Computational efficient robustness analysis of aircraft component distortion

accounting for stochastic pre-stressed stock material in reductive manufacturing processes

By Y.C.E. Janssens

Department of Precision and Microsystems Engineering

Report no	: 2019.005
Coach	: D.P. Munro
Professor	: Prof. dr. ir. A. van Keulen
Specialisation	: Structural Optimization and Mechanics
Type of report	: Thesis
Date	: 08-02-2019

TU Delft

AIRBUS

Challenge the future

Computational efficient robustness analysis of aircraft component distortion

accounting for stochastic pre-stressed stock material in reductive manufacturing processes

by

Y.C.E. Janssens

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday February 8, 2019 at 4:00 PM

Student number:	4104897	
Project duration:	March 1, 2017 – February 8, 2019	
Thesis committee:	Prof. dr. ir. A. van Keulen,	TU Delft, supervisor
	dr.ir. M. Tichem,	TU Delft
	S. van der Veen,	AIRBUS Operations S.A.S.
	D.P. Munro,	AIRBUS Operations S.A.S.

This thesis is confidential and cannot be made public until ...., ...

TU Delft

Abstract

In this thesis, a methodology is developed that allows for both reliable and computational efficient robustness analysis of aircraft component distortion. This research applies to distortion caused by reductive manufacturing processes that are used to obtain monolithic components from pre-stressed stock material. With the developed methodology, orientations of any component within rolled plate stock material can be found where distortion is most robust. Part distortion is defined as a deviation in shape of an aircraft component from original intent as a result of the component's reductive manufacturing process. As extreme precision is required in aircraft component assembly, the distortion phenomenon is highly undesired. The developed methodology in this thesis contributes to AIRBUS' objectives to minimize part distortion related issues.

In this thesis, the fundamentals of part distortion are studied. It is found that aircraft components distort as a result of residual stresses that are present in stock material from which the components are manufactured. As residual stress in rolled plate is subjected to substantial variation, part distortion is stochastic in nature. Positions of the components in rolled plate are searched for where distortion is most robust. The robustness of distortion refers to the insensitivity of distortion to uncertainty in residual stress. A mathematical stochastic representation of residual stress in rolled plate is developed showing high coherence with experimental measurement data provided by AIRBUS. For elementary geometries, the relationship between distortion robustness and residual stress is derived analytically.

The developed method for predicting distortion robustness is more than one hundred times more efficient in terms of computation cost compared to state-of-the-art methods and allows for reliable robustness predictions. In the developed method, three-dimensional positioning of a component in rolled plate can be simulated where state-of-the-art distortion modeling tools usually stick to one dimension.

The developed methodology is put to the test in a case study concerning an aircraft stiffener component. The case study emphasizes the significance of robustness predictions; distortion dispersion is found to be relatively large compared to the distortion magnitude and significant correlation is found between the component's orientation in rolled plate and the level of robustness. Positions of components in rolled plate can be found where distortion is extremely robust. Moreover, a relationship is found between the component's degree of symmetry and the level of robustness.

Contents

1 Introduction **4**

2 Residual stress in rolled plate and its origin **8**
2.1 Residual stress due to primary- and secondary machining processes 9
2.2 A stochastic description of experimental residual stress data 11

3 The fundamentals of part distortion **18**
3.1 Self-equilibrated stress 18
3.2 Homogeneity of stress in rolled plate 19
3.3 Machining considered as an one-to-one mapping of stress 21
3.4 Analytical derivation of distortion 23
3.5 Example: T-section 28
3.6 Distortion minimization based on analytical equations 30
3.7 Robust optimization based on analytical equations 31
3.8 Till what point can analytical equations be used? 33

4 A computational efficient method for distortion robustness analyses **35**
4.1 Relation between residual stress and displacements in Finite Element Analyses 36
4.2 The residual stress matrix $\boldsymbol{\sigma}^{\dagger}$ as a function of translation (z_1) and rotation (α, β, γ) of the residual stress field 37
4.3 State-of-the-art methods for robustness analyses 43
4.4 Improved method for robustness analyses 47

5 Case study **52**
5.1 Translation z_1 of the stiffener component in z-direction 53
5.2 Rotation β of the stiffener component about the y-axis 58
5.3 Optimal position of stiffener component in piece of rolled plate in terms of offset z_1 and angle β 61
5.4 Physical explanation for the presence of robustness troughs where distortion is extremely robust 68

6 Conclusions **70**

7 Recommendations for future research **73**

Bibliography **75**

APPENDICES **78**

A Analytical distortion derivations **79**
A.1 Analytical distortion derivation of a beam cut in half 79
A.2 Analytical distortion derivation of a beam machined from both sides 81
A.3 Analytical distortion derivation of a tee section 83

B Distortion and robustness optimization based on analytical equations **87**
B.1 Distortion optimization of a Tee Section for offset z_1 87
B.2 Distortion optimization of a tee section for offset z_1 and flank thickness t 90
B.3 Robustness optimization of a tee section for offset z_1 and flank thickness t . . . 92

C Finite element simulations concerning an elementary beam with rectangular cross section **94**
C.1 Deterministic static analysis 94
C.2 State-of-the-art method 103
C.3 Improved method 110

D Case study **128**
D.1 Distortion curves for translation z_1 in z-direction 128
D.2 Distortion curves for rotation β about the y-axis 130
D.3 The improved method applied to the case study 134

Chapter 1

Introduction

Environmental challenges, a rise in the crude oil price and an ever increasing competitive market have demanded the aerospace industry as well as the automotive industry for more fuel-efficient design and manufacture. The demand for fuel-efficient and lightweight aircraft has fueled the use of high-strength alloys to enable design of thinner web and wall features in structural components. It is estimated that usage of topology optimization on the AIRBUS' A350 aircraft wing ribs demonstrated in Fig. 1.1 resulted in 1000 kg of weight savings per aircraft [1]. The downside is that these thin and lightweight designs come with higher inherent residual stresses and have, due to their thin and lightweight design, low stiffness to prevent part distortion from happening [2–4].

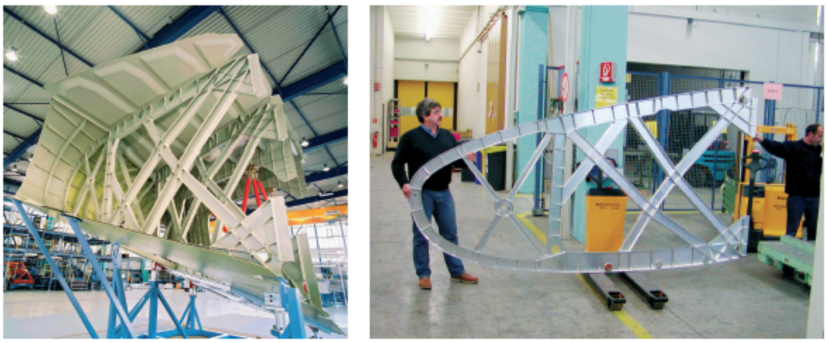


Figure 1.1: An optimized wing rib section (right) using topology optimization that fits into the wing box (left) belonging to the AIRBUS A350 aircraft [1].

Part distortion is a common phenomenon in the product manufacturing life cycle and is costing billions of loss in profit each year. Part distortion is defined as the deviation of shape from original intent as a result of the component's manufacturing process after it is released from fixture [5]. This research concerns only components that are manufactured with reductive manufacturing techniques, like milling, drilling and cutting. In Fig. 1.2, distortion as a result of a cutting reductive manufacturing process is demonstrated [6]. It is estimated that heat treatment distortion in German machine tool, automotive and transmission industry alone, is costing an economic loss of €850M [7]. Similarly, a study by Boeing, based on four aircraft programs, estimated that rework and scrap costs related to part distortion comes to in excess of 290 million dollars [8]. Due to the large impact of distortion, in 2005, a €5.4M project named COMPACT was launched by the European Commission. The aim

was to advance the understanding of residual stress and provide solutions to reduce and eliminate part distortion in aluminum alloys for the aerospace industry.

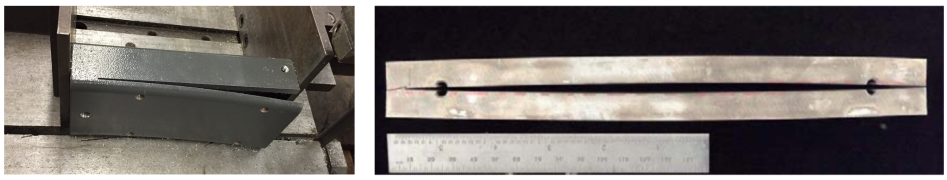


Figure 1.2: Two examples of distortion as a result of a cutting reductive manufacturing process [6].

In current industry practice, managing part distortion during or after component manufacture is not properly understood. Often designs are manufactured without any simulation tool in advance, leading to long and expensive machining iterations on components [4]. In a further attempt to minimize distortion, material specifications or even suppliers are changed. As a last resort, additional correction processes such as shot peening are applied.

Amongst others, COMPACT has advanced the state of the art in terms of understanding residual stress and part distortion. Numerical modeling tools have been developed to predict distortion. First attempts have been made to minimize distortion by changing the orientation of the component in stock material [5], like is shown in Fig. 1.3.

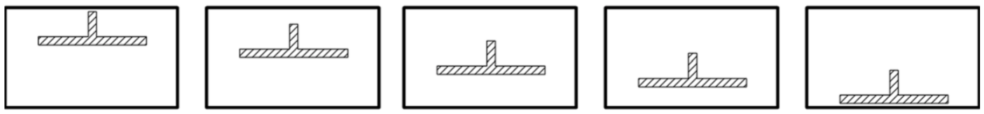


Figure 1.3: A cross sectional view of a piece of stock material is shown from which a typical tee section component is machined. The figure shows five different offsets for the tee section within a piece of stock material at which part distortion is evaluated [5].

Furthermore, components with various geometrical features have been studied to understand the influence of *design* on residual stress distribution and part distortion [9]. An example where different designs for a so-called H-plate are evaluated for distortion [10] is demonstrated in Fig. 1.4.

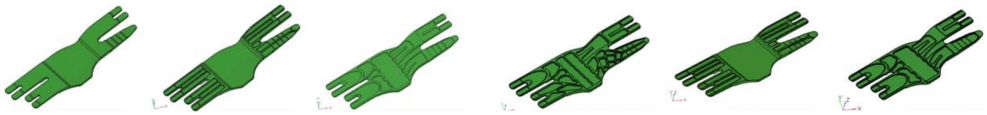


Figure 1.4: The effect of variable design features on distortion is evaluated for a *H-plate* component. Six different designs were considered which vary in thickness and in the type of stiffeners used [10].

In addition to the work of COMPACT, initiatives have been taken by AIRBUS/EADS to mature the technology for industrial exploitation. Industrial workflows were developed by AIRBUS that empowers manufacturing engineers at shop floor level to find the optimal positioning of components in stock material for minimal distortion [5].

The current state of the art falls short on dealing with the *stochastic* nature of part distortion. As will be elaborated upon later in this thesis, part distortion happens as a result of the presence of *residual stress* in stock material. Residual stress in stock material is stochastic in nature due to the uniqueness of the metallurgical and mechanical history of each individual rolled plate production and due to different plate suppliers [6]. Residual stresses are different for each rolled plate production since it is impossible to create equal conditions during each production [6]. Since residual stress is stochastic in nature, part distortion as a result of residual stress is stochastic in nature as well. In literature, the term *robustness* is used for expressing the amount of variation (or dispersion) in distortion. Part distortion is classified as robust when it is has low sensitivity to variation in rolled plate residual stress. Experimental research has shown that robustness analyses are highly significant. A case study performed by AIRBUS concerning a side cockpit window frame demonstrated that a 10% variation in residual stress predicts a variation as much as 50% in part distortion [11]. In order to save as much weight as possible and to cut material costs, aircraft components tend to have highly complex design features (see Fig. 1.5). These complex design features tend to make aircraft components highly sensitive to variation in residual stress. In other words, a component manufactured from a given plate or batch of material might result in distortion that is significantly different then distortion for another batch of material, or for another material supplier.

Figure 1.5: Three different structural aircraft components, i.e. complex wing rib structures and landing gear components, are shown that are machined from titanium alloy rolled plate [12].

Due to its high significance, efforts have been done to include robustness analyses in distortion modeling tools [10]. Methodologies were developed in which next to part distortion also robustness can be evaluated. This way, positions of components in rolled plate can be found where distortion is acceptable and has low sensitivity to variation in residual stresses. In current state-of-the-art research, however, efforts of propagating the residual stress uncertainty to distortion have proven to drastically worsen the computational efficiency of the modeling tools. Brute-force *Monte Carlo* stochastic analyses are employed that are often very costly in computational effort and commercial license requirements since a great number of numerical simulations is required to obtain reliable robustness results [11].

Up to now, state-of-the-art methodologies have failed to be of any use in industry. In July 2017, I myself have visited GROUPE ROSSI AERO based in Toulouse (France), which is one of AIRBUS' subcontractors specialized in aircraft component manufacturing. During this visit, I learned that in industry no simulation tools at all are used to deal with part distortion. Long machining iterations, referred to as *multiple step machining*, are current practice to minimize part distortion. During my internship at AIRBUS from April to July 2017, I in-

vestigated the effect of multiple step machining on part distortion. I discovered that stress relaxation in between machining steps significantly diminishes part distortion [13].
The reason why state-of-the-art distortion prediction tools are not yet incorporated in industry is that they are not yet ready. Deterministic distortion tools have been developed, however are meaningless due to the large variation in part distortion as a result of the stochastic nature of rolled plate residual stress. Thereupon, robustness prediction tools have been developed, however tend to worsen the computational efficiency of the simulation tools in such a way that robustness predictions have become infeasible in terms of time and costs.

In this thesis, a methodology will be developed that can predict both part distortion as well as its robustness as a function of the part's position in stock material whilst requiring relatively little computational effort. The title for this research is therefore as follows.

Computational efficient robustness analysis of aircraft component distortion accounting for stochastic pre-stressed stock material in reductive manufacturing processes.

In Chapter 2, residual stress in rolled plate stock material and its origin will be examined. With the help of by AIRBUS' experimental stress measurements, a stochastic description for residual stress in rolled plate will be formulated.
In Chapter 3, the fundamentals of part distortion will be examined. Analytical equations relating part distortion to residual stress will be formulated. In several examples, part distortion will be derived analytically. Based upon the analytical equations derived for distortion and robustness, several optimization problems will be solved. Distortion and distortion robustness will be evaluated as a function of the component's position in stock material. This Chapter will conclude with a remark on to what extent analytical equations can be used to solve distortion related problems and why the use of Finite Element Analyses are more appropriate for solving distortion related problems for complex aircraft components.
In Chapter 4, a methodology will be formulated which allows for three-dimensional positioning of a component within rolled plate. Next, the state-of-the-art computational inefficient method for evaluating robustness will be examined. Subsequently, an improved method will be introduced that is capable of evaluating robustness in a computational efficient manner. In Chapter 5, the developed improved method will be applied to a case study concerning a characteristic aircraft stiffener component. Distortion robustness will be evaluated as a function of the position of the component in stochastic pre-stressed stock material, like is shown in Fig. 5.1. Chapter 6 will conclude with a final conclusion and in Chapter 7, recommendations for future research will be listed.

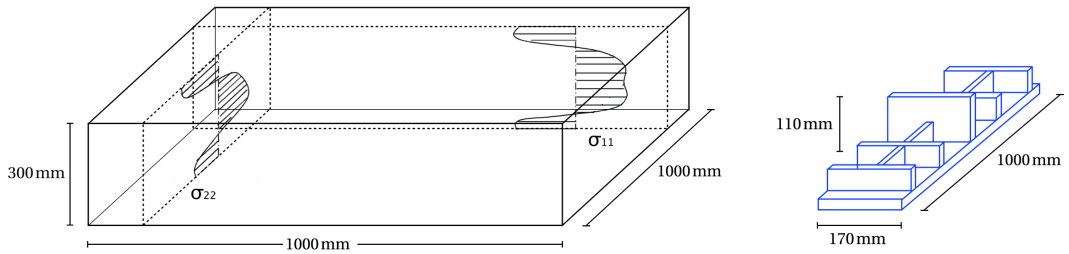


Figure 1.6: For a stiffener component (blue) distortion robustness is evaluated as a function of its position (green) in stochastic pre-stressed rolled plate stock material (black).

Chapter 2

Residual stress in rolled plate and its origin

Residual stresses are stresses locked into a rigid part in the absence of external forces or thermal gradients and are the result of metallurgical and mechanical history of each material point in the part and of the part as a whole during its manufacturing life cycle, according to its definition [14]. Residual stress originates from manufacturing processes like quenching, stretching, forging, casting, welding, machining, forming, etc., that are applied in the manufacturing chain of the component from raw material. These processes are complex combinations of heat transfer, mechanical deformation and metallurgical changes [5, 15]. A distinction is made between *primary*- and *secondary* process induced residual stress. The role of the primary- and secondary processes in the manufacturing chain of the component from raw material is illustrated in Fig. 2.1.

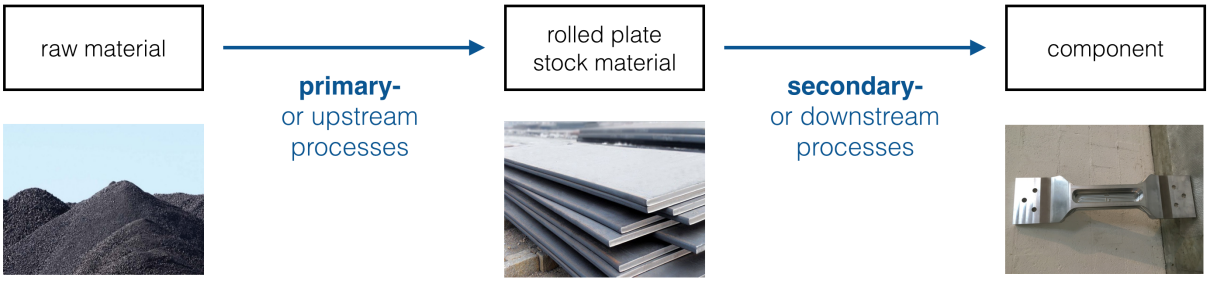


Figure 2.1: The role of primary- and secondary machining processes in the manufacturing chain of a component from raw material. Both primary- and secondary machining processes induce residual stress [16].

Primary or *upstream* processes imply machining processes used for the manufacture of stock material from raw material. Secondary or *downstream* processes imply machining processes used for the manufacture of the component from stock material. In the aerospace industry, aircraft components are commonly made from stock material in the form of rolled plate, extrusions or forgings [3] of which the production processes are illustrated in a nutshell in Fig. 2.2. In the aerospace industry, aluminum and titanium alloys are used to achieve both lightweight- and high-strength components. In this research, rolled plate material will be considered as stock material due to its wide use and its distinctive residual stress profile.

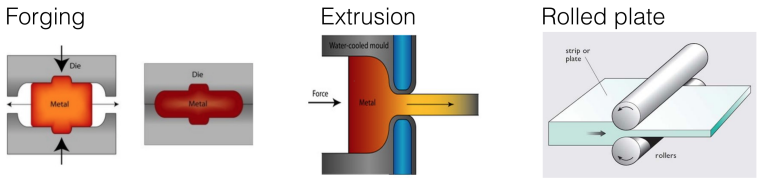


Figure 2.2: Simplified schematic representations of the manufacturing processes of three types of stock material used in the aeronautical industry, i.e. forgings (left), extrusions (middle) and rolled plate (right). From these types of stock material, aircraft components are commonly manufactured. In this research, only rolled plate will be considered [17].

As both primary- and secondary processes induce residual stress, both processes are examined.

2.1 Residual stress due to primary- and secondary machining processes

The primary production processes where rolled plate is formed from raw material consist of a number of steps. First, different alloying elements are mixed, melted, casted into an ingot and cooled. The cast ingot is then heated, hot rolled and subsequently quenched. Quenching is the most dominant stress-inducing process. In this process, the plate is rapidly cooled in cold water in order to achieve desirable physical and mechanical material properties. In Fig. 2.3, the two most dominant stress-inducting processes - being hot rolling and quenching - in the manufacturing chain of rolled plate are displayed along with the final product, rolled plate material.

Figure 2.3: The two most dominant stress-inducing processes in the primary production processes, being hot rolling (left) and quenching (middle), where rolled plate stock material (right) is formed from raw material [18].

Quenching induces undesirable high levels of residual stresses that approach even yield strength magnitude as a result of large temperature gradients due to non-uniform cooling in between intermediate layers of the plate [2, 3, 5, 19, 20]. When relatively thick parts are initially immersed in the quench bath, the surfaces cool first and thus contract more rapidly than the interior. Early in the quench, the hot interior provides little resistance to the contraction of the surfaces. The soft interior plastically deforms to accommodate surface contraction. Later in the quench, when the interior cools and wants to contract, its contraction is resisted by the now cold and relatively strong near-surface material. As a result, tensile stresses develop in the interior. The material there wants to contract, but cannot. These tensile interior stresses are balanced by compressive stresses that develop near the

surface. These represent the forces that resist contraction of the cooling interior. A symmetric through-thickness so-called *M-shaped* residual stress profile, typically found in rolled plate, develops with maximum compression on each surface and maximum tension along the centerline [6] in both rolling as transverse direction, such as is illustrated in Fig. 2.4.

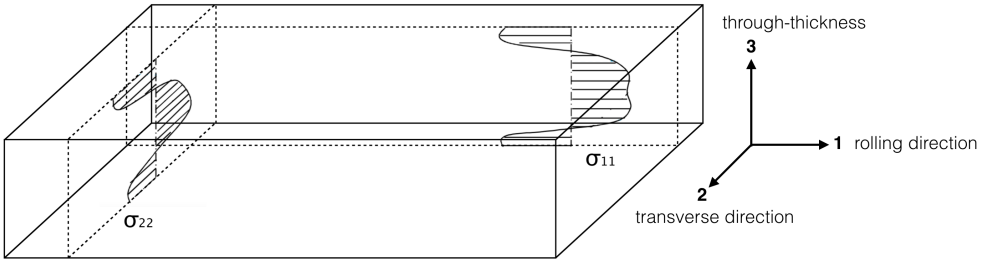


Figure 2.4: A schematic drawing of the characteristic *M-shaped* residual stress profiles present in rolled plate. These residual stress profiles are induced by the primary manufacturing processes and own their shape mainly to the quenching process. The stress profiles vary through the thickness and are homogeneous throughout rolling- and transverse direction.

After quenching, the stress peak magnitudes approach even yield strength. In order to relieve these high quench-induced residual stresses, the rolled plate is mechanically stretched at room temperature in the rolling direction to 1.5 to 3% plastic deformation [21]. In Fig. 2.5, the residual stress distributions in (7050-T74 Aluminum) rolled plate in both rolling as transverse direction are illustrated before (left) and after (right) the mechanical stretching stress-relieve process. The uniaxial stretch has relieved the stresses in both directions by a factor of 10 approximately, while maintaining the characteristic M-shaped profile.

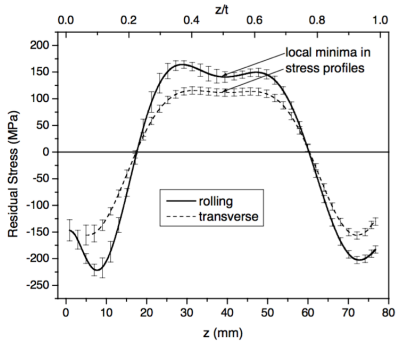


Figure 4. Residual stresses in 7050-T74 aluminum plate.

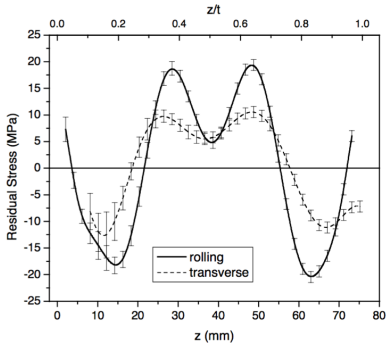


Figure 5. Residual stresses after stretch stress relief (7050-T7451).

Figure 2.5: Residual stress profiles in both rolling- and transverse direction in (7050-T74 Aluminum) rolled plate before (left) and after (right) the mechanical stretching stress-relieve process [21]. The z-coordinate in the graphs corresponds with the 3-direction, or the *through-thickness* direction, that is illustrated in Fig. 2.4.

The secondary production processes involve the reductive manufacturing techniques that are used to obtain the component from rolled plate material. These reductive manufacturing techniques are based upon the principle of removing material in order to obtain a monolithic component. In this research, the reductive manufacturing technique focused upon is milling, which is demonstrated in Fig. 2.6.

Figure 2.6: Milling is a reductive manufacturing technique based upon the principle of removing material in order to obtain a component such as the one that is illustrated (right) [22].

Reductive manufacturing techniques, such as milling, cutting and drilling, involve a combination of mechanical and thermal loading by which residual stress is induced. The induced residual stress effects the already existing residual stress induced by the primary manufacturing processes. Secondary processes tend to induce compressive residual stress in the sub-surface of the plate as it is a mechanically dominant process [3]. Fig. 2.7 shows the through-thickness sub-surface residual stress profile induced by a typical cutting process of (Aluminum 7050) rolled plate.

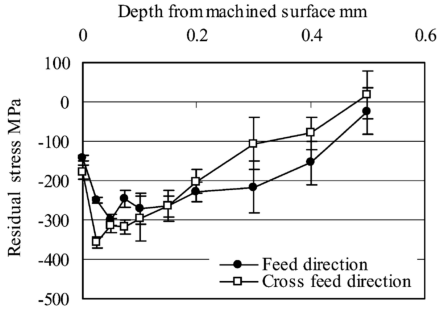


Figure 2.7: The penetration of the secondary machining induced stresses through the sub-surface of (Aluminum 7050) rolled plate [23]. The secondary machining technique concerned is cutting.

As can be seen, the induced residual stresses are in the order of magnitude of −200 MPa to −350 MPa which penetrate to a depth of 0.5 mm. The magnitude of these secondary induced residual stresses are significantly larger than the magnitude of primary induced residual stresses which are in the order of magnitude of 15 MPa to 25 MPa [23, 24]. However, secondary induced stresses only penetrate into the 0.5 mm subsurface of the geometry, hence influencing only a small portion given that the rolled plate of interest has a thickness in the range of 60-90 mm. Therefore it can be concluded that secondary induced residual stresses can be neglected given the focus of this research. Only primary induced residual stresses will be considered in this research.

2.2 A stochastic description of experimental residual stress data

AIRBUS has conducted experimental measurements in order to determine the residual stress present in a batch of twelve rolled plate samples from the type Al-Cu-Li 2050. An often used

- destructive - method for extracting residual stress from stock material is the so-called *layer removal method* [25], of which the functioning is explained in Fig. 2.8.

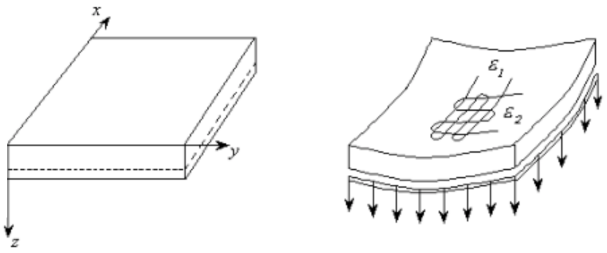


Figure 2.8: A schematic drawing of the functioning of the layer removal method. Strain gauges are fixed onto the rolled plate stock material. Layers of material are removed and strain is measured. Based on the strain measurements, the original residual stress profile is deduced [25].

As was demonstrated in Fig. 2.5, residual stress in rolled plate stock material is present in both rolling as in transverse direction. In Fig. 2.9, the experimental residual stress data in both rolling as transverse direction of the batch of twelve rolled plate samples is plotted against ζ. ζ is defined as $\zeta = \frac{z}{T}$ with T being the thickness of the rolled plate and z being the coordinate in through-thickness direction. In this thesis, residual stress will be denoted with the symbol $\sigma^{\dagger}$, with stress in the rolling direction as $\sigma^{\dagger}_{11}$ and stress in the transverse direction as $\sigma^{\dagger}_{22}$.

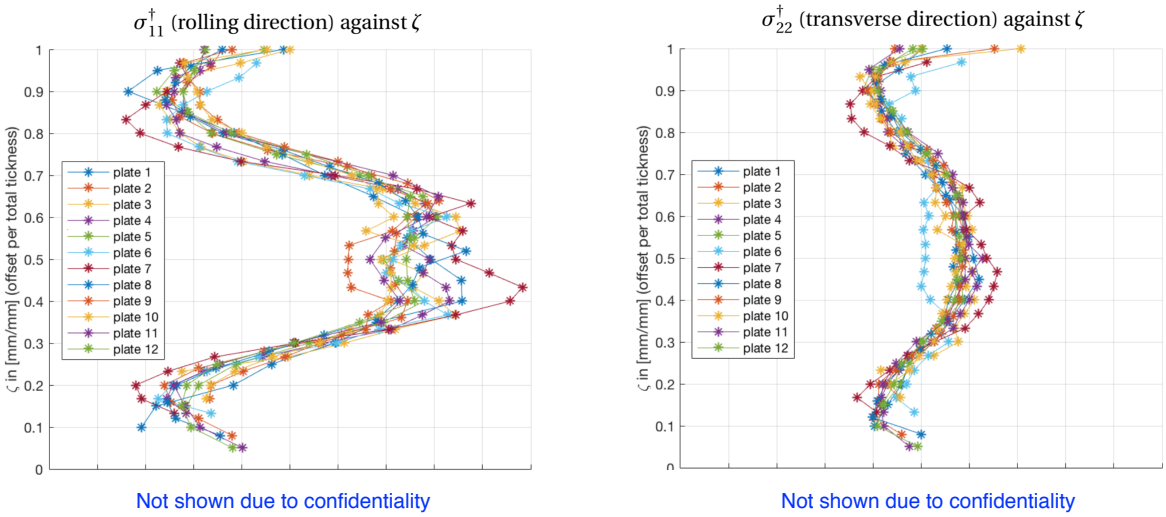


Figure 2.9: Experimental residual stress data provided by AIRBUS of a batch containing twelve rolled plate samples. In both rolling (left) and transverse direction (right), stress is plotted against ζ, where $\zeta = \frac{z}{T}$. The z-direction corresponds with the 3-direction or through-thickness direction which is illustrated in Fig. 2.4 along with the rolling- and transverse direction.

Fig. 2.9 demonstrates that the residual stress in each plate generally follows the same M-shaped pattern of compressive stress near the surfaces and tensile stress in the core. It also becomes evident that large variation is present between different rolled plate samples. It is therefore inevitable to include stochastics in the mathematical description for residual

stress. In this Chapter, a stochastic description of residual stress is found by curve-fitting each rolled plate sample in both rolling as transverse direction to a chosen fit function and, subsequently, by defining a comprehensive stochastic fit function based upon the twelve curve-fitted fit functions. Since the obtained stochastic fit function will be based on only twelve samples, it will not describe stochastics of rolled plate in general very accurately, however it will provide certain insight.

First a fit function is chosen. The definition of residual stress, as will be elaborated upon further in Chapter 3, implies that the residual stress profile in rolled plate must satisfy both force equilibrium

$$\int_A \sigma^\dagger(z)\, dA = 0 \tag{2.1}$$

and moment equilibrium

$$\int_A \sigma^\dagger(z) z\, dA = 0. \tag{2.2}$$

A curve fit function must be chosen satisfying both conditions stated in Eq. 2.1 and Eq. 2.2. The stress data shown in Fig. 2.9 appears to resemble the shape of sinusoidal functions. In addition, a summation of sinusoidal functions with integer amounts of periods in ζ, such as

$$\sigma^\dagger(z) = \sum_i^n c_i^\dagger \cos(i2\pi\zeta(z)), \tag{2.3}$$

will satisfy both conditions for force- and moment equilibrium. In the equation above, n represents the total number of individual cosine modes i. $c_i^\dagger$, the amplitude of cosine mode i, is referred to as a fit function coefficient. In literature [5, 11, 21], summations of cosine modes are frequently used to describe residual stress in rolled plate. Similarly will be done in this research.

For each rolled plate sample $k = 1 \ldots l$ with $l = 12$, the residual stress data is curve-fitted to the above stated fit function. For curve-fitting, Matlab build-in Nonlinear Least-Squares Solver *lsqcurvefit* is used. This solver solves nonlinear data-fitting problems in least-squares sense [26].
In the least-squares method, a simple data set consists of m data points of data pairs (p_j, q_j) for data points $j = 1 \cdots m$ where p_j is an *independent* variable and q_j is a *dependent* variable whose value is found by observation [27]. In this occasion, the measured residual stress in transverse- $\sigma^\dagger_{11,j}(z)$ or in rolling direction $\sigma^\dagger_{22,j}(z)$ of data point j, represents the dependent variables, q_j, and the offset in rolled plate z_j represents the independent variable, p_j.

The least-squares method finds the optimal parameter values listed in $\boldsymbol{\beta}$ by minimizing the sum,

$$S = \sum_{j=1}^m r_j^2, \tag{2.4}$$

of squared residuals, r_j. In this occasion, vector $\boldsymbol{\beta}$ embodies a list of the fit function coefficients $c^\dagger_{11,i}$ and $c^\dagger_{22,i}$ for cosine modes $i = 1 \ldots n$ that are to be determined with the least-squares method.

The residual, r_j, describes the fit of a model to the data points and is defined as the difference between the actual value of the dependent variable, q_j, and the value predicted by the fit function $f(p_j, \boldsymbol{\beta})$,

$$r_j = q_j - f(p_j, \boldsymbol{\beta}). \tag{2.5}$$

For both the rolling as transverse direction, twelve curve-fits were realized using the least-squares method. For all twelve rolled plate samples, the optimal parameter values $\boldsymbol{\beta}$ have been determined, like

$$\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3, \ldots, \boldsymbol{\beta}_{12} = \begin{bmatrix} c^\dagger_{11,1} \\ c^\dagger_{11,2} \\ c^\dagger_{11,3} \\ c^\dagger_{11,4} \\ c^\dagger_{11,5} \\ c^\dagger_{11,6} \end{bmatrix}_1, \begin{bmatrix} c^\dagger_{11,1} \\ c^\dagger_{11,2} \\ c^\dagger_{11,3} \\ c^\dagger_{11,4} \\ c^\dagger_{11,5} \\ c^\dagger_{11,6} \end{bmatrix}_2, \begin{bmatrix} c^\dagger_{11,1} \\ c^\dagger_{11,2} \\ c^\dagger_{11,3} \\ c^\dagger_{11,4} \\ c^\dagger_{11,5} \\ c^\dagger_{11,6} \end{bmatrix}_3, \ldots, \begin{bmatrix} c^\dagger_{11,1} \\ c^\dagger_{11,2} \\ c^\dagger_{11,3} \\ c^\dagger_{11,4} \\ c^\dagger_{11,5} \\ c^\dagger_{11,6} \end{bmatrix}_{12}. \tag{2.6}$$

The sum of the squared residuals, S, provides a measure for the quality of the fit. Only modes $i = 1 \ldots 6$ are considered for the fit functions, since adding other more modes leads to negligible improvement of the quality of the fit. Three examples of the curve-fits of residual stress in the rolling direction, $\sigma^\dagger_{11}$, where the obtained fit functions are plotted together with the experimental stress data, are shown in Fig. 2.10.

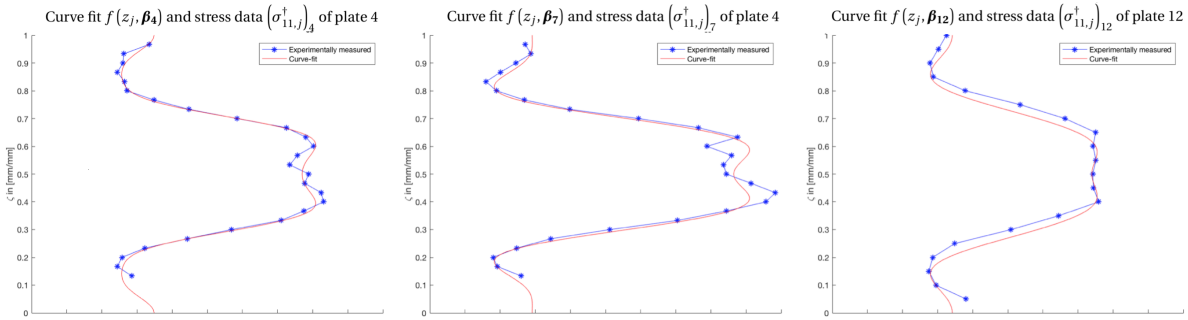


Figure 2.10: Three examples, being plate 4 (left), plate 7 (middle) and plate 12 (right), of the rolling direction where the experimental data of a specific plate is plotted (blue) together with the - by the least-squares method - obtained curve-fit (red).

The obtained curve-fits appear to approximate the experimental data well.
Now that twelve fit functions $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \ldots \boldsymbol{\beta}_{12}$ having six fit function coefficients each have been obtained by using the least-squares method for both the rolling- and transverse direction, the stochastics can be deduced. First, the twelve fit functions for the rolling direction, obtained in Eq. 2.6, are organized per fit function coefficient, like

$$\boldsymbol{c^\dagger_{11,1}}, \boldsymbol{c^\dagger_{11,2}}, \ldots, \boldsymbol{c^\dagger_{11,6}} = \begin{bmatrix} \left(c^\dagger_{11,1}\right)_1 \\ \left(c^\dagger_{11,1}\right)_2 \\ \left(c^\dagger_{11,1}\right)_3 \\ \vdots \\ \left(c^\dagger_{11,1}\right)_{12} \end{bmatrix}, \begin{bmatrix} \left(c^\dagger_{11,2}\right)_1 \\ \left(c^\dagger_{11,2}\right)_2 \\ \left(c^\dagger_{11,2}\right)_3 \\ \vdots \\ \left(c^\dagger_{11,2}\right)_{12} \end{bmatrix}, \ldots, \begin{bmatrix} \left(c^\dagger_{11,6}\right)_1 \\ \left(c^\dagger_{11,6}\right)_2 \\ \left(c^\dagger_{11,6}\right)_3 \\ \vdots \\ \left(c^\dagger_{11,6}\right)_{12} \end{bmatrix}. \tag{2.7}$$

Each fit function coefficient, $c^{\dagger}_{11,i}$, now has a dataset with a sample size of $l = 12$. Due to the small sample size not much can be said about the distribution of the fit function coefficients. A *normal* distribution will be assumed for each fit function coefficient. Normal distributions are often used to represent real-valued random variables whose distributions are unknown [28]. A normal distribution is defined by two parameters, being the mean, commonly denoted by μ, and the standard deviation, commonly denoted by σ. In Fig. 2.11, the relation between these two parameters and the normal distribution is illustrated.

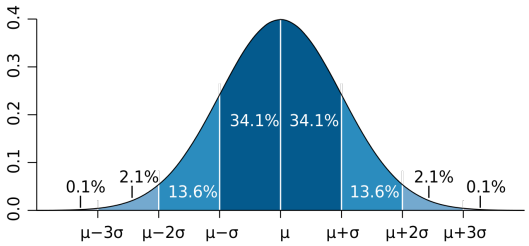


Figure 2.11: A plot of the normal distribution in which the significance of the two parameters - the mean μ and standard deviation σ - that describe the normal distribution is demonstrated graphically [28].

There is a distinction between the *population* mean and -standard deviation, symbolized by μ and σ, respectively, and the *sample* mean and -standard deviation, symbolized by $\overline{x}$ and s, respectively. Since this case concerns fit function coefficients having a *finite* sample size of $l = 12$, the discrete random variable notation should be used. Since the symbol x is already used for the x-coordinate in the coordinate system notation, the symbol of the population mean, μ, will be used for the sample mean. The symbols μ and s, will thus be used to indicate the sample mean and standard deviation, respectively. The obtained sample mean and standard deviation will not very accurately describe the population mean- and standard deviation due the small sample size. For each fit function coefficient $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$ for $i = 1 \dots n$ with $n = 6$, the sample mean

$$\mu = \frac{1}{l} \sum_{k=1}^{l} t \tag{2.8}$$

and the variance

$$s^2 = \frac{1}{l-1} \sum_{k=1}^{l} \left(t - \mu\right)^2, \tag{2.9}$$

are determined. In these equations, the variable t has no physical meaning. The variable t should be substituted with the fit function coefficients $c^{\dagger}_{11,i}\Big|_k$ and $c^{\dagger}_{22,i}\Big|_k$ with k being the sample for $k = 1 \dots l$ with sample size $l = 12$.
The concluding fit functions that describe stochastic residual stress in rolled plate in rolling direction is

$$\sigma^{\dagger}_{11}(z) = \sum_{i}^{n} c^{\dagger}_{11,i} \cos\left(i 2\pi\zeta(z)\right) \qquad \text{for } i = 1 \dots 6 \tag{2.10}$$

and in transverse direction is

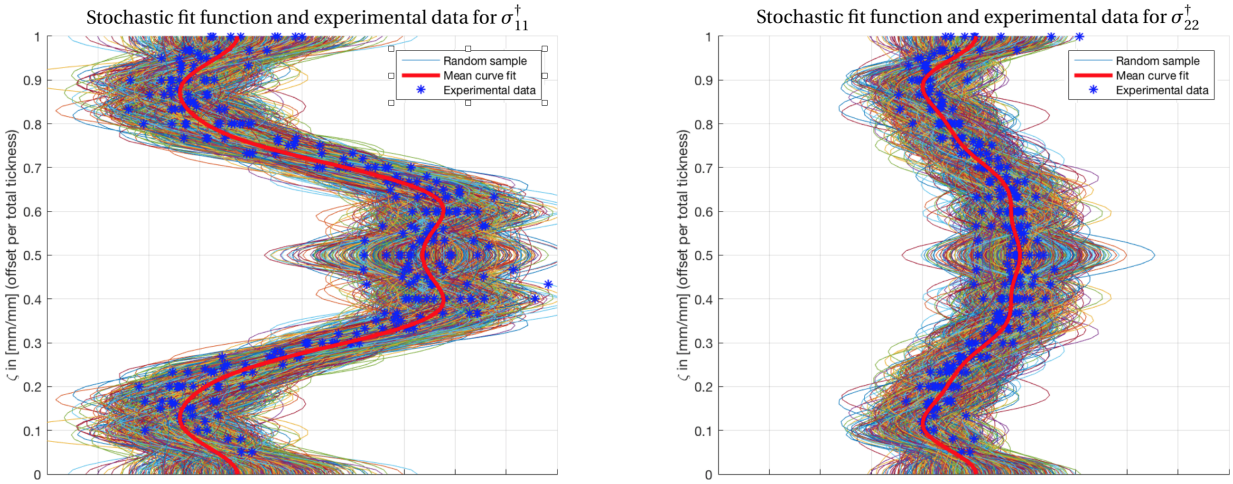
$$\sigma^{\dagger}_{22}(z) = \sum_{i}^{n} c^{\dagger}_{22,i} \cos\left(i 2\pi\zeta(z)\right) \qquad \text{for } i = 1 \dots 6. \tag{2.11}$$

In these equations, the fit function coefficients, $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$, are normally distributed. The mean and standard deviation defining the normal distributions are listed in table 2.1.

Table 2.1: In this table the two parameters, being the mean μ and standard deviation s, defining the normal distributions of the stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$ are listed.

$c^{\dagger}_{11,i} \sim \mathcal{N}(\mu, s)$		$c^{\dagger}_{22,i} \sim \mathcal{N}(\mu, s)$	
$c^{\dagger}_{11,1} \sim$		$c^{\dagger}_{22,1} \sim$	
$c^{\dagger}_{11,2} \sim$		$c^{\dagger}_{22,1} \sim$	
$c^{\dagger}_{11,3} \sim$	Not shown due to confidentiality	$c^{\dagger}_{22,1} \sim$	Not shown due to confidentiality
$c^{\dagger}_{11,4} \sim$		$c^{\dagger}_{22,1} \sim$	
$c^{\dagger}_{11,5} \sim$		$c^{\dagger}_{22,1} \sim$	
$c^{\dagger}_{11,6} \sim$		$c^{\dagger}_{22,1} \sim$	

In order to see whether the determined fit functions for $\sigma^{\dagger}_{11}(z)$ and $\sigma^{\dagger}_{22}(z)$ formulated in Eq. 2.10 and Eq. 2.11, respectively, with normally distributed stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$ as are listed in Tab. 2.1 correlate with the experimental stress data, a *Monte Carlo* experiment is done. For both rolling as transverse direction, the stress profiles of one thousand random rolled plates are simulated by drawing random samples from the stress coefficient distributions. For each of the one thousand samples, $\sigma^{\dagger}_{11}(z)$ and $\sigma^{\dagger}_{22}(z)$ are subsequently plotted in a graph along with the experimental stress data. These one thousand random samples represent a batch of a thousand rolled plate samples. These plots, for both rolling- and transverse direction, are shown in Fig. 2.12.



Not shown due to confidentiality

Not shown due to confidentiality

Figure 2.12: The experimental stress data (blue asterisks) is plotted together with one thousand random samples drawn from the normally distributed stress coefficients (thin colored lines). The mean curve fit is also plotted (thick red line). The one thousand random samples (representing a batch of one thousand random rolled plates) appears to correlate with the distribution of the experimental stress data.

From Fig. 2.12, it appears that the experimental stress data lies within the area covered by the one thousand random samples. Not much can be concluded about the accuracy of the stochastic representation, other than that the stochastic representation appears to correlate with the experimental stress data. The stochastic representation, however, is based on a batch of twelve rolled plates. The reliability can obviously be improved by increasing the sample size.

Chapter 3

The fundamentals of part distortion

By definition, residual stresses are stresses locked into a rigid part in the *absence* of external forces or thermal gradients [14]. Residual stresses in a body do not result in deformation of the body as the *equilibrium conditions* are satisfied. In order for the body to be in equilibrium, the forces and moments produced by the residual stresses inside the body must be equal to the external forces and moments acting on the body [29–32]. Since, by definition, the external forces and moments acting on the body are zero, the condition for the residual stress profile is implied: internal forces and moments generated by residual stress should be equal to zero. In other words, residual stress inside a body must be *self-equilibrated*.

