NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF MECHANICAL ENGINEERING

MACHINE DESIGN LABORATORY

Development of gear-specific methodologies for FEM to include tooth modification and misalignment modelling in MATLAB and ANSYS Workbench environment.

THESIS

BY

MICHAIL SKIADOPOULOS

SUPERVISOR: Christophoros Provatidis
Professor N.T.U.A.

Athens, July 2021

Abstract

The goal of this thesis is to propose methods of numerical modelling of spur gear loading through the Finite Element Method. This is achieved both via the implementation of the commercial software ANSYS Workbench and by the development of an original code in the computational environment of MATLAB/Simulink. The process begins with the analysis of the theory of Hertzian contact between cylindrical bodies that links the geometrical features of the gears with the surface pressure that arises during their operation. The Hertz theory provides the base and the benchmark for the setup of static loading model of two-dimensional of ideal spur gears in ANSYS Workbench. More importantly, it helps to determine details about the mesh size in the areas of contact. The two-dimensional results are then used for the creation of the equivalent three-dimensional model which concludes to the development of sufficiently working models for the simulation of loading in the presence of misalignment and the presence of crowing in spur gears. Therefore, a methodology is established in which specific steps and adjustments are used of the modelling of gear loading in ANSYS Workbench. All the above lead to the development of a finite element code in MATLAB/Simulink, where the conclusions from the modelling structure from ANSYS Workbench are implemented. The purpose of this code is to replace the modelling in a commercial software with more simple and straightforward procedures that offers the same accuracy with reduced computational cost. Moreover, once again the results from the Hertzian contact theory are used in combination with Lo's algorithms for triangular mesh. The modelling is done in a single gear tooth using both CST (Constant Strain Triangular Element) and LST (Linear Strain Triangular Element) and a comparison is done between the two. In addition, the nodal load distribution exploits the theoretical elliptical shape of surface pressure. The results, remarks and conclusions drawn from this research will serve as a guide for the numerical modelling of gear loading and can also be applied to the modelling of contact phenomena in general. Furthermore, the MATLAB code will be used by the Machine Design Laboratory of N.T.U.A. for the study of gears with Finite Elements and its enrichment in the modelling options will continue from there.

Περίληψη

Ο σκοπός αυτής της διπλωματικής είναι η πρόταση μεθόδων για την αριθμιτική μοντελοποίηση της φόρτισης ευθείων οδοντωτών τροχών εξειλειγμένης καμπύλης με την Μέθοδο των Πεπερασμένων Στοιχείων. Αυτό επιτυγχάνεται με την χρήση του εμπορικού λογισμικού ANSYS Workbench αλλά και με την ανάπτυξη πρωτότυπου κώδικα στο υπολογιστικό περιβάλλον MATLAB/Simulink. Η διαδικασία για την εξαγωγή των μεθοδών μοντελοποίησης ξεκινά με την ανάλυση της θεωρίας επαφής του Hertz μεταξύ κυλινδρικών σωμάτων η οποία συνδέει τα γεωμετρικά χαρατκηριστικά των οδοντωτών τροχών με την πίεση επιφανείας που προκύπτει κατά την λειτουργία τους. Η θεωρία Hertz αποτελεί την βάση και το σημείο αναφοράς για την δημιουργία του δισδιάστατου μοντέλου της στατικής φόρτισης των ιδεατών οδοντωτών τροχών στο ANSYS Workbench. Σημαντικότερα, βοηθάει στον προσδιορισμο του μεγέθους του πλέγματος στην περιοχή της επαφής μεταξύ των οδόντων. Εν συνεχεία, τα αποτελέσματα που εξάγονται από την δισδιάστατη ανάλυση χρησιμοποιούνται για την δημιουργία της αντίστοιχης τρισιδιάστατης ανάλυσης, η οποία καταλήγει στην δημιουργία ικανοποιητικά λειτουργικών μοντέλων για την προσωμείωση της φόρτισης των οδοντωτών τροχών υπό την παρουσία σφάλματος ευθυγράμμισης και έχοντας υποστεί κατεργασία crowning. Επομένως, δημιουργείται μια μεθοδολογία στην οποία ακολουθούνται συγκεκριμμένα βήματα και γίνονται συγκεκριμμένες ρυθμίσεις για την μοντελοποίηση της φόρτισης οδοντωντών τροχών στο ANSYS Workbench. Όλα τα παραπάνω οδηγούν τελικά στην ανάπτυξη ενός κώδικα πεπερασμένων στοιχείων στο MATLAB/Simulink, όπου εφαρμόζονται όλα τα συμπεράσματα που έχουν εξαχθεί για την δομή της μοντελοποίησης από το ANSYS Workbench. Ο στόχος αυτού του κώδικα είναι να αντικαταστήσει την διαδικασία μοντελοποίησης σε εμπορικά λογισμικά, με έναν πιο απλό και άμεσο τρόπο μοντελοποίησης που προσφέρει την ίδια ακρίβεια με μειωμένο υπολογιστικό κόστος. Επιπλέον, και εδώ χρησιμοποιούνται τα αποτελέσματα από την θεωρία Hertz σε συνδυασμό με τους αλγορίθμους του Lo για την πλεγματοποίηση ενός κλειστού χωρίου με τριγωνικα στοιχεία. Η μοντελοποίηση γίνεται με την χρήση τρικομβικών και εξακομβικών τριγωνικών στοιχείων η αλλιώς CST και LST και γίνεται μία σύγκριση μεταξύ τους. Ακόμη, η κομβική κατανομή της δύναμης βασίζεται στην θεωρητική ελλειπτική κατανομή της πίεσης επιφανείας. Τα αποτελέσματα, τα σχόλια και τα συμπεράσματα τα οποία προκυποτυν από αυτήν την έρευνα θα λειτουργήσουν σαν όδηγός για την αριθμιτική μοντελοποίηση της φόρτισης οδοντωντών τροχών αλλά και γενικά φαινομένων στα οποία υπάρχει επαφή μεταξύ σωμάτων. Επιπρόσθετα, ο κώδικας σε MATLAB θα χρησιμοποιηθεί από το Εργαστήριο Στοιχείων Μηχανών του Ε.Μ.Π. για την μέλετη οδοντωτών τροχών με την χρήση Πεπερασμένων Στοιχείων και ο εμπλουτισμός του με περισσότερες επιλογές μοντελοποίησης θα συνεχίσει μέσα από αυτό.

Thanks

In this point I would like to express my gratitude to all the people that aid me throughout the whole experience of this thesis and through the course of my studies. I would like to thank all the teaching stuff of N.T.U.A and all my fellow students that contributed to the cultivation of my engineering knowledge and skills and accompanied me for five years that I will forever cherish.

Especially, I would like to thank my supervisor Mr Christophoros Provatidis and the head of Machine Design Laboratory of N.T.U.A. Mr Vasilis Spitas. These two honourable men will always constitute a role model for me not only because they nurtured me the passion for manufacturing, innovative design, and numerical analysis but also because they acted as advisors and guides through all my studies. I will always be grateful for all their help, and I am confident that they will continue to thrive both as scientists and as educators.

Moreover, I would like to thank the candidate Ph.D. student Christos Kalligeros for his precious help and support. His support was not only in a cognitive but also in a psychological level as he was ready to assist me whenever it was necessary at every step of this process.

Finally, I would like to thank all those people who with their love and support conduced to my development not only as an engineer but also as a human being. Above everything is the love and guidance from family and friends and I hope that someday I will be able to repay it.

Contents

1. Introduction

Gears are one of the most commonly used machine element in various industrial and civil applications. In fact, gears are used since ancient years, but their spread became greater with the development of more accurate manufacturing methods and the design of more complex tooth profiles that contributed even more to their ability both in the steady and the efficient transmission of large amount of mechanical power. Consequently, the industry is in great need robust methods for the design and testing of gears that can ensure their quality and their functionality.

The current study will be focusing on the contact analysis between the very widely used spur gears. Spur gears are one of the most popular types of precision cylindrical gears. These gears feature a simple design of straight, parallel teeth positioned around the circumference of a cylinder body with a central bore that fits over a shaft. In many variants, the gear is machined with a hub which thickens the gear body around the bore without changing the gear face. The central bore can also be broached as to allow the spur gear to fit onto a spline or keyed shaft. Spur gears are used in mechanical applications to increase or decrease the speed of a device or multiply torque by transmitting motion and power from one shaft to another through a series of mated gears. Spur gears provide several benefits to industrial applications and processes, including:

- ***Simplicity***. Spur gears feature a simple, compact design that makes them easy to design and install, even in limited or restricted spaces.
- ***Constant Speed Drive***. These gears increase or decrease shaft speed with a high degree of precision at a constant velocity.
- ***Reliability.*** Unlike other power and motion transmission components, spur gears are unlikely to slip during operation. Additionally, their durability decreases their risk of premature failure.
- ***Cost-Effectiveness.*** The simplicity of their design also allows for greater manufacturability, making them less expensive to fabricate and purchase even with highly specific or customized dimensions.
- ***Efficiency.*** Spur gear systems have power transmission efficiencies between 95% and 99% and can transfer large amounts of power across multiple gears with minimal power loss.

Spur gears are used to transfer motion and power from one shaft to another in a mechanical setup. This transference can alter machinery's operating speed, multiply torque, and allow for the fine-tuned control of positioning systems. Their design makes them suitable for lower speed operations or operational environments with a higher noise tolerance. Some of the typical industrial applications include:

- Transmissions
- Conveyor systems
- Speed reducers
- Engines and mechanical transportation systems
- Gear pumps and motors
- Machining tools

In spur gears, the ideal contact will be investigated along with the contact with a misalignment error and with the tooth modification of crowning which will be analysed in their respective chapters.

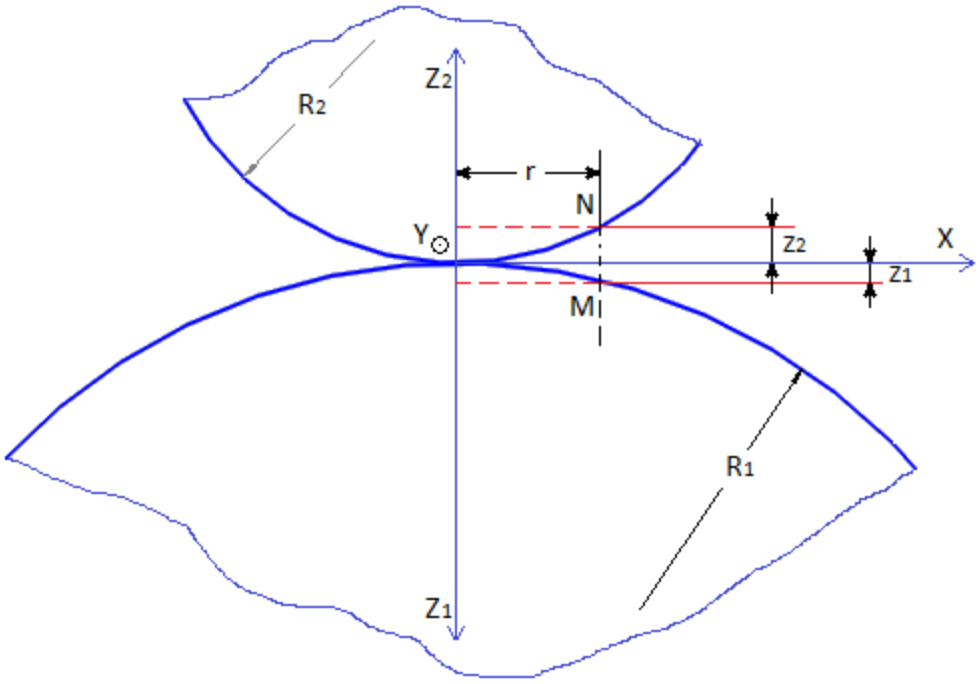
Nowadays, when it comes to designing a product there are two main schools of thought. The first one is investing in the fast prototyping. Meaning that the industry will use the analytical formulas and experience to make a conceptual design and after some changes manufacture a prototype to submit it to several tests and verify its strength for example. This is usually done by using manufacturing methods such 3D Printing or Injection Molding which are manufacturing procedures that can produce one prototype of the product at a fast pace. The second one is the use of numerical simulations to test the product before it is even manufactured and then proceed to the creation of prototype and test it. In this way, the industry can save time form the physical tests and money both form the use of the testing machined and from the loss of prototypes especially if the tests are destructive which they usually are. The most common method applied in these numerical simulations is the Finite Element Method.

The Finite Element Method (FEM) is a widely used method for numerically solving differential equations arising in engineering and mathematical modelling. Typical problem areas of interest include the traditional fields of structural analysis, heat transfer, fluid flow, mass transport, and electromagnetic potential. The FEM is a general numerical method for solving partial differential equations in two or three space variables (i.e., some boundary value problems). To solve a problem, the FEM subdivides a large system into smaller, simpler parts that are called finite elements. This is achieved by a particular space discretization in the space dimensions, which is implemented by the construction of a mesh of the object: the numerical domain for the solution, which has a finite number of points. The finite element method formulation of a boundary value problem finally results in a system of algebraic equations. The method approximates the unknown function over the domain. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. There are also other methods that have been based in the idea of FEM like the Applied Element Method (AEM), the Extended Finite Element Method (X-FEM), the Spectral Elements Method etc but these will not be part of the discussion in this thesis. The Finite Element method combined with the analytical formulas of Hertz theory about elastic are the heart of the concept of the methodologies to be presented.

The primary goal of the present thesis is the proposal of functional methodologies for the numerical modelling of contact between spur gears including the cases of ideal contact, contact with a misalignment error and contact with crowing effect. As it was previously stated, the modern industry mainly follows two paths but the one of the numerical modelling can save time and money from it and aid as a prediction tool of its existing products. Thus, this study will constitute a source of steps, comments, and conclusion for effective ways to conduct these numerical simulations and it will also serve as benchmark for the testing of the results of the simulations. In the chapters that will follow, an overview of the Hertzian contact theory will be presented that will be followed by the presentation of the setup of FEM models in ANSYS Workbench and the MATLAB code for the gear analysis. Each modelling procedure will be accompanied by the results and their verification. Finally, there will be a discussion on the presented results.

2. Overview of Hertz contact theory

Let us assume that two bodies come into contact as in Picture 2.1 and that these bodies have been deformed under the influence of a vertical force acting on one of the two bodies. As a result, a contact area is created.



Picture 2.1. Bodies that gradually come into contact.[3]

By ignoring the higher order terms the surfaces of the bodies in the area of contact can be expressed by the following formula:

$$z_1 = A_1x^2 + A_2xy + A_3y^2 \text{ (2.1)}$$

$$z_2 = B_1x^2 + B_2xy + B_3y^2 \text{ (2.2)}$$

where z_1 and z_2 are the distances from the area of contact to the points M and N, respectively. If we add (2.1) and (2.2) and choose as coordinate system, the one of the principal curvatures then we receive the following formula:

$$z_1 + z_2 = Ax^2 + By^2 \text{ (2.3)}$$

So, by choosing this coordinate system we were able to erase the xy terms and replace $A_1 + B_1$ and $A_3 + B_3$ by A and B, respectively.

Moreover, R_1 and R_1' are the principal curvatures of Body 1 and R_2 and R_2' are the principal curvatures of Body 2. R_1 and R_1' belong to perpendicular planes and the same applies for R_2 and R_2'. Furthermore, we define ψ the angle between the planes of R_1 and R_2. Thus, the values of the constants A, B can be found by solving the following system:

$$A + B = \frac{1}{2}\left(\frac{1}{R_1} + \frac{1}{R_1'} + \frac{1}{R_2} + \frac{1}{R_2'}\right) \text{ (2.4)}$$

$$B - A = \frac{1}{2}\sqrt{((\frac{1}{R_1} - \frac{1}{R_1'})^2 + (\frac{1}{R_2} - \frac{1}{R_2'})^2 + 2(\frac{1}{R_1} - \frac{1}{R_1'})(\frac{1}{R_2} - \frac{1}{R_2'})\cos\psi)} \text{ (2.5)}$$

To simplify the problem, we assume that $\psi = 0$. As the two bodies come into contact the points M, N are moving by w_1, w_2 respectively along the Z axis. If the initial distance between M and N was d then:

$$w_1 + w_2 + z_1 + z_2 = d \text{ (2.6)}$$

By combining (2.3) and (2.6) we get:

$$w_1 + w_2 = d - Ax^2 - By^2 \text{ (2.7)}$$

Assuming now that the two bodies are semi-infinite (that suits the case of gear contact since their materials usually have low compliance and the area of contact is very small comparable to the dimensions of the gears) we can use the following expression:

$$w_1 + w_2 = (\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}) \int_{\Omega} rqd\Omega \text{ (2.8)}$$

where q is the normal pressure applied to an infinitely small part $d\Omega$ that lies in a distance r from the centre of the contact area Ω. In addition, ν_1, ν_2 are the Poisson ratio of the material of the Body 1 and 2 and E_1, E_2 their Young Modulus.

We insert the variable E :

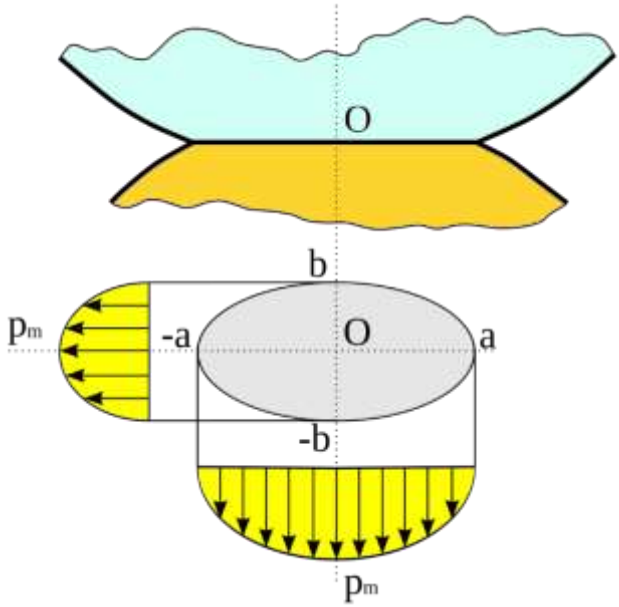
$$\frac{1}{E} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \text{ (2.9)}$$

where $E$is the equivalent Young's Modulus.

By combining (2.7), (2.8) we receive the following expression:

$$\frac{1}{E}\int_{\Omega} rqd\Omega = d - Ax^2 - By^2 \text{ (2.10)}$$

Based on (13) all it takes is to determine a suitable pressure distribution that can satisfy the formula and Hertz chose as distribution the one of the half ellipsoid. This distribution has also been verified experimentally. It is evident that if the pressure follows an elliptical distribution, then the shape of the contact area will be an ellipse itself with dimensions a and b as shown in the Picture 2.2.



Picture 2.2. Hertz pressure distribution.[6]

If P is the compressional load acting on the contact surfaces, then the maximum pressure p_{max} is given by:

$$p_{max} = \frac{2P}{3\pi ab} \text{ (2.11)}$$

The axes of the ellipsoid are calculated by the following formulas:

$$a = m\sqrt[3]{\frac{3\pi P}{4E(A+B)}} \text{ (2.12)}$$

$$b = n\sqrt[3]{\frac{3\pi P}{4E(A+B)}} \text{ (2.13)}$$

The values of the constants m, n are given by table like the following Table 1 that H.L. Whittemore and N.S. Petrenko proposed.

θ(deg)	30º	35º	40º	45º	50º	55º	60º	60º	70º	75º	80º	85º	90º
m	2.371	2.397	2.137	1.926	1.754	1.611	1.486	1.378	1.284	1.202	1.128	1.061	1.000
n	0.493	0.53	0.567	0.607	0.641	0.678	0.717	0.759	0.802	0.846	0.893	0.944	1.000

Table 2.1. H.L. Whittemore and N.S. Petrenko table for selection m and n of values.

where θ is calculated is equal to:

$$\theta = \cos^{-1}\frac{B-A}{A+B} \text{ (2.14)}$$

Since primarily we want to study the contact behaviour in spur gears, we can make the following simplifications:

1. The contact surfaces of gears are cylindrical. Consequently R_1', $R_2' \to \infty$ and (4), (5) becomes:

$$A + B = \frac{1}{2}\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \text{ (2.15)}$$

$$B - A = \frac{1}{2}\left|\frac{1}{R_1} - \frac{1}{R_2}\right| \text{ (2.16)}$$

2. The assumption that $\psi = 0$ applies in this case since lie R_1, R_2 on the same plane.
3. The type of contact is line contact instead of the general point contact. Therefore, the contact area does not have the shape of an ellipse but instead it is shaped as a rectangle with height h, and width w equal to:

$$w = \text{ gear width}$$

$$h = 2\sqrt{\frac{4PR}{Eh}} \text{ (2.17)}$$

where R is the equivalent curvature radius and is given by the formula:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \text{(2.18)}$$

4. The pressure distribution still follows the one of the half ellipsoid but it does not vary in the direction of the gear width.
5. The maximum pressure p_{max} is given by:

$$p_{max} = \frac{2P}{\pi h w} \text{ (2.19)}$$

and the pressure distribution is given by the following formula:

$$p = p_{max}\sqrt{1 - \left(\frac{x}{h}\right)^2} \text{ (2.20), where } x \in (-h, h)$$

The setting up and the verification of the numerical simulations with the FEM modelling will be based in this theory.

3. FEM models in ANSYS WORKBENCH

In this chapter an algorithm for the effective FEM modelling of contact between gears will be presented. First, the two-dimensional modelling of ideal spur gears will be explained followed by the three-dimensional. Afterwards, the modelling of the misalignment error of the gear shafts on the plane of action will be displayed and finally there will be the modelling of the crowning effect. The dimensions and material of gears that will be used in the simulation are the shown in Table 3.1.

Involute spur gears	**Pinion**	**Conjugate gear**
Pressure angle (deg)	20°	20°
Module (mm)	4	4
Tooth number	20	30
Width (mm)	40	40
Tip diameter (mm)	88	128
Root diameter (mm)	70	110
Backlash(mm)	-	0.05-0.1
Gear material	SCM415	SCM415
Torque load (Nm)	-	98

Table 3.1. Dimensions and material of gears.

The characteristics of the gears were selected to be able to verify the results of the simulations by comparing them to [2].

3.1 Two-dimensional modelling of involute spur gears

In the following paragraphs the two-dimensional modelling of gear contact in ANSYS Workbench will be discussed. This will be done by explaining every step of the process.

3.1.1 Pre simulation calculations

Before the setting up of the simulation some calculations are necessary to assist the mesh process and to verify our results. For this purpose, the MATLAB program [] was created. This program was given the characteristics of gears as an input and had as an output the maximum theoretical pressure from Hertz theory and the maximum bending stress.

First and foremost, we must calculate some basic quantities of gears like the radius of the pitch and form circle and the maximum degree of overlap. The radii of the pitch circle and form circle for each gear is given by:

$$r_{o1} = \frac{mZ_1}{2} \quad (3.1)$$

$$r_{o2} = \frac{mZ_2}{2} \quad (3.2)$$

$$r_{g1} = r_{o1} \cos a_o \quad (3.3)$$

$$r_{g2} = r_{o2} \cos a_o \quad (3.4)$$

where m is the gear module, a_o is the pressure angle, Z_1 and $Z_2$are the number of teeth of the conjugate gear and pinion respectively, r_{o1} and r_{o2} are the radii of pitch circles of the gear in collaboration and pinion respectively and finally, r_{g1} and r_{g2} are the radii of form circles of the gear in collaboration and pinion respectively. This results in:

r_{o1} **(mm)**	40
r_{o2} **(mm)**	60
r_{g1} **(mm)**	37.59
r_{g2} **(mm)**	56.38

Table 3.2. Basic gear radii

The maximum degree of overlap ε can be calculates via the following formula:

$$\varepsilon = \frac{AB}{t_{g1}} \text{ (3.5)}$$

where t_{o1} is the tooth width of gear in collaboration in pitch circle and AB is the total length of the contact trajectory of the gears. This is straight line passing from the centre of axes at (0,0), with an incline equal to the pressure angle since we study involute spur gears. These quantities are given by the following formulas:

$$t_{g1} = \pi m \cos a_o \text{ (3.6)}$$

$$BC = \sqrt{(r_{o1} + m)^2 - (r_{o1} \cos a_o)^2} - r_{o1} \sin a_o \text{ (3.7)}$$

$$CA = \sqrt{(r_{o2} + m)^2 - (r_{o2} \cos a_o)^2} - r_{o2} \sin a_o \text{ (3.8)}$$

$$AB = BC + CA \text{ (3.9)}$$

This results in:

t_{g1} **(mm)**	11.81
AB **(mm)**	18.95
ε	1.61

Table 3.3. Maximum degree of overlap calculation

Secondly, we need to specify the position in which the teeth of the gears are engaged. This is a necessary step that provides us with information of the gear geometry at the area of contact and of the LSR (Load Sharing Ratio). In the [2] all the simulation results were shown for position 6, which is shown in the LSR diagram of Fig 1. Therefore, for using [2] to compare our results and for running the simulation in a more complex tooth engagement position (because in position 6 we there is a double pair tooth contact), position 6 was selected.

Having chosen the position in which the gears engage we can calculate quantities needed to do the Hertz analysis. From Table 2, it is given that the torque M on the conjugate gear is equal to 98 Nm. In this case we must define two separate normal forces P_1 , P_2 each for one contact pair and for the LSR diagram we know that at position 6 (it is evident that P_1 corresponds to the pair that is engaged in position 6 and P_2 to the second pair):

$$LSR = 0.575 \text{ (3.10)}$$

$$P_1 = LSR * P \text{ (3.11)}$$

$$P_2 = LSR * P \text{ (3.12)}$$

where P is the force that acts if there is only one pair in phase of engagement. This happens in positions 8 through 12 and to have an average value of it we choose to calculate it in position 10. Can be calculated by the following formula:

$$P_x = \frac{M}{r_{10}} \text{ (3.13)}$$

$$P = \frac{P_x}{\cos a_o} \text{ (3.14)}$$

where P_x is the x-component of P and r_{10} is the radius from the centre of conjugate gear in which it comes in contact in position 10. In this position we observe from Fig 1. that the degree of overlap ε_{10} is 0.8.

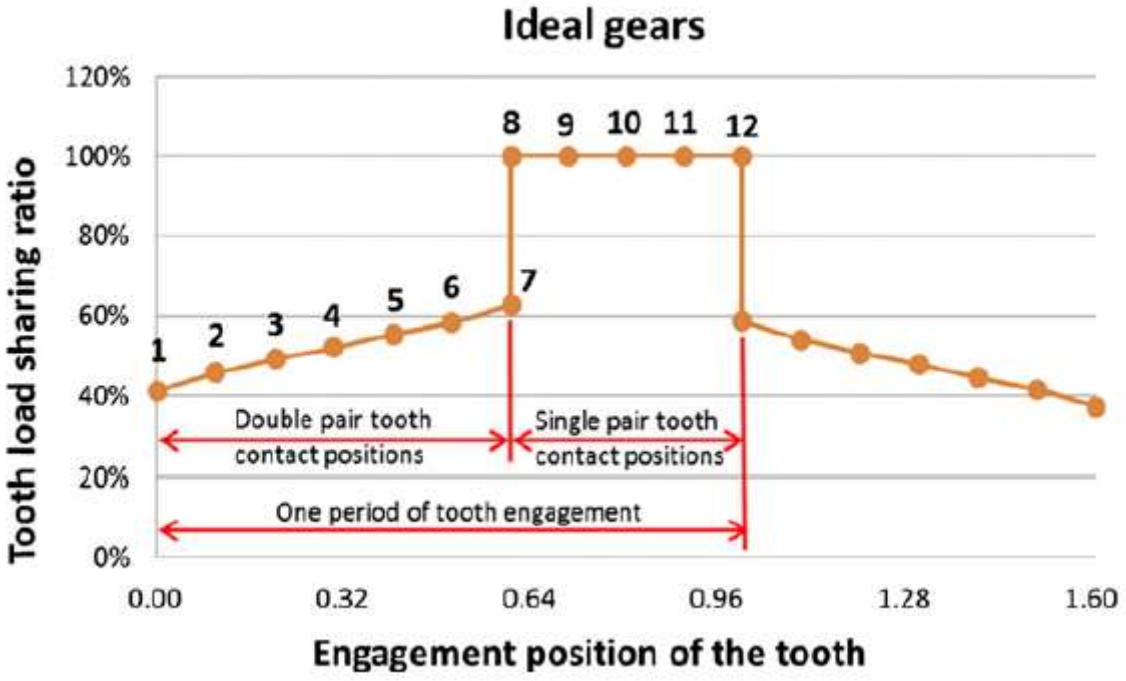


Fig 3.1. LSR of the ideal gears.[2]

Now we can find the exact coordinates of this position (x_{10}, y_{10}) which are equal to:

$$x_{10} = CA - \frac{\varepsilon_{10}}{\varepsilon} AB * \cos a_o \text{ (3.15)}$$

$$y_{10} = CA - \frac{\varepsilon_{10}}{\varepsilon} AB * \sin a_o \text{ (3.16)}$$

This results in $(x_{10}, y_{10}) = (0.268, 0.097)$. Finally, we can calculate the radius r_{10}:

$$r_{10} = \sqrt{{x_{10}}^2 + (y_{10} - r_{o1})^2} \text{ (3.17)}$$

This results in $r_{10} = 59.90$ mm and from equation (3.14) it gives us that $P = 1{,}740.97$ N. In addition, from equations (3.11), (3.12) we receive:

$\boldsymbol{P_1}$ **(N)**	1,001.05
$\boldsymbol{P_2}$ **(N)**	739.91

Table 3.4. LSR in the contact pairs.

Now that the normal forces acting on each pair are known we also require the equivalent Young's Modulus and the equivalent radius of curvature in the points of contact to fully define the loading and geometry of contact area according to Hertz theory. Concerning the equivalent

Young's Modulus, it is stated that the material of both gears is SCM415 which is a typical steel with following properties:

$\boldsymbol{\nu_{SCM415}}$	0.3
$\boldsymbol{E_{SCM415}}$ **(MPa)**	$2 \cdot 10^5$

Table 3.5. Material properties of gears.

From (2.9) by setting we get $\nu_1 = \nu_2 = \nu_{SCM415}$ and $E_1 = E_2 = E_{SCM415}$ we get $E = 1.099 \cdot 10^5$ MPa. To find the equivalent radius of curvature we need to find the radii from the centre of gears at the point of each contact pair. From Fig 1. We know that in position 6, $\varepsilon_6 = 0.48$ so by using again (3.15) through (3.17) we find that $r_{61} = 58.89$ mm (we now use 61 instead 6 to denote that this concerns the radius of the gear in collaboration and 62 will be used to denote the radius of pinion.). For r_{62} the following formula will be used:

$$r_{63} = \sqrt{{x_6}^2 + (y_6 + r_{o2})^2} \text{ (3.18)}$$

and thus $r_{63} = 41.34$ mm.

After having computed the radii from the centre of each gear at the point of contact in position 6, we must find the radii of curvatures R_{61} and R_{62} of gear in collaboration and pinion, respectively. These will be obtained by the following relations:

$$a_{61} = \cos^{-1}\frac{r_{g1}}{r_{61}} \text{ (3.19)}$$

$$a_{62} = \cos^{-1}\frac{r_{g2}}{r_{62}} \text{ (3.20)}$$

$$R_{61} = r_{g1} \sin a_{61} \text{ (3.21)}$$

$$R_{62} = r_{g2} \sin a_{62} \text{ (3.22)}$$

Thus, by replacing R_{61} and R_{62} to equation (2.18) we get $R_6 = 8.55$ mm. The same procedure from (3.15) to (3.22) is done for the second pair of contact that has $\varepsilon_{6'} = \varepsilon_6 + 1 = 1.48$ .The results about the radii of curvature for both gears are presented in Table 6 and Table 7.

$\boldsymbol{R_{61}}$ **(mm^{-1})**	16.99
$\boldsymbol{R_{62}}$ **(mm^{-1})**	17.20
$\boldsymbol{R_6}$ **(mm^{-1})**	8.55

Table 3.6. Radii of curvature for position 6.

$\boldsymbol{R_{61}}$ **(mm^{-1})**	28.80
$\boldsymbol{R_{62}}$ **(mm^{-1})**	5.39
$\boldsymbol{R_6}$ **(mm^{-1})**	4.54

Table 3.7. Radii of curvature for position 6'.

By replacing all the above to (2.17) and (2.19) we get the height of the rectangle that will form as contact area and the maximum pressure at each pair of contact which are presented in Table 8 and Table 9.

$\boldsymbol{h_6}$ **(mm)**	0.099
$\boldsymbol{p_{max,6}}$ **(MPa)**	319.98

Table 3.8. Height of rectangle of contact area and maximum contact pressure at position 6.

$h_{6'}$ **(mm)**	0.062
$p_{max,6'}$ **(MPa)**	377.32

Table 3.9. Height of rectangle of contact area and maximum contact pressure at position 6'.

The last calculation before we set up the simulation is that of the maximum theoretical bending stress σ_{bmax}. This is given by the following formula:

$$\sigma_{bmax,n} = \frac{P_{x,n} q_k}{wm} \text{ (3.23),}$$

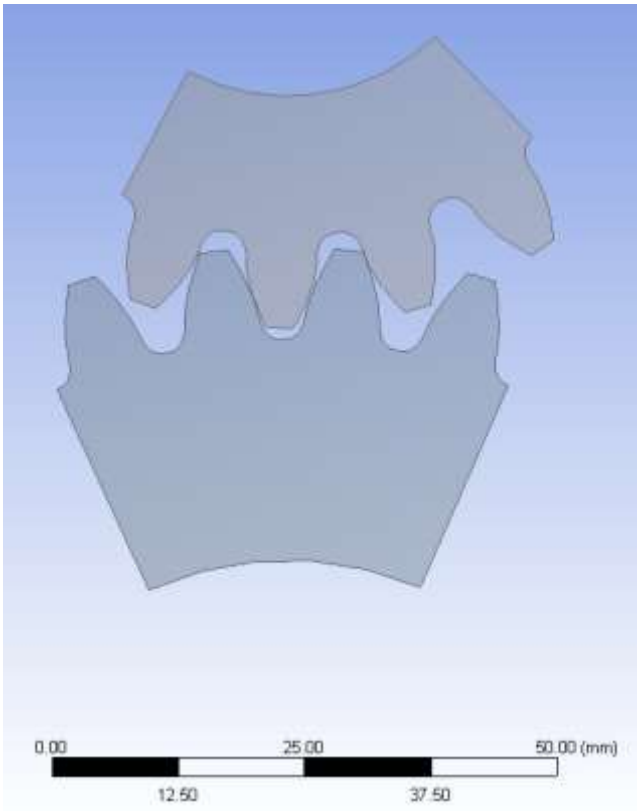
where w is the gear width and the value of constant q_k depends on the number of teeth and the displacement of the axes of gears So from tables we get that for zeros displacement and for $Z_1 = 30$, $q_k = 2.6$. In Table 10 the theoretical maximum bending stress for each position in the teeth of conjugate gear are presented (the computation of bending stress at the teeth of pinion are omitted because it is sufficient only the ones for the gear in collaboration to be verified through the simulation).

$\sigma_{bmax,61}$ **(MPa)**	15.29
$\sigma_{bmax,61'}$ **(MPa)**	11.30

Table 3.10. Maximum theoretical bending stresses at each contact pair in the teeth of conjugate gear.

3.1.2 Importing the CAD models into ANSYS Workbench

The CAD models of gears were created via KISSOFT and Solidworks and then they were imported into Design Modeler of Static Structural module of ANSYS Workbench as shown in Picture 3.1.



Picture 3.1.CAD models of gears in Design Modeler.

The assembly of gears was initially created in Solidworks, but at the first time that the simulation run a problem was observed that led to specific modifications of the assembly that were done using the tools of Design Modeler. This problem was that Solidworks' assembly tools were not accurate enough and thus the gears did not engage at the desired position but in some position very close to it. An even bigger problem was that due to the fault positioning of gears unwanted penetrations were created between the gears and there was a considerable difference between the penetration of the first and the second pair which results in having false values in contact pressure. So, to avoid these the gears were inserted from Solidworks as shown in Picture and then they were rotated by a specific angle that was computed in MATLAB program [].

Picture 3.2. Gears at the initial state of their insertion in Design Modeler.

The way that the Matlab program calculates these angles is that it specifies the coordinates on the tooth profile of each gear by the equation of involute curve since we know the radius from the centre of each gear to the point of contact. These equations are shown below:

$$t = \sqrt{(\frac{r_n}{r_{gn}})^2 - 1} \quad (3.24)$$

$$x = (\sin t - t\cos t) \quad (3.25)$$

$$y = (\cos t + t\sin t) \quad (3.26)$$

$$x_p = x \cos fi - y \sin fi \quad (3.27)$$

$$y_p = x \sin fi + y \cos fi \quad (3.28)$$

where r_n is the radius of each gear at the point of contact and x_p, y_p the coordinates of the points in this radius when the tooth is in the vertical position.

Furthermore, in the previous section we have computed the coordinates of the contact point of each pair, so the angle of rotation is just the angle that is formed between the lines that are creates if lines are drawn from these points to the centre of the axes. For this case, the angles are shown in Table 3.11:

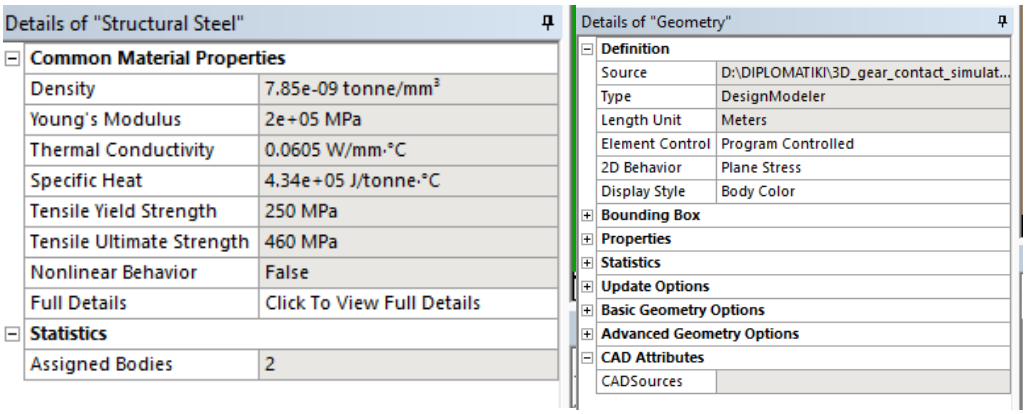
Rotation angle of gear in collaboration (deg)	180.94
Rotation angle of pinion (deg)	-6.54

Table 3.11. Angles of rotations of gears in Design Modeler.

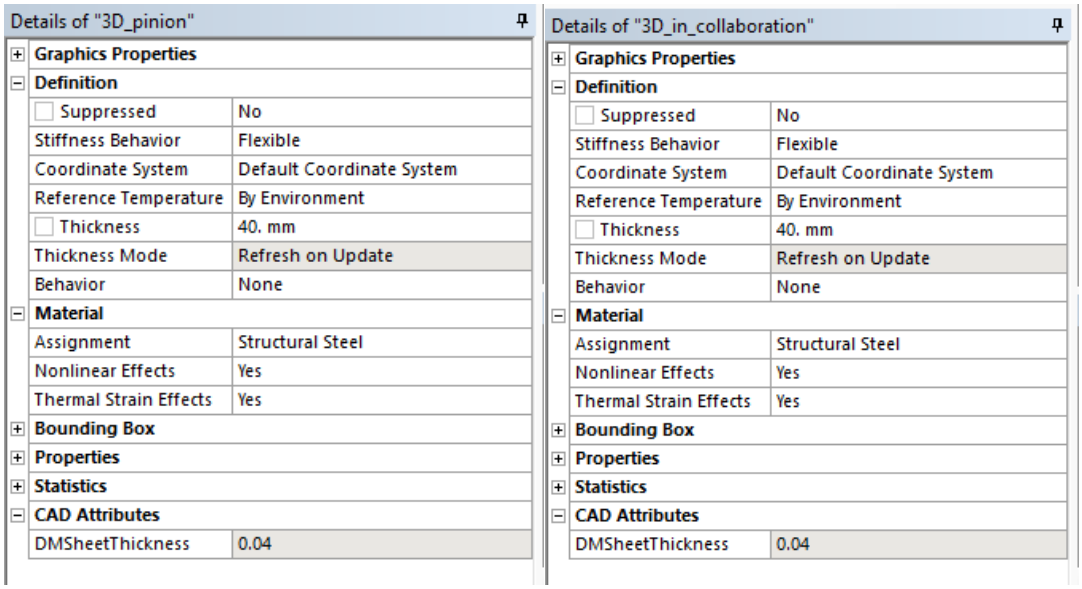
It must be noted that this procedure was done using the Rotate tool of Design Modeler which is found in Create → Body transformation and the sign of angles depends on the positive direction of the axes of rotation. Another important configuration of the model that was done in Design Modeler was the creation of the two-dimensional surfaces of gears which was done using Mid-Surface tool which is found in Tools. Finally, the model does not include the whole gear but just the two pairs of teeth that are engages and two more teeth in each gear. This is done to decrease the computational time since a smaller model means less nodes in our mesh.

3.1.3 Setting up the material properties

Since the material of gears has the properties of a common steel, in the simulation the default material of Structural Steel is selected that has the properties of Table 3.5. Moreover, it also required to choose the behaviour of the material which is set to Plane Stress and to define the width of the gear. All of these are done in the Geometry Section inside the Model module of Static Structural and are shown in Picture 3.3 and Picture 3.4.



Picture 3.3. Material Properties and behaviour in Static Structural.



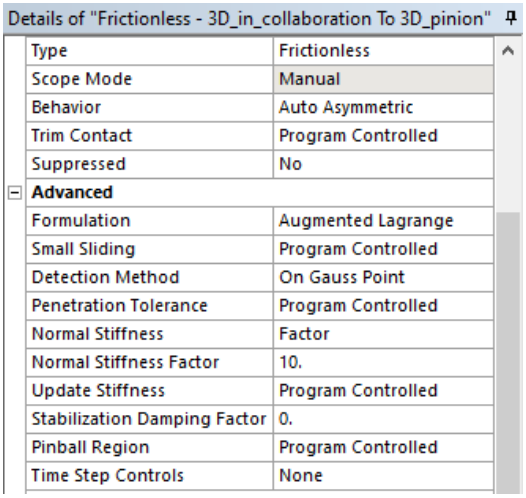
Details of "3D_pinion"	
Graphics Properties	
Definition	
Suppressed	No
Stiffness Behavior	Flexible
Coordinate System	Default Coordinate System
Reference Temperature	By Environment
Thickness	40. mm
Thickness Mode	Refresh on Update
Behavior	None
Material	
Assignment	Structural Steel
Nonlinear Effects	Yes
Thermal Strain Effects	Yes
Bounding Box	
Properties	
Statistics	
CAD Attributes	
DMSheetThickness	0.04

Details of "3D_in_collaboration"	
Graphics Properties	
Definition	
Suppressed	No
Stiffness Behavior	Flexible
Coordinate System	Default Coordinate System
Reference Temperature	By Environment
Thickness	40. mm
Thickness Mode	Refresh on Update
Behavior	None
Material	
Assignment	Structural Steel
Nonlinear Effects	Yes
Thermal Strain Effects	Yes
Bounding Box	
Properties	
Statistics	
CAD Attributes	
DMSheetThickness	0.04

Picture 3.4. Width of gears in Static Structural.

3.1.4. Setting up of the contact parameters

The basic contact parameters that were tuned were the following. The first one and most important one if the Type of contact. This was set to frictionless since we want to verify our results with Hertz's theory where contact is assumed to be frictionless. The next parameter was the Normal Stiffness Factor. The function of Normal Stiffness factor is basically to soften of harden the stiffness of the contact area. We may think of it as a spring between the contact surfaces that when the value of its stiffness is increased it is more rigid and when it is decreased it is more flexible. The default value of this parameter is 1 but to have more accurate results its value was set to 10. This was done because making the contact pair more stiff decreases the penetration between the bodies and provides us with more realistic solution (this stands true up to a point because if the Normal Stiffness Factor becomes too large then oscillations are created, and the program is facing convergence issues). All the rest parameters were set to default values. The setting up of the two contact pairs (since there are two contact areas in this gear engagement) are shown in Picture 3.5 and Picture 3.6.



Details of "Frictionless - 3D_in_collaboration To 3D_pinion"	
Type	Frictionless
Scope Mode	Manual
Behavior	Auto Asymmetric
Trim Contact	Program Controlled
Suppressed	No
Advanced	
Formulation	Augmented Lagrange
Small Sliding	Program Controlled
Detection Method	On Gauss Point
Penetration Tolerance	Program Controlled
Normal Stiffness	Factor
Normal Stiffness Factor	10.
Update Stiffness	Program Controlled
Stabilization Damping Factor	0.
Pinball Region	Program Controlled
Time Step Controls	None

Picture 3.5. Setting up of contact parameters of contact pair in position 6.

Details of "Frictionless - 3D_in_collaboration To 3D_pinion"	
Type	Frictionless
Scope Mode	Manual
Behavior	Auto Asymmetric
Trim Contact	Program Controlled
Suppressed	No
Advanced	
Formulation	Augmented Lagrange
Small Sliding	Program Controlled
Detection Method	On Gauss Point
Penetration Tolerance	Program Controlled
Normal Stiffness	Factor
Normal Stiffness Factor	10.
Update Stiffness	Program Controlled
Stabilization Damping Factor	0.
Pinball Region	Program Controlled
Time Step Controls	None

Picture 3.6. Setting up of contact parameters of contact pair in position 6'.

Another essential tool that can save the user from errors is the Contact Tool which is found in the section Contacts. This tool is a pre-processing tool that calculates the penetrations or gaps between the two meshed bodies and can prove to be extremely useful since for instance when the type of contact is set to Frictionless the simulation cannot run if a gap is detected between the two bodies. In Picture 3.7 the results of contact tool for the case of gears are shown.

Initial Information

For additional options, please visit the context menu for this table (right mouse button)

Name	Contact Side	Type	Status	Number Contacting	Penetration (mm)	Gap (mm)	Geometric Penetration (mm)	Geometric Gap (mm)	Resulting Pinball (mm)	Real Constant
Frictionless - 3D_in_collaboration To 3D_pinion	Contact	Frictionless	Inactive	N/A	N/A	N/A	N/A	N/A	N/A	3.
Frictionless - 3D_in_collaboration To 3D_pinion	Target	Frictionless	Closed	205.	4.4023e-004	0.	4.4023e-004	3.0205e-007	1.7865e-002	4.
Frictionless - 3D_in_collaboration To 3D_pinion	Contact	Frictionless	Inactive	N/A	N/A	N/A	N/A	N/A	N/A	5.
Frictionless - 3D_in_collaboration To 3D_pinion	Target	Frictionless	Closed	166.	4.2041e-004	0.	4.2041e-004	1.8307e-006	2.0676e-002	6.

Color Legend

Red	The contact status is open but the type of contact is meant to be closed. This applies to bonded and no separation contact types.
Yellow	The contact status is open. This may be acceptable.
Orange	The contact status is closed but has a large amount of gap or penetration. Check penetration and gap compared to pinball and depth.
Gray	Contact is inactive. This can occur for MPC and Normal Lagrange formulations. It can also occur for auto asymmetric behavior.

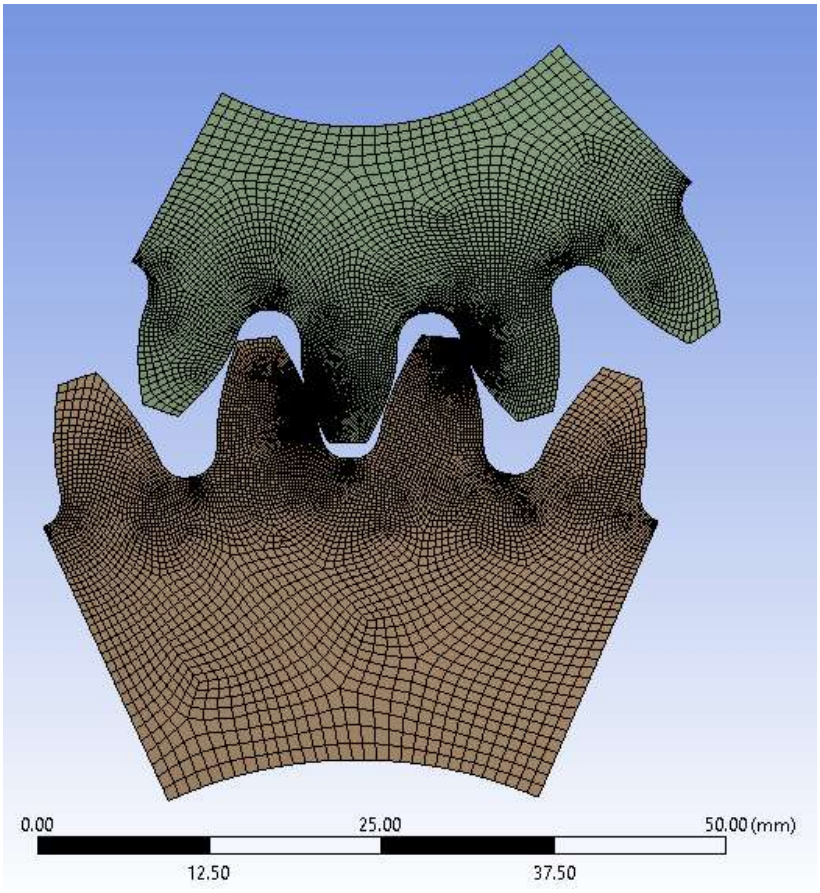
Picture 3.7. Results of Contact Tool.

Form Picture 3.7, the status on both contact pairs is closed and the penetration is at an acceptable degree and almost the same in both pairs which is also crucial. If the difference in the value of penetration among the pairs is large, then one pair the pair with the larger penetration will present larger stresses than the actual ones and this is the reason why the assembly was not completed in Solidworks, and the gears were positioned correctly in Design Modeler as it is mentioned in chapter.

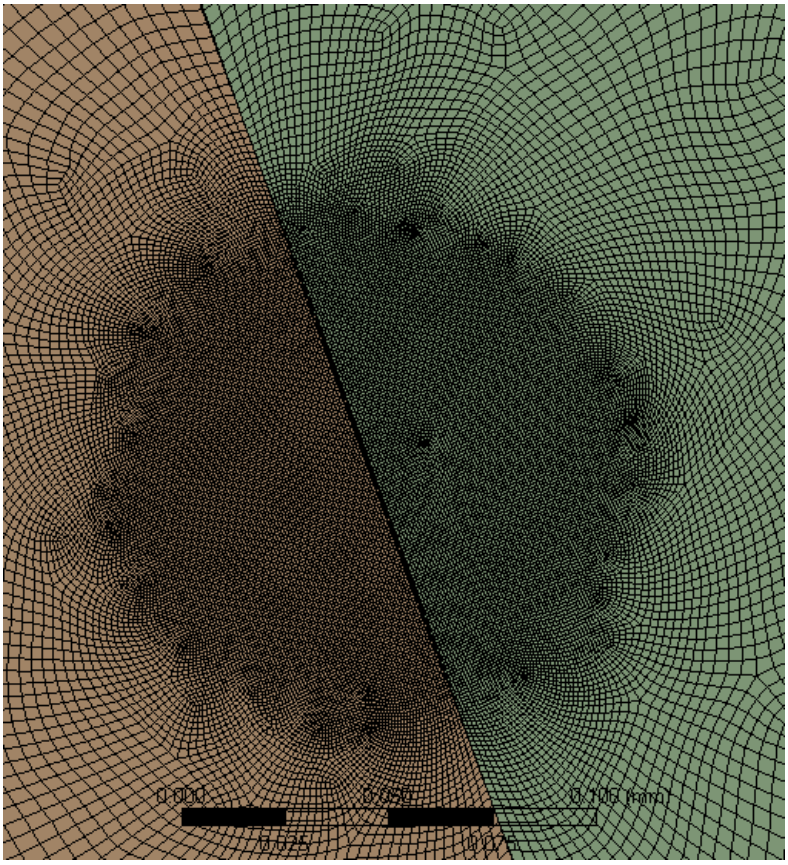
3.1.5 Setting up of the mesh parameters

The meshing was done using quadratic quadrilaterals and the initial focus was on the area where the contact area is going to be formed be formed. From Table 3.8 and Table 3.9, we

know that the height of the rectangular contact area is 0.099 for position 6 and 0.062 for position 6'. So the element size was selected to be $3 \cdot 10^{-3}$ to have at least 20 elements in the area. It is crucial to understand that if the element size in this area is not at least a tenth of the length of the heights then the mesh cannot fully model the pressure distribution in the area and the model will yield wrong results. This is performed using the option of Face Sizing combined with the option of Sphere of Influence, otherwise we would end up with a fine mesh in all the gear profile which is unnecessary and would cost more computational time. The radius of the Sphere of Influence was set to be a little higher that the half of the heights of the contact areas. The basic criterion for how much bigger the radius depends on how much of the most loaded area is in the fine mesh. For example, if a big part of the most loaded area is outside of the fine mesh, then the radius should increase to capture that part also and observe if there is a significant change in the results. Furthermore, a total refinement in the mesh of the gear is done to create mesh whose element size changes more gradually as we move from the gear hub to the contact area. In Picture 3.9(a), (b).the initial mesh is shown:



Picture 3.8(a)



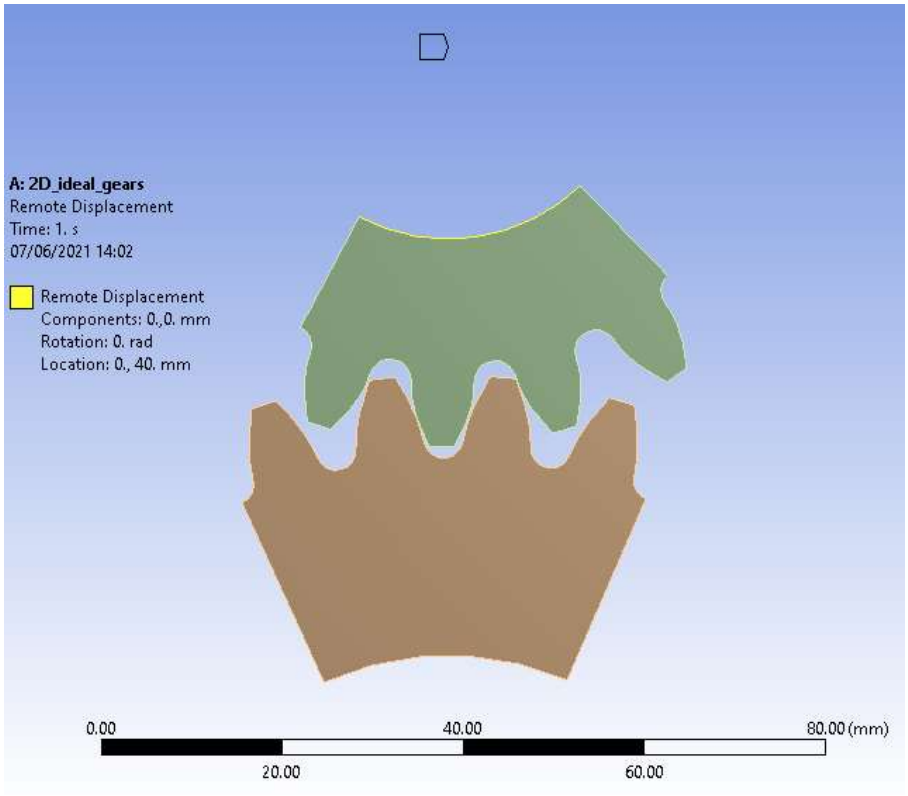
Picture 3.8(b)

(a) Initial mesh in Static Structural, (b) Mesh around the contact area.

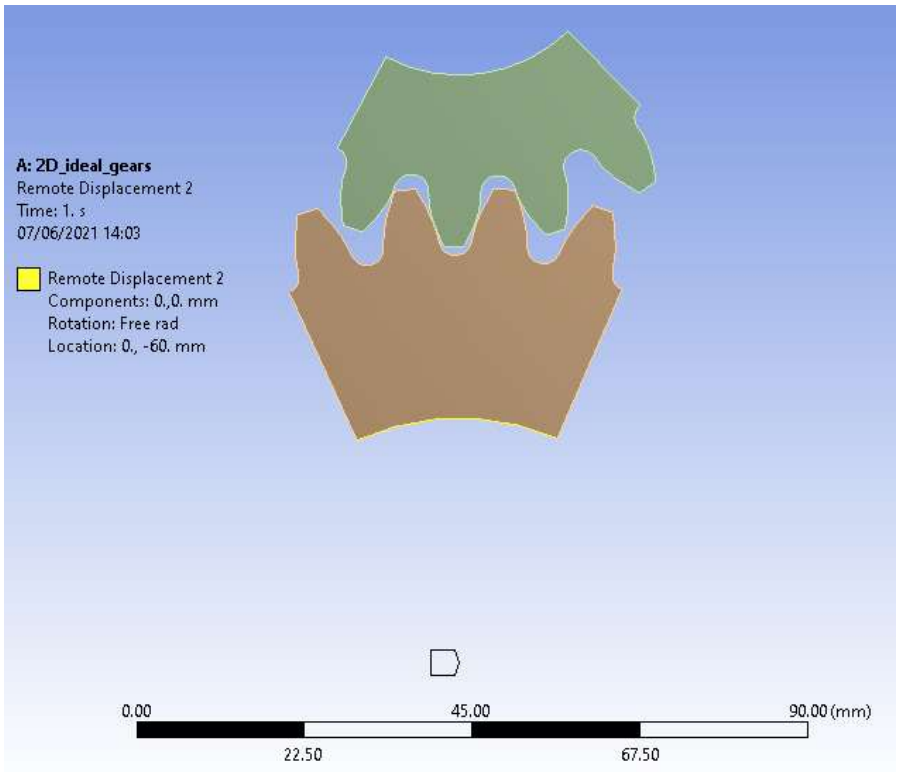
Finally, someone could argue that another area of interest is the trochoid of the gear where we expect to see the maxim bending stress being developed. That would be right but after the previous mesh refinement we already have a sufficiently small element size in these areas too, but if there is a problem there, it will be corrected in the mesh convergence that would be presented in the las paragraph of this chapter.

