 (mm)	0.2	0.3	0.3	0.2
Laminate 3 (mm)	0.2	0.3	0.5	0.5

ence between the top and bottom of the laminates, i.e. there is no significant temperature gradient. In addition, the temperature is stable for several seconds before each pressing cycle (cooling stages of approximately 40 seconds). In some cases, one of the thermocouples showed some measurement errors (fluctuations in Figure 3.10), but the results are consistent through different measurements. Therefore, the presence of a temperature gradient and the subsequent asymmetric buildup of residual stresses is ruled out as the source of warpage. This is explained by the small thickness of the laminates and the compatibility layer sheet. However, Tables 3.8, 3.9, 3.10 and 3.11 also clearly show that the laminates are warped. Therefore, the probable cause for this phenomenon remains unclear at this point (motivating the subsequent investigations of this thesis).

3.2.3. CONCLUSIONS FROM HEAT TRANSFER ANALYSIS

The heat transfer simulations demonstrate that there is a negligible thermal gradient occurring due to the presence of the composite compatibility layer sheet at the bottom of the laminate during the thermoforming process. This conclusion is merely qualitative, as a good estimation of material properties was not possible. However, an experimental verification led to the same conclusion (even less temperature difference is observed).

Figures 3.12 and 3.13 show finite element predictions when the composite compatibility layer sheet is present or not during the manufacturing process, in order to highlight the small thermal asymmetry induced by the composite compatibility layer sheet. If this difference was larger, the top part of the laminate would solidify faster than the bottom which would lead to residual stresses that cause undesirable deformation.

Therefore, the laminate warpage that is observed experimentally cannot be explained by the heat transfer process. Instead, the focus should be on the mechanical deformation and the tool-part interations. Yet, before analyzing the mechanical deformation, the next section 3.3 includes Coupled Temperature Displacement simulations to illustrate the effect of the small temperature differences when the laminate temperature is around the glass transition temperature.

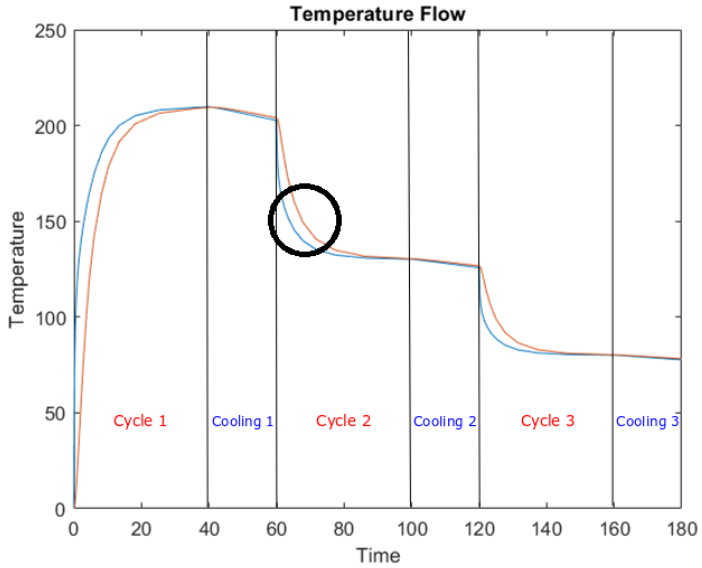


Figure 3.12: Temperature gradient with the use of compatibility layer sheet

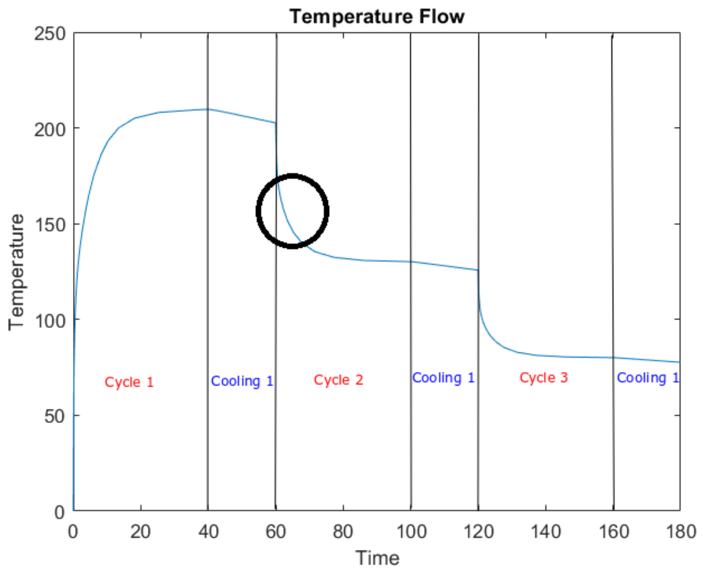


Figure 3.13: Temperature gradient without the use of compatibility layer sheet

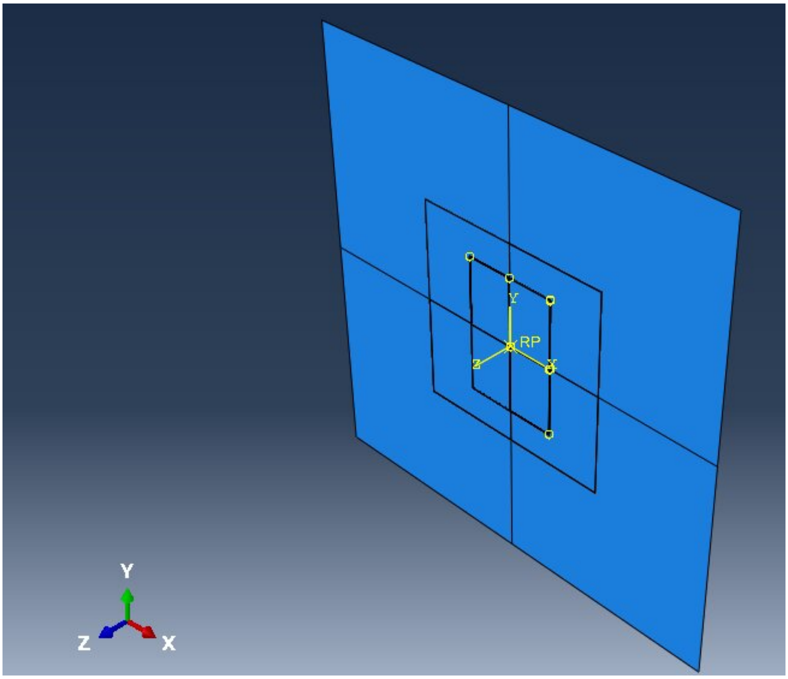


Figure 3.14: Assembly for Coupled Temperature-Displacement simulation

3.3. Coupled Temperature-Displacement Simulations

The temperature difference in the beginning of cycle 2 occurs during the glass transition, where the material goes from rubbery to glassy. At this stage, the material closer to the top would become more rigid and constrain the bottom material, introducing residual stresses. Considering coupled temperature-displacement simulations it is possible to assess what is the distortion caused at the moment when the temperature difference is highest. The top steel plate is omitted in order to let the laminate deform freely. Figure 3.14 shows the setup assembly, and the simulation steps are summarized as follows:

1. The simulations lasts for 20-60 sec (several tries to observe the difference).
2. Surface interactions between the materials are considered, where the tangential and normal behavior is specified, as well as the Thermal conductance (Table 3.12). The values for the tangential behavior are estimated from the literature [69]. The friction between compatibility layer sheet and steel is small compared to the friction between compatibility layer sheet and laminate. For the latter, no-slippage condition is considered [69].
3. The boundary conditions of the problem include a fixed bottom steel plate and a fixed node at the top surface of the laminate (to hold the material in place during deformation).
4. From the heat transfer simulation it is possible to pre-define a temperature profile at the time the material transitioned from a rubbery to solid form, defined here

Table 3.12: Interaction properties

	SS to compatibility layer sheet	compatibility layer sheet to Lamina
Tangential Behaviour (Penalty)	0.04	0.4
Normal Behaviour	Hard contact	Hard contact
GAPCON	1500	500

as the time when the top layers reach a percentage of the glass transition temperature. Looking at figure 3.12 and with the well-said assumption that $T_g \approx 150^oC$, the time the top ply is reaching T_g is different than the bottom ply. This can be illustrated with the circle at figure 3.12, where there is a delay from the bottom side. This specific moment is captured in Abaqus, which represent the temperature gradient. This temperature gradient is super-imposed as the initial condition/state of the compatibility layer sheet and laminate.

5. Surface Convection at the top surface of the laminate and at the bottom of the composite compatibility layer sheet's surface is considered, in order to cool the materials.

6. Meshing of the 2 materials (rigid plates cannot be meshed) with 8-node thermally coupled brick elements (C3D8T elements in Abaqus, suitable for thermo-mechanical analysis).

The Tangential behavior [69] was implemented by a Penalty method that approximates contact without introducing Lagrange multipliers (that can lead to convergence issues [70]). The Normal behavior was used as a Hard contact, which does not let any nodes penetrate the surface. Moreover, Thermal conductance was used with the same values as in the Heat Transfer module.

For comparison, simulations with homogeneous temperature profile as a predefined step were also conducted. Figures 3.15 and 3.16 illustrates the pre and post analysis when an initial thermal gradient is applied.

Unsurprisingly, the temperature gradient would cause warpage of the laminate. Since the central node of the top surface is fixed, all four edges warp outwards with similar values, approximately around 0.9 to 1.2 mm depending the step time, where for 60 seconds, maximum warpage was observed. After that (if the step is bigger than 60 seconds), the material tend to return to its original non-deformed shape due to its elastic nature. Evidently, if there is no temperature gradient, then there is no warpage of the composite laminate – see Figure 3.17.

Note that these simulations are significant simplifications of reality, but they illustrate how the temperature gradient can lead to warpage by creating local stress imbalances at the time when the plies are becoming glassy. However, these simulations do not include viscoelastic effects, neither phase transformations, nor mechanical loads. There-

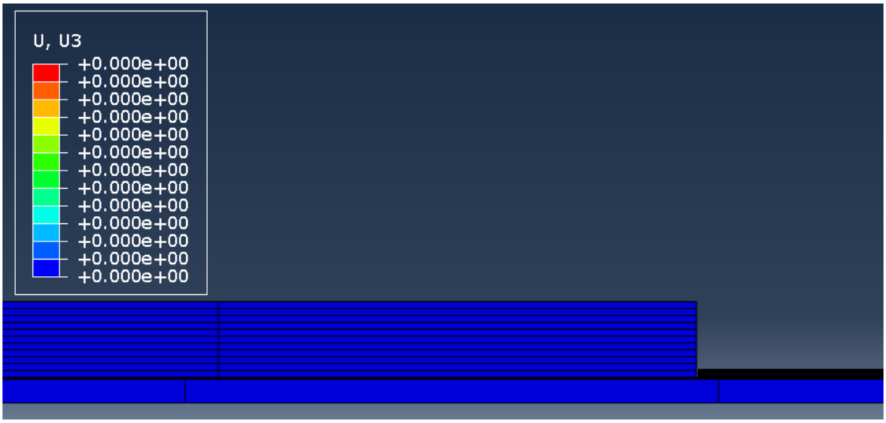


Figure 3.15: Pre-analysis

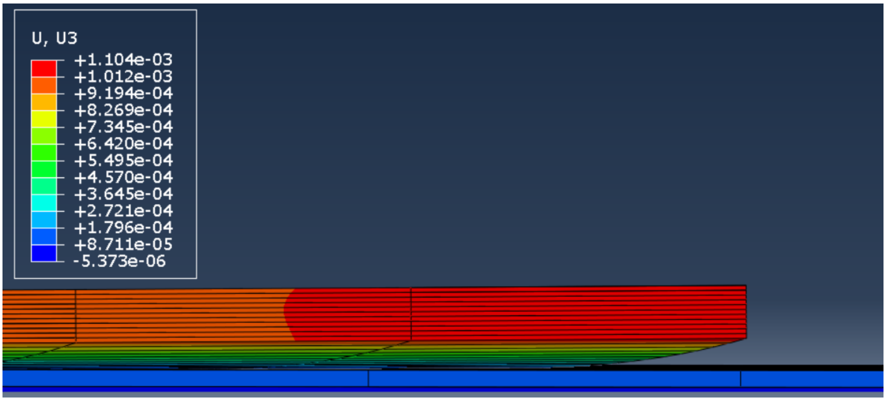


Figure 3.16: Post-analysis with temperature initial gradient

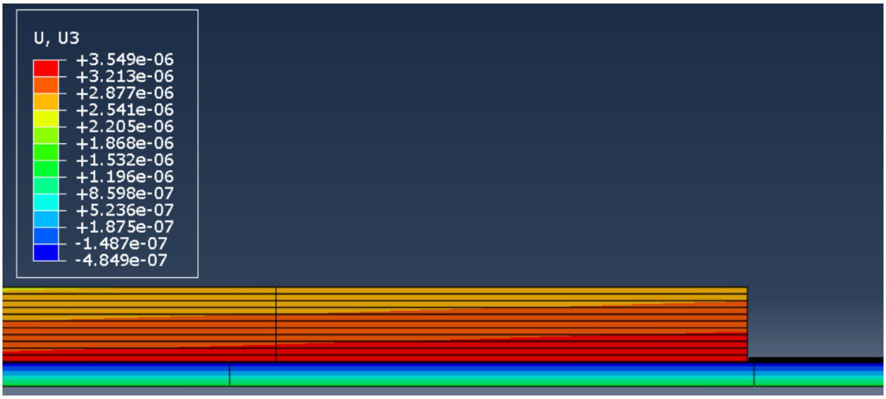


Figure 3.17: Post-analysis with uniform temperature initial profile

fore, these simulations just provide a preliminary estimation about the influence of temperature gradients through the thickness on the warpage of the laminate. Nevertheless, as repeatedly mentioned, this effect is concluded to be small in practice, excluding heat transfer effects as the main cause for warpage.

4

Tool-Part Interaction

Heat transfer analyses and temperature profile measurements have demonstrated that temperature gradients are unlikely to be the cause of significant laminate warpage in the practical thermoforming process under analysis (Chapter 3).

For reminder, placing a tool in one side will introduce several effects:

1. Different heating and also cooling rate comparing both sides that it will introduce a significant Temperature Gradient (Chapter 3).
2. Different friction coefficients between both sides that it will introduce constraints and eventually a stress gradient profile.
3. Introduction of imperfections and fiber misalignment due to the interaction of the laminate with the tool and most important,
4. The compatibility layer will expand due to it's higher thermal expansion coefficient, letting the bottom layers expand with it.

Therefore, this Chapter focuses on the mechanical tool-part interaction and its possible effect on warpage. Similarly to the previous Chapter, the main goal is to automate the analysis process by creating parametric finite element models that can be used for future design and analysis tasks. The reason of that is due to the several input properties of the laminates, compatibility layer (compatibility layer sheet for the simulations) and also the system's configurations that can influence the stress development. The codes for the simulations of this Chapter are provided in Appendix B.

As discussed in Chapter 2 and illustrated in Figure 2.12, in general if the part being formed is subjected to pressure on the top surface and there is a tool at the bottom surface that constrains deformation, when the part cools down it can lock the induced stress state which has a through the thickness gradient. This creates asymmetrical residual stresses that lead to warpage.

The process under analysis has two similar press plates at the top and bottom of the laminate, but there is an additional composite compatibility layer sheet sheet placed in between the bottom press plate and the laminate which can potentially introduce a similar effect because there are different friction coefficients between the different surfaces, and the composite compatibility layer sheet has a higher coefficient of thermal expansion (CTE) compared to the laminate. In addition, warpage could also be caused by manufacturing or microstructural imperfections such as fiber misalignment.

Section 4.1 details the parametric simulations for the tool-part interaction predictions, while Section 4.2 includes additional experiments of the manufacturing process conducted at different pressure cycles to assess the influence of the mechanical load in warpage.

4.1. TOOL-PART FINITE ELEMENT ANALYSES

As discussed in Chapters 2 and 3, most residual stresses in the laminate are arrested when the material stiffness steeply increases after the part is cooled below the glass transition temperature. In the process under analysis, this occurs at the second pressure cycle (see for example Figure 3.12). The first pressure cycle heats up both the laminate and the composite compatibility layer sheet sheet to high temperatures where the laminate is in a viscous state but the sheet is not. In this cycle, the stress build-up at the laminate is expected to be negligible, but the deformation of the bottom layers is larger due to the higher thermal expansion of the sheet compared to the top press plate (metal). During cool down, the laminate transforms from a rubbery state into a solid state and higher stresses start to develop (Cycle 2), so capturing the stress gradient at this moment is expected to be of critical importance for predicting warpage.

The parametric finite element analyses undergo three steps to simulate Cycle 1, Cycle 2 and the warping effect. Each step enables to create appropriate Predefined fields for the subsequent step. Note that the laminate's geometry changes due to mechanical interactions between the laminate closer to the composite compatibility layer sheet sheet but also between the plies themselves [5]. As discussed in Section 2.3.4 and shown in Figure 2.12, the bottom plies of the laminate "slide" as the tool expands (here, the composite compatibility layer sheet).

The 1^{st} step (Cycle 1) is simulated with top and bottom stainless steel plates as Rigid surfaces and assuming frictionless interaction between the plate and the laminate and a very small friction with the composite compatibility layer sheet. Assuming rigid surfaces reduces the computational time and facilitates convergence. This finite element analysis is summarized as follows:

1. Laminate and composite compatibility layer sheet sheet are modeled as 3D solids and the stainless steel plates as Rigid Surfaces.

2. Abaqus Implicit analyses are used, despite potential convergence issues arising from contact conditions, because larger time steps can be considered when compared to Explicit analyses. Nonetheless, convergence issues can occur for some of the automated simulations, depending on the input parameters chosen. It was noticed that by considering smaller simulation time when compared to the real time of the process, convergence of the simulations improved. A second difficulty when preprocessing these analyses pertains the contact heating condition. Rigid surfaces cannot transfer heat (by nature of Abaqus), so in this case, heating conditions are applied on the surfaces of the composite laminate and the compatibility layer sheet

Table 4.1: Interaction properties

	Compatibility layer sheet-BP	Composite-TP
Tangential Behaviour	0.04	Frictionless
Normal Behaviour	Hard contact	Hard contact
GAPCON	1500	1500

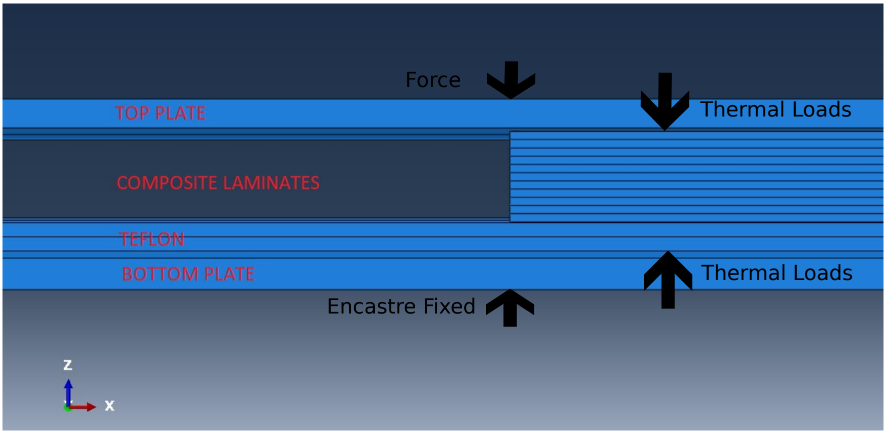


Figure 4.1: Structure layup of the Assembly

(so the heating is faster comparing to the heat transfer simulations from Chapter 3). When imposing the total simulation time, the main condition is to ensure that the material is heated well above it's T_g. As in Chapter 3, **Coupled Temperature Displacement** simulations are considered, using thermo-mechanical finite elements to predict both the temperature distributions as well as the deformation of the laminate. Laminate and composite compatibility layer sheet are meshed with 8-node thermally coupled brick elements (C3D8T in Abaqus).

3. Contact interactions between the top plate and the laminate, and between the composite compatibility layer sheet and the bottom plate are imposed according to Table 4.1. For the interaction between the Laminate and the compatibility layer sheet a no-slip condition is considered (as if the are perfectly bonded, which corresponds to a Tie constraint in Abaqus). Viscous damping is assumed in the contact areas, in order to stabilize the model [71].

4. Boundary conditions are defined by a concentrated mechanical load at the top plate applied in a reference point of the rigid surface. The bottom plate is fixed. Temperature is applied on top of the Laminate's surface and on the bottom of compatibility layer sheet's surface. The analyses start with the Laminate and compatibility layer sheet at room temperature using a Predefined Temperature Field. Figure 4.1 shows the assembly of the complete model. The same simulation strategy is used for Cycle 2 but with different input conditions.

Once Cycle 1 is simulated, warpage can be predicted by removing the plates and applying

Table 4.2: Compatibility layer sheet Isotropic Properties

T (^oC)	$E(GPa)$	μ	$a(\frac{\mu m}{mK})$	$k(\frac{W}{mK})$	$C(\frac{J}{kgK})$
20	2	0.4	120	0.25	970
70	2	0.4	125	0.27	980
120	1.9	0.4	150	0.3	1000
170	1.8	0.4	177	0.32	1100
210	1.78	0.4	200	0.33	1200

a stress gradient created at the end of Cycle 1 as an initial step for the whole assembly (same procedure as the warpage model for the heat transfer module, but this time with stress gradient rather than thermal). Moreover, the Tie condition was also replaced by a friction interaction, in order to let the laminate warp. The cooling simulation is an implicit static analyses where the only applied load in the structure is the stress gradient. The boundary conditions are defined by fixing the center node of the laminate's top surface and also the center node of the bottom surface of the composite compatibility layer sheet. This allows the material to deform according to the stress gradient, causing warpage due to it's unbalanced profile. This step is only for comparison with the stress profile that is generated through Cycle 2.

Cycle 2 is simulated similarly to Cycle 1, but considering different input conditions. Following the work of Ghayoor [12], Cycle 2 starts with an undeformed part but defining two predefined fields (temperature and stress) according to the end of the analysis of Cycle 1. These predefined fields are superimposed in the structure and then the loads are applied according to the specified cycle conditions. Implementing Cycle 1 and Cycle 2 in the same simulation leads to convergence issues; therefore, this strategy is adopted instead. A different strategy was also attempted, where Cycle 2 starts with a deformed structure, but where the thermal loads were implemented differently (convection cooling instead of an assigned temperature), and similar predictions were obtained.