3.1 Self-equilibrated stress

Self-equilibrated residual stress implies two conditions, the first being the force equilibrium condition and the second being the moment equilibrium condition. Force equilibrium for any plane section area A of the whole body requires

$$\int_A \sigma^{\dagger}(z)\, dA = 0 \tag{3.1}$$

and moment equilibrium with respect to an arbitrary reference line in the section area A demands

$$\int_A \sigma^{\dagger}(z) z\, dA = 0. \tag{3.2}$$

$\sigma^{\dagger}(z)$ symbolizes residual stress and z symbolizes the reference line distance of an element dA of the area A. An example concerning a simple beam with rectangular cross section where the two conditions for self-equilibrated stress are verified is shown in Fig. 3.1.
In Fig. 3.1, a beam is shown with rectangular cross section having a simplified one-dimensional residual stress profile on plane section area $A-A$. This residual stress profile, $\sigma^{\dagger}_{11}(z)$, can be described by the function

$$\sigma^{\dagger}_{11}(z) = c\cos\big(\beta(z-z_0)\big) \qquad \text{with } \beta = \frac{2\pi}{h}.$$

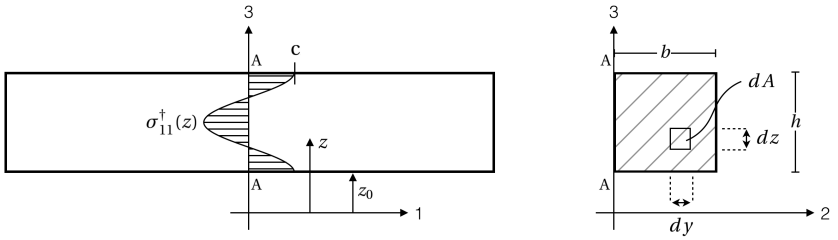


Figure 3.1: An example of a simple beam (left) having a simplified one-dimensional residual stress profile $\sigma^{\dagger}_{11}(z)$ on plane section area $A-A$ (right). Note: the location of the coordinate system's origin is chosen arbitrarily in space.

The internal force generated by $\sigma^{\dagger}_{11}(z)$ on plane section area $A-A$ is equal to

$$\begin{aligned} F_{\text{int}} &= \int_A \sigma^{\dagger}_{11}(z)\, dA \\ &= \int_{y=0}^{b} \int_{z=z_0}^{z_0+h} c\cos\big(\beta(z-z_0)\big)\, dz\, dy \\ &= \frac{bc}{\beta}\Big(\sin\big(\beta(z-z_0)\big)\big|_{z=z_0+h} - \sin\big(\beta(z-z_0)\big)\big|_{z=z_0}\Big) \\ &= 0. \end{aligned}$$

and the internal moment generated by $\sigma^{\dagger}_{11}(z)$ on plane section area $A-A$ is equal to

$$\begin{aligned} M_{\text{int}} &= \int_A \sigma^{\dagger}_{11}(z)\, z\, dA \\ &= \int_{y=0}^{b} \int_{z=z_0}^{z_0+h} c\cos\big(\beta(z-z_0)\big)\, z\, dz\, dy \\ &= \frac{bc}{\beta} z\sin\big(\beta(z-z_0)\big)\big|_{z=z_0+h} \\ &+ \frac{bc}{\beta^2}\Big(\cos\big(\beta(z-z_0)\big)\big|_{z=z_0+h} - \cos\big(\beta(z-z_0)\big)\big|_{z=z_0}\Big) \\ &= 0. \end{aligned}$$

Since both the force equilibrium condition (Eq. 3.1) as the moment equilibrium condition (Eq. 3.2) is met, the residual stress profile on plane section area $A-A$ illustrated in Fig. 3.1 is said to be self-equilibrated and no deformation will thus occur on this plane section.

3.2 Homogeneity of stress in rolled plate

In literature, residual stress in rolled plate stock material is generally assumed to be homogeneous. This means that the residual stress state does not vary along the rolling- and transverse direction. This is illustrated in Fig. 3.2, where identical stress profiles are shown for three arbitrarily chosen cross sections throughout the rolled plate's rolling direction.
In theory however, the stress state throughout the rolling- and transverse direction cannot be fully homogeneous due to the finite nature of rolled plate. The equilibrium condition and

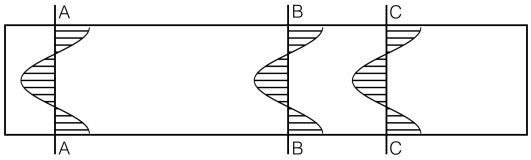


Figure 3.2: A two-dimensional view of rolled plate stock material where the stress state is assumed to be homogeneous. The figure shows three arbitrarily chosen plane section areas $A-A$, $B-B$ and $C-C$ all having identical stress profiles.

the absence of external loads require the boundaries of rolled plate to be *traction-free*. The boundary conditions of the traction-free surfaces require that the normal and shear stress components along the boundary plane are zero [33]. When, for example, a beam having a one-dimensional stress profile like is shown in Fig. 3.3 is sectioned into two halves along a cutting plane, the residual stresses on the cutting plane are fully released and new traction-free surfaces are created. Stress relaxation at the cutting plane happens through deformation of the cut surface. The deformed boundary planes as a result of stress relaxation are referred to by the rolled plate's 'end-effects'.

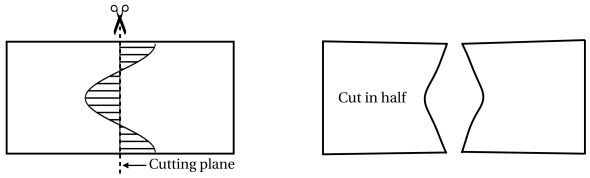


Figure 3.3: An example where a beam having a one-dimensional residual stress profile is sectioned into two halves along a cutting plane (left). The boundary conditions require the created cut surfaces to be traction-free. As a result, the cut surfaces will deform (right) until the stress at the boundaries is fully relieved [33].

In theory, residual stress in rolled plate is thus not fully homogeneous. Near the free surfaces a transition region exists as the residual profile reduces to zero. In the application of aircraft component manufacture, rolled plate typically has large dimensions relatively to the length of the transition region, like is shown in Fig. 3.4. Since the majority of the rolled plate is unaffected by the end-effects, the end-effects can be ignored and residual stress can be assumed homogeneous.

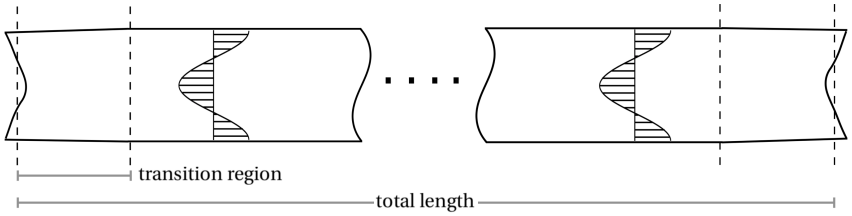


Figure 3.4: A schematic two-dimensional view of rolled plate stock material and illustrates the fact that in the application of aircraft component manufacture the dimensions of the transition region (or end-effects) are negligible compared with the dimensions of the entire rolled plate. A homogenous stress profile exists in the majority of rolled plate.

The presence of end-effects at the boundaries of rolled plate material can be *corrected* by

applying boundary conditions such as the ones that are illustrated in Fig. 3.5. Having applied these boundary conditions, residual stress in rolled plate can be rightfully qualified as homogenous.

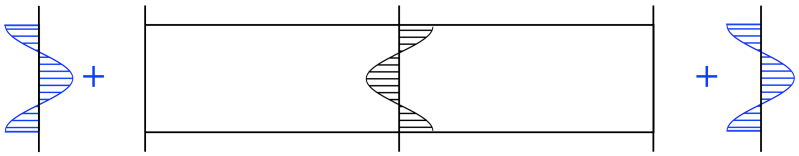


Figure 3.5: A two-dimensional view of rolled plate with stress and applied external loads (blue) at the boundaries. Due to the applied external loads, residual stress in the beam is now fully homogenous.

In this research, residual stress in rolled plate will be assumed homogeneous, which implies that boundary conditions such as ones illustrated in Fig. 3.5 are considered.

3.3 Machining considered as an one-to-one mapping of stress

Part distortion happens when stresses in the body *do not* satisfy the equilibrium conditions, or, in other words, when stresses do not self-equilibrate. In this research, machining will be considered as a one-step removal of material without influencing the stress in the remaining material. This is illustrated in Fig. 3.6. The stress in the remaining geometry is considered to be a one-to-one mapping of the stress from the rolled plate.

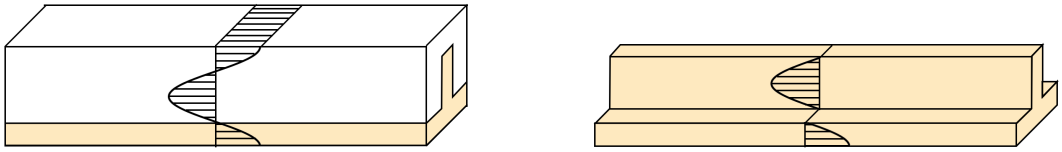


Figure 3.6: A piece of rolled plate stock material (left) from which a T-section component (right) is manufactured. Machining is assumed to be a one-step removal of material with a one-to-one mapping of residual stress from rolled plate to the remaining material. Consequently, distortion happens through stress relaxation since the stresses in the remaining material do no longer self-equilibrate.

By considering machining as an one-to-one mapping of stress, several assumptions are made. To begin with, secondary process induced stresses are assumed to be negligible, the reason for which was elaborated upon in Section 2.1. Furthermore, machining is assumed to be a *one-step* removal of material. This would be a correct assumption if in practice the entire part is machined from rolled plate in a single step while the material is held fixed and cannot deform during the machining process. In reality, however, this is not the case. During my internship at AIRBUS from April to July 2017, I visited an AIRBUS subcontractor responsible for component manufacture named GROUPE ROSSI AERO, based in Toulouse (France). I discovered that long machining iterations - referred to as *multiple step* machining - are current practice in component manufacture [13]. During a machining step, the material is held fixed and in between machining steps, the component is released from fixture as a result of which stress relaxation occurs. As a result of stress relaxation in between steps,

part distortion differs from when the part would have been machined in a single step. In other words, part distortion is *path dependent*. The reason behind path dependency of part distortion is explained by means of illustrations in Fig. 3.7 and Fig. 3.8 [13]. Fig. 3.7 illustrates a one-step machining sequence of a T-section from rolled plate. Beforehand, a set of material points is marked. After the one-step machining sequence, the remaining material contains the same set of marked material points at start.

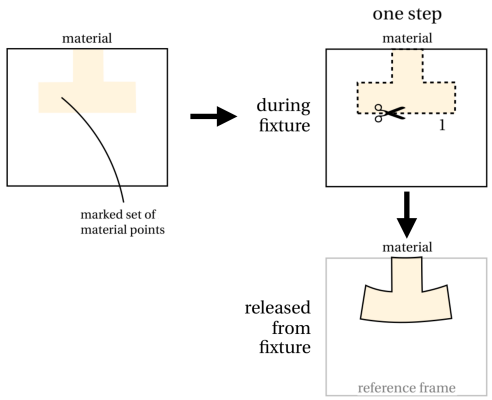


Figure 3.7: A scenario where a T-section is manufactured from rolled plate in a single step. First, a set of material points is marked. Subsequently, the marked set of material points is machined in one-step while held in fixture. When released from fixture the remaining material deforms. The remaining material contains the same marked material points that were marked in the beginning [13].

Fig. 3.8 illustrates a multiple step machining sequence of a T-section from rolled plate. The same set of material points is marked at start. In between steps, the material is released from fixture as a result of which stress relaxation (or deformation) occurs in between machining steps.

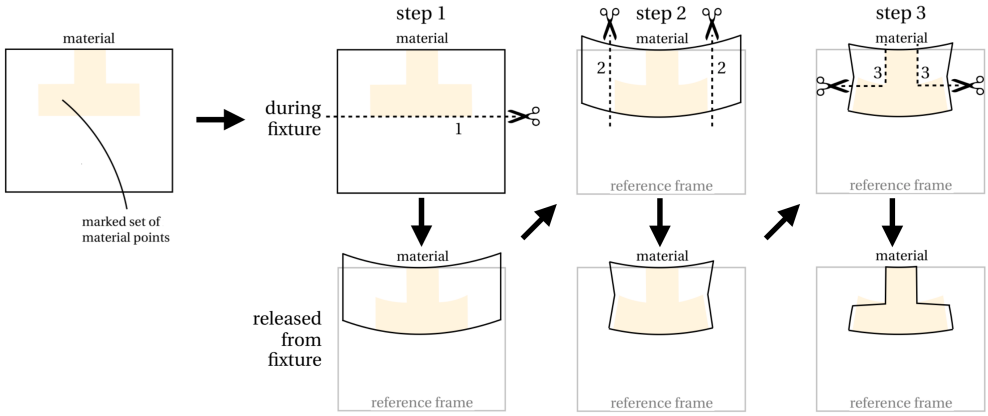


Figure 3.8: A scenario where the same T-section is manufactured from rolled plate in a multiple step machining sequence. The same set of marked material points does not completely end up in the remaining material after a multiple step machining sequence. The reason for this is that the material deforms in between steps while the configuration of machining paths does not change. As a result, a different set of material points ends up in the remaining material [13].

Since the configuration of the machining paths does not change, a different set of material points ends up in the remaining material than the marked set of material points at start.
To conclude, by assuming one-to-one mapping of stress from the rolled plate onto the remaining material, one-step machining is assumed and secondary process induced stresses are not taken into account.

3.4 Analytical derivation of distortion

A slender beam with rectangular cross section is considered which will be cut in half such as is illustrated in Fig. 3.9. As external loads - representing the clamping forces - are acting on the two boundary planes, the residual stress profile in the beam, $\sigma_{11}(z)$, is homogeneous throughout its length. Like is illustrated in the figure, *isostatic* boundary conditions are put in place. In total, six displacement constraints are placed at three chosen material points positioned on cross section A-A, like is shown in the figure. This way, the beam's six rigid body modes are blocked. The beam can deform, however the beam as a whole cannot translate in any of the three directions nor rotate about any of the three axes.

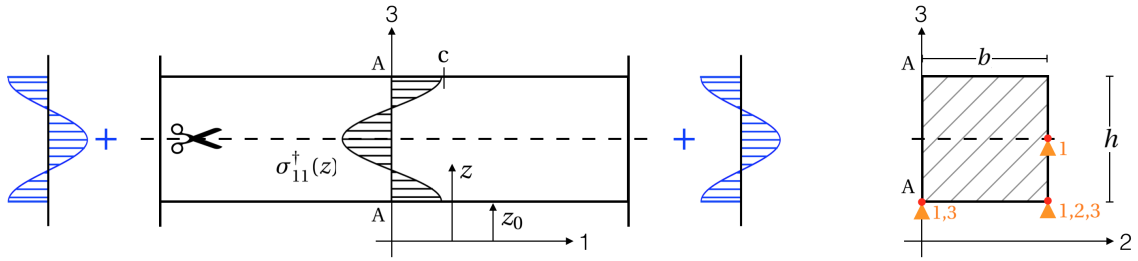


Figure 3.9: A beam which is in fixture (left) with a rectangular cross section (right) which is about to be sectioned in half at the beam's middle plane (scissor on dotted line). The beam's stress profile, $\sigma_{11}^{\dagger}(z)$, is homogeneous through x as external loads representing the clamping forces (blue) act on the boundary planes. Six displacement constraints (orange) are put in place at three material points (red) positioned on cross section A-A.

The piece of stock material illustrated in Fig. 3.9 has material- and geometrical properties that are characteristic for aircraft components where distortion is relevant. For rolled plate stock material, a 7000 series wrought Aluminum-Titanium alloy of the type 7050-T7451 (stretched), is considered. This alloy is favored for of its combination of high strength, stress-corrosion-cracking resistance and toughness [34] and has a Young's Modulus of $E = 70.07\,\text{GPa}$. The magnitude of the residual stress is $c = 20\,\text{MPa}$, a magnitude typically found in rolled plate. The piece of stock material furthermore has a thickness of $h = 30\,\text{mm}$, a length of $l = 1000\,\text{mm}$ and a width of $b = 80\,\text{mm}$. The piece of stock material is positioned at an arbitrarely chosen offset of $z_0 = 100\,\text{mm}$ with respect to the coordinate system in Fig. 3.9.

After the beam is cut in half, like was indicated in Fig. 3.9, the upper half of material is removed. Fig. 3.10 shows the beam after machining.
The beam is still in fixture as the boundary loads are still acting at the boundary planes. As the remaining material is still in fixture, deformation or *stress relaxation* has not yet occurred. For the sake of convenience, a symmetry plane can be defined, like is shown in Fig. 3.11.

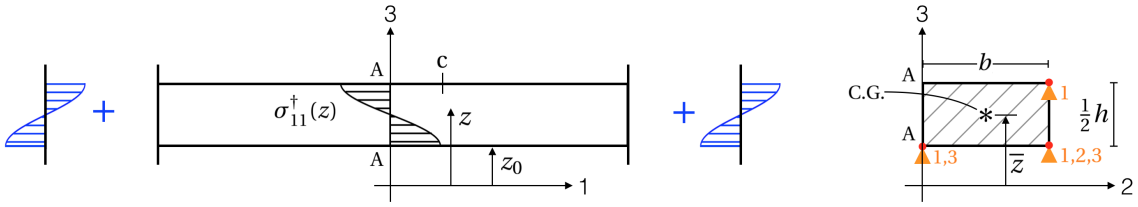


Figure 3.10: The remaining material of the beam after the beam is sectioned in half like was indicated in Fig. 3.9. The external boundary loads are still acting as the beam is still in fixture.

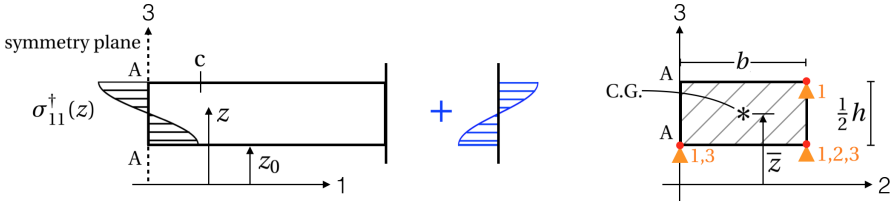


Figure 3.11: The same illustration of the remaining material after the beam is sectioned in half as in Fig. 3.11. This time, a symmetry plane has been defined at cross section A-A marked with a dotted line.

Since the stress profile present in the remaining material does not satisfy the equilibrium conditions mentioned in Chapter 3.1, the beam deforms upon removal of the external boundary loads (or clamping force). As can be concluded by observation of Fig. 3.11, the stress profile does satisfy force equilibrium however does not satisfy moment equilibrium. Upon removal of the external boundary loads, stress relaxation will therefore *not* lead to contraction, however will result in bending, like is illustrated in Fig. 3.12. The remaining material deforms until the beam has relaxed in such a manner that the stress profile is self-equilibrated once more. Fig. 3.12 illustrates the fact that some residual stress is still present after stress relaxation. However, the stress profile that is present is self-equilibrated.

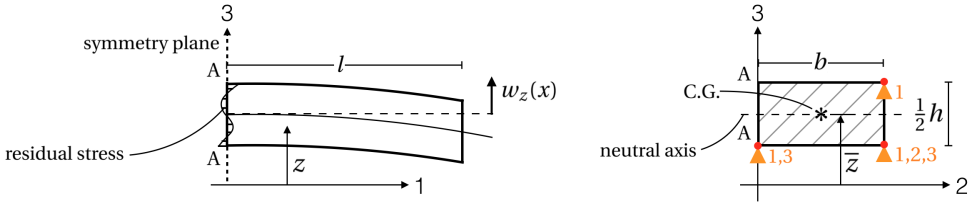


Figure 3.12: The remaining material upon removal of the external boundary loads, in other words: when the remaining material is released from fixture. Stress relaxation or deformation has occurred in the form of bending. Note that some residual stress is still present after deformation. However, this remaining residual stress is self-equilibrated once more. Note that the neutral y-axis is indicated (right).

Distortion of the beam can be derived analytically. Since in this case distortion occurs in the form of bending, distortion is defined as $w_z(x = l)$, being the translation of the neutral plane in z-direction at the tip of the beam ($x = l$). First, a mathematical description for the stress profile, $\sigma^{\dagger}_{11}(z)$, is formulated.

$$\sigma^{\dagger}_{11}(z) = c\cos\left(\frac{2\pi}{h}(z - z_0)\right).$$

The moment about the neutral y-axis, generated by the *non*-self-equilibrated stress profile after material removal, M_{y}, is obtained by

$$\begin{aligned} M_{\mathrm{y}} &= \int_A \sigma_{11}^{\dagger}(z)(z-\overline{z})\, dA_x \\ &= \iint \sigma_{11}^{\dagger}(z)(z-\overline{z})\, dz\, dy. \end{aligned} \tag{3.3}$$

Note that since the stresses do no longer satisfy the equilibrium conditions, the moment can no longer be taken with respect to an *arbitrary* reference line, like with the moment equilibrium condition stated in Eq. 3.2. As the stresses do not satisfy the equilibrium conditions, the moment must be taken about the body's *neutral y-axis*. The neutral y-axis passes through the body's geometric centroid or centre of gravity (C.G.) and is indicated in Fig. 3.12. The z-coordinate of the geometric centroid, $\overline{z}$, is obtained by

$$\overline{z} = \frac{S_y}{A_x}, \tag{3.4}$$

in which

$$S_y = \int_{A_x} z\, dA_x = \iint z\, dz\, dy \tag{3.5}$$

and represents the first moment of area of the cross section about the neutral y-axis and

$$A_x = \int_{A_x} dA_x = \iint dz\, dy \tag{3.6}$$

and represents the area normal to x, which is the area of the cross section shown on the right in Fig. 3.12.

The relationship between the deflection of the beam's neutral xy-plane along the x in z-direction, $w_z(x)$, and the moment generated by the stress, M_y, is sought. First, the material is assumed to be homogeneous and to behave in a linear-elastic manner, as a result of which Hooke's law applies [35].

$$\epsilon = \frac{\sigma}{E}. \tag{3.7}$$

The *flexure formula* relating the stress distribution to the internal resulting bending moment applies since it is assumed that a linear variation in normal stress results in a linear variation of normal strain [36].

$$\sigma = -\frac{Mz}{I_y}. \tag{3.8}$$

The relationship between the beam's curvature and Eq. 3.7 and Eq. 3.8 gives [37]

$$\frac{1}{\rho} = \frac{M}{EI_y}. \tag{3.9}$$

When assuming small displacements with respect to the beam's length - which can be assumed in aircraft distortion related problems - the relationship between the curvature and the deflection, $w_z(x)$, can be approximated to be [38]

$$\frac{1}{\rho} = \frac{d^2 w_z}{dx^2}. \tag{3.10}$$

Combining Eq. 3.9 and Eq. 3.10 yields

$$\frac{d^2 w_z}{dx^2} = \frac{M}{EI_y}. \tag{3.11}$$

Integrating this expression and assuming boundary conditions of a beam clamped at $x = 0$, by which $w(x = 0) = 0$ and $\frac{dw(x=0)}{dx} = 0$ hold, yields the expression

$$w_z(x) = \frac{M_y x^2}{2EI_y}, \tag{3.12}$$

relating deflection $w_z(x)$ with the moment generated by the stress profile, M_y [38]. This expression is equivalent to the simple beam flexure formula ("Vergeet-mij-nietje") for a cantilever beam fixed at one end and a couple moment at the beam's free end.

$$I_y = \int_{A_x} (z - \overline{z})^2 \, dA_x = \iint (z - \overline{z})^2 \, dz \, dy \tag{3.13}$$

is the second moment of area of the cross section about the neutral y-axis.
In Appendix A.1, the analytical distortion derivation is worked out for the beam that is cut in half, which was illustrated in Fig. 3.9. The analytical solution for distortion, that in this case is defined as $w_z(x{=}l)$, was found to be

$$w_z(x{=}l) = -\frac{24cl^2}{Eh\pi^2}.$$

Substituting geometrical- and material properties l, h, c and E gives a distortion of

$$w_z(x{=}l) = -23.1\,\text{mm}.$$

Now, the same piece of stock material as was illustrated in Fig. 3.9 is considered, however in this case a lower and upper part of material will be removed, like is illustrated in Fig. 3.13.

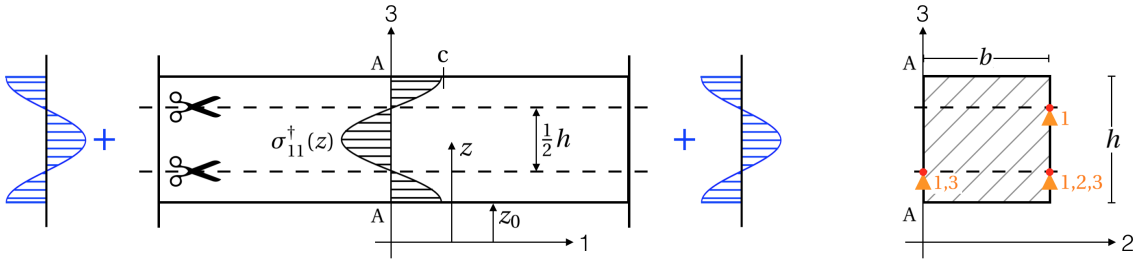


Figure 3.13: The same piece of stock material that was illustrated in Fig. 3.9. This time, material is removed symmetrically from both sides. The displacement constraints are applied on the remaining material.

After the beam is symmetrically machined from both sides, like was indicated in Fig. 3.13, the upper and lower parts of material are removed. Fig. 3.14 shows the remaining material after machining. The beam is still in fixture and stress relaxation has not yet occurred as the boundary loads are still acting at the boundary planes.

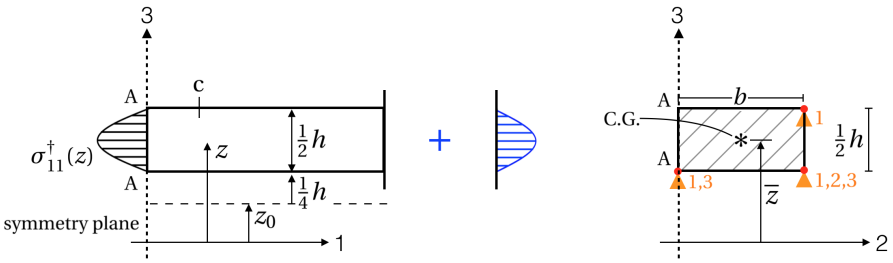


Figure 3.14: The remaining material of the beam after the beam is machined symmetrically from both sides, as was indicated in Fig. 3.13. As the boundary loads are still acting on the boundary planes, no deformation has yet occurred. A symmetry plane has been defined at cross section A-A marked with a dotted line.

The stress profile in the remaining material does not satisfy the equilibrium conditions as a result of which the beam deforms upon removal of the external boundary loads. As can be concluded by observation of Fig. 3.14, this time the stress profile does satisfy moment equilibrium however does not satisfy force equilibrium. Upon removal of the external boundary loads, stress relaxation will therefore *not* result to bending, however will lead to contraction like is illustrated in Fig. 3.15. The remaining material deforms until the beam has relaxed in such a manner that the stress profile is self-equilibrated once more.

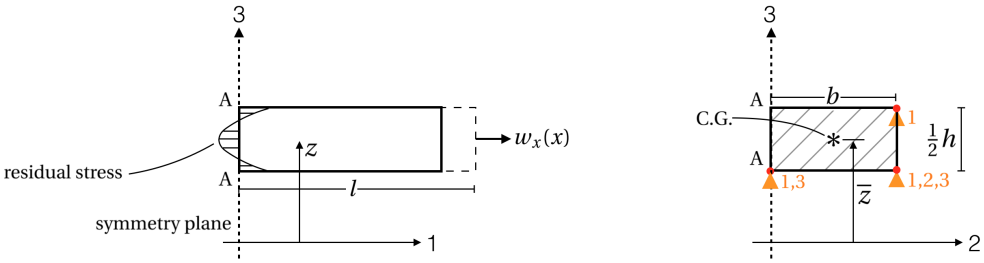


Figure 3.15: The deformed remaining material after removal of the external boundary loads. Stress relaxation or deformation has occurred in the form of contraction. Some residual stress is still present after deformation, however the remaining stress profile is self-equilibrated once more.

The resultant force, F_x, generated by the *non*-self-equilibrated stress profile after material removal is obtained by

$$\begin{aligned} F_{\mathrm{x}} &= \int_{A_x} \sigma_{11}^{\dagger}(z)\, dA_x \\ &= \iint \sigma_{11}^{\dagger}(z)\, dz\, dy. \end{aligned} \tag{3.14}$$

Since in this case distortion occurs in the form of contraction, distortion is defined as the displacement of the beam's tip in x-direction, $w_x(x{=}l)$. The displacement of beam's tip relatively to the fixture, $w_x(x{=}l)$, caused by the net force of the non-self-equilibrated stress profile is obtained by [39]

$$w_x(x{=}l) = \int_0^l \frac{F_x(x)\, dx}{A_x(x)E}. \tag{3.15}$$

Since both $F_x(x)$ and $A_x(x)$ are constant throughout x, $F_x(x) = F_x$ and $A_x(x) = A_x$, and as a result Eq. 3.15 can be integrated to yield

$$w_x(x{=}l) = \frac{F_x l}{A_x E}. \tag{3.16}$$

In Appendix A.2, the analytical distortion derivation is worked out for the beam that is symmetrically machined from both sides, as was illustrated in Fig. 3.13. The analytical solution for distortion, that in this case is defined as $w_x(x{=}l)$, was found to be

$$w_x(x{=}l) = -\frac{2cl}{\pi E}.$$

Substituting geometrical- and material properties l, c and E gives a distortion of

$$w_z(x{=}l) = -0.182\,\text{mm}.$$

In the first scenario, moment equilibrium was not satisfied which resulted in distortion in the form of bending with $w_z(x{=}l) = -23.1\,\text{mm}$. In the second scenario, force equilibrium was not satisfied resulting in distortion in the form of contraction with $w_x(x{=}l) = -0.182\,\text{mm}$. A bending moment seems to result in distortion significantly larger than distortion due to net force. Aircraft components tend to have slender and lightweight designs with low stiffness in bending and large stiffness in contraction. This explains why only distortion in the form of bending is considered and distortion in the form of contraction can often be neglected.

3.5 Example: T-section

Next to a beam with rectangular cross section, distortion can be analytically derived for other geometries as well. Suppose that a structural beam with a T shaped cross section - i.e. *T-section* - is manufactured from a piece of rolled plate stock material such as is illustrated in Fig. 3.16. Distortion of a T-section can be analytically derived as well.

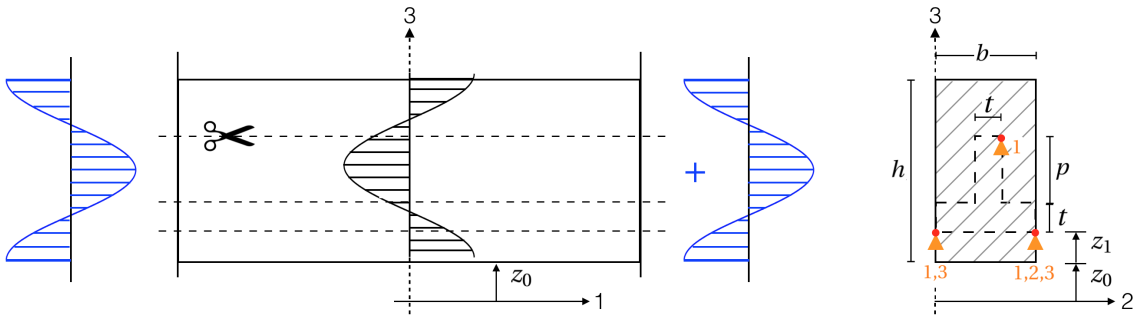


Figure 3.16: A piece of stock material from which a T-section (dotted lines) is to be manufactured. The T-section is machined with an offset of z_1 with respect to the rolled plate's bottom face.

The same piece of stock material as in Chapter 3.4 is considered with a Young's Modulus of $E = 70.07\,\text{GPa}$ and a residual stress magnitude of $c = 20\,\text{MPa}$. This time, the piece of rolled

plate has slightly different geometry features, being a thickness of $h = 80\,\text{mm}$, a length of $l = 1000\,\text{mm}$ and a width of $b = 40\,\text{mm}$. The flanks of the T-section geometry have a thickness of $t = 10\,\text{mm}$ and a height of $p = 30\,\text{mm}$. The piece of stock material is positioned at an arbitrarily chosen offset of $z_0 = 100\,\text{mm}$ with respect to the coordinate system. The T-section has an offset of $z_1 = 10\,\text{mm}$ with respect to the piece of rolled plate's bottom face. After manufacturing, a non-self-equilibrated stress profile is present in the remaining material, such as is illustrated in Fig. 3.17.

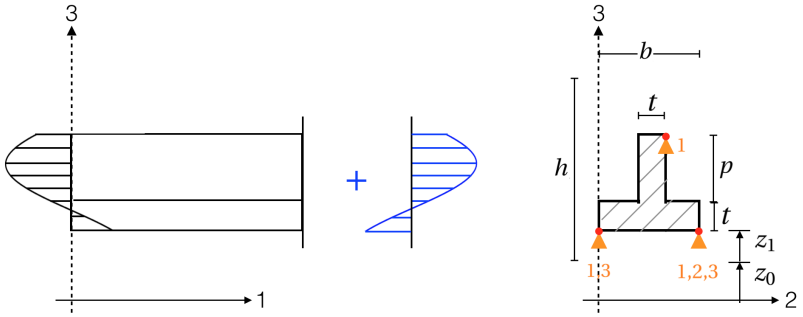


Figure 3.17: The remaining material - a structural beam with a T shaped cross section - with its non-self-equilibrated stress profile after manufacturing. Since external boundary loads are still acting, distortion has not yet occurred.

When external boundary loads are removed, distortion occurs. Since Chapter 3.4 demonstrated that distortion due to contraction is negligible compared with distortion due to bending, the deflection of the T-section's tip at $x = l$, $w_z(x{=}l)$, will be considered as a measure for distortion.

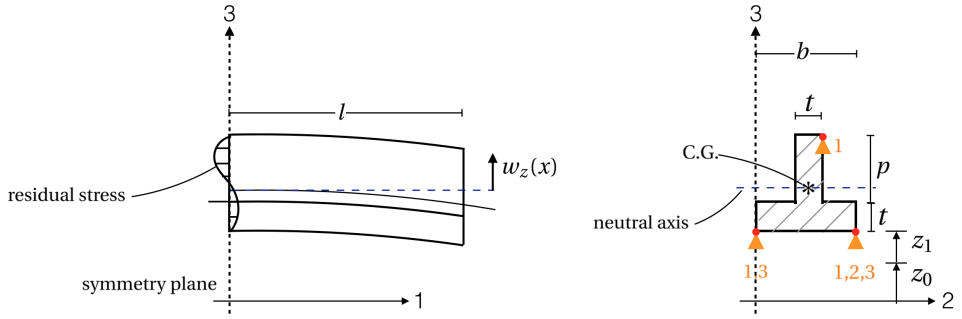


Figure 3.18: The remaining material after stress relaxation due to removal of the boundary loads. Note that the geometry bends about its neutral y-axis which passes through the geometry's centre of gravity (C.G.). Residual stress is still present after distortion.

In Appendix A.3, the analytical distortion derivation is worked out for the T-section that is manufactured from rolled plate stock material. The analytically derived expression for $w_z(x{=}l)$ is considerably complex and is shown in a Matlab m-file in Fig. B.1. Substituting geometric- and material properties yields a distortion of

$$w_z(x{=}l) = -6.9\,\text{mm}.$$

It can be concluded that distortion, $w_z(x{=}l)$, can be analytically derived for a T-section machined from a piece of rolled plate. However, the derivations for the bending moment, M_y,

and the second moment of area, I_y, and thereby distortion, $w_z(x{=}l)$, already become considerably complex.

3.6 Distortion minimization based on analytical equations

Distortion has been analytically derived for *prismatic* beams - i.e. beams with constant cross section throughout the beam's length - with rectangular or T shaped cross sections. Based on the obtained analytical equations for distortion, simple optimization problems can be solved. For instance, a T-section which is to be manufactured at offset z_1 with respect to the bottom face of the piece of rolled plate, such as was illustrated in Fig. 3.16. The offset z_1 can be optimized for distortion $w_z(x{=}l, z_1)$. The optimization problem formulation would be

$$\begin{aligned} \underset{z_1}{\text{minimize}} \quad & f(z_1) \\ \text{subject to} \quad & g_1 = \frac{t+p+z_1}{h} - 1 \leq 0 \\ \text{with} \quad & f(z_1) = w_z(x{=}l, z_1)^2. \end{aligned}$$

Inequality constraint g_1 makes sure that the T-section stays inside the piece of rolled plate $z \leq (h-(p+t))$. In order to obtain a function where the minimal value represents the value where distortion is zero, the objective function $f(z_1)$ is chosen to be equal to $f(z_1) = w_z(x{=}l, z_1)^2$. Since a beam with a T shaped cross section is prismatic, even an offset z_1^* can be found for which no distortion at all in the form of bending will occur, $w_z(x{=}l, z_1^*) = 0$. In Fig. 3.19 distortion $w_z(x{=}l, z_1)$ is plotted against offset z_1 that ranges from $0 \leq z_1 \leq (p+t)$.

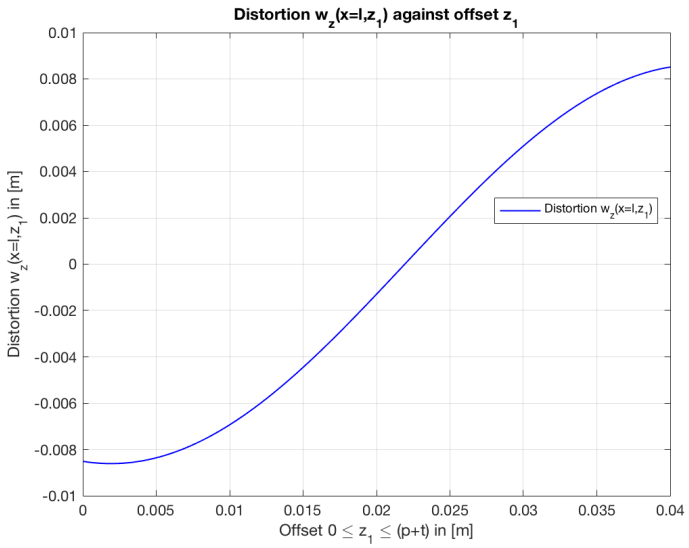


Figure 3.19: Distortion $w_z(x{=}l, z_1)$ of the T-section illustrated in Fig. 3.16 which is machined from rolled plate at offset z_1 with respect to the bottom face of the piece of rolled plate.

As can be observed, distortion $w_z(x{=}l)$ passes through $w_z(x{=}l, z_1^*) = 0$. The optimization problem is solved in Appendix B.1 and the offset z_1^* for which distortion is approximately equal to zero is found to be $z_1^* = 21.9\,\text{mm}$. Fig. 3.20 shows the positioning of the T-section at

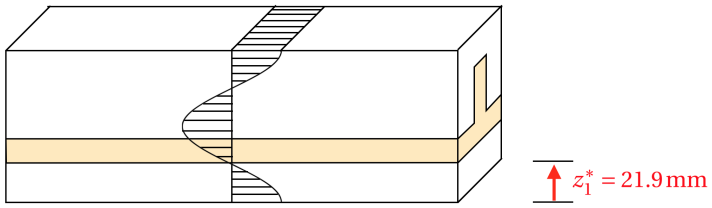


Figure 3.20: The offset z_1^* at which the T-section should be manufactured with respect to the bottom face of the piece of rolled plate in order to have zero distortion in the form of bending.

offset z_1^* at which the T-section should be machined as a result of which no distortion in the form of bending will occur.
A second design variable can be added to the optimization problem, such as for example the thickness of the flanks, t, which was specified in Fig. 3.16. This time, offset z_1 and flank thickness t are optimized for distortion. The optimization problem formulation becomes

$$\begin{aligned} \underset{z_1,t}{\text{minimize}} \quad & f(z_1,t) \\ \text{subject to} \quad & g_1 = \frac{t+p+z_1}{h} - 1 \leq 0 \\ & g_2 = \frac{t}{b} - 1 \leq 0, \\ \text{with} \quad & f(z_1,t) = w_z(x{=}l, z_1, t)^2. \end{aligned}$$

A second inequality constraint g_2 is added that dictates that the flank thickness does not exceed the piece of rolled plate's width $t \leq b$. The objective function as a function of the two design variables $f(z_1,t)$ can be visualized in a contour plot which is shown in Fig. 3.21. Appendix B.2 elaborates on the Matlab code used for obtaining the contour plot.
The solution for which the objective function is equal to zero, $f(z_1,t) = 0$, lies on a line which is plotted in Fig. 3.21.

3.7 Robust optimization based on analytical equations

This time, the stock material from which the T-section is manufactured consists of a batch of rolled plates that each have slightly different stress profiles. An optimization problem can be formulated in which the offset z_1 and flank thickness t of the T-section - manufactured from rolled plate, as was illustrated in Fig. 3.16 - are optimized for *robustness*. A mathematical description that resembles the distribution of stress in the batch was found by subjecting stress coefficient c in $\sigma_{11}^\dagger(z)$ to a normal distribution. As was agreed upon in Chapter 2.2, the symbols μ and s will be used for the mean and standard deviation, respectively.

$$\sigma_{11}^\dagger(z) = c\cos\big(\beta(z-z_0)\big) \qquad \text{with } c \sim \mathcal{N}\left(\mu_c = 20\,\text{MPa},\, s_c = 5\,\text{MPa}\right).$$

The standard deviation of distortion, s_{w_z}, can be used to relate to robustness. A high standard deviation of distortion s_{w_z} means that distortion has poor robustness; when the standard deviation of distortion is low, distortion is said to be robust. Since the standard deviation of stress coefficient c, s_c, is known and the relationship between stress $\sigma_{11}^\dagger(z)$ and distortion $w_z(x{=}l, z_1, t)$ has been determined analytically (in Appendix A.3), the standard

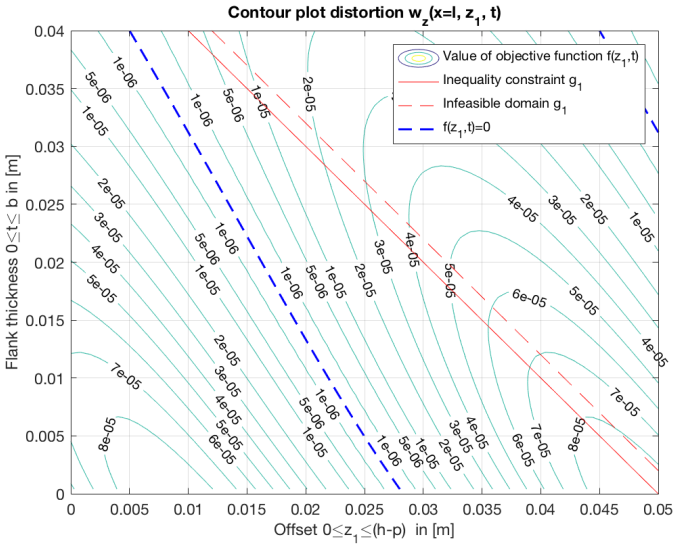


Figure 3.21: A contour plot showing objective function $f(z_1, t)$, which relates to distortion magnitude as a function of two design variables, being offset z_1 and flank thickness t. The solution where distortion is zero, $f(z_1, t) = 0$, lies on a (blue dotted) line which is shown in the figure.

deviation of distortion can be determined analytically as well. Probability theory [40] states that if the relation between function $f_2(z_1, t, c)$ and function $f_1(z_1, t)$ is like

$$f_2(z_1, t, c) = c f_1(z_1, t) \qquad \text{with } c \sim \mathcal{N}\left(\mu_c, s_c\right), \tag{3.17}$$

than $f_2(z_1, t, c)$ is normally distributed as well, $f_2 \sim \mathcal{N}\left(\mu_{f_2}, s_{f_2}\right)$. The mean, μ_{f_2}, can be determined through

$$\mu_{f_2} = \mu_c f_1(z_1, t) \tag{3.18}$$

and the standard deviation, s_{f_2}, can be determined through

$$s_{f_2} = s_c f_1(z_1, t). \tag{3.19}$$

In this occasion, the function $f_1(z_1, t)$ is equal to

$$f_1(z_1, t) = \frac{w_z(x{=}l, z_1, t, c)}{c}$$

and is not stochastic since c is excluded. Function $f_2(z_1, t, c)$ symbolizes distortion and is stochastic:

$$f_2(z_1, t, c) = w_z(x{=}l, z_1, t, c) \qquad \text{with } f_2 \sim \mathcal{N}\left(\mu_{f_2}, s_{f_2}\right).$$

Now that a measure for distortion robustness - i.e. the standard deviation s_{w_z} - is analytically determined, offset z_1 and flank thickness t can be optimized for robustness. An optimization problem is formulated below where offset z_1 and flank thickness t are optimized for

robustness while constraining the distortion to a maximum of $w_{z,\max} = 6\,\text{mm}$ (arbitrary).

$$
\begin{aligned}
\underset{z_1,t}{\text{minimize}} \quad & f(z_1,t) \\
\text{subject to} \quad & g_1 = \frac{t+p+z_1}{h} - 1 \leq 0, \\
& g_2 = \frac{t}{b} - 1 \leq 0, \\
& g_3 = \frac{w_z(x{=}l,z_1,t)}{w_{z,\max}} - 1 \leq 0 \\
\text{with} \quad & f(z_1,t) = s_{w_z}(z_1,t) \\
& w_{z,\max} = 6\,\text{mm}.
\end{aligned}
$$

The objective function as a function of the two design variables $f(z_1,t)$ is visualized in a contour plot which is shown in Fig. 3.22 of which the Matlab code is included in Appendix B.3. Two inequality constraints g_1 and g_2 dictate that T-section dimensions may not exceed rolled plate dimensions.