3.1.6 Application of boundary conditions and load

The boundary conditions that are applied were the following. Firstly, we want to prevent the pinion gear from rotating and from moving axially or radially. The latter is already fixed since the analysis is two-dimensional. The radial and rotational displacement could easily be prevented if we just fix the edge of the gear hub by using the option of Fixed Support, Displacement or Cylindrical Support. Even if this way of modelling does not make much difference in the results it is not sufficient because, we would like to fix the centre of the gear and not the edge of the hub. To do that we use the option of Remote Displacement that allows us to impose the displacement and rotation values in a Remote Point and in this case the centre of the gear, and we can select the edge of the hub to be correlated with this point. So, for the gear we set and the rotation around Z axis to be zero. For the gear in collaboration, we also set displacements along X axis and Y axis to be zero but the rotation around Z axis will be left free because in this gear the moment is applied. The boundary conditions of the two gears are shown in Picture 3.9(a), (b).



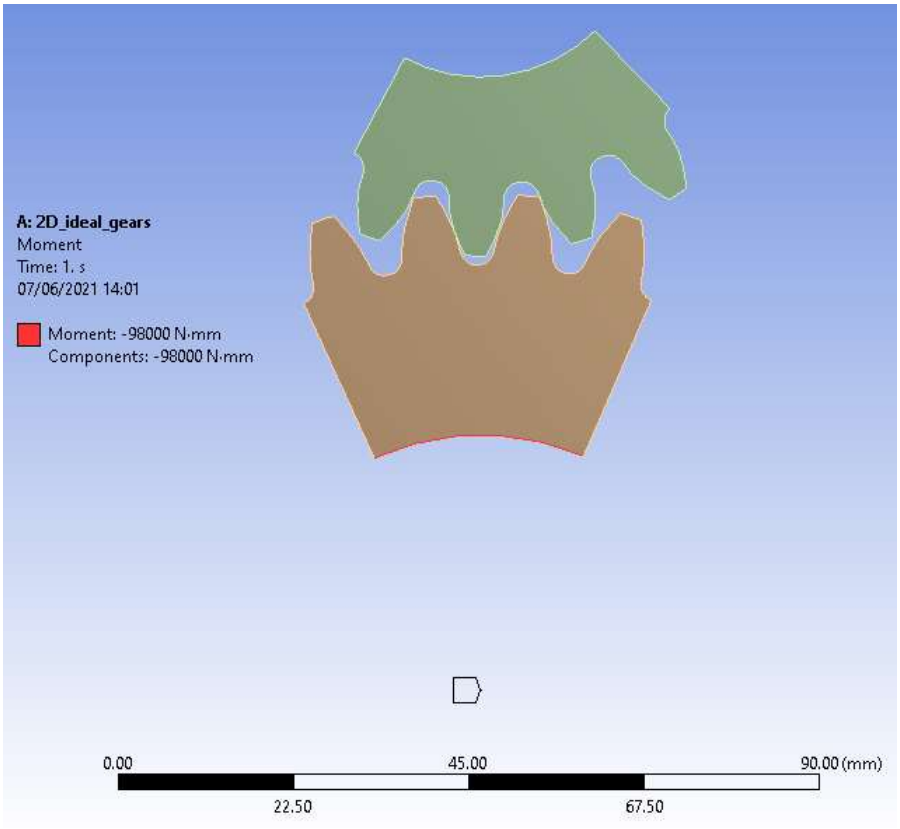
Picture 3.9(a)



Picture 3.9(b)

(a) Remote Displacement in the centre of pinion gear, (b) Remote Displacement in the centre of conjugate gear.

As far as the load is concerned, a moment of 98 Nm is applied in the Remote Point at the centre of the gear in collaboration. Again, the moment could also be applied directly to the edge of the hub of the gear without having much difference in the results, but this would be a less realistic way to model the problem. The load application is shown in Picture 3.10.



Picture 3.10. Moment application in conjugate gear.

3.1.7 Results and verification

After the simulation is run the results in Reaction Force, Equivalent Von-Mises Stress, Contact Pressure and Penetration in each contact pair will be presented. To be completely sure that the results are independent from the mesh in the areas of interest which are the contact areas and the trochoid gears, a mesh refinement procedure was performed to have a stress convergence in these areas. The Adaptive Mesh refinement can be enables by the inserting the Convergence option in the results that interest us and its parameters can change in the tool of Convergence and in the section Solution and are presented in Picture 3.11 and in Picture 3.12.

Details of "Convergence"	
Definition	
Type	Maximum
Allowable Change	5. %

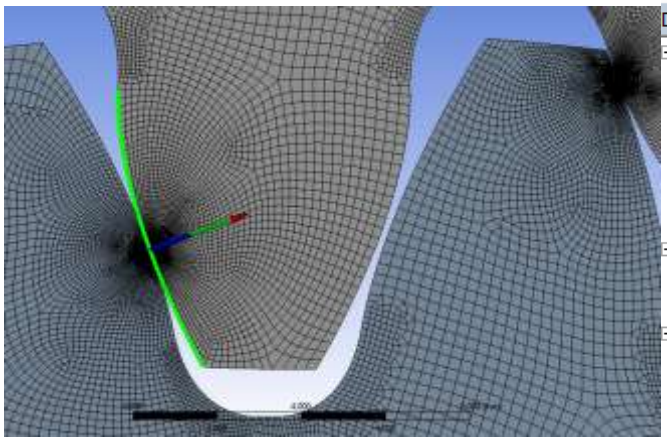
Picture 3.11. Paramteres of Convergence tool.

Details of "Solution (A6)"	
Solution	
Number Of Cores to Use (Beta)	Solve Process Settings
Adaptive Mesh Refinement	
Max Refinement Loops	3.
Refinement Depth	3.

Picture 3.12. Parameters of Adaptive Mesh Refinement.

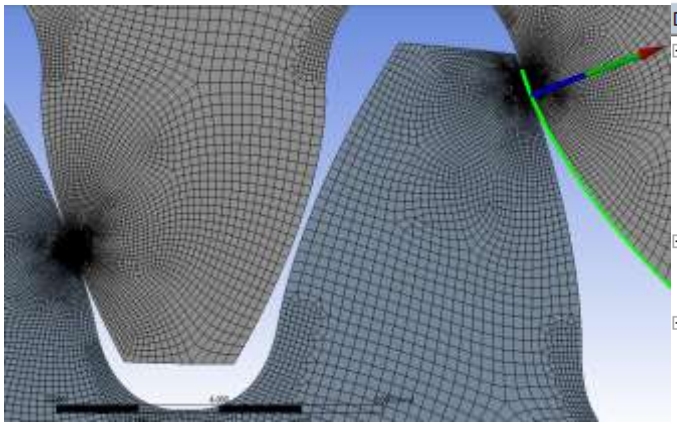
Reaction Force

The results of the Reaction Force are shown in Picture 3.13 and Picture 3.14. Their values match well with the theoretical ones and their actual error are presented in Table 3.12.



Details of "Force Reaction"	
Definition	
Type	Force Reaction
Location Method	Contact Region
Contact Region	Frictionless - 3D_pinion To 3D_in_col...
Orientation	Global Coordinate System
Extraction	Contact (Underlying Element)
Suppressed	No
Options	
Result Selection	All
Display Time	End Time
Results	
X Axis	905.21 N
Y Axis	329.4 N
Total	963.28 N

Picture 3.13. Reaction Force at Position 6.



Details of "Force Reaction 2"	
Definition	
Type	Force Reaction
Location Method	Contact Region
Contact Region	Frictionless - 3D_pinion To 3D_in_col...
Orientation	Global Coordinate System
Extraction	Contact (Underlying Element)
Suppressed	No
Options	
Result Selection	All
Display Time	End Time
Results	
X Axis	728.11 N
Y Axis	265. N
Total	774.84 N

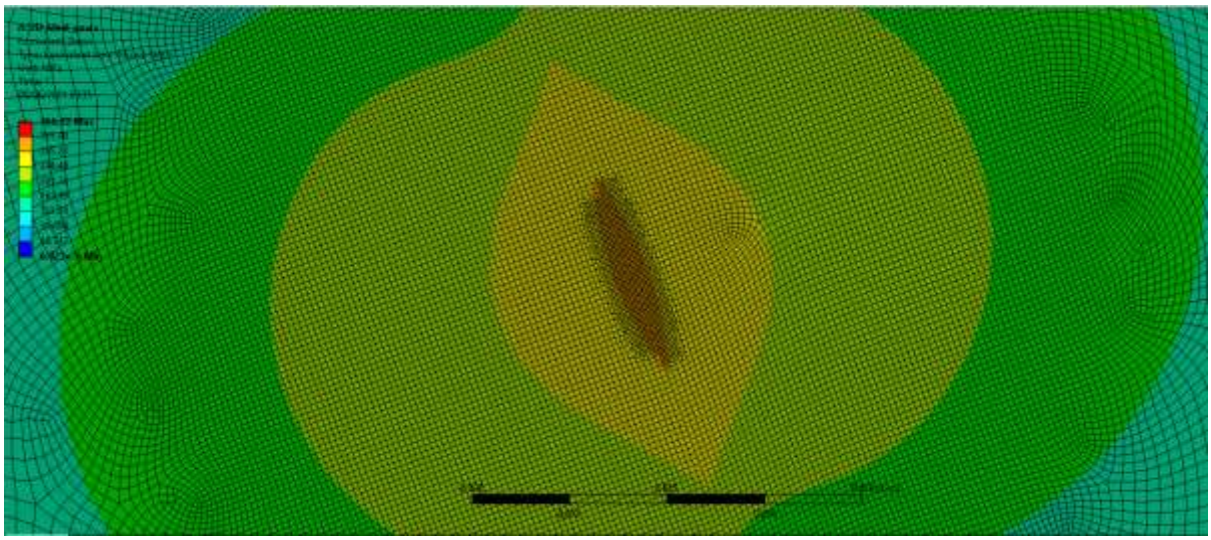
Picture 3.14. Reaction Force at Position 6'.

	Position 6	**Position 6'**
Reaction Force ANSYS (N)	963.28	774.84
Reaction Force Theoretical (N)	1,001.05	739.91
Error (%)	3.77	4.72

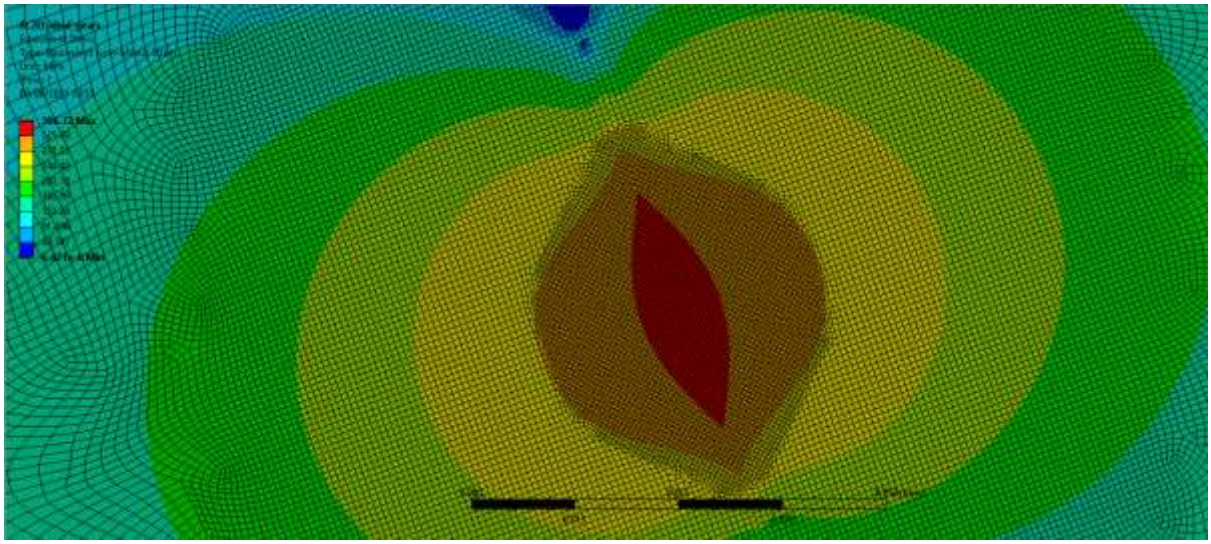
Table 3.12. Reaction Forces at each pair of contact.

Von Mises Stress

The results of Equivalent Von-Mises Stress are shown in Picture 3.15, Picture 3.16. As we can see, the stresses follow the elliptical distribution that we were expecting. Moreover, the most loaded area is located well inside the finer mesh, so this is another indication that the mesh is probably ok.

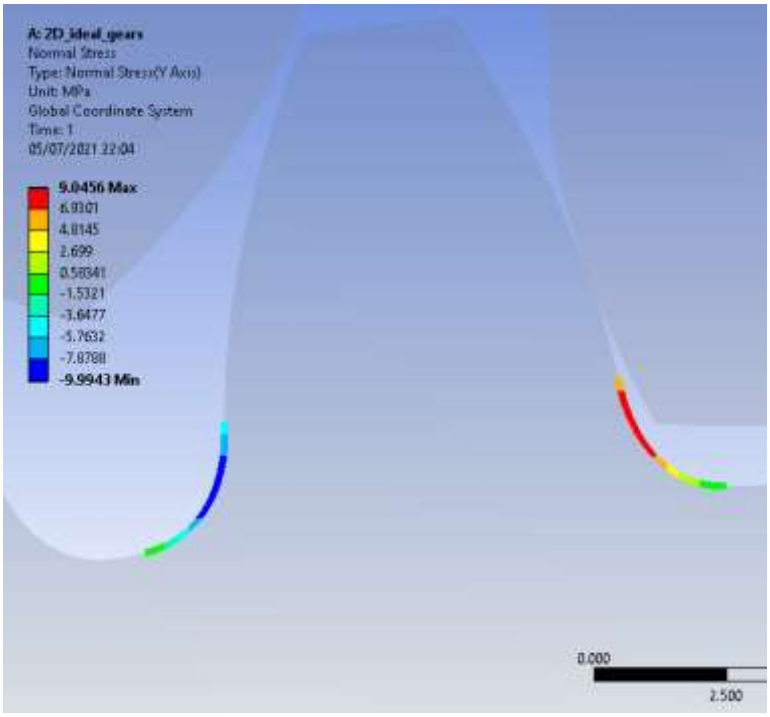


Picture 3.15. Equivalent Von-Mises stress in Position 6. Max value is 297.25 MPa.

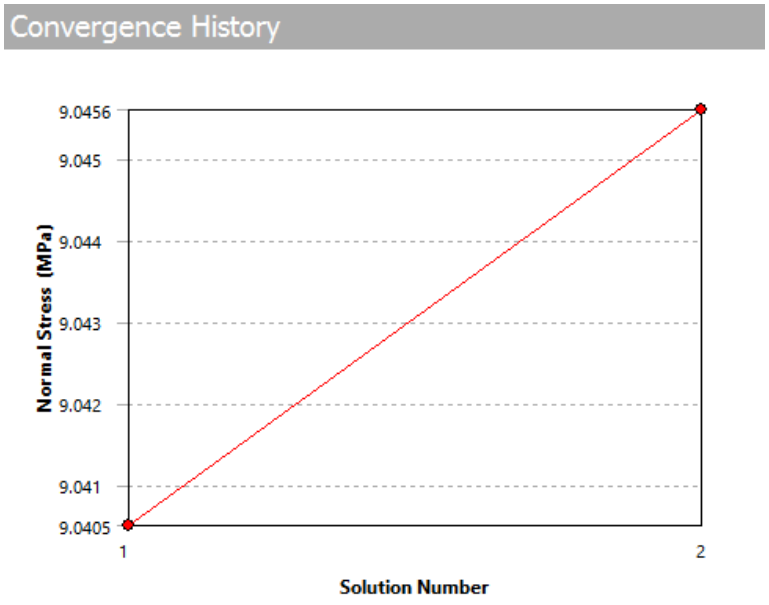


Picture 3.16. Equivalent Von-Mises stress in Position 6'. Max value is 366.72 MPa.

Finally in Picture 3.17 and in Picture 3.19 we can see the values of the stress in the foot of the conjugate gear. These values match well with the theoretical ones calculated in paragraph 3.1.1, if we also think that the theoretical values assume that the force is applied in the tip of the tooth. This was expected since the Reaction Forces also confirm the theoretical model. Moreover, we can see in Picture 3.18 and in Picture 3.20 that the mesh convergence is achieved. Their actual errors are presented in Table 3.13. For the calculation of errors, the average value from bending stress at each position was used.

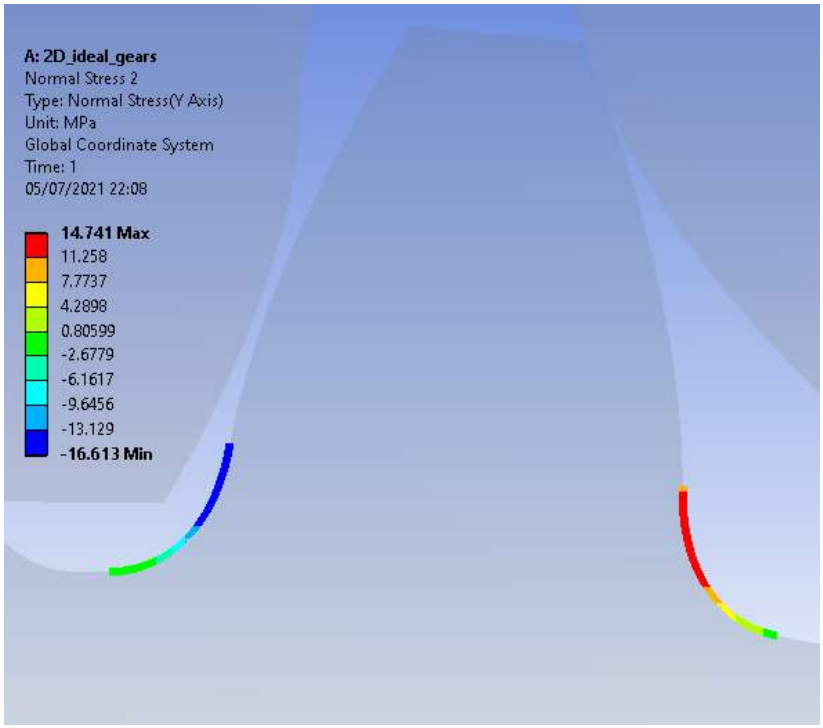


Picture 3.17. Bending stress in conjugate gear at Position 6.

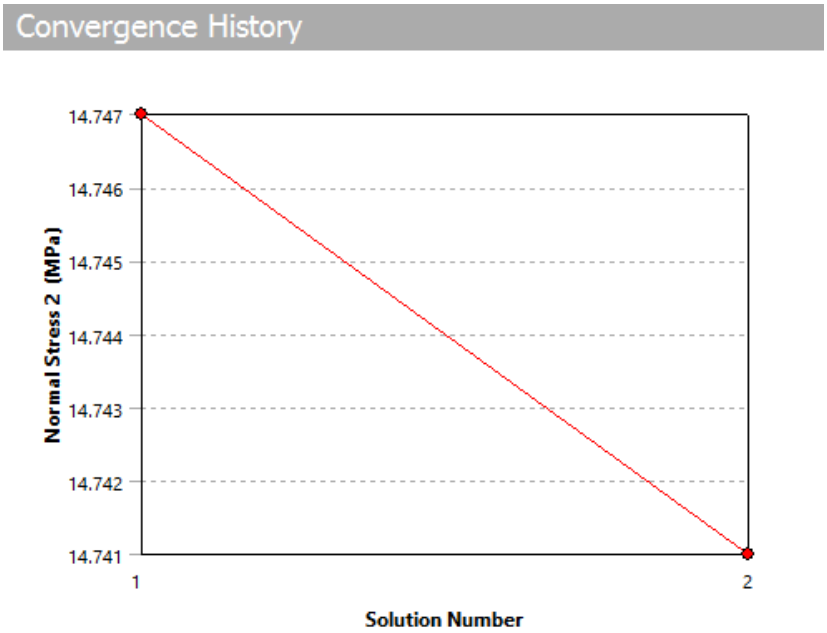


	Normal Stress (MPa)	Change (%)	Nodes	Elements
1	9.0405		306491	101365
2	9.0456	5.6811e-002	310046	102520

Picture 3.18. Convergence with Adaptive Meshing in tooth of gear in collaboration at Position 6.



Picture 3.19. Bending stress in conjugate gear at Position 6'.



	Normal Stress 2 (MPa)	Change (%)	Nodes	Elements
1	14.747		306491	101365
2	14.741	-3.7803e-002	310046	102520

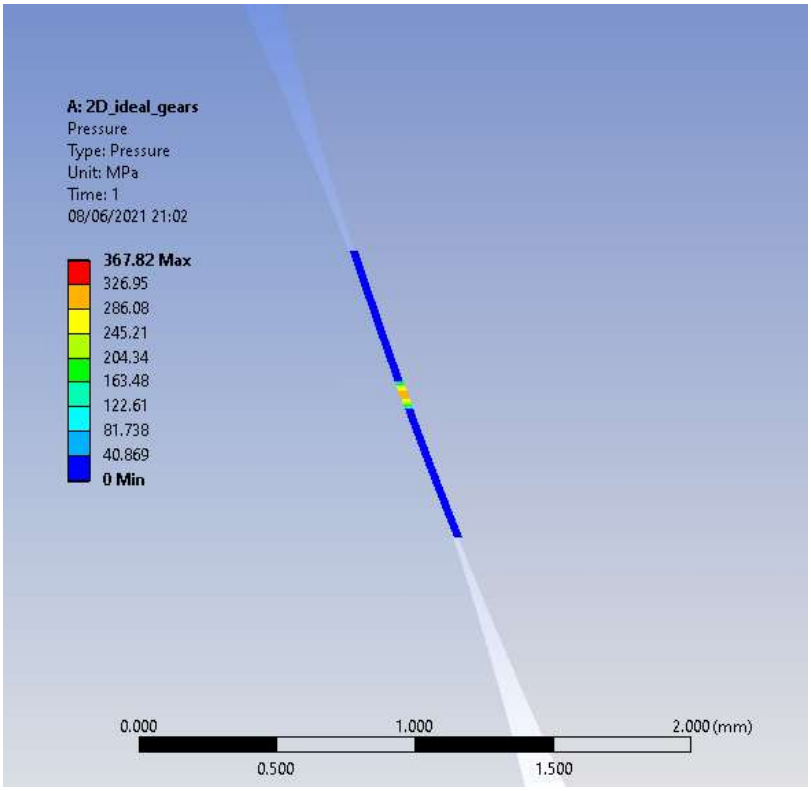
Picture 3.20. Convergence with Adaptive Meshing in tooth of gear in collaboration at Position 6'.

	Position 6		Position 6’	
	Left	**Right**	**Left**	**Right**
Bending stress ANSYS (MPa)	9.99	9.05	16.61	14.74
Bending stress Theoretical (MPa)	15.29		11.30	
Error (%)	37.74		38.7	

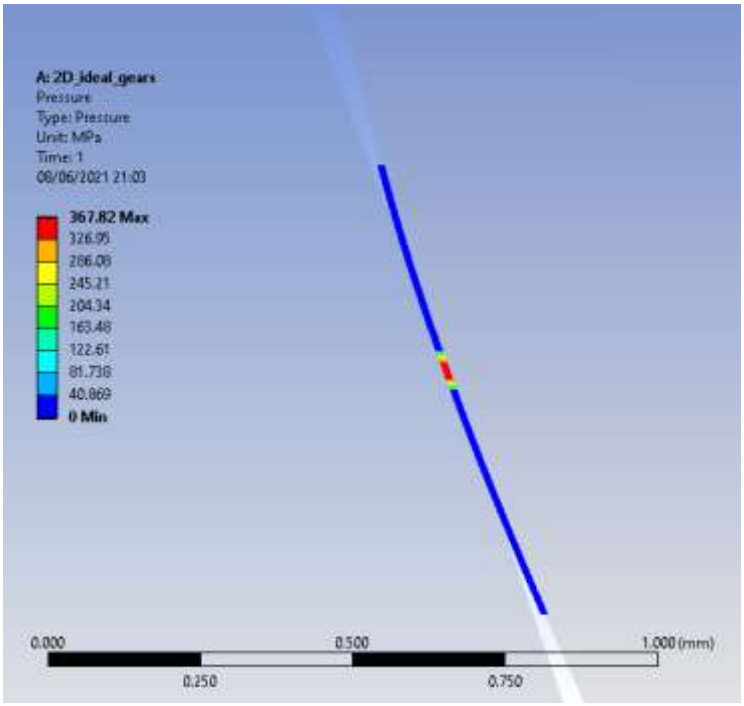
Table 3.13. Bending stress of conjugate gear at each contact pair.

Contact Pressure

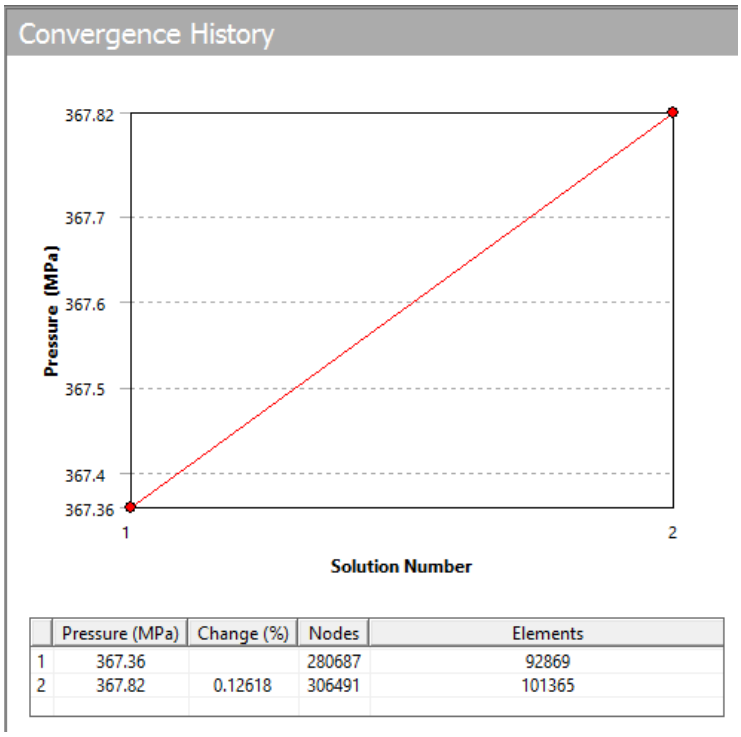
The results of the Contact Pressure are shown in Picture 3.21 and Picture 3.22. Again, we expect that also the values of contact pressure will be close to the theoretical ones. Indeed, the maximum values match well with the theoretical ones their actual error are presented in Table 3.14. Furthermore, we see in picture 3.23 that the mesh convergence is achieved.



Picture 3.21. Contact Pressure in Position 6.



Picture 3.22. Contact Pressure in Position 6'.



	Pressure (MPa)	Change (%)	Nodes	Elements
1	367.36		280687	92869
2	367.82	0.12618	306491	101365

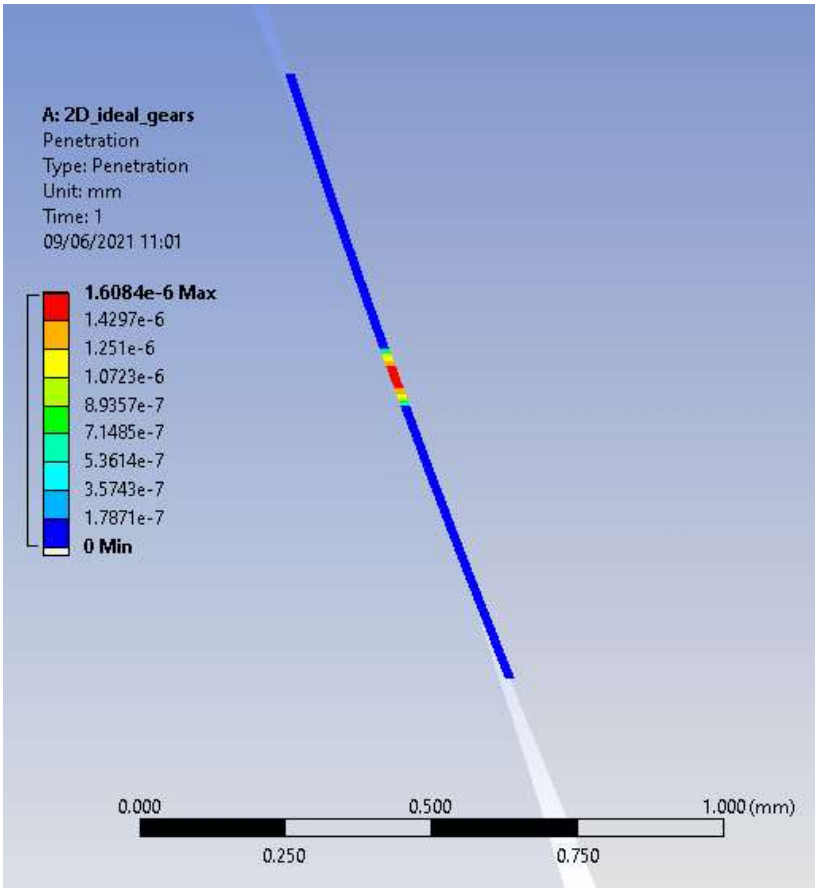
Picture 3.23. Convergence with Adaptive Meshing in Contact Pressure.

	Position 6	Position 6'
Contact Pressure ANSYS (MPa)	299.99	367.82
Contact Pressure Theoretical (MPa)	319.98	377.32
Error (%)	6.25	2.52

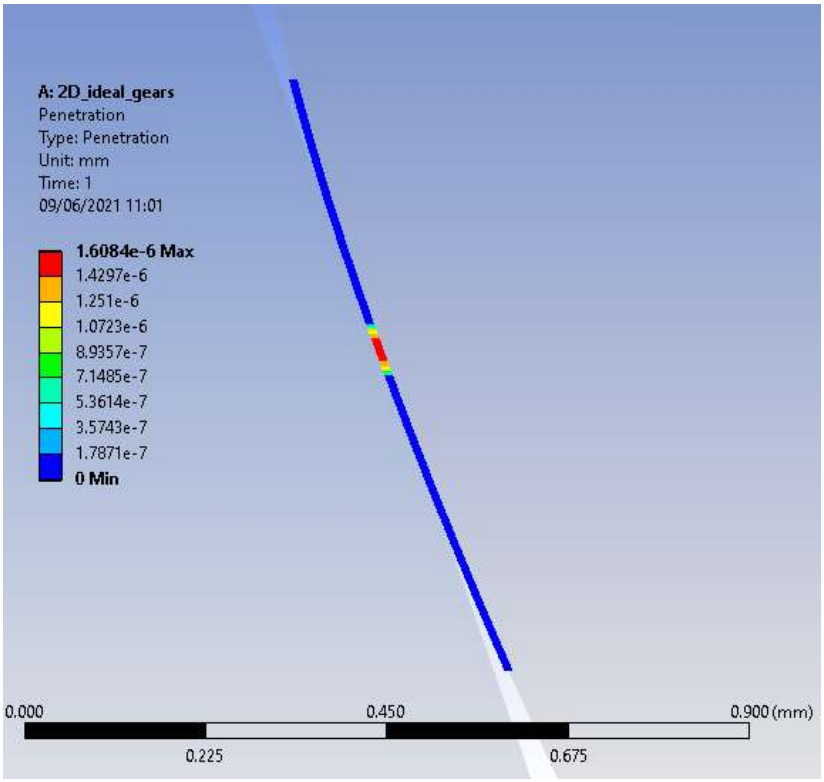
Table 3.14. Contact Pressure at each contact pair.

Penetration

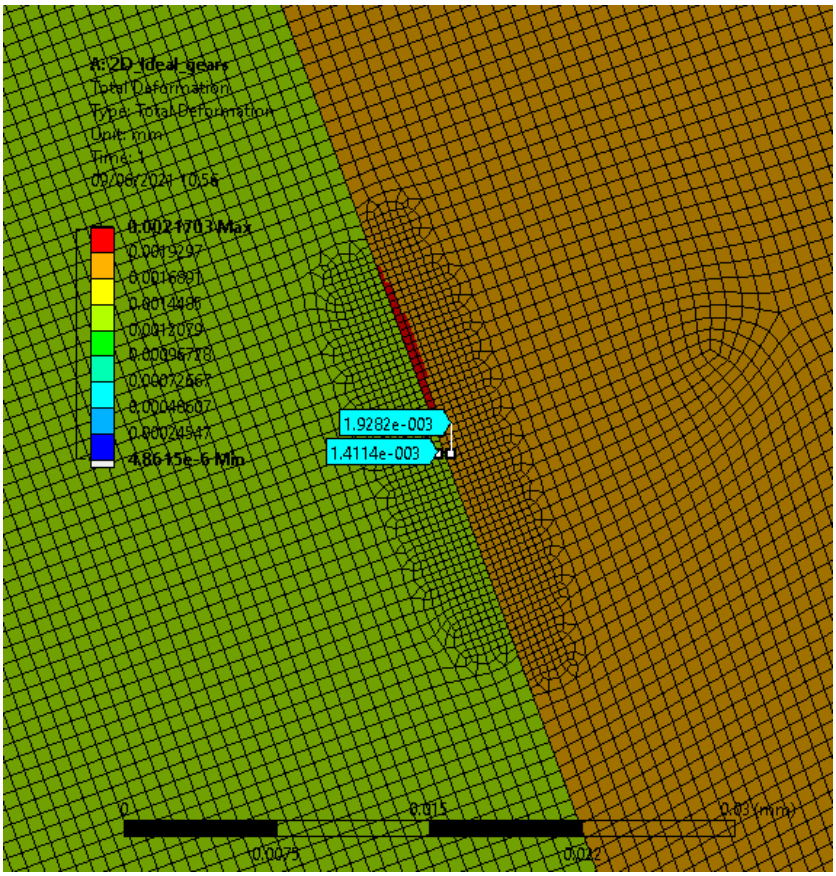
The results of the Penetration are shown in Picture 3.24 and in Picture 3.25. To verify that the selection of the value for the Normal Stiffness Factor is ok we must compare the values of Penetration to the values of Displacement in the contact areas which are shown in Picture 3.26 and Picture 3.27. For the calculation of errors, the average value from total displacement at each position was used.



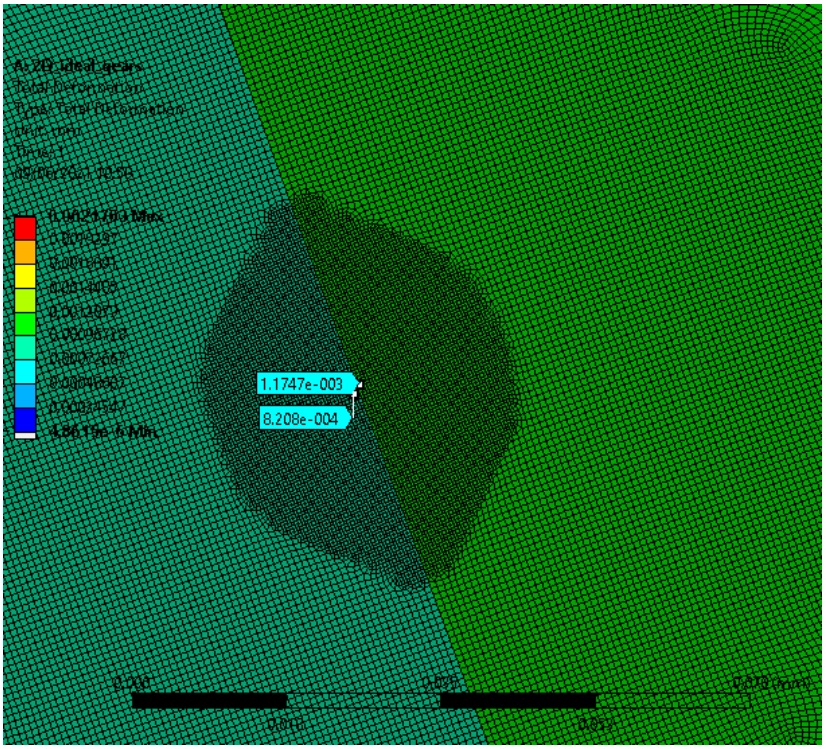
Picture 3.24. Penetration in position 6.



Picture 3.25. Penetration in position 6'.



Picture 3.26. Total Displacement in position 6 in each gear.



Picture 3.27. Total Displacement in position 6' in each gear.

	Position 6		Position 6'	
	Conjugate gear	**Pinion**	**Conjugate gear**	**Pinion**
Total Displacement ANSYS (mm)	$1.41 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	$8.21 \cdot 10^{-4}$	$1.17 \cdot 10^{-3}$
Penetration ANSYS (mm)	$1.56 \cdot 10^{-6}$		$1.61 \cdot 10^{-6}$	
Penetration Percent (%)	0.09		0.16	

Table 3.15. Penetration at each contact pair.

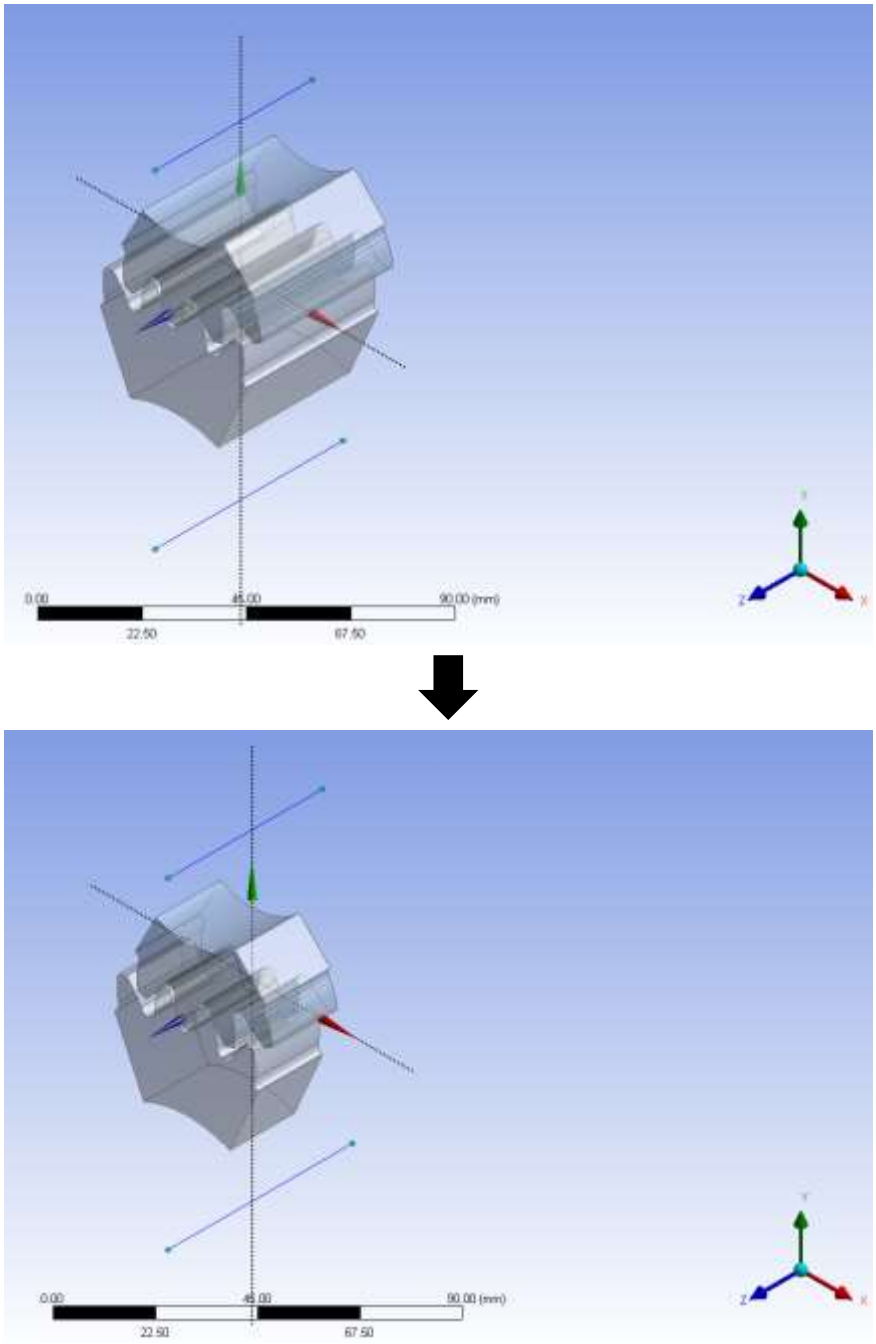
As we can see in Table 3.15, the maximum Penetration is 0.09% of the Displacement in contact pair of position 6 and 0.16% in contact pair of position 6'. These are both acceptable values so Normal Stiffness Factor value is also acceptable.

3.2 Three-dimensional modelling of involute spur gears

The setting up of the three-dimensional analysis of involute spur gears resembles with the one described in the two-dimensional analysis except some changes in the meshing process and in the application of boundary conditions.

3.2.1 Pre-processing in Design Modeler

To save computational time by reducing the number of nodes and elements in our mesh the Symmetry Tool is used in Design Modeler. In particular, the gears are divided in half in the direction of their width and a symmetry condition is applied in the plane that divided them which is the XY-plane. The process is shown in Picture 3.28.



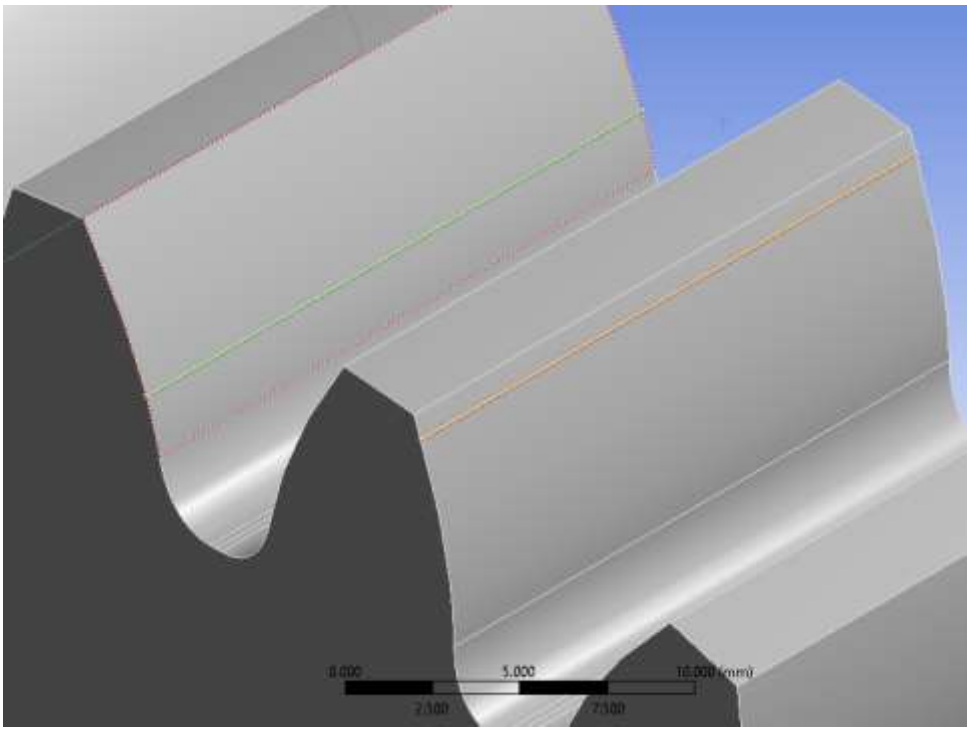
Picture 3.28. Application of symmetry condition.

Apart from the benefits in meshing process, the symmetry condition also allows to leave free the displacement in Z-direction and not put another constraint in it.

3.2.2 Setting up of the mesh parameters

To have the same mesh quality as in two-dimensional mesh some modifications are necessary. First, for the finer meshing of the contact area the option of Body of Influence is used (instead of Sphere of Influence in two-dimensional analysis) which can be enabled if in the Mesh menu the Solver changes from Mechanical to CFD. This tool makes things a lot easier since we create a new body in Design Modeler, and more specifically a cylinder with radius equal to the radius

of Sphere of Influence in two-dimensional modelling and use it to mesh the whole contact area in each contact pair as it is shown in Picture 3.29.



Picture 3.29. The green and orange cylinders are the bodies used in the Body of Influence option.

Of course, this body is sacrificed or in other words it is not included in the analysis beyond the meshing process. Secondly, since the Refinement tool is not suitable for the three-dimensional modelling, the areas of trochoid were mesh separately by using the Edge Sizing option with an Element Size of 0.2 mm. Finally, in the three-dimensional analysis the bodies need also to be meshed in the Z-direction and for this purpose the Sweep Method option was used and the width of each gear was meshed with 160 elements or with an element length of 0.125 mm.

3.2.3 Application of boundary conditions and load

The major changes in this part of the analysis are in the application of displacement conditions. Again, the option of Remote Displacement applied in Remote Points is used but in the pinion the displacement in X-direction, Y -direction and the rotations around all the three axes are set to zero and in the gear in collaboration the displacement in X-direction, Y -direction and the rotation around X axis and Y axis are set to zeros and the rotation around Z axis is left free. Finally, concerning the load application, again the Moment tool is used but the load must only be applied in the Z-component of Moment and since we have used a symmetry condition it must be half of the load used in two-dimensional analysis; namely 49 Nm.

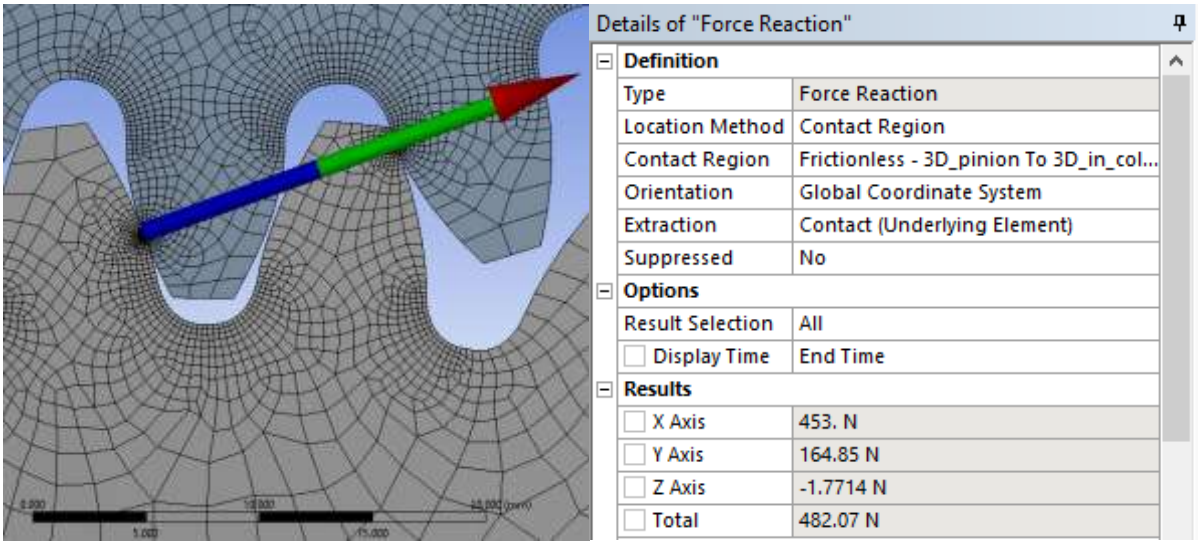
3.2.4 Results and verification

After the simulation is run the results in Reaction Force, Equivalent Von-Mises Stress, Contact Pressure and Penetration in each contact pair will be presented. In the three-dimensional analysis the refinement of the mesh by the Adaptive Meshing option was not used since the

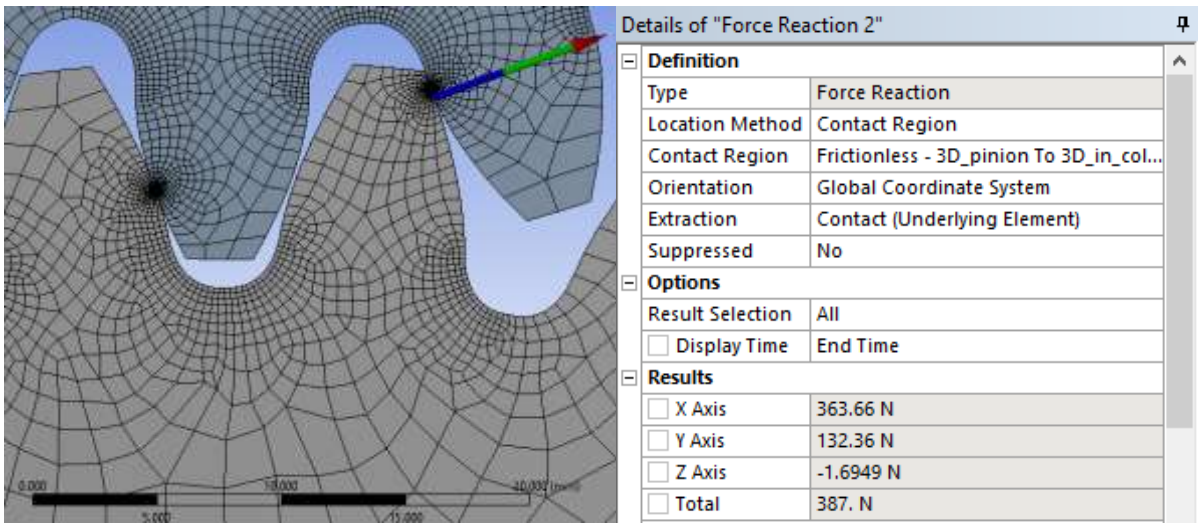
way of modelling the gears and the mesh independence was verified in the two-dimensional analysis.

Reaction Force

The results of the Reaction Force are shown in Picture3.30 and Picture 3.31. Their values match well with the theoretical ones and their actual error are presented in Table3.16. Of course, since there is a symmetry condition the Reaction Force in the results is the half of the actual force so we are comparing it to the half of the theoretical one.



Picture 3.30. Reaction Force at Position 6.



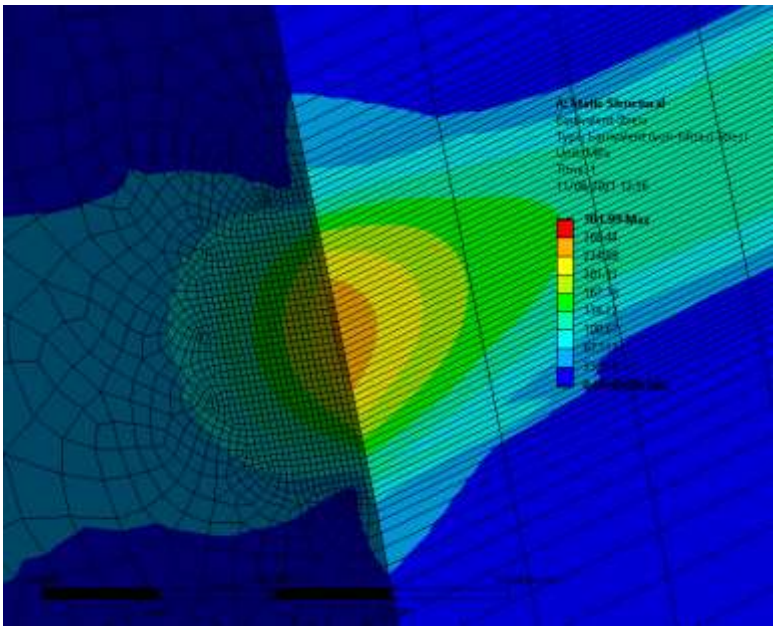
Picture 3.31. Reaction Force at Position 6'.

	Position 6	Position 6'
Reaction Force ANSYS (N)	482.07	387
Half of the Reaction Force Theoretical (N)	500.52	369.96
Error (%)	3.68	3.53

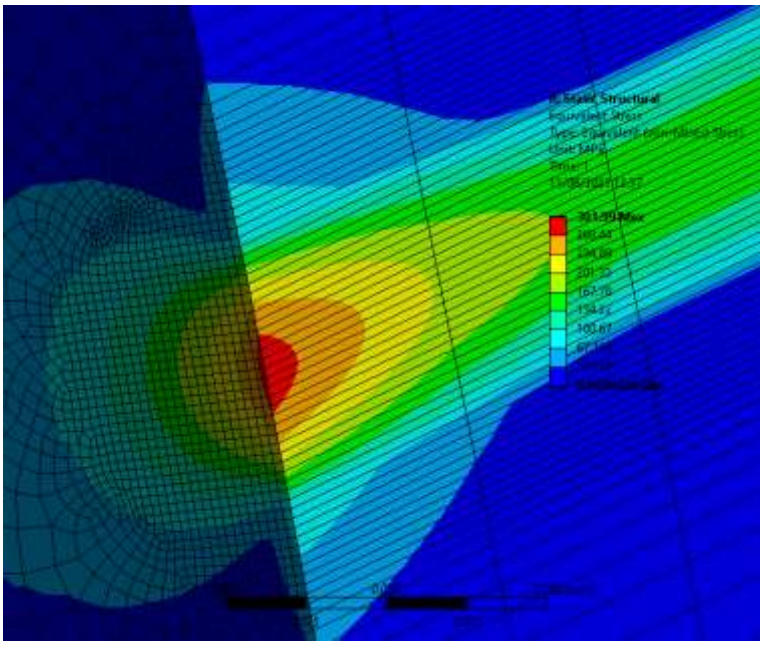
Table 3.16. Reaction Forces at each pair of contact.

Von Mises Stress

The results of Equivalent Von-Mises Stress are shown in Picture 3.32 and Picture 3.33. As we can see, the stresses follow the elliptical distribution that we were expecting. Moreover, the most loaded area is located well inside the finer mesh, so this is another indication that the mesh is probably ok.



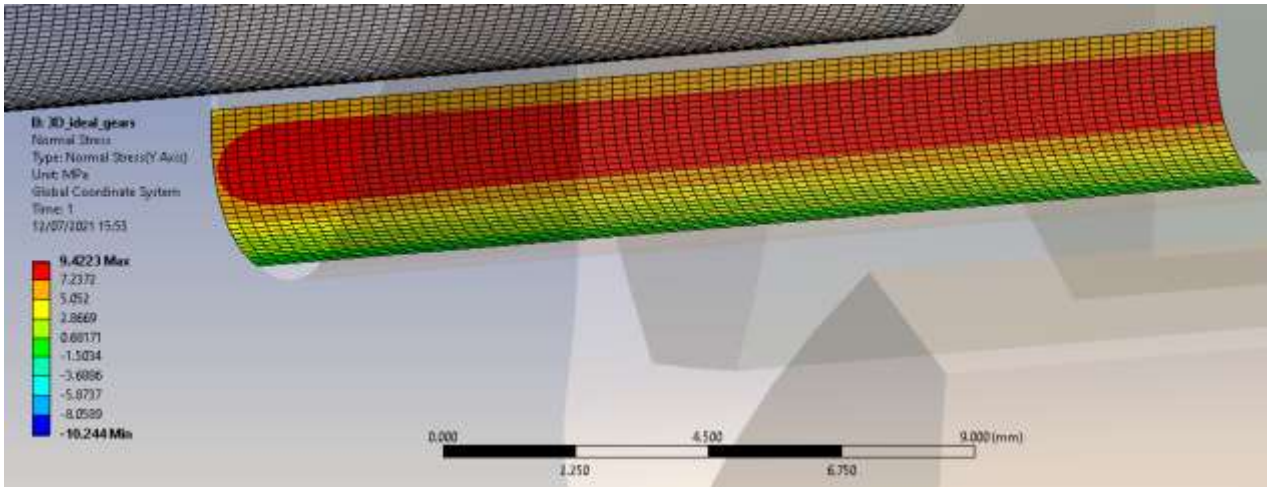
Picture 3.32. Equivalent Von-Mises stress in Position 6. Max value is 262.4 MPa.



