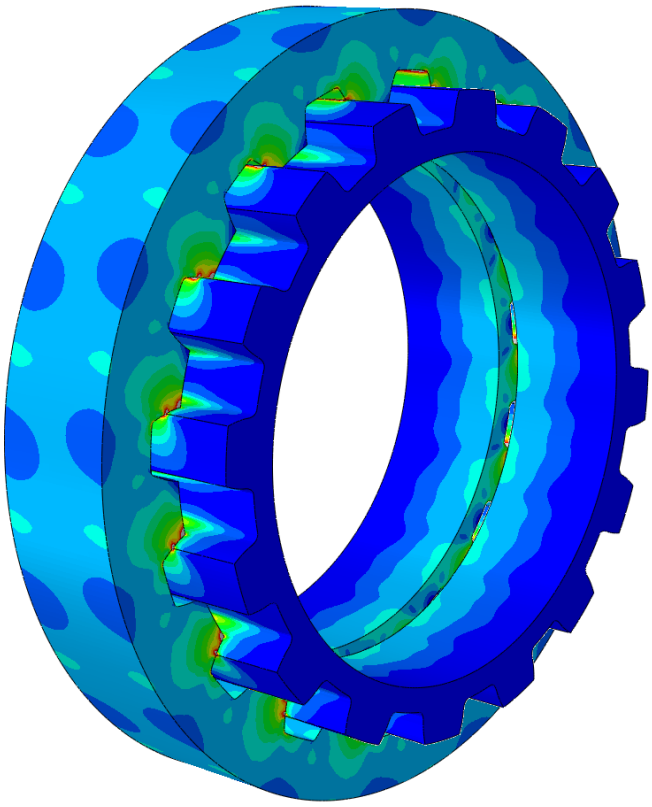
CHALMERS
UNIVERSITY OF TECHNOLOGY



Spline optimization tool using Finite Element Analysis

Master's thesis in Applied Mechanics

Anton Stenstrand

Department of Industrial and Materials Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

MASTER'S THESIS IN APPLIED MECHANICS

Spline optimization tool using Finite Element Analysis

ANTON STENSTRAND

Department of Industrial and Materials Science
Division of Materials & Computational Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2018

Spline optimization tool using Finite Element Analysis

ANTON STENSTRAND

© ANTON STENSTRAND, 2018

Master's thesis 2018
Department of Industrial and Materials Science
Division of Materials & Computational Mechanics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
3-part involute spline coupling showing the effective *Von Mises* stress distribution

Chalmers Reproservice
Göteborg, Sweden 2018

Spline optimization tool using Finite Element Analysis
Master's thesis in Applied Mechanics
ANTON STENSTRAND
Department of Industrial and Materials Science
Division of Materials & Computational Mechanics
Chalmers Univeristy of Technology

Abstract

The current method for spline design at the *Driveline department* consists of several steps. First the initial design is established using analytical calculations, then a 3-dimensional CAD-model is created based on the analytically derived geometrical data. If deemed necessary this model is then manually discretisized with necessary boundary conditions introduced and solved using the Finite Element Method. If needed additional changes are made to the CAD-model and the final design is obtained by through iteration. The main goal of this thesis is to create a program that can automate the pre-processing process and act as an intermediate tool between the analytical calculations and the full Finite Element models during spline design. This report covers the steps taken in order to create a standalone application with a graphical user interface that can generate meshed models of two and three component spline couplings, with all necessary boundary conditions and setup parameters, ready to be solved in the Finite Element program *Abaqus*. This report also includes a parameter study that investigates the effects different geometrical parameters have on the strength of spline couplings. Lastly the effects of manufacturing tolerances and assembling deviations are studied.

Keywords: Spline, Gear, Involute, Trochoid, FEM, MATLAB

Preface

This master thesis has been carried out as the last part of the masters program in *Applied Mechanics* at Chalmers University of Technology for the degree of *Master of Science*. The work has been performed during the spring of 2018 at the *Concept and Simulation* group under the *Driveline* department at *Volvo Group Trucks Technology*. M.Sc Sven Andersson and M.Sc Jon Elfridsson have been the supervisors at Volvo and Associate Professor Ralf Jänicke has been the examiner at Chalmers.

Acknowledgements

I would first like to thank all employees at the *Concept and simulation* group for their helpfulness and for creating a great and welcoming work environment for me to perform my thesis work. A special thanks goes to my supervisors Jon Elfridsson and Sven Aronsson for their continuous support and helpful insights throughout this project. I would also like to give a special thanks to Lennart Johansson at the *Rotating parts and housings* group for enduring all my questions regarding spline design and manufacturing techniques. Finally I would like to thank my examiner Ralf Jänicke for his support and the manager at the *Concept and Simulation* group Erik Nordlander for selecting me and giving me the opportunity to work on this master thesis project.

Anton Stenstrand
Göteborg 2018

Nomenclature

Spline specific

z	Number of teeth
m	Module
α_0	Pressure angle
x	Profile shift coefficient
D_{ii}	Minor diameter internal spline
D_{ie}	Minor diameter external spline
D_{ei}	Major diameter internal spline
D_{ee}	Major diameter external spline
D_{fi}	Form diameter internal spline
D_{fe}	Form diameter external spline
D_{base}	Base diameter
D_{pitch}	Pitch diameter
E	Space width (internal spline)
S	Tooth thickness (external spline)
E_v	Space width effective (internal spline)
S_v	Tooth thickness effective (external spline)
w_i	width internal spline
w_e	width external spline
inv ()	Involute function

General

PTO	Power Take-Off
CAD	Computer Aided Design
FEM	Finite Element Method
Abaqus	Finite element software
MATLAB	Programming language
Python	Programming language
ANSA	Pre-processing software
GUI	Graphical User Interface
Pa	Pascal
SS	Swedish Standard
SMS	Sveriges Maskinindustriförenings Standardcentral (Swedish mechanical standard)
.inc file	Include file
.inp file	Analysis input file
.txt file	Text file
.mat file	*MATLAB* MAT-file

Contents

1 Introduction

1.1 Background

This chapter gives an introduction to the machine element known as a spline coupling and gives a brief overview of its history, main function and its applications in powertrain design.

1.1.1 Spline coupling

A spline coupling is a machine element that is used in order to transmit torque between two components that are not permanently joined together. This feature makes them useful in vast number of applications and they are frequently used in the automotive industry in everything from driveshafts to clutches. Spline couplings ability to be translationally moved while rotating also makes them useful for connecting and disconnecting parts in a rotating assembly. This is utilized in powertrain engineering in for instance slip yokes and engaging sleeves. The most simple form of a torque transmitting coupling that works on the same principle as a spline coupling is a simple keyway joint, where torque is transfered between a hub and an axle trough a key. The design features a slot cut out of both the axle and the hub with a key fitted in order to limit rotational movement between the two components, as shown in Figure 1[1]. This is a joint that has been used frequently in the past and is still used today for low torque applications, usually featuring a flat-key or a Woodruff-key.

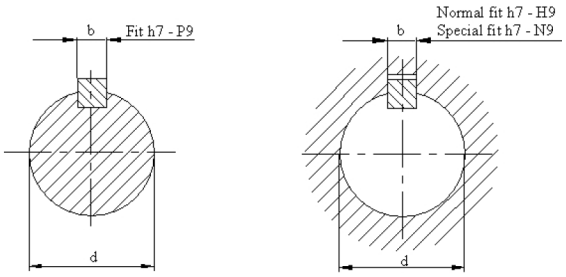


Figure 1: Keyway coupling [1]

The geometrically most simple spline is the rectangular spline where the axle have teeth of rectangular shape, much like the flat-key joint, but with the keys incorporated in the axle. The hub in its turn has the inverse geometry with broached slots on the inside. This results in higher possible torque transfer compared to a key joint design since the stress is distributed around the entire joint. These types of joints where used frequently in the past, but have since been surpassed by involute splines. The reason for this is that the rectangular shape with sharp edges results in high stress concentrations. Since the superior involute splines can be manufactured with the same machines and tools as involute gears at a lower cost there are few reasons to implement rectangular splines in industrial components. The same goes for splines with a triangular cog shape. Figure 2 shows a cross section of a rectangular spline coupling and in Figure 3 a cross section of a triangular spline coupling is shown [2][3].

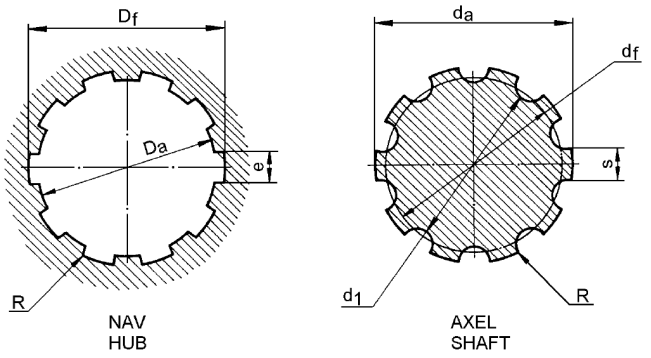


Figure 2: Spline coupling with straight sided teeth[2]

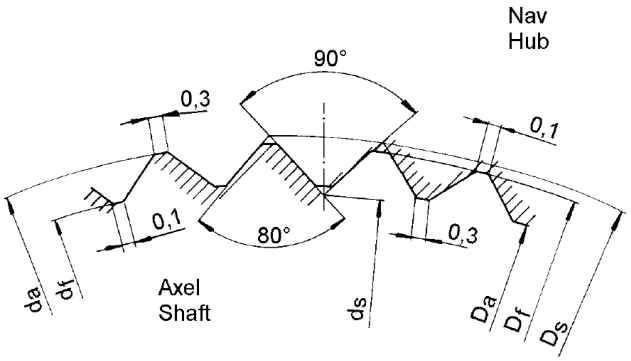


Figure 3: Spline coupling with triangular teeth[3]

The by far most used type of spline couplings today is the involute spline, which as previously mentioned has the same geometry as involute gears. This geometry, which is shown in Figure 4[4] beneath results in much lower stress concentrations and thus also results in the ability to transfer higher torque in relation to the physical size of the coupling. Furthermore the involute profile makes the coupling self align under load even if it is designed as a loose fit coupling which also benefits the torque transmitting capabilities.

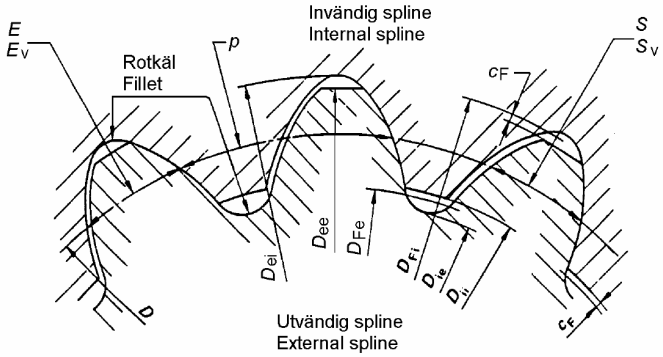


Figure 4: Involute spline coupling [4]

1.1.2 Uses in gearboxes

In most gearboxes there exists mainly two types of spline couplings. Permanent spline couplings that are constantly engaged such as the clutch disc to input shaft interface or the interface between the counter shaft and the power take off outlet (PTO). The other type are spline couplings that are disconnectable, such as the engaging sleeves that are used to change gear. Permanently engaged spline couplings are couplings between one external spline and one internal spline that is used to transfer torque between two shafts or rotating parts. The main reason for incorporating these types of couplings is to simplify production and assembly as well as allow for modularity. For instance in the case of the PTO, the manufacturing and assembly of the gearbox would be more complicated if it was permanently attached. Additionally it is a feature that is only used by some customers and therefore the spline coupling allows for gearboxes both with and without PTO outlets. Figure 5 beneath shows the principal working of a two component permanent spline coupling and and how it can be used in practice in a clutch assembly[5].

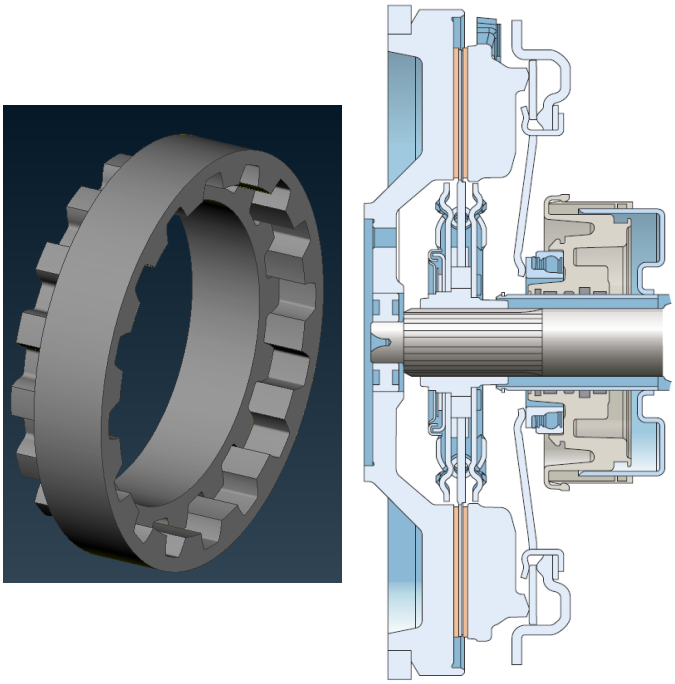


Figure 5: 2-component spline configuration[5]

The engaging sleeve spline coupling is usually a coupling between four splined parts, although a maximum of three parts are meshing simultaneously. Engaging sleeves utilizes the spline couplings ability to perform translational movement while rotating. In a gear changing mechanism there are usually three external splines and one internal spline where the internal spline can be slided in either direction in order to engage a gear, Figure 6 shows one side of such a coupling consisting of three parts. In the figure the yellow part represents a shaft to which the input torque is supplied, the red part represents the splines on the gear wheel and the green part represents the engaging sleeve. In the left Figure the engaging sleeve is disconnected and no torque is transferred to the gear wheel. By sliding the engaging sleeve over the output spline (red spline) as shown in the right Figure torque is transmitted to the output spline through the engaging sleeve. This way of transferring torque between shafts and gear wheels through spline couplings is the basic functioning of all manual constant meshing gearboxes.

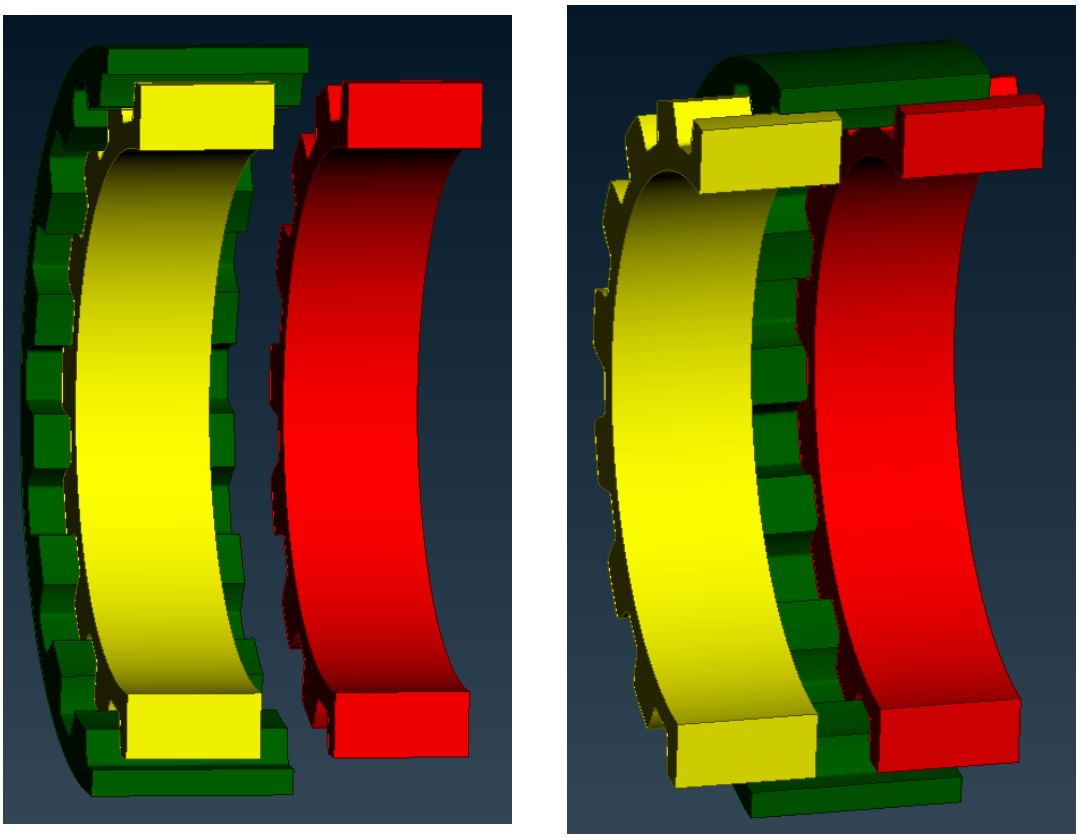


Figure 6: Principal working of an engaging sleeve

Figure 7 and 8 shows a principal sketch of a longitudinal constant meshing manual gearbox [6], where the parts in Figure 6 have been highlighted. The gearwheels on the counter shaft are solid mounted on the shaft and follows the rotation of the shaft, whilst the gearwheels on the main shaft are fitted with bearings and are free to rotate relative to the main shafts rotation. In Figure 7 the gearbox is in neutral and no torque is transmitted through the gearbox. In Figure 8 the engaging sleeve for the first and reverse gear have been slided to engage first gear and the red arrow shows the torque path through the gearbox.

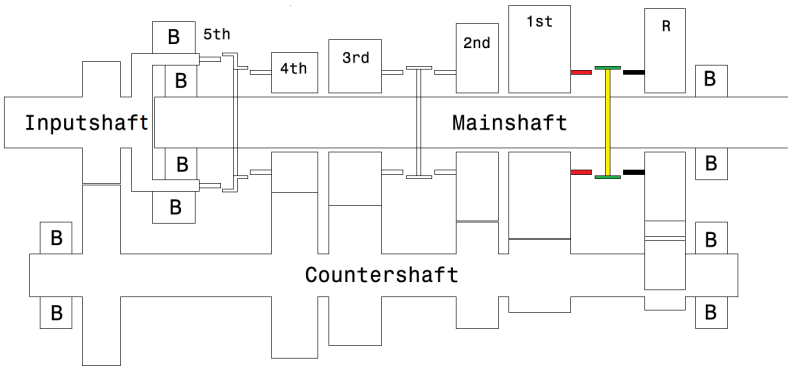


Figure 7: Gearbox in neutral

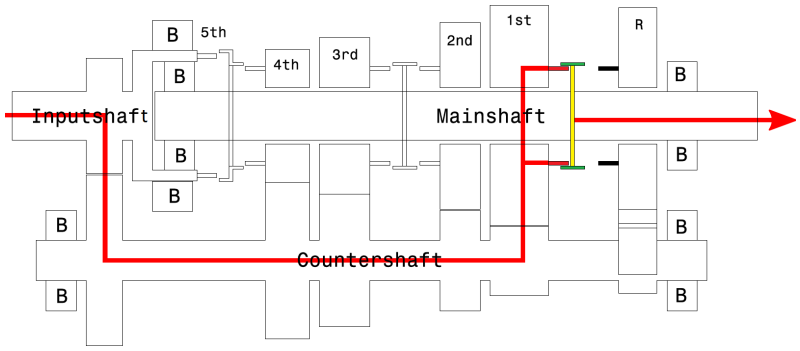


Figure 8: Gearbox in 1st gear

1.2 Problem description

Spline couplings are used for many different types of torque transmitting couplings in the powertrain of a vehicle. Currently there exists a in house developed script that can generate basic discretisized geometries of straight involute splines. The generated geometry only includes the flange surfaces i.e. the involute contact profile. This script can therefore only be used to study the stress levels at the contact surfaces of the spline, which means that areas of interest such as the root radius is not included. In order to be able to analyze the spline joint in its entirety a CAD-model has to be created and then discretisized manually. This is a very time consuming way of analyzing a spline coupling, especially if it is in an early stage of design when the scope of possible spline combinations is large.

1.3 Objective

The objective of this master thesis is to develop a program that can speed up the design process of spline couplings as well as work as a tool for further analysis of current spline configurations, in terms of quickly investigating sensitive design parameters. The program shall be well tested on a wide range of spline configurations in order to secure a stable algorithm.

1.4 Scope and limitations

The created program should generate the correct geometry of the entire spline including the root radius/trochoid based on standardized input parameters, such as module and number of teeth, for both external and internal splines. The output from the program should be a complete three dimensional meshed model with all necessary boundary conditions and contact definitions in *Abaqus* syntax ready for solving. The program should be able to handle both two and three part configurations with user defined assembly options such as overlap, misalignment and desired outputs. The program shall be able to handle this for all typical straight involute spline geometries. The project will also include a parameter study in order to identify sensitive design parameters in spline design. This also applies to manufacturing tolerances where different tolerance outcomes can be studied in order to gain further knowledge of what tolerances are the most important. The script will not be able to perform complete fatigue analysis due to the time frame of the project, but instead comparison of stress levels will be used in order to find the sensitive parameters.

2 Theory

This chapter describes the theory behind spline couplings, their uses and the mathematical description of their geometry. The chapter first gives an introduction to the basic parameters of involute spline and gear design together with the nomenclature used and lastly an explicit derivation of the mathematical functions needed is given.

2.1 General

An involute spline, both external and internal shares all of its fundamental geometry with an involute gear. The mathematical functions describing them are the same and they are manufactured with the same techniques, the only thing that usually distinguishes them from one another are the dimensions in terms of limiting circles, such as the major diameter for the external spline. The involute spline profile has a lot of advantages compared to rectangular and triangular spline couplings. The width of the profile increases from the top diameter down to the end of the involute profile and the transition from the involute profile to the bottom radius is smooth which helps lowering the stress concentrations in the root of the teeth. As mentioned previously the profile makes the coupling self align under load even if the two parts are fitted loosely together. Figure 9 shows an example of an external and an internal spline where the dimensions on the axles are in millimeters as they are for all following figures in this report unless otherwise stated.

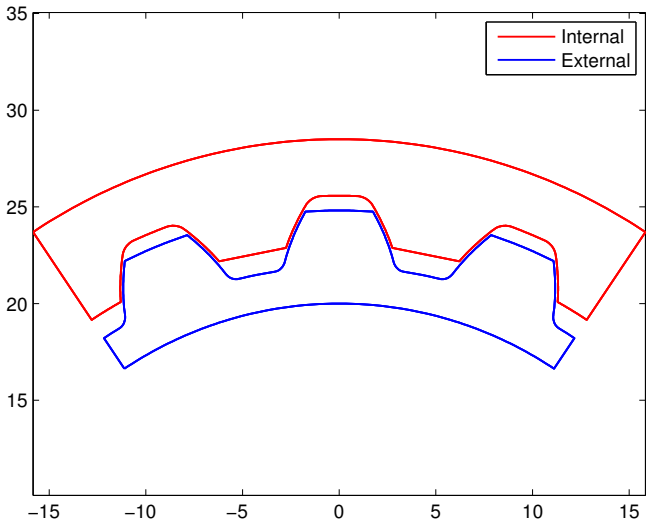


Figure 9: Internal and external spline section

2.2 Module system

The far most used system in terms of involute gear and spline design, and the only one covered in this report is the standardized module system. The module is a normalized dimension that is used to define the profile of a gear rack. Since the gear rack is a tool that can be used to manufacture gears the profile of the gear rack in itself defines the geometry of the gear. The dimensions of the gear rack profile varies depending on which standard that is used for the gear or spline design. *Volvo* has their own standardized rack profiles and tools for both gears and splines, but since they are classified and can not be shown in this report the gear rack profile according to Swedish standard (*SMS 296*) will be referred to instead. Figure 10 shows the reference profile according to *SMS 296* together with its dimensions [[7] *p.422*]. All derivations as well as the method used

is the same regardless of the gear rack profile and only the actual dimension convention of the parts differ. For gears the pressure angle, α_0 is specified at 20 degrees according to the standard, however for splines used within *Volvo* this angle can vary. In gear terminology the units in the module system are often divided into primary and secondary units, where the primary units are the units needed to describe the function of a gear transmission, for instance the speed ratio. The secondary units are the units needed in order to fully describe the geometry of the gear tooth including the root radius. These units are shown in Table 1 in Section 2.2.1 and Table 2 in Section 2.2.3. With these standard units known all other dimensions for a gear profile and thus also for a spline profile can be derived.

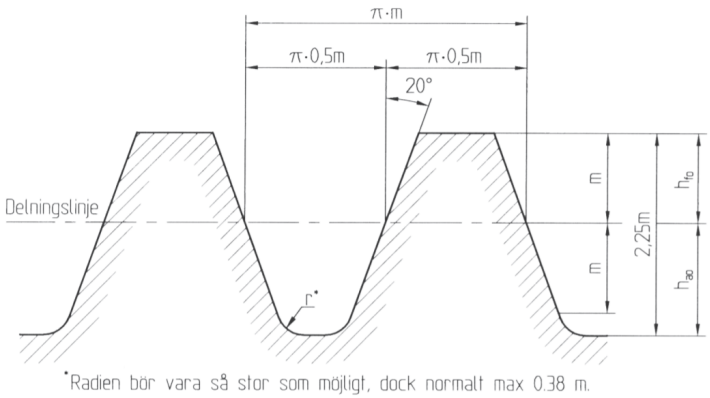


Figure 10: Reference profile according to *SMS 296* [[7] *p.422*]

2.2.1 Primary units

The primary units of the module system are shown in Table 1, as mentioned in the previous section these four units are adequate in order to fully describe the function of a gear transmission. The basis of the module system is the gear module m, as can be seen in Figure 10 it specifies the overall shape of the gear rack and in extension the gear itself. The unit of the module is in millimeters and is chosen from a set of predefined standard modules, in the Swedish standard (*SS 52*) these range from 0,5mm to 32mm. There is of course the option to use modules outside of the predefined modules, however for small volumes that can increase the cost vastly and is avoided unless necessary. As mentioned earlier the pressure angle is fixed at 20° for *SMS 296*, this effectively means that all gearwheels manufactured according to the standard will work together given the same module. The condition for this to work is that the two gear wheels have the same base pitch, which is defined according to Equation 3. As can be seen this criterion will be met if the two wheels have the same pressure angle. For splines this is of less importance since all teeth are engaged at once and the entire coupling therefore has to be designed as a unit. The standard used at *Volvo* specifies the pressure angle for splines at 30°, 37,5° and 45° with a corresponding set of modules ranging from 0,25mm to 10mm. This standard is usually followed during spline design, although deviations from this standard is quite common due to the many special applications of spline couplings used. In this case this is of limited importance from an economic perspective due to the large production volumes, moreover the majority of the tools used for both splines and gears are in *Volvos* case custom made for their own purposes. Regardless of to which extent the standard is used the mathematics and the way of describing the geometry stays the same.

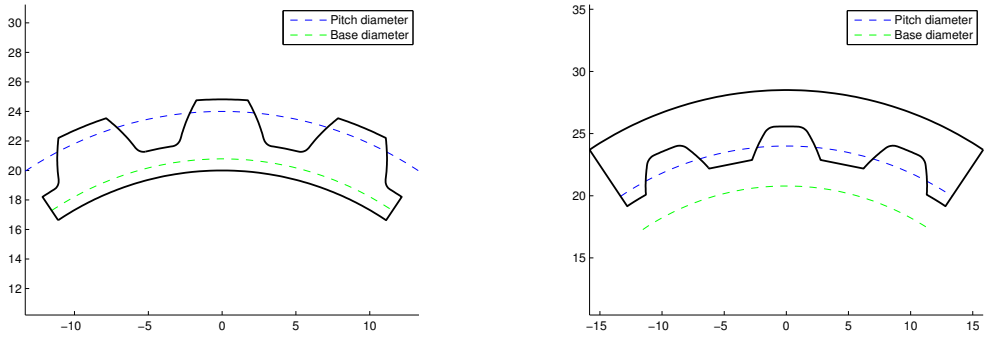
Table 1: Primary units

z	Number of teeth
m	Gear module
α_0	Pressure angle
x	Profile shift coefficient

From the *primary units* the two key diameters known as the base diameter and the pitch diameter can be derived. The base diameter is the diameter from which the involute profile is unrolled, this is explained further in Section 2.3.3. The pitch diameter is the diameter of the correct path of contact for a gearwheel, effectively the rolling diameter which for a gear transmission defines the speed ratio. The base diameter is defined according to Equation 1 and the pitch diameter is defined according to Equation 2. Figure 11 shows an example of the relation between these circles for a spline with a pressure angle, α_0, of 30°.

$$D_{base} = mz \cdot \cos(\alpha_0) \tag{1}$$

$$D_{pitch} = mz \tag{2}$$



(a) External involute spline

(b) Internal involute spline

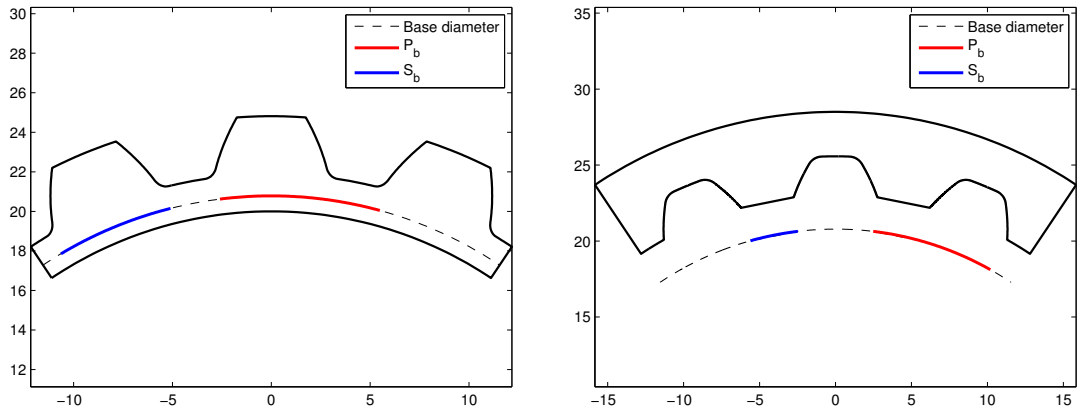
Figure 11: Base and pitch diameter

With the primary units known it is possible to derive both the pitch and the base thickness of the teeth, which are needed in order to describe the spline geometry. The pitch is defined according to Equation 3 for both external and internal splines. The base thickness is the thickness of the teeth on the base diameter and is defined according to Equation 4 for an external spline and according to Equation 5 for an internal spline [7]. These dimensions are shown in Figure 12 for both types of splines.

$$P_b = \pi m \cos(\alpha_0) \tag{3}$$

$$S_{b,ext} = \left(\frac{\pi}{2} + 2x\tan(\alpha_0) + z \cdot \text{inv}(\alpha_0)\right) m\cos(\alpha_0) \tag{4}$$

$$S_{b,int} = \left(\frac{\pi}{2} - 2x\tan(\alpha_0) - z \cdot \text{inv}(\alpha_0)\right) m\cos(\alpha_0) \tag{5}$$



(a) External involute spline

(b) Internal involute spline

Figure 12: Pitch and base thickness for involute splines

2.2.2 Profile shift

The last of the primary units is the profile shift coefficient, x, which is a dimensionless coefficient defined according to equation 6. A profile shift of a gear or spline means as the name would suggest that the profile is shifted either outwards or inwards in the radial direction. This is in practice done by shifting the gear cutting tool towards or away from the stock that is being cut. In the case of a gear rack being used for manufacturing, the profile shift is created by offsetting the distance between the gear racks pitch line and the pitch diameter of the gear as can be seen in Figure 13[8]. A positive profile shift is defined as moving the gear rack outwards radially from the spline or gear wheel and analogously moving it inwards for a negative profile shift, this is the case for both external and internal splines. Obviously a gear rack can not be used to cut internal gears and instead a circular gear shaper is usually used, but for the sake of the theory an imaginary gear rack works fine. A gear without profile shift is often called an uncorrected gear. For gears profile shifting is used in order to get the desired backlash between two gearwheels or to get the correct centre distance for multiple gear pairs of different size that are mounted on the same axle as they would in a gearbox. For splines it is commonly used in order to get the desired fit between the mating splines in a coupling. For an external spline a positive profile shift results in thicker teeth and a negative profile shift will result in thinner teeth. This can sometimes be utilized for spline wheels with a low number of teeth where a positive profile shift can be used in order to lower the stress levels in the root of the cog.

$$X = x \cdot m \quad [\text{mm}] \tag{6}$$

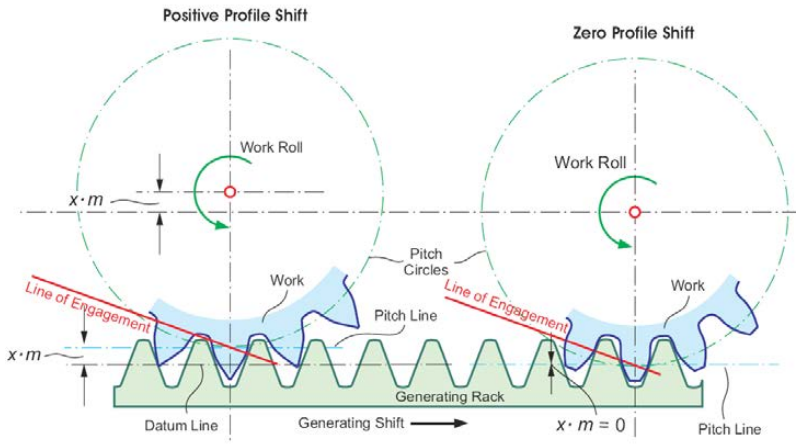


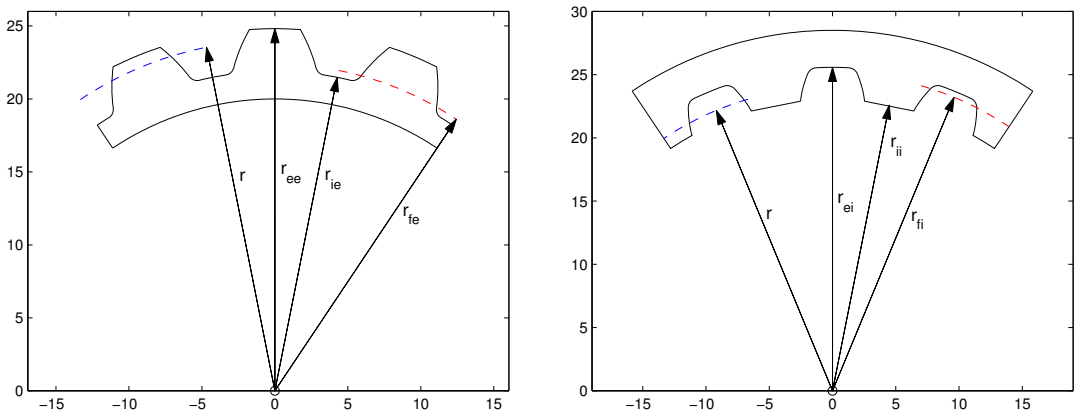
Figure 13: Profile shift using gear rack cutting[8]

2.2.3 Secondary units

The secondary units are the dimensions needed in order to describe the geometric representation of the involute spline and are listed in Table 2 as radiuses, it is also common to refer to these dimensions diametrically. For an external spline as shown in Figure 14a the major radius, r_{ee}, is the outermost radius of the spline and is machined prior to the spline cutting. The minor radius, r_{ie} is the innermost radius and is effectively the maximum cutting depth of the gear rack. For the internal spline, shown in Figure 14b, it is the other way around. The minor diameter, r_{ii}, is done prior to the gear cutting and the major diameter, r_{ei}, is the result of the maximum cutting depth. The last dimension that is needed in order to fully describe the spline profile is the form radius or form diameter that describes where the transition between the trochoid and involute function is located, which is explained further in Section 2.3. This means that the radial length of the active involute profile becomes $r_{major} - r_{form}$. This dimension can be used in order to define the geometry of the spline and thereafter derive the effective tool tip radius for the gear rack. According to the standard used at *Volvo* this is done the other way around where the form radius is a dimension derived from the effective tool tip radius. The pitch radius, r_{pitch}, is also a derived dimension and is defined diametrically according to Equation 2.

Table 2: Secondary units

	External	Internal
r_{major}	r_{ee}	r_{ei}
r_{minor}	r_{ie}	r_{ii}
r_{form}	r_{fe}	r_{fi}
r_{pitch}	r	r
$Width$	w_e	w_i



(a) External (b) Internal

Figure 14: Secondary units

2.3 Basic geometric description

2.3.1 External involute spline

The external involute spline, as shown in Figure 15 can be described by two different functions. From the start of the involute, also known as the form radius (r_{fe}), to the major radius (r_{ee}) it is described by the involute function shown in Equation 10. From the minor radius (r_{ie}) to the form radius (r_{fe}) it is described by a trochoid function, which represents the radius created by the tool tip between two teeth during gear cutting. These two functions are explained explicitly in section 2.4.

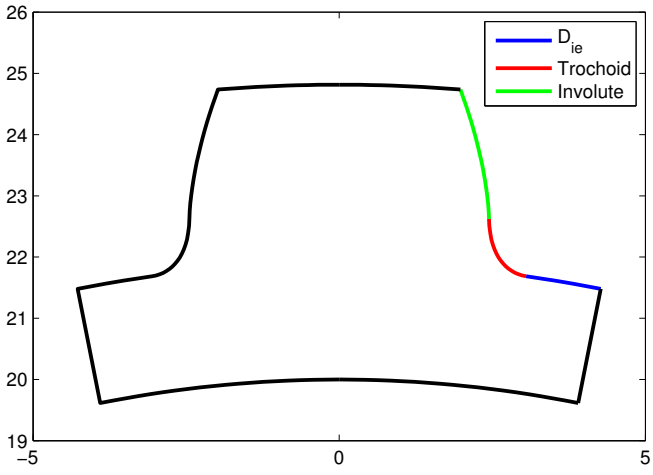


Figure 15: Buildup of an external involute spline tooth

2.3.2 Internal involute spline

An internal spline can for obvious reasons not be manufactured in the same way as an external spline. The most common way of manufacturing internal splines and gears is with a gear shaper. This means that theoretically there is a lot more design freedom when it comes to the geometry since it is effectively the inverse geometry of the tool. Additionally this manufacturing method does not result in the complex trochoid shape of the root radius. The common practice in internal involute spline design and the one practiced according to the *Volvo standard* is to have a root radius that is tangent to the the minor radius, r_{ii}, and intersects the involute profile, as can be seen in Figure 16. In technical specifications this is specified with a minimum tool tip radius and a maximum value for the form radius, r_{fi}. This effectively specifies the minimum root radius of the spline as well as the minimum length of the involute profile.