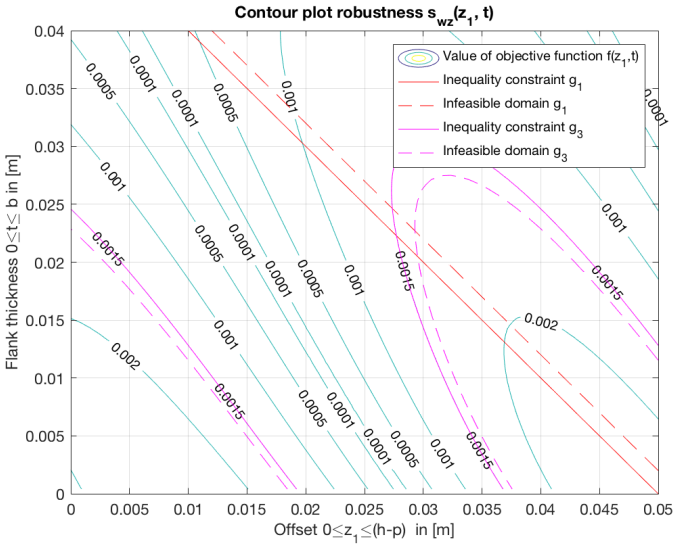


Figure 3.22: A contour plot visualizing the objective function $f(z_1,t)$ as a function of two design variables, being offset z_1 and flank thickness t. The objective function relates to *robustness*. Two inequality constraints g_1 and g_2 dictate that T-section dimensions may not exceed rolled plate dimensions. A third inequality constraint g_3 (purple) dictates that distortion may not exceed the maximum value of $w_z(x{=}l,z_1,t) \leq 6\,\text{mm}$.

3.8 Till what point can analytical equations be used?

For simple problems concerning *prismatic* beams, distortion and distortion robustness can be determined analytically, as was done for beams with rectangular and T shaped cross sections in Sections 3.4 and 3.5. The analytical distortion derivation for a T-section was already found to be considerably complex (Fig. A.2). Based on the analytical equations for distortion and robustness, simple optimization problems can be solved, as was done in Sections

3.6 and 3.7. Offset z_1 of the beam in rolled plate and a geometric property such as the flank thickness t have been optimized for distortion and robustness. The question is, till what point can distortion and robustness be derived analytically and until what point can optimization problems be solved based on analytical equations?
Two examples of aircraft components are shown in Fig. 3.23, which shows a structural bracket belonging the AIRBUS A350 aircraft and in Fig. 3.24, which shows an engine aircraft component. As becomes obvious, aircraft components are highly complex and *non-prismatic*. Due to this complexity, it is nearly impossible to analytically derive distortion and robustness. Therefore, Finite Element Analysis will be used for structural analyses concerning complex aircraft components.
Note: for prismatic beams in homogeneous stress fields, optima can be found where distortion in the form of bending of the whole component is equal to zero, like was discovered in Sections 3.6 and 3.7. However, for *non*-prismatic beams in homogeneous stress fields, distortion in the form of bending is never equal to zero (unless the component is *symmetrical*) since moment generated by the stress field is varying throughout the beam's length.

Figure 3.23: A structural bracket belonging to the AIRBUS A350 aircraft [41].

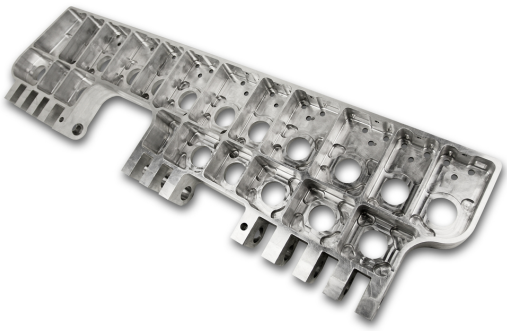


Figure 3.24: An engine aircraft component [42].

Chapter 4

A computational efficient method for distortion robustness analyses

As was elaborated upon in Chapter 1, state-of-the-art modeling tools have been developed that can predict distortion and robustness for aircraft components positioned in pre-stressed stock material. *Deterministic* distortion modeling tools have been developed, however have turned out to be meaningless due to the large variation in part distortion as a result of the stochastic nature of rolled plate residual stress; Low distortion deduced for a given piece of rolled plate might result in significant distortion for another piece of rolled plate. Consequently, efforts have been made to develop robustness prediction tools. However, these robustness prediction tools have proven to worsen the computational efficiency of Finite Element Analyses drastically. In this Chapter, state-of-the-art robustness modeling tools are evaluated and the reason for the computational inefficiency is elaborated upon. Subsequently, based upon the governing equations in the Finite Element Analyses, an improved - *computational efficient* - method for evaluating robustness is introduced. Throughout this Chapter, statements that are made will be supported by Finite Element Analyses concerning a meshed beam that is illustrated in Fig. 4.1.

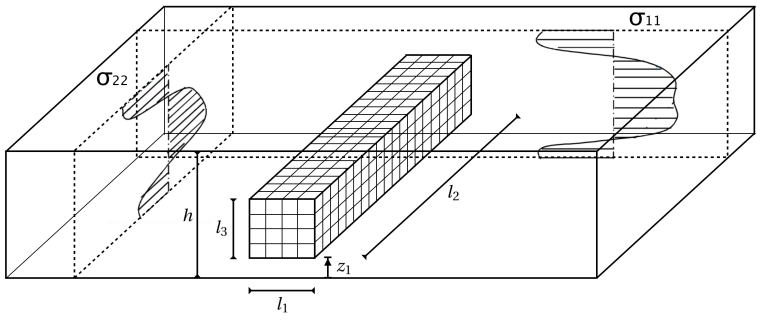


Figure 4.1: A meshed rectangular beam consisting of hexahedron elements which is positioned in stock material at offset $z_1 = 50\,\text{mm}$ and is subjected to a two-dimensional stress field of which the specifications were derived in Chapter 2.2. The beam has dimensions $l_1 = 50\,\text{mm}$, $l_2 = 1000\,\text{mm}$ and $l_3 = 50\,\text{mm}$. The thickness of stock material is $h = 300\,\text{mm}$. The beam has a Young's Modulus of $E = 70.07\,\text{GPa}$.

The accuracy of the Finite Element analyses is not our main concern herein, hence a coarse mesh will suffice. Fig. 4.2 shows the mesh including the chosen set of iso-static boundary

conditions and the centre node for which displacement in z-direction will be assessed as a measure for distortion.

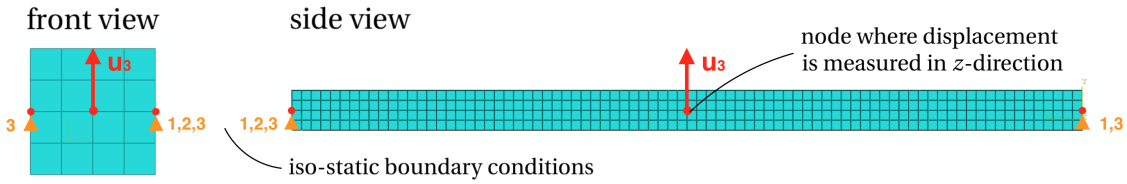


Figure 4.2: The meshed beam for which distortion and robustness will be evaluated. Rigid body modes are blocked through the iso-static boundary conditions. Distortion is measured as being the displacement in z-direction, u_3, of the beam's centre node.

The Finite Element simulations in this Chapter concern a beam that is positioned at an offset of $z_1 = 50\,\text{mm}$ (arbitrarily chosen) with respect to the rolled plate's bottom face. Furthermore, the beam is not rotated about any of the x-, y- and z-axes ($\alpha = 0\,\text{rad}, \beta = 0\,\text{rad}, \gamma = 0\,\text{rad}$).

4.1 Relation between residual stress and displacements in Finite Element Analyses

Before elaborating on the state of the art of distortion robustness evaluation methods, the governing equations relating displacements with residual stress are elaborated upon. The governing equations consist of three types [43] which will be formulated in the numerical discretized notation. The continuity equation relates the displacement field, $\boldsymbol{u}$, with strain, $\boldsymbol{\epsilon}$. The strain components are related to the displacements by the linear operation

$$\boldsymbol{\epsilon} = \boldsymbol{D}\boldsymbol{u}, \tag{4.1}$$

wherein $\boldsymbol{D}$ is itself a function of $\boldsymbol{u}$ if geometric linearity assumptions are not invoked. In general, geometric linearity assumptions are valid if the displacement gradient can be assumed small. Since in the context of aircraft component distortion, displacements as a result of residual stress are usually small with respect to component dimensions, geometric linearity will be assumed like was done in the analytical equation relating displacement to curvature in Eq. 3.10. The relation between strain, $\boldsymbol{\epsilon}$, and displacement, $\boldsymbol{u}$, is thus assumed linear as a result of which $\boldsymbol{D}$ does not depend on $\boldsymbol{u}$,

$$\boldsymbol{D} = \boldsymbol{D}[\boldsymbol{0}]. \tag{4.2}$$

The constitutive equation relates stress, $\boldsymbol{\sigma}$, with internal strain, $\boldsymbol{\epsilon}$. The residual stress matrix, which is denoted by $\boldsymbol{\sigma}^\dagger$, can be added as an extra term to the constitutive equation, like

$$\boldsymbol{\sigma} = \boldsymbol{S}\boldsymbol{\epsilon} + \boldsymbol{\sigma}^\dagger. \tag{4.3}$$

The equilibrium equation relates the external forces, $\boldsymbol{f}$, with the stresses, $\boldsymbol{\sigma}$, by

$$\boldsymbol{f} = \boldsymbol{D}^T\boldsymbol{\sigma}. \tag{4.4}$$

Substituting the continuity equation in the constitutive equation and subsequently the constitutive equation in the equilibrium equation, yields the total equation relating external forces, $\boldsymbol{f}$, with displacements, $\boldsymbol{u}$

$$\boldsymbol{f} = \boldsymbol{D}^T \left(\boldsymbol{S}\boldsymbol{D}\boldsymbol{u} + \boldsymbol{\sigma}^\dagger \right). \tag{4.5}$$

Residual stress fields are those stresses present in a body in the absence of external loads, as a result of which

$$\boldsymbol{f} = \boldsymbol{0} \tag{4.6}$$

holds. The stiffness matrix can be defined as

$$\boldsymbol{D}^T \boldsymbol{S}\boldsymbol{D} = \boldsymbol{K}. \tag{4.7}$$

Since, due to geometric linearity, $\boldsymbol{D}$ is independent of $\boldsymbol{u}$, $\boldsymbol{D}[\boldsymbol{0}]$, stiffness matrix $\boldsymbol{K}$ is independent of $\boldsymbol{u}$ as well, $\boldsymbol{K}[\boldsymbol{0}]$. Substituting Eq. 4.6 and Eq. 4.7 in the total equation in Eq. 4.5, yields

$$\boldsymbol{0} = \boldsymbol{K}[\boldsymbol{0}]\boldsymbol{u} + \boldsymbol{D}[\boldsymbol{0}]^T \boldsymbol{\sigma}^\dagger. \tag{4.8}$$

Since $\boldsymbol{K}$ and $\boldsymbol{D}$ do not depend on the displacement field $\boldsymbol{u}$, Eq. 4.8 can be rewritten to the static equilibrium equation relating the residual stress field, $\boldsymbol{\sigma}^\dagger$, to the displacement field, $\boldsymbol{u}$, by

$$\boldsymbol{K}\boldsymbol{u} = -\boldsymbol{D}^T \boldsymbol{\sigma}^\dagger. \tag{4.9}$$

This equation is solved during a Finite Element Analysis and solving this equation requires computational effort.

4.2 The residual stress matrix $\boldsymbol{\sigma}^\dagger$ as a function of translation (z_1) and rotation (α, β, γ) of the residual stress field

The aim of the Finite Element Analysis is to obtain the displacement field $\boldsymbol{u}$ as a function of a body's orientation in a two-dimensional stress field. The relationship between the displacement field $\boldsymbol{u}$ and residual stress matrix $\boldsymbol{\sigma}^\dagger$ has been derived in Section 4.1 and the equation which is to be solved through the Finite Element Analysis was given in Eq. 4.9. The residual stress matrix $\boldsymbol{\sigma}^\dagger$ forms the input for the Finite Element Analysis and the displacement field $\boldsymbol{u}$ is the output of the Finite Element Analysis. In this Chapter, the relationship between the residual stress matrix $\boldsymbol{\sigma}^\dagger$ that forms the input for the Finite Element Analysis and translation (z_1) and rotation (α, β, γ) of the stress field with respect to the mesh is determined.

In the Finite Element Analysis, orienting the body within a stress field can be simulated by orienting the stress field with respect to the mesh which is held fixed, such as is illustrated in Fig. 4.3. The orientation of the body within stock material can be described by six degrees of freedom or rigid body motions, being displacement of the body in three directions and rotation of the body about three axes. Since the stress field is homogeneous along the x- and

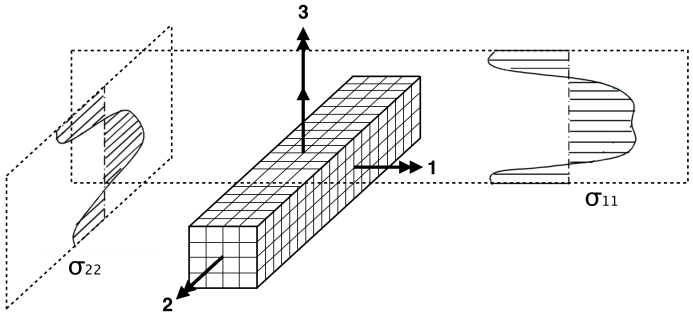


Figure 4.3: The meshed rectangular beam that was shown earlier in Fig. 4.1 which is positioned in stock material having a two-dimensional stress field. Positioning of the beam inside stock material is simulated by translating and rotating the stress field with respect to the (fixed) mesh. The positioning of the beam within stock material is determined via four degrees of freedom, being translation in z-direction and rotation about the x-, y- and z-axis.

y-direction, translating the body x- and y-direction has no effect on the stress mapping of the stress fields on the elements. Only translation in z-direction is relevant. In other words, the positioning of a body in stock material is determined by four degrees of freedom, being translation in z-direction and rotation about the x-, y- and z-axes.
Let's start with simulating translation in z-direction of a two-dimensional mesh with respect to a one-dimensional stress field, such as is illustrated in Fig. 4.4. Translation of the body in z-direction can be simulated by translating the stress field in z-direction with respect to the mesh.

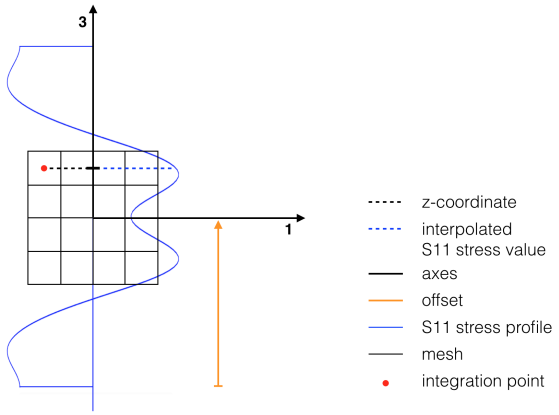


Figure 4.4: A two-dimensional mesh containing quadrilateral elements which is subjected to a one-dimensional stress field σ_{11}. Translating the mesh in $z-$direction with respect to the stress field is simulated by translating the stress field $z-$direction and interpolating the translated stress field based upon the z-coordinate of the element's integration points.

For each two-dimensional quadrilateral element having a single integration point in the centre, a stress tensor must be defined. This stress tensor [44] has three unique stress tensor components, being σ_{11}, σ_{13} and σ_{33}. As the stress field in Fig. 4.4 contains only σ_{11}, the stress tensor components σ_{13} and σ_{33} are equal to zero, like is stated below.

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{13} \\ \sigma_{31} & \sigma_{33} \end{bmatrix} = \begin{bmatrix} \sigma_{11} & 0 \\ 0 & 0 \end{bmatrix}.$$

$\sigma_{11}(z, z_1)$ depends only on the integration's point z-coordinate and the offset, z_1, of the stress field with respect to the mesh via the relationship that was obtained in Chapter 2, being

$$\sigma^{\dagger}_{11}(z, z_1) = \sum_{i=1}^{6} c^{\dagger}_{11,i} \cos\left(\frac{i2\pi}{h}\left(z + z_1\right)\right), \tag{4.10}$$

in which h is the height of the stock material and z_1 is the offset of the stress field with respect to the origin of the coordinate system, which in this example is positioned at the geometric centroid. Such as is illustrated in Fig. 4.4, for each element in the mesh, the stress tensor is determined based upon the z-coordinate of the integration point and the offset, z_1, by solving Eq. 4.10. Suppose that the stress field is rotated with an angle θ about the y-axis, such as is illustrated in Fig. 4.5.

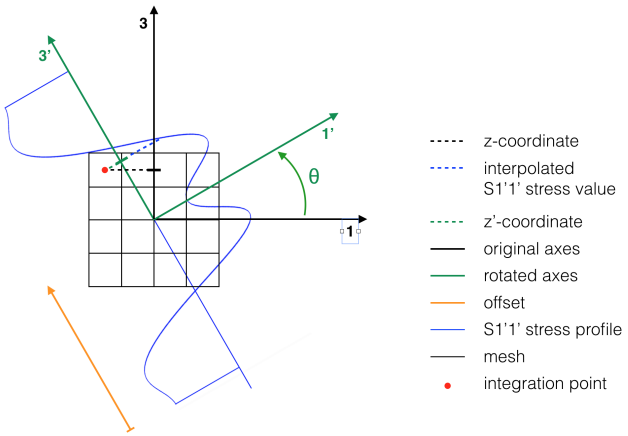


Figure 4.5: A two-dimensional mesh containing quadrilateral elements which is subjected to a rotated one-dimensional stress field σ_{11}. Rotation of the mesh about the $y-$axis with respect to the stress field is simulated by rotation of the stress field about the $y-$axis and interpolating the rotated stress field based upon the z'-coordinate of the element's integration points.

The coordinates of the element's integration points are known in the *untransformed* (i.e. reference) coordinate system of the mesh. The integration point's coordinates can be expressed in the *transformed* (i.e. rotated) coordinate system by making use of the so-called *coordinate transforms*. Coordinate transforms represent rotations of the coordinate system while the object is held constant by making use of the $\boldsymbol{Q}$ matrix [45]. The $\boldsymbol{Q}$ matrix for a two-dimensional system is equal to

$$\boldsymbol{Q} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{4.11}$$

The $\boldsymbol{Q}$ matrix describes the relationship between the coordinates of a point with respect to the reference coordinate system and the coordinates of the same point with respect to the transformed coordinate system. The relationship between a vector described in the reference coordinate system and the same vector in the transformed coordinate system is

$$\boldsymbol{v}' = \boldsymbol{Q}\boldsymbol{v}. \tag{4.12}$$

and in matrix notation is

$$\begin{bmatrix} v'_1 \\ v'_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

In this equation, $\boldsymbol{v}$ represents the coordinates of the integration point with respect to the reference coordinate system and $\boldsymbol{v}'$ the coordinates of the integration point with respect to the transformed coordinate system. From vector $\boldsymbol{v}'$, the z'-coordinate - i.e. v'_2 - can be used to obtain σ'_{11} via Eq. 4.10, which is the xx'-component of the stress tensor with respect to the transformed coordinate system, $\boldsymbol{\sigma}'$, like

$$\boldsymbol{\sigma}' = \begin{bmatrix} \sigma'_{11} & \sigma'_{13} \\ \sigma'_{31} & \sigma'_{33} \end{bmatrix} = \begin{bmatrix} \sigma'_{11} & 0 \\ 0 & 0 \end{bmatrix}.$$

The stress tensor of the integration point has been derived with respect to the transformed coordinate system, $\boldsymbol{\sigma}'$. However, the stress tensor of the integration point should be defined with respect to the untransformed coordinate system, $\boldsymbol{\sigma}$. In order to transform a stress tensor to a different coordinate system, a *tensor transformation* must be done. A stress tensor in the transformed coordinate system, $\boldsymbol{\sigma}'$, is obtained from a stress tensor in the undeformed coordinate system, $\boldsymbol{\sigma}$, by [45]

$$\boldsymbol{\sigma}' = \boldsymbol{Q}\boldsymbol{\sigma}\boldsymbol{Q}^T. \tag{4.13}$$

In this case, the stress tensor in the transformed coordinate system, $\boldsymbol{\sigma}'$, is known and the stress tensor in the untransformed coordinate system, $\boldsymbol{\sigma}$, is to be determined, like is illustrated in Fig. 4.6. The element with the integration point can be represented as an infinitesimal material point having a two-dimensional stress state.

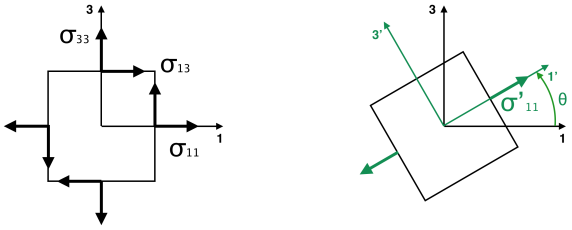


Figure 4.6: The two-dimensional stress state of an infinitesimal material point in the transformed coordinate system (right) and in the untransformed coordinate system (left). In order to obtain the stress tensor in the untransformed coordinate system, a tensor back transformation must be done.

When pre-multiplying both sides with $\boldsymbol{Q}^T$ and post-multiplying both sides with $\boldsymbol{Q}$ and making use of the fact that $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$ by which $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$, the *tensor back transformation* relation is obtained [45], which is

$$\boldsymbol{\sigma} = \boldsymbol{Q}^T\boldsymbol{\sigma}'\boldsymbol{Q}. \tag{4.14}$$

Up to this point a two-dimensional situation was considered. Both translation in z-direction and rotation about the y-axis of the stress field with respect to the mesh was simulated. The next step is to consider a three-dimensional situation in which the stress field can be rotated

about three axes. The second stress direction σ_{22} will be taken into account as well.

In the three-dimensional situation, three *basic* coordinate transforms are formulated. A basic coordinate transform describes the relationship between a transformed and an untransformed coordinate system which is rotated about one of three axes. The three basic coordinate transform matrices are [45]

$$\boldsymbol{Q}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}, \tag{4.15}$$

$$\boldsymbol{Q}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & & \cos\beta \end{bmatrix} \tag{4.16}$$

and

$$\boldsymbol{Q}_z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.17}$$

where α, β and γ embody the Euler angles about the x-, y- and z-axes, respectively. Then, the *general* coordinate transform can be obtained from these three basic coordinate transform matrices by using matrix multiplication. The general coordinate transform describes the relationship between the coordinates of a point with respect to the reference coordinate system and the coordinates of the same point with respect to the transformed coordinate system which is rotated about three axes simultaneously and is given by

$$\boldsymbol{Q}(\alpha,\beta,\gamma) = \boldsymbol{Q}_z(\gamma)\boldsymbol{Q}_y(\beta)\boldsymbol{Q}_x(\alpha). \tag{4.18}$$

With the use of the general coordinate transform that is stated in Eq. 4.18, the stress tensors for all elements in the mesh as result of rotation of the stress field about three axes α, β and γ can be determined. In Fig. 4.7, the stress tensor components of a material point are shown in the transformed and in the untransformed configuration.

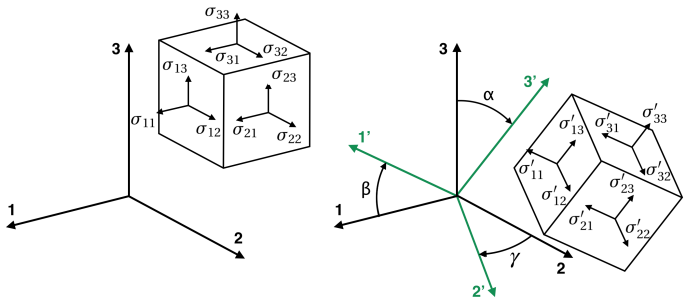


Figure 4.7: The three-dimensional stress state of an infinitesimal material point in the transformed coordinate system (right) and in the untransformed coordinate system (left). In order to obtain the stress tensor in the untransformed coordinate system, a tensor back transformation must be done.

The procedure for obtaining the stress tensors with respect to the untransformed configuration for all elements in the mesh remains equal to the procedure described in the two-dimensional stress state scenario and is summarized.

1. The rotations α, β, γ and the offset z_1 of the stress field with respect to the mesh are determined.

2. The basic coordinate transforms $\boldsymbol{Q}_z(\gamma)$, $\boldsymbol{Q}_y(\beta)$ and $\boldsymbol{Q}_x(\alpha)$ are determined based upon rotations α, β, γ, by using Eq. 4.15, 4.16 and 4.17. Subsequently, the general coordinate transform $\boldsymbol{Q}(\alpha,\beta,\gamma)$ is obtained by using Eq. 4.18.

3. The coordinates of the element's integration points with respect to the *untransformed* coordinate system are stored in vector

$$\boldsymbol{v}(x,y,z) = \begin{bmatrix} v_1(x) \\ v_2(y) \\ v_3(z) \end{bmatrix}.$$

4. A coordinate transform is performed in order to express the integration point's coordinates with respect to the *transformed* coordinate system, $\boldsymbol{v}'$, by using Eq. 4.12, which is

$$\boldsymbol{v}'(x,y,z,\alpha,\beta,\gamma) = \boldsymbol{Q}(\alpha,\beta,\gamma)\,\boldsymbol{v}(x,y,z),$$

   in which $\boldsymbol{Q}(\alpha,\beta,\gamma)$ is the general coordinate transform which was obtained in step 2.

5. The stress components σ'_{11} and σ'_{22} of the transformed stress tensor $\boldsymbol{\sigma}'$ are determined by substituting the z'-coordinate of $\boldsymbol{v}'(x,y,z,\alpha,\beta,\gamma)$ - i.e. $v'_3(x,y,z,\alpha,\beta,\gamma)$ - into Eq. 4.19 and Eq. 4.20.

$$\sigma^{\dagger}_{11}(x,y,z,z_1,\alpha,\beta,\gamma) = \sum_{i=1}^{6} c^{\dagger}_{11,i}\phi_i(x,y,z,z_1,\alpha,\beta,\gamma). \tag{4.19}$$

$$\sigma^{\dagger}_{22}(x,y,z,z_1,\alpha,\beta,\gamma) = \sum_{i=1}^{6} c^{\dagger}_{22,i}\phi_i(x,y,z,z_1,\alpha,\beta,\gamma). \tag{4.20}$$

   In these equations,

$$\phi_i(x,y,z,z_1,\alpha,\beta,\gamma) = \cos\left(\frac{i2\pi}{h}\left(v'_3(x,y,z,\alpha,\beta,\gamma) + z_1\right)\right). \tag{4.21}$$

6. The integration point's stress tensor in the transformed configuration $\boldsymbol{\sigma}'$ is determined

$$\boldsymbol{\sigma}' = \begin{bmatrix} \sigma'_{11} & \sigma'_{12} & \sigma'_{13} \\ \sigma'_{21} & \sigma'_{22} & \sigma'_{23} \\ \sigma'_{31} & \sigma'_{32} & \sigma'_{33} \end{bmatrix} = \begin{bmatrix} \sigma^{\dagger}_{11}(x,y,z,z_1,\alpha,\beta,\gamma) & 0 & 0 \\ 0 & \sigma^{\dagger}_{22}(x,y,z,z_1,\alpha,\beta,\gamma) & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

7. A stress tensor back transformation is performed in order to obtain the integration point's stress tensor in the untransformed configuration $\boldsymbol{\sigma}$

$$\boldsymbol{\sigma} = \boldsymbol{Q}^T\boldsymbol{\sigma}'\boldsymbol{Q}.$$

This procedure is repeated for each element in the mesh. The final matrix consisting all stress tensors for each element in the mesh is the residual stress matrix $\boldsymbol{\sigma}^{\dagger}$. This matrix

forms the input for the Finite Element Analyses.

Now that the relation between the residual stress matrix $\boldsymbol{\sigma}^{\dagger}$ and the orientation of the stress fields with respect to the mesh has been determined, a *deterministic* static analysis can be done concerning the elementary beam shown in Fig. 4.1 with the mesh which was shown in Fig. 4.2. In this static analysis, the stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$ for $i = 1 \cdots 6$ will be

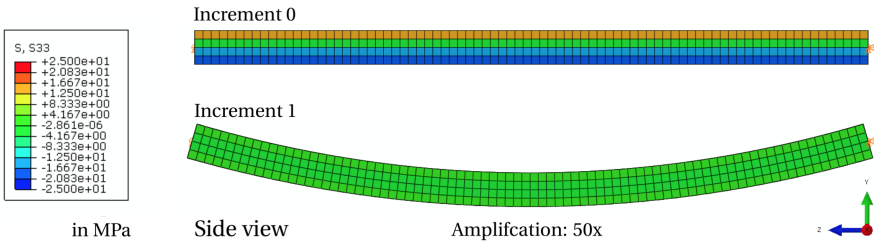


Figure 4.8: The mesh of the elementary beam before the static analysis (increment 0) and after the static analysis (increment 1).

assumed to be *deterministic* by nature, as a result of which the obtained displacement field $\boldsymbol{u}$ is deterministic by nature as well. Fig. 4.8 shows the undeformed beam before the static analysis (increment 0) and the deformed beam after the static analysis (increment 1). The static analysis was performed in Abaqus. The implementation of the stress field mapping and the deterministic static analysis in Abaqus and Python is elaborated in Appendix C.1. As can be observed, the stress profile has fully relaxed as a result of stress relaxation.
The outcome of the deterministic static analysis is listed in Tab. 4.1.

Table 4.1: An overview of the results of the static analysis that are relevant. The static analysis concerns an elementary beam subjected to a *deterministic* stress field having an offset z_1 with respect to the mesh, such as is illustrated in Fig. C.1.

Outcome of deterministic static analysis	Symbol	Value
Centre node displacement	u_3	$-1.442\,096\,\text{mm}$
Number of static analyses required	m	1
Computation time	t	$\approx 13\,\text{sec}$

The value for distortion of the centre node obtained with a deterministic static analyses will be compared to the mean value obtained using the state-of-the-art method and the improved method for robustness evaluation obtained in Section 4.3 and 4.4, respectively. Furthermore, the computational efficiency of the state-of-the-art method will be compared with the computational efficiency of the improved method, by inspecting the number of static analyses required, m, and the computation time, t.

4.3 State-of-the-art methods for robustness analyses

State-of-the-art methodologies for evaluating distortion robustness struggle with computational efficiency. A frequently used method to obtain stochastic information on distortion - i.e. robustness - is the *Monte Carlo* method, also referred to by *Monte Carlo* exper-

iments [10, 11]. The Monte Carlo method is used when one wants to know the stochastics of an unknown output distribution that is a function of an input of which the distribution is known. One resorts to Monte Carlo experiments when the relation between the output distribution and input distribution cannot be established analytically. The Monte Carlo method relies on repeated random sampling of the input distribution and calculating the output as a function of the random samples in order to obtain numerical results for the output distribution. By the law of large numbers, the numerically obtained distribution converges to the analytical output distribution by increasing sample size m. Since the output distribution is obtained by propagating random samples with a large sample size, this method is considered to be a rough method. One can imagine that the Monte Carlo method is a highly inefficient approach since a great deal of Finite Element Analyses are needed to obtain stochastic information on the distortion distribution, which is extremely costly. Often concessions are made to the sample size m in order to reduce computational costs. This however weakens the reliability of the obtained distribution since a substantial sample size m is needed to approach the analytical output distribution. The Monte Carlo method applied to obtaining information about the distortion distribution is illustrated in Fig. 4.9.

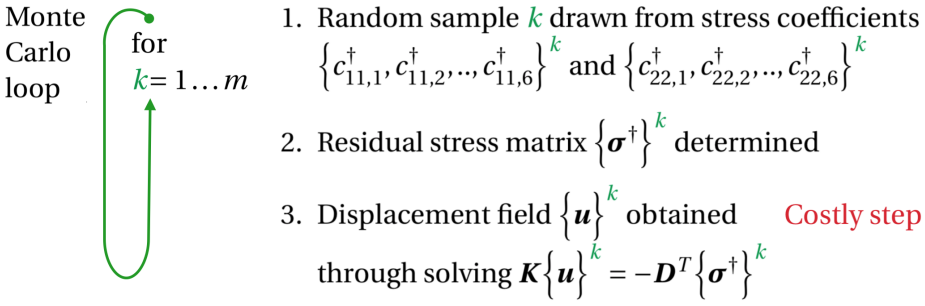


Figure 4.9: The Monte Carlo loop used for obtaining the distortion distribution. For each iteration, a static analysis must be performed which is computationally costly. If sample size m is large enough, the numerically obtained output distribution converges - by the law of large numbers - to the analytical distortion distribution.

At the end of the Monte Carlo experiments, m displacement fields $\{\boldsymbol{u}\}^1 \cdots \{\boldsymbol{u}\}^m$ are obtained, in which $\{\boldsymbol{u}\}^k$ is a vector containing displacements for each node in the mesh. For each node in the mesh, m displacements are obtained which form a displacement distribution for each node. The mean and standard deviation of this displacement distribution can subsequently be obtained by applying Eq. 2.8 and 2.9, which were used in Section 2.2 for obtaining the distribution the stress coefficients. If the sample size m is chosen large enough, the obtained displacement distribution approaches the analytical displacement distribution. The individual steps in the Monte Carlo experiments shown in Fig. 4.9 are elaborated below.

1. A random sample set $\{c^{\dagger}_{11,1}, c^{\dagger}_{11,2}, .., c^{\dagger}_{11,6}\}^k$ and $\{c^{\dagger}_{22,1}, c^{\dagger}_{22,2}, .., c^{\dagger}_{22,6}\}^k$ is drawn from the normally distributed fit coefficients $c^{\dagger}_{11,i} \sim \mathcal{N}(\mu_{c^{\dagger}_{11,i}}, s_{c^{\dagger}_{11,i}})$ and $c^{\dagger}_{22,i} \sim \mathcal{N}(\mu_{c^{\dagger}_{22,i}}, s_{c^{\dagger}_{22,i}})$, like is illustrated in Fig. 4.10. The stochastic parameters for the normally distributed fit coefficients were determined in Section 2.2.

2. Based upon the random sample set of stress coefficients $\{c^{\dagger}_{11,1}, c^{\dagger}_{11,2}, .., c^{\dagger}_{11,6}\}^k$ and $\{c^{\dagger}_{22,1}, c^{\dagger}_{22,2}, .., c^{\dagger}_{22,6}\}^k$, which were determined in the previous step, and a mapping of

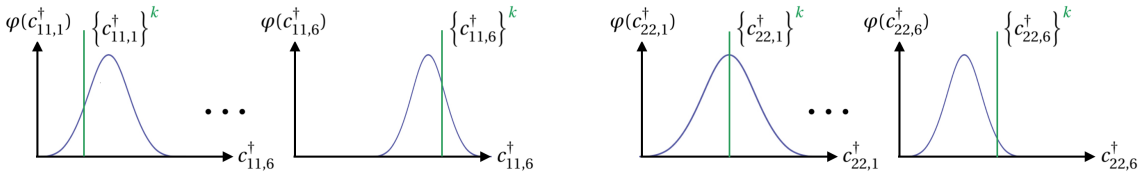


Figure 4.10: This figure illustrates that a random sample set k is drawn from the normally distributed stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$

the rotated (α, β and γ) and translated (z_1) stress fields to the elements in the mesh, the residual stress matrix $\{\boldsymbol{\sigma}^{\dagger}\}^k$ is determined by employing the methodology that was elaborated upon in Section 4.2. The residual stress matrix $\{\boldsymbol{\sigma}^{\dagger}\}^k$ includes stress tensors of all elements in the mesh of which the individual stress tensor components were related to the stress coefficients by

$$\{\sigma^{\dagger}_{11}(x,y,z,z_1,\alpha,\beta,\gamma)\}^k = \sum_{i=1}^{6} \{c^{\dagger}_{11,i}\}^k \phi_i(x,y,z,z_1,\alpha,\beta,\gamma)$$

and by

$$\{\sigma^{\dagger}_{22}(x,y,z,z_1,\alpha,\beta,\gamma)\}^k = \sum_{i=1}^{6} \{c^{\dagger}_{22,i}\}^k \phi_i(x,y,z,z_1,\alpha,\beta,\gamma)$$

The residual stress matrix $\{\boldsymbol{\sigma}^{\dagger}\}^k$ forms the input for the Finite Element Analysis.

3. A static analysis is executed in Abaqus in order to obtain the displacement field $\{\boldsymbol{u}\}^k$. The input for the static analysis is the residual stress matrix $\{\boldsymbol{\sigma}^{\dagger}\}^k$ which was obtained in the previous step. As was elaborated upon in Section 4.1, the displacement field $\{\boldsymbol{u}\}^k$ is obtained through Finite Element Analyses by solving the equation

$$\boldsymbol{K}\{\boldsymbol{u}\}^k = -\boldsymbol{D}^T\{\boldsymbol{\sigma}^{\dagger}\}^k.$$

The obvious downside of applying the state-of-the-art method is that a large sample size m is necessary in order to reliably describe the distortion distribution. In practice, a sample size of $m > 1000$ is needed to obtain reliable results. Since for each sample a separate static analysis must be executed, this method is very costly in terms of computational effort. In practice, often concessions are made to the sample size, weakening the reliability of the robustness analysis.

Since the relation that is solved by Finite Element Analyses in step 3 has a linear appearance, the question rises why the state of the art resorts to the expensive Monte Carlo experiments at all. This is since often *geometric nonlinearity* is assumed, as a result of which $\boldsymbol{D}$ does depend on displacement field $\boldsymbol{u}$, $\boldsymbol{D}[\boldsymbol{u}]$, by which stiffness matrix $\boldsymbol{K}$ does depend on $\boldsymbol{u}$ as well, $\boldsymbol{K}[\boldsymbol{u}]$. When assuming geometric nonlinearity, the relation in step 3 does become nonlinear. Furthermore, the state of the art resorts to Monte Carlo experiments since no efforts have been made to establish the relation between a stress field having multiple stochastic input variables and a single stochastic output *analytically*. The improved method in Section 4.4 will show that a single output distribution depending on multiple input distributions can be

determined analytically.

The state-of-the-art methodology for evaluating robustness, which was elaborated upon in this Chapter, will be employed for the elementary beam that was shown in Fig. 4.1. Fig. 4.11 visualizes the outcome of the state-of-the-art method in a scatter plot (left) and a histogram (right). The scatter plot shows the centre node displacement for each Monte Carlo iteration and the histogram visualizes the occurrence of the centre mode displacement, which appears to be similar to the histogram of a normal distribution.

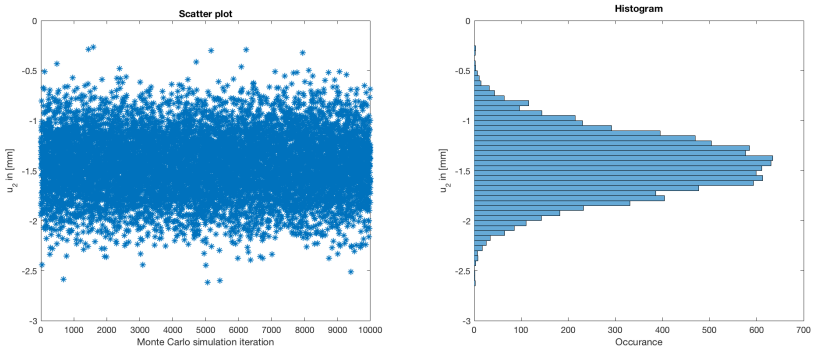


Figure 4.11: A scatter plot (left) of the centre node (node 1038) displacement u_3 per Monte Carlo iteration and a histogram (right) of the occurance of the centre node displacement.

The implementation of the state-of-the-art method in Abaqus and Python is elaborated in Appendix C.2. Tab. 4.2 summarizes the outcome of the state-of-the-art method for obtaining distortion robustness of the elementary beam that was shown in Fig. 4.1.

Table 4.2: An overview of the outcome of the *state-of-the-art* method employed for evaluating distortion robustness of the elementary beam that was shown in 4.1 and that was subjected to a stochastic stress field as was specified in Section 2.2.

Outcome of the state-of-the-art method	Symbol	Value
Mean value of centre node displacement	μ_{u_3}	$-1.445\,049\,\text{mm}$
Standard deviation of centre node displacement	s_{u_3}	$3.078\,326 \times 10^{-1}\,\text{mm}$
Number of static analyses required	m	$10\,000$
Computation time	t	$\approx 85\,157\,\text{sec}$

As can be concluded, the state-of-the-art method is highly costly in computational effort, since a great number of static analyses were required to evaluate distortion robustness. 10000 static analyses were performed in order to achieve high reliability. The mean value of the centre node displacement, μ_{u_3}, is approximately equal to the centre node displacement obtained in the deterministic static analysis of which the outcome was listed in Tab. 4.1.

4.4 Improved method for robustness analyses

It can be concluded that the state-of-the-art method for evaluating distortion robustness which was elaborated upon in Section 4.3 is highly inefficient in terms of computational effort. Particularly considering the complex Finite Element geometries and large number of degrees of freedom encountered in practice. Considering the fact that linearity assumptions are invoked in the governing equations, as was substantiated in Section 4.1, distortion robustness can be evaluated whilst requiring relatively little computational effort.

Let's start by recalling Eq. 4.9 which was derived in Section 4.1 and describes the relationship between the displacement field, $\boldsymbol{u}$, and the residual stress matrix, $\boldsymbol{\sigma}^{\dagger}$, which was

$$\boldsymbol{K}\boldsymbol{u} = -\boldsymbol{D}^{T}\boldsymbol{\sigma}^{\dagger}.$$

For explanation purposes, an elementary mesh will be considered consisting of a *single element* having a single integration point. Furthermore, a stress field is considered which is not rotated about any of the three axes with respect to the mesh ($\alpha = 0, \beta = 0, \gamma = 0$). The statements derived in this Chapter will also hold for meshes containing multiple elements and for stress fields that are rotated with respect to the mesh. Considering these assumptions, the residual stress tensor, $\boldsymbol{\sigma}^{\dagger}$, is constituted as

$$\boldsymbol{\sigma}^{\dagger} = \begin{bmatrix} \sigma^{\dagger}_{11} & \sigma^{\dagger}_{12} & \sigma^{\dagger}_{13} \\ \sigma^{\dagger}_{21} & \sigma^{\dagger}_{22} & \sigma^{\dagger}_{23} \\ \sigma^{\dagger}_{31} & \sigma^{\dagger}_{32} & \sigma^{\dagger}_{33} \end{bmatrix} = \begin{bmatrix} \sigma^{\dagger}_{11}(x,y,z,z_1) & 0 & 0 \\ 0 & \sigma^{\dagger}_{22}(x,y,z,z_1) & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

where the stress tensor components $\sigma^{\dagger}_{11}(x,y,z,z_1)$ and $\sigma^{\dagger}_{22}(x,y,z,z_1)$ depend on the coordinates of the integration point (x,y,z) and the offset of the stress field with respect to the mesh (z_1), as was stated in Eq. 4.19 and in Eq. 4.20, respectively and as is repeated below.

$$\sigma^{\dagger}_{11}(x,y,z,z_1) = \sum_{i=1}^{6} c^{\dagger}_{11,i}\phi_i(x,y,z,z_1).$$

$$\sigma^{\dagger}_{22}(x,y,z,z_1) = \sum_{i=1}^{6} c^{\dagger}_{22,i}\phi_i(x,y,z,z_1).$$

In these equations, $\phi_i(x,y,z,z_1)$ was stated to be (Eq. 4.21)

$$\phi_i(x,y,z,z_1) = \cos\left(\frac{i2\pi}{h}\Big(z+z_1\Big)\right).$$

$\boldsymbol{\sigma}^{\dagger}$ can be rewritten into two terms, ${\boldsymbol{\sigma}^{\dagger}}_{11}$ and ${\boldsymbol{\sigma}^{\dagger}}_{22}$, like

$$\boldsymbol{\sigma}^{\dagger} = {\boldsymbol{\sigma}^{\dagger}}_{11} + {\boldsymbol{\sigma}^{\dagger}}_{22} = \begin{bmatrix} \sigma^{\dagger}_{11}(x,y,z,z_1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma^{\dagger}_{22}(x,y,z,z_1) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.22}$$

Substituting Eq. 4.19 and Eq. 4.20 for $\sigma^{\dagger}_{11}(x,y,z,z_1)$ and $\sigma^{\dagger}_{22}(x,y,z,z_1)$, respectively, yields

$${\boldsymbol{\sigma}^{\dagger}}_{11} = \sum_{i=1}^{6} c^{\dagger}_{11,i} \begin{bmatrix} \phi_i(x,y,z,z_1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \qquad {\boldsymbol{\sigma}^{\dagger}}_{22} = \sum_{i=1}^{6} c^{\dagger}_{22,i} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \phi_i(x,y,z,z_1) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.23}$$

For the sake of convenience, *individual stress mode* matrices $\boldsymbol{\phi}_{11,i}$ and $\boldsymbol{\phi}_{22,i}$ are defined for $i = 1 \cdots 6$. $\boldsymbol{\phi}_{11,i}$ and $\boldsymbol{\phi}_{22,i}$ are independent of the stochastic stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$, as a result of which these matrices are deterministic by nature. These individual stress mode matrices are stated below.

$$\boldsymbol{\phi}_{11,i} = \begin{bmatrix} \phi_i(x,y,z,z_1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \qquad \boldsymbol{\phi}_{22,i} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \phi_i(x,y,z,z_1) & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.24}$$

Substituting $\boldsymbol{\phi}_{11,i}$ and $\boldsymbol{\phi}_{22,i}$ in Eq. 4.23 and ${\boldsymbol{\sigma}^{\dagger}}_{11}$ and subsequently ${\boldsymbol{\sigma}^{\dagger}}_{22}$ in Eq. 4.22, yields

$$\boldsymbol{\sigma}^{\dagger} = \sum_{i=1}^{6} c^{\dagger}_{11,i} \boldsymbol{\phi}_{11,i} + \sum_{i=1}^{6} c^{\dagger}_{22,i} \boldsymbol{\phi}_{22,i}. \tag{4.25}$$

Substituting the obtained expression for $\boldsymbol{\sigma}^{\dagger}$ into the equation for static equilibrium in Eq. 4.9, which was derived in Chapter 4.1, yields

$$\begin{aligned} \boldsymbol{K}\boldsymbol{u} &= -\boldsymbol{D}^T \boldsymbol{\sigma}^{\dagger} \\ &= -\boldsymbol{D}^T \left(\sum_{i=1}^{6} c^{\dagger}_{11,i} \boldsymbol{\phi}_{11,i} + \sum_{i=1}^{6} c^{\dagger}_{22,i} \boldsymbol{\phi}_{22,i} \right). \end{aligned} \tag{4.26}$$

This expression can be rewritten to

$$\boldsymbol{K}\boldsymbol{u} = \sum_{i=1}^{6} c^{\dagger}_{11,i} \left(-\boldsymbol{D}^T \boldsymbol{\phi}_{11,i} \right) + \sum_{i=1}^{6} c^{\dagger}_{22,i} \left(-\boldsymbol{D}^T \boldsymbol{\phi}_{22,i} \right). \tag{4.27}$$

Twelve *individual mode* displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ can be defined for $i = 1 \cdots 6$ that are independent of the stochastic stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$, respectively. These displacements fields are deterministic by nature and are obtained through solving the equations below.

$$\begin{aligned} \boldsymbol{K}\boldsymbol{U}_{11,i} &= -\boldsymbol{D}^T \boldsymbol{\phi}_{11,i}. \\ \boldsymbol{K}\boldsymbol{U}_{22,i} &= -\boldsymbol{D}^T \boldsymbol{\phi}_{22,i}. \end{aligned} \tag{4.28}$$

Substituting the expressions obtained in Eq. 4.28 into the equation in Eq. 4.27, yields

$$\boldsymbol{K}\boldsymbol{u} = \sum_{i=1}^{6} c^{\dagger}_{11,i} \left(\boldsymbol{K}\boldsymbol{U}_{11,i} \right) + \sum_{i=1}^{6} c^{\dagger}_{22,i} \left(\boldsymbol{K}\boldsymbol{U}_{22,i} \right).$$

Simplifying this expression, leads to the relation between the individual mode displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ that do not depend on the stochastic stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$, and the *total* displacement field $\boldsymbol{u}$ that does depend on the stochastic stress coefficients, with

$$\boldsymbol{u} = \sum_{i=1}^{6} c^{\dagger}_{11,i} \boldsymbol{U}_{11,i} + \sum_{i=1}^{6} c^{\dagger}_{22,i} \boldsymbol{U}_{22,i}. \tag{4.29}$$

The purpose of rewriting expression 4.9 to expression 4.29 is that displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ have been defined that are deterministic by nature. The stochastics of the total

displacement field $\boldsymbol{u}$ can subsequently be determined through a linear combination of the deterministic displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ and the stochastic stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$, like is formulated in Eq. 4.29.