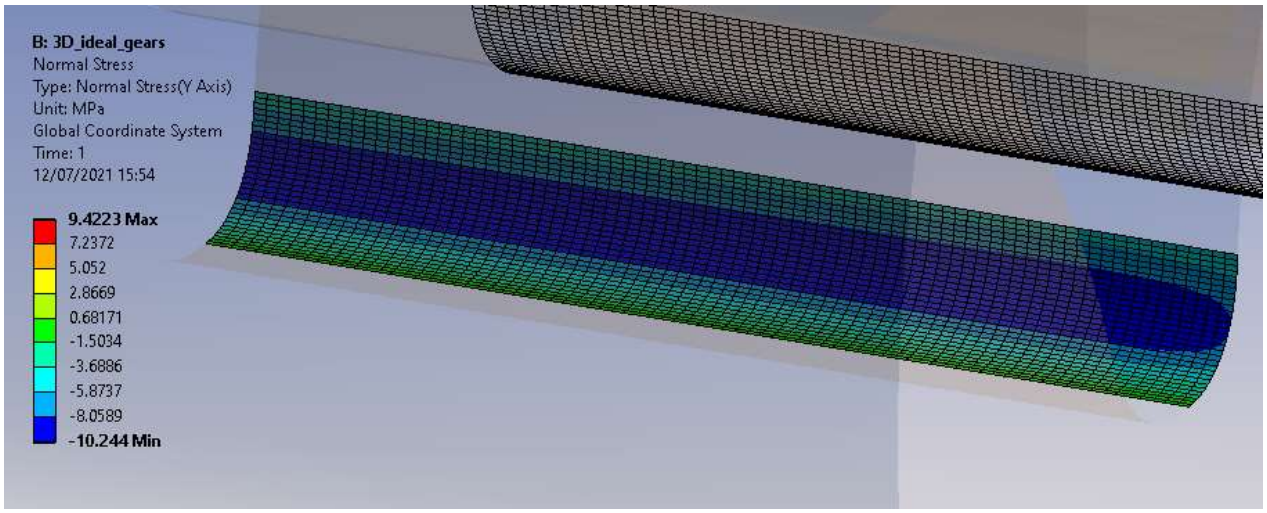
Picture 3.33. Equivalent Von-Mises stress in Position 6'. Max value is 301.99 MPa.

Finally in Picture we can see the values of the stress in the foot of the conjugate gear. These values match well with the theoretical ones calculated in paragraph. This was expected since the Reaction Forces also confirm the theoretical model. Their actual errors are presented in

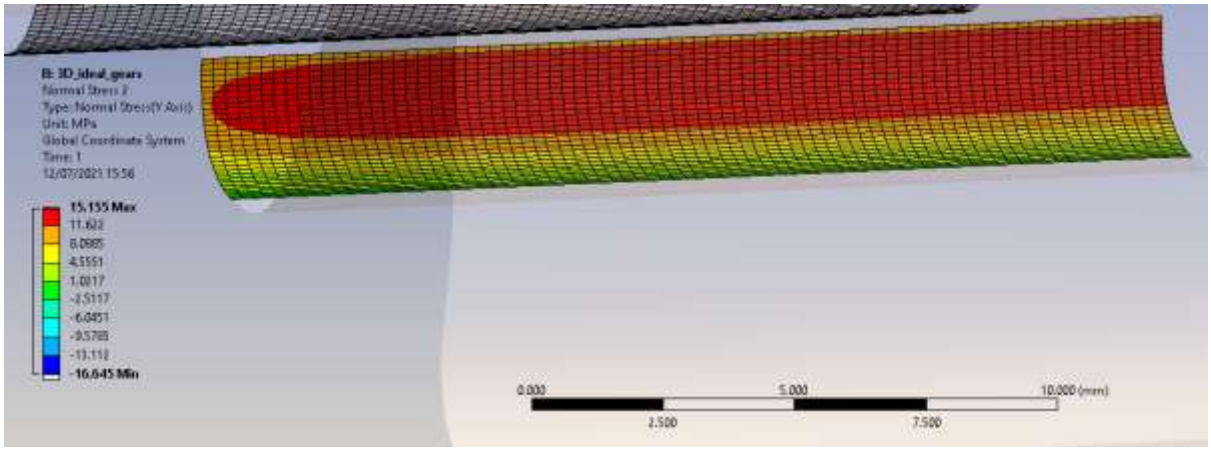
Table 3.17. For the calculation of errors, the average value of bending stress at each position was used.



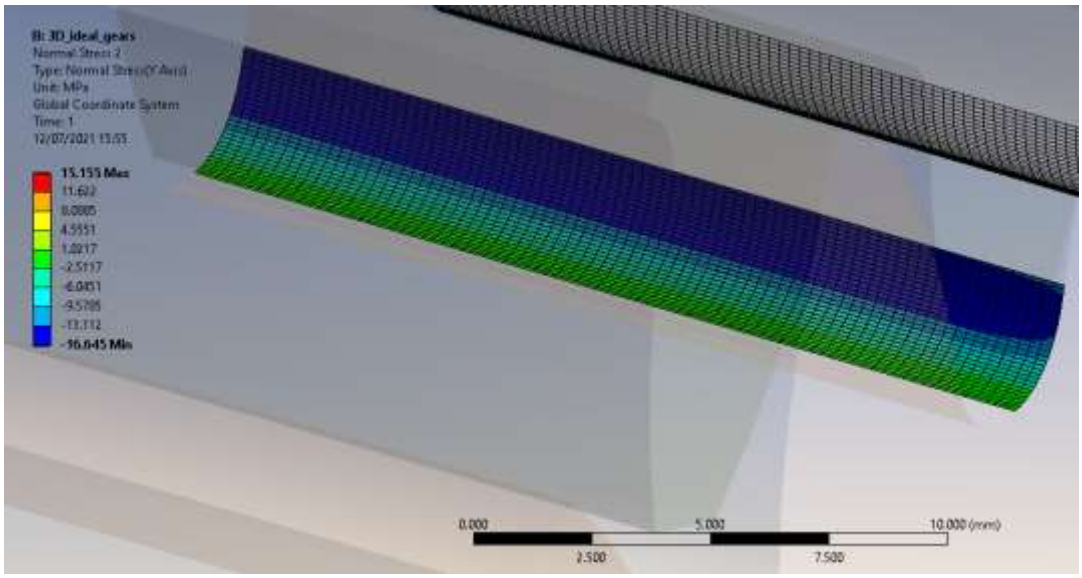
Picture 3.34. Bending stress in the right side of the tooth of gear in collaboration at Position 6.



Picture 3.35. Bending stress in the left side of the tooth of gear in collaboration at Position 6.



Picture 3.36. Bending stress in the right side of the tooth of gear in collaboration at Position 6'.



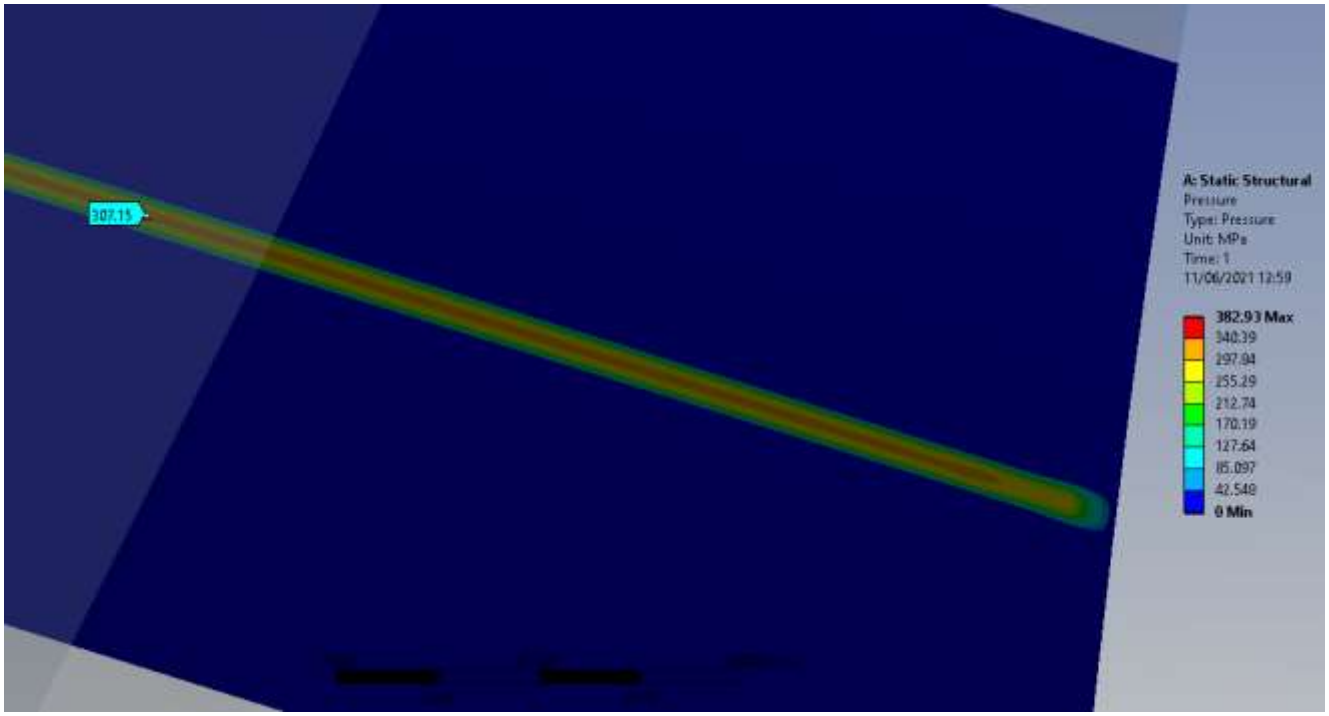
Picture 3.37. Bending stress in the left side of the tooth of gear in collaboration at Position 6'.

	Position 6		Position 6'	
	Left	**Right**	**Left**	**Right**
Bending stress ANSYS (MPa)	10.24	9.42	16.65	15.16
Bending stress Theoretical (MPa)	15.29		11.30	
Error (%)	35.71		40.75	

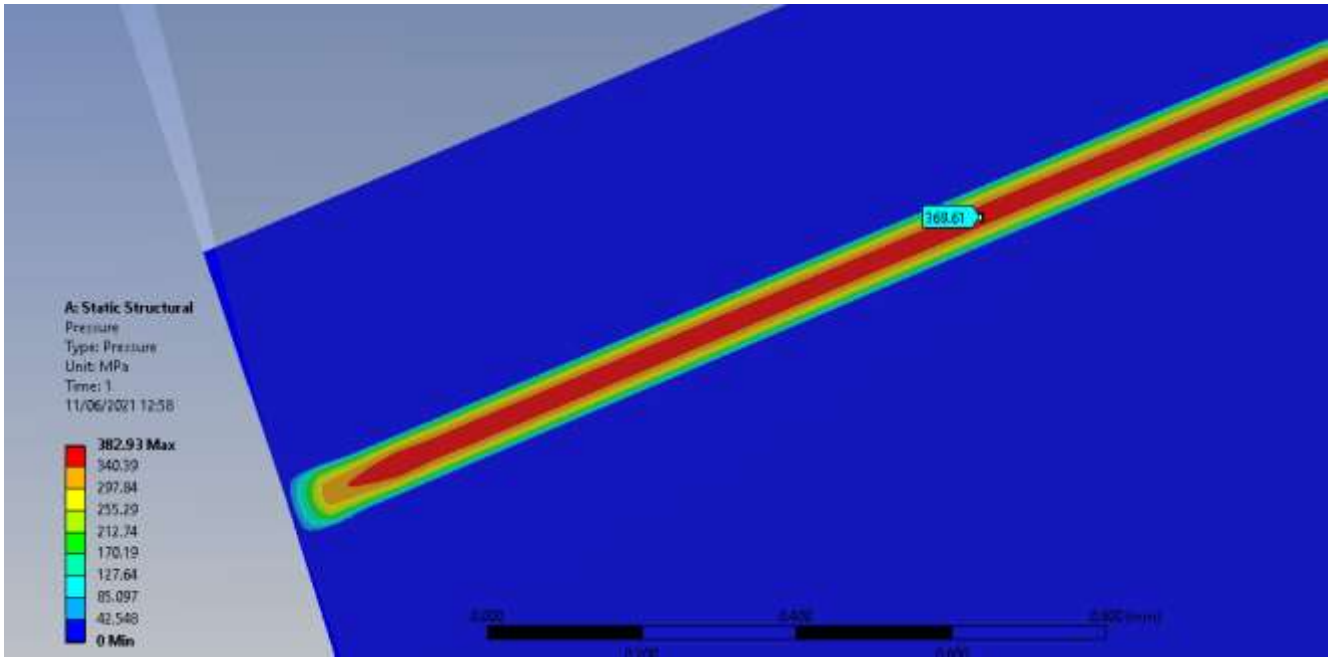
Table 3.17. Bending stress at each contact pair.

Contact Pressure

The results of the Contact Pressure are shown in Picture 3.38 and Picture 3.39. Again, we expect that also the values of contact pressure will be close to the theoretical ones. Indeed, the maximum values match well with the theoretical ones their actual error are presented in Table 3.18.



Picture 3.38. Contact Pressure in position 6.



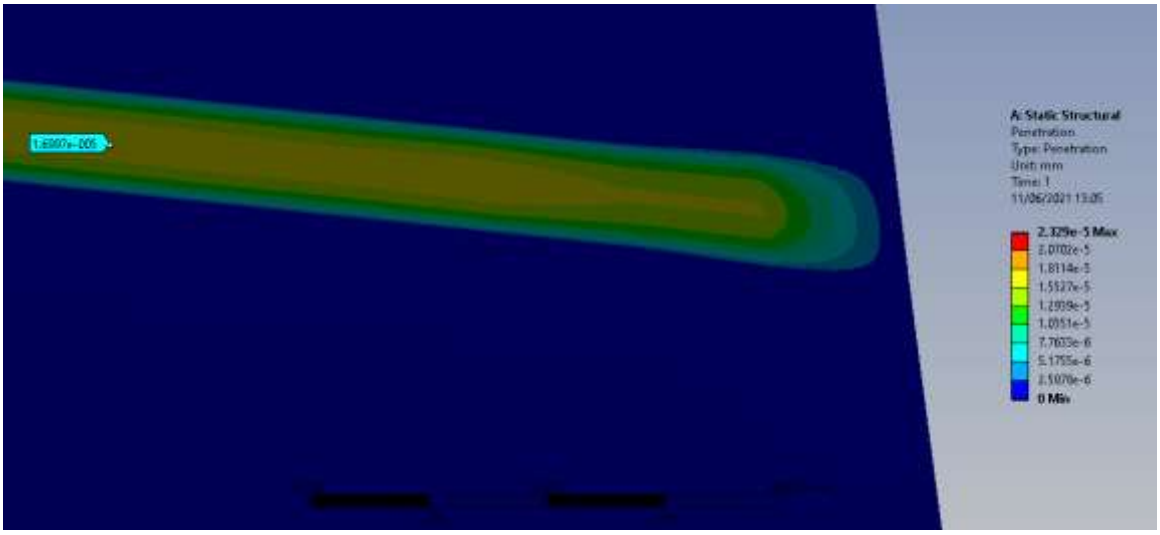
Picture 3.39. Contact pressure in position 6'.

	Position 6	Position 6'
Contact Pressure ANSYS (MPa)	307.15	369.61
Contact Pressure Theoretical (MPa)	319.98	377.32
Error (%)	4.01	2.04

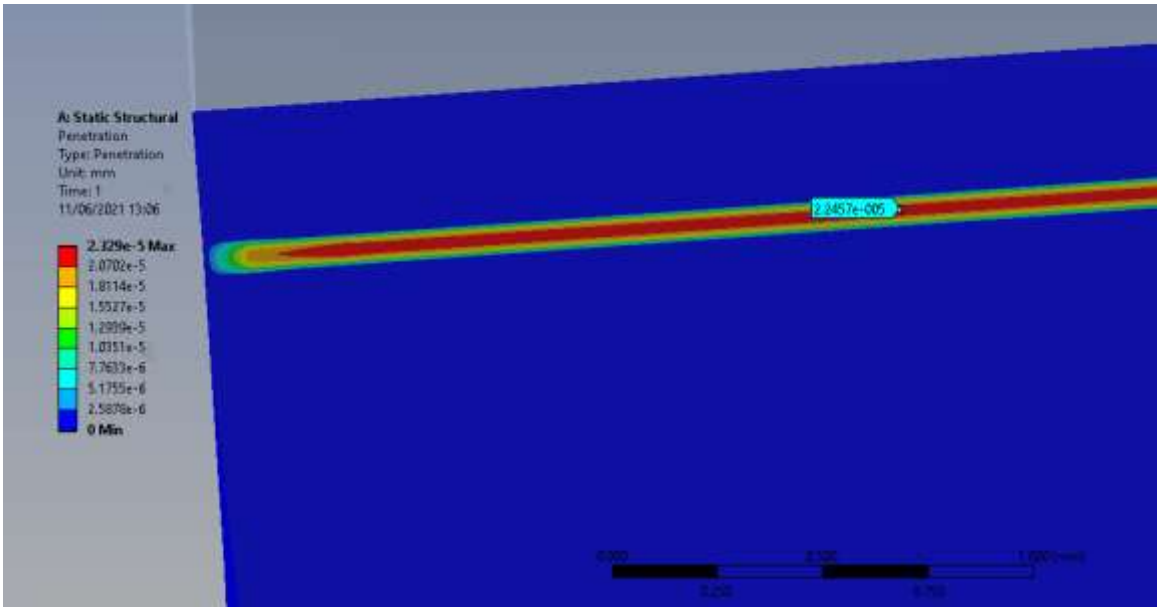
Table 3.18. Contact Pressure at each contact pair.

Penetration

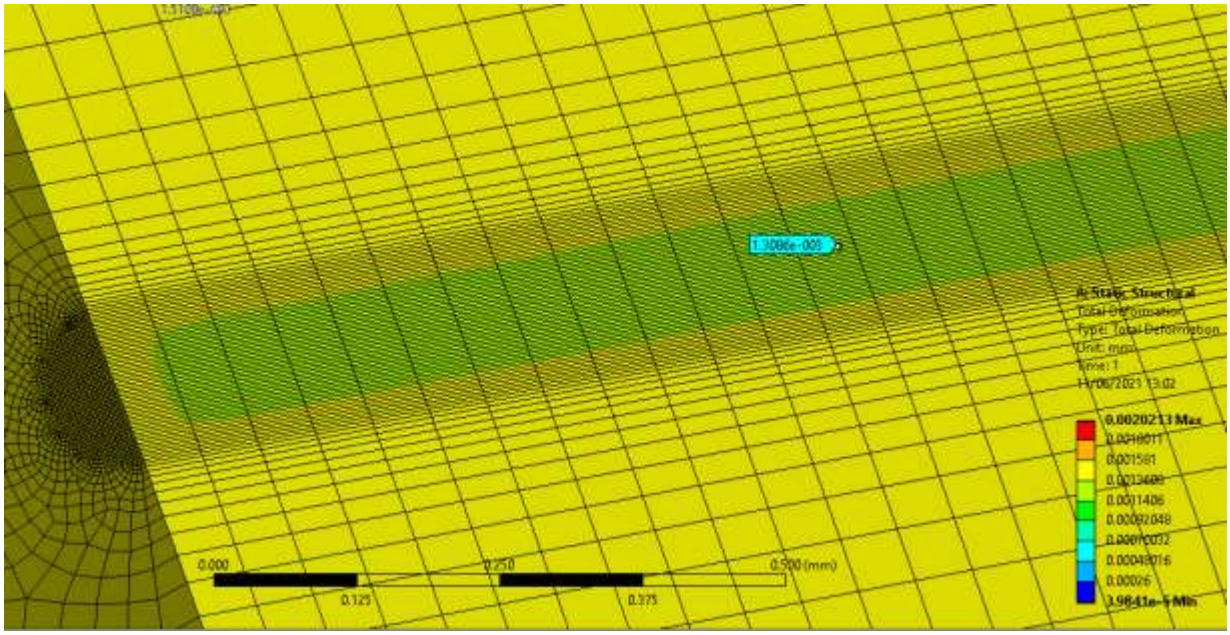
The results of the Penetration are shown in Picture 3.40 and Picture 3.41. To verify that the selection of the value for the Normal Stiffness Factor is ok we must compare the values of Penetration to the values of Displacement in the contact areas which are shown in Picture 3.42, Picture 3.43, Picture 3.44 and Picture 3.45. For the calculation of errors, the average value from total displacement at each position was used.



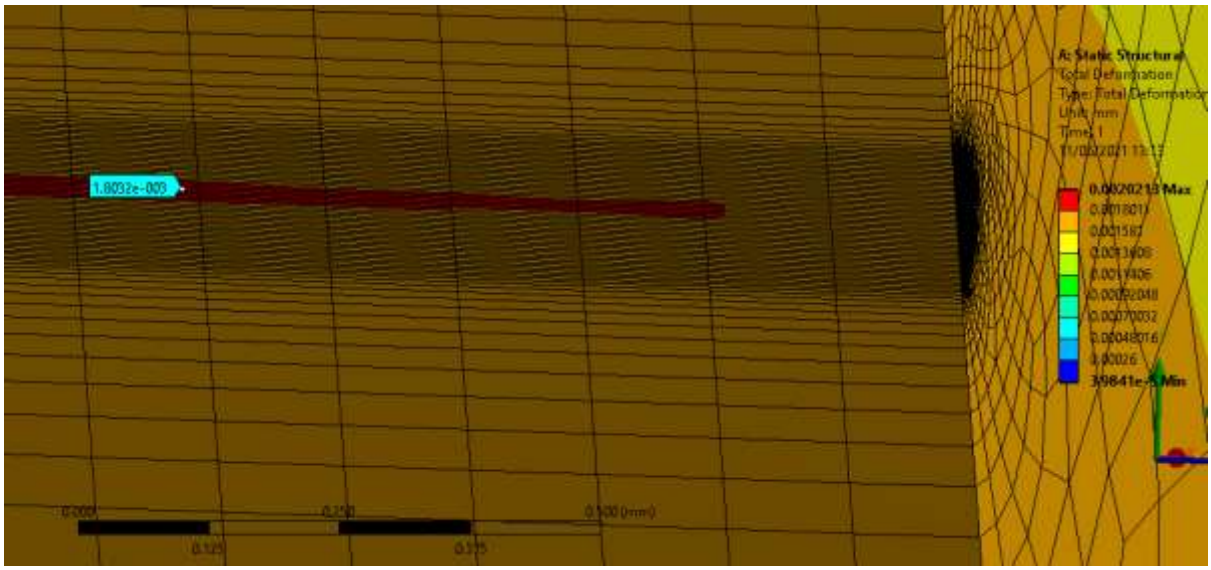
Picture 3.40. Penetration in position 6.



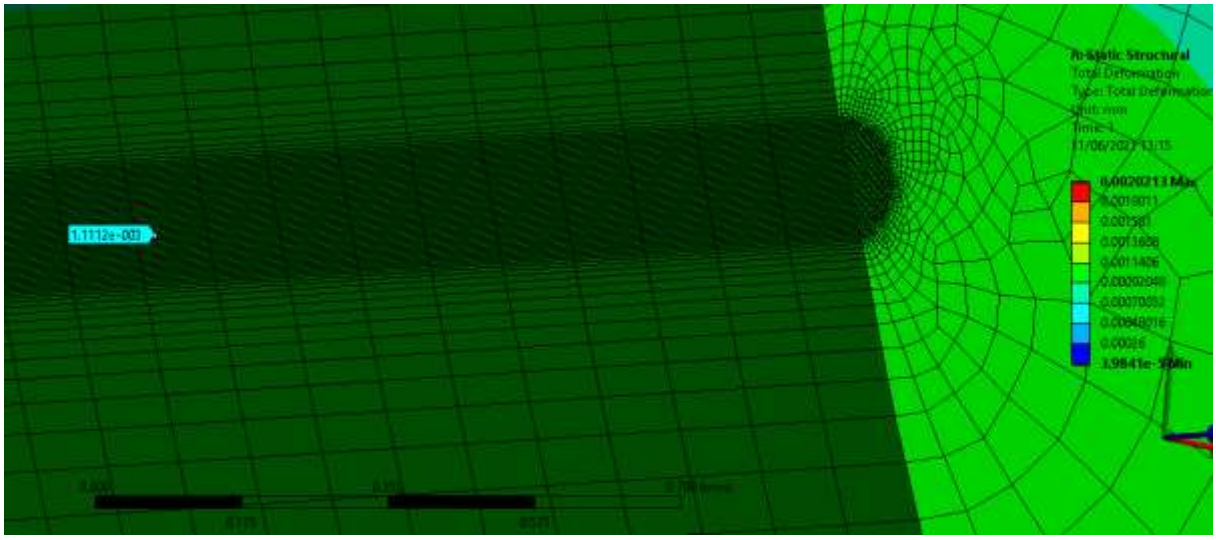
Picture 3.41. Penetration in position 6'.



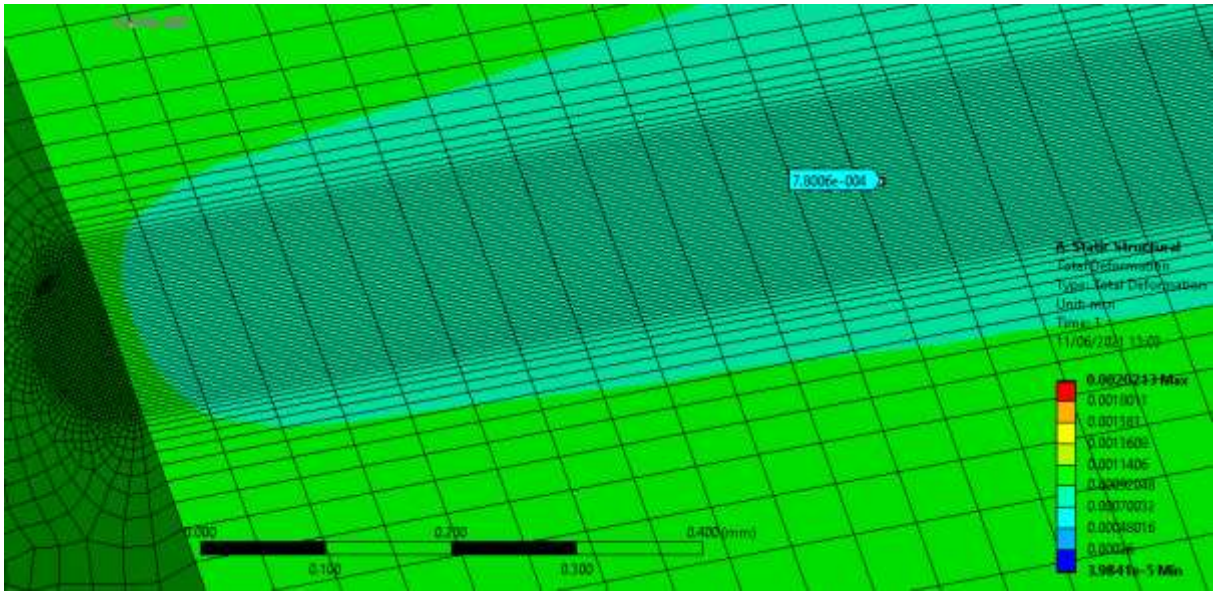
Picture 3.42. Total Displacement in position 6 of conjugate gear.



Picture 3.43. Total Displacement in position 6 of pinion.



Picture 3.44. Total Displacement in position 6' of pinion.



Picture 3.45. Total Displacement in position 6' of conjugate gear.

	Position 6		Position 6'	
	Conjugate gear	Pinion	Conjugate gear	Pinion
Total Displacement ANSYS (mm)	$1.31 \cdot 10^{-3}$	$1.80 \cdot 10^{-3}$	$7.8 \cdot 10^{-4}$	$1.11 \cdot 10^{-3}$
Penetration (mm)	$1.69 \cdot 10^{-5}$		$2.24 \cdot 10^{-5}$	
Penetration Percent (%)	1.09		2.54	

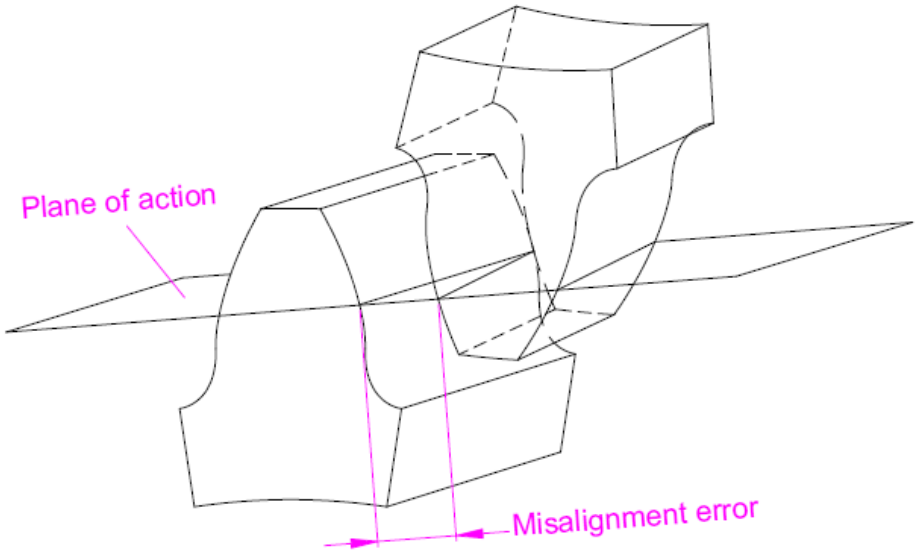
Table 3.19. Penetration at each contact pair.

As we can see in Table 3.19. Penetration is 1.09% of the Displacement in contact pair of position 6 and 2.54% in contact pair of position 6'. These are both acceptable values so Normal Stiffness Factor value is also acceptable.

3.3 Modelling of misalignment in involute spur gears

In this paragraph the modelling of misalignment error in involute spur gears is explained. First, a definition of this error must be given.

It is widely known that assembly errors of a pair of gears have significant effects on vibration, noise, and strength of them. One of these errors is the misalignment error of the gear shafts on the plane of action. The misalignment error of the gear shafts on the plane of action can be expressed by an inclination angle of the contact teeth on the plane of action as shown in Picture 3.46.

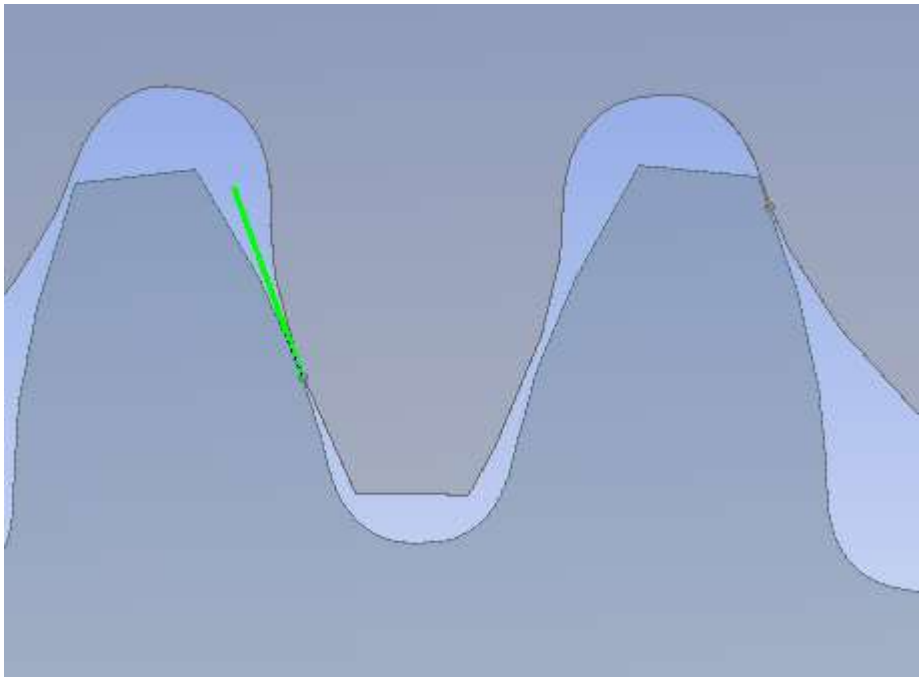


Picture 3.46. Misalignment error.[2]

The setting up of this simulation does not differ from the previous one about ideal fear contact (apart from the symmetry condition of course that cannot be applied in this case). Nevertheless, there is on necessary modification that must be made in the Design Modeler to create this misalignment in the assembly on purpose.

3.3.1 Modifications in Design Modeler

To create the misalignment error, one must follow the following steps. Frist of all from the definition of the error we know that one of the two gears-for instance here we choose the pinion-must be acquire a relative angle to the other gear in the direction of the plane of action. For this purpose, a line drawn at the point of contact in position which is perpendicular to the contact trajectory of the gears which as explained previously is a line that passes from the centre of the plane XY and has an incline of angle equal to a_o. This line will serve as rotational axis and is shown in Picture 3.47.



Picture 3.47. Design of rotation axis in Design Modeler.

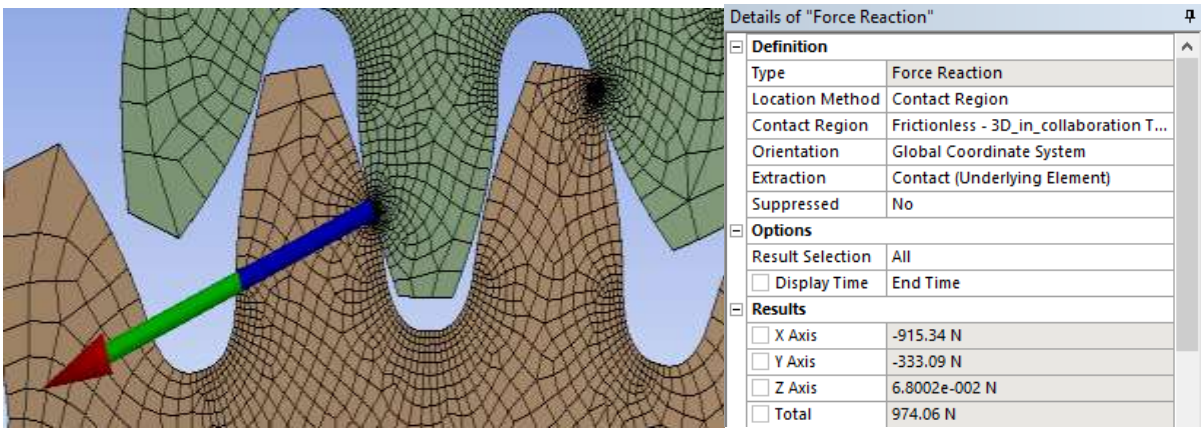
Secondly, by using the Rotate tool and giving as inputs the line created as axis of rotation and the value of 0.004 deg for the angle of rotation the misalignment error is created. Now the setting up of the model can continue with the setting up of the parameters discussed in the previous paragraphs.

3.3.2 Results and verification

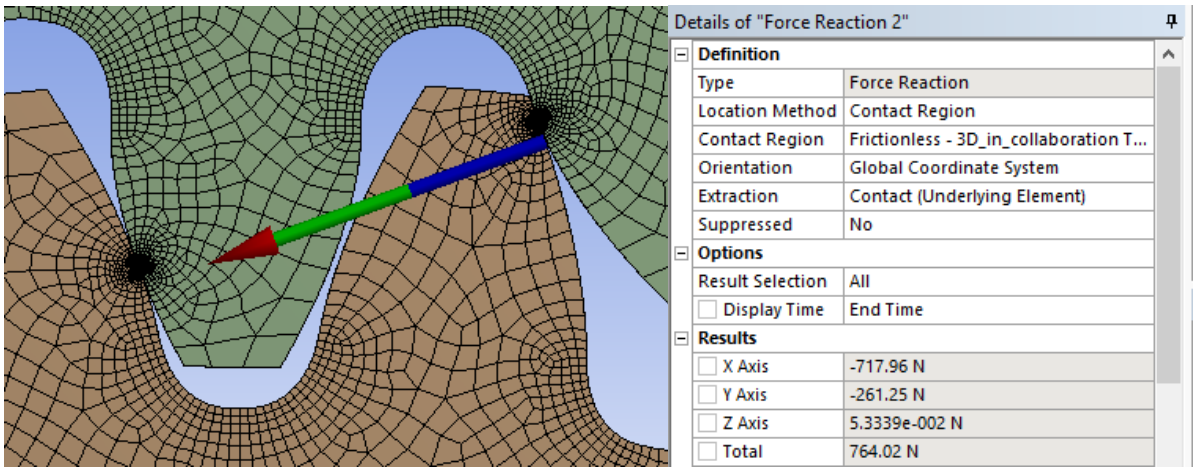
After the simulation is run the results in Reaction Force, Equivalent Von-Mises Stress, Contact Pressure and Penetration in each contact pair will be presented. In contrast to the other simulations, since the Hertz theory cannot easily be applied to the geometry of the crowned gears, the verification was intended to be done by comparing the results of the simulation with results of [2]. The results of [2] only provide information about the contact pressure which was found to be different than the which will be presented in the following paragraphs, but the stress filed is very similar. Therefore, for this simulation there will not be any verification provided other than the comparison of the stress fields.

Reaction Force

The results of the Reaction Force are shown in Picture 3.48 and Picture 3.49.



Picture 3.48. Reaction Force at Position 6.



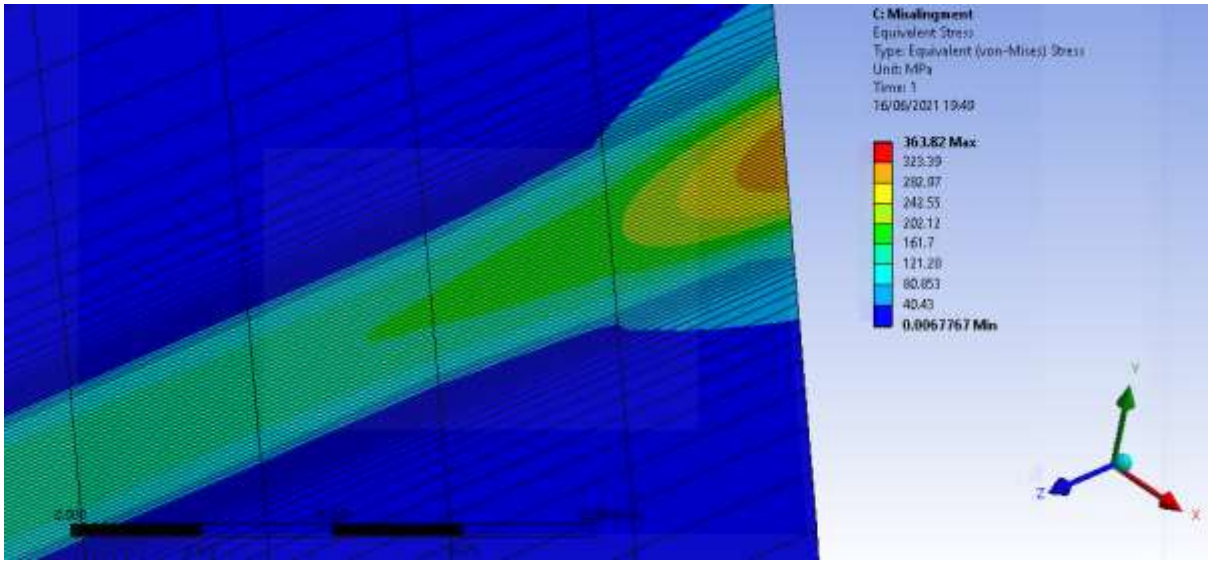
Picture 3.49. Reaction Force at Position 6'.

	Position 6	Position 6'
Reaction Force ANSYS (N)	974.06	764.02

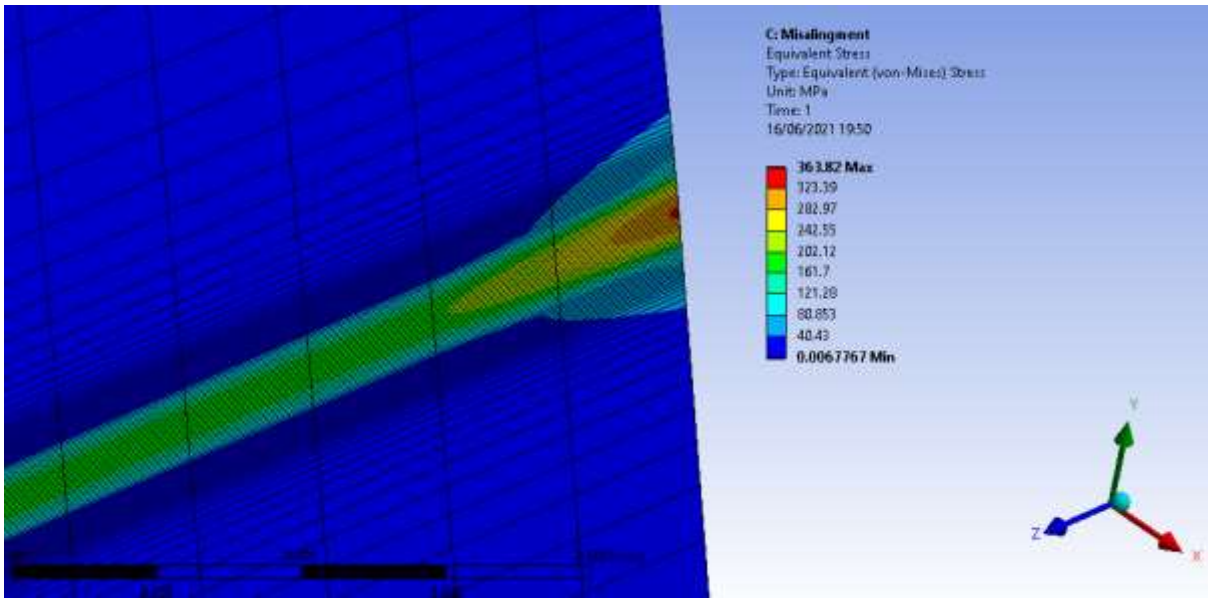
Table 20. Reaction Forces at each pair of contact.

Von Mises Stress

The results of Equivalent Von-Mises Stress are shown in Picture 3.50 and Picture 3.51. As we can see, the stresses follow the elliptical distribution that we were expecting. Moreover, the most loaded area is located well inside the finer mesh, so this is another indication that the mesh is probably ok.

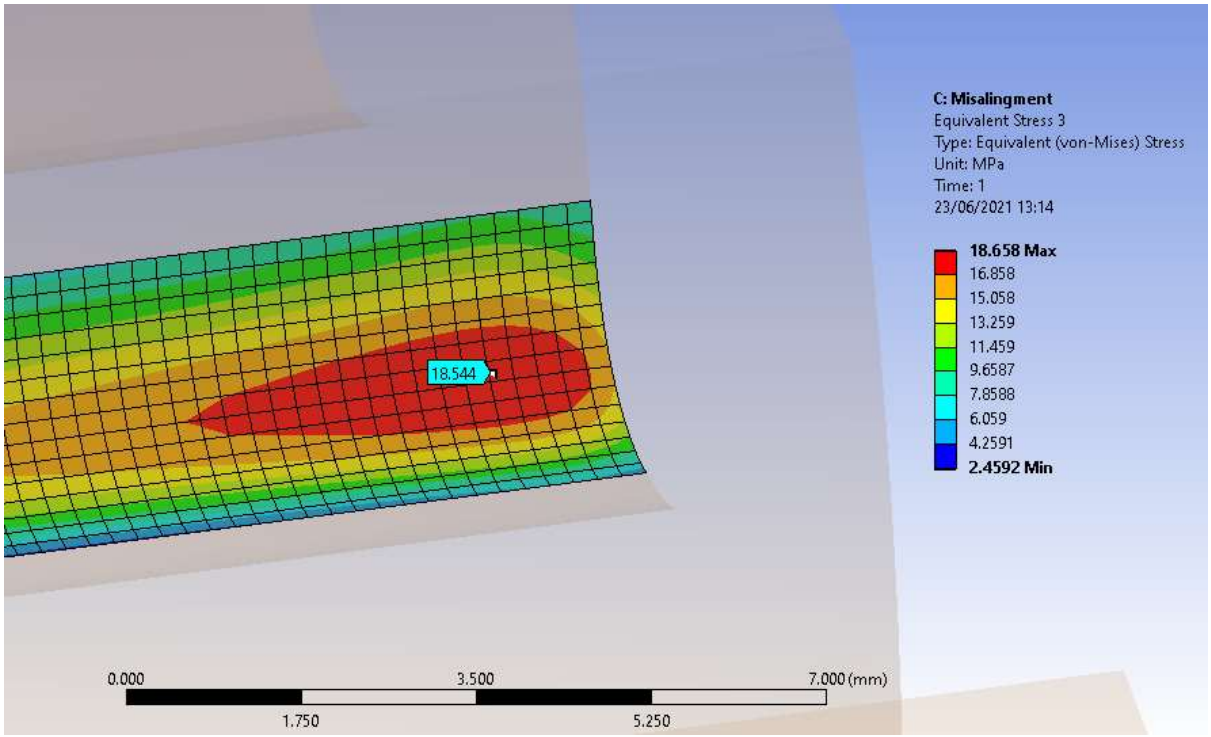


Picture 3.50. Equivalent Von-Mises stress in Position 6. Max value is 315.61 MPa.

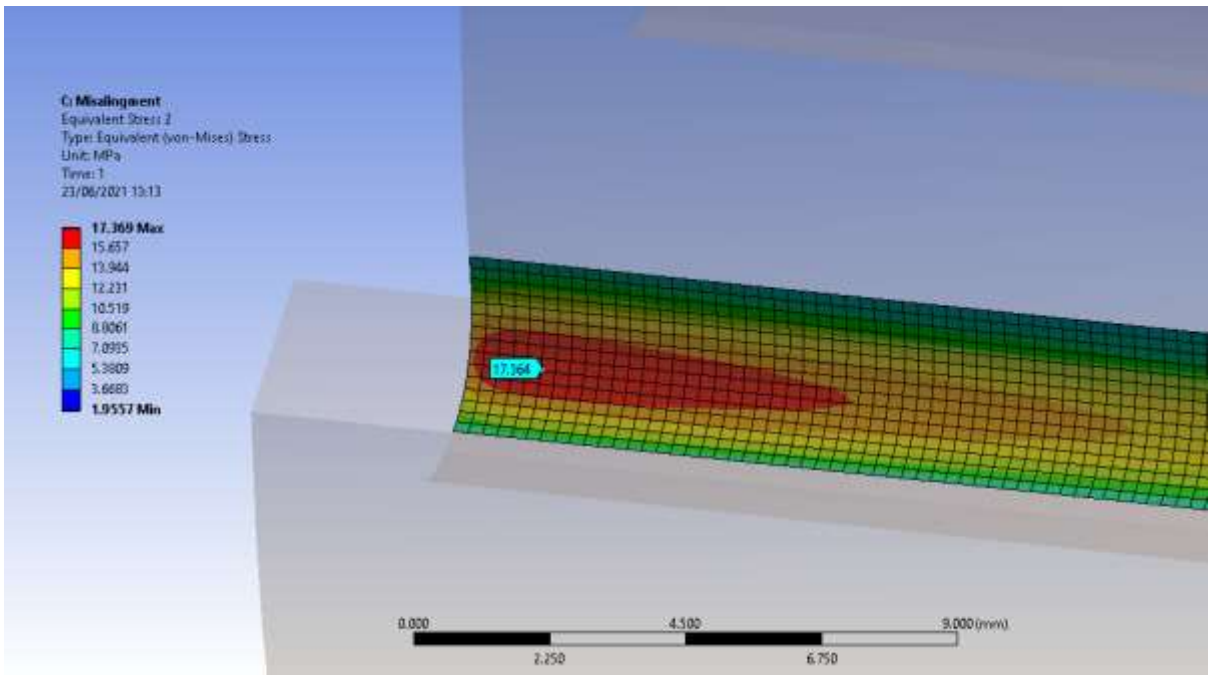


Picture 3.51. Equivalent Von-Mises stress in Position 6'. Max value is 363.82 MPa.

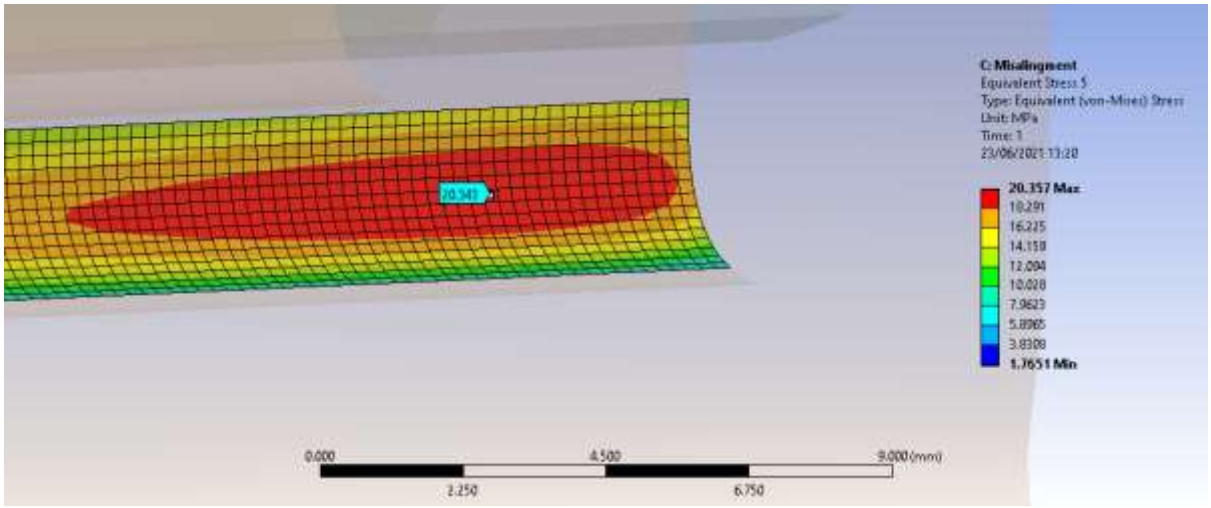
Finally in Picture we can see the values of the stress in the foot of the gear. Their values are presented in Table 3.21.



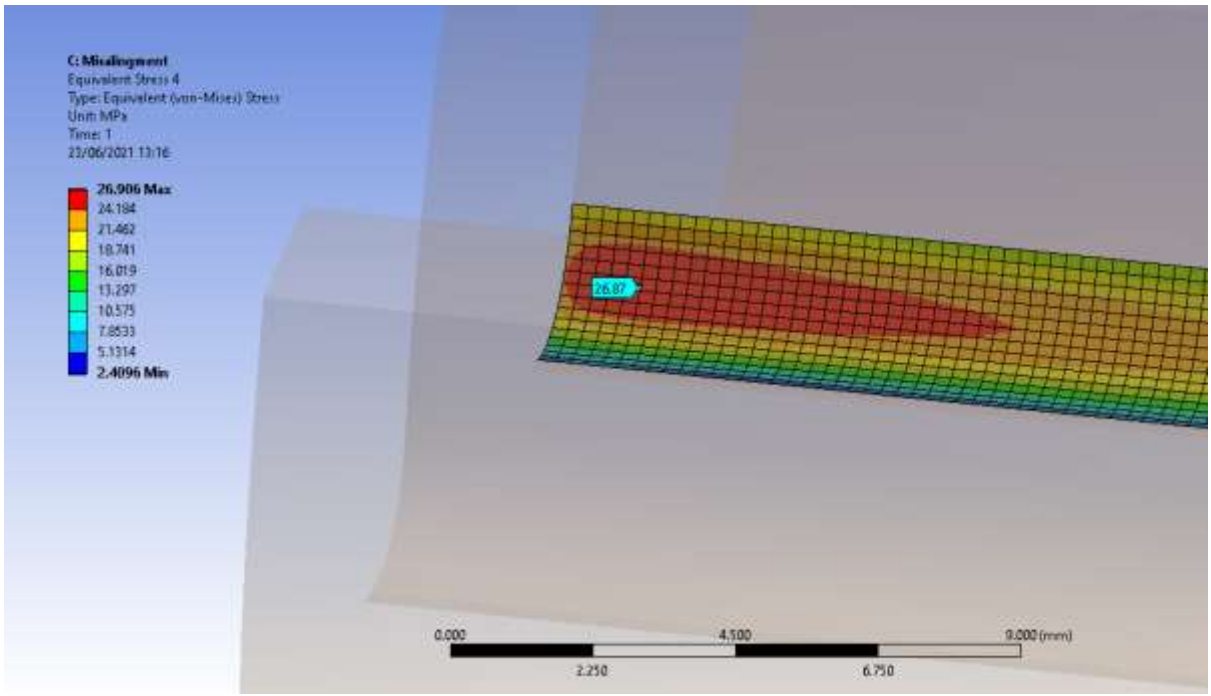
Picture 3.52. Bending stress in the right side of the tooth of conjugate gear at Position 6.



Picture 3.53. Bending stress in the left side of the tooth of conjugate gear at Position 6.



Picture 3.53. Bending stress in the right side of the tooth of conjugate gear ion at Position 6'.



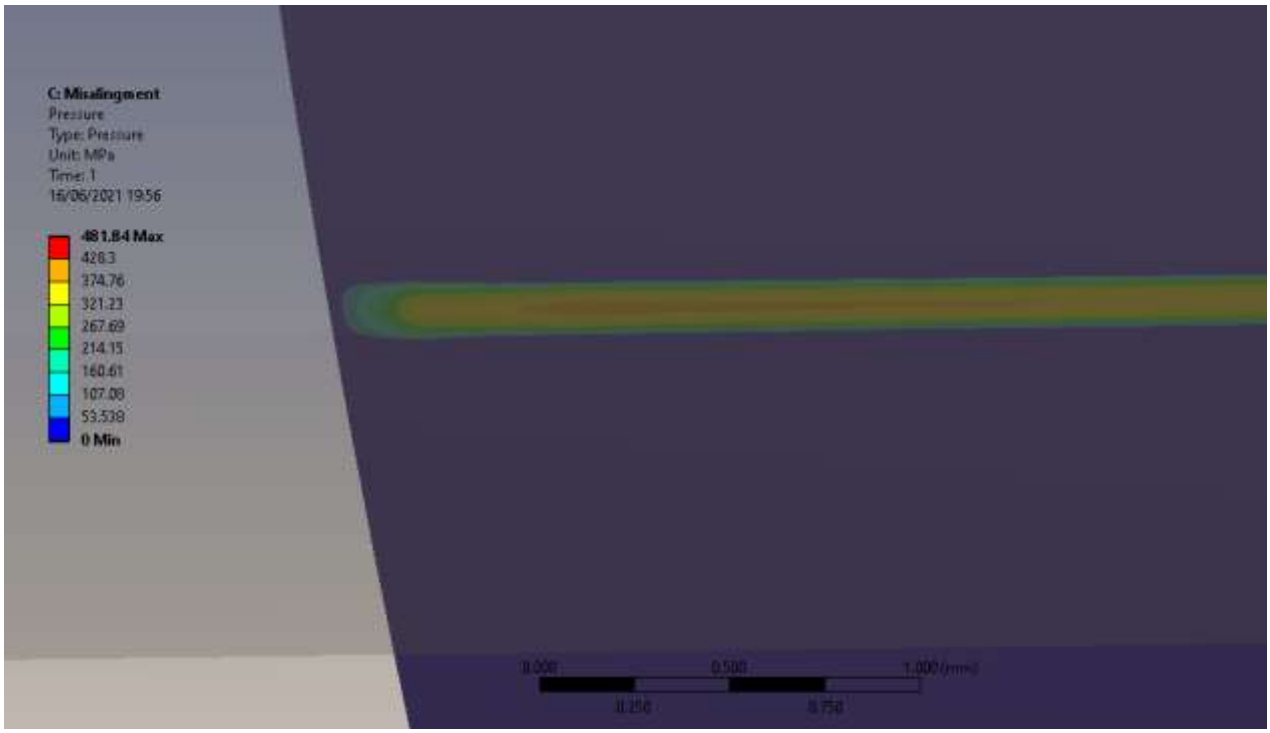
Picture 3.54. Bending stress in the left side of the tooth of conjugate gear at Position 6'.

	Position 6		Position 6'	
	Left	Right	Left	Right
Bending stress ANSYS (MPa)	17.36	18.54	26.87	20.34

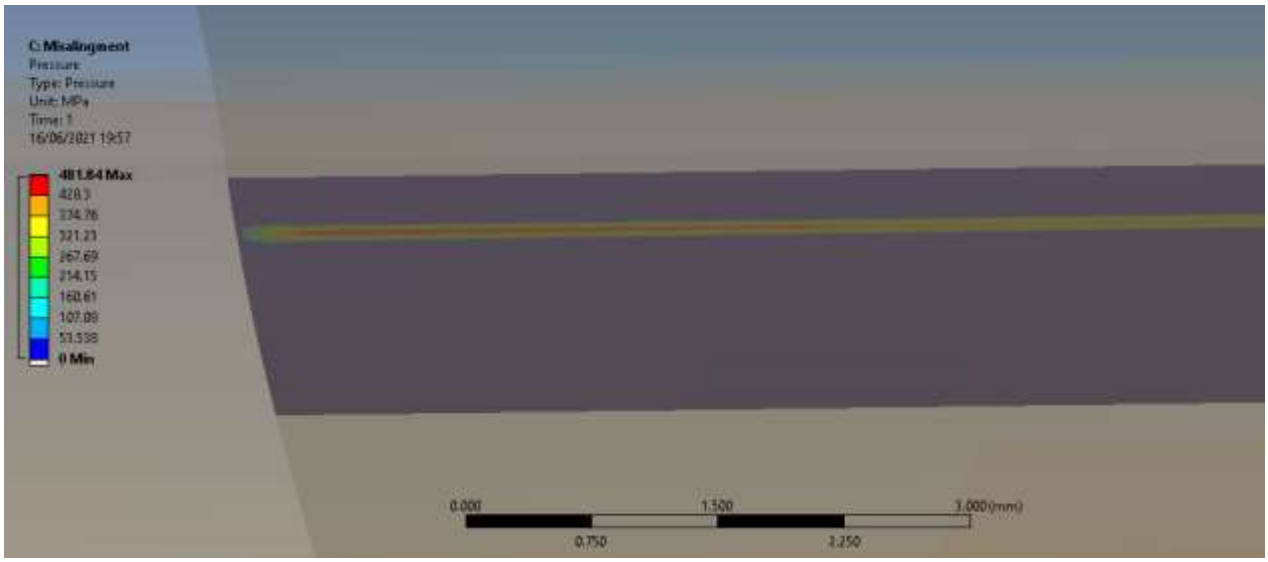
Table 3.21. Bending stress at each contact pair.