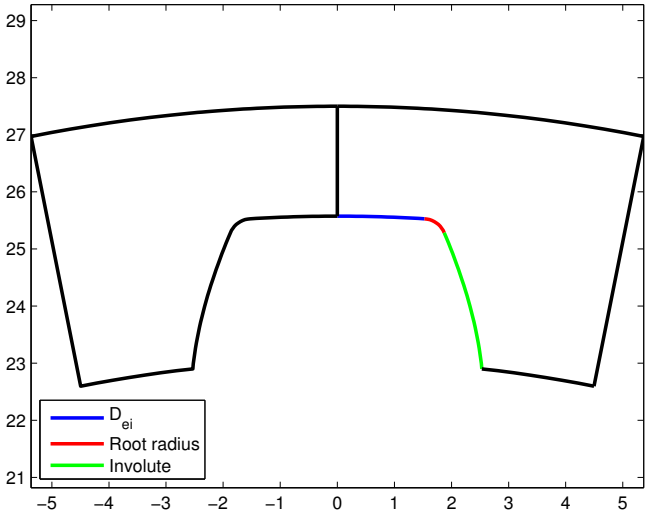


Figure 16: Buildup of an internal involute spline tooth

2.3.3 Involute profile

The involute profile can be described as a string rolled of a stationary cylinder with the string kept tensioned, the resulting path of the end of the string will result in an involute profile with the diameter of the cylinder as the base diameter. This is illustrated in Figure 17 below where the vector r_k represents the tensioned string, B the start point and P the end of the string. From Figure 17 the definition of the involute function can be derived which is done in Equation 7 through 10 according to *MÄGI*[7].

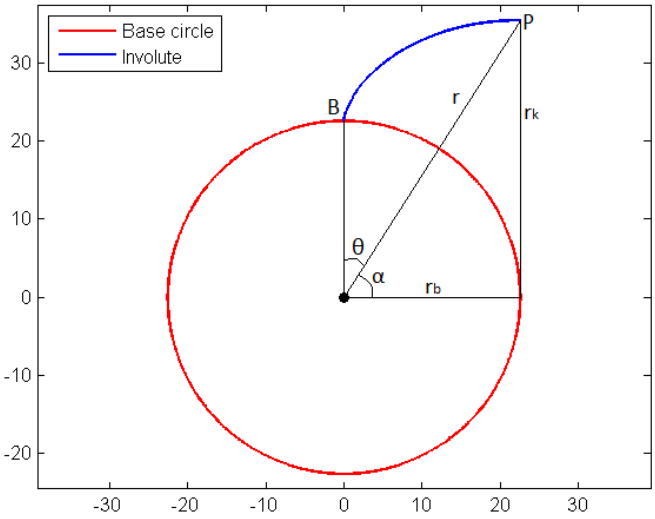


Figure 17: Involute profile rolled of the base circle

From Figure 17 the lengths of the vectors r and r_k can be derived according to Equation 7

$$\begin{aligned} r &= \frac{r_b}{\cos(\alpha)} \\ r_k &= r_b \tan(\alpha) \end{aligned} \tag{7}$$

The length of the vector r_k will be equal to the length of the involute according to the definition. r_k can therefore be expressed according to Equation 8

$$r_k = (\theta + \alpha)\, r_b \tag{8}$$

Using Equation 7 and 8 r_k can be eliminated according to Equation 9

$$\begin{aligned} &r_k = r_b \tan(\alpha) = (\theta + \alpha)\, r_b \longrightarrow \\ &\theta = \tan(\alpha) - \alpha \end{aligned} \tag{9}$$

θ in Equation 9 is called the *involute* of the angle α which is a function frequently used in gear and spline terminology and has its own function defined according to Equation 10

$$\mathrm{inv}(\alpha) = \tan(\alpha) - \alpha \tag{10}$$

2.3.4 Trochoid curve

A trochoid curve is defined as the path created from a point on a circle or connected to a circle that rolls along a straight line. In gear terminology the path that the tool tip of the cutter takes while cutting the profile of a gear tooth results in a trochoid curve between two teeth of a gear wheel. Since involute splines are manufactured in the same way as involute gears the root radius of a spline is also represented by a trochoid curve. Figure 18 shows the basics of how a trochoid curve is generated.

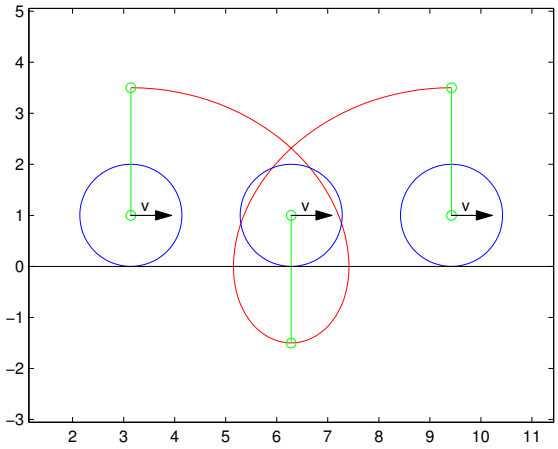


Figure 18: Trochoid curve

2.4 Mathematical geometric description

This chapter covers the derivation of the angles and distances needed in order fully describe the geometry of both the internal and external involute spline.

2.4.1 Trochoid curve for the external involute spline

In order to determine the coordinates needed for creating the trochoid curve, a set of angles and distances needs to be introduced. For the derivation of the trochoid curve of the external involute spline a modified version of *Von Ingo Maier* will be used[9]. This method is originally developed for involute gears with a helix angle, but can with some alterations be applied to straight involute splines. The sketches shown in Figure 20 and 21 are made in the transverse plane and the tool is normal to the normal plane. Because of this the tool tip in the sketches have an elliptical form as it would assuming a non zero helix angle. For a gear with a helix angle the transverse plane is orthogonal to the symmetry axis of the gearwheel, whereas the normal plane is rotated with the helix angle and thus orthogonal to the direction of the gear teeth. Figure 19 shows the definition of the two planes for a spur gear with a helix angle of 10°. For straight involute splines the helix angle is 0° and the transverse plane becomes coincident with the normal plane. In Figure 20 the trochoid profile with the angles and distances needed in order to calculate it is shown for an involute gear, Figure 21 shows the same geometry as Figure 20 but is zoomed in on the tool tip of the cutting rack. Since this derivation is done for a straight spline the geometry of the tool tip will in reality be a circle and not an ellipse as shown in Figure 20 and 21.

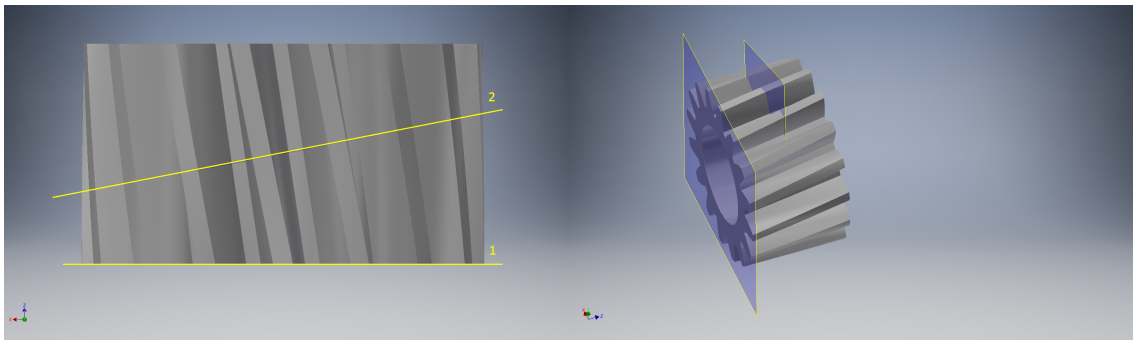


Figure 19: Normal (2) and transverse (1) plane for a spur gear with a helix angle

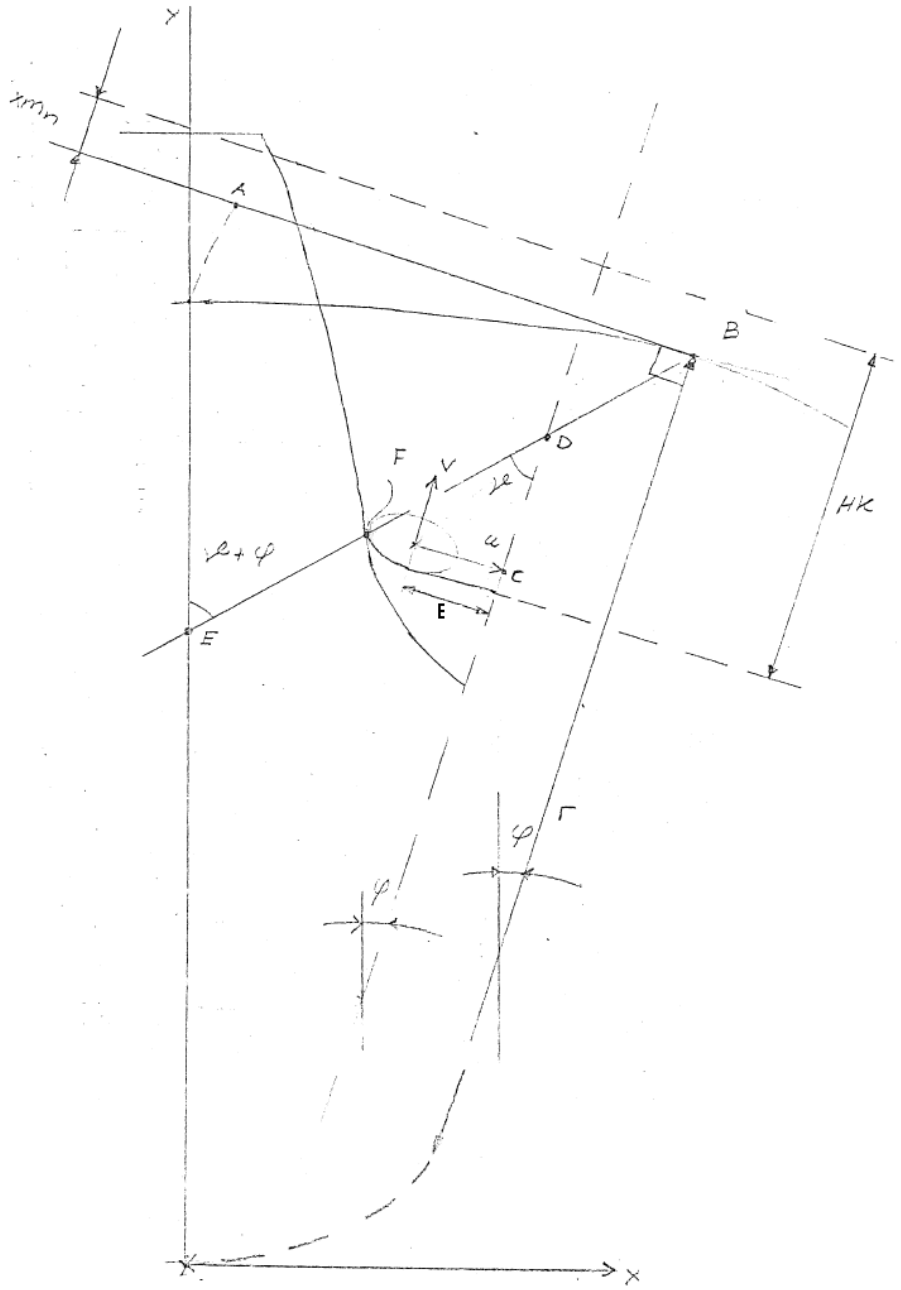


Figure 20: Trochoid curve

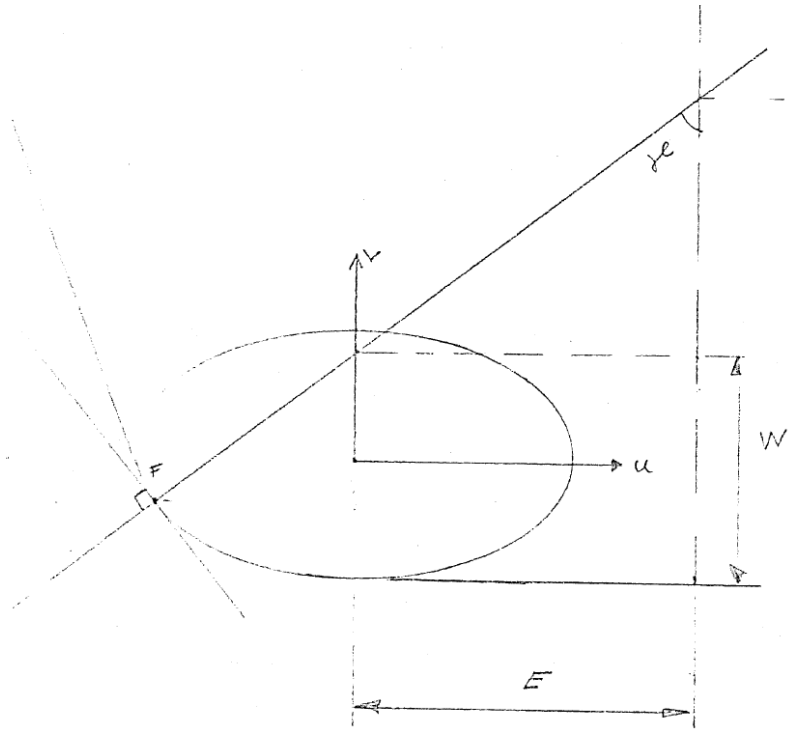


Figure 21: Tool tip

Equations 11 through 21 shows the necessary steps in order to calculate the curve for a straight involute spline. Equations 22 and 23 describes the geometry between the two trochoids in case of a specified radius smaller than the distance between two teeth at the form radius r_{fe}.

First the tool height h_k is calculated according to equation 11

$$h_k = r + x \cdot m - r_{ie} \tag{11}$$

Where:

$$\begin{aligned} r &= \text{Pitch radius} \\ x &= \text{Profile shift coefficient} \\ m &= \text{Gear module} \\ r_{ie} &= \text{Minor radius} \end{aligned} \tag{12}$$

Now the distance from the center of the trochoid to the center of the tool can be calculated

$$E = \frac{\pi \cdot m}{4} - h_k \cdot \tan(\alpha_0) - \frac{R}{cos(\alpha_0)} \left(1 - \sin(\alpha_0)\right) \tag{13}$$

Where:

$$\begin{aligned} \alpha_0 &= \text{Preassure angle} \\ R &= \text{Tool tip radius} \end{aligned} \tag{14}$$

The coordinate u in the local coordinate system showed in Figure 21 is defined according to equation 15. This is used as the governing coordinate in order to create points on the trochoid curve in the program.

$$-R\cos(\alpha_0) \leq u < 0 \tag{15}$$

In the normal plane and without a helix angle the tool tip is a circle, which means that the coordinate v can be solved according to Equation 16. v becomes negative since the point of engagement will always be in the third or fourth quadrant.

$$v^2 + u^2 = R^2 \rightarrow v = -\sqrt{R^2 - u^2} \tag{16}$$

Now the angle ν, which is the angle between the symmetry line of the tool and the vector stretching from the roll point to the engagement point as can be seen in Figure 21, can be calculated.

$$\begin{aligned} \tan(\nu) &= \left|\frac{dv}{du}\right| = \frac{|u|}{\sqrt{R^2 - u^2}} \\ \nu &= arc\tan\left(\frac{|u|}{\sqrt{R^2 - u^2}}\right) \end{aligned} \tag{17}$$

The distance from the bottom of the tool tip to the point where the vector $\overrightarrow{BF}$ intersects the v-axis is calculated according to equation 18

$$W = (R + v) - u \cdot \cot(\nu) \tag{18}$$

With W known the roll angle ϕ can now be determined.

$$\phi = \frac{1}{r}\left(\frac{\pi \cdot m}{2} - E + (h_k - x \cdot m - W)\tan(\nu)\right) \tag{19}$$

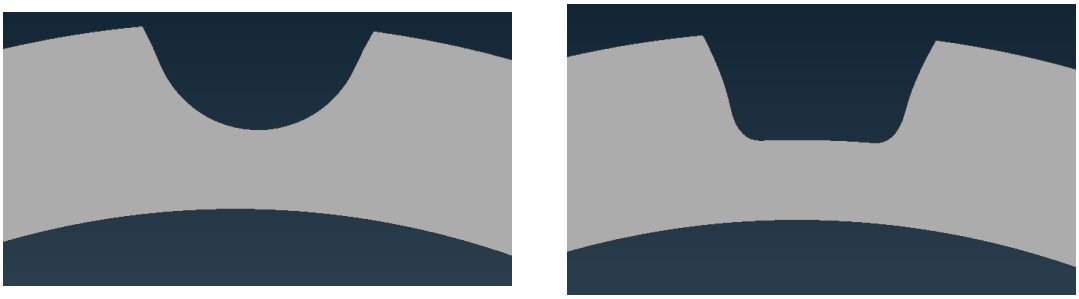
The distance from the roll point to the point of engagement described by the vector $\overrightarrow{BF}$ is given by equation 20

$$\overrightarrow{BF} = \frac{h_k - x \cdot m - W}{\cos(\nu)} - \frac{u}{\sin(\nu)} \tag{20}$$

The coordinates of the trochoid curve in the global Cartesian coordinate system can now be calculated according to Equation 21

$$\begin{aligned} x &= r \cdot \sin(\phi) - \overrightarrow{BF} \cdot \sin(\phi + \nu) \\ y &= r \cdot \cos(\phi) - \overrightarrow{BF} \cdot \cos(\phi + \nu) \end{aligned} \tag{21}$$

If the spline is not designed with a full radius an additional curve needs to be calculated for the part between the two trochoids. Figure 22 shows the difference between a spline with a full radius and a spline designed with smaller tool tip radius resulting in a flat part between the trochoids of two teeth.



(a) Full radius

(b) Non full radius spline root

Figure 22: Different types of spline roots

Figure 23 shows the introduced angles needed in order to define the geometry for this part of the root.

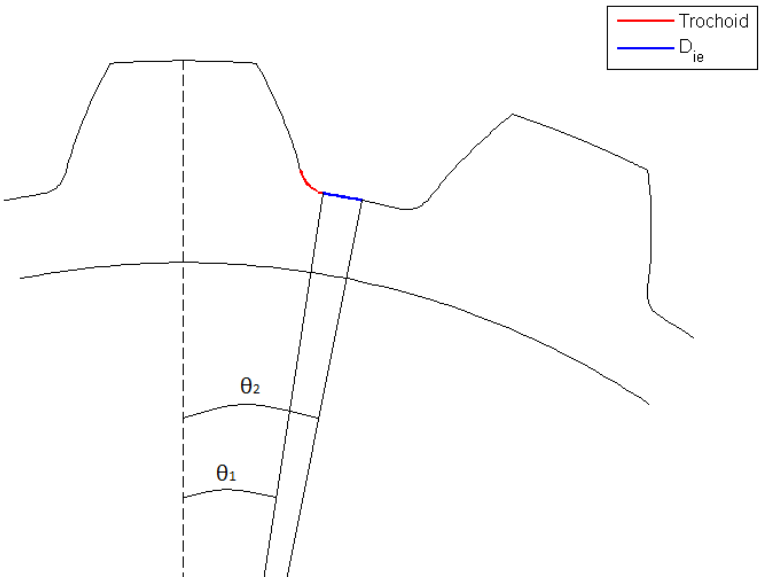


Figure 23: Flat part between the trochoids

The angle θ_1 becomes the same as the angle ϕ for $u \to 0$ and the two angles angles defining the flat part of the root are shown in Equation 22

$$\begin{aligned} \theta_1 &= \phi \qquad u \to 0 \\ \theta_2 &= \frac{2 \cdot \pi}{z \cdot 2} = \frac{\pi}{z} \end{aligned} \tag{22}$$

The Cartesian coordinates in the global system for the part between the two trochoids can then be calculated according to equation 23. Where $\theta_1 \leq \theta \leq \theta_2$.

$$\begin{aligned} x_f &= r_f \cdot \sin(\theta) \\ y_f &= r_f \cdot \cos(\theta) \end{aligned} \tag{23}$$

2.4.2 Involute profile for the external involute spline

When the coordinates for the trochoid curve is known then so is the form radius, r_{fe}, defining the start point of the involute curve. The outer radius,r_{ee}, is a secondary unit and thus also specified and therefore the radius span for the entire involute profile is known. What remains to be determined is the corresponding angle (ϕ) in the polar coordinate system.

The thickness for any given radius (S_r) on the involute profile is defined according to Equation 24[7].

$$S_r = \left(\frac{pi + 4x \tan(\alpha_0)}{2z} + \text{inv}(\alpha_0) - \arccos\left(\frac{r_b}{r_r}\right) \right) 2r_r \tag{24}$$

The angle can then be calculated according to equation 25

$$\phi_n = \frac{S_r}{2 \cdot r} \tag{25}$$

2.4.3 Root radius for the internal involute spline

The first step in creating the geometry for the internal spline is to create the root radius. This is necessary because the size of the bottom radius defines the form radius r_{fi}, which in itself defines the starting point of the involute profile as well as the starting point of the bottom part of the spline (r_f). Figure 24 shows a sketch of the root radius (red) and involute profile with the lengths and angles needed in order to determine its coordinates.

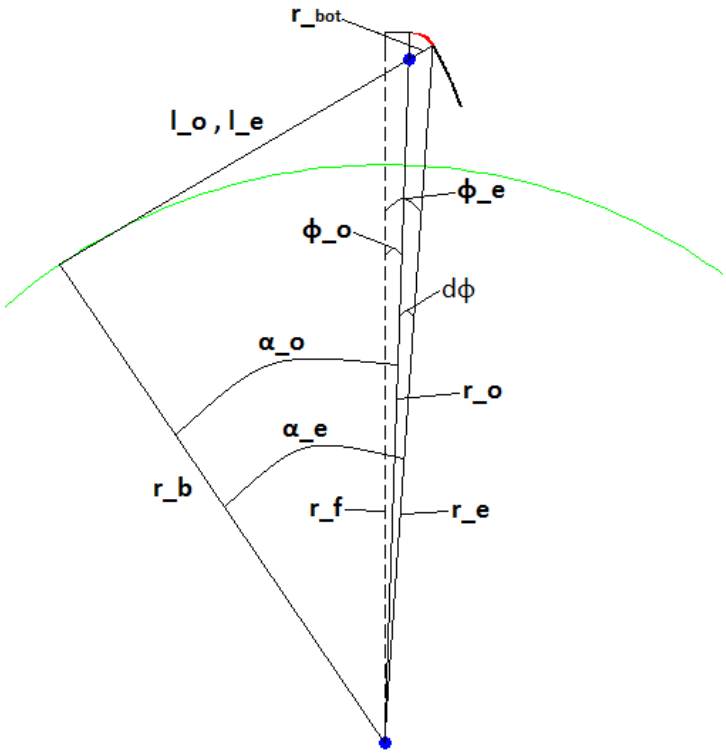


Figure 24: Root radius for the internal involute spline

The radial coordinate of the origin of the root radius is calculated according to equation 26

$$r_o = r_f - r_{bot} \tag{26}$$

The roll distance to the origin of the bottom radius, l_o, can now be calculated according to equation 27

$$l_o = \sqrt{r_o^2 - r_b^2} \tag{27}$$

The roll distance to the start of the involute, l_e, then becomes

$$l_e = l_o + r_{bot} \tag{28}$$

The radial coordinate of the form radius can now be calculated according to equation 29

$$r_e = \sqrt{l_e^2 + r_b^2} \tag{29}$$

The angles α_e and α_o then becomes:

$$\begin{aligned} \alpha_e &= arctan(\frac{r_b}{r_e}) \\ \alpha_o &= arctan(\frac{r_b}{r_o}) \end{aligned} \tag{30}$$

The angle $d\phi$ can be calculated as:

$$d\phi = \alpha_e - \alpha_o \tag{31}$$

The thickness at the form radius is given by equation 32[7]

$$S_e = \left(\frac{\pi + 4 * x * tan(\alpha_{po})}{2 \cdot z} + inv(\alpha_{po}) - inv\left(arccos\left(\frac{r_b}{r_e} \right) \right) \right) 2r_e \tag{32}$$

Where r_b is the base circle radius defined as[7]:

$$r_b = \left(\frac{m \cdot z}{2} \right) \cdot cos(\alpha_{po}) \tag{33}$$

The angle ϕ_e can now be calculated according to equation 34

$$\phi_e = \frac{S_e}{2r_e} \tag{34}$$

Which gives the angle ϕ_o as:

$$\phi_o = \phi_e - d\phi \tag{35}$$

With both the angle and radius for the three points (o,e and f) known they can know be converted into cartesian coordinates according to equation 36 to simplify the creation of the bottom radius.

$$
\begin{aligned}
x_e &= r_e \cdot sin(\phi_e) \\
y_e &= r_e \cdot cos(\phi_e) \\
x_o &= r_o \cdot sin(\phi_o) \\
y_o &= r_o \cdot cos(\phi_o) \\
x_f &= r_f \cdot sin(\phi_f) \\
y_f &= r_f \cdot cos(\phi_f)
\end{aligned}
\tag{36}
$$

Figure 25 shows the root radius zoomed in where the angles γ_e and γ_f have been introduced as well as the sweep angle for the root radius γ_{ef}.

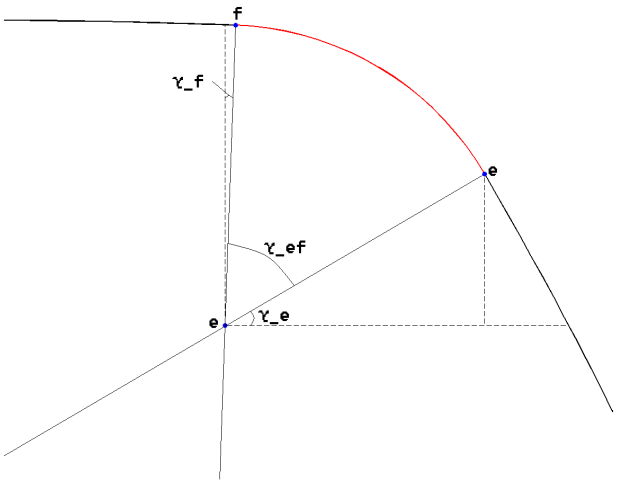


Figure 25: Root radius for the internal involute spline

Since the coordinates of all three points (o,e and f) are known the angles γ_e and γ_f can be calculated according to Equation 37.

$$
\begin{aligned}
\gamma_e &= arctan\left(\frac{(y_e - y_o)}{(x_e - x_o)}\right) \\
\gamma_f &= arctan\left(\frac{(x_f - x_o)}{(y_f - y_o)}\right)
\end{aligned}
\tag{37}
$$

The sweep angle for the bottom radius can now be defined according to Equation 38.

$$
\gamma_e \leq \gamma_{ef} \leq \frac{\pi}{2} - \gamma_f \tag{38}
$$

In a numerical enviroment the root radius will consist of a number of coordinates along the radius. These coordinates can now easily be created since the root radius r_{bot} and the sweep angle γ_{ef} are known. The coordinates will however be defined in the local coordinate system with its origin coinciding with the origin of the root radius and needs to be transformed to the global polar coordinate system. Let $\mathbf{r_{bot}}$ and γ_{ef} be vectors containing the coordinates for the root radius, then the coordinates in the global Cartesian coordinate system can be expressed according to Equation 39.

$$\begin{aligned} \mathbf{X}_{bot} &= x_o + \mathbf{r_{bot}} \cdot cos(\underline{\gamma_{ef}}) \\ \mathbf{Y}_{bot} &= y_o + \mathbf{r_{bot}} \cdot sin(\underline{\gamma_{ef}}) \end{aligned} \tag{39}$$

The global carteesian coordinates can then be transformed into global polar coordinates according to equation 40.

$$\begin{aligned} \mathbf{r_{botr}} &= \sqrt{\mathbf{X}_{bot}^2 + \mathbf{Y}_{bot}^2} \\ \underline{\phi_{botr}} &= arctan\left(\frac{\mathbf{X}_{bot}}{\mathbf{Y}_{bot}}\right) \end{aligned} \tag{40}$$

2.4.4 Involute profile for the internal involute spline

With the root radius calculated the start point of the involute profile at the form radius, r_{fi}, is now known as well as the end point at the major radius, r_{ei}. The remaining part is to determine the angle in the polar coordinate system ϕ for any given radius on the profile.

The thickness for a given radius on the involute profile is given by Equation 41[7].

$$S_r = \left(\frac{\pi + 4 \cdot x \cdot tan(\alpha_{po})}{2 \cdot z} + inv(\alpha_{po}) - inv\left(arccos\left(\frac{r_b}{r}\right)\right)\right) 2r \tag{41}$$

With the thickness known the corresponding angle ϕ in the global polar coordinate system can be calculated according to equation 42

$$\phi_{inv} = \frac{S_r}{2 \cdot r} \tag{42}$$

2.5 Contact mechanics

Contact mechanics in the finite element method becomes relevant whenever two different bodies interact with each other through a contact constraint. The contact could be between a deformable and rigid body or more commonly as in this case between two deformable bodies. This type of simulations adds more requirements to the simulation setup and are also much more demanding from a computational point of view compared to a single body analysis. This is due to the fact that the system of equations become nonlinear if contact is included even if dealing with small strain linear elasticity and therefore an iterative solver needs to be used in order to solve it.

2.5.1 Methods

When applying contact theory for the finite element method there are two main methods that are used, the *penalty method* and the *Lagrange multiplier method*. Common for both these methods is that the solver needs to keep track of which nodes that comes in contact with each other when the load is applied. In some less determined simulations where its hard to predict what surfaces will come in contact during the simulation this is done with a global or semi global node search where large parts of the geometry needs to be checked for clashes like for instance in crash simulations. The method of searching the entire domain for contact between every loadstep would theoretically work fine for any contact simulation, but since it can be very computationally demanding it is much more efficient to predefine the contact surfaces if possible. In the case of a spline coupling this becomes a non issue since the contact surfaces of the spline is defined by its geometry and can easily be predefined in the pre-processing step. Figure 26 shows a simple 1-D contact mechanics problem taken from *Ekh 2015* [10]. In this case the solver needs to keep track of the gap between nodes 3 and 4 with the initial gap g_0 for each loadstep and when the two nodes come into contact with each other the iterative solving of the system can begin.

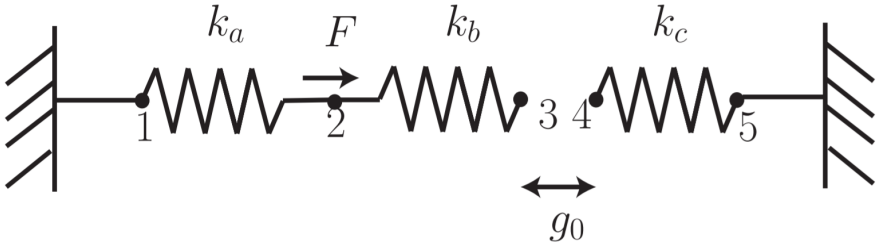


Figure 26: 1-D contact mechanics problem [10]

Penalty method
The essential workings of the penalty method is that a stiffness is added to to the stiffness matrix for the nodes that clash together with the corresponding force added to the internal force vector. This can be visualized as spring added between two nodes of the parts which results in a force pushing them apart. For the system shown in Figure 26 the program would enter the iterative contact solver of the program when the absolute distance between node 3 and 4 becomes less than 0, i.e penetration occurs. The resulting system of equations then becomes according to Equation 43[10].

$$\begin{bmatrix} K_a + K_b & -K_b & 0 \\ -K_b & K_b + \epsilon_N & -\epsilon_N \\ 0 & -\epsilon_N & K_c + \epsilon_N \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F \\ \epsilon_N g_0 \\ -\epsilon_N g_0 \end{bmatrix} \tag{43}$$

Lagrange multiplier method
The Lagrange multiplier method is based on adding an external force to the force vector for the nodes that comes into contact. This force known as a *Lagrange multiplier* becomes an unknown in the system of equations and therefore a constraint equation needs to be added in order to make the system solvable. For the 1-D problem shown in Figure 26 assuming contact between nodes 3 and 4 have occurred the system of equations for the problem becomes according to Equation 44[10].

$$\begin{bmatrix} K_a + K_b & -K_b & 0 & \\ -K_b & K_b & 0 & -1 \\ 0 & 0 & K_c & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ \lambda \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \\ -g_0 \end{bmatrix} \tag{44}$$

Differences
For the penalty method the resulting contact force is a function of the penalty stiffness and the penetration and thus there will be a small resulting penetration when the tolerance for equilibrium is reached by the solver. The only way of obtaining zero penetration for the penalty method is if the penalty stiffness goes towards infinity which in itself will result in an ill conditioned stiffness matrix and renders the problem unsolvable by numeric methods. This is the main disadvantage of the penalty method. The main advantage of the penalty method is that the size of the problem remains constant regardless of the amount of nodes in contact as can be seen in Equation 43. For the *Lagrange multiplier* method the system of equations needs to be redefined every time the number of nodes in contact changes as can be seen in Equation 44. On the other hand the *Lagrange multiplier* method results by definition in a gap of 0 between the contact nodes. This method will therefore result in a more accurate solution compared to the *penalty* method, but the *penalty* method is less computationally demanding since the number of equations are kept constant.

2.5.2 Contact mechanics in *Abaqus*

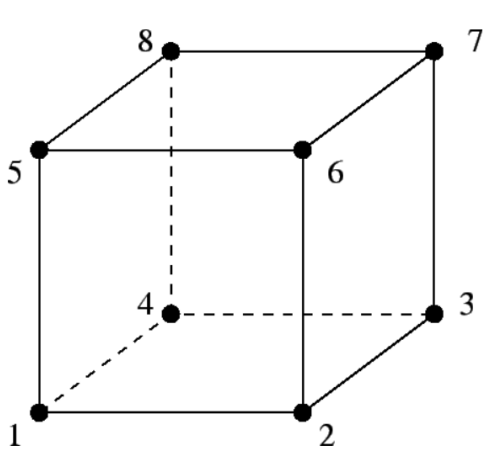
In *Abaqus* there are several options when it comes to contact formulations, both in terms of solving method and surface interactions. The models created by the *Optimization tool* have the *Penalty method* as the default solving technique. The main reason for this is that some of the features included in the program are only compatible with the *Penalty method* definition in the *Abaqus* version used at *Volvo*. The contact formulation used in the *Optimization tool* is a finite sliding surface-to-surface formulation. The surface-to-surface option means that the contact surfaces are defined as surface-sets instead of node-sets which makes it possible for *Abaqus* to interpolate between the nodes of both the master and the slave surface. For the example shown in Figure 26 the geometrical formulation of the gap function is simple since its 1-dimensional, but for two 2-dimensional surfaces with different mesh refinement grade the gap function becomes more complex. By using the surface-to-surface definition instead of the regular node-to-node definition *Abaqus* is able to establish a more exact gap function through interpolation which results in a more accurate solution. The trade-off for this increased accuracy is the computational effort which means that it increases the solving time.

3 Method

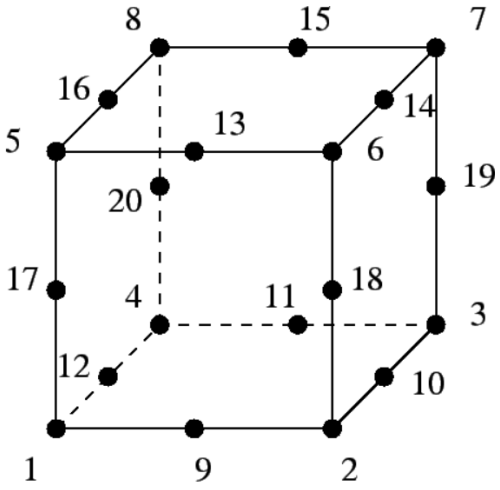
In this chapter the theory and underlying structure of the program will be explained from defining the basic measurements of a single spline to a complete discretisized model.

3.1 Meshing strategy

The element type used for the discretization of the model is hexhedral elements, also known as brick elements. The choice of this element type was a demand from *Volvo*. The program is written in such a way that both 8-node and 20-node hex elements can be used in order to allow for flexibility with respect to precision and computational effort, these element types are shown in Figure 27 beneath. Since the geometry of a spline includes radiuses and to some extent relatively steep derivatives in its shape this element type creates some challenges in the discretization.



(a) 8-node hex element (C3D8)



(b) 20-node hex element (C3D20)

Figure 27: Element types used for discretization

In order to create a meshing algorithm that provides as good quality mesh as possible while at the same time being possible to implement in code for all different types of splines and inputs a study of different mesh schemes was made in the pre-processing program *ANSA*. A symmetry model of two geometrically different splines were created where different layouts could be tested quickly and effectively. After experimenting with different layouts and consulting with the supervisors at *Volvo* a promising meshing structure was obtained which can be seen in Figure 28. Figure 28 only shows the external spline, but since the internal spline is the inverse of the external spline it is sufficient in order to determine the mesh structure for both parts.

