ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΚΟΠΗΣ & ΚΑΤΑΣΚΕΥΑΣΤΙΚΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

ΑΡΙΘΜΗΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ ΤΗΣ ΣΥΓΚΡΟΥΣΗΣ ΤΩΝ ΑΠΟΒΛΙΤΤΩΝ ΣΤΟ ΦΡΑΙΖΑΡΙΣΜΑ ΜΕ ΚΥΛΙΣΗ ΟΔΟΝΤΩΣΕΩΝ

ΔΗΜΗΤΡΙΟΣ ΣΦΑΚΙΑΝΑΚΗΣ

ΕΠΙΒΛΕΠΩΝ: ΑΡΙΣΤΟΜΕΝΗΣ ΑΝΤΩΝΙΑΔΗΣ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ

ΑΡ. ΔΙΠΛ. : 9

ΧΑΝΙΑ 2010

Αφιερώνω αυτή την εργασία

στη γιαγιά μου

Μαρία Τσακαλάκη

Μετά την ολοκλήρωση της πτυχιακής μου εργασίας θα ήθελα να ευχαριστήσω θερμά

Τους γονείς μου και τον αδελφό μου για την υποστήριξη που μου έδωσαν τα χρόνια που φοίτησα στο πολυτεχνείο Κρήτης

Τον καθηγητή κύριο Αριστομένη Αντωνιάδη, Δρ μηχανολόγο μηχανικό που μου έδωσε την ευκαιρία να συνεργαστώ μαζί του και με καθοδήγησε κατά τη διάρκεια της εκπόνησης της εργασίας

Τον υποψήφιο διδάκτορα Ταπόγλου Νίκο για την υπερπολύτιμη βοήθεια του και την απεριόριστη υπομονή του

και τέλος στους μεταπτυχιακούς φοιτητές Μπελλή Ταξιάρχη και Δημήτρη Βακόνδιο για την εξίσου σημαντική βοήθεια τους.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια η ραγδαία ανάπτυξη της επιστήμης δίνει ολοένα και περισσότερες λύσεις στην εξυπηρέτηση των αναγκών του ανθρώπου. Συγκεκριμένα στη βιομηχανία, η ανακάλυψη και πρακτική εφαρμογή αυτοματοποιημένων διαδικασιών έχουν φέρει την επανάσταση στην παραγωγική διαδικασία και την έχουν εξελίξει σε επίπεδα που ήταν πέρα από κάθε ανθρώπινη φαντασία ορισμένες δεκαετίες πριν. Σημαντικός καταλύτης για την προσπάθεια αυτή που εξακολουθεί να γίνεται, είναι η ανάγκη του ανθρώπου για βελτιστοποίηση των συνθηκών παραγωγής, δηλαδή αύξηση του ρυθμού παραγωγής και μείωση του κόστους κατεργασίας χωρίς υποβάθμιση του τελικού προϊόντος.

Μία από τις διαδικασίες παραγωγής πάνω στην οποία έχει γίνει προσπάθεια εξέλιξής και στην οποία είναι βασισμένη αυτή η εργασία, είναι η διαμόρφωση οδοντώσεων με φραιζάρισμα με κύλιση (Gear Hobbing). Παλαιότερα η διαδικασία διαμόρφωσης οδοντώσεων γινόταν χειροκίνητα με αποτέλεσμα η ποιότητα του τελικού προϊόντος να ήταν συνάρτηση της δεξιότητας του χειριστή της εκάστοτε μηχανής παραγωγής. Τώρα πια ο ανθρώπινος παράγοντας λάθους έχει εξαλειφθεί καθώς με τη βοήθεια κατάλληλου λογισμικού η διαδικασία κοπής έχει αυτοματοποιηθεί πλήρως. Επίσης εκτός από τη δυνατότητα για προγραμματισμό της επικείμενης κατεργασίας, υπάρχει η δυνατότητα προσομοίωσής της σε εικονικό περιβάλλον συμβάλλοντας έτσι στην αποφυγή τυχόν προγραμματιστικών λαθών.

Ωστόσο παρά την πλήρη αυτοματοποίηση της διαδικασίας κοπής οδοντώσεων με φραιζάρισμα με κύλιση, δεν παύουν να υπάρχουν προβλήματα όπως αυτό της φθοράς του κοπτικού εργαλείου. Είναι φυσιολογικό μετά το πέρας ενός μεγάλου αριθμού κοπών, το εργαλείο κοπής να έχει υποστεί αλλεπάλληλες καταπονήσεις με αποτέλεσμα τη φθορά του, αλλοιώνοντας έτσι το τελικό προϊόν.

Σκοπός της παρούσας εργασίας είναι η παροχή πληροφοριών σχετικά με το πώς η σύγκρουση του υλικού αποβλίττου που παράγεται κατά τη διαδικασία της κοπής στην εκάστοτε θέση κύλισης, επηρεάζει τη φθορά του κοπτικού εργαλείου.

Πιο συγκεκριμένα με τη βοήθεια κατάλληλου λογισμικού (πρόγραμμα Inventor και Matlab) και τη δημιουργία κατάλληλου κώδικα προγραμματισμού (σε γλώσσα Visual Basic for Applications ‘VBA’), γίνεται η απαραίτητη επεξεργασία του παραγόμενου στερεού αποβλίττου το οποίο έχει ήδη παραχθεί από μία προσομοίωση της διαδικασίας διαμόρφωσης, με σκοπό την απεικόνιση των περιοχών των δοντιών του κοπτικού εργαλείου που φθείρονται από τη δημιουργία του αποβλίττου. Επίσης γίνεται μία γενική περιγραφή της κατεργασίας κοπής οδοντώσεων με φραιζάρισμα με κύλιση και μία εκτενή περιγραφή της φθοράς του κοπτικού εργαλείου κατά την κατεργασία αυτή.

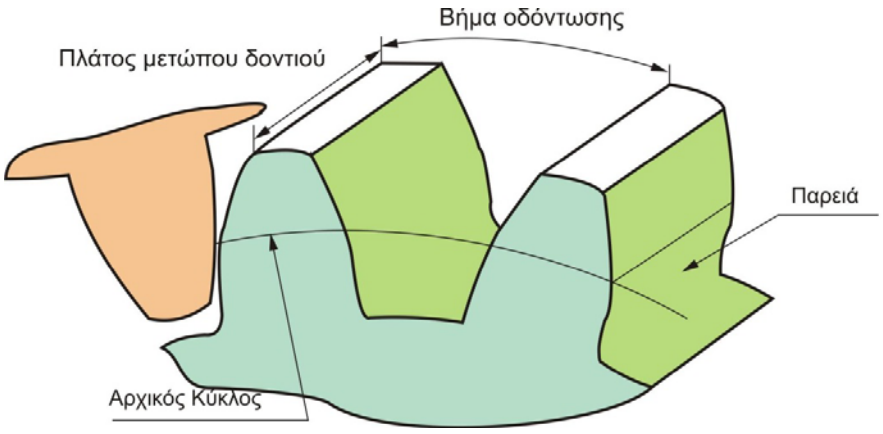
2. ΦΘΟΡΑ ΤΟΥ ΚΟΠΤΙΚΟΥ ΚΑΤΑ ΤΗ ΔΙΑΜΟΡΦΩΣΗ ΟΔΟΝΤΩΣΕΩΝ ΜΕ ΦΡΑΙΖΑΡΙΣΜΑ ΜΕ ΚΥΛΙΣΗ

2.1 Η κατεργασία του Gear Hobbing

Η διαμόρφωση οδοντώσεων είναι μια διαδικασία που λαμβάνει χώρα εδώ και χιλιάδες χρόνια και έχει ωφελήσει την ανθρωπότητα με ποικίλους τρόπους. Είναι φυσιολογικό με την εξέλιξη της επιστήμης να έχουν ανακαλυφθεί νέες μέθοδοι για την παραγωγή γραναζιών ταχύτερα και με μεγαλύτερη ακρίβεια.

Η κυριότερη και πιο παραγωγική από αυτές τις μεθόδους είναι η κοπή οδοντώσεων με φραιζάρισμα με κύλιση. Απαιτούμενα εργαλεία για την υλοποίηση αυτής της κατεργασίας είναι μια μηχανή κοπής γραναζιών, καθώς και η απαραίτητη θεωρητική γνώση για τον κατάλληλο προγραμματισμό αυτής για την εξαγωγή των επιθυμητών γραναζιών.

Μια μηχανή κοπής γραναζιών χρησιμοποιεί μια σειρά ειδικά διαμορφωμένων κομματιών που συμβάλουν στη δημιουργία συγκεκριμένων διαδικασιών κοπής και διαμόρφωσης. Τα κομμάτια αυτά κατέχουν τα απαραίτητα χαρακτηριστικά για να χρησιμοποιηθούν σε συγκεκριμένους τύπους εξοπλισμού. Οι μοντέρνες μηχανές κοπής οδοντώσεων είναι πλήρως αυτοματοποιημένες και ποικίλουν σε μεγέθη αφού πρέπει να είναι σε θέση να παράγουν γρανάζια με διάμετρο που φτάνει έως τα τρία μέτρα. Κάθε μηχανή κοπής αποτελείται από ένα σφιγκτήρα (τσοκ), από μια κατασκευή που χρησιμεύει στην επιπλέον στήριξη του κατεργαζόμενου κομματιού, από μια άτρακτο στην οποία τοποθετείται το κοπτικό εργαλείο και έναν ηλεκτροκινητήρα. Οι μηχανές χαρακτηρίζονται από το μεγαλύτερο module ή τη μεγαλύτερη διάμετρο αρχικού κύκλου (pitch diameter) που μπορούν να κατεργαστούν. Για παράδειγμα μια μηχανή κοπής χωρητικότητας 250mm μπορεί να διαμορφώσει οδοντώσεις με διάμετρο αρχικού κύκλου έως 250mm. Στο σχήμα 2.1 φαίνεται ο αρχικός κύκλος της οδόντωσης καθώς και ορισμένα άλλα βασικά χαρακτηριστικά του γραναζιού.



Σχήμα 2.1: Βασική ορολογία στα γρανάζια

Το module είναι μονάδα μέτρησης που καθορίζει το μέγεθος ενός γραναζιού. Ορίζεται σαν το πηλίκο της διαμέτρου αρχικού κύκλου με τον αριθμό των δοντιών του γραναζιού και είναι η σημαντικότερη πληροφορία κατά τη διαμόρφωση οδοντώσεων καθώς όλα τα μεγέθη και η γεωμετρία του κοπτικού εκφράζονται συναρτήσει του. Επίσης άλλος ένας σημαντικός όρος είναι το βήμα της οδόντωσης (pitch) του προς κοπή γραναζιού, που φαίνεται στο σχήμα 2.1,

το οποίο ορίζεται ως το πηλίκο της περιμέτρου του εξωτερικού κύκλου, προς τον αριθμό των δοντιών του γραναζιού.

Το κοπτικό εργαλείο που χρησιμοποιείται κατά την κοπή οδοντώσεων με φραιζάρισμα με κύλιση, είναι κυλινδρικό, με ελικοειδή δόντια κοπής όπως φαίνεται στο σχήμα 2.2.(α). Κατά μήκος του κυλίνδρου υπάρχουν αυλάκια τα οποία βοηθάνε στην κοπή και στην αφαίρεση του αποβλίττου που παράγεται. Υπάρχουν πολλά είδη κοπτικών για κοπή γραναζιών, κάθε ένα από τα οποία βρίσκει εφαρμογή σε διαφορετική διαδικασία, εξαρτάται δηλαδή από το παραγόμενο γρανάζι. Τα κοπτικά αυτά κατασκευάζονται από πολύ ανθεκτικά μέταλλα κατάλληλα τροποποιημένα να αντέχουν σε υψηλές καταπονήσεις (π.χ. M35 και ASP2030 ταχυχάλυβα). Ορισμένα από αυτά φαίνονται στο σχήμα 2.2.(β).

(α)

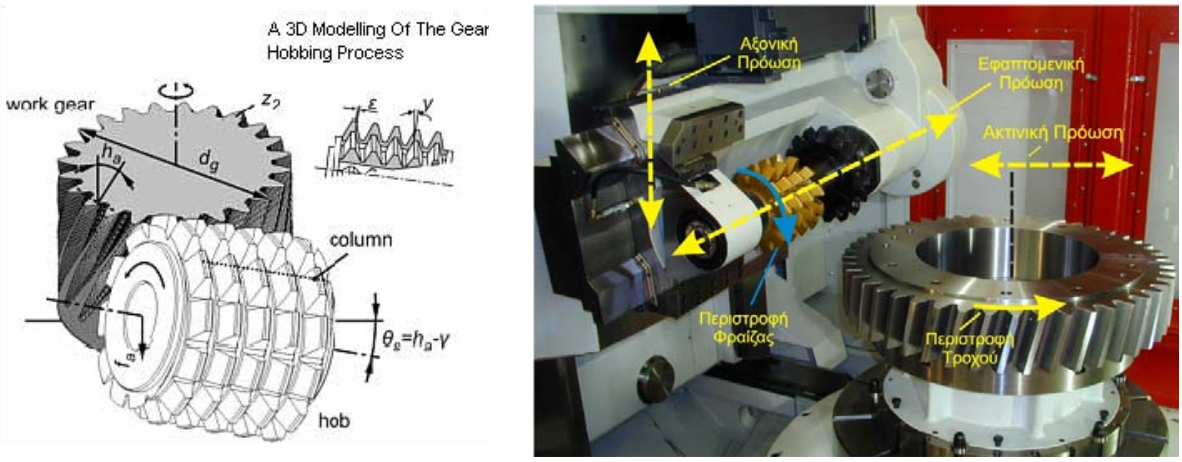
(β)

Σχήμα 2.2: (α) Κοπτικό εργαλείο κατά την κατεργασία οδόντωσης με φραιζάρισμα με κύλιση.
(β) Διάφορα είδη κοπτικών

Οι μηχανές κοπής χωρίζονται σε 2 μεγάλες κατηγορίες ανάλογα με το πώς γίνεται η κατεργασία του τεμαχίου. Στις κάθετες (vertical), όπου το κοπτικό διαμορφώνει το τεμάχιο κάθετα και στις οριζόντιες (horizontal) όπου αντίστοιχα η κατεργασία του τεμαχίου γίνεται οριζόντια. Οι περισσότερες μηχανές κοπής είναι κάθετες, ενώ οι οριζόντιες χρησιμεύουν στην κατεργασία μεγαλυτέρων κομματιών.

Η διαδικασία με την οποία επιτυγχάνεται η διαμόρφωση των οδοντώσεων είναι πολύπλοκη και απαιτεί μεγάλη ακρίβεια. Αρχικά το κατάλληλο κοπτικό εργαλείο τοποθετείται στην περιστρεφόμενη άτρακτο και το προς κατεργασία κομμάτι τοποθετείται στο σφιγκτήρα. Κατά τη δημιουργία της οδοντώσεως το κοπτικό εργαλείο καθώς και το προς κατεργασία τεμάχιο περιστρέφονται, έχοντας μια σχετική περιστροφική κίνηση μεταξύ προς. Έπειτα προσδίδεται μια πρόωση και ένα συγκεκριμένο βάθος κοπής στο κοπτικό εργαλείο. Καθώς λοιπόν το κοπτικό εργαλείο περιστρέφεται και προωθείται παράλληλα με την άτρακτο περιστροφής του τεμαχίου, το κατεργάζεται σταδιακά δημιουργώντας τελικά την οδόντωση, προς φαίνεται στο σχήμα 2.3.

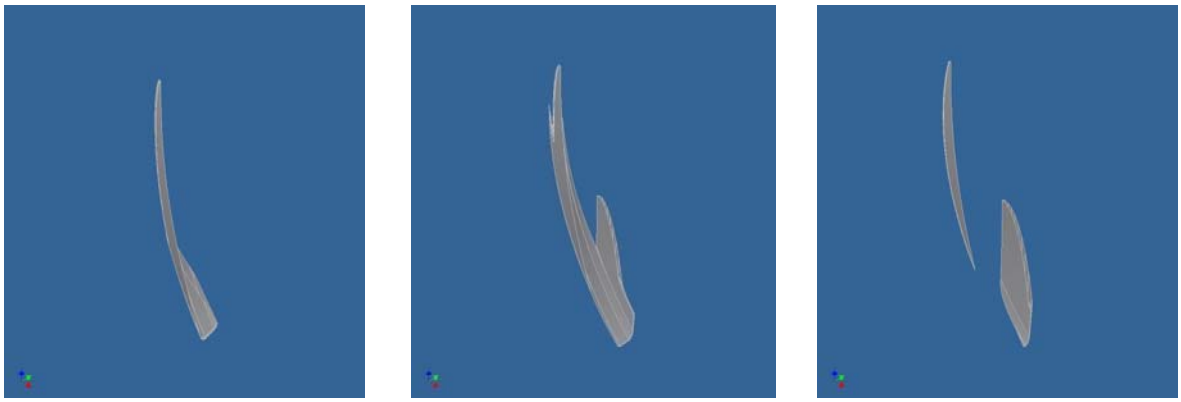
Η διαμόρφωση οδοντώσεων με φρεζάρισμα με κύλιση, διακρίνεται σε δύο κατηγορίες, σύμφωνα με τη σχετική κίνηση του κοπτικού με το προς κατεργασία κομμάτι. Στην ομόρροπη και στην αντίρροπη διαμόρφωση. Κατά την ομόρροπη, η διεύθυνση πρόωσης του κατεργαζόμενου τεμαχίου και η διεύθυνση περιστροφής του κοπτικού εργαλείου, εξεταζόμενες στη θέση κοπής, είναι ομόρροπες, ενώ κατά την αντίρροπη, είναι αντίρροπες.



(α) (β)

Σχήμα 2.3 (α): Απεικόνιση σχέσης μεταξύ κοπτικού εργαλείου και κατεργαζόμενου κομματιού. (β): Χώρος κατεργασίας κάθετης μηχανής κοπής γραναζιών Hoefler Gear Hobbing Machine

Όπως αναφέρεται στην προηγούμενη παράγραφο, η δημιουργία ενός αυλακιού στο προς κατεργασία τεμάχιο επιτυγχάνεται σταδιακά και όχι με ένα πέρασμα. Αρχικά κάποιο δόντι αφαιρεί λίγο υλικό, το επόμενο περισσότερο κ.ο.κ εωσότου δημιουργηθεί η αυλάκωση. Αυτή η διαδικασία έχει σαν αποτέλεσμα τη δημιουργία ενός υλικού αποβλίττου για κάθε μπάσιμο του εκάστοτε δοντιού κάτι που φαίνεται στο σχήμα 2.4. Σε κάθε δόντι που κόβει, αντιστοιχεί και μία θέση κοπής. Η θέση κοπής 0, ονομάζεται αυτή κατά την οποία το δόντι του κοπτικού εισέρχεται πλήρως στην αυλάκωση. Οι προηγούμενες θέσεις αριθμούνται με αρνητικές τιμές και οι επόμενες με θετικές. Εφόσον λοιπόν τα δόντια του κοπτικού δεν αφαιρούν την ίδια ποσότητα υλικού, είναι φυσιολογικό όλα τα απόβλιττα να μην είναι γεωμετρικά όμοια. Για παράδειγμα το απόβλιττο στο πρώτο μπάσιμο είναι πολύ μικρό σε αντίθεση με το απόβλιττο στη θέση κοπής 0. Μελετώντας λοιπόν αυτά τα απόβλιττα μπορούμε να αντλήσουμε πολλές πληροφορίες που έχουν σχέση με την καταπόνηση του εργαλείου κοπής καθώς και με τη διάρκεια ζωής του.



Σχήμα 2.4 Τρισδιάστατη απεικόνιση υλικού αποβλίττου μέσω του προγράμματος Autodesk Inventor στις θέσεις κοπής -9, 0 και 4 που παρήχθει ύστερα από προσομοίωση της κατεργασίας κοπής με ομόρροπη φορά.

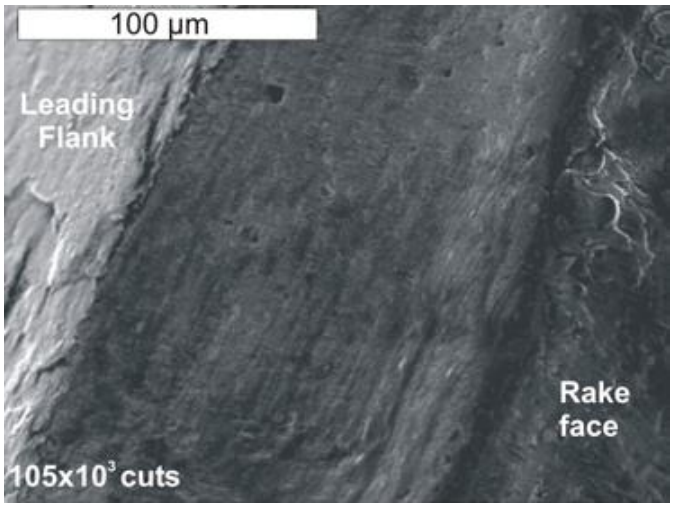
2.2 Φθορά των μετάλλων λόγω κόπωσης

Με τον όρο φθορά στα μέταλλα εννοείται είτε η απώλεια, είτε η μετατόπιση είτε η καταστροφή υλικού. Όταν η φθορά θεωρηθεί ως απώλεια υλικού, αυτό συμβαίνει περισσότερο από το σχηματισμό σωματιδίων παρά από την απώλεια μεμονωμένων ατόμων. Μια παρόμοια διαπίστωση μπορεί να γίνει όταν η φθορά θεωρηθεί ως μετατόπιση ή ακόμα ως καταστροφή με την έννοια ότι αυτές οι εκδηλώσεις της φθοράς είναι γενικά προάγγελοι της απώλειας υλικού και ότι η αρχική τους εμφάνιση είναι σε πολύ μεγαλύτερη κλίμακα από αυτή των μεμονωμένων ατόμων. Ένα σημαντικό σημείο που σχετίζεται με αυτή την παρατήρηση είναι ότι οι μηχανισμοί της φθοράς μπορούν να θεωρηθούν ως τυπικοί μηχανισμοί αποτυχίας υλικού που συμβαίνει πάνω η κοντά στην επιφάνεια. Οι μηχανισμοί της φθοράς συνήθως εκφράζονται με έννοιες όπως εύθραυστη θραύση, πλαστική παραμόρφωση, κόπωση και συγκολλητικές και συνεκτικές αποτυχίες στις συνδεδεμένες δομές. Στην περίπτωση της φθοράς η πολυπλοκότητα που συνδέεται σε καθέναν από τους μηχανισμούς γίνεται ακόμα πιο σύνθετη από το γεγονός ότι εμπλέκονται περισσότερα από ένα σώματα όπως επίσης και οι μοναδικές ιδιότητες και τα χαρακτηριστικά των επιφανειών.

Ανατρέχοντας σε βιβλία τριβολογίας διαπιστώνεται πως υπάρχει ένας υπερβολικά μεγάλος αριθμός μηχανισμών φθοράς. Ωστόσο, οι περισσότεροι τριβολόγοι τείνουν να διαιρούν τους βασικούς μηχανισμούς φθοράς σε μερικές κύριες υποκατηγορίες. Στη δεκαετία του ’50 αναπτύχθηκαν οι εξής κατηγορίες: Φθορά με σχηματισμό και λύση συγκολλητών δεσμών, φθορά απόξεσης, διάβρωση, φθορά από κόπωση και άλλες μικρότερες κατηγορίες. Ενώ η κατανόηση και ο ακριβής ορισμός αυτών των κατηγοριών έχει μεταβληθεί με την αυξημένη γνώση, αυτές οι κύριες κατηγορίες ισχύουν ακόμα και χρησιμοποιούνται από τους περισσότερους τριβολόγους.

Κατά την κατεργασία της κοπής οδοντώσεων με φραιζάρισμα με κύλιση, το κοπτικό εργαλείο καταπονείται από τις συνεχείς κρούσεις της οδοντώσεως με το προς κατεργασία τεμάχιο και υπόκειται σε φθορά λόγω κόπωσης όπως μπορούμε να διακρίνουμε στο σχήμα 2.5.

Η βασική έννοια της φθοράς λόγω κόπωσης είναι ότι με την επαναλαμβανόμενη ολίσθηση, κύλιση ή κρούση το υλικό στο χώρο της επιφάνειας υπόκειται σε μια κυκλική τάση. Με τη συνέχιση των κύκλων οι ρωγμές εξαπλώνονται και τελικά διατέμνουν την επιφάνεια και τέμνονται μεταξύ τους. Αυτό το δίκτυο των ρωγμών δημιουργεί τότε ελεύθερα σωματίδια που απομακρύνονται εύκολα από τη φθορά από την κίνηση και αυτό δημιουργεί φθορά. Αυτή η φθαρμένη επιφάνεια δέχεται επίσης κυκλικές τάσεις και η διαδικασία συνεχίζεται με αποτέλεσμα τη σταδιακή απώλεια υλικού από την επιφάνεια. Αυτός ο μηχανισμός φθοράς είναι πιο εμφανής σε περιπτώσεις κύλισης και κρούσης, όπου αναγνωρίζεται ως κύριος μηχανισμός. Στην περίπτωση της κύλισης και σε μικρότερο βαθμό στην κρούση, τα τοπολογικά χαρακτηριστικά της ρωγμής δείχνουν αρκετά καλά την αρχή της ρωγμής και τη διάδοσή της.