The computational analysis are conducted considering different materials as the bottom sheet in the process. This illustrates the influence of this added material in the process. Therefore, a composite compatibility layer sheet with properties outlined in Table 4.3 is used, as well as an isotropic sheet of pure compatibility layer sheet (Table 4.2). Note that the composite compatibility layer sheet is a cross-ply laminate with two layers (0 and 90 degrees).

4.1.1. SIMULATION RESULTS

CYCLE 1

The choice of pure compatibility layer sheet (isotropic) or composite compatibility layer sheet should affect significantly the warpage of the composite laminate because the coefficients of thermal expansion are significantly higher for the pure polymer. Figure 4.2 shows the undeformed configuration of the analyses, while Figure 4.3 shows the deformed configuration at the end of Cycle 1 when using only pure compatibility layer sheet at the

Table 4.3: Compatibility layer sheet Composite Properties

T (^oC)	20	70	120	170	210
$E_{11}(GPa)$	20.37	20.20	20	19.8	19.1
$E_{22}(GPa)$	6.65	6.5	6.2	6	5.9
$E_{33}(GPa)$	6.65	6.5	6.2	6	5.9
$G_{12}(GPa)$	4.4	4.3	4	3.50	3.4
$G_{13}(GPa)$	4.4	4.3	4	3.50	3.4
$G_{23}(GPa)$	3	2.8	2.6	2.5	2.4
μ_{12}	0.35	0.35	0.35	0.35	0.35
μ_{13}	0.35	0.35	0.35	0.35	0.35
μ_{23}	0.4	0.4	0.4	0.4	0.4
$a_{11}(\frac{\mu m}{mK})$	2.7	2.8	3	3.1	3
$a_{22}(\frac{\mu m}{mK})$	126	127	200	210	250
$a_{33}(\frac{\mu m}{mK})$	126	127	200	210	250
$C(\frac{J}{kgK})$	1000	1050	1100	1150	1200
$k_{11}(\frac{W}{mK})$	0.49	0.5	0.52	0.55	0.6
$k_{22}(\frac{W}{mK})$	0.31	0.32	0.33	0.35	0.4
$k_{33}(\frac{W}{mK})$	0.31	0.32	0.33	0.35	0.4

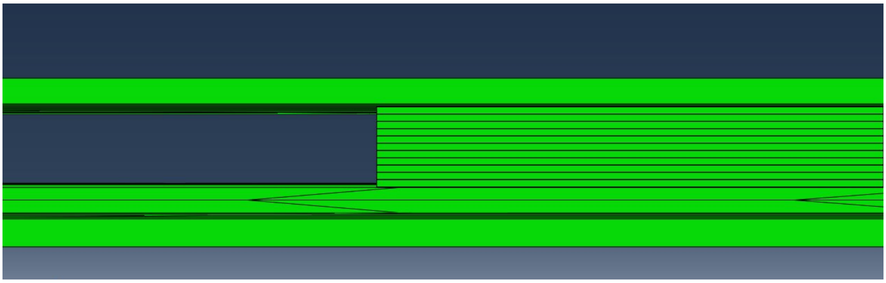


Figure 4.2: Undeformed configuration

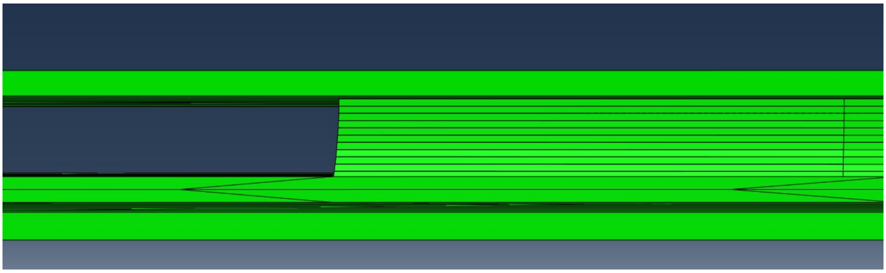


Figure 4.3: Deformed configuration with Isotropic compatibility layer sheet

bottom of the laminate. The deformation of the laminate is clearly visible, as compared with Figure 2.12. Figure 4.4 shows the deformed configuration when a composite compatibility layer sheet is used instead of the pure polymer. In this case, the deformation also exists but it is significantly less pronounced.

The displacement on the x-direction (sliding/dragging of the laminate due to the compatibility layer sheet expansion) is *1.26 mm* for the simulation with pure compatibility layer sheet and *0.09 mm* for the simulation with composite compatibility layer sheet. These values are indicative because, as discussed previously, the input material properties have not been accurately characterized. Figures 4.5 and 4.6 show the stresses along the fibers di-

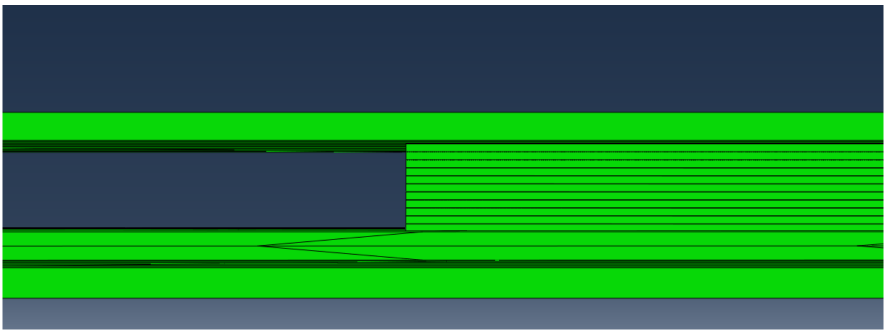


Figure 4.4: Deformed configuration with Composite compatibility layer sheet

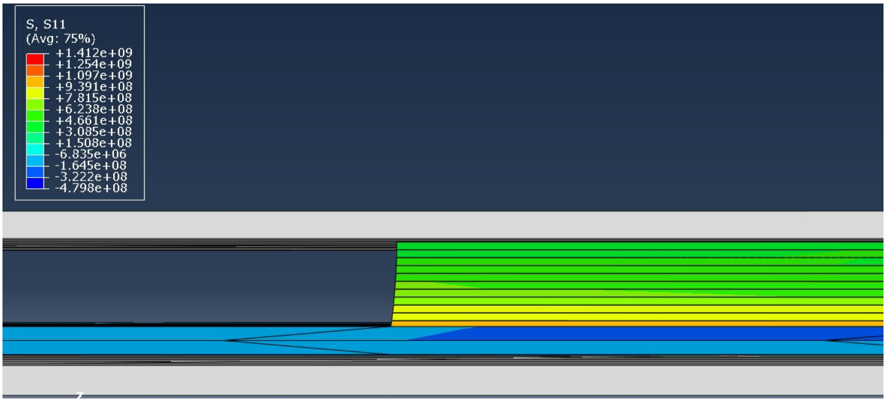


Figure 4.5: Stresses along the fibers direction ($\sigma_{11} \equiv S11$) when using a compatibility layer sheet sheet at the bottom.

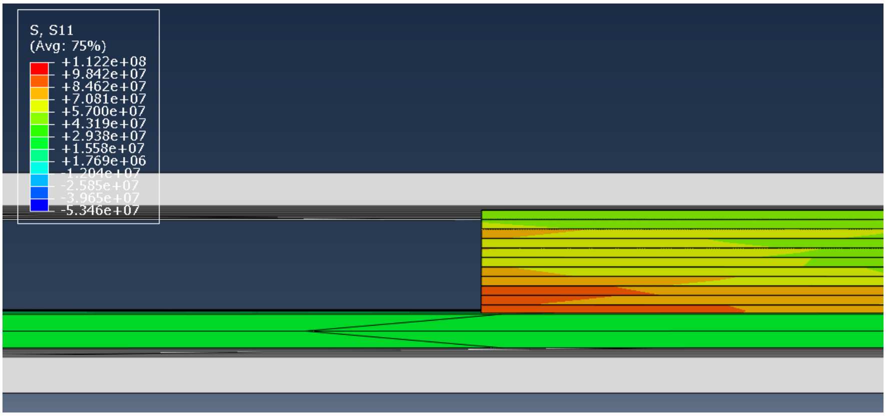


Figure 4.6: Stresses along the fibers direction ($\sigma_{11} \equiv S11$) when using a composite compatibility layer sheet sheet at the bottom.

rection through the thickness of the laminate when considering pure compatibility layer sheet or composite compatibility layer sheet. Note that showing the stresses along the fibers directions ($\sigma_{11} \equiv S11$) can be misleading in the sense that 0^o plies correspond to stresses along the horizontal (left to right) direction, while 90^o plies correspond to stresses aligned perpendicular to the paper plane (coming towards the reader). However, since the laminate is rectangular, the strains in the two directions are the same which implies that the stresses σ_{11} become continuous. Moreover, Figure 4.9 shows the stress gradient through the global x-direction which is discontinuous. However, the point of showing the stress gradient on the local coordinates (S11) is to strongly pinpoint the steep increase of the stresses through thickness, so, the subsequent analyses will only show the local stress gradients.

Note that the stress components are very large, which is unphysical because unlike the simulations, in reality the material during Cycle 1 is in a viscous state and has very limited

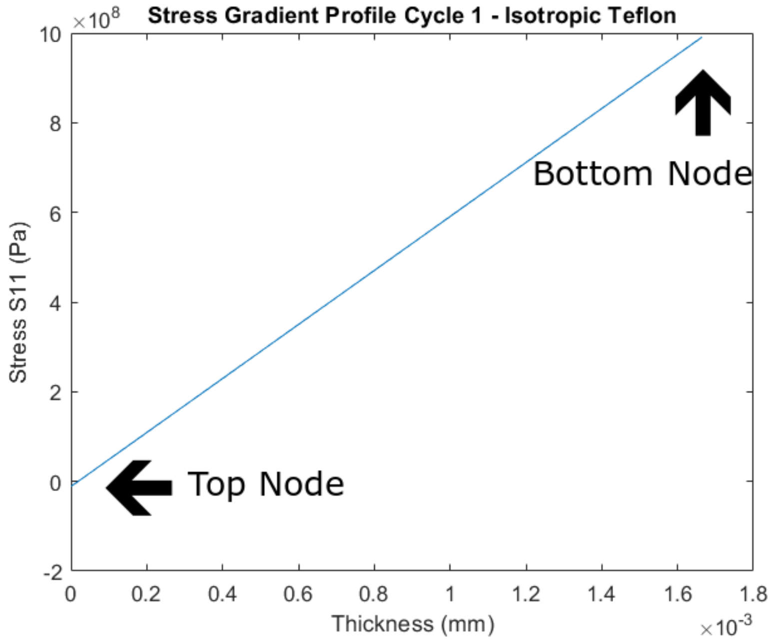


Figure 4.7: Stress S11 through the thickness for a laminate on top of a pure compatibility layer sheet sheet.

ability to hold stresses. This is confirmed by plotting the stress S11 variation along the thickness of the laminate, in Figures 4.7 and 4.8. Obviously, the induced stresses are higher when using pure compatibility layer sheet instead of the composite compatibility layer sheet.

One way to visualize the effects of the stress gradients introduced by the expansion of the compatibility layer sheet, is to create a new simulation where the plates are released and the laminate is allowed to relax in the presence of the residual stresses. For the pure compatibility layer sheet, Figure 4.10 shows how the laminate warps reaching a maximum displacement of **0.54 mm**. For the Composite compatibility layer sheet, Figure 4.11 shows a value of warpage that is one order of magnitude lower, around **0.03 mm**. The laminate warps in the same way as shown in Figure 2.12, i.e. bending outwards (the edges stayed in contact with the compatibility layer sheet).

CYCLE 2

Simulations of Cycle 2 are conducted only considering a pure compatibility layer sheet, given that this exaggerates the warping effect (although other conditions can easily be considered, as the simulations are parameterized). Cycle 2 is influenced by the stresses of Cycle 1, so the loads are applied similarly as in Cycle 1 but not with the same value (Temperature below the T_g and Pressure of several kN). Also, as already explained in section 4.1, the structures are undeformed initially and the simulation starts by predefining temperature and stresses. Figure 4.12 shows the deformation of the laminate, where the horizontal displacement U2 is 0.3 mm. The stress S11 through the thickness is shown in

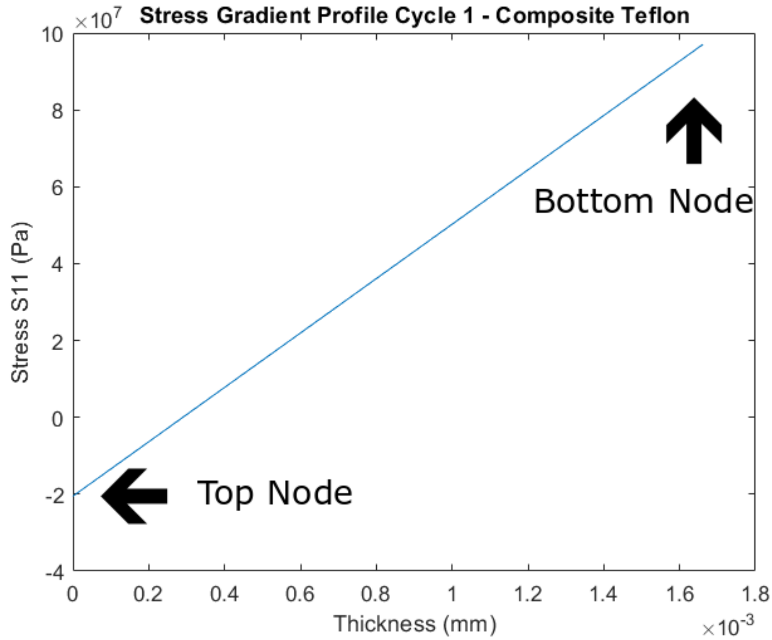


Figure 4.8: Stress S11 through the thickness for a laminate on top of a composite compatibility layer sheet.

Figure 4.13.

The next step is to show how the material warps. First, the stress gradient at the end of the step (when the temperature of the material is 130 ^oC through the thickness) is predicted (Figure 4.14). The second stress gradient that was used is during the solidification temperature range, where part of the laminate is under its $T_g = 150^oC$ and part of it over the glass transition (so half of it is solid and the other half still at rubbery stage). Figures 4.14, 4.15 and 4.16 show the calculated warpage at each Temperature. Using the stress gradient at the end of Cycle 2, the warpage is small and around **0.3 mm**. However, when the stress gradient was captured around it's T_g, the warpage deformation is higher, since the maximum displacement reaches **1 mm** which is very close to the real experimental results. The second case is higher due to the additional effect of the temperature gradient discussed in Chapter 3, which is sufficient to enhance warpage.

4.1.2. SIMULATION CONCLUSIONS

The presence of a compatibility layer sheet (pure or composite) is concluded to induce non-negligible warpage into the laminates after manufacturing. During Cycle 1, the compatibility layer sheet expands due to the increase of temperature, so the laminate deforms with it. However, since the temperature is above the glass transition, the internal stresses should not be large (contrary to linear elastic predictions). Then, in Cycle 2 there is cooling below the glass transition and there the stresses are locked. Since the compatibility layer sheet contracts significantly it drags the laminate with it and leads to the final

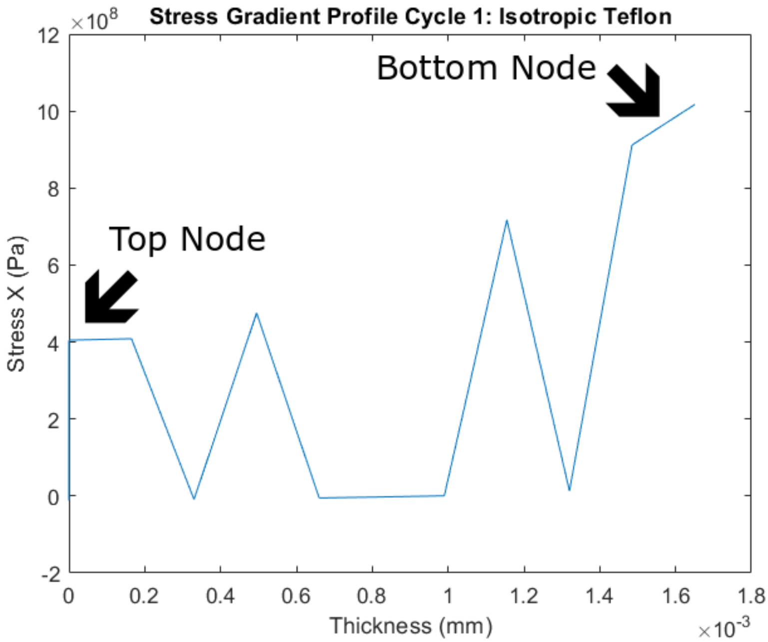


Figure 4.9: Stress x-direction through the thickness for a laminate on top of a pure compatibility layer sheet.

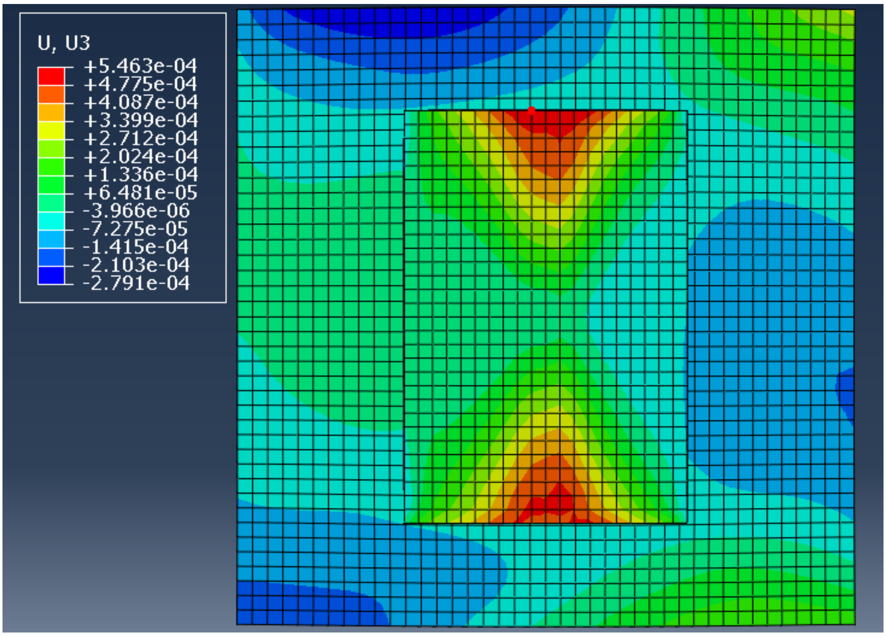


Figure 4.10: Top view warpage from Isotropic compatibility layer sheet Simulation

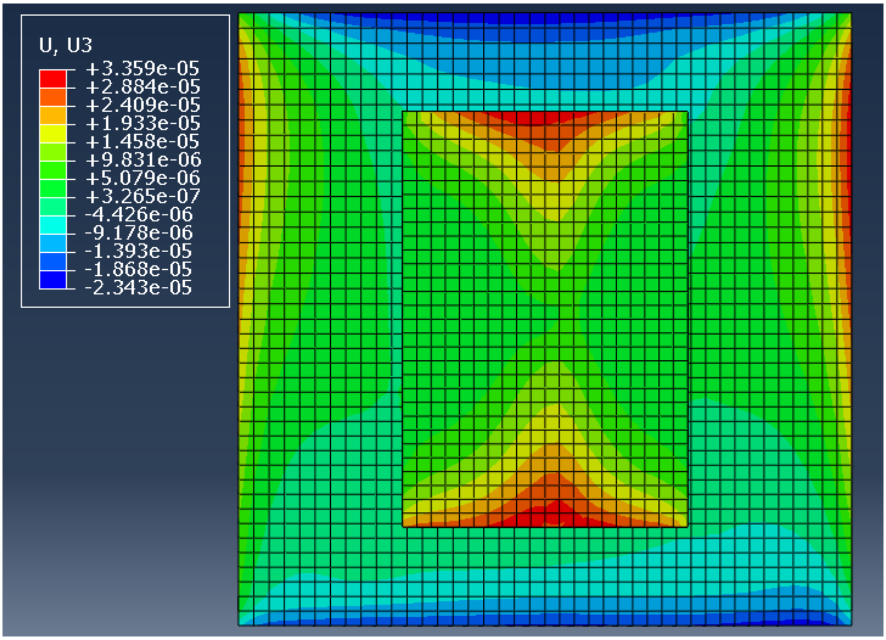


Figure 4.11: Top view warpage from Composite compatibility layer sheet Simulation

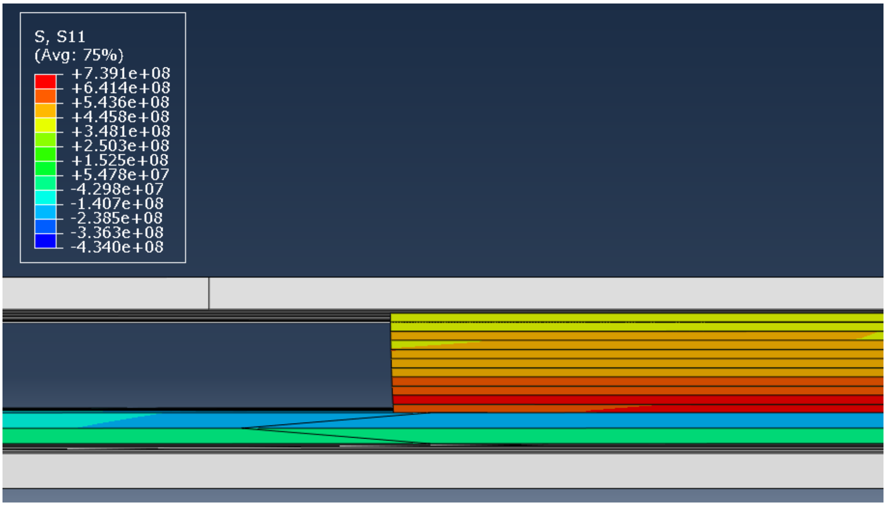


Figure 4.12: Stress S11 through the thickness for laminate on top of pure compatibility layer sheet for Cycle 2