Contact Pressure

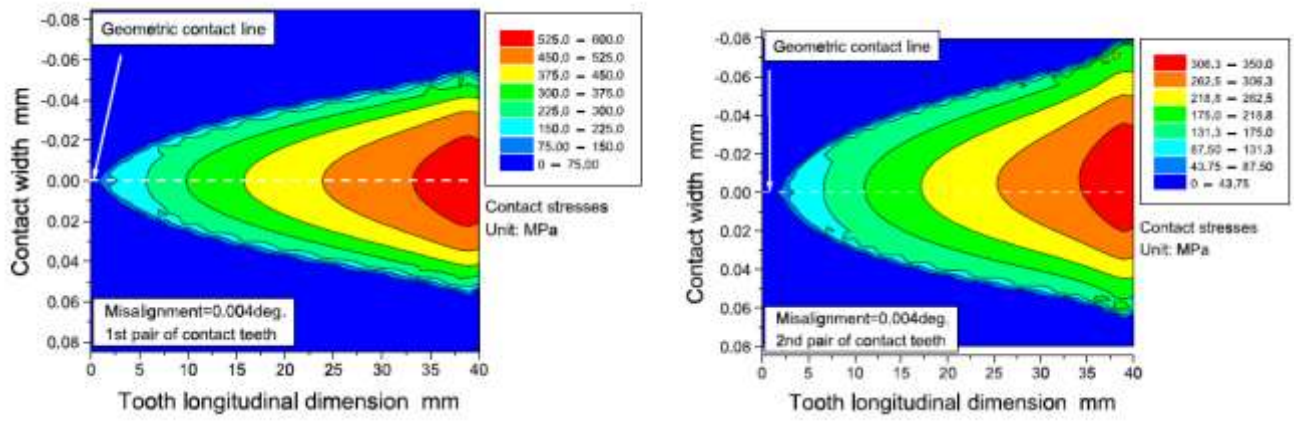
The results of the Contact Pressure are shown in Picture 3.55 and Picture 3.56.



Picture 3.55. Contact Pressure in position 6.



Picture 3.56. Contact Pressure in position 6'.



Picture 3.57. Contact Pressure from [] in position 6 and 6'.[2]

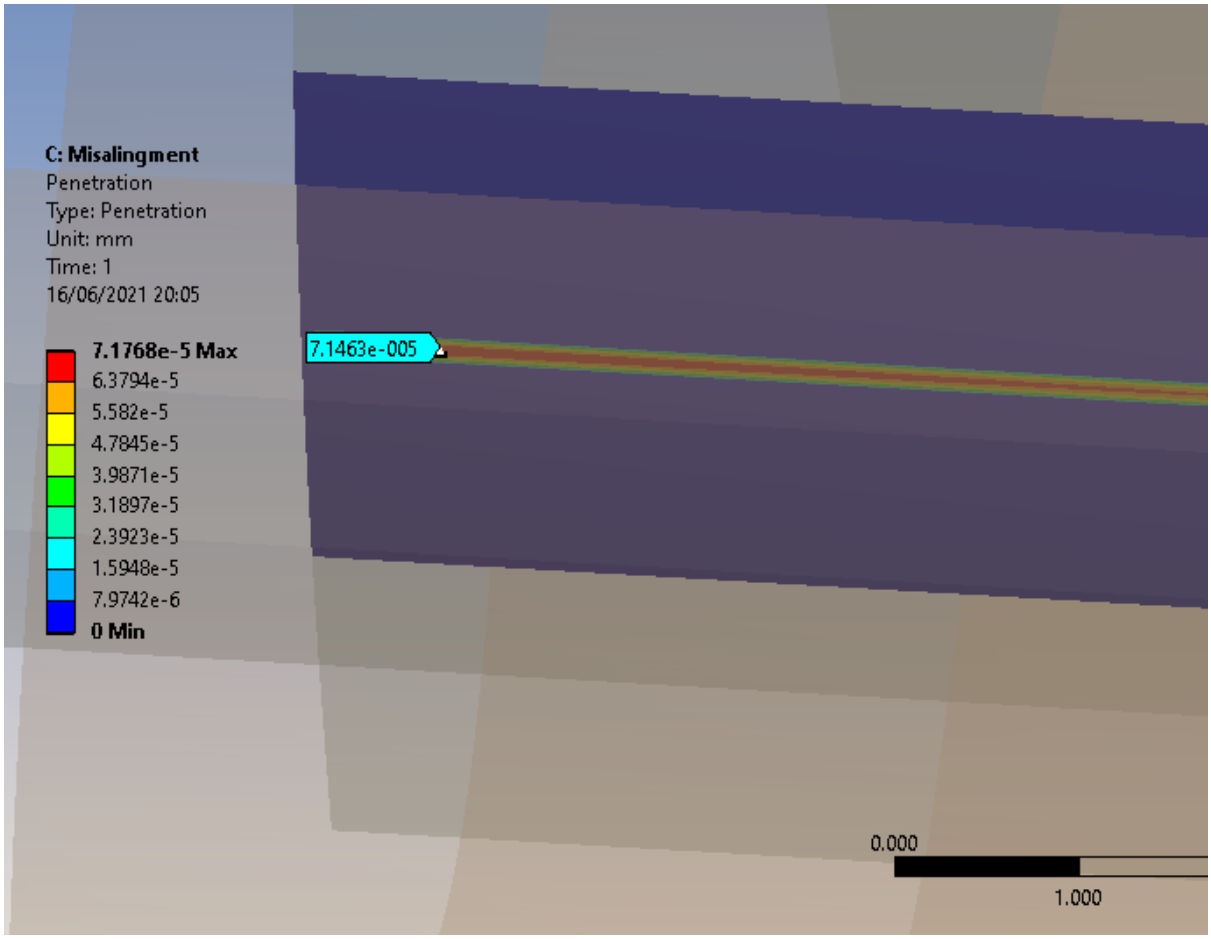
	Position 6	Position 6'
Contact Pressure ANSYS (MPa)	394.22	481.84
Contact Pressure in [2] (MPa)	350	600

Table 3.22. Contact Pressure at each contact pair.

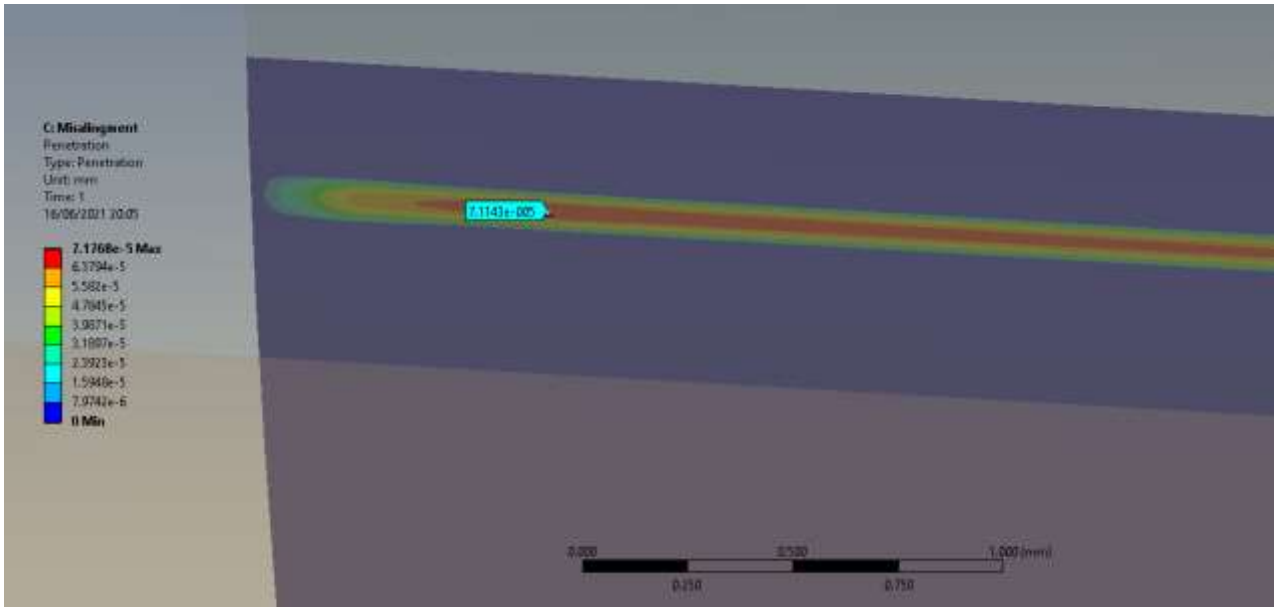
As we can see in Picture 3.73 and in Table 3.22, even if the results are not the same, the fields of contact pressure are very much alike.

Penetration

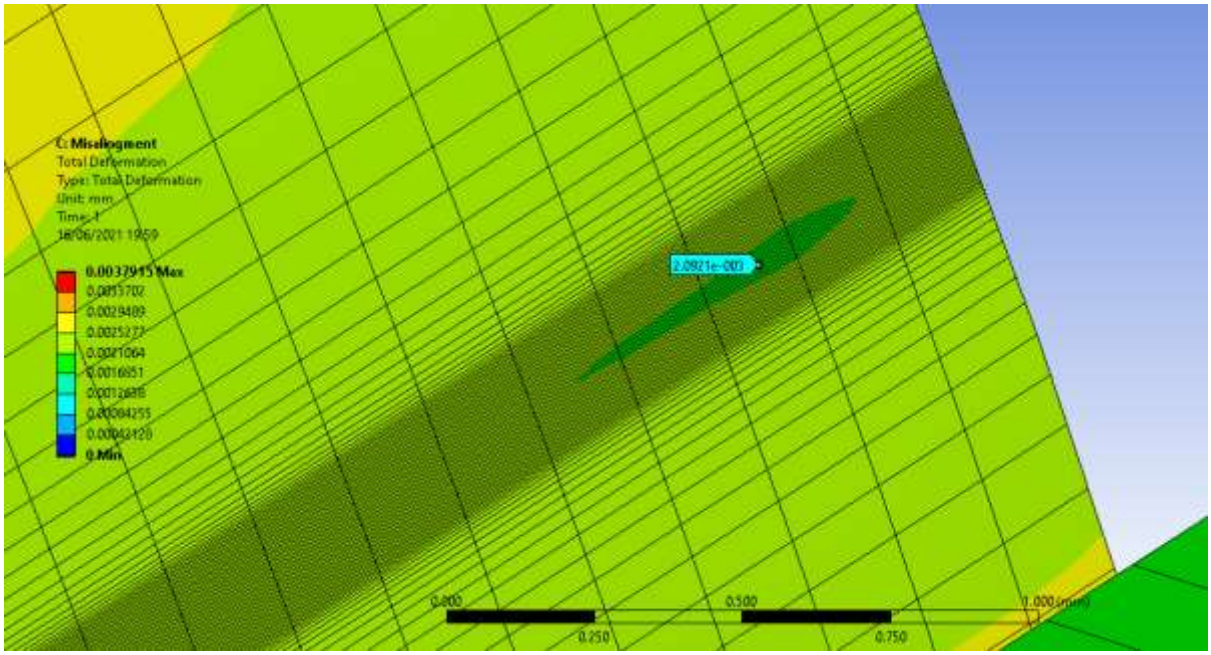
The results of the Penetration are shown in Picture 3.58 and Picture 3.59. To verify that the selection of the value for the Normal Stiffness Factor is ok we must compare the values of Penetration to the values of Displacement in the contact areas which are shown in Picture 3.60, Picture 3.61, Picture 3.62 and Picture 3.63.



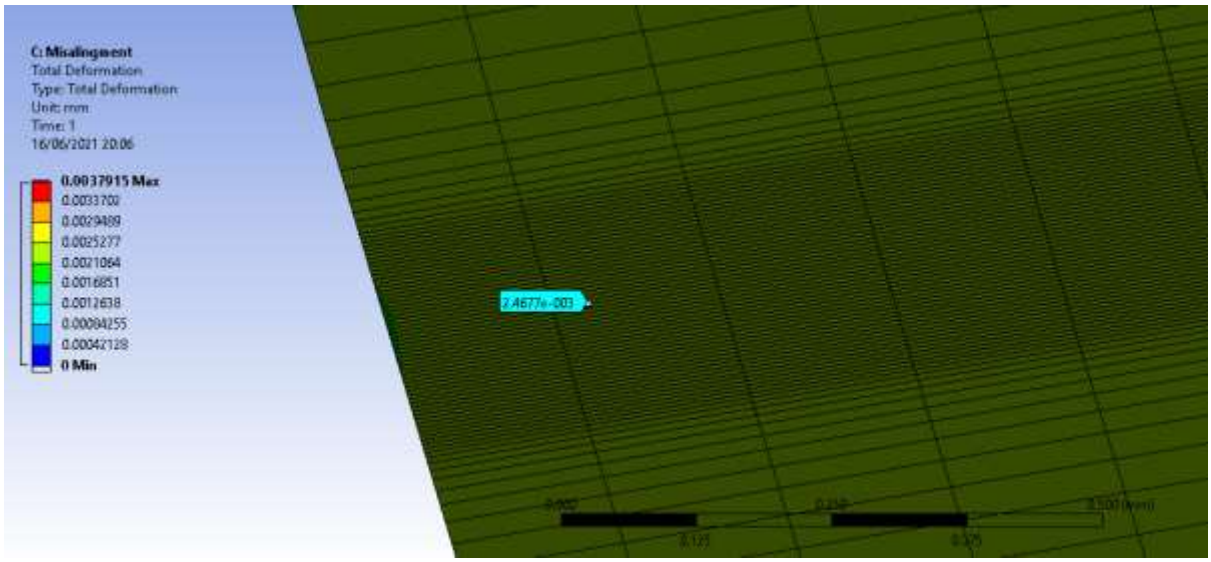
Picture 3.58. Penetration in position 6.



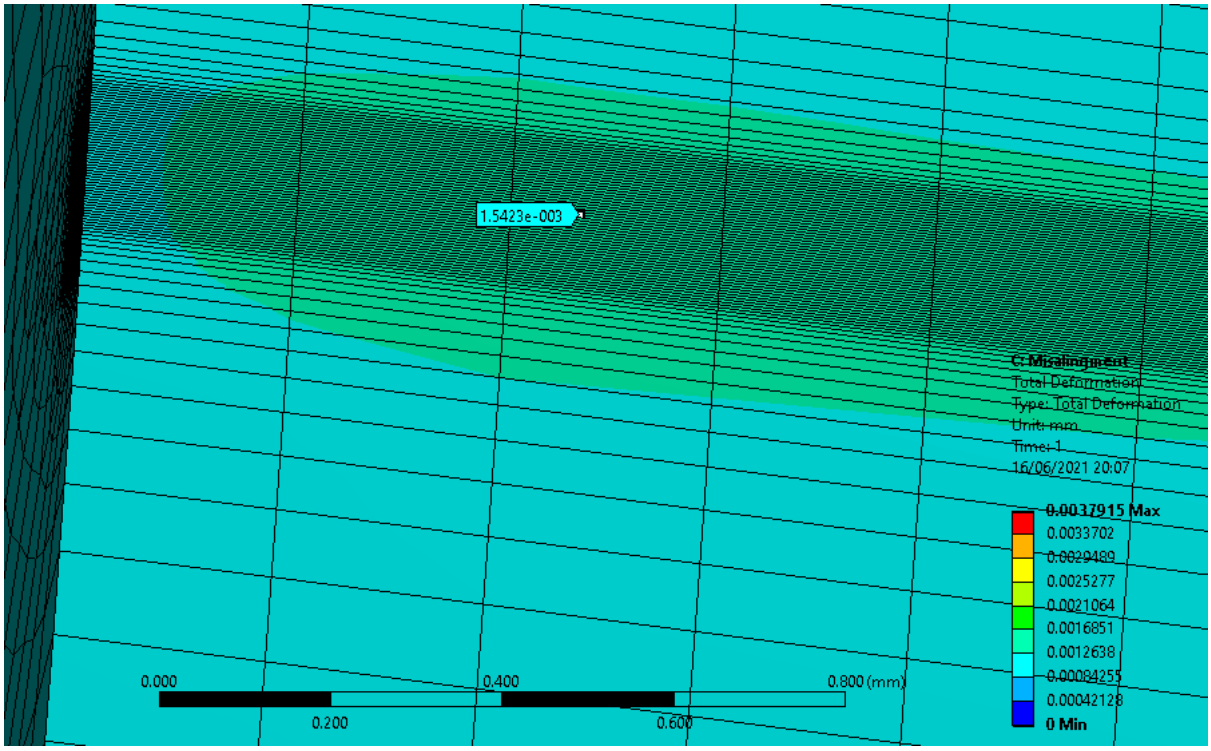
Picture 3.59. Penetration in position 6'.



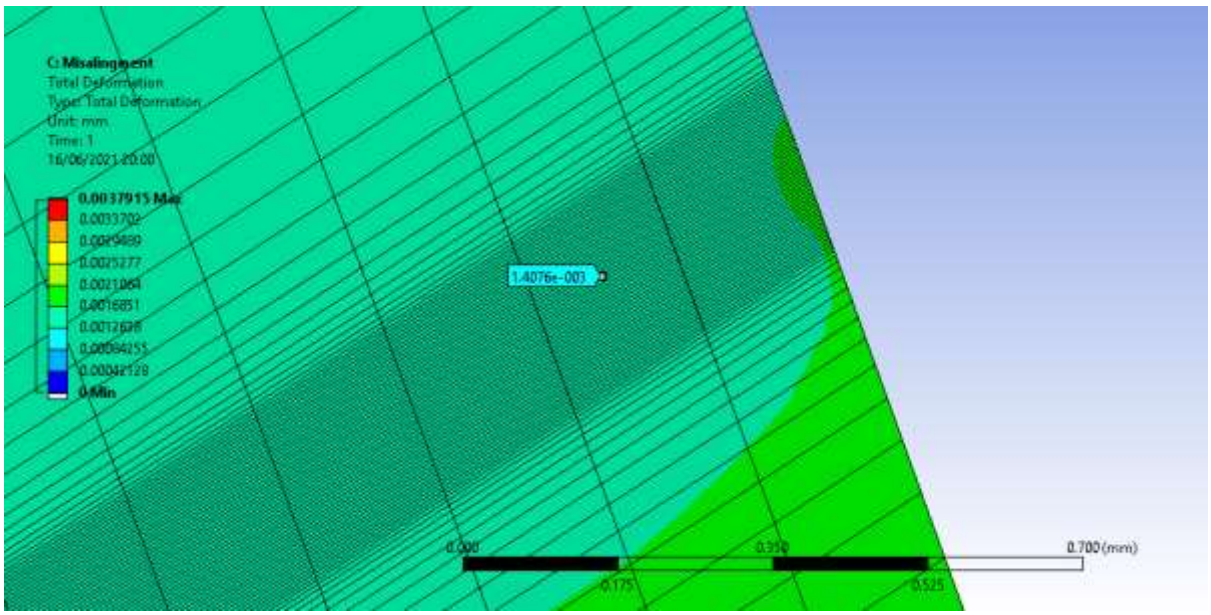
Picture 3.60. Total Displacement in position 6 of conjugate gear.



Picture 3.61. Total Displacement in position 6 of pinion.



Picture 3.62. Total Displacement in position 6' of pinion.



Picture 3.59. Total Displacement in position 6' of conjugate gear.

	Position 6		Position 6'	
	Conjugate gear	**Pinion**	**Conjugate gear**	**Pinion**
Total Displacement ANSYS (mm)	$2.09 \cdot 10^{-3}$	$2.46 \cdot 10^{-3}$	$1.41 \cdot 10^{-3}$	$1.54 \cdot 10^{-3}$
Penetration ANSYS (mm)	$7.15 \cdot 10^{-5}$		$7.11 \cdot 10^{-5}$	
Penetration Percent (%)	3.14		4.82	

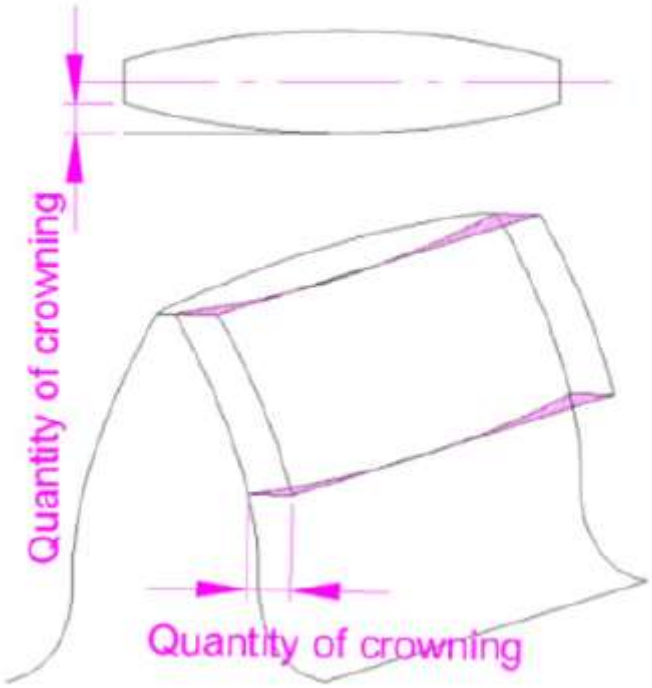
Table 3.23. Penetration at each contact pair.

As we can see in Table 3.23. Penetration is 3.14% of the Displacement in contact pair of position 6 and 4.82% in contact pair of position 6'. These are both acceptable values so Normal Stiffness Factor value is also acceptable.

3.4 Modelling of crowning in involute spur gears

In this chapter the modelling of crowning in involute spur gears is described. But what is crowning?

Some of the most common problems that gear manufacturers and users of gears in several application come across are gear noise and misalignment errors and. One of the simple solutions proposed to fix that is gear crowning. Gear crowning is a process that involves changing the chordal thickness of the tooth along its axis as shown in Picture 3.64. This solution is also used to counter offset loads commonly found in cantilevered gear shafts. In this case the crown on the lead would be offset to the centre of the gear face width. Crowning can be done by using several auxiliary geometries but here the study of crowning using a circular arc with an amount of crowing equal to 5 μm is examined. This value was selected to be able to compare the simulation results to the results of [2].



Picture 3.64. Crowning process.[2]

As far as the setting up of the simulation is concerned it is very similar to the one of the three-dimensional ideal involute spur gears apart from some modifications in the defining of contact and in the meshing procedure which will be discussed in the next paragraphs. Moreover, a symmetry condition was used with exact same way as described in the modelling of three-dimensional ideal involute spur gears.

3.4.1 Setting up of the contact parameters

In the modelling of crowning the value of the Normal Stiffness Factor is set to one.

3.4.2 Setting up of the mesh parameters

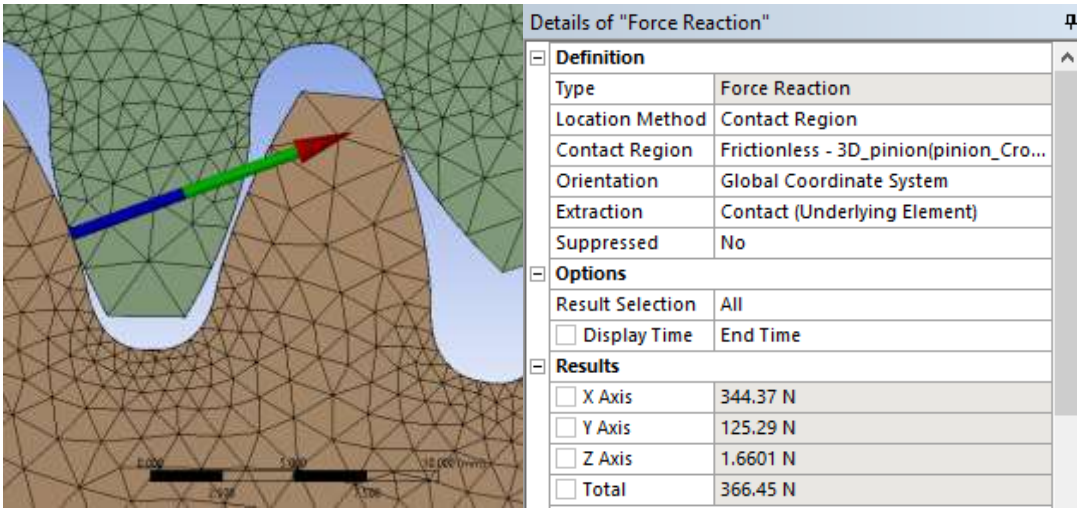
In the simulation of crowning there was no need to create a fine mesh across a whole strip like in the other simulations. This happens because the high stresses are mainly concentrated at a small area close to the initial point of contact. Therefore, instead of using the option Body Influence, the option Sphere of Influence was used with its centre located in each point of contact. In addition, because of the more complexed shape of the crowned gears the meshing was performed using LST elements instead of the quadratic quadrilaterals used in the rest of the simulations.

3.4.3 Results and verification

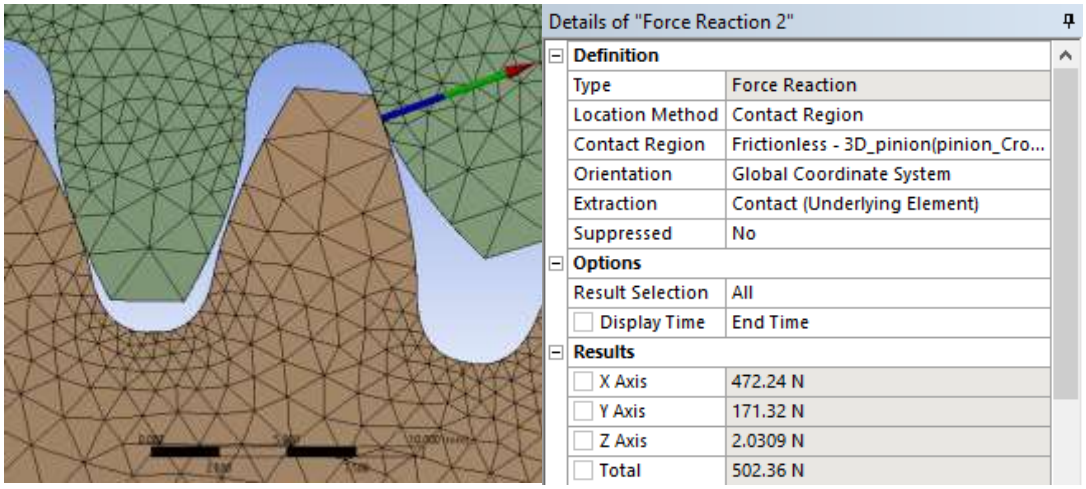
After the simulation is run the results in Reaction Force, Equivalent Von-Mises Stress, Contact Pressure and Penetration in each contact pair will be presented. Again, since the Hertz theory cannot easily be applied to the geometry of the crowned gears, the verification will be done by comparing the results of the simulation with results of [2]. However, the results of [] only provide information about the contact pressure so the rest of the results will be taken as realistic if the error between it is confirmed that the contact pressure of the model matches the one of [2].

Reaction Force

The results of the Reaction Force are shown in Picture 3.65 and Picture 3.66. Of course, since there is a symmetry condition the Reaction Force in the results is the half of the actual force.



Picture 3.65. Reaction Force at Position 6.



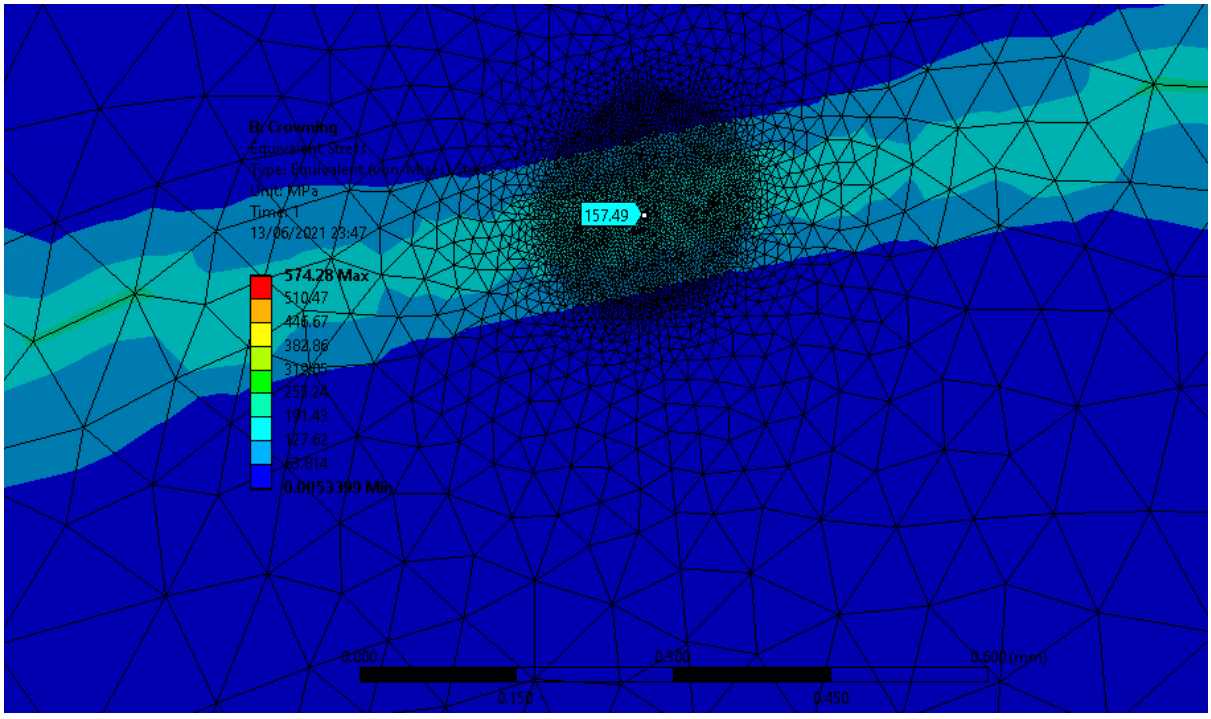
Picture 3.66. Reaction Force at Position 6'.

	Position 6	Position 6'
Reaction Force ANSYS (N)	366.45	502.36

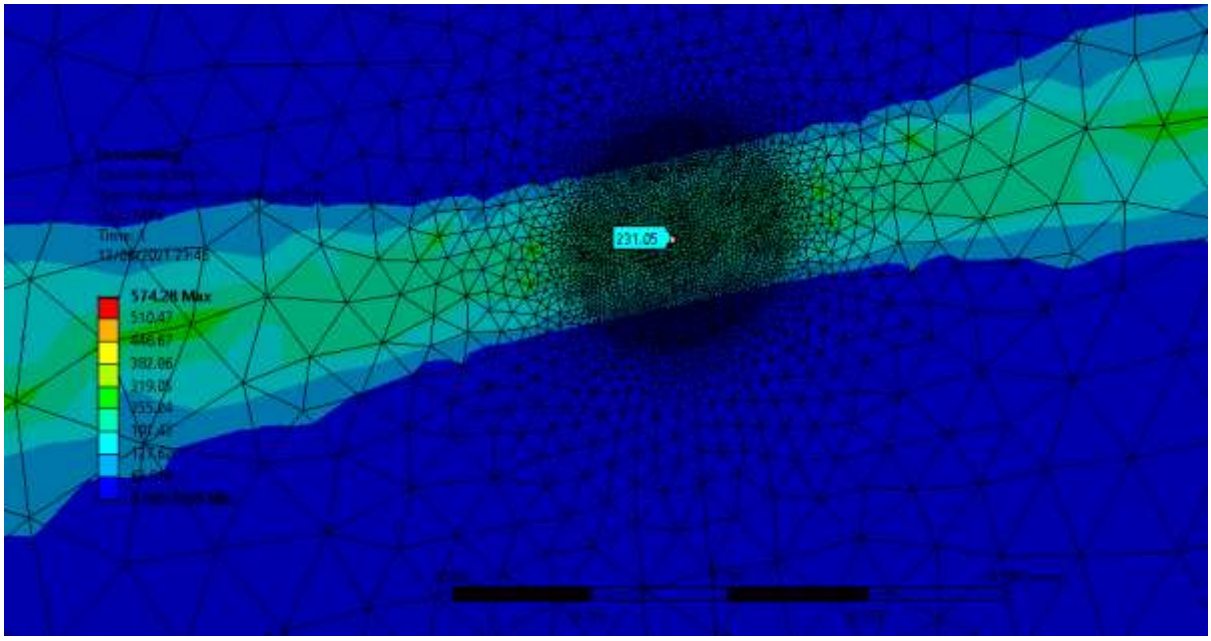
Table 3.24. Reaction Forces at each pair of contact.

Von Mises Stress

The results of Equivalent Von-Mises Stress are shown in Picture 3.67 and Picture 3.68. Moreover, the most loaded area is located well inside the finer mesh, so this is another indication that the mesh is probably ok.

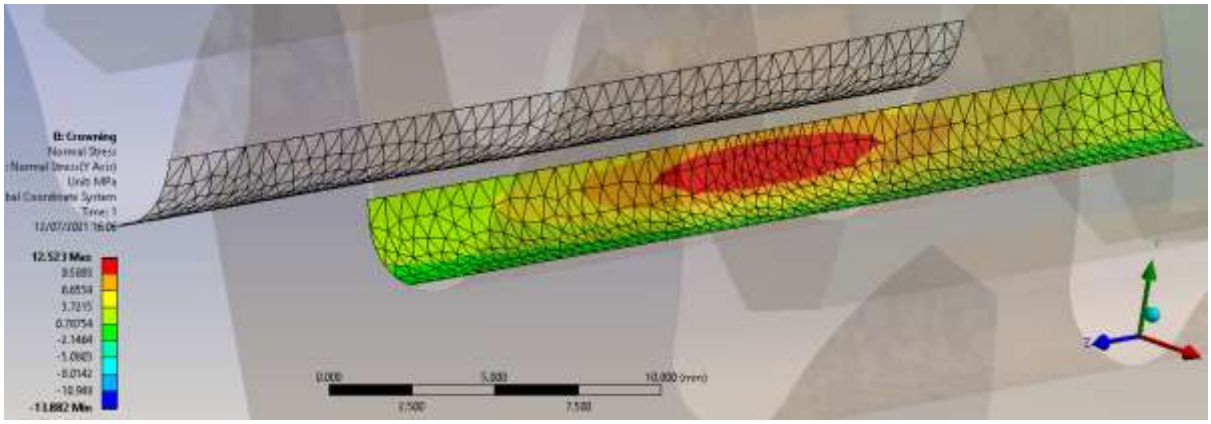


Picture 3.67. Equivalent Von-Mises stress in Position 6. Max value is157.49 MPa.

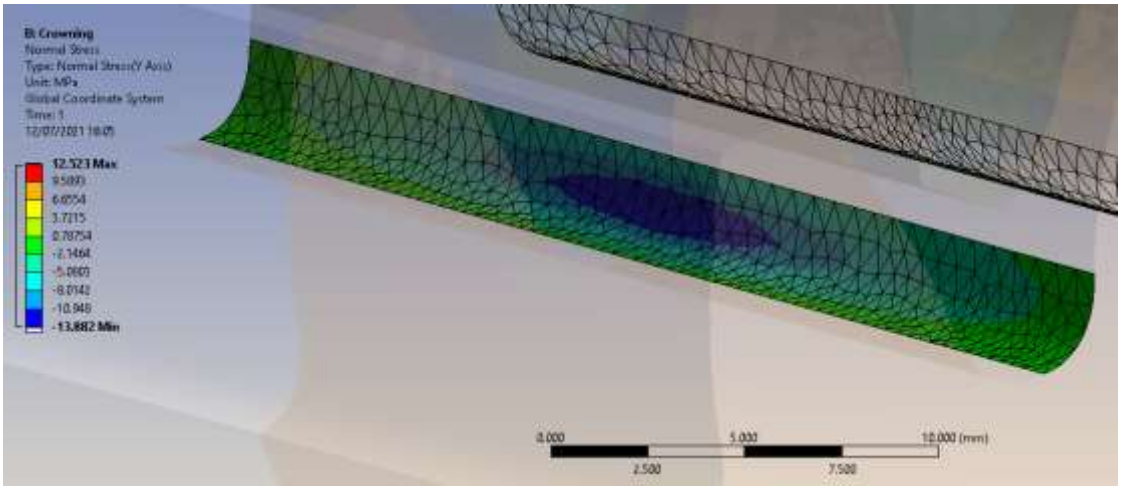


Picture 3.68. Equivalent Von-Mises stress in Position 6'. Max value is 231.05 MPa.

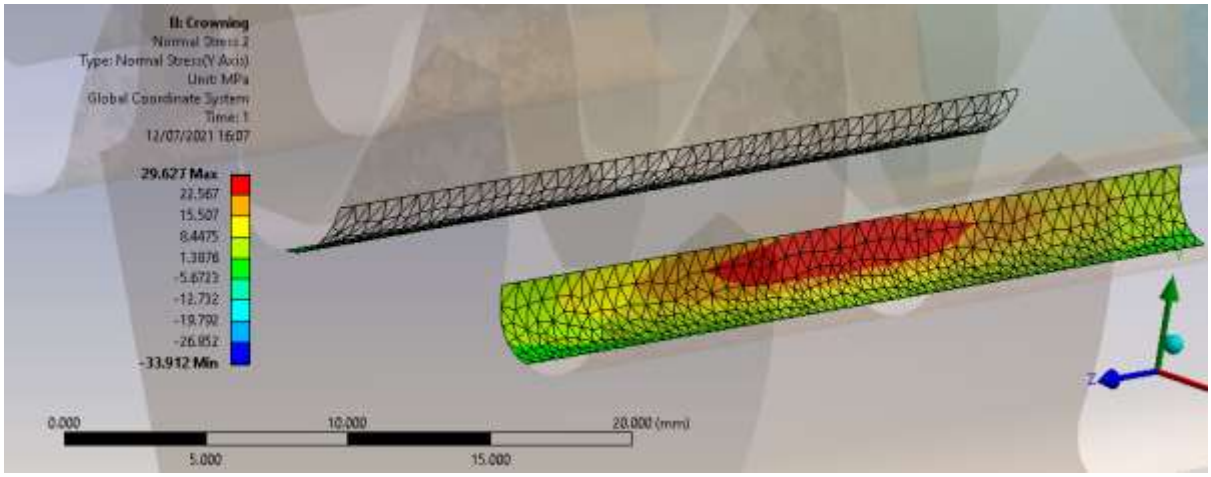
Finally, in Picture 3.69, Picture 3.70, Picture 3.71 and Picture 3.72, we can see the values of the stress in the foot of the gear. Their values are presented in Table 3.25.



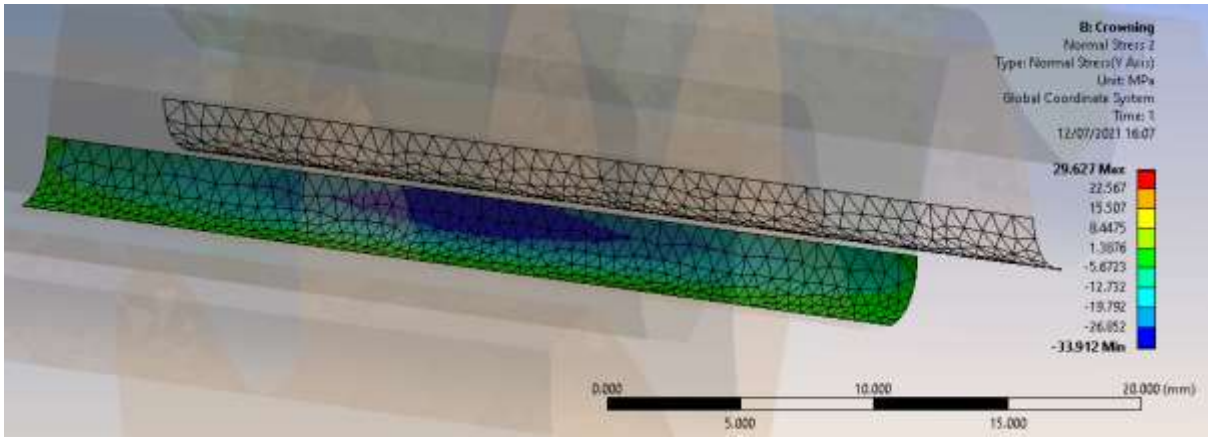
Picture 3.69. Bending stress in the right side of the tooth of conjugate gear at Position 6.



Picture 3.70. Bending stress in the left side of the tooth of conjugate gear at Position 6.



Picture 3.71. Bending stress in the right side of the tooth of conjugate gear at Position 6'.



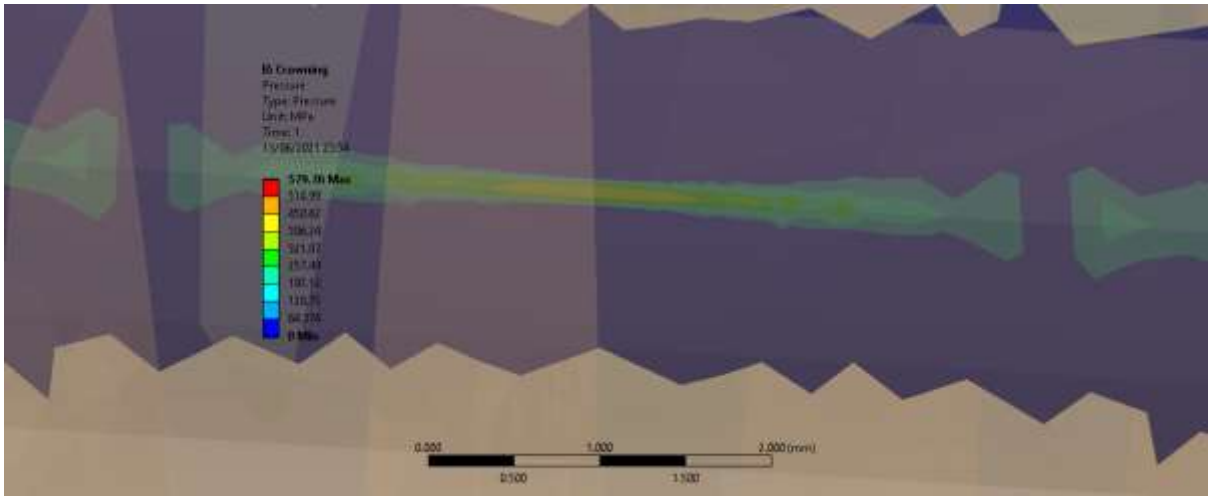
Picture 3.72. Bending stress in the left side of the tooth of conjugate gear at Position 6'.

	Position 6		Position 6'	
	Left	**Right**	**Left**	**Right**
Bending stress ANSYS (MPa)	13.88	12.52	33.91	29.63

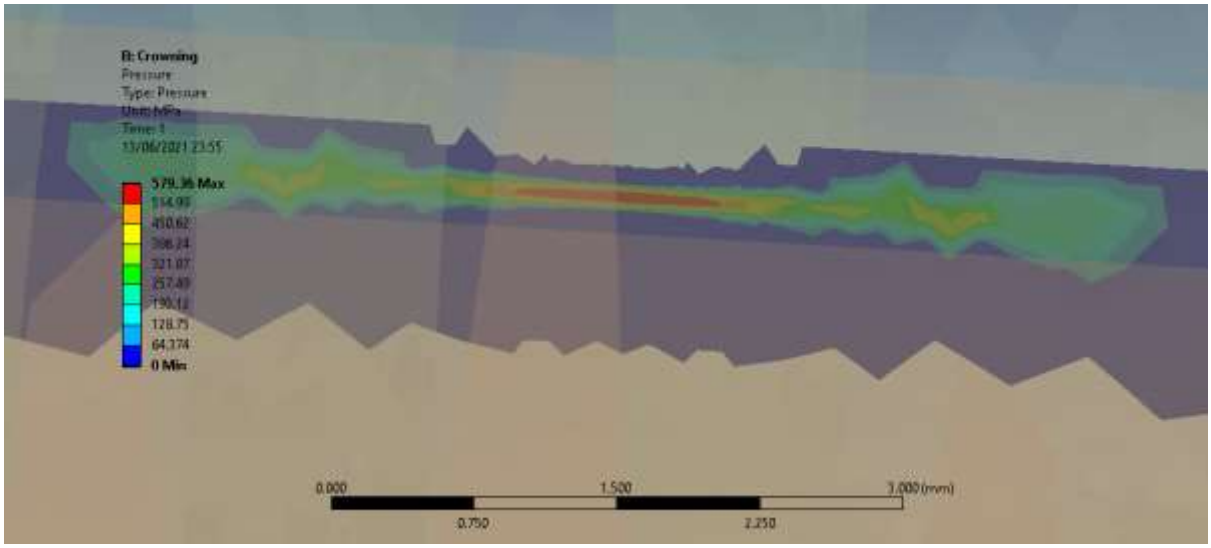
Table 3.25. Bending stress at each contact pair.

Contact Pressure

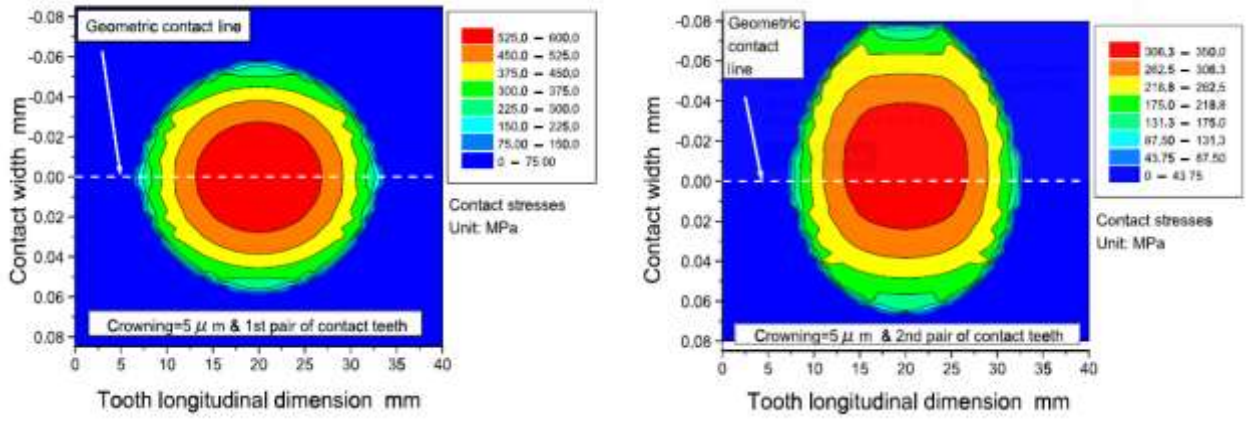
The results of the Contact Pressure are shown in Picture 3.73 and Picture 3.74 and in Picture 3.75 we have the results from [2]. We can see in Table 3.26 that the maximum values match well with the ones presented in [2] and the pressure fields are alike Thus, we also expect the rest of the simulation results to be correct.



Picture 3.73. Contact Pressure in position 6.



Picture 3.74. Contact Pressure in position 6’.



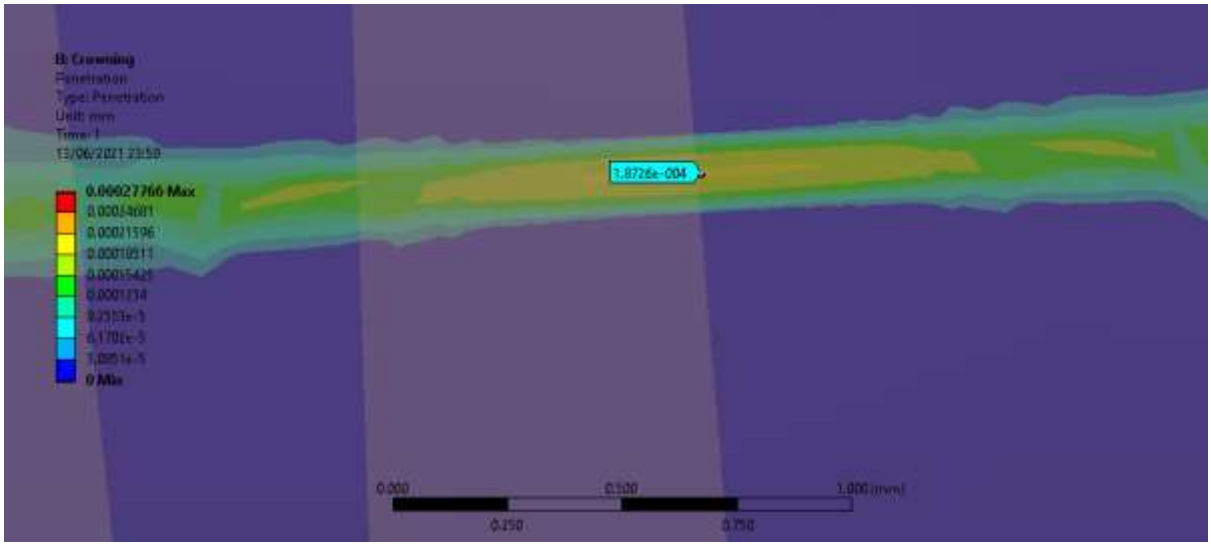
Picture 3.75. Contact Pressure from [] in position 6 and 6’.[2]

	Position 6	Position 6’
Contact Pressure ANSYS (MPa)	377.51	579.36
Contact Pressure in [2] (MPa)	350	600
Error (%)	7.86	3.4

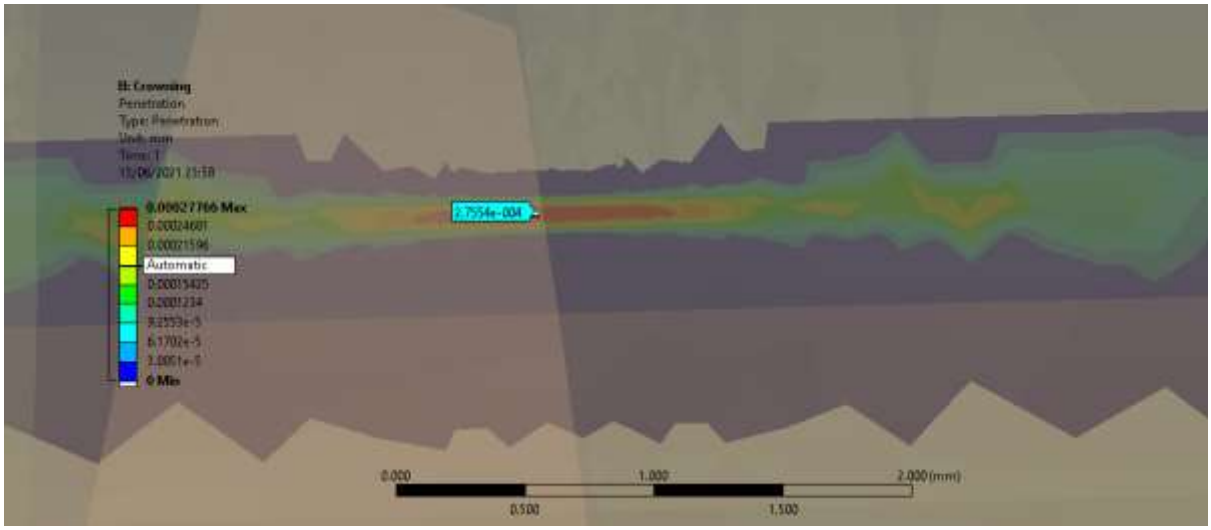
Table 3.26. Contact Pressure at each contact pair.

Penetration

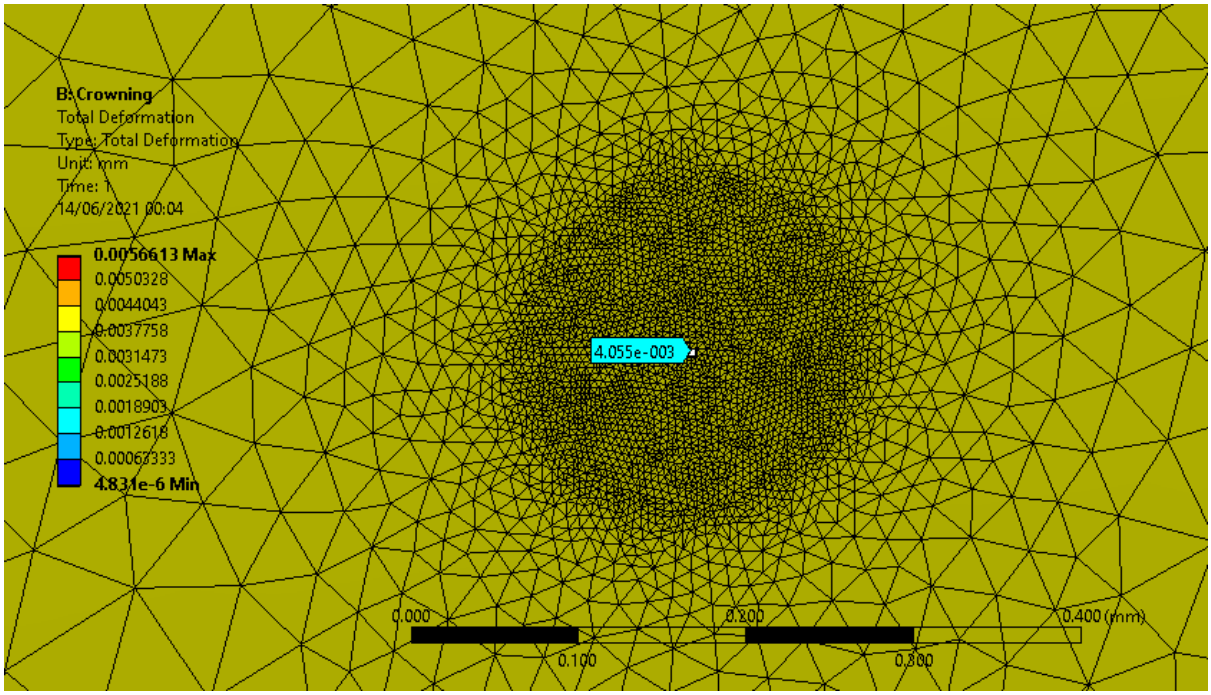
The results of the Penetration are shown in Picture 3.76 and in Picture 3.77. To verify that the selection of the value for the Normal Stiffness Factor is ok we must compare the values of Penetration to the values of Displacement in the contact areas which are shown in Picture 3.78, Picture 3.79, Picture 3.80 and Picture 3.81.



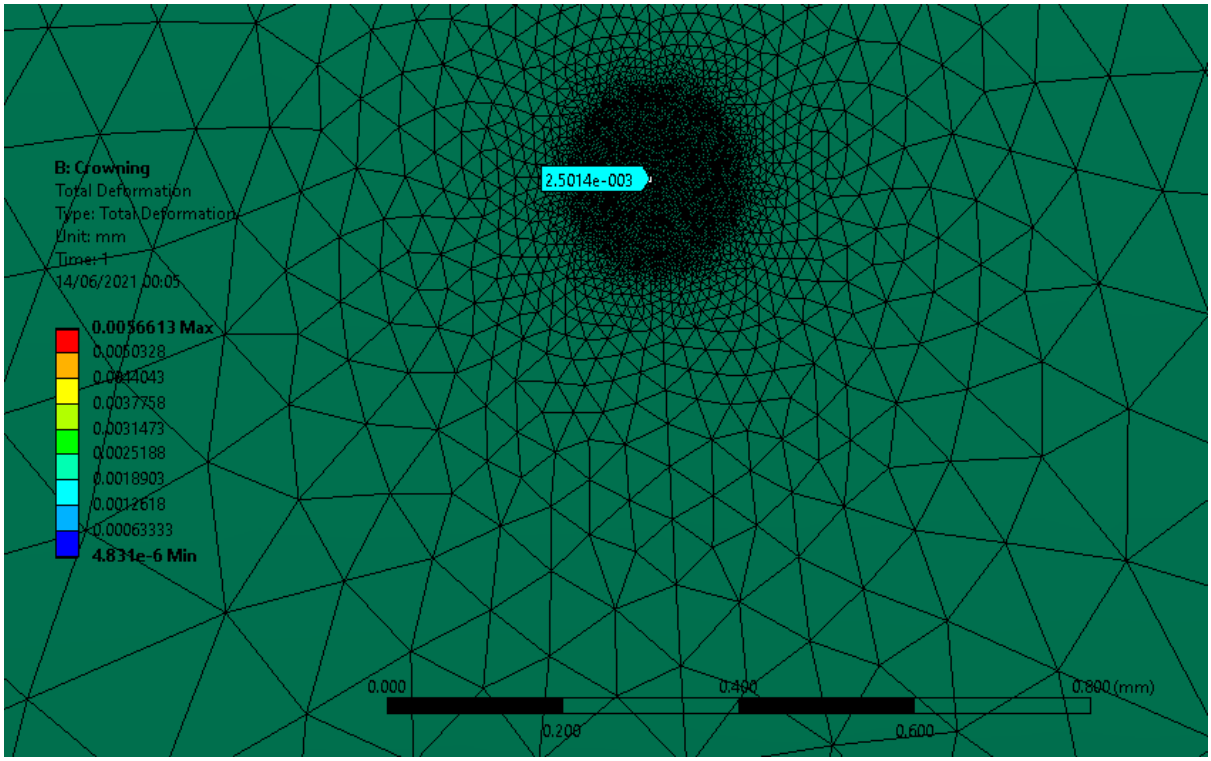
Picture3.76. Penetration in position 6.