Σχήμα 2.5 Φθορά κόπωσης ενός κοπτικού δοντιού ύστερα από $105 * 10^3$ κοψίματα

Ένα συνηθισμένο χαρακτηριστικό της φθοράς λόγω κόπωσης και της κανονικής κόπωσης είναι η ύπαρξη αυτού που συχνά αναφέρεται ως περίοδο επώασης. Κατά τη διάρκεια αυτής της αρχικής περιόδου, σχηματίζονται ρωγμές και εξαπλώνονται στην επιφάνεια. Μερικές τοπολογικές αλλαγές μπορεί να είναι φανερές σε αυτή την περίοδο, όπως κάποιες ενδείξεις πλαστικής παραμόρφωσης. Ωστόσο δεν υπάρχει απώλεια υλικού από τη φθορά ή σχηματισμός ελευθέρων σωματιδίων. Μετά από αυτή τη φάση, μπορεί να έχουμε απώλεια υλικού με το σχηματισμό ελευθέρων σωματιδίων. Στην περίπτωση της κανονικής κόπωσης αυτή θα ισοδυναμούσε με το σπάσιμο του κομματιού και ο αριθμός των κύκλων μέχρι αυτό το σημείο θα ήταν η ανοχή στην κόπωση. Στην περίπτωση της φθοράς λόγω κόπωσης, η διαδικασία θα επαναλαμβανόταν πολλές φορές με αποτέλεσμα ένα όλο και πιο βαθύ ρήγμα. Επιπλέον υπάρχει κι άλλη διαφορά που πρέπει να επισημανθεί. Στη συμβατική κόπωση, τα περισσότερα υλικά έχουν ένα όριο αντοχής δηλαδή μια τιμή κάτω από την οποία δεν υφίσταται σπάσιμο. Στην περίπτωση της φθοράς λόγω κόπωσης δεν φαίνεται να υπάρχει ένα τέτοιο όριο τουλάχιστον όσον αφορά μακροσκοπικά φορτία και τάσεις. Για πρακτικές συνθήκες φορτίων, ανεξάρτητα από το πόσο μικρά είναι τα φορτία ή οι τάσεις ή η κρούση, θα έχει σαν αποτέλεσμα τη δημιουργία φθοράς λόγω κόπωσης.

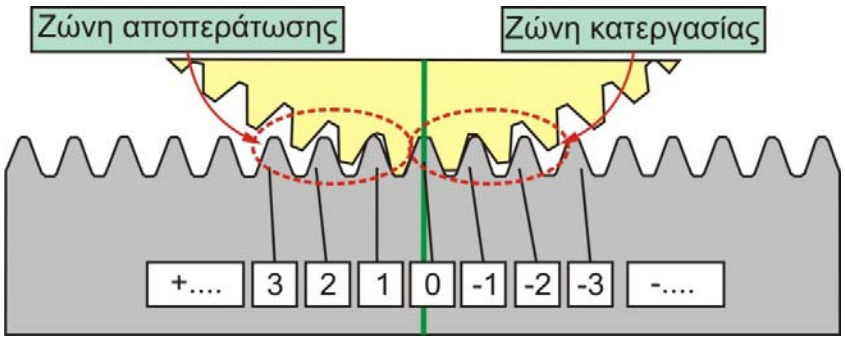
2.3 Πειραματικός προσδιορισμός της φθοράς κοπτικού εργαλείου κατά την κατεργασία οδοντώσεων με φραιζάρισμα με κύλιση.

Όλες οι βασικές κατεργασίες κοπής όπως η τόρνευση και το φραιζάρισμα, έχουν επωφεληθεί τα τελευταία χρόνια από καινοτόμες ανακαλύψεις, όπως νέα είδη υποστρωμάτων και επικαλύψεων στα κοπτικά εργαλεία, νέες μηχανές κοπής υψηλής ταχύτητας καθώς και νέα συστήματα λίπανσης. Η κοπή οδοντώσεων με φραιζάρισμα με κύλιση, αντιμετωπίζει πολλά προβλήματα στην ενσωμάτωση αυτών των καινοτομιών, λόγω των πολύ χρονοβόρων πειραμάτων φθοράς τα οποία μπορεί να διαρκέσουν ορισμένες βδομάδες, λόγω της υψηλής τιμής των μηχανών κοπής (άνω των 400000€), λόγω της υψηλής τιμής των εργαλείων κοπής (άνω των 1000€) και τέλος λόγω της καθυστέρησης παράδοσής τους. Επίσης τέτοιου είδους μηχανές κατεργασίας υπάρχουν μόνο σε μεγάλα εργοστάσια παραγωγής κάτι που κάνει την εκπόνηση των πειραμάτων δυσμενέστερη.

Στο εργαστήριο WZL του πανεπιστημίου TU Aachen, δημιουργήθηκε για πρώτη φορά μια νέα, εναλλακτική πειραματική μέθοδος για την κατεργασία οδοντώσεων με φραιζάρισμα με κύλιση, με το όνομα fly-hobbing. Αυτή η πειραματική μέθοδος είναι παρόμοια με την κανονική

κατεργασία που εξετάζεται απλά περιλαμβάνει μονάχα ένα κοπτικό δόντι. Λαμβάνοντας υπ' όψιν πως το κοπτικό εργαλείο έχει πολλά κοπτικά δόντια, κάθε ένα από αυτά αφαιρεί ένα διαφορετικό απόβλιττο. Έτσι η φθορά σε κάθε δόντι είναι διαφορετική από τα γειτονικά του. Όμως στο fly-hobbing, το ένα κοπτικό δόντι που υπάρχει συσσωρεύει όλη τη φθορά όλων των κοπτικών δοντιών, αφού με το που κόψει σε μία θέση κοπής, αμέσως κόβει και στην επόμενη. Το μικρό αυτό κοπτικό λοιπόν, δίνει τη δυνατότητα να παρατηρείται ευκολότερα η φθορά που έχει υποστεί μετά από μια ολοκληρωμένη προσομοίωση με τη βοήθεια ενός ψηφιακού μικροσκοπίου (SEM), σε αντίθεση με ένα κανονικό κοπτικό εργαλείο που έχει συνήθως διάμετρο 100mm και μήκος 200mm. Το fly-hobbing βοηθάει πολύ στην απόκτηση βιομηχανικών αποτελεσμάτων για τη φθορά του κοπτικού εργαλείου. Ωστόσο δεν μπορεί να πληροφορήσει για το ποια περιοχή δέχεται την περισσότερη καταπόνηση στη ζώνη επαφής, στην οποία συμμετέχουν πολλά δόντια και όχι μόνο ένα όπως στο πείραμα.

Η ζώνη επαφής κατά την κατεργασία οδοντώσεων με φραιζάρισμα με κύλιση, αποτελείται από δύο ζώνες όπως φαίνεται στο σχήμα 2.6. Μία ζώνη κατεργασίας, κατά την οποία το κοπτικό αφαιρεί μεγάλη ποσότητα αποβλίττου και μία ζώνη αποπεράτωσης, στην οποία το κοπτικό αφαιρεί πιο μικρή ποσότητα αποβλίττου και ουσιαστικά διαμορφώνει την οδόντωση. Όπως είναι φυσιολογικό τα δόντια που ανήκουν στην ζώνη κατεργασίας είναι και τα πιο καίρια καθώς υπόκεινται σε μεγαλύτερη καταπόνηση. Ωστόσο η χρήση καρβιδιούχων κοπτικών, αντιτίθεται σε αυτό το συλλογισμό, καθώς υπάρχουν διαδικασίες κοπής που οδηγούν σε πρόωρη αποτυχία των κοπτικών δοντιών που ανήκουν στη ζώνη αποπεράτωσης λόγω κατεργασίας πολύ λεπτών αποβλίττων. Άρα γίνεται επιτακτική η ανάγκη για την εύρεση μιας πειραματικής μεθόδου που θα μπορεί να υπολογίσει τη φθορά και στις δύο ζώνες κοπής ούτως ώστε να μπορεί να φανεί ποιο δόντι αποτελεί το αδύναμο σημείο του κοπτικού.

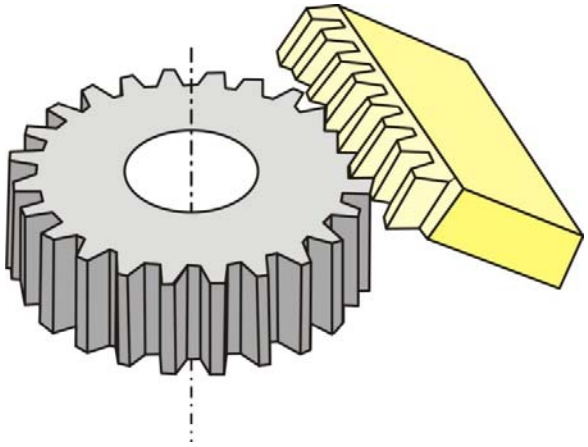


Σχήμα 2.6 Θέσεις κοπής και ζώνες κοπής κοπτικού εργαλείου

Άλλο ένα πρόβλημα της τεχνολογίας του fly-hobbing είναι πως το κόστος πραγματοποίησης ενός πειράματος είναι πολύ υψηλό και η διαθεσιμότητα μηχανών κατεργασίας είναι περιορισμένη. Μία νέα μέθοδος προτάθηκε το 2006 για μείωση του κόστους της πειραματικής αυτής τεχνολογίας. Αυτή η ανανεωμένη μέθοδος ήταν μια μεταφορά του fly-hobbing σε μια φραίζα 5 διαστάσεων η οποία είναι ευκολότερα διαθέσιμη καθώς βρίσκεται σε πολλά ερευνητικά κέντρα και όχι μόνο στη βιομηχανία. Ωστόσο και πάλι τα αποτελέσματα των πειραμάτων βασίζονται στην καταγραφή της φθοράς σε ένα μόνο κοπτικό δόντι οπότε παραμένει άλυτο ακόμα το πρόβλημα του υπολογισμού της φθοράς στην εκάστοτε ζώνη κοπής.

Τη λύση σε αυτό το πρόβλημα ήρθε να δώσει μια νέα πειραματική μέθοδος υπολογισμού φθοράς το 'flute-hobbing'. Η κινηματική στο flute-hobbing είναι ίδια με αυτή του gear hobbing που έχει ήδη περιγραφεί στην παράγραφο 2.1. Όπως βλέπουμε στο σχήμα 2.7, η διαφορά

της πειραματικής αυτής μεθόδου με την κανονική διαδικασία διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση, είναι πως το κοπτικό εργαλείο αποτελείται από μονάχα μία σειρά κοπτικών δοντιών. Αυτή η σειρά κοπτικών δοντιών έχει ακριβώς την ίδια γεωμετρία και είναι από το ίδιο υλικό με το κανονικό κοπτικό εργαλείο οπότε τελικά έχουμε την παραγωγή μιας κανονικής οδοντώσεως. Φυσικά οι συνθήκες κοπής (ιδιαίτερα η πρόωση ανά περιστροφή του προς κοπή γραναζιού) και ο χρόνος κατεργασίας διαφέρουν από την κανονική κατεργασία καθώς μία σειρά μονάχα κοπτικών δοντιών δεν μπορεί να έχει το ίδιο αποτέλεσμα με ένα κανονικό εργαλείο.



Σχήμα 2.7 Κοπτικό εργαλείο στην πειραματική διαδικασία του Flute-Hobbing

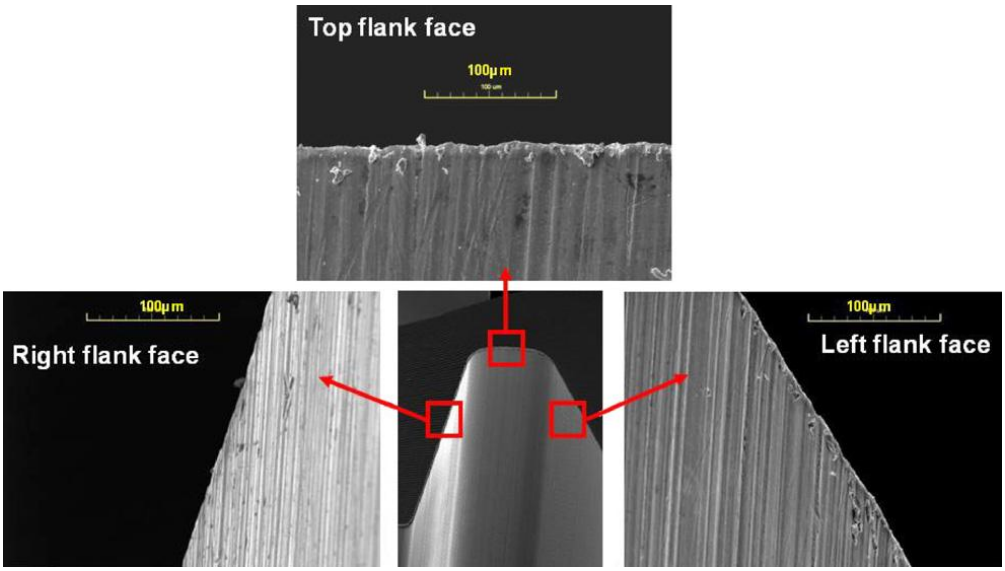
Χαρακτηριστικό του flute-hobbing είναι πως δεν υπάρχει μετατόπιση στον Y άξονα ανάμεσα στην κατεργασία δύο γραναζιών αντίθετα με την κανονική κατεργασία διαμόρφωσης οδοντώσεων. Στην παραγωγή, αυτή η μετατόπιση χρησιμοποιείται για να υπάρχει ομοιόμορφη κατανομή της φθοράς σε όλα τα κοπτικά δόντια του εργαλείου. Αντιθέτως, το αντικείμενο του flute-hobbing είναι να προσδιορίσει το συγκεκριμένο μηχανισμό φθοράς σε διάφορες θέσεις κοπής, έτσι κάθε δόντι πρέπει να διατηρεί συγκεκριμένη θέση καθ' όλη τη διάρκεια του πειράματος. Έτσι το μήκος της σειράς δοντιών που χρησιμοποιείται σαν κοπτικό πρέπει να είναι απλά λίγο μεγαλύτερο από το μήκος της επαφής του κοπτικού με το προς κατεργασία τεμάχιο. Για παράδειγμα, όπως φαίνεται στο σχήμα 2.6, πέντε δόντια είναι σε επαφή οπότε χρειάζονται μονάχα ένα με δύο δόντια από κάθε πλευρά επιπλέον. Για να διατηρείται λοιπόν συνεχώς η ίδια θέση του κοπτικού το μεσαίο δόντι της σειράς αυτής δοντιών βρίσκεται πάντα στη θέση κοπής 0 όπως φαίνεται στο σχήμα 2.6. Επίσης και πάλι στο σχήμα 2.6 φαίνεται πως τα δόντια που ανήκουν στη ζώνη κατεργασίας έχουν αρνητική θέση κοπής ενώ αυτά που ανήκουν στη ζώνη αποπεράτωσης έχουν θετική τιμή.

Το flute-hobbing μπορεί να πραγματοποιηθεί σε οποιαδήποτε μηχανή κατεργασίας. Το μόνο που είναι απαραίτητο για την πραγματοποίηση του είναι η δημιουργία του κατάλληλου κοπτικού, δηλαδή της σειράς δοντιών που φαίνεται στο σχήμα 2.7. Επίσης το πείραμα μπορεί να μεταφερθεί σε φραίζα πέντε αξόνων κάτι που είναι ευνοϊκό αφού όπως αναφέρθηκε η πρόσβαση σε πενταξονική φραίζα είναι ευκολότερη από την πρόσβαση σε μηχανή διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση.

Για την εκκίνηση ενός πειράματος χρειάζεται να καθοριστούν δύο μεγέθη. Η ταχύτητα κοπής (δηλαδή το πόσο γρήγορα περιστρέφεται το κοπτικό) και η ταχύτητα πρόωσης. Στην περίπτωση του flute-hobbing όσο και του fly-hobbing η ταχύτητα κοπής διαφέρει από αυτή

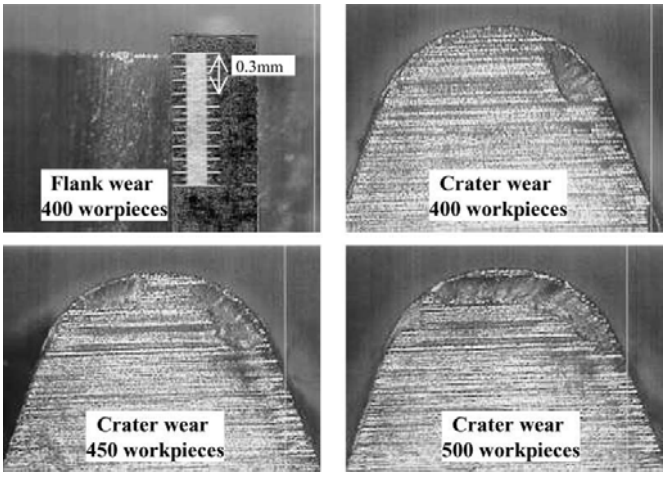
της κανονικής διαδικασίας κοπής με κανονικό κοπτικό εργαλείο και υπολογίζεται στην εκάστοτε περίπτωση.

Σε πειράματα που διεξήχθησαν το 2008 στο πανεπιστήμιο της Lyon, στην εθνική σχολή μηχανολόγων μηχανικών, εφαρμόστηκε πρώτη φορά η μέθοδος του flute-hobbing. Τα πειράματα έδωσαν πληροφορίες για τη φθορά του κοπτικού όπως φαίνεται στο σχήμα 2,8



Σχήμα 2.8 Παρουσίαση φθοράς ενός κοπτικού δοντιού ύστερα από διεξαγωγή πειραμάτων με τη διαδικασία flute-hobbing

Παίρνοντας την πληροφορία για τη φθορά στην οποία υπόκειται ένα κοπτικό εργαλείο ύστερα από συγκεκριμένο αριθμό κοπών σε συγκεκριμένες συνθήκες, μπορεί να γίνει σύγκριση σε διαφορετικά κοπτικά. Ο βαθμός της φθοράς, μαζί με το κόστος παραγωγής και την ποιότητα κατασκευής είναι λοιπόν οι τρεις παράγοντες που επηρεάζουν την επιλογή του κατάλληλου κοπτικού.



Σχήμα 2.9 Σταδιακή φθορά κοπτικού δοντιού ύστερα από κατεργασία 400, 450 και 500 κομματιών αντίστοιχα

2.4 Βελτιστοποίηση του κοπτικού εργαλείου

Η διεξαγωγή πειραμάτων για τον υπολογισμό της φθοράς του κοπτικού εργαλείου που περιγράφηκε στην προηγούμενη παράγραφο γίνεται για τη βελτιστοποίηση του κοπτικού. Τα βασικά χαρακτηριστικά που μπορούν να επιδεχθούν τροποποιήσεις και να καταστήσουν το εργαλείο πιο αποδοτικό είναι τρία. Η προετοιμασία της άκρης του δοντιού, το υλικό του υποστρώματος του κοπτικού και το υλικό επικάλυψης του δοντιού κοπής.

Η προετοιμασία της άκρης του κοπτικού δοντιού δεν απασχολούσε τους κατασκευαστές κοπτικών τα προηγούμενα χρόνια επειδή δεν υπήρχαν κατάλληλες κατασκευαστικές μέθοδοι που να οδηγούν σε ακριβή αποτελέσματα και δεν υπήρχε σύστημα ελέγχου για να προσδώσει τα κατάλληλα χαρακτηριστικά που ο κατασκευαστής επιθυμούσε. Πολλές μελέτες έδειξαν πως με τη χρήση της κατάλληλης ακτίνας στην άκρη του κοπτικού δοντιού μπορεί να μειωθεί αισθητά η φθορά του στο χρόνο. Στις περισσότερες αυτές μελέτες τα κοπτικά που χρησιμοποιούνταν ήταν είτε κεραμικά είτε από καρβίδια και τα πειράματα γινόταν σε φραίζα.

Στις μέρες μας, όπου η τεχνολογία έχει κάνει τεράστια άλματα, η κατεργασία του κοπτικού δοντιού είναι μια πολύ ακριβής διαδικασία. Σε συνδυασμό με το κατάλληλο λογισμικό, γίνεται να προγραμματιστεί με μεγάλη ακρίβεια η τελική μορφή που θέλουμε να έχει το κοπτικό και να δοθεί η επιθυμητή γεωμετρία στην άκρη του δοντιού.

Επίσης μεγάλη ανάπτυξη έχει παρατηρηθεί στη μετρολογία στον κλάδο του gear-hobbing. Έχουν ήδη κατασκευαστεί μηχανήματα επιθεώρησης των διαστάσεων και της γεωμετρίας του κοπτικού, ένα από τα οποία φαίνεται στο σχήμα 2.10. Πρόκειται για τη μηχανή μέτρησης Gleason-M&M's η οποία δίνει ακριβής πληροφορίες για το μήκος, το προφίλ και το πάχος του κάθε δοντιού, το μήκος και τη διάμετρο όλου του κοπτικού και πολλές άλλες μετρήσεις. Σε συνεργασία με το κατάλληλο λογισμικό, η μηχανή μέτρησης εκτελώντας αρχικά μια σειρά από τεστ, εκτυπώνει στον υπολογιστή τα αποτελέσματα των μετρήσεων.

Σχήμα 2.10 Μηχάνημα μέτρησης διαστάσεων κοπτικού εργαλείου Gleason-M&M

Το άλλο τώρα χαρακτηριστικό που επιδέχεται τροποποιήσεως είναι το υλικό επικάλυψης του κάθε δοντιού. Κατά τη διαδικασία της διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση, όπως έχει ήδη αναφερθεί, υπάρχει κρούση μεταξύ δύο μετάλλων. Αυτό έχει σαν αποτέλεσμα την αύξηση της θερμοκρασίας κάτι που έως τώρα αντιμετωπιζόταν με τη χρήση ψυκτικού υγρού. Όμως μελέτες έδειξαν πως η χρήση των ψυκτικών υγρών μπορεί να αποβεί μοιραία για το περιβάλλον. Έτσι λόγω της αποφυγής της μόλυνσης του περιβάλλοντος γίνεται μια προσπάθεια η κατεργασία διαμόρφωσης οδοντώσεων να γίνεται υπό ξηρές συνθήκες. Μία λύση σε αυτό το πρόβλημα ήρθε να δώσει η επικάλυψη των κοπτικών δοντιών με το κατάλληλο υλικό.

Οι κατασκευαστές κοπτικών εργαλείων χρησιμοποιούν κυρίως συμβατικά υποστρώματα όπως M35 (SM), σε συνδυασμό με επικάλυψη (PVD) TiN. Η έρευνες που γίνονται από τους

κατασκευαστές εργαλείων κοπής για τη δημιουργία κοπτικών για ξηρές συνθήκες κοπής περιλαμβάνουν τη χρήση σκληρότερων και αυτό-λιπαινόμενων επικαλύψεων σε συνδυασμό με σκληρότερα υποστρώματα όπως ημιτετηγμένο ταχυχάλυβα (HSS, high speed steel) ή καρβίδια. Στις μέρες μας η τεχνολογία των καρβιδίων δεν είναι αξιόπιστη, λόγω της δυσχέρειας στην επανάληψη πειραματικών μετρήσεων.

Η εφαρμογή του ημιτετηγμένου HSS, είναι έως τώρα το πιο υποσχόμενο υπόστρωμα και αποτελεί την καλύτερη επιλογή ανάμεσα σε συμβατικούς HSS και τα καρβίδια. Η υψηλή περιεκτικότητα του σε βανάδιο του προσδίδει μεγάλη σκληρότητα. Έτσι λοιπόν μία πολύ καλή και ομογενής δομή, με μεγάλη προσθήκη κοβάλτιου επιφέρει μεγάλη σκληρότητα και αντίσταση στη θραύση.

Πέρα από τα υποστρώματα οι κατασκευαστές ψάχνουν και για νέες αποδοτικότερες επικαλύψεις των κοπτικών. Η (Ti,Al)N και η MoS2 επικαλύψεις, είναι πιθανώς ορισμένες από τις πιο ενδιαφέρουσες στην αγορά υψηλής ταχύτητας διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση.

Όπως αναφέρθηκε και πριν, κατά την κατεργασία οδοντώσεων δημιουργούνται υψηλές θερμοκρασίες και υψηλές μηχανικές τάσεις, κάτι που επιφέρει όλων των ειδών φθορές στα κοπτικά δόντια όπως εκδορές, προσκόλληση υλικού, οξείδωση κ.α. Το (Ti,Al)N είναι γνωστό για την εφαρμογή σε υψηλής ταχύτητας κατεργασία κοπής οδοντώσεων, λόγω της υψηλής σκληρότητας που διατηρεί σε υψηλές θερμοκρασίες έως 1000 βαθμούς κελσίου. Επίσης έχει τη δυνατότητα δημιουργίας ενός στρώματος Al2O3 κατά την οξείδωσή του σε υψηλές θερμοκρασίες και όσο η θερμοκρασία αυξάνεται, το στρώμα που προκύπτει είναι παχύτερο. Έτσι κατατάσσεται ως μία από τις καλύτερες επικαλύψεις των κοπτικών.

Σχήμα 2.11 Κοπτικά εργαλεία με διαφορετικές επικαλύψεις

3. ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

3.1 Εισαγωγικά για τον κώδικα

Όπως αναφέρεται στην εισαγωγή, κύριος σκοπός της εργασίας αυτής είναι να προσδιοριστεί πώς η κρούση του παραγόμενου αποβλίττου σε κάθε θέση κύλισης επηρεάζει τη φθορά του κοπτικού εργαλείου. Αυτό επιτυγχάνεται με το συνδυασμό του προγράμματος Autodesk Inventor το οποίο πρόκειται για ένα τρισδιάστατο σχεδιαστικό πρόγραμμα και της γλώσσας προγραμματισμού Visual Basic for Applications. Φτιάχνοντας τον κατάλληλο κώδικα, γίνεται η επεξεργασία του τρισδιάστατου αποβλίττου και εξάγονται τα επιθυμητά αποτελέσματα τα οποία αποθηκεύονται σε ένα αρχείο κειμένου (text). Τέλος γίνεται μια απεικόνιση του αποτελέσματος με τη βοήθεια του προγράμματος MatLab.