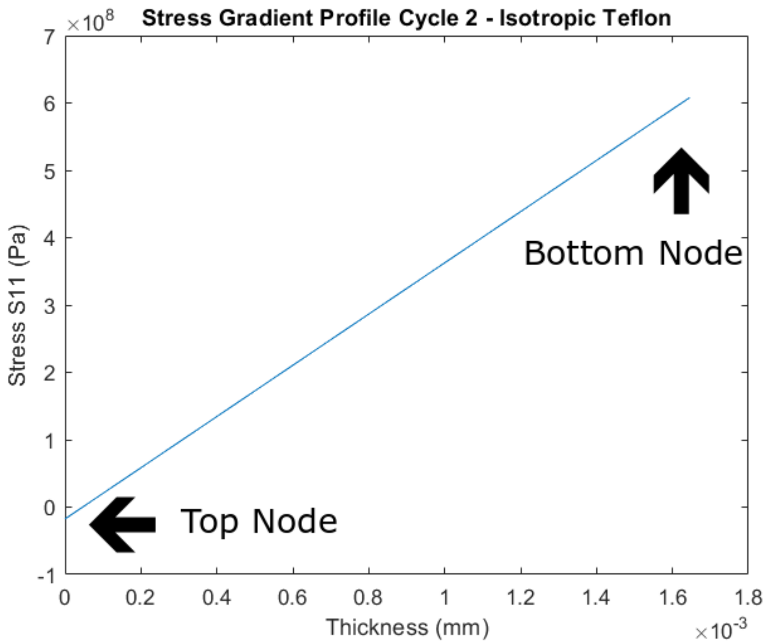


Figure 4.13: Stress S11 through the thickness for laminate on top of pure compatibility layer sheet for Cycle 2

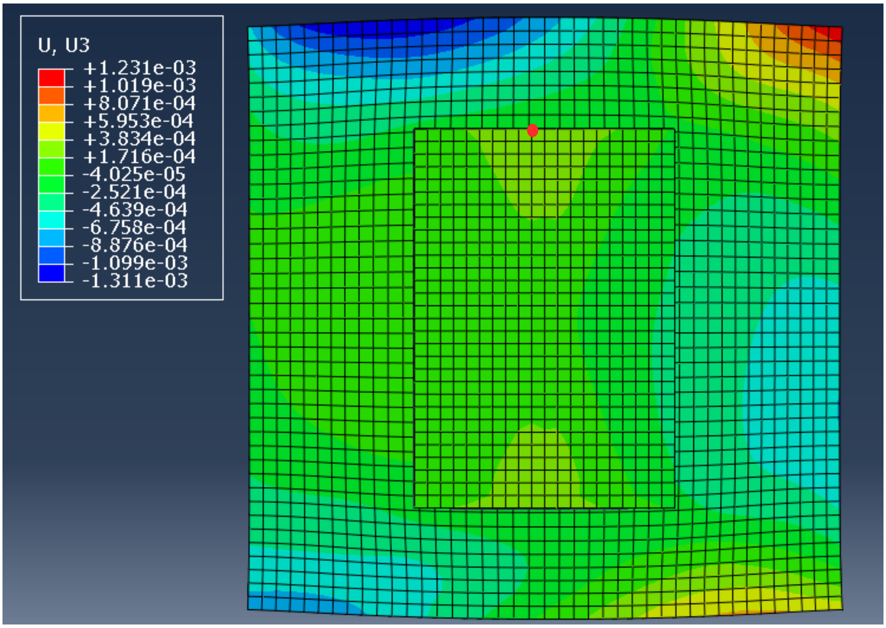


Figure 4.14: Cycle 2 warpage at the end of the step $T = 130^oC$

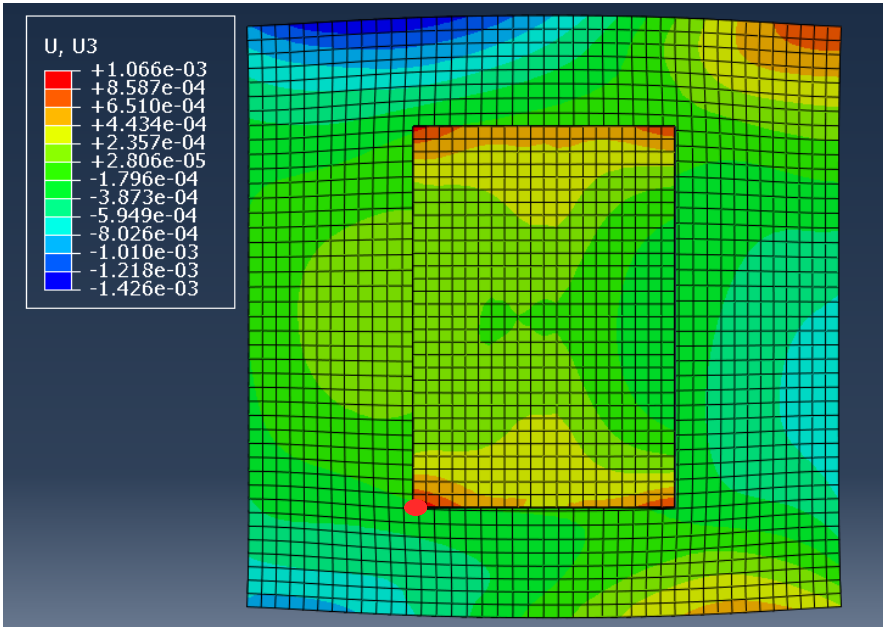


Figure 4.15: Cycle 2 warpage at the solidification $T \sim 150^oC$

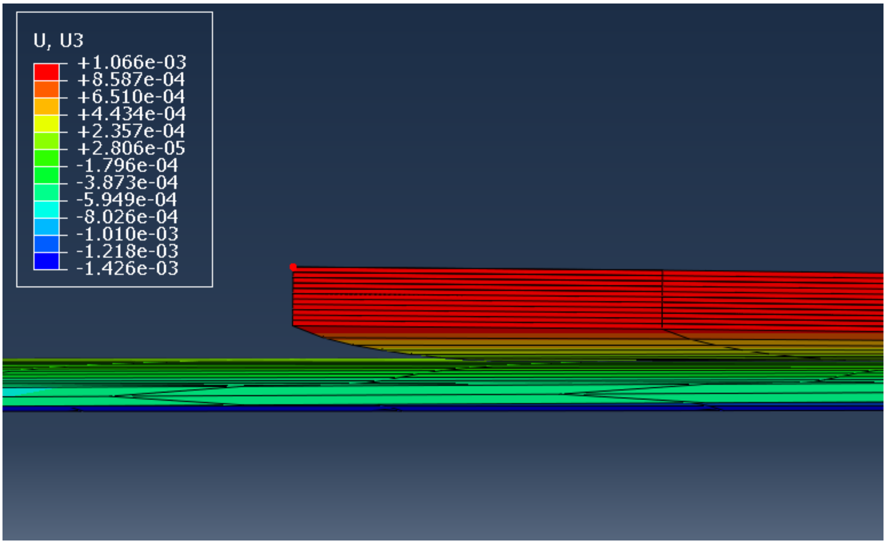


Figure 4.16: Cycle 2 edge warpage at the solidification $T \sim 150^oC$

warpage.

Figures 4.7 and 4.13 show a steep stress gradient for the first cycle, but an almost uniform temperature across the thickness ($\sim 180^oC$). For Cycle 2, the stress gradient is similar but there is also a temperature gradient during the time when stresses are arrested, which imposes extra constraints and deforms the material further.

4.2. TOOL-PART EXPERIMENTS

Subsequently to the simulations, a simple experimental investigation was conducted to assess the impact of the presence of the compatibility layer sheet on the warpage of the laminate. Three pressure configurations were used, ranking from Low to High. For these 3 configurations, 6 total runs were conducted where 3 of them used the compatibility layer sheet and the other 3 without it. Moreover, 3 more tests with different compatibility layer sheet and laminate sequence were done for the sake of comparison. The temperatures for each cycle were the same for each pressure configuration.

1. Low Pressure Configuration: Cycle 1: 100kN, Cycle 2: 100kN, Cycle 3: 50kN
2. Medium Pressure Configuration: Cycle 1: 200kN, Cycle 2: 200kN, Cycle 3: 50kN
3. High Pressure Configuration: Cycle 1: 300kN, Cycle 2: 300kN, Cycle 3: 50kN

Figure 4.17 shows the configuration of the laminates before the press. To make the data collection easier and also to observe if there is any correlation or warpage pattern, the laminates were numbered, where 1 is the top left, 2 is the top right, 3 is the bottom left and 4 is the bottom right. Also, the edges of each laminate where numbered in the same way (1 is the top left edge and so on). The compatibility layer has thickness of $500\mu m$.

After the runs, manual measurements were conducted using a metallic handling tool for every edge of all the laminates. Tables 4.4 and 4.5 show the results from the measurements with the standard 11 layers ($[0_2,90,0,90_2]_s$ lamination sequence with and without the compatibility layer respectively.

To check if the lamination sequence affects the warpage initiation, one extra experiment was conducted with different layup sequence $[0_4,90_2]_s$ (Table 4.6). It was decided for the Pressure configuration to be High due to the assumption that the higher the Pressure, the better the consolidation of the laminates.

The last experiment was with the use of a different compatibility layer sheet, which was thinner with value of $230\mu m$ and also of smaller strength (Table 4.7).

4.2.1. EXPERIMENTAL CONCLUSIONS

The experiments clearly demonstrate that the use of compatibility layer sheet introduces significant deformation in the laminates. The warpage was in the range of **1.3-1.8 mm** on average with the use of compatibility layer sheet and for all the pressure configurations. So probably, the pressure does not affect significantly the warpage initiation, but it is possible that the extracted laminates may have different properties depending of the consolidation pressure cycle that was used. Without compatibility layer sheet, the warpage was minimal since there is no asymmetry introduced by the top and bottom press plates. When thinner compatibility layer sheet was used, it does not seem to affect the warpage differently than

Figure 4.17: Laminate Configuration before the press

Table 4.4: Warpage measurements with the use of regular Airborne's compatibility layer sheet

Low Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.44	1.9	2	1.4
Laminate 2 (mm)	1.35	1.4	1	0.9
Laminate 3 (mm)	1.4	1.4	1.5	1
Laminate 4 (mm)	1	1.2	1.3	1
Medium Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.2	1.3	1.1	1.45
Laminate 2 (mm)	0.7	2.3	2	1.1
Laminate 3 (mm)	1.3	0.6	0.8	0.8
Laminate 4 (mm)	1.45	1.25	1.1	1.8
High Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	1.5	2.1	1.5	1.9
Laminate 2 (mm)	0.6	2.4	1.5	1
Laminate 3 (mm)	1.1	1.7	1.4	1.3
Laminate 4 (mm)	1.5	1.2	0.8	1.5

Table 4.5: Warpage measurements without the use of compatibility layer sheet

Low Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	0.5	0.4	0.5	0.3
Laminate 2 (mm)	0.3	0.7	0.8	0.6
Laminate 3 (mm)	1	0.6	1.1	0.1
Laminate 4 (mm)	0.1	0.3	0.1	0
Medium Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	0.5	0.2	0.55	0.7
Laminate 2 (mm)	0.1	0.65	0.9	0.7
Laminate 3 (mm)	0.4	0.6	0.8	0.3
Laminate 4 (mm)	1	0.6	0.5	0.7
High Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	0.4	0.1	0.3	0.1
Laminate 2 (mm)	0.1	0.2	0.1	0.1
Laminate 3 (mm)	0.2	0.1	0.5	0.1
Laminate 4 (mm)	0.2	0.1	0.1	0.4

Table 4.6: Warpage measurements with the use of the regular compatibility layer sheet and different layup sequence

High Pressure with compatibility layer sheet				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	2.5	1.4	2.1	2.5
Laminate 2 (mm)	0.5	2.4	4.1	3.7
Laminate 3 (mm)	1.8	2	2.45	1.7
Laminate 4 (mm)	2.1	1.6	3.1	3.2
High Pressure without compatibility layer sheet				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	0.5	0.6	1.5	1.4
Laminate 2 (mm)	2	2.5	2.4	1.6
Laminate 3 (mm)	4.6	4.6	5.6	5.4
Laminate 4 (mm)	0.5	0.4	0.3	0.3

Table 4.7: Warpage measurements with the use of thinner compatibility layer sheet

High Pressure				
	Edge 1	Edge 2	Edge 3	Edge 4
Laminate 1 (mm)	2	0.7	0.9	1.6
Laminate 2 (mm)	1.6	1.8	0.6	0.8
Laminate 3 (mm)	1.6	1.4	1.6	0.7
Laminate 4 (mm)	0.9	0.7	1.1	1.7

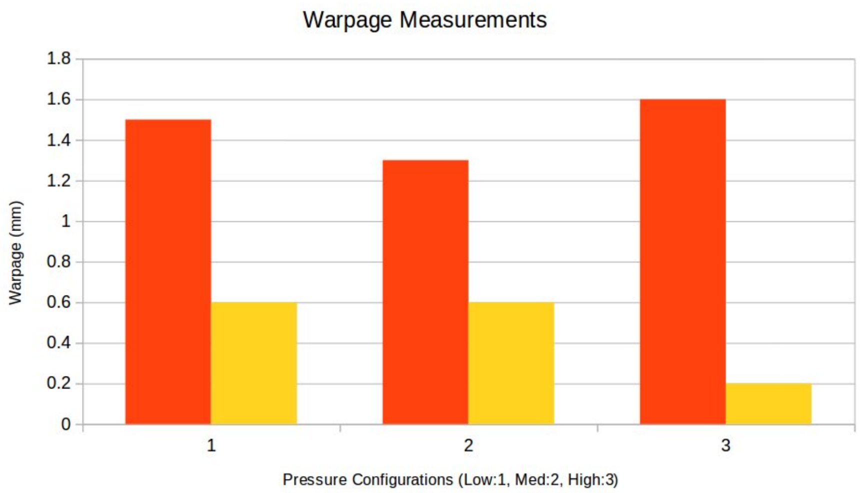


Figure 4.18: Average Warpage Measurements with (red) and without (yellow) compatibility layer sheet sheet.

the regular one. It should be noted that the laminates adhered well to the compatibility layer sheet, i.e. the compatibility layer sheet was not as effective in facilitating extraction as intended. Concerning the experiments with different laminates, it was evident that due to the block effect (high number of consequent layers with the same direction) the interlaminar stresses would have been high enough to bend the laminate in higher values concerning the regular laminate (see Figure 2.5 and section 2.2). Figure 4.18 illustrates an average of the warpage measurements for the 3 pressure configurations for the sake of visualization (for the meaning of low, medium and high pressures, you can refer to the start of this section).

As an overall conclusion, simulations showed that the compatibility layer sheet affects the warpage initiation, where the experiments validate this result. However, the input properties and constitutive models that were considered in this Chapter are not truly representative of reality. In addition, manufacturing imperfections such as fiber misalignments were not taken into account.

5

VISCOELASTICITY

POLYMERS are viscoelastic, i.e. their elastic behavior is time-dependent. This has implications for residual stresses because the material can relax with time. Therefore, this chapter follows a simple modeling strategy to predict viscoelasticity of composite laminates using commercial finite element software.

5.1. IMPLEMENTATION IN FINITE ELEMENT ANALYSIS

The commercial finite element software used herein, Abaqus, does not have orthotropic viscoelastic material models implemented. However, a simple strategy [44] is to merge two separate material models to as shown in Figure 5.1. In practice, this is achieved as follows:

1. Create two materials and assign the corresponding constitutive laws having exactly the same part dimensions: one with an isotropic viscoelastic law, and one with an elastic orthotropic law.

2. Mesh the parts with the same element configuration.

3. Merge the 2 parts as one, such that they occupy the same location (perfect superposition). Note that this merge operation should be a "node" merge, not a "geometry" merge.

This simple strategy is equivalent to having a parallel model between the elastic response (that is orthotropic) and an isotropic viscoelastic response. Due to the parallel addition of an extra elastic element (spring), the stifness of the material will be stiffer comparing to the one with only one elastic part (see equation 2.21 and Figure 2.24).

After merging the two models, the Teflon sheet and the two plates are added into the assembly with the proper assignment of interactions and loads, as discussed previously in Chapter 4. There are different viscoelastic models that can be considered. Typically, their properties can be assigned using three types of time dependencies:

1. Relaxation data (time dependent shear modulus vs time)

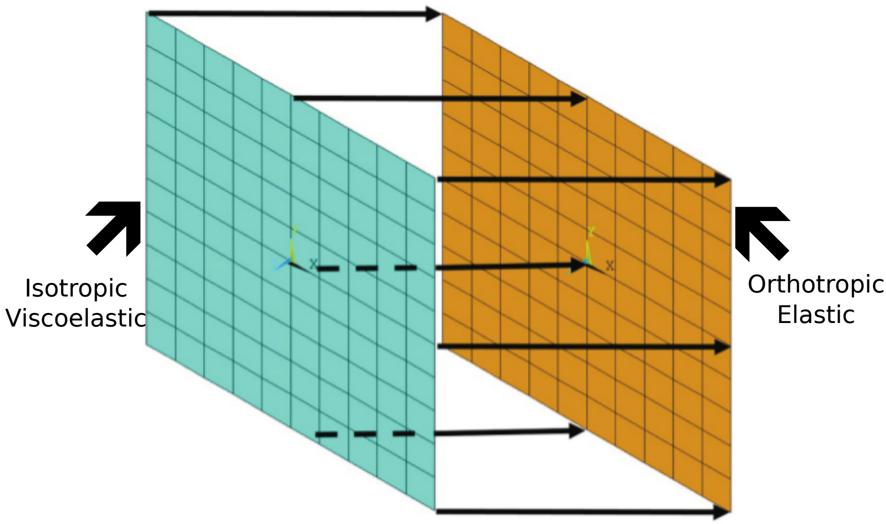


Figure 5.1: Merging of coincident nodes for elastic (orange) and viscoelastic (blue) finite element models [44]

2. Creep data (time dependent creep compliance vs time)
3. Prony series

The Prony series can be defined directly or calculated from the relaxation or creep data. Moreover, they can be implemented as shear relaxation tensor (G) or/and bulk relaxation function (K) with the equations:

$$\hat{G}_t = \hat{G}_o[E^G_\infty + \sum_i^n E^G_i exp(-\frac{t}{\tau^G_i})] \tag{5.1}$$

$$K_t = K_o[E^K_\infty + \sum_i^n E^K_i exp(-\frac{t}{\tau^K_i})] \tag{5.2}$$

where the same notation of Section 2.3.3 is used.

For this thesis, the material properties are selected from Jazouli et al. [35], who report on creep data for polycarbonate. The creep compliance must be normalized according to Figure 5.2. As highlighted in the figure, the data associated to creep at higher applied stress is considered herein, as the manufacturing process occurs at high pressure. Figures 5.3 and 5.4 show the corresponding viscoelastic data implemented in the finite element code. After the generation of the Figures 5.3 and 5.4, Abaqus also provide a way to evaluate the viscoelastic model. What it does is to calculate the Prony series using the equations 5.1 and 5.2 with all the data properties that the user provided to the software. The extracted Prony series is then defined by the coefficients listed in table 5.1. The user can

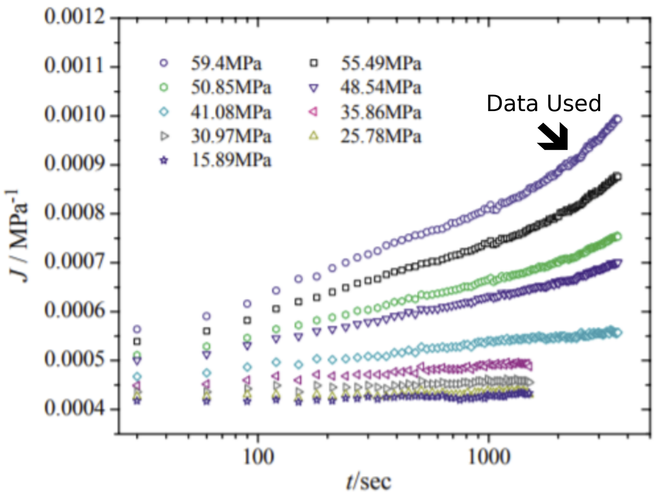


Figure 5.2: Creep compliance curves of PC for various stress levels [35]

Table 5.1: Prony series from Abaqus evaluation from data [35]

Linear	Isotropic	Prony Series	
I	G(I)	K(I)	TAU(I)
1	-2.23E-02	0.0	5.19E-03
2	0.25	0.0	156.3

also define the Prony series immediately, if they are known through numerical analysis and experimental results.

5.2. COMPUTATIONAL PREDICTIONS WITH VISCOELASTICITY

CYCLE 1

The simulations are similar to Chapter 4, but now using the merged finite element meshes as described in the previous section. Recall that during Cycle 1 the temperature is 210 ^oC and the force is 400 kN. Figures 5.5 and 5.6 show the pre- and post-deformation of the material after the applied loads (only the laminate for better illustration). Figure 5.7 illustrates the overall stresses and also the residual stresses along the direction of the fibers.

Concerning the deformed laminate, the displacement on the x-axis is small, about 0.4 mm of horizontal dragging. However, the deformed shape of the laminate is different to the one observed for the purely elastic case seen in Chapter 4. The laminate's top layers are more deformed than the bottom ones – the opposite of what was observed in the previous Chapter. The stress gradient is also smaller, as expected due to relaxation (Figure 5.8).