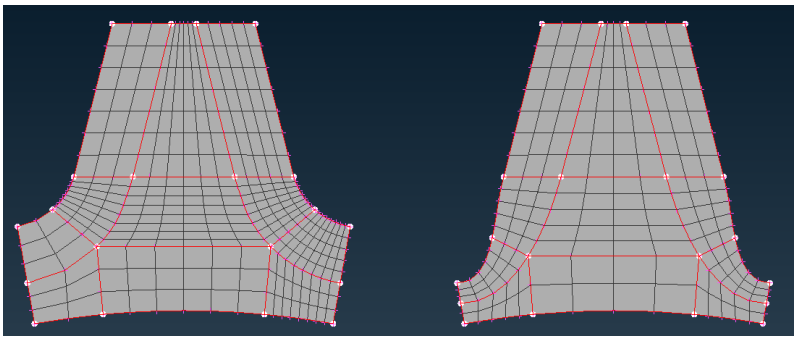


Figure 28: Mesh structure

The greatest advantage of this mesh structure is that it provides high quality elements along the contact surfaces of the splines, as well as in the root radius which is advantageous since this is the area of most interest and with the largest discontinuities in the geometry. This layout will however result in some skewed elements especially between the sections around the projected root radius. This skewness of course varies with the geometry input parameters of the spline and one of the big challenges is to minimize the amount of skewed elements for a wide range of different spline geometries. Some bad elements are however unavoidable due to the basic geometry of a spline coupling and the shape of the elements used. The major reasoning behind this mesh structure is to make sure that these elements are located where they have a small impact on the discretization in its entirety.

3.2 Geometry

This section describes the creation of the complete geometry of both the internal and external involute spline. The input data in terms of both geometrical data and mesh settings are used in order create points in space representing the node coordinates. Since the spline is rotationally symmetric it is enough to create the geometry for half a tooth, which can then be used to create the entire spline later in the mesh algorithm.

3.2.1 External geometry

As mentioned earlier the node coordinates only have to be created for half of a tooth in two dimensions. The outer geometry is created based on the mathematical description explained in Section 2.4, this geometry is then divided into three different sections in order to imitate the mesh strategy shown in Figure 28. Figure 29 shows the geometry divided into the three different mesh sections. The size and position of these mesh sections are defined automatically based on the input, but can also be altered manually in order to allow for flexibility.

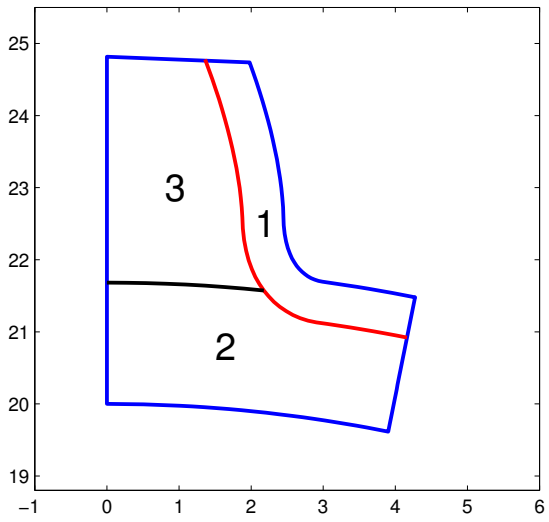


Figure 29: Mesh sections external spline

Section one consists of the active involute profile together with the root of the tooth, which in itself consists of one or two parts depending on if the spline has a full radius or not. If the spline has a full radius the root is described solely by the trochoid function and if it is not it also has a flat part between the trochoids of two teeth with the diameter D_{ie}. The different parts of the profile and root is shown in Figure 30.

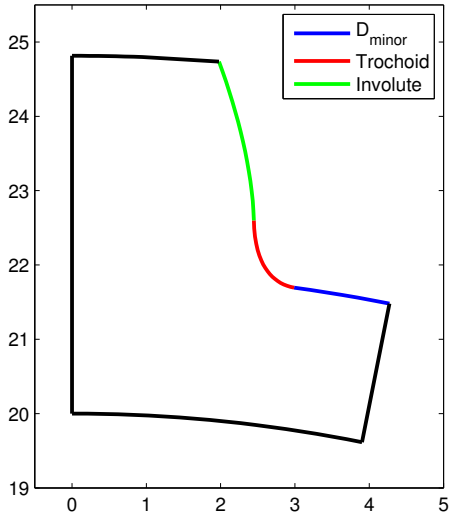


Figure 30: Profile and root sections for an external spline

The number of node points that are created along this profile is decided by user input. The user can either specify a total number of elements along the entire profile and the program automatically distributes them in order to obtain an equal element side length. It is also possible to specify the number of elements along the involute profile and the root and let the program distribute the number of elements for the two root sections or specify the number of elements for each section manually. The size of the section radially, i.e the position of the projected profile (red line in Figure 29) is determined by user input and is in the program created by adjusting the distance between the roll point and the cutting point, the vector $\overrightarrow{BF}$ in Figure 20, in the trochoid function and projecting the involute in the circumferential direction. The number of node points created in radial direction is specified by user input determining the number of elements radially for section 1.

The rest of the geometry is divided into section 2 and 3 by the black line in Figure 29, this line is a radial projection of the inner diameter and completes the outer boundary for both section 2 and 3. The nodal points for section 3 and thus also the resolution of the mesh are in the radial direction decided by the number of elements chosen for section 1. In the circumferential direction a new variable is introduced which sets the number of elements in this direction. The nodal points are in this case linearly spaced between the projected profile (red line in Figure 30) and the left boundary. This parameter then also sets the number of elements in the circumferential direction for the left part of section 2 (green part in Figure 31), whereas the right part of section 2 is set by the mesh resolution in section 1. For the radial direction a new variable is introduced which sets the number of projected nodal lines between the upper and lower boundary. Figure 31 shows the different mesh sections, their common node boundaries and the input variables that define the number of node coordinates along each section. In Figure 32 a basic coarse mesh for the symmetry part of the external spline is shown with the input parameters, as shown in Figure 31, set to the values displayed in Table 3.

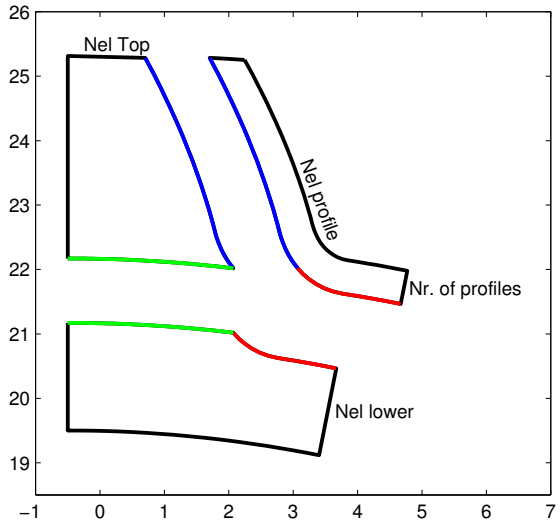


Figure 31: Meshing sections with input parameters for the external spline

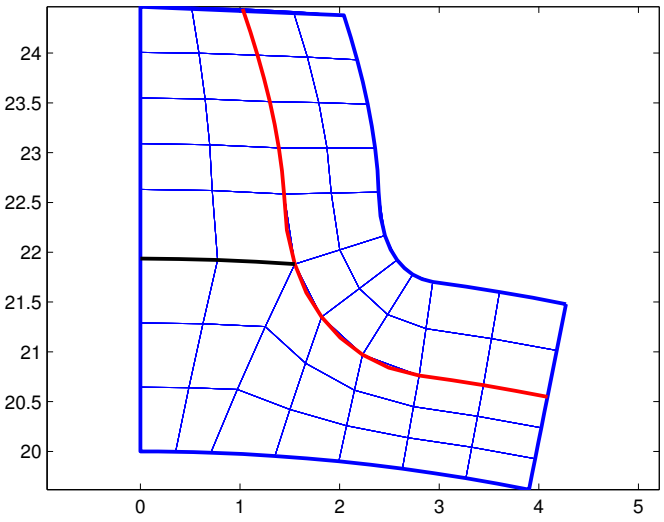


Figure 32: Basic mesh for symmetry part of external involute spline defined by Table 3

Table 3: Mesh input parameters for Figure 32

Nel profile	10
Nr. of profiles	2
Nel top	2
Nel lower	3

3.2.2 Internal geometry

The creation of the discretisized geometry for the internal involute spline is very similar to the external spline. The internal spline is as the external spline rotationally symmetric and therefore it is in the same way as for the external spline only necessary to create the geometry for half of a tooth explicitly. The mathematical description on how to derive the outer geometry of the internal involute spline is explained in Section 2.4. In the same way as for the external spline the internal spline has been divided into three different mesh sections according to Figure 33 in order to mimic the mesh mesh strategy set out in Section 3.1. The dividing line that makes up section 1 (red line in Figure 33) is a projection of the active profile consisting of the involute, the root radius and the major diameter (blue line in Figure 33) inwards. This is done by numerically differentiating the profile in order to obtain the unit normal orthogonal to the profile that specifies the projection direction. The line that splits up the remaining geometry into section 2 and 3 (black line in Figure 33) is simply a line that is drawn from a point on the dividing profile (red line in Figure 33) to a point on the outer diameter, the positioning and angle of this line is then altered in order to minimize the skweness of the elements in the intersecting boundary. The theory and application of this is explained further in Section 3.4.

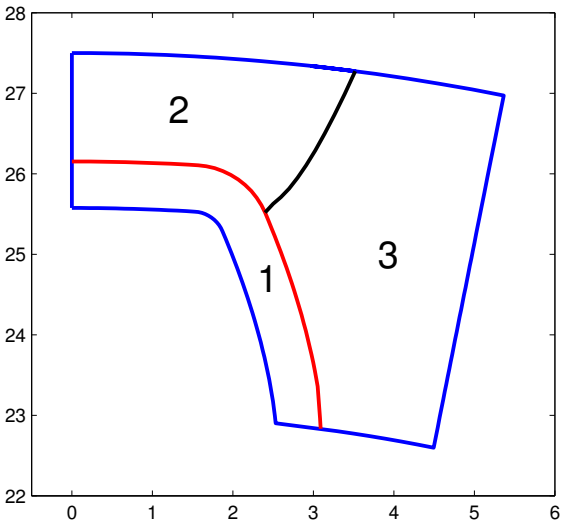


Figure 33: Mesh sections internal spline

As is the case for the external spline there is a set of input variables that sets the number of node coordinates both radially and circumferentially for each section. Section 1 has two variables that define these points. Section 2 has one variable that defines the number of points circumferentially and the number of points in the radial direction becomes specified by the input for section 1 due to their shared boundary. Section 3 has a variable that specifies the number of points circumferentially whilst the number of points radially are determined by the inputs for section 1 and 2. The input parameters determining the number of node coordinates for each mesh section are shown in Figure 34 where the boundaries with shared node coordinates have been highlighted. Figure 35 shows an example mesh of the internal involute spline with the input parameters defined according to Table 4.

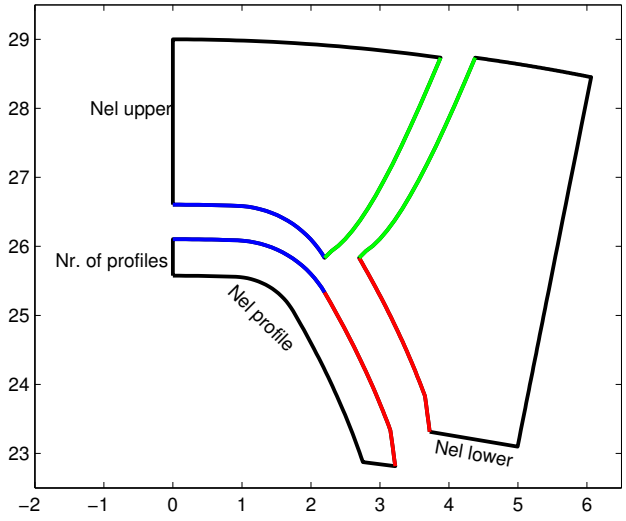


Figure 34: Meshing sections with input parameters for the internal spline

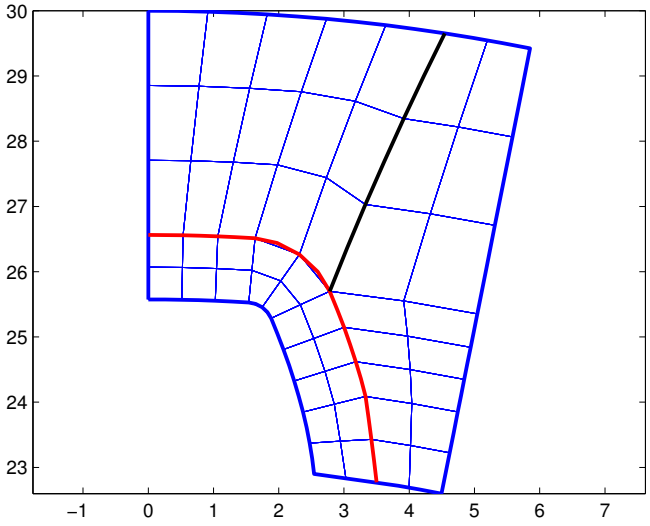


Figure 35: Basic mesh for symmetry part of internal involute spline defined by Table 4

Table 4: Mesh input parameters for Figure 35

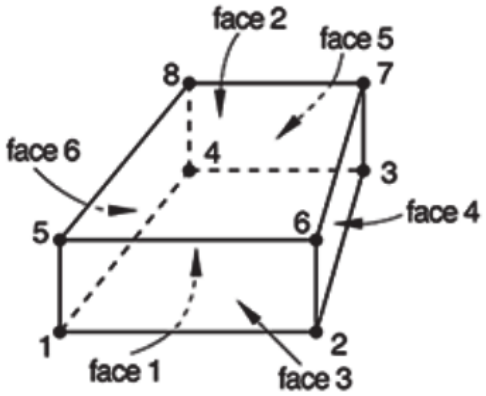
Nel profile	10
Nr. of profiles	2
Nel right	2
Nel upper	3

3.3 Finite Element discretization

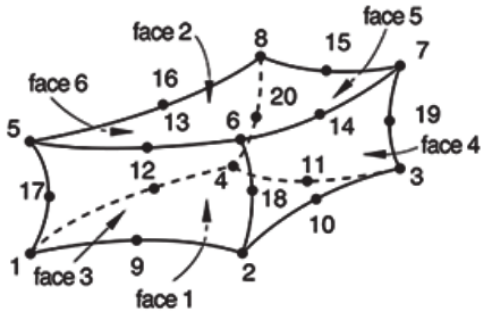
This chapter covers the method used in order to create a meshed 3 dimensional model of the spline coupling, from discretisizing the 2 dimensional mesh sections of the symmetric part of the spline shown in Section 3.2 to assembling it into a complete 3 dimensional model. As can be seen in Section 3.2 the creation of the node coordinates are done very similiarly for the internal and external spline, therefore the meshing technique is very similar for the two different parts. Because of this the meshing technique will only be shown for the external spline in order to avoid repetition. This chapter also includes the creation of the node and element sets that are needed in order to solve the Finite Element problem, as well as the sets that are needed for the desired outputs.

3.3.1 Element definition

Figure 36 beneath shows the numbering and face convention that *Abaqus* uses for 8 and 20 node hex elements. As can be seen from the figure both elements have the same face definition and numbering convention, with the exception that the 20 node element has mid nodes. The numbering and face convention is of great importance in order to make sure that the node numbers are collected in the element coupling matrix in an order that allows for an easy extraction of element sets and faces after the discretization is done. For instance making sure that all elements along the involute profile have the same face pointing outwards makes defining the contact areas later much less cumbersome.



(a) 8-node hex element (C3D8)



(b) 20-node hex element (C3D20)

Figure 36: Face definition

3.3.2 2 dimensional mesh

As described in Section 3.2 the node coordinates are only explicitly derived for half of a tooth of the spline coupling since it is rotationally symmetric. This part is then in its turn divided into 3 sections which are meshed separately and then joined together. The output from the geometry program is one matrix containing the node coordinates for each mesh section. Figure 37 beneath shows a scatter plot of the output matrix created by the geometry subprogram which is used as input for the mesh subprogram.

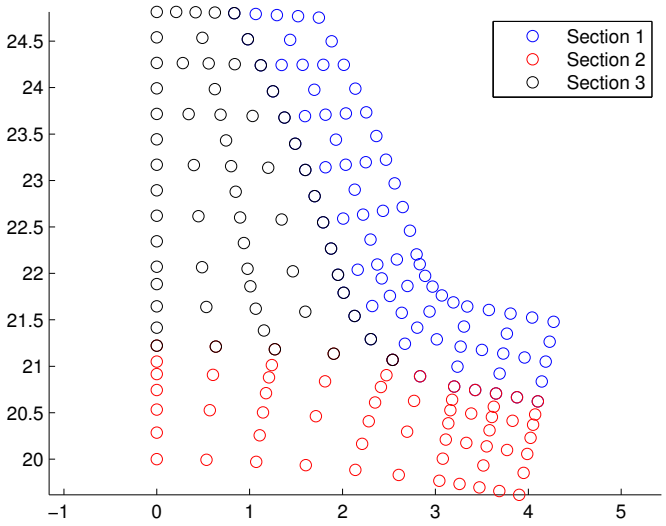


Figure 37: Scatter plot of node coordinates

In order to define the geometry of the mesh two matrices are needed, one node coordinate matrix hereby referred to as *node_coord* and defined according to Equation 45 and one element coupling matrix hereby referred to as *Enod* and defined according to Equation 46 in 2 dimensions. *node_coord* defines the spatial coordinates for each node and *Enod* defines which nodes that make up each element. In Equation 45 and 46 n_1,n_2,.. represents arbitrary node numbers and e_1,e_2,... represents element numbers.

$$\mathbf{node\_coord} = \begin{bmatrix} n_1 & x_1 & y_1 \\ n_2 & x_2 & y_2 \\ . & . & . \\ . & . & . \\ . & . & . \end{bmatrix} \tag{45}$$

$$\mathbf{Enod} = \begin{bmatrix} e_1 & n_1 & n_2 & n_3 & ... & n_n \\ e_2 & n_1 & n_2 & n_3 & ... & n_n \\ . & . & . & & & \\ . & . & . & & & \\ . & . & . & & & \end{bmatrix} \tag{46}$$

Figure 38 shows how the mesh algorithm is programmed for an example mesh, each section is divided into node lines which are then numbered chronologically. By using a set of loops to go through the node coordinate data the *Enod* matrix can be assembled. Equation 47 shows the first row of of the 2 dimensional *Enod* matrix for this example mesh.

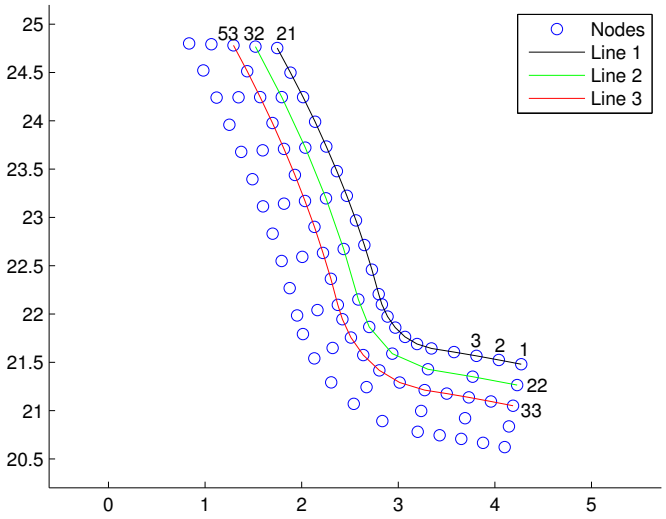


Figure 38: Mesh technique

$$\mathbf{Enod} = \begin{bmatrix} e_1 & 1 & 3 & 35 & 33 & 2 & 23 & 34 & 22 \\ . & & & & & & & & \\ . & & & & & & & & \end{bmatrix} \tag{47}$$

When the *Enod* and *node_coord* matrix have been created for each mesh section the matrices are merged together and the double nodes at the intersecting boundaries, as can be seen in Figure 31, are removed and the sections are connected in the *Enod* matrix. An example of the resulting 2 dimensional mesh for the symmetric part of the spline is shown in Figure 39.

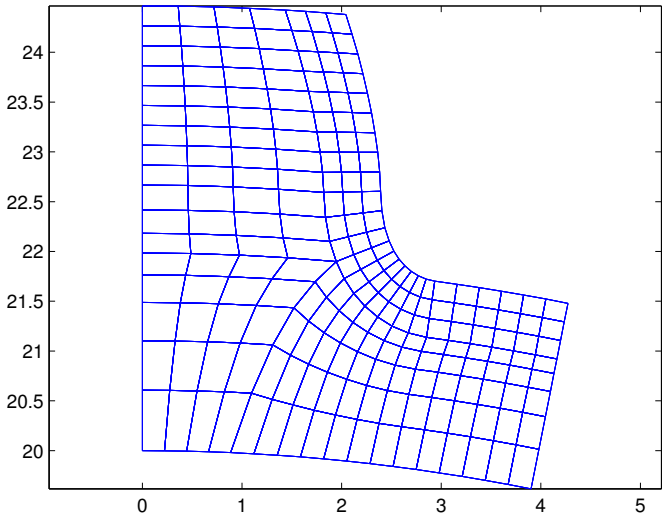


Figure 39: 2 dimensional mesh of the symmetric part of the external spline

The next step in the mesh program is to first mirror the symmetric part of the geometry in order to create a full tooth and then rotate it in a circular pattern in order to create the complete 2 dimensional geometry. The method of doing this is greatly simplified by defining all node coordinates throughout the geometry creation and mesh algorithm in polar coordinates. This means that mirroring can be done by coping the coordinates for the symmetry part of the tooth and changing the sign of the angle. The two parts can then be merged together at their common boundary. The same method can then be used for the circular pattern where only the angle coordinate needs to be altered for each tooth. Figure 40 beneath shows the mirrored geometry and Figure 41 shows the complete 2 dimensional mesh for the external spline.

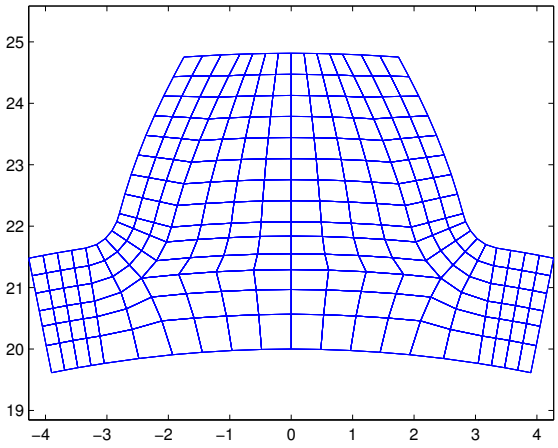


Figure 40: 2 dimensional mesh of a complete spline tooth

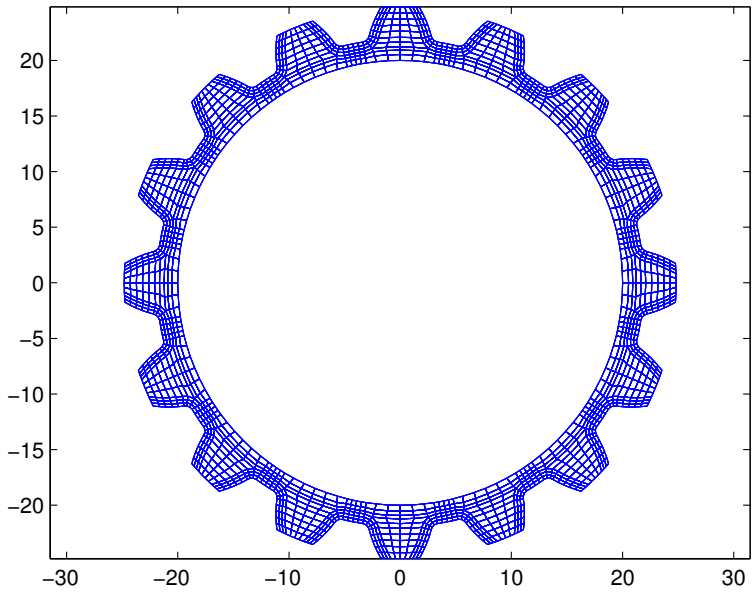


Figure 41: Complete 2 dimensional mesh

3.3.3 3 dimensional mesh

The final step of creating the mesh is to extrude the 2 dimensional mesh consisting of quad elements and turning it into a 3 dimensional mesh built up by hex elements. This is in the program done by copying the 2 dimensional mesh and extruding it in the depth direction in two steps for each element row. The first extrusion is to create the mid nodes of the elements if 20-node hex elements are to be used and the second extrusion is to create the back side of the elements. This is then done for the number of elements specified in the depth direction where the intersecting boundaries in the layers between the element rows are merged together and the surplus nodes are deleted. The resulting 3 dimensional mesh can be seen in Figure 42.

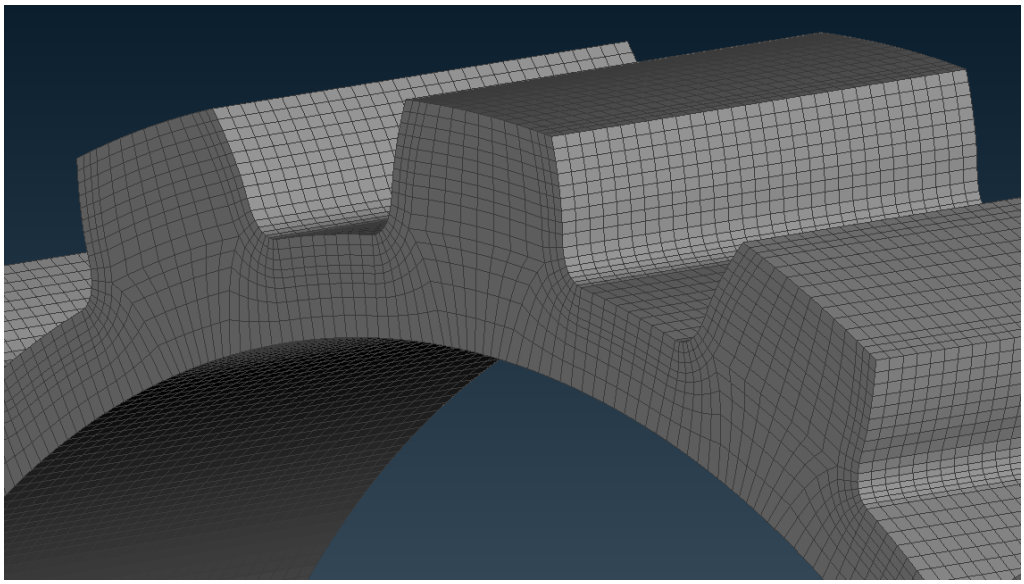


Figure 42: 3 dimensional mesh of external spline

3.3.4 Node and element sets

In order to be able to set up the problem element sets needs to be extracted for the boundaries and the mating contact surfaces of the spline coupling. When creating the mesh for the spline care has been taken to make sure that all sets have an equal spacing between element and node numbers, this makes the sets easier to extract and makes post processing easier for the user. If the user for instance wants to investigate a set of elements or nodes outside of the predefined post processing sets it is very easy to find the corresponding sets for the other teeth. This step size is written out in the *Abaqus* input file for both node numbers and element numbers. For example if the user has performed a simulation where the stress distrubution is uneven, as it would be if there is a misalignment or eccentricity included and has defined a set containing the elements 1,2 and 3. Then the corresponding sets for the rest of the teeth can be found according to Equation 48, where s denotes the set and x the step size which is the number spacing between the symmetric parts that build up the mesh.

$$\begin{aligned} s_1 &= [1 \quad 2 \quad 3] \\ s_2 &= [1 \quad 2 \quad 3] + x \cdot 1 \\ s_3 &= [1 \quad 2 \quad 3] + x \cdot 2 \\ s_n &= [1 \quad 2 \quad 3] + x \cdot (n-1) \end{aligned} \tag{48}$$

The default sets that are extracted for all simulations are shown in Table 5, some of these sets needs to be extracted in order to set up the simulation and some are extracted in order to aid in post processing. All sets are created for both parts except for the inner and outer boundary since they are part specific. From these sets the face definitions for the involute surfaces, the front and back surface and the inner and outer boundary surfaces are extracted. For the involute profile of the spline one set is created for both the left and the right side for each tooth individually. These sets are created primarily in order to be able to define the contact areas for the solver, but they can also of course be used to quickly extract stress or any other data from the contact surfaces of the spline. The reason for creating individual sets for each involute is to be able to emulate different contact ratios that can stem from manufacturing tolerances, which is explained further under Section 6.5 and to allow for simple extraction of the contact pressure. The inner boundary of the external spline and the outer boundary of the internal spline together with the front and back face of both parts are used for boundary conditions. The rest of the sets are created in order to aid in post processing. The use of the sets and the actual setup of the simulation is explained further under Section 3.5.2. Figure 43 shows an example of some of the default sets for the external involute spline.

Table 5: Node and element sets

Set	Description
Involute left side	One element set for each tooth
Involute right side	One element set for each tooth
All involutes	One node set containing involute profiles (left and right)
Root	One node set and one element set for each root (tooth)
Inner boundary	An element set for the inner boundary of the external spline
Outer boundary	An element set for the outer boundary of the internal spline
Front	An element set for the front face of each part
Back	An element set for the back face of each part

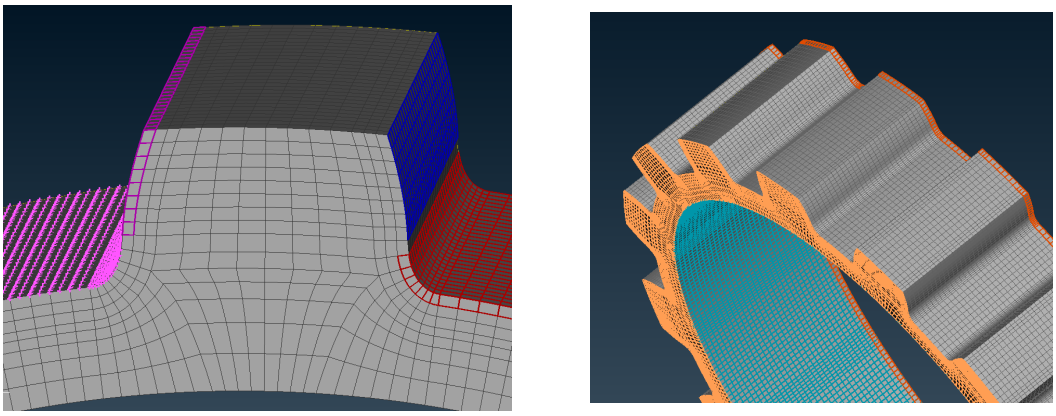


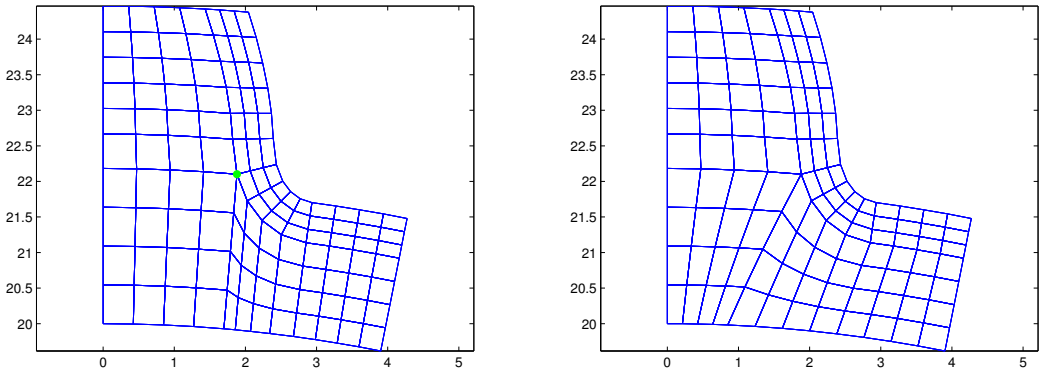
Figure 43: Root elements, root nodes, involute elements, involute surface, front surface, back elements and inner surface sets

3.4 Mesh quality improvement

The involute spline geometry is a circular geometry that contains radiuses and relatively steep derivatives in its shape, whilst the element type used in the program is rectangular. This means that simply building the spline geometry with elements defined in two directions, radial and circumferential, would result in a lot of elements with poor element geometry. The first step in addressing this is as mentioned earlier in the report to create three different mesh sections by projecting parts of the outer geometry of the spline in order to minimize the amount of bad elements and control where the worst elements end up. This also makes the mesh more flexible for the relatively large variations in geometry between different spline geometries. With the mesh sections defined there are still ways to further improve the mesh quality within the sections and to reduce the computational effort by allowing for finer mesh sizes in places of interest and coarser mesh in places where the need for resolution is low.

3.4.1 Element angle adjustment

By design the elements of mesh sections 1 and 3 of the external spline will become of sufficient quality regardless of the spline geometry used given that the mesh settings have been set up properly. The elements of mesh section 2 however will become skewed along the upper boundary due to the geometry difference between the elements and the domain, this is especially prominent around the area where all sections intersect each other (the green dot in Figure 44a). A few elements with bad geometry in this area does not affect the solution as a whole to great extent and as mentioned earlier some bad elements are unavoidable due to the geometry. However by adjusting the radial element lines stretching from the upper boundary of mesh section 2 to the inner boundary the skewness of the elements can be minimized as well as the sharp corners of the elements close to the root radius. Figure 44a beneath shows an example mesh of the external spline where the radial lines of mesh section 2 are purely radial and Figure 44b shows the same mesh where the radial lines have been moved in the negative ϕ direction in the polar coordinate system in order straighten up the elements.



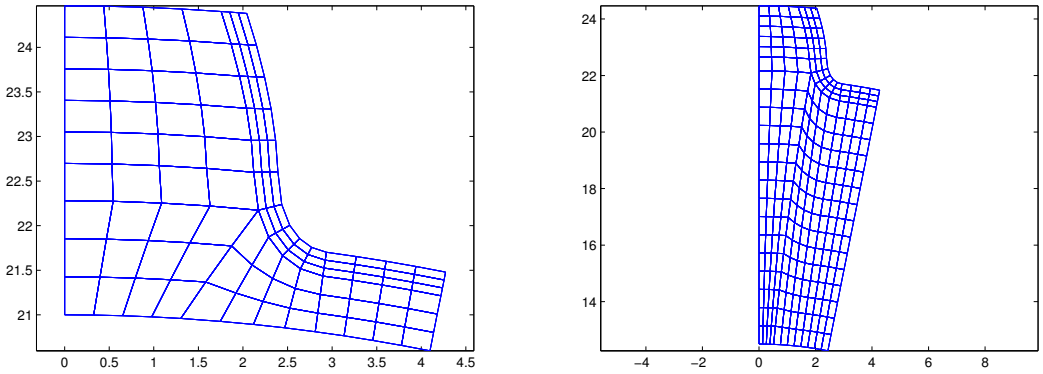
(a) Without angle adjustment

(b) With angle adjustment

Figure 44: External mesh with and without angle adjustment

The method used for adjusting the elements of mesh section 2 is to first calculate the element angle for the two elements with a common node in the point where all mesh sections intersect, the node highlighted in green in Figure 44a. The node points on the inner boundary of mesh section 2 are then moved in order to make these two angles equal up to a certain threshold depending on the size of mesh section 2 in comparison to the other mesh sections. If the inner diameter of the external spline, which can vary considerable between different spline configurations, becomes small mesh section 2 becomes large and it is not possible to have an equal angle between the two elements of

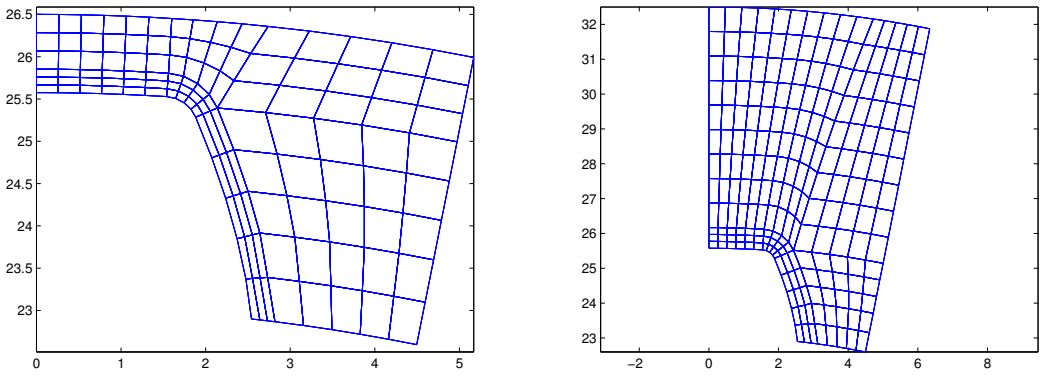
interest. If the corner angle of these two are held equal and the inner diameter becomes small the elements to the left in section 2 would end up outside of the domain and therefore a threshold on the angle of adjustment depending on the size of mesh section 2 has been implemented. Figure 45a shows an example of a symmetry section of an external spline with a large inner diameter and Figure 45b one with a small inner diameter. The mesh algorithm will not work for external splines on a solid axle i.e with an inner diameter of 0, but this is not necessary due to how the simulation is set up which is explained further in Section 3.5.2.



(a) Large inner diameter (b) Small inner diameter

Figure 45: Mesh behaviour for difference in inner diameter for the external spline

The same problem that occurs for mesh section 2 for the external spline also occurs for mesh section 2 of the internal spline. It is in this case the elements along the lower boundary of mesh section 2 that obtains small corner angles if the element lines are purely radial. Therefore the same technique to improve the element geometry has been applied to the internal spline and the behaviour of the mesh corresponding to Figure 45 for the internal spline with a varying outer diameter is shown in Figure 46.