Σχήμα 3.1 Λογότυπα των προγραμμάτων Visual Basic και Autodesk Inventor

Τα δεδομένα που χρειάζονται για να ξεκινήσει ο κώδικας είναι το τρισδιάστατο απόβλιττο σε αρχείο μορφής του προγράμματος Autodesk Inventor και το προφίλ του κοπτικού δοντιού που πραγματοποιεί την κοπή, επίσης σε αρχείο μορφής του προγράμματος Autodesk Inventor. Αυτά τα δεδομένα έρχονται από μια προσομοίωση της διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση στο πρόγραμμα Autodesk Inventor που έχει ήδη πραγματοποιηθεί. Ουσιαστικά ο κώδικας που δημιουργείται σε αυτή τη διπλωματική με τα αποτελέσματα που μας δίνει, είναι ένα κομμάτι μιας μεγαλύτερης εργασίας στην οποία προσομοιώνεται ολόκληρη η διαδικασία της κοπής οδοντώσεων. Με τα αποτελέσματα που εξάγονται σε αυτήν τη διπλωματική λαμβάνεται μια επιπλέον πληροφορία για το ποιες θέσεις του κάθε δοντιού του κοπτικού καταπονούνται περισσότερο.

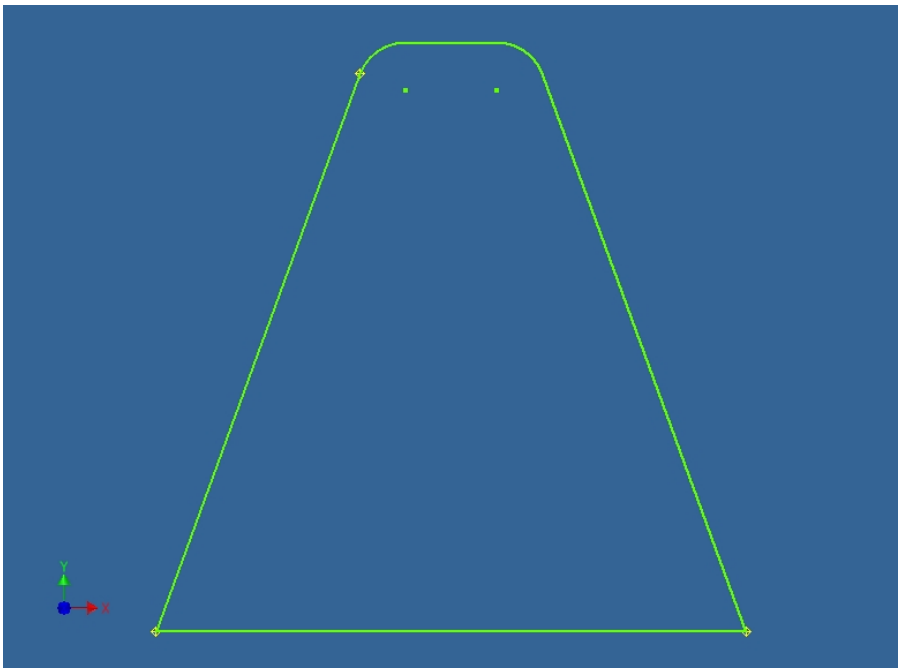
Ο κώδικας που δημιουργείται είναι περίπου 1800 γραμμές και αρκετά πολύπλοκος σε ορισμένα σημεία. Εδώ θα γίνει μια προσπάθεια επεξήγησης του, όσο εκτενέστερα γίνεται. Επίσης είναι κατάλληλα διαμορφωμένος έτσι ώστε να λειτουργεί σε οποιοδήποτε απόβλιττο της κατεργασίας διαμόρφωσης οδοντώσεων με φραιζάρισμα με κύλιση. Για να γίνει όμως μια καλύτερη περιγραφή του κώδικα, μαζί με την ανάλυση των εντολών θα γίνεται και μία οπτική απεικόνιση των αποτελεσμάτων καθώς ‘τρέχει’ ο κώδικας. Οι εικόνες που παρουσιάζονται κατά την ανάλυση του, προκύπτουν καθώς τρέχει ο κώδικας για το απόβλιττο που προκύπτει κατά την ομόρροπη διαμόρφωση οδοντώσεως με φραιζάρισμα με κύλιση στη θέση κοπής 0 γιατί το απόβλιττο αυτό έχει πολύ υλικό.

Τέλος για επιπλέον κατανόηση του αναγνώστη όπου κρίνεται σκόπιμο θα υπάρχει παραπομπή στην γραμμή του κώδικα που περιγράφεται εκείνη τη στιγμή.

3.2 Κύρια συνάρτηση

Ο κώδικας ξεκινάει προτρέποντας το χρήστη να επιλέξει ποιο απόβλιττο θέλει να επεξεργαστεί. Με έναν έλεγχο ο κώδικας συνεχίζει μόνο όταν υπάρχει αυτό το απόβλιττο. Το αρχείο πρέπει να βρίσκεται σε μια συγκεκριμένη τοποθεσία μέσα στον ηλεκτρονικό υπολογιστή. Στη συγκεκριμένη περίπτωση ο φάκελος που περιέχει όλες τις πληροφορίες είναι ο '`C:\Gear Hobbing\omoropo`'. Αφού λοιπόν ορίζεται ποιο απόβλιττο έχει επιλεγεί, ο κώδικας δημιουργεί έναν φάκελο με το όνομα του αποβλίττου στον οποίο θα αποθηκεύονται όλες οι πληροφορίες με την εντολή '`MkDir ("C:\Gear Hobbing\omoropo\Chip" & ch & "")`' όπου στη μεταβλητή '`ch`' έχει αποθηκευτεί ο αριθμός του επεξεργαζόμενου αποβλίττου. Στη συγκεκριμένη περίπτωση που τρέχει ο κώδικας παράλληλα με την ανάλυση του, όταν ζητείται επιλέγεται το απόβλιττο 0.

Τώρα ορίζεται ως τρέχον αρχείο για επεξεργασία το αρχείο που περιέχει το προφίλ του κοπτικού δοντιού όπως φαίνεται στο σχήμα 3.2. Αυτό γίνεται με την εντολή '`Set oPartDoc1 = ThisApplication.Documents.Open("C:\Gear Hobbing\omoropo\Profiles\Profil" & ch & ".ipt", True)`'. Αυτή η εντολή θα χρησιμοποιηθεί αρκετά στη συνέχεια, όποτε πρέπει να οριστεί ποιο είναι το αρχείο προς επεξεργασία. Μια εντολή που επίσης χρησιμοποιείται όταν ορίζεται ένα νέο αρχείο προς επεξεργασία είναι η '`Set oCompDef1 = oPartDoc1.ComponentDefinition`' η οποία θέτει τη μεταβλητή '`oCompDef1`' ως εργαλείο για την επεξεργασία κάθε ορίσματος του τρέχοντος αρχείου. Με την '`Set osketch1 = oCompDef1.Sketches.Item(1)`' ορίζεται σαν σκίτσο προς επεξεργασία το πρώτο και μοναδικό σκίτσο που υπάρχει στο αρχείο το οποίο περιλαμβάνει το προφίλ του δοντιού.

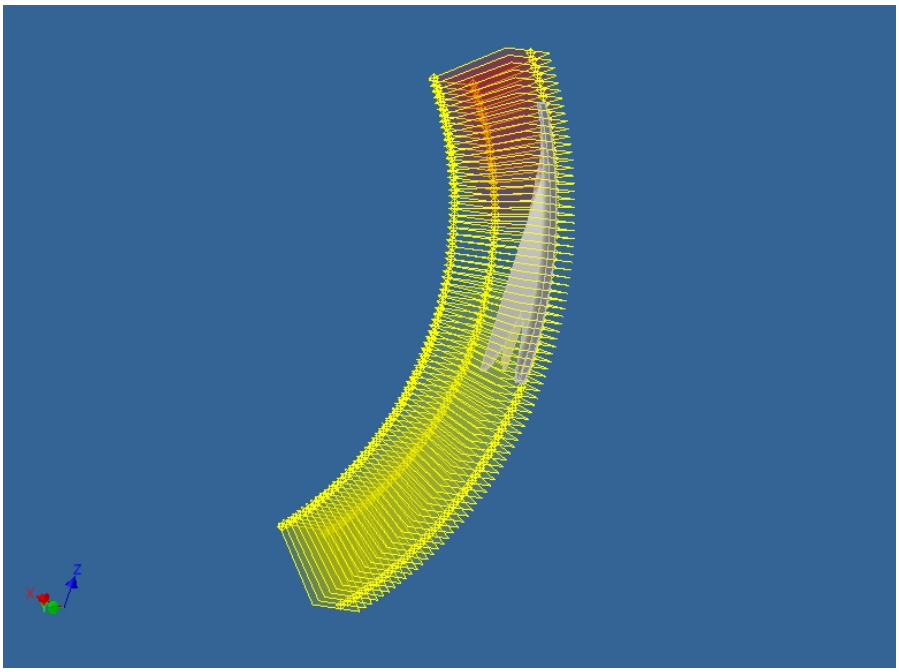


Σχήμα 3.2 Προφίλ κοπτικού δοντιού

Έπειτα με μια επαναληπτική διαδικασία σαρώνονται όλα τα στοιχεία του σκίτσου (SketchEntities) και με έναν έλεγχο, όταν το στοιχείο πρόκειται για τόξο (Arc) αποθηκεύεται στη μεταβλητή '`mikos`' το μήκος της ακτίνας του τόξου (γραμμή 35). Αυτό χρησιμεύει αργότερα και γίνεται για να υπάρχει μια τιμή που να συσχετίζεται με το δόντι που

χρησιμοποιείται και όχι μια αυθαίρετη τιμή. Η επαναληπτική διαδικασία θα συναντήσει δύο τόξα όπως μπορούμε να δούμε στο σχήμα 3.2 τα οποία είναι όμοια, τέσσερεις ευθείες και οκτώ σημεία.

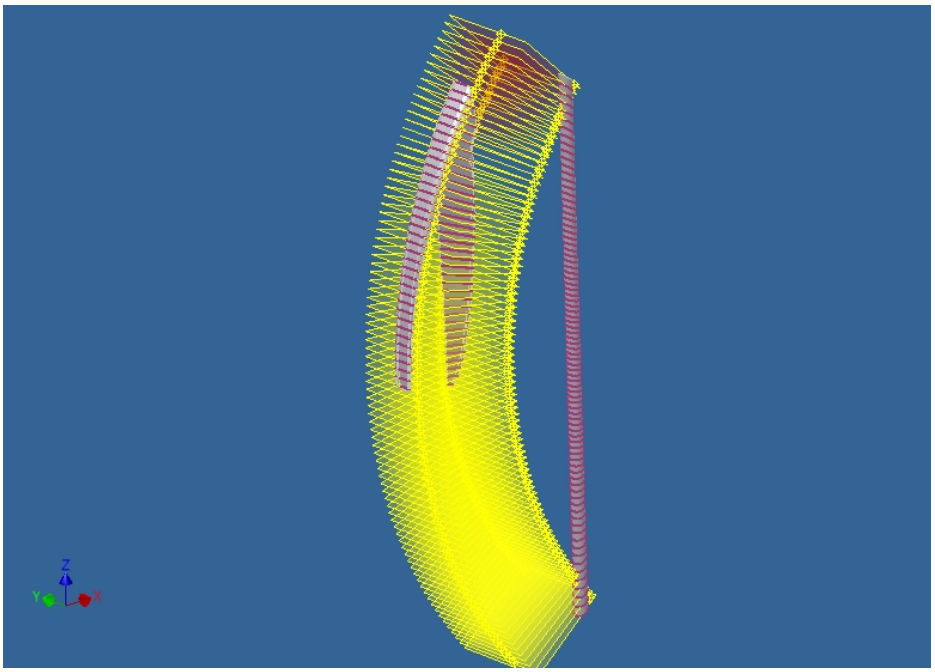
Έπειτα κλείνει το αρχείο του προφίλ του δοντιού και ανοίγει το αρχείο που περιέχει το τρισδιάστατο απόβλιττο όπως φαίνεται στο σχήμα 3.3. Βλέπουμε πως κατά μήκος του αποβλίττου έχουν δημιουργηθεί επίπεδα που το τέμνουν ανά μία μοίρα.



Σχήμα 3.3 Τρισδιάστατη απεικόνιση αποβλίττου με επίπεδα να το τέμνουν ανά μία μοίρα.

Αποθηκεύονται στη μεταβλητή '`iPlanes1`' (γραμμή 52) όλα τα επίπεδα που έχουν απόσταση μια μοίρα και φαίνονται στο σχήμα 3.3, μετρώντας όλα τα επίπεδα που υπάρχουν στο τρέχον αρχείο αφαιρώντας τέσσερα, καθώς τρία είναι τα αυτά που ορίζονται από τους άξονες (επίπεδα XY, XZ και YZ) και λόγω του ότι η μέτρηση των επιπέδων ανά 1^{o} ξεκινάει από το 0, αφαιρείται το άλλο.

Τώρα σκοπός είναι να προσδιοριστεί σε ποια επίπεδα υπάρχει τομή με το απόβλιττο, δηλαδή ποια επίπεδα 'βρίσκουν' υλικό. Με μια πρώτη σκέψη θα μπορούσαν να σαρωθούν ένα-ένα τα επίπεδα, να διαπιστωθεί σε ποια δεν υπάρχει καθόλου υλικό και να κρατηθούν τα υπόλοιπα. Όμως εάν σαρωθεί ένα επίπεδο και δεν βρεθεί καθόλου υλικό τότε ο κώδικας σταματάει γιατί ενώ του έχουμε πει πως περιμένει να βρει υλικό, τελικά κάτι τέτοιο δε συμβαίνει. Αυτό το πρόβλημα αντιμετωπίζεται με την εξής διαδικασία (γραμμή 55-101). Επιλέγεται το πρώτο και το τελευταίο επίπεδο και σχεδιάζονται δύο κύκλοι με ακτίνα '`mikos`*2'. Έπειτα ενώνονται αυτοί οι δύο κύκλοι με προσθήκη υλικού και με μία επαναληπτική διαδικασία προβάλλονται σε κάθε επίπεδο το υλικό που συναντάει ο κώδικας όπως φαίνεται στο σχήμα 3.4.

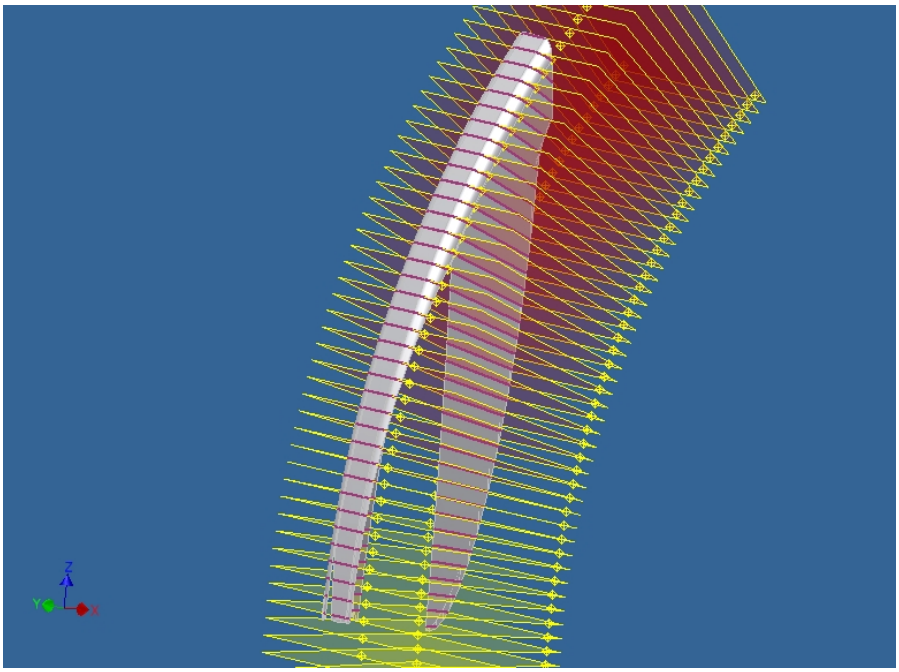


Σχήμα 3.4 Απόβλιττο με βοηθητικό κύλινδρο

Άρα τώρα κάθε επίπεδο έχει τουλάχιστον δύο Sketch Entities, τον κύκλο και την ακτίνα του. Τώρα λοιπόν γίνεται μια διαδικασία για να βρεθούν τα επίπεδα στα οποία υπάρχει υλικό απόβλιττο (γραμμή 104-131).

Εφαρμόζεται μια επαναληπτική διαδικασία η οποία ξεκινά από το επίπεδο 0 (`ChipPlane0`). Σαρώνει το επίπεδο και αν βρει μονάχα δύο sketch entities σημαίνει πως στο επίπεδο αυτό δεν υπάρχει τομή με το απόβλιττο. Ταυτόχρονα ένας μετρητής '`m`' αυξάνεται κατά μια μονάδα. Όταν τώρα στο επίπεδο υπάρχει υλικό, η διαδικασία σταματάει και ο μετρητής έχει πάρει μια τιμή που υποδηλώνει ποια επίπεδα δεν τέμνουν το απόβλιττο. Εφαρμόζεται η ίδια διαδικασία από πάνω, δηλαδή ο κώδικας ξεκινάει από το τελευταίο επίπεδο και κατεβαίνει προς τα κάτω μέχρι ένα επίπεδο να τέμνει το απόβλιττο και παρομοίως αποθηκεύονται σε ένα μετρητή '`n`' τα επίπεδα που δεν υπάρχει υλικό. Άρα λοιπόν εφόσον είναι γνωστά ποια επίπεδα δε συναντάνε απόβλιττο, είναι γνωστά και ποια επίπεδα τέμνουν το απόβλιττο. Όπως φαίνεται στον κώδικα το πρώτο επίπεδο που συναντάει υλικό θα είναι πάντα το '`m-1`' και το τελευταίο θα είναι το '`iPlanes1-(n+1)`'. Το αρχείο αυτό δεν είναι αναγκαίο πλέον, όμως δε γίνεται απλά να κλείσει γιατί το πρόγραμμα ρωτά εάν πρέπει να αποθηκευτούν οι αλλαγές που έχουν γίνει. Έτσι για να μη σταματάει η ροή του κώδικα, το αρχείο αποθηκεύεται με την ονομασία 'Temp' και έπειτα κλείνει (γραμμή 135).

Τώρα εφόσον υπάρχει η πληροφορία για το ποια επίπεδα τέμνουν το απόβλιττο, ο κώδικας ανοίγει εκ νέου το αρχείο με το απόβλιττο και με μία επαναληπτική διαδικασία προβάλλει σε κάθε επίπεδο το υλικό του αποβλίττου που αυτό συναντάει (γραμμή 137-151) όπως φαίνεται και στο σχήμα 3.5. Τέλος αποθηκεύεται το αρχείο με το όνομα '`ChipProjected`' και κλείνει (γραμμή 154).



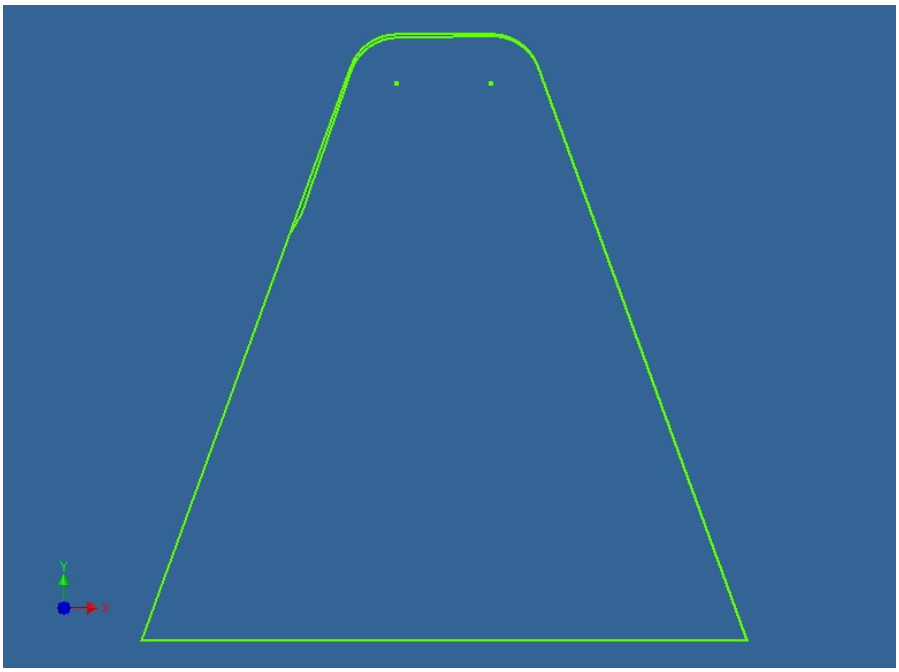
Σχήμα 3.5 Τρισδιάστατη απεικόνιση αποβλίττου με προβολές σε κάθε επίπεδο που το τέμνει

Έπειτα στην κύρια συνάρτηση καλούνται οι συναρτήσεις `CopySketch`, `Fixing` και `ChipCollision` μέσα σε επαναληπτικούς βρόγχους για όλα τα επίπεδα, καθώς και η συνάρτηση `MakeToothPerigram`. Η ανάλυση των συναρτήσεων αυτών γίνεται παρακάτω.

3.3 Συναρτήσεις CopySketch, Fixing και MakeToothPerigram

Η συνάρτηση `CopySketch` έχει σκοπό να αποτυπώσει σε ένα αρχείο το προφίλ του κοπτικού δοντιού μαζί με την προβολή του αποβλίττου σε κάθε θέση κύλισης. Παίρνει δύο ορίσματα που είναι δύο ακέραιοι αριθμοί. Τον '`f`' που αντιπροσωπεύει τη θέση κύλισης και τον '`ch`' που αντιπροσωπεύει τη θέση κοπής.

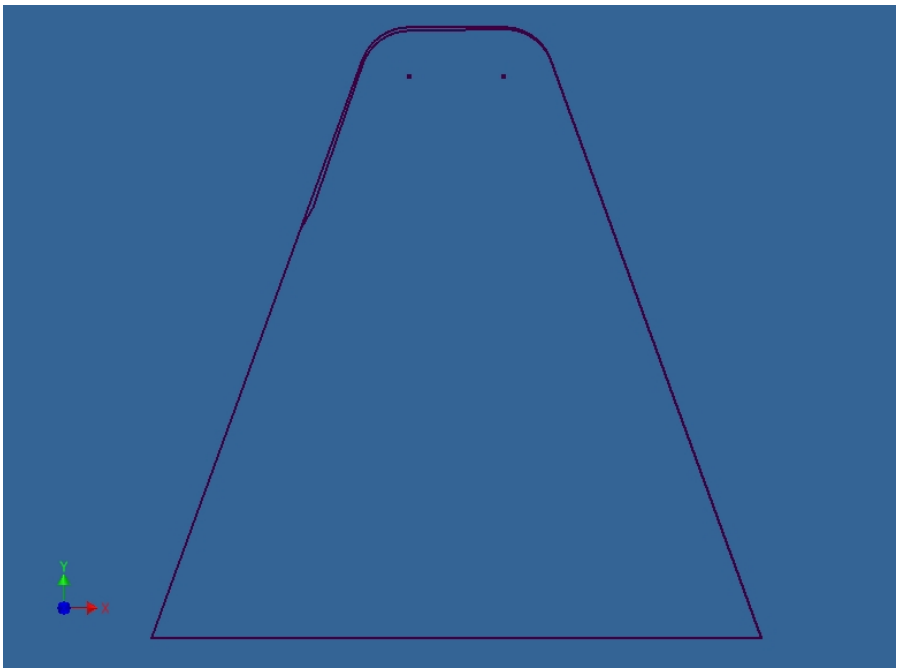
Η συνάρτηση λειτουργεί ως εξής. Αρχικά ανοίγει το αρχείο με το προφίλ του δοντιού (γραμμή 176). Στη συνέχεια αντιγράφει το προφίλ του κοπτικού δοντιού (γραμμές 178-194), κλείνει το τρέχον αρχείο, δημιουργεί ένα καινούριο και κάνει επικόλληση το προφίλ που αντέγραψε πριν. Μετά σώνει το νέο αυτό αρχείο με το όνομα '`ChipCollision`' (γραμμή 211) και κλείνει τα δύο ανοικτά αρχεία. Στη συνέχεια με παρόμοιο τρόπο γίνεται και η αντιγραφή – επικόλληση της προβολής του αποβλίττου σε κάθε θέση κύλισης. Ο κώδικας ανοίγει πρώτα το αρχείο '`ChipProjected" & ch & ".ipt`' που είδαμε στο σχήμα 3.5 (γραμμή 218) και με την εντολή '`Set oSketchToCopy = oDef.Sketches.Item(f)`' επιλέγει το σκίτσο που θα αντιγραφεί. Φαίνεται πως εμφανίζεται το όρισμα '`f`'. Αυτό σημαίνει πως καθώς η συνάρτηση καλείται μέσα σε μία επαναληπτική διαδικασία, το '`f`' που αντιπροσωπεύει τη θέση κύλισης θα παίρνει κάθε φορά διαφορετική τιμή, με αποτέλεσμα να αντιγράφει κάθε φορά το σκίτσο της εκάστοτε θέσης κύλισης που αντιπροσωπεύει, μέχρι αυτές να τελειώσουν. Στη συνέχεια ο κώδικας ανοίγει το αρχείο '`ChipCollision" & ch & "_" & f & ".ipt`' και κάνει επικόλληση το σκίτσο που αντέγραψε πριν. Το τελικό αποτέλεσμα φαίνεται στο σχήμα 3.6.