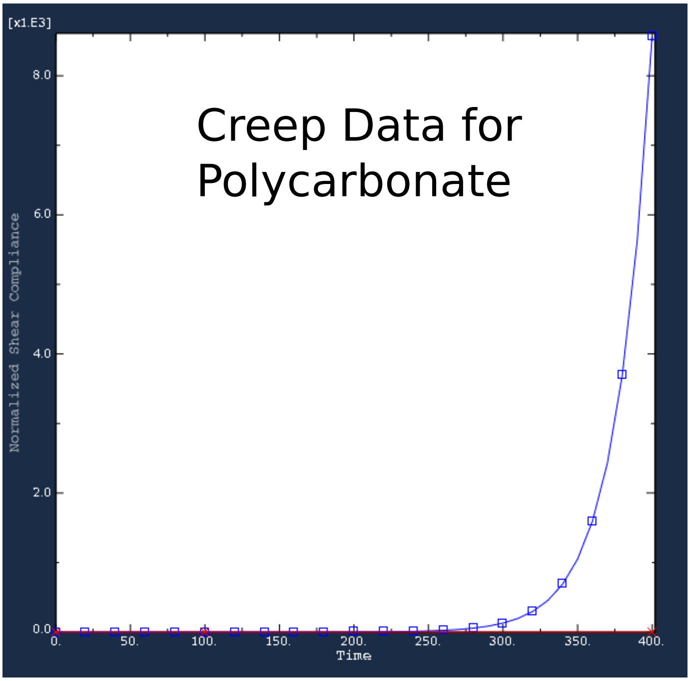


Figure 5.3: Creep compliance vs Time from ABAQUS CAE evaluation

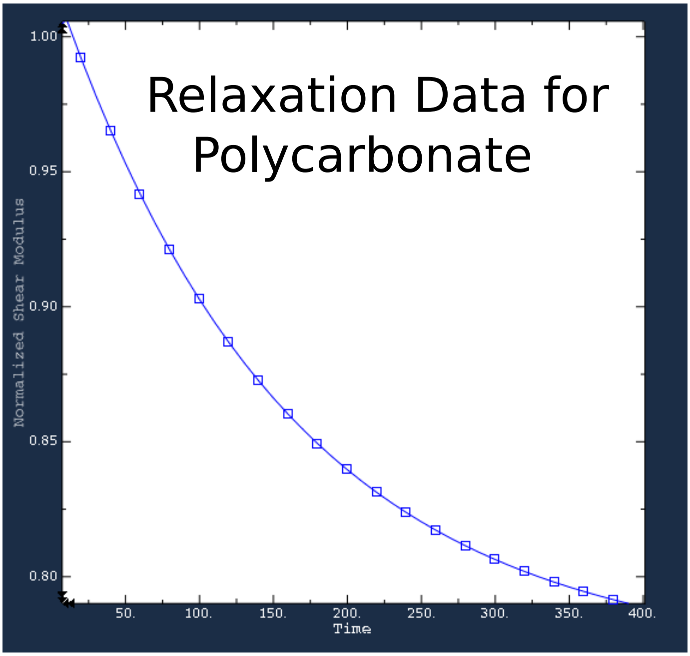


Figure 5.4: Time Dependent Shear Modulus vs Time from ABAQUS CAE evaluation

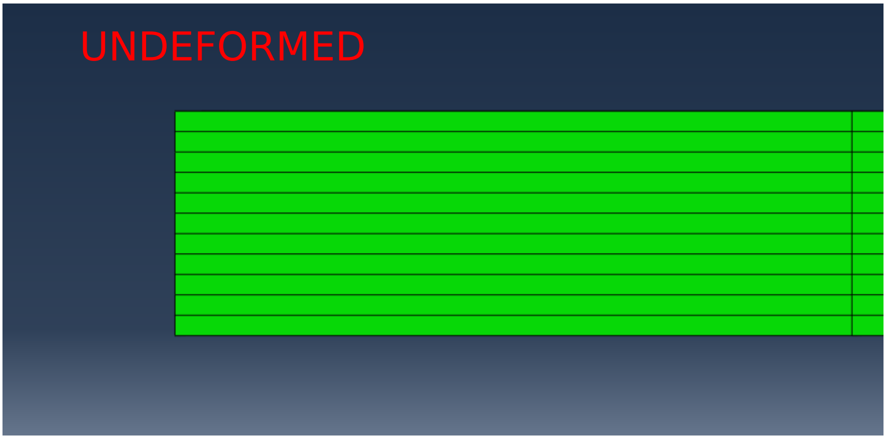


Figure 5.5: Undeformed configuration of the viscoelastic laminate

Figure 5.6: Deformed configuration of the viscoelastic laminate for Cycle 1

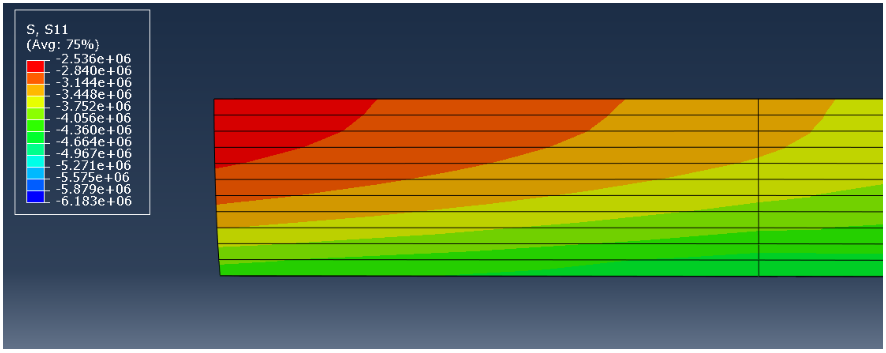


Figure 5.7: Stress profile along the fibers direction for Cycle 1

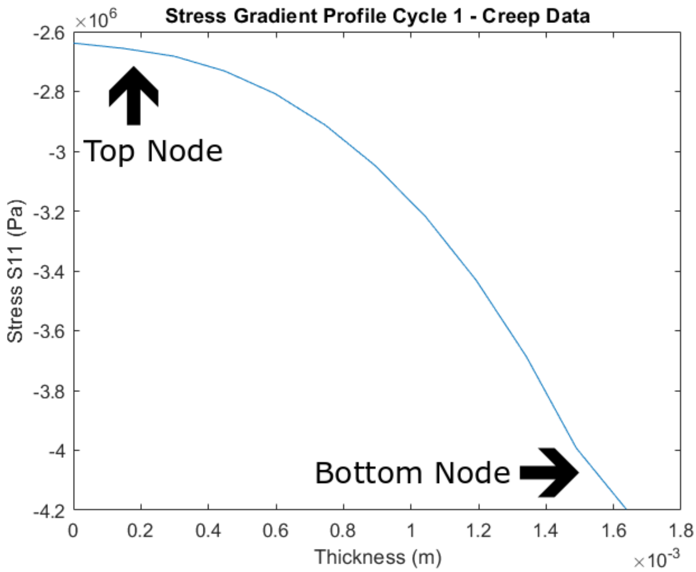


Figure 5.8: Stress profile along the fibers direction for Cycle 1 graph

After the stress gradient was obtained at the end of the cycle, it was used as a predefined field for the warpage simulation. Figure 5.9 illustrates how the material warped. As can be seen, the warpage is very small compared to the purely elastic material. The value is approximately 0.06 mm. The reason of the limited warpage is associated to the low residual stresses due to relaxation.

CYCLE 2

In Cycle 2 recall that the stress and temperature profiles are predefined from the last increment of Cycle 1, and then the loads of Cycle 2 are applied. Figure 5.10 shows the deformed configuration and Figure 5.11 illustrates the stress profile along the fibers direction.

The horizontal displacement along the x-axis is very small, approximately 0.056 mm. Also, the stress gradient in this case is not what was expected. As can be seen from Figure 5.11, there are fluctuations of the stresses between the layers, where the middle layers have negative residual stresses comparing with the top layers (tensile forces are applied on top layers and compressive stresses of the same magnitude on the middle layers). The bottom layers have smaller compressive stresses than the middle layers and in general, the residual stresses are small. Figure 5.12 shows a top view of the laminate and the corresponding vertical displacement. The warpage of the laminate is higher than the one obtained from Cycle 1, achieving a value around 0.1 mm. However, this value is small compared to experimental results.

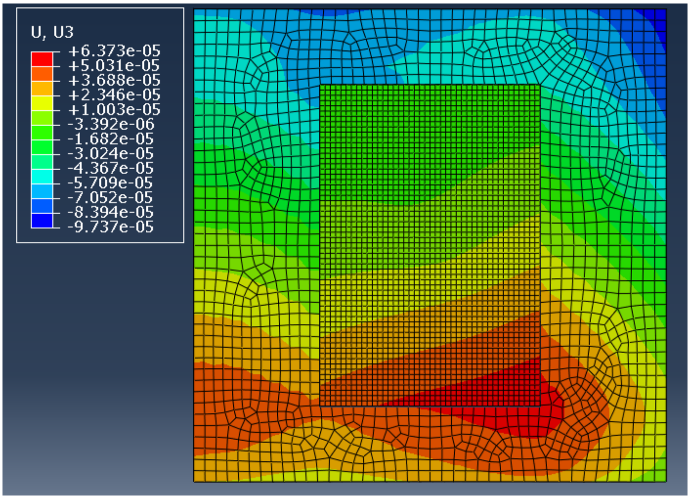


Figure 5.9: Warpage from stress gradient from Cycle 1

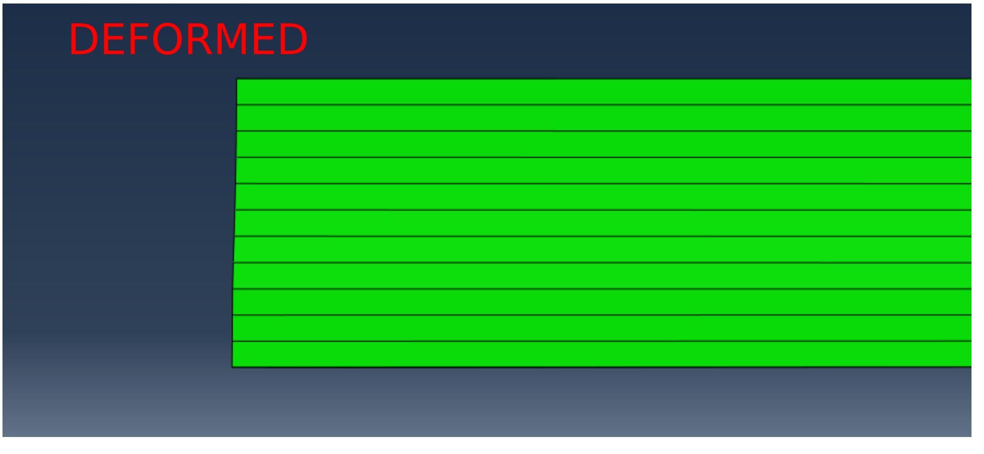


Figure 5.10: Deformed configuration of the viscoelastic laminate for Cycle 2

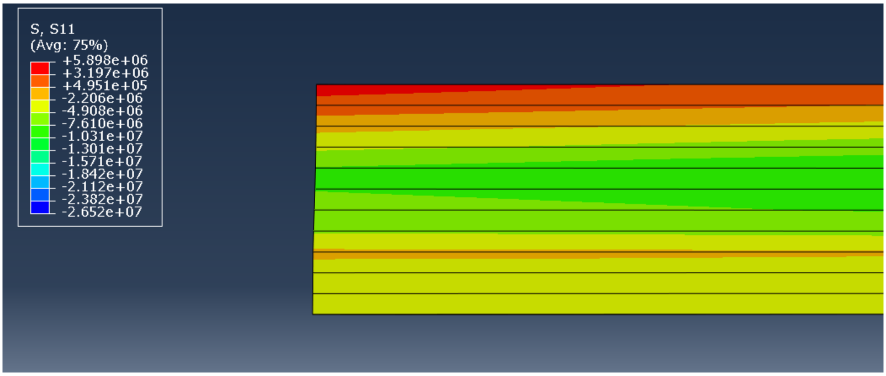


Figure 5.11: Stress gradient on the x-direction for Cycle 2

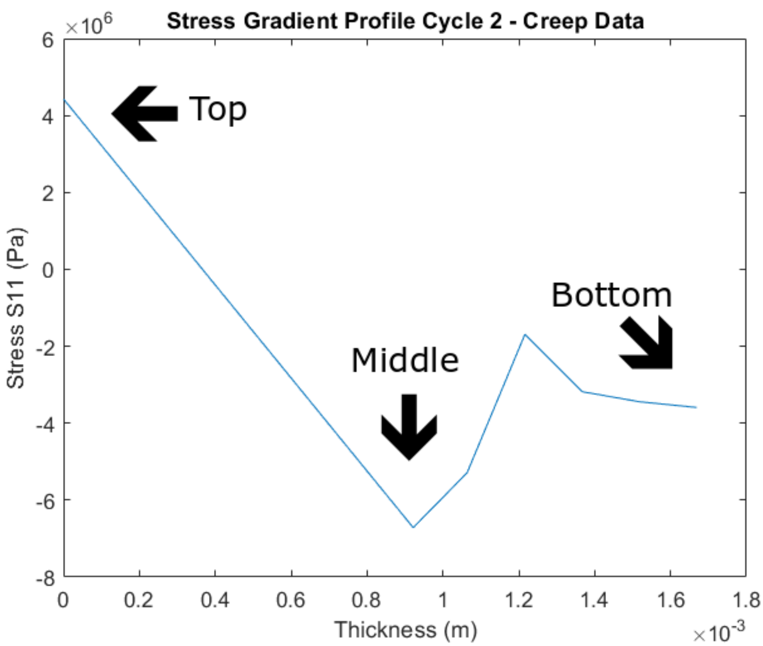


Figure 5.12: Stress gradient on the x-direction for Cycle 2 graph

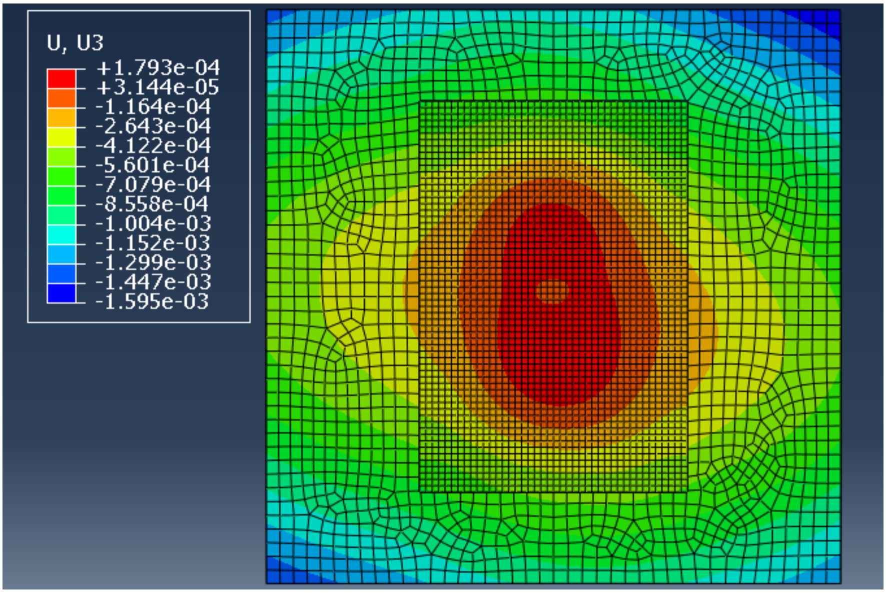


Figure 5.13: Warpage from stress gradient from Cycle 2

5.3. VISCOELASTICITY CONCLUSION

Due to time constraints in performing this work, the viscoelasticity simulations can only be considered preliminary. However, the modeling strategy is implemented and future work can be developed to improve the quality of the predictions by including adequate material properties and experimentally validating them. Nevertheless, the effect of relaxation mechanisms is demonstrated where the residual stresses become lower when compared to an elastic material, leading to less warpage.

6

Discussion

This work aims at understanding warpage of thermoformed composite laminates. Although this investigation focused on a particular manufacturing line, the codes developed herein are parametric and applicable to other thermoforming processes with different conditions.

Concerning the particular thermoforming process under analysis, it was observed that the laminates at the end of the process were undesirably warped. The literature on the subject offers several origins for the residual stresses that explain this behavior: from micro-scale to macro-scale phenomena. This thesis concentrates on simulating the main macro-scale phenomena.

In Chapter 3, finite element analyses have demonstrated that thermal gradients are not significant, so they cannot explain the formation of residual stresses because heat transfers sufficiently quickly in each press cycle of the process. The simulations can still be improved significantly:

1. Input material properties have not been experimentally determined. This is a major limitation of this work because the properties found in the literature cannot be representative of the particular material under analysis, which severely affects the quality of the simulations;

2. Viscous/rubbery phases of the composite laminate have not been simulated;

3. The temperature profile is obtained without considering mechanical loads (pressure)

4. Few simulations have been conducted, only providing qualitative information, instead of determining the dependency of warpage on the input material properties and process conditions.

Notwithstanding the above stated, the computational analyses of Chapter 3 enabled an estimation of the influence of thermal conductance and contact conditions on the thermal profile, and excluded thermal effects as being the only ones responsible for residual stresses.

In Chapter 4, the interaction with the tool is shown to significantly affect warpage by introducing a stress gradient due to the expansion of a Teflon sheet at the bottom of the composite laminate. If that sheet is itself a composite with Teflon matrix, then warpage decreases significantly, although still being observed. Nevertheless, appreciable limitations in the developed models should be taken into consideration:

1. Again, reliable input material properties have not been used;
2. Elastic material laws are used, so the rubbery and viscous phases are not properly modeled;
3. Cycle 2 is modeled from an undeformed configuration and by importing the thermal and stress profiles at the end of Cycle 1 as predefined fields;
4. Similar issue when simulating warpage after Cycle 2;
5. No microstructural defects have been considered.

Finally, Chapter 5 introduced a practical strategy to predict viscoelastic behavior in composite laminates where two finite element models are merged into one by superimposing orthotropic elasticity with isotropic viscoelasticity. These simulations show that the relaxation of the polymer matrix leads to lower residual stresses and less warpage, as compared to a purely elastic material (Chapter 4). However, simulating orthotropic viscoelasticity with this strategy involves important simplifications:

1. The elastic and viscoelastic model are in parallel. In micromechanics, parallel models provide a lower bound for composite properties (and series models an upper bound). Therefore, this modeling strategy should be validated appropriately by comparing against other modeling strategies, and especially against experimental results;
2. Once again, experimental input should be carefully determined to enhance the predictive capabilities of the model;

7 CONCLUSION

This thesis provides basic knowledge about thermoforming of thermoplastics and the role of residual stresses in warping composite laminates. This is intended to be a modest first step towards automating the simulation process such that more in depth investigations can be conducted, including sensitivity analysis, machine learning, and optimization. Sensitivity analyses can help understand what are the input material properties and process parameters that affect more significantly the properties of interest (e.g. warpage), while machine learning can map the input-output relationship. Yet, these techniques can only be used after automating the analyses process, which is the main focus of this thesis.

The macro-scale models created showed that the presence of a Teflon sheet in between the bottom press and the composite laminate leads to non-negligible warpage of the material. This warpage was identified to arise from a mechanical interaction between the Teflon and the laminate, due to a higher thermal expansion of the first when compared to the latter. This stretches the bottom part of the laminate while the temperatures are above the glass transition temperature, i.e. when the laminate is in a viscous/rubbery state and offers little resistance to deformation. Yet, after cooling, there is a stress build-up and these residual stresses are arrested, causing the subsequent warpage when the laminate is released from the press. The finite element simulations also excluded the possibility of the residual stresses arising from a thermal gradient through the thickness of the laminate. In summary, the computational predictions showed that warpage can be predicted, and assisted in isolating the causes behind this warpage.

Due to the limited duration of a masters research project, a vast parametric study was not possible. However, the parametric finite element models developed herein are shared in the Appendix of this work for assisting future investigations. A future investigation should start by carefully characterizing the material properties and developing adequate constitutive models that include viscoelasticity and phase transformations. Predicting the viscous/rubbery state of the material may involve different modeling techniques, such as Arbitrary Lagrangian-Eulerian finite element methods, meshfree methods, or even Computational Fluid Dynamics.

References

[1] G. K. Jeyakodi, *Finite Element Simulation of the In - Situ AFP process for Thermoplastic Composites using Abaqus*, Master's thesis, TU Delft (2016).

[2] I. Zewi, I. Daniel, and J. Gotro, *Residual stresses and warpage in woven-glass/epoxy laminates,* Experimental mechanics **27**, 44 (1987).

[3] G. Fernlund, A. Poursartip, G. Twigg, and C. Albert, *Residual stress, spring-in and warpage in autoclaved composite parts,* TECHNICAL PAPERS-SOCIETY OF MANUFACTURING ENGINEERS-ALL SERIES- (2003).

[4] I. M. Daniel, O. Ishai, I. M. Daniel, and I. Daniel, *Engineering mechanics of composite materials*, Vol. 3 (Oxford university press New York, 1994).

[5] G. Twigg, A. Poursartip, and G. Fernlund, *Tool–part interaction in composites processing. part i: experimental investigation and analytical model,* Composites Part A: Applied Science and Manufacturing **35**, 121 (2004).

[6] T. E. of Encyclopaedia Britannica, *Van der waals forces chemistry and physics,* (2019).

[7] M. Mayer, *What is a thermoplastic polymer?* (2018).

[8] Sabic, *Sabic® pc resin,* (2019).

[9] E. Britannica, *Amorphous solid,* (2019).

[10] A. Van der Vegt, *From polymers to plastics* (VSSD Delft, The Netherlands, 2006).

[11] P. S. L. Center, *The glass transition,* (2019).

[12] H. G. Karimiani, *Analysis of Residual Stresses in Thermoplastic Composites Manufactured by Automated Fiber Placement*, Master's thesis, Concordia University (2015).

[13] N. Rudolph, I. Kühnert, E. Schmachtenberg, and G. Ehrenstein, *Pressure solidification of amorphous thermoplastics,* Polymer Engineering & Science **49**, 154 (2009).

[14] A. Jungmeier, W. Wildner, D. Drummer, and I. Kühnert, *Compression-induced solidification: A novel processing technique for precise thermoplastic optical components with negligible internal stresses,* ISRN Optics **2012** (2012).

[15] formlabs, *Guide to manufacturing processes for plastics,* (2019).

[16] S. Faris, *Four primary types of manufacturing processes,* (2018).

[17] R. M. Stack and F. Lai, *Development in thermoforming thermoplastic composites,* (2018).

[18] F. Saraiva, *Development of press forming techniques for thermoplastic composites,* Master's thesis, TU Delft (2017).

[19] M. Shokrieh and A. G. Mohammadi, *The importance of measuring residual stresses in composite materials,* in *Residual Stresses in Composite Materials* (Elsevier, 2014) pp. 3–14.

[20] N. Zobeiry and A. Poursartip, *The origins of residual stress and its evaluation in composite materials,* in *Structural integrity and durability of advanced composites* (Elsevier, 2015) pp. 43–72.

[21] P. P. Parlevliet, H. E. Bersee, and A. Beukers, *Residual stresses in thermoplastic composites—a study of the literature—part i: Formation of residual stresses,* Composites Part A: Applied Science and Manufacturing **37**, 1847 (2006).

[22] D. Plastics, *How engineering plastics expand with temperature,* (2013).

[23] N. K. MacVarish, *The difference between amorphous and semi crystalline polymers,* (2017).