Via Eq. 4.29, the parameters describing the stochastics, being the mean and standard deviation, of the total displacement field $\boldsymbol{u}$ can be derived *analytically*. Probability theory [40] states that a function containing a summation of normally distributed parameters itself has a normal distribution as well. Consider a function $f(x)$ which is a function of variable x which is normally distributed and constant a which is deterministic by nature, like

$$f(x) = ax \qquad \text{with } x \sim \mathcal{N}\left(\mu_x, s_x\right). \tag{4.30}$$

As was agreed upon in Chapter 2.2, the symbols μ and s are used for the mean and standard deviation, respectively. The mean, μ_f, and standard deviation, s_f, describing the normal distribution of function $f(x)$, are subsequently obtained through solving the equations below.

$$\mu_f = a\mu_x \tag{4.31}$$

$$s_f = as_x \tag{4.32}$$

Now, consider a function $g(x_1, x_2, x_3)$ which is a function of variables x_1, x_2 and x_3 which are normally distributed and constants a, b and c which are deterministic by nature, like

$$\begin{aligned} g(x_1, x_2, x_3) = ax_1 + bx_2 + cx_3 \qquad & \text{with } x_1 \sim \mathcal{N}\left(\mu_{x_1}, s_{x_1}\right) \\ & x_2 \sim \mathcal{N}\left(\mu_{x_2}, s_{x_2}\right) \\ & x_3 \sim \mathcal{N}\left(\mu_{x_3}, s_{x_3}\right). \end{aligned} \tag{4.33}$$

This time, the mean, μ_g, and standard deviation, s_g, describing the normal distribution of function $g(x_1, x_2, x_3)$, are obtained through solving the equations below.

$$\mu_g = a\mu_{x_1} + b\mu_{x_2} + c\mu_{x_3}. \tag{4.34}$$

$$s_g = \sqrt{\left(as_{x_1}\right)^2 + \left(bs_{x_2}\right)^2 + \left(cs_{x_3}\right)^2}. \tag{4.35}$$

In the same manner, based upon Eq. 4.29, the mean and standard deviation, $\boldsymbol{\mu_u}$ and $\boldsymbol{s_u}$ respectively, of the total displacement field $\boldsymbol{u}$ can be derived analytically. The normal distributions of stress coefficients $c^{\dagger}_{11,i} \sim \mathcal{N}\left(\mu_{c^{\dagger}_{11,i}}, s_{c^{\dagger}_{11,i}}\right)$ and $c^{\dagger}_{22,i} \sim \mathcal{N}\left(\mu_{c^{\dagger}_{22,i}}, s_{c^{\dagger}_{22,i}}\right)$ are known. This time, the mean and standard deviation, $\boldsymbol{\mu_u}$ and $\boldsymbol{s_u}$ respectively, are in vector form since they describe the normal distributions of the displacements for all nodes in the mesh. The mean and standard deviation, $\boldsymbol{\mu_u}$ and $\boldsymbol{s_u}$ respectively, of the normal distributions of the displacements for each node in the mesh can be determined through solving the equations below.

$$\boldsymbol{\mu_u} = \sum_{i=1}^{6} \mu_{c^{\dagger}_{11,i}} \boldsymbol{U}_{11,i} + \sum_{i=1}^{6} \mu_{c^{\dagger}_{22,i}} \boldsymbol{U}_{22,i}. \tag{4.36}$$

$$\boldsymbol{s_u} = \sqrt{\sum_{i=1}^{6} \left(s_{c^{\dagger}_{11,i}} \boldsymbol{U}_{11,i}\right)^2 + \sum_{i=1}^{6} \left(s_{c^{\dagger}_{22,i}} \boldsymbol{U}_{33,i}\right)^2}. \tag{4.37}$$

The methodology developed in this Chapter for obtaining the distortion distribution in a computational efficient manner can be summarized as:

1. Twelve static analyses are executed in Abaqus in order to obtain the twelve *individual mode* displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ for $i = 1 \cdots 6$. The inputs for the static analyses are the individual mode matrices $\boldsymbol{\phi}_{11,i}$ and $\boldsymbol{\phi}_{22,i}$ for $i = 1 \cdots 6$, which were formulated in Eq. 4.24. The individual mode displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ are obtained through Finite Element Analyses by solving the equations below.

$$\begin{aligned} \boldsymbol{K}\boldsymbol{U}_{11,i} &= -\boldsymbol{D}^T\boldsymbol{\phi}_{11,i}, \\ \boldsymbol{K}\boldsymbol{U}_{22,i} &= -\boldsymbol{D}^T\boldsymbol{\phi}_{22,i}, \qquad\qquad \text{for } i = 1 \cdots 6. \end{aligned}$$

2. The two parameters describing the distortion distribution of the *total* displacement field $\boldsymbol{u}$, being the mean and standard deviation, $\boldsymbol{\mu_u}$ and $\boldsymbol{s_u}$ respectively, are *analytically* determined through solving the equations stated in Eq. 4.36 and Eq. 4.37 and repeated below. The mean and standard deviations of the stress coefficients $c^{\dagger}_{11,i} \sim \mathcal{N}\left(\mu_{c^{\dagger}_{11,i}}, s_{c^{\dagger}_{11,i}}\right)$ and $c^{\dagger}_{22,i} \sim \mathcal{N}\left(\mu_{c^{\dagger}_{22,i}}, s_{c^{\dagger}_{22,i}}\right)$ for $i = 1 \cdots 6$ are known and were stated in Tab. 2.1.

$$\boldsymbol{\mu_u} = \sum_{i=1}^{6} \mu_{c^{\dagger}_{11,i}} \boldsymbol{U}_{11,i} + \sum_{i=1}^{6} \mu_{c^{\dagger}_{22,i}} \boldsymbol{U}_{22,i}.$$

$$\boldsymbol{s_u} = \sqrt{\sum_{i=1}^{6} \left(s_{c^{\dagger}_{11,i}} \boldsymbol{U}_{11,i}\right)^2 + \sum_{i=1}^{6} \left(s_{c^{\dagger}_{22,i}} \boldsymbol{U}_{33,i}\right)^2}.$$

   The obtained mean and standard deviation of the distortion distribution, $\boldsymbol{\mu_u}$ and $\boldsymbol{s_u}$ respectively, are vectors in which the distortion distribution for each node in the mesh is stored.

The developed methodology has substantially improved the state-of-the-art methodology for evaluating distortion robustness. With the developed methodology, $m = 12$ static analyses suffice and the distortion robustness can be derived *analytically*. With the state-of-the-art methodology, distortion robustness is evaluated *numerically* and more than $m > 1000$ static analyses were needed to obtain a reliable evaluation of distortion robustness. Fewer than $m < 1000$ static analyses would weaken the reliability of the obtained distribution. With the developed methodology, significant simulation time can be spared since robustness can be evaluated in a computational efficient manner.

The improved methodology for evaluating robustness, which was elaborated upon in this Chapter, will be employed for the elementary beam that was shown in Fig. 4.1. The implementation of the improved method in Abaqus and Python is elaborated in Appendix C.3. Tab. 4.3 summarizes the outcome of the improved method for evaluating distortion robustness.

Table 4.3: This table provides an overview of the outcome of the *improved* method employed for evaluating distortion robustness of the elementary beam that was shown in 4.1 and that was subjected to a stochastic stress field as was specified in Chapter 2.2.

Outcome of the improved method	Symbol	Value
Mean value of centre node displacement	μ_{u_3}	$-1.442\,096\,\text{mm}$
Standard deviation of centre node displacement	s_{u_3}	$3.078\,827 \times 10^{-1}\,\text{mm}$
Number of static analyses required	m	12
Computation time	t	$\approx 76\,\text{sec}$

The mean value for distortion, μ_{u_3}, and the standard deviation of distortion, s_{u_3}, obtained with the improved method is approximately equal to the values obtained with the state-of-the-art method which were listed in Tab. 4.2. Furthermore, it can be concluded that the improved method for evaluating robustness significantly reduces computational effort. Where $m = 10\,000$ static analyses were needed with the state-of-the-art method to reliably evaluate the distortion robustness, only $m = 12$ static analyses are needed for evaluating robustness with the improved method.

Chapter 5

Case study

Now that an improved - *computational efficient* - method for predicting distortion robustness has been validated for the elementary beam that was considered in Chapter 4, the improved method can be applied to a case study that concerns a *real*, i.e. characteristic, aircraft component. In this case study, distortion robustness is to be predicted for a *stiffener* component that is manufactured from a piece of rolled plate, such as is illustrated in Fig. 5.1. The piece of rolled plate is subjected to a two-dimensional stress field that is stochastic by nature. The stochastics of the stress fields in the piece of rolled plate will be considered to be equal to the stochastics derived in Chapter 2.2 which were based on experimental measurements and are summarized in Tab. 2.1.

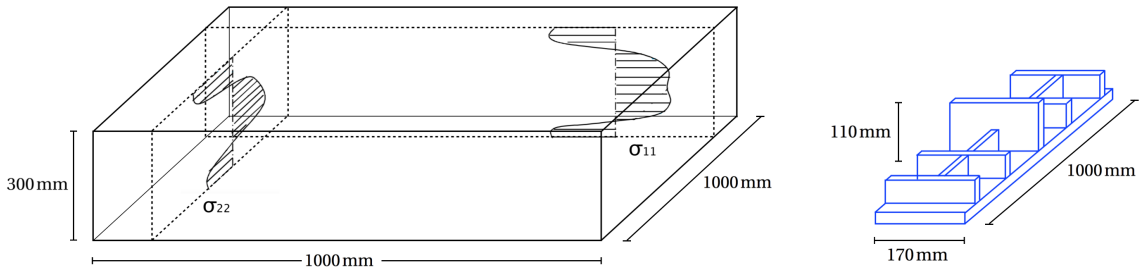


Figure 5.1: A piece of rolled plate stock material (left) from which a *stiffener* component (right) is manufactured. The two-dimensional stress field present in the piece of rolled plate is stochastic by nature such as was specified in Chapter 2.2. In this case study, distortion robustness will be assessed as a function of the position of the component in the piece of rolled plate.

First, in Section 5.1, the distortion magnitude and distortion robustness will be evaluated as a function of translation of the stiffener component in z-direction throughout the thickness of the piece of rolled plate. Subsequently, in Section 5.2, the component will be rotated about the y-axis at a fixed offset z_1. At last, in Section 5.3, the optimal position in terms of offset z_1 and angle β for robust distortion of the stiffener component will be determined.

5.1 Translation z_1 of the stiffener component in z-direction

First, the stochastic two-dimensional stress field will be translated in z-direction with respect to the iso-statically fixed stiffener component, such as is illustrated in Fig. 5.2. Throughout $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ both the distortion magnitude as well as distortion robustness will be assessed. The domain $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ will be discretized into 21 design values for z_1 for which the distortion magnitude and distortion robustness will be assessed by using Finite Element Analyses. Distortion robustness will be predicted in computational efficient manner by using the *improved method* which was elaborated in Section 4.4. As a measure for distortion, the displacement in z-direction of the centre node, $u_{c,3}$, located on the bottom centre line (indicated as a red line in Fig. 5.2), will be considered. As a measure for distortion robustness, $s_{u_{c,3}}$, being the *standard deviation* of the centre node's displacement in z-direction, will be considered.

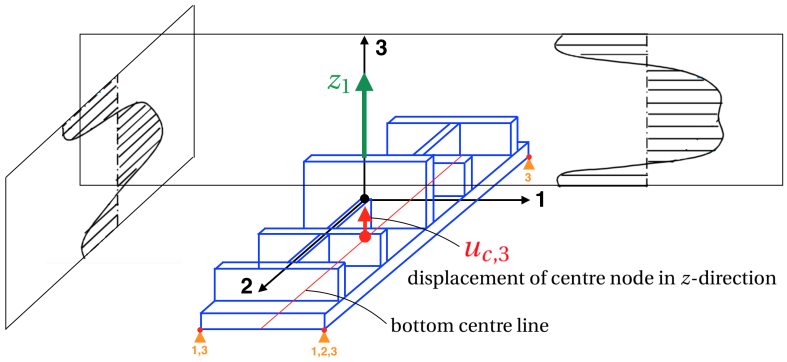


Figure 5.2: The two-dimensional stochastic stress field will be translated in z-direction throughout the range $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ with respect to the iso-statically fixed stiffener component. As a measure for distortion and distortion robustness, the displacement in z-direction of the centre node located on the bottom centre line (red line) and its standard deviation, $u_{c,3}$ and $s_{u_{c,3}}$ respectively, will be considered. The coordinate system shown in the illustration is affiliated with the piece of rolled plate, i.e. the two-dimensional stress field.

An optimization problem can be formulated in which an offset z_1 is sought at which distortion is most robust while constraining the magnitude of distortion to a maximum. The optimization problem formulation is as follows.

$$\begin{aligned} \underset{z_1}{\text{minimize}} \quad & f(z_1) \\ \text{subject to} \quad & g = \left(\frac{u_{c,3}(z_1)}{u_{3,\max}} \right)^2 - 1 \leq 0 \\ & 0\,\text{mm} \leq z_1 \leq 190\,\text{mm} \\ \text{with} \quad & f(z_1) = 2 s_{u_{c,3}}(z_1) \\ & u_{3,\max} = \text{undetermined}. \end{aligned} \tag{5.1}$$

In Fig. 5.3, the objective function $f(z_1)$ is plotted together with the *mean* distortion curve $\mu_{u_{c,3}}(z_1)$. At each instance of z_1, displacement of the centre node in z-direction can be described by a normal distribution having a mean and standard deviation. In order to illustrate distortion robustness throughout $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$, the *mean-distortion-plus-*

two-standard-deviations curve $\mu_{u_{c,3}} + 2s_{u_{c,3}}$ and the *mean-distortion-minus-two-standard-deviations* curve $\mu_{u_{c,3}} - 2s_{u_{c,3}}$ are plotted as well. These two curves form a bandwidth containing four standard deviations $s_{u_{c,3}}$. 95.4 %, being the equivalent of *four* standard deviations (see Fig. 2.11), of distortion of the centre node in z-direction, $u_{c,3}$, occurs within this interval.

Not shown due to confidentiality

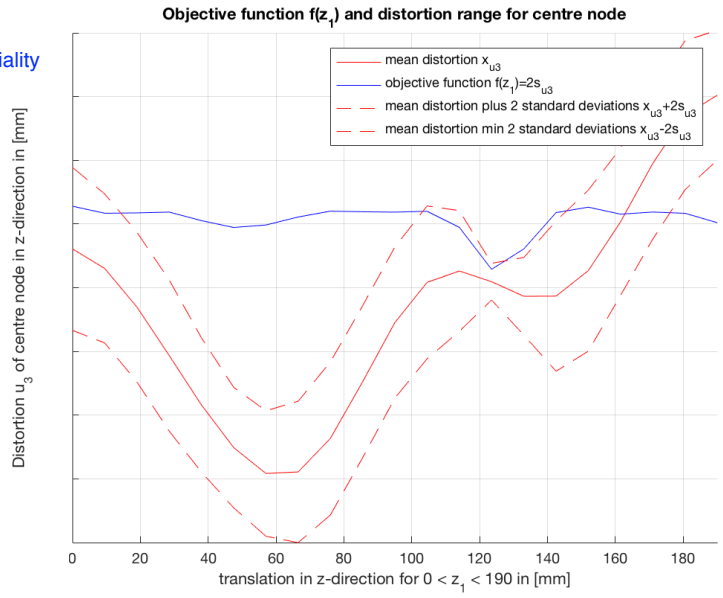


Figure 5.3: A plot of the objective function $f(z_1)$ (blue line), a plot of the mean distortion $\mu_{u_{c,3}}$ (red line) and plots of the mean distortion plus two standard deviations $\mu_{u_{c,3}} + 2s_{u_{c,3}}$ and the mean distortion minus two standard deviations $\mu_{u_{c,3}} - 2s_{u_{c,3}}$ (red dotted lines). 95.4 % - being the equivalent of four standard deviations - of distortion of the centre node in z-direction, $u_{c,3}$, falls within the area between the red dotted lines.

On closer inspection of Fig. 5.3, several observations can be made.

- The *dispersion* of distortion is relatively large - i.e. robustness is relatively poor - with respect to the magnitude of distortion. Including robustness analyses in part distortion problems thus seems highly relevant.
- A single destinctive *trough* is present at $z_1 = 123.5\,\text{mm}$ at which distortion seems to be extremely *robust*, i.e $2s_{u_{c,3}}(z_1) = 0.1915\,\text{mm}$. At other values for z_1, robustness seems to be fairly constant - i.e. fluctuating between $2s_{u_{c,3}}(z_1) \approx 0.5$ - $0.7\,\text{mm}$. Note that since the z_1 domain is discretized, a lower trough is plausibly existing.
- At some instances of z_1, the distortion interval is located in the positive as well as the negative domain, i.e. distortion can be either positive or negative implying that for individual pieces of rolled plate in a batch, the curvature of the machined component can be either convex or concave.
- Deterministic distortion passes through zero at four occasions of z_1. One occasion seems more favorable to the others for it has high robustness. At the Optimum, which is located at $z_1 \approx 123.5\,\text{mm}$, distortion robustness is high $2s_{u_{c,3}} \approx 0.2\,\text{mm}$ while at the same time the magnitude of distortion is low, $u_{c,3} \approx 0\,\text{mm}$.

As was mentioned earlier in Section 3.8, aircraft components tend to be highly non-prismatic due to their highly complex design features. As a result, components can deform in a different way than would be expected from the deformation behavior of a prismatic component. In Fig. 5.4, both prismatic- as non-prismatic deformation behavior is illustrated.

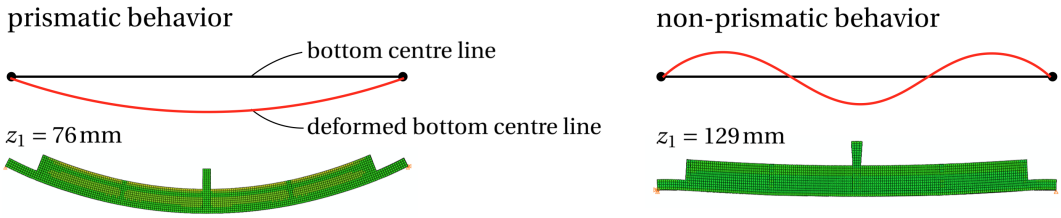


Figure 5.4: Examples of prismatic- and non-prismatic deformation behavior of the *bottom centre line* of the stiffener component illustrated in Fig. 5.2. Both the undeformed (black line) as the deformed bottom centre line (red line) are illustrated together with an example of the stiffener component showing prismatic (left) and non-prismatic behavior (right) at given offsets z_1. Prismatic deformation behavior occurs with both prismatic and non-prismatic components; non-prismatic deformation behavior could occur with non-prismatic components.

When non-prismatic deformation behavior occurs, deformation of the bottom centre node (indicated in Fig. 5.3) will not be a proper measure for distortion. For instance, distortion can be zero at one node and non-zero at another node. Therefore, it seems wise to consider a line of nodes throughout the component's length along which distortion is measured, such as is illustrated in Fig. 5.5.

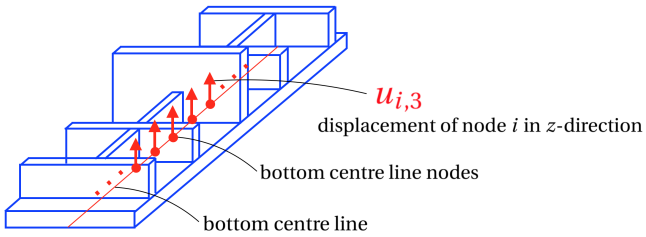


Figure 5.5: The stiffener component together with the bottom centre line. Distortion along the bottom centre line is measured by considering a number of equally distributed nodes i at which distortion is evaluated in z-direction.

The deformation of the stiffener component's bottom centre line can be visualized. The deformed bottom centre line will be referred to as *distortion curve*. In this case, 21 equally spaced nodes along the bottom centre line were chosen. The distortion curves of the stiffener component are plotted for all of the 21 discrete steps between $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ and are depicted in Fig. D.1 in Appendix D.1.

On closer inspection of Fig. D.1 in Appendix D.1, it can be concluded that the stiffener component shows mainly prismatic deformation behavior (one example of non-prismatic deformation behavior was found at $z_1 = 129\,\text{mm}$ in Fig. 5.4). At four instances of z_1, which are shown in Fig. 5.6, several interesting distortion curve scenario's were found. At offset $z_1 = 0\,\text{mm}$, the distortion curves can turn out to be either convex or concave, depending on

from which rolled plate in the batch the component is manufactured. At offsets $z_1 = 76\,\text{mm}$ and $z_1 = 190\,\text{mm}$, all stiffener components manufactured from the entire batch will show exclusively convex and concave deformation, respectively. At offset $z_1 = 123.5\,\text{mm}$, robustness is very high and the magnitude of distortion relatively low, which corresponds to the results found in Fig. 5.3.

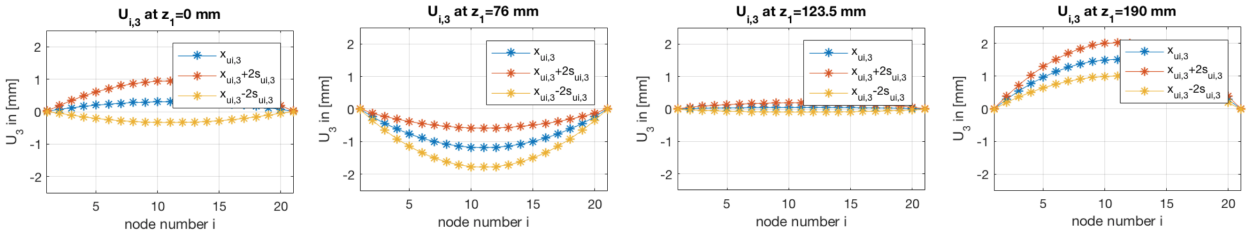


Figure 5.6: The distortion curves of the stiffener components for four instances of offset z_1, being $z_1 = 0\,\text{mm}$, $z_1 = 76\,\text{mm}$, $z_1 = 123.5\,\text{mm}$ and $z_1 = 190\,\text{mm}$, respectively. All 21 instances throughout $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ are included in Appendix D.1. The blue line represents the mean distortion curve, the yellow line represents the mean-minus-two-standard-deviations distortion curve and the red line represents the mean-plus-two-standard-deviations distortion curve. 95.4 % - being the equivalent of four standard deviations - of distortion curves will fall within the area between the yellow and red line.

A more appropriate measure for distortion and distortion robustness of the stiffener component can be formulated. One that takes non-prismatic deformation behavior into consideration. The amount of distortion, symbolized by f_u, can be quantified by taking the *Euclidean* magnitude of all displacements in z-direction of the selection of bottom centre line nodes i with $i = 1 \cdots n$ with n being the number of nodes, like

$$f_u = \sqrt{\sum_{i=1}^{n} \left(u_{i,3}\right)^2}. \tag{5.2}$$

Subsequently, the amount of robustness, symbolized by f_{s_u}, can be quantified by taking the Euclidean magnitude of the standard deviations of the displacements $u_{i,3}$ for bottom centre line nodes $i = 1 \cdots n$, like

$$f_{s_u} = \sqrt{\sum_{i=1}^{n} \left(s_{u_{i,3}}\right)^2}. \tag{5.3}$$

Since the stiffener component shows mainly prismatic deformation behavior, displacement in z-direction of the centre node, $u_{c,3}$, as was formulated in the optimization problem in Eq. 5.1, will suffice as a measure for distortion. For components that show non-prismatic deformation behavior, Eq. 5.2 and Eq. 5.3 would be more appropriate as a measure for distortion and distortion robustness, respectively.

The optimal offset z_1 at which the stiffener component is to be manufactured from the piece of rolled plate for which distortion robustness is highest, together with the resulting distortion distribution of the bottom centre line is visualized in Fig. 5.7.

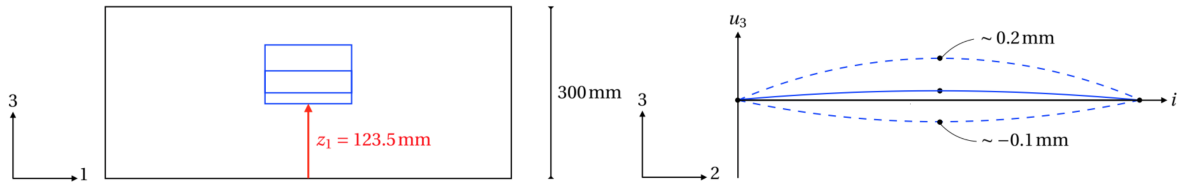


Figure 5.7: On the left, the optimal position of the stiffener component (blue) with respect to the piece of rolled plate at offset $z_1 = 123.5\,\text{mm}$ is shown for which distortion robustness is highest. On the right, the distortion distribution of the stiffener component's bottom centre line is visualized. The blue curve represents the mean distortion curve μ_{u_3}, the blue dotted curves represent the mean-minus-two-standard-deviations and the mean-plus-two-standard-deviations curves, $\mu_{u_3} - 2s_{u_3}$ and $\mu_{u_3} + 2s_{u_3}$, respectively. Distortion of the bottom centre line will occur in between the blue dottes lines with a probability of 95.4 %.

In Tab. 5.2, the computational effort that was required for the robustness analysis of translation of the component in z-direction with respect to the piece of rolled plate is summarized. Equivalent to the robustness analysis in Section 5.1, in total, 252 static analyses were required.

Table 5.1: An overview describing the computational effort that was required for performing a robustness analysis for the stiffener component as a function of its *offset* z_1 inside the piece of rolled plate using the *improved* method which was elaborated in Chapter 4.4.

Computational effort parameter	Symbol	Value
Number of design value iterations	n	21
Number of static analyses per design value iteration	m	12
Total number of static analyses required	–	252
Total computation time	t	1 h 45 min 50 sec

5.2 Rotation β of the stiffener component about the y-axis

Next, the stochastic two-dimensional stress field is rotated with angle β about the y-axis with respect to the iso-statically fixed stiffener component, such as is illustrated in Fig. 5.8. The origin of the stress field is located at the component's centre, as is illustrated in Fig. 5.8, such that rotation of the component about its centre is simulated. During the rotation, the component is held fixed at offset $z_1 = 38\,\text{mm}$ (arbitrary) with respect to the piece of rolled plate's bottom face, which is equivalent to a distance of 93 mm between the origin and the bottom face of the piece of rolled plate.

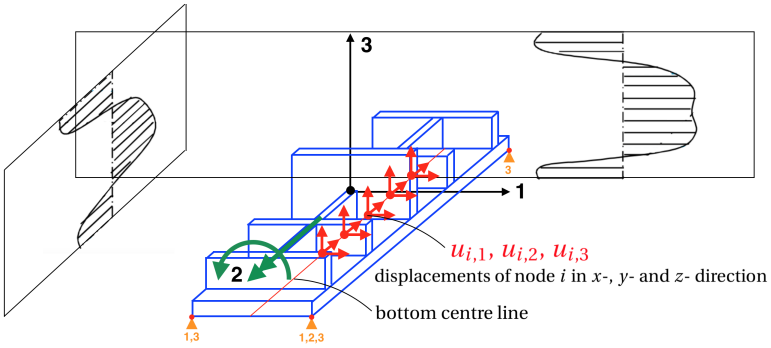


Figure 5.8: The two-dimensional stochastic stress field will be rotated about the y-axis throughout the discretized range $0\,\text{mm} \leq \beta \leq 360°$ with respect to the iso-statically fixed stiffener component. As a measure for distortion and distortion robustness, the Euclidean magnitude of all centre line node displacements in x-, y- and z-direction, $u_{i,1}$, $u_{i,2}$ and $u_{i,3}$ and their standard deviations, $s_{u_{i,1}}$, $s_{u_{i,2}}$ and $s_{u_{i,3}}$ respectively, will be considered.

In Section 5.1, only displacements in z-direction were considered for determining the amount of distortion and distortion robustness. In the case of rotation of the component about the y-axis, displacements in other directions will become significant as well. This becomes apparent in Appendix D.2 where the distortion curves of the stiffener component are depicted throughout the discretized interval $0° \leq \beta \leq 360°$. Distortion curves for displacement in x-, y- and z-direction are depicted in Fig. D.2, Fig. D.3 and D.4, respectively. It seems wise to consider displacements in all three directions (x, y and z) for expressing the amount of distortion and distortion robustness.

As was mentioned in Fig. 5.8, the Euclidean magnitude of all centre line node displacements in x-, y- and z-direction, $u_{i,1}$, $u_{i,1}$ and $u_{i,1}$ respectively and their standard deviations, $s_{u_{i,1}}$, $s_{u_{i,1}}$ and $s_{u_{i,1}}$ respectively, will be considered as measure for distortion and distortion robustness. The amount of distortion, symbolized by f_u, is obtained through solving

$$f_u = \sqrt{\sum_{i=1}^{n} \left(u_{i,1}\right)^2 + \sum_{i=1}^{n} \left(u_{i,2}\right)^2 + \sum_{i=1}^{n} \left(u_{i,3}\right)^2}. \tag{5.4}$$

Subsequently, the amount of robustness, symbolized by f_{s_u}, is obtained through solving

$$f_{s_u} = \sqrt{\sum_{i=1}^{n} \left(s_{u_{i,1}}\right)^2 + \sum_{i=1}^{n} \left(s_{u_{i,2}}\right)^2 + \sum_{i=1}^{n} \left(s_{u_{i,3}}\right)^2}. \tag{5.5}$$

Again, an optimization problem can be formulated in which rotation β is sought for which distortion is most robust while constraining the magnitude of distortion to a maximum. The optimization problem formulation is as follows.

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & f(\beta) \\ \text{subject to} \quad & g = \left(\frac{f_u(\beta)}{f_{u,\max}}\right)^2 - 1 \leq 0 \\ & 0° \leq \beta \leq 360° \\ \text{with} \quad & f(\beta) = 2f_{s_u}(\beta) \\ & f_{u,\max} = \text{undetermined}. \end{aligned} \tag{5.6}$$

In Fig. 5.9, the objective function $f(\beta) = 2f_{s_u}(\beta)$, which relates to robustness as was formulated in Eq. 5.5, is plotted along with the magnitude of distortion, f_u, which was formulated in Eq. 5.4. The mean-distortion-plus-two-standard-deviations curve, $f_u + 2f_{s_u}$, and the mean-distortion-minus-two-standard-deviations curve, $f_u - 2f_{s_u}$, are plotted as well. These two curves form a bandwith containing four standard deviations f_{s_u}. Within this bandwith, distortion will occur with a probability of 95.4 %.

Not shown due to confidentiality

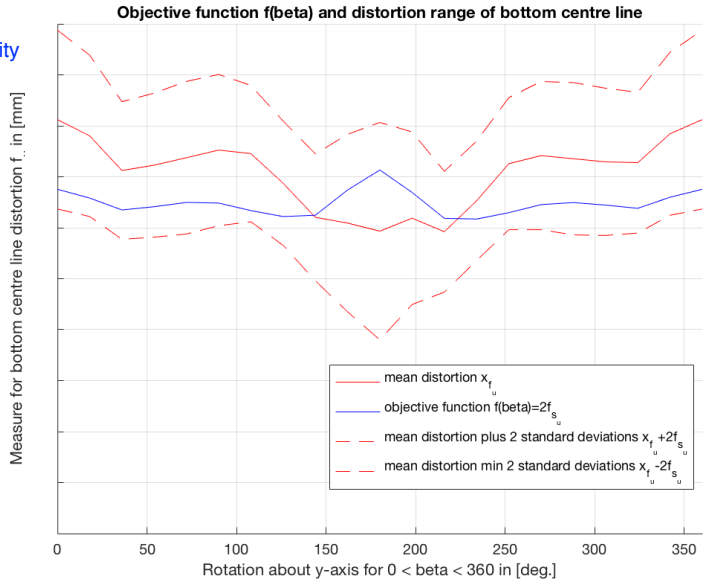


Figure 5.9: A plot of the objective function $f(\beta)$ (blue line), the *magnitude* of distortion f_u (red line), the mean-distortion-plus-two-standard-deviations $f_u + 2f_{s_u}$ and the mean-distortion-minus-two-standard-deviations $f_u - 2f_{s_u}$ (red dotted lines). The Euclidean magnitude of distortion will occur between the blue dotted lines with a probability of 95.4 %.

On closer inspection of Fig. 5.9, several observations can be made.

- The distortion pattern seems to be symmetrical about $\beta = 180°$. This seems logical since distortion should be indifferent to whether the component is rotated in clockwise- or counterclockwise sense.
- Robustness, which is manifested by $2f_{s_u}(\beta)$ in Fig. 5.9, varies between $2f_{s_u}(\beta) \approx 1.3 - 2.1$ mm and seems thus to be fairly consistent.

- Distortion is most robust ($f_{s_u} \approx 1.3\,\text{mm}$) at $\beta \approx 140°$ and least robust ($f_{s_u} \approx 2.1\,\text{mm}$) at $\beta \approx 180°$. The Optimum seems to be at $\beta \approx 140°$ (and obviously $\beta \approx 220°$) where robustness is highest and the distortion magnitude is relatively low.

The optimal angle β at which the stiffener component is to be manufactured from the piece of rolled plate (while the distance from the centre of the component to the piece of rolled plate's bottom face is held fixed at 93 mm) for which distortion robustness is highest is visualized in Fig. 5.10 together with the resulting distortion distribution of the bottom centre line.

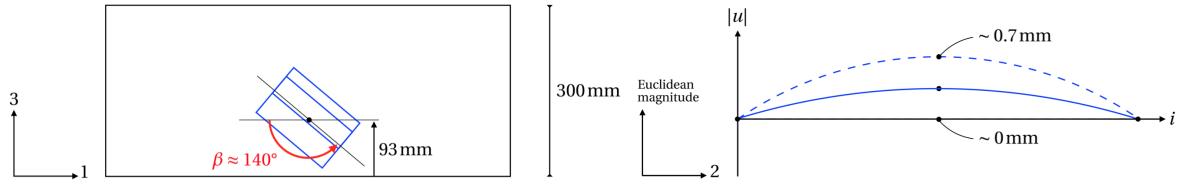


Figure 5.10: On the left, the optimal position of the stiffener component (blue) with respect to the piece of rolled plate at angle $\beta = 140°$ at fixed offset is shown for which distortion robustness is highest. On the right, the distortion distribution of the stiffener component's bottom centre line is visualized, in which distortion is defined as the magnitude $|u|$ of distortion in x-, y- and z-direction. The blue curve represents the mean distortion curve $\mu_{|u|}$, the blue dotted curves represent the mean-minus-two-standard-deviations and the mean-plus-two-standard-deviations curves, $\mu_{|u|} - 2s_{|u|}$ and $\mu_{|u|} + 2s_{|u|}$, respectively. Distortion of the bottom centre line will occur in between the blue dotted lines with a probability of 95.4 %.

In Tab. 5.2, the computational effort that was required for the robustness analysis of rotation β of the component about the y-axis at fixed offset is shown. Like for the robustness analysis in Section 5.1, in total, 252 static analyses were required.

Table 5.2: An overview of the computational effort that was required for the robustness analysis for the stiffener component as a function of its *angle* β at fixed offset in the piece of rolled plate using the *improved* method which was elaborated in Chapter 4.4.

Computational effort parameter	Symbol	Value
Number of design value iterations	n	21
Number of static analyses per design value iteration	m	12
Total number of static analyses required	–	252
Total computation time	t	1 h 31 min 32 sec

5.3 Optimal position of stiffener component in piece of rolled plate in terms of offset z_1 and angle β

Distortion magnitude and distortion robustness have been evaluated as a function of translation of the stiffener component in z-direction and rotation about its y-axis at fixed offset z_1 within the piece of rolled plate, separately. The optimal position for robust distortion can now be determined by assessing distortion robustness for translation in z-direction and rotation about the y-axis *at the same time*. The stochastic two-dimensional stress field will now be rotated with angle β about the y-axis and translated with offset z_1 in z-direction with respect to the iso-statically fixed stiffener component, such as is illustrated in Fig. 5.11.

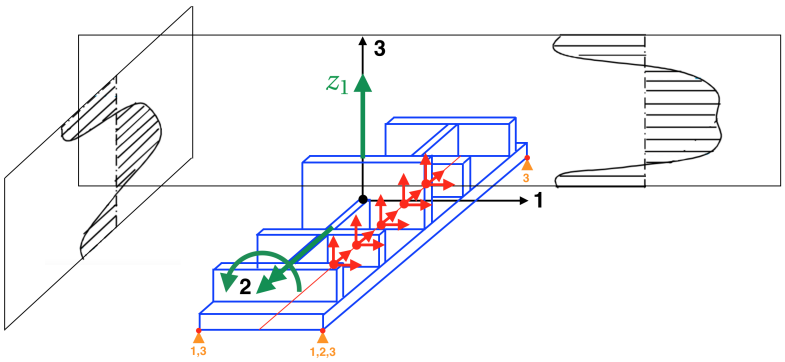


Figure 5.11: The two-dimensional stochastic stress field will be rotated about the y-axis throughout the discretized range $0\,\text{mm} \leq \beta \leq 360°$ and translated in z- direction throughout the discretized range $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ with respect to the iso-statically fixed stiffener component. As a measure for distortion and distortion robustness, again the Euclidean magnitude of all centre line node displacements in x-, y- and z-direction, $u_{i,1}$, $u_{i,2}$ and $u_{i,3}$ and their standard deviations, $s_{u_{i,1}}$, $s_{u_{i,2}}$ and $s_{u_{i,3}}$ respectively, will be considered.

As was elaborated in Section 5.2, the Euclidean magnitude of displacements of all nodes on the bottom centre line in three directions, $u_{i,1}$, $u_{i,2}$ and $u_{i,3}$ and their standard deviations, $s_{u_{i,1}}$, $s_{u_{i,2}}$ and $s_{u_{i,3}}$ will be considered as a measure for distortion magnitude f_u (Eq. 5.4) and distortion robustness f_{s_u} (Eq. 5.5), respectively. The optimization problem formulation in which both offset z_1 and rotation β are optimized for robustness, is as follows.

$$
\begin{aligned}
&\underset{z_1,\beta}{\text{minimize}} && f(z_1,\beta) \\
&\text{subject to} && g = \left(\frac{f_u(z_1,\beta)}{f_{u,\max}}\right)^2 - 1 \leq 0 \\
& && 0\,\text{mm} \leq z_1 \leq 190\,\text{mm} \\
& && 0° \leq \beta \leq 360° \\
&\text{with} && f(z_1,\beta) = 2 f_{s_u}(z_1,\beta) \\
& && f_{u,\max} = 7 \text{ (arbitrarily chosen)}.
\end{aligned}
\tag{5.7}
$$

Note that the maximum distortion magnitude, $f_{u,\max}$, was chosen to be equal to 7 mm (arbitrary). The maximum distortion magnitude refers to the Euclidean magnitude of all node

displacements in x-, y- and z-direction (Eq. 5.4) and not to the maximum value of distortion.

For each combination of offset z_1 throughout the discretized range $0\,\text{mm} \leq z_1 \leq 190\,\text{mm}$ and of angle β throughout the discretized range $0° \leq \beta \leq 360°$, distortion magnitude and distortion robustness are evaluated. In total, distortion magnitude, f_u, and distortion robustness, f_{s_u}, is evaluated for $21 \times 21 = 441$ combinations of z_1 and β. The computational effort that was required for the robustness analysis of the positioning, i.e. offset z_1 and angle β, of the stiffener component in the piece of rolled plate is summarized in Tab. 5.3.

Table 5.3: An overview of the computational effort that was required for the robustness analysis for the optimal position in terms of offset z_1 and angle β of the stiffener component in the piece of rolled plate using the *improved* method which was elaborated in Section 4.4.

Computational effort parameter	Symbol	Value
Number of design value iterations	n	$21 \times 21 = 441$
Number of static analyses per design value iteration	m	12
Total number of static analyses required	–	5292
Total computation time	t	34 h 53 min 0 sec

Distortion magnitude $f_u(z_1, \beta)$ as a function of offset z_1 and angle β can be visualized in a *surface* plot which is shown in Fig. 5.12.

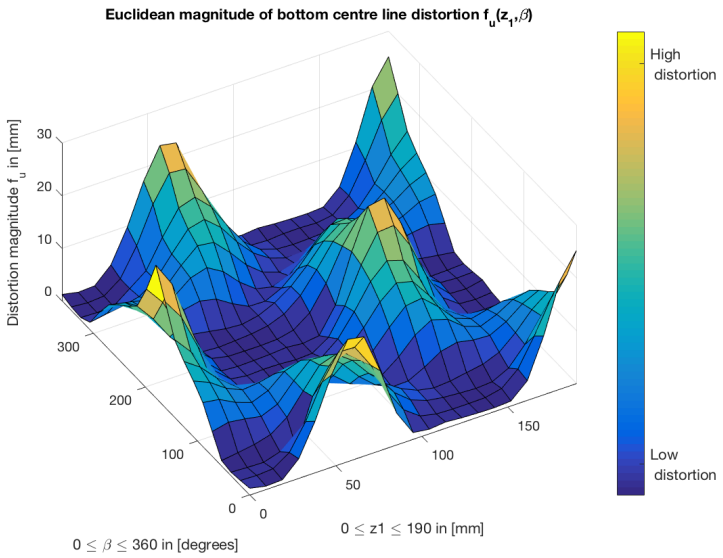


Figure 5.12: A surface plot visualizing distortion magnitude $f_u(z_1, \beta)$ as a function of both offset z_1 and angle β of the stiffener component within the piece of rolled plate.

On closer inspection of Fig. 5.12, several observations can be made.

- Distortion magnitude $f_u(z_1, \beta)$ seems to be *symmetrical* about $\beta = 180°$. This seems logical since distortion should be indifferent to whether the component is rotated in clockwise- or counterclockwise sense.

- The distortion path $f_u(z_1, \beta = 180°)$ seems to equivalent to the distortion path $f_u(z_1, \beta = 0°)$, but then *mirrored* about $z_1 = 190\,\text{mm}$: $f_u(z_1, \beta = 180°) = f_u(190\,\text{mm} - z_1, \beta = 0°)$. This seems logical since distortion should be equivalent when translating a component faced upwards ($\beta = 0°$) in positive z-direction from bottom to top, to when translating the component faced downwards ($\beta = 180°$) in negative z-direction from top to bottom, due to symmetry of the stress profile.

In the same manner, distortion robustness $f_{s_u}(z_1, \beta)$ as a function of offset z_1 and angle β can be visualized in a surface plot which is shown in Fig. 5.13.

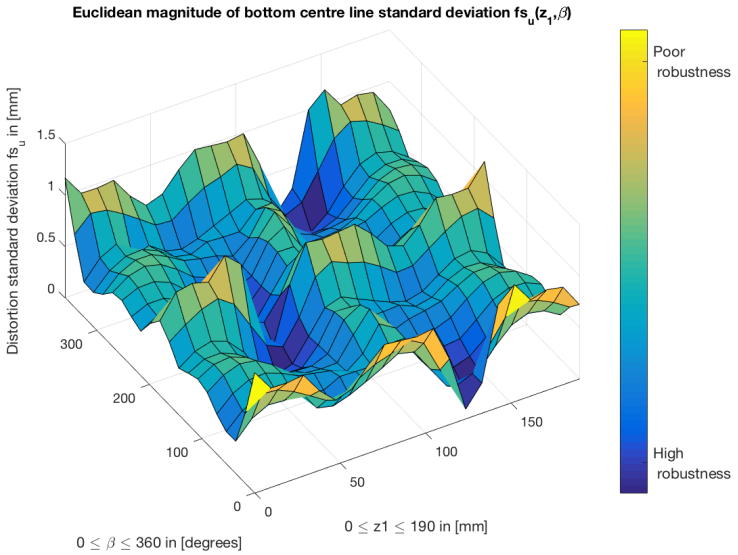


Figure 5.13: A surface plot visualizing distortion robustness $f_{s_u}(z_1, \beta)$ as a function of both offset z_1 and angle β of the stiffener component within the piece of rolled plate.

On closer inspection of Fig. 5.13, several observations can be made.

- Similarly as for distortion magnitude $f_u(z_1, \beta)$, distortion robustness $f_{s_u}(z_1, \beta)$ seems to be *symmetrical* about $\beta = 180°$ and the distortion path $f_u(z_1, \beta = 180°)$ seems to equivalent to the distortion path $f_u(z_1, \beta = 0°)$, but then *mirrored* about $z_1 = 190\,\text{mm}$.

- Significant variation in distortion robustness is perceived for different combinations of offset z_1 and angle β. In other words, the degree of distortion robustness is highly dependent on the position of the component inside the piece of rolled plate. This demonstrates that robustness evaluation of components for their position inside rolled plate is *highly relevant*. Combinations of offset z_1 and angle β can be found where distortion is considerably more robust then for other combinations of offset z_1 and angle β.