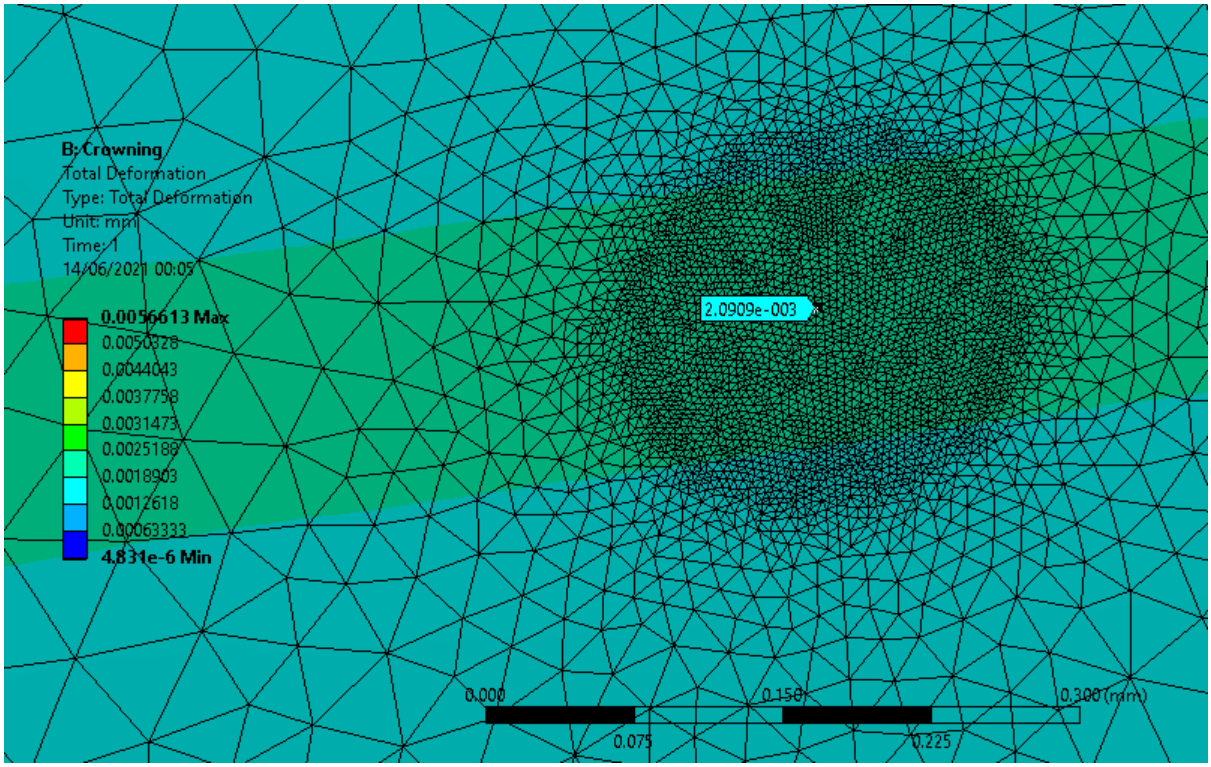
Picture 3.77. Penetration in position 6’.



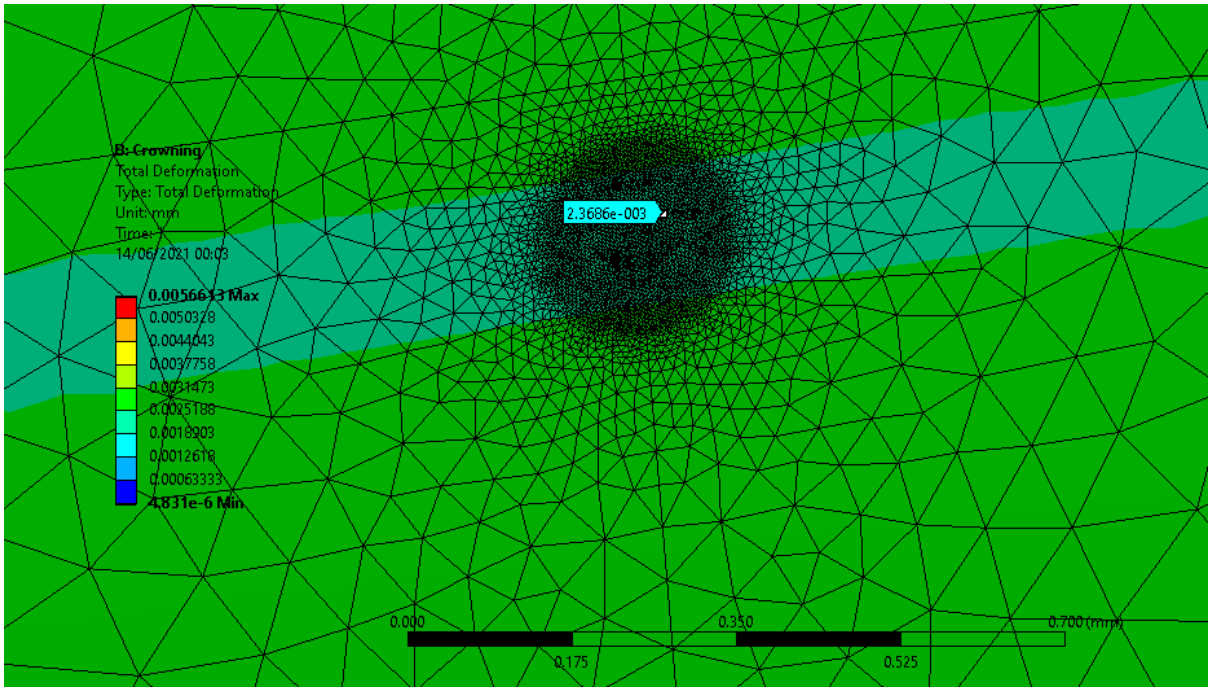
Picture 3.78. Total Displacement in position 6 of conjugate gear.



Picture 3.79. Total Displacement in position 6 of pinion.



Picture 3.80. Total Displacement in position 6' of pinion.



Picture 3.81. Total Displacement in position 6' of conjugate gear.

	Position 6		Position 6'	
	Conjugate gear	Pinion	Conjugate gear	Pinion
Total Displacement ANSYS (mm)	$4.01 \cdot 10^{-3}$	$2.50 \cdot 10^{-3}$	$2.37 \cdot 10^{-3}$	$2.09 \cdot 10^{-3}$
Penetration ANSYS (mm)	$1.87 \cdot 10^{-4}$		$2.75 \cdot 10^{-4}$	
Penetration Percent (%)	5.75		12.33	

Table 3.27. Penetration at each contact pair.

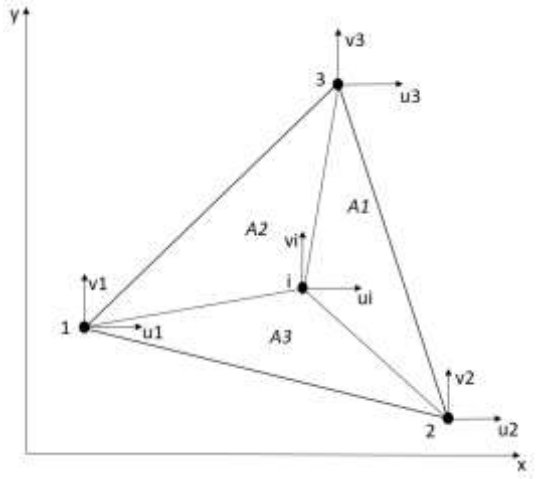
As we can see in Table 3.27. Penetration is 5.74% of the Displacement in contact pair of position 6 and 12.33% in contact pair of position 6'. These value in position 6 is acceptable but the value in position 6' which is over 10% may be problematic but in this case is our results are ok we can accept it.

4. MATLAB code for modelling of gear contact

In this paragraph the structure of the MATLAB for the modelling of involute spur gear contact will be described along with the FEM used in it. The case used for the presentation of the code will be the same for gears engaged in the same position as in the previous chapter. As it is already mentioned the analysis will be two-dimensional and the elements used will be CST and LST. The code with the CST elements was first written by Prof Christophoros Provatidis in collaboration with the student at the time Emanouil Sakaridis and I made the necessary changes to function properly and yield the desired results.

4.1 Description of CST element

Since the meshing process is well established, the equations used for the modelling of the problem with the CST element must also be explained. For this reason, one of the triangular elements is taken and its nodes are numbered as shown in Picture 4.1.



Picture 4.1. CST element

The displacements u and v along the axes x and y respectively of a random point included in the element are given by the following equations:

$$u = N_1 u_1 + N_2 u_2 + N_3 u_3 \text{ (4.1)}$$

$$v = N_1 v_1 + N_2 v_2 + N_3 v_3 \text{ (4.2)}$$

Where (x_1, y_1), $(x_2, y_2)$and (x_3, y_3) and (u_1, v_1), $(u_2, v_2)$and (u_3, v_3) are the coordinates and displacements of nodes 1, 2 and 3 respectively and N_1, N_2 and N_3 are called shape functions. The shape functions are the functions used to interpolate the coordinates and displacement of any given point inside the element. Since we are talking about the CST element, these shape functions are chosen to be the triangular coordinates. This means that:

$$N_1 = \frac{A_1}{A} \text{ (4.3)}$$

$$N_2 = \frac{A_2}{A} \text{ (4.4)}$$

$$N_3 = \frac{A_3}{A} \text{ (4.5)}$$

$$A = A_1 + A_2 + A_3 \text{ (4.6)}$$

$$1 = N_1 + N_2 + N_3 \text{ (4.7)}$$

The areas A_1, A_2 and A_3 are shown in Picture and for instance if we area node 1 then $A_1 = A$ and $A_2, A_3 = 0$. Thus, they area called triangular coordinates. From equations () through () we end up with:

$$N_1 = \frac{1}{2A}((x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y) \text{ (4.8)}$$

$$N_2 = \frac{1}{2A}((x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y) \text{ (4.9)}$$

$$N_3 = \frac{1}{2A}((x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y) \text{ (4.10)}$$

As far as the computation of strains of any given point inside the triangle is concerned, from equations of two-dimensional linear elasticity we know that:

$$s = E_{tot}\varepsilon \text{ (4.11)}$$

$$s = \begin{bmatrix} s_x \\ s_y \\ t_{xy} \end{bmatrix} \text{ (4.12)}$$

$$\varepsilon = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \text{ (4.13)}$$

where s_x and $s_y$are the normal stress along the axes x and y respectively and t_{xy} the shear stress and ε_x, ε_y and γ_{xy} are the equivalent strains and E_{tot} is equal to:

$$E_{tot} = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{12} & d_{11} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \text{ (4.14)}$$

If the state of the linear elasticity model is Plane Stress, then the parameters d_{11}, d_{12} and d_{22} are defined as:

$$d_{11} = \frac{E}{(1-\nu^2)} \text{ (4.15)}$$

$$d_{12} = \frac{\nu E}{(1-\nu^2)} \text{ (4.16)}$$

$$d_{33} = \frac{E}{2(1+\nu)} \text{ (4.17)}$$

And if the state of linear elasticity model is Plane Strain, then the parameters d_{11}, d_{12} and d_{22} are defined as:

$$d_{11} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \text{ (4.18)}$$

$$d_{12} = \frac{\nu E}{(1+\nu)(1-2\nu)} \text{ (4.19)}$$

$$d_{33} = \frac{E}{2(1+\nu)} \text{ (4.20)}$$

To receive the stresses, first the strains must be calculated, and this is done with the following equations:

$$\varepsilon = \begin{bmatrix} \frac{dN_1}{dx} & 0 & \frac{dN_2}{dx} & 0 & \frac{dN_3}{dx} & 0 \\ 0 & \frac{dN_1}{dy} & 0 & \frac{dN_2}{dy} & 0 & \frac{dN_3}{dy} \\ \frac{dN_1}{dx} & \frac{dN_1}{dy} & \frac{dN_2}{dx} & \frac{dN_2}{dy} & \frac{dN_3}{dx} & \frac{dN_3}{dy} \end{bmatrix} U \text{ (4.21)}$$

$$U = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} \text{ (4.22)}$$

From () we observe that for the strains to be calculated it is necessary to get the nodal displacements and this is done with the general equations of FEM in the static analysis which is:

$$KU = F \text{ (4.23)}$$

$$F = \begin{bmatrix} F_{x1} \\ F_{y1} \\ F_{x2} \\ F_{x2} \\ F_{x3} \\ F_{y3} \end{bmatrix} \text{ (4.24)}$$

where K is the stiffness matrix and F is the array of nodal forces.

To calculate the stiffness matrix is calculated from the theory of Possible Works with the following formula:

$$K = \int_A B^T E_{tot} B \, dA \text{ (4.25)}$$

where B is equal to:

$$B = \begin{bmatrix} \frac{dN_1}{dx} & 0 & \frac{dN_2}{dx} & 0 & \frac{dN_3}{dx} & 0 \\ 0 & \frac{dN_1}{dy} & 0 & \frac{dN_2}{dy} & 0 & \frac{dN_3}{dy} \\ \frac{dN_1}{dx} & \frac{dN_1}{dy} & \frac{dN_2}{dx} & \frac{dN_2}{dy} & \frac{dN_3}{dx} & \frac{dN_3}{dy} \end{bmatrix} (4.26)$$

For the simple case of CST element, the stiffness matrix is equal to:

$$K = \frac{Eh}{4A(1-\nu^2)} K_1 + \frac{Eh}{8A(1+\nu)} K_2 \ (4.27)$$

where for the case of Plane Stress K_1 and K_2 are equal to:

$$K_1 = \begin{bmatrix} {b_1}^2 & b_1c_1\nu & b_1b_2 & b_1c_2\nu & b_1b_3 & b_1c_3\nu \\ b_1c_1\nu & {c_1}^2 & b_2c_1\nu & c_1c_2 & b_3c_1\nu & c_1c_3 \\ b_1b_2 & b_2c_1\nu & {b_2}^2 & b_2c_2\nu & b_3b_2 & b_2c_3\nu \\ b_1c_2\nu & c_1c_2 & b_2c_2\nu & {c_2}^2 & b_3c_2\nu & c_3c_2 \\ b_1b_3 & b_3c_1\nu & b_3b_2 & b_3c_2\nu & {b_3}^2 & b_3c_3\nu \\ b_1c_3\nu & c_1c_3 & b_2c_3\nu & c_3c_2 & b_3c_3\nu & {c_3}^2 \end{bmatrix} (4.28)$$

$$K_2 = \begin{bmatrix} {c_1}^2 & b_1c_1 & c_1c_2 & b_2c_1 & c_1c_3 & b_3c_1 \\ b_1c_1 & {b_1}^2 & b_1c_2 & b_1b_2 & b_1c_3 & b_1b_3 \\ c_1c_2 & b_1c_2 & {c_2}^2 & b_2c_2 & c_3c_2 & b_3c_2 \\ b_2c_1 & b_1b_2 & b_2c_2 & {b_2}^2 & b_2c_3 & b_3b_2 \\ c_1c_3 & b_1c_3 & c_3c_2 & b_2c_3 & {c_3}^2 & b_3c_3 \\ b_3c_1 & b_1b_3 & b_3c_2 & b_3b_2 & b_3c_3 & {b_3}^2 \end{bmatrix} (4.29)$$

$$b_1 = y_2 - y_3 \ (4.30) \qquad c_1 = x_3 - x_2 \ (4.31)$$

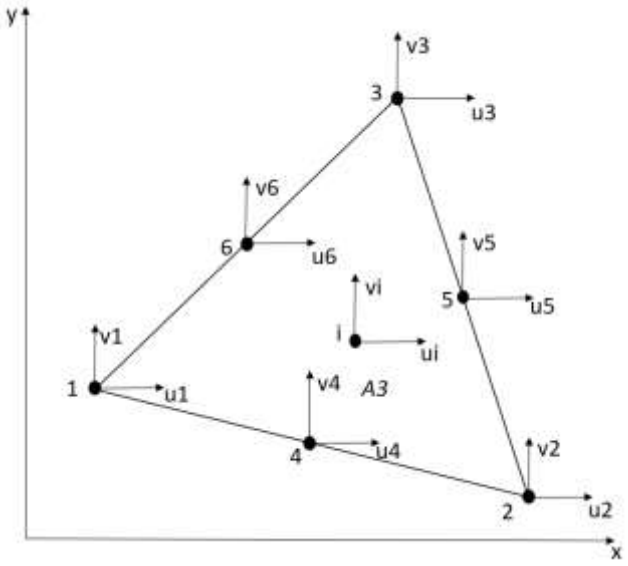
$$b_2 = y_3 - y_1 \ (4.32) \qquad c_2 = x_1 - x_3 \ (4.33)$$

$$b_3 = y_1 - y_2 \ (4.34) \qquad c_3 = x_2 - x_1 \ (4.35)$$

Therefore, the general procedure is first to calculate the displacements which lead us to the calculation of strains which result in the stresses.

4.2 Description of LST element

Despite all if its advantages like the vast variety of meshing algorithms and the simple mathematical modelling, the CST element has one very important disadvantage which is as its name denotes that throughout the whole element there is a constant strain. This happens because if we examine closely the equation () we will observe that the derivatives of shape functions are constant. So, to achieve a satisfying level of accuracy we need to have much of those elements that results in more computational time. For this reason, the LST element is used which is shown in Picture 4.2. In contrast to the CST element, the LST element has 6 nodes instead of 3 (the extra nodes are found in the middle of each side of the triangle).



Picture 4.2. LST element.

Again, to obtain the stresses we must first calculate the displacements and then the strains. The difference is that now is there are 6 nodes the displacements are given by the following equations:

$$u = S_1u_1 + S_2u_2 + S_3u_3 + S_4u_4 + S_5u_5 + S_6u_6 \text{ (4.36)}$$

$$v = S_1v_1 + S_2v_2 + S_3v_3 + S_4v_4 + S_5v_5 + S_6v_6 \text{ (4.37)}$$

where the new shape functions S_1, S_2, S_3, S_4, S_5 and S_6 are equal to:

$$S_1 = N_1(2N_1 - 1) \text{ (4.38)}$$

$$S_2 = N_2(2N_2 - 1) \text{ (4.39)}$$

$$S_3 = N_3(2N_3 - 1) \text{ (4.40)}$$

$$S_4 = 4N_1N_2 \text{ (4.41)}$$

$$S_5 = 4N_2N_3 \text{ (4.42)}$$

$$S_6 = 4N_3N_1 \text{ (4.43)}$$

All the other equation presented for CST element also apply here but with the new shape functions in them. However, looking back at equation () for the calculation of stiffness matrix, now because the shape functions contain x and y in the second power we must follow a different process to solve the integral. For this reason, we use the Gauss-Hammer integration and to do that we modify the shape functions. In particular, the Gauss-Hammer integration requires that the integration range be from 0 to 1 and from equation () we know that N_1, N_2 and N_3. Therefore, we set:

$$s = N_1 \text{ (4.44)}$$

$$t = N_2 \text{ (4.45)}$$

$$1 - s - t = N_3 \text{ (4.46)}$$

So, the shape functions take the following form:

$$S_1 = s(2s - 1) \text{ (4.47)}$$

$$S_2 = t(2t - 1) \text{ (4.48)}$$

$$S_3 = (1 - s - t)(2(1 - s - t) - 1) \text{ (4.49)}$$

$$S_4 = 4st \text{ (4.50)}$$

$$S_5 = 4t(1 - s - t) \text{ (4.51)}$$

$$S_6 = 4(1 - s - t)s \text{ (4.52)}$$

After this transformation of shape functions another problem that arises is the calculation of the derivatives of shape functions in respect to x and y to obtain matrix B, because now we have the shape functions in respect to s and t. This problem is solved with the use of the Jacobian matrix J with the following formula:

$$\begin{bmatrix} \frac{dS_i}{dx} \\ \frac{dS_i}{dy} \end{bmatrix} = J \begin{bmatrix} \frac{dS_i}{ds} \\ \frac{dS_i}{dt} \end{bmatrix} \text{ (4.53)}$$

The Jacobian matrix is defined as:

$$J = \begin{bmatrix} \frac{dx}{ds} & \frac{dy}{ds} \\ \frac{dx}{dt} & \frac{dy}{dt} \end{bmatrix} \text{ (4.54)}$$

and we can calculate the derivatives of x and y in respect to s and t by differentiating the following equations:

$$x = S_1x_1 + S_2x_2 + S_3x_3 + S_4x_4 + S_5x_5 + S_6x_6 \text{ (4.55)}$$

$$y = S_1y_1 + S_2y_2 + S_3y_3 + S_4y_4 + S_5y_5 + S_6y_6 \text{ (4.56)}$$

This happens because as it is already mentioned, shape functions are used to interpolate the displacements and the coordinates of any random point inside the element. To summarize, after all the necessary modification the equation () becomes:

$$K = \int_A B^T E_{tot} B \, dA \Rightarrow K = \int_y \int_x B^T E_{tot} B \, dx \, dy \Rightarrow$$

$$K = \int_0^1 \int_0^1 0.5 B^T E_{tot} B \, |J| ds \, dt \text{ (4.57)}$$

Now from equation (), we can perform the Gauss-Hammer integration which is given by the following formula:

$$K = \sum_i^m \sum_j^m 0.5 B^T E_{tot} B \, |J| \, w_i w_j \quad (4.58)$$

where w_i and w_j are the Gauss-Hammer integration weights. The $0.5 B^T E_{tot} B \, |J|$ is evaluated in the Gauss-Hammer integration points which are set in the position of s and t. After trials in the code, it is observed that if four or more Gauss-Hammer points are used then the integral yields an acceptable result. The results that will be presented are obtained for seven points. For the case of four points the values of w_i and w_j are given in Table 4.1 and for seven in Table 4.2.

s	$\frac{1}{3}$	$\frac{3}{5}$	$\frac{1}{5}$	$\frac{1}{5}$
t	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{3}{5}$	$\frac{1}{5}$
w	$-\frac{27}{48}$	$\frac{25}{48}$	$\frac{25}{48}$	$\frac{25}{48}$

Table 4.1. Values of w_i and w_j for four Gauss-Hammer weights.

s	$\frac{1}{3}$	0.597158717	0.4701420641	0.4701420641	0.7974269853	0.1012865073	0.1012865073
t	$\frac{1}{3}$	0.4701420641	0.597158717	0.4701420641	0.1012865073	0.7974269853	0.1012865073
w	0.225	0.1323941527	0.1323941527	0.1323941527	0.1259391805	0.1259391805	0.1259391805

Table 4.2. Values of w_i and w_j for four Gauss-Hammer weights.

4.3 Description of the meshing method

The meshing algorithm used for the creation of triangular elements over the gear tooth is chosen to be the Advancing Front Triangulation (AFT) method. This method was initially proposed by Lo and it was rapidly developed by the FE communities. The process of the AFT method is basically composed of the following steps:

Step 1: Creation of the outline nodes for the initiation of the algorithm.

In the beginning the user must provide a specific spacing function to create outline points in the boundaries of the geometry. These will serve as the first nodes. If the boundary is external, then nodes must be created in a counter clockwise direction and if the boundary is internal nodes must be created in a clockwise direction. In this way the boundaries of the geometry to be meshed is decomposed into straight lines.

Step 2: Selection of the base line segment.

Next, a line segment from the boundaries must be selected for the formation of the first element. Usually, this line segment is the last one created in Step 1.

Nevertheless, if there is a rapid change in the size in the element size then the stability of the meshing process and the quality of the elements can be improved if the shortest line segment is selected as base segment. The base segment is called AB.

Step 3: Creation of a new element

Since we have selected the base segment, we can now form a new element. For this purpose, the ideal location of the third node must be found which can belong to the boundary or it can be a totally new point inside our geometry. If the third node C belongs to the boundary, then to select it, we use the metric λ whose value is given by the following formula:

$$\lambda = \alpha\delta \quad (4.59)$$

where α is the geometrical shape factor of the triangle and is defined as:

$$\alpha = \frac{2\sqrt{3}AB \times AC}{\|AB\|^2 + \|BC\|^2 + \|CA\|^2} \quad (4.60)$$ (if the triangle is equilateral then $\alpha = 1$)

and δ is the node conformity factor and is defined as:

$$\delta = \delta_1 \delta_2 \quad (4.61)$$

$$\delta_1 = \min\left(\frac{\|AC\|}{\rho_1}, \frac{\rho_1}{\|AC\|}\right) \quad (4.62)$$

$$\delta_2 = \min\left(\frac{\|BC\|}{\rho_2}, \frac{\rho_2}{\|BC\|}\right) \quad (4.63)$$

In which $\rho_1$and ρ_2 are values of node spacing function evaluated at the mid-points of AC and CB, respectively. If the third node I does not belong to the boundary the same metrics are used for its evaluation but to the searching is done across the normal bisector of segment AB. Of course, this new node must be selected in a way that it does not intersect the boundary. Finally, the λ values of node C and I are compared and the node that has the biggest λ value is chosen for the formulation of the new element.

Step 4: Boundary update

If the node C is selected for the new element, then:

- If CA belongs to the boundary, then it is deleted from the boundary.
- If BC belongs to the boundary, then it is deleted from the boundary.

But, if node I is selected for the new element, then both CA and BC are added to the boundary. Thus, a new boundary is formed. For the creation of a new element, we return to Step 2 and the whole process is repeated.

Step 5: End of meshing process

The meshing process ends when there are no sides left in the boundary and this is when we receive the final triangular mesh.

4.4 Structure of MATLAB code

In the following paragraphs the MATLAB codes of CST and LST elements for the modelling of gear contact will be presented. The general idea of the code is to model only one of the two teeth that participate in a contact and apply the normal force in it.

4.4.1 Definition of gear parameters

The first section of the code is the definition of gear parameters. All these parameters are given by the user and stored in the variable `gear` which is defined as struct in MATLAB. The fields of this struct are shown in Table 4.3.

`gear.z=30`	Number of teeth for pinion gear
`gear.m=4`	Module
`gear.a0`	Pressure angle
`gear.cs`	Thickness coefficient at rolling circle
`gear.ck`	Addendum coefficient
`gear.cf`	Dedendum coefficient
`gear.cc=0.3`	Rack curvature coefficient
`gear.width`	Gear width
`gear.i`	Gear ratio
`gear.contact_r`	Array of dimensions 1×2, in which the first element is the radius from the centre of pinion gear to the point of contact and the second is the radius from the centre of conjugate gear to the point of contact

Table 4.3. User defined gear parameters.

The struct *gear* is then inserted to the equation `gear_radii_angles` for the calculation of some secondary gears parameters which are also saved as fields as shown in Table 4.4.

`gear.r0`	Pitch radius of pinion gear
`gear.rg`	Form radius of pinion gear
`gear.rk`	Tip radius of pinion gear
`gear.rf`	Root radius of pinion gear
`gear.rc`	Trochoid radius of pinion gear
`gear.fi`	Involute rotation angle
`gear.w`	Trochoid rotation angle

Table 4.4. Secondary gear parameters.

4.4.2 Definition of material properties

In this section the user provides the Young Modulus and the Poisson ratio of the material of each gear. In addition, the user can choose if the problem is in Plane Stress or in Plane Strain. These are stored in the struct `material`. The fields of this struct are shown in Table 4.5.

`material.E`	Array of dimensions 1×2, in which the first element is the Young Modulus of pinion and the second is the Young Modulus of conjugate gear
`material.v`	Array of dimensions 1×2, in which the first element is the Poisson ratio of pinion and the second is the Poisson ratio of conjugate gear
`material.PlaneType`	If this is set to 0 then we are in Plane Stress and if it is set to 1 we are in Plane Strain.

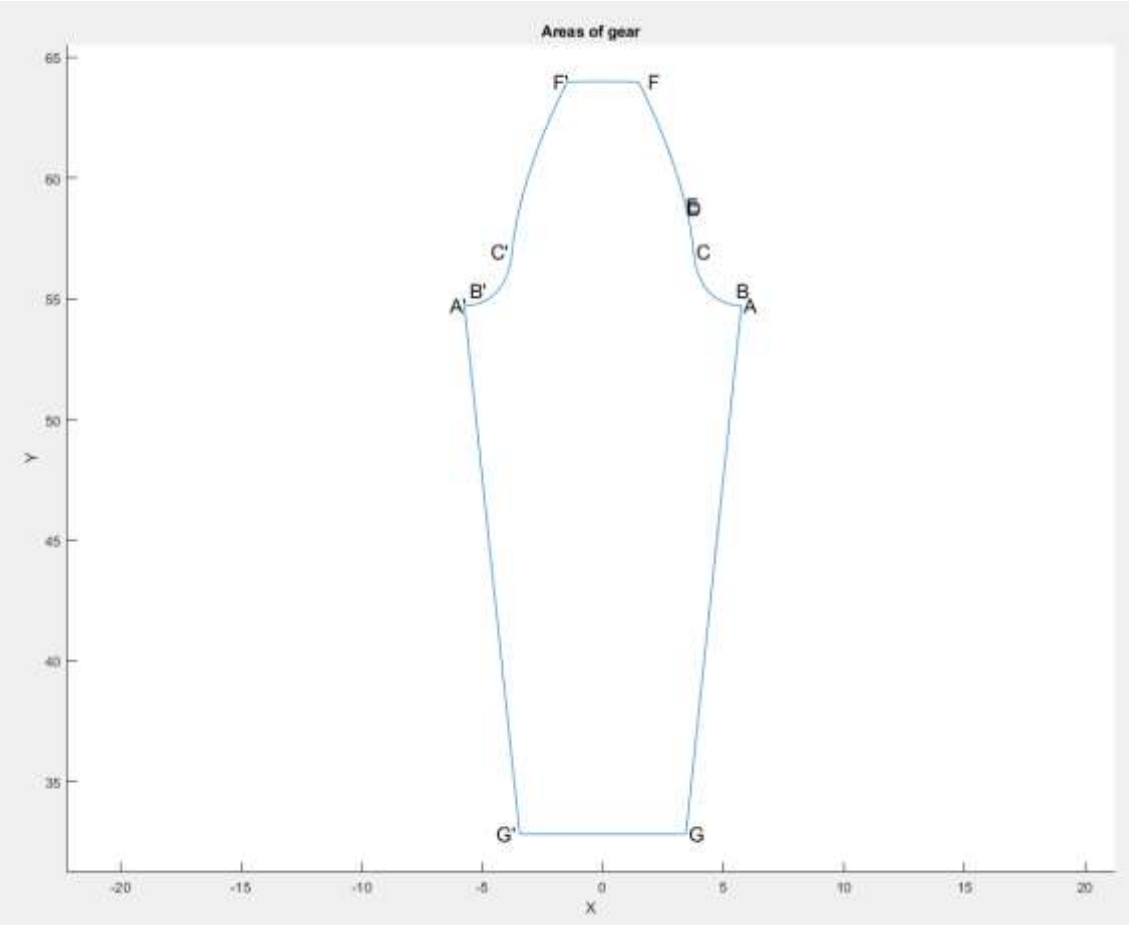
Table 4.5. User defined material properties.

4.4.3 Hertz calculations

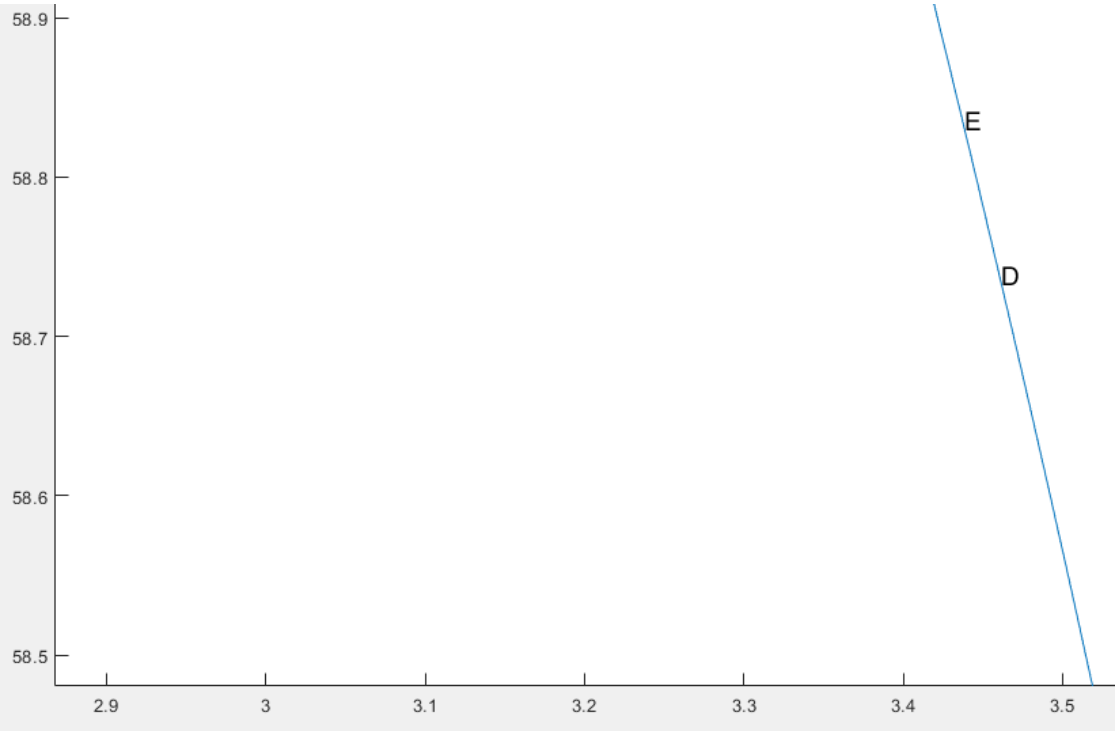
In this section the calculation of Hertz quantities is done. This is performed through the equation `hertz_calc` which takes as input the structs `gear` and `material` and the contact force `F` provided by the user and gives as outputs the maximum theoretical contact pressure `pmax_th` and the theoretical height `b_th` of the rectangular contact area as described in chapter 2. These are used both to verify our results and to aid the meshing process and the load application process. In addition, the force `F` is calculated by a MATLAB program written by the Ph.D. student Christos Papalexis.

4.4.4 Code implementation of the AFT method

As it is already mentioned to use the AFT method, we first need to create outline nodes in the boundary of the gear tooth to divide it into line segments. Therefore, we initially divide the boundary into the parts shown in Picture 4.3 and Picture 4.4 for the engagement of gears in Position 6. The type of the curve in every part is explained in Table 4.6.



Picture 4.3. Gear boundary.



Picture 4.4. DE section of gear boundary.

AB	Root circle
BC	Trochoid
CD	Involute curve
DE	Contact area in the involute curve
EF	Involute curve
FF´	Tip circle
F´C´	Involute curve
C´B´	Symmetrical part of **BC**
B´A´	Symmetrical part of **AB**
A´G´	
G´G	Gear hub circle
GA	Symmetrical part of **G´G**

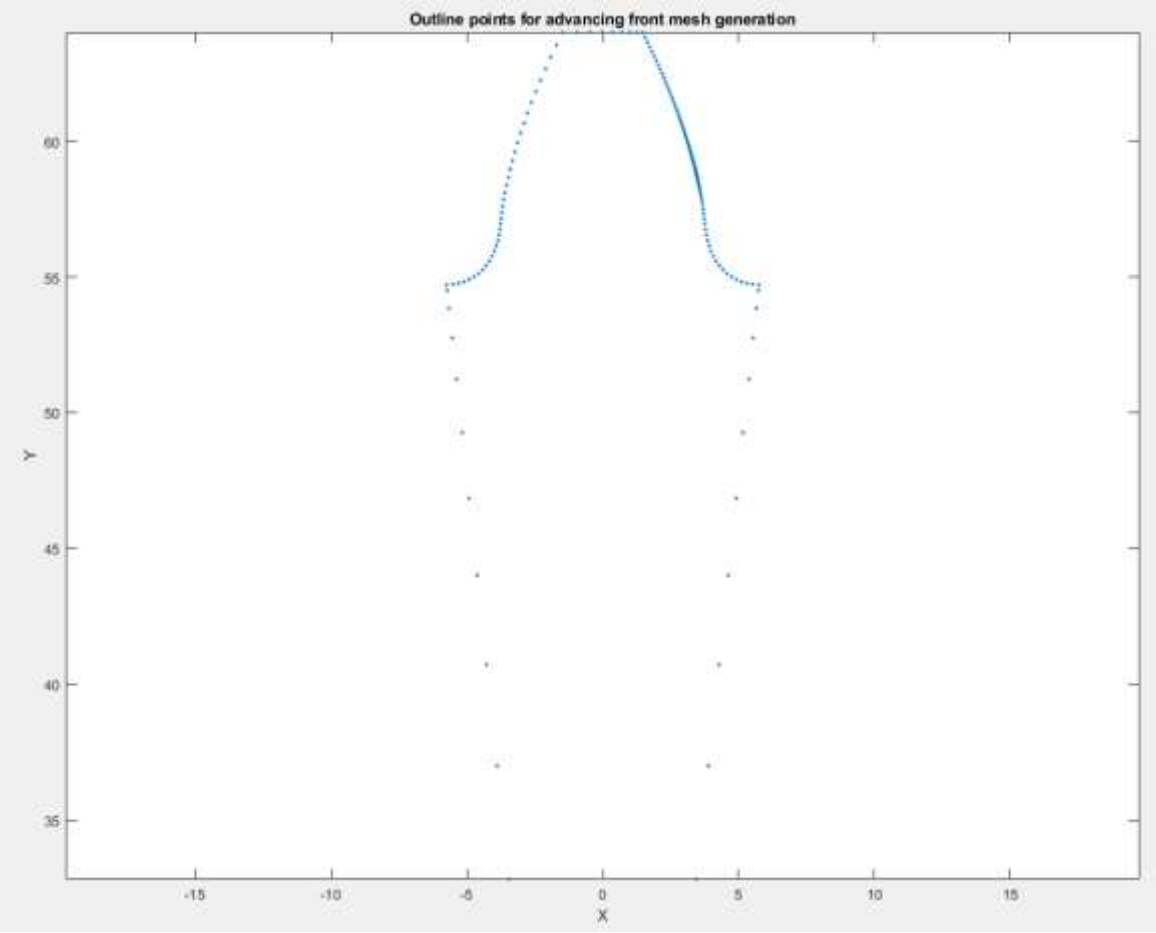
Table 4.6. Divisions of the boundary of gear.

This is initial division of the boundary is done for the user to be able to set a different element size in each part, because for example in the part of trochoids and in the part of contact area we want to have a finer mesh. So, for each one of those parts we use a desired mesh size the value if which is stores in the matrix `mesh_seeds`. The dimensions of this matrix are 12×2, because the boundary was initially divided to 12 areas and its row of the matrix contains the mesh size or in other words the spacing for the nodes of the boundary. The code is structured in a way that if in a row the first element is a certain value and the second is 0 then the part of the boundary associated to this row will have nodes with a spacing equal to the first element of the row. However, if both elements of the row are non-zero then, the part of the boundary corresponding to this row will contain nodes whom spacing will vary linearly from the value of the first element to the value of the second element of this row. For the values in `mesh_seeds` shown in Picture 4.5 we receive the outline nodes shown in Picture 4.6. This

procedure takes place in the equation `outline_points` which gives as output the matrix `outline_points` in which each row contains the coordinates of each outline point. In addition, from this equation we receive the matrices `loaded_nodes` which contains the nodes that belong to the section **DE** or in other words the contact area, `fixed_nodes` which contains the nodes that belongs to the sections **A΄G΄**, **G΄G**, and **GA** which will be fixed and `bending_nodes` which contains the nodes found in the sections **BC** and **C΄B΄** or in other words the trochoids.

```
mesh_seeds = [0.2,0;...        %(A-B)
              0.2,0;...        %(B-C)
              0.2,b_th/10;...        %(C-D)
              b_th/10,0;...        %(D-E)
              b_th/10,0.2;...      %(E-F)
              0.2,0.5;...        %(F-F')
              0.2,0.5;...      %(F'-C')
              0.2,0;...        %(C'-B')
              0.2,0;...        %(B'-A')
              4,0.2;         %(A'-G')
              5,0;           %(G'-G)
              4,0.2];      %(G-A)
```

Picture 4.5. Mesh seeds

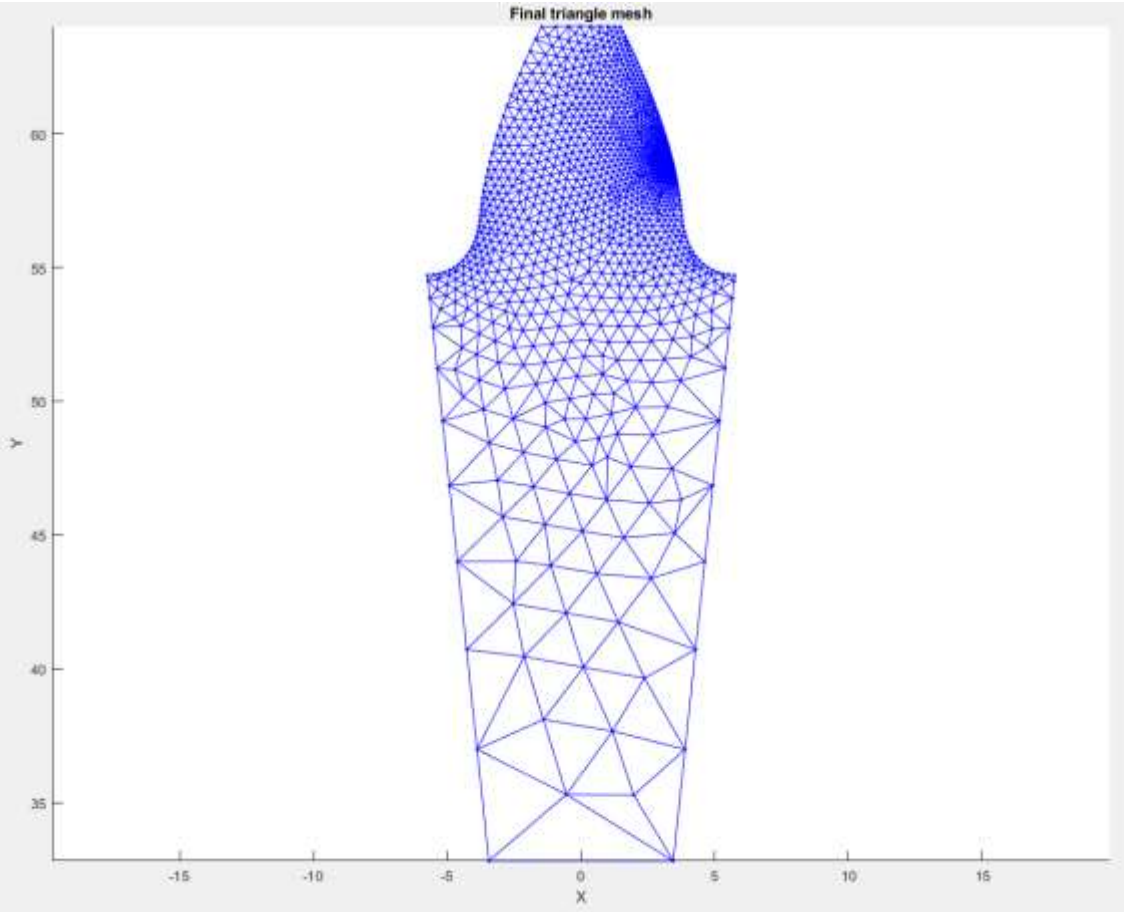


Picture 4.6. Outline points

Afterwards the triangular mesh is created with the steps described in paragraph . This is done with the equation `Advancing_Front_LO_2D` which takes as main input the matrix `outline_points` and returns as output the coordinates of the nodes of the final mesh which are stored in the column matrices `Xnodes` and `Ynodes` which are the x-intercepts and the y-intercepts of the nodes respectively, the number of nodes which is stored in the variable `NN` , the number of elements which is stored in the variable `NE` and the connectivity matrix which is stored in the matrix `ME` and as its name states it provides information about the way that the nodes are connected to form the elements. Finally, the column matrices `Xnodes` and `Ynodes` are stored in the two-column matrix `nodes` and the matrix `ME` is inserted into the following $for-loop$ for the formation of the matrix `elements`:

```
elements=zeros(NE,3);
for i=1:NE
    elements(i,1)=ME(3*(i-1)+1);
    elements(i,2)=ME(3*(i-1)+2);
    elements(i,3)=ME(3*(i-1)+3);
end
```

Each row of the matrix `elements` corresponds to one element and contains the 3 nodes that compose it if we are referring to a CST element. For instance, the first row corresponds to the first element and so on. In Picture 4.7. the final mesh appears which is done with the equation `plot_tri_mesh`.

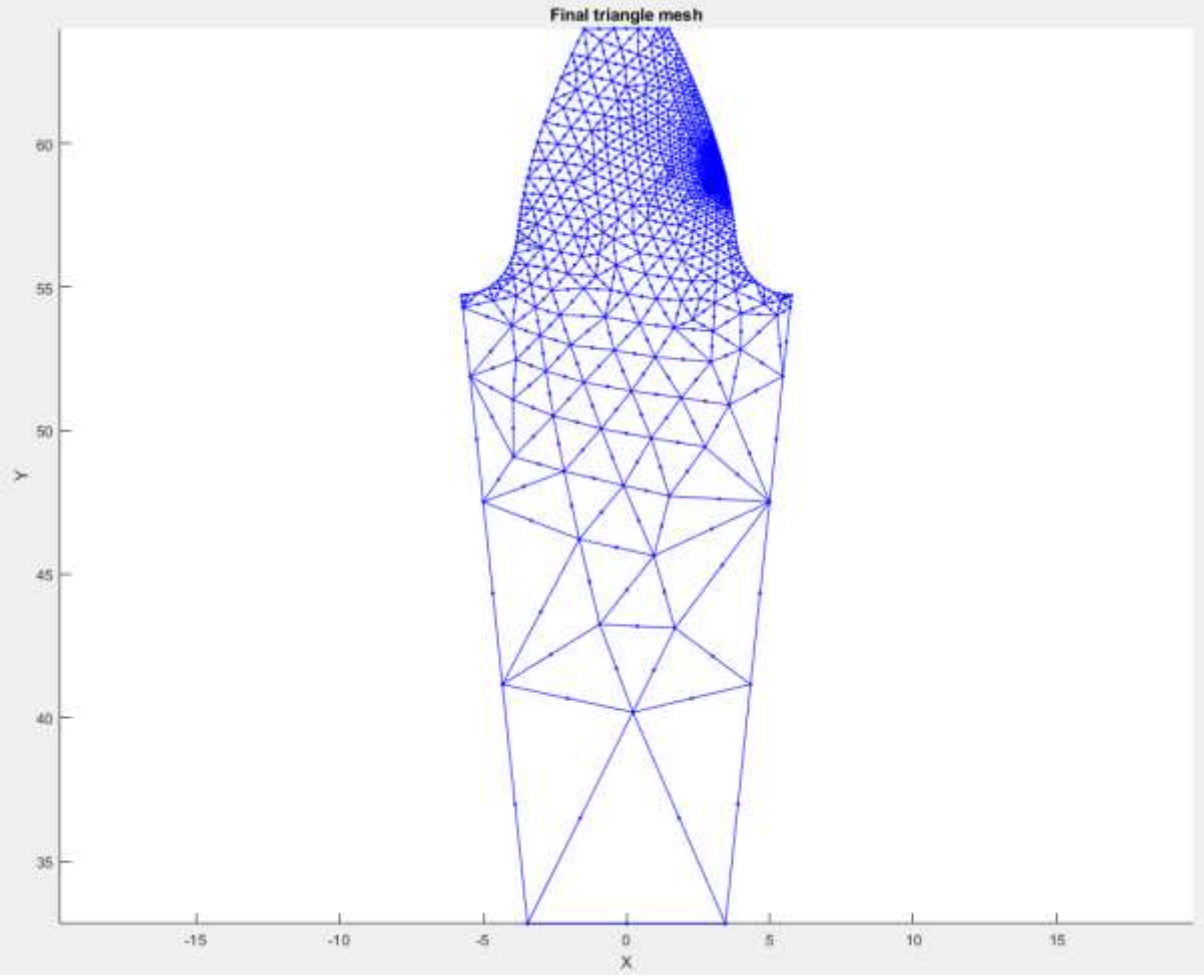


Picture 4.7. Meshed gear with CST elements.

The code also offers the possibility for the creation of figures with the numbered nodes and elements which are done with the following equations:

- `plot_tri_mesh_nodal_points_numbered`
- `plot_tri_mesh_elements_numbered`
- `plot_tri_mesh_nodes_and_elements_numbered`

Although, this procedure is enough for the CST element, when the code with the LST elements is used, a function is also required for the insertion of mid-nodes in the sides of each triangular element. This is done with the equation. Now, the matrix `elements` contains the 6 nodes that compose each LST element. Again, the final mesh is shown in Picture .4.8 and is done with the equation `plot_six_node_tri_mesh`.



Picture 4.8. Meshed gear with LST elements.

The creation of figures with the numbered nodes and elements which are done with the following equations:

- `plot_six_node_tri_mesh_nodal_points_numbered`
- `plot_six_node_tri_mesh_elements_numbered`
- `plot_six_node_tri_mesh_node_and_elements_numbered`

4.4.5 Load application

In this section the load application is done. This is one of the most important parts of the code because this offers a way to avoid the full modelling of the contact between the gears -like it was done in ANSYS Workbench- and to decouple the gears, by just moving the contact force to the pinion gear. This procedure is done via the equation `load_distribution` and `load_distribution_six_nodes` for the CST and LST elements, respectively. This equation has as output the array `Fext` that contains all the nodal forces. The core function of this equation of is to divide the contact force `F` to the nodes found in the contact area by following the theoretical elliptical distribution of contact pressure in Hertz theory. given in equation () These nodes are stored in the matrix `loaded_nodes` as mentioned earlier. If in this equation we replace x by l where $l \in [-b\_th, b\_th]$ and we replace the fraction $\frac{l}{b\_th}$ by div where $div \in [0,1]$ then the equation () can be rewritten as:

$$p = pmax\_th\sqrt{1 - div^2} \text{ (4.64)}$$

In the code dive is an array of dimensions $1 \times number\ of\ loaded\_nodes$. So, each two consecutive elements of div correspond to two consecutive values of p and to two consecutive nodes that form the side of the triangular element that belongs to the contact area. Since we are interested only in the side of the element that belongs to the contact area, we can assume without loss of generality that in a CST element this side is formed by nodes 1 and 2 and on this side, it is true that:

$$ss = N_1 \text{ (4.65)}$$

$$1 - ss = N_2 \text{ (4.66)}$$

$N_3 = 0$ (4.67), this applies because we are on side 12 of the element.

The total nodal force F_1that corresponds to node 1 is equal to:

$$F_1 = \int_A N_1 p dA \Rightarrow F_1 = gear.width \int_l N_1 pmax\_th \sqrt{1 - (\frac{l}{b\_th})^2} dl \Rightarrow$$

$$F_1 = gear.width \int_0^1 ss\ pmax\_th\sqrt{1 - ((1-ss)div_1 + ss * div_2)^2}\sqrt{(\frac{dx}{dss})^2 + (\frac{dy}{dss})^2} dss$$

(4.68)

Where div_1 and div_2 are the values of the array div for nodes 1 and 2. Of course instead of node 1 and 2 we could have nodes i and $i+1$ and the total nodal forces F_i and F_{i+1}. Respectively the total nodal force F_2 for node 2 is equal to:

$$F_2 = gear.width \int_0^1 (1 -$$

$$ss)\ pmax\_th\sqrt{1 - ((1-ss)div_1 + ss * div_2)^2}\sqrt{(\frac{dx}{dss})^2 + (\frac{dy}{dss})^2} dss \text{ (4.69)}$$

We said that these are the total nodal forces, but we also need to find the x-component and the y-component of those forces and insert them into the array F_{ext}. In general, when we refer to the nodes i and $i+1$ we set as:

$$F_{ext}(2i-1) = F_{i,x} = F_i \cos angle \text{ (4.70)}$$

$$F_{ext}(2i) = F_{i,y} = F_i \sin angle \text{ (4.71)}$$

$$F_{ext}(2(i+1)-1) = F_{i+1,x} = F_{i+1} \cos angle \text{ (4.72)}$$

$$F_{ext}(2(i+1)) = F_{i+1,y} = F_{i+1} \sin angle \text{ (4.73)}$$

where $angle$ is the angle that is formed between the x-axis and the total force vector, which always is perpendicular to the contact area. Thus, a formula for the calculation of thus angle is:

$$angle = \tan^{-1} \frac{y_{loaded\_nodes(end)} - y_{loaded\_nodes(1)}}{x_{loaded\_nodes(end)} - x_{loaded\_nodes(1)}} - \frac{pi}{2} \text{ (4.74)}$$

For the case of LST element we apply all the above but now since its side of the element has 3 nodes, we must calculate at first 3 total nodal forces. Again, without loss of generality we can assume that the nodes that interest as are 1,2 and 4 where the node 4 is found in the middle of the side 12. From paragraph we have the shape functions of LST element in parametric form, so we can just replace s with ss and t with $1 - ss$ and receive the following formulas for the total nodal forces:

$$F_1 = gear.width \int_0^1 ss(2 * ss - 1)\, pmax\_th \sqrt{1 - ((1-ss)div_1 + ss * div_2)^2} \sqrt{(\frac{dx}{dss})^2 + (\frac{dy}{dss})^2} dss \text{ (4.75)}$$

$$F_2 = gear.width \int_0^1 (1-ss)(2 * (1-ss) - 1)\, pmax\_th \sqrt{1 - ((1-ss)div_1 + ss * div_2)^2} \sqrt{(\frac{dx}{dss})^2 + (\frac{dy}{dss})^2} dss \text{ (4.76)}$$

$$F_4 = gear.width \int_0^1 4ss(1 - ss)\, pmax\_th \sqrt{1 - ((1-ss)div_1 + ss * div_2)^2} \sqrt{(\frac{dx}{dss})^2 + (\frac{dy}{dss})^2} dss \text{ (4.77)}$$

An important difference here is that div_1 and div_2 still corresponds to the values of the array div for nodes 1 and 2 but nodes 1 and 2 are no longer consecutive because between them there is node 4. After that, we can fill the array F_{ext} as shown previously.

4.4.6 Boundary conditions application

The application of boundary conditions is done via the array `BC` in the code. This is an array of size `2NN`, because there are `NN` nodes and consequently $DOFs$ (Degrees Of Freedom) are `2NN`. Thus, there are two cases:

$$BC(DOF) = \begin{cases} 0, & if\ not\ fixed \\ 1, & if\ fixed \end{cases} \text{ (4.77)}$$

The tooth is modelled in a way that the nodes of sides **A΄G΄**, **G΄G**, and **GA** are completely fixed. So, if node q belongs to one of these sides, then:

$$BC(2q-1) = 1 \text{ (4.78)}$$

$$BC(2q) = 1 \text{ (4.79)}$$

Of course, these nodes are already found previously in the code and stored in the array `fixed_nodes`.

4.4.7 Solver and Post processor

In this final section the solving of the finite element model is performed and its post processing. The model is solved through the equation `FEA` for the code with CST elements and through the equation `FEA_six_nodes` for the code with the LST elements. In both equations the inputs are the matrices `nodes`, `elements`, `BC`, `Fext` and `U` and the variables `NODES` and `NELE` and the output is the matrix `U` which contains the nodal displacements calculates as discussed in paragraphs 4.1 and 4.2 . For the CST element the post processing is done in the equation `Element_Strain_Stress` where we receive the element stresses and strains, because as it is already mentioned the CST element has a single value for strain and stress throughout the element. The inputs of the equation are the matrices *nodes* and *elements* and `U`, the variable `NELE` and the struct `material`. The outputs of equation are presented in Table 4.7. Furthermore, the code through the equations `SmoothNodalStress` and `SmoothElementStress` which provide the user with the opportunity to see the smoothed stresses.

`Strain`	Element normal and shear strains
`Stress`	Elements stresses. The first three columns contain the normal and shear stresses and the rest contain the principal stresses and the equivalent Von-Mises stresses
`PrincipStress`	Element principal stresses
`Svm`	Element equivalent Von-Mises stresses

Table 4.7. Outputs of the equation `Element_Strain_Stress`.

The post processing for the LST element is don with the equation `Nodal_Strain_Stress_six_nodes`. Through the LST element we can directly receive nodal strain and nodal stresses so there is no need for smoothening. Once again, the inputs are the matrices `nodes`, `elements`, `BC`, `Fext` and `U`, the variable `NELE` and the struct `material`. The outputs of equation are shown in Table 4.8.

`StrainNodal`	Nodal normal and shear strains
`StressNodal`	Nodal stresses. The first three columns contain the normal and shear stresses and the rest contain the principal stresses and the equivalent Von-Mises stresses
`PrincipStressNodal`	Nodal principal stresses
`SvmNodal`	Nodal equivalent Von-Mises stresses

Table 4.8. Outputs of the equation `Nodal_Strain_Stress_six_nodes`.