[24] B. Lotz, *Phase Transitions and Structure Of Crystalline Polymers*, Tech. Rep. (Institut Charles Sadron).

[25] Y. Guo, Y. Wang, G. Guo, and Y. Xie, *Finite element analysis of the thermal residual stress distribution in the interphase of unidirectional fiber-reinforced resin matrix composites,* Composite Interfaces **25**, 823 (2018).

[26] F. Teixeira-Dias and L. Menezes, *Numerical aspects of finite element simulations of residual stresses in metal matrix composites,* International Journal for Numerical Methods in Engineering **50**, 629 (2001).

[27] C. Ridgard, *Accuracy and distortion of composite parts and tools: causes and solutions,* TECHNICAL PAPERS-SOCIETY OF MANUFACTURING ENGINEERS-ALL SERIES- (1993).

[28] Autodesk, *The causes of warpage,* (2019).

[29] J. Vlachopoulos and D. Strutt, *Basic heat transfer and some applications in polymer processing,* Plastics Technician's Toolbox **2**, 21 (2002).

[30] A. Ferrous and N.-F. M. Stockist, *Copper - specifications, properties, classifications and classes,* (2005).

[31] H. Ghayoor, F. Shadmehri, and S. Van Hoa, *Development of experimental technique for measuring strain and deformation in manufacturing of thermoplastic composites using automated fiber placement (afp),* .

[32] Bergsma, *Design of lightweight materials,* (2018).

[33] G. Jeronimidis and A. Parkyn, *Residual stresses in carbon fibre-thermoplastic matrix laminates,* Journal of Composite Materials **22**, 401 (1988).

[34] P. Kelly, *Solid mechanics lecture notes,* Reological Models (2013).

[35] S. Jazouli, W. Luo, F. Bremand, and T. Vu-Khanh, *Application of time-stress equivalence to nonlinear creep of polycarbonate,* Polymer testing **24**, 463 (2005).

[36] Sorbothane, *The difference between elastic materials and viscoelastic materials,* (2019).

[37] A. Y. Malkin and A. I. Isayev, *Rheology: concepts, methods, and applications* (Elsevier, 2017).

[38] Sorbothane, *The difference between elastic materials and viscoelastic materials,* .

[39] Zobeiry and Nima, *iscoelastic constitutive models for evaluation of residual stresses in thermoset composites during cure*, Ph.D. thesis, Diss. University of British Columbia (2006).

[40] D. Roylance, *ENGINEERING VISCOELASTICITY*, Tech. Rep. (Department of Materials Science and Engineering Massachusetts Institute of Technology, 2001).

[41] D. S. C. Shit, *Approach for ascertaining service temperature of plastics based on heat deflection temperature,* (2019).

[42] Kelly, *Rheological models,* in *Solid Mechanics Part I*.

[43] A. Ding, S. Li, J. Sun, J. Wang, and L. Zu, *A thermo-viscoelastic model of process-induced residual stresses in composite structures with considering thermal dependence,* Composite Structures **136**, 34 (2016).

[44] V. G. Martynenko, *An original technique for modeling of anisotropic viscoelasticity of orthotropic materials in finite element codes applied to the mechanics of plates and shells,* Mechanics and Mechanical Engineering **21**, 389 (2017).

[45] T. Sakai and S. Somiya, *Estimating creep deformation of glass-fiber-reinforced polycarbonate,* Mechanics of Time-Dependent Materials **10**, 185 (2006).

[46] P. P. Database, *Temperature dependence of polymer viscosity,* (2015).

[47] T. Instruments, *Thermal Analysis Application Brief*, Tech. Rep. (Thermal Analysis and Rheology, 2019).

[48] G. Twigg, *TOOL-PART INTERACTION IN COMPOSITES PROCESSING*, Master's thesis, BA.Sc. (Materials and Metallurgical Engineering), Queen's University (1997).

[49] A. Aleksandrov, Archer and Azzopardi, *Thermal contact resistance,* (2011).

[50] K. N. Babu, *Thermal Contact Resistance: Experiments and Simulation*, Master's thesis, Department of Applied Mechanics CHALMERS UNIVERSITY OF TECHNOLOGY (2015).

[51] C. Demerchant, *Thermal conductivity of carbon fiber, and other carbon based materials,* (2019).

[52] R. Progelhof, J. Throne, and R. Ruetsch, *Methods for predicting the thermal conductivity of composite systems: a review,* Polymer Engineering & Science **16**, 615 (1976).

[53] E. System, *Parameters of polycarbonate,* .

[54] M. Velea and S. Lache, *Thermal expansion of composite laminates,* Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I **8**, 25 (2015).

[55] C. Demerchant, *Youngs modulus is a measure of stiffness,* (2019).

[56] D. Edie, *The effect of processing on the structure and properties of carbon fibers,* Carbon **36**, 345 (1998).

[57] P. C. LTD, *Mechanical properties of carbon fibre composite materials, fibre / epoxy resin (120°c cure),* (2009).

[58] AZoM, *Polycarbonate (pc) - conductive polycarbonate - properties - supplier data by goodfellow,* .

[59] M. MInus and S. Kumar, *The processing, properties, and structure of carbon fibers,* Jom **57**, 52 (2005).

[60] N. Morton, *Design and Manufacture of an Advanced Composite,* Master's thesis, University of Glasgow, Meng in Mechanical Engineering with Aeronautics (2010-2011).

[61] C. C. Chamis, *Mechanics of composite materials: past, present, and future,* Journal of Composites, Technology and Research **11**, 3 (1989).

[62] G. B. Media, *The effects of temperature,* (2011).

[63] S. Krop, *Constitutive modeling of rate and temperature dependent strain hardening in polycarbonate,* Master's thesis, Eindhoven University of Technology (2011).

[64] NETZSCH, *Polycarbonate — thermal conductivity,* (2019).

[65] DuPont, *Teflon ptfe, properties handbook,* .

[66] S. M. Wahid and C. Madhusudana, *Gap conductance in contact heat transfer,* International Journal of Heat and Mass Transfer **43**, 4483 (2000).

[67] S. Song, M. Yovanovich, and F. Goodman, *Thermal gap conductance of conforming surfaces in contact,* Journal of Heat Transfer **115**, 533 (1993).

[68] E. Marotta and L. Fletcher, *Thermal contact conductance of selected polymeric materials,* Journal of Thermophysics and Heat Transfer **10**, 334 (1996).

[69] E. ToolBox, *Friction and friction coefficients,* (2004).

[70] R. Weyler, J. Oliver, T. Sain, and J. Cante, *On the contact domain method: A comparison of penalty and lagrange multiplier implementations,* Computer methods in applied mechanics and engineering **205**, 68 (2012).

[71] A. 6.14, *Contact controls,* (2019).

[72] J. HERMAN, *Running sobol sensitivity analysis using salib,* (2017).

[73] N. Hsieh, *Sensitivity analysis by python salib,* (2017).

A

Appendix A

In the Appendix A, some basic Python scripts for simulating the Heat Transfer simulation from Chapter 3 will be provided. The basic structural code will be given, which will be the same for simulations of Chapters 4 and 5. Off course, for each Chapter, extra structural coding will be needed (such as the simulation of the plates as rigid or deformable). The interactions, boundary conditions and mesh of the elements will be given in each Chapter individually. Moreover, for display methods, the viewer can't copy and paste the codes, as the need of splitting them in order to be fully displayed in the document. But, the code is fully automated if correctly displayed in Python programming. There are two ways to run the scripts. One is to use the "Run Script" command in Abaqus, where you can choose the desirable file and the second one is to run script by script in Abaqus GUI (this is better if the user wants to add different scripts and also using the tools from the interface in conjuction).

For better understanding, comments will be given in the script (the comments are illustrated in Python with "hash-tag"). The code is one big script, but each smaller script can be used to simulate separate needs.

A.1. Heat Transfer Simulation

```
## Heat Transfer

from abaqus import *
from abaqusConstants import *
from caeModules import *
from driverUtils import executeOnCaeStartup
executeOnCaeStartup()
import os
#os.chdir(r"C:\temp")
os.chdir(r"E:\HDD BACKUP 03.09.19\temp")

# Import important modulus
```

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,
 recoverGeometry=COORDINATE)

Grid = 0.04
## Composite Parameters
#Diamensions and laminate sequence,
sheet_Size = 1 #If i want to change parameters
Length = 0.3
Width = 0.2

Width_=Width/2.0
Length_=Length/2.0

thick_ply=0.00015
#Airborne 11 plies composite
OrientationPlyes= [ 0, 0, 90, 0, 90, 90, 90, 0, 90, 0, 0 ]
#Thickness of the cohesive zone (change it concerning the units)
delta=0.00001

Nplies=(len(OrientationPlyes))-1

## Create the laminate with adding the thickness of the plies

ThicknessPlyes=[]

for i in range(len(OrientationPlyes)):
        ThicknessPlyes.append(thick_ply)

# Calculate the Thickness of the composite

Thickness_=0.0
ThicknessPlyesCumulative=[]
```

```
for i in range(len(ThicknessPlyes)):
        Thickness_=Thickness_+ThicknessPlyes[i]
        ThicknessPlyesCumulative.append(Thickness_)

# Create the cohesive layer (pure matrix material)
ThicknessPlyesCohesive=[]

for i in range(len(ThicknessPlyesCumulative)):
        ThicknessPlyesCohesive.append(ThicknessPlyesCumulative[i]-delta/2.0)
        ThicknessPlyesCohesive.append(ThicknessPlyesCumulative[i]+delta/2.0)

ThicknessPlyesCohesive.remove(Thickness_+delta/2.0)

# Teflon Parameters
Sides = 0.45
Sides_=Sides/2.0
thickness = 0.0005

# Heat Transfer Steps
time_Heat = 40
time_Couling = 20

# Interaction
Thermal_Conductance_Comp_Teflon=500
Thermal_Conductance_Plates_Polymers=1500
ContactValue = 0.1

# Convection: Not needed now
Convection = 7
sink_temperature = 20

##################################################################

## Composite
# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize= sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(
 Width_,Length_),
    point2=( -Width_,-Length_))
mdb.models['Model-1'].Part(dimensionality=THREE_D,
 name='Airborne-Composite', type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Airborne-Composite'].BaseSolidExtrude(
depth=Thickness_, sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
```

```

# partition for the edge with cohesive elements

for i in range(len(ThicknessPlyesCohesive)):
    # partition of the edge
    p = mdb.models['Model-1'].parts['Airborne-Composite']
    e1, v1, d1 = p.edges, p.vertices, p.datums
    c = p.cells
    p.DatumPointByCoordinate(coords=(Width_,Length_,
    ThicknessPlyesCohesive[i]))
    Datum_1 = p.datums[p.datums.keys()[-1]]
    p.DatumPointByCoordinate(coords=( Width_,-Length_,
    ThicknessPlyesCohesive[i]))
    Datum_2 = p.datums[p.datums.keys()[-1]]
    p.DatumPointByCoordinate(coords=( -Width_,Length_,
    ThicknessPlyesCohesive[i]))
    Datum_3 = p.datums[p.datums.keys()[-1]]
    p.PartitionCellByPlaneThreePoints(point1=Datum_1, point2=Datum_2,
    point3=Datum_3, cells=c)

# Vertical cut (partition) of the top and bottom face
p = mdb.models['Model-1'].parts['Airborne-Composite']
e1, v1, d1 = p.edges, p.vertices, p.datums
c = p.cells
p.DatumPointByCoordinate(coords=(0.0, 0.0,0.0))
Datum_1 = p.datums[p.datums.keys()[-1]]
p.DatumPointByCoordinate(coords=(0.0, Length_,0.0))
Datum_2 = p.datums[p.datums.keys()[-1]]
p.DatumPointByCoordinate(coords=(0.0, Length_, Thickness_))
Datum_3 = p.datums[p.datums.keys()[-1]]
p.PartitionCellByPlaneThreePoints(point1=Datum_1, point2=Datum_2,
 point3=Datum_3, cells=c)

# Horizontal cut (partition) of the top and bottom face
p = mdb.models['Model-1'].parts['Airborne-Composite']
e1, v1, d1 = p.edges, p.vertices, p.datums
c = p.cells
p.DatumPointByCoordinate(coords=(0.0, 0.0,0.0))
Datum_1 = p.datums[p.datums.keys()[-1]]
p.DatumPointByCoordinate(coords=(Width_, 0.0,0.0))
Datum_2 = p.datums[p.datums.keys()[-1]]
p.DatumPointByCoordinate(coords=(Width_, 0.0, Thickness_))
Datum_3 = p.datums[p.datums.keys()[-1]]
p.PartitionCellByPlaneThreePoints(point1=Datum_1, point2=Datum_2,
 point3=Datum_3, cells=c)

# Define Lamina for Composite
```

```
p = mdb.models['Model-1'].parts['Airborne-Composite']
lamina_Composite=[]
for i in range(len(ThicknessPlyesCumulative)):
        c = p.cells
        cells = c.findAt(
                        (((Width_)/2.0,(Length_)/2.0,
                        (ThicknessPlyesCumulative[i]-ThicknessPlyes[i]/2.0)
                        ),),
                        ((-(Width_)/2.0,(Length_)/2.0,
                        (ThicknessPlyesCumulative[i]-ThicknessPlyes[i]/2.0)
                        ),),
                        (((Width_)/2.0,-(Length_)/2.0,
                        (ThicknessPlyesCumulative[i]-ThicknessPlyes[i]/2.0)
                        ),),
                        ((-(Width_)/2.0,-(Length_)/2.0,
                        (ThicknessPlyesCumulative[i]-ThicknessPlyes[i]/2.0)
                        ),),)
        string='Lamina-'+str(i+1)
        p.Set(cells=cells, name=string)
        lamina_Composite.append(cells)

p.Set(cells=lamina_Composite, name='Lamina')

# Define Lamina for Cohesive
p = mdb.models['Model-1'].parts['Airborne-Composite']
lamina_Cohesive=[]
for i in range(len(ThicknessPlyesCumulative)):
        c = p.cells
        cells = c.findAt(
                        (((Width_)/2.0,(Length_)/2.0,
                        ThicknessPlyesCumulative[i] ),),
                        ((-(Width_)/2.0,(Length_)/2.0,
                        ThicknessPlyesCumulative[i] ),),
                        (((Width_)/2.0,-(Length_)/2.0,
                        ThicknessPlyesCumulative[i] ),),
                        ((-(Width_)/2.0,-(Length_)/2.0,
                        ThicknessPlyesCumulative[i] ),),)
        string='Lamina-Cohesive-'+str(i+1)
        p.Set(cells=cells, name=string)
        lamina_Cohesive.append(cells)

p.Set(cells=lamina_Cohesive, name='Lamina-Cohesive')


## Assign material properties
# Airborne-Composite
mdb.models['Model-1'].Material(name='Airborne-Composite')
```

```
mdb.models['Model-1'].materials['Airborne-Composite'].Density(
 table=((1390,
    ), ))
mdb.models['Model-1'].materials['Airborne-Composite'].Elastic(
    type=ENGINEERING_CONSTANTS, table=((166010000000.0, 4950000000.0,
    4950000000.0, 0.3115, 0.3115, 0.0093, 1810000000.0, 1810000000.0,
    1810000000.0, 20.0), (165990000000.0, 4840000000.0, 4840000000.0,
        0.3115,
    0.3115, 0.00908, 1760000000.0, 1760000000.0, 1760000000.0, 70.0),
        (
    165900000000.0, 4410000000.0, 4410000000.0, 0.3115, 0.3115, 0.00828,
    1600000000.0, 1600000000.0, 1600000000.0, 393.0), (165800000000.0,
    3740000000.0, 3750000000.0, 0.3115, 0.3115, 0.00705, 1370000000.0,
    1370000000.0, 1370000000.0, 443.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Airborne-Composite'].Expansion(
    type=ORTHOTROPIC, table=((4e-07, 3.9e-05, 3.9e-05, 293.0), (4.2e-07,
        3.9e-05, 3.9e-05,
    343.0), (3.9e-07, 4e-05, 4e-05, 393.0), (3.36e-07, 4.1e-05, 4.1e-05,
    433.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Airborne-Composite'].SpecificHeat(
 table=((919.0, 293.0), (938.4, 393.0), (996.7,
    413.0), (1035.5, 433.0), (1074.4, 453.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Airborne-Composite'].Conductivity(
    type=ORTHOTROPIC, table=((20.0, 1.0, 1.0, 20.0), (21.0, 2.0,
    2.0, 393.0), (22.0, 3.0, 3.0, 443.0)), temperatureDependency=ON)

# Cohesive
mdb.models['Model-1'].Material(name='Cohesive')
mdb.models['Model-1'].materials['Cohesive'].Density(table=((1200,
    ), ))
mdb.models['Model-1'].materials['Cohesive'].Elastic(
    type=ISOTROPIC, table=((2250000000.0, 0.3, 293.0), (2200000000.0,
    0.3, 343.0), (2000000000.0, 0.3, 393.0), (1700000000.0, 0.3, 443.0)),
        temperatureDependency=ON)
mdb.models['Model-1'].materials['Cohesive'].Expansion(
    type=ISOTROPIC, table=((
    6.7e-05, 293.0), (7.2e-05, 333.0), (7.4e-05, 353.0), (7.5e-05, 373.0),
        (
    7.7e-05, 393.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Cohesive'].SpecificHeat(table=((1100.0,
  293.0), (1150.0, 393.0), (1300.0,
    413.0), (1400.0, 433.0), (1500.0, 453.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Cohesive'].Conductivity(
    type=ISOTROPIC, table=((0.2, 293.0), (0.26, 393.0), (0.28,
    443.0), (0.3, 493.0)), temperatureDependency=ON)

## Create material sections (Composite and Cohesive)
```

```
mdb.models['Model-1'].HomogeneousSolidSection(name='Composite-Section',
    material='Airborne-Composite', thickness=None)

mdb.models['Model-1'].HomogeneousSolidSection(name='Cohesive-Section',
    material='Cohesive', thickness=None)

# Assign material sections
mdb.models['Model-1'].parts['Airborne-Composite'].SectionAssignment(
offset=0.0, region=
    mdb.models['Model-1'].parts['Airborne-Composite'].sets['Lamina'],
        sectionName=
    'Composite-Section')

mdb.models['Model-1'].parts['Airborne-Composite'].SectionAssignment(
offset=0.0, region=
    mdb.models['Model-1'].parts['Airborne-Composite'].
        sets['Lamina-Cohesive'], sectionName=
    'Cohesive-Section')

## Define the orientation and assign Orientation for Composite

mdb.models['Model-1'].parts['Airborne-Composite'].DatumCsysByThreePoints(
coordSysType=
    CARTESIAN, line1=(1.0, 0.0, 0.0), line2=(0.0, 1.0, 0.0), name=
    'Datum csys-1', origin=(0.0, 0.0, 0.0))

DatumOrientation_1 = p.datums[p.datums.keys()[-1]]

for i in range(len(OrientationPlyes)):
        string='Lamina-'+str(i+1)
        mdb.models['Model-1'].parts['Airborne-Composite'].
        MaterialOrientation(angle=OrientationPlyes[i], axis=
                        AXIS_3, localCsys=DatumOrientation_1,
                        orientationType=SYSTEM, region=mdb.models['Model-1'].
                        parts['Airborne-Composite'].sets[string])

## Teflon

# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize=sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=
(Sides_, Sides_),
    point2=(-Sides_, -Sides_))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Teflon',
 type=
    DEFORMABLE_BODY)
```

```
mdb.models['Model-1'].parts['Teflon'].BaseSolidExtrude(depth=thickness,
 sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition

# First Side
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
 name='__profile__',
    sheetSize=sheet_Size, transform=
    mdb.models['Model-1'].parts['Teflon'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['Teflon'].faces.findAt((
        Sides/6.0,
    Sides/6.0, thickness), ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Teflon'].edges.findAt((
        -Sides/2.0,
    -Sides/4.0, thickness), ), sketchOrientation=RIGHT, origin=(0.0,
        0.0, thickness)))
mdb.models['Model-1'].parts['Teflon'].projectReferencesOntoSketch(
 filter=
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
 Sides/2.0), point2=
    (0.0, -Sides/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
addUndoState=
    False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
        0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
        Sides/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        Sides/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
```

```
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            Sides/2.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        -Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
        -Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        -Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        -Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        -Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-Sides/2.0,
  0.0),
        point2=(Sides/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
        addUndoState=False, entity=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
            0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
            -Sides/2.0,
        0.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
            0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        0.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
```

```
            -Sides/2.0,
        0.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        -Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0, 0.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        Sides/2.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        -Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0, 0.0),
        ))
mdb.models['Model-1'].parts['Teflon'].PartitionFaceBySketch(faces=
        mdb.models['Model-1'].parts['Teflon'].faces.findAt(((Sides/6.0,
            Sides/6.0, thickness),
        ), ), sketch=mdb.models['Model-1'].sketches['__profile__'],
            sketchUpEdge=
        mdb.models['Model-1'].parts['Teflon'].edges.findAt((-Sides/2.0,
            -Sides/4.0,
        thickness), ))
del mdb.models['Model-1'].sketches['__profile__']

# Second Side
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
  name='__profile__',
        sheetSize=sheet_Size, transform=
        mdb.models['Model-1'].parts['Teflon'].MakeSketchTransform(
        sketchPlane=mdb.models['Model-1'].parts['Teflon'].faces.findAt((
```

```
            -Sides/6.0,
        Sides/6.0, 0.0), ), sketchPlaneSide=SIDE1,
        sketchUpEdge=mdb.models['Model-1'].parts['Teflon'].edges.findAt((
            -Sides/4.0,
        Sides/2.0, 0.0), ), sketchOrientation=RIGHT, origin=(0.0, 0.0,
            0.0)))
mdb.models['Model-1'].parts['Teflon'].projectReferencesOntoSketch(
 filter=
        COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
   Sides/2.0), point2=
        (0.0, -Sides/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
 addUndoState=
        False, entity=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
            0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
            Sides/2.0,
        Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
            0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            Sides/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        Sides/2.0, Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            Sides/2.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
-Sides/2.0,
        -Sides/2.0))
```

```
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
    -Sides/2.0), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -Sides/2.0,
    -Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -Sides/2.0,
    -Sides/2.0), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        Sides/2.0,
    -Sides/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
    -Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-Sides/2.0,
  0.0),
    point2=(Sides/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -Sides/2.0,
    0.0), ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -Sides/2.0,
    Sides/2.0), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -Sides/2.0,
    0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -Sides/2.0,
    0.0), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -Sides/2.0,
    Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -Sides/2.0,
    Sides/2.0), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
```