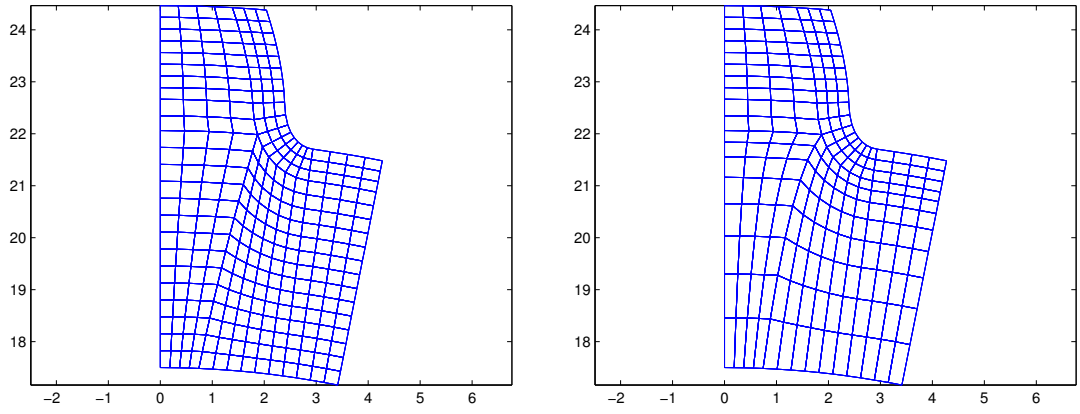


(a) Small outer diameter (b) Large outer diameter

Figure 46: Mesh behaviour for difference in outer diameter for the internal spline

3.4.2 Progressive element sizing

The inner diameter of the part with external splines and the outer diameter of the part with internal splines can vary substantially between different couplings. If the inner diameter of the external spline for instance is relatively small the total size of the part becomes large with a big portion of the part where resolution in the solution is of limited interest. This creates some challenges with respect to precision and computational effort when it comes to the discretization. If the elements stretching from the inner diameter to the upper boundary of mesh section 2 are of equal size the amount of elements will rapidly increase with a decreasing inner diameter if the elements are to be kept roughly equal in size to the rest of the part, this can be seen in Figure 47a beneath. On the other hand if the elements of this section are made large the precision and resolution at the upper boundary of mesh section 2 is compromised. In order to address this problem three different element sizing options have been implemented for this section. The user can choose from using linear element sizing, progressive element sizing or automatic element sizing, which is a progressive element sizing where the initial size is determined by the the mesh size used for mesh section 1. The automatic and progressive element sizing is based on an exponential function where the user can alter the progressiveness of the function by changing the exponent. An example mesh with linearly sized elements for section 2 is shown in Figure 47a, in Figure 47b the same geometry is meshed using the automatic element sizing with an exponent set to 2. It is clear from Figure 47 that the part using automatic element sizing will require less computational effort with small sacrifice to resolution close to the root. As mentioned the same problem occurs when the outer diameter of the part with internal splines becomes large and thus the same type of element sizing algorithms has been implemented for the internal spline.



(a) Linear (b) Automatic

Figure 47: Difference between linear and automatic element sizing

3.4.3 Element adjustment internal spline

For mesh section 1 of the internal spline the element sides in the radial direction are based on the involute profile and the projected involute profile and the element sides in the circumferential direction are orthogonal to these two profiles. The elements in the lower part of mesh section 3 are in the radial direction based on projections of the minor diameter (D_{ii}) and are linearly spaced in the circumferential direction between the projected involute profile and the left boundary. This method results in some skewness in the elements closest to the minor diameter as shown in Figure 49a. This skewness is small compared to many other elements in the discretization, but by comparing them to other elements in mesh section 1 the skewness is noticeable when calculating the determinant of the Jacobian in the element mapping. Since these elements are close to the

contact surface a function that adjusts these elements have been implemented and the result, although difficult to see, is shown in Figure 49b. Figure 48 shows the part of the spline covered in Figure 49.

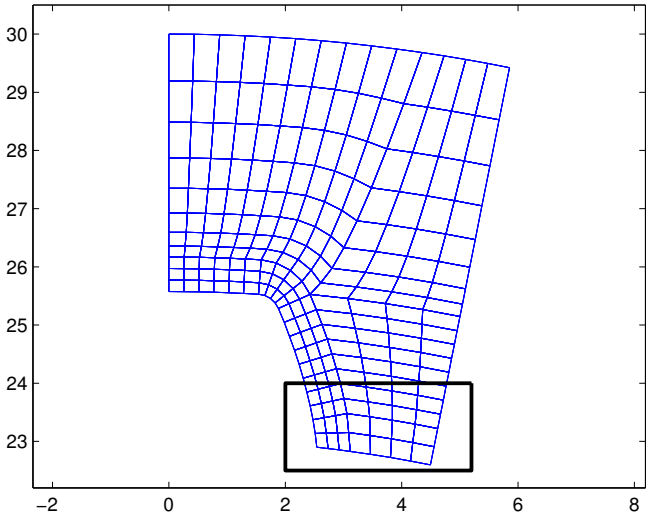
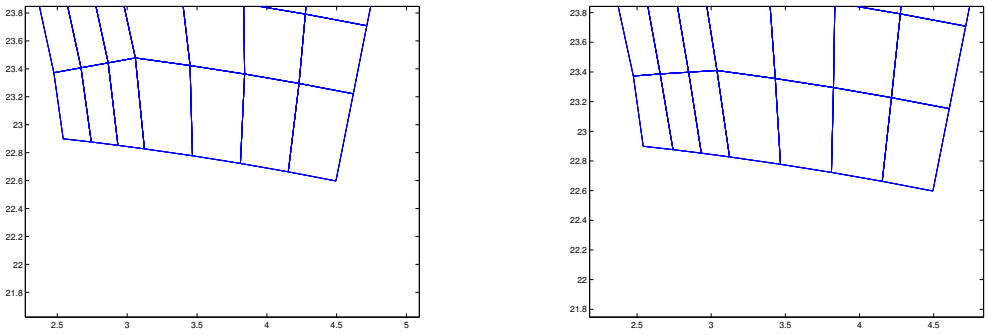


Figure 48: Part of interest



(a) Before adjustment (b) After adjustment

Figure 49: Adjustment of lower elements for internal spline

3.5 Program output

The program generates output files in *Abaqus* syntax in two levels, it generates two or three part files (.inc) depending on the configuration and one assembly file (.inp). The program also generates a .mat file containing all the data specified in the GUI, this file is used in order to be able to reset the program to settings specified in a previous analysis.

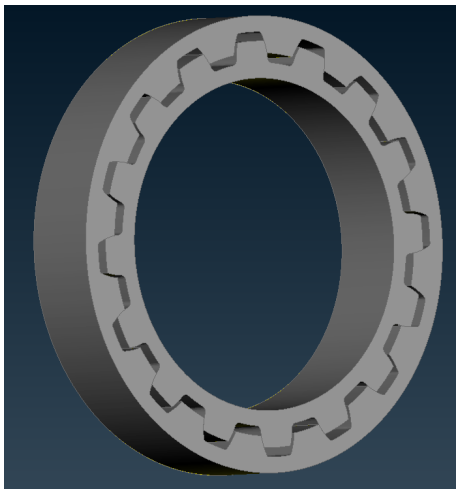
3.5.1 Output files

The setup of the simulation and the definition of parts are done in separate files in order to allow for flexibility, separating the part definition from the actual setup makes it possible to alter between a two part configuration and a three part configuration by only changing the assembly file. The part files are built up by three parts, the first is purely informational and contains the geometric description of the part, the mesh settings used in its creation and general information such as name, creation date and software version. The second part of the file is the actual geometric description of the mesh defined by the node coordinate matrix and the element coupling matrix. The last part is where all the node sets, element sets and surfaces are defined. The assembly file consists of all the information needed in order to setup the simulation as well as the output requests.

3.5.2 Simulation setup

The program can in total generate four different types of analyses, first of all it can generate a spline coupling in either a 2 part configuration with an internal spline and an external spline or in a 3 part configuration with 2 external splines and one internal spline used to emulate an engaging sleeve. For both these two configurations it is possible to run them as either a standalone simulation or as part of a shaft assembly. The standalone simulation type introduces all necessary boundary conditions in order to solve the finite element problem, where as the shaft assembly option creates all necessary element and surface definitions as well as creates tie constraints in order to allow the user to quickly couple the spline coupling to an existing model.

The assembly input files starts with reading the geometry and sets from the part files and places out the parts in space according to the users input, an example of this is shown for a 2 part configuration in Figure 50a and for a 3-part configuration in Figure 51b.

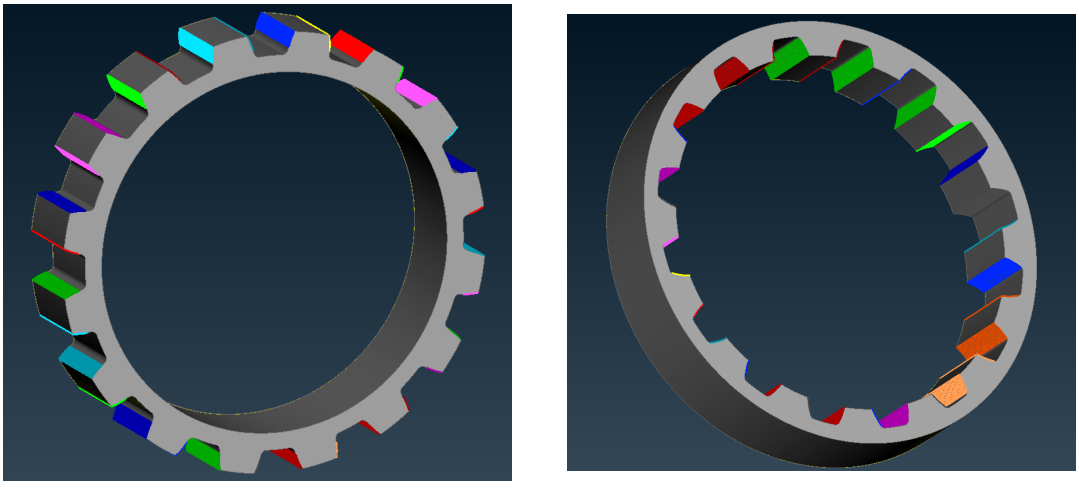


(a) 2 part configuration

(b) 3 part configuration

Figure 50: Spline coupling assembly

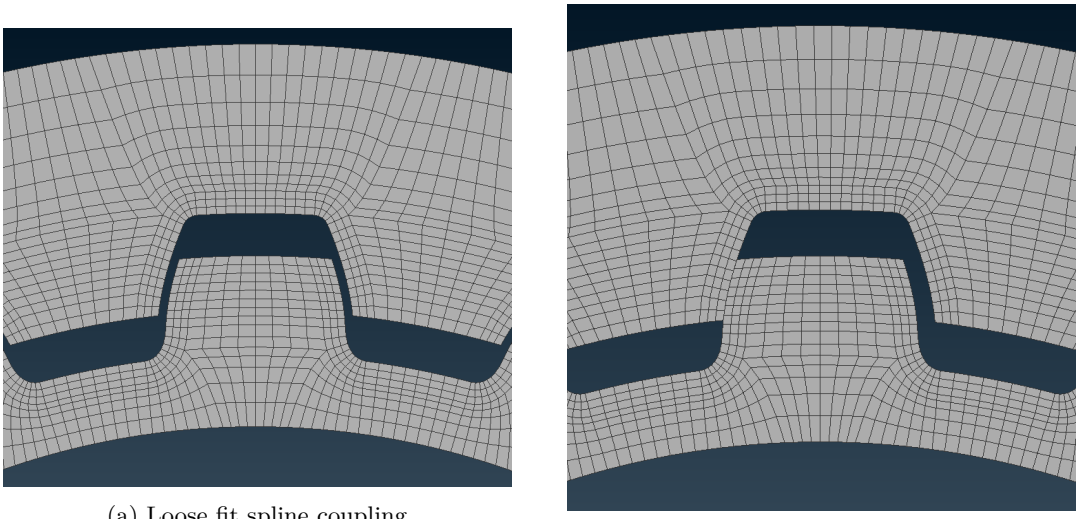
If the simulation is of a standalone 2-part configuration the next step in the setup is creating couplings in the centre of each part on to which the boundary conditions are applied, these couplings are connected to the surface pertaining to the inner boundary of the external spline and the outer boundary of the internal spline respectively. Effectively this means that the boundary conditions applied to these couplings points to their respective surface. The program can create two different types of couplings, either *kinematic* couplings or *distributing* couplings which differ in the transfer of the boundary conditions and are explained more thoroughly under Section 6.3. The program then goes on to defining the contact surfaces for the simulation, i.e the involute profile of the teeth. If the simulation is of the shaft assembly type both the left and the right involute of the teeth are defined as contact surfaces since the loading is unknown. If on the other hand the simulation is of the standalone type the side that is defined as contact surface is dependent on the direction of the torque applied and to which part the torque is applied, both these parameters are user defined in the program. In both the 2 part configuration as well as the 3 part configuration the involute surfaces of the external spline are defined as slave surfaces and the corresponding surfaces of the internal spline are defined as master surfaces. Several different setups of master and slave surfaces have been tested with little to no difference in result, so in order to maintain consistency the internal spline involute surfaces are always defined as master surfaces. There are several guidelines to follow when it comes to defining these surfaces, for instance it is often common practice to set the surface with the finest mesh as the slave surface which in this special application is hard to determine. It is a fair assumption to assume that mesh size of the involute surfaces for the internal and the external involute profile to be of roughly equal size. Furthermore the contact type has been set to *surface to surface* which means that the solver will interpolate between contact nodes if necessary in order to get the contact normal and gap function as close to reality as possible. As mentioned previously all involute profiles both left and right for all parts are defined individually. This is done partly for ease of post processing since it makes it easy to extract contact force and stress level variation. The other reason for defining these surfaces individually is order to be able to alter the gap function for each contact pair independently. This feature allows the user to emulate the effect of manufacturing tolerances by manually changing the gap between each contact surface with the *Abaqus* command *Clearance* and effectively changing the contact ratio. Figure 51 shows the contact surfaces for both types of parts.



(a) External (b) Internal

Figure 51: Extracted contact surfaces

The boundary conditions used in this analysis are quite straight forward, for the two part configuration the coupling corresponding to the spline without any torque applied to it is clamped in all six degrees of freedom. The other part is clamped in all degrees of freedom except for the one to which the torque is applied. The driven spline is only bound by its contacts with the clamped part and no inertia forces are taken into account due to it being a static analysis. This means that theoretically the part would rotate an infinite amount even by a infinitely small load step. Of course *Abaqus* has built in algorithms that establishes contacts and counteracts this problem, but to build in robustness in the program and speed up the solving process the program has a built in subroutine that automatically takes up any rotational gap in the coupling and establishes the contacts. An example of this is shown in Figure 52 where Figure 52a shows a spline coupling with a loose fit and Figure 52b how the program automatically rotates the parts in order to establish the contact.

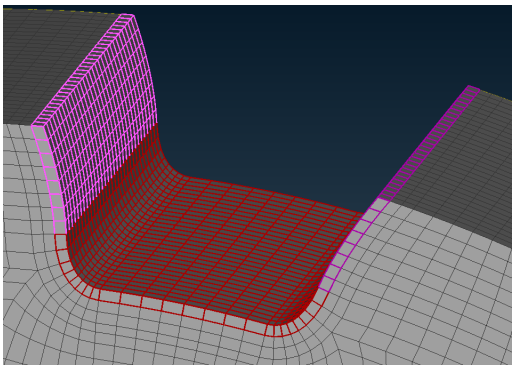


(a) Loose fit spline coupling

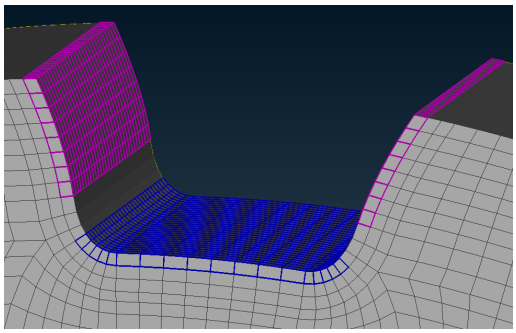
(b) Contact established

Figure 52: Automatic contact establishment

The last part of the simulation setup covers the output requests, this part of the setup mostly consists of reading the output requested by the user and translating them into *Abaqus* syntax. The user can choose all outputs recognized by *Abaqus* for all predefined node sets, element sets and contact surfaces as well as the print frequency for these outputs. There is also an additional output parameter created specifically for the node and element set pertaining to the roots of the parts, the user can select the starting point for these sets in terms of number of elements from the involute profile. This has been created in order for the user to quickly be able to define the boundary of the root set, which is one of the parts of most interest in the post processing. By for instance setting the start point of these sets one or a few elements from the start of the involute profile stress concentrations due to the contact can be avoided when analyzing the stress levels in the root. Figure 53a shows an example of an element set corresponding to the root with the boundary set all the way to the start of the involute profile and Figure 53b shows an example where the root element set boundary is moved three elements down from the start of the involute profile.



(a) 0 elements from start of involute



(b) 3 elements from start of involute

Figure 53: Root element set definition

4 Program structure

This section deals with the overall structure of the program from reading the input parameters to creating the *Abaqus* input file. The chapter starts from an holistic perspective and briefly explains the function of the main subprograms and then explains how the subprograms are built up in terms of their subscripts. Included is also an overview and explanation of the graphical user interface (GUI) and the subprograms and sub GUI:s embedded into it.

4.1 Program overview

Figure 54 beneath shows a flow chart of the main subprograms that works under the GUI. The user enters input values to the GUI and when the GUI gets the signal to create an input file it goes through the steps showed in Figure 54. It first saves all input data such as geometry and mesh settings as well as all logical data needed in order to remember the state of the interface. This data is then written to a *.mat*-file called *raw_data* which is saved in the output folder. The next step is saving the data necessary in order to set up the simulation so that the subprograms can reach it. Once this is done the geometry function is called which creates all node coordinates that are then sent to the mesh algorithm. The mesh algorithm creates the FE discretisized geometry and extracts all element and node sets. This data is then read by the program that generates the part input files (.inc) which translates the data to *Abaqus* syntax, defines all parameters needed on the part level and creates the part file. Lastly the program creating the assembly input file (.inp) is called which reads the user input data as well as the part files and sets up all parameters for the simulation which are then written into the assembly input file. As can be seen in Figure 54 the path and methods from the input data to the input files on the part level are very similar for the internal and external spline. Due to the modular build of the program the 3-part configuration spline coupling generation becomes very similar to the 2-part configuration. The major difference lays in what order the different subprograms and subscripts are called and in what order data is saved. In general the only difference is that the external section of Figure 54 is called twice and two external input files are created and a different program creating the assembly file for the 3-part configuration is called.

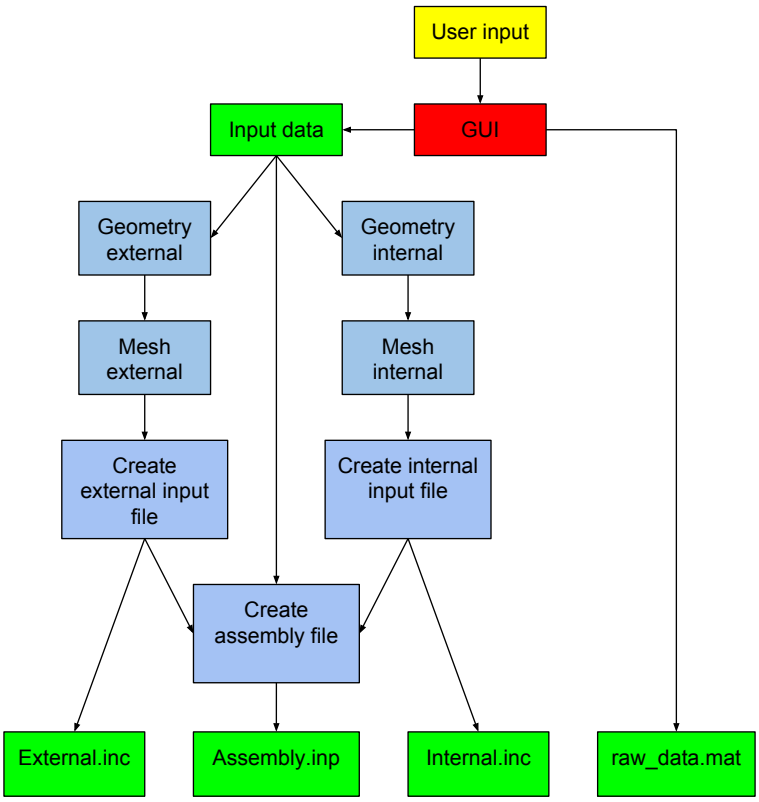


Figure 54: Program overview

4.2 Graphical user interface

The GUI part of the program is designed to help the user to set up the simulation and help to make sure that the spline coupling, mesh settings and simulation settings are according to the desired specification before the creation of the input file is initiated. The GUI is shown in Figure 55 and as can be seen it is rather large due to the many input parameters and options. It is divided into three main sections: geometry creation, meshing and simulation setup with supporting control functions for each section in order to make sure that the output is according to what the user anticipates before any computational time is put into setting up the analysis. This section will cover the typical path from input parameters to a complete ready to be solved model in the order shown in Figure 55 for a 2-part spline coupling. The creation of a 3-part model requires a few more inputs compared to the 2-part configuration and the difference is explained further in section 4.2.4.

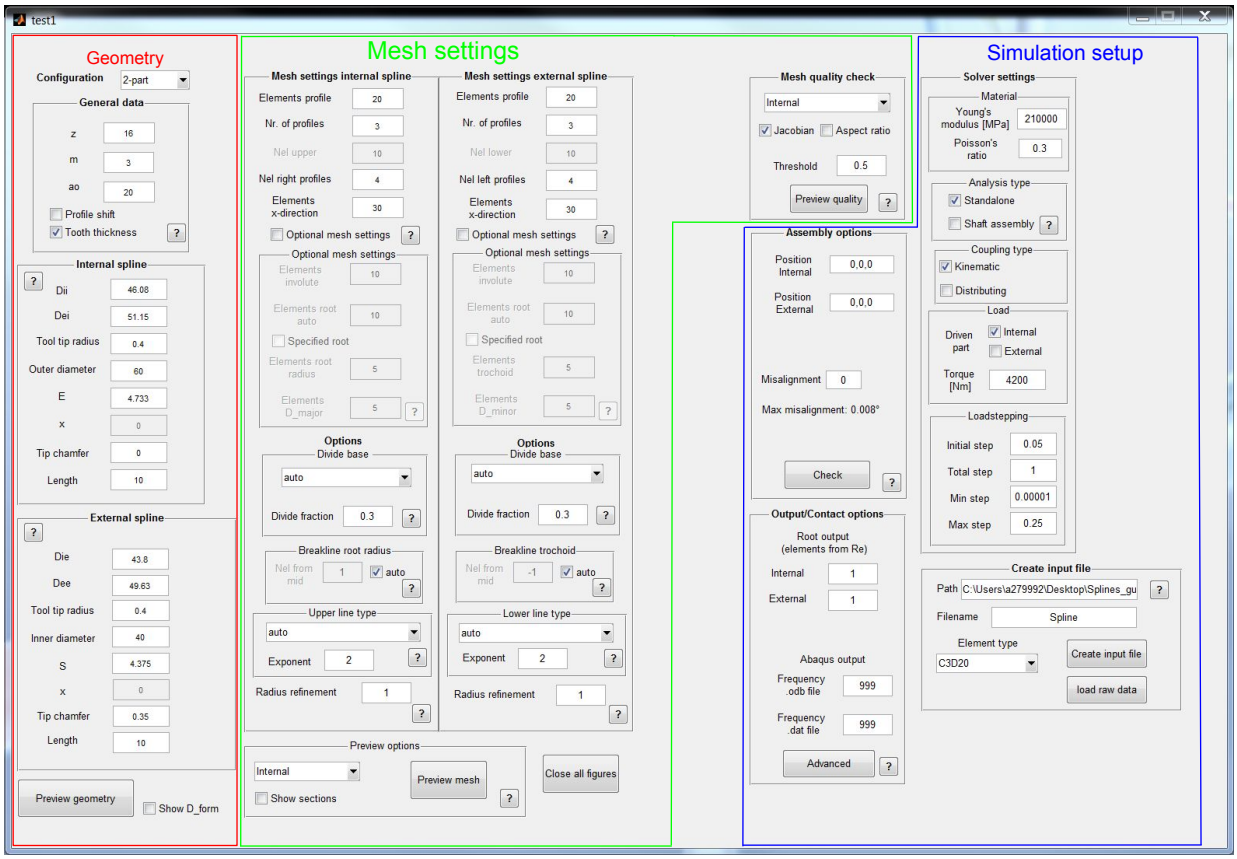


Figure 55: Graphical user interface with highlighted main sections

4.2.1 Geometry

This section deals with the geometry creation part of the GUI, this part in itself has been divided into three different parts as shown in Figure 56.

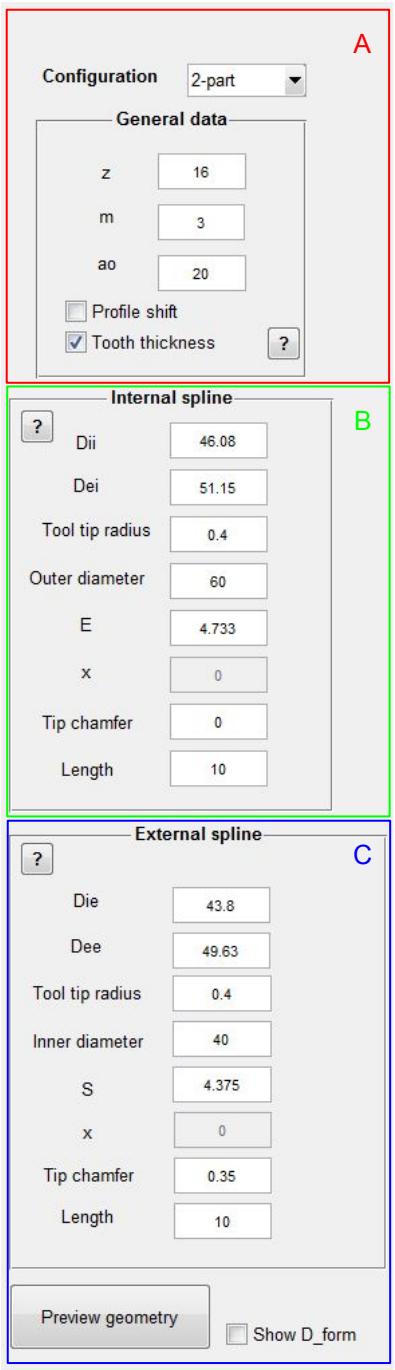
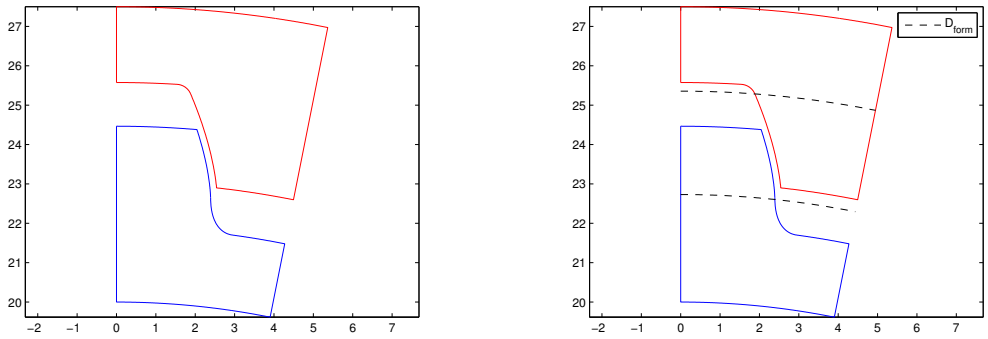


Figure 56: Geometry section of the GUI

Part A
This part consists of the *Configuration* pop up menu and the general data part of the geometric description. The *Configuration* menu lets the user select between a 2-part or 3-part configuration, triggering this menu sets the layout of the GUI between its 2-part layout and 3-part layout. The general data section is where the user sets the common geometric data that is the same for both the internal and external spline. This part also includes two check buttons where the user can choose to define the spline geometry either with a profile shift coefficient or by specifying the tooth thickness and space width. The profile shift coefficient is explained in more detail in Section 2.2.2 and the tooth thickness and space width is defined in Section 2.2.1.

Part B
In this section the geometric parameters specific to the internal spline are defined. These dimensions are quite self explanatory and have already been covered under Section 2 in this report.

Part C
This part is the external splines counterpart to part B where the geometric data for the external spline is defined. As with part B these dimensions have already been covered in Section 2. However at the bottom of part C there is a push button and a check box that are coupled to the entire geometric part of the GUI. Pushing the *Preview geometry* button calls on a script that plots a two dimensional symmetry section of the spline coupling based on the inputs entered by the user in the geometric part of the GUI, as the one shown in Figure 57a. By enabling the check box, *Show D_form*, the plot also includes the form diameter for both the internal and external spline in order to simplify for the user to see if the path of contact is correct. Figure 57b shows an example plot where the form diameter is included.



(a) Standard preview (b) Preview with D_{form}

Figure 57: Preview of the spline coupling triggered by the *Preview geometry* button

4.2.2 Mesh settings

After the geometry has been established the next step is to set the mesh settings for the discretization. The mesh control parameters are the ones within the green box in Figure 55, these have been expanded and are shown beneath in Figure 58. The mesh control parameters themselves have been divided into three parts consisting of element settings (part A), mesh section controls (part B) and visualization (Part C). The left column contains the settings for the internal spline and the right column contains the settings corresponding to the external spline.

Figure 58: Mesh section of the GUI

Part A

This part of the mesh settings controls the number of elements for the three different mesh sections specified in Section 3.2.1. In its standard state the GUI is set as displayed in Figure 58 where both the *Optional mesh settings* and *Specified root* buttons are turned off, additionally the *Lower line type* option in part B is set to *auto*. This means that the number of elements for the three different mesh sections are only governed by the active fields shown in Figure 58. As can be seen the field *Nel lower* is inactive, this is coupled to the fact that the field *Lower line type* is set to auto and is explained further under part B. Figure 59 shows the three different states of part A for the mesh settings of the external spline.

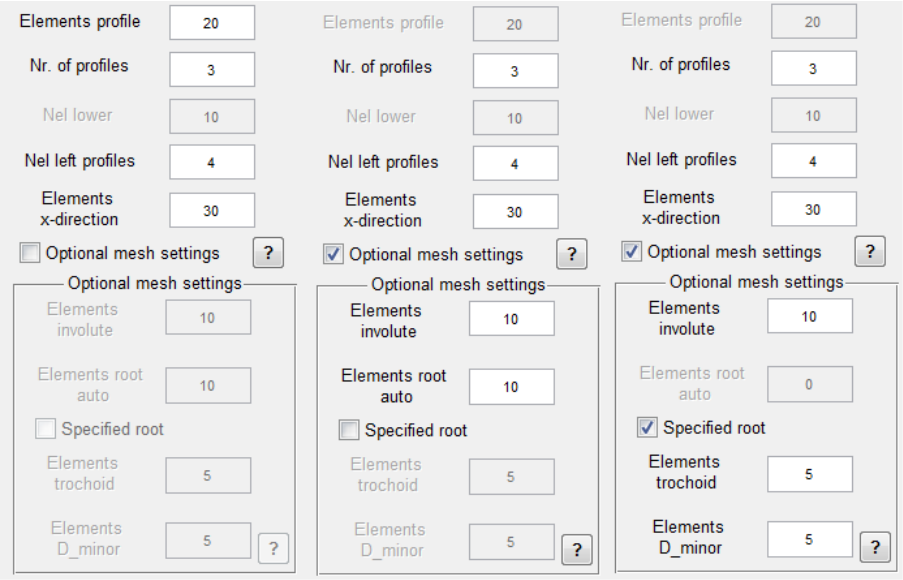


Figure 59: Part A of the mesh settings

The two check boxes in part A controls the element settings for the active profile, where the user have three different options as shown in Figure 59. The leftmost image in Figure 59 shows the standard state of the element settings where the number of elements are specified by the four active input parameters. In this case the number of elements gets distributed as shown in Figure 32 in Section 3.2.1, where a subscript calculates the length of the three different parts that make up the active profile and distributes the elements in order to get an equal element side length. If the check boxes are ticked according to the middle image in Figure 59 the user has the option to specify the involute profile and the root of the tooth separately where the distribution of elements among the two different parts of the root are done automatically based on element side length. The last option is shown in the right image of Figure 59 with both check boxes ticked. With the settings set in this manner the user can specify the number of elements for each of the three parts of the active profile as shown in Figure 30 in Section 3.2.1

Part B

The second part of the mesh parameters are shown in Figure 60 for the external spline, this part includes the settings that determines the size and position of the three mesh regions of the spline as well as some mesh enhancing parameters.

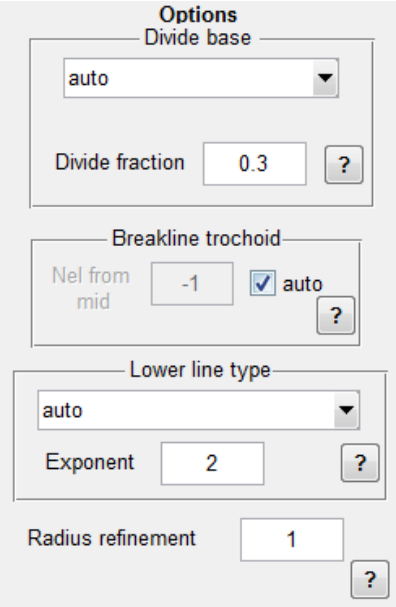
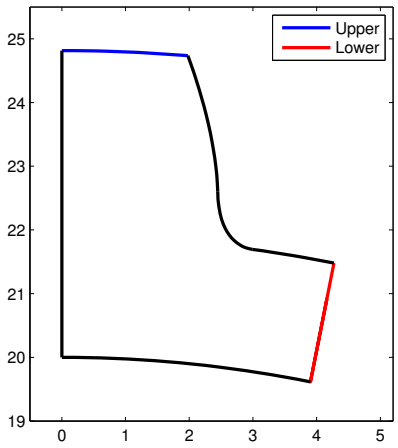
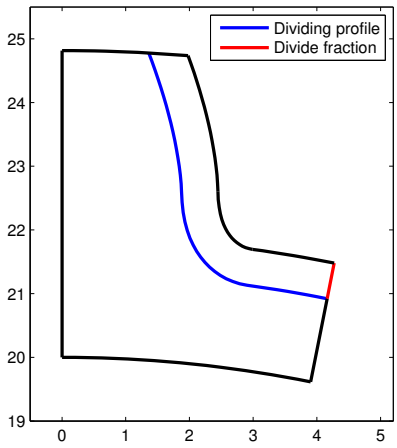


Figure 60: Part B of the mesh settings

The first parameter labeled *Divide base* is used to determine the size of mesh section 1 of the external spline. This is done by setting the distance of which the active profile is projected onto the geometry which is defined by the two parameters *Divide base* and *Divide fraction*. The divide base sets which charecteristic length of the spline to base the distance on and the divide fraction is the fraction of this distance. Figure 61a shows the two lengths of which the divide base is based on and Figure 61b shows an example where the divide base is set to lower and the divide fraction is set to 0.3 as in Figure 60. For the divide base option there is a third alternative which is the *auto* setting. By setting the divide base to auto the program calculates the length of both characteristic lengths and selects the shortest of the two, this setting yields the most stable meshing and is therefore set as default.



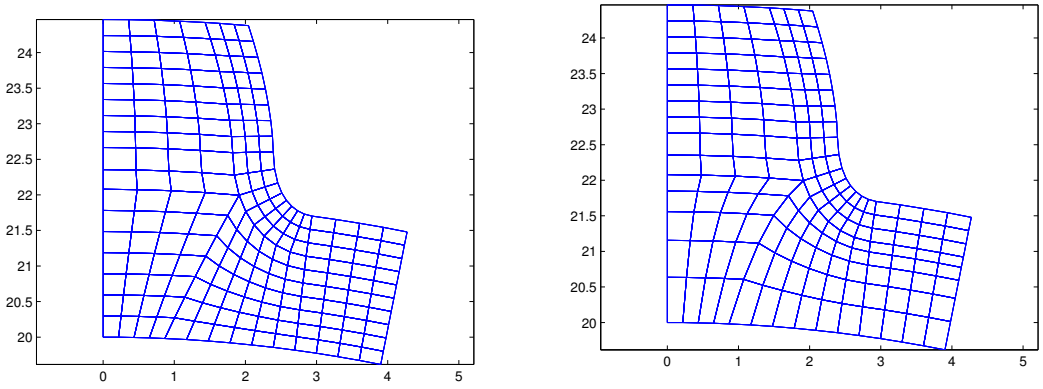
(a) Divide base external



(b) Divide fraction example

Figure 61: Mesh section 1 settings

The settings within the box named *Lower line type* in Figure 60 are used to control the mesh scheme used in mesh section 2 (see Figure 29) of the external spline. The reasoning behind including this parameter is explained more thoroughly under Section 3.4.2, but in short it allows to set the element side length in the radial direction in a linear or progressive manner. The popup menu itself has three different settings: linear, progressive and auto, where the linear setting spaces the elements linearly in the radial direction and the progressive and auto setting spaces the elements progressively based on an exponential function. The parameter *Exponent* sets the exponent of the exponential spacing function and is therefore only active if the *progressive* or *auto* setting is enabled. The difference between the *progressive* and *auto* setting is that with the progressive setting enabled the user can freely choose the number of elements, but with the automatic setting enabled the number of elements are based on the element size used in mesh section 1. The program calculates the element side length of the elements in mesh section 1 and calculates the number of elements needed for mesh section 2 in order to get a smooth transition between the two mesh sections. Figure 62 beneath shows an example of the difference between the two mesh spacing functions, Figure 62a shows a linear spacing and Figure 62b shows a progressive spacing using the *auto* setting.