Σχήμα 3.6 Προφίλ κοπτικού δοντιού μαζί με προβολή αποβλίττου στη θέση κοπής 0 και θέση κύλισης 10.

Επόμενη συνάρτηση είναι η '`Fixing`'. Σκοπός αυτής της συνάρτησης είναι να καθηλώσει τα σημεία του αρχείου '`ChipProjected" & ch & ".ipt`' που δημιουργήθηκε προηγουμένως. Ο κώδικας λοιπόν ανοίγει το αρχείο '`ChipProjected" & ch & ".ipt`' (γραμμή 254) και έπειτα αποθηκεύει σε μια μεταβλητή τον αριθμό των στοιχείων του σκίτσου (Sketch Entities) με την εντολή '`k = osketch.SketchEntities.Count`'. Τώρα κύριο σκεπτικό της συνάρτησης είναι με μία επαναληπτική διαδικασία από το 1 έως το 'k' να αναγνωρίζεται το κάθε Sketch Entity και αν αυτό πρόκειται για σημείο τότε να το καθηλώνει με την εντολή '`Call osketch.GeometricConstraints.AddGround(osketch.SketchEntities.Item(i))`'. Έτσι εφόσον κάθε γραμμή έχει ένα σημείο για αρχή και ένα άλλο για τέλος, καθηλώνεται και αυτή όπως φαίνεται και στο σχήμα 3.7. Παρατηρείται επίσης πως όταν ένα Sketch Entity γίνει 'fixed' αλλάζει χρώμα και από πράσινο γίνεται μοβ.

Εδώ όμως υπάρχει ένα πρόβλημα που θα συναντηθεί και πιο μετά. Ενώ όλα τα Sketch Entities της προβολής του αποβλίττου φαίνονται να είναι είτε ευθείες γραμμές, είτε τόξα, είτε splines, ορισμένες φορές συναντάται ένα άλλο είδος γραμμής, το ελλειπτικό τόξο (elliptical arc) το οποίο χρήζει ειδικής μεταχείρισης. Αυτό δεν ορίζεται από ένα σημείο αρχής και ένα τέλους οπότε προσθέτεται κι άλλος ένας έλεγχος στην επαναληπτική διαδικασία προτρέποντας τον κώδικα αν συναντήσει ένα στοιχείο που να είναι ελλειπτικό τόξο, να το καθηλώσει (γραμμές 269-271). Τέλος αποθηκεύεται και κλείνει το αρχείο.



Σχήμα 3.7 Το προφίλ του δοντιού με την προβολή του αποβλίττου στη θέση κοπής 0 και θέση κύλισης 10 έπειτα από την καθήλωση (fix) των στοιχείων του σκίτσου

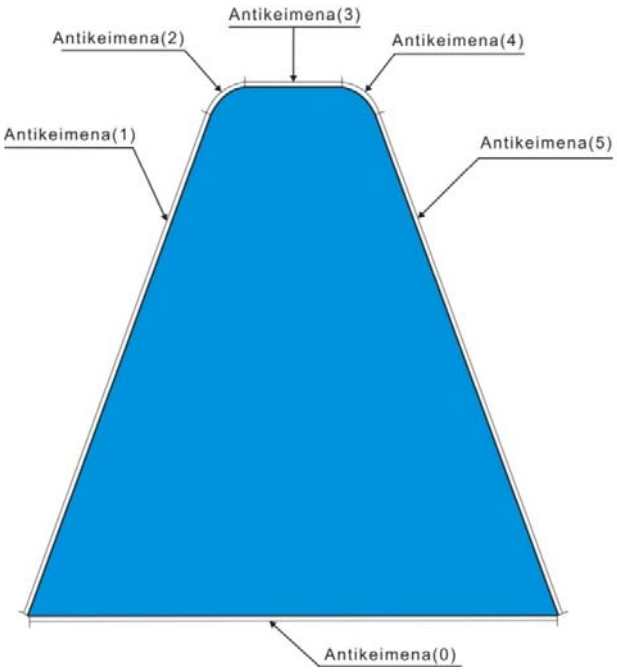
Τρίτη στη σειρά συνάρτηση που καλείται στο κυρίως πρόγραμμα είναι η '`MakeToothPerigram`'. Σκοπός αυτής της συνάρτησης είναι να εξάγει σε ένα αρχείο κειμένου ορισμένα σημεία που βρίσκονται πάνω στο περίγραμμα του προφίλ του δοντιού. Τα σημεία αυτά εξάγονται σε μορφή συντεταγμένων X, Y. Η συνάρτηση αυτή παίρνει σαν όρισμα έναν ακέραιο, που υποδηλώνει τη θέση κοπής στην οποία βρισκόμαστε.

Αρχικά ανοίγει το αρχείο '`Profil" & ch & ".ipt`' που όπως προαναφέρθηκε περιέχει το προφίλ του δοντιού που κόβει στη θέση κοπής '`ch`'. Έπειτα ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν και φαίνεται πως μέσα σ' αυτές βρίσκεται και η μεταβλητή '`klimaka`' η οποία ορίζει την πυκνότητα με την οποία θα παρθούν τα σημεία πάνω στο προφίλ του δοντιού. Αργότερα θα φανεί πως επιτυγχάνεται αυτό.

Μετά τη δήλωση των μεταβλητών ακολουθεί μια επαναληπτική διαδικασία για την καθήλωση όλων των στοιχείων του σκίτσου, μια διαδικασία παρόμοια με αυτή που συναντήθηκε στη συνάρτηση '`fixing`'. Στη συνέχεια γίνεται ο διαχωρισμός των γραμμών του προφίλ του δοντιού. Αρχικά έχει ορισθεί μια μεταβλητή '`Antikeimena(0 to 5)`' στη οποία μπορούν να καταχωρηθούν στοιχεία του σκίτσου π.χ. μια γραμμή, ένα σημείο κ.α. Σκοπός εδώ είναι να καταχωρηθούν οι γραμμές και τα τόξα του προφίλ στον πίνακα '`Antikeimena()`'. Ξεκινάει μια επαναληπτική διαδικασία από το 1 έως το 14 (αφού 14 είναι τα στοιχεία του σκίτσου του προφίλ του δοντιού). Εδώ εισέρχεται ένας έλεγχος,με τον οποίο όταν ο τύπος του στοιχείου που θα συναντήσει είναι τόξο (`If osketch.SketchEntities(i).Type = kSketchArcObject`), τότε μπαίνει σε έναν δεύτερο έλεγχο που λέει πως εάν η συντεταγμένη Y του αρχικού σημείου του τόξου είναι μεγαλύτερη από τη συντεταγμένη Y του τελικού σημείου (τα τόξα σχεδιάζονται από τα δεξιά προς τα αριστερά) τότε καταχωρεί στη μεταβλητή '`Antikeimena(2)`' το στοιχείο αυτό. Εάν συμβαίνει το αντίθετο τότε καταχωρεί στη μεταβλητή '`Antikeimena(4)`' το στοιχείο. Έτσι έχουν καταχωρηθεί τα δύο τόξα του προφίλ.

Στη συνέχεια γίνεται ο διαχωρισμός των ευθειών. Με μια επαναληπτική διαδικασία πάλι εξερευνούνται τα σημεία του σκίτσου μας από το 1 έως το 14. Όταν το πρόγραμμα συναντήσει ευθεία γραμμή '`If osketch.SketchEntities(i).Type = kSketchLineObject`' τότε μπαίνει σε έναν έλεγχο. Εκεί συναντάει μια σειρά από ελέγχους. Αρχικά αν η συντεταγμένη X του τελικού σημείου της ευθείας είναι 0 και η συντεταγμένη Y του τελικού σημείου είναι ίση με τη συντεταγμένη Y του αρχικού σημείου, τότε η ευθεία αυτή πρόκειται για τη βάση του δοντιού (που πρακτικά δεν υπάρχει) και αποθηκεύεται στη μεταβλητή '`Antikeimena(0)`'. Εάν η συντεταγμένη X του τελικού σημείου της ευθείας δεν είναι μηδέν και οι συντεταγμένες Y της αρχής και του τέλους της ευθείας είναι ίδιες τότε το στοιχείο αυτό καταχωρείται στο '`Antikeimena(3)`'. Εάν τώρα η συντεταγμένη Y του αρχικού σημείου είναι διαφορετική από τη συντεταγμένη Y του τελικού σημείου της γραμμής και η συντεταγμένη X του αρχικού μικρότερη από τη συντεταγμένη X του τελικού τότε το στοιχείο αυτό καταχωρείται στο '`Antikeimena(1)`'. Εάν η συντεταγμένη Y του αρχικού σημείου είναι διαφορετική από τη συντεταγμένη Y του τελικού σημείου της γραμμής και η συντεταγμένη X του αρχικού μεγαλύτερη από τη συντεταγμένη X του τελικού τότε το στοιχείο αυτό καταχωρείται στο '`Antikeimena(5)`'. Αυτή είναι μια διαδικασία που θα συναντηθεί και αργότερα. Σχηματικά τα στοιχεία του πίνακα `Antikeimena()` φαίνονται στο σχήμα 3.8.

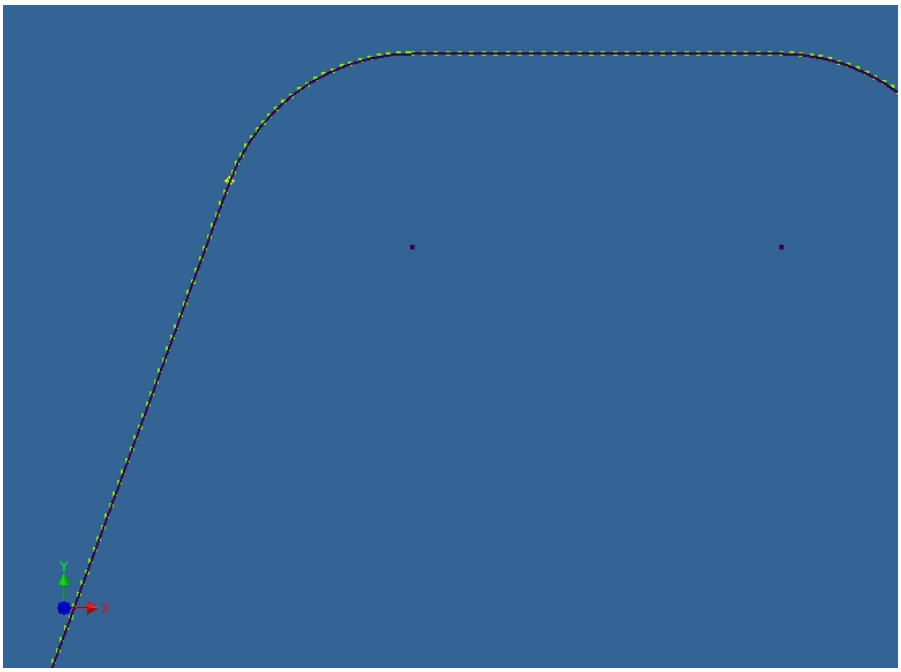
Τώρα αφού έχουν ορισθεί οι γραμμές του προφίλ, ο κώδικας πηγαίνει σε κάθε μία ξεχωριστά και δημιουργεί πάνω της σημεία με συγκεκριμένη απόσταση το ένα από το άλλο. Αρχικά πηγαίνει στο τόξο '`Antikeimena(2)`', αποθηκεύει το τέλος και την αρχή του παραμετρικά, στις μεταβλητές '`min`' και '`max`' αντίστοιχα με την εντολή '`Call CurveEval.GetParamExtents`



Σχήμα 3.8 Τα στοιχεία του σκίτσου του προφίλ του δοντιού, διαχωρισμένα και καταχωρημένα στον πίνακα Antikeimena()

(min, max)' και ύστερα με την εντολή 'Call CurveEval.GetLengthAtParam(min, max, Length(2))' αποθηκεύει στη μεταβλητή 'Length(2)' το μήκος μεταξύ της αρχής και του τέλους. Ύστερα ορίζει το διάστημα 'dx' που θα απέχουν μεταξύ τους τα σημεία, σε σχέση με το μήκος αυτό με την εντολή 'dx = Length(2) / klimaka' όπου η μεταβλητή 'klimaka' ορίζει το πόσο πυκνά θα είναι τα σημεία. Όσο μεγαλύτερη τιμή παίρνει τόσο πιο πυκνά είναι. Για τη γραμμή 'Antikeimena(1)' τώρα ακολουθείται η ίδια διαδικασία με τη διαφορά του ότι ορίζεται ένας ακέραιος 'n' ίσος με 'n = Int(Length(1) / dx)' δηλαδή με το ακέραιο μέρος της διαίρεσης του μήκους του 'Antikeimena(1)' προς το 'dx' που υπολογίσθηκε από πάνω. Με αυτόν τον τρόπο γίνεται εφικτό όλα τα σημεία στο περίγραμμα του δοντιού να ισαπέχουν και να μπορεί να μεταβληθεί η απόστασή τους αλλάζοντας μονάχα μία μεταβλητή. Ομοίως ο κώδικας λειτουργεί και για τις άλλες γραμμές του προφίλ. Άρα τελικά το βήμα που θα προχωράει η διαδικασία της δημιουργίας των σημείων θα έχει ίδιο μήκος για κάθε διαφορετική γραμμή όμως διαφορετική τιμή παραμετρικά. Άρα για την γραμμή Antikeimena(1) θα είναι 'vima(1) = 1 / n', για το τόξο 'Antikeimena(2)' θα είναι 'vima(2) = -1 / klimaka' (το μείον υπάρχει γιατί όπως προαναφέραμε η σχεδίαση του τόξου ξεκινάει από τα δεξιά προς τα αριστερά), για τη γραμμή 'Antikeimena(3)' θα είναι 'vima(3) = 1 / n1' και ομοίως για τις υπόλοιπες γραμμές.

Η διαδικασία για τη δημιουργία των σημείων που ακολουθεί είναι σχετικά απλή. Αρχικά με μία επαναληπτική διαδικασία επιλέγεται μία από τις 5 γραμμές. Στη γραμμή 'Antikeimeno(0)' δε θα δημιουργηθούν σημεία αφού στην πραγματικότητα δεν υφίσταται. Αφού επιλεγεί μια γραμμή τότε επιλέγονται παραμετρικά η αρχή και το τέλος του και με την εντολή 'Call CurveEval.GetPointAtParam(paramval, ptVal)' αποθηκεύονται στον πίνακα 'ptVal()' οι συντεταγμένες X και Y του σημείου αρχής της γραμμής. Παρομοίως αποθηκεύονται οι συντεταγμένες του τέλους στον πίνακα 'ptVal1()'. Έπειτα με μία επαναληπτική διαδικασία, ξεκινώντας από την αρχή της εκάστοτε επιλεγμένης γραμμής με βήμα που έχει ορισθεί παραπάνω, δημιουργούνται σημεία, οι συντεταγμένες των οποίων αποθηκεύονται στον πίνακα 'perigramma(:,1)'. Έτσι η μεταβλητή 'perigramma(0,0)' είναι η συντεταγμένη X του πρώτου σημείου και η μεταβλητή 'perigramma(0,1)' η συντεταγμένη Y του πρώτου σημείου. Με έναν μετρητή 'v', η γραμμή του πίνακα αυξάνεται κατά ένα και έτσι στον πίνακα 'perigramma(v,1)' αποθηκεύονται οι συντεταγμένες όλων των σημείων του περιγράμματος όπως φαίνεται στο σχήμα 3.9. Τέλος καλώντας τη συνάρτηση 'Write2File' τυπώνονται τα στοιχεία του πίνακα 'perigramma(v,1)' σε ένα αρχείο κειμένου 'Perigramma" & ch & ".txt' όπου η μεταβλητή 'ch' όπως έχουμε πει αντιπροσωπεύει τη θέση κοπής.



Σχήμα 3.9 Μεγέθυνση στο προφίλ του δοντιού με δημιουργημένα σημεία στις γραμμές

3.4 Συνάρτηση ChipCollision

Η συνάρτηση αυτή είναι και η κυριότερη του κώδικα. Καλείται στην κύρια συνάρτηση μέσα σε μία επαναληπτική διαδικασία για όλες τις θέσεις κύλισης. Παίρνει δύο ορίσματα, τον ακέραιο '`kl`' που υποδεικνύει τη θέση κύλισης και τον ακέραιο '`ch`' που υποδεικνύει τη θέση κοπής. Σκοπός της συνάρτησης είναι να εξάγει ένα αρχείο κειμένου το οποίο θα περιέχει πληροφορίες για το πού υπάρχει επικάλυψη του αποβλίττου. Αυτό επιτυγχάνεται με μια σειρά βημάτων που θα φανούν παρακάτω.

3.4.1 Επεξεργασία αποβλίττου

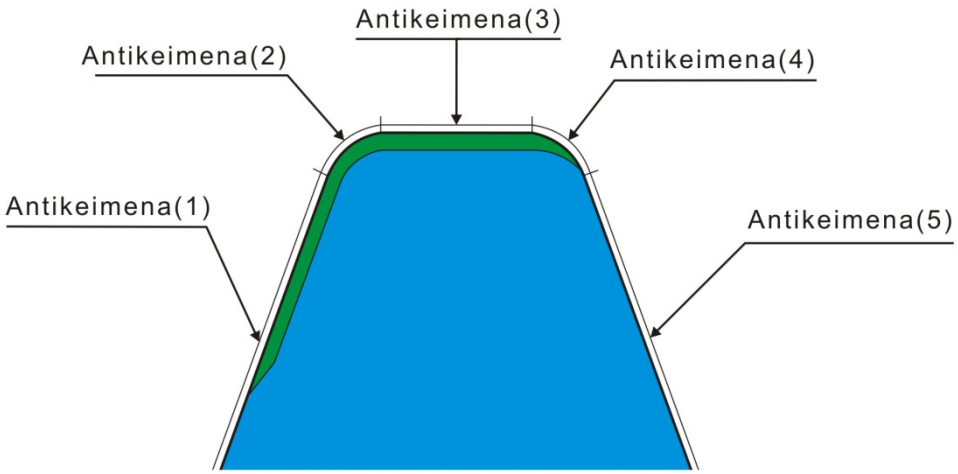
Η συνάρτηση ξεκινάει ανοίγοντας το αρχείο '`ChipCollision" & ch & "_" & kl & ".ipt`' (γραμμή 283) με τρόπο παρόμοιο που είδαμε προηγουμένως. Έπειτα εφαρμόζεται η ίδια διαδικασία που περιγράφηκε στη συνάρτηση '`MakeToothPerigram`' για διαχωρισμό των γραμμών του προφίλ του δοντιού, με αποτέλεσμα την αποθήκευσή τους σε ένα πίνακα με όνομα `Antikeimena()` όπως φαίνεται στο σχήμα 3.8 (γραμμές 291-332).

Στη συνέχεια ορίζονται οι γραμμές του αποβλίττου, ταξινομούνται κατά τον άξονα X και αποθηκεύονται σε έναν πίνακα '`EntsApovlitou()`'. Αυτό επιτυγχάνεται με τον ακόλουθο τρόπο. Αρχικά υπάρχει μια επαναληπτική διαδικασία που ξεκινάει από το στοιχείο 15 του σκίτσου (αφού τα πρώτα 14 στοιχεία ανήκουν στο προφίλ του δοντιού) έως το συνολικό αριθμό των στοιχείων που υπάρχουν στο σκίτσο. Με το που ξεκινήσει η διαδικασία υπάρχει ένας έλεγχος για να διαπιστωθεί εάν το στοιχείο που εξετάζεται είναι μια γραμμή εφαπτόμενη στο προφίλ του δοντιού (γραμμές 348-359). Εάν όντως είναι τότε δεν είναι χρήσιμη και δεν αποθηκεύεται. Εάν όμως ο έλεγχος δείξει πως το στοιχείο αυτό δεν είναι γραμμή που εφάπτεται στο προφίλ τότε το στοιχείο αυτό αποθηκεύεται στον πίνακα '`EntsApovlitou()`' ο οποίος είναι ένας πίνακας στον οποίο καταχωρούνται στοιχεία του σκίτσου. Τα στοιχεία είναι τυχαία διατεταγμένα στον πίνακα. Για να κατανεμηθούν τώρα κατά αύξουσα σειρά σύμφωνα με τη συντεταγμένη X του αρχικού τους σημείου, καλείται η συνάρτηση '`BubbleSort1`'

(γραμμή 384) η οποία παίρνει σαν όρισμα τη συντεταγμένη Χ του αρχικού σημείου του κάθε στοιχείου του σκίτσου, τον πίνακα '`EntsApovlitou()`' και το ελάχιστο και το μέγιστο αριθμό των στοιχείων του πίνακα. Έτσι λοιπόν ο πίνακας '`EntsApovlitou()`' περιέχει όλες τις γραμμές του αποβλίττου της θέσης κύλισης στην οποία βρίσκεται ο κώδικας κατανεμημένες κατά τον άξονα Χ.

Τώρα πρέπει με κάποιο τρόπο να προσδιοριστεί πάνω σε ποιο μέρος του προφίλ του δοντιού ξεκινάει να υφίσταται απόβλιττο και σε ποιο μέρος σταματάει. Δηλαδή πάνω σε ποιο στοιχείο του πίνακα '`Antikeimena()`'. Για να γίνει αυτό αρχικά αποθηκεύονται σε δύο πίνακες οι συντεταγμένες του πρώτου και του τελευταίου σημείου του αποβλίττου ως προς τον άξονα Χ (γραμμές 388-404). Στη συνέχεια με μία σειρά ελέγχων, ο κώδικας βλέπει πάνω σε ποιο στοιχείο του πίνακα '`Antikeimena()`' βρίσκεται το πρώτο και το τελευταίο σημείο της απεικόνισης του αποβλίττου, κάτι που γίνεται με σύγκριση των συντεταγμένων Χ. Αυτή η πληροφορία αποθηκεύεται σε δύο μεταβλητές που παίρνουν ακέραιες τιμές την 'p' και την 'q'. Εάν δηλαδή το απόβλιττο ξεκινάει από τη γραμμή '`Antikeimena(1)`' τότε '`p=1`', αν ξεκινάει από την '`Antikeimena(2)`' τότε '`p=2`' κ.ο.κ. Αν τώρα το απόβλιττο τελειώνει στη γραμμή '`Antikeimena(5)`' τότε '`q=1`', αν τελειώνει στο τόξο '`Antikeimena(4)`' τότε '`q=2`' κ.ο.κ.

Στο σχήμα 3.10 φαίνεται το άνω μέρος του προφίλ του δοντιού με το απόβλιττο που έχει κόψει σε μία θέση κύλισης. Διακρίνεται πως το πρώτο σημείο του αποβλίττου ως προς τον άξονα Χ βρίσκεται πάνω στη γραμμή '`Antikeimena(1)`' άρα '`p=1`' και το τελευταίο βρίσκεται πάνω στο τόξο '`Antikeimena(4)`' άρα '`q=2`'.



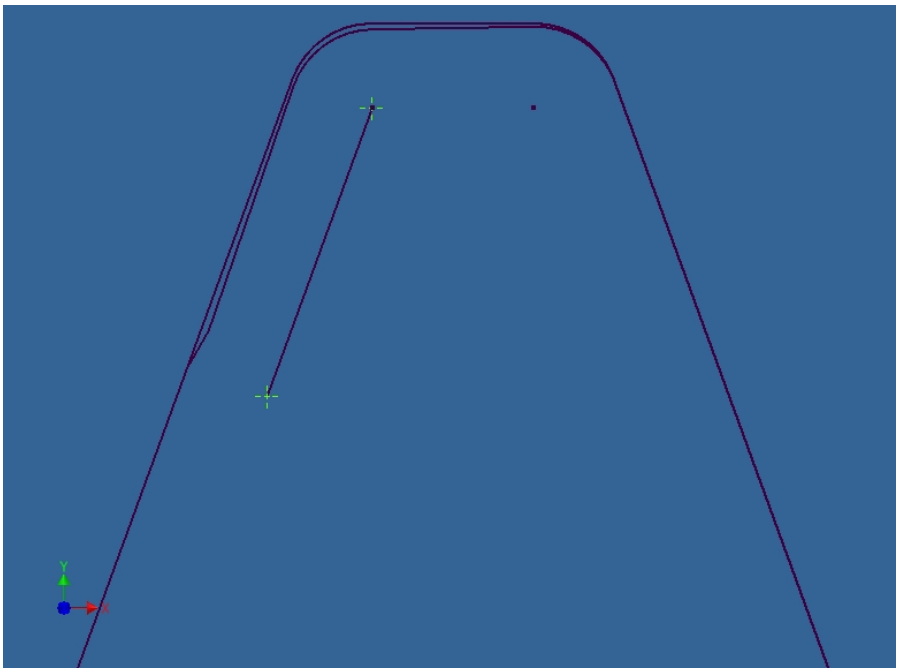
Σχήμα 3.10 Άνω μέρος του κοπτικού δοντιού με το απόβλιττο το οποίο αφαιρεί σε μία θέση κύλισης

Στη συνέχεια (γραμμές 420-438), εφαρμόζεται μια διαδικασία που εφαρμόσθηκε και στη συνάρτηση '`MakeToothPerigram`' με την οποία ορίζεται ένα διάστημα '`dx`' σε σχέση με το μήκος του τόξου '`Antikeimena(2)`', το οποίο είναι χρήσιμο αργότερα. Τέλος στις επόμενες γραμμές μέχρι να αρχίσει η δημιουργία του πλέγματος δηλώνονται κάποιες μεταβλητές που θα χρειαστούν αργότερα και γίνονται κάποιοι έλεγχοι που έχουν να κάνουν με το εάν κάποιο στοιχείο του αποβλίττου είναι τόξο έλλειψης και οι οποίοι θα αναλυθούν παρακάτω.