```
            -Sides/2.0,
        -Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -Sides/2.0,
        0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        Sides/2.0, 0.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
            Sides/2.0,
        -Sides/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        -Sides/2.0), ), entity2=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            Sides/2.0,
        Sides/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        Sides/2.0, 0.0), ))
mdb.models['Model-1'].parts['Teflon'].PartitionFaceBySketch(faces=
        mdb.models['Model-1'].parts['Teflon'].faces.findAt(((-Sides/6.0,
            Sides/6.0, 0.0),
        )), sketch=mdb.models['Model-1'].sketches['__profile__'],
            sketchUpEdge=
        mdb.models['Model-1'].parts['Teflon'].edges.findAt((-Sides/4.0,
            Sides/2.0, 0.0),
        ))
del mdb.models['Model-1'].sketches['__profile__']

# Teflon Properties

mdb.models['Model-1'].Material(name='Teflon')
mdb.models['Model-1'].materials['Teflon'].Density(table=((2200.0, ), ))
mdb.models['Model-1'].materials['Teflon'].Elastic(table=((200000000000.0,
  0.3), ))
mdb.models['Model-1'].materials['Teflon'].Expansion(table=((0.00012, ),
  ))
mdb.models['Model-1'].materials['Teflon'].Conductivity(table=((0.25, ),
  ))
mdb.models['Model-1'].materials['Teflon'].SpecificHeat(table=((970.0, ),
  ))

# Section Assigned

```

```
mdb.models['Model-1'].parts['Teflon'].Set(cells=
    mdb.models['Model-1'].parts['Teflon'].cells.findAt(((-Sides/6.0,
        Sides/6.0, 0.0),
    )), name='Teflon')
mdb.models['Model-1'].HomogeneousSolidSection(material='Teflon', name=
    'Teflon-Section', thickness=None)
mdb.models['Model-1'].parts['Teflon'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Teflon'].sets['Teflon'], sectionName=
    'Teflon-Section', thicknessAssignment=FROM_SECTION)

#Stainless Steels
#### Plates ####
# Dimensions
Grid = 0.04
TopPlate = 0.6
TopPlate_ = TopPlate/2.0
BottomPlate = 1.0
BottomPlate_= BottomPlate/2.0

sheet_Size = 1
PlatesDepth = 0.0005


## SS-Top

# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize=sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(
TopPlate_, TopPlate_),
    point2=(-TopPlate_, -TopPlate_))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='SS_Top',
 type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['SS_Top'].BaseSolidExtrude(depth=PlatesDepth,
 sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition Top Plate
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
 name='__profile__',
    sheetSize=sheet_Size, transform=
    mdb.models['Model-1'].parts['SS_Top'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['SS_Top'].faces.findAt((
        TopPlate/6.0, TopPlate/6.0,
```

```
    PlatesDepth), ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['SS_Top'].edges.findAt((
        -TopPlate/2.0,
    -TopPlate/4.0, PlatesDepth), ), sketchOrientation=RIGHT, origin=(0.0,
        0.0, PlatesDepth)))
mdb.models['Model-1'].parts['SS_Top'].projectReferencesOntoSketch(
filter=
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
 TopPlate/2.0), point2=(
    0.0, -TopPlate/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
addUndoState=
    False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
        0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -TopPlate/2.22, TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        TopPlate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -TopPlate/2.22, TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -TopPlate/2.0, TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    TopPlate/2.0, TopPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        TopPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        -TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
```

```
    TopPlate/2.22, -TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        TopPlate/2.0, -TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -TopPlate/2.0, -TopPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        -TopPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-TopPlate/2.0,
 0.0), point2=(
    TopPlate/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -TopPlate/2.22, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -TopPlate/2.0,
    -TopPlate/2.22), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -TopPlate/2.22, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -TopPlate/2.0, 0.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -TopPlate/2.0, -TopPlate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -TopPlate/2.0, -TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -TopPlate/2.0, TopPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -TopPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
```

```
            TopPlate/2.0, 0.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        TopPlate/2.0, TopPlate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            TopPlate/2.0, TopPlate/2.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
            findAt((
        TopPlate/2.0, -TopPlate/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            TopPlate/2.0, 0.0),
        ))
mdb.models['Model-1'].parts['SS_Top'].PartitionFaceBySketch(faces=
        mdb.models['Model-1'].parts['SS_Top'].faces.findAt(((TopPlate/6.0,
            TopPlate/6.0, PlatesDepth), )),
        sketch=mdb.models['Model-1'].sketches['__profile__'], sketchUpEdge=
        mdb.models['Model-1'].parts['SS_Top'].edges.findAt((-TopPlate/2.0,
            -TopPlate/4.0, PlatesDepth),
        ))
del mdb.models['Model-1'].sketches['__profile__']


#BottomPlate

# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
   sheetSize=sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=
(BottomPlate_, BottomPlate_),
        point2=(-BottomPlate_, -BottomPlate_))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='SS_Bottom',
   type=
        DEFORMABLE_BODY)
mdb.models['Model-1'].parts['SS_Bottom'].BaseSolidExtrude(depth=
PlatesDepth, sketch=
        mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition Bottom Plate
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
  name='__profile__',
        sheetSize=sheet_Size, transform=
        mdb.models['Model-1'].parts['SS_Bottom'].MakeSketchTransform(
        sketchPlane=mdb.models['Model-1'].parts['SS_Bottom'].faces.findAt((
        BottomPlate/6.0, BottomPlate/6.0, PlatesDepth), ),
```

```
            sketchPlaneSide=SIDE1,
        sketchUpEdge=mdb.models['Model-1'].parts['SS_Bottom'].edges.findAt((
            -BottomPlate/2.0,
        -BottomPlate/4.0, PlatesDepth), ), sketchOrientation=RIGHT,
            origin=(0.0, 0.0, PlatesDepth)))
mdb.models['Model-1'].parts['SS_Bottom'].projectReferencesOntoSketch(
 filter=
        COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
  BottomPlate/2.0), point2=(
        0.0, -BottomPlate/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
 addUndoState=
        False, entity=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
            0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
            -BottomPlate/2.22, BottomPlate/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            BottomPlate/2.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        -BottomPlate/2.22, BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            -BottomPlate/2.0, BottomPlate/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
            findAt((
        BottomPlate/2.0, BottomPlate/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            BottomPlate/2.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
            -BottomPlate/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
```

```
    BottomPlate/2.22, -BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        BottomPlate/2.0, -BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -BottomPlate/2.0, -BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        -BottomPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(
-BottomPlate/2.0, 0.0), point2=(
    BottomPlate/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -BottomPlate/2.22, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -BottomPlate/2.0,
    -BottomPlate/2.22), ), entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
        -BottomPlate/2.22, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -BottomPlate/2.0, 0.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -BottomPlate/2.0, -BottomPlate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -BottomPlate/2.0, -BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -BottomPlate/2.0, BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
        -BottomPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
```

```
            BottomPlate/2.0, 0.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        BottomPlate/2.0, BottomPlate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            BottomPlate/2.0, BottomPlate/2.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
            findAt((
        BottomPlate/2.0, -BottomPlate/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((
            BottomPlate/2.0, 0.0),
        ))
mdb.models['Model-1'].parts['SS_Bottom'].PartitionFaceBySketch(faces=
        mdb.models['Model-1'].parts['SS_Bottom'].faces.findAt(((
            BottomPlate/6.0, BottomPlate/6.0,
        PlatesDepth), )), sketch=mdb.models['Model-1'].sketches
            ['__profile__'],
        sketchUpEdge=mdb.models['Model-1'].parts['SS_Bottom'].edges.findAt((
            -BottomPlate/2.0,
        -BottomPlate/4.0, PlatesDepth), ))
del mdb.models['Model-1'].sketches['__profile__']

# Stainless Steel Properties
mdb.models['Model-1'].Material(name='Stainless␣Steel')
mdb.models['Model-1'].materials['Stainless␣Steel'].Density(table=
((8000.0, ),
        ))
mdb.models['Model-1'].materials['Stainless␣Steel'].Elastic(table=((
        200000000000.0, 0.28), ))
mdb.models['Model-1'].materials['Stainless␣Steel'].Conductivity(table=
((16.2,
        20.0), (16.3, 70.0), (16.4, 120.0), (16.6, 170.0), (17.0, 210.0)),
        temperatureDependency=ON)
mdb.models['Model-1'].materials['Stainless␣Steel'].SpecificHeat(table=
((500.0,
        ), ))
mdb.models['Model-1'].materials['Stainless␣Steel'].Expansion(table=
((1.76e-05,
        ), ))

# Section for Stainless Steels
mdb.models['Model-1'].HomogeneousSolidSection(material='Stainless␣Steel',
  name=
        'SS-Section', thickness=None)
mdb.models['Model-1'].parts['SS_Top'].Set(cells=
```

```
        mdb.models['Model-1'].parts['SS_Top'].cells.findAt(((TopPlate/2.0,
            TopPlate/6.0, PlatesDepth/1.5),
        )), name='TopPlate_Whole_Set')
mdb.models['Model-1'].parts['SS_Top'].SectionAssignment(offset=0.0,
        offsetField='', offsetType=MIDDLE_SURFACE, region=
        mdb.models['Model-1'].parts['SS_Top'].sets['TopPlate_Whole_Set'],
        sectionName='SS-Section', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['SS_Bottom'].Set(cells=
        mdb.models['Model-1'].parts['SS_Bottom'].cells.findAt(((
            BottomPlate/2.0, BottomPlate/6.0,
        PlatesDepth/1.5), )), name='BottomPlate_Whole_Set')
mdb.models['Model-1'].parts['SS_Bottom'].SectionAssignment(offset=0.0,
        offsetField='', offsetType=MIDDLE_SURFACE, region=
        mdb.models['Model-1'].parts['SS_Bottom'].sets
            ['BottomPlate_Whole_Set'],
        sectionName='SS-Section', thicknessAssignment=FROM_SECTION)

# Steps for Heat Transfer
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Heat
        , minInc=0.0004, name='Heat1', previous='Initial',
            timePeriod=time_Heat)
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Couling
        , minInc=0.0002, name='Cooling1', previous='Heat1',
            timePeriod=time_Couling)
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Heat
        , minInc=0.0004, name='Heat2', previous='Cooling1',
            timePeriod=time_Heat)
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Couling
        , minInc=0.0002, name='Cooling2', previous='Heat2',
            timePeriod=time_Couling)
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Heat
        , minInc=0.0004, name='Heat3', previous='Cooling2',
            timePeriod=time_Heat)
mdb.models['Model-1'].HeatTransferStep(deltmx=60.0, initialInc=0.1,
  maxInc=time_Couling
        , minInc=0.0002, name='Cooling3', previous='Heat3',
            timePeriod=time_Couling)

## Assembly
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name=
        'Airborne-Composite-1', part=
        mdb.models['Model-1'].parts['Airborne-Composite'])
```

```
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
  name='Teflon-1',
      part=mdb.models['Model-1'].parts['Teflon'])
mdb.models['Model-1'].rootAssembly.translate(instanceList=(
      'Airborne-Composite-1', ), vector=(0.0, 0.0, thickness))

# Assign Teflon Surf and Set
mdb.models['Model-1'].rootAssembly.Set(faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.findAt(((
      Sides/6.0, -Sides/6.0, 0.0), ), ((Sides/3.0, Sides/6.0, 0.0), ),
          ((-Sides/3.0, -Sides/6.0, 0.0), ), (
      (-Sides/6.0, Sides/6.0, 0.0), ), ), name='Teflon-Set-Bottom')

mdb.models['Model-1'].rootAssembly.Surface(name='Teflon-Surf-Bottom',
      side1Faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.
          findAt(((
      Sides/6.0, -Sides/6.0, 0.0), ), ((Sides/3.0, Sides/6.0, 0.0), ),
          ((-Sides/3.0, -Sides/6.0, 0.0), ), (
      (-Sides/6.0, Sides/6.0, 0.0), ), ))

mdb.models['Model-1'].rootAssembly.Surface(name='Teflon-Surf-Top',
  side1Faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.
          findAt(((
      Sides/6.0, -Sides/3.0, thickness), ), ((-Sides/6.0, -Sides/6.0,
          thickness), ), ((Sides/6.0, Sides/6.0,
      thickness), ), ((-Sides/6.0, Sides/3.0, thickness), ), ))

# Create a set for the Laminate
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.Set(cells=lamina_Composite + lamina_Cohesive, name='C-Set-InitialTemp')

mdb.models['Model-1'].rootAssembly.Set(faces=
     mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
          faces.findAt(
      ((Width/3.0, Length/6.0, Thickness_+thickness), ),
          ((-Width/3.0, Length/6.0, Thickness_+thickness), ), ((Width/3.0,
     -Length/6.0, Thickness_+thickness), ), ((-Width/6.0, -Length/6.0,
          Thickness_+thickness), ), ), name='Comp-Set-Top')

# Surf for convection
p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
surf_contact_1_A=[]

surf_contact_1_A.append(s.findAt(
```

```
                              (((Width_)/2.0,(Length_)/2.0, Thickness_ ),),
                              ((-(Width_)/2.0,(Length_)/2.0, Thickness_ ),),
                              (((Width_)/2.0,-(Length_)/2.0, Thickness_ ),),
                              ((-(Width_)/2.0,-(Length_)/2.0, Thickness_ ),),))

surf_contact_1_A.append(s.findAt(
                              (((Width_)/2.0,(Length_)/2.0, 0.0 ),),
                              ((-(Width_)/2.0,(Length_)/2.0, 0.0 ),),
                              (((Width_)/2.0,-(Length_)/2.0, 0.0 ),),
                              ((-(Width_)/2.0,-(Length_)/2.0, 0.0 ),),))
p.Surface(side1Faces=surf_contact_1_A, name='C-Surf-Conv-Radiation')

# Top and bottom surfaces for composite
p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
p.Surface(side1Faces=surf_contact_1_A[0], name='C-Surf-Top')

p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
p.Surface(side1Faces=surf_contact_1_A[1], name='C-Surf-Bottom')

# Assembly Plates

mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
 name='SS_Top-1',
    part=mdb.models['Model-1'].parts['SS_Top'])
mdb.models['Model-1'].rootAssembly.translate(instanceList=('SS_Top-1', ),
    vector=(TopPlate, 0.0, 0.0))
mdb.models['Model-1'].rootAssembly.rotate(angle=180.0,
 axisDirection=(TopPlate/2.0, 0.0,
    0.0), axisPoint=(TopPlate, 0.0, PlatesDepth), instanceList=
        ('SS_Top-1', ))
mdb.models['Model-1'].rootAssembly.CoincidentPoint(fixedPoint=
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
        vertices.findAt(
    (0.0, 0.0, Thickness_+thickness), ), movablePoint=
    mdb.models['Model-1'].rootAssembly.instances['SS_Top-1'].vertices.
        findAt((
    TopPlate, 0.0, PlatesDepth), ))
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
 name='SS_Bottom-1',
    part=mdb.models['Model-1'].parts['SS_Bottom'])
mdb.models['Model-1'].rootAssembly.translate(
 instanceList=('SS_Bottom-1', ),
    vector=(1.0, 0.0, 0.0))
mdb.models['Model-1'].rootAssembly.CoincidentPoint(fixedPoint=
    mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].vertices.
```

```
            findAt((
    0.0, 0.0, 0.0), ), movablePoint=
    mdb.models['Model-1'].rootAssembly.instances['SS_Bottom-1'].
            vertices.findAt(
    (1.0, 0.0, PlatesDepth), ))

# Assign Sets and Surfaces for the Plates

mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['SS_Top-1'].faces.
            findAt(((
    -TopPlate/6.0, -TopPlate/6.0, Thickness_+thickness+PlatesDepth),
            )), name='SS_Top_Set')
mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['SS_Bottom-1'].faces.
            findAt(((
    -BottomPlate/6.0, BottomPlate/6.0, -PlatesDepth), )),
            name='SS_Bottom_Set')

mdb.models['Model-1'].rootAssembly.Surface(name='SS_Top_Surf',
  side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['SS_Top-1'].faces.
            findAt(((
    TopPlate/6.0, TopPlate/3.0, thickness+Thickness_), ),
            ((-TopPlate/6.0, TopPlate/6.0, thickness+Thickness_), ),
            ((TopPlate/6.0, -TopPlate/6.0, thickness+Thickness_), ),
    ((-TopPlate/6.0, -TopPlate/3.0, thickness+Thickness_), ), ))
mdb.models['Model-1'].rootAssembly.Surface(name='SS_Bottom_Surf',
  side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['SS_Bottom-1'].faces.
            findAt(((
    BottomPlate/6.0, -BottomPlate/3.0, 0.0), ), ((-BottomPlate/6.0,
            -BottomPlate/6.0, 0.0), ), ((BottomPlate/6.0,
    BottomPlate/6.0, 0.0), ), ((-BottomPlate/6.0, BottomPlate/3.0, 0.0),
            ), ))


## Surface Contact

# Assign Contact Property
mdb.models['Model-1'].ContactProperty('Comp-Teflon')
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
TangentialBehavior(
    dependencies=0, directionality=ISOTROPIC, formulation=LAGRANGE,
    pressureDependency=OFF, shearStressLimit=None,
            slipRateDependency=OFF,
    table=((ContactValue, ), ), temperatureDependency=OFF)
```

```
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
NormalBehavior(
    allowSeparation=ON, clearanceAtZeroContactPressure=0.0,
    constraintEnforcementMethod=AUGMENTED_LAGRANGE,
        contactStiffness=DEFAULT,
    contactStiffnessScaleFactor=1.0, pressureOverclosure=HARD)
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
ThermalConductance(
    clearanceDepTable=((Thermal_Conductance_Comp_Teflon, 0.0),
        (0.0, 0.001)), clearanceDependency=ON,
    definition=TABULAR, dependenciesC=0, massFlowRateDependencyC=OFF,
    pressureDependency=OFF, temperatureDependencyC=OFF)

mdb.models['Model-1'].ContactProperty('Plates-Polymers')
mdb.models['Model-1'].interactionProperties['Plates-Polymers'].
TangentialBehavior(
    dependencies=0, directionality=ISOTROPIC, formulation=LAGRANGE,
    pressureDependency=OFF, shearStressLimit=None,
        slipRateDependency=OFF,
    table=((ContactValue, ), ), temperatureDependency=OFF)
mdb.models['Model-1'].interactionProperties['Plates-Polymers'].
NormalBehavior(
    allowSeparation=ON, clearanceAtZeroContactPressure=0.0,
    constraintEnforcementMethod=AUGMENTED_LAGRANGE,
        contactStiffness=DEFAULT,
    contactStiffnessScaleFactor=1.0, pressureOverclosure=HARD)
mdb.models['Model-1'].interactionProperties['Plates-Polymers'].
ThermalConductance(
    clearanceDepTable=((Thermal_Conductance_Plates_Polymers, 0.0),
        (0.0, 0.001)), clearanceDependency=ON,
    definition=TABULAR, dependenciesC=0, massFlowRateDependencyC=OFF,
    pressureDependency=OFF, temperatureDependencyC=OFF)

# Assign interaction
mdb.models['Model-1'].SurfaceToSurfaceContactStd(adjustMethod=
OVERCLOSED,
    clearanceRegion=None, createStepName='Initial', datumAxis=None,
    initialClearance=OMIT, interactionProperty='Comp-Teflon', master=
    mdb.models['Model-1'].rootAssembly.surfaces['Teflon-Surf-Top'],
        name=
    'C-T-inter', slave=
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
        surfaces['C-Surf-Bottom']
    , sliding=FINITE, surfaceSmoothing=AUTOMATIC, thickness=ON,
        tied=OFF)
mdb.models['Model-1'].SurfaceToSurfaceContactStd(adjustMethod=
OVERCLOSED,
```

```
    clearanceRegion=None, createStepName='Initial', datumAxis=None,
    initialClearance=OMIT, interactionProperty='Plates-Polymers',
        master=
    mdb.models['Model-1'].rootAssembly.surfaces['SS_Top_Surf'],
        name=
    'Comp-TopPlate', slave=
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
        surfaces['C-Surf-Top']
    , sliding=FINITE, surfaceSmoothing=AUTOMATIC, thickness=ON, tied=OFF)
mdb.models['Model-1'].SurfaceToSurfaceContactStd(adjustMethod=OVERCLOSED,
    clearanceRegion=None, createStepName='Initial', datumAxis=None,
    initialClearance=OMIT, interactionProperty='Plates-Polymers',
        master=
    mdb.models['Model-1'].rootAssembly.surfaces['SS_Bottom_Surf'],
        name=
    'Teflon_BottomPlate', slave=
    mdb.models['Model-1'].rootAssembly.surfaces['Teflon-Surf-Bottom'],
        sliding=
    FINITE, surfaceSmoothing=AUTOMATIC, thickness=ON, tied=OFF)

# Convection
mdb.models['Model-1'].rootAssembly.Surface(name='SS_Convection',
 side1Faces=
    mdb.models['Model-1'].rootAssembly.instances['SS_Top-1'].faces.
        findAt(((
    -TopPlate/6.0, -TopPlate/6.0, Thickness_+thickness+PlatesDepth), ),
        )+\
    mdb.models['Model-1'].rootAssembly.instances['SS_Bottom-1'].faces.
        findAt(((
    BottomPlate/6.0, -BottomPlate/6.0, -PlatesDepth), ), ))
mdb.models['Model-1'].FilmCondition(createStepName='Cooling1',
 definition=
    EMBEDDED_COEFF, filmCoeff=Convection, filmCoeffAmplitude='',
        name='Convection_1',
    sinkAmplitude='', sinkDistributionType=UNIFORM, sinkFieldName='',
    sinkTemperature=sink_temperature, surface=
    mdb.models['Model-1'].rootAssembly.surfaces['SS_Convection'])
mdb.models['Model-1'].FilmCondition(createStepName='Cooling2',
 definition=
    EMBEDDED_COEFF, filmCoeff=Convection, filmCoeffAmplitude='',
        name='Convection_2',
    sinkAmplitude='', sinkDistributionType=UNIFORM, sinkFieldName='',
    sinkTemperature=sink_temperature, surface=
    mdb.models['Model-1'].rootAssembly.surfaces['SS_Convection'])
mdb.models['Model-1'].FilmCondition(createStepName='Cooling2',
 definition=
    EMBEDDED_COEFF, filmCoeff=Convection, filmCoeffAmplitude='',
```