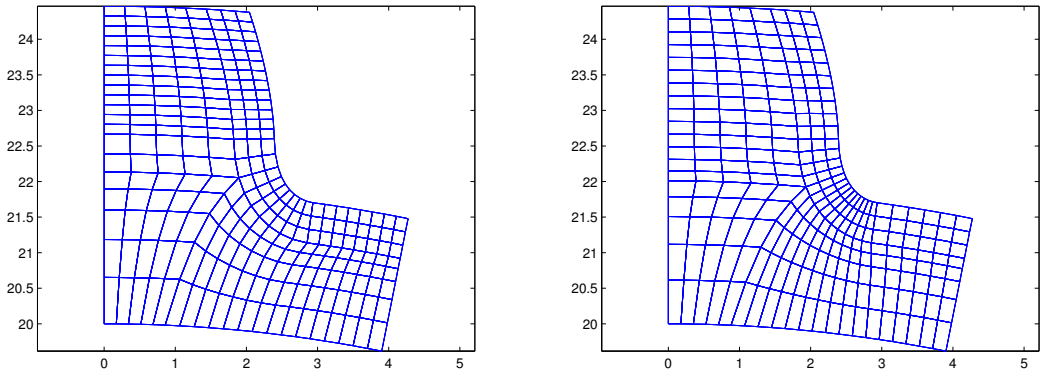


(a) Linear spacing

(b) Progressive spacing using automatic number of elements

Figure 62: Mesh schemes for mesh section 2

The last setting included in Part B of the mesh settings is the *Radius refinement* option which is as the name would suggest a parameter that is used to alter the mesh size at the root radius (trochoid) of the spline tooth. This parameter is implemented in such a way that it refines the mesh along the profile based on the curvature, i.e the larger the spatial derivative of the profile is for a certain point the more the mesh will be refined in that point. This is done in practice by setting the projected distance on which the elements on the profile have an equal side length. If the *Radius refinement* setting is set to 0 the elements in mesh section 1 will have an equal side length along the active profile (black profile line in Figure 61b) and if it is set to 1 the elements will have an equal side length along the projected profile that make up the left boundary of mesh section 1 (blue line in Figure 61b). Since this projected line on which the elements have an equal side length is imaginary the parameter *Radius refinement* can take values both higher and lower than 0. Figure 63 below shows an example where in Figure 63a the radius refinement has been set to 0 and in Figure 63b the radius refinement has been set to 2.



(a) Radius refinement = 0 (b) Radius refinement = 2

Figure 63: Example of *Radius refinement* setting

Part C

The last part of the mesh section of the GUI is shown in Figure 64 and consists of a visualization tool that can be used in order to quickly get a preview of the mesh settings applied. This function has been included in the GUI in order for the user to be able to find the desired mesh without having to go through the time consuming process of creating the entire 3-dimensional geometry. This part also includes the *Mesh quality check* box located to the far right in the *Mesh settings* section in Figure 55. This is a function that can be used to quickly obtain key quality measures of the mesh before the entire model is created.

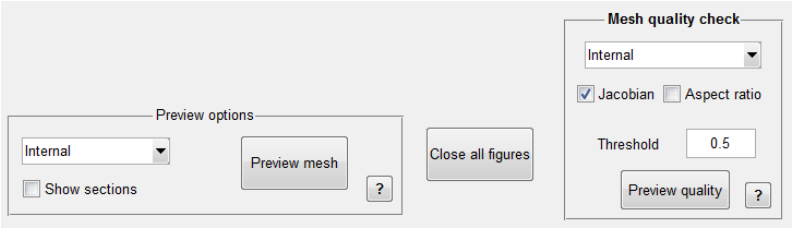
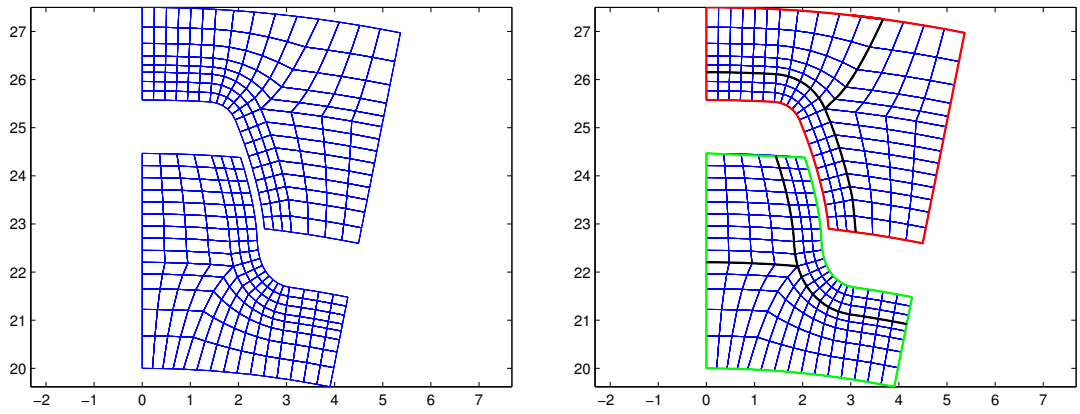


Figure 64: Mesh visualization tool

From the popup menu in the *preview options* box the user can choose which part or parts to include in the plot and by ticking the checkbox named *Show sections* the dividing boundaries of the mesh sections are included in the plot. The plot shows a 2-dimensional symmetry section of the spline, the reason for displaying this instead of a full 3-dimensional model or a full 2-dimensional model is in order to save time. Plotting large meshes in *MATLAB* takes a considerable amount of time and plotting a symmetry section is enough in order for the user to get an idea as to how the full mesh will look. Figure 65 shows an example mesh plotted by the function where both parts have been included, Figure 65a shows what it looks like with the *Show sections* checkbox disabled and Figure 65b shows the same mesh with the checkbox enabled.

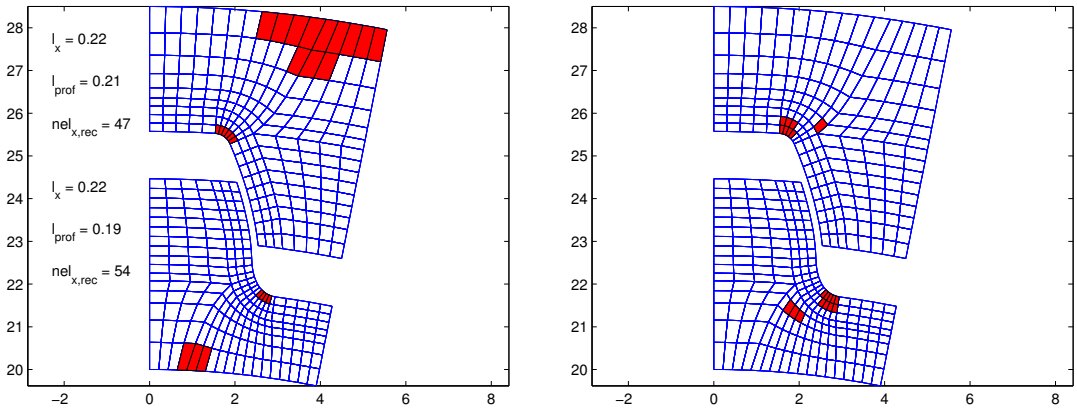


(a) Standard (b) With highlighted mesh sections

Figure 65: Mesh preview

The *Mesh quality check* box is another tool that has been implemented in order to aid the user in making sure that the final model will have a mesh of adequate quality. This function can be used in order to calculate either the aspect ratio or the ratio of the determinant of the jacobian for all elements. The aspect ratio is defined as the maximum side length of an element divided by the minimum side length of an element and thus gives a measure of the deviation from having all sides of equal length for an element. Due to the mesh algorithm used where all elements have the same side length in the extruded direction this quantity can be calculated for the full 3-dimensional geometry without having to create it explicitly. This means that the aspect ratio can be visulaized and calculated for the 2-dimensional symmetry section of the spline and be valid for the complete geometry which saves a lot of computational effort. The derivation of the aspect ratio used in the program is shown in Appendix A.2. The quantity of interest named *Jacobian* in the GUI is defined according to equation 49 and the explicit derivation is shown in Appendix A.1. For a straight involute spline without a helix angle with the mesh algorithm and elements used this quantity can also be calculated for the full 3-dimensional model without having to create it. Since the element sides in the extruded direction will be orthogonal to the 2-dimensional plane the Jacobian of the mapping between local and global coordinates calculated for a 8-node quadrilateral element will be the same as for a 20-node hex element. In the same way a 4-node quadrilateral element will have the same jacobian as a 8-node hex element. This jacobian ratio becomes effectively a measure of the skewness of the elements where a value of 1 represents a perfect cube. The major advantage of being able to do this for a small 2-dimensional model compared to the full 3-dimensional model is that the computational effort needed is exponentially less. The visualization process starts with the user selecting which part or parts to include, selects the quantity of interest to study and then enters a threshold value. The program then calculates the quantity of interest for all elements and shows all elements with a quantity of interest lower than the given threshold for the jacobian and higher for the aspect ratio. Figure 66a shows a plot of the aspect ratio for an example mesh with the threshold set to 3 and Figure 66b shows a plot of the same mesh with the quantity of interest set to *Jacobian* with a threshold of 0.8. As can be seen in Figure 66a the plot of the aspect ratio includes some information in the plot window, this information tells the user what the average element side length for mesh section 1 is in the 2-dimensional plane as well as the element length in the extruded direction. This information is given in order to aid the user in setting the number of elements in the extruded direction since the aspect ratio is a 3-dimensional quantity and the plot is 2-dimensional. The information also includes a benchmark value showing what number of elements to set in the extruded direction for a theoretical mean aspect ratio of 1 in mesh section 1.

$$\mathbf{J}_{ratio} = \frac{\min\left(\left|\det(\mathbf{J})\right|\right)}{\max\left(\left|\det(\mathbf{J})\right|\right)} \tag{49}$$



(a) Aspect ratio (b) Jacobian

Figure 66: Quantity of interest plots

4.2.3 Simulation setup

When the user is done with the mesh settings part of the GUI the geometry of both the internal and external spline is complete and the only remaining part is to set up the simulation and output requests. The part of the GUI covering this aspect of the simulation is highlighted i blue in Figure 55 and has as the other main sections been divided into subsections that are shown in Figure 67.

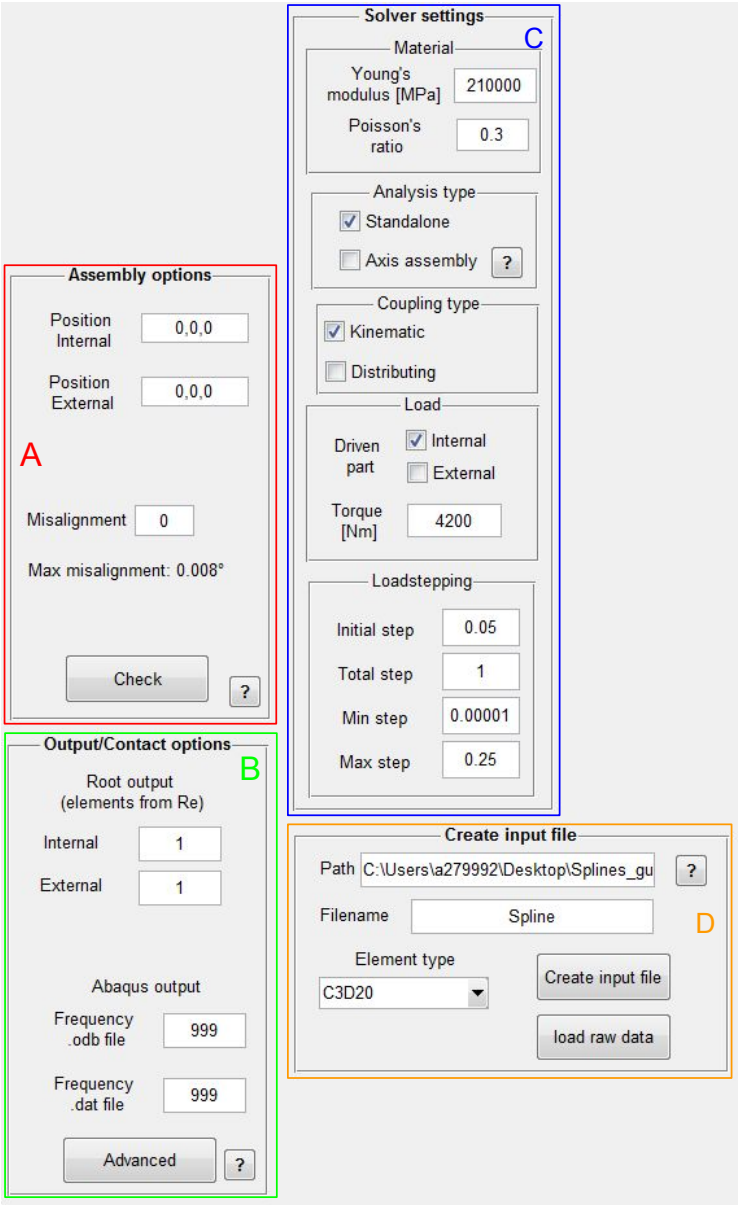


Figure 67: Simulation setup

Part A

The first part of the simulation setup (highlighted in red in Figure 67) covers the positioning of the parts for the simulation. The position of each part is defined by a three component vector that sets the position of the part relative to its centre node. The second segment of part A includes the misalignment settings for the assembly. The misalignment is designed to emulate a misalignment in the assembly of a spline coupling or between two shafts with a splined interface. The input is given in degrees and the program sets the angle between the two symmetry axes of the parts with the centre node of the external spline as rotation centre. Figure 68 shows how the misalignment is defined with the splined parts replaced with two cylinders for clarity. The *Check* button at the bottom of this section calls on a function that helps visualize the misalignment, the function plots a meshed symmetry section of the spline coupling as seen from the front face. The reason for implementing this function is to allow the user to inspect the kinematics of the contact and to help make sure that the misalignment is in the desired range before creating the input file. Figure 69a beneath shows an example spline coupling with an extruded length of 10 mm generated by the misalignment visualization tool with a misalignment set to 0 degrees and Figure 69b shows the same coupling with a misalignment set to 1.5 degrees.

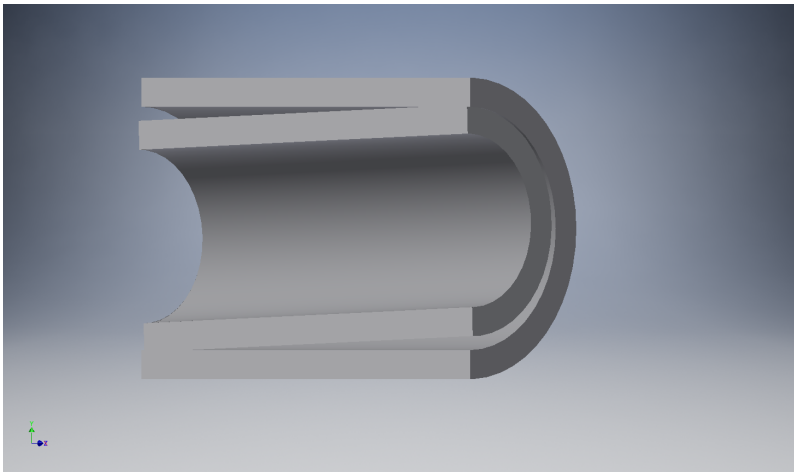
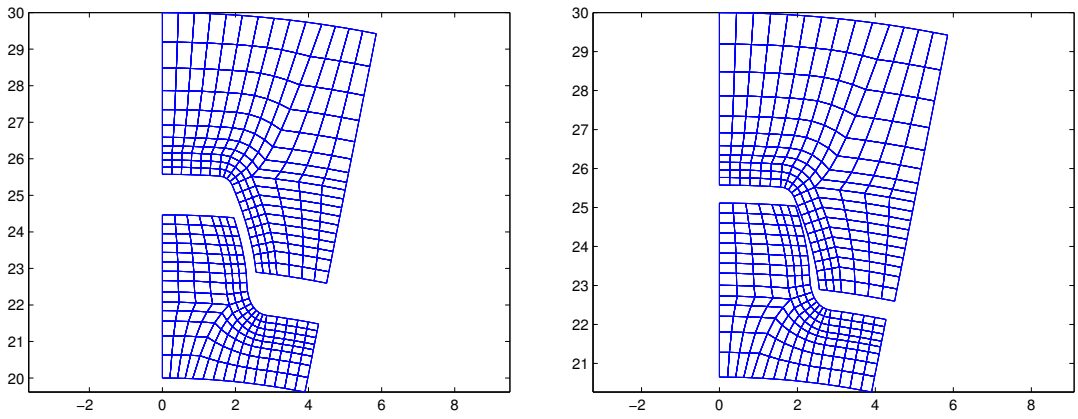


Figure 68: Misalignment definition



(a) Without misalignment (b) 1.5 degrees of misalignment

Figure 69: Misalignment visualization as shown in the GUI

Part B

Part B of the simulation setup (highlighted in green in Figure 67) is where the user can configure the settings for the contact constraints in the simulation as well as define the desired output from the analysis. In the first part of the output options the user can specify the boundary of the default node and element set extracted from the root of both parts. This is explained more thoroughly under Section 3.5.2. The second part of this region is where the user can specify the output frequency given in increments for the output files. The output for the last time increment will always be written unless the frequency is set to 0 in which case the output is suppressed. By pressing the push button named *Advanced* at the bottom of Part B a subroutine is called which starts another GUI as shown in Figure 70, where the user can modify the default contact parameters and output variables. In Figure 70 the default settings for the analysis are shown, these can be changed to any output variables recognized by *Abaqus*. The numbering section allows the user to choose the start number for both the node numbering and the element numbering. The model is built up in such a way that the node and element numbers for the parts are always unique. This setting is intended to be used when creating a spline coupling for a shaft assembly when the user might want unique numbering for all parts of the assembly. The last settings of in this part of GUI concerns the contact interface, the user can set the friction coefficient in the contact as well as the clearance setting which is used in order to manipulate the gap function in order to mimic manufacturing tolerances which is explained in more detail in Section 6.5.

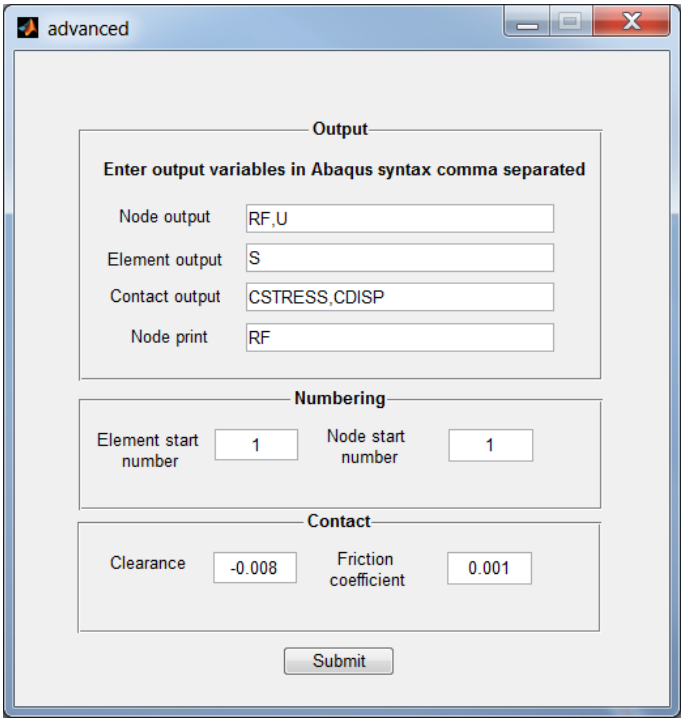
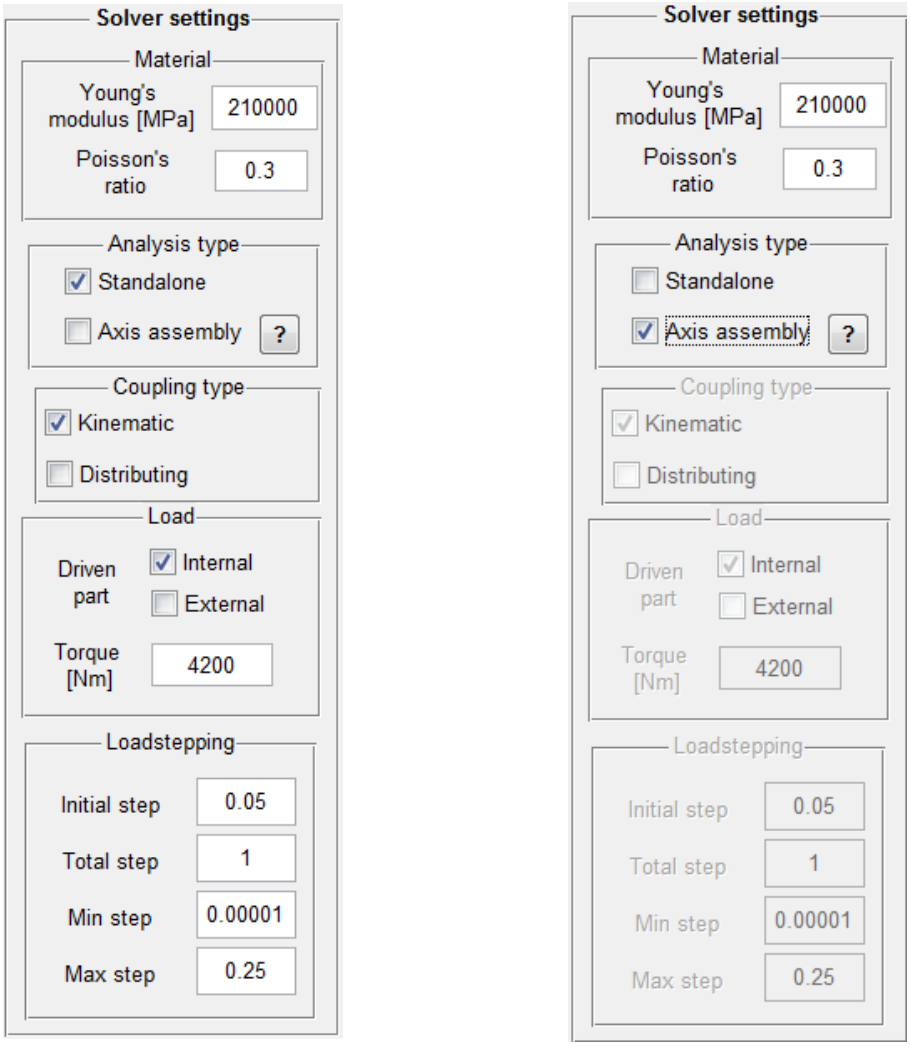


Figure 70: Advanced settings

Part C

The main setting in this part is where the user can choose between a standalone or shaft assembly analysis, by selecting *Shaft assembly* all options except for the material settings are disabled. The difference in available settings for the two types of simulations are shown in Figure 71, where Figure 71a shows the settings for the standalone application and Figure 71b shows the settings for the shaft assembly setup. As can be seen from the figure the solver settings are only relevant for the standalone application since for a shaft assembly these parameters are defined in the main simulation. Assuming a standalone simulation is to be created the user is next asked to define

the coupling type used in the simulation. The coupling is created in the middle of each part and connected to the inner surface of the external spline and the outer surface of the internal spline. It is to these couplings the boundary conditions are applied, i.e the torque and the fixed constraint. The user can choose from either *kinematic* type couplings or *distributing* type couplings for the two parts which are explained more thoroughly in Section 6.3. The last step of setting up the solver settings is to choose which part to apply the torque to, the magnitude of the torque and choosing the load stepping settings. The load stepping settings are inputs to *Abaqus* inbuilt load stepping algorithm where the initial step is load step applied in the first step of the solver, in this example 5 % of the total load according to Figure 71. The min and max step settings sets as the name would suggest the minimum and maximum step the solver is allowed to take, the solver can take any step size between these values and will determine the step size based on its own load stepping optimization algorithm.



(a) Standalone settings

(b) Shaft assembly settings

Figure 71: Simulation type

Part D

The last part of the simulation setup concerns the naming of the output files created by the program and the choice of which elements to use for the discretization. As mentioned under Section 3.1 in the report the base elements used for the meshing are either 8-node or 20-node hex elements, however *Abaqus* has some special versions of these two elements that are available for the user. Figure 72 shows the different element types the user can choose from.

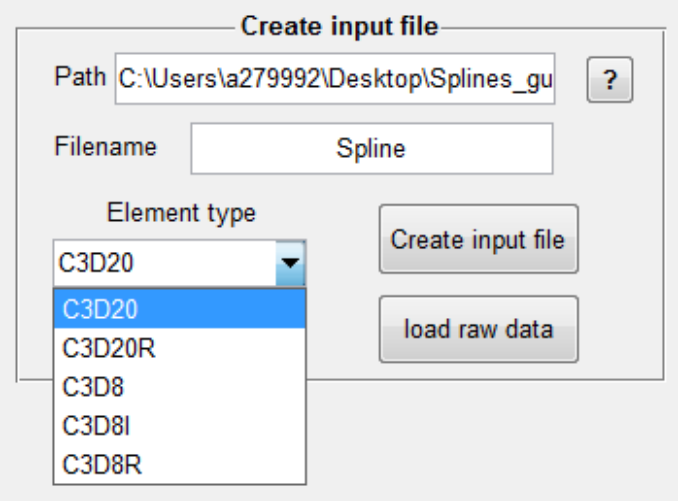


Figure 72: Element types available

The C3D8 is a standard first order 8-node hex element with linear shape functions and 2x2x2 integration points. The C3D20 is a 20-node second order element with 3x3x3 integration points and quadratic shape functions. Sub index R denotes reduced number of integration points which lowers the accuracy of the integration, but then also lowers the computational time. In this case the CD20R has 2x2x2 integration points and the C3D8R has only one in the middle of the element. Sub index I is by *Abaqus* referred to as *incompatible mode elements* which have an improved behaviour in bending. *Abaqus* adds internal incompatible deformation degrees of freedom which helps eliminate the parasitic shear stress that occurs in linear elements loaded in bending and thus makes the elements less stiff in bending, furthermore the artificial bending stiffness due to *Poisson's* effect is reduced[11]. The rest of the fields and buttons in part D concerns the output files generated by the program. In the first field the user can select the path to where the files are saved and in the second field the name of the simulation is entered. When the user clicks on the *Create input file* button the program goes to the specified save path and checks if there is a folder with the same name as entered under the field *filename*, if it already exists the content of the folder is deleted and if it does not exist a folder with the name specified will be created. Figure 73 shows an example of how the different output files are named given that the specified name of the simulation is *Spline* as it is in Figure 72.

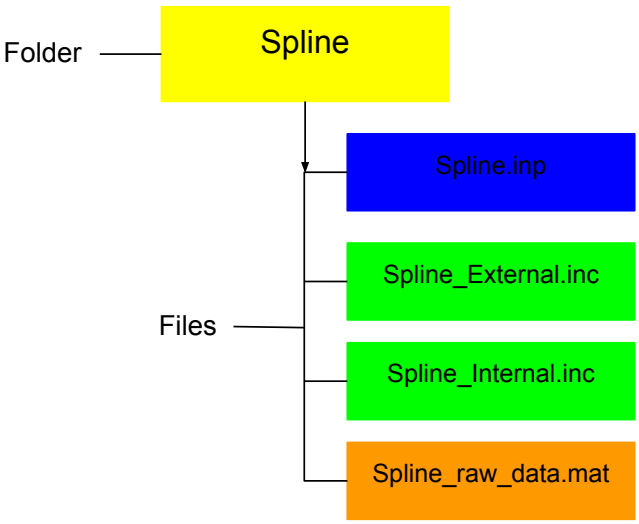


Figure 73: Program output

For every generated simulation setup a *raw_data* file (highlighted in orange in Figure 73) is saved in the same folder as the input files. By pushing the *Load raw data* button shown in Figure 72 a subroutine within the GUI is called that opens up a blank *Windows explorer* window in which the user is asked to browse and select the *raw_data* file from a simulation. By doing so the program reads the *raw_data* file and resets all the settings in the GUI according to that simulation. This functionality has been included in the program in order to save time for the user, by having this function it is very easy for the user to correct mistakes or tweak the settings for an old simulation without having to reenter all settings manually. The last feature that has been included in the program is a status bar at the bottom under the *Create input file* box where the program writes out information regarding where in the process it is as well as prints some statistics regarding the created setup once the creation is complete. This function is included in order for the user to be able to estimate the computation time needed to create the files and showing the user that the program is working. Figure 74 shows a snap of the status bar while the program is in use.

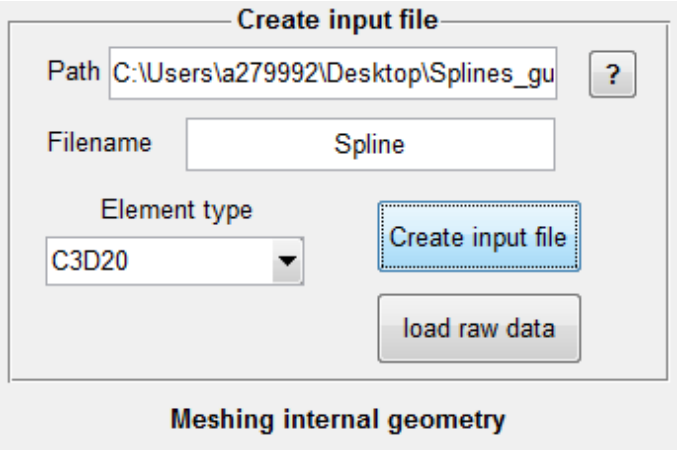


Figure 74: Status bar

4.2.4 3-part configuration

In chapters 4.2.1 through 4.2.3 all the steps from creating the geometry to writing the input files have been shown for a 2-part spline configuration. The reason for only showing the complete process for the 2-part configuration is because the method for creating the 2-part and 3-part configuration are very similar. The only thing that separates them is that the 3-part configuration has more parameters that need to be specified for obvious reasons. Figure 75 beneath shows the complete GUI when the configuration popup menu is set to 3-part configuration.

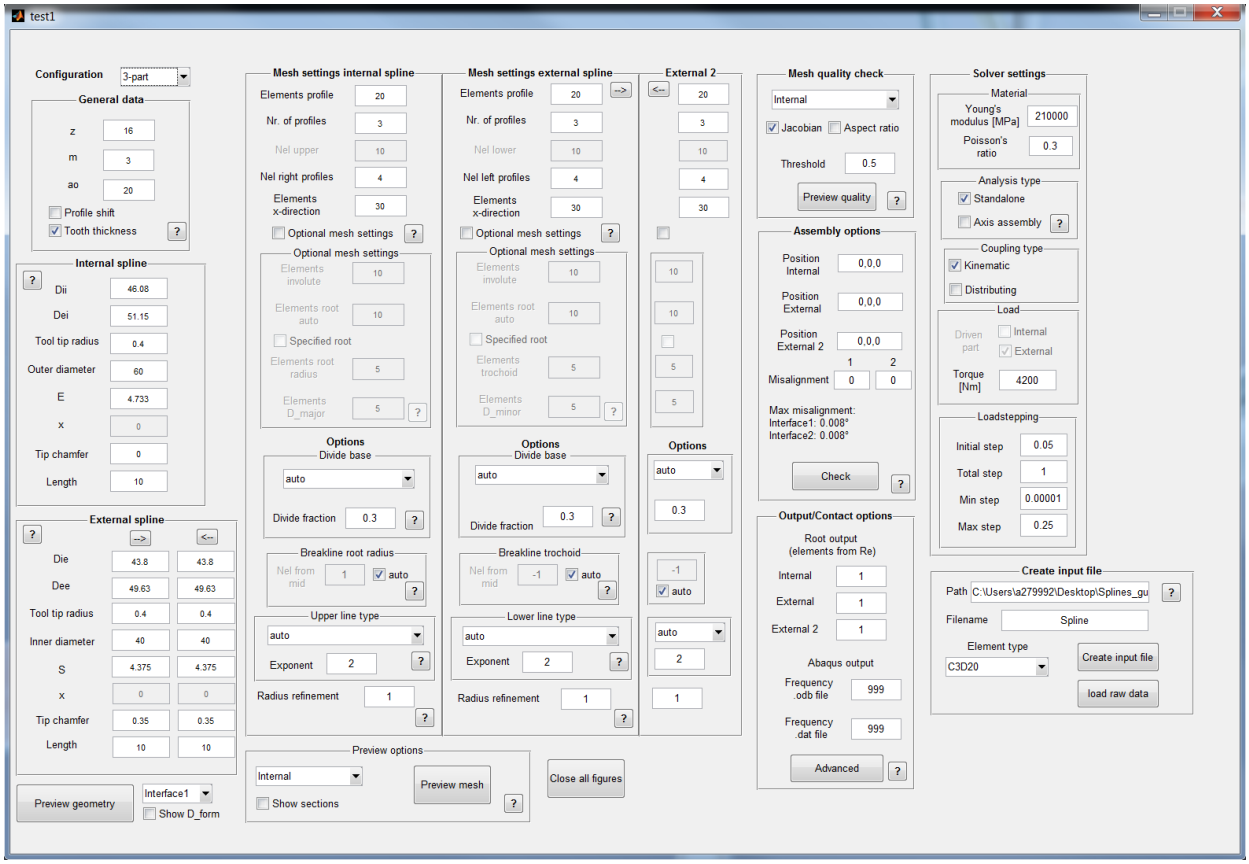


Figure 75: Graphical user interface for 3-part configuration

As can be seen in Figure 75 the GUI has been expanded to include an additional external spline for both the geometry settings as well as the mesh settings. Since the 3-part configuration has two interfaces the different supporting preview functions has been updated to be able to show both interfaces, furthermore the simulation setup becomes adapted to the 3-part configuration. Figure 76 shows the geometry section for the external spline for a 3-part configuration, as can be seen a second column with the same inputs becomes activated. The same goes for the mesh settings where a new column with the same inputs and functionality is activated to the right of the first external spline, as can be seen in Figure 75. The only difference compared to the 2-part configuration apart from the extra column are the added push buttons at the top of Figure 76, which have also been implemented in the mesh settings part of the GUI. These buttons are used in order to transfer data between the two columns which simplifies the process of creating the coupling since the geometry and mesh settings of the two external splines are often very similar if not the same.

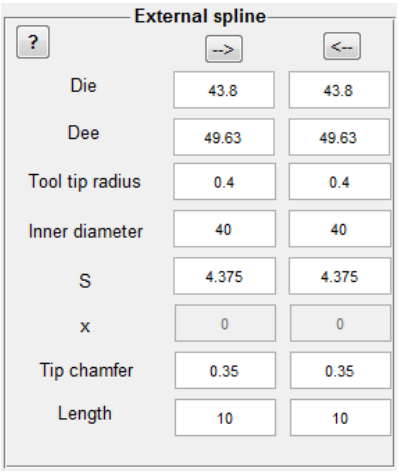


Figure 76: External geometry inputs for 3-part configuration

The second difference between the 2-part and 3-part configuration is that all supporting functions that plots the spline coupling becomes redefined in order to be able to show all parts and interfaces of the 3-part coupling. In Figure 77 beneath the mesh preview function for the 3-part configuration is shown as an example where the selection of parts has been expanded, the same goes for all other plot functions.

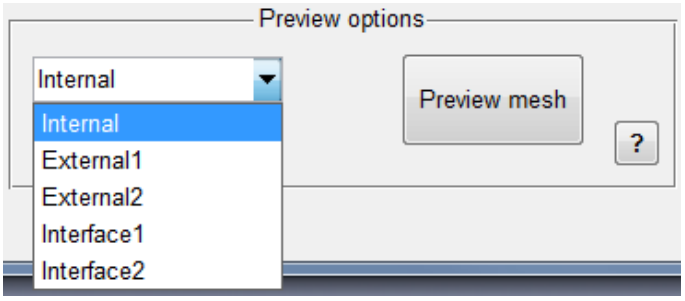


Figure 77: Support functions for 3-part configuration

Regarding the simulation setup part of the GUI most of the input parameters stay the same for the 3-part configuration although expanded to include the third part where necessary. As an example the *Assembly options* for the 3-part configuration is shown in Figure 78. The other main difference is the loading where the choice of which part to apply the load to is removed since the load case will always be that the engaging sleeve (internal spline) transfers the torque between the two externally splined parts. When it comes to the output of the program the output works in the same way as shown in Figure 73 for the 2-part configuration with an added part file for the second external spline and of course with the content of the assembly file differing.

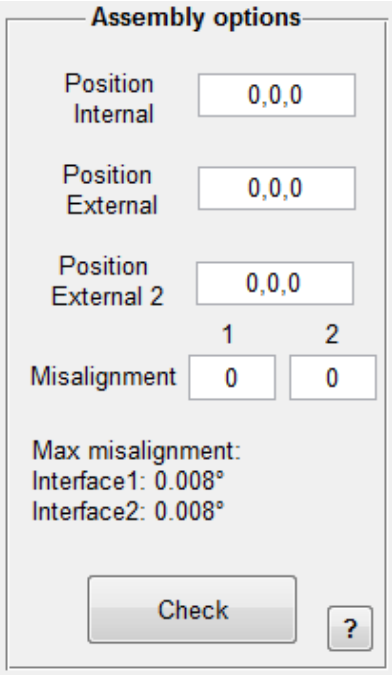


Figure 78: 3-part configuration assembly options

4.3 Supporting scripts

When the main script is started it builds up and shows the graphical interface into which all input parameters are entered and all supporting scripts are called from. The main program itself contains several sub functions that handle all data management as well as all the logic for the interface. The more extensive supporting scripts as well as the supporting scripts with a graphical interface are coded as separate programs that are called from the main GUI. This section covers the structure and basic function of all programs and scripts underneath the main GUI in the hierarchy.

4.3.1 Preview scripts

The preview functions have been somewhat covered under Section 4.2 and includes the geometric preview, mesh preview, mesh quality preview and the misalignment preview. The functionality of the geometric preview together with its subscripts is shown in Figure 79. As shown in the figure the GUI triggered by the user calls the two supporting scripts that with the help of their own subscripts generates the coordinates needed and returns the data to the main GUI where the plot is created.