- Several distinctive troughs can be found where distortion seems to be extremely robust. Similarly, various peaks can be found where robustness is very poor.

The optima can be identified more conveniently by displaying top views of the surface plots for $f_u(z_1, \beta)$ and $f_{s_u}(z_1, \beta)$ side by side, such as is done in Fig. 5.14.

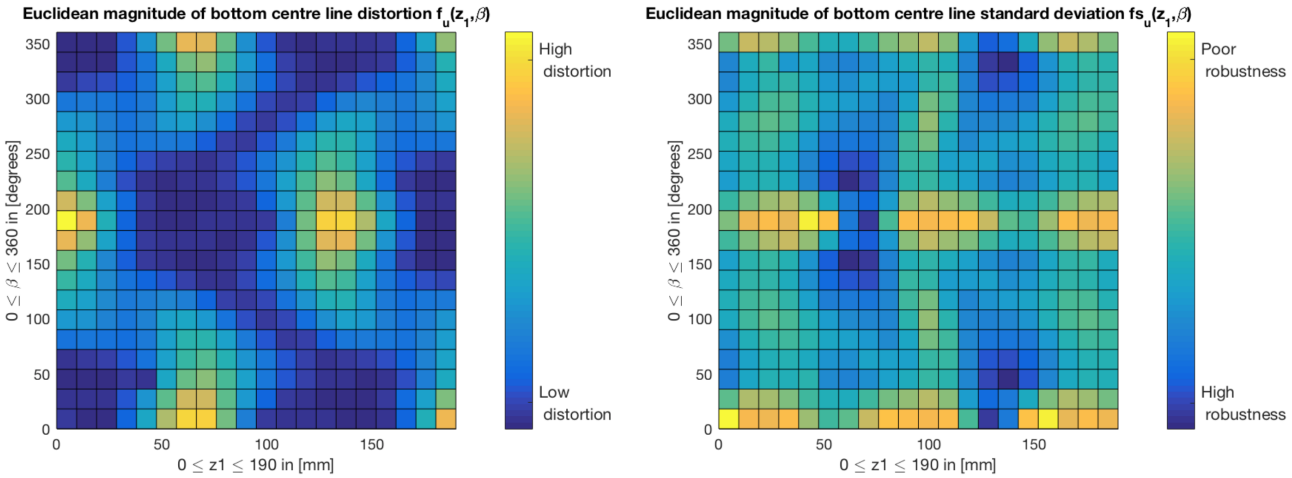


Figure 5.14: On the left, a top view of the surface plot of distortion magnitude $f_u(z_1, \beta)$ is shown. On the right, a top view of distortion robustness $f_{s_u}(z_1, \beta)$ is shown. Unfavorable combinations of z_1 and β are colored yellow: large distortion magnitudes and poor robustness. Favorable combinations of z_1 and β are colored blue: small distortion magnitudes and high robustness. Six troughs can be identified (right) as being blue squares where distortion is extremely robust.

On closer inspection of Fig. 5.14, several observations can be made.

- In total, *six* troughs can be identified where distortion is extremely robust. All of these six troughs seems to be located in areas where distortion magnitude is extremely low as well and thus can all six be considered *local optima*.
- It can *not* be stated that when distortion magnitude is low, distortion is robust as well. Troughs can be identified with extremely low distortion magnitudes, however at which distortion robustness is at its highest (for example, see lower left corner and upper left corner: distortion magnitude troughs and distortion robustness peaks).

A more convenient way of visualizing both the objective function $f(z_1, \beta)$ and constraint $g(z_1, \beta)$, formulated in the optimization problem formulation (Eq. 5.7), in a single figure is by employing a contour plot which is shown in Fig. 5.15. The areas enclosed by the red lines represent the infeasible domains of constraint $g(z_1, \beta)$. Within these domains, distortion magnitude, $f_u(z_1, \beta)$, is larger than the arbitrarily chosen value of $f_u(z_1, \beta) = 7\,\text{mm}$ (see optimization problem formulation in Eq. 5.7).
On closer inspection of Fig. 5.15, several observations can be made.

- As was observed in Fig. 5.14, six optima can be distinguished which are all located in the feasible domain of constraint $g(z_1, \beta)$.
- Areas can be found where distortion robustness is relatively good but where distortion magnitude is high. Contrarily, areas can be found where distortion robustness is poor but where distortion magnitude satisfies the constraint.
- It thus seems relevant to consider both distortion robustness, $f_u(z_1, \beta)$, as distortion magnitude, $f_{s_u}(z_1, \beta)$, when choosing the right combination of offset z_1 and angle β where distortion is both acceptable and robust.

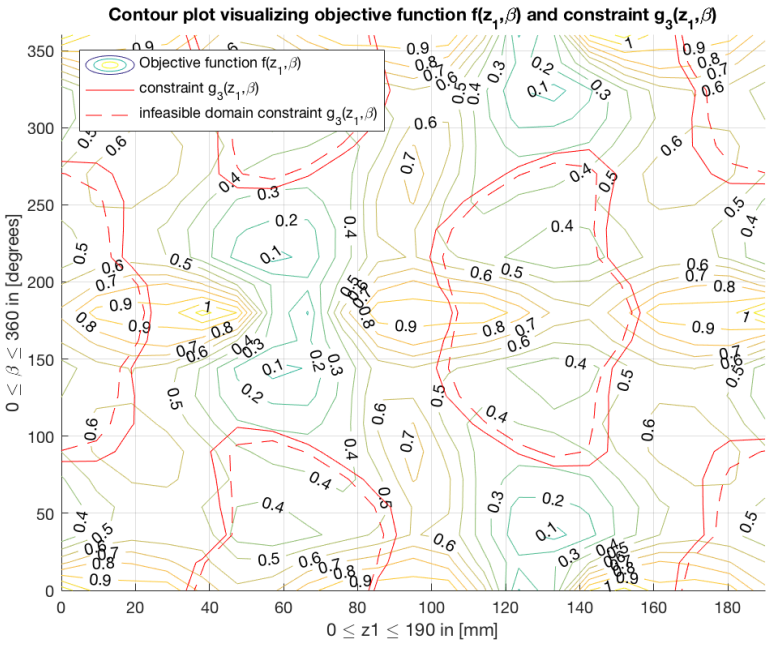


Figure 5.15: A contour plot visualizing the objective function, $f(z_1, \beta)$, relating to distortion robustness and the constraint, $g(z_1, \beta)$, relating to the distortion magnitude, which were defined in the optimization problem formulation in Eq. 5.7. The red lines indicate where the maximum value for distortion magnitude is reached, $f_u(z_1, \beta) = 7\,\text{mm}$, and where the constraint is equal to $g(z_1, \beta) = 0$. The areas enclosed by the red lines point out the infeasible domains of the constraint where the distortion magnitude exceeds $f_u(z_1, \beta) > 7\,\text{mm}$ (arbitrarily chosen).

The six optima that were found are listed in Tab. 5.4. All six optima seem to have very low and approximately equivalent values for distortion magnitude, f_u, and distortion robustness, f_{s_u}.

Table 5.4: The location in terms of offset z_1 and angle β of the six optima that were found in the contour plot in Fig. 5.15, together with the corresponding values for distortion magnitude $f_u(z_1, \beta)$ and distortion robustness $f_{s_u}(z_1, \beta)$.

Variable	Optimum 1	Optimum 2	Optimum 3	Optimum 4	Optimum 5	Optimum 6
z_1	123.5 mm	66.5 mm	57 mm	57 mm	133 mm	133 mm
β	0°	180°	216°	144°	36°	324°
f_u	0.87	0.87	0.8093	0.7205	0.8093	0.7205
f_{s_u}	0.092 87	0.092 87	0.063 52	0.058 22	0.063 52	0.058 22

In Fig. 5.4, for all six optima, the front view of the positioning (i.e. offset z_1 and angle β) of the stiffener component within the piece of rolled plate's through-thickness is illustrated.

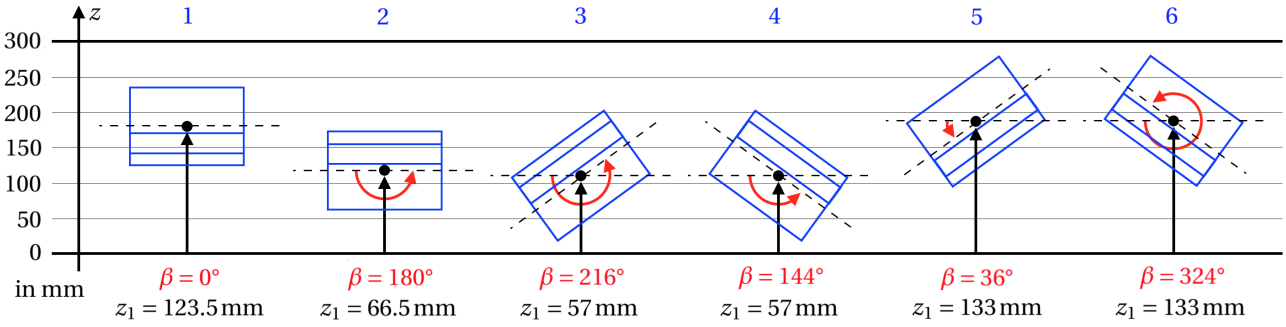


Figure 5.16: A front view of the positioning in terms of offset z_1 and angle β of the stiffener component within the rolled plate's through-thickness is illustrated for all six optima that were listed in Tab. 5.4.

On closer inspection of Fig. 5.16, several observations can be made.

- It appears that Optimum 1 and Optimum 2 involve a *similar* configuration of the component in the piece of rolled plate. The positioning of the component in the piece of rolled plate for Optimum 2 is equal to the positioning at Optimum 1 if it was mirrored about an imaginary xy-symmetry plane located at half the rolled plate's thickness. This makes sense since the stress profile within the piece of rolled plate is *symmetrical*.

- It can be concluded that optima 3-6 involve a *similar* configuration of the component in the piece of rolled plate as well. At Optimum 3, the stiffener component is rotated *counter*-clockwise with 144° and at Optimum 4, the stiffener component is rotated clockwise with 144°. As was concluded in Chapter 5.2, distortion is indifferent to whether the component is rotated in clockwise or counterclockwise sense. Optima 5-6 have configurations that are equivalent to the configurations at optima 3-4 if they were mirrored about an imaginary xy-symmetry plane located at half the rolled plate's thickness. These optima have equivalent distortion as optima 3-4 due to the symmetrical nature of the rolled plate's stress profile.

- Hence, *two* different *configurations* exist at which the stiffener component is to be manufactured inside the piece of rolled plate as a result of which distortion magnitude is low and distortion robustness is high. Distortion magnitude and -robustness is equivalent when the component would be mirrored about the yz-symmetry plane and the xy-symmetry plane. This since i) distortion is equivalent whether the component is rotated in clockwise sense or in counterclockwise sense and ii) since the rolled plate's residual stress profile is symmetrical by nature.

The *first* configuration at which the stiffener component is to be manufactured inside rolled plate in order for distortion to be both low and robust, is illustrated in Fig. 5.17. Distortion magnitude and -robustness is equivalent when the component would be mirrored about the xy-symmetry plane. Strictly speaking, the component can be mirrored about the yz-symmetry plane as well, however, since the component is symmetrical with respect to the yz-symmetry plane, this would imply the same position.

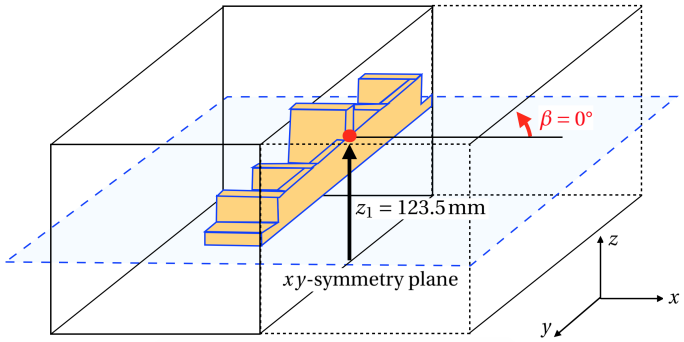


Figure 5.17: The first configuration is shown where distortion magnitude is low and distortion robustness to be high. Distortion magnitude and -robustness is equivalent when the component would manufactured at the position mirrored about the xy-symmetry plane (offset $z_1 = 66.5\,\text{mm}$ and angle $\beta = 180°$). Note that the component can be mirrored about a yz-symmetry plane as well, however this will yield the same two positions.

The *second* configuration at which the stiffener component is to be manufactured inside rolled plate in order for distortion to be both low and robust, is illustrated in Fig. 5.18. Distortion magnitude and -robustness is equivalent when the component would be mirrored about the xy-symmetry plane and when the original- and mirrored position would be mirrored about the yz-symmetry plane.

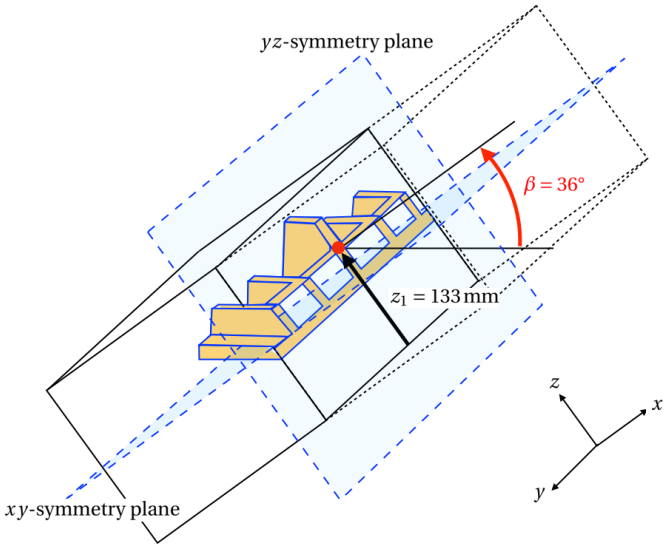


Figure 5.18: The second configuration is shown. Distortion magnitude and -robustness is equivalent when the component would manufactured at the position mirrored about the xy-symmetry plane and when the original and mirrored position would be mirrored about the yz-symmetry plane.

In Appendix D.3, the application of the *improved* method for the robustness analysis of distortion of the stiffener component as a function of its position in terms of offset z_1 and angle β inside the piece of rolled plate is elaborated.

5.4 Physical explanation for the presence of robustness troughs where distortion is extremely robust

In this case study it appeared that, generally, distortion has *poor* robustness throughout the entire domain of offset z_1 and angle β. At certain locations, however, sudden *troughs* occur where both distortion is extremely robust and distortion magnitude is low. When for instance recalling Fig. 5.3, in which the distortion distribution was shown as a function of translation of the stiffener component in z-direction; more or less significant variation in distortion can be observed throughout the entire domain of offset z_1, except for a sudden dip that is present around $z_1 \approx$ 110-140 mm. Likewise is the case in Fig. 5.13, in which distortion robustness was visualized as a function of both translation z_1 and rotation β of the stiffener component. At six certain locations, sudden troughs were observed where distortion is extremely robust and distortion magnitude is low.

In order to explain the presence of these robustness troughs where distortion is extremely robust, an *elementary* beam with rectangular cross section is considered. The slender beam (dimensions: $l = 1000$ mm, $b = 170$ mm, $h = 50$ mm) is translated in z-direction through the same piece of rolled plate that was considered in this case study (see Fig. 5.1) having a stochastic stress field of which the specifics were listed in Tab. 2.1. The distortion distribution of the rectangular beam as a function of translation in z-direction through the piece of rolled plate is shown in Fig. 5.19. This distribution was obtained analytically.

Not shown due to confidentiality

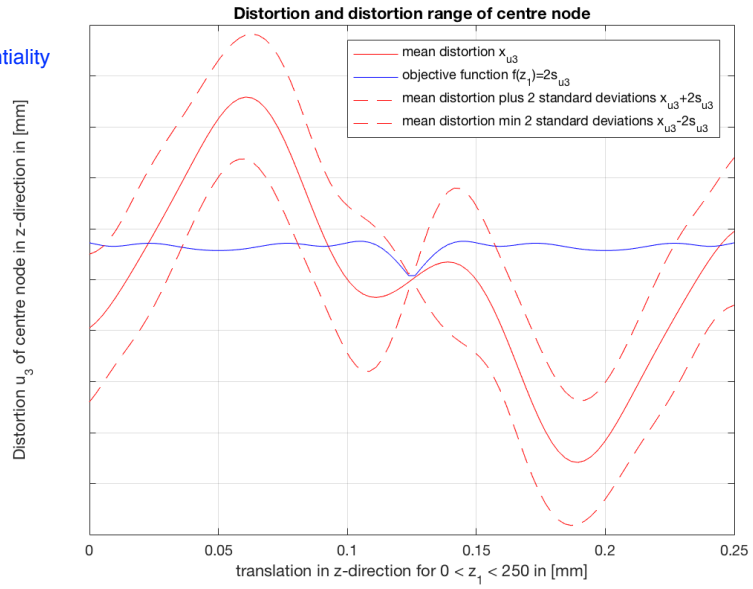


Figure 5.19: The distortion distribution of a slender *prismatic* beam with rectangular cross section as function of translation in z-direction throught the piece of rolled plate. The blue line relates to robustness and the red line represents mean distortion $\mu_{u_{c,3}}$. The area enclosed between the red dotted lines contains four standard deviations implying that 95.4 % of distortion of the centre node will occur in between these two dotted lines.

On closer inspection of Fig. 5.19, it can be observed that the same phenomenon occurs where distortion has poor robustness throughout the entire domain of z_1, however a sudden

dip is present halfway at $z_1 = 125\,\text{mm}$ where distortion is extremely robust. At $z_1 = 125\,\text{mm}$, the standard deviation of distortion of the centre node, s_{u_3}, is even equivalent to zero.

The reason for the *standard deviation* of distortion to be equivalent to zero when the rectangular beam is positioned exactly halfway the piece of rolled plate's thickness, is that 1) the stochastic cosine fit function is always *symmetrical* with respect to the rolled plate's middle xy-symmetry plane since it consists of a summation of *integer* cosine periods (Eq. 2.10-2.11), and 2) that the beam with rectangular cross section is *symmetrical* with respect to the rolled plate's middle xy-symmetry plane. When the cross section is symmetrical with respect to the rolled plate's middle xy-symmetry plane and when the geometry is centered with respect to z, the moment generated by the stress profile will *always* be equivalent to zero, no matter what values are chosen for the stress coefficients. This since the stress fit function is also symmetrical with respect to the rolled plate's middle xy-symmetry plane. Fig. 5.20 shows three examples of prismatic beams with symmetrical cross sections that are centered with respect the rolled plate's height. Since the stochastic stress profile is symmetrical by nature as well, distortion of the symmetrical geometries will be zero for all possible stress profiles from the residual stress distribution.

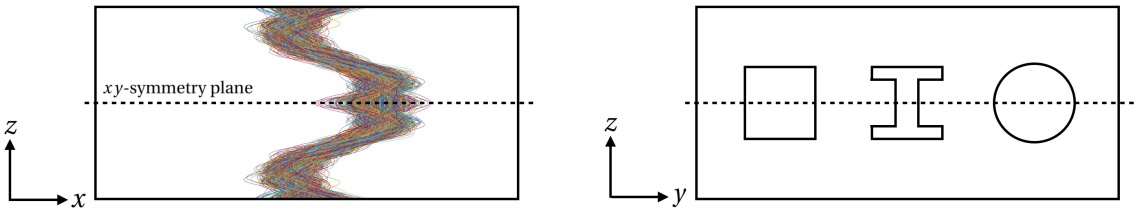


Figure 5.20: On the left, a side view of the piece of rolled plate is shown which is subjected to a residual stress distribution. 1000 random samples are shown and each one of them is *symmetrical* about the xy-symmetry plane. On the right, a front view is shown with three cross sections that are *symmetrical* about the xy-symmetry plane. When the cross sections are centered with respect to z, distortion will be equivalent to zero for *all* 1000 random residual stress profiles.

Residual stress profiles can, in theory, be *asymmetrical* by nature as well and still satisfy the equilibrium equations (Eq. 3.1-3.2). In this thesis however, the stress fit functions were chosen to be symmetrical by nature. The question that remains is whether it was *realistic* to assume that residual stress in rolled plate is symmetrical by nature. In order to validate this assumption, experimental data of residual stress in a batch of rolled plate, which was elaborated upon in Section 2.2, can be examined for symmetry. By inspection of Fig. 2.9 and Fig. 2.10 it can be concluded that experimental measurements of residual stress in rolled plate show indeed a *high degree* of symmetry. The assumption that residual stress in rolled plate is symmetrical seems to be realistic.

Aircraft components tend to have highly complex designs features and are as a result often not symmetrical. The stiffener component considered in this case study, for example, is not symmetrical. Optima however still seem to exist where distortion is extremely robust. This is since components can be asymmetrical, however, still have a *degree* of symmetry. Components that have a high degree of symmetry, have optima with high robustness; components with a low degree of symmetry have optima that are less robust.

Chapter 6

Conclusions

With respect to the developed methodology in this thesis, several conclusions can be drawn.

- A methodology has been developed that allows for both *reliable* and *computational efficient* robustness prediction of aircraft component distortion which is a result of a reductive manufacturing process from stochastic pre-stressed rolled plate material. The developed methodology can be applied to *any* aircraft component.

  In the current state of the art, *deterministic* modeling tools were developed to predict part distortion. Deterministic distortion predictions, however, appeared to be meaningless since residual stress in rolled plate is subjected to substantial variation. Consequently, efforts were made to predict the *stochastics* of distortion - referred to as robustness predictions. These efforts, however, worsened the computational efficiency of the Finite Element Analyses *drastically*, such that robustness predictions became infeasible in terms of time and costs. Brute force *Monte Carlo* simulations were employed relying on random sampling of the residual stress distribution in order to obtain numerical results of the distortion distribution. In these Monte Carlo simulations, a *trade-off* had to be made between *computational efficiency* and *reliability*. A low sample size meant acceptable computation costs but poor reliability; a high sample size meant good reliability but high computation costs. Assuring reliability of the robustness prediction, a sample size of $m > 1000$ is recommended suggesting that at least 1000 static analyses were required.

  The developed *improved* - computational efficient - method requires only 12 static analyses based upon which robustness is subsequently determined *analytically*. The developed method for predicting robustness is thus > 100 times more efficient in terms of computation cost. Moreover, the robustness prediction is more reliable since the reliability now only depends on the quality of the experimental measurements and the choice for its mathematical stochastic representation. Tab. 6.1 shows a brief comparison between the state-of-the-art and improved method for robustness prediction.

Table 6.1: A comparison between the state-of-the-art- and improved method for robustness evaluation. *number of static analyses required per design iteration **derivation of standard deviation

	State-of-the-art method	Improved method
Computation costs	> 1000 static analyses*	12 static analyses*
Reliability	numerical derivation** & limited to sample size	analytical derivation**

- The developed methodology developed in this thesis (see Section 4.2) allows for simulation of *three-dimensional* positioning of a component within rolled plate. The three-dimensional position of a component can be described by *four* degrees of freedom, being rotation about the x-, y- and z-axes and translation in z-direction. Translation in x- and y-direction are not relevant since residual stress in rolled plate only varies in z-direction and is constant in both x- and y-direction.

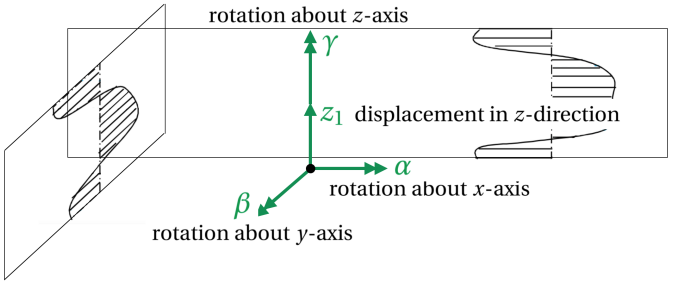


Figure 6.1: Four degrees of freedom that determine the three dimensional position of a component in rolled plate.

In the current state of the art, positioning of the component within rolled plate was limited to a *single* degree of freedom, being translation of the component in z-direction of rolled plate. Note that in the case study, only translation in z-direction and rotation about the y-axis were considered. The developed methodology, however, allows for rotation about the x- and z-axes as well.

- A mathematical stochastic representation of residual stress in rolled plate was developed showing *coherence* with the experimental data provided by AIRBUS (see Fig. 2.12). The reliability of the mathematical description is limited, however, since the experimental data concerns only twelve pieces of rolled plate. The fit function was chosen to be *symmetrical* since each individual measurement showed a high degree of symmetry.

- Distortion and distortion robustness can be obtained *analytically* for *prismatic* components. Robustness optimization problems concerning prismatic components can be solved based on analytical equations (see Section 3.8). Since aircraft components tend to have highly complex design features and are as a result highly non-prismatic, Finite Element Analyses will be resorted to for distortion and distortion robustness analyses.

The developed methodology in this thesis was employed in a case study in Chapter 5. In this case study, distortion robustness was evaluated for an aircraft component as a function of its position in rolled plate. Several conclusions regarding distortion robustness predictions were drawn in the case study.

- Robustness analyses are *highly relevant* since the dispersion of distortion is relatively large with respect to the magnitude of distortion. It would therefore be meaningless to only take into account deterministic distortion. For example, in some cases, distortion can be dispersed in such a way that components can distort in either a convex or a concave manner, depending from which rolled plate in the batch the component is manufactured (see Fig. 5.6).

- Robustness analyses are *highly relevant* since robustness is highly *correlated* with the positioning of a component within rolled plate (see Fig. 5.13). Positions can be found where distortion is significantly more robust than for other positions.

- For components that are *symmetrical* in shape about a certain plane, positions within rolled plate can be found where distortion is both *fully* robust (i.e. standard deviation equal to zero) and where distortion has a magnitude of zero (see Fig. 5.19). The component should be positioned in such a way that the plane about which the component is symmetrical *coincides* with the rolled plate's residual stress symmetry plane. At this position, distortion magnitude and -standard deviation are both equal to zero since the stochastic fit function is formulated in such a way that residual stress is always *symmetrical* with respect to the rolled plate's middle xy-symmetry plane. For each random sample drawn from the stochastic fit function, the bending moment induced by residual stress on the component will always be equal to zero.

- For components that are *not* symmetrical about a certain plane, still, positions can be found where distortion is both highly robust and approaches zero. This is since components can be asymmetrical, however, still can show a *degree* of symmetry. Components that show a high degree of symmetry have optima with high robustness; components that show a low degree of symmetry have optima that are less robust.

- *Non*-symmetrical components should be positioned in such a way that the plane about which the components is *most* symmetrical coincides with the rolled plate's residual stress symmetry plane.

- A component should be *designed* in such a way that is has a *high degree of symmetry* about a certain plane. This is, in literature, referred to by *design for distortion and robustness* which forms one of AIRBUS' future objectives in the scope of this research. This will be mentioned as a recommendation for future research in Chapter 7.

Chapter 7

Recommendations for future research

Several recommendations for future research can be made with respect to improving the computational efficiency of the developed methodology.

- The twelve static analyses that are required to determine distortion robustness in the improved method can be run *simultaneously*, since the twelve static analyses do not depend on one another. This way, computational efficiency can be improved by a factor of 12.

- Explore alternative fit functions for the stochastic experimental data. Fit functions can possible be found that have a lower number of stochastic variables (currently being twelve) as a result of which the computational costs can possibly be lowered even more.

- In the case study, the optimal position - in terms of offset z_1 and angle β - of the aircraft component was determined by assessing every combination of offset z_1 and angle β. In literature, this is referred to as *design space exploration*; the whole design space, with offset z_1 and angle β as design variables, has been explored for distortion robustness. At the end, the optimal combination of offset z_1 and angle β at which distortion was most robust was chosen *by observation*.

  One can imagine that this is a time consuming process. In the future, gradient-based optimization can be introduced. A gradient-based optimizer uses a optimization sequence in which the choice of the subsequent design iteration depends on both the output and gradient information of the previous design iteration. This way, not the entire design space has to be explored, but the optimal combination of the design variables can be found by converging to local and global minima.

  It must be said that a great deal of static analyses would be required when employing gradient-based optimization, since gradients (also referred to by *sensitivies*) must be determined for each of the twelve displacement fields per design iteration. Furthermore, when introducing more design variables, for each design variable extra sensitivities must be determined. For an optimization problem containing many design variables, so-called *adjoint sensitivity* can be used in order to scale down the number of static analyses needed for determining sensitvities.

Several recommendations for future research can be made with respect to improving the reliability of the distortion robustness predictions.

- Improve reliability of mathematical stochastic representation of the experimentally measured residual stress data by increasing the sample size, i.e. by increasing the number of rolled plate's that are measured.
- Compare the developed methodology for distortion and robustness analysis with *reality*. See whether a machined component at a certain position in a number of pieces of rolled plate shows the same distortion behavior as is predicted with the developed simulation tool.

Several recommendations for future research can be made with respect to *design for distortion robustness*.

- Up to now, distortion robustness was optimized by varying the *position* of components within rolled plate. One of AIRBUS' objectives, however, is to change the component's *design* itself in order to maximize distortion robustness. This is referred to in literature as *design for distortion robustness*. You could say that so far we dealt with *positioning for distortion robustness*. To this end, *topology optimization* would be a suitable method. The material layout itself will be optimized in such a way that distortion robustness is maximized.

  Since topology optimization basically is an optimization problem with many design variables, adjoint sensitivities should be resorted to in order to save computation costs for the sensitivity analyses. When a computational efficient method has been developed for the sensitivity analyses, the developed methodology in this thesis can be implemented in a topology optimization sequence.

- As was already mentioned in the conclusions, the case study demonstrated that the degree of symmetry of a component design relates to the degree of robustness. For a design with a high degree of symmetry about a certain plane, distortion would be highly robust and approach to zero when the component is positioned in such a way that its symmetry plane coincides with the rolled plate's residual stress symmetry plane.

  This knowledge can be applied to the topology optimization sequence. The optimization problem can be formulated in such a way that a *symmetrical* design is more favorable than a non-symmetrical design.

Bibliography

[1] Topology Optimization airbus. http://www.topology-opt.com/2011/03/airbus/. Accessed: 2018-09-06.

[2] W.M. Sim. Residual stress engineering in manufacture of aerospace structural parts. *Filton, UK, Airbus S.A.S*, 2009.

[3] W.M. Sim. Challenges of residual stress and part distortion in the civil airframe industry. *International Journal of Microstructure and Materials Properties*, 5(4/5):446–455, 2010.

[4] H. Karaouni, B. Souvestre, and Y. Ahipo. Machining advanced simulation: Distortion prediction of prestressed machined parts in ncsimul environment. *Proceedings of IDMME - Virtual Concept 2010*, 2010.

[5] D. Chantzisa, S. Van der Veen, J.Zettler, and W.M. Sim. An industrial workflow to minimise part distortion for machining of large monolithic components in aerospace industry. In Conference Chair Prof. Luca Settineri, editor, *14th CIRP Conference on Modeling of Machining Operations (CIRP CMMO)*, Filton, Bristol, UK, 6 2013. Modeling of Machining Operations, Elsevier.

[6] M.S. Younger and K.H. Eckelmeyerr. Overcoming Residual Stresses and Machining Distortion in the Production of Aluminum Alloy Satellite Boxes. Technical report, Sandia National Laboratories, 11 2007.

[7] Th. Luebben, B. Clausen, Chr. Prinz, A. Schulz, R. Rentsch, R. Kusmierz, H. Surm L. Nowag, F. Frerichs, M. Hunkel, and P. Mayr. Distortion engineering: A system-oriented view on the distortion of component-parts. In Klaus-Dieter Thoben and Dieter Klein, editors, *Jinshu Rechuli/Heat Treatment of Metals*, pages 43–50, Bremen, Germany, 11 2004. Bremen Institute of Industrial Technology and Applied Work Science (BIBA), Institute for Material Science (IWT).

[8] D.M. Bowden and J.E. Halley. Aluminium reliability improvement program. Technical report, The Boeing Company, Chicago, IL, USA, 2001. Final Report.

[9] W.M. Sim. Deliverable publishable final activity report version 1.0 - a concurrent approach to manufacturing induced part distortion in aerospace components (compact). Technical report, AIRBUS in the UK (AUK), AIRBUS in the UK (AUK), 9 2009.

[10] K.K. Gadhamsetty and V.G.Araveti. Machining distortion prediction - design principles rp1221201. Technical report, AIRBUS Engineering Centre India, Unit 301, 3rd

Floor, Tower B, RMZ Infinity campus,No 3, Old Madras Road,Bangalore 560016,Karnataka,India, 6 2012.

[11] K.K. Gadhamsetty, V.G. Araveti, S. Thakur, and J. Willaume. Machining Distortion Prediction - RP1144484. Technical report, AIRBUS Engineering Centre India, 12 2011.

[12] Demands of titanium milling in aerospace. https://www.mscdirect.com/betterMRO/metalworking/demands-titanium-milling-aerospace. Accessed: 2019-01-01.

[13] Y.C.E. Janssens. Multiple step reductive machining of aircraft components from prestressed stock material and its influence on residual stress and part distortion. Technical report, Delft University of Technology and AIRBUS Operations S.A.S., Toulouse, France, 7 2017.

[14] P.J. Withers and H.K.D.H. Bhadeshia. Residual stress. part 2 – nature and origins. *Materials Science and Technology*, 17(4):366–375, 2001.

[15] G.S. Schajer. Overview of residual stresses and their measurement. In G.S. Schajer and C.O. Ruud, editors, *Practical residual stress measurement methods*, chapter 1, pages 1–27. John Wiley and Sons, 2013.

[16] Titanium sheets and plates manufacturer. https://www.jaydeepsteels.com/titanium-sheets-plates-supplier-exporter/. Accessed: 2019-04-01.

[17] Hot working and cold working (forging, extrusion, rolling). http://mech413.blogspot.com/2013/07/hot-working-and-cold-working-forging.html. Accessed: 2019-01-01.

[18] Plate rolling gets hot. https://www.thefabricator.com/article/bending/plate-rolling-gets-hot. Accessed: 2019-04-01.

[19] X. Ficquet, C. E. Truman, E. Kingston, and D. J. Smith. Measurement of residual stresses in aluminium alloy aerospace components. In *25th international congress of the aeronautical science*. Department of Mechanical Engineering, University of Bristol, U.K., 2006.

[20] J.S. Robinson, S. Hossain, C.E. Truman, A.M. Paradowska, D.J. Hughes, R.C. Wimpory, and M.E. Fox. Residual stress in 7449 aluminium alloy forgings. *Materials Science and Engineering: A*, 2008.

[21] M.B. Prime and M.R. Hill. Residual stress, stress relief, and inhomogeneity in aluminum plate. *Scripta Materialia*, 46(1):77–82, 2002.

[22] A new approach to titanium machining. https://www.radical-departures.net/articles/a-new-approach-to-titanium-machining/. Accessed: 2019-04-01.

[23] T. Segawa, H. Sasahara, and M. Tsutsumi. Development of a new tool to generate compressive residual stress within a machined surface. *International Journal of Machine Tools and Manufacture*, 44:1215–1221, 3 2004.

[24] T.D. Marusich and E. Askari. Modeling residual stress and workpiece quality in machined surfaces. Technical report, Third Wave Systems Inc and Boeing, Minneapolis, MN USA, 2003.

[25] S. Dreier and B. Denkena. Determination of residual stresses in plate material by layer removal with machine-integrated measurement. In B. Denkena, editor, *New Production Technologies in Aerospace Industry - Machining Innovations Conference (MIC)*, Procedia CIRP 24, pages 103–107, Leibniz University of Hannover, Institute of Production Engeneering and Machine Tools, Germany, 2014. Elsevier B.V.

[26] Inc. The MathWorks. lsqcurvefit. https://nl.mathworks.com/help/optim/ug/lsqcurvefit.html, 2018.

[27] J. Wolberg. The method of least-squares: Extracting the most information from experiments. In *Data analysis using the method of least-squares*, chapter 2. Springer Science and Business media, Berlin, Germany, 2006.

[28] C. Kraaikamp, F. M. Dekking, H.P. Lopuhaa, and L.E. Meester. Discrete random variables. In *A Modern Introduction to Probability and Statistics: Understanding Why and How*, chapter 4, pages 41–51. Springer, 2005.

[29] E. Macherauch. Introduction to residual stress. In A. Niku-Lari, editor, *Advances in Surface Treatments: Technology-Application-Effect*, volume 4, chapter 1, pages 1–6. Pergamon Press, Oxford, UK, 1986.

[30] F. Heymes, B. Commet, B. Dubost, P. Lassince, P. Lequeu, and G.M. Raynaud. Development of new al alloys for distortion free machined aluminium aircraft components. In T. Bains, D. S.Mackenzie, and G. M. Davidson, editors, *International Non-Ferrous Processing and Technology Conference*, volume 1, pages 249–258, Materials Park, Ohio, 1997. ASM International.

[31] A. van Keulen. *Advanced Mechanics Part: Continuum Mechanics (course ME46000 Nonlinear Mechanics)*. 2013.

[32] J.S. Robinson, D.A. Tanner, and C.E. Truman. Measurement and prediction of machining induced redistribution of residual stress in the aluminium alloy 7449. *Experimental Mechanics*, 51(6):981–993, 7 2011.

[33] M.E. Kartal. Analytical solutions for determining residual stresses in two-dimensional domains using the contour method. In *Proceedings of the Royal Society A*, volume 469, Oxford University, Parks Road, Oxford OX1 3PJ, UK, 2013. Department of Engineering Science, Royal Society Publishing.

[34] Z. Zhang, L. Li, Y. Yang, N. He, and W. Zhao. Machining distortion minimization for the manufacturing of aeronautical structure. *International Journal of Advanced Manufacturing Technology*, (73):1765–1773, August 2014.

[35] R.C. Hibbeler. *Mechanics of Materials*, chapter Hooke's Law, pages 90–91. Pearson Education, Upper Saddle River, N.J., 8th edition, 2011.

[36] R.C. Hibbeler. *Mechanics of Materials*, chapter The Flexure Formula, pages 285–287. Pearson Education, Upper Saddle River, N.J., 8th edition, 2011.

[37] R.C. Hibbeler. *Mechanics of Materials*, chapter The Elastic Curve, pages 569–572. Pearson Education, Upper Saddle River, N.J., 8th edition, 2011.

[38] R.C. Hibbeler. *Mechanics of Materials*, chapter Slope and Displacement by Integration, pages 573–574. Pearson Education, Upper Saddle River, N.J., 8th edition, 2011.

[39] R.C. Hibbeler. *Mechanics of Materials*, chapter Axial Load, pages 119–164. Pearson Education, Upper Saddle River, N.J., 8th edition, 2011.

[40] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaa, and L.E. Meester. *A Modern Introduction to Probability and Statistics*, chapter Discrete Random Variables, pages 41–51. Springer, Delft, The Netherlands, 8th edition, 2005.

[41] Structural bracket for airbus a350. https://weststarprecision.com/aircraft-structural/.

[42] Aircraft engine component airbus. http://www.wsiindustries.com/aerospace/.

[43] A. van Keulen. Eindige-elementenmethode. In *Advanced Mechanics Part: Continuum Mechanics (course ME46000 Nonlinear Mechanics)*, chapter 4, pages 93–112. 2013.

[44] G.A. Holzapfel. The concept of stress. In *Nonlinear Solid Mechanics: a Continuum Approach for Engineering*, chapter 1, pages 109–127. John Wiley and Sons, LTD, 2000.

[45] G.A. Holzapfel. Transformation laws for basis vectors and components. In *Nonlinear Solid Mechanics: a Continuum Approach for Engineering*, chapter 1, pages 28–32. John Wiley and Sons, LTD, 2000.

Appendix A

Analytical distortion derivations

A.1 Analytical distortion derivation of a beam cut in half

This appendix concerns the analytical derivation for distortion of a piece of stock material that is cut in half, like is illustrated in Fig. A.1.

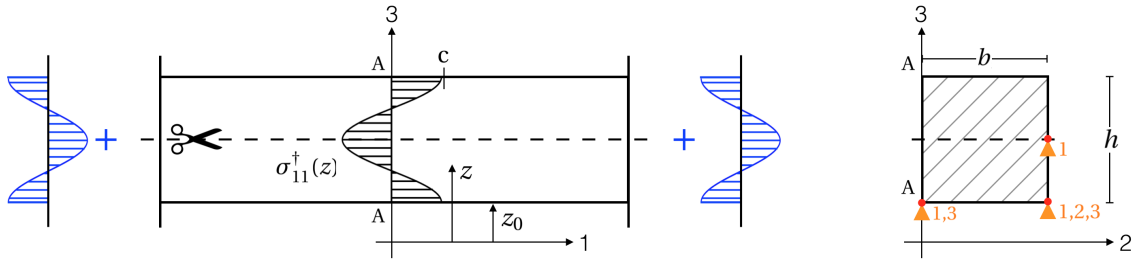


Figure A.1: This figure shows a beam or piece of stock material with rectangular cross section that is about to be cut in half as a result of which distortion will occur in the form of bending. Details shown in this figure were elaborated earlier in Fig. 3.9.

The area of the cross section, A_x

$$A_x = \int_{y=0}^{b} \int_{z=z_0}^{z_0+\frac{h}{2}} dz\,dy = y\Big|_0^b z\Big|_{z_0}^{z_0+\frac{h}{2}} = \frac{1}{2}bh$$

The first moment of area, S_y

$$S_y = \int_{y=0}^{b} \int_{z=z_0}^{z_0+\frac{h}{2}} z\,dz\,dy = y\Big|_0^b \frac{1}{2}z^2\Big|_{z_0}^{z_0+\frac{h}{2}} = \frac{1}{2}bh\left(z_o + \frac{1}{4}h\right)$$

The z-coordinate of the geometric centroid, $\overline{z}$

$$\overline{z} = \frac{S_y}{A_x} = \frac{\frac{1}{2}bh\left(z_o + \frac{1}{4}h\right)}{\frac{1}{2}bh} = z_0 + \frac{1}{4}h$$

A mathematical description for $\sigma^{\dagger}_{11}(z)$

$$\sigma^{\dagger}_{11}(z) = c\cos\big(\beta(z - z_0)\big) \qquad \text{with } \beta = \frac{2\pi}{h}$$

The moment, M_y, induced by $\sigma_{11}^{\dagger}(z)$

$$\begin{aligned}
M_{\mathrm{y}} &= \int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} \sigma_{11}^{\dagger}(z)(z-\overline{z})\,dz\,dy \\
&= \int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} c\cos\big(\beta(z-z_0)\big)\big(z-\overline{z}\big)\,dz\,dy \\
&= c\int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} z\cos\big(\beta(z-z_0)\big)\,dz\,dy - c\overline{z}\int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} \cos\big(\beta(z-z_0)\big)\,dz\,dy \\
&= \frac{c}{\beta^2}y\Big|_0^b\big(\cos\big(\beta(z-z_0)\big)+\beta z\sin\big(\beta(z-z_0)\big)\big)\Big|_{z_0}^{z_0+\frac{h}{2}} - \frac{c\overline{z}}{\beta}y\Big|_0^b\sin\big(\beta(z-z_0)\big)\Big|_{z_0}^{z_0+\frac{h}{2}} \\
&= \frac{cb}{\beta^2}\left[-1+0-(1+0)\right] - \frac{cb\overline{z}}{\beta}\left[0-0\right] \\
&= -\frac{bch^2}{2\pi^2}
\end{aligned}$$

The second moment of area about the neutral y-axis, I_y

$$\begin{aligned}
I_y &= \int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} \left(z-\left(z_0+\frac{1}{4}h\right)\right)^2 dz\,dy \\
&= \int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} \left(z^2-2z\left(z_0+\frac{1}{4}h\right)+\left(z_0+\frac{1}{4}h\right)^2\right) dz\,dy \\
&= \int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} z^2\,dz\,dy - 2\left(z_0+\frac{1}{4}h\right)\int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} z\,dz\,dy + \left(z_0+\frac{1}{4}h\right)^2\int_{y=0}^{b}\int_{z=z_0}^{z_0+\frac{h}{2}} dz\,dy \\
&= y\Big|_0^b\frac{1}{3}z^3\Big|_{z_0}^{z_0+\frac{h}{2}} - 2\left(z_0+\frac{1}{4}h\right)y\Big|_0^b\frac{1}{2}z^2\Big|_{z_0}^{z_0+\frac{h}{2}} + \left(z_0+\frac{1}{4}h\right)^2 y\Big|_0^b z\Big|_{z_0}^{z_0+\frac{h}{2}} \\
&= \frac{1}{3}b\left(\left(z_0+\frac{h}{2}\right)^3-z_0^3\right) - b\left(z_0+\frac{1}{4}h\right)\left(\left(z_0+\frac{h}{2}\right)^2-z_0^2\right) + b\left(z_0+\frac{1}{4}h\right)^2\left(\left(z_0+\frac{h}{2}\right)-z_0\right) \\
&= \frac{bh^3}{96}
\end{aligned}$$

Deflection of the beam $w_z(x)$ at $x=l$ gives

$$w_z(x{=}l) = \frac{M_y l^2}{2EI_y} = -\frac{24cl^2}{Eh\pi^2}$$

Substituting geometrical- and material properties l, h, c and E as were declared earlier in chapter 3.4 gives

$$\begin{aligned}
w_z(x{=}l) &= -0.0231\,\mathrm{m} \\
&= -23.1\,\mathrm{mm}
\end{aligned}$$

A.2 Analytical distortion derivation of a beam machined from both sides

This appendix concerns the analytical derivation for distortion of a piece of stock material that is symmetrically machined from both sides, like is illustrated in Fig. A.2.

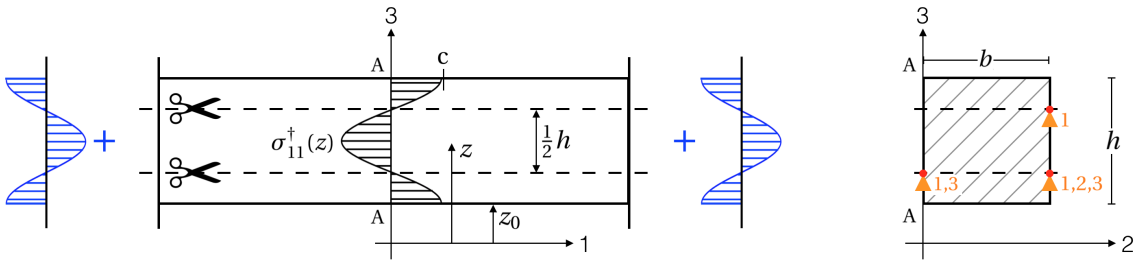


Figure A.2: This figure shows a beam or piece of stock material with rectangular cross section that is about to be symmetrically machined from top and bottom as a result of which distortion will occur in the form of contraction. Details shown in this figure were elaborated earlier in Fig. 3.13.