3.4.2 Δημιουργία πλέγματος (Grid)

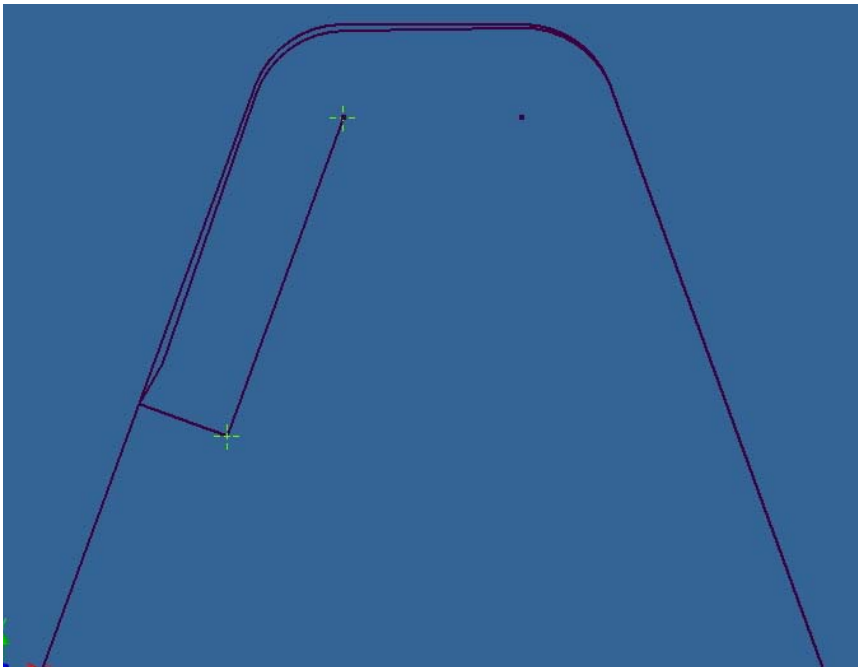
Σε αυτό το τμήμα του κώδικα θα δημιουργηθεί ένα πλέγμα από σημεία τα οποία θα βοηθήσουν να αντληθεί και η τελική πληροφορία που χρειαζεται στο τέλος της συνάρτησης. Αρχικά (γραμμές 487-493) δηλώνονται κάποιες μεταβλητές που χρειάζονται. Έπειτα δηλώνεται το 'π' το οποίο θεωρητικά δίνεται από την ισότητα '`pi=4*Atn(1)`' όπου '`Atn(1)`' είναι η γωνία της οποίας η εφαπτομένη είναι ίση με 1. Ορίζεται η γωνία μεταξύ της γραμμής '`Antikeimena(1)`' και του άξονα X η οποία είναι 20^o. Όπως είδαμε και στη συνάρτηση '`MakeToothPerigram`' έτσι και εδώ θα ορισθεί μια κλίμακα με όνομα μεταβλητής '`klimakaGrid`' από την τιμή της οποίας θα ορίζεται η πυκνότητα του πλέγματος. Η προκαθορισμένη τιμή της μέσα στο πρόγραμμα είναι 20. Ορίζεται η μεταβλητή '`dxGrid`' ίση με το πηλίκο του μήκους του τόξου '`Antikeimena(2)`' προς την τιμή της '`klimakaGrid`'. Άρα από εδώ και πέρα όλες οι αποστάσεις θα εξαρτώνται από τη μεταβλητή '`dxGrid`' άρα από την '`klimakaGrid`'.

Τώρα ξεκινάει η δημιουργία του πλέγματος. Ξεκινάει με έναν έλεγχο (γραμμή 505) για το αν υπάρχει υλικό πάνω στη γραμμή '`Antikeimena(1)`'. Εάν υπάρχει, δηλαδή εάν το '`p=1`' όπως φαίνεται στο σχήμα 3.10, τότε μπαίνει στον έλεγχο. Αρχικά δημιουργείται ένα σημείο, στη θέση που βρίσκεται η αρχή του αποβλίττου με την ονομασία '`GridPoint1`'. Στη συνέχεια δημιουργούνται δύο νέα σημεία κάθετα στη γραμμή '`Antikeimena(1)`' με απόσταση από αυτή ίση με την ακτίνα του τόξου '`Antikeimena(2)`' τα οποία φαίνονται στο σχήμα 3.11. Αυτά τα δύο σημεία ενώνονται όπως φαίνεται στο σχήμα 3.11 και η γραμμή που δημιουργείται αποτελεί μία βοηθητική γραμμή όπως θα φανεί, με την ονομασία '`HelpLine`'. Επίσης ενώνεται το '`GridPoint1`' με το τελικό σημείο της γραμμής '`Antikeimena(1)`' και η γραμμή που δημιουργείται ονομάζεται '`GridLine`'.

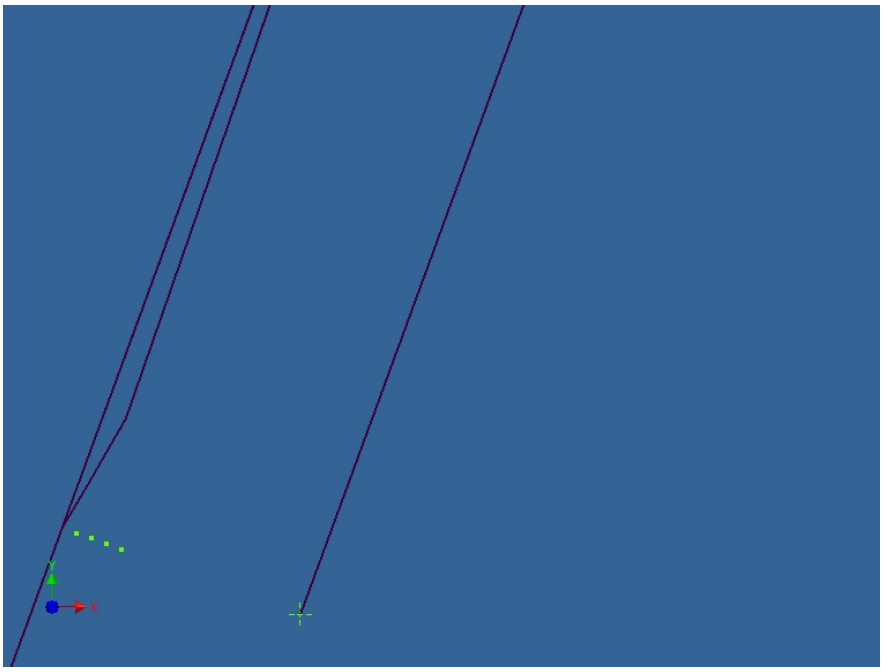


Σχήμα 3.11 Βοηθητική γραμμή κατά τη δημιουργία του πλέγματος αν υπάρχει απόβλιττο κατά μήκος της γραμμής '`Antikeimena(1)`'

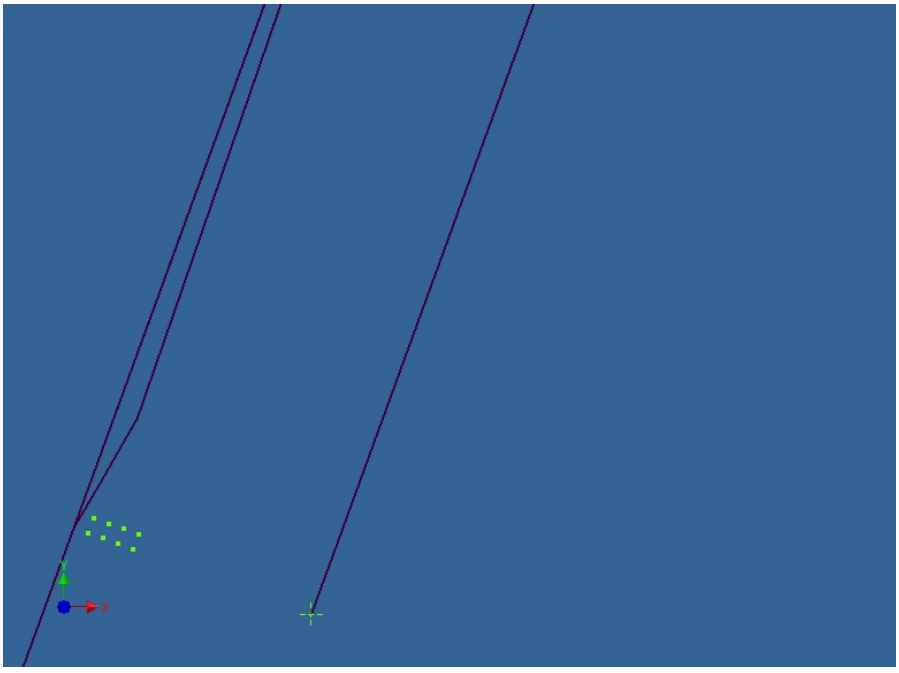
Η διαδικασία λοιπόν που δημιουργείται το πλέγμα περιλαμβάνει δύο επαναληπτικές διαδικασίες. Αρχικά ξεκινάει η πρώτη επαναληπτική διαδικασία από το αρχικό σημείο της γραμμής '`GridLine`' όπου και δημιουργείται μία γραμμή, κάθετη στην '`GridLine`' η οποία προεκτείνεται έως την '`HelpLine`' όπως φαίνεται στο σχήμα 3.12. Στη συνέχεια ξεκινάει μια δεύτερη επαναληπτική διαδικασία κατά την οποία δημιουργούνται ισαπέχοντα σημεία πάνω στη γραμμή που έχει δημιουργηθεί με βήμα που έχουμε ορίσει εμείς και είναι πάντα σχετικό με το μήκος του τόξου '`Antikeimena(2)`' και μπορούμε να μεταβάλλουμε αλλάζοντας την τιμή της μεταβλητής '`klimakaGrid`'. Η δεύτερη επαναληπτική διαδικασία λαμβάνει χώρα έως το ¼ της γραμμής που έχουμε δημιουργήσει όπως φαίνεται στο σχήμα 3.13 καθώς δεν είναι απαραίτητο να δημιουργηθούν όλα τα σημεία κατά μήκος της γραμμής. Όταν ο κώδικας βγει από τη δεύτερη επαναληπτική διαδικασία, διαγράφεται η γραμμή που είχε δημιουργηθεί και η πρώτη επαναληπτική διαδικασία προχωράει κατά ένα βήμα πάνω στη '`GridLine`' και επαναλαμβάνει την ίδια διαδικασία δημιουργώντας νέα σημεία όπως φαίνεται στο σχήμα 3.14. Όταν λοιπόν ολοκληρωθεί η επαναληπτική διαδικασία για τη γραμμή '`Antikeimena(1)`' το πρώτο μέρος του πλέγματος δημιουργείται όπως φαίνεται στο σχήμα 3.15.



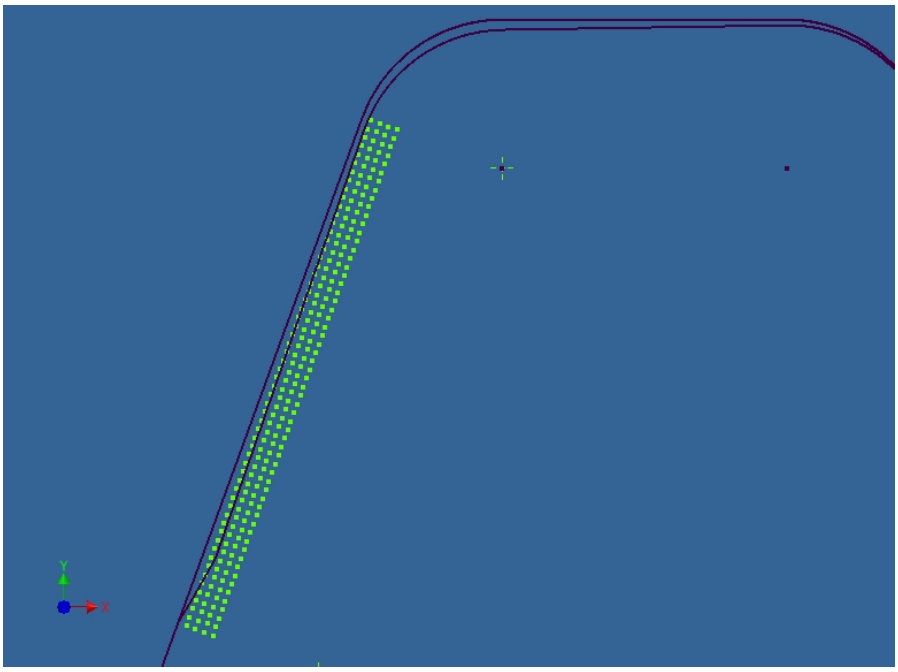
Σχήμα 3.12 Δημιουργία κάθετης γραμμής για εκκίνηση της δημιουργίας του πλέγματος



Σχήμα 3.13 Τα σημεία του πλέγματος που έχουν δημιουργηθεί μετά από μία επανάληψη



Σχήμα 3.14 Τα σημεία του πλέγματος που έχουν δημιουργηθεί μετά από δύο επαναλήψεις

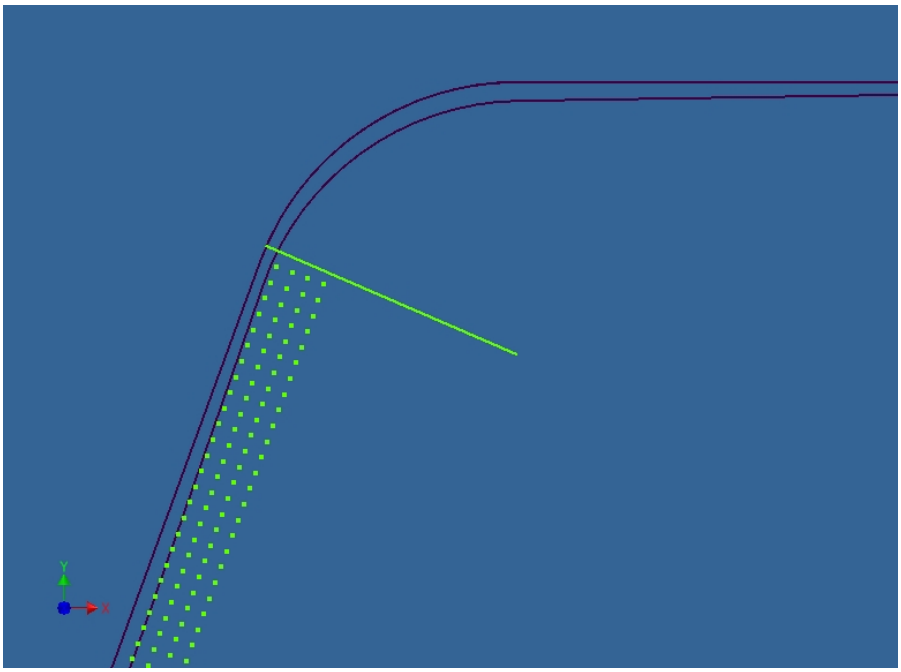


Σχήμα 3.15 Τελική μορφή του πλέγματος για τη γραμμή 'Antikeimena(1)'

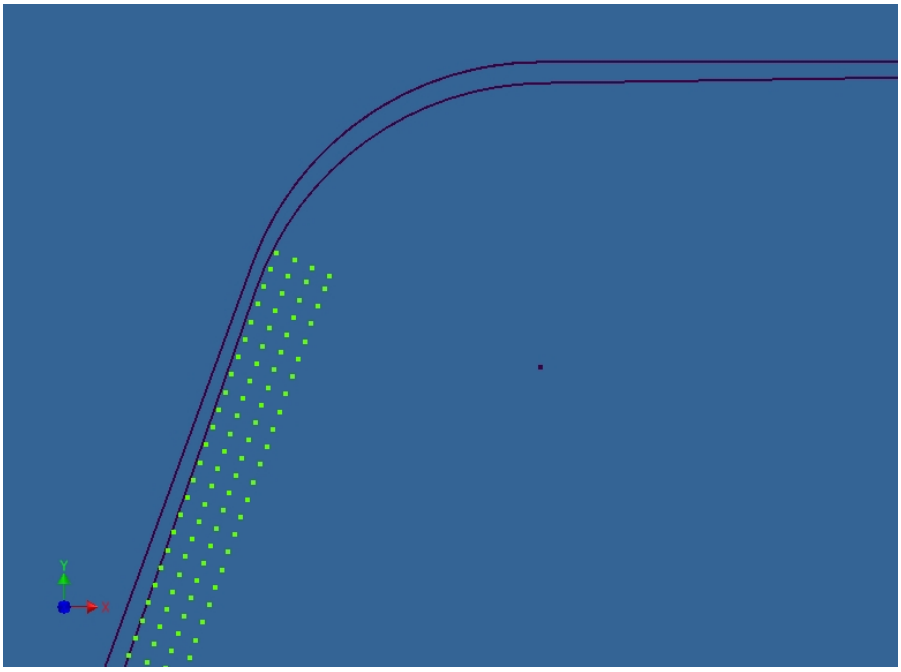
Μετά το πέρας της δημιουργίας του πλέγματος για τη γραμμή '`Antikeimena(1)`' υπάρχει ένας έλεγχος για το αν υπάρχει απόβλιττο στην περιοχή του τόξου '`Antikeimena(2)`' δηλαδή εάν '`p=2`' (γραμμή 579) και για το αν το απόβλιττο δεν τελειώνει πάνω στη γραμμή '`Antikeimena(1)`' δηλαδή '`q≠5`'. Εδώ ακολουθείται παρόμοια διαδικασία με προηγουμένως με μοναδική διαφορά πως τώρα δε χρειάζεται βοηθητική γραμμή. Ο κώδικας ξεκινάει να κινείται πάνω στο τόξο '`Antikeimena(2)`' με μία επαναληπτική διαδικασία και με συγκεκριμένο βήμα που ορίζεται από τη μεταβλητή '`klimakagrid`'. Δημιουργεί ένα σημείο εκεί που βρίσκεται και το ενώνει με το κέντρο του τόξου '`Antikeimena(2)`' δημιουργώντας μία νέα γραμμή όπως φαίνεται στο σχήμα 3.16 και την ονομάζει '`GridLine`'.

Έπειτα με μία επαναληπτική διαδικασία δημιουργούνται κατά μήκος της γραμμής '`GridLine`' τα σημεία του πλέγματος με συγκεκριμένο βήμα και στη συνέχεια σβήνεται η γραμμή. Μετά το πέρας μιας επανάληψης έχουμε το αποτέλεσμα που φαίνεται στο σχήμα 3.17. Έτσι ο κώδικας κινείται κατά μήκος το τόξου και με τον ίδιο τρόπο δημιουργούνται τα σημεία του πλέγματος. Το τελικό αποτέλεσμα για το τόξο φαίνεται στο σχήμα 3.18

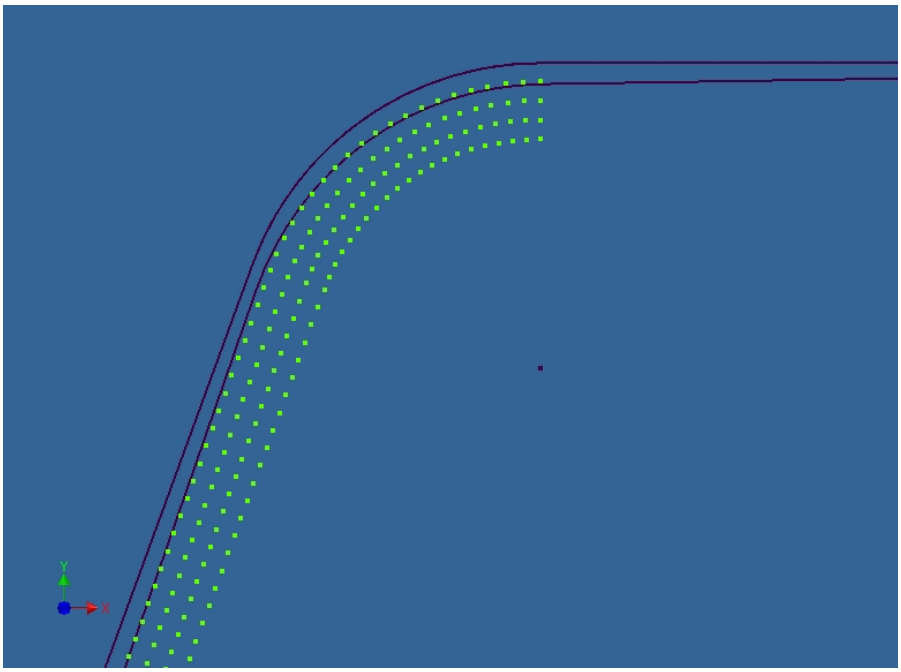
Η διαδικασία δημιουργίας του πλέγματος συνεχίζεται παρομοίως μέχρι το σημείο που υπάρχει απόβλιττο. Γίνεται πάντα έλεγχος για το αν υπάρχει απόβλιττο στο εξεταζόμενο στοιχείο (δηλαδή σε ποια γραμμή του πίνακα '`Antikeimena()`' βρίσκεται ο κώδικας) και εάν βγει θετικός τότε ξεκινάει η δημιουργία του πλέγματος. Στο σχήμα 3.19 φαίνεται το τελικό πλέγμα για την απεικόνιση του αποβλίττου στη θέση κύλισης 10 της θέσης κοπής 0.



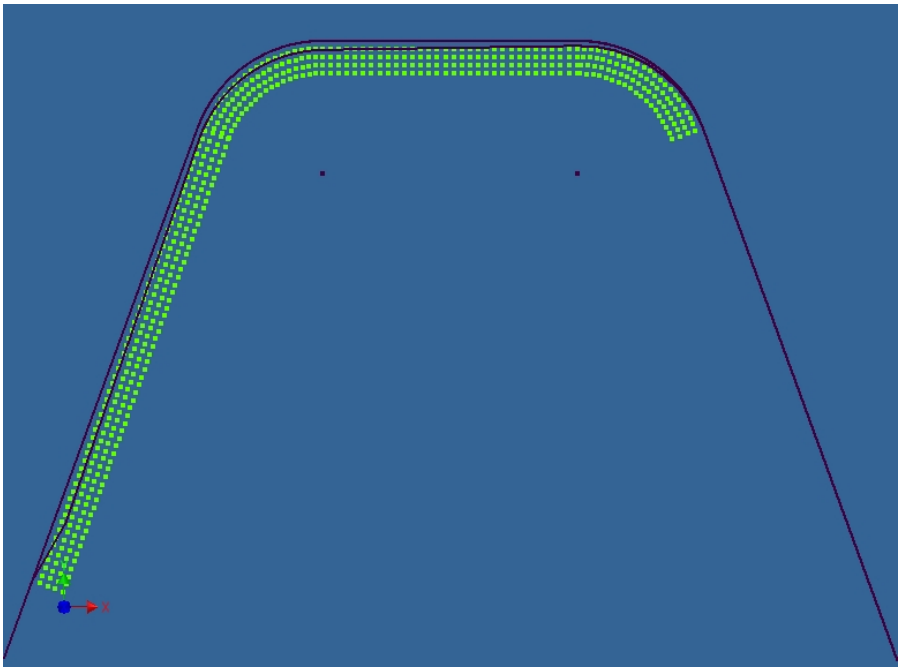
Σχήμα 3.16 Δημιουργία ‘GridLine’



Σχήμα 3.17 Δημιουργία πλέγματος στο τόξο μετά από μια επανάληψη



Σχήμα 3.18 Τελική μορφή πλέγματος στο τόξο



Σχήμα 3.19 Τελικό πλέγμα στη θέση κύλισης 10 θέση κοπής 0

Φαίνεται πως ίσως ήταν πιο εύκολο να δημιουργηθεί ένα πλέγμα πιο γενικό, με λιγότερους περιορισμούς. Όμως στη συνέχεια θα φανεί πως τα σημεία του πλέγματος εξετάζονται ένα προς ένα έτσι όσο λιγότερα υπάρχουν τόσο μειώνεται ο υπολογιστικός φόρτος. Γι' αυτό το λόγο ο κώδικας μπορεί να είναι μεγαλύτερος και πιο πολύπλοκος όμως έγινε εφικτό να δημιουργηθούν μόνο τα απαραίτητα σημεία του πλέγματος.