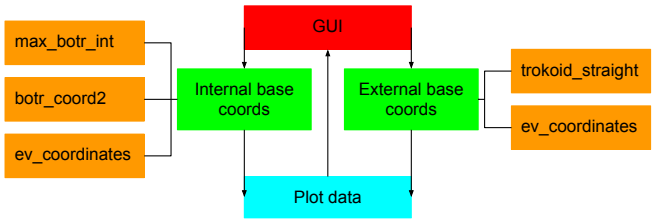


Figure 79: Flowchart for creating the geometric preview

The mesh preview, mesh quality check and misalignment check all uses the same supporting script for their visualization although the input parameters emitted to the program and the subscripts called from within the program differs. Figure 80 shows the call sequence for creating the mesh preview. The subprograms *Geometry internal* and *Geometry external* that in this case are under the program *mesh2d* in the hierarchy are the subprograms used in order to define the geometry based on the input data for the mesh algorithms as shown in Figure 54. These programs in themselves have several subscripts beneath them which are explained more thoroughly under Section 4.4.

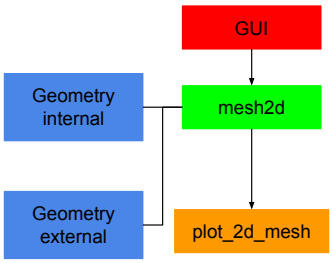


Figure 80: Flowchart for creating the mesh preview

The preview for the mesh quality check and the misalignment check are very similar to the mesh preview with the only difference being additional subscripts that calculate the quantity of interest. Figure 81 shows the buildup of the mesh quality check and Figure 82 shows the buildup of the misalignment check.

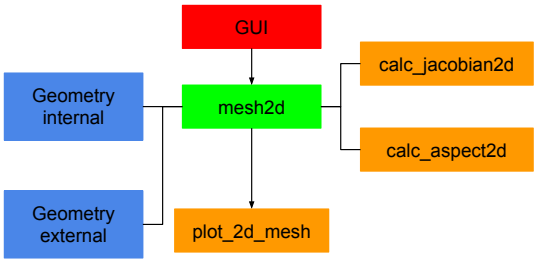


Figure 81: Flowchart for creating the mesh quality check

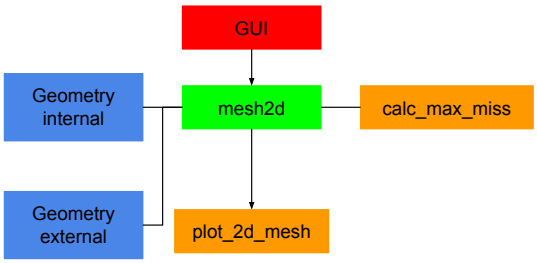


Figure 82: Flowchart for creating the misalignment preview

4.3.2 Help buttons

As can be seen in the figures showing the GUI throughout Section 4.2 all of the different sections of the GUI have a push button with a question mark symbol. These buttons are help buttons for the respective parts of the GUI which have been implemented in order to guide the user through the setup of the simulation and contains information regarding the optional parameters within each section. The help programs that are triggered by the help buttons are built as GUIs in *MATLAB* and are under the main GUI in the hierarchy. In total there are 20 help sub GUIs corresponding to each section of the main GUI. The reason for including these help functions in the program is that they give the user just the information needed in an effective way compared to having a separate user guide, additionally all the information the user needs is collected under one program which makes it more user friendly. Figure 83 shows an example of the help provided through the help buttons, in this case it is the help function for the optional mesh settings for the external spline (shown in Figure 84).

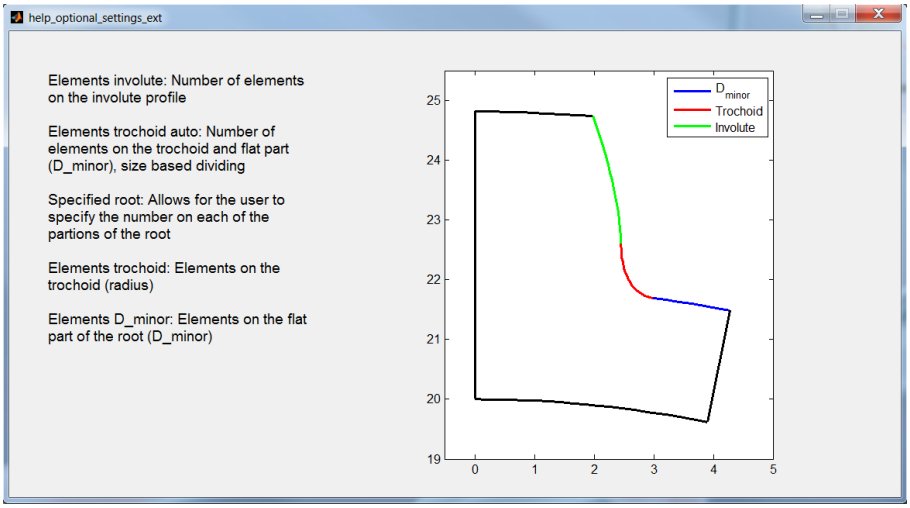


Figure 83: Help for optional mesh settings external

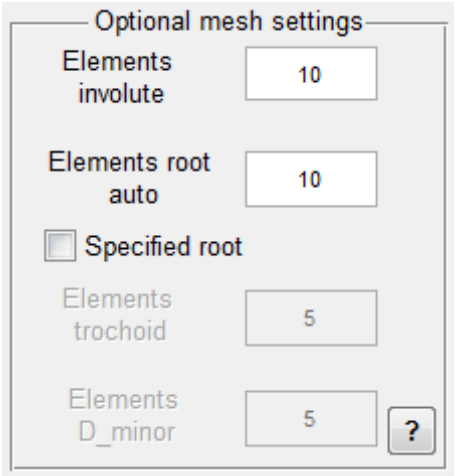


Figure 84: Optional mesh settings external

4.4 Subprogram overview

The main functionality of the subprograms have already been explained under Section 3 so the focus in this section will be put on their structure and the call sequence of the subscripts they use. The subscripts themselves are quite narrow in their use and are therefore explained further under Appendix B. Figure 85 shows an overview of all subprograms, subscripts, help scripts and supporting scripts used in some way by the main program. The general path for creating the *Abaqus* input files is shown in Figure 54 under Section 4.1 and the main GUI, the supporting scripts and the help scripts are covered under Section 4.2.

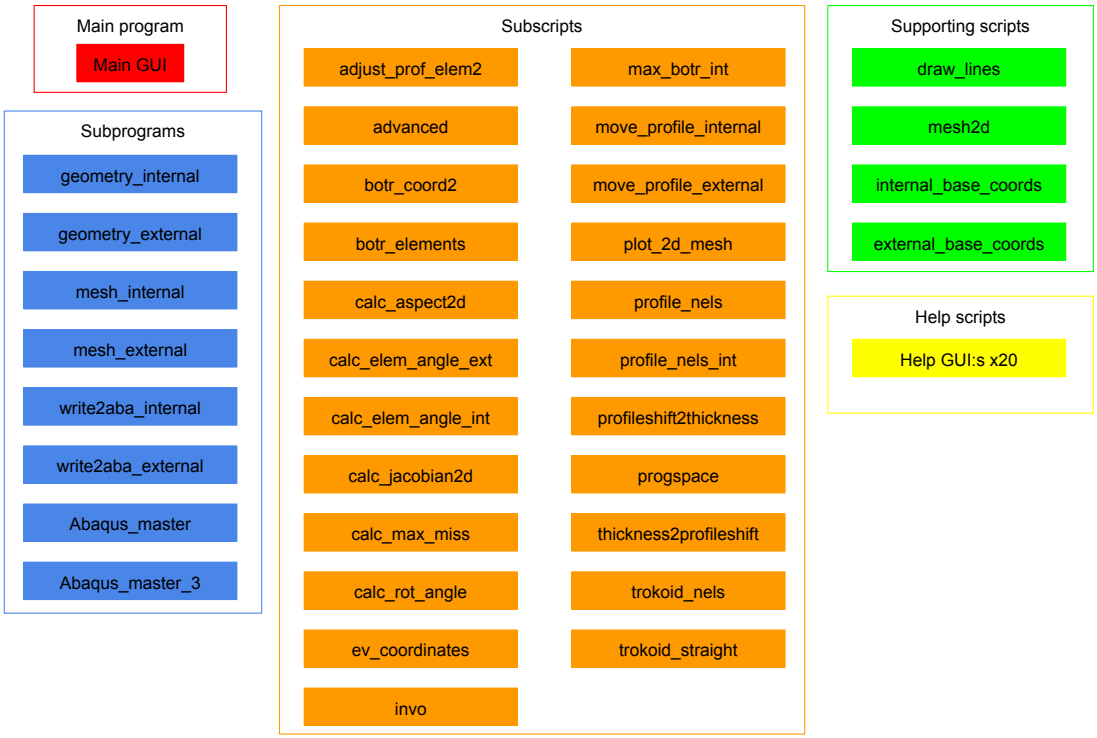


Figure 85: Program overview

4.4.1 Geometry creation

The purpose of the two geometry subprograms, *geometry_internal* and *geometry_external*, is to create the coordinates for all nodes of the 2-dimensional symmetric part of the spline. The input to both programs is the geometric and mesh data read from the main GUI. The output for each program are three matrices containing the node coordinates for each of the three mesh sections. Figure 86 shows the internal geometry subprogram together with the subscripts called from the program. The subscripts highlighted in orange are called from the subprogram and are listed in order of appearance vertically, the subscripts highlighted in yellow are called from within another subscript and are listed in order of appearance horizontally. Figure 87 shows the corresponding buildup for the external geometry subprogram *geometry_external* where the subscripts are listed in order of appearance in the same way as in Figure 87.

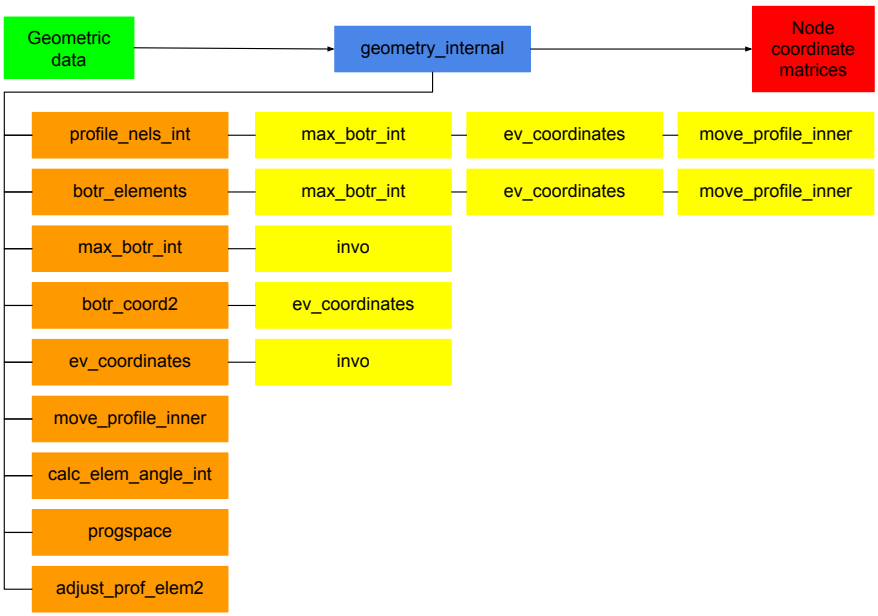


Figure 86: Internal geometry script overview

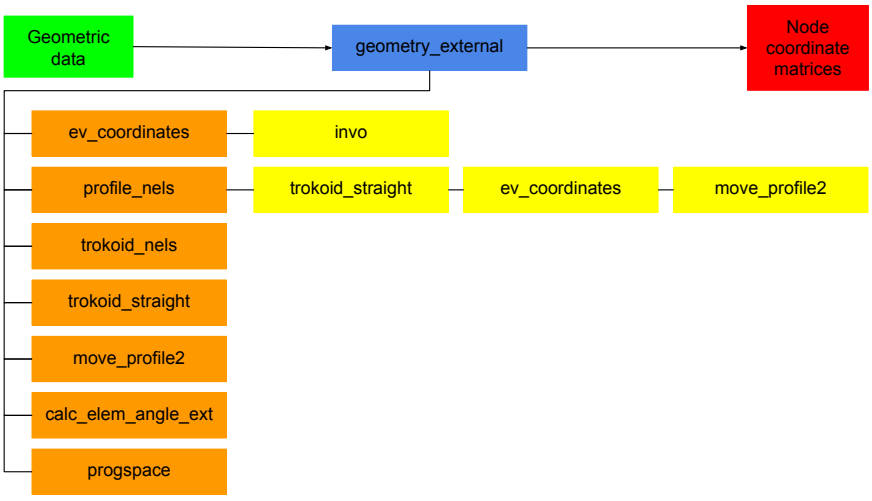


Figure 87: External geometry script overview

4.4.2 Mesh algorithm

In terms of program structure the mesh algorithms *mesh_internal* and *mesh_external* are much simpler. These subprograms do not need any subscripts in order to create the mesh and the structure in terms of inputs and outputs are also simple as can be seen in Figure 88 and 89. The mesh algorithms takes the node coordinate matrices for the mesh sections created by the geometry subprograms and creates the *node_coord* and *Enod* matrices needed to define the mesh as well as creating all the node and element sets needed for the simulation setup and post processing.

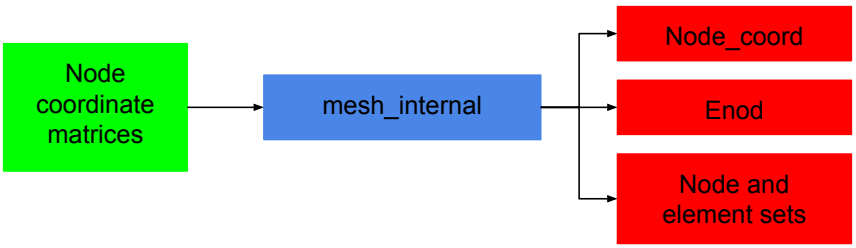


Figure 88: Internal mesh script overview

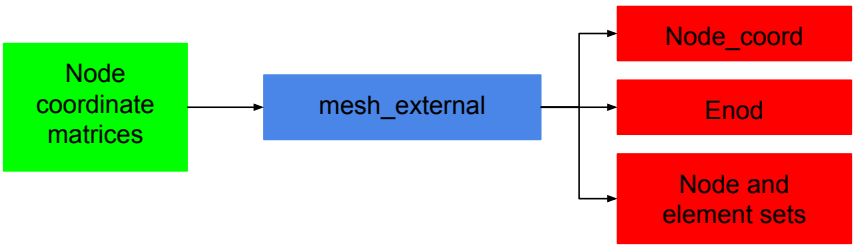


Figure 89: External mesh script overview

4.4.3 Creating *Abaqus* files

The subprograms creating the part input files, shown in Figure 90, takes the matrices created by the mesh subprograms and writes them in *Abaqus* syntax in the part files. The program also takes the input parameters for the geometry and mesh settings in order to write out information of the part in the file. The subprogram creating the assembly input file takes the data from the simulation setup part of the main GUI and creates the main input file based on those inputs, as shown in Figure 91.

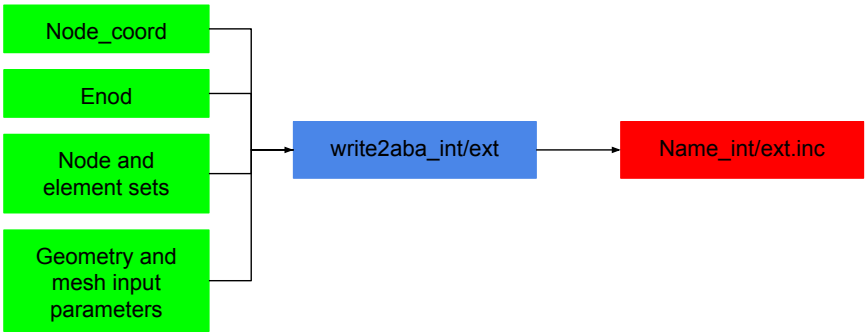


Figure 90: Creating part input file

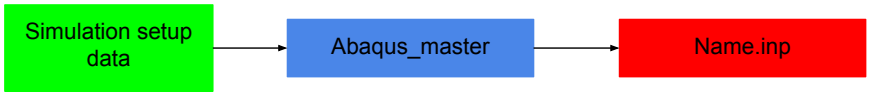


Figure 91: Creating main input file

5 Geometrical verification

The geometrical verification is done in order to make sure that the parts created by the program have the correct geometry. This is crucial for the results obtained from the program to be regarded as valid. The verification of the geometry has been done for a number of different spline geometries, where all the geometrical input parameters have been varied in order to make sure that the correct tooth profile is created for all different types of straight involute splines. The verification has been performed with the help of an in-house developed software called *TGA norm* which is used to check the involute profile for meshed models of splines and gears. The program is intended to be used to check that the nodes of the discretisized involute profile follows an analytically derived curve. The program needs an *Abaqus* file that contains a node set for the involute profile for all teeth of the spline as input. For this reason both part output files from the *spline optimization tool* contains a node set for the involute profiles in order to simplify the usage of the *TGA norm* script. The program controls the geometry of the involute profile by creating the analytical curve based on the input given by the user and then compares this curve to the nodal coordinates of the node set for the involute profiles. The program then uses the least square method in order to determine the error between the nodes and the curve and the user can then choose to alter the model according to the results obtained. Figure 92 shows the result from running the program with an example part generated by the *optimization tool* as input. As can be seen from the figure both the suggested average adjustment and the suggested maximum adjustment are in the order of 10^{-6} and since the dimensions are given in millimeters this means that the maximum error is in nanometers which can be considered as a numerical fault.

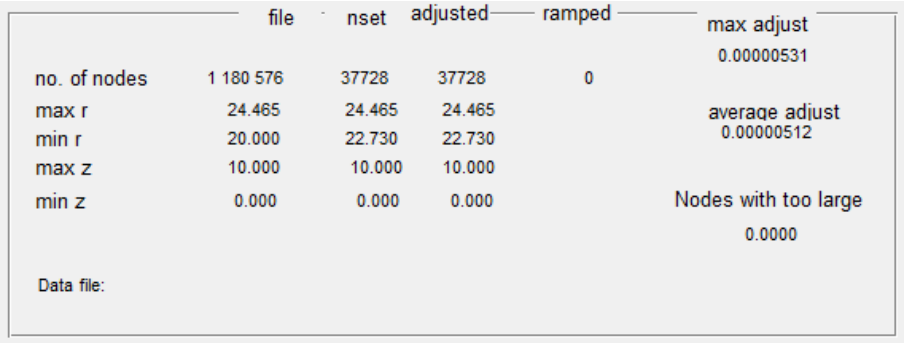


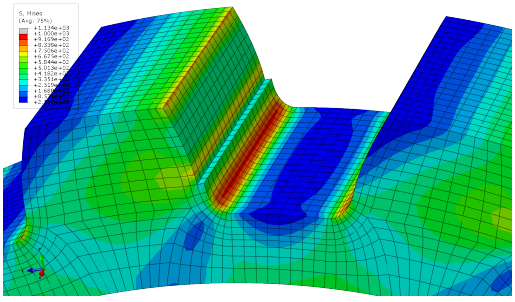
Figure 92: Results from *TGA norm*

6 Testing and results

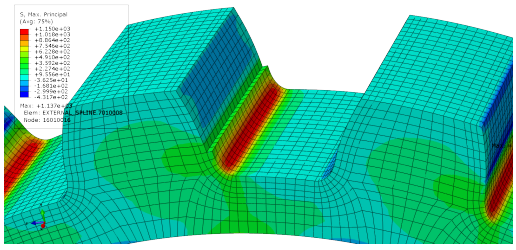
In order to evaluate the performance and usability of the program a number of simulations has been run first in order to evaluate the stability of the program and secondly to identify how different settings affect the end result. As included in the scope of this thesis some key sensitivity parameters have been investigated in this section. Some of the parameters investigated are purely oriented as to how different spline designs affect the stress levels in the coupling and some are oriented towards how different settings in the discretization and simulation setup affect the accuracy of the result.

6.1 Element analysis

In this section a comparison of the different element types available in the program are compared in terms of precision and computational effort. The quantity of interest studied is the maximum principal stress and the effective *Von Mises* stress in the root radius (trochoid) against the computational time. The second order element type (C3D20) is the element that should theoretically converge to the exact solution first and is therefore used as the reference point in this analysis. For this test the same spline geometry, mesh and loading parameters have been tested for all different element types and the results are shown in the figures below. Figure 93 through 97 shows the effective *Von Mises* and maximum principal stress for the different element types. For all contour plots showing the *Von Mises* stress the upper limit is set to 1000 MPa and for the maximum principal stress contour plots the upper limit is set to 1150 Mpa. Furthermore as can be seen from the contour plots the first element row on the involute profile has been removed in order to remove the high contact stresses from the plots.

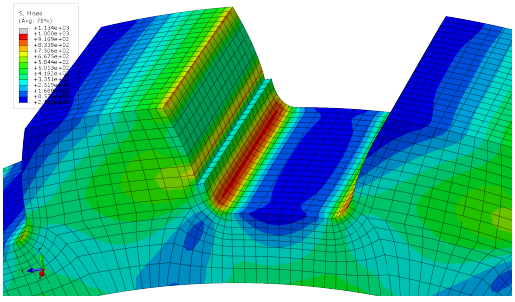


(a) Effective *Von Mises* stress

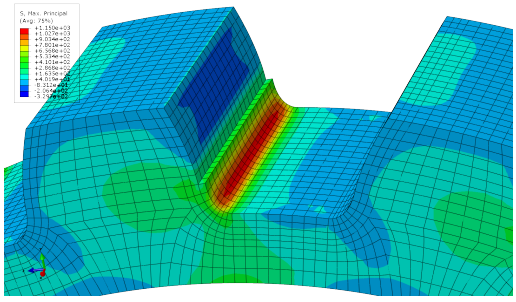


(b) Maximum principal stress

Figure 93: 2nd order elements (C3D20)

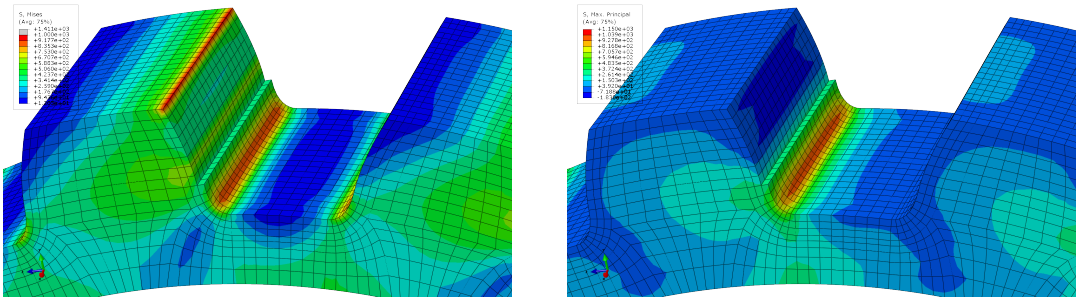


(a) Effective *Von Mises* stress



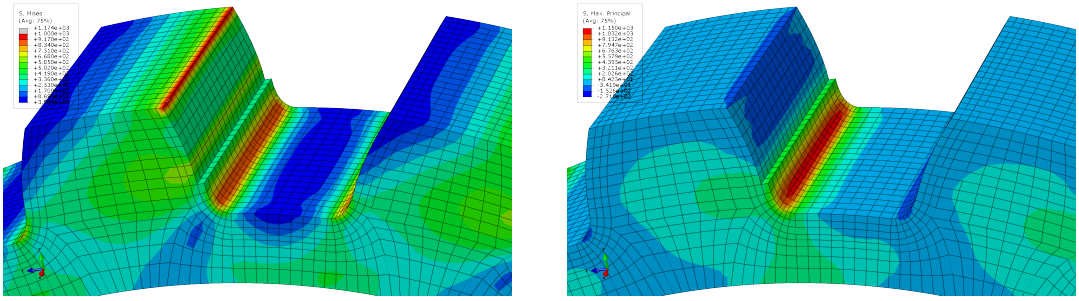
(b) Maximum principal stress

Figure 94: 2nd order elements with reduced integration (C3D20R)



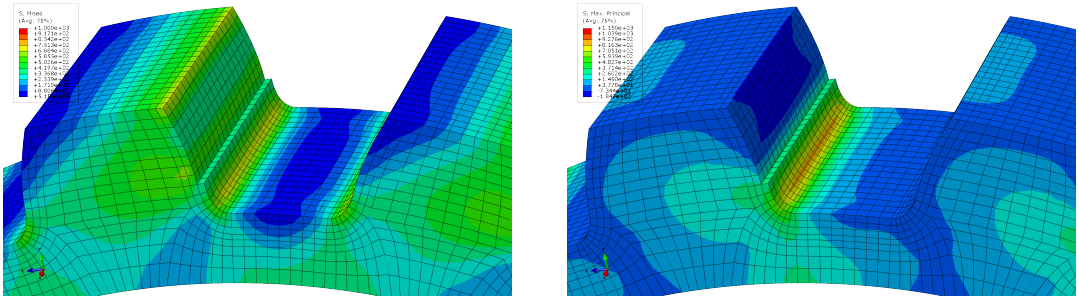
(a) Effective *Von Mises* stress (b) Maximum principal stress

Figure 95: 1st order elements (C3D8)



(a) Effective *Von Mises* stress (b) Maximum principal stress

Figure 96: 1st order incompatible modes elements (C3D8I)



(a) Effective *Von Mises* stress (b) Maximum principal stress

Figure 97: 1st order elements with reduced integration (C3D8R)

By studying the contour plots above it seems as if all element types underestimate the stress levels to some extent compared to the regular 2nd order elements for both types of stress measures. In order to show this in a more quantifiable way the maximum stress levels for both types of stresses have been extracted from the root set and are shown in Table 6. As can be seen the maximum stress shows the same results as the contour plots above, that all element types underestimate the stress levels. Worth noting is that for *Von Mises* stress the C3D20R elements gives the best approximation, but for maximum principal stress the C3D8I elements give the closest approximation, although not by much. This is a bit unexpected since the C3D20R have shape functions of a higher degree and should theoretically converge faster. On the other hand are bending stresses dominant in the root and the C3D8I elements are supposed to be particularly good at capturing bending stresses[11].

Table 6: Maximum stress in root radius (trochoid)

Element type	*Von Mises* [MPa]	Max. principal [MPa]
C3D20	969.1	1137.0
C3D20R	946.3	1083.0
C3D8	879.6	967.3
C3D8I	905.5	1092.0
C3D8R	688.0	837.5

The other measure of this analysis which is probably the most important is the computational effort needed in order to solve the system of equations for the different element types. This is quantified as the time needed solve the simulation with only the element type differing. In Table 7 beneath the time needed for all element types are shown and as can be seen the big time difference seems to be between the polynomial degree whereas the number of integration points seem to have less impact on the time parameter, at least for an analysis of this size.

Table 7: Computing time for the different element types

Element type	Time
C3D20	39min 19s
C3D20R	36min 37s
C3D8	3min 44s
C3D8I	4min 29s
C3D8R	2min 56s

In conclusion the second order elements with reduced integration seem to give the most accurate result although with marginal computational savings compared to ordinary second order elements. The incompatible modes elements yield the best accuracy for the first order elements which seem reasonable based on the fact that they are designed to handle bending stresses well according to *Abaqus*[11]. The first order elements solves the problem by a factor of around 10 times quicker than the second order elements which can be considered as a lot based on the fact that the spline coupling used in this study is relatively small. With this in mind it is possible that the C3D8I elements can outperform the second order element types since a much finer mesh can be used at a lower computational cost. This is investigated further in Section 6.6.

6.2 Tool tip radius

The first geometrical parameter that has been investigated is how the chosen tool tip radius for the manufacturing of an external spline affects the stress levels in the root of the external spline. The primary quantity of interest that has been investigated is the stress levels in the root of the external spline and how it varies with a varying tool tip radius (effectively the root radius of the tooth). Besides giving information regarding the stress levels for the spline this study also shows how the predefined node and element sets that are implemented in order to aid in post processing works in practice. The geometrical data for the spline coupling used in this study is shown in Table 8 and the simulation setup parameters are shown in Table 9.

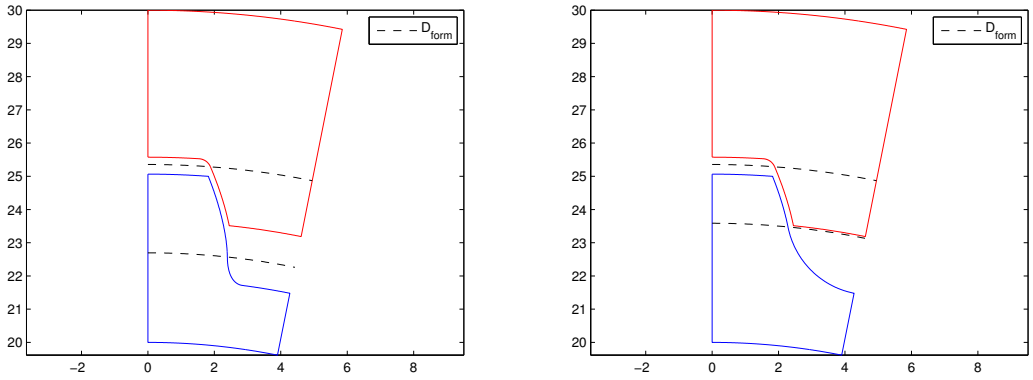
Table 8: Geometric spline data

Dimension	Internal / External
m	3
z	16
α_0	20°
D_{ii} / D_{ie}	47.28 mm / 43.8 mm
D_{ei} / D_{ee}	51.15 mm / 50.83 mm
Tool tip radius	0.4 mm / Variable
D_{outer} / D_{inner}	60 mm / 40 mm
Space width (E) /Space thickness (S)	4.733 mm / 4.375 mm
Tip chamfer (Y_i)	0 mm / 0.35 mm
Length	10 mm / 10 mm

Table 9: Simulation setup parameters

Young's modulus	210 GPa
Poisson's ratio	0.3
Coupling type	Kinematic
Driven part	Internal
Torque	1000 Nm
Element type	C3D20

The data for this spline has been chosen in such a way that it allows for a big variation in tool tip radius for the external spline while still maintaining a correct contact patch between the internal and external spline. For this study simulations of the same spline coupling with a tool tip radius varying from 0.25mm to a full radius corresponding to roughly 2.5mm in steps of 0.25mm have been preformed. Figure 98 beneath shows the two extremes of the external spline in terms of tool tip radius plotted with the geometry preview tool built in to the program.



(a) Tool tip radius = 0.25 mm (b) Tool tip radius = full radius

Figure 98: Varying root radius

The quantity of interest considered in this study is the effective *Von Mises* stress on the surface of the root where the root boundary has been set to 1 element from the start of the involute (D_{form}). The root *Von Mises* surface stress can then easily be extracted from *Abaqus viewer* with the help of the predefined default node set *External_spline.root_nodes_1* which is shown in Figure 99.

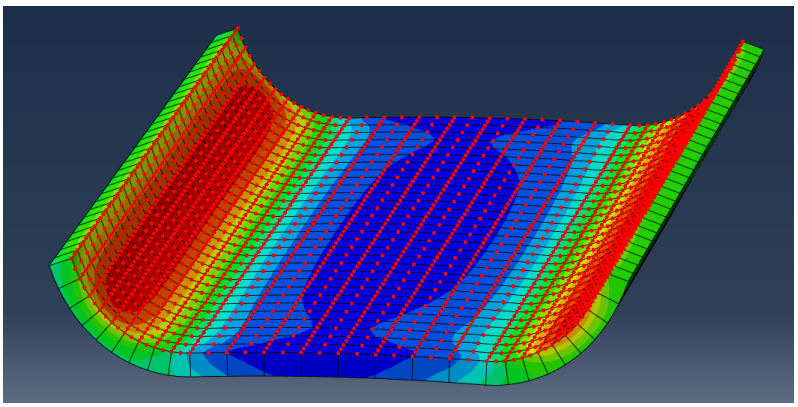
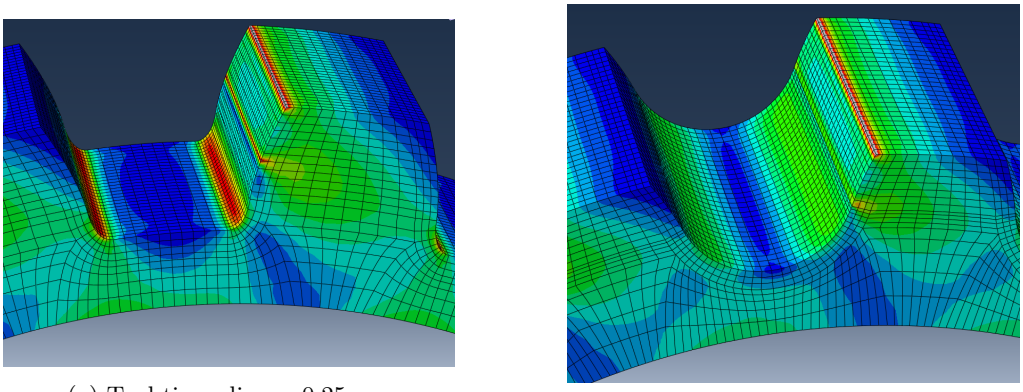


Figure 99: The node set *External_spline.root_nodes_1*

Figure 100 shows a comparison of the stress distribution in the spline tooth between the smallest tool tip radius (0.25mm) and the full radius spline (2.5mm) where the maximum limit has been set to 280 MPa.



(a) Tool tip radius = 0.25 mm

(b) Tool tip radius = full radius

Figure 100: Stress distribution in the external spline tooth

Figure 101 shows the maximum effective *Von Mises* extracted from the surface nodes of the spline root as a function of the tool tip radius. As can be seen from the graph the stress levels in the root decrease with an increasing tool tip radius which is expected, although worth noting is the smoothness of the curve. As mentioned in the beginning of this chapter this study is only meant to show the effect the tool tip radius has on the stress levels in the root. In reality this parameter would be set as a trade-off primarily between bending stress levels in the root and contact stress levels on the involute, otherwise all spline couplings would be of the full radius type.

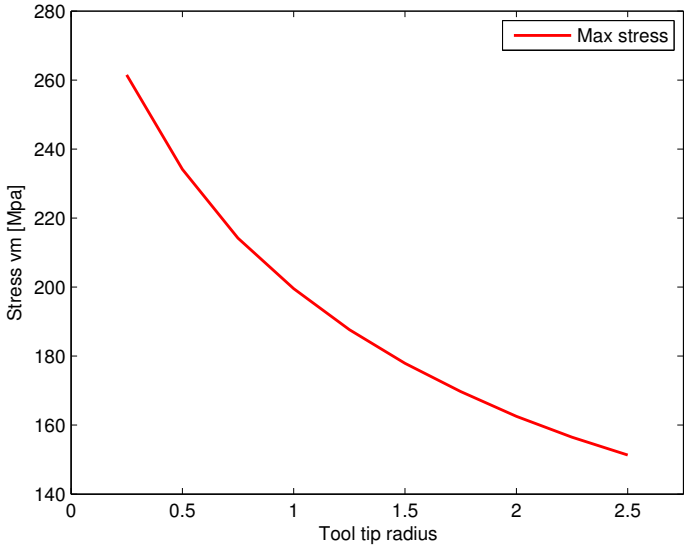


Figure 101: Stress levels on the surface of the root as a function of the tool tip radius

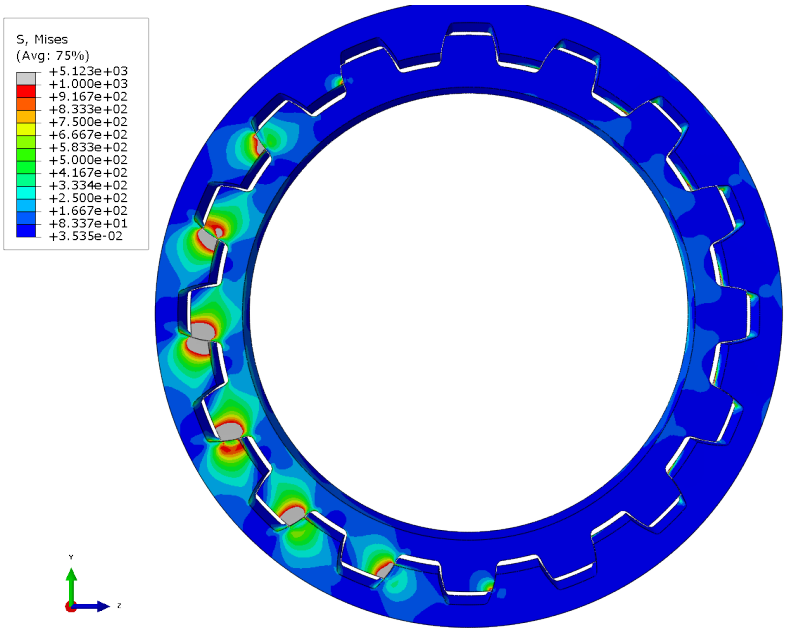
6.3 Coupling effect on misalignment

As mentioned under Section 4.2 the user can choose between two types of couplings from which the boundary conditions are transferred to the boundary surfaces. This Section covers the effect the coupling type can have on the results when running a 2-part standalone type simulation and is thus only valid for that simulation type. For a 2-part setup one of these couplings is attached to the outer boundary of the internal spline and one is attached to the inner boundary of the external spline. The two types of couplings that are available in the program are *kinematic* couplings and *distributing* couplings. The coupling consists of a "master" node located in the centre of the spline that is connected to a node set consisting of slave nodes pertinent to the surface of the boundary. The difference between these two coupling types lays in the kinematic relationship between the master node and the slave nodes. The kinematic coupling enforces a strict relationship between the master and slave nodes where the degrees of freedom of the slave nodes are removed and the slave nodes are only allowed to move with the rigid body motion of the master node, this also means that the slave nodes are not allowed to move relative to each other. This could be envisioned as having an axle with rigid body properties connected to the splined parts. The distributing type coupling works in a way where the average of the distributed loads at the slave nodes fulfills the equilibrium condition of the applied load at the master node and thus are the slave nodes also allowed to move relative each other to a certain degree. With this definition the distributing coupling could be considered as less stiff in its behaviour compared to the kinematic coupling. This can have a significant impact on the results when performing a misalignment study for a standalone type simulation since the stiffness of the rest of the assembly will not be included in the analysis. The only way of controlling the stiffness of the system in order to emulate the actual implementation of the spline coupling is by altering the radial thickness of the parts and by selecting the appropriate coupling. In order to investigate the effects of these parameters on the result of a misaligned spline coupling a study of two different geometries each tested with both types of couplings have been performed. The input data for the spline couplings are given in Table 10 underneath with the only difference among the two different variants being the radial thickness of the parts. The mesh parameters of the parts have been kept as consistent as possible with the only difference being the number of elements in the radial direction in order to get a similar mesh size for the entirety of the parts.