The area of the cross section, A_x

$$A_x = \int_{y=0}^{b} \int_{z=z_0+\frac{h}{4}}^{z_0+\frac{3h}{4}} dz\,dy = y\Big|_0^b z\Big|_{z_0+\frac{h}{4}}^{z_0+\frac{3h}{4}} = \frac{1}{2}bh$$

A mathematical description for $\sigma^{\dagger}_{11}(z)$

$$\sigma^{\dagger}_{11}(z) = c\cos\big(\beta(z-z_0)\big) \qquad \text{with } \beta = \frac{2\pi}{h}$$

The force, F_x, induced by $\sigma^{\dagger}_{11}(z)$

$$\begin{aligned}
F_{\mathrm{x}} &= \int_{y=0}^{b} \int_{z=z_0+\frac{h}{4}}^{z_0+\frac{3h}{4}} \sigma^{\dagger}_{11}(z)\,dz\,dy \\
&= \int_{y=0}^{b} \int_{z=z_0+\frac{h}{4}}^{z_0+\frac{3h}{4}} c\cos\big(\beta(z-z_0)\big)\,dz\,dy \\
&= y\Big|_0^b \frac{c}{\beta}\sin\big(\beta(z-z_0)\big)\Big|_{z_0+\frac{h}{4}}^{z_0+\frac{3h}{4}} \\
&= \frac{bc}{\beta}\left[\sin\left(\beta\frac{3h}{4}\right) - \sin\left(\beta\frac{h}{4}\right)\right] \\
&= \frac{bc}{\beta}\left[-1-1\right] \\
&= -\frac{bch}{\pi}
\end{aligned}$$

Displacement of the beam in x-direction at $x = l$, $w_x(x{=}l)$, is obtained by

$$w_x(x{=}l) = \frac{F_x l}{A_x E} = -\frac{2cl}{\pi E}$$

Substituting geometrical- and material properties l, c and E as were declared earlier in chapter 3.4 gives

$$\begin{aligned} w_z(x{=}l) &= -0.000\,182\,\mathrm{m} \\ &= -0.182\,\mathrm{mm} \end{aligned}$$

A.3 Analytical distortion derivation of a tee section

This appendix concerns the analytical derivation for distortion of a Tee Section that is machined from a piece of stock material, like is illustrated in Fig. A.3.

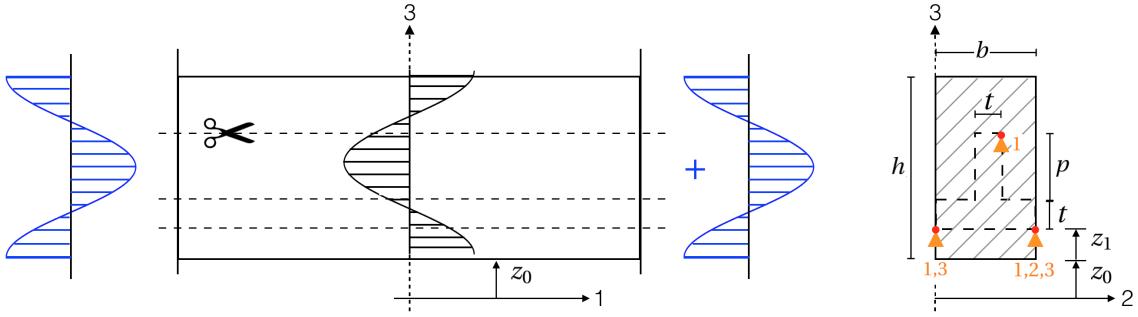


Figure A.3: This figure shows a piece of stock material from which a Tee Section (dotted lines) is to be manufactured. The Tee Section is machined with an offset of z_1 with respect to the rolled plate's bottom face. This is the same figure as Fig. 3.16

The area of the cross section, A_x

$$\begin{aligned}
A_x &= \int_y \int_z dz\,dy \\
&= \int_{y=0}^{b} \int_{z=z_0+z_1}^{z_0+z_1+t} dz\,dy + \int_{y=\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}} \int_{z=z_0+z_1+t}^{z_0+z_1+t+p} dz\,dy \\
&= y\Big|_0^b z\Big|_{z_0+z_1}^{z_0+z_1+t} + y\Big|_{\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}} z\Big|_{z_0+z_1+t}^{z_0+z_1+t+p} \\
&= bt + pt
\end{aligned}$$

The first moment of area, S_y

$$\begin{aligned}
S_y &= \int_y \int_z z\,dz\,dy \\
&= \int_{y=0}^{b} \int_{z=z_0+z_1}^{z_0+z_1+t} z\,dz\,dy + \int_{y=\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}} \int_{z=z_0+z_1+t}^{z_0+z_1+t+p} z\,dz\,dy \\
&= y\Big|_0^b \frac{1}{2} z^2\Big|_{z_0+z_1}^{z_0+z_1+t} + y\Big|_{\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}} \frac{1}{2} z^2\Big|_{z_0+z_1+t}^{z_0+z_1+t+p} \\
&= \frac{pt}{2}\left(p+2t+2z_0+2z_1\right) + \frac{bt}{2}\left(t+2z_0+2z_1\right)
\end{aligned}$$

The z-coordinate of the geometric centroid, $\overline{z}$

$$\overline{z} = \frac{S_y}{A_x} = \frac{\frac{pt}{2}\left(p+2t+2z_0+2z_1\right) + \frac{bt}{2}\left(t+2z_0+2z_1\right)}{bt+pt}$$

A mathematical description for $\sigma_{11}^{\dagger}(z)$

$$\sigma_{11}^{\dagger}(z) = c\cos\left(\beta(z-z_0)\right) \qquad \text{with } \beta = \frac{2\pi}{h}$$

The moment, M_y, induced by $\sigma^{\dagger}_{11}(z)$, with help of MATLAB® symbolic integration

$$
\begin{aligned}
M_{\mathrm{y}} &= \int_{y=0}^{b}\int_{z=z_0+z_1}^{z_0+z_1+t} \sigma^{\dagger}_{11}(z)(z-\overline{z})\,dz\,dy + \int_{y=\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}}\int_{z=z_0+z_1+t}^{z_0+z_1+t+p} \sigma^{\dagger}_{11}(z)(z-\overline{z})\,dz\,dy \\
&= c\,t\left(\frac{\cos\left(\beta\left(p+t+\mathrm{z}_1\right)\right)+\beta\sin\left(\beta\left(p+t+\mathrm{z}_1\right)\right)\left(p+t+\mathrm{z}_0+\mathrm{z}_1\right)}{\beta^2}\right) \\
&- c\,t\left(\frac{\cos\left(\beta\left(t+\mathrm{z}_1\right)\right)+\beta\sin\left(\beta\left(t+\mathrm{z}_1\right)\right)\left(t+\mathrm{z}_0+\mathrm{z}_1\right)}{\beta^2}\right) \\
&- b\,c\left(\frac{\cos\left(\beta\,\mathrm{z}_1\right)+\beta\sin\left(\beta\,\mathrm{z}_1\right)\left(\mathrm{z}_0+\mathrm{z}_1\right)}{\beta^2} - \frac{\cos\left(\beta\left(t+\mathrm{z}_1\right)\right)+\beta\sin\left(\beta\left(t+\mathrm{z}_1\right)\right)\left(t+\mathrm{z}_0+\mathrm{z}_1\right)}{\beta^2}\right) \\
&- b\,c\,\overline{z}\left(\frac{\sin\left(\beta\left(t+\mathrm{z}_1\right)\right)-\sin\left(\beta\,\mathrm{z}_1\right)}{\beta}\right) \\
&+ c\,t\,\overline{z}\left(\frac{\sin\left(\beta\left(t+\mathrm{z}_1\right)\right)-\sin\left(\beta\left(p+t+\mathrm{z}_1\right)\right)}{\beta}\right)
\end{aligned}
$$

The second moment of area about the neutral y-axis, I_y, with help of MATLAB® symbolic integration

$$
\begin{aligned}
I_y &= \int_{y=0}^{b}\int_{z=z_0+z_1}^{z_0+z_1+t} (z-\overline{z})^2\,dz\,dy + \int_{y=\frac{b}{2}-\frac{t}{2}}^{\frac{b}{2}+\frac{t}{2}}\int_{z=z_0+z_1+t}^{z_0+z_1+t+p} (z-\overline{z})^2\,dz\,dy \\
&= t\left(\overline{z}\left(t+\mathrm{z}_0+\mathrm{z}_1\right)^2 - \overline{z}^2\left(t+\mathrm{z}_0+\mathrm{z}_1\right) - \frac{\left(t+\mathrm{z}_0+\mathrm{z}_1\right)^3}{3} - \overline{z}\left(p+t+\mathrm{z}_0+\mathrm{z}_1\right)^2\right) \\
&+ t\left(\overline{z}^2\left(p+t+\mathrm{z}_0+\mathrm{z}_1\right) + \frac{\left(p+t+\mathrm{z}_0+\mathrm{z}_1\right)^3}{3}\right) \\
&+ \frac{b\,t\left(t^2+3\,t\,\mathrm{z}_0+3\,t\,\mathrm{z}_1-3\,t\,\overline{z}+3\,{\mathrm{z}_0}^2+6\,\mathrm{z}_0\,\mathrm{z}_1-6\,\mathrm{z}_0\,\overline{z}+3\,{\mathrm{z}_1}^2-6\,\mathrm{z}_1\,\overline{z}+3\,\overline{z}^2\right)}{3}
\end{aligned}
$$

The expressions for M_y and I_y displayed above were already quite complex. The final expression for the deflection of the Tee Section's tip at $x = l$, $w_z(x{=}l)$ is not displayed in this appendix since this expression contains too many terms. $w_z(x{=}l)$ is obtained by means of MATLAB® symbolic integration which can be found in m-file *fanalyticalderivation.m* shown in Fig. A.1. The full expression for $w_z(x{=}l)$ is shown in m-file *fwz.m* shown in Fig. A.2. Substituting geometrical- and material properties as were stated in chapter 3.5, yields a deflection of

$$
\begin{aligned}
w_z(x{=}l) &= \frac{M_y l^2}{2EI_y} \\
&= -0.0069\,\mathrm{m} \\
&= -6.9\,\mathrm{mm}
\end{aligned}
$$

Matlab code A.1: fanalayticalderivation.m

```
1 clear; close all; clc;
```

```
2
3  syms c h z z0 z1 t b y p l E
4  beta=2*pi/h;
5  sigma11=c*cos(beta*(z-z0));
6
7  Ax=int(int(1,z,(z0+z1),(z0+z1+t)),y,0,b)+int(int(1,z,(z0+z1+t),...
8      (z0+z1+t+p)),y,(b/2-t/2),(b/2+t/2))
9  Sy=int(int(z,z,(z0+z1),(z0+z1+t)),y,0,b)+int(int(z,z,(z0+z1+t),...
10     (z0+z1+t+p)),y,(b/2-t/2),(b/2+t/2))
11 z_=Sy/Ax
12
13 f=sigma11*(z-z_);
14 My=int(int(f,z,(z0+z1),(z0+z1+t)),y,0,b)+int(int(f,z,(z0+z1+t),...
15     (z0+z1+t+p)),y,(b/2-t/2),(b/2+t/2))
16 g=(z-z_)^2;
17 Iy=int(int(g,z,(z0+z1),(z0+z1+t)),y,0,b)+int(int(g,z,(z0+z1+t),...
18     (z0+z1+t+p)),y,(b/2-t/2),(b/2+t/2))
19
20 wz=(My*l^2)/(2*E*Iy)
```

Matlab code A.2: fwz.m

```
1  function wz=fwz(l,b,h,p,t,z0,z1,c,E)
2  wz=-(l^2*(b*c*((h^2*cos((2*pi*z1)/h))/(4*pi^2) ...
3      - (h^2*cos((2*pi*(t + z1))/h))/(4*pi^2) ...
4      + (h*sin((2*pi*z1)/h)*(z0 + z1))/(2*pi) ...
5      - (h*sin((2*pi*(t + z1))/h)*(t + z0 + z1))/(2*pi)) ...
6      - c*t*((h^2*cos((2*pi*(p + t + z1))/h))/(4*pi^2) ...
7      - (h^2*cos((2*pi*(t + z1))/h))/(4*pi^2) ...
8      + (h*sin((2*pi*(p + t + z1))/h)*(p + t + z0 + z1))/(2*pi) ...
9      - (h*sin((2*pi*(t + z1))/h)*(t + z0 + z1))/(2*pi)) ...
10     + (b^2*c*h*t^2*(sin((2*pi*(t + z1))/h) ...
11     - sin((2*pi*z1)/h)))/(4*pi*(b*t + p*t)) ...
12     - (c*h*p^2*t^2*(sin((2*pi*(t + z1))/h) ...
13     - sin((2*pi*(p + t + z1))/h)))/(4*pi*(b*t + p*t)) ...
14     - (b*c*h*t^3*(sin((2*pi*(t + z1))/h) ...
15     - sin((2*pi*(p + t + z1))/h)))/(4*pi*(b*t + p*t)) ...
16     - (c*h*p*t^3*(sin((2*pi*(t + z1))/h) ...
17     - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
18     + (b*c*h*p*t^2*(sin((2*pi*(t + z1))/h) ...
19     - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
20     + (b*c*h*p^2*t*(sin((2*pi*(t + z1))/h) ...
21     - sin((2*pi*z1)/h)))/(4*pi*(b*t + p*t)) ...
22     + (b^2*c*h*t*z0*(sin((2*pi*(t + z1))/h) ...
23     - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
24     + (b^2*c*h*t*z1*(sin((2*pi*(t + z1))/h) ...
25     - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
26     - (b*c*h*t^2*z0*(sin((2*pi*(t + z1))/h) ...
27     - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
28     - (b*c*h*t^2*z1*(sin((2*pi*(t + z1))/h) ...
29     - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
30     - (c*h*p*t^2*z0*(sin((2*pi*(t + z1))/h) ...
31     - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
32     - (c*h*p*t^2*z1*(sin((2*pi*(t + z1))/h) ...
```

```
33      - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
34      + (b*c*h*p*t*z0*(sin((2*pi*(t + z1))/h) ...
35      - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
36      + (b*c*h*p*t*z1*(sin((2*pi*(t + z1))/h) ...
37      - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t))))/(2*E*((b*t^3)/12 ...
38      - (b*((p^2*t^3)/12 - (t*(3*p^4 + 6*p^3*t ...
39      + 4*p^2*t^2))/12))/(b + p)^2 ...
40      + (p*t*(4*b^2*p^2 + 6*b^2*p*t + 3*b^2*t^2 ...
41      + 2*b*p^3 + p^4))/(12*(b + p)^2)));
```

Appendix B

Distortion and robustness optimization based on analytical equations

B.1 Distortion optimization of a Tee Section for offset z_1

This appendix includes the MATLAB® code used for an optimization problem where the optimal offset, z_1^*, is to be found where distortion, $w_z(x{=}l, z_1^*)$, is minimized. The *fmincon* solver in MATLAB® is used to solve this optimization problem.

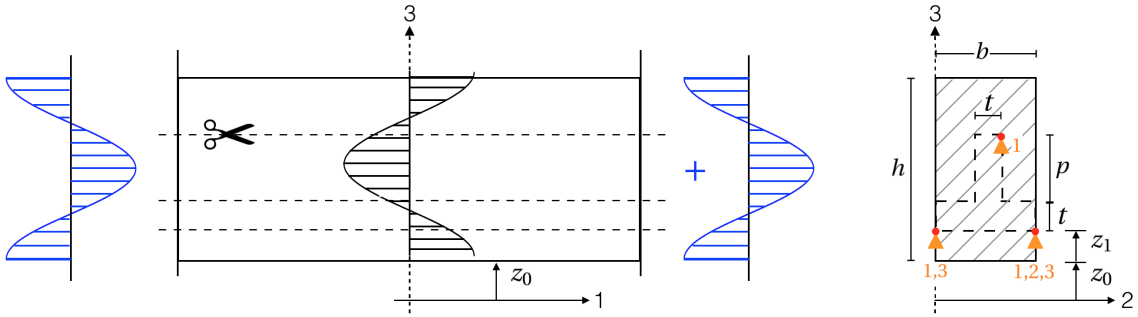


Figure B.1: This figure shows the same illustration as in Fig. 3.16 where a Tee Section is to be manufactured at an offset z_1 from the bottom face of a piece of rolled plate material. The optimal offset z_1^* is to be found where distortion is minimized.

The objective function relating the objective $f(z_1) = w_z(x{=}l, z_1)^2$ with offset z_1, as was elaborated in chapter 3.6, is captured in a separate function m-file, *objective.m*, shown in Fig. B.1. The expression for $w_z(x{=}l, z_1)$ was analytically derived in chapter 3.5.

The fmincon code is captured in a second m-file, *fminconsolve.m*, shown in Fig. B.2. The offset z_1^* where distortion $w_z(x{=}l, z_1^*)$ is minimal was found to be

$$z_1^* = 0.0219\,\text{m} = 21.9\,\text{mm}$$

At which distortion is found to be

$$w_z(x{=}l, z_1^*) = 1.0684 \times 10^{-13}\,\text{m}$$

Matlab code B.1: objective.m

```
function f=objective(l,b,h,p,t,z0,z1,c,E)
wz=-(l^2*(b*c*((h^2*cos((2*pi*z1)/h))/(4*pi^2) ...
    - (h^2*cos((2*pi*(t + z1))/h))/(4*pi^2) ...
    + (h*sin((2*pi*z1)/h)*(z0 + z1))/(2*pi) ...
    - (h*sin((2*pi*(t + z1))/h)*(t + z0 + z1))/(2*pi)) ...
    - c*t*((h^2*cos((2*pi*(p + t + z1))/h))/(4*pi^2) ...
    - (h^2*cos((2*pi*(t + z1))/h))/(4*pi^2) ...
    + (h*sin((2*pi*(p + t + z1))/h)*(p + t + z0 + z1))/(2*pi) ...
    - (h*sin((2*pi*(t + z1))/h)*(t + z0 + z1))/(2*pi)) ...
    + (b^2*c*h*t^2*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(4*pi*(b*t + p*t)) ...
    - (c*h*p^2*t^2*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(4*pi*(b*t + p*t)) ...
    - (b*c*h*t^3*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(4*pi*(b*t + p*t)) ...
    - (c*h*p*t^3*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
    + (b*c*h*p*t^2*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
    + (b*c*h*p^2*t*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(4*pi*(b*t + p*t)) ...
    + (b^2*c*h*t*z0*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
    + (b^2*c*h*t*z1*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
    - (b*c*h*t^2*z0*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
    - (b*c*h*t^2*z1*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
    - (c*h*p*t^2*z0*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
    - (c*h*p*t^2*z1*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*(p + t + z1))/h)))/(2*pi*(b*t + p*t)) ...
    + (b*c*h*p*t*z0*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t)) ...
    + (b*c*h*p*t*z1*(sin((2*pi*(t + z1))/h) ...
    - sin((2*pi*z1)/h)))/(2*pi*(b*t + p*t))))/(2*E*((b*t^3)/12 ...
    - (b*((p^2*t^3)/12 - (t*(3*p^4 + 6*p^3*t ...
    + 4*p^2*t^2))/12))/(b + p)^2 ...
    + (p*t*(4*b^2*p^2 + 6*b^2*p*t + 3*b^2*t^2 ...
    + 2*b*p^3 + p^4))/(12*(b + p)^2)));
f=wz^2;
```

Matlab code B.2: fminconsolve.m

```
close all; hold off; clear; clc;

%constants
l=1.000;
b=0.040;
h=0.080;
p=0.030;
t=0.010;
```

```
c=20*10^6;
E=70.07*10^9;
z0=0.100;

%plot of objective function
n=100;
z1_vec=linspace(0,(p+t),n);
for i=1:n
    f(i)=objective(l,b,h,p,t,z0,z1_vec(i),c,E)
end
figure(1)
plot(z1_vec,f)
grid on
set(gca,'fontsize',11.5)

%fmincon
A=[];
B=[];
Aeq=[];
Beq=[];
LB=0;
UB=p+t;
z1_0=t;
f_0=objective(l,b,h,p,t,z0,z1_0,c,E);
[z1,f_val]=fmincon(@(z1) ...
   objective(l,b,h,p,t,z0,z1,c,E),z1_0,A,B,Aeq,Beq,LB,UB)
```

B.2 Distortion optimization of a tee section for offset z_1 and flank thickness t

In this appendix, the MATLAB® code is shown for an optimization problem where offset z_1 and flank thickness t of the Tee Section in Fig. B.2 manufactured from a piece of rolled plate are optimized for distortion.

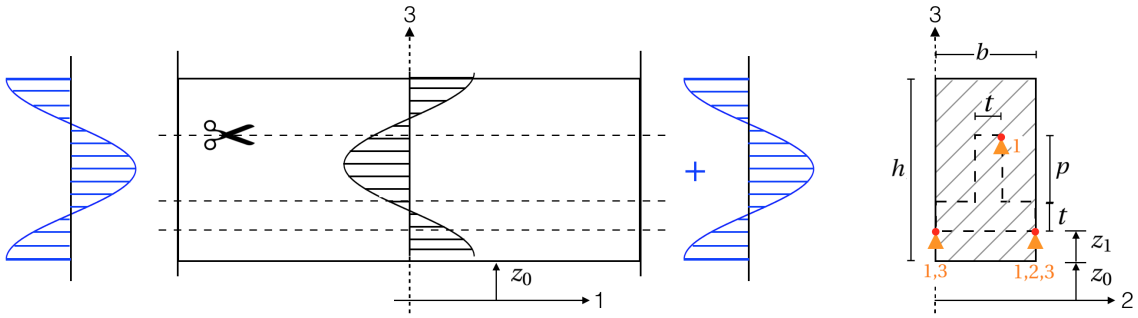


Figure B.2: This figure shows the same illustration as in Fig. 3.16. The optimal offset z_1 and flank thickness t is to be found where distortion is minimized.

The m-file *fcontourplot.m*, shown in Fig. B.3, contains the MATLAB® code used for creating the contour plot. The inequality constraints are coded in a separate function m-file, *fconstraints.m*, shown in Fig. B.4.

Matlab code B.3: fcontourplot.m

```
clf; hold off; clear; clc; close all;

%constants
l=1.000;
b=0.040;
h=0.080;
p=0.030;
c=20*10^6;
E=70.07*10^9;
z0=0.100;

%contour plot and surf plot
n=100;
z1_vec=linspace(0,h,n);
t_vec=linspace(0,b,n);

for i=1:1:length(z1_vec)
    for j=1:1:length(t_vec)
        z1=z1_vec(i);
        t=t_vec(j);
        f(j,i)=objective(l,b,h,p,t,z0,z1,c,E);
        [c1,c2,ceq]=fconstraints(z1,t,h,b,p);
        fc1(j,i)=c1;
        fc2(j,i)=c2;
    end
end
```

```
27
28 figure(1)
29 contour(z1_vec,t_vec,f,[1e-6 5e-6 1e-5 2e-5 3e-5 4e-5 5e-5 6e-5 7e-5 ...
       8e-5],'ShowText','on')
30 hold on
31 contour(z1_vec,t_vec,fc2,[0.0 0.0],'r')                %c2
32 contour(z1_vec,t_vec,fc2,[0.025 0.025],'r--')
33 grid
34 xlabel('Offset 0\leqz_1\leq(h-p)  in [m]')
35 ylabel('Flank thickness 0\leqt\leq b in [m]')
36 title('Contour plot distortion w_z(x=l, z_1, t)')
37 axis([0 (h-p) 0 b])
38 legend('Value of objective function f(z_1,t)','Inequality ...
       constraint','Infeasible domain')
39 set(gca,'fontsize',11.5)
```

Matlab code B.4: fconstraints.m

```
1 function [c1,c2,ceq] = fconstraints(z1,t,h,b,p)
2 %normalised
3
4 c1=t/b-1;
5 c2=(t+p+z1)/h-1;
6 ceq=[];
7
8 end
```

B.3 Robustness optimization of a tee section for offset z_1 and flank thickness t

In this appendix, the MATLAB® code is shown for an optimization problem where offset z_1 and flank thickness t of the Tee Section in Fig. B.3 manufactured from a piece of rolled plate are optimized for *robustness*.

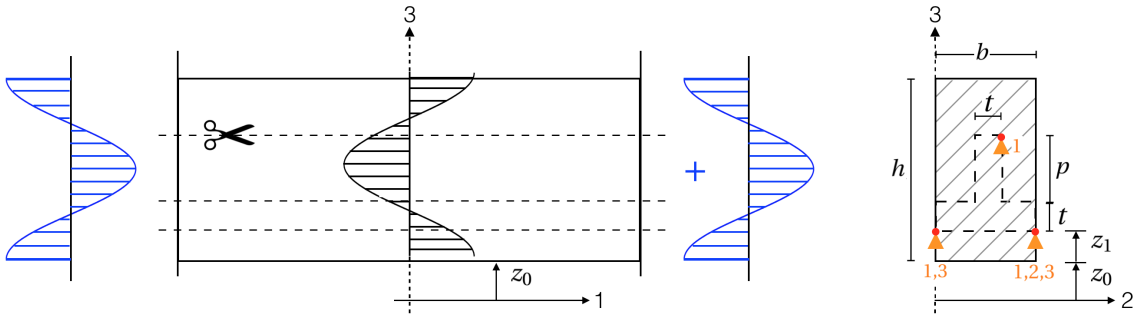


Figure B.3: This figure shows the same illustration as in Fig. 3.16. The offset z_1 and flank thickness t are optimized for robustness.

The m-file *fcontourplot.m*, shown in Fig. B.5, contains the MATLAB® code used for creating the contour plot. The inequality constraints are coded in a separate function m-file, *fconstraints.m*, shown in Fig. B.6.

Matlab code B.5: fcontourplot.m

```
clf; hold off; clear; clc; close all;

%constants
l=1.000;
b=0.040;
h=0.080;
p=0.030;
E=70.07*10^9;
z0=0.100;

%contour plot and surf plot
n=100;
z1_vec=linspace(0,h,n);
t_vec=linspace(0,b,n);

%stochastics
c_mean=20*10^6;
c_std=5*10^6;

for i=1:1:length(z1_vec)
    for j=1:1:length(t_vec)
        z1=z1_vec(i);
        t=t_vec(j);
        wz_specific=objective(l,b,h,p,t,z0,z1,1,E);
        wz_mean=c_mean*wz_specific;
        wz_std(j,i)=c_std*wz_specific;
```

```
        wz_std_abs(j,i)=sqrt((wz_std(j,i))^2);
        f(j,i)=(wz_mean)^2;
        [c1,c2,c3] = fconstraints(f(j,i),z1,t,h,b,p);
        fc1(j,i)=c1;
        fc2(j,i)=c2;
        fc3(j,i)=c3;
    end
end

figure(1)
contour(z1_vec,t_vec,wz_std_abs,[0.0001 0.0005 0.001 0.0015 0.002 ...
    0.0025 0.003],'ShowText','on')
grid
hold on
contour(z1_vec,t_vec,fc2,[0.0 0.0],'r')
contour(z1_vec,t_vec,fc2,[0.025 0.025],'r--')
contour(z1_vec,t_vec,fc3,[0.0 0.0],'m')
contour(z1_vec,t_vec,fc3,[0.15 0.15],'m--')
xlabel('Offset 0\leqz_1\leq(h-p)  in [m]')
ylabel('Flank thickness 0\leqt\leq b in [m]')
title('Contour plot robustness s_{wz}(z_1, t)')
axis([0 (h-p) 0 b])
legend('Value of objective function f(z_1,t)','Inequality constraint ...
    g_2','Infeasible domain g_3','Inequality constraint ...
    g_3','Infeasible domain g_3')
set(gca,'fontsize',11.5)
```

Matlab code B.6: fconstraints.m

```
function [c1,c2,c3] = fconstraints(fobj,z1,t,h,b,p)
%normalised

c1=t/b-1;
c2=(t+p+z1)/h-1;
c3=fobj/(3.6e-5)-1;

end
```

Appendix C

Finite element simulations concerning an elementary beam with rectangular cross section

C.1 Deterministic static analysis

This appendix concerns a Finite Element Analysis for an elementary beam that is illustrated in Fig. 4.1 with a mesh that was shown in Fig. 4.2. The stress field is positioned at offset $z_1 = 50\,\text{mm}$ and is not rotated with respect to the x-, y- or z-axis ($\alpha = 0\,\text{rad}, \beta = 0\,\text{rad}, \gamma = 0\,\text{rad}$). The two-dimensional stress field is assumed to be *deterministic* in nature and the stress coefficients $c^{\dagger}_{11,i}$ and $c^{\dagger}_{22,i}$ for $i = 1 \cdots 6$ are chosen to be equal to the mean values that were defined in chapter 2.2.

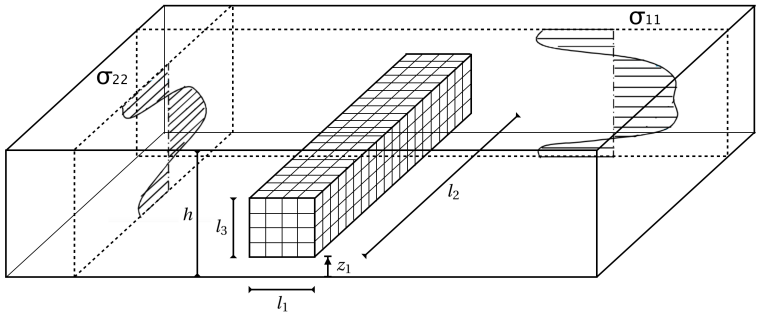


Figure C.1: This figure is identical to the figure shown in Fig. 4.1. The rectangular beam is positioned at offset $z_1 = 50\,\text{mm}$ having a two-dimensional stress field such as the one that is mathematically derived in chapter 2.2. The mesh consist of three-dimensional hexahedron elements. The beam has dimensions $l_1 = 50\,\text{mm}$, $l_2 = 1000\,\text{mm}$ and $l_3 = 50\,\text{mm}$. The thickness of stock material is $h = 300\,\text{mm}$. The beam has a Young's Modulus of $E = 70.07\,\text{GPa}$.

The static analyses are executed in Finite Element software Abaqus©. The input for Finite Element Analyses in Abaqus© are written in so-called *input files* (.inp). In these input files elements, nodes, and element sets are created. Furthermore, material properties are defined and boundary conditions are formulated. The input file *sim.inp*, shown in Fig. C.1, contains the main structure for the Finite Element Analysis in which the separate input files for nodes definition, elements definition and element set definition are summoned besides

the separate input files for boundary condition definition, material definition and step definition.

Abaqus Input file code C.1: sim.inp

```
*Heading
*Initial conditions, type=stress, user
*Part, name=Part-1
*End Part
*Assembly, name=Assembly
*Instance, name=Part-1-1, part=Part-1
         15.,        13.75,           0.
**
*Include, input=sim_nds.inp
**
*Include, input=sim_els.inp
**
*Include, input=sim_sts.inp
**
*Include, input=sim_bcs.inp
**
*Include, input=sim_simp.inp
**
*Include, input=sim_stp.inp
**
```

The input file *simnds.inp*, shown in Fig. C.2 contains the coordinate definition of the nodes in the mesh. Only a part is shown, since the mesh contains 2075 nodes.

Abaqus Input file code C.2: simnds.inp

```
*Node
      1,         -15.,       -13.75,        1000.
      2,         -15.,        -1.25,        1000.
      3,         -15.,        11.25,        1000.
      4,         -15.,        23.75,        1000.
      5,         -15.,        36.25,        1000.
      6,         -15.,       -13.75,   987.804871
      7,         -15.,        -1.25,   987.804871
      8,         -15.,        11.25,   987.804871
      9,         -15.,        23.75,   987.804871
     10,         -15.,        36.25,   987.804871
     11,         -15.,       -13.75,   975.609741
     12,         -15.,        -1.25,   975.609741
     13,         -15.,        11.25,   975.609741
     14,         -15.,        23.75,   975.609741
     15,         -15.,        36.25,   975.609741
     16,         -15.,       -13.75,   963.414612
  ..........................................
  a part has been left out
  ..........................................
   2072,          35.,        -1.25,           0.
   2073,          35.,        11.25,           0.
   2074,          35.,        23.75,           0.
```

```
24     2075,          35.,        36.25,           0.
```

The input file *simels.inp*, shown in Fig. C.3 contains the definition of the elements in the mesh based upon the nodes which were defined in the input file *simnds.inp*. Only a part is shown, since the mesh contains 1312 elements.

Abaqus Input file code C.3: simels.inp

```
1  *Element, type=C3D8R
2     1,  416,  417,  422,  421,    1,    2,    7,    6
3     2,  417,  418,  423,  422,    2,    3,    8,    7
4     3,  418,  419,  424,  423,    3,    4,    9,    8
5     4,  419,  420,  425,  424,    4,    5,   10,    9
6     5,  421,  422,  427,  426,    6,    7,   12,   11
7     6,  422,  423,  428,  427,    7,    8,   13,   12
8     7,  423,  424,  429,  428,    8,    9,   14,   13
9     8,  424,  425,  430,  429,    9,   10,   15,   14
10    9,  426,  427,  432,  431,   11,   12,   17,   16
11   10,  427,  428,  433,  432,   12,   13,   18,   17
12   11,  428,  429,  434,  433,   13,   14,   19,   18
13   12,  429,  430,  435,  434,   14,   15,   20,   19
14   13,  431,  432,  437,  436,   16,   17,   22,   21
15   14,  432,  433,  438,  437,   17,   18,   23,   22
16   15,  433,  434,  439,  438,   18,   19,   24,   23
17   16,  434,  435,  440,  439,   19,   20,   25,   24
18   ..................................................
19   a part has been left out
20   ..................................................
21 1309, 2066, 2067, 2072, 2071, 1651, 1652, 1657, 1656
22 1310, 2067, 2068, 2073, 2072, 1652, 1653, 1658, 1657
23 1311, 2068, 2069, 2074, 2073, 1653, 1654, 1659, 1658
24 1312, 2069, 2070, 2075, 2074, 1654, 1655, 1660, 1659
```

The input file *simsts.inp*, shown in Fig. C.4 contains the definition of element- and node sets used for the boundary conditions (*Set-A123*, *Set-B12*, *Set-C2*) and the centre node of the mesh at which displacement u_3 is measured (*Set-NodeDist*).

Abaqus Input file code C.4: simsts.inp

```
1  *Nset, nset=Set-all, generate
2      1,  2075,     1
3  *Elset, elset=Set-all, generate
4      1,  1312,     1
5  *Solid Section, elset=Set-all, material=Material-1
6  ,
7  *End Instance
8  *Nset, nset=Set-A123, instance=Part-1-1
9   413,
10 *Nset, nset=Set-B12, instance=Part-1-1
11  3,
12 *Nset, nset=Set-C2, instance=Part-1-1
13  2073,
14 *Nset, nset=Set-NodeDist, instance=Part-1-1
15  1038,
```

```
16 *End Assembly
```

The input file *simsimp.inp*, shown in Fig. C.5 contains the definition of the material properties of the mesh.

Abaqus Input file code C.5: simsimp.inp

```
1 *Material, name=Material-1
2 *Elastic
3 70070., 0.33
```

The input file *simbcs.inp*, shown in Fig. C.6 contains the definition of the iso-static boundary conditions acting on the mesh, preventing rigid body modes.

Abaqus Input file code C.6: simbcs.inp

```
1 *Boundary
2 Set-A123, 1, 1
3 Set-A123, 2, 2
4 Set-A123, 3, 3
5 *Boundary
6 Set-B12, 1, 1
7 Set-B12, 2, 2
8 *Boundary
9 Set-C2, 2, 2
```

The input file *simstp.inp*, shown in Fig. C.7 contains the *step* definition of the Finite Element Analysis. The step definition determines, amongst other things, that the Finite Element Analysis concerns a *static* analysis. Furthermore, a *node output* is created in which displacement u_3 is stored, which is used in this example as a measure for distortion of the beam.

Abaqus Input file code C.7: simstp.inp

```
1 *Step, name=Step-1, nlgeom=NO
2 *Static
3 1., 1., 1e-05, 1.
4 *Restart, write, frequency=0
5 *Output, field, variable=PRESELECT
6 *Output, history, variable=PRESELECT
7 *Output, history, frequency=99999
8 *Node Output, nset=Set-NodeDist
9 U2,
10 *End Step
```

The Abaqus© static analysis of which the details are defined in the input files, is launched from the *Cluster* with the *.sub* file *submit.sub* which is shown in Fig. C.8.

Cluster job code C.8: submit.sub

```
1 #!/bin/bash
2 #PBS -l nodes=1:ppn=1
```

```
#PBS -N yanj
#PBS -M N
#PBS -e hpc06.err
#PBS -o hpc06.log

#Print time and date
date
echo ------------------------------------------------------
echo -n 'Job is running on nodes '; cat $PBS_NODEFILE
echo ------------------------------------------------------
echo PBS: qsub is running on $PBS_O_HOST
echo PBS: originating queue is $PBS_O_QUEUE
echo PBS: executing queue is $PBS_QUEUE
echo PBS: working directory is $PBS_O_WORKDIR
echo PBS: execution mode is $PBS_ENVIRONMENT
echo PBS: job identifier is $PBS_JOBID
echo PBS: job name is $PBS_JOBNAME
echo PBS: node file is $PBS_NODEFILE
echo PBS: current home directory is $PBS_O_HOME
echo PBS: PATH = $PBS_O_PATH
echo ------------------------------------------------------

#Go to working directory
cd ${PBS_O_WORKDIR}

#Clean up:
rm *.so
rm *.msg
rm *.dat
rm *.odb
rm *.log
rm *.err

#Make scratch directory
DIR=${PWD}
TMP=${DIR}

#Write directory name and scratch path
echo -n $TMP > scr.dir

TMP=${PWD##*/}

#Total number of procs
#cat ${PBS_NODEFILE} | wc -l | tr -d " \t\n\r" > m_c.int

#Get num procs per node
#sort ${PBS_NODEFILE} | uniq -c | tr -d " \t\n\r" > node.file
#cat node.file | sed 's/n.*//' > n_c.int

n_c=$( cat n_c.int )
tmp=$( cat scr.dir )

module load intel/2013sp1
module load mpi/openmpi-1.8.8-intel
module load abaqus
```

```
export LM_LICENSE_FILE=27000@flexserv1.tudelft.nl

ulimit -s unlimited

python StaticAnalysis.py

#Clean up:
rm *.sta
rm *.prt
rm *.sim
rm *.dir
rm *.com
rm *.pyc
rm *.so
rm U3.dat
rm .modules


echo -n "END: "
date
echo "-----------------------------------------"
```

Within the *submit.sub* file, the python script *StaticAnalysis.py* is launched, which is shown in Fig. C.9. The python script *StaticAnalysis.py* writes the offzet z_1 to a seperate *.dbl* file and launches *funcStaticAnalysis.py*, in which the static analysis is executed.

Python code C.9: StaticAnalysis.py

```
from __future__ import division              #decimal division
#import matplotlib.pyplot as plt
from inputvalues import L,h
import funcStaticAnalysis
import timeit
import os
import shutil

##------------ 1: Create Plot directory ------------##
#delete old 'Plots' directory
Plotsdir='Plots'
if os.path.exists(Plotsdir):
    shutil.rmtree(Plotsdir)

#create new 'Plots' directory
os.makedirs(Plotsdir)

#total DSE run-time count
start_time_totalDSE = timeit.default_timer()


##------------ 2: single simulation  ------------##
z1=50

#create z1.dbl
fo=open('z1.dbl','w+')       #creates it for us if not already created
fo.write( '%e' % z1 )
```

```
28 fo.close()
29
30 U3_centrenode=funcStaticAnalysis.fStaticAnalysis(z1)
31
32 #end timer StaticAnalyses
33 elapsed_totalScript = timeit.default_timer() - start_time_totalDSE
34 print("--- computation time for total Script run is %s seconds ---" ...
       % elapsed_totalScript)
```

The script *funcStaticAnalysis.py*, which is shown in Fig. C.10, executes the Abaqus® static analysis and collects the results.

Python code C.10: funcStaticAnalysis.py

```
1  from __future__ import division              #decimal division
2  import math
3  import os
4  import random
5  import timeit
6  #import matplotlib.pyplot as plt
7  import shutil
8
9  def fStaticAnalysis(z1):
10
11     ##------------ 1: Run sim.inp, extract displacement for ...
          ------------##
12     #start timer Static Analyses
13     start_time_StaticAnalyses = timeit.default_timer()
14
15     os.system('abaqus double job=sim user=StressMapping cpus=1 ...
          scratch=/home/ycejanssens memory=32gb')
16     os.system("abaqus python ExtractU3fromodb.py")
17
18     with open('U3.dat') as file:
19         for line in file:
20             U3_centrenode=float(line)
21
22
23     ##-print mode displacements-##
24     print("***--------- RESULTS for z1%e ---------***" %z1)
25     print( 'U3_centrenode at z1=%e [mm] is %e [mm]' % ...
          (z1,U3_centrenode) )
26
27     ##-end timer StaticAnalyses-##
28     elapsed_StaticAnalyses = timeit.default_timer() - ...
          start_time_StaticAnalyses
29     print("--- computation time for Static Analyses is %s seconds ...
          ---" % elapsed_StaticAnalyses)
30
31     return U3_centrenode
```

The static analysis that is launched in *funcStaticAnalysis.py* summons a *SIGINI* subroutine *StressMapping.f* which is written in Fortran®. The SIGINI subroutine facilitates the assignment of the stress tensors to each element in the mesh based on the orientation of the stress

field and the coordinates of the elements. The methodology used for defining the stress tensors based on the element's coordinates and the orientation of the stress field was elaborated upon in chapter 4.2. The SIGINI subrouting written in *StressMapping.f* is the same as the one that is used in the case study in appendix **??** which is shown in Fig. **??**.

The script *ExtractU3fromodb.py*, which is shown in Fig. C.11, extracts the displacement of the centre node (node 1038)in z-direction from the *.odb* file *sim.odb* which contains the results of the static analysis.

Python code C.11: ExtractU3fromodb.py

```
1  if __name__ == "__main__":
2
3      from odbAccess import openOdb
4      odb=openOdb('sim.odb')
5
6      dataU3 = odb.steps['Step-1'].historyRegions['Node ...
           PART-1-1.1038'].historyOutputs['U3'].data[1]
7      dataU3str=str(dataU3)
8      len_firstpart=len('(1.0, ')
9      U3=float(dataU3str[len_firstpart:-1])
10
11     fo=open('U3.dat','w+')        #creates it for us if not already ...
           created
12     fo.write( '%e' % U3 )
13     fo.close()
```

Fig. C.2 shows the undeformed beam before the static analysis (increment 0) and the deformed beam after the static analysis (increment 1).

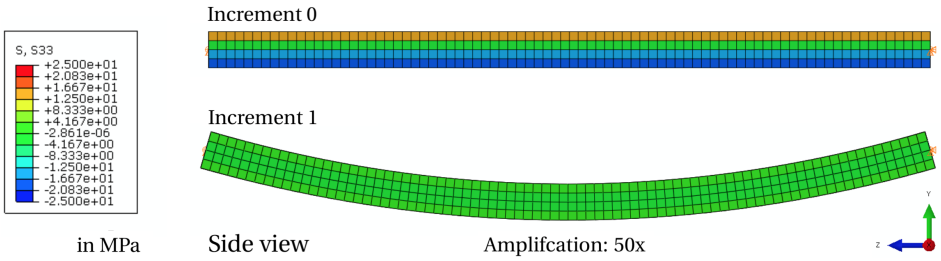


Figure C.2: This figure shows the mesh of the elementary beam before the static analysis (increment 0) and after the static analysis (increment 1). The mapping of the stress field to the elements is visible at increment 0. The internal stress is fully relaxed at increment 0.

The results of the static analysis are listed in Tab. C.1.

Table C.1: This table provides an overview of the results of the static analysis that are relevant. The static analysis concerns an elementary beam subjected to a *deterministic* stress field having an offset z_1 with respect to the mesh, such as is illustrated in Fig. C.1.

Outcome of deterministic static analysis	Symbol	Value
Centre node displacement	u_3	$-1.442\,096\,\mathrm{mm}$
Number of static analyses required	m	1
Computation time	t	$\approx 13\,\mathrm{sec}$

C.2 State-of-the-art method

In this appendix, the state-of-the-art method for evaluating distortion robustness is applied to the elementary beam positioned in rolled plate stock material as was illustrated in Fig. 4.1 with a mesh such as was shown in Fig. 4.2. The stress field is positioned at offset $z_1 = 50\,\text{mm}$ with respect to the bottom face of the rolled plate and is not rotated with respect to the x-, y- or z-axis ($\alpha = 0\,\text{rad}, \beta = 0\,\text{rad}, \gamma = 0\,\text{rad}$). The two-dimensional stress field is *stochastic* in nature with the specifics that were determined in chapter 2.1.

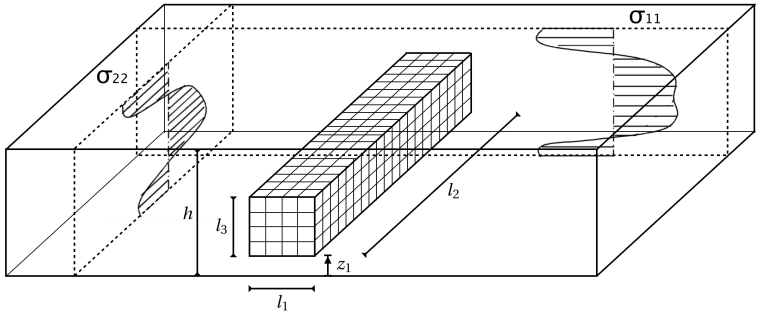


Figure C.3: This figure is identical to the figure shown in Fig. 4.1. The rectangular beam is positioned at offset $z_1 = 50\,\text{mm}$ having a two-dimensional stress field such as the one that is mathematically derived in chapter 2.2. The mesh consist of three-dimensional hexahedron elements. The beam has dimensions $l_1 = 50\,\text{mm}$, $l_2 = 1000\,\text{mm}$ and $l_3 = 50\,\text{mm}$. The thickness of stock material is $h = 300\,\text{mm}$. The beam has a Young's Modulus of $E = 70.07\,\text{GPa}$.