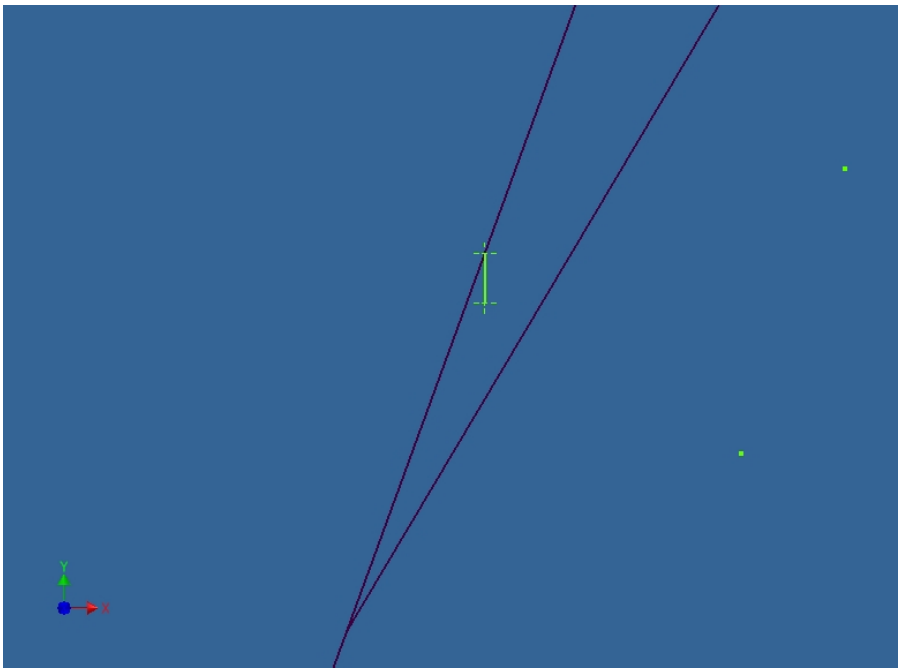
3.4.3 Δημιουργία προεκτάσεων (Constraints)

Σε αυτό το τμήμα του κώδικα, δημιουργούνται γραμμές οι οποίες ξεκινούν από το τις γραμμές του προφίλ του δοντιού και καταλήγουν πάνω στο απόβλιττο. Όπως και προηγουμένως το σκεπτικό εργασίας παραμένει το ίδιο. Επεξεργάζεται δηλαδή η κάθε γραμμή του προφίλ ξεχωριστά.

Αρχικά υπάρχει ένας έλεγχος για να διαπιστωθεί εάν υπάρχει απόβλιττο στη γραμμή '`Antikeimena(1)`' του προφίλ, δηλαδή εάν '`p=1`' όπως είδαμε στο σχήμα 3.10. Αν όντως '`p=1`' τότε ο κώδικας μπαίνει στον έλεγχο αλλιώς τον προσπερνάει και μπαίνει σε αυτόν που έχει την τιμή της μεταβλητής '`p`'. Ας υποθέσουμε ότι το '`p=1`' οπότε μπαίνει στον πρώτο έλεγχο. Αρχικά ορίζεται η γραμμή του προφίλ '`Antikeimena(1)`' ως η γραμμή που επεξεργάζεται ο κώδικας τη συγκεκριμένη στιγμή και αποθηκεύει στους πίνακες '`ptVal`' και '`ptVal1`' τις συντεταγμένες της αρχής και του τέλους της γραμμής αντίστοιχα. Στην συνέχεια επειδή θα σχεδιαστούν πάνω στη γραμμή νέες κάθετες γραμμές με συγκεκριμένο βήμα, όπως έγινε και στη δημιουργία του πλέγματος ορίζεται μία κλίμακα η οποία θα είναι ενδεικτική για το πόσο απέχουν αυτά τα σημεία μεταξύ τους κάτι που έχει γίνει πιο πριν στον κώδικα(γραμμή 423). Επίσης πρέπει να ορισθεί το βήμα έτσι ώστε όποια κι αν είναι η τιμή της μεταβλητής '`klimaka`' οι γραμμές να ισαπέχουν και στο τέλος να μην περισσεύει κομμάτι της γραμμής χωρίς να έχει αξιοποιηθεί. Έτσι ορίζεται μια νέα μεταβλητή '`ypo`' και της καταχωρείται η τιμή '`ypo = (((max - min) * n * dx) / Length(1)) + min`' όπου '`max, min`' είναι το μέγιστο και το ελάχιστο παραμετρικό σημείο της γραμμής '`Antikeimena(1)`' και το '`n = Int(Length(1) / dx)`'. Έπειτα ορίζεται το βήμα καταχωρώντας στη μεταβλητή '`vima`' την τιμή '`(ypo-min)/n`'. Έτσι ο κώδικας έχει το βήμα με το οποίο θα κινηθεί πάνω στη γραμμή.

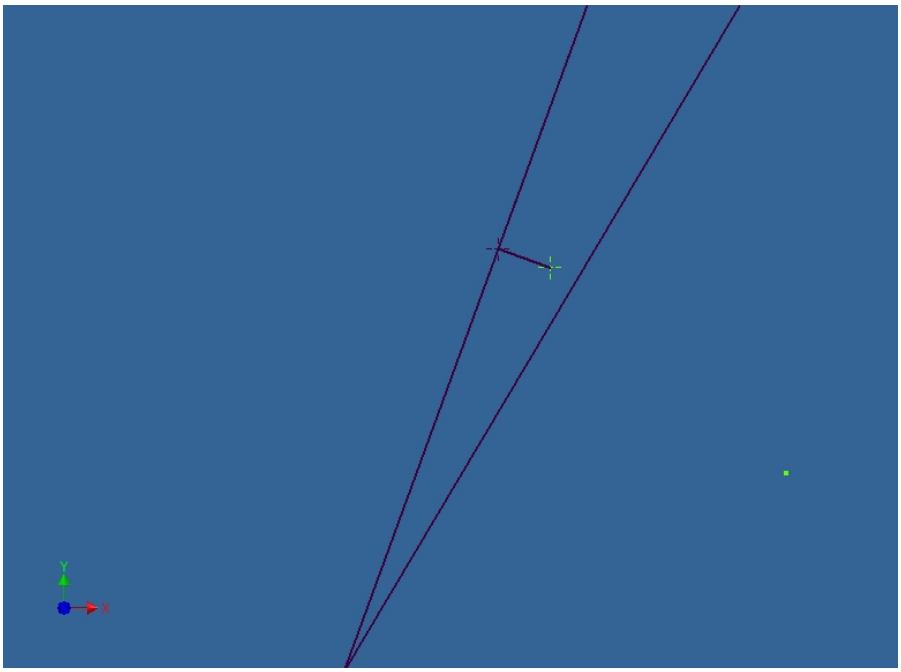
Τώρα θέλουμε να καθοριστεί το σημείο από το οποίο θα ξεκινήσει αυτή η διαδικασία και δεν είναι άλλο από το σημείο που ξεκινάει το απόβλιττο και βρίσκεται πάνω στη γραμμή '`Antikeimena(1)`'. Αυτό γίνεται με την εντολή '`Call CurveEval.GetParamAtPoint(ptval5, paramval10, paramval11, paramval3, paramval12)`'. Αυτή η εντολή παραμετροποιεί το σημείο που ξεκινάει το απόβλιττο (οι συντεταγμένες του οποίου είναι αποθηκευμένες στον πίνακα '`ptVal5()`') και αποθηκεύει την τιμή στη μεταβλητή '`paramval3`'. Οι άλλες μεταβλητές '`paramval10`', '`paramval11`' και '`paramval12`' είναι βοηθητικές και η τιμή τους δεν παίζει κανένα ρόλο. Σε αυτό το σημείο όμως μετά από εφαρμογή του κώδικα σε διάφορες θέσεις κύλισης παρατηρήθηκε πως υπάρχουν φορές που το σημείο της αρχής του αποβλίττου δε συσχετίζεται ακριβώς με κάποιο σημείο της γραμμής '`Antikeimena(1)`'. Αυτό συμβαίνει είτε λόγω απλοποίησης κάποιων δεκαδικών του προγράμματος είτε λόγω σχεδιαστικού λάθους του προγράμματος. Έτσι όταν ο κώδικας το προτρέπει να παραμετροποιήσει πάνω στη γραμμή ένα σημείο το οποίο δεν ανήκει σε αυτή, τότε το πρόγραμμα εμφανίζει σφάλμα. Για να αντιμετωπιστεί αυτό, πριν γίνει η παραμετροποίηση, μπαίνει η εντολή '`On Error Resume Next`' (γραμμή 834) η οποία προτρέπει το πρόγραμμα αν συναντήσει σφάλμα απλά να συνεχίσει στην επόμενη γραμμή. Έπειτα γίνεται ένας έλεγχος για το αν υπάρχει σφάλμα αυτό να σβηστεί ('`If Err Then Err.Clear`') και απλά θέτει στη μεταβλητή '`paramval3`' την τιμή 0. Αυτό σημαίνει πως η διαδικασία σάρωσης της γραμμής '`Antikeimena(1)`' θα ξεκινήσει από την αρχή της και όχι από το σημείο που ξεκινάει τα υπάρχει απόβλιττο. Αυτό δε θα μας δώσει λάθος αποτελέσματα απλά θα δημιουργήσει κάποιες γραμμές παραπάνω οι οποίες δε θα χρησιμεύουν σε κάτι. Επίσης ορίζεται και η μεταβλητή '`paramval4`' η οποία υποδεικνύει πού θα σταματήσει η διαδικασία δημιουργίας των γραμμών και παίρνει την τιμή 1 αφού παραμετρικά το 1 είναι το τέλος της γραμμής.

Τώρα υπάρχει κι άλλος ένας έλεγχος. Αν το απόβλιττο τελειώνει στη γραμμή ‘Antikeimena(1)’ δηλαδή εάν ‘q=5’ τότε δίνεται στη μεταβλητή ‘paramval4’ η παραμετρική τιμή του σημείου του τέλους του αποβλίττου (γραμμές 844-853) με παρόμοια διαδικασία που δώθηκε και στη μεταβλητή ‘paramval3’ η τιμή της. Επίσης δίνεται και στον μετρητή ‘kpointer’ η τιμή 1, πληροφορία που θα μας χρησιμεύσει αργότερα για να μπορεί να ελέγθεί που τελειώνει το απόβλιττο. Εφόσον υπάρχει λοιπόν η αρχή, το τέλος και το βήμα ξεκινάει μια επαναληπτική διαδικασία από το ‘paramval3 + vima’ έως το ‘paramval4 + eps’ με βήμα ‘vima’. Η μεταβλητή ‘eps’ συμβολίζει έναν πολύ μικρό αριθμό και την μπαίνει γιατί μπορεί το πρόγραμμα που επεξεργάζεται τα δεδομένα να κόψει ορισμένα δεκαδικά ψηφία από την τιμή της ‘paramval4’ κι έτσι η διαδικασία να τελειώσει νωρίτερα από το αναμενόμενο. Οπότε τώρα ο κώδικας βρίσκεται πάνω στη γραμμή ‘Antikeimena(1)’ σε απόσταση ‘vima’ από το σημείο αρχής του αποβλίττου. Εκεί δημιουργείται ένα σημείο ακριβώς εκεί που βρίσκεται ο κώδικας και ένα λιγάκι πιο κάτω όπως φαίνεται στο σχήμα 3.20 τα οποία ενώνονται με μία γραμμή η οποία αποθηκεύεται στη μεταβλητή ‘oLine1’. Στη συνέχεια ορίζεται αυτή τη γραμμή κάθετη ως προς τη γραμμή ‘Antikeimena(1)’ και με την εντολή ‘Call osketch1.GeometricConstraints.AddGround(oLine1.StartSketchPoint)’ φιξάρεται το αρχικό σημείο της το οποίο είναι αυτό που βρίσκεται πάνω στη γραμμή ‘Antikeimena(1)’ όπως φαίνεται στο σχήμα 3.21 και αποθηκεύεται το μήκος της στη μεταβλητή ‘mikosLine1’



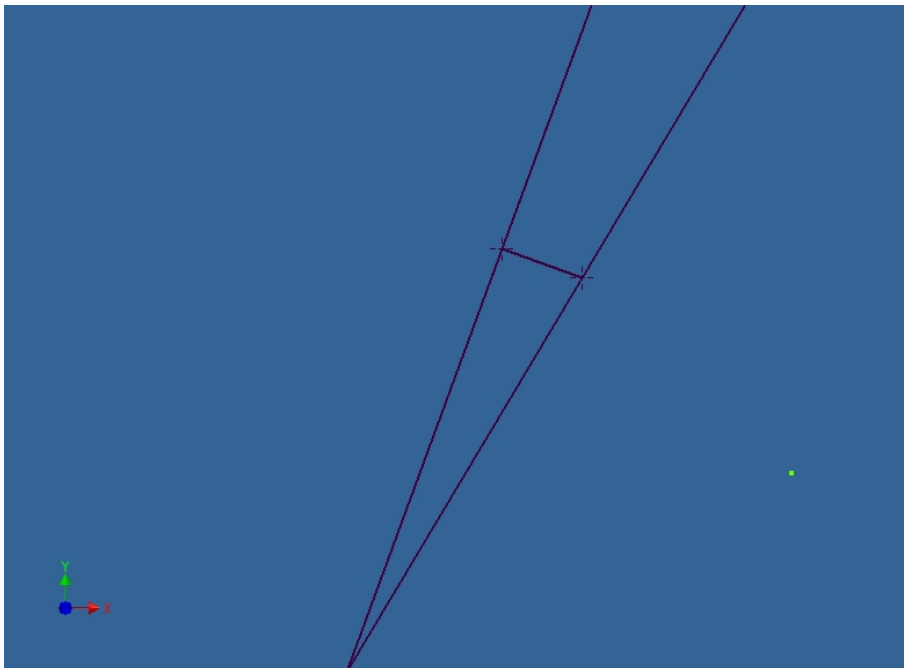
Σχήμα 3.20 Αρχική δημιουργία γραμμής

.

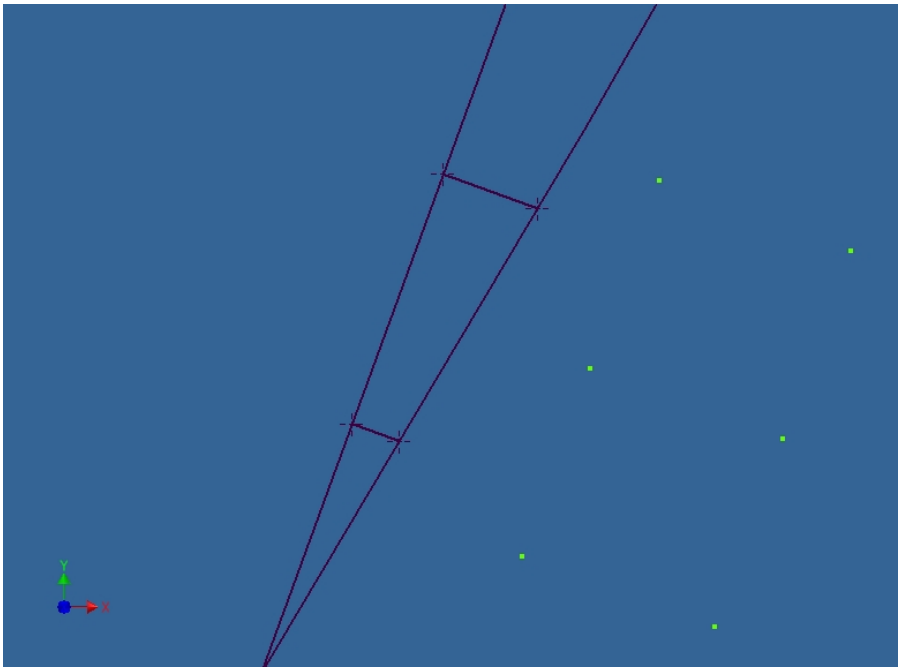


Σχήμα 3.21 Η γραμμή ενώ έχει γίνει κάθετη ως προς τη γραμμή 'Antikeimena(1)'

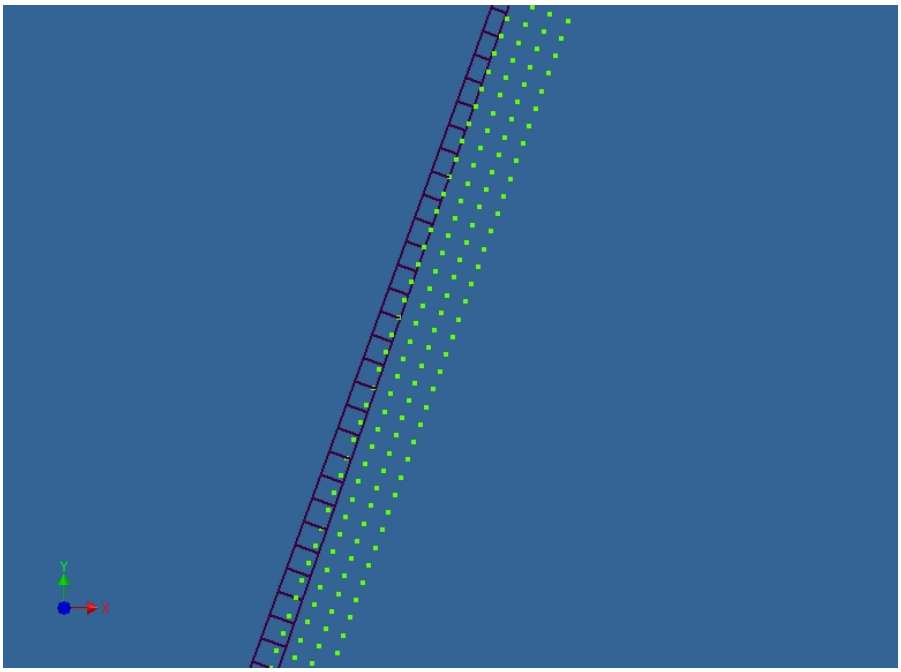
Επόμενο βήμα είναι να προεκταθεί η γραμμή μέχρι να συναντήσει το απόβλιττο. Αυτό επιτυγχάνεται με την εντολή 'Call osketch1.GeometricConstraints.AddCoincident (oLine1.EndSketchPoint, EntsApovlitou(PrevTrap))'. Η μεταβλητή 'PrevTrap' είναι αυτή που καθορίζει έως ποιο στοιχείο του αποβλίττου θα προεκταθεί η γραμμή. Είναι φυσιολογικό πως καθώς ο κώδικας προχωράει πάνω στη γραμμή 'Antikeimena(1)' το στοιχείο του αποβλίττου έως το οποίο θα γίνεται η προέκταση θα είναι διαφορετικό. Έτσι με μία σειρά από ελέγχους (γραμμές 869-875, 879-897) καθορίζεται η τιμή της μεταβλητής 'PrevTrap'. Στο σχήμα 3.22 φαίνεται η μορφή της γραμμής ενώ έχει προεκταθεί. Μετά τη δημιουργία της γραμμής λοιπόν, αυτή αποθηκεύεται σε ένα πίνακα που θα αποθηκευτούν όλες οι γραμμές με την εντολή 'Set pinakasLines(v) = oLine1' όπου 'v' ένας μετρητής που αυξάνεται κατά ένα σε κάθε βήμα. Επίσης σε έναν άλλο πίνακα 'pinakasxyl(:, 2)' αποθηκεύεται για κάθε γραμμή, η συντεταγμένη X και Y του σημείου αρχής της γραμμής 'oLine1' καθώς και το μήκος της. Υπάρχει και ένας επιπλέον έλεγχος (γραμμή 905) ο οποίος αν δει πως το μήκος της γραμμής 'oLine1' δεν έχει αλλάξει μετά την προέκτασή της, τότε αποθηκεύει την τιμή 0 για το μήκος της στον πίνακα 'pinakasxyl(:, 2)'. Αυτό συμβαίνει σε περιπτώσεις που δεν υπάρχει απόβλιττο για να επεκταθεί η γραμμή δηλαδή όταν το απόβλιττο διακόπτεται και δεν είναι συνεχόμενο κατά μήκος του προφίλ του δοντιού. Όταν ολοκληρωθεί όλη αυτή η διαδικασία η επαναληπτική διαδικασία αυξάνεται κατά 'vima' και συνεχίζει δημιουργώντας την επόμενη γραμμή με τον ίδιο τρόπο όπως φαίνεται στο σχήμα 3.23. Με το πέρας της διαδικασίας έχουν δημιουργηθεί όλες οι γραμμές που ξεκινάνε από τη γραμμή 'Antikeimena(1)' όπως βλέπουμε στο σχήμα 3.24 και έχουν αποθηκευτεί οι απαραίτητες πληροφορίες.



Σχήμα 3.22 Τελική μορφή της γραμμής (constraint)



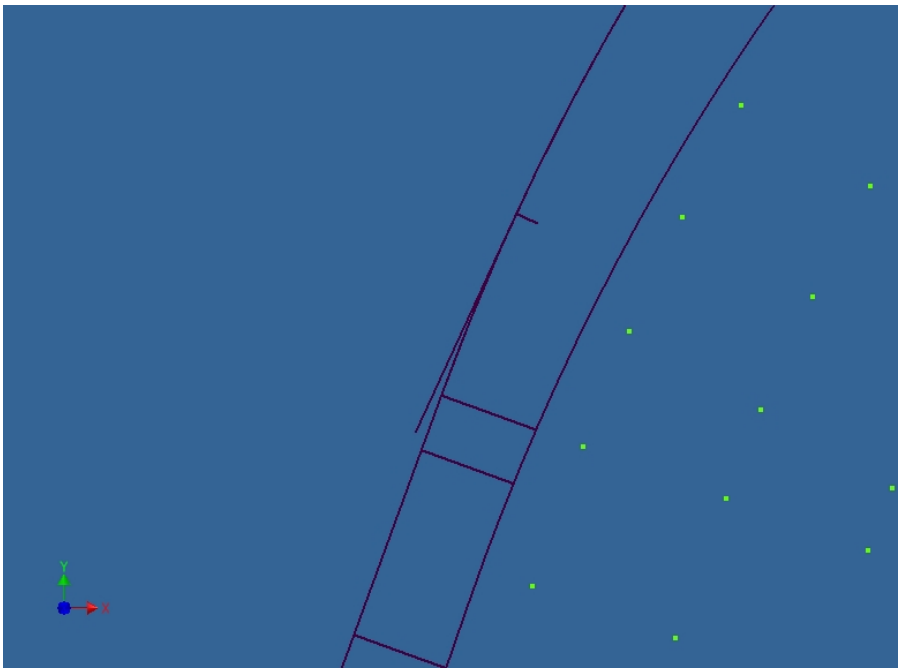
Σχήμα 3.23 Εξέλιξη της δημιουργίας γραμμών μετά από δύο βήματα



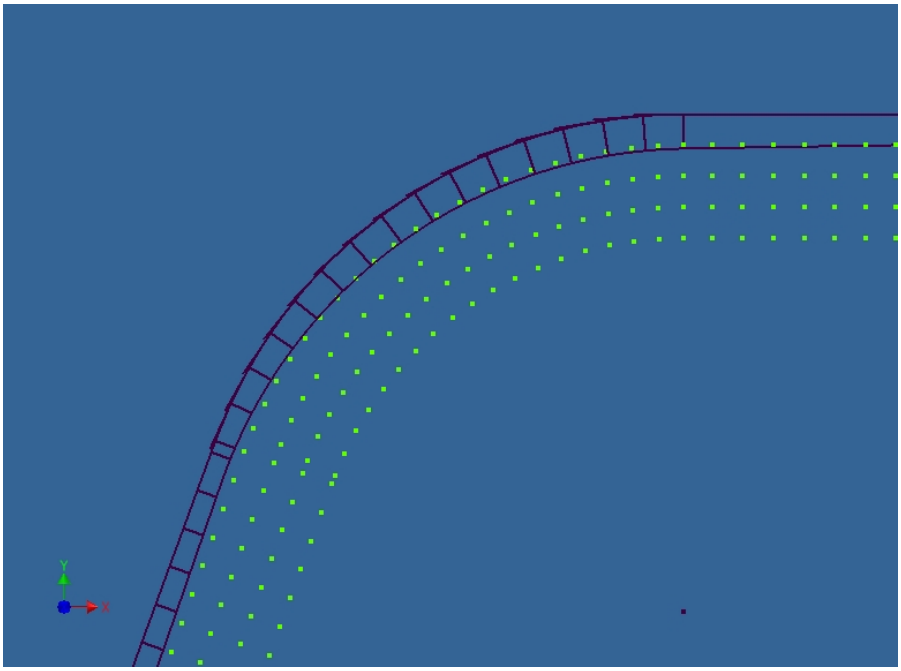
Σχήμα 3.24 Δημιουργία όλων των γραμμών (constraints) για το πρώτο τμήμα του δοντιού

Η διαδικασία συνεχίζει και προχωράει στο τόξο '`Antikeimena(2)`'. Αρχικά υπάρχει ένας έλεγχος ο οποίος εάν δει πως δεν υπάρχει άλλο απόβλιττο δηλαδή '`If kpointer = 1`' τότε παραλείπει τα επόμενα βήματα δημιουργίας των γραμμών και συνεχίζει τον κώδικα από το σημείο που ξεκινάει η επόμενη διαδικασία (γραμμή 1295) .Ο τρόπος τώρα με τον οποίο δημιουργούνται οι γραμμές είναι ίδιος με αυτόν που περιγράφηκε από πάνω με μία μοναδική διαφορά. Μετά που θα δημιουργηθεί η πρώτη γραμμή όπως φαίνεται στο σχήμα 3.20 πρέπει να έρθει κάθετη στο τόξο κάτι που δε γίνεται. Γι' αυτό το λόγο ο κώδικας φέρνει μια εφαπτομένη στο τόξο στο σημείο στο οποίο βρισκόμαστε εκείνη τη στιγμή (γραμμή 991) και στη συνέχεια ορίζει τη γραμμή που έχει δημιουργηθεί ως κάθετη στην εφαπτόμενη κάτι που βλέπουμε στο σχήμα 3.25. Έπειτα την προεκτείνει έως το απόβλιττο όπως και προηγουμένως. Αυτή η διαδικασία συνεχίζεται μέχρι να φτάσουμε στο τέλος του τόξου. Το αποτέλεσμα φαίνεται στο σχήμα 3.26.