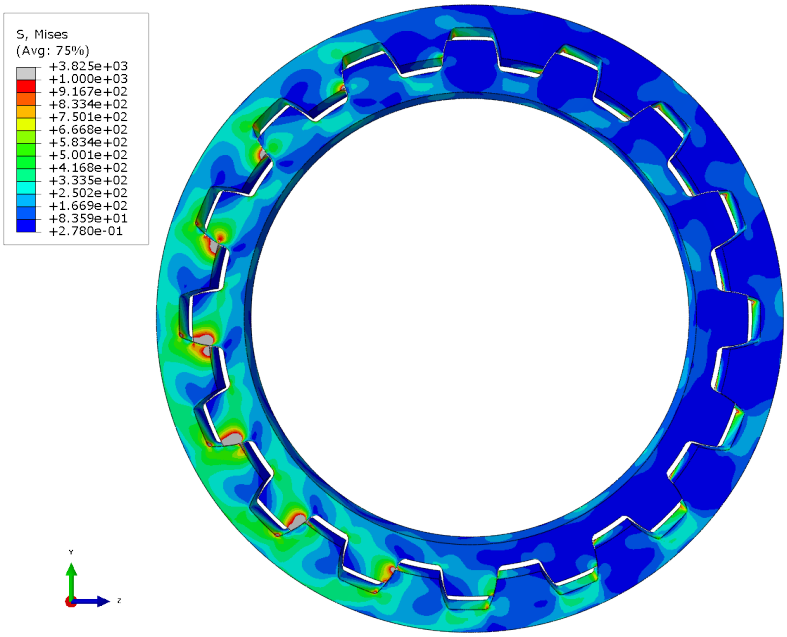
Table 10: Geometric input data

Dimension	Internal	External
z	16	16
m	3	3
α_0	20 °	20 °
D_{ii}/D_{ie}	46.08 mm	43.8 mm
D_{ei}/D_{ee}	51.15 mm	49.63 mm
Tool tip radius	0.4 mm	0.4 mm
D_{outer}/D_{inner}	61.15 mm/55.15 mm	33.8 mm/39.8 mm
E/S	4.733 mm	4.375 mm
Y_i/Y_e	0 mm	0.35 mm
Length	5 mm	5 mm

Figure 102 shows the resulting *Von Mises* stresses for the two different coupling types for the thin walled spline coupling, the upper figure shows the result for the kinematic coupling and the lower figure for the distributing coupling. The applied torque are for both simulations 500 Nm and the misalignment is set to 1°. In the post processing step the maximum limit of the legend is set to 1000 MPa in order to filter out the singularities that occur in the contact surfaces due to the misalignment. As can be seen in Figure 102 the theory that the kinematic coupling will yield a stiffer response seems to be accurate.



Kinematic coupling

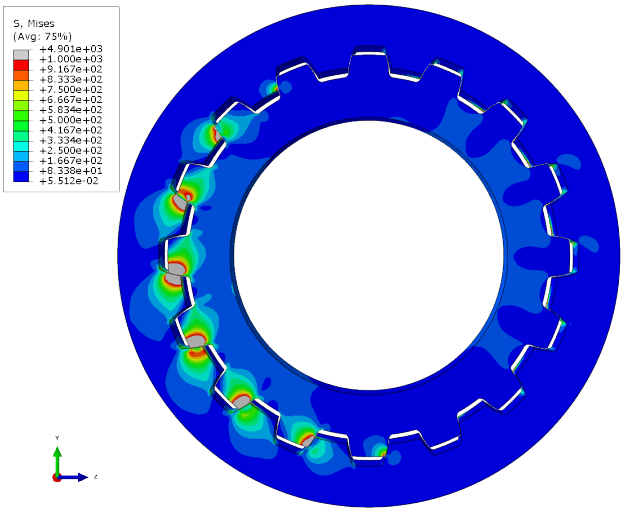


Distributing coupling

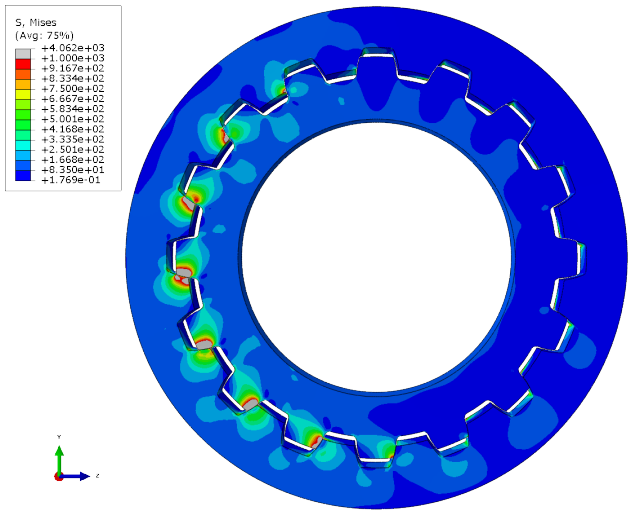
Figure 102: Stress distribution for the thin walled spline coupling

Figure 103 shows the resulting *Von mises* stress distribution for the thick walled spline coupling with the same torque and misalignment input values, as well as the same maximum limit set on the legend. As expected the influence of the coupling type is not as prominent in this case since the overall stiffness of the system is lower compared to the thin walled version. It is still possible to see a difference in the distribution of the stress between the two though. It is important to keep in mind that the misalignment is in the program defined as a rotation about the Z-axis and the figures are created in the ZY-plane. This means that the initial contact will be established at the leftmost tooth on this side and analogously the rightmost tooth will come in contact first on the back side. Studying Figure 103 while bearing this in mind it can be seen that the increased

stress levels on the right side of the spline coupling for the kinematic coupling stems from a lower contact ratio resulting in higher stress concentrations on the backside that is propagating to the front side, rather than from contact occurring in the front plane.



Kinematic coupling



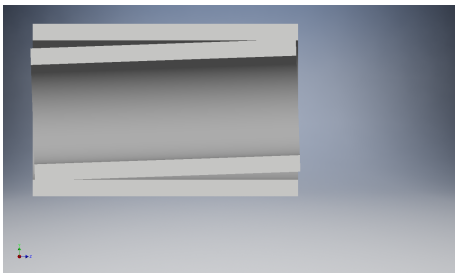
Distributing coupling

Figure 103: Stress distribution for the thick walled spline coupling

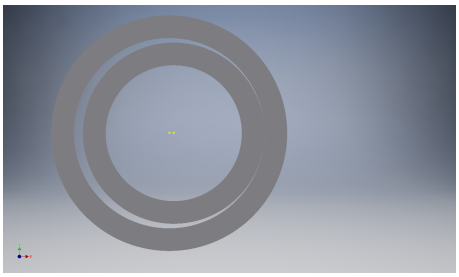
In conclusion it seems like the type of coupling used in a standalone type study that involves either an angular misalignment or an eccentric misalignment can have a relatively large impact on the end result. This will of course not become a problem when running simulations of the shaft assembly type where the stiffness of the setup is defined by parts included in the model. As for the standalone type simulations the difference between the coupling types creates some challenges especially as *Abaqus* limits the number of slave nodes for a distributing type coupling to 21845 nodes. This means that the distributing type coupling can not be used for applying boundary conditions when the size of the problem becomes to large. In this case the user can either run a shaft assembly type simulation instead or as shown above modify the stiffness response of the coupling with the radial thickness of the parts if possible.

6.4 Misalignment study

The program can be used to investigate the effects of two types of misalignment, both angular misalignment where the two parts engage each other at an angle that is not 0 and eccentric misalignment where the centre axis of the two parts are not coincident. Figure 104 shows an exaggerated illustration of the two types of misalignment with the spline coupling replaced by two cylinders. In order to demonstrate the capability of the program a study of the stress levels and distribution for varying degrees of both types of misalignment have been performed. The study also to some extend helps to show how the predefined sets and build-up of the mesh algorithm aids in the actual post processing.



(a) Angular misalignment



(b) Eccentric misalignment

Figure 104: Types of misalignment

6.4.1 Angular misalignment

The first parameter that has been investigated is the effect of an angular misalignment in the spline coupling. For the study a spline coupling with the geometric properties shown in Table 11 have been used with the setup parameters and boundary conditions as defined in Table 12.

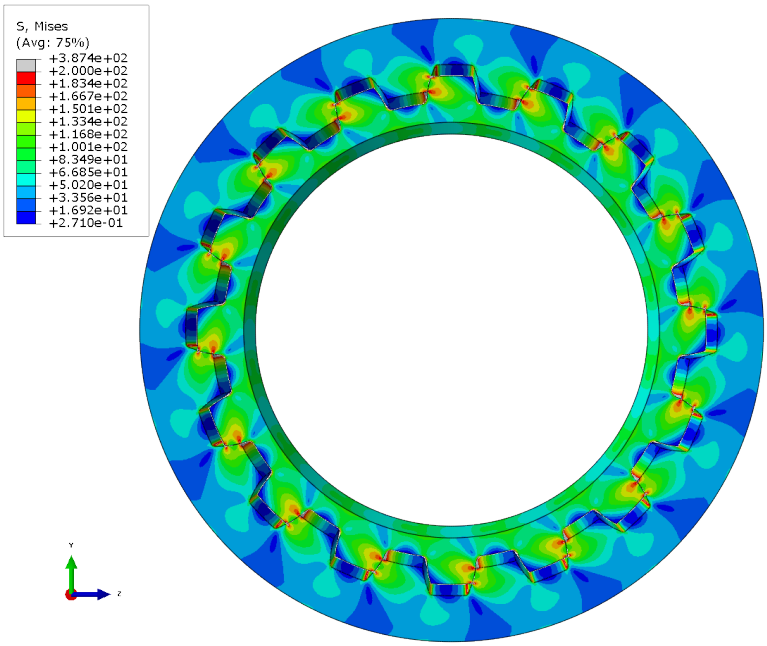
Table 11: Geometric input data

Dimension	Internal	External
z	16	16
m	3	3
α_0	20 °	20 °
D_{ii}/D_{ie}	46.08 mm	43.8 mm
D_{ei}/D_{ee}	51.15 mm	49.63 mm
Tool tip radius	0.4 mm	0.4 mm
D_{outer}/D_{inner}	60 mm	40 mm
E/S	4.75 mm	4.25 mm
Y_i/Y_e	0 mm	0.35 mm
Length	10 mm	10 mm

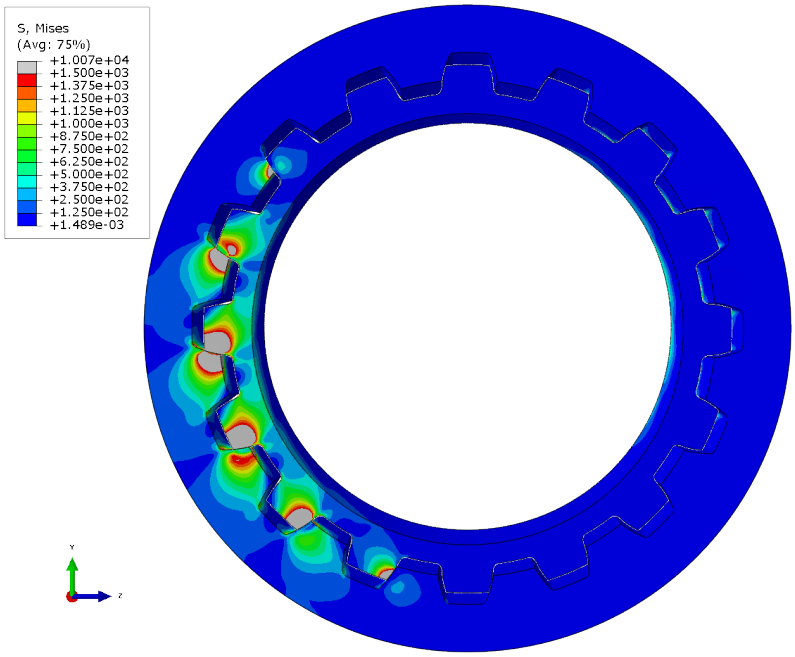
Table 12: Setup parameters

Coupling type	Kinematic
Driven spline	Internal
Torque	1000 Nm
Element type	C3D20
Young's modulus	210 GPa
Poisson's ratio	0.3

The angular misalignment have been varied from 0 degrees to 1.5 degrees in steps of 0.25 degrees. The quantity of interest that have been investigated in the study is primarily the root stress in the middle of the trochoid as a function of the coordinate stretching along the root for the root with the highest stress levels, i.e the root of the leftmost tooth. The study also includes some stress plots in order to visualize how the stress distribution changes with the misalignment. The results shown in this section can not be seen as general in the sense that they apply to a wide number of different splines. This is due to the fact that the overall stiffness of the model has a large impact on the resulting stress levels which is dependent on the geometry of the spline. In reality a misalignment study would have to be done as an shaft assembly with the boundary conditions representing the physical model fully in order get results that apply to the actual part. The results of this study can therefore be seen as accurate for this particular model, but not applicable to spline couplings with either different geometry or different overall stiffness. Figure 105 shows how a misalignment affect the stress distribution for the entire coupling. The figure shows the two extremes, one without misalignment and one where the misalignment is set to 1.5 degrees, the effect this has on the contact ratio and thus also the stress distribution between the teeth. Worth noting is that in Figure 105 the upper limit is set to 200 MPa for the non misaligned coupling and 1500 MPa for the misaligned coupling, which shows the large impact of the edge effects created by the misalignment.



0°



1.5°

Figure 105: Effect of misalignment for the stress distribution in the coupling

Since the misalignment is defined as a rotation about the Z-axis the maximum stress in the YZ-plane (as shown in Figure 105) will occur at the leftmost tooth of the spline coupling. Figure 106 shows how the stress distribution varies in the spline root of the external spline for three different angles.

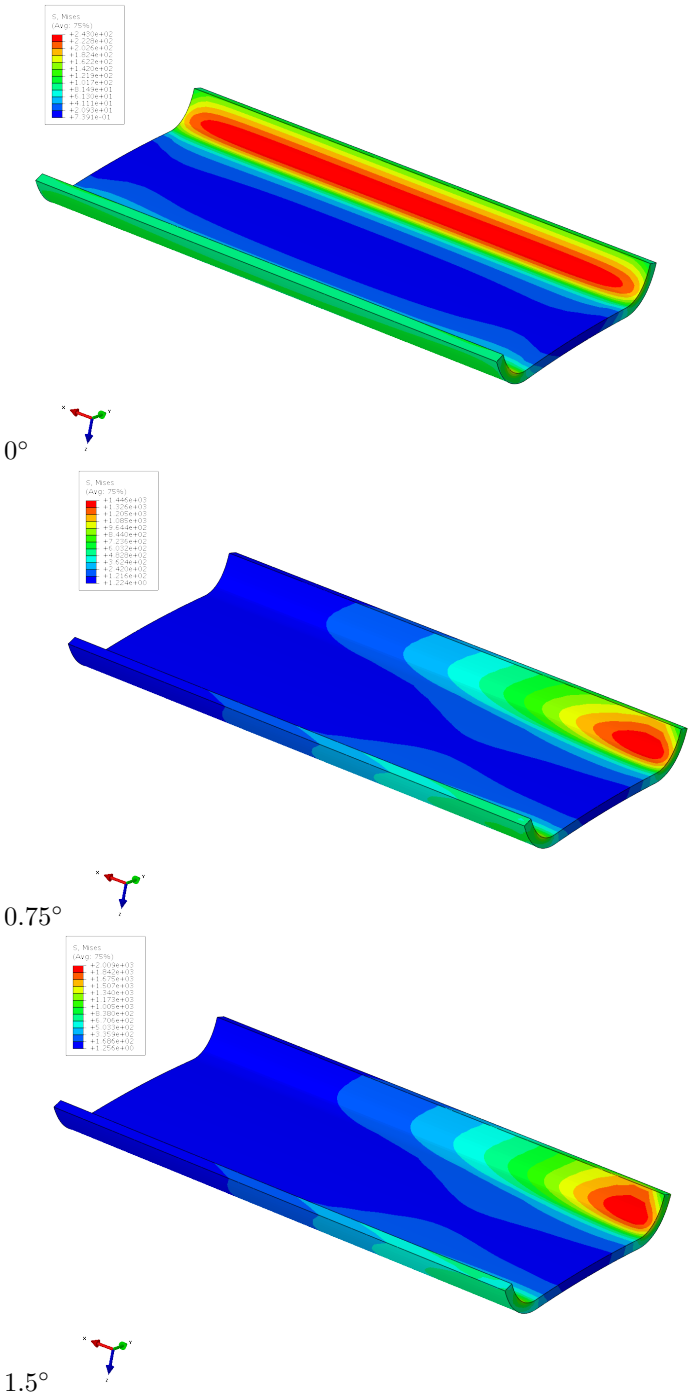


Figure 106: Stress distribution in spline root due to misalignment

To better visualize the impact the degree of misalignment has on the stress distribution in the root a path has been created that stretches over the maximum stress in the trochoid according to Figure 107. This path have been recreated for all simulations and the *Von Mises* stresses have been extracted and post processed in *MATLAB*. The result of the study is shown in Figure 108, where the stress has been plotted as a function of the coordinate along the root for varying degrees of misalignment.

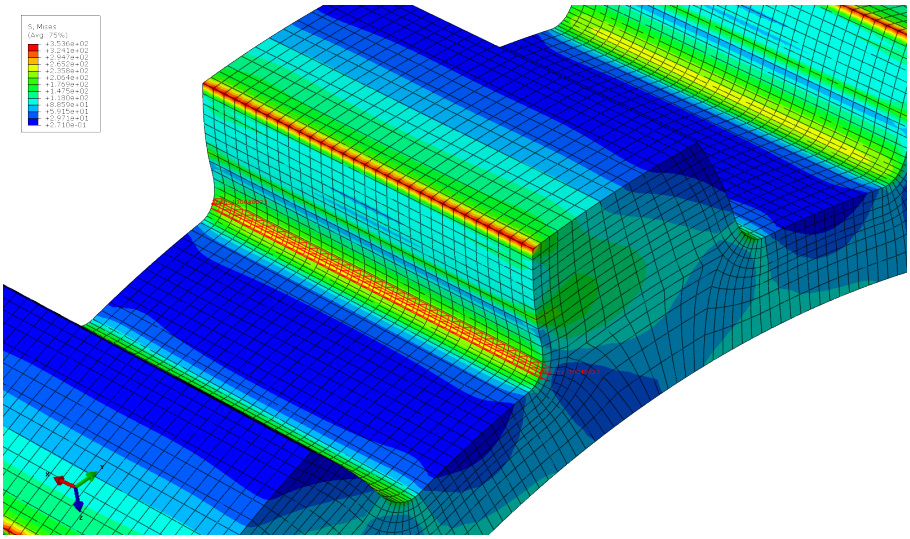


Figure 107: Path from which stresses have been extracted

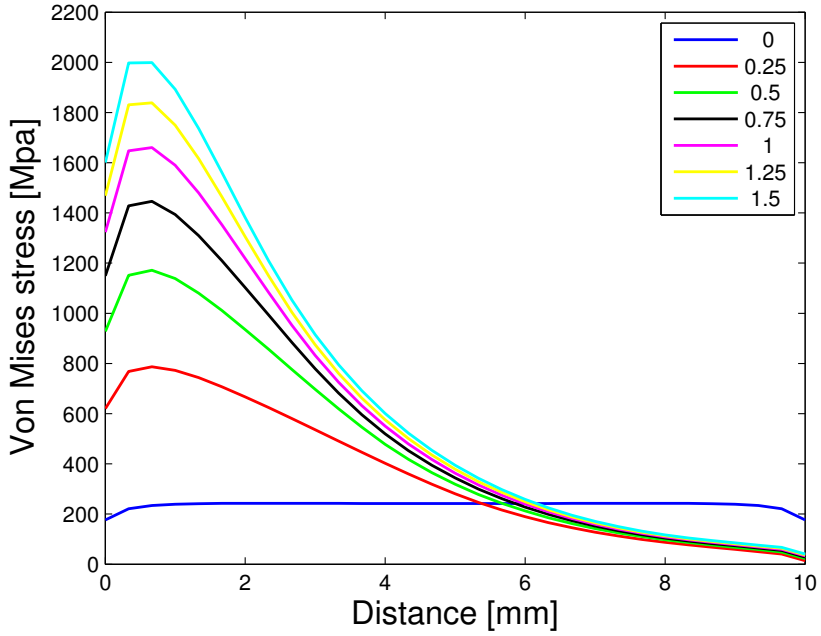
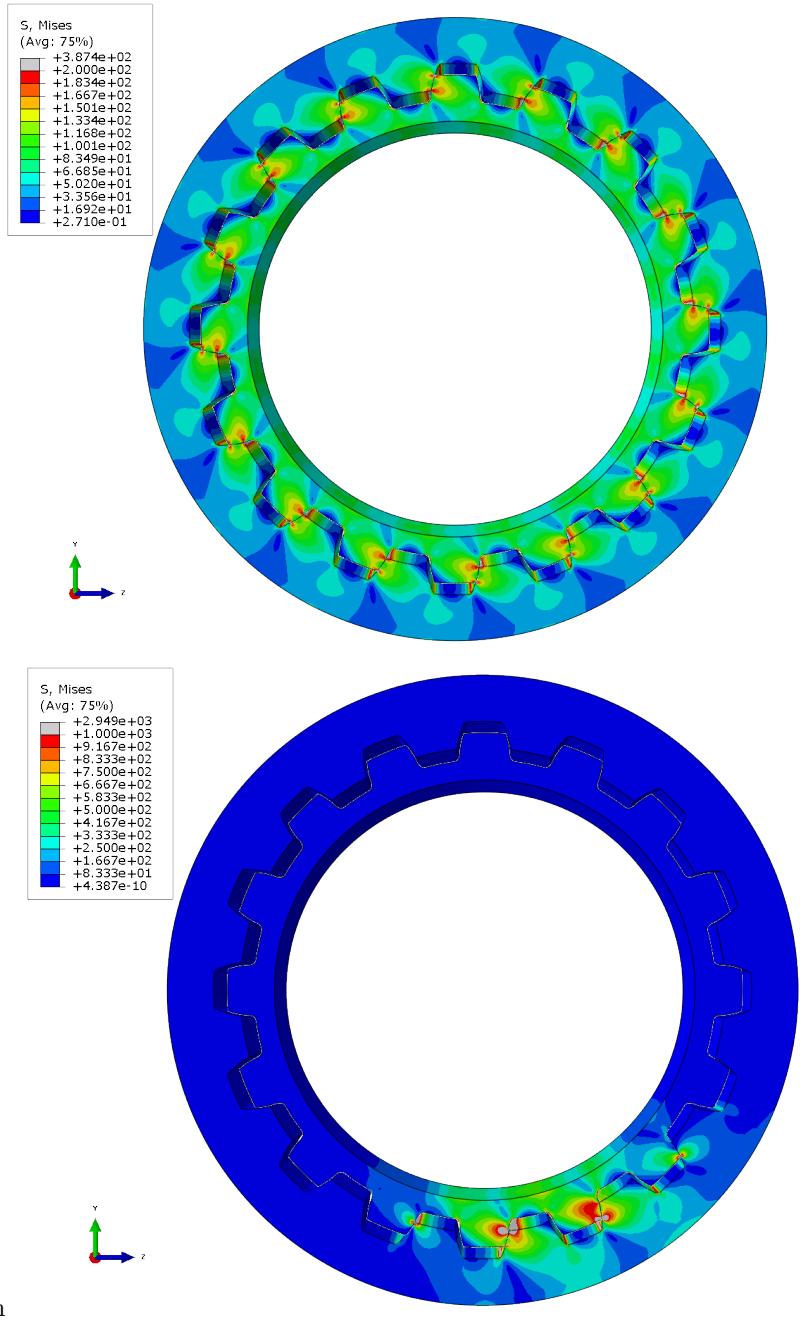


Figure 108: *Von Mises* stress as a function of the coordinate along the root

As can be seen from Figure 108 the maximum stress in the root increases significantly already at a small misalignment in the coupling. This is most likely due to the fact the model used in this study has a very high overall stiffness where the contact ratio decreases considerably already at a small angle of misalignment. This means that the leftmost tooth from which the stresses in Figure 108 are extracted will already at small angle take up a large part of the applied torque. More interesting is the pattern between the different angles which seems to be rather consistent with a similar stress at the non engaged side, a stress curve very similar to a second order polynomial of varying degree which then decreases close to the free boundary.

6.4.2 Eccentric misalignment

The second type of assembly deviation that can be studied with the program is an eccentric mounting as shown in Figure 104. This has been implemented by moving the external spline in the Z-direction relative the internal spline and thus creating an offset between the two centre axes of the parts. The study has been performed with the same spline coupling used in the angular misalignment defined according to Table 11 and with the same setup and loading parameters as shown in Table 12. Figure 109 shows the two extremes of the study, one coupling without an offset and one coupling with an offset of 0.2 mm.

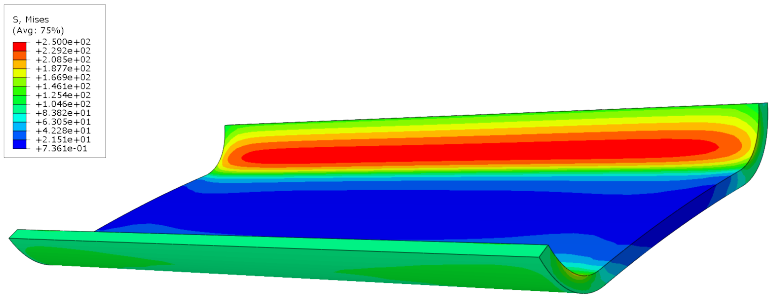


0 mm

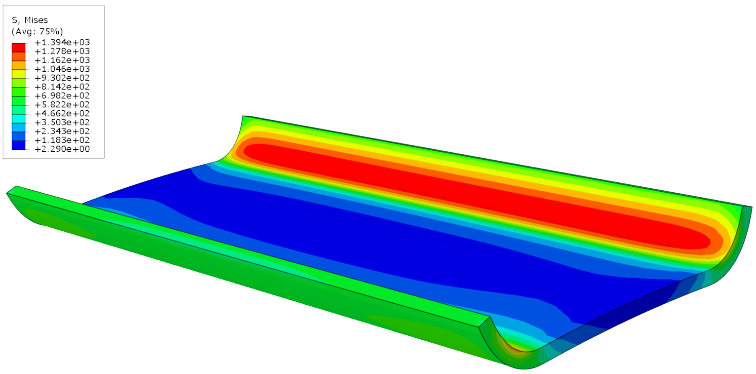
0.2 mm

Figure 109: Effect of eccentricity for the stress distribution in the coupling

Similar to the angular misalignment a spline coupling that is mounted eccentric will result in a lower contact ratio with fewer teeth taking up the load, but in contrast to the angular misalignment the contact pressure in the XY-plane and XZ-plane will theoretically be constant. The root stress as a function of the coordinate normal to the root, as plotted for the angular misalignment in Figure 108, should thereby not be affected by the eccentricity apart from the magnitude. Figure 110 shows a contour plot of the stress distribution in the root for a concentric spline coupling and a coupling with a centre axis offset of 0.2 mm. In Figure 111 the stress in the middle of the trochoid is plotted as a function of the distance parallel to the root for the root with the highest stress level, in the same way as for the angular misalignment. As expected the stress across the root does not change in appearance, but only in magnitude due to the eccentricity.



0 mm



0.2 mm

Figure 110: Effect of eccentricity for the stress distribution in the root

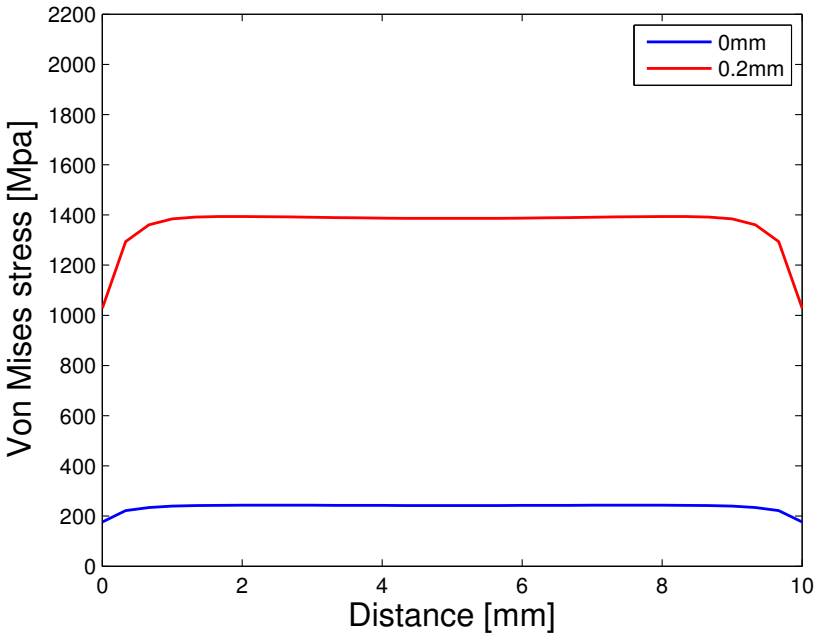


Figure 111: *Von Mises* stress as a function of the coordinate along the root

As can be seen in Figure 111 the stress across the root is not of much interest when studying the effects of eccentricity instead the focus of this study has been put on investigating the stress distribution between the teeth as a function of the deviation. In order to do so the surface *Von Mises* stress has been extracted for each root separately for all simulations using the predefined node sets. In Figure 112 the maximum and mean *Von Mises* stress is plotted for the root with the highest stress level. As can be seen from the figure both stresses increases quite rapidly already at a small eccentricity. The reason for this is most likely due to the same reasons as for the angular misalignment since the spline coupling used is the same, which is that the overall stiffness of the model is high. This kind of response is probably not very common in reality, but the study shows the kind of analyses that can be preformed with the program.

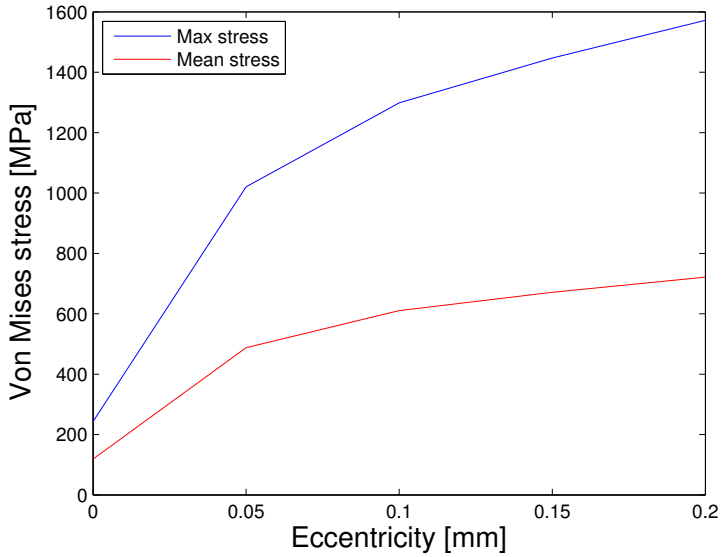


Figure 112: *Von Mises* stress as a function of eccentricity

By using a *Python* script the node sets pertaining to each root of the spline can be extracted easily from *Abaqus* as separate *.txt* files, which can then be loaded into *MATLAB* for effective post processing. With the data in *MATLAB* there is a lot of different visualization methods that can be used in order to show the effects of different manufacturing and assembly deviations. Figure 113 shows a plot where the maximum stress for each root of the external spline is plotted. The circumferential ϕ coordinate represents the position of each root and the radial coordinate, i.e the vector stretching from the origin to the dot illustrating the root, represents the maximum *Von Mises* stress in the root in GPa. In Figure 114 an example is shown where the stress in root 3 for the concentric coupling is defined by the length of the vector a and the stress in root 8 of the coupling with 0.005 mm sinusoidal deviation is given by the length of vector b. As can be seen from Figure 113 the stress in each root is evenly distributed when there is no eccentricity between the two parts (blue dots) and as the eccentricity increases the stress is diverted more and more to the roots at the bottom of the spline coupling in the YZ-plane.

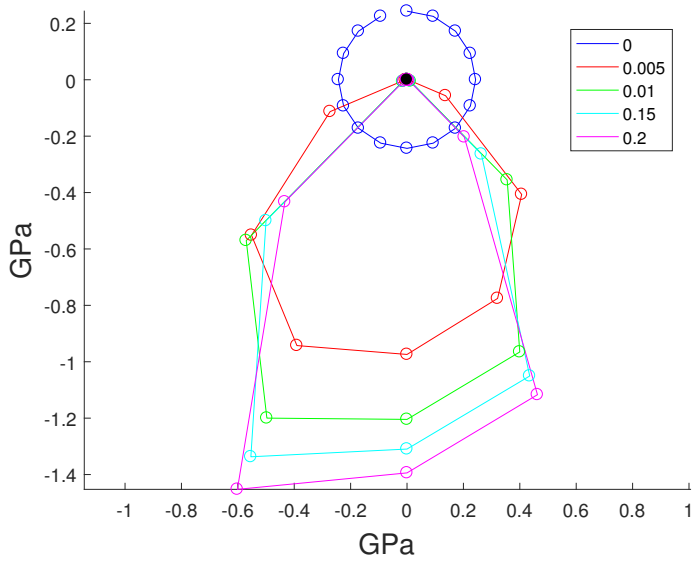


Figure 113: *Von Mises* stress for each root as a function of eccentricity

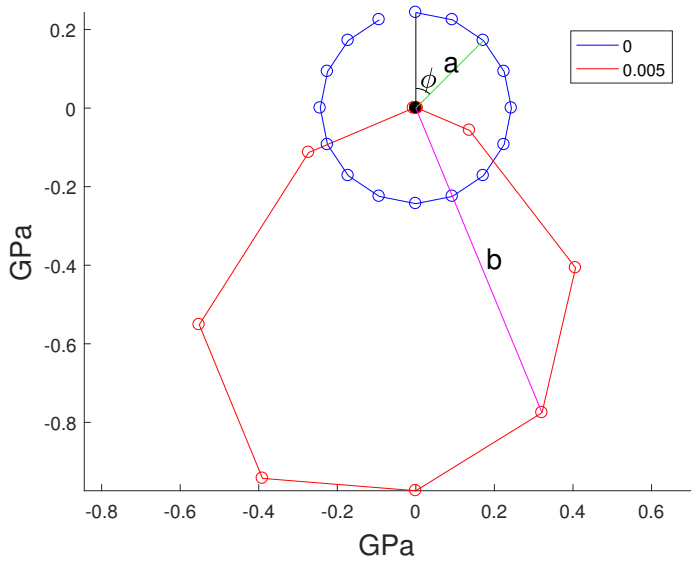


Figure 114: *Von Mises* stress for each root as a function of eccentricity

6.5 Manufacturing tolerances

In order to be able to replicate the effects of manufacturing tolerances in the program the *Abaqus* command *clearance* has been incorporated in the program. This command is used in order to offset the gap function used by the iterative solver for the contacts of the spline coupling. The command can be used to specify either an additional offset between the master and slave nodes for a contact surface or allow an additional penetration without adding any penalty stiffness to the stiffness matrix. Since the program predefines each involute surface separately this parameter can be used individually for each tooth in order to replicate a deviation in the thickness/width of a tooth and also to replicate a pitch deviation between the teeth of a splined part. This method will not replicate the geometry of an actual part completely since in reality a pitch deviation would also affect the actual geometry of the spline coupling, but since the deviation is usually very small this will have a negligible effect on the results. The alternative would have been to create the mesh algorithm in such a way that it meshes the parts in their entirety without using any symmetry which would most likely not be worth the small gains in resembling the geometrical deviations fully. In this section a few examples are shown in order to demonstrate this functionality and how manufacturing deviations can affect the stress distrubution in a spline coupling. The actual deviations used in the study are made up , but based on actual tolerances and typical tolerance outcomes. This is of little importance though since the main objective is to test the actual functionality of the program. The geometric data for the spline coupling used is shown in Table 13 beneath together with the setup parameters in Table 14

Table 13: Geometric input data

Dimension	Internal	External
z	28	28
m	4.25	4.25
α_0	20 °	20 °
D_{ii}/D_{ie}	116.3 mm	115.05 mm
D_{ei}/D_{ee}	122.85 mm	121.2 mm
Tool tip radius	0.3 mm	0.4 mm
D_{outer}/D_{inner}	132.85 mm	105.05 mm
E/S	6.706 mm	6.576 mm
Y_i/Y_e	0.2 mm	0.2 mm
Length	10 mm	10 mm

Table 14: Setup parameters

Coupling type	Kinematic
Driven spline	External
Torque	2500 Nm
Element type	C3D20
Young's modulus	210 GPa
Poisson's ratio	0.3

6.5.1 Sinusoidal pitch deviation

Pitch deviation is when the teeth of a spline have a deviation in the circumferential distance between each other, or rather deviates from the theoretical pitch. A usual trend in this type of deviation is that it to some degree has a sinusoidal appearance. In order to replicate this behaviour a deviation has been created based on a sinus wave where the amplitude has been set to a usual manufacturing deviation based on old measuring results as shown in Figure 115.