The input files used in the state-of-the-art method are identical to the input files that were shown in Fig. C.1 - C.7. The state-of-the-art method concerning the Monte Carlo simulations is launched from the Cluster using the same *submit.sub* file as the one that was shown in Fig. C.8. This time, however, the python script *StateoftheArtMethod.py* is launched, which is shown in Fig. C.12.

Python code C.12: StateoftheArtMethod.py

```
1  from __future__ import division              #decimal division
2  #import matplotlib.pyplot as plt
3  from inputvalues import L,h
4  import funcStateoftheArtMethod
5  import timeit
6  import math
7  import random
8  import os
9  import shutil
10
11
12 #total DSE run-time count
13 start_time_totalDSE = timeit.default_timer()
14
15 ##------------ 0: constants ------------##
16 s11_1_mean=-8.1936
17 s11_2_mean=2.4317
18 s11_3_mean=2.9584
```

```
s11_4_mean=1.3215
s11_5_mean=0.9406
s11_6_mean=1.0299

s22_1_mean=-27.0343
s22_2_mean=6.1362
s22_3_mean=7.8890
s22_4_mean=-1.4510
s22_5_mean=1.0052
s22_6_mean=0.8000

s11_1_std=2.9313
s11_2_std=1.2281
s11_3_std=1.2097
s11_4_std=1.0429
s11_5_std=0.9294
s11_6_std=0.7252

s22_1_std=4.0475
s22_2_std=3.8186
s22_3_std=1.9898
s22_4_std=2.4577
s22_5_std=2.0155
s22_6_std=0.9293

m_MonteCarlo=10000

# make sure s_11_log.dbl, s_22_log.dbl, ...
    U3_MonteCarloIteration_log.dbl, U3_mean_log.dbl, U3_std_log.dbl ...
    are empty

logfile_s11='s_11_log.dbl'
if os.path.isfile(logfile_s11):
    os.remove(logfile_s11)

open('s_11_log.dbl', 'a').close()

logfile_s22='s_22_log.dbl'
if os.path.isfile(logfile_s22):
    os.remove(logfile_s22)

open('s_22_log.dbl', 'a').close()

logfile_U3_MonteCarloIteration='U3_MonteCarloIteration_log.dbl'
if os.path.isfile(logfile_U3_MonteCarloIteration):
    os.remove(logfile_U3_MonteCarloIteration)

open('U3_MonteCarloIteration_log.dbl', 'a').close()

logfile_U3_mean='U3_mean_log.dbl'
if os.path.isfile(logfile_U3_mean):
    os.remove(logfile_U3_mean)

open('U3_mean_log.dbl', 'a').close()

```

```
72 logfile_U3_std='U3_std_log.dbl'
73 if os.path.isfile(logfile_U3_std):
74     os.remove(logfile_U3_std)
75
76 open('U3_std_log.dbl', 'a').close()
77
78
79
80 ##------------ 1: Monte Carlo  ------------##
81
82 z1=50
83
84 #create z1.dbl
85 fo=open('z1.dbl','w+')       #creates it for us if not already created
86 fo.write( '%e' % z1 )
87 fo.close()
88
89 U3_MonteCarloIteration=[0]*m_MonteCarlo
90
91 start_time_StaticAnalyses = timeit.default_timer()
92 for i in range(m_MonteCarlo):
93     s11_1_MonteCarlo=random.normalvariate(s11_1_mean,s11_1_std)
94     s11_2_MonteCarlo=random.normalvariate(s11_2_mean,s11_2_std)
95     s11_3_MonteCarlo=random.normalvariate(s11_3_mean,s11_3_std)
96     s11_4_MonteCarlo=random.normalvariate(s11_4_mean,s11_4_std)
97     s11_5_MonteCarlo=random.normalvariate(s11_5_mean,s11_5_std)
98     s11_6_MonteCarlo=random.normalvariate(s11_6_mean,s11_6_std)
99
100    s22_1_MonteCarlo=random.normalvariate(s22_1_mean,s22_1_std)
101    s22_2_MonteCarlo=random.normalvariate(s22_2_mean,s22_2_std)
102    s22_3_MonteCarlo=random.normalvariate(s22_3_mean,s22_3_std)
103    s22_4_MonteCarlo=random.normalvariate(s22_4_mean,s22_4_std)
104    s22_5_MonteCarlo=random.normalvariate(s22_5_mean,s22_5_std)
105    s22_6_MonteCarlo=random.normalvariate(s22_6_mean,s22_6_std)
106
107
108    #create s_11.dbl and s_22.dbl which will be used in analyses
109    fo=open('s_11.dbl','w+')       #creates it for us if not already ...
           created
110    fo.write( '%e\n%e\n%e\n%e\n%e\n%e' % ...
111              (s11_1_MonteCarlo,s11_2_MonteCarlo,s11_3_MonteCarlo, ...
112               s11_4_MonteCarlo,s11_5_MonteCarlo,s11_6_MonteCarlo))
113    fo.close()
114
115    fo=open('s_22.dbl','w+')       #creates it for us if not already ...
           created
116    fo.write( '%e\n%e\n%e\n%e\n%e\n%e' % ...
117              (s22_1_MonteCarlo,s22_2_MonteCarlo,s22_3_MonteCarlo, ...
118               s22_4_MonteCarlo,s22_5_MonteCarlo,s22_6_MonteCarlo))
119    fo.close()
120
121    #calculate U3_MonteCarloIteration
122    U3_MonteCarloIteration[i] ...
123    =funcStateoftheArtMethod.fStateoftheArtMethod(z1)
124
```

```
125     #writes s_11, s_22 and U3_MonteCarloSimulations to logfiles
126     fo=open('s_11_log.dbl','a')       #appends
127     fo.write( 's11 at Monte Carlo iteration %e:\n' % (i))
128     fo.close()
129
130     fo=open('s_11_log.dbl','a')       #appends
131     fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n' % ...
132               (s11_1_MonteCarlo,s11_2_MonteCarlo,s11_3_MonteCarlo, ...
133                s11_4_MonteCarlo,s11_5_MonteCarlo,s11_6_MonteCarlo))
134     fo.close()
135
136     fo=open('s_22_log.dbl','a')       #appends
137     fo.write( 's22 at Monte Carlo iteration %e:\n' % (i))
138     fo.close()
139
140     fo=open('s_22_log.dbl','a')       #appends
141     fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n' % ...
142               (s22_1_MonteCarlo,s22_2_MonteCarlo,s22_3_MonteCarlo, ...
143                s22_4_MonteCarlo,s22_5_MonteCarlo,s22_6_MonteCarlo))
144     fo.close()
145
146     fo=open('U3_MonteCarloIteration_log.dbl','a')       #appends
147     fo.write( 'U3_MonteCarloIteration at Monte Carlo iteration ...
            %e:\n' % (i))
148     fo.close()
149
150     fo=open('U3_MonteCarloIteration_log.dbl','a')       #appends
151     fo.write( '%e\n' % (U3_MonteCarloIteration[i]))
152     fo.close()
153
154     print("***--------- RESULTS for iteration_%e ---------***" %i)
155     print( 'U3_MonteCarloIteration at iteration_%e [mm] is %e [mm]' ...
            % (i,U3_MonteCarloIteration[i]) )
156
157
158 elapsed_StaticAnalyses = timeit.default_timer() - ...
        start_time_StaticAnalyses
159 print("--- computation time for State-of-the-art method is %s ...
        seconds ---" % elapsed_StaticAnalyses)
160
161
162 ##------------ 2: determine displacement distribution  ------------##
163
164 #mean
165 print( 'number of samples (m_MonteCarlo) is %e' % m_MonteCarlo )
166
167 U3_mean=sum(U3_MonteCarloIteration)/m_MonteCarlo
168 print( 'U3_mean is %e in [mm]' % U3_mean )
169
170 fo=open('U3_mean_log.dbl','a')       #appends
171 fo.write( 'U3_mean for %e Monte Carlo iterations is:\n' % ...
        (m_MonteCarlo))
172 fo.close()
173
174 fo=open('U3_mean_log.dbl','a')       #appends
```

```
175 fo.write( '%e\n' % (U3_mean))
176 fo.close()
177
178 #standard deviation
179 sumsq=0
180 for j in range(m_MonteCarlo):
181     sumsq=sumsq+(U3_MonteCarloIteration[j]-U3_mean)**2
182
183 U3_std=math.sqrt(sumsq/(m_MonteCarlo-1))
184 print( 'U3_std is %e in [mm]' % U3_std )
185
186 fo=open('U3_std_log.dbl','a')       #appends
187 fo.write( 'U3_std for %e Monte Carlo iterations is:\n' % (m_MonteCarlo))
188 fo.close()
189
190 fo=open('U3_std_log.dbl','a')       #appends
191 fo.write( '%e\n' % (U3_std))
192 fo.close()
193
194
195
196 #end timer StaticAnalyses
197 elapsed_totalScript = timeit.default_timer() - start_time_totalDSE
198 print("--- computation time for total Script run is %s seconds ---" ...
      % elapsed_totalScript)
```

Within python script *StateoftheArtMethod.py*, python script *funcStateoftheArtMethod.py* shown in Fig. C.13 is launched which launches the static analysis for each Monte Carlo simulation and subsequently extracts the u_3 displacement of the centre node after the static analysis. The python script *ExtractU3fromodb.py*, which was shown in Fig. C.11, extracts the displacement of the centre node (node 1038)in z-direction from the *.odb* file after each static analysis.

Python code C.13: funcStateoftheArtMethod.py

```
1  from __future__ import division             #decimal division
2  import math
3  import os
4  import random
5  import timeit
6  #import matplotlib.pyplot as plt
7  import shutil
8
9  def fStateoftheArtMethod(z1):
10
11     ##------------ 1: Run sim.inp, extract displacement ------------##
12
13     os.system('abaqus double job=sim user=StressMapping cpus=1 ...
          scratch=/home/ycejanssens memory=32gb')
14     os.system("abaqus python ExtractU2fromodb.py")
15
16     with open('U3.dat') as file:
17         for line in file:
18             U3_MonteCarloIteration=float(line)
```

```
19
20      return U3_MonteCarloIteration
```

The static analyses that are launched in *funcStateoftheArtMethod.py* summon the *SIGINI* subroutine *StressMapping.f* which is shown in appendix **??** in Fig. **??**. The *SIGINI* subroutine facilitates the assignment of the stress tensors to each element in the mesh based on the orientation of the stress field and the coordinates of the elements.

At the end of the state-of-the-art method for robustness evaluation, a vector named $\boldsymbol{u_3}$ is obtained containing the displacements in z-direction (u_3) of the centre node (node 1038) for all 10000 Monte Carlo iterations. In Fig. C.4, on the left, a scatter plot is shown of all displacements obtained at each Monte Carlo iteration. In the same figure, on the right, the occurance of each displacement is depicted in a histogram.

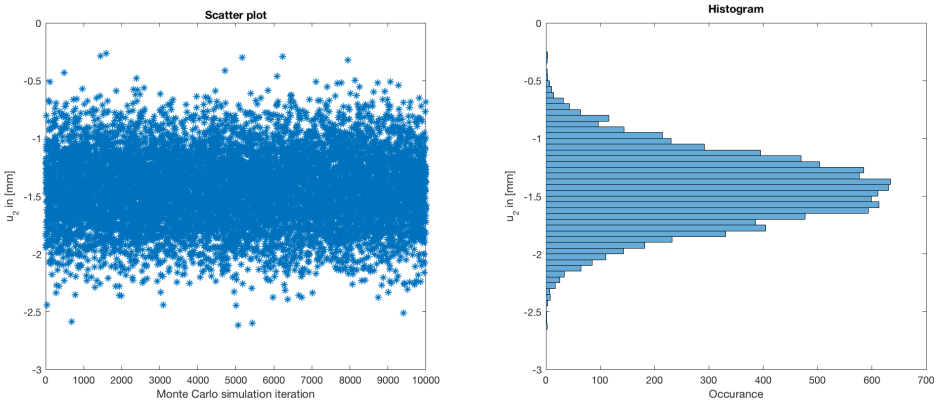


Figure C.4: This figure shows a scatter plot (left) of the centre node (node 1038) displacement u_3 per Monte Carlo iteration and a histogram (right) of the occurance of centre node (node 1038) displacement u_3.

The obtained distribution appears to resemble the shape of a normal distribution. The mean displacement $\overline{x}_{u_3}$ and standard deviation of the displacement s_{u_3} can be obtained from this data set as was elaborated upon in chapter 4.3. The outcome of the state-of-the-art method used for obtaining the distortion distribution of the elementary beam shown in fig. 4.1 is listed in Tab. C.2.

Table C.2: This table provides an overview of the outcome of the state-of-the-art method employed for evaluating distortion robustness of the elementary beam that was shown in 4.1 and that was subjected to a stochastic stress field as was specified in chapter 2.2.

Outcome of the state-of-the-art method	Symbol	Value
Mean value of centre node displacement	$\overline{x}_{u_3}$	$-1.445\,049\,\text{mm}$
Standard deviation of centre node displacement	s_{u_3}	$3.078\,326 \times 10^{-1}\,\text{mm}$
Number of static analyses required	m	10000
Computation time	t	$\approx 85\,157\,\text{sec}$

It becomes evident that even for an elementary mesh such as the one that is considered in this appendix, obtaining reliable robustness information using the state-of-the-art method

is computationally highly inefficient, as the computation time was equal to $t \approx 85\,157$ sec. Furthermore, the mean value of the centre node displacement, $\overline{x}_{u_3}$, seems to be approximately equal to the deterministic value for the centre node displacement which was obtained with the *deterministic* static analysis of which the results were listed in Tab. 4.1.

C.3 Improved method

In this appendix, the improved method for evaluating distortion robustness is applied to the elementary beam positioned in rolled plate stock material as was illustrated in Fig. 4.1 with a mesh such as was shown in Fig. 4.2. The stress field is positioned at offset $z_1 = 50\,\text{mm}$ with respect to the bottom face of the rolled plate and is not rotated with respect to the x-, y- or z-axis ($\alpha = 0\,\text{rad}, \beta = 0\,\text{rad}, \gamma = 0\,\text{rad}$). The two-dimensional stress field is *stochastic* in nature with the specifics that were determined in chapter 2.1.

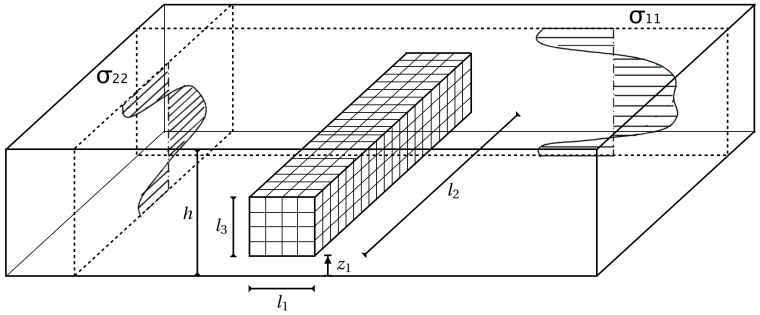


Figure C.5: This figure is identical to the figure shown in Fig. 4.1. The rectangular beam is positioned at offset $z_1 = 50\,\text{mm}$ having a two-dimensional stress field such as the one that is mathematically derived in chapter 2.2. The mesh consist of three-dimensional hexahedron elements. The beam has dimensions $l_1 = 50\,\text{mm}$, $l_2 = 1000\,\text{mm}$ and $l_3 = 50\,\text{mm}$. The thickness of stock material is $h = 300\,\text{mm}$. The beam has a Young's Modulus of $E = 70.07\,\text{GPa}$.

The input files used in the state-of-the-art method are identical to the input files that were shown in Fig. C.1 - C.7. The improved method for robustness evaluation is launched from the Cluster using the same *submit.sub* file as the one that was shown in Fig. C.8. This time, however, the python script *ImprovedMethod.py* is launched, which is shown in Fig. C.14.

Python code C.14: ImprovedMethod.py

```
1  from __future__ import division              #decimal division
2  #import matplotlib.pyplot as plt
3  from inputvalues import L,h
4  import funcImprovedMethod
5  import timeit
6  import os
7  import shutil
8
9
10 #total DSE run-time count
11 start_time_totalDSE = timeit.default_timer()
12
13
14 ##------------ 2: single simulation  ------------##
15 z1=50
16
17 #create z1.dbl
18 fo=open('z1.dbl','w+')        #creates it for us if not already created
19 fo.write( '%e' % z1 )
```

```
fo.close()

deterministic_U3,mean_U3,std_U3=funcImprovedMethod.fImprovedMethod(z1)

print('deterministic_U3 is %e' % (deterministic_U3))
print('mean_U3 is %e' % (mean_U3))
print('std_U3 is %e' % (std_U3)

#end timer StaticAnalyses
elapsed_totalScript = timeit.default_timer() - start_time_totalDSE
print("--- computation time for total Script run is %s seconds ---" ...
    % elapsed_totalScript)
```

Within the script *ImprovedMethod.py*, the script *funcImprovedMethod.py* is called, which is shown in Fig. C.15. In this script, the improved method for evaluating robustness, which is further elaborated in chapter 4.4, is executed. Twelve static analyses are performed in order to obtain the individual mode displacement fields $\boldsymbol{U}_{11,i}$ and $\boldsymbol{U}_{22,i}$ for $i = 1 \cdots 6$. Subsequently, the final displacement field $\boldsymbol{u}$ including stochastic information is obtained via the twelve displacement fields.

Python code C.15: funcImprovedMethod.py

```
from __future__ import division              #decimal division
import math
import os
import random
import timeit
#import matplotlib.pyplot as plt
import shutil
import numpy

def fImprovedMethod(z1):


    ##------------ 0: stress coefficients ------------##
    c_11_1=-8.1936
    c_11_2=2.4317
    c_11_3=2.9584
    c_11_4=1.3215
    c_11_5=0.9406
    c_11_6=1.0299

    c_22_1=-27.0343
    c_22_2=6.1362
    c_22_3=7.8890
    c_22_4=-1.4510
    c_22_5=1.0052
    c_22_6=0.8000


    ##------------ 1: Run sim.inp, extract displacement ------------##
    #start timer Static Analyses
    start_time_StaticAnalyses = timeit.default_timer()

```

```
    ### Displacement of individual mode U11,1
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode1 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode1=float(line)

    ### Displacement of individual mode U11,2
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode2 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode2=float(line)

    ### Displacement of individual mode U11,3
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode3 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode3=float(line)

    ### Displacement of individual mode U11,4
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode4 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode4=float(line)

    ### Displacement of individual mode U11,5
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode5 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode5=float(line)

    ### Displacement of individual mode U11,6
    os.system('abaqus double job=sim user=StressMapping_Phi11_mode6 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_11_mode6=float(line)


```

```
    ### Displacement of individual mode U22,1
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode1 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode1=float(line)

    ### Displacement of individual mode U22,2
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode2 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode2=float(line)

    ### Displacement of individual mode U22,3
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode3 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode3=float(line)

    ### Displacement of individual mode U22,4
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode4 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode4=float(line)

    ### Displacement of individual mode U22,5
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode5 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode5=float(line)

    ### Displacement of individual mode U22,6
    os.system('abaqus double job=sim user=StressMapping_Phi22_mode6 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU3fromodb.py")

    with open('U3.dat') as file:
        for line in file:
            U3_22_mode6=float(line)
```

```
131     ##-print mode displacements-##
132     print("***--------- RESULTS for ystar_%e ---------***" %z1)
133     print( 'U3_11_mode1 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode1) )
134     print( 'U3_11_mode2 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode2) )
135     print( 'U3_11_mode3 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode3) )
136     print( 'U3_11_mode4 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode4) )
137     print( 'U3_11_mode5 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode5) )
138     print( 'U3_11_mode6 at z1=%e [mm] is %e [mm]' % (z1,U3_11_mode6) )
139     print( 'U3_22_mode1 at z1=%e [mm] is %e [mm]' % (z1,U3_22_mode1) )
140     print( 'U3_22_mode2 at z1=%e [mm] is %e [mm]' % ...
            (z1,U3_22_mode2)) )
141     print( 'U3_22_mode3 at z1=%e [mm] is %e [mm]' % ...
            (z1,U3_22_mode3)) )
142     print( 'U3_22_mode4 at z1=%e [mm] is %e [mm]' % ...
            (z1,U3_22_mode4)) )
143     print( 'U3_22_mode5 at z1=%e [mm] is %e [mm]' % ...
            (z1,U3_22_mode5)) )
144     print( 'U3_22_mode6 at z1=%e [mm] is %e [mm]' % ...
            (z1,U3_22_mode6)) )
145
146     ##-Total deterministic displacement U3-##
147     U3_deterministic=c_11_1*U3_11_mode1+c_11_2*U3_11_mode1+ \
148                                     c_11_3*U3_11_mode1+c_11_4*U3_11_mode1+ ...
                                        \
149                                     c_11_5*U3_11_mode1+c_11_6*U3_11_mode1+ ...
                                        \
150                                     c_22_1*U3_11_mode1+c_22_2*U3_11_mode1+ ...
                                        \
151                                     c_22_3*U3_11_mode1+c_22_4*U3_11_mode1+ ...
                                        \
152                                     c_22_5*U3_11_mode1+c_22_6*U3_11_mode1
153     print( 'deterministic_U3 at z1=%e [mm] is %e [mm]' % ...
            (z1,deterministic_U3) )
154
155     ##-end timer StaticAnalyses-##
156     elapsed_StaticAnalyses = timeit.default_timer() - ...
            start_time_StaticAnalyses
157     print("--- computation time for Static Analyses is %s seconds ...
            ---" % elapsed_StaticAnalyses)
158
159
160     ##------------ 2: Analytical robustness evaluation  ------------##
161     #start timer Monte-Carlo
162     start_time_MonteCarlo = timeit.default_timer()
163
164     c11_1_std=2.9313
165     c11_2_std=1.2281
166     c11_3_std=1.2097
167     c11_4_std=1.0429
168     c11_5_std=0.9294
169     c11_6_std=0.7252
170
171     c22_1_std=4.0475
172     c22_2_std=3.8186
173     c22_3_std=1.9898
```

```
    c22_4_std=2.4577
    c22_5_std=2.0155
    c22_6_std=0.9293

    #theoretical mean derivation
    mean_U3=c_11_1*U3_11_mode1 + c_11_2*U3_11_mode2 + \
                          c_11_3*U3_11_mode3 + c_11_4*U3_11_mode4 + \
                          c_11_5*U3_11_mode5 + c_11_6*U3_11_mode6 + \
                          c_22_1*U3_22_mode1 + c_22_2*U3_22_mode2 + \
                          c_22_3*U3_22_mode3 + c_22_4*U3_22_mode4 + \
                          c_22_5*U3_22_mode5 + c_22_6*U3_22_mode6


    #theoretical var and std derivation
    var_U3=(c11_1_std*U3_11_mode1)**2 + (c11_2_std*U3_11_mode2)**2 + \
                               (c11_3_std*U3_11_mode3)**2 + ...
                                   (c11_4_std*U3_11_mode4)**2 + \
                               (c11_5_std*U3_11_mode5)**2 + ...
                                   (c11_6_std*U3_11_mode6)**2 + \
                               (c22_1_std*U3_22_mode1)**2 + ...
                                   (c22_2_std*U3_22_mode2)**2 + \
                               (c22_3_std*U3_22_mode3)**2 + ...
                                   (c22_4_std*U3_22_mode4)**2 + \
                               (c22_5_std*U3_22_mode5)**2 + ...
                                   (c22_6_std*U3_22_mode6)**2


    std_U3=math.sqrt(var_U3)


    elapsed_MonteCarlo = timeit.default_timer() - start_time_MonteCarlo
    print("--- computation time for State-of-the-Art method is %s ...
        seconds ---" % elapsed_MonteCarlo)


    return deterministic_U3, mean_U3, std_U3
```

At each static analysis for $\boldsymbol{U}_{11,i}$ or $\boldsymbol{U}_{22,i}$ that are performed in *funcImprovedMethod.py*, the corresponding stress mode matrices $\boldsymbol{\phi}_{11,i}$ and $\boldsymbol{\phi}_{22,i}$ form the input. There are thus *twelve* Fortran scripts needed for the assignment of stress tensors to the elements in the mesh for each of the twelve individual static analyses, which are shown in Fig. C.16-C.27.

Fortran code C.16: StressMappingS11Mode1.f

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--------------------------------------------------------------!
!       lop.eq.0 -> start of the analysis                      !
!       lop.eq.1 -> start of the current analysis increment    !
!       lop.eq.2 -> end of the current analysis increment      !
!       lop.eq.3 -> end of the analysis                        !
!       lop.eq.4 -> beginning of a restart analysis            !
!       lop.eq.5 -> start of the step (#kstep)                 !
```

```
!       lop.eq.6 -> end of the step (#kstep)                    !
!---------------------------------------------------------------!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
        subroutine uexternaldb(lop,lrestart,time,dtime,kstep,kinc)
!
        include 'aba_param.inc'
!
#include <SMAAspUserSubroutines.hdr>
!
!       General
!
        integer lop,io
        integer i,j,k
        logical exists
        dimension time(2)
        double precision dmp
!
!       Strings
!
        character(99) :: scr
        character(99) :: fln
!
!       Arrays
!
        DOUBLE PRECISION s_11(*)
        POINTER (s_11_ptr, s_11)
!
        DOUBLE PRECISION C_2(*)
        POINTER (C_2_ptr, C_2)
!
        DOUBLE PRECISION s_33(*)
        POINTER (s_33_ptr, s_33)
!
        DOUBLE PRECISION O_F(*)
        POINTER (O_F_ptr, O_F)
!
        DOUBLE PRECISION H_T(*)
        POINTER (H_T_ptr, H_T)
!
        DOUBLE PRECISION alpha_deg(*)
        POINTER (alpha_deg_ptr, alpha_deg)
!
        DOUBLE PRECISION beta_deg(*)
        POINTER (beta_deg_ptr, beta_deg)
!
        DOUBLE PRECISION gamma_deg(*)
        POINTER (gamma_deg_ptr, gamma_deg)
!
!       ALWAYS GET SCRATCH
!
        exists=.false.
        inquire(file='../scr.dir', exist=exists)
```

```
66          if(exists) then
67                  open(unit=666,file='../scr.dir')
68                  read(666,'(a)') scr
69                  close(666)
70          else
71                  print*, 'FILE ERROR; scr.dir not found'
72                  stop
73          endif
74  !
75  !       START OF THE ANALYSIS
76  !
77          if(lop.eq.0) then
78          !
79          print*, trim(scr)
80          !
81                  s_11_ptr = SMAFloatArrayCreate(11,6,0.d0)
82                  C_2_ptr = SMAFloatArrayCreate(12,6,0.d0)
83                  s_33_ptr = SMAFloatArrayCreate(13,6,0.d0)
84                  O_F_ptr = SMAFloatArrayCreate(14,3,0.d0)
85                  H_T_ptr = SMAFloatArrayCreate(15,1,0.d0)
86          alpha_deg_ptr = SMAFloatArrayCreate(16,1,0.d0)
87          beta_deg_ptr = SMAFloatArrayCreate(17,1,0.d0)
88          gamma_deg_ptr = SMAFloatArrayCreate(18,1,0.d0)
89          !
90          !       Read stress components, part height, and offset
91          !
92                  fln= trim(scr)//'/s_11.dbl'
93                  exists=.false.
94                  inquire(file=fln, exist=exists)
95                  if(exists) then
96                          open(unit=205,file=fln)
97                          i=1
98                          do
99                                  read(205,*,iostat=io) dmp
100                                 if(io/=0) exit
101                                 s_11(i) = dmp
102                                 i=i+1
103                         enddo
104                         close(205)
105                 else
106                         print*, 'FILE ERROR; s_11.dbl not found'
107                         stop
108                 endif
109         !
110                 fln= trim(scr)//'/C_2.dbl'
111                 exists=.false.
112                 inquire(file=fln, exist=exists)
113                 if(exists) then
114                         open(unit=205,file=fln)
115                         i=1
116                         do
117                                 read(205,*,iostat=io) dmp
118                                 if(io/=0) exit
119                                 C_2(i) = dmp
120                                 i=i+1
```

```
121                     enddo
122                     close(205)
123             else
124                     print*, 'FILE ERROR; C_2.dbl not found'
125                     stop
126             endif
127     !
128             fln= trim(scr)//'/s_33.dbl'
129             exists=.false.
130             inquire(file=fln, exist=exists)
131             if(exists) then
132                     open(unit=205,file=fln)
133                     i=1
134                     do
135                             read(205,*,iostat=io) dmp
136                             if(io/=0) exit
137                             s_33(i) = dmp
138                             i=i+1
139                     enddo
140                     close(205)
141             else
142                     print*, 'FILE ERROR; s_33.dbl not found'
143                     stop
144             endif
145     !
146             fln= trim(scr)//'/O_F.dbl'
147             exists=.false.
148             inquire(file=fln, exist=exists)
149             if(exists) then
150                     open(unit=205,file=fln)
151                     i=1
152                     do
153                             read(205,*,iostat=io) dmp
154                             if(io/=0) exit
155                             O_F(i) = dmp
156                             i=i+1
157                     enddo
158                     close(205)
159             else
160                     print*, 'FILE ERROR; OFF.dbl not found'
161                     stop
162             endif
163     !
164             fln= trim(scr)//'/H_T.dbl'
165             exists=.false.
166             inquire(file=fln, exist=exists)
167             if(exists) then
168                     open(unit=205,file=fln)
169                             read(205,*,iostat=io) dmp
170                             H_T(1) = dmp
171                     close(205)
172             else
173                     print*, 'FILE ERROR; H_T.dbl not found'
174                     stop
175             endif
```

```
176       !
177                   fln= trim(scr)//'/alpha_deg.dbl'
178                   exists=.false.
179                   inquire(file=fln, exist=exists)
180                   if(exists) then
181                           open(unit=205,file=fln)
182                                   read(205,*,iostat=io) dmp
183                                   alpha_deg(1) = dmp
184                           close(205)
185                   else
186                           print*, 'FILE ERROR; alpha_deg.dbl not found'
187                           stop
188                   endif
189       !
190                   fln= trim(scr)//'/beta_deg.dbl'
191                   exists=.false.
192                   inquire(file=fln, exist=exists)
193                   if(exists) then
194                           open(unit=205,file=fln)
195                                   read(205,*,iostat=io) dmp
196                                   beta_deg(1) = dmp
197                           close(205)
198                   else
199                           print*, 'FILE ERROR; beta_deg.dbl not found'
200                           stop
201                   endif
202       !
203                   fln= trim(scr)//'/gamma_deg.dbl'
204                   exists=.false.
205                   inquire(file=fln, exist=exists)
206                   if(exists) then
207                           open(unit=205,file=fln)
208                                   read(205,*,iostat=io) dmp
209                                   gamma_deg(1) = dmp
210                           close(205)
211                   else
212                           print*, 'FILE ERROR; gamma_deg.dbl not found'
213                           stop
214                   endif
215           !
216           else
217   !
218   !       IF NOT START OF THE ANALYSIS (access values)
219   !
220                   s_11_ptr = SMAFloatArrayAccess(11)
221                   C_2_ptr = SMAFloatArrayAccess(12)
222                   s_33_ptr = SMAFloatArrayAccess(13)
223                   O_F_ptr = SMAFloatArrayAccess(14)
224                   H_T_ptr = SMAFloatArrayAccess(15)
225           alpha_deg_ptr = SMAFloatArrayAccess(16)
226           beta_deg_ptr = SMAFloatArrayAccess(17)
227           gamma_deg_ptr = SMAFloatArrayAccess(18)
228   !
229           endif
230   !
```

```
231 !
232 !
233           end subroutine
234 !
235           SUBROUTINE SIGINI(SIGMA,COORDS,NTENS,NCRDS,NOEL,NPT,LAYER,
236       1 KSPT,LREBAR,NAMES)
237 C
238         INCLUDE 'ABA_PARAM.INC'
239 C
240 #include <SMAAspUserSubroutines.hdr>
241 !
242
243 !       !!!-------0: declarations-------!!!
244         DOUBLE PRECISION, DIMENSION(3,3) :: Rx
245         DOUBLE PRECISION, DIMENSION(3,3) :: Ry
246         DOUBLE PRECISION, DIMENSION(3,3) :: Rz
247         DOUBLE PRECISION, DIMENSION(3,3) :: Rx_backrotation
248         DOUBLE PRECISION, DIMENSION(3,3) :: Ry_backrotation
249         DOUBLE PRECISION, DIMENSION(3,3) :: Rz_backrotation
250         DOUBLE PRECISION, DIMENSION(3,3) :: R1
251         DOUBLE PRECISION, DIMENSION(3,3) :: R
252         DOUBLE PRECISION, DIMENSION(3,3) :: R1_backrotation
253         DOUBLE PRECISION, DIMENSION(3,3) :: R_backrotation
254         DOUBLE PRECISION, DIMENSION(3,3) :: R_T
255         DOUBLE PRECISION, DIMENSION(3,3) :: R_backrotation_T
256         DOUBLE PRECISION, DIMENSION(3,1) :: COORDS_wrtC_transformed
257         DOUBLE PRECISION, DIMENSION(3,1) :: COORDS_wrtC
258         DOUBLE PRECISION, DIMENSION(3,3) :: sigma_transformed
259         DOUBLE PRECISION, DIMENSION(3,3) :: sigma_originalconfiguration
260         DOUBLE PRECISION, DIMENSION(3,3) :: sigma_step1
261         DOUBLE PRECISION alpha
262         DOUBLE PRECISION beta
263         DOUBLE PRECISION gamma
264         DOUBLE PRECISION xbar
265         DOUBLE PRECISION ybar
266         DOUBLE PRECISION zbar
267
268         INTEGER i
269         DIMENSION SIGMA(NTENS),COORDS(NCRDS)
270         CHARACTER NAMES(2)*80
271 !
272         DOUBLE PRECISION PI
273 !
274         DOUBLE PRECISION s_11(*)
275         POINTER (s_11_ptr, s_11)
276 !
277         DOUBLE PRECISION C_2(*)
278         POINTER (C_2_ptr, C_2)
279 !
280         DOUBLE PRECISION s_33(*)
281         POINTER (s_33_ptr, s_33)
282 !
283         DOUBLE PRECISION O_F(*)
284         POINTER (O_F_ptr, O_F)
285 !
```

```
286         DOUBLE PRECISION H_T(*)
287         POINTER (H_T_ptr, H_T)
288 !
289         DOUBLE PRECISION alpha_deg(*)
290         POINTER (alpha_deg_ptr, alpha_deg)
291 !
292         DOUBLE PRECISION beta_deg(*)
293         POINTER (beta_deg_ptr, beta_deg)
294 !
295         DOUBLE PRECISION gamma_deg(*)
296         POINTER (gamma_deg_ptr, gamma_deg)
297 !
298         s_11_ptr = SMAFloatArrayAccess(11)
299         C_2_ptr = SMAFloatArrayAccess(12)
300         s_33_ptr = SMAFloatArrayAccess(13)
301         O_F_ptr = SMAFloatArrayAccess(14)
302         H_T_ptr = SMAFloatArrayAccess(15)
303         alpha_deg_ptr = SMAFloatArrayAccess(16)
304         beta_deg_ptr = SMAFloatArrayAccess(17)
305         gamma_deg_ptr = SMAFloatArrayAccess(18)
306 !
307         PI=4.D0*DATAN(1.D0)
308 !
309
310 !       !!!-------1: COORD with respect to Centroid-------!!!
311         xbar=85d0
312         ybar=55d0
313         zbar=500d0
314
315         COORDS_wrtC(1,1)=COORDS(1)-xbar
316         COORDS_wrtC(2,1)=COORDS(2)-ybar
317         COORDS_wrtC(3,1)=COORDS(3)-zbar
318
319
320 !       !!!-------2: rotation matrices Rx, Ry, Rz, R, R_T and their ...
    backrotation matrices-------!!!
321
322 !       Rx and Rx_backrotation
323         alpha=alpha_deg(1)/360d0*2d0*PI
324
325         Rx(1,1) = 1d0
326         Rx(1,2) = 0d0
327         Rx(1,3) = 0d0
328         Rx(2,1) = 0d0
329         Rx(2,2) = dcos(alpha)
330         Rx(2,3) = dsin(alpha)
331         Rx(3,1) = 0d0
332         Rx(3,2) = -dsin(alpha)
333         Rx(3,3) = dcos(alpha)
334
335         Rx_backrotation(1,1) = 1d0
336         Rx_backrotation(1,2) = 0d0
337         Rx_backrotation(1,3) = 0d0
338         Rx_backrotation(2,1) = 0d0
339         Rx_backrotation(2,2) = dcos(-alpha)
```

```
340         Rx_backrotation(2,3) = dsin(-alpha)
341         Rx_backrotation(3,1) = 0d0
342         Rx_backrotation(3,2) = -dsin(-alpha)
343         Rx_backrotation(3,3) = dcos(-alpha)
344
345 !       Ry and Ry_backrotation
346         beta=beta_deg(1)/360d0*2d0*PI
347
348         Ry(1,1) = dcos(beta)
349         Ry(1,2) = 0d0
350         Ry(1,3) = -dsin(beta)
351         Ry(2,1) = 0d0
352         Ry(2,2) = 1d0
353         Ry(2,3) = 0d0
354         Ry(3,1) = dsin(beta)
355         Ry(3,2) = 0d0
356         Ry(3,3) = dcos(beta)
357
358         Ry_backrotation(1,1) = dcos(-beta)
359         Ry_backrotation(1,2) = 0d0
360         Ry_backrotation(1,3) = -dsin(-beta)
361         Ry_backrotation(2,1) = 0d0
362         Ry_backrotation(2,2) = 1d0
363         Ry_backrotation(2,3) = 0d0
364         Ry_backrotation(3,1) = dsin(-beta)
365         Ry_backrotation(3,2) = 0d0
366         Ry_backrotation(3,3) = dcos(-beta)
367
368 !       Rz and Rz_backrotation
369         gamma=gamma_deg(1)/360d0*2d0*PI
370
371         Rz(1,1) = dcos(gamma)
372         Rz(1,2) = dsin(gamma)
373         Rz(1,3) = 0d0
374         Rz(2,1) = -dsin(gamma)
375         Rz(2,2) = dcos(gamma)
376         Rz(2,3) = 0d0
377         Rz(3,1) = 0d0
378         Rz(3,2) = 0d0
379         Rz(3,3) = 1d0
380
381         Rz_backrotation(1,1) = dcos(-gamma)
382         Rz_backrotation(1,2) = dsin(-gamma)
383         Rz_backrotation(1,3) = 0d0
384         Rz_backrotation(2,1) = -dsin(-gamma)
385         Rz_backrotation(2,2) = dcos(-gamma)
386         Rz_backrotation(2,3) = 0d0
387         Rz_backrotation(3,1) = 0d0
388         Rz_backrotation(3,2) = 0d0
389         Rz_backrotation(3,3) = 1d0
390
391 !       R and R_backrotation (R=Rz*Ry*Rx)
392         R1=matmul(Ry,Rx)
393         R=matmul(Rz,R1)
394
```

```
395         R1_backrotation=matmul(Ry_backrotation,Rx_backrotation)
396         R_backrotation=matmul(Rz_backrotation,R1_backrotation)
397
398 !       R_T transposed and R_backrotation_T transposed
399         R_T(1,1) = R(1,1)
400         R_T(1,2) = R(2,1)
401         R_T(1,3) = R(3,1)
402         R_T(2,1) = R(1,2)
403         R_T(2,2) = R(2,2)
404         R_T(2,3) = R(3,2)
405         R_T(3,1) = R(1,3)
406         R_T(3,2) = R(2,3)
407         R_T(3,3) = R(3,3)
408
409         R_backrotation_T(1,1) = R_backrotation(1,1)
410         R_backrotation_T(1,2) = R_backrotation(2,1)
411         R_backrotation_T(1,3) = R_backrotation(3,1)
412         R_backrotation_T(2,1) = R_backrotation(1,2)
413         R_backrotation_T(2,2) = R_backrotation(2,2)
414         R_backrotation_T(2,3) = R_backrotation(3,2)
415         R_backrotation_T(3,1) = R_backrotation(1,3)
416         R_backrotation_T(3,2) = R_backrotation(2,3)
417         R_backrotation_T(3,3) = R_backrotation(3,3)
418
419 !       !!!-------3: determine COORDS_wrtC in rotated/transformed ...
    orientation-------!!!
420 !           -------                 COORDS_wrtC'=R*COORDS_wrtC ...
                    -------
421         COORDS_wrtC_transformed=matmul(R,COORDS_wrtC)
422
423
424 !       !!!-------4: fill sigma_transformed-------!!!
425         sigma_transformed(1,1)=
426         dcos(1d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
427         sigma_transformed(1,2)=0d0
428         sigma_transformed(1,3)=0d0
429         sigma_transformed(2,1)=0d0
430         sigma_transformed(2,2)=0d0
431         sigma_transformed(2,3)=0d0
432         sigma_transformed(3,1)=0d0
433         sigma_transformed(3,2)=0d0
434         sigma_transformed(3,3)=0d0
435
436
437 !       !!!-------5: determine sigma in original ...
    configuration/coordinate system-------!!!
438 !           -------         sigma=R_backrotation*sigma'*R_backrotation_T ...
                  -------
439         sigma_step1=matmul(sigma_transformed,R_backrotation_T)
440         sigma_originalconfiguration=matmul(R_backrotation,sigma_step1)
441
442         SIGMA(1)=sigma_originalconfiguration(1,1)
443         SIGMA(2)=sigma_originalconfiguration(2,2)
444         SIGMA(3)=sigma_originalconfiguration(3,3)
445         SIGMA(4)=sigma_originalconfiguration(1,2)
```

```
446         SIGMA(5)=sigma_originalconfiguration(1,3)
447         SIGMA(6)=sigma_originalconfiguration(2,3)
448
449       RETURN
450       END
451 !
```

The Fortran scripts for the other *eleven* stress modes are equal to the Fortran script that was shown in Fig. C.16, except for a small section containing the definition of the transformed stress matrix. For each stress mode, the sections shown in Fig. C.17-C.27 must be substituted in Fig. C.16.

Fortran code C.17: Section of StressMappingS11Mode2.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=
3       dcos(2d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
4          sigma_transformed(1,2)=0d0
5          sigma_transformed(1,3)=0d0
6          sigma_transformed(2,1)=0d0
7          sigma_transformed(2,2)=0d0
8          sigma_transformed(2,3)=0d0
9          sigma_transformed(3,1)=0d0
10         sigma_transformed(3,2)=0d0
11         sigma_transformed(3,3)=0d0
```

Fortran code C.18: Section of StressMappingS11Mode3.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=
3       dcos(3d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
4          sigma_transformed(1,2)=0d0
5          sigma_transformed(1,3)=0d0
6          sigma_transformed(2,1)=0d0
7          sigma_transformed(2,2)=0d0
8          sigma_transformed(2,3)=0d0
9          sigma_transformed(3,1)=0d0
10         sigma_transformed(3,2)=0d0
11         sigma_transformed(3,3)=0d0
```

Fortran code C.19: Section of StressMappingS11Mode4.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=
3       dcos(4d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
4          sigma_transformed(1,2)=0d0
5          sigma_transformed(1,3)=0d0
6          sigma_transformed(2,1)=0d0
7          sigma_transformed(2,2)=0d0
8          sigma_transformed(2,3)=0d0
9          sigma_transformed(3,1)=0d0
10         sigma_transformed(3,2)=0d0
```

```
11          sigma_transformed(3,3)=0d0
```

Fortran code C.20: Section of StressMappingS11Mode5.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2           sigma_transformed(1,1)=
3       dcos(5d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
4           sigma_transformed(1,2)=0d0
5           sigma_transformed(1,3)=0d0
6           sigma_transformed(2,1)=0d0
7           sigma_transformed(2,2)=0d0
8           sigma_transformed(2,3)=0d0
9           sigma_transformed(3,1)=0d0
10          sigma_transformed(3,2)=0d0
11          sigma_transformed(3,3)=0d0
```

Fortran code C.21: Section of StressMappingS11Mode6.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2           sigma_transformed(1,1)=
3       dcos(6d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
4           sigma_transformed(1,2)=0d0
5           sigma_transformed(1,3)=0d0
6           sigma_transformed(2,1)=0d0
7           sigma_transformed(2,2)=0d0
8           sigma_transformed(2,3)=0d0
9           sigma_transformed(3,1)=0d0
10          sigma_transformed(3,2)=0d0
11          sigma_transformed(3,3)=0d0
```

Fortran code C.22: Section of StressMappingS22Mode1.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2           sigma_transformed(1,1)=0d0
3           sigma_transformed(1,2)=0d0
4           sigma_transformed(1,3)=0d0
5           sigma_transformed(2,1)=0d0
6           sigma_transformed(2,2)=0d0
7           sigma_transformed(2,3)=0d0
8           sigma_transformed(3,1)=0d0
9           sigma_transformed(3,2)=0d0
10          sigma_transformed(3,3)=
11      dcos(1d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

Fortran code C.23: Section of StressMappingS22Mode2.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2           sigma_transformed(1,1)=0d0
3           sigma_transformed(1,2)=0d0
4           sigma_transformed(1,3)=0d0
```

```
5          sigma_transformed(2,1)=0d0
6          sigma_transformed(2,2)=0d0
7          sigma_transformed(2,3)=0d0
8          sigma_transformed(3,1)=0d0
9          sigma_transformed(3,2)=0d0
10         sigma_transformed(3,3)=
11       dcos(2d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

Fortran code C.24: Section of StressMappingS22Mode3.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=0d0
3          sigma_transformed(1,2)=0d0
4          sigma_transformed(1,3)=0d0
5          sigma_transformed(2,1)=0d0
6          sigma_transformed(2,2)=0d0
7          sigma_transformed(2,3)=0d0
8          sigma_transformed(3,1)=0d0
9          sigma_transformed(3,2)=0d0
10         sigma_transformed(3,3)=
11       dcos(3d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

Fortran code C.25: Section of StressMappingS22Mode4.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=0d0
3          sigma_transformed(1,2)=0d0
4          sigma_transformed(1,3)=0d0
5          sigma_transformed(2,1)=0d0
6          sigma_transformed(2,2)=0d0
7          sigma_transformed(2,3)=0d0
8          sigma_transformed(3,1)=0d0
9          sigma_transformed(3,2)=0d0
10         sigma_transformed(3,3)=
11       dcos(4d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