```
            name='Convection_3',
        sinkAmplitude='', sinkDistributionType=UNIFORM, sinkFieldName='',
        sinkTemperature=sink_temperature, surface=
        mdb.models['Model-1'].rootAssembly.surfaces['SS_Convection'])
mdb.models['Model-1'].interactions['Convection_1'].deactivate('Heat2')
mdb.models['Model-1'].interactions['Convection_2'].deactivate('Heat3')
mdb.models['Model-1'].interactions['Convection_3'].move('Cooling2',
  'Heat3')
mdb.models['Model-1'].interactions['Convection_3'].move('Heat3',
  'Cooling3')

# Assign Temperature Loads
# Loads
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET,
  createStepName='Heat1',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=210.0, name=
        'Heat1_Top', region=mdb.models['Model-1'].rootAssembly.sets
            ['SS_Top_Set'])
mdb.models['Model-1'].boundaryConditions['Heat1_Top'].deactivate
('Cooling1')
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET,
  createStepName='Heat2',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=130.0, name=
        'Heat2_Top', region=mdb.models['Model-1'].rootAssembly.
            sets['SS_Top_Set'])
mdb.models['Model-1'].boundaryConditions['Heat2_Top'].deactivate
('Cooling2')
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET,
  createStepName='Heat3',
        distributionType=UNIFORM, fieldName='', fixed=OFF, magnitude=80.0,
            name=
        'Heat3_Top', region=mdb.models['Model-1'].rootAssembly.
            sets['SS_Top_Set'])
mdb.models['Model-1'].boundaryConditions['Heat3_Top'].deactivate
('Cooling3')
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET,
  createStepName='Heat1',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=210.0, name=
        'Heat1_Bottom', region=
        mdb.models['Model-1'].rootAssembly.sets['SS_Bottom_Set'])
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET,
  createStepName='Heat2',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=130.0, name=
```

```
    'Heat2_Bottom', region=
    mdb.models['Model-1'].rootAssembly.sets['SS_Bottom_Set'])
mdb.models['Model-1'].boundaryConditions['Heat1_Bottom'].deactivate
('Cooling1')
mdb.models['Model-1'].boundaryConditions['Heat2_Bottom'].deactivate
('Cooling2')
mdb.models['Model-1'].TemperatureBC(amplitude=UNSET, createStepName
='Heat3',
    distributionType=UNIFORM, fieldName='', fixed=OFF, magnitude=80.0,
        name=
    'Heat3_Bottom', region=
    mdb.models['Model-1'].rootAssembly.sets['SS_Bottom_Set'])
mdb.models['Model-1'].boundaryConditions['Heat3_Bottom'].deactivate
('Cooling3')

# Field Output (Thermal Analysis)
mdb.models['Model-1'].FieldOutputRequest(createStepName='Heat1', name=
    'F-Output-1', variables=('NT', 'TEMP', 'FTEMP', 'HFL', 'HFLA', 'HTL',
    'HTLA', 'RFLE', 'RFL', 'CFL', 'NFLUX', 'RADFL', 'RADFLA', 'RADTL',
    'RADTLA', 'VFTOT', 'SJD', 'SJDA', 'SJDT', 'SJDTA', 'WEIGHT', 'FLUXS',
    'HBF', 'FILMCOEF', 'SINKTEMP'))

# History Output
mdb.models['Model-1'].HistoryOutputRequest(createStepName='Heat1',
 name=
    'H-Output-1', variables=('FTEMP', 'HFLA', 'HTL', 'HTLA', 'RADFL',
        'RADFLA',
    'RADTL', 'RADTLA', 'VFTOT', 'SJD', 'SJDA', 'SJDT', 'SJDTA',
        'WEIGHT'))

# Mesh Generation for Laminate and Teflon
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.seedPart(size=0.01, deviationFactor=0.1, minSizeFactor=0.1)
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.generateMesh()

t = mdb.models['Model-1'].parts['Teflon']
t.seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=0.011)
mdb.models['Model-1'].parts['Teflon'].generateMesh()

# Mesh Generation for Plates

mdb.models['Model-1'].parts['SS_Top'].seedPart(deviationFactor=0.1,
    minSizeFactor=0.1, size=0.03)
mdb.models['Model-1'].parts['SS_Top'].generateMesh()

mdb.models['Model-1'].parts['SS_Bottom'].seedPart(deviationFactor=0.1,
```

```
    minSizeFactor=0.1, size=0.05)
mdb.models['Model-1'].parts['SS_Bottom'].generateMesh()


#Elements for Laminate and Teflon
elemType1 = mesh.ElemType(elemCode=DC3D6, elemLibrary=STANDARD,
    secondOrderAccuracy=OFF, distortionControl=DEFAULT)
elemType2 = mesh.ElemType(elemCode=DC3D4, elemLibrary=STANDARD)
p = mdb.models['Model-1'].parts['Airborne-Composite']
c = p.cells
cells = lamina_Composite + lamina_Cohesive
pickedRegions =(cells, )
p.setElementType(regions=pickedRegions, elemTypes=(elemType1,
 elemType2, ))

mdb.models['Model-1'].parts['Teflon'].setElementType(elemTypes=
(ElemType(
    elemCode=DC3D8, elemLibrary=STANDARD), ElemType(elemCode=DC3D6,
    elemLibrary=STANDARD), ElemType(elemCode=DC3D4, elemLibrary=
        STANDARD)),
    regions=(mdb.models['Model-1'].parts['Teflon'].cells.findAt(((
        -Sides/6.0,
    Sides/6.0, 0.0), )), ))
mdb.models['Model-1'].rootAssembly.regenerate()

# Elements for Plates

mdb.models['Model-1'].parts['SS_Top'].setElementType(elemTypes=
(ElemType(
    elemCode=DC3D8, elemLibrary=STANDARD), ElemType(elemCode=DC3D6,
    elemLibrary=STANDARD), ElemType(elemCode=DC3D4, elemLibrary=
        STANDARD)),
    regions=(mdb.models['Model-1'].parts['SS_Top'].cells.findAt(((
        TopPlate/2.0, TopPlate/6.0,
    PlatesDepth/1.5), )), ))

mdb.models['Model-1'].parts['SS_Bottom'].setElementType(elemTypes=(
ElemType(
    elemCode=DC3D8, elemLibrary=STANDARD), ElemType(elemCode=DC3D6,
    elemLibrary=STANDARD), ElemType(elemCode=DC3D4, elemLibrary=
        STANDARD)),
    regions=(mdb.models['Model-1'].parts['SS_Bottom'].cells.findAt(((
        BottomPlate/2.0,
    BottomPlate/6.0, PlatesDepth/1.5), )), ))
```

B

Appendix B

In the Appendix B, the Python code for the simulations from Chapter 4 will be provided. The structure was given in Appendix A, so in this appendix, the rest of the code that is needed to simulate the Cycles will be provided. Four smaller codes are typed, first is the simulation of the Cycles, second is the warpage simulation, third is the discrete plates scripting and fourth is the viscoelastic merging. Furthermore, the structure of the parts will only be shown in the simulation of the Cycles, where in the warpage model, only the most important changes will be given (such as the step).

B.1. Cycle's Simulation

```
# Teflon Properties for Composite Material (change if not isotropic)

mdb.models['Model-1'].Material(name='Teflon-Composite')
mdb.models['Model-1'].materials['Teflon-Composite'].Density(
table=((2275.0, ),
    ))
mdb.models['Model-1'].materials['Teflon-Composite'].Elastic(table=((
    20375000000.0, 6650000000.0, 6650000000.0, 0.35, 0.35, 0.4,
        4400000000.0,
    4400000000.0, 3000000000.0, 20.0), (20000000000.0, 6500000000.0,
    6500000000.0, 0.35, 0.35, 0.4, 4300000000.0, 4300000000.0,
        2800000000.0,
    70.0), (20000000000.0, 6200000000.0, 6200000000.0, 0.35, 0.35, 0.4,
    4000000000.0, 4000000000.0, 2600000000.0, 120.0), (20000000000.0,
    6000000000.0, 6000000000.0, 0.35, 0.35, 0.4, 3500000000.0,
        3500000000.0,
    2500000000.0, 170.0), (19000000000.0, 5900000000.0, 5900000000.0,
        0.35,
    0.35, 0.4, 3400000000.0, 3400000000.0, 2400000000.0, 210.0)),
    temperatureDependency=ON, type=ENGINEERING_CONSTANTS)
```

```
mdb.models['Model-1'].materials['Teflon-Composite'].Expansion(
 table=((2.7e-06,
    0.000126, 0.000126, 20.0), (2.8e-06, 0.000127, 0.000127, 70.0),
        (3e-06,
    0.0002, 0.0002, 120.0), (3.1e-06, 0.0002, 0.0002, 160.0),
        (3e-06, 0.00025,
    0.00025, 210.0)), temperatureDependency=ON, type=ORTHOTROPIC)
mdb.models['Model-1'].materials['Teflon-Composite'].Conductivity(
 table=((
    0.4875, 0.3115, 0.3115, 20.0), (0.5, 0.32, 0.32, 70.0),
        (0.52, 0.33, 0.33,
    120.0), (0.55, 0.35, 0.35, 170.0), (0.6, 0.4, 0.4, 210.0)),
    temperatureDependency=ON, type=ORTHOTROPIC)
mdb.models['Model-1'].materials['Teflon-Composite'].SpecificHeat(
 table=((
    1000.0, 20.0), (1050.0, 70.0), (1100.0, 120.0), (1150.0, 170.0),
        (1200.0,
    210.0)), temperatureDependency=ON)

# Section assigned for Teflon Composite and material orientation

mdb.models['Model-1'].HomogeneousSolidSection(material='Teflon-Composite',
    name='Teflon-Composite-Section', thickness=None)
mdb.models['Model-1'].parts['Teflon'].Set(cells=
    mdb.models['Model-1'].parts['Teflon'].cells.findAt(((Sides/2.0,
        Sides/3.0, thickness/6.02),
    ), ((Sides/6.0, Sides/6.0, thickness), ), ), name='Teflon_Set')
mdb.models['Model-1'].parts['Teflon'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Teflon'].sets['Teflon_Set'],
        sectionName=
    'Teflon-Composite-Section', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Teflon'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_ANGLE,
        angle=
    0.0, axis=AXIS_3, fieldName='', localCsys=None,
        orientationType=SYSTEM,
    region=Region(cells=mdb.models['Model-1'].parts['Teflon'].cells.
        findAt(((
    Sides/6.0, Sides/6.0, thickness), ), )), stackDirection=STACK_3)
mdb.models['Model-1'].parts['Teflon'].MaterialOrientation(
    additionalRotationField='', additionalRotationType=ROTATION_ANGLE,
        angle=
    90.0, axis=AXIS_3, fieldName='', localCsys=None,
        orientationType=SYSTEM,
    region=Region(cells=mdb.models['Model-1'].parts['Teflon'].cells.
        findAt(((
```

```
    Sides/2.0, Sides/3.0, thickness/6.02), ), )),
        stackDirection=STACK_3)
mdb.models['Model-1'].rootAssembly.regenerate()

# Top Plate

# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize=sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].Line(
point1=(-TopPlate, 0.0), point2=
    (0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (-TopPlate, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].
ObliqueDimension(textPoint=(
    -TopPlate, -TopPlate), value=TopPlate, vertex1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-TopPlate, 0.0),
    ), vertex2=mdb.models['Model-1'].sketches['__profile__'].
        vertices.findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].Part(dimensionality=THREE_D,
 name='Top␣Plate', type=
    ANALYTIC_RIGID_SURFACE)
mdb.models['Model-1'].parts['Top␣Plate'].AnalyticRigidSurfExtrude(
depth=TopPlate,
    sketch=mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition Cross Section
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
 name='__profile__',
    sheetSize=sheet_Size, transform=
    mdb.models['Model-1'].parts['Top␣Plate'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['Top␣Plate'].faces.findAt(
        (-TopPlate/3.0,
    0.0, -TopPlate/6.0), (0.0, 1.0, 0.0)), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Top␣Plate'].edges.findAt(
        (-TopPlate,
    0.0, TopPlate/4.0), ), sketchOrientation=RIGHT,
        origin=(-TopPlate/2.0, 0.0, 0.0)))
mdb.models['Model-1'].parts['Top␣Plate'].projectReferencesOntoSketch(
filter=
```

```
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(
 point1=(0.0, TopPlate/2.0), point2=(
    0.0, -TopPlate/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
 addUndoState=
    False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (0.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (TopPlate/2.0, TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, TopPlate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    TopPlate/2.0, TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (TopPlate/2.0, TopPlate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -TopPlate/2.0, TopPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, TopPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, -TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -TopPlate/2.0, -TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-TopPlate/2.0, -TopPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
```

```
        TopPlate/2.0, -TopPlate/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (0.0, -TopPlate/2.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=
 (-TopPlate/2.0, 0.0), point2=(
        TopPlate/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
        addUndoState=False, entity=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
            (-TopPlate/2.0, 0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
            (-TopPlate/2.0, TopPlate/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        -TopPlate/2.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (-TopPlate/2.0, 0.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        -TopPlate/2.0, TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (-TopPlate/2.0, TopPlate/2.0),
        ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
            findAt((
        -TopPlate/2.0, -TopPlate/2.0), ), midpoint=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (-TopPlate/2.0, 0.0),
        ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (TopPlate/2.0, 0.0), )
        , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
            findAt((
        TopPlate/2.0, -TopPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
        addUndoState=False, entity1=
        mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
            (TopPlate/2.0, -TopPlate/2.0),
```

```
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    TopPlate/2.0, TopPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (TopPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].parts['Top␣Plate'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Top␣Plate'].faces.findAt((
        (-TopPlate/3.0, 0.0, -TopPlate/6.0),
    )), sketch=mdb.models['Model-1'].sketches['__profile__'],
        sketchUpEdge=
    mdb.models['Model-1'].parts['Top␣Plate'].edges.findAt(
        (-TopPlate, 0.0, TopPlate/4.0), ))
del mdb.models['Model-1'].sketches['__profile__']

# Reference Point
mdb.models['Model-1'].parts['Top␣Plate'].ReferencePoint(point=
    mdb.models['Model-1'].parts['Top␣Plate'].vertices.findAt(
        (-TopPlate/2.0, 0.0, 0.0),
    ))

# Bottom Plate

# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize=sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-BottomPlate, 0.0),
 point2=
    (0.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (-BottomPlate, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].
ObliqueDimension(textPoint=(
    -BottomPlate, -BottomPlate), value=BottomPlate, vertex1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-BottomPlate, 0.0),
    ), vertex2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Bottom␣Plate',
 type=
    ANALYTIC_RIGID_SURFACE)
mdb.models['Model-1'].parts['Bottom␣Plate'].
AnalyticRigidSurfExtrude(depth=BottomPlate,
```

```
    sketch=mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition Cross Section
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
 name='__profile__',
    sheetSize=sheet_Size, transform=
    mdb.models['Model-1'].parts['Bottom␣Plate'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['Bottom␣Plate'].faces.
        findAt((-BottomPlate/3.0,
    0.0, -BottomPlate/6.0), (0.0, 1.0, 0.0)), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Bottom␣Plate'].edges.
        findAt((-BottomPlate,
    0.0, BottomPlate/4.0), ), sketchOrientation=RIGHT,
        origin=(-BottomPlate/2.0, 0.0, 0.0)))
mdb.models['Model-1'].parts['Bottom␣Plate'].
projectReferencesOntoSketch(filter=
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].
Line(point1=(0.0, BottomPlate/2.0), point2=(
    0.0, -BottomPlate/2.0))
mdb.models['Model-1'].sketches['__profile__'].
VerticalConstraint(addUndoState=
    False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((0.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (BottomPlate/2.0, BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, BottomPlate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    BottomPlate/2.0, BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (BottomPlate/2.0, BottomPlate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
```

```
    -BottomPlate/2.0, BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, BottomPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, -BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -BottomPlate/2.0, -BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-BottomPlate/2.0, -BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    BottomPlate/2.0, -BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, -BottomPlate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].
 Line(point1=(-BottomPlate/2.0, 0.0), point2=(
    BottomPlate/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((-BottomPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (-BottomPlate/2.0, BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -BottomPlate/2.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-BottomPlate/2.0, 0.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -BottomPlate/2.0, BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-BottomPlate/2.0, BottomPlate/2.0),
```

```
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -BottomPlate/2.0, -BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-BottomPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (BottomPlate/2.0, 0.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    BottomPlate/2.0, -BottomPlate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (BottomPlate/2.0, -BottomPlate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    BottomPlate/2.0, BottomPlate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (BottomPlate/2.0, 0.0),
    ))
mdb.models['Model-1'].parts['Bottom Plate'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Bottom Plate'].faces.findAt((
        (-BottomPlate/3.0, 0.0, -BottomPlate/6.0),
    )), sketch=mdb.models['Model-1'].sketches['__profile__'],
        sketchUpEdge=
    mdb.models['Model-1'].parts['Bottom Plate'].edges.findAt(
        (-BottomPlate, 0.0, BottomPlate/4.0), ))
del mdb.models['Model-1'].sketches['__profile__']

# Reference Point
mdb.models['Model-1'].parts['Bottom Plate'].ReferencePoint(point=
    mdb.models['Model-1'].parts['Bottom Plate'].vertices.findAt(
        (-BottomPlate/2.0, 0.0, 0.0),
    ))


## Assembly
# Connect Comp with Teflon
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name=
    'Airborne-Composite-1', part=
    mdb.models['Model-1'].parts['Airborne-Composite'])
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
 name='Teflon-1',
    part=mdb.models['Model-1'].parts['Teflon'])
```

```
mdb.models['Model-1'].rootAssembly.translate(instanceList=(
    'Airborne-Composite-1', ), vector=(0.0, 0.0, thickness))


# Connect Bottom Plate and assign Set(RP) and Surf
mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
 name='Bottom␣Plate-1'
    , part=mdb.models['Model-1'].parts['Bottom␣Plate'])
mdb.models['Model-1'].rootAssembly.rotate(angle=90.0,
 axisDirection=(-BottomPlate/2.0, 0.0,
    0.0), axisPoint=(-BottomPlate/2.0, 0.0, 0.0),
        instanceList=('Bottom␣Plate-1', ))

mdb.models['Model-1'].rootAssembly.CoincidentPoint(fixedPoint=
    mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].vertices.
        findAt((
    0.0, 0.0, 0.0), ), movablePoint=
    mdb.models['Model-1'].rootAssembly.instances['Bottom␣Plate-1'].
        referencePoints[3])


# Assign RP and surf for contact
mdb.models['Model-1'].rootAssembly.Set(name='BottomPlate_RP',
 referencePoints=(
    mdb.models['Model-1'].rootAssembly.instances['Bottom␣Plate-1'].
        referencePoints[3],
    ))
mdb.models['Model-1'].rootAssembly.Surface(name='BottomPlate_Surf',
    side2Faces=
    mdb.models['Model-1'].rootAssembly.instances['Bottom␣Plate-1'].
        faces.findAt(
    ((0.,0.,0),),) )

# Connect Top Plate and assign Set(RP) and Surf
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,
 name='Top␣Plate-1',
    part=mdb.models['Model-1'].parts['Top␣Plate'])
mdb.models['Model-1'].rootAssembly.rotate(angle=90.0,
 axisDirection=(-TopPlate/2.0, 0.0,
    0.0), axisPoint=(-TopPlate/1.5, 0.0, 0.0),
        instanceList=('Top␣Plate-1', ))
mdb.models['Model-1'].rootAssembly.CoincidentPoint(fixedPoint=
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
        vertices.findAt(
    (0.0, 0.0, thickness+Thickness_), ), movablePoint=
    mdb.models['Model-1'].rootAssembly.instances['Top␣Plate-1'].
```

```
            referencePoints[3])

# Assign RP and surf for contact
mdb.models['Model-1'].rootAssembly.Set(name='TopPlate_RP',
 referencePoints=(
     mdb.models['Model-1'].rootAssembly.instances['Top␣Plate-1'].
         referencePoints[3],
     ))
mdb.models['Model-1'].rootAssembly.Surface(name='TopPlate_Surf',
 side1Faces=
     mdb.models['Model-1'].rootAssembly.instances['Top␣Plate-1'].faces.
         findAt(
     ((0.,0.,thickness+Thickness_),)) )


# Assign Teflon Surf and Set
mdb.models['Model-1'].rootAssembly.Set(cells=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].cells.
         findAt(((
     -Sides/6.0, Sides/6.0, 0.0), )), name='Teflon_Set')

mdb.models['Model-1'].rootAssembly.Set(faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.
         findAt(((
     Sides/6.0, -Sides/6.0, 0.0), ), ((Sides/3.0, Sides/6.0, 0.0), ),
         ((-Sides/3.0, -Sides/6.0, 0.0), ), (
     (-Sides/6.0, Sides/6.0, 0.0), ), ), name='Teflon-Set-Bottom')

mdb.models['Model-1'].rootAssembly.Surface(name='Teflon-Surf-Bottom',
     side1Faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.
         findAt(((
     Sides/6.0, -Sides/6.0, 0.0), ), ((Sides/3.0, Sides/6.0, 0.0), ),
         ((-Sides/3.0, -Sides/6.0, 0.0), ), (
     (-Sides/6.0, Sides/6.0, 0.0), ), ))

mdb.models['Model-1'].rootAssembly.Surface(name='Teflon-Surf-Top',
 side1Faces=
     mdb.models['Model-1'].rootAssembly.instances['Teflon-1'].faces.
         findAt(((
     Sides/6.0, -Sides/3.0, thickness), ), ((-Sides/6.0, -Sides/6.0,
         thickness), ), ((Sides/6.0, Sides/6.0,
     thickness), ), ((-Sides/6.0, Sides/3.0, thickness), ), ))

# Set for the whole laminate
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.Set(cells=lamina_Composite + lamina_Cohesive,
```

```
    name='C-Set-InitialTemp')

mdb.models['Model-1'].rootAssembly.Set(faces=
        mdb.models['Model-1'].rootAssembly.
            instances['Airborne-Composite-1'].faces.findAt(
        ((Width/3.0, Length/6.0, Thickness_+thickness), ),
            ((-Width/3.0, Length/6.0, Thickness_+thickness), ), ((Width/3.0,
        -Length/6.0, Thickness_+thickness), ),
            ((-Width/6.0, -Length/6.0, Thickness_+thickness), ), ),
            name='Comp-Set-Top')