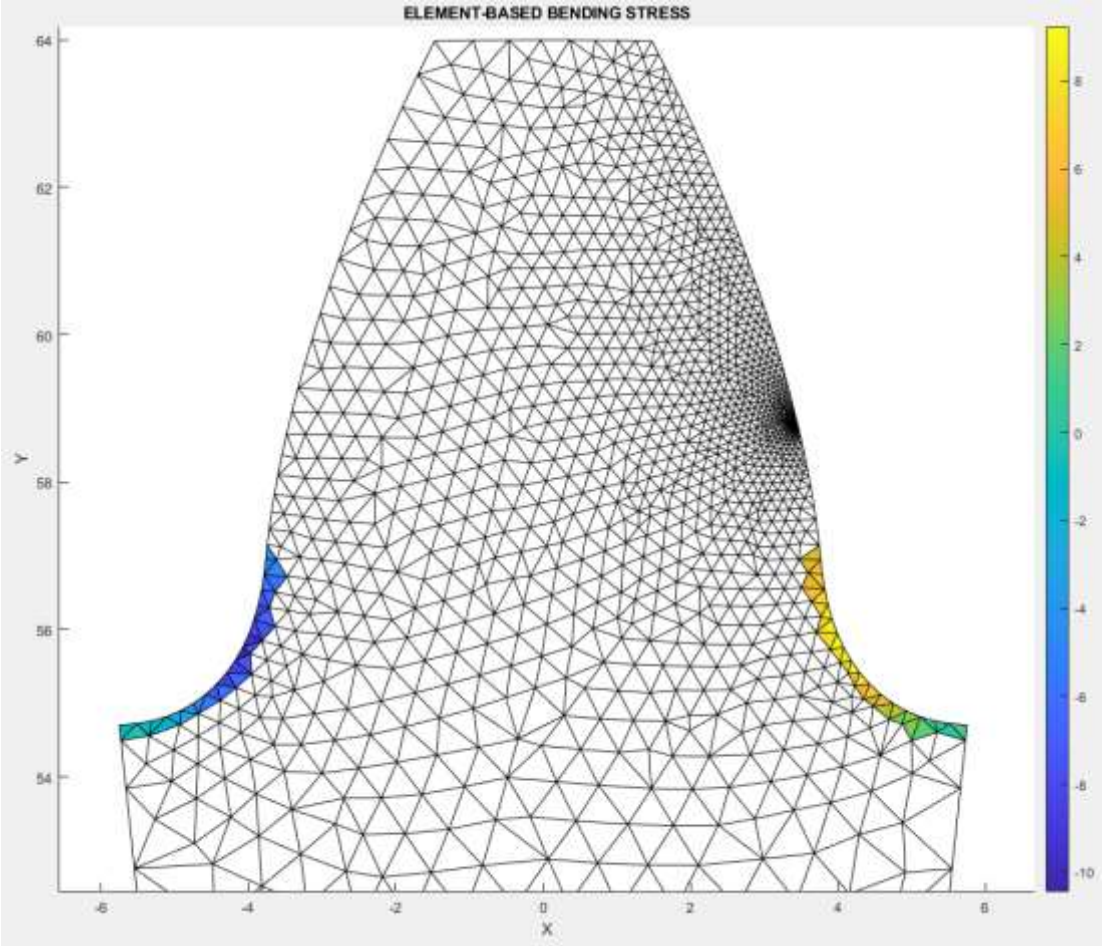
4.5 Results and verification of MATLAB code

After the simulation has run the results in Contact Pressure and Bending stress in each contact pair will be presented. Moreover, there will be a comparison between the theoretical results,

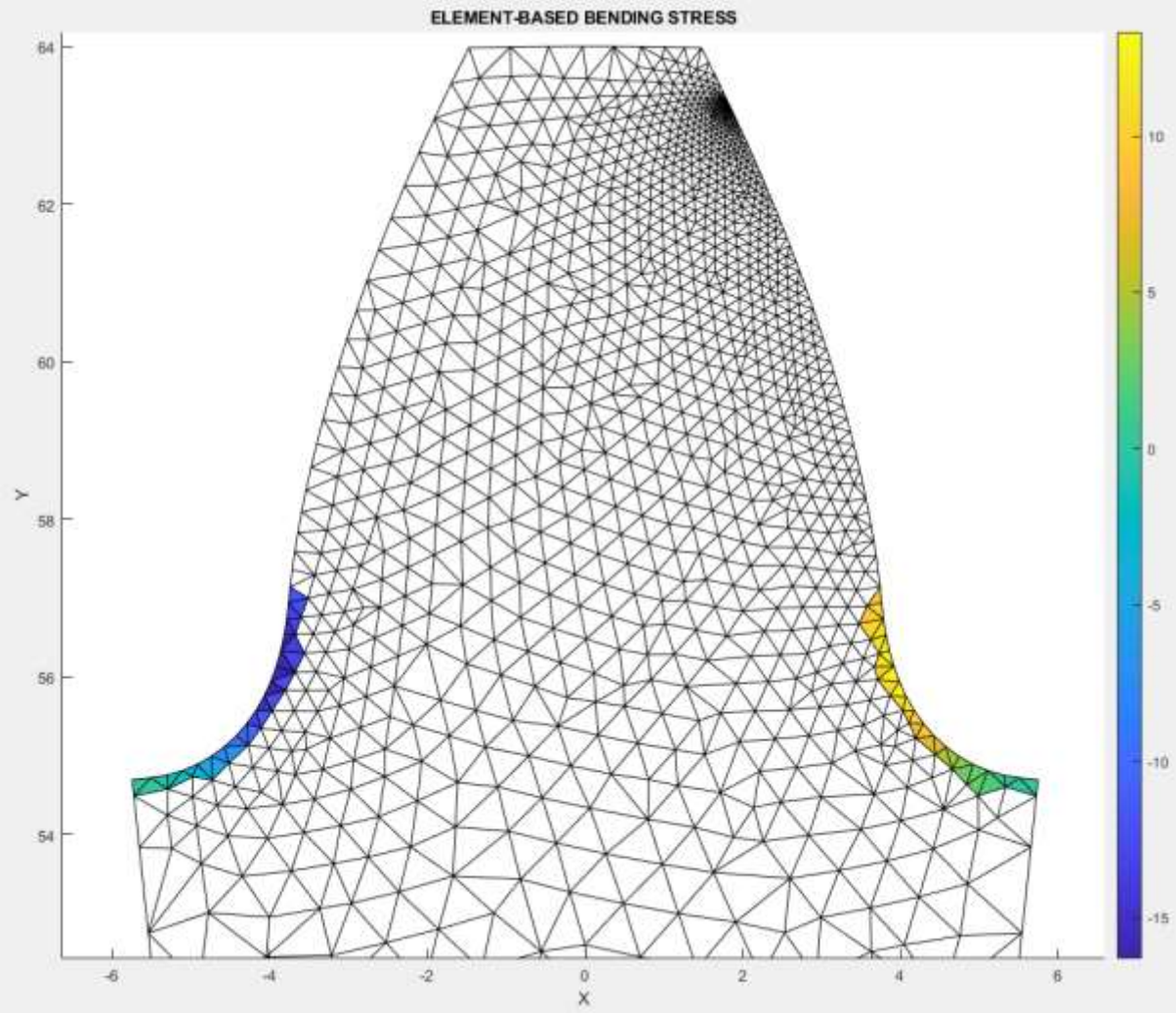
the results from the two-dimensional analysis in ANSYS and the results from the MATLAB codes.

Bending Stress

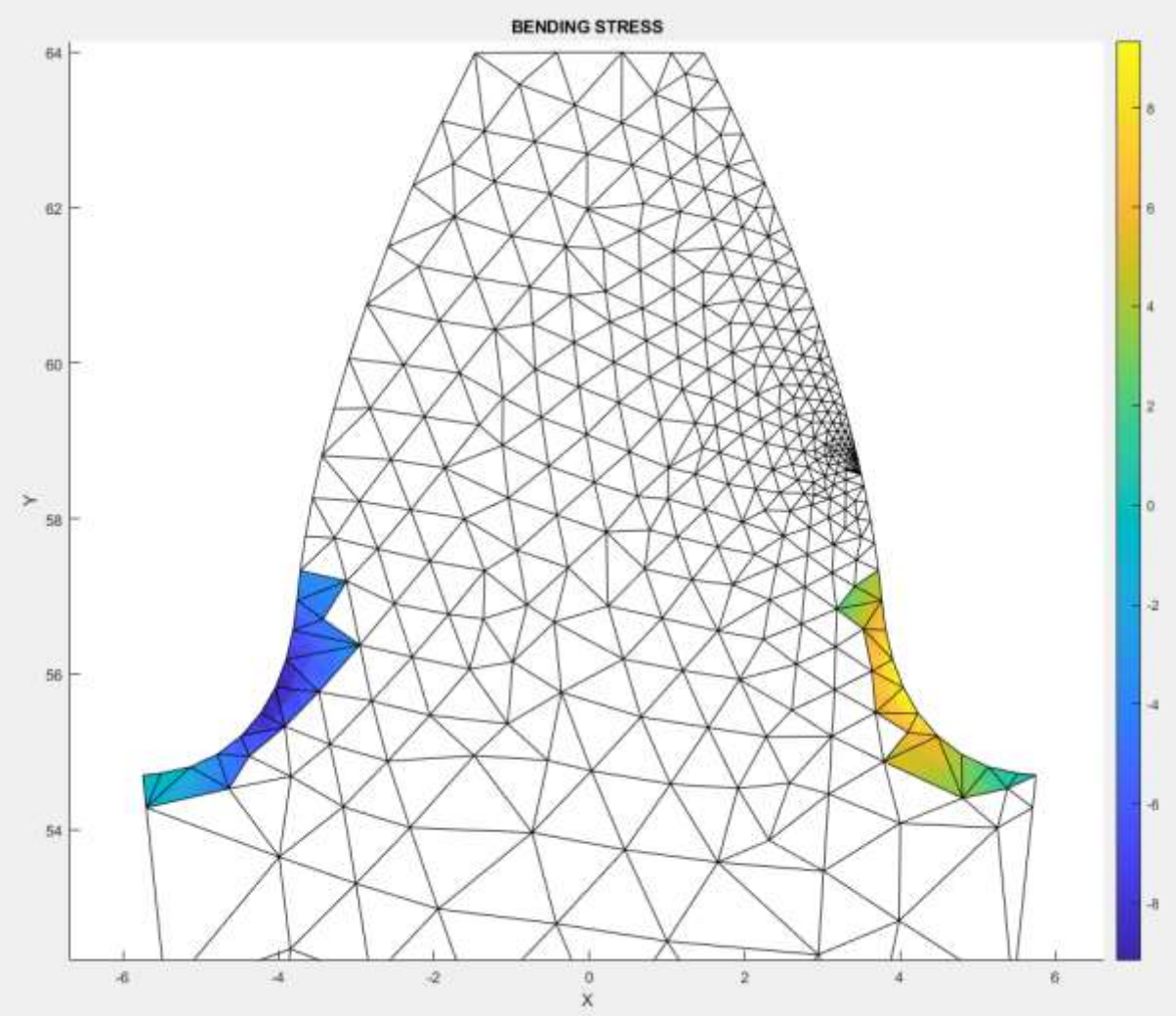
The Bending Stress of conjugate gear for Position 6 and Position 6'are shown in Picture 4.9 and Picture 4.10 for the CST elements and in Picture 4.11 and Picture 4.12 for the LST elements. In Table 4.9 a comparison is done between the results from MATLAB and the theoretical ones and the ANSYS results from the two-dimensional analysis.



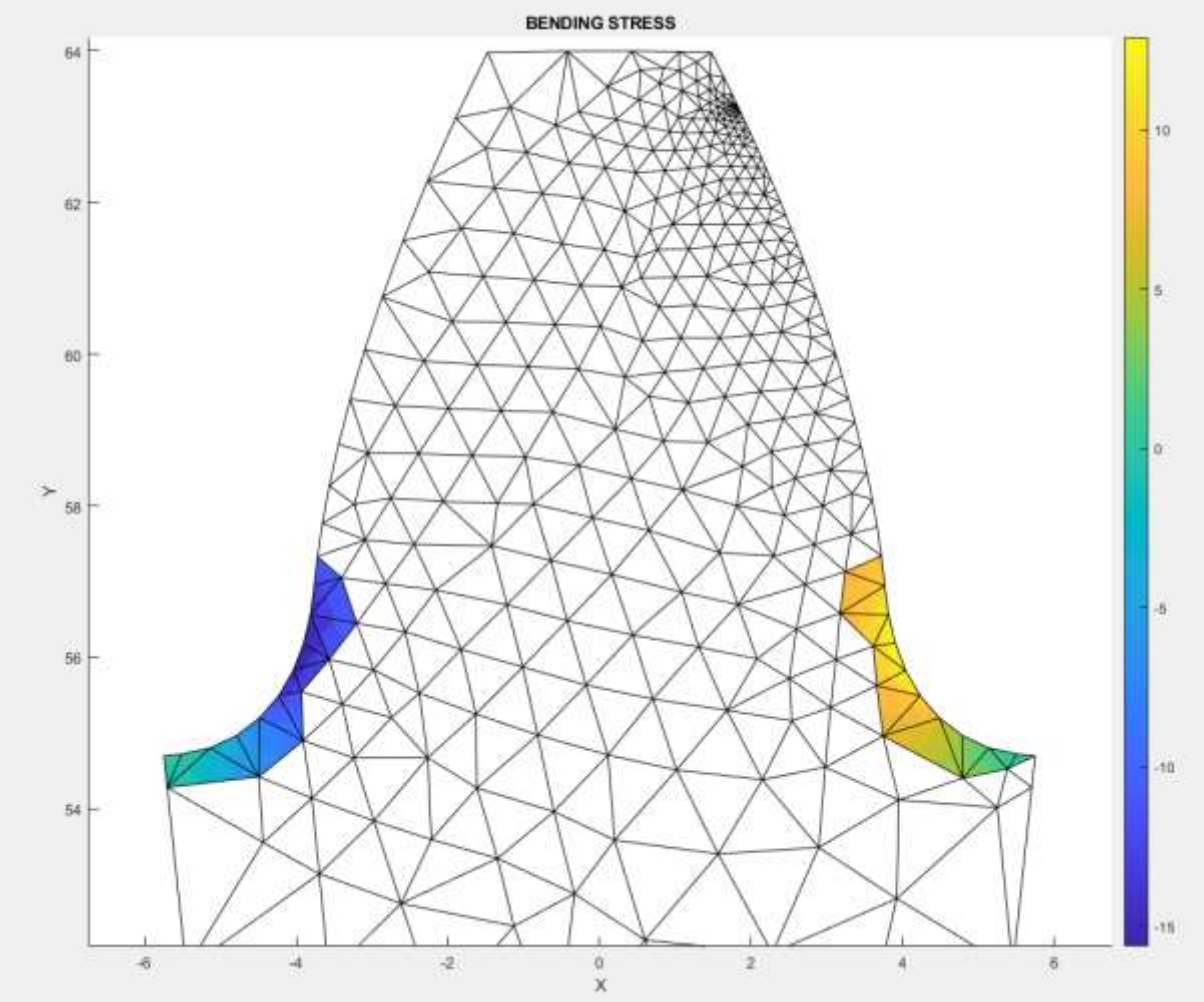
Picture 4.9. Bending stress of conjugate gear in Position 6 with CST elements.



Picture 4.10. Bending stress of conjugate gear in Position 6' with CST elements.



Picture 4.11. Bending stress of conjugate gear in Position 6 with LST elements.



Picture 4.12. Bending stress of conjugate gear in Position 6' with LST elements.

	Position 6		Position 6'	
	Left	**Right**	**Left**	**Right**
Bending stress ANSYS (MPa)	9.99	9.05	16.61	14.74
Bending stress MATLAB CST (MPa)	10.44	9.24	16.31	13.32
Bending stress MATLAB LST (MPa)	9.16	9.33	15.64	12.92
Bending stress Theoretical (MPa)	15.29		11.30	
Error ANSYS (%)	37.74		38.7	
Error MATALAB CST (%)	35.64		31.11	
Error MATALAB LST (%)	39.54		26.37	

Table 4.9. Comparison of Bending Stress results of conjugate gear in Position 6 and Position 6'.

From Table 4.9 it is obvious that the MATLAB code yields better results since the errors are smaller. Moreover, from Table 4.10 we observe that the MATLAB models save 75% more computational time for the CST elements and 85% more computational time for the LST elements than the ANSYS model.

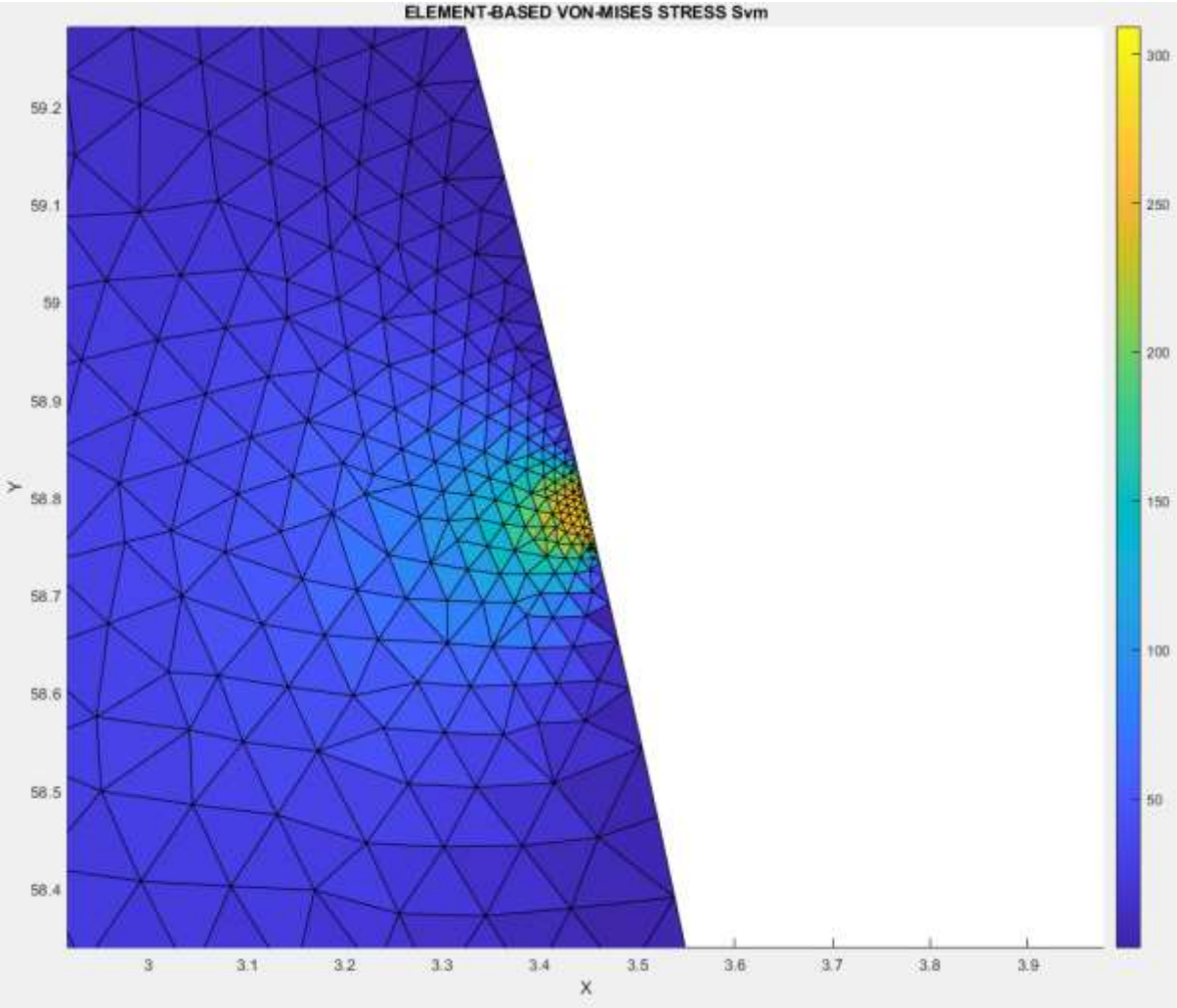
	ANSYS	MATLAB CST	MATLAB LST
Setup and Solving time (min)	20	5	3

Table 4.10 . Comparison of simulation times between MATLAB and ANSYS models.

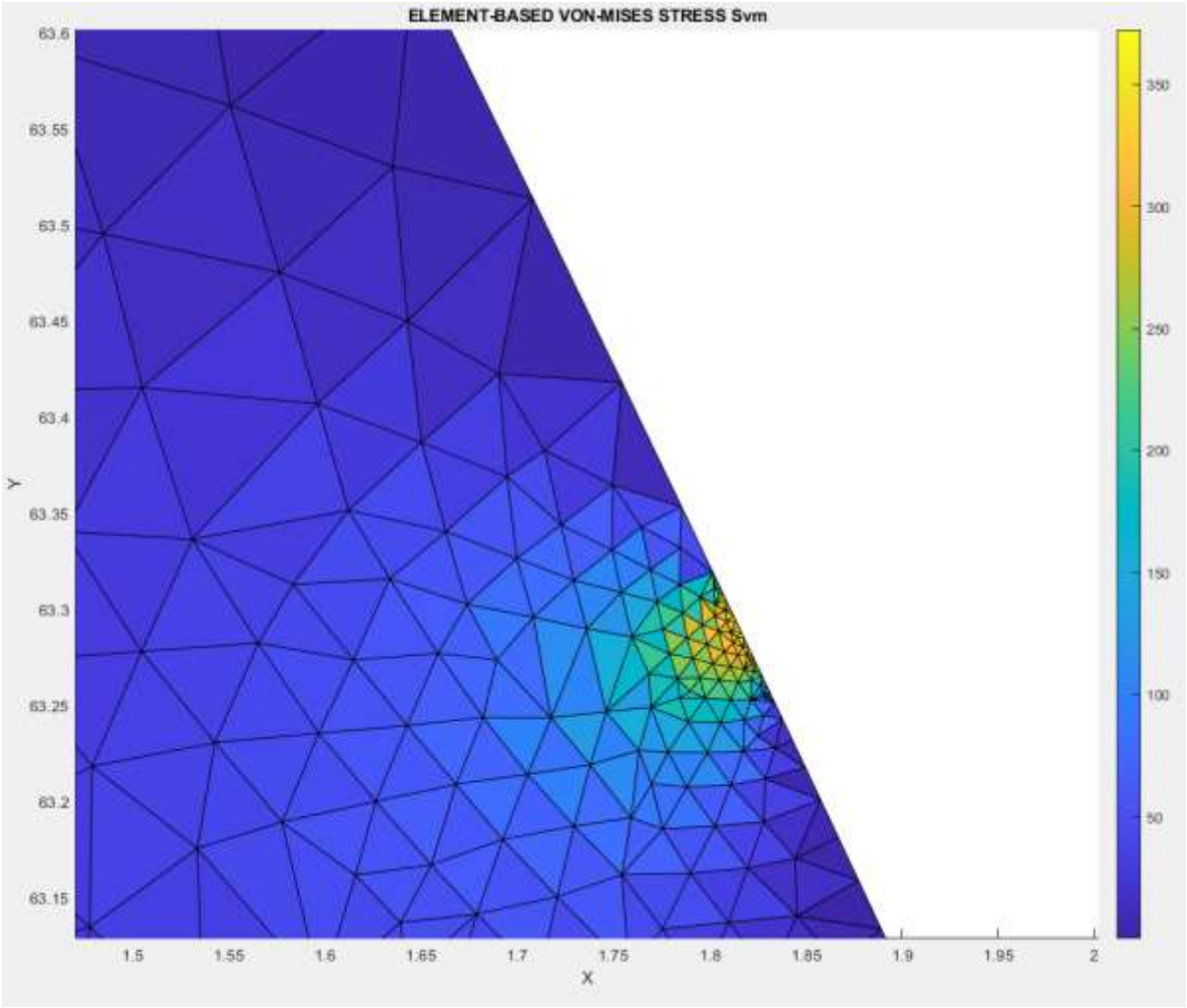
Finally, another thing worth mentioned is that model with LST elements was able to provide the same quality of results and even better by using half of the mesh sizes used in CST elements and thus using of the computational time.

Contact Pressure

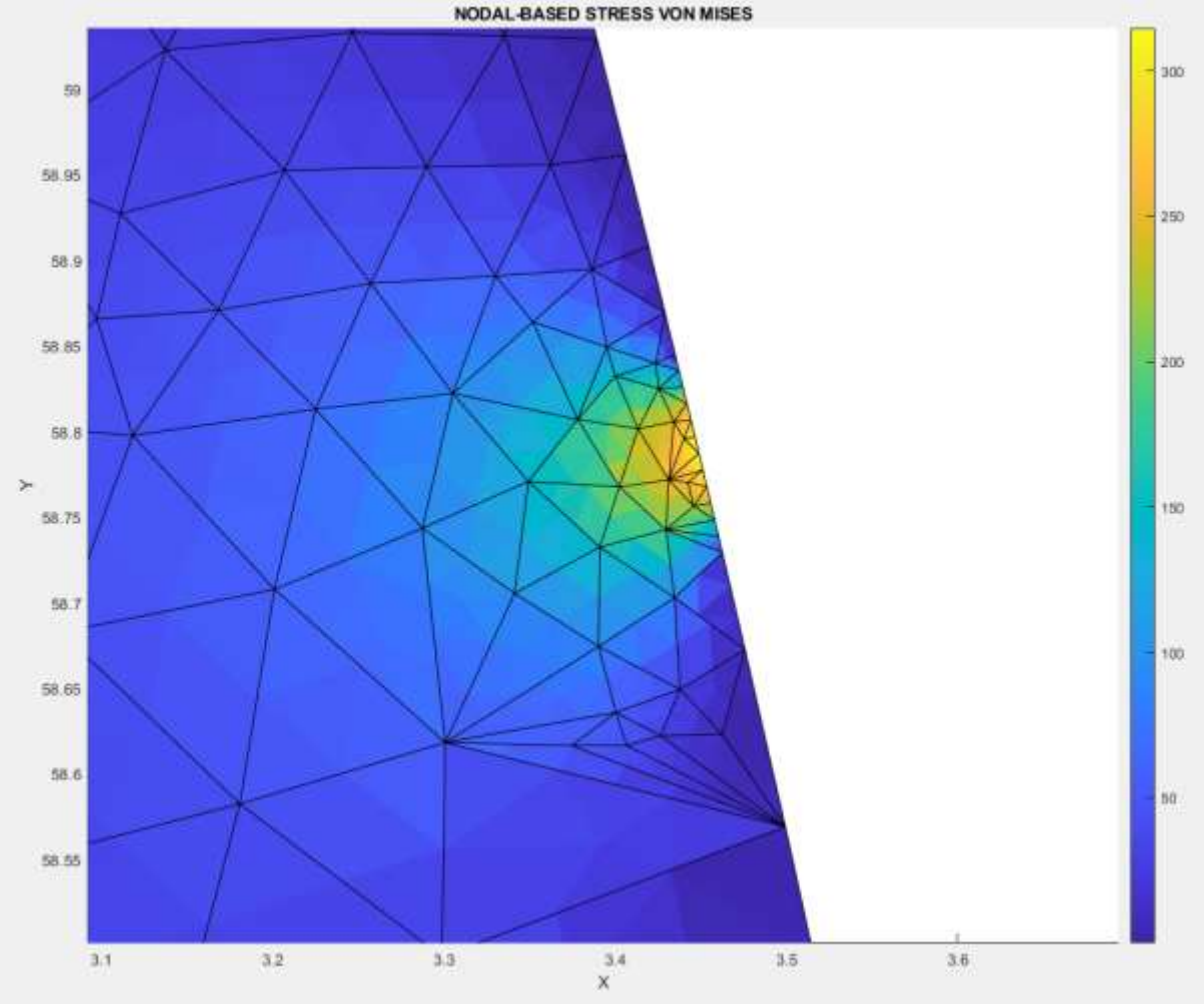
The Contact Pressure for Position 6 and Position 6'are shown in Picture 4.13 and Picture 4.14 for the CST elements and in Picture 4.15 and Picture 4.16 for the LST elements. In Table 4.11 a comparison is done between the results from MATLAB and the theoretical ones and the ANSYS results from the two-dimensional analysis.



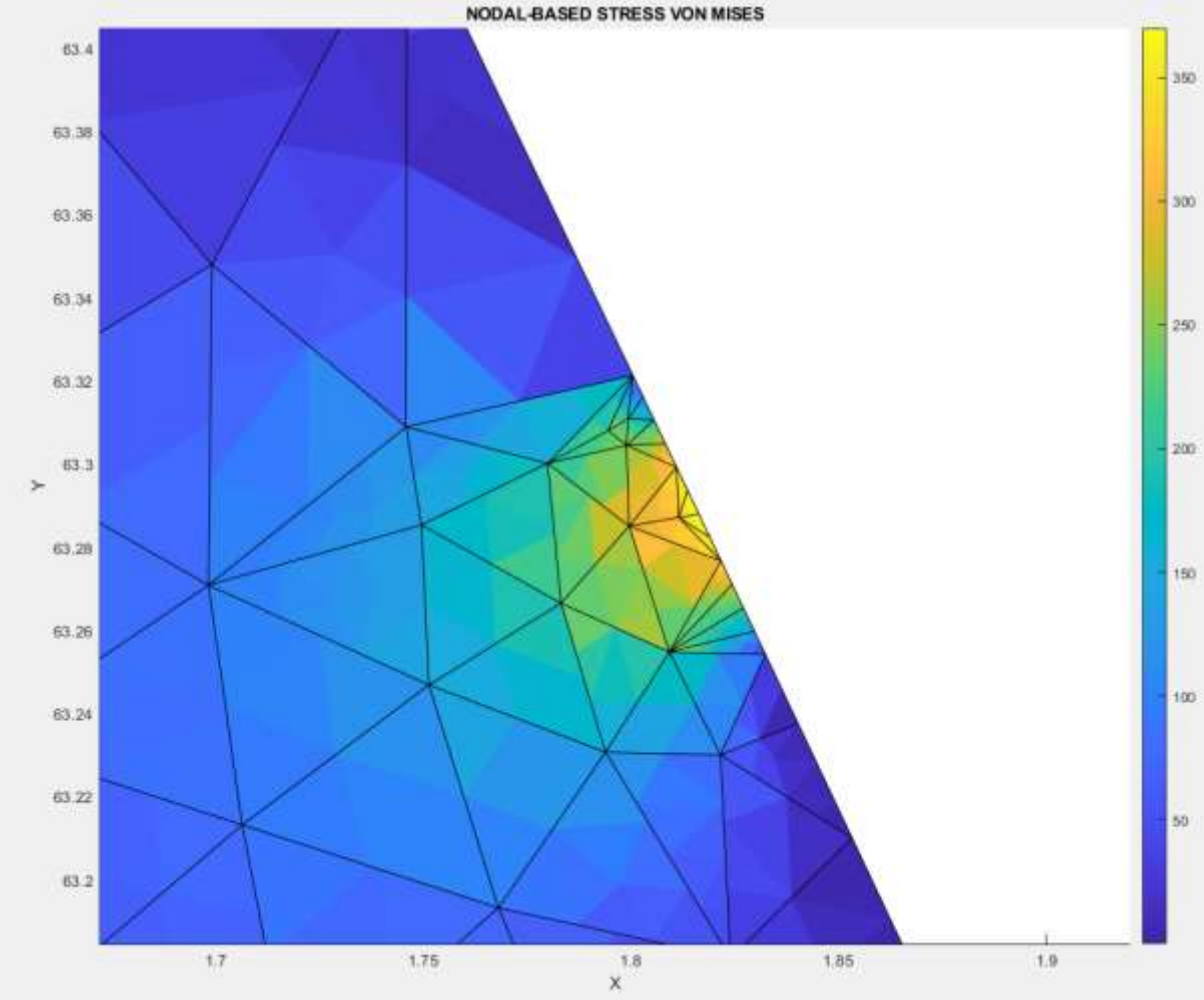
Picture 4.13. Contact Pressure in Position 6 with CST elements.



Picture 4.14. Contact Pressure in Position 6' with CST elements.



Picture 4.15. Contact Pressure in Position 6 with LST elements.



Picture 4.16. Contact Pressure in Position 6' with LST elements.

	Position 6	Position 6'
Contact Pressure ANSYS (MPa)	299.99	367.82
Contact Pressure MATLAB CST (MPa)	309.30	372.41
Contact Pressure MATLAB LST (MPa)	314.74	369.82
Contact Pressure Theoretical (MPa)	319.98	377.32
Error ANSYS(%)	6.25	2.52
Error MATLAB CST (%)	3.34	1.30
Error MATLAB LST (%)	1.64	1.98

Table 4.11. Comparison of Contact Pressure results in Position 6 and Position 6'.

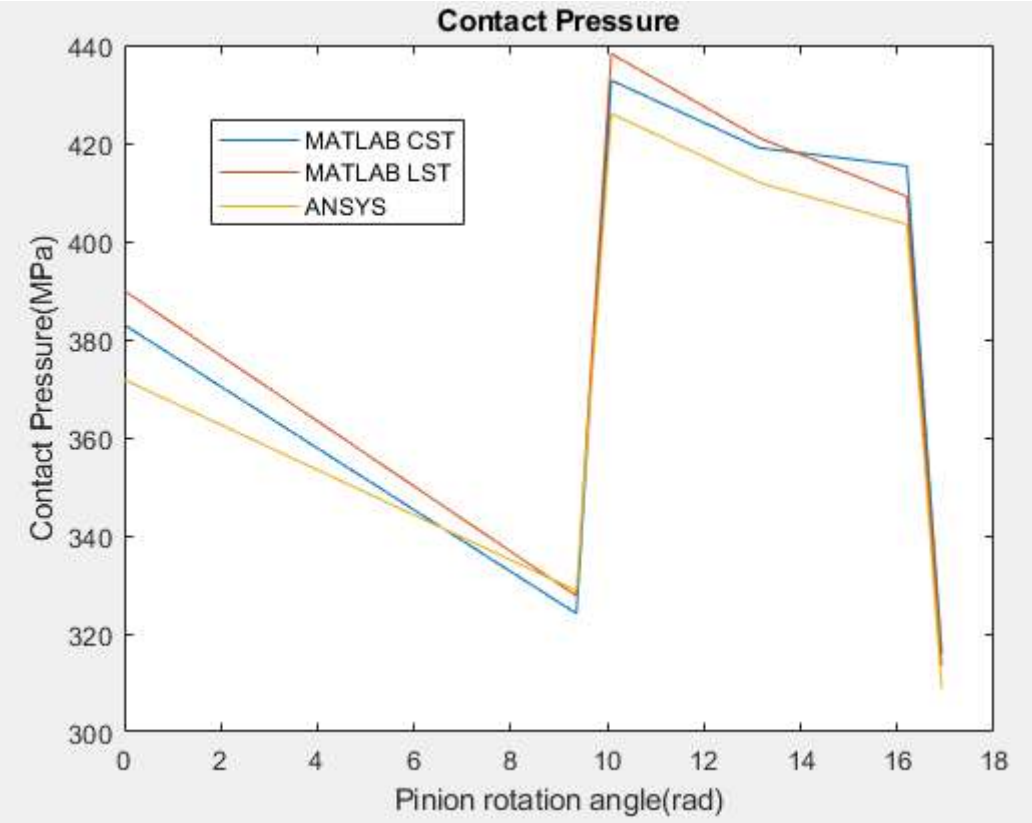
Comparison between ANSYS and MATLAB code in several positions

Apart from the testing of the MATLAB code in Position 6 and Position 6', the simulations with the CST and LST elements also run for the positions presented in Table 4.12. These positions of gear contact run in two-dimensional models in ANSYS too.

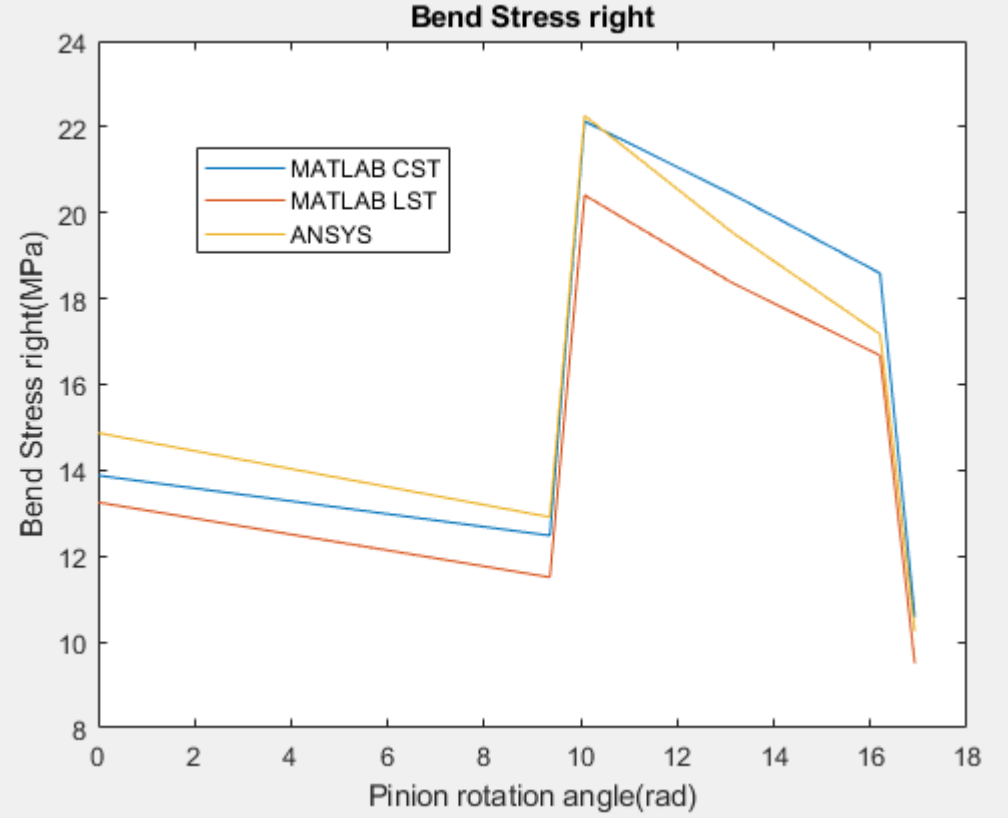
Pinion's rotation angle(deg)	0	9.36	10.08	13.14	16.20	16.92
Radius of conjugate gear(mm)	63.46	60.97	60.80	60.11	59.42	59.28

Table 4.12. Comparing positions for the testing of MATLAB code

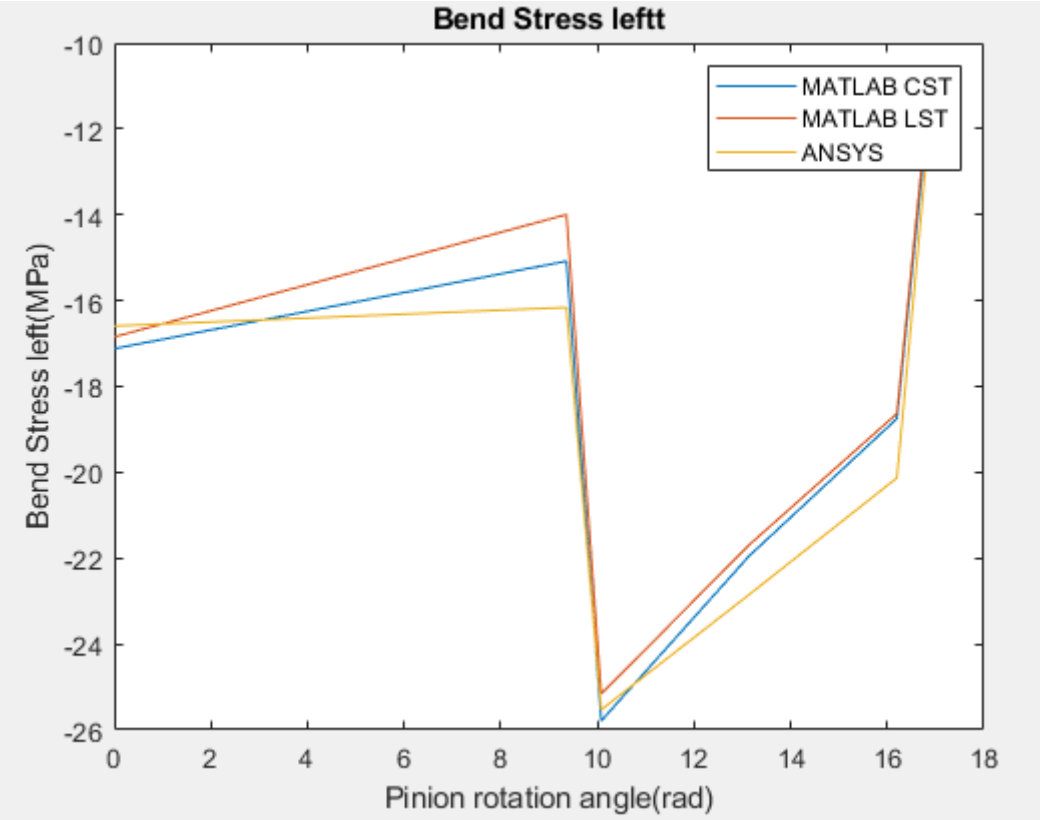
The results of the simulations regarding the contact pressure and the bending stress of conjugate gear are presented in Picture 4.17, Picture 4.18 and Picture 4.19 respectively.



Picture 4.17. Contact pressure in comparing positions.



Picture 4.18. Bending stress in the right side of conjugate gear in comparing positions.



Picture 4.19. Bending stress in the left side of conjugate gear in comparing positions.

5. Conclusion

The goal of this thesis has been the presentation step by step of several methodologies for the modelling of gear contact. In particular, the case that were investigated were the modelling of ideal gear contact in two-dimensional and three-dimensional modelling, the modelling of the misalignment error and finally the modelling of the crowning modification. The simulations of those case were done with the combination of FEM and the Hertz theory of contact and aimed at a fast and efficient modelling. They done both in the software ANSYS and with the development of an original MATALB code.

In the ANSYS modelling all the above cases were examined and explained. It was given emphasis in the CAD treatment of the model especially in the cases of misalignment error and crowning where specific geometric modifications are necessary, to the setting up of the mesh parameters, the setting up of the contact parameters and the application of boundary conditions and of the load. Finally, the results of each case are presented and are compared with the ones yielded from theoretical calculations.

As far as the MATLAB code is concerned, the purpose was the creation of a code that it is easily handled from a user with a basic background in finite element analysis and the code should be fast to save much of the computational time that is required in ANSYS. Both goals were achieved as it was shown in chapter 4 and thus a complete code that models one tooth with a distributed force applied in it with the abilities of two-dimensional modelling with CST elements and LST elements was handed.

All the methodologies of modelling gear contact that are presented in this project aim at making the gears design process more effective both in cost and in product functionality and offer some ideas about modelling of contact phenomena in general which can also be applied in various cases. Some future improvements, to the methods presented in this thesis would be the modelling in ANSYS of a case that combines the misalignment error with the crowing modification to build an effective model that can predict the necessary quantity of crowing for each case and the enrichment of the MATLAB code with the insertion of quadrilateral elements and with the option of automated mesh refinement.

6. References

[1] Daniel S.H. Lo, *Finite Element Mesh Generation*, CRC Press, 2015, pp. 121-134

[2] Shuting Li, *Effects of misalignment error, tooth modifications and transmitted torque on tooth engagements of a pair of spur gears*, Mechanisms and Machine Theory, October 2014, pp. 1-12

[3] Φίλλιπος Φ. Κωστάκης, *Βέλτιστος σχεδιασμός κατατομών μετωπικών οδοντωτών τροχών κλειστής τροχιάς επαφώνγια ισοκατανομή του κινδύνου εμφάνισης εκκοιλάνσεων*, Διπλωματική Εργασία, Σχολή Μηχανολόγων Μηχανικών , Ε.Μ.Π., 2013, σελ. 17-23

[4] Προβατίδης Χ. Γ., *Βελτιστοποίηση & Λογισμικό Κατασκευών: Πεπερασμένα Στοιχεία, Ισογεωμετρικά Στοιχεία, Συνοριακά Στοιχεία*, Εκδόσεις ΤΖΙΟΛΑ, 2017, σελ. 189-198,σελ 297-306, σελ 766-768

[5] Κωστόπουλος Θ., *Οδοντώσεις και μειωτήρες στροφών*, Β' Έκδοση, Εκδόσεις Συμεών, Αθήνα, 2010, σελ. 53

[6]Wikipedia, *Contact Mechanics*

[7] H.L. Whittemore and S.N. Petrenko, Natl. Bur. Std. Tech. Paper 201,1921 in : S.P. Timoshenko and J.N. Goodier, (Eds.) *Theory of Elasticity*, Third Edition, McGRAW-HILL INTERNATIONAL EDITIONS, New York, 1970

Appendix

MATLAB codes for the positioning of gears in ANSYS

```
clc
clear all
format long
%Gear parameters
gear.a0=deg2rad(20);
gear.m=4;
gear.z=30;
gear.width=40;%mm
gear.r0=gear.m*gear.z/2;
gear.rg=cos(gear.a0)*gear.r0;
gear.i=1/1.5;
%gear.contact_r=[82.64 117.92;75.88 126.86]/2;
BC=sqrt((gear.r0+gear.m)^2-gear.r0^2*cos(gear.a0)^2)-gear.r0*sin(gear.a0);
CA=sqrt((gear.i*gear.r0+gear.m)^2-(gear.i*gear.r0)^2*cos(gear.a0)^2)-
gear.i*gear.r0*sin(gear.a0);
e=(BC+CA)/(2*pi*gear.r0*cos(gear.a0)/gear.z);
x_desired=-(CA-(((1+e-1)/2)/e)*(BC+CA))*cos(gear.a0);
y_desired=-(CA-(((1+e-1)/2)/e)*(BC+CA))*sin(gear.a0);
contact_r=[sqrt(x_desired^2+(y_desired-gear.r0)^2)
sqrt(x_desired^2+(y_desired+gear.i*gear.r0)^2)];
gear.contact_r=contact_r;
%Pinion load
M=98;%Nm
load_ratio=[1 0.575 1-0.575];
%Material properties
material.E=[2e5 2e5];%MPa
material.v=[0.3 0.3];
%
b_th=zeros(3,1);
pmax_th=zeros(3,1);
R_eq=zeros(3,1);
R=zeros(3,2);
d=zeros(3,1);
F_tot=zeros(3,1);
[b_th(1),pmax_th(1,:),R_eq(1),R(1,:),d(1),F_tot(1)] =
hertz_calc(M,gear,material);

x_desired2=-(CA-(0.48/e)*(BC+CA))*cos(gear.a0);
y_desired2=-(CA-(0.48/e)*(BC+CA))*sin(gear.a0);
contact_r(2,:)=[sqrt(x_desired2^2+(y_desired2+gear.r0)^2)
sqrt(x_desired2^2+(y_desired2-gear.i*gear.r0)^2)];


x_desired3=-(CA-(1.48/e)*(BC+CA))*cos(gear.a0);
y_desired3=-(CA-(1.48/e)*(BC+CA))*sin(gear.a0);
contact_r(3,:)=[sqrt(x_desired3^2+(y_desired3+gear.r0)^2)
sqrt(x_desired3^2+(y_desired3-gear.i*gear.r0)^2)];

bend_stress=zeros(2,1);
for i=2:3
  rg=[gear.rg gear.i*gear.rg];
  a0=gear.a0;
  L=gear.width;
  E=material.E;
  v=material.v;
  F_tot(i)=load_ratio(i)*F_tot(1);
  a=acos(rg./contact_r(i,:));
```

```
  R(i,:)=sin(a).*contact_r(i,:);
  R_eq(i)=R(i,1)*R(i,2)/(R(i,1)+R(i,2));
  E_eq=1/((1-v(1)^2)/E(1)+(1-v(2)^2)/E(2));
  pmax_th(i)=sqrt(F_tot(i)*E_eq/(pi*L*R_eq(i)));
  b_th(i)=2*F_tot(i)/(pi*L*pmax_th(i));
  d(i)=(2*F_tot(i)*(1-
v(1)^2)/(pi*E(1)*L))*(2/3+log(4*R(i,1)/b_th(i))+log(4*R(i,2)/b_th(i)));
  qk=2.6;
  bend_stress(i-1)=F_tot(i)*cos(a0)*qk/(L*gear.m);
end
function [b_th,pmax_th,R_eq,R,d,F_tot] = hertz_calc(M,gear,material)
%pmax_th is the theoritical maximum surface pressure(MPa)
%b_th is the theoritical halfwidth of contact(mm)
%F is force between gears(N)
rg=[gear.rg gear.i*gear.rg];
contact_r=gear.contact_r;
a0=gear.a0;
L=gear.width;
E=material.E;
v=material.v;
%-------------------------------------------------------------
F(1)=-M*1e3/contact_r(1);
F(2)=tan(a0)*F(1);
F_tot=sqrt(F(1)^2+F(2)^2);
%-------------------------------------------------------------
a=acos(rg./contact_r);
R=sin(a).*contact_r;
R_eq=R(1)*R(2)/(R(1)+R(2));
%-------------------------------------------------------------
E_eq=1/((1-v(1)^2)/E(1)+(1-v(2)^2)/E(2));
pmax_th=sqrt(F_tot*E_eq/(pi*L*R_eq));
b_th=2*F_tot/(pi*L*pmax_th);
d=(2*F_tot*(1-v(1)^2)/(pi*E(1)*L))*(2/3+log(4*R(1)/b_th)+log(4*R(2)/b_th));
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HISTORY of GEARCALC-PRO:
% First draft by C. Provatidis (21.02.2020), for BEM analysis.
% Generalization and gear-tooth mirroring by E. Sakaridis (03.04.2020)
% Introduction of Finite Element Analysis (FEA) by C. Provatidis (15.04.20)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------------------------
% Continued after line #163 by C.Provatidis (13-15.04.2020): Plot & FEA
% Functions for the advancing front method (advancing_front_functions) are
% directly borrowed from GEARCALC-PRO
%-------------------------------------------------------------------------
%                  Runs with MATLAB R2014b and later
%      (Some secondary operations require the PDE-Toolbox)
%-------------------------------------------------------------------------
% This code generates a triangular finite element mesh for an arbitrary
% gear tooth profile and with arbitrary mesh seeds, set by the user
% Then FEA is performed and postprocessing outpouts are plotted.
%-------------------------------------
addpath(genpath('advancing_front_functions'))
addpath(genpath('geometry_functions'))
addpath(genpath('outline_functions'))
addpath(genpath('plotting_functions'))
addpath(genpath('FEA_functions'))           %by C.P. (15.04.2020)
```

```
addpath(genpath('display_functions'))       %by C.P. (15.04.2020)
addpath(genpath('meshGen_functions'))       %by C.P. (25.11.2020)
addpath(genpath('advancing_front_functions'))
%--------------------------------------
clear all
clc
format long
hold on
%--------------------------------------
% no cutting tool displacement allowed!!!
%
% tooth outline descriprion
%      E   D        (A-B) Root circle
%      |---         (B-C) Trochoid
%      |   \C       (C-D) Involute
%      |    \B   A  (D-E) Tip circle
%      |     -- |   (E-F) Straight symmetry line
%      |        |   (F-G) Hub circle
%      |--------|   (G-A) Straight line border with next tooth
%      F         G
%
%% algorithm parameters
tol=10^-8; % universal tolerance for geometry calculations in length units
npoints=1000; % number of points to sample from the tooth flank
%
% basic geometric parameters
gear.z=20; % # of teeth
gear.m=4; % module
gear.a0=20*(pi/180);  %deg2rad(20); % pressure angle [Convert angle from
degrees to radians]
gear.cs=0.4928;%0.492835; % thickness coefficient at rolling circle
gear.ck=1; % addendum coefficient
gear.cf=1.25; % dedendum coefficient
gear.cc=0.3; % rack curvature coefficient
gear.i=1.5;
%
% Mesh size array: mesh seeds
% Define the seed size for the corresponding segments, in length units.
% Two options:
% A) Constant element size: To invoke this option set the second array
% element for each segment to zero, for example [0.1,0]
% B) Linearly changing element size: To invoke this option set the second
% array element for each segment to a positive real. For example using the
% value [0.1,0.2] for segment AB will result in element size 0.1 at A and
% 0.2 at B, whereas [0.2,0.1] gives 0.2 at A and 0.1 at B.
%
%
mesh_seeds = [0.001,0;...       %(A-B)
              0.01,0;...       %(B-C)
              0.01,0;...       %(C-D)
              0.01,0;...     %(D-E)
              0.1,0;...     %(E-F)
              0.01,0;...       %(F-G)
              0.001,0;];...  %(G-A)
%
%% supplementary geometry calculations
[gear.r0,gear.rg,gear.rf,gear.rk,gear.rc,gear.fi,gear.w] = ...
                                                    gear_radii_angles(gear);
%
% calculating the trochoid and involute meeting radius
gear.rs = trochoid_meet_involute(gear,tol);
```

```
%
% assigning the gear hub radius
gear.rh=22.%15%gear.rf-2*gear.m;
%
% sampling the flank points at equidistant y
[gear.flank_points,gear.trochoid_points,gear.involute_points] = ...
                                                    flank_points(gear,npoints);
%
%% extracting the outline points
outline_points=zeros(1,2);
%
% (A-B)
fi_start=pi/2-pi/gear.z;
fi_end=pi/2-gear.w;
r=gear.rf;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(1,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
%
% (B-C)
outline_nodes = outline_nodes_curve(gear,mesh_seeds(2,:),'trochoid',tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
%
% (C-D)
outline_nodes = outline_nodes_curve(gear,mesh_seeds(3,:),'involute',tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
%
% (D-E)
ak=acos(gear.rg/gear.rk); % pressure angle on tip circle
invak=tan(ak)-ak; % involute function
%
fi_start=pi/2-gear.fi+invak;
fi_end=pi/2;
r=gear.rk;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(4,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
%
% (E-F)
point_start=[0,gear.rk];
point_end=[0,gear.rh];
outline_nodes = outline_nodes_line(point_start,point_end,mesh_seeds(5,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];

% (F-G)
fi_start=pi/2;
fi_end=pi/2-pi/gear.z;
r=gear.rh;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(6,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
%
% % (G-A)
% point_start=[gear.rh*cos(pi/2-pi/gear.z),gear.rh*sin(pi/2-pi/gear.z)];
% point_end=[gear.rf*cos(pi/2-pi/gear.z),gear.rf*sin(pi/2-pi/gear.z)];
% outline_nodes =
outline_nodes_line(point_start,point_end,mesh_seeds(7,:));
% outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];
% This is the last segment, so the last node is omitted, as it coincides
% with the first node of the first segment
%gear.contact_r=[82.64 117.92;75.88 126.86]/2;
% a=acos(gear.rg/gear.contact_r(1,2));
% phi=tan(a)-a;
% phi0=tan(gear.a0)-gear.a0;
```

```
% S0=gear.cs*pi*2*gear.r0/gear.z;
% S=gear.contact_r(1,2)*(S0/gear.r0+2*(phi0-phi));
% x=rad2deg(S/(2*gear.contact_r(1,2)));
%
%%
%
BC=sqrt((gear.r0+gear.m)^2-gear.r0^2*cos(gear.a0)^2)-gear.r0*sin(gear.a0);
CA=sqrt((gear.i*gear.r0+gear.m)^2-(gear.i*gear.r0)^2*cos(gear.a0)^2)-
gear.i*gear.r0*sin(gear.a0);
% x_desired2=-(BC-(5*(BC+CA)/16))*cos(gear.a0);
% y_desired2=-(BC-(5*(BC+CA)/16))*sin(gear.a0);
% x_desired=(CA-(0.1/1.6)*(BC+CA))*cos(gear.a0);
% y_desired=(CA-(0.1/1.6)*(BC+CA))*sin(gear.a0);
x_desired=-2.077183160124260;
y_desired=-0.756032841404238;
%
gear.contact_r=sqrt(x_desired^2+(y_desired-gear.r0)^2);
t=sqrt(gear.contact_r^2/gear.rg^2-1);
x=gear.rg*(sin(t)-t.*cos(t));
y=gear.rg*(cos(t)+t.*sin(t));
x1=-(cos(gear.fi)*x-sin(gear.fi)*y);
y1=sin(gear.fi)*x+cos(gear.fi)*y;
angle=atan2(y_desired-gear.r0,x_desired)-atan2(y1,x1);
rad2deg(angle)
t=sqrt((linspace(gear.rg,gear.rk,1e3).^2)/gear.rg^2-1);
x=gear.rg*(sin(t)-t.*cos(t));
y=gear.rg*(cos(t)+t.*sin(t));
x1=-(cos(gear.fi)*x-sin(gear.fi)*y);
y1=sin(gear.fi)*x+cos(gear.fi)*y;
x2=x1*cos(angle)-y1*sin(angle);
y2=x1*sin(angle)+y1*cos(angle);
plot(outline_points(:,1),outline_points(:,2))
plot(x2,y2+gear.r0)
%

gear.z=30;
[gear.r0,gear.rg,gear.rf,gear.rk,gear.rc,gear.fi,gear.w] =
gear_radii_angles(gear);
gear.contact_r=sqrt(x_desired^2+(y_desired+gear.r0)^2);
t=sqrt(gear.contact_r^2/(gear.rg)^2-1);
x=gear.rg*(sin(t)-t.*cos(t));
y=gear.rg*(cos(t)+t.*sin(t));
x1=-(cos(gear.fi)*x-sin(gear.fi)*y);
y1=sin(gear.fi)*x+cos(gear.fi)*y;
angle=atan2(y_desired+gear.r0,x_desired)-atan2(y1,x1);
rad2deg(angle)
t=sqrt((linspace(gear.rg,gear.rk,1e3).^2)/gear.rg^2-1);
x=gear.rg*(sin(t)-t.*cos(t));
y=gear.rg*(cos(t)+t.*sin(t));
x1=-(cos(gear.fi)*x-sin(gear.fi)*y);
y1=sin(gear.fi)*x+cos(gear.fi)*y;
x2=x1*cos(angle)-y1*sin(angle);
y2=x1*sin(angle)+y1*cos(angle);
plot(x2,y2-gear.r0)
plot(x_desired,y_desired,'o')
axis equal
%
%%
x_ansys=outline_nodes(:,1);
y_ansys=outline_nodes(:,2);
z=zeros(size(x_ansys,1),1);
```

```
coordinates=[ones(size(x_ansys,1),1) (1:size(x_ansys,1))' x_ansys y_ansys-
gear.r0 z ];
fileID = fopen('coordinates.txt','w');
for i=1:size(x_ansys,1)
    fprintf(fileID,'%d %d %d %d %d\n',coordinates(i,:));
end
fclose(fileID);
```