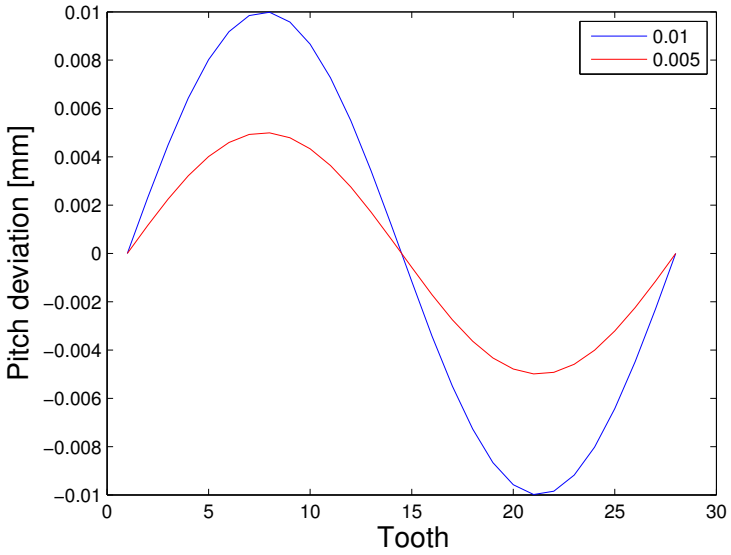


Figure 115: Sinusoidal pitch deviation

As can be seen in Figure 115 the maximum total pitch deviation has been set to 0.005 mm and 0.01 mm respectively. Figure 116, 117 and 118 underneath shows the contour plots of the *Von Mises* stress distribution in the coupling where Figure 116 is the baseline coupling without any deviation.

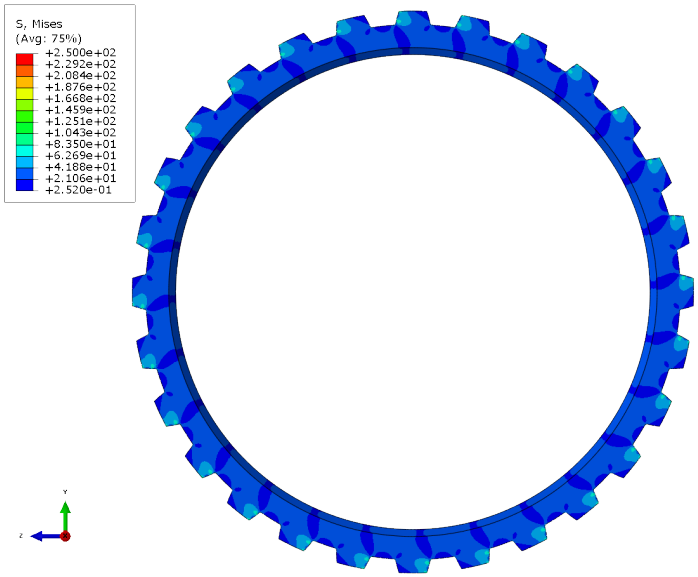


Figure 116: Spline coupling without any deviation

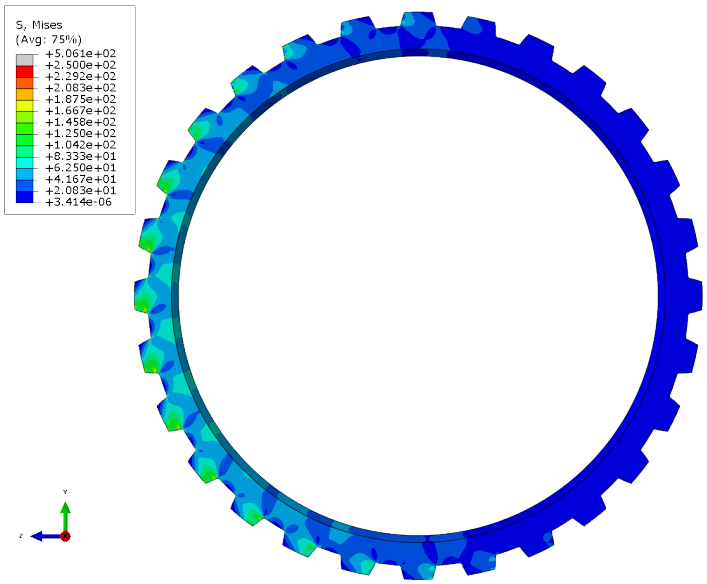


Figure 117: 0.005 mm pitch deviation

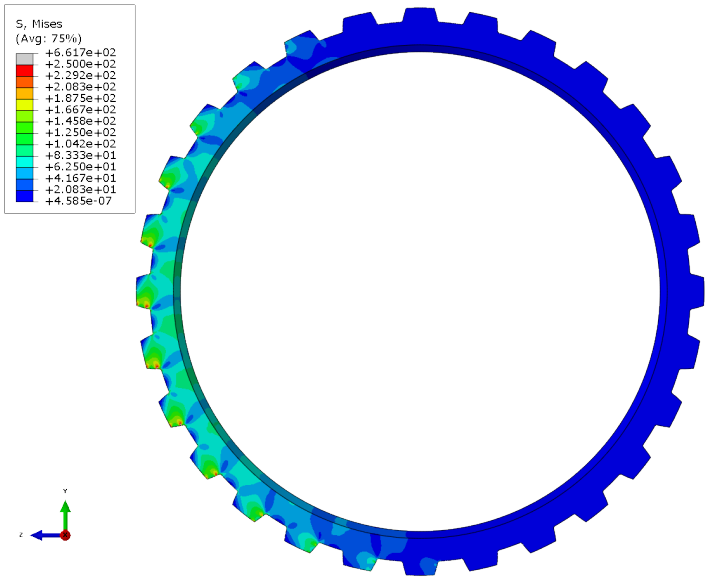


Figure 118: 0.01 mm pitch deviation

The effect of a sinusoidal deviation is as can be seen in the figures above that the load is increasingly shifting to one side of the coupling as the deviation increases. In Figure 119 the effective *Von Mises* stress levels for the root element sets have been extracted and plotted as a polar coordinate system where the circumferential coordinate represents the position of the tooth and the radial coordinate represents the stress in GPa, in the same way as in Figure 114 in Section 6.4.2. The result of a sinusoidal deviation seems to be that the stress and thus the tooth load is shifted quite uniformly towards one side of the coupling.

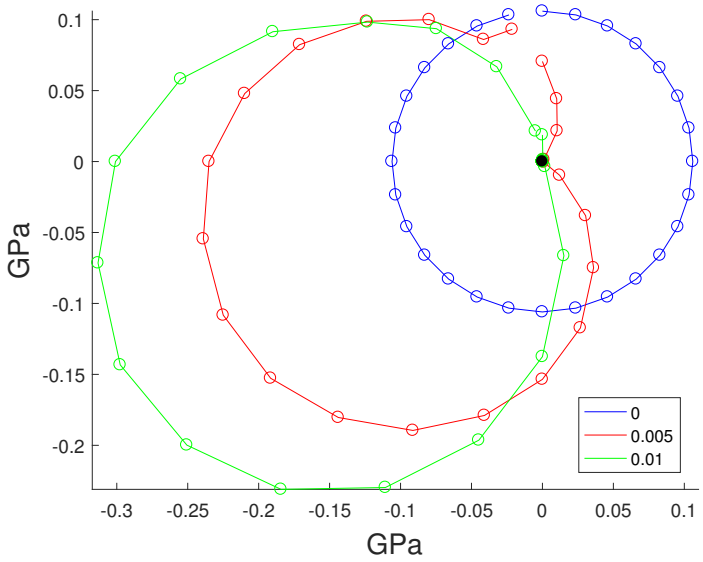


Figure 119: Root stress as a function of deviation

6.5.2 Random pitch deviation

The second test that has been performed is a test where the pitch deviation has been set in a randomized fashion. The same spline coupling with the same parameters as defined in Table 13 and 14 have been used with a maximum total deviation of 0.005 mm distributed with the help of the *MATLAB* function *randi*. The input in terms of deviation is shown for each tooth in Figure 120 beneath.

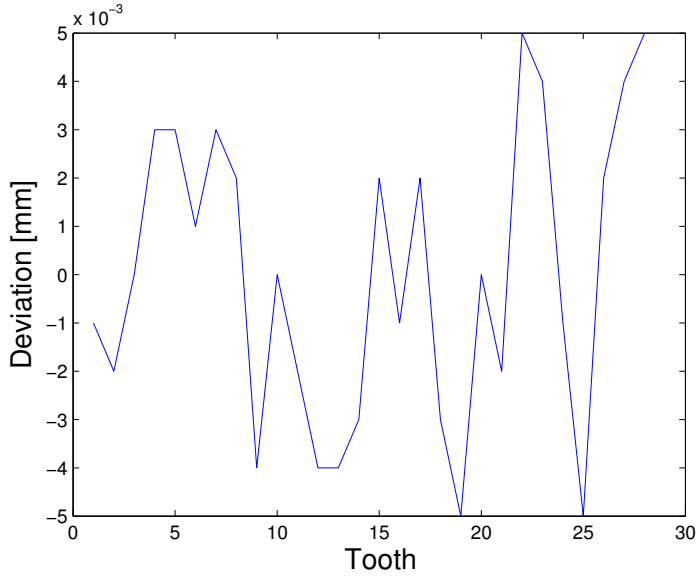


Figure 120: Pitch deviation for each tooth of the coupling

In Figure 121 the effective *Von Mises* stress distribution is shown as a contour plot and it is as expected a lot more irregular compared to the sinusoidal deviation curve. Figure 122 shows the corresponding polar coordinate system plot over the stress distribution in the roots of the coupling and as can be seen it is a lot more chaotic compared to the sinusoidal deviation which is expected with the deviation inputs used.

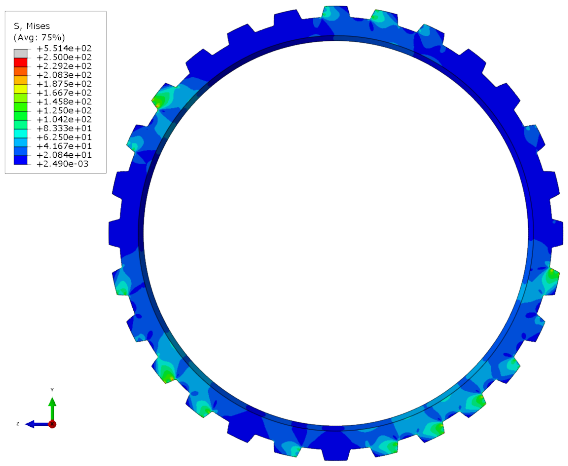


Figure 121: Random deviation with 0.005 mm in amplitude

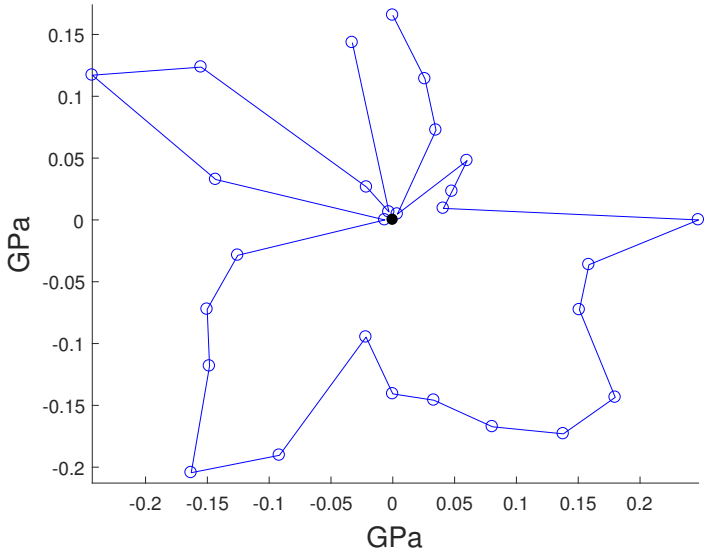


Figure 122: Root stress as a function of deviation

6.6 Convergence study

To get an idea of the convergence rate and behaviour of the different element types available in the program a convergence study has been performed for a 2-part spline coupling with the parameters defined according to Table 15 and 16. In order to keep the study consistent with the other tests performed the quantity of interest in the study has been chosen to be the maximum effective *Von Mises* stress in the root of the external spline. The stress has been normalized according to Equation 50 beneath where the subscript *ok* stands for *overkill* which is the solution derived from a simulation with an extremely fine mesh size compared to the other simulations. For this simulation C3D20 elements have been used and the results from this simulation is used as the substitute for the exact analytic solution which is not available.

$$e = \frac{\sigma_{vm,max,u_h}}{\sigma_{vm,max,u}} \approx \frac{\sigma_{vm,max,u_h}}{\sigma_{vm,max,ok}} \tag{50}$$

Table 15: Geometric input data

Dimension	Internal	External
z	16	16
m	3	3
α_0	20 °	20 °
D_{ii}/D_{ie}	46.08 mm	43.8 mm
D_{ei}/D_{ee}	51.15 mm	49.63 mm
Tool tip radius	0.4 mm	0.4 mm
D_{outer}/D_{inner}	55 mm	40 mm
E/S	4.733 mm	4.375 mm
Y_i/Y_e	0 mm	0.35 mm
Length	10 mm	10 mm

Table 16: Setup parameters

Coupling type	Kinematic
Driven spline	Internal
Torque	1000 Nm
Young's modulus	210 GPa
Poisson's ratio	0.3
Root output (nel from r_fe)	2

The study has been performed for 4 different types of elements, regular 1st and 2nd order elements (C3D8 and C3D20), 1st order incompatible modes elements (C3D8I) and 2nd order elements with reduced number of integration points (C3D20R). These elements have been chosen because they are the ones most likely to perform best for this application according to the element analysis performed in Section 6.1. Since the geometry of the spline is quite complex the element distribution is not as straight forward as it would be for a simpler geometry like a beam. The number of elements used in the different mesh sections in the domain affect the end result in terms of accuracy to varying degrees. With the quantity of interest being the stress in the root radius the mesh sections where the resolution is of most importance in this study should be in the root of the cog and the contact surfaces. For example a model with fine mesh everywhere except in these places would result in a large number of total degrees of freedom but not necessarily an accurate solution. To maintain consistency throughout the simulations the mesh scheme has been set to linear for all sections and starting with a very fine mesh the number of elements for each dimension has then been reduced in the same way for each new simulation. The same mesh size has been used for all

element types, Figure 123 shows the finest mesh used apart from the *overkill* solution, Figure 124 shows an intermediate mesh size and Figure 125 shows the coarsest mesh used in the study.

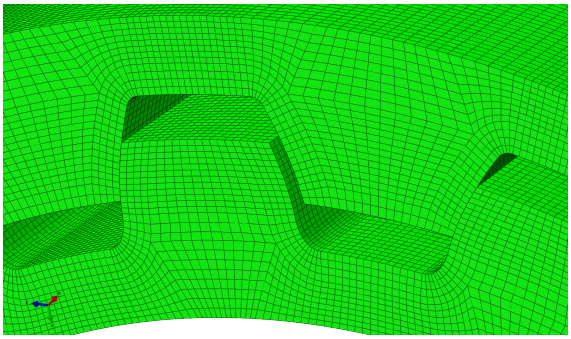


Figure 123: Finest mesh

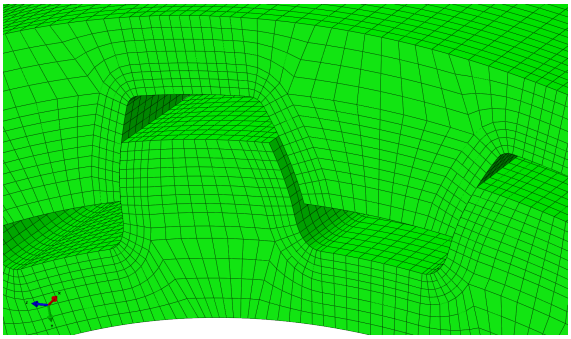


Figure 124: Intermediate mesh

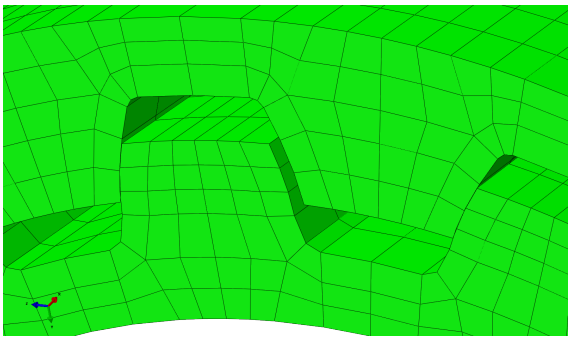


Figure 125: Coarsest mesh

The results of the study are shown in Figure 126 and 127 where the error defined in Equation 50 has been plotted against the total number of degrees of freedom for the model. As can be seen from the figures the maximum stress converges rapidly for all element types and around $2 \cdot 10^6$ degrees of freedom the solution can be considered more or less converged for this model. What is interesting is that both types of 1st order elements as well as the 2nd order elements with reduced integration seem to underestimate the maximum stress to different degrees compared to the regular 2nd order elements even as the refinement of the mesh increases, which was unexpected. All these element types result in a smaller equation system compared to the normal 2nd order elements , which is the reason for including them in the study, and are thus less computationally demanding. Based

on the results obtained the element of choice in the case a less demanding element type is to be used for stress studies in the root the C3D20R elements should be be the first choice. For this setup they resulted in an underestimation of about 3% whereas the C3D8I and C3D8 elements resulted in an underestimation of about 6% and 9% respectively, all values are taken for the last points in Figure 127. If the computational time is of great importance the choice falls on the C3D8I elements which as shown in Section 6.1 are much less time consuming. These results can however not be seen as general for all types of quantities in the entire domain for obvious reasons, but could be used as a guideline when the quantity studied are stress levels in the root which should be transferable to most geometrical configurations.

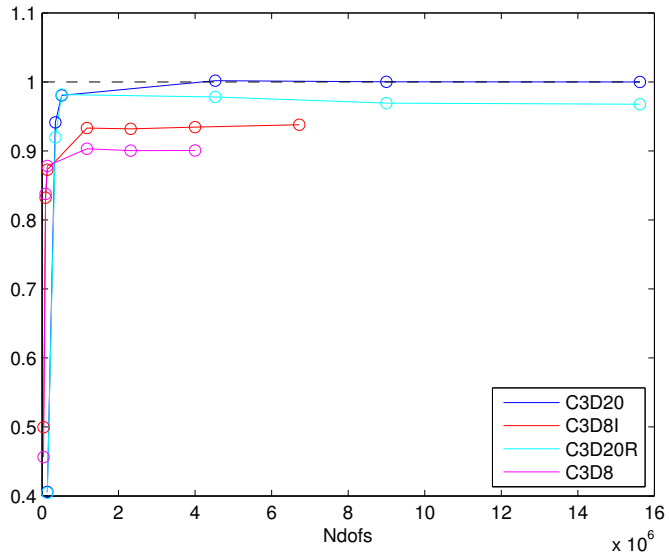


Figure 126: Convergence as a function of number of degrees of freedom

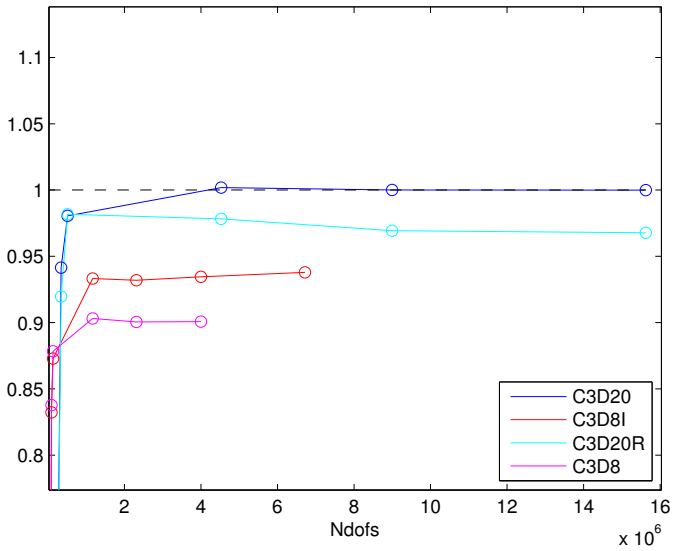


Figure 127: Convergence as a function of number of degrees of freedom

7 Discussion

The program created as a result of this project fulfills the objectives set out in the original scope. The *optimization tool* has been able to create ready to solve FEM discretisized models for all different straight involute spline geometries tested during this project. The program is as explained under Section 4.1 built up by several subprograms and subscripts, the idea of creating the program in this manner was established already at the beginning of the project in order to make it modular. The intention behind this arrangement was to allow for the program to be easily extendable for including other spline geometries. During this project a decision was made between extending the program to include more types of spline geometry features, primarily involute splines that included a back taper angle, or creating a graphical user interface. The decision fell on the latter as can be told from this report with the argument that the original scope in terms of geometries was reached and that a graphical user interface would drastically improve the usability of the program. In hindsight this seems like the right decision since now that the entire structure of the program from graphics to complete models are done new functionalities can be added relatively easy and obtain the advantages that a graphical interface entails.

When it comes to the testing of the program all different functionalities embedded in the program have been tested with satisfying results. The studying of the sensitive parameters in general spline design have not been tested to the extent desired at the initial stages of this project due to a lack of time, which stems from the graphical programming part requiring more work than expected. Most of the sensitivity studies presented in this report are not particularly transferable to real life applications since that demands for a more thorough model including the geometry surrounding the spline coupling. However the simpler studies included in this report shows that the functionality exists in the program as well as the methodology for performing the different types of analyses.

7.1 Recommendations

As mentioned under the discussion part many of the parametric studies intended to be performed during this work where either neglected or simplified due to a lack of time. Since the programs functionality has been verified a first natural step could be to build more realistic models of existing spline couplings in order to further investigate the effects of different geometrical parameters and deviations. A more extensive element type and convergence study should also be performed in order to get a clearer view as how the mesh settings affect the end result.

The post processing of the simulations has been performed in *Abaqus* with the help of a simple *Python* script that is used in order to extract stresses from predefined sets efficiently. The data extracted with the script is then imported to *MATLAB* which is used for visualization. By creating a more advanced *Python* script the post processing could be made much quicker and smoother by eliminating the need to use *Abaqus*. By including communication with the cluster and a feedback loop with the *Optimization tool* the performance could be enhanced even more.

The programs modular buildup allows for more geometric spline parameters to be included, a first step would probably be to implement splines with a back taper angle since this is a feature commonly used today. This can be done by implementing a new function in the mesh algorithms for the extrusion of the elements in the length direction. There are several other parameters that could be included in the program such as a helix angle and a geometrically correct tip chamfer, as of now the program only corrects the diameter with the tip chamfer in order to get the correct contact area. By extending the program to include these parameters more or less all spline couplings could be modeled by the program. This in itself opens up for further expansion of the program to also include involute gears. As mentioned in the report the geometrical definition of involute splines and involute gears are the same which means that the program could be used as a base for creating a similar tool for evaluating involute gears, by modifying the assembly of

the parts. This would probably be a natural step to take after implementing the functionality of helical splines since then if a crowning parameter is added all transmitting gear pairs used today could be modeled by the program including planetary gear sets.

References

[1] Volvo Group design guideline. *Key joints*. 1st ed. 2015.

[2] Volvo Group Standard. *Straight-sided splines - General 5083,922*. 2nd ed. 1994.

[3] Volvo Group standard. *Spline joints with triangular profiles (serrations) - Tapered hubs and shafts 5083,971*. 3rd ed. 2000.

[4] Volvo Group standard. *Splines - Straight, cylindrical involute splines - Metric module - Drawing data 5083,17*. 5th ed. 2004.

[5] Volvo Bus Corporation. *Gearbox AT2412C, I-Shift 2, Design and function*. 1st ed. 2006.

[6] Tobias Mattsson and Anton Stenstrand. *Powertrain mechanics (TME170) assignment 2*. 1st ed. 2017.

[7] Mart Mägi and Kjell Melkersson. *LÄROBOK I MASKINELEMENT*. 2nd ed. 2014.

[8] Hermann J. Stadtfeld. *The basics of gear theory*. URL: `https://www.geartechnology.com/articles/0615/The_Basics_of_Gear_Theory/` (visited on 06/07/2018).

[9] Von Ingo Maier. "Kopfflanken-Rucknahme und kopfkantenbruch bei Zahnrädern". In: *Werkstatt und Betrieb* 104.4 (1971).

[10] Magnus Ekh. *Contact mechanics (lecture notes)*. 1st ed. 2015.

[11] Dassault systemes. *Abaqus documentation*. 6.14. 2014.

A Mesh quality check

A.1 Jacobian

The shapefunctions of a 8-node quadrilateral element are defined according to equation 51

$$\begin{aligned}
N_1 &= \frac{1}{4}\left(1+\eta+\xi+\xi\eta\right)\left(\xi+\eta-1\right) \\
N_2 &= \frac{1}{4}\left(1+\eta-\xi-\xi\eta\right)\left(-\xi+\eta-1\right) \\
N_3 &= \frac{1}{4}\left(1-\eta-\xi+\xi\eta\right)\left(-\xi-\eta-1\right) \\
N_4 &= \frac{1}{4}\left(1-\eta+\xi-\xi\eta\right)\left(\xi-\eta-1\right) \\
N_5 &= \frac{1}{2}\left(1-\xi^2\right)\left(1+\eta\right) \\
N_6 &= \frac{1}{2}\left(1-\xi\right)\left(1-\eta^2\right) \\
N_7 &= \frac{1}{2}\left(1-\xi^2\right)\left(1-\eta\right) \\
N_8 &= \frac{1}{2}\left(1+\xi\right)\left(1-\eta^2\right)
\end{aligned} \tag{51}$$

The derivatives with respect to ξ becomes:

$$\begin{aligned}
\frac{\partial N_1}{\partial \xi} &= \frac{1}{4}\left(\eta^2+\eta+2\xi+2\xi\eta\right) \\
\frac{\partial N_2}{\partial \xi} &= \frac{1}{4}\left(-\eta^2-\eta+2\xi+2\xi\eta\right) \\
\frac{\partial N_3}{\partial \xi} &= \frac{1}{4}\left(-\eta^2+\eta+2\xi-2\xi\eta\right) \\
\frac{\partial N_4}{\partial \xi} &= \frac{1}{4}\left(\eta^2-\eta+2\xi-2\xi\eta\right) \\
\frac{\partial N_5}{\partial \xi} &= -\xi\left(1+\eta\right) \\
\frac{\partial N_6}{\partial \xi} &= -\frac{1}{2}\left(1-\eta^2\right) \\
\frac{\partial N_7}{\partial \xi} &= -\xi\left(1-\eta\right) \\
\frac{\partial N_8}{\partial \xi} &= \frac{1}{2}\left(1-\eta^2\right)
\end{aligned} \tag{52}$$

The derivatives with respect to η becomes:

$$\begin{aligned}
\frac{\partial N_1}{\partial \eta} &= \frac{1}{4}\left(\xi^2 + \xi + 2\xi\eta + 2\eta\right) \\
\frac{\partial N_2}{\partial \eta} &= \frac{1}{4}\left(\xi^2 - \xi - 2\xi\eta + 2\eta\right) \\
\frac{\partial N_3}{\partial \eta} &= \frac{1}{4}\left(-\xi^2 + \xi - 2\xi\eta + 2\eta\right) \\
\frac{\partial N_4}{\partial \eta} &= \frac{1}{4}\left(-\xi^2 - \xi + 2\xi\eta + 2\eta\right) \\
\frac{\partial N_5}{\partial \eta} &= \frac{1}{2}\left(1 - \xi^2\right) \\
\frac{\partial N_6}{\partial \eta} &= -\eta\left(1 - \xi\right) \\
\frac{\partial N_7}{\partial \eta} &= -\frac{1}{2}\left(1 - \xi^2\right) \\
\frac{\partial N_8}{\partial \eta} &= -\eta\left(1 + \xi\right)
\end{aligned} \tag{53}$$

With the local and global orthonormal carteesian coordinate systems defined according to equation 54, the gradient of the shape functions can now be defined according to equation 55.

$$\begin{aligned}
\xi &= \xi \cdot e_\xi + \eta \cdot e_\eta \\
\mathbf{x} &= x \cdot e_x + y \cdot e_y
\end{aligned} \tag{54}$$

$$\nabla\xi = \frac{\partial \mathbf{N_i}}{\partial \xi_\mathbf{i}} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_1}{\partial \eta} \\ \frac{\partial N_2}{\partial \xi} & \frac{\partial N_2}{\partial \eta} \\ \dots & \dots \\ \frac{\partial N_n}{\partial \xi} & \frac{\partial N_n}{\partial \eta} \end{bmatrix} \tag{55}$$

The Jacobian of the mapping can now be expressed according to equation 56

$$\mathbf{J} = \mathbf{X} \cdot \nabla\xi \tag{56}$$

Where X is the matrix containing the global coordinates according to equation 57

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix} \tag{57}$$

The determinant is calculated at each Gauss point and the Jacobian ratio is then calculated for an element according to equation 58

$$\mathbf{J}_{ratio} = \frac{\min\left(\left|\det(\mathbf{J})\right|\right)}{\max\left(\left|\det(\mathbf{J})\right|\right)} \tag{58}$$

A.2 Aspect ratio

The aspect ratio is defined according to equation 59

$$A = \frac{\max(\mathbf{l}_i)}{\min(\mathbf{l}_i)} \tag{59}$$

Where $\mathbf{l}_i$ are the element side lengths and A is the aspect ratio for the element. The element side lengths in the two dimensional plane are straight forward and because of the meshing methodology used all element sides that extrudes into the plane are the same. These lengths can be calculated according to equation 60 beneath.

$$l_z = \frac{t}{z_{nel}} \tag{60}$$

Where t is the thickness of the spline and z_{nel} is the number of elements in the z-direction.

B Subscripts

The subscripts used by the program and the subprograms are shown in Figure 128 underneath. There are in total 23 programs classified as subscripts some of which are solely used by either the main GUI or the subprograms and some that are used frequently by many programs throughout the hierarchy. In this section a brief explanation of the function and purpose of each subscript is given together with their respective inputs and outputs.

Figure 128: Subscripts

adjust_prof_elem2

MATLAB code

```
1 function [mid_prof_line,div_prof,right_midprof,right,ldiv] = ...
2     adjust_prof_elem2(mid_prof_line,div_prof,right_midprof,right,ldiv)
```

Is used to modify the elements closest to the inner diameter of the internal spline in order to reduce the skewness of them as explained under Section 3.4.3. The program takes all matrices containing the node lines in this section as input, the script then adjusts the coordinates of the nodes pertinent to the elements closest to the inner diameter and then returns the same matrices with the new coordinates.

Advanced

advanced is the only subscript which is written as a GUI and is called when the *Advanced button* is triggered in the main GUI. This subscript handles the output requests given by the user under the *Advanced* section. It reads a *.mat-file* from the data library which initially contains the default output parameters, it then redefines this file according to the users input and saves in the data library where it can be reached by the subprograms that create the assembly input file.

botr_coord2

MATLAB code

```
1 function [r,phi,re] = botr_coord2(botr,ao,m,z,x,rb,rf,botr_p)
```

This subscript is used to create the polar coordinates for the root radius of the internal spline and the method used is the same as described in Section 2.4.3.

botr_elements

MATLAB code

```
1 function [nel_rf,nel_botr] =
      botr_elements(nel_botr_auto,botr,ao,m,z,x,rb,...
2 rf,divide_dist,radius_refine_int)
```

Calculates the amount of elements to distribute between the root radius and the major diameter part of mesh section 1 of the internal spline. The subscript calculates the length of each part and then distributes the number of elements for each section in order to obtain elements with equal side length along the profile. The amount for each section is also affected by the *radius refine* parameter which sets the distance to project the imaginary profile along which the elements are distributed.

calc_aspect2d

MATLAB code

```
1 function [Aspect,nelz_rec,l_plane_avg] =
      calc_aspect2d(Enod,node_coord,t,nelz,nelprof)
```

This subscript is used in order to calculate the aspect ratio when the user calls for an aspect ratio plot in the main GUI. The program calculates the side lengths for all elements for the symmetric 2-D part from the node coordinate and element coupling matrix. The length in the extruded direction is calculated from the length and number of elements in this direction. Lastly the program calculates the mean element side length for all elements in mesh section 1 for both parts and uses this length in order to return a suggested number of elements in the extruded direction.

calc_elem_angle_ext and calc_elem_angle_int

MATLAB code

```
function inner = calc_elem_angle_ext(upper,inner,start,break_pos)
```

MATLAB code

```
function outer_prof = calc_elem_angle_int(div_prof,outer_prof,...
start,break_pos)
```

These two subscripts are used in order to improve the geometry of the elements in mesh section 2 and 3 for each respective part. The results and reasoning behind this is explained more explicitly in Section 3.4.1. For the external spline the inputs to the program are the matrices containing the node coordinates for the inner diameter and the upper boundary of mesh section 2. For the internal spline the inputs are node coordinate matrices for the outer diameter and the projected profile. The subscripts uses the node coordinates for these lines to calculate the angle for the hypothetical elements between them and strives to make them as equal as possible based on the geometry of the spline. The subscripts then returns the node matrix containing the nodes on the outer diameter for the internal spline and the inner diameter for the external spline with the altered positions of the nodes. The nodes are only altered for the outer boundary of the mesh section and the intermediate nodes for the elements within the section are added afterwards which means that the angle adjustment will be added gradually for all elements within the mesh section.

calc_jacobian2d

MATLAB code

```
    function jacobian = calc_jacobian2d(Enod,node_coord)
```

Calculates the ratio of the Jacobian of the mapping between local and global coordinates as explained in Appendix A.1. The subscript takes the *node_coord* and *Enod* matrices as input and returns a vector containing the value for each element in the same order as in the *Enod* matrix.

calc_max_miss

MATLAB code

```
    function [miss_angle,miss_dist,d,theta] =
        calc_max_miss(m,z,ao,x_int,x_ext,radie,botr,t_inv,t,...
    Dii,Dei,Die,Dee)
```

This subscript is used in order to calculate the maximum possible misalignment without interference for a given spline coupling, the value calculated by this subscript is shown in the main GUI under the misalignment section and is updated every time the geometric data of the spline is changed. The output variable *miss_angle* is the maximum angle for the coupling and the variable *miss_dist* is the corresponding distance in the YZ-plane. The variables *d* and *theta* are internal variables used to convert a given angle to a distance in the YZ-plane which is used for the 2-D visualization of the misalignment in the GUI.

calc_rot_angle

MATLAB code

```
1  function angle = calc_rot_angle(m,z,ao,x_int,x_ext)
```

Calculates the angular play in the spline coupling given that the coupling is designed with a loose fit. The angle calculated by this subscript is then used in order to rotate the coupling in order to establish contact as shown in Figure 52 in Section 3.5.2.

ev_coordinates

MATLAB code

```
1  function phi = ev_coordinates(r,m,z,x,ao)
```

This subscript is used in order to calculate the circumferential coordinate for a given radial coordinate on the involute.

invo

MATLAB code

```
1  function involute = invo(angle)
```

invo is defined as the involute function derived in Section 2.3.3 and calculates the involute for a given angle.

max_botr_int

MATLAB code

```
1  function botr = max_botr_int(ao,m,z,x_int,rb,rf)
```

The function *max_botr_int* is used in order to calculate the maximum possible root radius for an internal spline. This function is called from the *geometry_internal* subprogram in order to make sure that the radius specified by the user does not exceed the physical maximum radius.

move_profile_internal

MATLAB code

```
1  function profile = move_profile_internal(org_prof,divide_dist)
```

This subscript is used to create mesh section 1 of the internal spline by creating the coordinates for the projected profile as shown in red in Figure 33 in Section 3.2.2. The program differentiates the original profile, calculates the normal for each node coordinate and projects the new profile with the distance specified in the input.

move_profile_external

MATLAB code

```
function coords = move_profile2(m,z,ao,bo,x,ro,dist,...
phi_trok,phi_e,hk,R,trok_p1,trok_p2)
```

Has the same function as *move_profile_internal*, but for the external spline and is shown in red in Figure 29 in Section 3.2.1. The program calculates new trochoid coordinates with an offset to the vector $\overrightarrow{BF}$ shown in Figure 20 in Section 2.4.1. The projection of the coordinates for the involute are approximated as a pure circumferential projection.

plot_2d_mesh

MATLAB code

```
function plot_2d_mesh(Enod,node_coord,fig_number)
```

Takes any *node_coord* and *Enod* matrices for hex-elements as input and plots the corresponding 2D mesh.

profile_nels

MATLAB code

```
function [nel_involute,nel_trok_root,nel_trok_botr]=...
    profile_nels(m,z,ao,x_ext,ro,radie,Die,...
nel_profile_rad,divide_dist,radius_refine_ext)
```

profile_nels is part of the automatic element distribution routines and calculates the number of elements to distribute over the three parts of the profile of the external spline. The subscript calculates the length of each section and distributes the elements in order to obtain an equal element side length in the same way as *botr_elements* described previously.

profile_nels_int

MATLAB code

```
function [nel_involute_inv,nel_rf,nel_botr] =...
    profile_nels_int(nel_profile_rad_int,botr,ao,...
m,z,x_int,rb,rf,ra,divide_dist,radius_refine_int)
```

Works in the same way as *profile_nels*, but for the internal spline.

profileshift2thickness

MATLAB code

```
function s = profileshift2thickness(x,m,ao)
```

Calculates the tooth thickness based on a profileshift coefficient as input.

progspace

MATLAB code

```
1  function y = progspace(start_point,end_point,n,prog_exp)
```

Creates a progressively spaced vector based on an exponential function which is used for creating the progressively sized elements in mesh section 2 of the external spline and mesh sections 2 and 3 of the internal spline.

thickness2profileshift

MATLAB code

```
1  function x = thickness2profileshift(s,m,ao)
```

Calculates the the profileshift coefficient for a given tooth thickness.

trokoid_nels

MATLAB code

```
1  function [nel_trok_root,nel_trok_botr]=
       trokoid_nels(m,z,ao,x,R,Die,nel,dist,radius_refine_ext)
```

Calculates how to distribute the elements for the root part of the external spline, in the same way as the other automatic element distribution routines. The corresponding subscript for the internal spline is the *botr_elements* script.

trokoid_straight

MATLAB code

```
1  function [x,y,hk,R]= trokoid_straight(m,z,ao,x,R,Die,nump1,nump2)
```

This script is used in order to calculate the node coordinates for the trochoid part of the external spline in the same way as explained explicitly in Section 2.4.1