# Surf for convection
p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
surf_contact_1_A=[]

surf_contact_1_A.append(s.findAt(
                            (((Width_)/2.0,(Length_)/2.0, Thickness_ ),),
                            ((-(Width_)/2.0,(Length_)/2.0, Thickness_ ),),
                            (((Width_)/2.0,-(Length_)/2.0, Thickness_ ),),
                            ((-(Width_)/2.0,-(Length_)/2.0, Thickness_ ),),))

surf_contact_1_A.append(s.findAt(
                            (((Width_)/2.0,(Length_)/2.0, 0.0 ),),
                            ((-(Width_)/2.0,(Length_)/2.0, 0.0 ),),
                            (((Width_)/2.0,-(Length_)/2.0, 0.0 ),),
                            ((-(Width_)/2.0,-(Length_)/2.0, 0.0 ),),))
p.Surface(side1Faces=surf_contact_1_A, name='C-Surf-Conv-Radiation')

# Top and bottom surfaces for composite
p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
p.Surface(side1Faces=surf_contact_1_A[0], name='C-Surf-Top')

p = mdb.models['Model-1'].parts['Airborne-Composite']
s = p.faces
p.Surface(side1Faces=surf_contact_1_A[1], name='C-Surf-Bottom')

# Creation of Step with Damping

mdb.models['Model-1'].CoupledTempDisplacementStep
(adaptiveDampingRatio=DampingFactor,
    continueDampingFactors=True, deltmx=100.0,
        initialInc=1e-08, maxInc=time,
    minInc=1e-015, name='Heat_1', nlgeom=ON, previous='Initial',
    stabilizationMethod=DAMPING_FACTOR, timePeriod=time)
```

```
## Surface Contact

# Contact Control
mdb.models['Model-1'].StdContactControl(dampFactor=ContactControl_Damp,
 name='ContCtrl-1',
    stabilizeChoice=AUTOMATIC)

# Assign Contact Properties
# Between Composite and Teflon
mdb.models['Model-1'].ContactProperty('Comp-Teflon')
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
TangentialBehavior(
    dependencies=0, directionality=ISOTROPIC,
        elasticSlipStiffness=None,
    formulation=PENALTY, fraction=0.005, maximumElasticSlip=FRACTION,
    pressureDependency=OFF, shearStressLimit=None,
        slipRateDependency=OFF,
    table=((0.4, ), ), temperatureDependency=OFF)
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
NormalBehavior(
    allowSeparation=ON, constraintEnforcementMethod=DEFAULT,
    pressureOverclosure=HARD)
mdb.models['Model-1'].interactionProperties['Comp-Teflon'].
ThermalConductance(
    clearanceDepTable=((Thermal_Conductance_Comp_Teflon, 0.0),
        (0.0, 0.001)),
        clearanceDependency=ON,
    definition=TABULAR, dependenciesC=0, massFlowRateDependencyC=OFF,
    pressureDependency=OFF, temperatureDependencyC=OFF)

# Between Composite and Top Plate
mdb.models['Model-1'].ContactProperty('Comp-TP')
mdb.models['Model-1'].interactionProperties['Comp-TP'].
TangentialBehavior(
    formulation=FRICTIONLESS)
mdb.models['Model-1'].interactionProperties['Comp-TP'].
NormalBehavior(
    allowSeparation=ON, constraintEnforcementMethod=DEFAULT,
    pressureOverclosure=HARD)
mdb.models['Model-1'].interactionProperties['Comp-TP'].
ThermalConductance(
    clearanceDepTable=((Thermal_Conductance_Plates, 0.0),
        (0.0, 0.001)),
        clearanceDependency=ON,
    definition=TABULAR, dependenciesC=0, massFlowRateDependencyC=OFF,
    pressureDependency=OFF, temperatureDependencyC=OFF)
```

```
# Between Teflon and Bottom Plate
mdb.models['Model-1'].ContactProperty('Teflon-BP')
mdb.models['Model-1'].interactionProperties['Teflon-BP'].
TangentialBehavior(
    dependencies=0, directionality=ISOTROPIC,
        elasticSlipStiffness=None,
    formulation=PENALTY, fraction=0.005, maximumElasticSlip=FRACTION,
    pressureDependency=OFF, shearStressLimit=None,
        slipRateDependency=OFF,
    table=((0.04, ), ), temperatureDependency=OFF)
mdb.models['Model-1'].interactionProperties['Teflon-BP'].
NormalBehavior(
    allowSeparation=ON, constraintEnforcementMethod=DEFAULT,
    pressureOverclosure=HARD)
mdb.models['Model-1'].interactionProperties['Teflon-BP'].
ThermalConductance(
    clearanceDepTable=((Thermal_Conductance_Plates, 0.0),
        (0.0, 0.001)),
        clearanceDependency=ON,
    definition=TABULAR, dependenciesC=0, massFlowRateDependencyC=OFF,
    pressureDependency=OFF, temperatureDependencyC=OFF)

# Assign interaction between surfaces, Teflon-BP and Comp-TP
mdb.models['Model-1'].SurfaceToSurfaceContactStd(
adjustMethod=OVERCLOSED,
    clearanceRegion=None, createStepName='Initial', datumAxis=None,
    initialClearance=OMIT, interactionProperty='Comp-TP', master=
    mdb.models['Model-1'].rootAssembly.surfaces['TopPlate_Surf'],
        name=
    'Comp-TP', slave=
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
        surfaces['C-Surf-Top']
    , sliding=FINITE, surfaceSmoothing=AUTOMATIC, thickness=ON,
        tied=OFF)
mdb.models['Model-1'].interactions['Comp-TP'].setValuesInStep(
contactControls=
    'ContCtrl-1', stepName='Heat_1')


mdb.models['Model-1'].SurfaceToSurfaceContactStd(
adjustMethod=OVERCLOSED,
    clearanceRegion=None, createStepName='Initial', datumAxis=None,
    initialClearance=OMIT, interactionProperty='Teflon-BP', master=
    mdb.models['Model-1'].rootAssembly.surfaces['BottomPlate_Surf'],
        name=
    'Teflon-BP', slave=
    mdb.models['Model-1'].rootAssembly.surfaces['Teflon-Surf-Bottom'],
```

```
            sliding=
        FINITE, surfaceSmoothing=AUTOMATIC, thickness=ON, tied=OFF)
mdb.models['Model-1'].interactions['Teflon-BP'].setValuesInStep(
        contactControls='ContCtrl-1', stepName='Heat_1')

# Tie Composite with Teflon
mdb.models['Model-1'].Tie(adjust=ON, master=
        mdb.models['Model-1'].rootAssembly.surfaces['Teflon-Surf-Top'],
            name='Tie',
        positionToleranceMethod=COMPUTED, slave=
        mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'].
            surfaces['C-Surf-Bottom']
        , thickness=ON, tieRotations=ON)

# Loads and Boundaries, Predifinied and Amplitude for Temperature Loads

mdb.models['Model-1'].SmoothStepAmplitude(data=((0.0, 0.0), (1.0, 0.2),
  (2.0,
        0.4), (5.0, 0.6), (10.0, 0.9), (12.0, 1.0)), name='Amp-1',
            timeSpan=STEP)

mdb.models['Model-1'].EncastreBC(createStepName='Initial',
  localCsys=None,
        name='Bottom_Encastre', region=
        mdb.models['Model-1'].rootAssembly.sets['BottomPlate_RP'])

mdb.models['Model-1'].TemperatureBC(amplitude='Amp-1',
  createStepName='Heat_1',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=Cycle1_Heat, name=
        'Bottom_Temp', region=
        mdb.models['Model-1'].rootAssembly.sets['Teflon-Set-Bottom'])

mdb.models['Model-1'].TemperatureBC(amplitude='Amp-1',
  createStepName='Heat_1',
        distributionType=UNIFORM, fieldName='', fixed=OFF,
            magnitude=Cycle1_Heat, name=
        'Top_Temp', region=
        mdb.models['Model-1'].rootAssembly.sets['Comp-Set-Top'])

mdb.models['Model-1'].Temperature(createStepName='Initial',
        crossSectionDistribution=CONSTANT_THROUGH_THICKNESS,
            distributionType=
        UNIFORM, magnitudes=(Predifinied_Heat, ),
            name='Temp_Uniform_Composite',
            region=
        mdb.models['Model-1'].rootAssembly.
```

```
        instances['Airborne-Composite-1'].sets['C-Set-InitialTemp'])

mdb.models['Model-1'].Temperature(createStepName='Initial',
    crossSectionDistribution=CONSTANT_THROUGH_THICKNESS,
        distributionType=
    UNIFORM, magnitudes=(Predifinied_Heat, ), name='Temp␣Uniform␣Teflon',
        region=
    mdb.models['Model-1'].rootAssembly.sets['Teflon_Set'])

mdb.models['Model-1'].ConcentratedForce(cf3=Top_Force,
 createStepName='Heat_1'
    , distributionType=UNIFORM, field='', localCsys=None,
        name='Top_Force',
    region=mdb.models['Model-1'].rootAssembly.sets['TopPlate_RP'])

# Mesh Generation
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.seedPart(size=0.01, deviationFactor=0.1, minSizeFactor=0.1)
p = mdb.models['Model-1'].parts['Airborne-Composite']
p.generateMesh()

t = mdb.models['Model-1'].parts['Teflon']
t.seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=0.011)
mdb.models['Model-1'].parts['Teflon'].generateMesh()

# Elements Assign
# Composite
elemType1 = mesh.ElemType(elemCode=C3D8T, elemLibrary=STANDARD,
    secondOrderAccuracy=OFF, distortionControl=DEFAULT)
elemType2 = mesh.ElemType(elemCode=C3D6T, elemLibrary=STANDARD)
elemType3 = mesh.ElemType(elemCode=C3D4T, elemLibrary=STANDARD)
p = mdb.models['Model-1'].parts['Airborne-Composite']
c = p.cells
cells = lamina_Composite + lamina_Cohesive
pickedRegions =(cells, )
p.setElementType(regions=pickedRegions, elemTypes=(elemType1,
 elemType2,
    elemType3))


# Teflon
mdb.models['Model-1'].parts['Teflon'].setElementType(elemTypes=
(ElemType(
    elemCode=C3D8T, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
    distortionControl=DEFAULT), ElemType(elemCode=C3D6T, elemLibrary=
        STANDARD),
    ElemType(elemCode=C3D4T, elemLibrary=STANDARD)), regions=(
```

```
    mdb.models['Model-1'].parts['Teflon'].cells.findAt(((-Sides/6.0,
        Sides/6.0, 0.0),
    )), ))
```

B.2. WARPAGE SIMULATION

```
## Python script for the warpage. Here, only the step, Predifined
#Stress Field and Boundary conditions are provided.  The structure
#can be simulated from the code of the Cycles removing the Plates.

# Assign Static Step
mdb.models['Model-1'].StaticStep(adaptiveDampingRatio=0.05,
    continueDampingFactors=False, initialInc=1e-10, maxInc=40.0,
        minInc=1e-15,
    name='Springback', nlgeom=ON, previous='Initial',
        stabilizationMagnitude=
    0.0002, stabilizationMethod=DISSIPATED_ENERGY_FRACTION,
        timePeriod=40.0)

# Predifined Stress Field -- In the "filename, the name of the
# file that produces the desirable profile must be used (here is one
# my own files)
#
mdb.models['Model-1'].Stress(distributionType=FROM_FILE, fileName=
    '/home/harris/temp/14June/Heat1_Test2_NP.odb', increment=45,
     name= 'Stress Field', step=1)

# Boundary Conditions, Fix Composite and Teflon Nodes
mdb.models['Model-1'].EncastreBC(createStepName='Initial',
 localCsys=None,
    name='Comp Encastre', region=
    mdb.models['Model-1'].rootAssembly.sets['Comp_Set_Enc'])

mdb.models['Model-1'].EncastreBC(createStepName='Initial',
 localCsys=None,
    name='Teflon Encastre', region=
    mdb.models['Model-1'].rootAssembly.sets['Teflon_Set_Encastre'])
```

B.3. DISCRETE RIGID PLATES

Another type of rigid plates is the Discrete rigid plates. The difference between those two is that discrete plates can be meshed as a real deformable part, but they have properties of a rigid plate. The user can try with both plates in order to compare the outcome, as discrete plates manipulate the contact regions better.

```
# Discrete Plate
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize=1.0)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=
```

```
(Plate/2.0, Plate/2.0),
    point2=(-Plate/2.0, -Plate/2.0))
mdb.models['Model-1'].sketches['__profile__'].ObliqueDimension(textPoint=
(
    -Plate/6.0, Plate/75.0), value=Plate, vertex1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-Plate/2.0, Plate/2.0),
    ), vertex2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -Plate/2.0, -Plate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].ObliqueDimension(textPoint=
(
    -Plate/15.0, -Plate/6.0), value=0.6, vertex1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-Plate/2.0,
    -Plate/2.0), ), vertex2=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (Plate/2.0,
    -Plate/2.0), ))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='Top_Plate',
 type=
    DISCRETE_RIGID_SURFACE)
mdb.models['Model-1'].parts['Top_Plate'].BaseSolidExtrude(depth=
PlatesDepth, sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Partition
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=Grid,
 name='__profile__',
    sheetSize=sheet_Size, transform=
    mdb.models['Model-1'].parts['Top_Plate'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['Top_Plate'].faces.findAt(
        (Plate/6.0,
    Plate/6.0, PlatesDepth), ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Top_Plate'].edges.findAt(
        (-Plate/2.0,
    -Plate/4.0, PlatesDepth), ), sketchOrientation=RIGHT, origin=(0.0,
        0.0, PlatesDepth)))
mdb.models['Model-1'].parts['Top_Plate'].projectReferencesOntoSketch(
filter=
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['__profile__'])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
 Plate/2.0), point2=(
    0.0, -Plate/2.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
addUndoState=
```

```
    False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((0.0,
        0.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
        (-Plate/2.22, Plate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    0.0, 0.0), ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt((0.0,
        Plate/2.0), )
    , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    -Plate/2.22, Plate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (-Plate/2.0, Plate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    Plate/2.0, Plate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, Plate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, -Plate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
        findAt((
    Plate/2.22, -Plate/2.0), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (Plate/2.0, -Plate/2.0),
    ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
        findAt((
    -Plate/2.0, -Plate/2.0), ), midpoint=
    mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
        (0.0, -Plate/2.0),
    ))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-Plate/2.0,
  0.0), point2=(
```

```
     Plate/2.0, 0.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
     addUndoState=False, entity=
     mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
          (-Plate/2.22, 0.0),
     ))
mdb.models['Model-1'].sketches['__profile__'].PerpendicularConstraint(
     addUndoState=False, entity1=
     mdb.models['Model-1'].sketches['__profile__'].geometry.findAt((
          -Plate/2.0,
     -Plate/2.22), ), entity2=
     mdb.models['Model-1'].sketches['__profile__'].geometry.findAt(
          (-Plate/2.22, 0.0),
     ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
     addUndoState=False, entity1=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (-Plate/2.0, 0.0),
     ), entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
          findAt((
     -Plate/2.0, -Plate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
     addUndoState=False, entity1=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (-Plate/2.0, -Plate/2.0),
     ), entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
          findAt((
     -Plate/2.0, Plate/2.0), ), midpoint=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (-Plate/2.0, 0.0),
     ))
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint(
     addUndoState=False, entity1=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (Plate/2.0, 0.0), )
     , entity2=mdb.models['Model-1'].sketches['__profile__'].geometry.
          findAt((
     Plate/2.0, Plate/2.22), ))
mdb.models['Model-1'].sketches['__profile__'].EqualDistanceConstraint(
     addUndoState=False, entity1=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (Plate/2.0, Plate/2.0), )
     , entity2=mdb.models['Model-1'].sketches['__profile__'].vertices.
          findAt((
     Plate/2.0, -Plate/2.0), ), midpoint=
     mdb.models['Model-1'].sketches['__profile__'].vertices.findAt(
          (Plate/2.0, 0.0),
```

```
    ))
mdb.models['Model-1'].parts['Top␣Plate'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Top␣Plate'].faces.findAt(((Plate/6.0,
        Plate/6.0, PlatesDepth),
    )), sketch=mdb.models['Model-1'].sketches['__profile__'],
        sketchUpEdge=
    mdb.models['Model-1'].parts['Top␣Plate'].edges.findAt(
        (-Plate/2.0, -Plate/4.0,
    PlatesDepth), ))
del mdb.models['Model-1'].sketches['__profile__']
```

B.4. VISCOELASTIC SCRIPTS

In this section, the basic extra script for merging the isotropic viscoelastic and the orthotropic elastic parts will be provided. These scripts are for Chapter 5. The rest of the scripts are the same from Chapter 4.

```
# Viscoelastic Isotropic
# Sketch and Part
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
 sheetSize= sheet_Size)
mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=
( Width_,Length_),
    point2=( -Width_,-Length_))
mdb.models['Model-1'].Part(dimensionality=THREE_D,
 name='Airborne-Composite-Viscoelastic', type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Airborne-Composite-Viscoelastic'].
BaseSolidExtrude(depth=Thickness_, sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']

# Merging
mdb.models['Model-1'].rootAssembly._previewMergeMeshes(instances=(
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'],
    mdb.models['Model-1'].rootAssembly.instances
        ['Airborne-Composite-Viscoelastic-1'])
    , mergeBoundaryOnly=False, nodeMergingTolerance=1e-06)
mdb.models['Model-1'].rootAssembly.InstanceFromBooleanMerge(domain=MESH,
    instances=(
    mdb.models['Model-1'].rootAssembly.instances['Airborne-Composite-1'],
    mdb.models['Model-1'].rootAssembly.
        instances['Airborne-Composite-Viscoelastic-1'])
    , mergeNodes=ALL, name='Part-1', nodeMergingTolerance=1e-06,
    originalInstances=DELETE)


# Viscoelastic Properties
```

```
mdb.models['Model-1'].Material(name='Viscoelastic')
mdb.models['Model-1'].materials['Viscoelastic'].Density(table=((1200.0,
  ), ))
mdb.models['Model-1'].materials['Viscoelastic'].Elastic(
        type=ISOTROPIC, table=((
        2250000000.0, 0.3, 20.0), (2200000000.0, 0.3, 70.0), (2000000000.0,
            0.3,
        120.0), (1700000000.0, 0.3, 170.0)), temperatureDependency=ON)
mdb.models['Model-1'].materials['Viscoelastic'].Viscoelastic(domain=TIME,
        table=(), time=RELAXATION_TEST_DATA)
mdb.models['Model-1'].materials['Viscoelastic'].viscoelastic.
 ShearTestData(
        shrinf=0.45, table=((1, 0.001), (0.9, 50.0), (0.72, 100.0), (0.55,
            200.0), (0.45,
        300.0)))
```

C

Appendix C

In this Appendix, the basic knowledge concerning the first steps of obtaining data and also how to do a sensitivity analysis will be illustrated. No solutions or results are given, but only the basic steps for future work, as due to limited time frame, the author couldn't retrieve any result. Also, due to the big amount of coding (many Matlab and Python scripts), here, only the main concept and some basic coding will be given.
For Data retrieveing, a Matlab code was created. The work is about changing important input parameters in the model in order to observe how it behaves in terms of output results. In our case, how the parameters of the system affect the warpage is the main topic. There are several parameters from material properties to system parameters. To mention a few:

1. Laminate's thermal and mechanical properties (such as thermal expansion and Young's Modulus).
2. Laminate's imperfections and fiber misalignment.
3. Teflon's thermal and mechanical properties.
4. Friction coefficients between the plates, Teflon and Laminate.
5. Pressure and Temperature configurations of the system.

Also, a combination of these properties may have different outcome. So, in order to have a full observation, a Design of Experiments must be implemented. What this does is to create a dimensional space of combination of parameters that the user chooses to analyze. The user must provide the Upper and the Lower bound which is the range of values for analysis. After the DOE (Design Of Experiments) is created, simulations using each pair of parameters must take place. Every pair will correspond to a specific output, which will be extracted using post-processing techniques concerning the software that the user uses.

After the data is extracted, many data-analysis methods can be implemented such as sensitivity analysis which show how sensitive the output is concerning a specific parameter or a pair of those. Another method of analysis is the well-known machine learning, which will show how the warpage is affected by the change of the most important parameters, mapping the inputs with the output.

To give an example, let us assume that the desirable parameters to be analyzed are the applied force from the top plate and the fiber misalignment of the first layer of the composite. Two ways can be used.
Through the Matlab code:

```
UserSettings.DoE_file_name = 'DOE'; % Name of the file
UserSettings.DoE_size       = 1000; % Number of DoE points
% Input design variables names
UserSettings.DoE_vars       = {'Force' , 'FiberMisal_layer'};
UserSettings.DoE_LowerBounds = [100e3, 0];
UserSettings.DoE_UpperBounds = [500e3, 90];
```

The 2^{nd} way is by using a Python library called Salib which can be used either through Python compiler or from a terminal/command line [72] (the example here is through Python):

```
# Input Parameters
parameter_dict = {'num_vars':2    # Number of Input variables
  'names':['Force','FiberMisal_layer'] # The parameters
  'bounds':[[100e3,500e3],      # Upper and Lower bounds
  [0,90]]}

# Design of Experiments using Morris Method
model_input = SALib.sample.morris.sample(parameter_dict,10)
```

The next step is to run the simulations. Using the Matlab code, 6 smaller codes (except of the main one which run all the smaller codes).

1. Structure design and mesh of the Cycles 1 and 2 models (Python into Matlab)
2. Two input file codes which are the loads and boundary conditions for the two Cycles (Input file into Matlab)
3. Structure design and mesh of the Warpage model (Python into Matlab)
4. Input file that runs the Warpage model (Input file into Matlab)
5. Post-processing for extracting the node displacement U3

The input files (.inp) mentioned is a different way that Abaqus uses to run it's simulations. In this case, Matlab provides ways to script different languages such as Python and also Input files.
After the output displacements are obtained, sensitivity analysis can be done using the input and the outputs:

```
# This step is just for illustration that to get the output,
# important is to run the simulations
```

```
model_output = run_simulation(model_input)

# The Python code below analyze the model using the input parameters
# and the output warpage to calculate how sensitive each parameter is.
Si = morris.analyze(problem,model_input,model_output)
```

Concerning sensitivity analysis, documentation is provided through articles, through Python developers and also online video seminars for better understanding [73].