Fortran code C.26: Section of StressMappingS22Mode5.f

```
1  !       !!!-------4: fill sigma_transformed-------!!!
2          sigma_transformed(1,1)=0d0
3          sigma_transformed(1,2)=0d0
4          sigma_transformed(1,3)=0d0
5          sigma_transformed(2,1)=0d0
6          sigma_transformed(2,2)=0d0
7          sigma_transformed(2,3)=0d0
8          sigma_transformed(3,1)=0d0
9          sigma_transformed(3,2)=0d0
10         sigma_transformed(3,3)=
11       dcos(5d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

Fortran code C.27: Section of StressMappingS22Mode6.f

```
!       !!!-------4: fill sigma_transformed-------!!!
        sigma_transformed(1,1)=0d0
        sigma_transformed(1,2)=0d0
        sigma_transformed(1,3)=0d0
        sigma_transformed(2,1)=0d0
        sigma_transformed(2,2)=0d0
        sigma_transformed(2,3)=0d0
        sigma_transformed(3,1)=0d0
        sigma_transformed(3,2)=0d0
        sigma_transformed(3,3)=
     dcos(6d0*2d0*PI*(COORDS_wrtC_transformed(2,1)+O_F(1)+ybar)/H_T(1))
```

The outcome of using the improved method to evaluate robustness of distortion is listed in Tab. C.3.

Table C.3: This table provides an overview of the outcome of the *improved* method employed for evaluating distortion robustness of the elementary beam that was shown in 4.1 and that was subjected to a stochastic stress field as was specified in chapter 2.2.

Outcome of static analysis	Symbol	Value
Mean value of centre node displacement	μ_{u_3}	$-1.442\,096\,\text{mm}$
Standard deviation of centre node displacement	s_{u_3}	$3.078\,827 \times 10^{-1}\,\text{mm}$
Number of static analyses required	m	12
Computation time	t	$\approx 76\,\text{sec}$

The improved method seems to yield correct results, since the values obtained for the mean and standard deviation of the centro node displacement, $\overline{x}_{u_3}$ and s_{u_3}, respectively, are equal to the values obtained with the state-of-the-art method of which the outcome was shown in Tab. C.2.

Appendix D

Case study

D.1 Distortion curves for translation z_1 in z-direction

The *distortion curves* of the bottom centre line of the stiffener component as a function of translation of the stiffener comonent in z-direction with respect to rolled plate in 21 discrete steps are shown In Fig. D.1. Only distortion curves in z-direction are concerned.

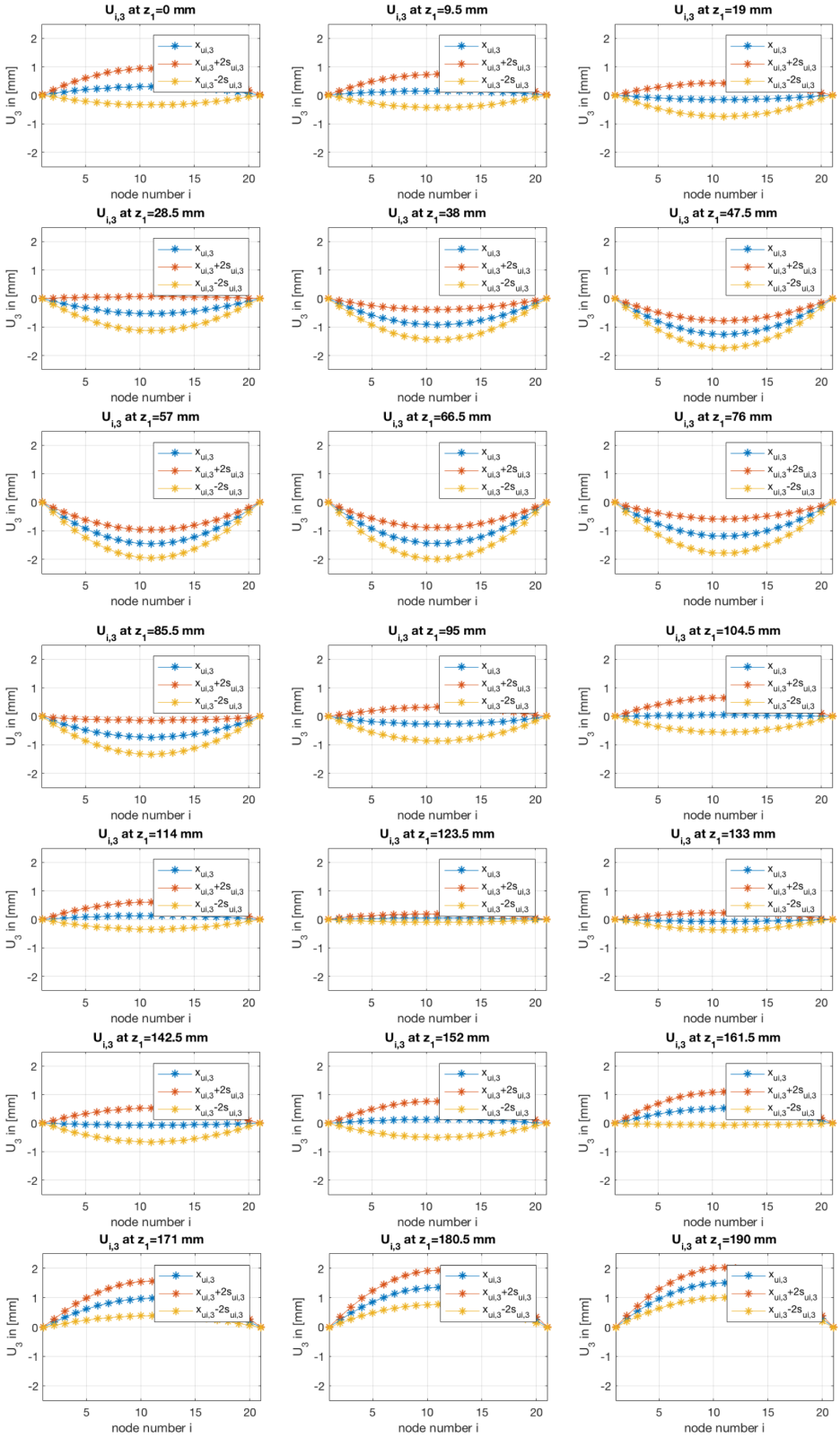


Figure D.1: The stiffener component is translated in *z-direction* in 21 discrete steps from 0 mm ≤ z_1 ≤ 190 mm. For each step of z_1, the mean distortion curve (blue line), the mean-plus-two-standard-deviations distortion curve (red line) and the mean-minus-two-standard-devations distortion curve (yellow line) are plotted. The area in between the red and yellow line represents the area where 95.4 % of distortion of the component's bottom centre lines will occur.

D.2 Distortion curves for rotation β about the y-axis

The *distortion curves* of the bottom centre line of the stiffener component as a function of rotation of the stiffener component about the y-axis with respect to rolled plate in 21 discrete steps are elaborated. Distortion curves in x-, y- and z-direction are depicted in Fig. D.2, D.3 and D.4, respectively.

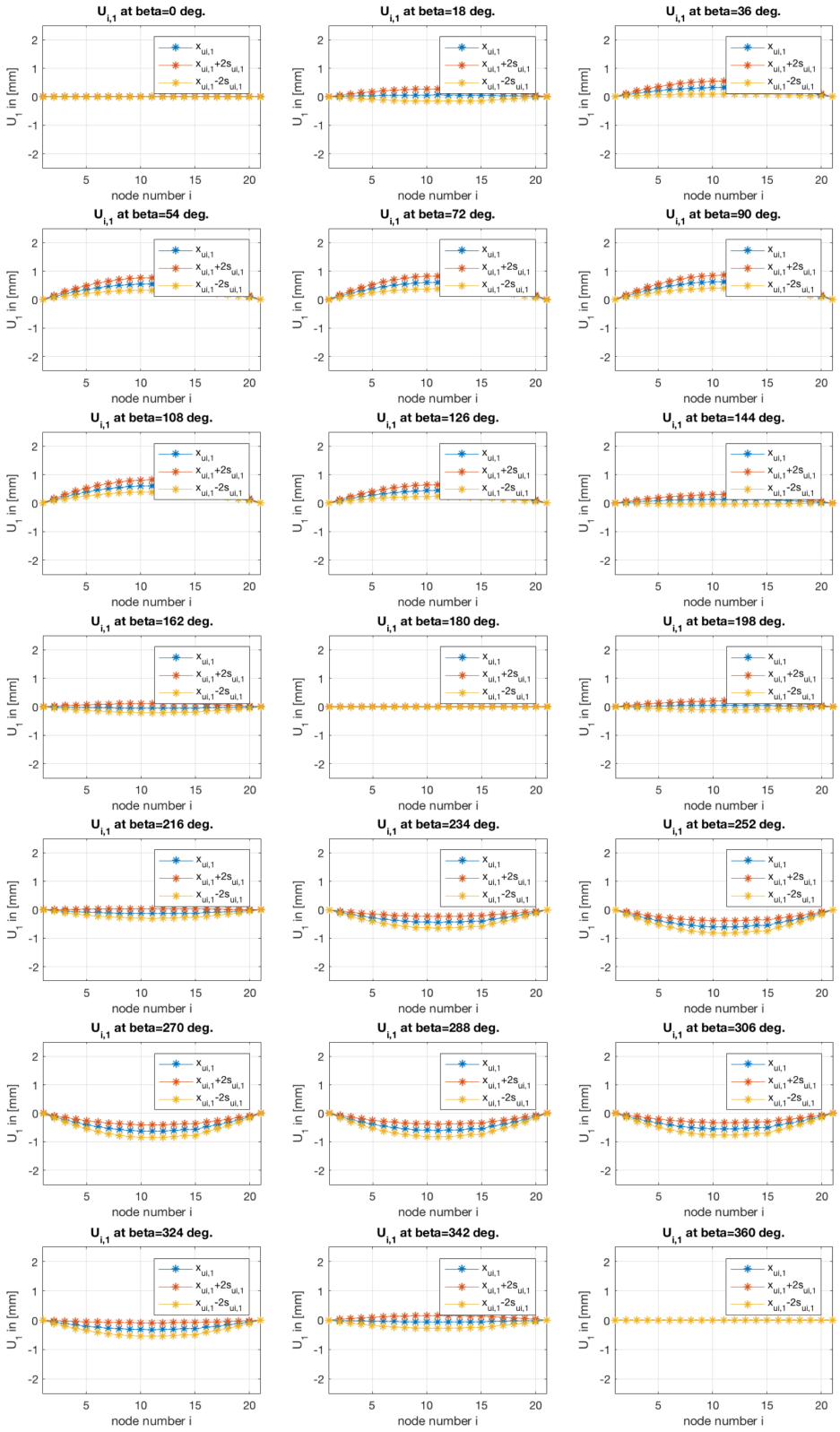


Figure D.2: The stiffener component is rotated about the *y-axis* in 21 discrete steps from $0° \leq \beta \leq 190°$. For each step of β, the mean distortion curve (blue line), the mean-plus-two-standard-deviations distortion curve (red line) and the mean-minus-two-standard-devations distortion curve (yellow line) are plotted in *x-direction*. The area in between the red and yellow line represents the area where 95.4 % of distortion of the component's bottom centre lines will occur.

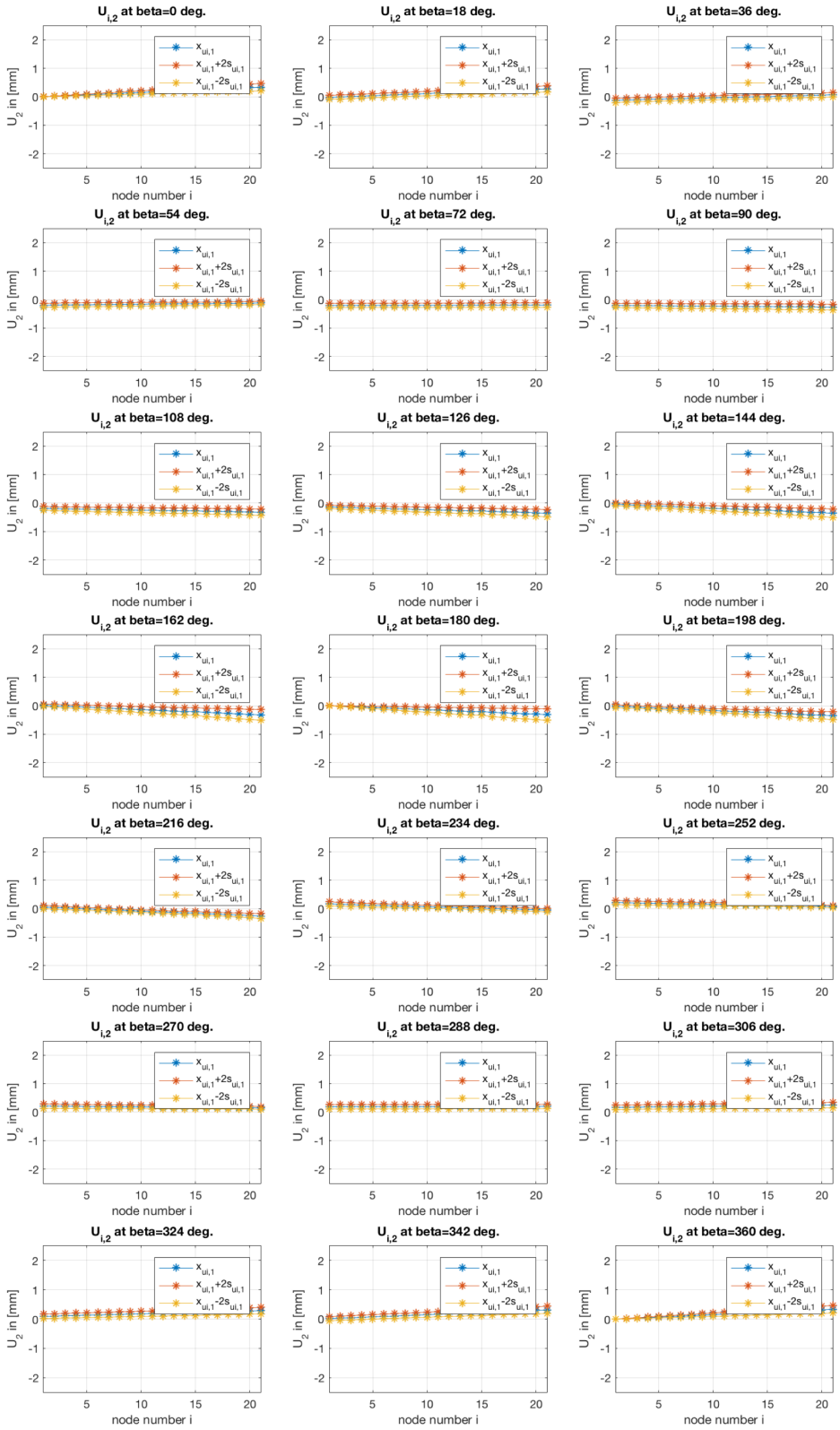


Figure D.3: The stiffener component is rotated about the *y-axis* in 21 discrete steps from $0° \leq \beta \leq 190°$. For each step of β, the mean distortion curve (blue line), the mean-plus-two-standard-deviations distortion curve (red line) and the mean-minus-two-standard-devations distortion curve (yellow line) are plotted in *y-direction*. The area in between the red and yellow line represents the area where 95.4 % of distortion of the component's bottom centre lines will occur.

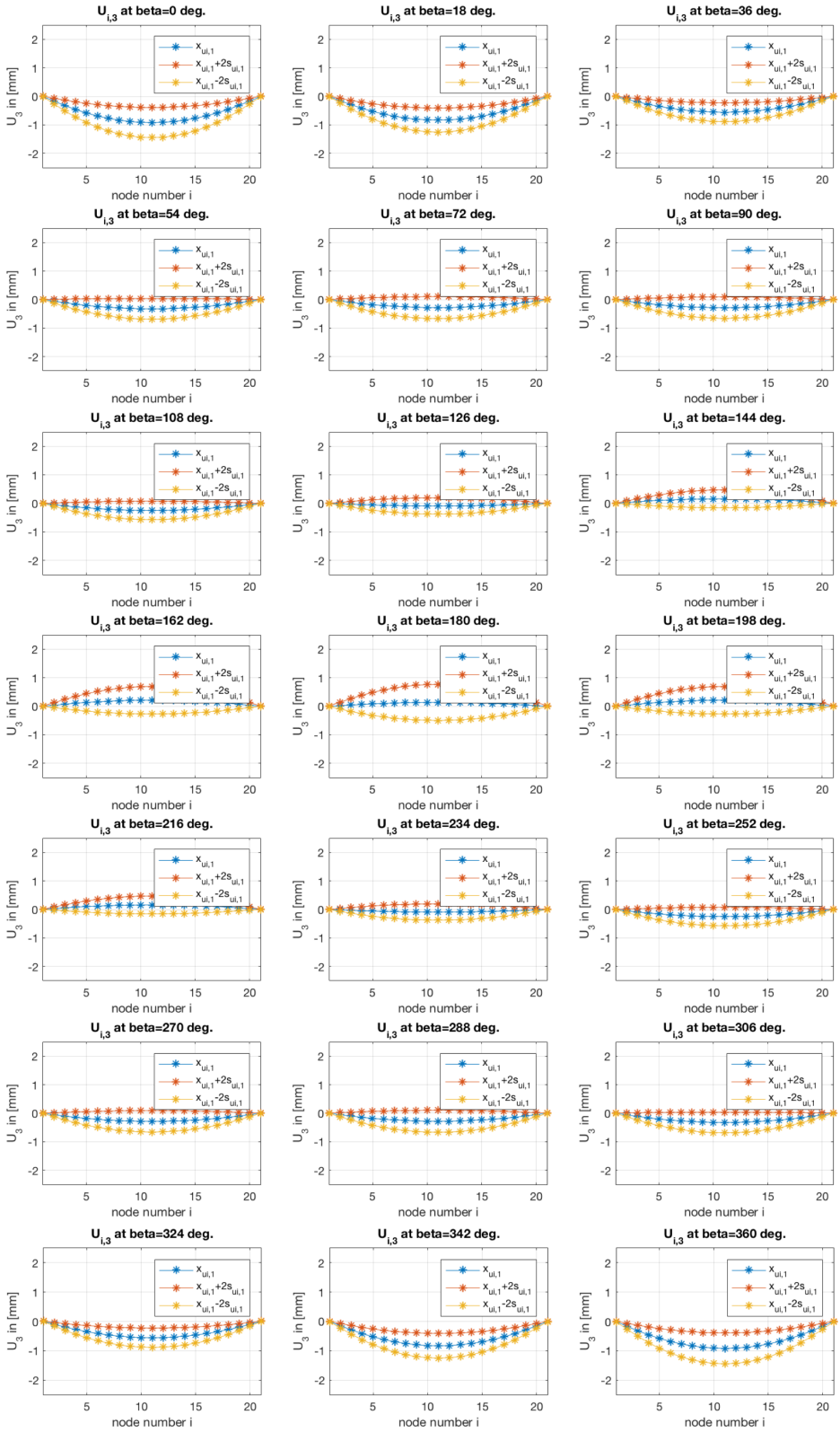


Figure D.4: The stiffener component is rotated about the *y-axis* in 21 discrete steps from $0° \leq \beta \leq 360°$. For each step of β, the mean distortion curve (blue line), the mean-plus-two-standard-deviations distortion curve (red line) and the mean-minus-two-standard-devations distortion curve (yellow line) are plotted in *z-direction*. The area in between the red and yellow line represents the area where 95.4 % of distortion of the component's bottom centre lines will occur.

D.3 The improved method applied to the case study

1. The geometry is imported in Abaqus/CAE©. In Abaqus/CAE©, the geometry is meshed, iso-static boundary conditions are put in place and node-sets for each node on the *bottom centre line* are created. Fig. D.5 shows the mesh of the stiffener component consisting of hexahedron elements.

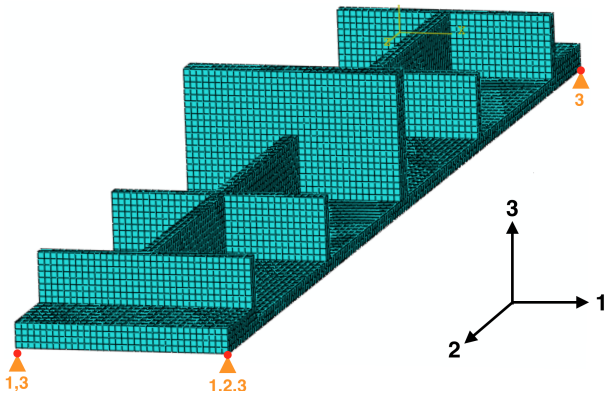


Figure D.5: This figure shows the mesh of the stiffener component containing hexahedron elements. Iso-static boundary conditions are put in place on the bottom plane of the component assuring that all rigid body modes are blocked.

2. An input file (.inp) is generated in Abaqus/CAE©. For reasons of clarity, the input file is splitted into multiple input files such that the main input file *sim.inp* calls *sim-els.inp* in which the elements are defined, *sim-nds.inp* in which the nodes are defined, *sim-sts.inp*, *sim-simp.inp* in which the material is defined, *sim-bcs.inp* in which the boundary conditions are defined and *sim-stp.inp* in which the step definition is defined. These input files were shown earlier - for a different geometry - in Fig. C.1-C.7.

3. The entire analysis is launched from the *cluster* computer network using a *.sub* file like the one that was used for the deterministic analysis of the elementary beam in Fig. C.8.

4. The *submit.sub* file launches a Python© script called *ImprovedMethod.py* which is shown in Fig. D.1. The Python© script *ImprovedMethod.py* calls the Python© script *funcImprovedMethod.py* for a number of combinations of offset z_1 and angle β.

5. In the Python© script *funcImprovedMethod.py* (Fig. D.2) the *improved method* for evaluating robustness (elaborated in chapter 4.4) is executed. For a given combination of offset z_1 and angle β, individual static analyses for all twelve cosine modes are performed. For each of the twelve analyses different stress mapping Fortran© scripts are called. These are equivalent to the Fortran© scripts that were shown earlier in Fig. C.16-C.27. Subsequently, displacements for all nodes on the bottom centre line in all three directions are collected from the *.odb* files.

6. Within *funcImprovedMethod.py*, the static analyses are launched after which the displacements in three directions of all nodes on the bottom centre line are extracted from the *.odb* files via *ExtractUfromodb.py* which is shown in Fig. D.5. After this, *mean*

displacement is obtained *analytically* for all nodes in all three directions separately by calling Python© script *funcAnMean.py* (Fig. D.3). Likewise, the *variance* and standard deviation of displacement is obtained analytically for all nodes in all three directions by calling Python© script *funcAnVariance.py* (Fig. D.4). At the end, for each combination of offset z_1 and angle β, mean displacement and standard deviation of all nodes for all three directions are written to text files (*.dat*).

7. The measures for distortion magnitude and distortion robustness, which were formulated in Eq. 5.4,5.5- are subsequently determined for each combination of offset z_1 and angle β.

Python code D.1: ImprovedMethod.py

```
from __future__ import division          #decimal division
#import matplotlib.pyplot as plt
import funcImprovedMethod
import timeit
import os
import shutil


#total DSE run-time count
start_time_totalDSE = timeit.default_timer()

L=300
h=110

n_intervals=20

#create alpha_deg.dbl
alpha_deg=0.0
fo=open('alpha_deg.dbl','w+')
fo.write( '%e' % alpha_deg )
fo.close()

#create beta_deg.dbl
beta_deg=0.0
fo=open('beta_deg.dbl','w+')
fo.write( '%e' % beta_deg )
fo.close()

##------------ 1: DSE  ------------##
for j in range(n_intervals+1):

    z1=(L-h)/n_intervals*j

    #create O_F.dbl
    fo=open('O_F.dbl','w+')
    fo.write( '%e' % z1 )
    fo.close()

    for i in range(n_intervals+1):

```

```
41          gamma_deg=360/n_intervals*i
42
43          #create gamma_deg.dbl
44          fo=open('gamma_deg.dbl','w+')
45          fo.write( '%e' % gamma_deg )
46          fo.close()
47
48          funcImprovedMethod.fImprovedMethod(i,j,z1,alpha_deg,beta_deg,gamma_deg)
49
50  ##------------ 1: DSE  ------------##
51
52
53  #end timer StaticAnalyses
54  elapsed_totalScript = timeit.default_timer() - start_time_totalDSE
55  print("--- computation time for total Script run is %s seconds ---" ...
        % elapsed_totalScript)
```

Python code D.2: funcImprovedMethod.py

```
1   from __future__ import division             #decimal division
2   import math
3   import os
4   import random
5   import timeit
6   #import matplotlib.pyplot as plt
7   import shutil
8   import numpy
9   import funcImportvaluefromFile
10  import funcAnMn
11  import funcAnVr
12
13  def fImprovedMethod(i,j,z1,alpha_deg,beta_deg,gamma_deg):
14
15
16      ##------------ 0: constants ------------##
17      s_11_1=-8.1936
18      s_11_2=2.4317
19      s_11_3=2.9584
20      s_11_4=1.3215
21      s_11_5=0.9406
22      s_11_6=1.0299
23
24      s_33_1=-27.0343
25      s_33_2=6.1362
26      s_33_3=7.8890
27      s_33_4=-1.4510
28      s_33_5=1.0052
29      s_33_6=0.8000
30
31
32      ##------------ 1: Run sim.inp, extract displacement ------------##
33      #start timer Static Analyses
34      start_time_StaticAnalyses = timeit.default_timer()
35
```

```
36      ### Sigma11, Mode 1
37      os.system('abaqus double job=sim user=StressMapping_S11_Mode1 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
38      os.system("abaqus python ExtractU2fromodb.py")
39
40      #node 1
41      U1_N1_Mode1_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
42      U2_N1_Mode1_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
43      U3_N1_Mode1_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)
44
45      ----------------------------------------
46      similar for other 20 nodes on distortion centre line
47      ----------------------------------------
48
49      ### Sigma11, Mode 2
50      os.system('abaqus double job=sim user=StressMapping_S11_Mode2 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
51      os.system("abaqus python ExtractU2fromodb.py")
52
53      U1_N1_Mode2_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
54      U2_N1_Mode2_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
55      U3_N1_Mode2_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)
56
57      ----------------------------------------
58      similar for other 20 nodes on distortion centre line
59      ----------------------------------------
60
61      ### Sigma11, Mode 3
62      os.system('abaqus double job=sim user=StressMapping_S11_Mode3 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
63      os.system("abaqus python ExtractU2fromodb.py")
64
65      U1_N1_Mode3_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
66      U2_N1_Mode3_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
67      U3_N1_Mode3_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)
68
69      ----------------------------------------
70      similar for other 20 nodes on distortion centre line
71      ----------------------------------------
72
73      ### Sigma11, Mode 4
74      os.system('abaqus double job=sim user=StressMapping_S11_Mode4 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
75      os.system("abaqus python ExtractU2fromodb.py")
76
77      U1_N1_Mode4_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
78      U2_N1_Mode4_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
79      U3_N1_Mode4_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)
80
81      ----------------------------------------
82      similar for other 20 nodes on distortion centre line
83      ----------------------------------------
84
85      ### Sigma11, Mode 5
86      os.system('abaqus double job=sim user=StressMapping_S11_Mode5 ...
```

```
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU2fromodb.py")

    U1_N1_Mode5_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
    U2_N1_Mode5_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
    U3_N1_Mode5_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------

    ### Sigma11, Mode 6
    os.system('abaqus double job=sim user=StressMapping_S11_Mode6 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU2fromodb.py")

    U1_N1_Mode6_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(1,1)
    U2_N1_Mode6_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(2,1)
    U3_N1_Mode6_Sigma11=funcImportvaluefromFile.fImportvaluefromFile(3,1)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------

    ### Sigma33, Mode 1
    os.system('abaqus double job=sim user=StressMapping_S22_Mode1 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU2fromodb.py")

    U1_N1_Mode1_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
    U2_N1_Mode1_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
    U3_N1_Mode1_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------

    ### Sigma33, Mode 2
    os.system('abaqus double job=sim user=StressMapping_S22_Mode2 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU2fromodb.py")

    U1_N1_Mode2_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
    U2_N1_Mode2_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
    U3_N1_Mode2_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------

    ### Sigma33, Mode 3
    os.system('abaqus double job=sim user=StressMapping_S22_Mode3 ...
        cpus=1 scratch=/home/ycejanssens/ memory=32gb')
    os.system("abaqus python ExtractU2fromodb.py")
```

```
137     U1_N1_Mode3_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
138     U2_N1_Mode3_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
139     U3_N1_Mode3_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)
140
141     ----------------------------------------
142     similar for other 20 nodes on distortion centre line
143     ----------------------------------------
144
145     ### Sigma33, Mode 4
146     os.system('abaqus double job=sim user=StressMapping_S22_Mode4 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
147     os.system("abaqus python ExtractU2fromodb.py")
148
149     U1_N1_Mode4_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
150     U2_N1_Mode4_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
151     U3_N1_Mode4_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)
152
153     ----------------------------------------
154     similar for other 20 nodes on distortion centre line
155     ----------------------------------------
156
157     ### Sigma33, Mode 5
158     os.system('abaqus double job=sim user=StressMapping_S22_Mode5 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
159     os.system("abaqus python ExtractU2fromodb.py")
160
161     U1_N1_Mode5_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
162     U2_N1_Mode5_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
163     U3_N1_Mode5_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)
164
165     ----------------------------------------
166     similar for other 20 nodes on distortion centre line
167     ----------------------------------------
168
169     ### Sigma33, Mode 6
170     os.system('abaqus double job=sim user=StressMapping_S22_Mode6 ...
            cpus=1 scratch=/home/ycejanssens/ memory=32gb')
171     os.system("abaqus python ExtractU2fromodb.py")
172
173     U1_N1_Mode6_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(1,1)
174     U2_N1_Mode6_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(2,1)
175     U3_N1_Mode6_Sigma33=funcImportvaluefromFile.fImportvaluefromFile(3,1)
176
177     ----------------------------------------
178     similar for other 20 nodes on distortion centre line
179     ----------------------------------------
180
181     ##-end timer StaticAnalyses-##
182     elapsed_StaticAnalyses = timeit.default_timer() - ...
            start_time_StaticAnalyses
183     print("--- computation time for Static Analyses is %s seconds ...
            ---" % elapsed_StaticAnalyses)
184
185
186     ##------------ 2: analytical distribution derivation  ------------##
```

```
187     #start timer Monte-Carlo
188     start_time_MonteCarlo = timeit.default_timer()
189 
190     s11_1_mean=s_11_1
191     s11_2_mean=s_11_2
192     s11_3_mean=s_11_3
193     s11_4_mean=s_11_4
194     s11_5_mean=s_11_5
195     s11_6_mean=s_11_6
196 
197     s33_1_mean=s_33_1
198     s33_2_mean=s_33_2
199     s33_3_mean=s_33_3
200     s33_4_mean=s_33_4
201     s33_5_mean=s_33_5
202     s33_6_mean=s_33_6
203 
204     s11_1_std=2.9313
205     s11_2_std=1.2281
206     s11_3_std=1.2097
207     s11_4_std=1.0429
208     s11_5_std=0.9294
209     s11_6_std=0.7252
210 
211     s33_1_std=4.0475
212     s33_2_std=3.8186
213     s33_3_std=1.9898
214     s33_4_std=2.4577
215     s33_5_std=2.0155
216     s33_6_std=0.9293
217 
218     ###-Analytical mean derivation-###
219 
220     ### N1 ###
221     #U1
222     mean_U1_N1_analytical=funcAnMn.fAnalyticalMean(U1_N1_Mode1_Sigma11,\
223                                                    U1_N1_Mode2_Sigma11,\
224                                                    U1_N1_Mode3_Sigma11,\
225                                                    U1_N1_Mode4_Sigma11,\
226                                                    U1_N1_Mode5_Sigma11,\
227                                                    U1_N1_Mode6_Sigma11,\
228                                                    U1_N1_Mode1_Sigma33,\
229                                                    U1_N1_Mode2_Sigma33,\
230                                                    U1_N1_Mode3_Sigma33,\
231                                                    U1_N1_Mode4_Sigma33,\
232                                                    U1_N1_Mode5_Sigma33,\
233                                                    U1_N1_Mode6_Sigma33)
234 
235     #U2
236     mean_U2_N1_analytical=funcAnMn.fAnalyticalMean(U2_N1_Mode1_Sigma11,\
237                                                    U2_N1_Mode2_Sigma11,\
238                                                    U2_N1_Mode3_Sigma11,\
239                                                    U2_N1_Mode4_Sigma11,\
240                                                    U2_N1_Mode5_Sigma11,\
241                                                    U2_N1_Mode6_Sigma11,\
```

```
                                                        U2_N1_Mode1_Sigma33,\
                                                        U2_N1_Mode2_Sigma33,\
                                                        U2_N1_Mode3_Sigma33,\
                                                        U2_N1_Mode4_Sigma33,\
                                                        U2_N1_Mode5_Sigma33,\
                                                        U2_N1_Mode6_Sigma33)

    #U3
    mean_U3_N1_analytical=funcAnMn.fAnalyticalMean(U3_N1_Mode1_Sigma11,\
                                                        U3_N1_Mode2_Sigma11,\
                                                        U3_N1_Mode3_Sigma11,\
                                                        U3_N1_Mode4_Sigma11,\
                                                        U3_N1_Mode5_Sigma11,\
                                                        U3_N1_Mode6_Sigma11,\
                                                        U3_N1_Mode1_Sigma33,\
                                                        U3_N1_Mode2_Sigma33,\
                                                        U3_N1_Mode3_Sigma33,\
                                                        U3_N1_Mode4_Sigma33,\
                                                        U3_N1_Mode5_Sigma33,\
                                                        U3_N1_Mode6_Sigma33)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------


    ###-Analytical Variance-###

    ### N1 ###
    #U1
    Variance_U1_N1_analytical=funcAnVr.fAnalyticalVar(U1_N1_Mode1_Sigma11,\
                                                        U1_N1_Mode2_Sigma11,\
                                                        U1_N1_Mode3_Sigma11,\
                                                        U1_N1_Mode4_Sigma11,\
                                                        U1_N1_Mode5_Sigma11,\
                                                        U1_N1_Mode6_Sigma11,\
                                                        U1_N1_Mode1_Sigma33,\
                                                        U1_N1_Mode2_Sigma33,\
                                                        U1_N1_Mode3_Sigma33,\
                                                        U1_N1_Mode4_Sigma33,\
                                                        U1_N1_Mode5_Sigma33,\
                                                        U1_N1_Mode6_Sigma33)

    #U2
    Variance_U2_N1_analytical=funcAnVr.fAnalyticalVar(U2_N1_Mode1_Sigma11,\
                                                        U2_N1_Mode2_Sigma11,\
                                                        U2_N1_Mode3_Sigma11,\
                                                        U2_N1_Mode4_Sigma11,\
                                                        U2_N1_Mode5_Sigma11,\
                                                        U2_N1_Mode6_Sigma11,\
                                                        U2_N1_Mode1_Sigma33,\
                                                        U2_N1_Mode2_Sigma33,\
                                                        U2_N1_Mode3_Sigma33,\
                                                        U2_N1_Mode4_Sigma33,\
                                                        U2_N1_Mode5_Sigma33,\
```

```
                                                        U2_N1_Mode6_Sigma33)

    #U3
    Variance_U3_N1_analytical=funcAnVr.fAnalyticalVar(U3_N1_Mode1_Sigma11,\
                                                        U3_N1_Mode2_Sigma11,\
                                                        U3_N1_Mode3_Sigma11,\
                                                        U3_N1_Mode4_Sigma11,\
                                                        U3_N1_Mode5_Sigma11,\
                                                        U3_N1_Mode6_Sigma11,\
                                                        U3_N1_Mode1_Sigma33,\
                                                        U3_N1_Mode2_Sigma33,\
                                                        U3_N1_Mode3_Sigma33,\
                                                        U3_N1_Mode4_Sigma33,\
                                                        U3_N1_Mode5_Sigma33,\
                                                        U3_N1_Mode6_Sigma33)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------


    ###-Analytical Standard deviation-###

    ### N1 ###
    std_U1_N1_analytical=math.sqrt(Variance_U1_N1_analytical)
    std_U2_N1_analytical=math.sqrt(Variance_U2_N1_analytical)
    std_U3_N1_analytical=math.sqrt(Variance_U3_N1_analytical)

    ----------------------------------------
    similar for other 20 nodes on distortion centre line
    ----------------------------------------


    ##------------ 3: Write to files  ------------##.
    ## Mean
    Textfile_mean_U1='mean_U1_j%s_i%s.dat' % (j,i)
    fo=open(Textfile_mean_U1,'w+')       #creates it for us if not ...
       already created
    fo.write('%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
              \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
              % (mean_U1_N1_analytical,mean_U1_N2_analytical,\
                 mean_U1_N3_analytical,\
                 mean_U1_N4_analytical,mean_U1_N5_analytical,\
                 mean_U1_N6_analytical,\
                 mean_U1_N7_analytical,mean_U1_N8_analytical,\
                 mean_U1_N9_analytical,\
                 mean_U1_N10_analytical,mean_U1_N11_analytical,\
                 mean_U1_N12_analytical,\
                 mean_U1_N13_analytical,mean_U1_N14_analytical,\
                 mean_U1_N15_analytical,\
                 mean_U1_N16_analytical,mean_U1_N17_analytical,\
                 mean_U1_N18_analytical,\
                 mean_U1_N19_analytical,mean_U1_N20_analytical,\
                 mean_U1_N21_analytical) )
    fo.close()
```

```
351
352     Textfile_mean_U2='mean_U2_j%s_i%s.dat' % (j,i)
353     fo=open(Textfile_mean_U2,'w+')       #creates it for us if not ...
            already created
354     fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
355               \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
356               % (mean_U2_N1_analytical,mean_U2_N2_analytical,\
357                  mean_U2_N3_analytical, \
358                  mean_U2_N4_analytical,mean_U2_N5_analytical,\
359                  mean_U2_N6_analytical, \
360                  mean_U2_N7_analytical,mean_U2_N8_analytical,\
361                  mean_U2_N9_analytical, \
362                  mean_U2_N10_analytical,mean_U2_N11_analytical,\
363                  mean_U2_N12_analytical, \
364                  mean_U2_N13_analytical,mean_U2_N14_analytical,\
365                  mean_U2_N15_analytical, \
366                  mean_U2_N16_analytical,mean_U2_N17_analytical,\
367                  mean_U2_N18_analytical, \
368                  mean_U2_N19_analytical,mean_U2_N20_analytical,\
369                  mean_U2_N21_analytical) )
370     fo.close()
371
372     Textfile_mean_U3='mean_U3_j%s_i%s.dat' % (j,i)
373     fo=open(Textfile_mean_U3,'w+')       #creates it for us if not ...
            already created
374     fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
375               \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
376               % (mean_U3_N1_analytical,mean_U3_N2_analytical,\
377                  mean_U3_N3_analytical, \
378                  mean_U3_N4_analytical,mean_U3_N5_analytical,\
379                  mean_U3_N6_analytical, \
380                  mean_U3_N7_analytical,mean_U3_N8_analytical,\
381                  mean_U3_N9_analytical, \
382                  mean_U3_N10_analytical,mean_U3_N11_analytical,\
383                  mean_U3_N12_analytical, \
384                  mean_U3_N13_analytical,mean_U3_N14_analytical,\
385                  mean_U3_N15_analytical, \
386                  mean_U3_N16_analytical,mean_U3_N17_analytical,\
387                  mean_U3_N18_analytical, \
388                  mean_U3_N19_analytical,mean_U3_N20_analytical,\
389                  mean_U3_N21_analytical) )
390     fo.close()
391
392
393     ## Std
394     Textfile_std_U1='std_U1_j%s_i%s.dat' % (j,i)
395     fo=open(Textfile_std_U1,'w+')       #creates it for us if not ...
            already created
396     fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
397               \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
398               % (std_U1_N1_analytical,std_U1_N2_analytical,\
399                  std_U1_N3_analytical, \
400                  std_U1_N4_analytical,std_U1_N5_analytical,\
401                  std_U1_N6_analytical, \
402                  std_U1_N7_analytical,std_U1_N8_analytical,\
```

```
                    std_U1_N9_analytical, \
                    std_U1_N10_analytical,std_U1_N11_analytical,\
                    std_U1_N12_analytical, \
                    std_U1_N13_analytical,std_U1_N14_analytical,\
                    std_U1_N15_analytical, \
                    std_U1_N16_analytical,std_U1_N17_analytical,\
                    std_U1_N18_analytical, \
                    std_U1_N19_analytical,std_U1_N20_analytical,\
                    std_U1_N21_analytical) )
    fo.close()

    Textfile_std_U2='std_U2_j%s_i%s.dat' % (j,i)
    fo=open(Textfile_std_U2,'w+')       #creates it for us if not ...
        already created
    fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
              \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
              % (std_U2_N1_analytical,std_U2_N2_analytical,\
                 std_U2_N3_analytical, \
                 std_U2_N4_analytical,std_U2_N5_analytical,\
                 std_U2_N6_analytical, \
                 std_U2_N7_analytical,std_U2_N8_analytical,\
                 std_U2_N9_analytical, \
                 std_U2_N10_analytical,std_U2_N11_analytical,\
                 std_U2_N12_analytical, \
                 std_U2_N13_analytical,std_U2_N14_analytical,\
                 std_U2_N15_analytical, \
                 std_U2_N16_analytical,std_U2_N17_analytical,\
                 std_U2_N18_analytical, \
                 std_U2_N19_analytical,std_U2_N20_analytical,\
                 std_U2_N21_analytical) )
    fo.close()

    Textfile_std_U3='std_U3_j%s_i%s.dat' % (j,i)
    fo=open(Textfile_std_U3,'w+')       #creates it for us if not ...
        already created
    fo.write( '%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e ... \
              \n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n%e\n'\
              % (std_U3_N1_analytical,std_U3_N2_analytical,\
                 std_U3_N3_analytical, \
                 std_U3_N4_analytical,std_U3_N5_analytical,\
                 std_U3_N6_analytical, \
                 std_U3_N7_analytical,std_U3_N8_analytical,\
                 std_U3_N9_analytical, \
                 std_U3_N10_analytical,std_U3_N11_analytical,\
                 std_U3_N12_analytical, \
                 std_U3_N13_analytical,std_U3_N14_analytical,\
                 std_U3_N15_analytical, \
                 std_U3_N16_analytical,std_U3_N17_analytical,\
                 std_U3_N18_analytical, \
                 std_U3_N19_analytical,std_U3_N20_analytical,\
                 std_U3_N21_analytical) )
    fo.close()

    return
```

Python code D.3: funcAnMean.py

```
from __future__ import division              #decimal division

def fAnalyticalMean(M1_S11,M2_S11,M3_S11,M4_S11,M5_S11,M6_S11, \
             M1_S33,M2_S33,M3_S33,M4_S33,M5_S33,M6_S33):

    S11_1=-8.1936
    S11_2=2.4317
    S11_3=2.9584
    S11_4=1.3215
    S11_5=0.9406
    S11_6=1.0299

    S33_1=-27.0343
    S33_2=6.1362
    S33_3=7.8890
    S33_4=-1.4510
    S33_5=1.0052
    S33_6=0.8000

    mean=S11_1*M1_S11+S11_2*M2_S11+S11_3*M3_S11+\
         S11_4*M4_S11+S11_5*M5_S11+S11_6*M6_S11+\
         S33_1*M1_S33+S33_2*M2_S33+S33_3*M3_S33+\
         S33_4*M4_S33+S33_5*M5_S33+S33_6*M6_S33

    return mean
```

Python code D.4: funcAnVariance.py

```
from __future__ import division              #decimal division

def fAnalyticalVar(M1_S11,M2_S11,M3_S11,M4_S11,M5_S11,M6_S11, \
                         M1_S33,M2_S33,M3_S33,M4_S33,M5_S33,M6_S33):

    S11_1_std=2.9313
    S11_2_std=1.2281
    S11_3_std=1.2097
    S11_4_std=1.0429
    S11_5_std=0.9294
    S11_6_std=0.7252

    S33_1_std=4.0475
    S33_2_std=3.8186
    S33_3_std=1.9898
    S33_4_std=2.4577
    S33_5_std=2.0155
    S33_6_std=0.9293

    variance=(S11_1_std*M1_S11)**2 + (S11_2_std*M2_S11)**2 + ...
        (S11_3_std*M3_S11)**2 + \
               (S11_4_std*M4_S11)**2 + (S11_5_std*M5_S11)**2 + ...
                   (S11_6_std*M6_S11)**2 + \
               (S33_1_std*M1_S33)**2 + (S33_2_std*M2_S33)**2 + ...
                   (S33_3_std*M3_S33)**2 + \
```

```
23                 (S33_4_std*M4_S33)**2 + (S33_5_std*M5_S33)**2 + ...
                       (S33_6_std*M6_S33)**2
24
25      return variance
```

Python code D.5: ExtractUfromodb.py

```
1  if __name__ == "__main__":
2
3      from odbAccess import openOdb
4      odb=openOdb('sim.odb')
5
6      #N1
7      dataU2 = odb.steps['Step-1'].historyRegions['Node ...
           PART-15-1.3221'].historyOutputs['U2'].data[1]
8      dataU2str=str(dataU2)
9      len_firstpart=len('(1.0, ')
10     U2_N1=float(dataU2str[len_firstpart:-1])
11
12     fo=open('U2_N1.dat','w+')        #creates it for us if not ...
           already created
13     fo.write( '%e' % U2_N1 )
14     fo.close()
15
16
17     dataU1 = odb.steps['Step-1'].historyRegions['Node ...
           PART-15-1.3221'].historyOutputs['U1'].data[1]
18     dataU1str=str(dataU1)
19     len_firstpart=len('(1.0, ')
20     U1_N1=float(dataU1str[len_firstpart:-1])
21
22     fo=open('U1_N1.dat','w+')        #creates it for us if not ...
           already created
23     fo.write( '%e' % U1_N1 )
24     fo.close()
25
26
27     dataU3 = odb.steps['Step-1'].historyRegions['Node ...
           PART-15-1.3221'].historyOutputs['U3'].data[1]
28     dataU3str=str(dataU3)
29     len_firstpart=len('(1.0, ')
30     U3_N1=float(dataU3str[len_firstpart:-1])
31
32     fo=open('U3_N1.dat','w+')        #creates it for us if not ...
           already created
33     fo.write( '%e' % U3_N1 )
34     fo.close()
35
36     ------------------------------------------------------------------------
37     same steps are repeated for the other 20 nodes on the distortion ...
           line
38     ------------------------------------------------------------------------
```

TU Delft

AIRBUS

Challenge the future