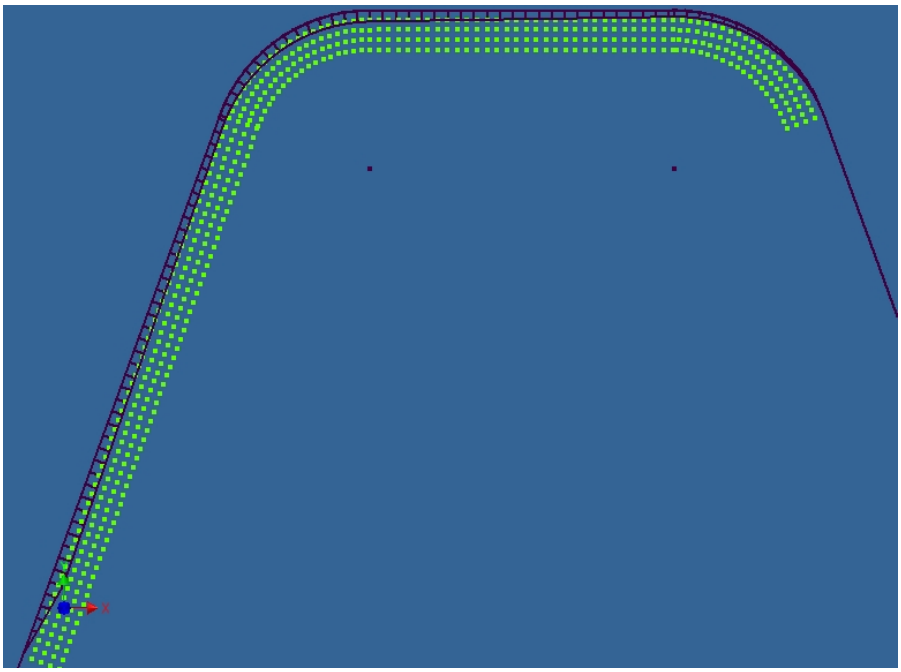
Η διαδικασία συνεχίζει στη γραμμή '`Antikeimena(3)`' στο τόξο '`Antikeimena(4)`' και στη γραμμή '`Antikeimena(5)`' εφόσον υπάρχει απόβλιττο. Οι διεργασίες που γίνονται είναι ίδιες με αυτές που περιγράφηκαν παραπάνω ανάλογα με το αν ο κώδικας συναντάει ευθεία γραμμή ή τόξο. Το τελικό αποτέλεσμα της διαδικασίας φαίνεται στο σχήμα 3.27.



Σχήμα 3.25 Η εφαπτόμενη και η γραμμή που έχει έρθει κάθετη σε αυτή.



Σχήμα 3.26 Δημιουργία γραμμών (constraints) στο τόξο



Σχήμα 3.27 Το αποτέλεσμα της διαδικασίας δημιουργίας γραμμών σε όλο το προφίλ

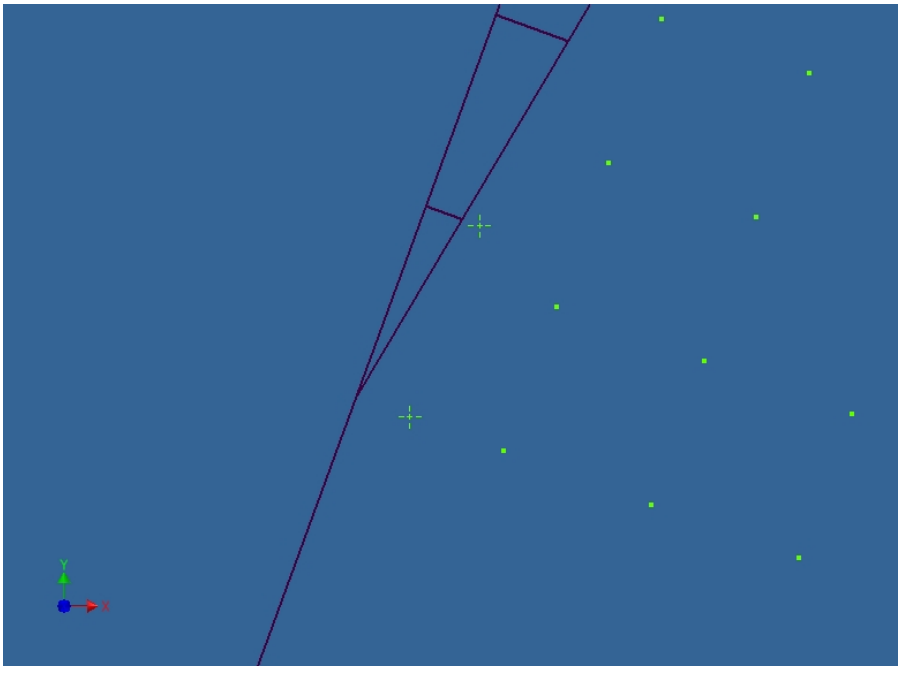
3.4.4 Δημιουργία παραλληλογράμμων και εξέταση αν τα σημεία του πλέγματος ανήκουν σε αυτά

Σε αυτό το σημείο του κώδικα θα δημιουργηθούν κάποια παραλληλόγραμμα με τρόπο που θα αναλυθεί παρακάτω και θα ελεγθεί ποια σημεία του πλέγματος επικαλύπτονται από αυτά τα παραλληλόγραμμα.

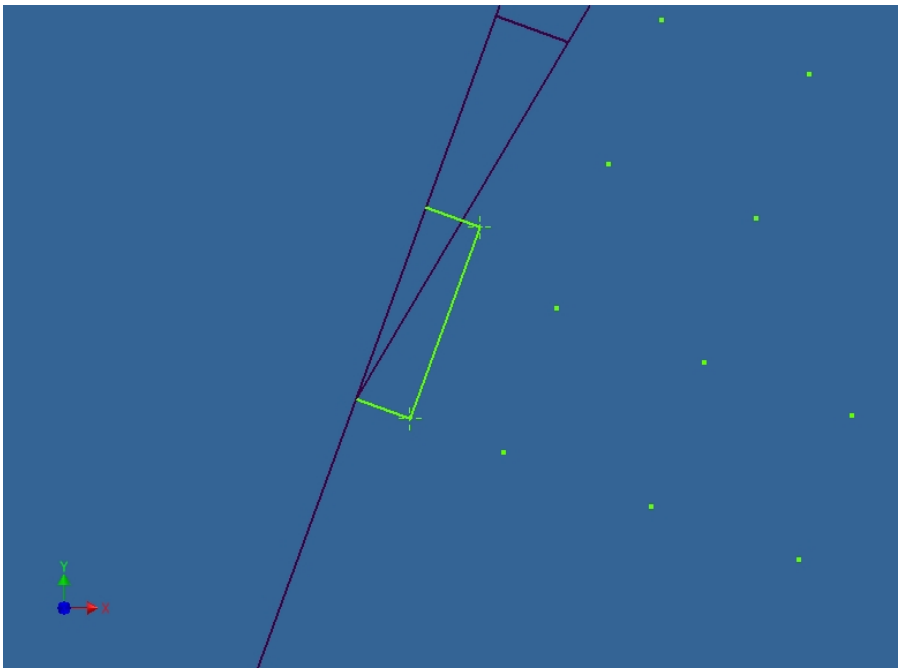
Αρχικά ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν στη συνέχεια. Έπειτα ο κώδικας συναντάει μία επαναληπτική διαδικασία (γραμμή 1312) που ξεκινάει από το '`i=0`' και φτάνει έως το '`i1-1`'. Η μεταβλητή '`i1`' έχει πάρει την τιμή της κατά τη δημιουργία των γραμμών (constraints) στην προηγούμενη διαδικασία και είναι ίση με τις γραμμές που δημιουργήθηκαν με αρχικό σημείο πάνω στη γραμμή '`Antikeimena(1)`'. Με το που μπαίνει στην επαναληπτική διαδικασία καταχωρεί στη μεταβλητή '`mesomikos`' την τιμή: '`mesomikos = (pinakasxyl(i, 2) + pinakasxyl(i + 1, 2)) / 2`' δηλαδή το μισό άθροισμα των μηκών δύο διαδοχικών γραμμών. Εάν η τιμή της '`mesomikos`' είναι διάφορη του μηδενός, δηλαδή εάν υπάρχουν όντως γραμμές (η έστω μια γραμμή) τότε ξεκινάει η δημιουργία τεσσάρων γραμμών που θα είναι και οι τέσσερεις πλευρές του παραλληλογράμμου. Ένα παραλληλόγραμμο ορίζεται από 3 σημεία. Τα δύο που είναι ήδη γνωστά είναι η αρχή της πρώτης γραμμής που έχει ήδη δημιουργήθεί στο προηγούμενο κομμάτι του κώδικα και η αρχή της επόμενης, δηλαδή τα στοιχεία '`i`' και '`i+1`' του πίνακα '`PinakasLines()`'. Το άλλο σημείο τώρα, αρχική εκτίμηση ήταν να απέχει κάθετη απόσταση από τη γραμμή '`Antikeimena(1)`' ίση με '`mesomikos`'. Όμως επειδή το παραλληλόγραμμο τότε θα ήταν σχετικά μικρό και δε θα επικάλυπτε αρκετά σημεία του πλέγματος, δίνεται στο τρίτο σημείο η τιμή '`mesomikos*f`' όπου '`f`' ένας ακέραιος που ορίζει το πόσο θα είναι το ύψος του παραλληλογράμμου και έχει την τιμή 3. Εάν για κάποιο λόγο θελήσει κάποιος να μεγαλώσει

τα παραλληλόγραμμα δεν έχει παρά να μεγαλώσει την τιμή της 'f'. Για να ορισθεί τώρα το σημείο αυτό να απέχει κάθετη απόσταση, προσθέτεται στην τιμή της μεταβλητής X του αρχικού σημείου της γραμμής 'PinakasLines(i)' η τιμή 'mesomikos * Cos(gwnia) * f' (γραμμή 1316) και από την τιμή της μεταβλητής Y του αρχικού σημείου της γραμμής 'PinakasLines(i)' αφαιρείται το 'mesomikos * Sin(gwnia) * f' (γραμμή 1317) όπου η μεταβλητή 'gwnia' είναι η γωνία που σχηματίζει η γραμμή 'PinakasLines(i)' με τον άξονα των X και είναι 20^o. Το σημείο αυτό το ονομάζεται 'opoint1'. Έπειτα δημιουργείται το αντίστοιχο σημείο προχωρώντας μια γραμμή και ονομάζεται 'oPointz'. Για καλύτερη κατανόηση τα σημεία 'opoint1' και 'oPointz' φαίνονται στο σχήμα 3.28.

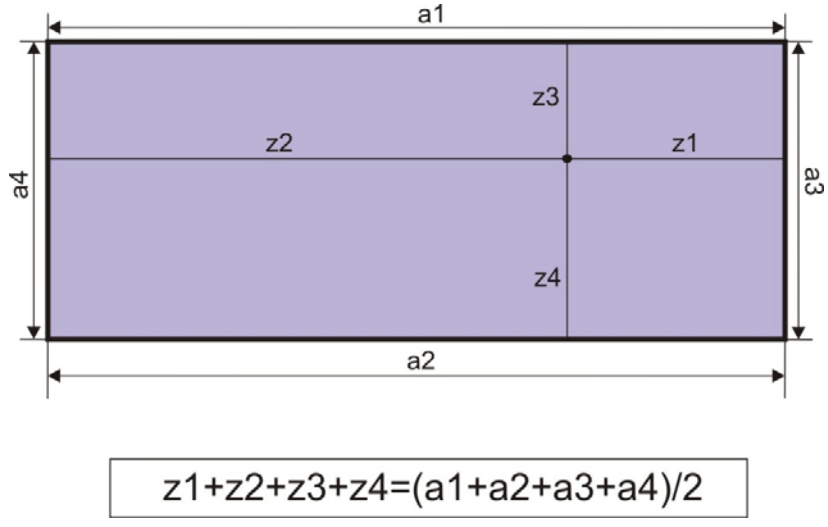
Επόμενη δουλειά είναι να δημιουργηθεί το παραλληλόγραμμο. Εφόσον λοιπόν υπάρχουν και οι τέσσερεις κορυφές του, ο κώδικας απλά τις ενώνει με γραμμές οι οποίες ονομάζονται 'oLinez1' 'oLinez2' 'oLinez3' και 'oLinez4' αντίστοιχα και φαίνονται στο σχήμα 3.29. Επόμενο βήμα είναι να ελεγχθούν ποια σημεία του πλέγματος βρίσκονται μέσα σε αυτό το παραλληλόγραμμο. Από την γεωμετρία είναι γνωστό πως για κάθε σημείο εντός ενός παραλληλογράμμου ισχύει πως το άθροισμα των κάθετων αποστάσεων του από τις πλευρές του παραλληλογράμμου είναι ίσο με το μισό της περιμέτρου κάτι που φαίνεται στο σχήμα 3.30. Άρα η διαδικασία που ακολουθεί ο κώδικας είναι να μετράει για κάθε σημείο το άθροισμα των κάθετων αποστάσεων από τις τέσσερεις πλευρές του παραλληλογράμμου και αν αυτό είναι ίσο ή μικρότερο (επειδή ίσως το πρόγραμμα απλοποιήσει ορισμένα δεκαδικά προς τα κάτω) του μισού του αθροίσματος των πλευρών τότε αυξάνεται η τρίτη στήλη του πίνακα 'GridPoints()' κατά μία μονάδα. Ο πίνακας 'GridPoints()' είναι ένας πίνακας που περιέχει πληροφορίες για όλα τα σημεία του πλέγματος. Στην πρώτη στήλη του έχει τις συντεταγμένες X, στη δεύτερη τις συντεταγμένες Y και στην τρίτη αποθηκεύει την πληροφορία της επικάλυψης από τα παραλληλόγραμμα.



Σχήμα 3.28 Τα σημεία 'opoint1' και 'oPointz'



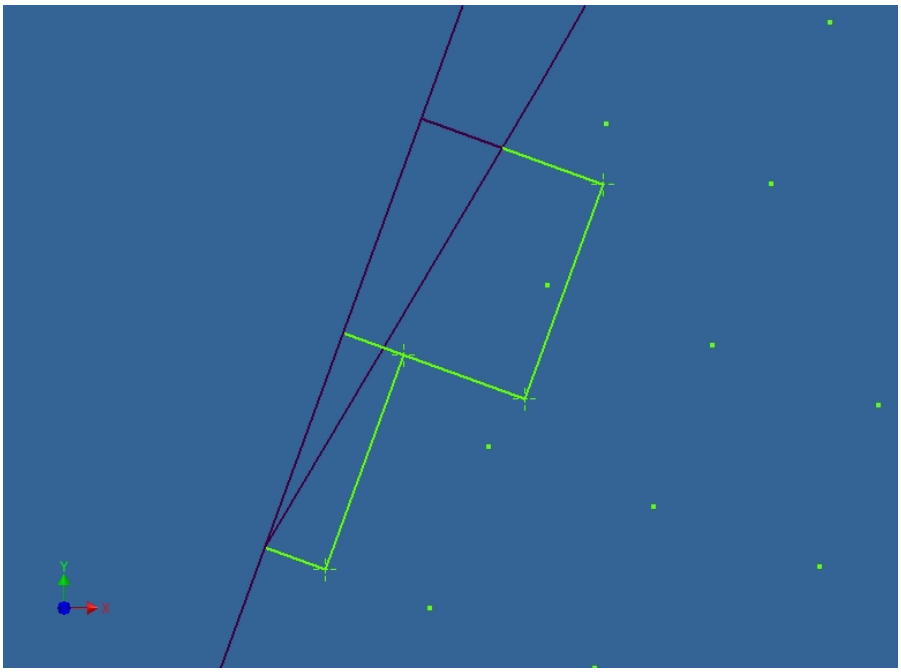
Σχήμα 3.29 Το πρώτο παραλληλόγραμμο



Σχήμα 3.30 Σχέση για κάθε σημείο εντός παραλληλογράμμου

Αφού γίνει ο έλεγχος λοιπόν για το πρώτο παραλληλόγραμμο ξεκινάει η δημιουργία του δεύτερου ακριβώς όπως και πριν όπως φαίνεται στο σχήμα 3.31. Έτσι δημιουργούνται όλα τα παραλληλόγραμμα και γίνεται ο έλεγχος των σημείων του πλέγματος κάθε φορά για κάθε παραλληλόγραμμο ξεχωριστά. Επειδή όμως η διαδικασία ελέγχου του κάθε σημείου του πλέγματος είναι αρκετά χρονοβόρα ήταν επιτακτική η ανάγκη να υπάρχει κάποιος έλεγχος για να ελέγχονται μόνο τα σημεία που υπάρχει πιθανότητα να επικαλύπτονται. Αυτός ο έλεγχος

διαφέρει ανάλογα με το ποια γραμμή του προφίλ εξετάζει ο κώδικας τη συγκεκριμένη στιγμή. Στην περίπτωση που εξετάζεται, δηλαδή αν ο κώδικας βρίσκεται στη γραμμή '`Antikeimena(1)`', μετά τη δημιουργία του παραλληλογράμμου ο κώδικας έχει έναν βρόγχο της μορφής 'Do While'. Δηλαδή να γίνεται η διαδικασία του ελέγχου των σημείων του πλέγματος (εάν αυτά ανήκουν μέσα στο παραλληλόγραμμο), εφόσον η συντεταγμένη Y του σημείου του πλέγματος που εξετάζεται είναι μικρότερη από τη συντεταγμένη Y του αρχικού σημείου της γραμμής '`PinakasLines(i+1)`' δηλαδή της κορυφής του παραλληλογράμμου με το μεγαλύτερο Y. Αυτό είναι εφικτό καθώς τα σημεία του πλέγματος δημιουργήθηκαν από κάτω προς τα πάνω έτσι τα σημεία ελέγχονται ανά σειρά. Αν αυτός ο έλεγχος βγει αρνητικός τότε ο κώδικας βγαίνει από το βρόγχο, ένας μετρητής αυξάνεται κατά ένα και πάει στο επόμενο σημείο του πλέγματος για να ελεγθεί. Επίσης μέσα στο βρόγχο 'Do While' που περιγράφηκε υπάρχει και άλλος ένας έλεγχος σύμφωνα με τον οποίο το πρόγραμμα μετράει τις αποστάσεις του σημείου από τις πλευρές του παραλληλογράμμου μόνο αν η συντεταγμένη Y του σημείου του πλέγματος είναι μεγαλύτερη η ίση της συντεταγμένης Y του σημείου '`opoint1`' το οποίο είναι και η κορυφή του παραλληλογράμμου με το μικρότερο Y. Τέλος γίνεται άλλος ένας έλεγχος. Όταν δημιουργείται ένα παραλληλόγραμμο, ξεκινάει ο έλεγχος για το ποια σημεία επικαλύπτει. Αποθηκεύεται λοιπόν στην μεταβλητή '`boing`' το πρώτο σημείο του πλέγματος που επικαλύπτεται από το τρέχον παραλληλόγραμμο. Έτσι όταν δημιουργείται το επόμενο παραλληλόγραμμο ο έλεγχος για τα σημεία που επικαλύπτονται δεν ξεκινάει από την αρχή του πλέγματος αλλά από το σημείο του πλέγματος με τιμή '`boing`', με αποτέλεσμα να ελέγχονται μόνο τα πιθανά σημεία που μπορεί να υπάρξει επικάλυψη. Έτσι εξοικονομείται πολύς χρόνος και ο κώδικας είναι αρκετά πιο γρήγορος.

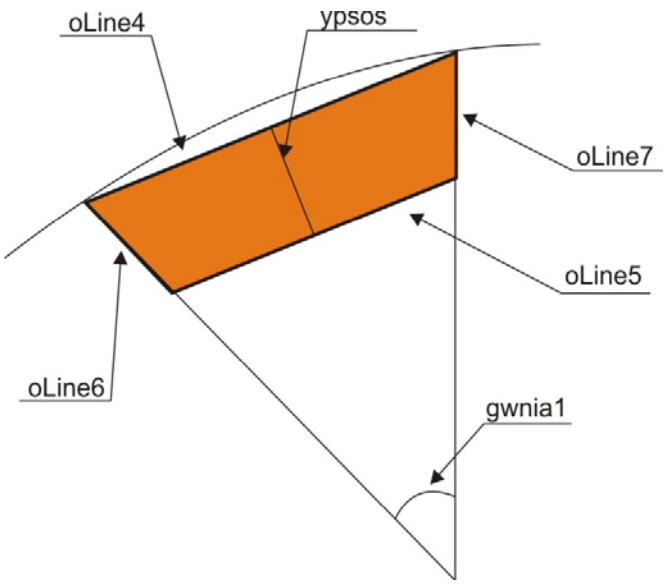


Σχήμα 3.31 Δημιουργία δεύτερου παραλληλογράμμου

Όταν τώρα τελειώσει η δημιουργία των παραλληλογράμμων στη γραμμή '`Antikeimena(1)`' ο κώδικάς προχωράει στο τόξο '`Antikeimena(2)`'. Εκεί τα πράγματα αλλάζουν λιγάκι καθώς οι γραμμές (constraints) που είχαν δημιουργηθεί προηγουμένως δεν είναι παράλληλες. Και πάλι θα δημιουργηθεί ένα παραλληλόγραμμο απλά θα υπολογισθεί διαφορετικά το ύψος του.

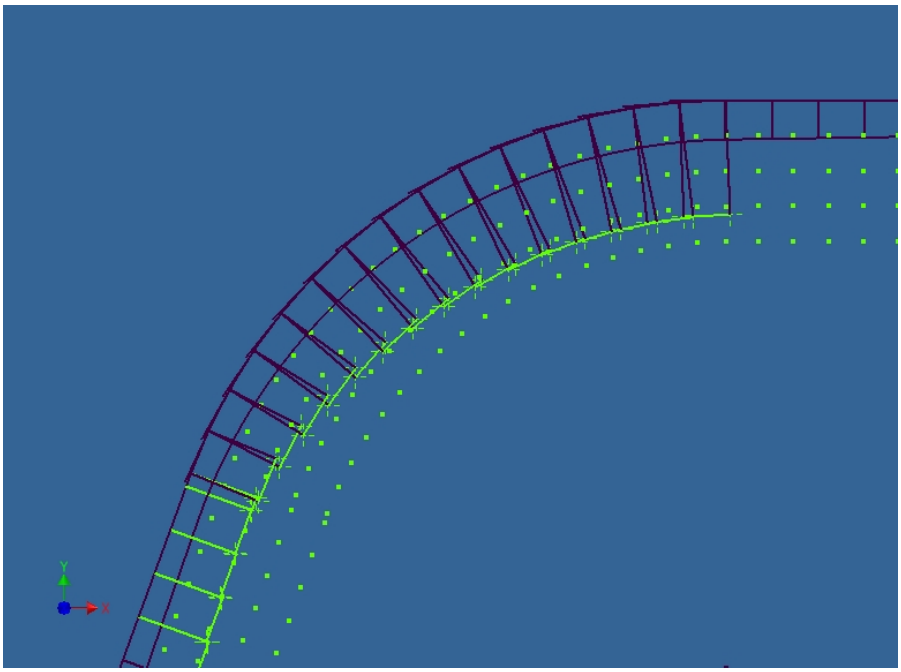
Αρχικά υπολογίζεται η μεταβλητή ‘mesomikos’ όπως και προηγουμένως. Έπειτα δημιουργείται ένα τραπέζιο που ορίζεται από τέσσερα σημεία. Από το αρχικό σημείο της γραμμής ‘PinakasLines(i)’, από το αρχικό σημείο της γραμμής ‘PinakasLines(i+1)’ και από δύο νέα σημεία ‘opoint1’ και ‘opoint2’ οι συντεταγμένες των οποίων προκύπτουν από τα αρχικά σημεία των γραμμών ‘PinakasLines(i)’ και ‘PinakasLines(i+1)’ μετατοπισμένα σε σχέση με τη μεταβλητή ‘mesomikos’. Στο σχήμα 3.32 φαίνονται οι πλευρές του τραπεζίου με τις ονομασίες τους όπως είναι στον κώδικα, που δημιουργούνται όταν ενωθούν τα σημεία.

Σκοπός είναι να υπολογισθεί το εμβαδό αυτού του τραπεζίου και να δημιουργηθεί ένα παραλληλόγραμμο με ίδιο εμβαδό. Γεωμετρικά υπολογίζεται το ύψος του τραπεζίου (‘ypsos = Sin(gwnia1) * oline6.Length’ όπου ‘oline6.Length’ είναι το μήκος της γραμμής ‘oline6’) και έπειτα υπολογίζεται το εμβαδό του που είναι [(βάση μεγάλη + βάση μικρή)*ύψος]/2 δηλαδή ‘emvado = ((oLine4.Length + oLine5.Length) * ypsos) / 2’. Είναι γνωστό πως το εμβαδό ενός παραλληλογράμμου είναι το γινόμενο δύο διαδοχικών πλευρών. Η μία είναι η γραμμή ‘oline4’ όπως αυτή φαίνεται στο σχήμα 3.32. Άρα το μήκος της άλλης θα είναι ‘h = emvado / oLine4.Length’. Εφόσον τώρα είναι γνωστό το μήκος της πλευράς του παραλληλογράμμου ο κώδικας το σχεδιάζει όπως και όταν ήταν στη γραμμή ‘Antikeimena(1)’ με τη διαφορά πως αντί την τιμή ‘mesomikos’ τοποθετείται η τιμή ‘h’. Έτσι έχει δημιουργηθεί ένα παραλληλόγραμμο που έχει ίδιο εμβαδό με το τραπέζιο που υπήρχε.