MATLAB code and functions for CST element

```
tic
close all
clear all
clc
format long
%% algorithm parameters
tol=10^-8; % universal tolerance for geometry calculations in length units
npoints=100; % number of points to sample from the tooth flank
%
% basic geometric parameters
gear.z=30; % # of teeth
gear.m=4; % module
gear.a0=deg2rad(20); % pressure angle [Convert angle from degrees to
radians]
gear.cs=0.5; % thickness coefficient at rolling circle
gear.ck=1; % addendum coefficient
gear.cf=1.25; % dedendum coefficient
gear.cc=0.38; % rack curvature coefficient
gear.width=40;%gear width
gear.i=1/1.5;%gear ratio
%
%%Material parameters
material.E=[2e5 2e5];%Elastic Modulus of gears(MPa)
material.v=[0.3 0.3];%Poisson coefficient
material.PlaneType=0;% plane-stress (0), plane-strain (=/0)
%
%%Load
F=1.008912416112217e+03;%N
%
%% supplementary geometry calculations
[gear.r0,gear.rg,gear.rf,gear.rk,gear.rc,gear.fi,gear.w] = ...
                                                    gear_radii_angles(gear);
gear.contact_r=[59.280370567072850  40.822387699923972];
%
%%Hertz calculations
[b_th,pmax_th] = hertz_calc(F,gear,material);
b=b_th;
%Theoretical  Bending stress calculation
s_bend_th = bending_stress_th(F,gear);
%
% Mesh size array: mesh seeds
% Define the seed size for the corresponding segments, in length units.
% Two options:
% A) Constant element size: To invoke this option set the second array
% element for each segment to zero, for example [0.1,0]
% B) Linearly changing element size: To invoke this option set the second
% array element for each segment to a positive real. For example using the
% value [0.1,0.2] for segment AB will result in element size 0.1 at A and
% 0.2 at B, whereas [0.2,0.1] gives 0.2 at A and 0.1 at B.
%
%
```

```
mesh_seeds = [0.2,0;...       %(A-B)
              0.2,0;...       %(B-C)
              0.2,b_th/10;...       %(C-D)
              b_th/10,0;...       %(D-E)
              b_th/10,0.2;...     %(E-F)
              0.2,0.5;...       %(F-F')
              0.2,0.5;...     %(F'-C')
              0.2,0;...       %(C'-B')
              0.2,0;...       %(B'-A')
              4,0.2;         %(A'-G')
              5,0;           %(G'-G)
              4,0.2];      %(G-A)
%
% calculating the trochoid and involute meeting radius
gear.rs = trochoid_meet_involute(gear,tol);
%
% assigning the gear hub radius
gear.rh=33;
%
%circle of influence
circle=circle_of_infuence_calculation(gear,b);
%
%% extracting the outline points
[division_points,outline_points,loaded_nodes,fixed_nodes,bending_nodes] =
outline_points(gear,circle,mesh_seeds,tol);
%plot(outline_points(:,1),outline_points(:,2),'-')
%
%% meshing
neleoutline=size(outline_points,1);
[NN,NE,ME,Xnodes,Ynodes] = Advancing_Front_LO_2D...
(outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                        [1:neleoutline,1]);
%
% reshaping the node and element connectivity matrices
nodes=[Xnodes,Ynodes,zeros(size(Xnodes))];
elements=zeros(NE,3);
for i=1:NE
   elements(i,1)=ME(3*(i-1)+1);
   elements(i,2)=ME(3*(i-1)+2);
   elements(i,3)=ME(3*(i-1)+3);
end
% plotting results

%plot_tooth_flank(gear);
%
plot_division_points(outline_points,division_points)
%
plot_outline_points(outline_points)
%
plot_tri_mesh(nodes,elements)%
%
%
%*************************************************************************
%Date: Friday 10.04.2020 (after SKYPE meeting) - 15.04.2020
%-------------------------------------------------------------------------
% CP-addons: continues with node numbering and element numbering
%---Node numbering (in red color):
plot_tri_mesh_nodal_points_numbered(nodes,elements)%
%---Element numbering (in black color, italics):
plot_tri_mesh_elements_numbered(nodes,elements)%
%---Both node and element numbering (red and black, as abovementioned):
```

```
plot_tri_mesh_node_and_element_numbering(nodes,elements)%

%-------------------------------------------------------------------------
%%                            FEM - analysis:                           *
%-------------------------------------------------------------------------
% Memory allocation and some useful nodal variables:
% NELE : total number of elements in the mesh
% NODES: total number of nodes in the mesh
% NDF  : number of DOF per node
% NDM  : dimension of problem (here is 2D-analysis)
% NEL  : number of nodes per element(linear triangle)
% Area : cross-sectional area)
% BC   : index for boundary conditions (0=free, 1=fixed)
% Fext : externally applied forces (initialization)
% U    : nodal displacements (initialization)
    [NELE,NODES,NDF,NDM,NEL,Area,BC,Fext,U] = AllocateMEMORY(NE,nodes);
%-------------------------------------------------------------------------
%Boundary Conditions
% Initially, all DOFs are free (BC=0)
% Then, we determine only the restrained (i.e. fixed) DOFs:
BC = plot_boundary_conditions(BC,nodes,elements,fixed_nodes);
% Externally Applied Forces
% default: all imposed forces, are initially zero.
% Thus, we define only the non-vanishing force components
%Fext = plot_load(Fext,nodes,elements,loaded_nodes,F,gear);
Fext = load_distribution(Fext,loaded_nodes,pmax_th,gear,nodes,circle);
% ENTER Thickness of elements (MANUALLY):
    Thickness(1:NELE,1)=gear.width;       % element thickness
%-------------------------------------------------------------------------

%% Calculate the unknown displacements:
    [U] = FEA(nodes,elements,NODES,NELE,material,BC,Fext,...
              Thickness,U);
%% POST-PROCESSING: Calculate the element strains and stresses:
    [Strain,Stress,PrincipStress,Svm] = Element_Strain_Stress(nodes ...
                                       ,elements,NELE,material,U,Thickness);
    [s_bend,bending_elements_right,bending_elements_left] =
bending_stress(Stress,elements,nodes,bending_nodes,NELE);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
% DISPLAY NODAL DISPLACEMENTS-STRAINS-STRESSES ON SCREEN and REPORT-file: %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENTER the name of Report file and Control the Printout-options:
    fid16 = fopen('REPORT_FEA.txt','w');    %Open file to print the results
    ScreenKey     = 1;  %SCREEN (1=performs printout, 0=cancels printout)
    ReportFileKey = 1;  %REPORT (1=performs printout, 0=cancels printout)
    PlotStressKey = 1;  %PLOTS (1) or cancels plot (0), accordingly.
%---
% Output of displacements:
%     Display_Displacements(NODES,NELE,nodeLoad,U,NDF,...
%                           ScreenKey,ReportFileKey,fid16);
% Output of element strains (as calculated):
    Display_Strains(NELE,Strain,ScreenKey,ReportFileKey,fid16);
% Output of element stresses (as calculated):

Display_Stresses(NELE,Stress,PrincipStress,ScreenKey,ReportFileKey,fid16)
%
%
%*************************************************************************
```

```
% %-------------------------------------------------------------------------
--
%% COLOR FILL and Plot (ELEMENT STRESSES)
% Sxx stress component:
    plotSxx(NELE,nodes,elements,Stress);
% Syy stress component:
    plotSyy(NELE,nodes,elements,Stress);
% Sxy=Txy stress component:
    plotSxy(NELE,nodes,elements,Stress);
% Von-Mises element stress:
% % %     titleChar = 'ELEMENT-BASED VON-MISES STRESS Svm';
    plot_VonMises(NELE,nodes,elements,Svm);
% S11 Principal element stress: PrincipStress(i,1:2)
    plotS11(NELE,nodes,elements,PrincipStress);
% S22 Principal element stress: PrincipStress(i,1:2)
    plotS22(NELE,nodes,elements,PrincipStress);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%% NODAL SMOOTHED STRESS COMPONENTS (Svm_nodal is in excess):
    [StressNodal,Svm_nodal] = SmoothNodalStress(NODES,NELE,elements, ...
                                Stress,Svm,PrincipStress);

%% PRINT NODAL STRESSES (Sxx, Syy, Sxy, Svm, S11, S22):
    fprintf('*** STRESSES ***\n');
    for i=1:NODES
        fprintf('node=%3i  Sxx=%12.5e  Syy=%12.5e
Sxy=%12.5e\n',i,StressNodal(3*i-2),StressNodal(3*i-1),StressNodal(3*i-0));
    end
    fprintf('
**********************************************************************\n');

%% RECALCULATE SMOOTHED ELEMENT STRESS COMPONENTS (Sxx,Syy,Sxy,Svm,S11,S22)
%   (based on smoothed nodal values at element centroids).
    [SmoothElementStress] = Smooth_Element_Stress(NELE,elements, ...
                                                  StressNodal);
% Plot smoothed von Mises stresses:
    plot_Smooth_VonMises(NELE,nodes,elements,StressNodal)
%---Check it!
    titleChar = '%%% ALTERNATIVE %%%';
    StressSmooth(1:NELE) = SmoothElementStress(1:NELE,4);
    plot_General_Smooth_Stress(NELE,nodes,elements,StressSmooth,titleChar)
%
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%% trisuf :
% Plot the FEM approximation U(x,y) with values U_1 to U_N at the nodes
clear t
t = elements;
p = nodes(:,1:2);
Ux=U(1:2:end);  %horizontal displacement
Uy=U(2:2:end);  %vertical displacement
Ures=sqrt(Ux.^2+Uy.^2);
trisurf(t,p(:,1),p(:,2),0*p(:,1),Ures,'edgecolor','k','facecolor','interp')
;
view(2),axis equal,colorbar
% view(2),axis([-1 1 -1 1]),axis equal,colorbar
%-------------------------------------------------------------------------
toc
```

```
function [r0,rg,rf,rk,rc,fi,w] = gear_radii_angles(gear)
%
% extracting data from the structure
z=gear.z;
m=gear.m;
a0=gear.a0;
cs=gear.cs;
ck=gear.ck;
cf=gear.cf;
cc=gear.cc;
%
% calculating radii and angles
r0=z*m/2; % rolling circle
rg=r0*cos(a0); % base circle
rf=r0-cf*m; % root circle
rk=r0+ck*m; % tip circle
rc=cc*m; % rack tip curvature
fi=pi*cs/z+tan(a0)-a0; % involute rotation angle
w=(pi*cs+2*(cf-cc)*tan(a0)+2*cc/cos(a0))/z; % trochoid rotation angle
end


function [b_th,pmax_th] = hertz_calc(F_tot,gear,material)
%pmax_th is the theoritical maximum surface pressure(MPa)
%b_th is the theoritical halfwidth of contact(mm)
%F is force between gears(N)
rg=[gear.rg gear.i*gear.rg];
contact_r=gear.contact_r;
L=gear.width;
E=material.E;
v=material.v;
%-------------------------------------------------------------
a=acos(rg./contact_r);
R=sin(a).*contact_r;
R_eq=R(1)*R(2)/(R(1)+R(2));
%-------------------------------------------------------------
E_eq=1/((1-v(1)^2)/E(1)+(1-v(2)^2)/E(2));
pmax_th=sqrt(F_tot*E_eq/(pi*L*R_eq));
b_th=2*F_tot/(pi*L*pmax_th);
end


function s_bend_th = bending_stress_th(F,gear)
m=gear.m;
l=gear.width;
a0=gear.a0;
qk=2.6;
s_bend_th=F*cos(a0)*qk/(m*l);
end


function [rs] = trochoid_meet_involute(gear,tol)
%
% extracting data from the struct
r0=gear.r0; % rolling circle
rg=gear.rg; % base circle
rf=gear.rf; % root circle
rk=gear.rk; % tip circle
rc=gear.rc; % rack tip curvature
fi=gear.fi; % involute rotation angle
w=gear.w; % trochoid rotation angle
```

```
%
% suplementary geometry calculations
%
% calculating ymin from trochoid
ymin=rf*cos(w);
%
% calculating ymax from involute
t_ymax=sqrt(rk^2/rg^2-1);
x_ymax=rg*(sin(t_ymax)-t_ymax*cos(t_ymax));
y_ymax=rg*(cos(t_ymax)+t_ymax*sin(t_ymax));
ymax=sin(fi)*x_ymax+cos(fi)*y_ymax;
%
% calculating the minimum y on which the involute is defined (rg)
yinvmin=rg*cos(fi);
%
% bisection related calculations
iter=ceil(log(ymax-ymin)-log(tol))/log(2);
%
for i=1:iter
    yrs=(ymax+ymin)/2;
    %
    if yrs<yinvmin % case 1: involute is not defined at yrs, yrs is
trochoid
        ymin=yrs;
    else % case 2: involute is defined at yrs, comparison needed
        xrs_tr=trochoid_y(r0,rk,rf,rc,w,yrs);
        xrs_inv=involute_y(rg,rk,fi,yrs);
        if xrs_tr<xrs_inv % case 2a: trochoid inside involute, yrs is
trochoid
            ymin=yrs;
        else % case 2b: trochoid outside involute, comparison needed
            ytrial=yrs+tol/10; % trial point at slightly higher y than yrs;
            % used to detect whether trochoid approaches or is moving away
            % from involute
            xtrial_tr=trochoid_y(r0,rk,rf,rc,w,ytrial);
            xtrial_inv=involute_y(rg,rk,fi,ytrial);
            if xtrial_tr-xtrial_inv < xrs_tr-xrs_inv
            % case 2b1: trochoid is moving towards involute for increasing
y
            % yrs is trochoid
                ymin=yrs;
            else
            % case 2b2: trochoid is moving away from involute for
increasing y
            % yrs is involute
                ymax=yrs;
            end
        end
    end
end
xrs=involute_y(rg,rk,fi,yrs); % it does not matter if involute or trochoid
is used
rs=sqrt(xrs^2+yrs^2);
end
```

```
function [ x ] = involute_y( rg,rk,fi,y )
%
ymin=rg*cos(fi);
ymax=rk;
if ( y<ymin )
    disp('Error in involute_y');
    x=0;
    return
elseif ( y>ymax ) %shifting the bisection limit
    ymax=ymax+2*(y-ymax);
end
%
tmin=0;
tmax=sqrt(ymax^2/rg^2-1); %this gives radius==rk==ymax
for i=1:100 %set number of biscections here
    t=(tmin+tmax)/2;
    %
    xinv=rg*(sin(fi-t)+t*cos(fi-t));
    yinv=rg*(cos(fi-t)-t*sin(fi-t));
    if (yinv>y)
        tmax=t;
    else
        tmin=t;
    end
end
%
x=xinv;
%
end


function [ x ] = trochoid_y( r0,rk,rf,rc,w,y )
%
B=r0-rf-rc;
%
ymin=rf*cos(w);
ymax=rk;
if ( y<ymin || y>ymax )
    disp('Error in trochoid_y');
    x=0;
    return
end
%
tmin=0;
tmax=sqrt(rk^2-rc^2-(rf+rc)^2)/r0; %this gives radius>rk
for i=1:100 %set number of biscections here
    t=(tmin+tmax)/2;
    %
    A=atan( (-t*r0*sin(t)+B*cos(t)) / (B*sin(t)+t*r0*cos(t)) );
    %
    xtr=-rc*cos(A-w)+(r0-B)*sin(t+w)-r0*t*cos(t+w);
    ytr=-rc*sin(A-w)+(r0-B)*cos(t+w)+r0*t*sin(t+w);
    if (ytr>y)
        tmax=t;
    else
        tmin=t;
    end
end
%
x=xtr;
%
```

```
end


function circle = circle_of_infuence_calculation(gear,b)
r0=gear.r0;
rk=gear.rk;
rg=gear.rg;
fi=gear.fi;
contact_r=gear.contact_r(1);

circle.r=b;

t=sqrt(contact_r^2/rg^2-1);
x=rg*(sin(t)-t.*cos(t));
y=rg*(cos(t)+t.*sin(t));
x1=-(cos(fi)*x-sin(fi)*y);
y1=sin(fi)*x+cos(fi)*y;
circle.C=[x1 y1];
%upper point
a=contact_r;
b=rk;
r=(a+b)/2;
t=sqrt(r^2/rg^2-1);
x=rg*(sin(t)-t.*cos(t));
y=rg*(cos(t)+t.*sin(t));
x1=-(cos(fi)*x-sin(fi)*y);
y1=sin(fi)*x+cos(fi)*y;
error=1e-7;
l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
e=abs(l-circle.r)/circle.r;
while e>error
    if l>circle.r
        b=r;
        r=(a+b)/2;
        t=sqrt(r^2/rg^2-1);
        x=rg*(sin(t)-t.*cos(t));
        y=rg*(cos(t)+t.*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
        error=1e-7;
        l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
        e=abs(l-circle.r)/circle.r;
    else
        a=r;
        r=(a+b)/2;
        t=sqrt(r^2/rg^2-1);
        x=rg*(sin(t)-t.*cos(t));
        y=rg*(cos(t)+t.*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
        error=1e-7;
        l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
        e=abs(l-circle.r)/circle.r;
    end
end
circle.cross_points(1,:)=[x1 y1 r];
%lower point
a=rg;
b=contact_r;
r=(a+b)/2;
```

```
t=sqrt(r^2/rg^2-1);
x=rg*(sin(t)-t.*cos(t));
y=rg*(cos(t)+t.*sin(t));
x1=-(cos(fi)*x-sin(fi)*y);
y1=sin(fi)*x+cos(fi)*y;
error=1e-7;
l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
e=abs(l-circle.r)/circle.r;
while e>error
    if l>circle.r
        a=r;
        r=(a+b)/2;
        t=sqrt(r^2/rg^2-1);
        x=rg*(sin(t)-t.*cos(t));
        y=rg*(cos(t)+t.*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
        error=1e-7;
        l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
        e=abs(l-circle.r)/circle.r;
    else
        b=r;
        r=(a+b)/2;
        t=sqrt(r^2/rg^2-1);
        x=rg*(sin(t)-t.*cos(t));
        y=rg*(cos(t)+t.*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
        error=1e-7;
        l=sqrt((x1-circle.C(1))^2+(y1-circle.C(2))^2);
        e=abs(l-circle.r)/circle.r;
    end
end
circle.cross_points(2,:)=[x1 y1 r];
end
```

```
function
[division_points,outline_points,loaded_nodes,fixed_nodes,bending_nodes] =
outline_points(gear,circle,mesh_seeds,tol)
outline_points=zeros(1,2);
%
% (A-B)
fi_start=pi/2-pi/gear.z;
fi_end=pi/2-gear.w;
r=gear.rf;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(1,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=outline_points([1 end],:);
%
% (B-C)
outline_nodes = outline_nodes_curve_trochoid(0,1,gear,mesh_seeds(2,:),tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
bending_nodes=size(outline_points,1)-
size(outline_nodes,1)+1:size(outline_points,1);
%
% (C-D)
```

```
outline_nodes =
outline_nodes_curve_involute(gear.rs,circle.cross_points(2,3),gear,mesh_see
ds(3,:),tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
%
% (D-E)
outline_nodes =
outline_nodes_curve_involute(circle.cross_points(2,3),circle.cross_points(1
,3),gear,mesh_seeds(4,:),tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
loaded_nodes=size(outline_points,1)-
size(outline_nodes,1)+1:size(outline_points,1);
division_points=[division_points;outline_points(end,:)];
%
% (E-F)
outline_nodes =
outline_nodes_curve_involute(circle.cross_points(1,3),gear.rk,gear,mesh_see
ds(5,:),tol);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
%
% (F-F')
ak=acos(gear.rg/gear.rk); % pressure angle on tip circle
invak=tan(ak)-ak; % involute function
%
fi_start=pi/2-gear.fi+invak;
fi_end=pi/2+gear.fi-invak;
r=gear.rk;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(6,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
%
% (F'-C')
outline_nodes =
outline_nodes_curve_involute(gear.rs,gear.rk,gear,mesh_seeds(7,:),tol);
outline_nodes(:,1)=-outline_nodes(:,1);
outline_nodes=flip(outline_nodes);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
%
% (C'-B')
outline_nodes = outline_nodes_curve_trochoid(0,1,gear,mesh_seeds(8,:),tol);
outline_nodes(:,1)=-outline_nodes(:,1);
outline_nodes=flip(outline_nodes);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
bending_nodes(2,:)=size(outline_points,1)-
size(outline_nodes,1)+1:size(outline_points,1);
%
% (B'-A')
fi_start=pi/2-pi/gear.z;
fi_end=pi/2-gear.w;
r=gear.rf;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(9,:));
outline_nodes(:,1)=-outline_nodes(:,1);
outline_nodes=flip(outline_nodes);
outline_points=[outline_points(1:end-1,:);outline_nodes];
division_points=[division_points;outline_points(end,:)];
%
% (A'-G')
```

```
point_start=[gear.rh*cos(pi/2-pi/gear.z),gear.rh*sin(pi/2-pi/gear.z)];
point_end=[gear.rf*cos(pi/2-pi/gear.z),gear.rf*sin(pi/2-pi/gear.z)];
outline_nodes = outline_nodes_line(point_start,point_end,mesh_seeds(10,:));
outline_nodes(:,1)=-outline_nodes(:,1);
outline_nodes=flip(outline_nodes);
outline_points=[outline_points(1:end-1,:);outline_nodes];
fixed_nodes=size(outline_points,1)-
size(outline_nodes,1)+1:size(outline_points,1);
division_points=[division_points;outline_points(end,:)];
%
% (G'-G)
fi_start=pi/2+pi/gear.z;
fi_end=pi/2-pi/gear.z;
r=gear.rh;
outline_nodes = outline_nodes_circle(r,fi_start,fi_end,mesh_seeds(11,:));
outline_points=[outline_points(1:end-1,:);outline_nodes];
fixed_nodes=[fixed_nodes(1:end-1) size(outline_points,1)-
size(outline_nodes,1)+1:size(outline_points,1)];
division_points=[division_points;outline_points(end,:)];
%
% (G-A)
point_start=[gear.rh*cos(pi/2-pi/gear.z),gear.rh*sin(pi/2-pi/gear.z)];
point_end=[gear.rf*cos(pi/2-pi/gear.z),gear.rf*sin(pi/2-pi/gear.z)];
outline_nodes = outline_nodes_line(point_start,point_end,mesh_seeds(12,:));
outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];
fixed_nodes=[fixed_nodes(1:end-1) size(outline_points,1)-
size(outline_nodes,1)+2:size(outline_points,1) 1];
end


function [outline_nodes] =
outline_nodes_circle(r,fi_start,fi_end,ele_lengths)
%
arc_length=r*abs(fi_start-fi_end);
unitary_distribution=unitary_length_distribution(arc_length,ele_lengths);
nnodes=length(unitary_distribution);
%fis=fi_start:(fi_end-fi_start)/(nnodes-1):fi_end;
fis=zeros(nnodes,1);
for i=1:nnodes
    fis(i)=(fi_end-fi_start)*unitary_distribution(i)+fi_start;
end
outline_nodes=zeros(nnodes,2);
for i=1:nnodes
    outline_nodes(i,1)=r*cos(fis(i));
    outline_nodes(i,2)=r*sin(fis(i));
end
end



function outline_nodes =
outline_nodes_curve_trochoid(divmin,divmax,gear,ele_lengths,tol)
%
% extracting data from the struct
r0=gear.r0;
rg=gear.rg; % base circle
rf=gear.rf; % root circle
rk=gear.rk; % tip circle
rc=gear.rc; % rack tip curvature
rs=gear.rs; % involute - trochoid meeting circle
fi=gear.fi; % involute rotation angle
```

```
w=gear.w; % trochoid rotation angle
%
npoints=1e2;
curve_points=trochoid_points(divmin,divmax,gear,npoints);
%
ymin_curve=rf*cos(w); % ymin at rf
%
t_ys=sqrt(rs^2/rg^2-1);
x_ys=rg*(sin(t_ys)-t_ys*cos(t_ys));
y_ys=rg*(cos(t_ys)+t_ys*sin(t_ys));
ymax_curve=sin(fi)*x_ys+cos(fi)*y_ys; % ymax at rs
ymin=ymin_curve;
ymax=ymax_curve;
ymin_curve=(1-divmin)*ymin+divmin*ymax;
ymax_curve=(1-divmax)*ymin+divmax*ymax;
%
% calculating an approximate curve length by summing all line segments
% between the sampled curve points
curve_dxs=curve_points(2:end,1)-curve_points(1:end-1,1);
curve_dys=curve_points(2:end,2)-curve_points(1:end-1,2);
%
curve_length=sum(sqrt(curve_dxs.^2+curve_dys.^2));
%
%
unitary_distribution =
unitary_length_distribution(curve_length,ele_lengths);
nnodes=length(unitary_distribution);
outline_nodes=zeros(nnodes,2);
outline_nodes(1,:)=[trochoid_y(r0,rk,rf,rc,w,ymin_curve),ymin_curve];
%
% The curve length calculated so far is approximate and using this length
% to distribute the nodes along the curve may result in the last node not
% being coincident with the last curve point. Thus, the nodes are
% distributed along the curve preserving the unitary distribution given
% above but not the curve length. This will result in elements with
% slightly different sizes than the ones specified.
%
curve_length_min=0.5*curve_length; % initial boundaries for bisection
curve_length_max=1.5*curve_length;
%
iter_out=ceil(log(curve_length_max-curve_length_min)-log(tol))/log(2);
interval_extension=1.1;
iter_in=ceil(log(interval_extension*(ymax_curve-ymin_curve))-
log(tol))/log(2);
%
for iout=1:iter_out
    curve_length_trial=(curve_length_min+curve_length_max)/2;
    point_distances=curve_length_trial*(unitary_distribution(2:end)-
unitary_distribution(1:end-1));
    for i=2:nnodes
        ymin=outline_nodes(i-1,2); % lower bound is set by the previous
point
        ymax=ymin_curve+(ymax_curve-ymin_curve)*interval_extension;
        % higher than ymax, so that overshoot above ymax is possible
        for iin=1:iter_in
            ytrial=(ymin+ymax)/2;
            xtrial=trochoid_y(r0,rk,rf,rc,w,ytrial);
            dist=sqrt((xtrial-outline_nodes(i-1,1))^2+(ytrial-
outline_nodes(i-1,2))^2);
            if dist>point_distances(i-1)
                ymax=ytrial;
```

```
            else
                ymin=ytrial;
            end
        end
        outline_nodes(i,:)=[xtrial,ytrial];
        if ytrial>ymax_curve % overshoot has occured
            curve_length_max=curve_length_trial;
            break % there is no need to calculate further points, they will
just overshoot more
        elseif i==nnodes % all points have been completed without overshoot
            curve_length_min=curve_length_trial;
        end
    end
    outline_nodes(:,2);
end
end


function outline_nodes =
outline_nodes_curve_involute(rmin,rmax,gear,ele_lengths,tol)
%
% extracting data from the struct
rg=gear.rg; % base circle
rk=gear.rk; % tip circle
fi=gear.fi; % involute rotation angle
%
npoints=1e2;
curve_points=involute_points(rmin,rmax,gear,npoints);
%
t_ymin=sqrt(rmin^2/rg^2-1);
x_ymin=rg*(sin(t_ymin)-t_ymin*cos(t_ymin));
y_ymin=rg*(cos(t_ymin)+t_ymin*sin(t_ymin));
ymin_curve=sin(fi)*x_ymin+cos(fi)*y_ymin; % ymin at rmin
%
t_ymax=sqrt(rmax^2/rg^2-1);
x_ymax=rg*(sin(t_ymax)-t_ymax*cos(t_ymax));
y_ymax=rg*(cos(t_ymax)+t_ymax*sin(t_ymax));
ymax_curve=sin(fi)*x_ymax+cos(fi)*y_ymax; % ymax at rmax
%
% calculating an approximate curve length by summing all line segments
% between the sampled curve points
curve_dxs=curve_points(2:end,1)-curve_points(1:end-1,1);
curve_dys=curve_points(2:end,2)-curve_points(1:end-1,2);
%
curve_length=sum(sqrt(curve_dxs.^2+curve_dys.^2));
%
%
unitary_distribution =
unitary_length_distribution(curve_length,ele_lengths);
nnodes=length(unitary_distribution);
outline_nodes=zeros(nnodes,2);
outline_nodes(1,:)=[involute_y(rg,rk,fi,ymin_curve),ymin_curve];
%
% The curve length calculated so far is approximate and using this length
% to distribute the nodes along the curve may result in the last node not
% being coincident with the last curve point. Thus, the nodes are
% distributed along the curve preserving the unitary distribution given
% above but not the curve length. This will result in elements with
% slightly different sizes than the ones specified.
%
curve_length_min=0.5*curve_length; % initial boundaries for bisection
```

```
curve_length_max=1.5*curve_length;
%
iter_out=ceil(log(curve_length_max-curve_length_min)-log(tol))/log(2);
interval_extension=1.1;
iter_in=ceil(log(interval_extension*(ymax_curve-ymin_curve))-
log(tol))/log(2);
%
for iout=1:iter_out
    curve_length_trial=(curve_length_min+curve_length_max)/2;
    point_distances=curve_length_trial*(unitary_distribution(2:end)-
unitary_distribution(1:end-1));
    for i=2:nnodes
        ymin=outline_nodes(i-1,2); % lower bound is set by the previous
point
        ymax=ymin_curve+(ymax_curve-ymin_curve)*interval_extension;
        % higher than ymax, so that overshoot above ymax is possible
        for iin=1:iter_in
            ytrial=(ymin+ymax)/2;
            xtrial=involute_y(rg,rk,fi,ytrial);
            dist=sqrt((xtrial-outline_nodes(i-1,1))^2+(ytrial-
outline_nodes(i-1,2))^2);
            if dist>point_distances(i-1)
                ymax=ytrial;
            else
                ymin=ytrial;
            end
        end
        outline_nodes(i,:)=[xtrial,ytrial];
        if ytrial>ymax_curve % overshoot has occured
            curve_length_max=curve_length_trial;
            break % there is no need to calculate further points, they will
just overshoot more
        elseif i==nnodes % all points have been completed without overshoot
            curve_length_min=curve_length_trial;
        end
    end
    outline_nodes(:,2);
end
end


function [outline_nodes] =
outline_nodes_line(start_point,end_point,ele_lengths)
%
line_vector=end_point-start_point;
line_length=sqrt(line_vector(1)^2+line_vector(2)^2);
%line_vector=line_vector/line_length;
%
unitary_distribution=unitary_length_distribution(line_length,ele_lengths);
nnodes=length(unitary_distribution);
outline_nodes=zeros(nnodes,2);
for i=1:nnodes
    outline_nodes(i,:)=start_point+unitary_distribution(i)*line_vector;
end
```

```
function [distribution] = unitary_length_distribution (length,ele_lengths)
%
if ele_lengths(2) == 0 % Case 1: No bias
    nele=max([round(length/ele_lengths(1)),1]); % a minimum of 1 element
    distribution=0:1/nele:1; % nele is number of intervals, not points
else % Case 2: Bias
    nele=2*length/sum(ele_lengths);
    if nele>2 % Case 2a: at least 2 elements fit as requested
        ele_lengths=nele*ele_lengths/round(nele); % scale element lengths
so that they fit exactly
        nele=round(nele);
        ele_length_distr=ele_lengths(1):(ele_lengths(2)-
ele_lengths(1))/(nele-1):ele_lengths(2);
        %
        % culminativery summing dimensionless element lengths
        distribution=zeros(1,nele+1);
        for i=1:nele
            distribution(i+1)=distribution(i)+ele_length_distr(i)/length;
        end
        distribution(nele+1)=1; % avoiding rounding errors
    else % Case 2b: 2 elements do not fit as requested
        if 2*min(ele_lengths)>length % Case 2b1: Small element does not fit
twice
            distribution=0:1; % assign single element
        else % Case 2b2: Small element fits more than twice
            distribution=0:0.5:1; % 2 element uniform distribution
        end
    end
end


%% This program is a MATLAB translaton of the FORTRAN 90 code cited
% in the CRC book (pp. 606-612, 2015) by Lo. It implements Lo(1992) code:
%-------------------------------------------------------------------------%
% S. H. LO, GENERATION OF HIGH-QUALITY GRADATION FINITE ELEMENT MESH, ...
%         Engineering Fracture Mechanics Vol. 41, No. 2, pp. 191-202, 1992.
%-------------------------------------------------------------------------%
% MEMO: Due to many GOTO-commands, a lot of corresponding break were made.
% Final code (15/5/2019)
%-------------------------------------------------------------------------%
% A.12 LIST OF FORTRAN PROGRAM ADF2D
% ADF2D is a mesh generation program of 2D triangular meshes based on the
% ADF approach (Lo 1992). The input is a list of boundary segments
% {MA(I), MB(I), I = 1,NB} following the convention, as depicted in
% Section 3.6.2. This is the final boundary for mesh generation by
% ADF2D, and in case boundary nodes need to be added by some boundary
% discretisation process, it should be done by a separate procedure earlier
% beforehand. The output is a mesh of triangular elements
% {ME(I), I = 1,3*NE}, where NE = number of elements. The element size
% is computed based on the line segments on the boundary, as described
% in Section 3.5.6.3.
%---To cope with a general node spacing function, the part related to the
% generation of interior nodes ought to be modified such that the height
% of the triangle to be created is no longer related to the boundary
% segments but computed from the specified node spacing function.
%--- An example of mesh generation is shown in Figure A.6. In this example,
% the four edges the triangular mesh is improved to 0.96120, as shown in
% Figure A.6b. Exercise: Following Section 3.6.3, introduce the background
% grid to speed up the mesh generation process.
%-------------------------------------------------------------------------%
%%
```

```
% Section 3.6.3, introduce the background grid to speed up the mesh
generation process.
%-------------------------------------------------------------------------
function [NN,NE,ME,X,Y] = Advancing_Front_LO_2D(X,Y,NB,MA,MB)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %...Close-the-loop, to make easier the computation of element lengths:
%     xb(1:nodesB)=X(1:nodesB); yb(1:nodesB)=Y(1:nodesB);
%     xb(nodesB+1)=xb(1); yb(nodesB+1)=yb(1);
%     plot(X,Y,'bo')
%     for i=1:nodesB
%        text(xb(i)+ds,yb(i)+ds,int2str(i),'Color','r')
%     end
%     axis equal
%     pause(1)
%%
    nodesB = NB;
    neleB = nodesB;
    for i=1:neleB-1
        MA(i)=i;
        MB(i)=i+1;
    end
    MA(neleB)=nodesB;
    MB(neleB)=1;
%
NB = nodesB;

% ! ADF MESHING ON PLANAR DOMAINS WITH ELEMENT SIZE BASED ON BOUNDARY
% SEGMENTS
% ! INPUT: COORDINATES OF NODAL POINTS, {X(I), Y(I), I=1,NN}
% ! BOUNDARY SEGMENTS, {MA(I),MB(I),I=1,NB}
% ! OUTPUT: TRIANGULAR ELEMENTS, {ME(I), I=1,3*NE}
% ! WORKING ARRAY: {XP(I),YP(I),DP(I), I=1,5000} MID-POINTS AND LENGTHS OF
% SEGMENTS
% ! {MA(I),MB(I),I=1,1000} CANDIDATE SEGMENTS IN ELEMENT CONSTRUCTION
% % % DIMENSION X(*),Y(*),ME(*),MA(*),MB(*)
% % % DIMENSION MS(1000),MT(1000),XP(5000),YP(5000),DP(5000)
% ! COMPUTE THE LENGTHS OF THE BOUNDARY SEGMENTS AND THEIR MID-POINTS
% fprintf('AFT2D: NB=%3i\n',NB);
%% My action: MY INITIALIZATION>>>>>>>>>>
    NN=NB; NE=0; ME = []; II=0;          %
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
    NP=NB;
    for I=1:NB
        IA=MA(I);
        IB=MB(I);
        XP(I)=(X(IA)+X(IB))/2;
        YP(I)=(Y(IA)+Y(IB))/2;
        DP(I)=(X(IB)-X(IA))^2+(Y(IB)-Y(IA))^2;
    end
% PREPARATION WORKS FOR THE BASE SEGMENT, J1-J2 = LAST SEGMENT ON THE FRONT
    NELnumber = 0;  % current element (ascending) number
    MaxNoElement = 10000;  %maximum expected number of elements
for i5=1:MaxNoElement  %dummy index (OUTER LOOP: Command 5 in FORTRAN)
%
fprintf('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$\n');
%
fprintf('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$\n');
%     fprintf('---EXTERNAL ITERATION i5 =%5i\n',i5);
```

```
    J3=0;
    J1=MA(NB);
    J2=MB(NB);
    NB=NB-1;    %here we reduce the boundary by-one
    X1=X(J1);
    Y1=Y(J1);
    X2=X(J2);
    Y2=Y(J2);
    A=Y1-Y2;
    B=X2-X1;
    DD=A*A+B*B;
    TOR=DD/100;
    XM=(X1+X2)/2;
    YM=(Y1+Y2)/2;
    RR=1.25*DD+TOR;
    XC=XM+A;
    YC=YM+B;
%     lc1=sqrt((XC-X1)^2+(YC-Y1)^2);  %length of produced side CA=C1
%     lc2=sqrt((XC-X2)^2+(YC-Y2)^2);  %length of produced side CB=C2
%     fprintf('lc1=%8.4f lc2=%8.4f\n',lc1,lc2);
% % %         hold on
% % %         plot(XC,YC,'m+')
    C=X2*Y1-X1*Y2+TOR;
% FILTER OFF SEGMENTS TOO FAR AWAY FROM THE BASE SEGMENT
 for i9=1:5    %dummy  (INNER LOOP: Command 9 in FORTRAN). Usually i9=1.
%     fprintf('---INTERNAL ITERATION i9 =%5i\n',i9);
    NS=0;
    for I=1:NB    %do 11
        IA=MA(I);
        IB=MB(I);
        if (~(DPL(X(IA),Y(IA),X(IB),Y(IB),XC,YC) > RR)) %GOTO 11
            NS=NS+1;
            MS(NS)=IA;
            MT(NS)=IB;
        end
    end
% DETERMINE CANDIDATE NODES ON THE GENERATION FRONT
for I=1:NS    %do 22
    J=MS(I);
    P=X(J);
    Q=Y(J);
    if (~((P-XC)^2+(Q-YC)^2 > RR || A*P+B*Q < C)) %GOTO 22
        [Index] = CHKINT (J1,J2,J,X1,Y1,X2,Y2,P,Q,NS,MS,MT,X,Y);    %*22)
        if(Index~=11 && Index~=22)
            [XC,YC,RR] = CIRCLE (X1,Y1,X2,Y2,P,Q);
            J3=J;
        end
    end
end

if (J3 == 0)
    H=sqrt(RR-TOR-DD/4);
    R=sqrt(RR-TOR);
    AREA=sqrt(DD)*(R+H);
    ALPHA=AREA/((R+H)^2+0.75*DD);
else
    AREA=A*X(J3)+B*Y(J3)+X1*Y2-X2*Y1;
    S=DD+(X(J3)-X1)^2+(Y(J3)-Y1)^2+(X(J3)-X2)^2+(Y(J3)-Y2)^2;
    ALPHA=sqrt(12.0)*AREA/S;
end
% CREATE INTERIOR NODES, CHECK THEIR QUALITIES AND COMPARE WITH FRONTAL
```

```
% NODE J3
    XX=XM+A/2;
    YY=YM+B/2;
    S1=0;
    S2=0;
    for I=1:NP     %do 44
        S=(XP(I)-XX)^2+(YP(I)-YY)^2+TOR;
        S1=S1+DP(I)/S;
        S2=S2+1/S;
    end

    F=sqrt(0.75*S1/(S2*DD));
    F1=F;

    for I=1:5     %do 111
        F1=(2*F1^3+3*F)/(3*F1*F1+2.25);
    end

    S=F*DD/AREA;

    if (S > 1)
        S=1/S;
    end

    BETA=S*(2-S)*ALPHA;
    T=1/ALPHA-sqrt(abs(1/ALPHA^2-1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    flag=0;
    for I=1:9      %do 66
        S=(11-I)*F1/10;
        GAMMA=sqrt(3.0)*S*S*(2-S/F)/(S*S*F+0.75*F);
        if (GAMMA < BETA) %GOTO 1
           flag=1;
           break   %OK
        end
        P=XM+A*S;
        Q=YM+B*S;
        if (~((P-XC)^2+(Q-YC)^2 > RR)) %GOTO 66
            %???
        [Index] = CHKINT (J1,J2,0,X1,Y1,X2,Y2,P,Q,NS,MS,MT,X,Y);
        if(Index~=11 && Index~=22)
        D=(X(MT(1))-X(MS(1)))^2+(Y(MT(1))-Y(MS(1)))^2;
        H=DPL(X(MS(1)),Y(MS(1)),X(MT(1)),Y(MT(1)),P,Q);
        for J=2:NS     %do 99
            S=DPL(X(MS(J)),Y(MS(J)),X(MT(J)),Y(MT(J)),P,Q);
            if (S < H)
                %GOTO 99
                H=S;
                D=(X(MT(J))-X(MS(J)))^2+(Y(MT(J))-Y(MS(J)))^2;
            end
        end  %99-continue
        if (H > D*T^2)
            flag=3; %GOTO 3
            break
        end
        end   %refers to check CHKINT
        end  % % 66 CONTINUE
    end
%------------- After 66-continue ----------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Here execution comes when flag=0,1,3
%     fprintf('---IMMEDIATELY AFTER 66-CONTINUE:  flag=%3i\n',flag);
    if(flag==3)
        break   %goes out the "9-continue"
    else
        II=3*NE;    %when iflag=0,1.
    end
%--------------------------------------------------------------------------
% IF NO NODE CAN BE FOUND TO FORM A VALID ELEMENT WITH THE BASE SEGMENT,
% ENLARGE THE SEARCH RADIUS
%     fprintf('      J3 =%3i\n',J3);
    if (J3 ~= 0) %GOTO 2
        flag=2;
%         fprintf('Immediately at definition: flag=%3i\n',flag);
        break    %goes out the "9-continue" and is controlled again
    end     %NEW (13/5/19, 11:05am) <------------------------------
    if (RR > 100*DD) %THEN
        fprintf('*** Mesh generation failed! ***\n');
        return
    end
    XC=XC+XC-XM;
    YC=YC+YC-YM;
    RR=(XC-X1)^2+(YC-Y1)^2+TOR;
%GOTO 9   %is replaced by an artificial "i9-loop"
 end    %refers to i9=500000 continue
%---------------------- 9-CONTINUE ------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Here the cursor comes when flag=3 (after corresponding break)
% fprintf('   Immediately After 9-CONTINUE, flag=%3i\n',flag);
% fprintf('i5=%3i  i9=%3i\n',i5,i9);
% fprintf('--->KUKU-1\n');

% NODE J3 IS FOUND TO FORM VALID ELEMENT WITH BASE SEGMENT J1-J2
% UPDATE GENERATION FRONT WITH FRONTAL NODE J3

 if(flag ~= 3)   %NEW

    if(flag==0 || flag==1 || flag==2)
%     fprintf('--->KUKU-2\n');
    NE=NE+1;
    ME(II+1)=J1;
    ME(II+2)=J2;
    ME(II+3)=J3;

    xloc=[X(J1) X(J2) X(J3) X(J1)];
    yloc=[Y(J1) Y(J2) Y(J3) Y(J1)];
% % %     hold on
% % %     plot(xloc,yloc,'k')
    NELnumber = NELnumber + 1;
    xc=(X(J1)+X(J2)+X(J3))/3;
    yc=(Y(J1)+Y(J2)+Y(J3))/3;
%-------------
%
text(xc,yc,int2str(NELnumber),'FontAngle','italic','FontSize',11,'FontWeigh
t','Bold') %activate or not!
%-------------

            icount=0;
    for I=1:NB   %do 77
%         fprintf('--->KUKU-3\n');
```

```
        if (~((MA(I)~=J3 || MB(I)~=J1))) %GOTO 77
            MA(I)=MA(NB);
            MB(I)=MB(NB);
            NB=NB-1;
            icount=icount+1;
                flag=7;
                break   %goes out the 77-loop
        end
    end
%             fprintf('icount = %3i\n',icount);

    if(~(flag==7))
% fprintf('--->KUKU-4\n');
        NB=NB+1;
        MA(NB)=J1;
        MB(NB)=J3;
    end
end   % flag=0,1,2

% fprintf('--->KUKU-5\n');
% 7 DO 88 I=1,NB
    for I=1:NB
        if(~((MA(I)~=J2 || MB(I)~=J3)))    % GOTO 88

            if (NB==1) %RETURN
                return
            end

            MA(I)=MA(NB);
            MB(I)=MB(NB);
            NB=NB-1;
            %GOTO 5
            flag = 5;    %NEW
            break    %goto 5
        end  %end-of-IF
    end  %% 88 CONTINUE

if(flag~=5)
% fprintf('--->KUKU-6\n');
    NB=NB+1;
    MA(NB)=J3;
    MB(NB)=J2;
%     break
end
    %GOTO 5

% INTERIOR NODE NN CREATED, UPDATE GENERATION FRONT WITH INTERIOR NODE NN

 else    %NEW
% fprintf('*****************  INTERIOR POINT  **********************\n');
% fprintf('   Immediately After INTERIOR POINT, flag=%3i\n',flag);
% fprintf('--->KUKU-7\n');
    NN=NN+1;
    X(NN)=P;
    Y(NN)=Q;
    II=3*NE;
    NE=NE+1;
    ME(II+1)=J1;
    ME(II+2)=J2;
    ME(II+3)=NN;
```

```
    NB=NB+1;
    MA(NB)=J1;
    MB(NB)=NN;
    NB=NB+1;
    MA(NB)=NN;
    MB(NB)=J2;
% %     fprintf('Interior Point was finished\n');
%     hold on
%     plot(P,Q,'c+')
%     ds=1/50;
% %---------------
% %     text(P+ds,Q+ds,int2str(NN),'Color','c') %activate or not!
% %---------------
%     xloc=[X(J1) X(J2) P X(J1)];
%     yloc=[Y(J1) Y(J2) Q Y(J1)];
%     plot(xloc,yloc,'k')
%     NELnumber = NELnumber + 1;
%     xc=(X(J1)+X(J2)+P)/3;
%     yc=(Y(J1)+Y(J2)+Q)/3;
%---------------
%
text(xc,yc,int2str(NELnumber),'FontAngle','italic','FontSize',11,'FontWeigh
t','Bold') %activate or not!
%---------------
%         fprintf('J1=%3i J2=%3i J3=%3i\n',J1,J2,J3);
 end    %end-of-flag3    %NEW
% fprintf('--->KUKU-8\n');
end   %refers to i5=1:1000000
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
function [Index] = CHKINT (J1,J2,J,X1,Y1,X2,Y2,P,Q,NB,MA,MB,X,Y)
% % % % IMPLICIT DOUBLE PRECISION (A-H,O-Z)
% % % % DIMENSION MA(*),MB(*),X(*),Y(*)
    TOL = 0.000001;
    Index=0;    %dummy value
% ! Check if there are any intersections between line segment (P,Q)-(X1,Y1)
% ! and the non-Delaunay segments MA(i)-MB(i), i=1,NB
    C1=Q-Y1;
    C2=P-X1;
    C=Q*X1-P*Y1;
    CC=C1*C1+C2*C2;
    TOR=-TOL*CC*CC;
for I=1:NB   %DO 11 I=1,NB
    IA=MA(I);
    IB=MB(I);
    if (~(J==IA || J==IB || J1==IA || J1==IB))      %GOTO 11
        XA=X(IA);
        YA=Y(IA);
        XB=X(IB);
        YB=Y(IB);
    if (~((C2*YA-C1*XA+C)*(C2*YB-C1*XB+C) > TOR))   %GOTO 11
        H1=YB-YA;
        H2=XB-XA;
        H=XA*YB-XB*YA;
    if ((H2*Y1-H1*X1+H)*(H2*Q-H1*P+H) < TOR) %RETURN 1
        Index=11;
        return
    end
    end
```

```
    end
%  11 CONTINUE
    end
% ! Check if there are any intersections between line segment (P,Q)-(X2,Y2)
% ! and the non-Delaunay segments MA(i)-MB(i), i=1,NB
    C1=Q-Y2;
    C2=P-X2;
    C=Q*X2-P*Y2;
    CC=C1*C1+C2*C2;
    TOR=-TOL*CC*CC;
for I=1:NB    %DO 22 I=1,NB
    IA=MA(I);
    IB=MB(I);
    if (~(J == IA || J == IB || J2 == IA || J2 == IB))  %GOTO 22
    XA=X(IA);
    YA=Y(IA);
    XB=X(IB);
    YB=Y(IB);
    if (~((C2*YA-C1*XA+C)*(C2*YB-C1*XB+C) > TOR))       %GOTO 22
    H1=YB-YA;
    H2=XB-XA;
    H=XA*YB-XB*YA;
    if (((H2*Y2-H1*X2+H)*(H2*Q-H1*P+H) < TOR))         %RETURN 1
        Index=22;
        return
    end
    end
    end
% 22 CONTINUE   % 22 CONTINUE
    end
% % % END
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%-------------------------------------------------------------------------
% CALCULATE THE CIRCUMCIRCLE OF TRIANGLE (X1,Y1),(X2,Y2),(P,Q)
%-------------------------------------------------------------------------
function [XC,YC,RR] = CIRCLE (X1,Y1,X2,Y2,P,Q)
% % % IMPLICIT DOUBLE PRECISION (A-H,O-Z)
A1=X2-X1;
A2=Y2-Y1;
B1=P-X1;
B2=Q-Y1;
AA=A1*A1+A2*A2;
BB=B1*B1+B2*B2;
AB=A1*B1+A2*B2;
DET=AA*BB-AB*AB;
C1=0.5*BB*(AA-AB)/DET;
C2=0.5*AA*(BB-AB)/DET;
XX=C1*A1+C2*B1;
YY=C1*A2+C2*B2;
RR=1.000001*(XX*XX+YY*YY);
XC=X1+XX;
YC=Y1+YY;
end
```

```
%-------------------------------------------------------------------------
%CALCULATE THE DISTANCE BETWEEN POINT (X3,Y3) TO LINE SEGMENT (X1,Y1)-
(X2,Y2)
%-------------------------------------------------------------------------
function [DPLvalue] = DPL(X1,Y1,X2,Y2,X3,Y3)
% % % IMPLICIT DOUBLE PRECISION (A-H,O-Z)
R=(X2-X1)^2+(Y2-Y1)^2;
S=(X2-X1)*(X3-X1)+(Y2-Y1)*(Y3-Y1);
T=(X3-X1)^2+(Y3-Y1)^2;
DPLvalue=T-S*S/R;
if (S > R)
    DPLvalue=(X3-X2)^2+(Y3-Y2)^2;
end

if (S < 0)
    DPLvalue=T;
end

end


function [] = plot_division_points(outline_points,division_points)
figure
hold on
plot([outline_points(:,1);outline_points(1,1)],[outline_points(:,2);outline
_points(1,2)])
text(division_points(1,1)+0.1,division_points(1,2),'A','FontSize',14)
text(division_points(2,1),division_points(2,2)+0.6,'B','FontSize',14)
text(division_points(3,1)+0.1,division_points(3,2),'C','FontSize',14)
text(division_points(4,1)+0.001,division_points(4,2),'D','FontSize',14)
text(division_points(5,1)+0.001,division_points(5,2),'E','FontSize',14)
text(division_points(6,1)+0.4,division_points(6,2),'F','FontSize',14)
text(division_points(7,1)-0.6,division_points(7,2),'F''','FontSize',14)
text(division_points(8,1)-0.9,division_points(8,2),'C''','FontSize',14)
text(division_points(9,1),division_points(9,2)+0.6,'B''','FontSize',14)
text(division_points(10,1)-0.6,division_points(10,2),'A''','FontSize',14)
text(division_points(11,1)-1,division_points(11,2),'G''','FontSize',14)
text(division_points(12,1)+0.1,division_points(12,2),'G','FontSize',14)
axis equal
title('Areas of gear')
hold off
xlabel('X')
ylabel('Y')
end



function [] = plot_outline_points(outline_points)
%
figure
plot(outline_points(:,1),outline_points(:,2),'.')
axis equal
title('Outline points for advancing front mesh generation')
xlabel('X')
ylabel('Y')
drawnow
end
```

```
function [] = plot_tri_mesh(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,:),:) ; nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh')
xlabel('X')
ylabel('Y')
drawnow
hold off
end



function [] = plot_tri_mesh_nodal_points_numbered(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,:),:) ; nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh with NODAL POINT numbering')
drawnow
hold off

% CP actions:
    xmax=max(nodes(:,1)); xmin=min(nodes(:,1));
    ymax=max(nodes(:,2)); ymin=min(nodes(:,2));
    dsx=(xmax-xmin); dsy=(ymax-ymin);
    hold on
    for i=1:size(nodes,1)
       text(nodes(i,1),nodes(i,2),int2str(i),'Color','red','FontSize',09)
    end

end  %end-of-function


function [] = plot_tri_mesh_elements_numbered(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,:),:) ; nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh with ELEMENT numbering')
drawnow
hold off
% Addon by C.P. (15.04.2020):
    for i=1:size(elements,1)
        xc=0.3333*sum(nodes(elements(i,:),1));
```

```
        yc=0.3333*sum(nodes(elements(i,:),2));

text(xc,yc,int2str(i),'FontAngle','italic','FontSize',08,'FontWeight','Bold
')
    end

end     %end-of-function


function [] = plot_tri_mesh_node_and_element_numbering(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,:),:) ; nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh with NODE and ELEMENT numbering')
drawnow
hold off

% CP actions:
    xmax=max(nodes(:,1)); xmin=min(nodes(:,1));
    ymax=max(nodes(:,2)); ymin=min(nodes(:,2));
    dsx=(xmax-xmin); dsy=(ymax-ymin);
    hold on
    for i=1:size(nodes,1)
       text(nodes(i,1),nodes(i,2),int2str(i),'Color','red','FontSize',09)
    end

% Addon by C.P. (15.04.2020):
    for i=1:size(elements,1)
        xc=0.3333*sum(nodes(elements(i,:),1));
        yc=0.3333*sum(nodes(elements(i,:),2));

text(xc,yc,int2str(i),'FontAngle','italic','FontSize',08,'FontWeight','Bold
')
    end

end     %end-of-function



function [NELE,NODES,NDF,NDM,NEL,Area,BC,Fext,U] = AllocateMEMORY(NE,nodes)

%% ===    PRE-PROCESSOR & MEMORY ALLOCATION:
NELE    = NE;               % total number of elements
NODES   = size(nodes,1);     % total number of nodes
% Prepare (initialize) matrices and vectors:
NDF = 2;                    % number of DOF per node
NDM = 2;                    % dimension of problem (here is 2D-analysis)
NEL = 3;                    % number of nodes per element(linear triangle)
Area = zeros(NELE,1);       % cross-sectional area)
BC   = zeros(2*NODES,1);     % boundary conditions)
Fext = zeros(2*NODES,1);     % externally applied forces (initialization)

U  = zeros(2*NODES,1);        % Nodal Displacements (global)
```

```
end     %end-of-function


function BC = plot_boundary_conditions(BC,nodes,elements,fixed_nodes)
% MEMO: Boundary Conditions (k=1,...,2*NODES):
    % BC(k) = 0 (unrestrained DOF, k-th DOF)
    % BC(k) = 1 (restrained DOF, k-th DOF)
for i=1:numel(fixed_nodes)
    BC(2*fixed_nodes(i)-1:2*fixed_nodes(i))=1;
end
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,:),:) ; nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Fixed nodes')
drawnow
hold off
hold on
plot(nodes(fixed_nodes,1),nodes(fixed_nodes,2),'ko-');
end



function [Fext] =
load_distribution(Fext,loaded_nodes,pmax_th,gear,nodes,circle)
div=linspace(-1,1,numel(loaded_nodes));
rg=gear.rg;
fi=gear.fi;
rmin=circle.cross_points(2,3);
rmax=circle.cross_points(1,3);
r=zeros(size(loaded_nodes));r(1)=rmin;r(end)=rmax;
e=1e-6;
for i=2:size(r,2)-1
    a=rmin;
    b=rmax;
    r_test=(a+b)/2;
    t=sqrt((r_test/rg)^2-1);
    x=(rg*(sin(t)-t*cos(t)));
    y=rg*(cos(t)+t*sin(t));
    x1=-(cos(fi)*x-sin(fi)*y);
    y1=sin(fi)*x+cos(fi)*y;
    error=(sqrt((nodes(loaded_nodes(i),1)-x1)^2+(nodes(loaded_nodes(i),2)-
y1)^2));
    while error>e
        if y1>nodes(loaded_nodes(i),2)
            b=r_test;
        else
            a=r_test;
        end
        r_test=(a+b)/2;
        t=sqrt((r_test/rg)^2-1);
        x=(rg*(sin(t)-t*cos(t)));
        y=rg*(cos(t)+t*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
```

```
        error=(sqrt((nodes(loaded_nodes(i),1)-
x1)^2+(nodes(loaded_nodes(i),2)-y1)^2));
    end
    r(i)=r_test;
end
theta=atan2(nodes(loaded_nodes(end),2)-
nodes(loaded_nodes(1),2),nodes(loaded_nodes(end),1)-
nodes(loaded_nodes(1),1))-pi/2;
for i=1:numel(div)-1
    rmin=r(i);
    rmax=r(i+1);

    divmin=div(i);
    divmax=div(i+1);

    s=linspace(0,1,100000);

    Fext(2*loaded_nodes(i)-1)=Fext(2*loaded_nodes(i)-
1)+trapz(s,fun1(gear,rmin,rmax,s,pmax_th,divmin,divmax))*cos(theta)*gear.wi
dth;

Fext(2*loaded_nodes(i))=Fext(2*loaded_nodes(i))+trapz(s,fun1(gear,rmin,rmax
,s,pmax_th,divmin,divmax))*sin(theta)*gear.width;

    Fext(2*loaded_nodes(i+1)-1)=Fext(2*loaded_nodes(i+1)-
1)+trapz(s,fun2(gear,rmin,rmax,s,pmax_th,divmin,divmax))*cos(theta)*gear.wi
dth;

Fext(2*loaded_nodes(i+1))=Fext(2*loaded_nodes(i+1))+trapz(s,fun2(gear,rmin,
rmax,s,pmax_th,divmin,divmax))*sin(theta)*gear.width;
end



function F = fun1(gear,rmin,rmax,s,pmax_th,divmin,divmax)
rg=gear.rg;
fi=gear.fi;

tmin=sqrt((rmin/rg)^2-1);
tmax=sqrt((rmax/rg)^2-1);

dx_ds=rg*((-tmin+tmax)*cos((1-s).*tmin+s*tmax)-(-tmin+tmax).*cos((1-
s)*tmin+s*tmax)+((1-s).*tmin+s*tmax).*(-tmin+tmax).*sin((1-
s)*tmin+s*tmax));
dy_ds=rg*(-(-tmin+tmax)*sin((1-s).*tmin+s*tmax)+(-tmin+tmax).*sin((1-
s)*tmin+s*tmax)+((1-s).*tmin+s*tmax).*(-tmin+tmax).*cos((1-
s)*tmin+s*tmax));

dx1_ds=cos(fi)*dx_ds-sin(fi)*dy_ds;
dy1_ds=sin(fi)*dx_ds+cos(fi)*dy_ds;

F=-s*pmax_th.*sqrt(1-((1-
s)*divmin+s*divmax).^2).*sqrt(dx1_ds.^2+dy1_ds.^2);
end
```

```
function F = fun2(gear,rmin,rmax,s,pmax_th,divmin,divmax)
rg=gear.rg;
fi=gear.fi;

tmin=sqrt((rmin/rg)^2-1);
tmax=sqrt((rmax/rg)^2-1);

dx_ds=rg*((-tmin+tmax)*cos((1-s).*tmin+s*tmax)-(-tmin+tmax).*cos((1-
s)*tmin+s*tmax)+((1-s).*tmin+s*tmax).*(-tmin+tmax).*sin((1-
s)*tmin+s*tmax));
dy_ds=rg*(-(-tmin+tmax)*sin((1-s).*tmin+s*tmax)+(-tmin+tmax).*sin((1-
s)*tmin+s*tmax)+((1-s).*tmin+s*tmax).*(-tmin+tmax).*cos((1-
s)*tmin+s*tmax));

dx1_ds=cos(fi)*dx_ds-sin(fi)*dy_ds;
dy1_ds=sin(fi)*dx_ds+cos(fi)*dy_ds;

F=-(1-s)*pmax_th.*sqrt(1-((1-
s)*divmin+s*divmax).^2).*sqrt(dx1_ds.^2+dy1_ds.^2);
end


%% FEM-analysis
function [U] = FEA(nodes,elements,NODES,NELE,material,BC,Fext,...
                   Thickness,U)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
% PREPARE FOR ANALYIS
% Matrix preparation (Initialization):
     Kglob = sparse(2*NODES,2*NODES); % Global Stiffness Matrix
      Kloc = zeros(6,6);              % Local Stiffness Matrix
         U  = zeros(2*NODES,1);       % Nodal Displacements (global)
   Strain  = zeros(NELE,3);           % Global Strains (åx,åy,ãxy)
   Stress  = zeros(NELE,3);           % Global Stresses (óx,óy,ôxy)
 free_dofs = [];                      % serial numbers of free DOFs
fixed_dofs = [];                      % serial numbers of fixed DOFs
%-------------------------------------------------------------------------
%===    FINITE ELEMENT ANALYSIS (FEA)
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)               %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
% LOOP Over All triangular elemenets:
for i=1:NELE     % For each triangular element
 % Find the triangle's area Area(i) of i-th element
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area(i) = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;
 % Determine the stiffness Kloc of i-th element:
   bi = y2 - y3;     ci = x3 - x2;
   bj = y3 - y1;     cj = x1 - x3;
   bm = y1 - y2;     cm = x2 - x1;
   K1   = [ bi^2*d11  bi*ci*d12 bi*bj*d11 bi*cj*d12 bi*bm*d11 bi*cm*d12;
```

```
            ci*bi*d12 ci^2*d11  ci*bj*d12 ci*cj*d11 ci*bm*d12 ci*cm*d11;
            bj*bi*d11 bj*ci*d12 bj^2*d11  bj*cj*d12 bj*bm*d11 bj*cm*d12;
            cj*bi*d12 cj*ci*d11 cj*bj*d12 cj^2*d11  cj*bm*d12 cj*cm*d11
            bm*bi*d11 bm*ci*d12 bm*bj*d11 bm*cj*d12 bm^2*d11  bm*cm*d12
            cm*bi*d12 cm*ci*d11 cm*bj*d12 cm*cj*d11 cm*bm*d12 cm^2*d11];
   K2    = [ ci^2  ci*bi ci*cj ci*bj ci*cm ci*bm;
            ci*bi bi^2  bi*cj bi*bj bi*cm bi*bm;
            cj*ci cj*bi cj^2  cj*bj cj*cm cj*bm;
            bj*ci bj*bi bj*cj bj^2  bj*cm bj*bm
            cm*ci cm*bi cm*cj cm*bj cm^2  cm*bm
            bm*ci bm*bi bm*cj bm*bj bm*cm bm^2];
   Kloc =   Thickness(i)/4/Area(i)* ( K1 + d33 * K2);
  % Define the global DOF-numbering of this element:
   edof=[2*elements(i,1)-1; 2*elements(i,1);   2*elements(i,2)-1; ...
         2*elements(i,2);   2*elements(i,3)-1; 2*elements(i,3)];
  % Add the global stiffness matrix Kglob of i-th element in Global matrix:
   Kglob(edof,edof) = Kglob(edof,edof) + Kloc;   %MATLAB's advantage
end

% Find the unrestrained (free) and restrained (fixed) DOFs:
    free_dofs  = find(BC==0);   % ïé á/á ôùí ìç-ðåñéïñéóìÝíùí ÂÅ
    fixed_dofs = find(BC==1);   % ïé á/á ôùí ðåñéïñéóìÝíùí ÂÅ

% Calculate the unknown displacement components solving a linear system:
    U (free_dofs,1) = Kglob(free_dofs,free_dofs) \ Fext (free_dofs,1);
% The above solutions concerns the free-DOF positions of U-vector.
% Afterwards, U consists of ALL displacement components, known nd unknown.


%**************************************************************************
function [Strain,Stress,PrincipStress,Svm] = Element_Strain_Stress(nodes...
                                  ,elements,NELE,material,U,Thickness)
%**************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)                %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
% We calculate Strains & Stresses within each element:
for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area(i) = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;
    Nx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area(i));
    Ny=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area(i));
    % Local displacements:
        U1=U(2*(elements(i,1))-1); U2=U(2*(elements(i,2))-1);
        U3=U(2*(elements(i,3))-1);
        V1=U(2*(elements(i,1))  ); V2=U(2*(elements(i,2))  );
        V3=U(2*(elements(i,3))  );
    % Element strains:
        eps1 = Nx(1)*U1 + Nx(2)*U2 + Nx(3)*U3;                          % åx
```

```
        eps2 = Ny(1)*V1 + Ny(2)*V2 + Ny(3)*V3;                          % åy
        eps3 = Ny(1)*U1+Ny(2)*U2+Ny(3)*U3+Nx(1)*V1+Nx(2)*V2+Nx(3)*V3; % ãxy
    % Store strains into the global vector:
        Strain(i,1) = eps1; Strain(i,2) = eps2; Strain(i,3) = eps3;
    % Element stresses:
        sig = [d11 d12 0;
               d12 d11 0;
               0   0   d33] * [eps1 eps2 eps3]';
   % Store all stresses into a global vector:
        Stress(i,1) = sig(1); Stress(i,2) = sig(2); Stress(i,3) = sig(3);
   % Principal element stresses:
        I1=sig(1)+sig(2);               % first stress invariant
        I2=sig(1)*sig(2) - sig(3)^2;    % second stress invariant
        s1=(I1-sqrt(I1^2-4*I2))/2; s2=(I1+sqrt(I1^2-4*I2))/2;
        sig11 = max(s1,s2);     % maximum principal stress
        sig22 = I1 - sig11;     % minimum principal stress
        PrincipStress(i,1:2) = [sig11 sig22];
   % Von Mises element stress:
        Svm(i) = sqrt(sig11^2+sig22^2-sig11*sig22);
   % Enlarge the stress vector by von-Mises and principal stresses:
        Stress(i,4) = Svm(i);
        Stress(i,5) = PrincipStress(i,1);
        Stress(i,6) = PrincipStress(i,2);
end
%
end     %end-of-function


function [s_bend,bending_elements_right,bending_elements_left] =
bending_stress(Stress,elements,nodes,bending_nodes,NELE)
Sy=Stress(:,2);
for i=1:NELE
    for j=1:size(bending_nodes,2)
        if elements(i,1)==bending_nodes(1,j) ||
elements(i,2)==bending_nodes(1,j) || elements(i,3)==bending_nodes(1,j)
            bending_elements_right(i)=i;
            break
        end
    end
end
bending_elements_right=bending_elements_right(bending_elements_right~=0);
for i=1:NELE
    for j=1:size(bending_nodes,2)
        if elements(i,1)==bending_nodes(2,j) ||
elements(i,2)==bending_nodes(2,j) || elements(i,3)==bending_nodes(2,j)
            bending_elements_left(i)=i;
            break
        end
    end
end
bending_elements_left=bending_elements_left(bending_elements_left~=0);
s_bend=max(Sy(bending_elements_right));
s_bend(2,1)=min(Sy(bending_elements_left));

figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'k');
end
%---
```

```
bending_elements=[bending_elements_right bending_elements_left];
for i=1:size(bending_elements,2)
    ex=[nodes(elements(bending_elements(i),1),1)
nodes(elements(bending_elements(i),2),1)
nodes(elements(bending_elements(i),3),1)];
    ey=[nodes(elements(bending_elements(i),1),2)
nodes(elements(bending_elements(i),2),2)
nodes(elements(bending_elements(i),3),2)];
    Svalue = Sy(bending_elements(i));
    fill(ex,ey,Svalue);
end
colorbar
title('ELEMENT-BASED BENDING STRESS')
xlabel('X')
ylabel('Y')
axis equal
%--------------------------------------------------------------------------
end



function Display_Displacements(NODES,NELE,nodeLoad,U,NDF, ...
                               ScreenKey,ReportFileKey,fid16)
%
%--------------------------------------------------------------------------
if(ScreenKey == 1)
% SCREEN:
    fprintf(' ************************************************************\n');
    fprintf(' GEAR TOOTH ANALYSIS\n');
    fprintf(' NUMBER OF NODES             =%5i\n',NODES);
    fprintf(' NUMBER OF ELEMENTS          =%5i\n',NELE);
    %fprintf(' NODE WHERE LOAD IS APPLIED =%5i\n',nodeLoad);
    fprintf('   \n');
    fprintf(' ************************************************************\n');
    fprintf('   \n');
    fprintf(' RESULTS\n');
    fprintf('   \n');
    fprintf(' NODAL DISPLACEMENTS\n');
    fprintf('       NODE           U              V        \n');
    for I=1:NODES
        K1=NDF*(I-1)+1;
        K2=K1+NDF-1;
        fprintf('%10i %15.7f %15.7f \n',I,U(K1),U(K1+1));
    end
end
%
%--------------------------------------------------------------------------
if(ReportFileKey == 1)
% REPORT-FILE:
    fprintf(fid16,'
************************************************************\n');
    fprintf(fid16,' GEAR TOOTH ANALYSIS\n');
    fprintf(fid16,' NUMBER OF NODES             =%5i\n',NODES);
    fprintf(fid16,' NUMBER OF ELEMENTS          =%5i\n',NELE);
    fprintf(fid16,' NODE WHERE LOAD IS APPLIED =%5i\n',nodeLoad);
    fprintf(fid16,'   \n');
    fprintf(fid16,'
************************************************************\n');
    fprintf(fid16,'   \n');
    fprintf(fid16,' RESULTS\n');
    fprintf(fid16,'   \n');
```

```
    fprintf(fid16,' NODAL DISPLACEMENTS\n');
    fprintf(fid16,'       NODE           U              V        \n');
    for I=1:NODES
        K1=NDF*(I-1)+1;
        K2=K1+NDF-1;
        fprintf(fid16,'%10i %15.7f %15.7f \n',I,U(K1),U(K1+1));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function Display_Strains(NELE,Strain,ScreenKey,ReportFileKey,fid16)
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%  DISPLAY THE ELEMENT STRAINS ON SCREEN and REPORT-file:                       %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
if(ScreenKey == 1)
% SCREEN:
    fprintf('   \n');
    fprintf(' ELEMENT STRAINS\n');
    fprintf('      ELEMENT        Exx              Eyy              Gxy\n');
%
 for I=1:NELE
    fprintf('%10i %15.4f %15.4f %15.4f\n',I,Strain(I,1),...
        Strain(I,2),Strain(I,3));
 end
    fprintf('
**************************************************************************\n');
end
%
if(ReportFileKey == 1)
% REPORT-FILE:
    fprintf(fid16,'   \n');
    fprintf(fid16,' ELEMENT STRAINS\n');
    fprintf(fid16,'      ELEMENT        Exx              Eyy
Gxy\n');
%
 for I=1:NELE
    fprintf(fid16,'%10i %15.4f %15.4f %15.4f\n',I,Strain(I,1),...
        Strain(I,2),Strain(I,3));
 end
    fprintf(fid16,'
**************************************************************************
************************\n');
end
%
end     %end-of-function
```

```
function
Display_Stresses(NELE,Stress,PrincipStress,ScreenKey,ReportFileKey,fid16)
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  DISPLAY THE ELEMENT STRESSES ON SCREEN and REPORT-file:
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
if(ScreenKey == 1)
% SCREEN:
    fprintf('   \n');
    fprintf(' ELEMENT STRESSES\n');
    fprintf('      MEMBER        SXX             SYY             SXY
PRINCIP-11        PRINCIP-22\n');
%
 for I=1:NELE
    fprintf('%10i %15.4f %15.4f %15.4f %15.4f %15.4f\n',I,Stress(I,1),...
        Stress(I,2),Stress(I,3),PrincipStress(I,1),PrincipStress(I,2));
 end
    fprintf('
**********************************************************************\n');
end
%
if(ReportFileKey == 1)
% REPORT-FILE:
    fprintf(fid16,'   \n');
    fprintf(fid16,' ELEMENT STRESSES\n');
    fprintf(fid16,'      MEMBER        SXX             SYY             SXY
PRINCIP-11        PRINCIP-22\n');
%
 for I=1:NELE
    fprintf(fid16,'%10i %15.4f %15.4f %15.4f %15.4f %15.4f\n',I, ...
            Stress(I,1),Stress(I,2),Stress(I,3),PrincipStress(I,1), ...
            PrincipStress(I,2));
 end
    fprintf(fid16,'
**********************************************************************
**********************\n');
end
%
end     %end-of-function.



function plotSxx(NELE,nodes,elements,Stress)

%% COLOR FILL ELEMENT STRESS Sxx:
%
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
for i=1:NELE
```

```
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Sxx = Stress(i,1);
    fill(ex,ey,Sxx);
end
colorbar
title('ELEMENT-BASED STRESS Sxx')
xlabel('X')
ylabel('Y')
axis equal

end     %end-of-function


function plotSyy(NELE,nodes,elements,Stress);
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Syy = Stress(i,2);
    fill(ex,ey,Syy);
end
colorbar
title('ELEMENT-BASED STRESS Syy')
xlabel('X')
ylabel('Y')
axis equal

end     %end-of-function



function plotSxy(NELE,nodes,elements,Stress)
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
```

```
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Sxy = Stress(i,3);
    fill(ex,ey,Sxy);
end
colorbar
title('ELEMENT-BASED STRESS Sxy')
xlabel('X')
ylabel('Y')
axis equal

end     %end-of-function



function plot_VonMises(NELE,nodes,elements,Svm)
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Svalue = Svm(i);
    fill(ex,ey,Svalue);
end
colorbar
title('ELEMENT-BASED VON-MISES STRESS Svm')
xlabel('X')
ylabel('Y')
axis equal
%-------------------------------------------------------------------------
end     %end-of-function



function plotS11(NELE,nodes,elements,PrincipStress)

%% COLOR FILL ELEMENT PRINCIPAL STRESS S11:
%
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
```

```
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Splot = PrincipStress(i,1);
    fill(ex,ey,Splot);
end
colorbar
title('ELEMENT-BASED PRINCIPAL STRESS S11')
xlabel('X')
ylabel('Y')
axis equal

end     %end-of-function


function plotS22(NELE,nodes,elements,PrincipStress)
%
%% COLOR FILL ELEMENT PRINCIPAL STRESS S22:
%
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%---
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Splot = PrincipStress(i,2);
    fill(ex,ey,Splot);
end
colorbar
title('ELEMENT-BASED PRINCIPAL STRESS S22')
xlabel('X')
ylabel('Y')
axis equal

end     %end-of-function


function [StressNodal,Svm_nodal] = SmoothNodalStress(NODES,NELE,elements,
...
                                    Stress,Svm,PrincipStress)
%% NODAL SMOOTHED STRESS COMPONENTS
%% COMPUTE and PRINT NODAL STRESSES:
    fprintf('*** STRESSES ***\n');
    StressNodal(1:6*NODES)=0;   % '6' refers to S(xx,yy,xy,vm,11,22).
    NodesConnected(1:NODES) =0; %number of elements connected to a node
    for i=1:NELE
        i1=elements(i,1);  i2=elements(i,2);       i3=elements(i,3);
        SIGxx=Stress(i,1); SIGyy=Stress(i,2);      SIGxy=Stress(i,3);
```

```
        SIGvm=Svm(i);      S11=PrincipStress(i,1); S22=PrincipStress(i,2);
        %
        StressNodal(6*i1-5)=StressNodal(6*i1-5)+SIGxx;
        StressNodal(6*i1-4)=StressNodal(6*i1-4)+SIGyy;
        StressNodal(6*i1-3)=StressNodal(6*i1-3)+SIGxy;
        StressNodal(6*i1-2)=StressNodal(6*i1-2)+SIGvm;
        StressNodal(6*i1-1)=StressNodal(6*i1-1)+S11;
        StressNodal(6*i1-0)=StressNodal(6*i1-0)+S22;
       %
        StressNodal(6*i2-5)=StressNodal(6*i2-5)+SIGxx;
        StressNodal(6*i2-4)=StressNodal(6*i2-4)+SIGyy;
        StressNodal(6*i2-3)=StressNodal(6*i2-3)+SIGxy;
        StressNodal(6*i2-2)=StressNodal(6*i2-2)+SIGvm;
        StressNodal(6*i2-1)=StressNodal(6*i2-1)+S11;
        StressNodal(6*i2-0)=StressNodal(6*i2-0)+S22;
        %
        StressNodal(6*i3-5)=StressNodal(6*i3-5)+SIGxx;
        StressNodal(6*i3-4)=StressNodal(6*i3-4)+SIGyy;
        StressNodal(6*i3-3)=StressNodal(6*i3-3)+SIGxy;
        StressNodal(6*i3-2)=StressNodal(6*i3-2)+SIGvm;
        StressNodal(6*i3-1)=StressNodal(6*i3-1)+S11;
        StressNodal(6*i3-0)=StressNodal(6*i3-0)+S22;
        %
        NodesConnected(i1)=NodesConnected(i1)+1;
        NodesConnected(i2)=NodesConnected(i2)+1;
        NodesConnected(i3)=NodesConnected(i3)+1;
    end
    for i=1:NODES
        StressNodal(6*i-5)=StressNodal(6*i-5)/NodesConnected(i); %Sxx
        StressNodal(6*i-4)=StressNodal(6*i-4)/NodesConnected(i); %Syy
        StressNodal(6*i-3)=StressNodal(6*i-3)/NodesConnected(i); %Sxy
        StressNodal(6*i-2)=StressNodal(6*i-2)/NodesConnected(i); %Von-Mises
        StressNodal(6*i-1)=StressNodal(6*i-1)/NodesConnected(i); %S11
        StressNodal(6*i  )=StressNodal(6*i  )/NodesConnected(i); %S22
        Svm_nodal(i)=StressNodal(6*i-2);
        fprintf('node=%3i  Sxx=%12.5e  Syy=%12.5e  Sxy=%12.5e\n',i, ...
                 StressNodal(6*i-5),StressNodal(6*i-4),StressNodal(6*i-3));
    end
    fprintf('
**********************************************************************\n');
end     %end-of-function
%-------------------------------------------------------------------------



function [SmoothElementStress] = Smooth_Element_Stress(NELE,elements, ...
                                                        StressNodal)
%% NODAL SMOOTHED STRESS COMPONENTS
%% COMPUTE and PRINT NODAL STRESSES:
    for i=1:NELE
        i1=elements(i,1);  i2=elements(i,2);       i3=elements(i,3);
        for j=1:6
            SmoothElementStress(i,j)=(StressNodal(6*(i1-1)+j)+...
                                      StressNodal(6*(i2-1)+j)+...
                                      StressNodal(6*(i3-1)+j))/3;
        end
    end
end     %end-of-function
```

```
function plot_Smooth_VonMises(NELE,nodes,elements,StressNodal)
figure
hold on
for i=1:NELE
    plot([nodes(elements(i,:),1)' nodes(elements(i,1),1)],
[nodes(elements(i,:),2)' nodes(elements(i,1),2)],'b');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
%-----------------------------------------------------
for i=1:NELE
    ex=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1)];
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
    Svm1=StressNodal(6*i1-2);
    Svm2=StressNodal(6*i2-2);
    Svm3=StressNodal(6*i3-2);
    SVMcm = 1/3*(Svm1+Svm2+Svm3); %at center of mass of element based on
nodes
    fill(ex,ey,SVMcm);
end
colorbar
title('NODAL-BASED (SMOOTHED) VON-MISES STRESS Svm')
% % % title(titleChar)
xlabel('X')
ylabel('Y')
axis equal
```

MATLAB code and functions for LST element

Some of the functions involved in the code of LST elements are also used in the code of CST elements so they will not be in this section.

```
tic
close all
clear all
clc
format long
%% algorithm parameters
tol=10^-8; % universal tolerance for geometry calculations in length units
npoints=100; % number of points to sample from the tooth flank
%
% basic geometric parameters
gear.z=30; % # of teeth
gear.m=4; % module
gear.a0=deg2rad(20); % pressure angle [Convert angle from degrees to
radians]
gear.cs=0.5; % thickness coefficient at rolling circle
gear.ck=1; % addendum coefficient
gear.cf=1.25; % dedendum coefficient
gear.cc=0.38; % rack curvature coefficient
gear.width=40;%gear width
gear.i=1/1.5;%gear ratio
%
%%Material parameters
material.E=[2e5 2e5];%Elastic Modulus of gears(MPa)
```

```
material.v=[0.3 0.3];%Poisson coefficient
material.PlaneType=0;% plane-stress (0), plane-strain (=/0)
%
%%Load
F=1.008912416112217e+03;%N
%
%% supplementary geometry calculations
[gear.r0,gear.rg,gear.rf,gear.rk,gear.rc,gear.fi,gear.w] = ...
                                                    gear_radii_angles(gear);
gear.contact_r=[59.280370567072850  40.822387699923972];
%
%%Hertz calculations
[b_th,pmax_th]=hertz_calc(F,gear,material);
b=b_th;
%Theoretical  Bending stress calculation
s_bend_th = bending_stress_th(F,gear);
% Mesh size array: mesh seeds
% Define the seed size for the corresponding segments, in length units.
% Two options:
% A) Constant element size: To invoke this option set the second array
% element for each segment to zero, for example [0.1,0]
% B) Linearly changing element size: To invoke this option set the second
% array element for each segment to a positive real. For example using the
% value [0.1,0.2] for segment AB will result in element size 0.1 at A and
% 0.2 at B, whereas [0.2,0.1] gives 0.2 at A and 0.1 at B.
%
%
mesh_seeds = [0.2,0;...       %(A-B)
              0.2,0;...       %(B-C)
              0.2,b_th/10;...       %(C-D)
              b_th/10,0;...       %(D-E)
              b_th/10,0.2;...     %(E-F)
              0.2,0.5;...       %(F-F')
              0.2,0.5;...     %(F'-C')
              0.2,0;...       %(C'-B')
              0.2,0;...       %(B'-A')
              4,0.2;        %(A'-G')
              5,0;          %(G'-G)
              4,0.2]*1.5;     %(G-A)
%
% calculating the trochoid and involute meeting radius
gear.rs = trochoid_meet_involute(gear,tol);
%
% assigning the gear hub radius
gear.rh=33;
%
%circle of influence
circle=circle_of_infuence_calculation(gear,b);
%
%% extracting the outline points
[division_points,outline_points,loaded_nodes,fixed_nodes,bending_nodes] =
outline_points(gear,circle,mesh_seeds,tol);
%plot(outline_points(:,1),outline_points(:,2),'-')
%
%% meshing
neleoutline=size(outline_points,1);
[NN,NE,ME,Xnodes,Ynodes] = Advancing_Front_LO_2D...
(outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                        [1:neleoutline,1]);
%
% reshaping the node and element connectivity matrices
```

```
nodes=[Xnodes,Ynodes,zeros(size(Xnodes))];
elements=zeros(NE,3);
for i=1:NE
   elements(i,1)=ME(3*(i-1)+1);
   elements(i,2)=ME(3*(i-1)+2);
   elements(i,3)=ME(3*(i-1)+3);
end
[nodes,elements] = tri_midnodes_insertion(NE,elements,nodes);
% for i=1:NE
%     X=[nodes(elements(i,1),1) nodes(elements(i,2),1)
nodes(elements(i,3),1) nodes(elements(i,1),1)];
%     Y=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2) nodes(elements(i,1),2)];
%     plot(X,Y,'k-')
% end
% plot(outline_points(loaded_nodes,1),outline_points(loaded_nodes,2),'ro')
% plot(outline_points(fixed_nodes,1),outline_points(fixed_nodes,2),'bo')
%% plotting results
%
%plot_tooth_flank(gear);
%
plot_division_points(outline_points,division_points)
%
plot_outline_points(outline_points)
%
plot_six_node_tri_mesh(nodes,elements) %
%
%*************************************************************************
% Date: Friday 10.04.2020 (after SKYPE meeting) - 15.04.2020
%-------------------------------------------------------------------------
%% CP-addons: continues with node numbering and element numbering
%---Node numbering (in red color):
plot_six_node_tri_mesh_nodal_points_numbered(nodes,elements)%
%---Element numbering (in black color, italics):
plot_six_node_tri_mesh_elements_numbered(nodes,elements)%
%---Both node and element numbering (red and black, as abovementioned):
plot_six_node_tri_mesh_node_and_element_numbering(nodes,elements)%
%
%-------------------------------------------------------------------------
%%                            FEM - analysis:                             *
%-------------------------------------------------------------------------
% Memory allocation and some useful nodal variables:
% NELE : total number of elements in the mesh
% NODES: total number of nodes in the mesh
% NDF  : number of DOF per node
% NDM  : dimension of problem (here is 2D-analysis)
% NEL  : number of nodes per element(linear triangle)
% Area : cross-sectional area)
% BC   : index for boundary conditions (0=free, 1=fixed)
% Fext : externally applied forces (initialization)
% U    : nodal displacements (initialization)
    [NELE,NODES,NDF,NDM,NEL,Area,BC,Fext,U] =
six_node_tri_AllocateMEMORY(NE,nodes);
%-------------------------------------------------------------------------
%Boundary Conditions
% Initially, all DOFs are free (BC=0)
% Then, we determine only the restrained (i.e. fixed) DOFs:
fixed_nodes = fixed_mid_nodes(nodes,fixed_nodes);
BC = plot_boundary_conditions_six_node_tri(BC,nodes,elements,fixed_nodes);
% Externally Applied Forces
% default: all imposed forces, are initially zero.
```

```
% Thus, we define only the non-vanishing force components
loaded_nodes = loaded_mid_nodes(nodes,loaded_nodes);
%Fext = plot_load_six_node_tri(Fext,nodes,elements,loaded_nodes,F,gear);
Fext =
load_distribution_six_nodes(Fext,loaded_nodes,pmax_th,gear,nodes,circle);
% ENTER Thickness of elements (MANUALLY):
    Thickness(1:NELE,1)=gear.width;       % element thickness
%---------------------------------------------------------------------

%% Calculate the unknown displacements:
    [U] = FEA_six_node_tri(nodes,elements,NODES,NELE,material,BC,Fext,...
              Thickness,U);
%% NODAL STRESS COMPONENTS
    [StrainNodal,StressNodal,PrincipStressNodal,SvmNodal] =
Nodal_Strain_Stress_six_nodes(nodes,elements...

,NODES,NELE,material,U);
[s_bend,bending_elements_right,bending_elements_left] =
bending_stress_six_nodes(elements,nodes,bending_nodes,NELE,material,U);
%% COLOR FILL and Plot (ELEMENT STRESSES)
% Sxx stress component:
    plotSxx_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
% Syy stress component:
    plotSyy_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
% Sxy=Txy stress component:
    plotSxy_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
% Von-Mises element stress:
% % %     titleChar = 'ELEMENT-BASED VON-MISES STRESS Svm';
    plot_VonMises_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
% S11 Principal element stress: PrincipStress(i,1:2)
    plotS11_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
% S22 Principal element stress: PrincipStress(i,1:2)
    plotS22_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
toc


function [nodes,elements] = tri_midnodes_insertion(NE,elements,nodes)
k=size(nodes,1);
for i=1:NE
    for j=1:k
        if nodes(j,1)==(nodes(elements(i,1),1)+nodes(elements(i,2),1))/2 &&
nodes(j,2)==(nodes(elements(i,1),2)+nodes(elements(i,2),2))/2
            elements(i,4)=j;
            break
        elseif j==k
            k=k+1;
            elements(i,4)=k;
            nodes(k,:)=(nodes(elements(i,1),:)+nodes(elements(i,2),:))/2;
        end
    end
    for j=1:k
        if nodes(j,1)==(nodes(elements(i,2),1)+nodes(elements(i,3),1))/2 &&
nodes(j,2)==(nodes(elements(i,2),2)+nodes(elements(i,3),2))/2
            elements(i,5)=j;
```

```
            break
        elseif j==k
            k=k+1;
            elements(i,5)=k;
            nodes(k,:)=(nodes(elements(i,2),:)+nodes(elements(i,3),:))/2;
        end
    end
    for j=1:k
        if nodes(j,1)==(nodes(elements(i,3),1)+nodes(elements(i,1),1))/2 &&
nodes(j,2)==(nodes(elements(i,3),2)+nodes(elements(i,1),2))/2
            elements(i,6)=j;
            break
        elseif j==k
            k=k+1;
            elements(i,6)=k;
            nodes(k,:)=(nodes(elements(i,3),:)+nodes(elements(i,1),:))/2;
        end
    end
end
end


function [] = plot_six_node_tri_mesh(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,1),:)
                 nodes(elements(i,4),:)
                 nodes(elements(i,2),:)
                 nodes(elements(i,5),:)
                 nodes(elements(i,3),:)
                 nodes(elements(i,6),:)
                 nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh')
xlabel('X')
ylabel('Y')
drawnow
hold off
end


function [] = plot_six_node_tri_mesh_nodal_points_numbered(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,1),:)
                 nodes(elements(i,4),:)
                 nodes(elements(i,2),:)
                 nodes(elements(i,5),:)
                 nodes(elements(i,3),:)
                 nodes(elements(i,6),:)
                 nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
```

```
axis equal
title('Final triangle mesh with NODAL POINT numbering')
drawnow
hold off

% CP actions:
    xmax=max(nodes(:,1)); xmin=min(nodes(:,1));
    ymax=max(nodes(:,2)); ymin=min(nodes(:,2));
    dsx=(xmax-xmin); dsy=(ymax-ymin);
    hold on
    for i=1:size(nodes,1)
       text(nodes(i,1),nodes(i,2),int2str(i),'Color','red','FontSize',09)
    end

end  %end-of-function


function [] = plot_six_node_tri_mesh_elements_numbered(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,1),:)
                 nodes(elements(i,4),:)
                 nodes(elements(i,2),:)
                 nodes(elements(i,5),:)
                 nodes(elements(i,3),:)
                 nodes(elements(i,6),:)
                 nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh with ELEMENT numbering')
drawnow
hold off
% Addon by C.P. (15.04.2020):
    for i=1:size(elements,1)
        xc=(1/6)*sum(nodes(elements(i,:),1));
        yc=(1/6)*sum(nodes(elements(i,:),2));

text(xc,yc,int2str(i),'FontAngle','italic','FontSize',08,'FontWeight','Bold
')
    end

end     %end-of-function



function [] =
plot_six_node_tri_mesh_node_and_element_numbering(nodes,elements)
%
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,1),:)
                 nodes(elements(i,4),:)
                 nodes(elements(i,2),:)
```

```
                nodes(elements(i,5),:)
                nodes(elements(i,3),:)
                nodes(elements(i,6),:)
                nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Final triangle mesh with NODE and ELEMENT numbering')
drawnow
hold off

% CP actions:
    xmax=max(nodes(:,1)); xmin=min(nodes(:,1));
    ymax=max(nodes(:,2)); ymin=min(nodes(:,2));
    dsx=(xmax-xmin); dsy=(ymax-ymin);
    hold on
    for i=1:size(nodes,1)
       text(nodes(i,1),nodes(i,2),int2str(i),'Color','red','FontSize',09)
    end

% Addon by C.P. (15.04.2020):
    for i=1:size(elements,1)
        xc=(1/6)*sum(nodes(elements(i,:),1));
        yc=(1/6)*sum(nodes(elements(i,:),2));

text(xc,yc,int2str(i),'FontAngle','italic','FontSize',08,'FontWeight','Bold
')
    end

end     %end-of-function


function [NELE,NODES,NDF,NDM,NEL,Area,BC,Fext,U] =
six_node_tri_AllocateMEMORY(NE,nodes)

%% ===    PRE-PROCESSOR & MEMORY ALLOCATION:
NELE    = NE;               % total number of elements
NODES   = size(nodes,1);     % total number of nodes
% Prepare (initialize) matrices and vectors:
NDF = 2;                     % number of DOF per node
NDM = 2;                     % dimension of problem (here is 2D-analysis)
NEL = 6;                     % number of nodes per element(linear triangle)
Area = zeros(NELE,1);        % cross-sectional area)
BC   = zeros(2*NODES,1);     % boundary conditions)
Fext = zeros(2*NODES,1);     % externally applied forces (initialization)

U  = zeros(2*NODES,1);        % Nodal Displacements (global)

end     %end-of-function


function BC =
plot_boundary_conditions_six_node_tri(BC,nodes,elements,fixed_nodes)
% MEMO: Boundary Conditions (k=1,...,2*NODES):
    % BC(k) = 0 (unrestrained DOF, k-th DOF)
    % BC(k) = 1 (restrained DOF, k-th DOF)
for i=1:numel(fixed_nodes)
    BC(2*fixed_nodes(i)-1:2*fixed_nodes(i))=1;
end
%
```

```
figure
hold on
plot(nodes(:,1),nodes(:,2),'.b')
for i=1:size(elements,1)
    nodes_local=[nodes(elements(i,1),:)
                 nodes(elements(i,4),:)
                 nodes(elements(i,2),:)
                 nodes(elements(i,5),:)
                 nodes(elements(i,3),:)
                 nodes(elements(i,6),:)
                 nodes(elements(i,1),:)];
    plot(nodes_local(:,1),nodes_local(:,2),'b')
end
axis equal
title('Fixed nodes')
drawnow
hold off
hold on
plot(nodes(fixed_nodes,1),nodes(fixed_nodes,2),'ko-');
end


function loaded_nodes = loaded_mid_nodes(nodes,loaded_nodes)
ln1=loaded_nodes;
ln2=zeros(1,numel(ln1)-1);
k=1;
for i=1:numel(ln1)-1
    for j=1:numel(nodes)
        if nodes(j,1)==(nodes(ln1(i),1)+nodes(ln1(i+1),1))/2 &&
nodes(j,2)==(nodes(ln1(i),2)+nodes(ln1(i+1),2))/2
            ln2(k)=j;
            k=k+1;
            break
        end
    end
end
loaded_nodes=zeros(1,2*numel(ln2));
for i=1:numel(ln2)
    loaded_nodes(2*i-1)=ln1(i);
    loaded_nodes(2*i)=ln2(i);
end
loaded_nodes(end+1)=ln1(end);
end


function [Fext] =
load_distribution_six_nodes(Fext,loaded_nodes,pmax_th,gear,nodes,circle)
div=linspace(-1,1,numel(loaded_nodes));
rg=gear.rg;
fi=gear.fi;
rmin=circle.cross_points(2,3);
rmax=circle.cross_points(1,3);
r=zeros(size(loaded_nodes));r(1)=rmin;r(end)=rmax;
e=1e-6;
for i=2:size(r,2)-1
    a=rmin;
    b=rmax;
    r_test=(a+b)/2;
    t=sqrt((r_test/rg)^2-1);
    x=(rg*(sin(t)-t*cos(t)));
    y=rg*(cos(t)+t*sin(t));
```

```
    x1=-(cos(fi)*x-sin(fi)*y);
    y1=sin(fi)*x+cos(fi)*y;
    error=(sqrt((nodes(loaded_nodes(i),1)-x1)^2+(nodes(loaded_nodes(i),2)-
y1)^2));
    while error>e
        if y1>nodes(loaded_nodes(i),2)
            b=r_test;
        else
            a=r_test;
        end
        r_test=(a+b)/2;
        t=sqrt((r_test/rg)^2-1);
        x=(rg*(sin(t)-t*cos(t)));
        y=rg*(cos(t)+t*sin(t));
        x1=-(cos(fi)*x-sin(fi)*y);
        y1=sin(fi)*x+cos(fi)*y;
        error=(sqrt((nodes(loaded_nodes(i),1)-
x1)^2+(nodes(loaded_nodes(i),2)-y1)^2));
    end
    r(i)=r_test;
end
theta=atan2(nodes(loaded_nodes(end),2)-
nodes(loaded_nodes(1),2),nodes(loaded_nodes(end),1)-
nodes(loaded_nodes(1),1))-pi/2;
for i=1:2:numel(div)-2
    rmin=r(i);
    rmax=r(i+2);

    divmin=div(i);
    divmax=div(i+2);

    s=linspace(0,1,100000);

    Fext(2*loaded_nodes(i)-1)=Fext(2*loaded_nodes(i)-
1)+trapz(s,fun1_six_nodes(gear,rmin,rmax,s,pmax_th,divmin,divmax))*cos(thet
a)*gear.width;

Fext(2*loaded_nodes(i))=Fext(2*loaded_nodes(i))+trapz(s,fun1_six_nodes(gear
,rmin,rmax,s,pmax_th,divmin,divmax))*sin(theta)*gear.width;

    Fext(2*loaded_nodes(i+1)-1)=Fext(2*loaded_nodes(i+1)-
1)+trapz(s,fun3_six_nodes(gear,rmin,rmax,s,pmax_th,divmin,divmax))*cos(thet
a)*gear.width;

Fext(2*loaded_nodes(i+1))=Fext(2*loaded_nodes(i+1))+trapz(s,fun3_six_nodes(
gear,rmin,rmax,s,pmax_th,divmin,divmax))*sin(theta)*gear.width;

    Fext(2*loaded_nodes(i+2)-1)=Fext(2*loaded_nodes(i+2)-
1)+trapz(s,fun2_six_nodes(gear,rmin,rmax,s,pmax_th,divmin,divmax))*cos(thet
a)*gear.width;

Fext(2*loaded_nodes(i+2))=Fext(2*loaded_nodes(i+2))+trapz(s,fun2_six_nodes(
gear,rmin,rmax,s,pmax_th,divmin,divmax))*sin(theta)*gear.width;
end
```

```
%% FEM-analysis
function [U] =
FEA_six_node_tri(nodes,elements,NODES,NELE,material,BC,Fext,...
                    Thickness,U)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
% PREPARE FOR ANALYIS
% Matrix preparation (Initialization):
     Kglob = sparse(2*NODES,2*NODES); % Global Stiffness Matrix
      Kloc = zeros(12,12);              % Local Stiffness Matrix
        U  = zeros(2*NODES,1);        % Nodal Displacements (global)
   Strain  = zeros(NELE,3);           % Global Strains (åx,åy,ãxy)
   Stress  = zeros(NELE,3);           % Global Stresses (óx,óy,ôxy)
 free_dofs = [];                      % serial numbers of free DOFs
fixed_dofs = [];                      % serial numbers of fixed DOFs
%-------------------------------------------------------------------------
%===    FINITE ELEMENT ANALYSIS (FEA)
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)               %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
E=[d11 d12 0 ;d12 d11 0;0 0 d33];
% LOOP Over All triangular elemenets:
for i=1:NELE     % For each triangular element
 % Find the triangle's area Area(i) of i-th element
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area(i) = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;
 % Determine the stiffness Kloc of i-th element:

u=[nodes(elements(i,1),1);nodes(elements(i,1),2);nodes(elements(i,2),1);...

nodes(elements(i,2),2);nodes(elements(i,3),1);nodes(elements(i,3),2);...

nodes(elements(i,4),1);nodes(elements(i,4),2);nodes(elements(i,5),1);...

nodes(elements(i,5),2);nodes(elements(i,6),1);nodes(elements(i,6),2)];
   Kloc = Kloc_triag2_calculation(u,E,Thickness(i));
  % Define the global DOF-numbering of this element:
   edof=[2*elements(i,1)-1; 2*elements(i,1);   2*elements(i,2)-1; ...
         2*elements(i,2);   2*elements(i,3)-1; 2*elements(i,3);...
         2*elements(i,4)-1; 2*elements(i,4);   2*elements(i,5)-1;...
         2*elements(i,5); 2*elements(i,6)-1;   2*elements(i,6)];
  % Add the global stiffness matrix Kglob of i-th element in Global matrix:
   Kglob(edof,edof) = Kglob(edof,edof) + Kloc;   %MATLAB's advantage
end

% Find the unrestrained (free) and restrained (fixed) DOFs:
    free_dofs  = find(BC==0);   % ïé á/á ôùí ìç-ðåñéïñéóìÝíùí ÂÅ
    fixed_dofs = find(BC==1);   % ïé á/á ôùí ðåñéïñéóìÝíùí ÂÅ

% Calculate the unknown displacement components solving a linear system:
    U (free_dofs,1) = Kglob(free_dofs,free_dofs) \ Fext (free_dofs,1);
% The above solutions concerns the free-DOF positions of U-vector.
```

```
% Afterwards, U consists of ALL displacement components, known and unknown.


function [s_bend,bending_elements_right,bending_elements_left] =
bending_stress_six_nodes(elements,nodes,bending_nodes,NELE,material,U)
for i=1:NELE
    for j=1:size(bending_nodes,2)
        if elements(i,1)==bending_nodes(1,j) ||
elements(i,2)==bending_nodes(1,j) || elements(i,3)==bending_nodes(1,j)
            bending_elements_right(i)=i;
            break
        end
    end
end
bending_elements_right=bending_elements_right(bending_elements_right~=0);
for i=1:NELE
    for j=1:size(bending_nodes,2)
        if elements(i,1)==bending_nodes(2,j) ||
elements(i,2)==bending_nodes(2,j) || elements(i,3)==bending_nodes(2,j)
            bending_elements_left(i)=i;
            break
        end
    end
end
bending_elements_left=bending_elements_left(bending_elements_left~=0);

Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)                %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on

%right_side
maxx=0;
for i=1:size(bending_elements_right,2)
    x1=nodes(elements(bending_elements_right(i),1),1);
y1=nodes(elements(bending_elements_right(i),1),2);
    x2=nodes(elements(bending_elements_right(i),2),1);
y2=nodes(elements(bending_elements_right(i),2),2);
    x3=nodes(elements(bending_elements_right(i),3),1);
y3=nodes(elements(bending_elements_right(i),3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(bending_elements_right(i),1)-1);
    U2=U(2*elements(bending_elements_right(i),2)-1);
    U3=U(2*elements(bending_elements_right(i),3)-1);
    U4=U(2*elements(bending_elements_right(i),4)-1);
    U5=U(2*elements(bending_elements_right(i),5)-1);
    U6=U(2*elements(bending_elements_right(i),6)-1);
```

```
    V1=U(2*elements(bending_elements_right(i),1));
    V2=U(2*elements(bending_elements_right(i),2));
    V3=U(2*elements(bending_elements_right(i),3));
    V4=U(2*elements(bending_elements_right(i),4));
    V5=U(2*elements(bending_elements_right(i),5));
    V6=U(2*elements(bending_elements_right(i),6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
           (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
           (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
```

```
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';
        s_bend=abs(sig(2));
        fill(ex,ey,s_bend,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
        if s_bend>maxx
            maxx=s_bend;
        end
    end
end
s_bend=maxx;
%left_side
minn=0;
for i=1:size(bending_elements_left,2)
    x1=nodes(elements(bending_elements_left(i),1),1);
y1=nodes(elements(bending_elements_left(i),1),2);
    x2=nodes(elements(bending_elements_left(i),2),1);
y2=nodes(elements(bending_elements_left(i),2),2);
    x3=nodes(elements(bending_elements_left(i),3),1);
y3=nodes(elements(bending_elements_left(i),3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(bending_elements_left(i),1)-1);
    U2=U(2*elements(bending_elements_left(i),2)-1);
    U3=U(2*elements(bending_elements_left(i),3)-1);
    U4=U(2*elements(bending_elements_left(i),4)-1);
    U5=U(2*elements(bending_elements_left(i),5)-1);
    U6=U(2*elements(bending_elements_left(i),6)-1);

    V1=U(2*elements(bending_elements_left(i),1));
    V2=U(2*elements(bending_elements_left(i),2));
    V3=U(2*elements(bending_elements_left(i),3));
    V4=U(2*elements(bending_elements_left(i),4));
    V5=U(2*elements(bending_elements_left(i),5));
    V6=U(2*elements(bending_elements_left(i),6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;
```

```
    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
           (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
           (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';
        s_bend(2,1)=sig(2);
        fill(ex,ey,s_bend(2,1),'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
        if s_bend(2,1)<minn
            minn=s_bend(2,1);
```

```
        end
    end
end
s_bend(2,1)=minn;

for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
end
colorbar
title('BENDING STRESS')
xlabel('X')
ylabel('Y')
axis equal
end



function plotSxx_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)                %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on

for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
```

```
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                        [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
            (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
            (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];
```

```
        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';

        Sxx=sig(1);
        fill(ex,ey,Sxx,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
    end
end
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
colorbar
title('NODAL-BASED STRESS Sxx')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function


function plotSyy_six_nodes(nodes,elements...
                          ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)                %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on

for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
```

```
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
            (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
            (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
```

```
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';

        Syy=sig(2);
        fill(ex,ey,Syy,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
    end
end
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
colorbar
title('NODAL-BASED STRESS Syy')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function


function plotSxy_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)               %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
```

```
%
div=2;
figure
hold on

for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
```

```
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
           (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
           (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';

        Sxy=sig(3);
        fill(ex,ey,Sxy,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
    end
end
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
colorbar
title('NODAL-BASED STRESS Sxy')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function


function plot_VonMises_six_nodes(nodes,elements...
                          ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
```

```
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)               %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on
maxx=0;
for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];
```

```
    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                        [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
           (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
           (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';

        I1=sig(1)+sig(2);
        I2=sig(1)*sig(2) - sig(3)^2;
        s1=(I1-sqrt(I1^2-4*I2))/2; s2=(I1+sqrt(I1^2-4*I2))/2;
        sig11 = max(s1,s2);
        sig22 = I1 - sig11;
        Svm=sqrt(sig11^2+sig22^2-sig11*sig22);
        fill(ex,ey,Svm,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
        if Svm>maxx
            maxx=Svm;
        end
    end
end
```

```
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
maxx
colorbar
title('NODAL-BASED STRESS VON MISES')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function


function plotS11_six_nodes(nodes,elements...
                          ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)                %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on

for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
```

```
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;

        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
           (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
           (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
            Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
            Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
            4*Sx(1)*S(2)+4*S(1)*Sx(2)
            4*Sx(2)*S(3)+4*S(2)*Sx(3)
            4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*2*Sy(1)
            Sy(2)*(2*S(2)-1)+S(2)*2*Sy(2)
            Sy(3)*(2*S(3)-1)+S(3)*2*Sy(3)
            4*Sy(1)*S(2)+4*S(1)*Sy(2)
            4*Sy(2)*S(3)+4*S(2)*Sy(3)
            4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;
```

```
eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
               d12 d11 0;
               0   0   d33] * [eps1 eps2 eps3]';

        I1=sig(1)+sig(2);
        I2=sig(1)*sig(2) - sig(3)^2;
        s1=(I1-sqrt(I1^2-4*I2))/2; s2=(I1+sqrt(I1^2-4*I2))/2;
        sig11 = max(s1,s2);
        sig22 = I1 - sig11;
        S11=sig11;
        fill(ex,ey,S11,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
    end
end
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
colorbar
title('NODAL-BASED STRESS S11')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function


function plotS22_six_nodes(nodes,elements...
                           ,NODES,NELE,material,U)

%% COLOR FILL ELEMENT STRESS Sxx:
%
%************************************************************************
% We define the analysis type through the variable 'PlaneType':
    %PLANE-STRESS: PlaneType=0 (plane-stress state)
    %PLANE-STRAIN: PlaneType~0 (plane-strain state)
Young=material.E(1);
xnu=material.v(1);
PlaneType=material.PlaneType;
d33=Young(1)/2/(1+xnu);
if (PlaneType == 0)               %plane-stress (0), plane-strain (=/0)
    d11=Young(1)/(1-xnu^2); d12=xnu*d11;
else
    d11=Young(1)*(1-xnu)/(1+xnu)/(1-2*xnu); d12=xnu/(1-xnu)*d11;
end
%
div=2;
figure
hold on

for i=1:NELE
    x1=nodes(elements(i,1),1); y1=nodes(elements(i,1),2);
    x2=nodes(elements(i,2),1); y2=nodes(elements(i,2),2);
    x3=nodes(elements(i,3),1); y3=nodes(elements(i,3),2);
```

```
    Area = ((x2-x1)*(y3-y1) - (y2-y1)*(x3-x1))/2;

    Sx=[(y2-y3) (y3-y1) (y1-y2)]/(2*Area);
    Sy=[(x3-x2) (x1-x3) (x2-x1)]/(2*Area);

    U1=U(2*elements(i,1)-1);
    U2=U(2*elements(i,2)-1);
    U3=U(2*elements(i,3)-1);
    U4=U(2*elements(i,4)-1);
    U5=U(2*elements(i,5)-1);
    U6=U(2*elements(i,6)-1);

    V1=U(2*elements(i,1));
    V2=U(2*elements(i,2));
    V3=U(2*elements(i,3));
    V4=U(2*elements(i,4));
    V5=U(2*elements(i,5));
    V6=U(2*elements(i,6));

    l1=sqrt((x2-x3)^2+(y2-y3)^2);
    l2=sqrt((x3-x1)^2+(y3-y1)^2);
    l3=sqrt((x1-x2)^2+(y1-y2)^2);
    ele_lengths=[min([l1 l2 l3])/div 0];

    start_point=[x1 y1];
    end_point=[x2 y2];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=outline_nodes;

    start_point=[x2 y2];
    end_point=[x3 y3];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes];

    start_point=[x3 y3];
    end_point=[x1 y1];
    outline_nodes = outline_nodes_line(start_point,end_point,ele_lengths);
    outline_points=[outline_points(1:end-1,:);outline_nodes(1:end-1,:)];

    neleoutline=size(outline_points,1);
    [NN_plot,NE_plot,ME_plot,Xnodes_plot,Ynodes_plot] =
Advancing_Front_LO_2D...
    (outline_points(:,1),outline_points(:,2),neleoutline,1:neleoutline,...
                                                    [1:neleoutline,1]);
    nodes_plot=[Xnodes_plot,Ynodes_plot,zeros(size(Xnodes_plot))];
    elements_plot=zeros(NE_plot,3);
    for j=1:NE_plot
       elements_plot(j,1)=ME_plot(3*(j-1)+1);
       elements_plot(j,2)=ME_plot(3*(j-1)+2);
       elements_plot(j,3)=ME_plot(3*(j-1)+3);
    end
    for j=1:NE_plot
        ex=[nodes_plot(elements_plot(j,1),1)
nodes_plot(elements_plot(j,2),1) nodes_plot(elements_plot(j,3),1)];
        ey=[nodes_plot(elements_plot(j,1),2)
nodes_plot(elements_plot(j,2),2) nodes_plot(elements_plot(j,3),2)];
        x=sum(ex)/3;
        y=sum(ey)/3;
```

```
        S=[(x2*y3-x3*y2)+x*(y2-y3)+y*(x3-x2)
             (x3*y1-x1*y3)+x*(y3-y1)+y*(x1-x3)
             (x1*y2-x2*y1)+x*(y1-y2)+y*(x2-x1)]/(2*Area);
        Nx=[Sx(1)*(2*S(1)-1)+S(1)*2*Sx(1)
             Sx(2)*(2*S(2)-1)+S(2)*2*Sx(2)
             Sx(3)*(2*S(3)-1)+S(3)*2*Sx(3)
             4*Sx(1)*S(2)+4*S(1)*Sx(2)
             4*Sx(2)*S(3)+4*S(2)*Sx(3)
             4*Sx(3)*S(1)+4*S(3)*Sx(1)];
        Ny=[Sy(1)*(2*S(1)-1)+S(1)*(2*Sy(1)-1) Sy(2)*(2*S(2)-
1)+S(2)*(2*Sy(2)-1) Sy(3)*(2*S(3)-1)+S(3)*(2*Sy(3)-1)...
             4*Sy(1)*S(2)+4*S(1)*Sy(2) 4*Sy(2)*S(3)+4*S(2)*Sy(3)
4*Sy(3)*S(1)+4*S(3)*Sy(1)];

        eps1=Nx(1)*U1+Nx(2)*U2+Nx(3)*U3+Nx(4)*U4+Nx(5)*U5+Nx(6)*U6;
        eps2=Ny(1)*V1+Ny(2)*V2+Ny(3)*V3+Ny(4)*V4+Ny(5)*V5+Ny(6)*V6;

eps3=Ny(1)*U1+Nx(1)*V1+Ny(2)*U2+Nx(2)*V2+Ny(3)*U3+Nx(3)*V3+Ny(4)*U4+Nx(4)*V
4+Ny(5)*U5+Nx(5)*V5+Ny(6)*U6+Nx(6)*V6;

        sig = [d11 d12 0;
                   d12 d11 0;
                   0   0   d33] * [eps1 eps2 eps3]';

        I1=sig(1)+sig(2);
        I2=sig(1)*sig(2) - sig(3)^2;
        s1=(I1-sqrt(I1^2-4*I2))/2; s2=(I1+sqrt(I1^2-4*I2))/2;
        sig11 = max(s1,s2);
        sig22 = I1 - sig11;
        S22=sig22;
        fill(ex,ey,S22,'LineStyle','none');
        %scatter(x,y,100,Sxx,'filled');
    end
end
for i=1:NELE
    plot([nodes(elements(i,[1 4 2 5 3 6]),1)' nodes(elements(i,1),1)],
[nodes(elements(i,[1 4 2 5 3 6]),2)' nodes(elements(i,1),2)],'k');
    ey=[nodes(elements(i,1),2) nodes(elements(i,2),2)
nodes(elements(i,3),2)];
    i1=elements(i,1); i2=elements(i,2); i3=elements(i,3);
end
colorbar
title('NODAL-BASED STRESS S22')
xlabel('X')
ylabel('Y')
axis equal
end     %end-of-function
```