Σχήμα 3.32 Με το πορτοκαλί χρώμα φαίνεται το τραπέζιο που δημιουργείται.

Η διαδικασία για τον έλεγχο της επικάλυψης των σημείων του πλέγματος γίνεται όπως και πριν με τη μόνη διαφορά πως ο έλεγχος τέλους του βρόγχου δεν είναι γεωμετρικός. Το τελικό αποτέλεσμα για το τόξο φαίνεται στο σχήμα 3.33.



Σχήμα 3.33 Δημιουργία παραλληλογράμμων στο τόξο ‘Antikeimena(2)’

Με παρόμοιο τρόπο γίνονται τα παραλληλόγραμμα και στις επόμενες γραμμές και ταυτόχρονα υπολογίζονται και οι επικαλύψεις των σημείων του πλέγματος όπου αυτές υπάρχουν και αποθηκεύονται στον πίνακα ‘GridPoints()’

Τέλος καλείται η συνάρτηση ‘Write2File’ η οποία τυπώνει σε ένα αρχείο κειμένου με ονομασία ‘Chip" & ch & "_Plane" & kl & ".txt’ όλα τα στοιχεία του πίνακα ‘GridPoints()’ δηλαδή τις συντεταγμένες X και Y του κάθε σημείου του πλέγματος καθώς και την πληροφορία για την επικάλυψη.

4. ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

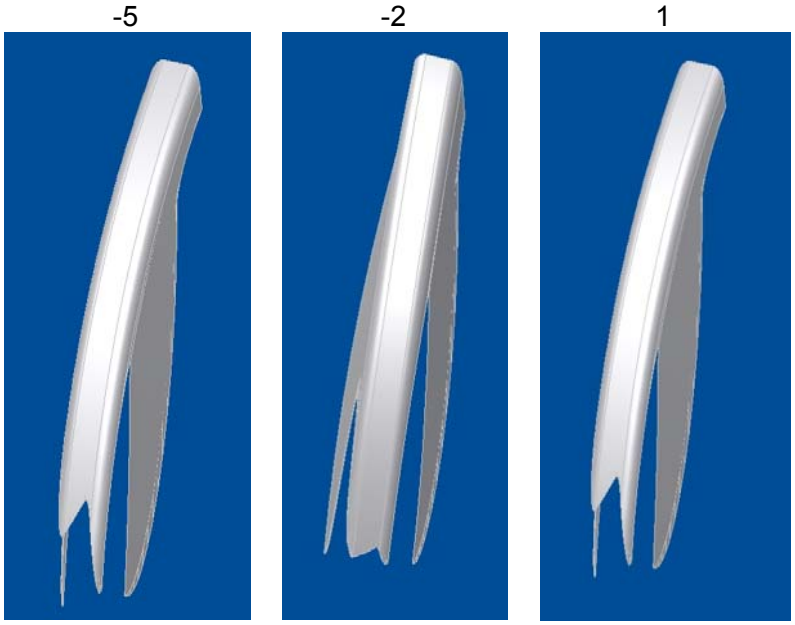
4.1 Περιγραφή τρόπου παρουσιάσεως

Σε αυτό το κεφάλαιο γίνεται η παρουσίαση των αποτελεσμάτων που έχουν εξαχθεί στα αρχεία κειμένου (text) έπειτα από την ολοκλήρωση του κώδικα που περιγράφηκε παραπάνω. Αρχικά εκτελείται το πρόγραμμα που έχει δημιουργηθεί στη Matlab, το οποίο παίρνει ως δεδομένα (input data) ένα από τα αρχεία κειμένου που δημιουργήθηκαν μετά το πέρας του κώδικα. Επίσης για την καλύτερη απεικόνιση και συνεπώς κατανόηση των αποτελεσμάτων στο πρόγραμμα πρέπει να εισαχθεί και το αρχείο που περιέχει το προφίλ ενός δοντιού του κοπτικού που χρησιμοποιήθηκε για την εξαγωγή των αποβλίττων που επεξεργάστηκε ο κώδικας, το οποίο έχει δημιουργηθεί από τη συνάρτηση '`MakeToothPerigram`'. Το πρόγραμμα προτρέπει το χρήστη να του υποδείξει που βρίσκονται αυτά τα δύο αρχεία. Στη συνέχεια εκτελείται το πρόγραμμα.

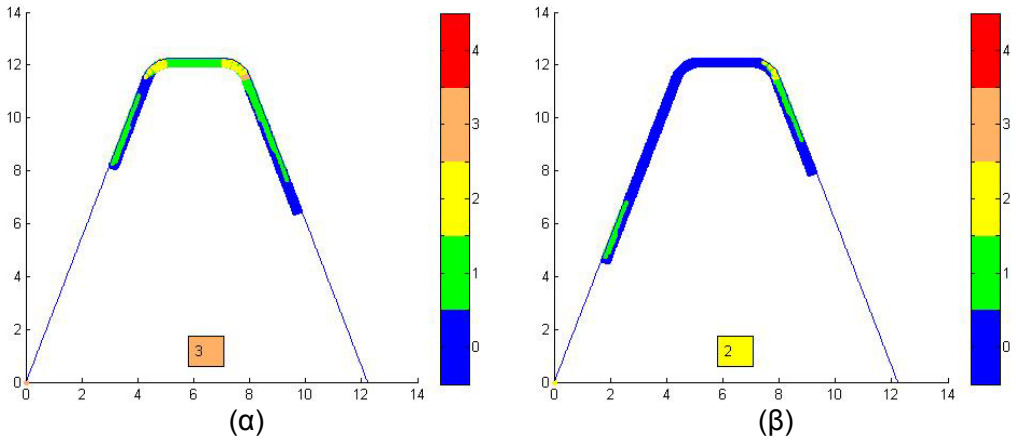
Στα αποτελέσματα που παρουσιάζονται υπάρχει μια ενδεικτική χρωματική κλίμακα στα δεξιά της κάθε φωτογραφίας που μας υποδεικνύει σε ποια σημεία υπάρχει επικάλυψη του αποβλίττου και πόσο έντονη είναι αυτή ανάλογα με το πώς έχει χρωματιστεί το σημείο. Επίσης σε κάθε χρώμα αντιστοιχεί και ένας αριθμός. Στα σημεία που το χρώμα είναι μπλε δεν υπάρχει καθόλου απόβλιττο. Στα σημεία που το χρώμα είναι πράσινο υπάρχει απόβλιττο χωρίς επικάλυψη όμως. Στα κίτρινα σημεία υπάρχει επικάλυψη δηλαδή υπάρχει σύγκρουση δύο διαφορετικών μερών του αποβλίττου, ενώ όπως είναι επακόλουθο στα πορτοκαλί σημεία υπάρχει επικάλυψη από τρία μέρη και στα κόκκινα από τέσσερα. Επίσης χαμηλά στο κέντρο κάθε αποτελέσματος υπάρχει ένας αριθμός ο οποίος δείχνει τη μέγιστη επικάλυψη στην εκάστοτε θέση περιστροφής. Τέλος εφόσον τα αποτελέσματα βρίσκονται σε δισδιάστατη μορφή, ο χρήστης πληροφορείται και για το ακριβές σημείο που υπάρχει αυτή η επικάλυψη.

Το να παρατεθούν όλα τα αποτελέσματα από μια διαδικασία κοπής με ένα συγκεκριμένο module θα απαιτούσε περίπου τριάντα θέσεις κύλισης, κάθε μια από τις οποίες θα παρήγαγε ένα απόβλιττο όπου κάθε απόβλιττο θα μας έδινε γύρω στις τριάντα θέσεις περιστροφής δηλαδή σύνολο 900 αποτελέσματα. Κάτι τέτοιο περισσότερο θα κούραζε τον αναγνώστη παρά θα τον βοηθούσε στην κατανόηση των αποτελεσμάτων. Για αυτό το λόγο επιλέχθηκαν να παρουσιαστούν τα αποτελέσματα για όλες τις θέσεις περιστροφής, για μία ενδεικτική θέση κύλισης και το αποτέλεσμα της θέσης περιστροφής με το μεγαλύτερο βαθμό σύγκρουσης για δύο άλλες θέσεις κύλισης για τέσσερεις διαφορετικές παραλλαγές της διαδικασίας κοπής. Αυτές οι τέσσερεις παραλλαγές είναι η ευθεία ομόρροπη και ευθεία αντίρροπη διαμόρφωση οδοντώσεων καθώς και η υπό γωνία ομόρροπη και υπό γωνία αντίρροπη διαμόρφωση οδοντώσεων.

4.2 Παρουσίαση αποτελεσμάτων για ευθεία ομόρροπη διαμόρφωση οδοντώσεων



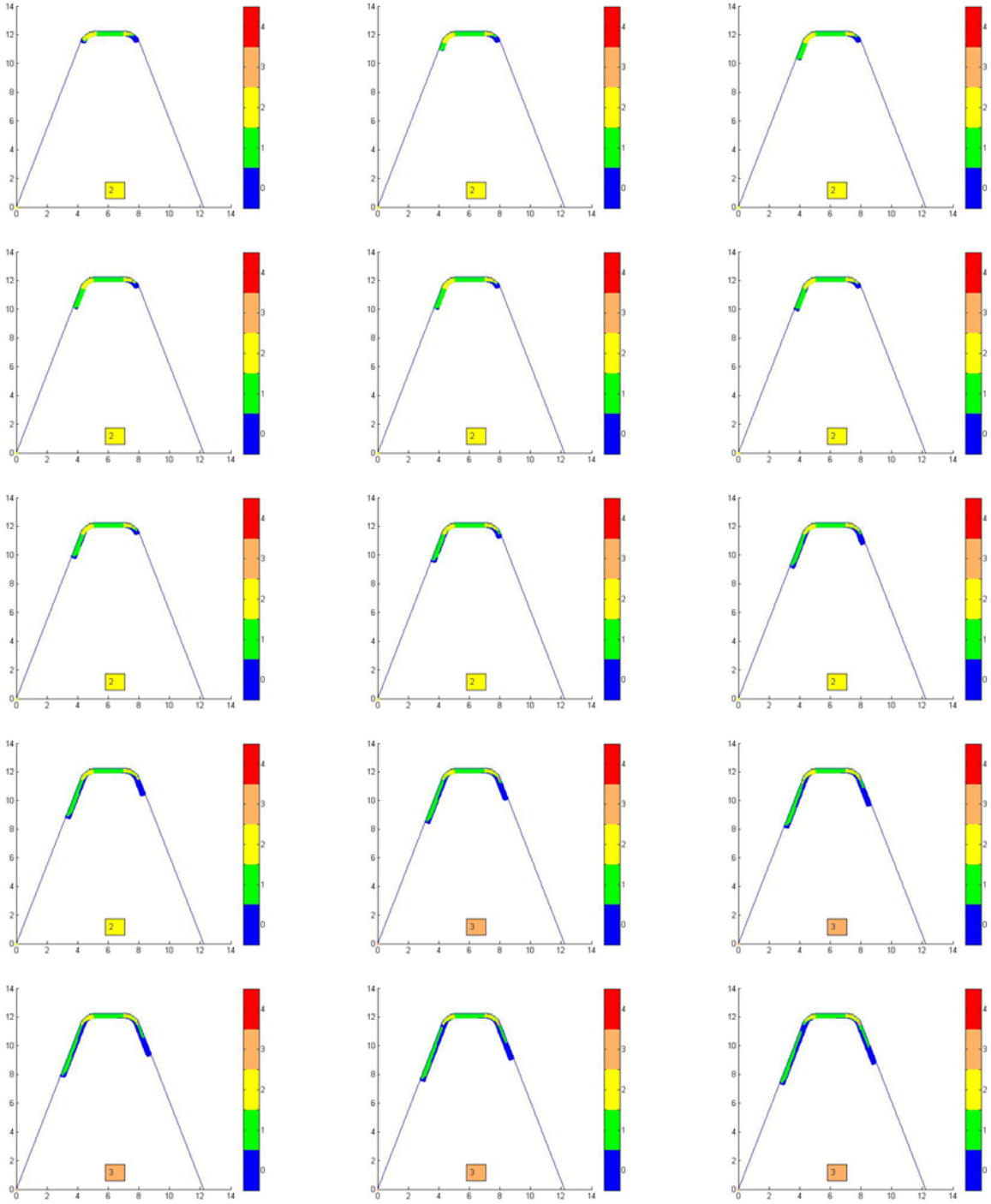
Σχήμα 4.1 Απόβλιττο για τρεις διαφορετικές θέσεις κύλισης

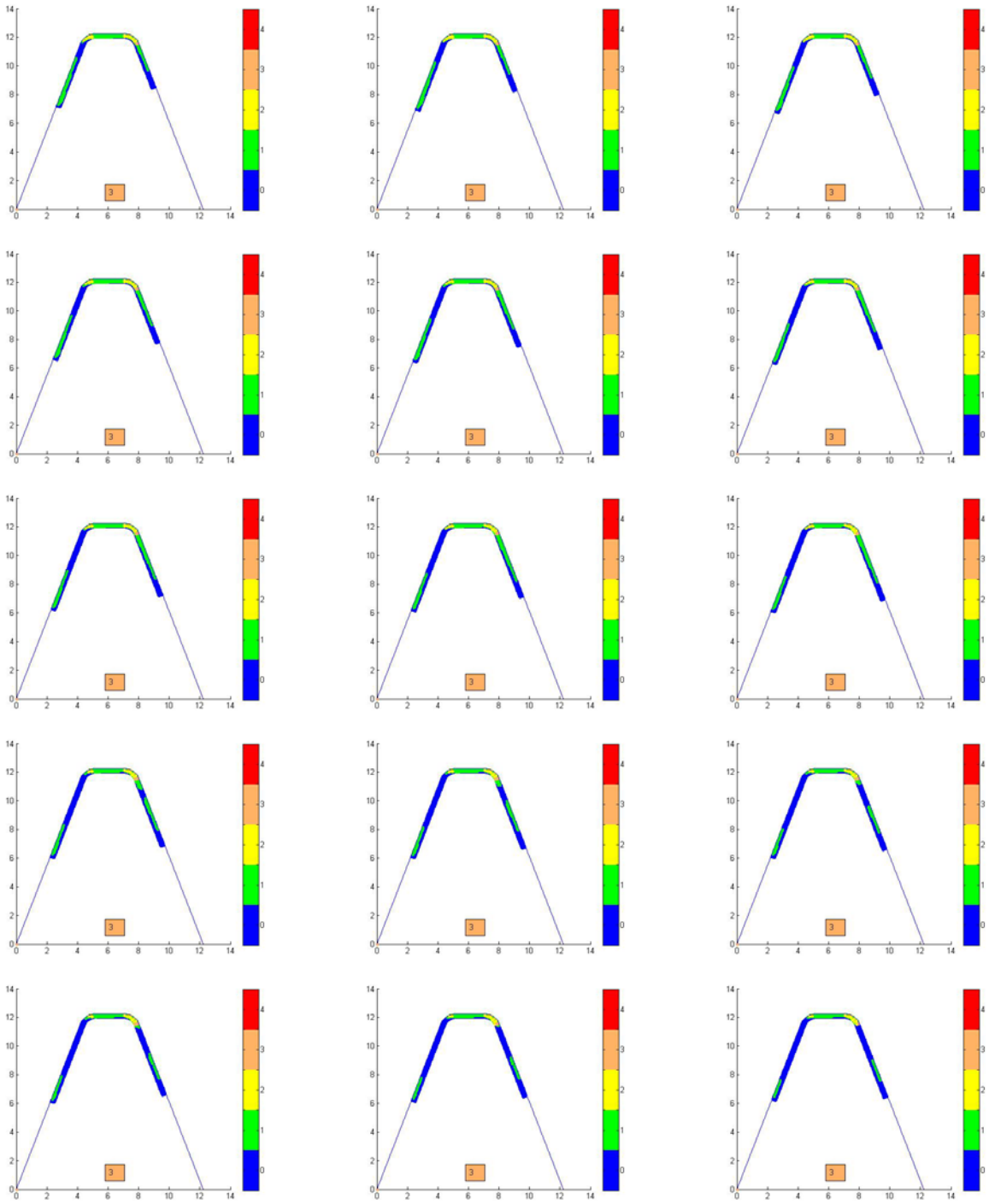


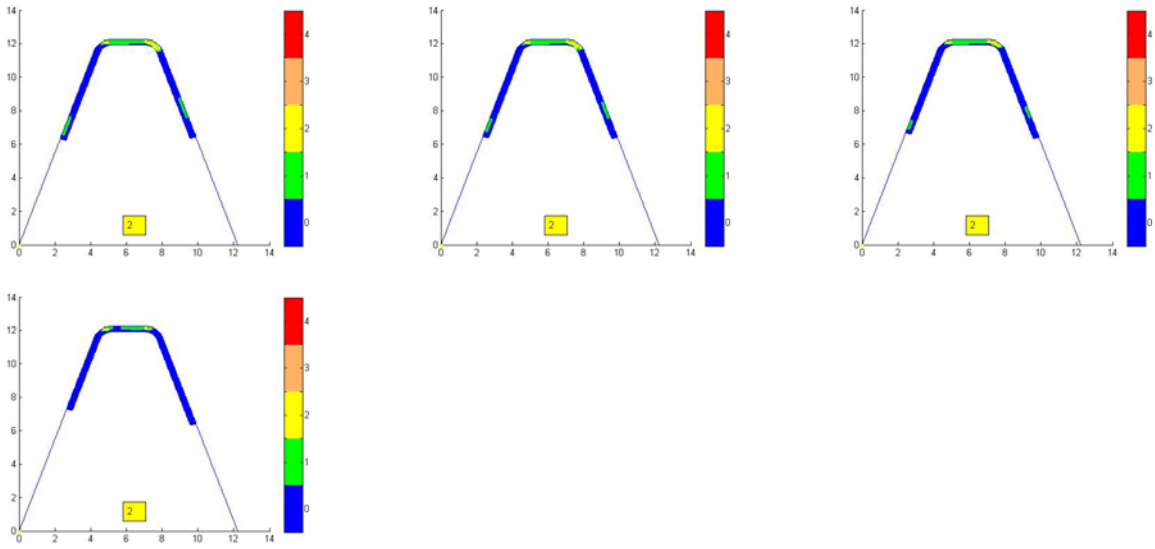
Σχήμα 4.2 Θέσεις περιστροφής με το μεγαλύτερο βαθμό σύγκρουσης για τις θέσεις κύλισης -5 (α) και 1 (β)

m=5mm , d_H=125mm, n_i=12, h_a=0^o, z_1=1, z_2=30, t=11mm, f_a=4mm/wrev

Τα αποτελέσματα για όλες τις θέσεις περιστροφής του αποβλίττου στη θέση κύλισης -2.

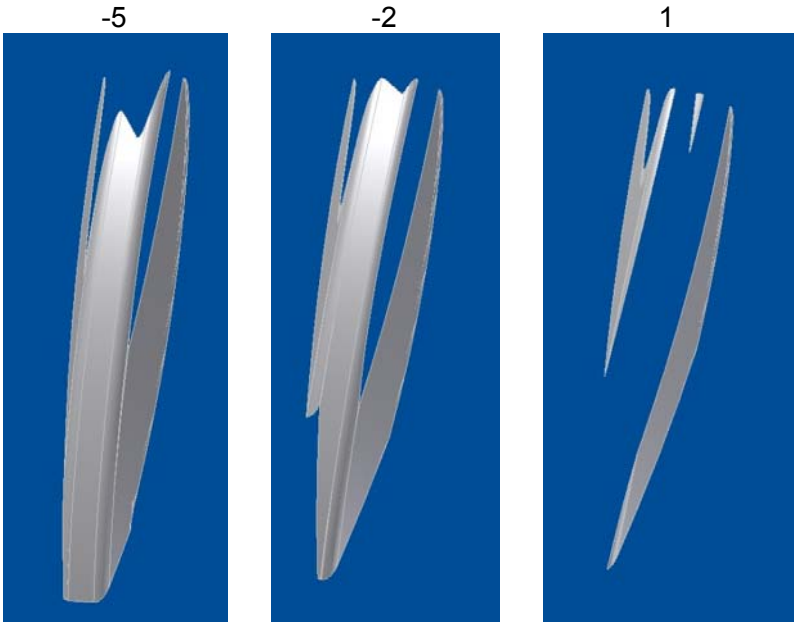




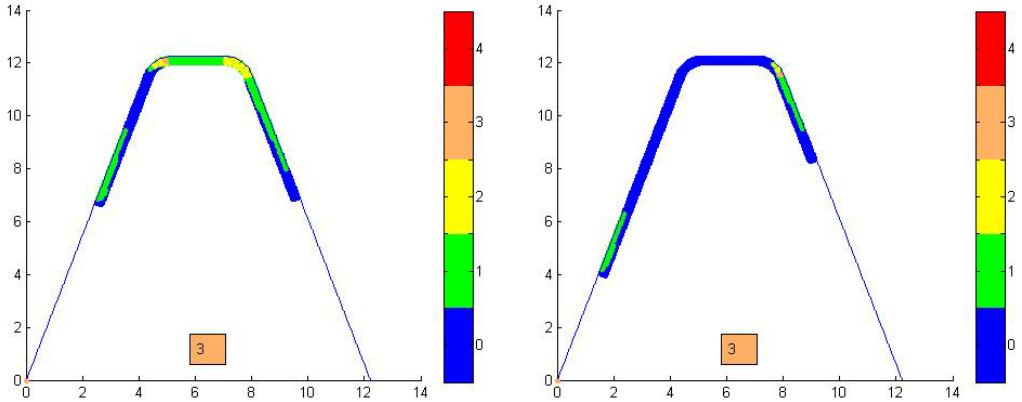


Σχήμα 4.3 Αποτελέσματα

4.3 Παρουσίαση αποτελεσμάτων για ευθεία αντίρροπη διαμόρφωση οδοντώσεων



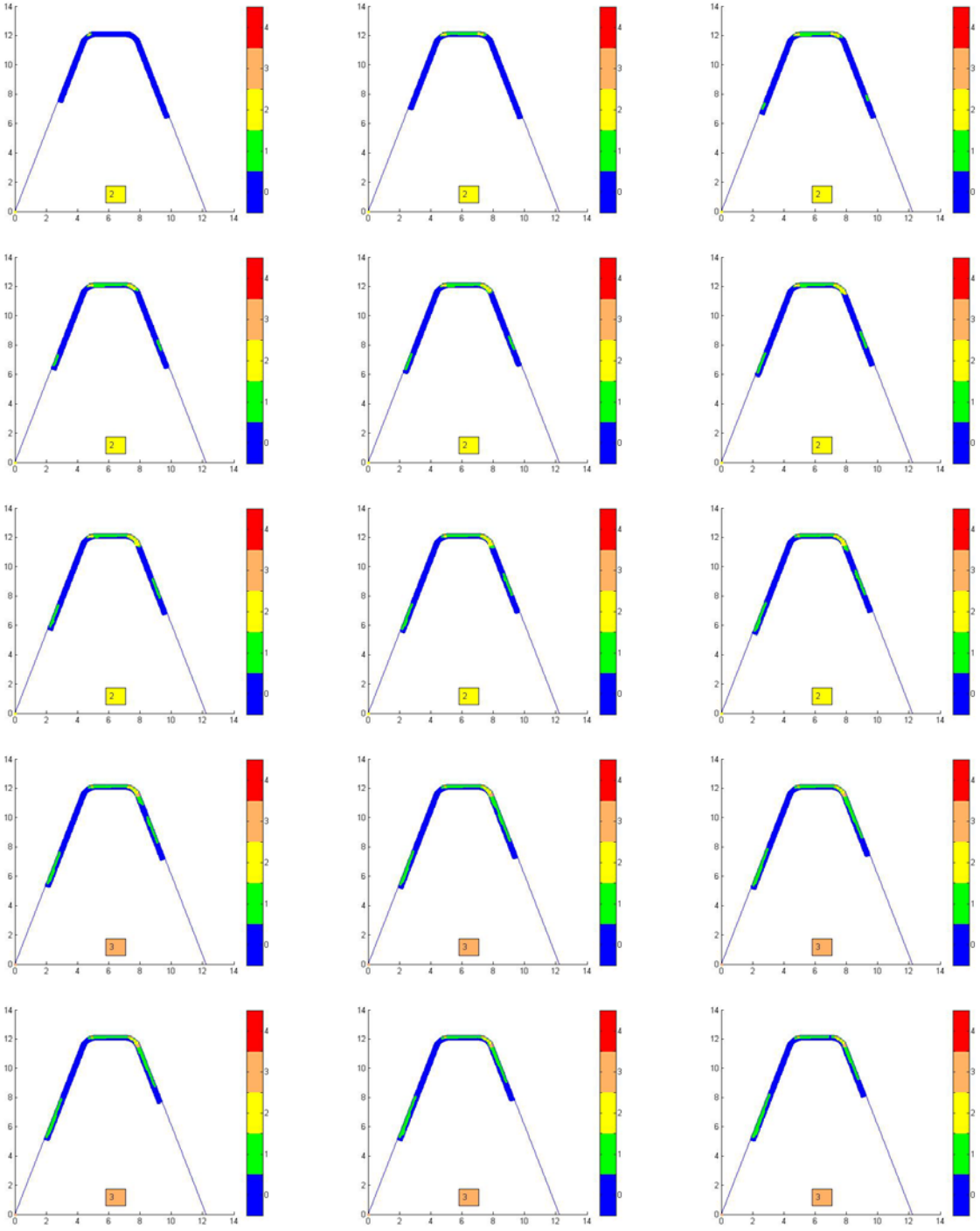
Σχήμα 4.4 Απόβλιττο για τρεις διαφορετικές θέσεις κύλισης

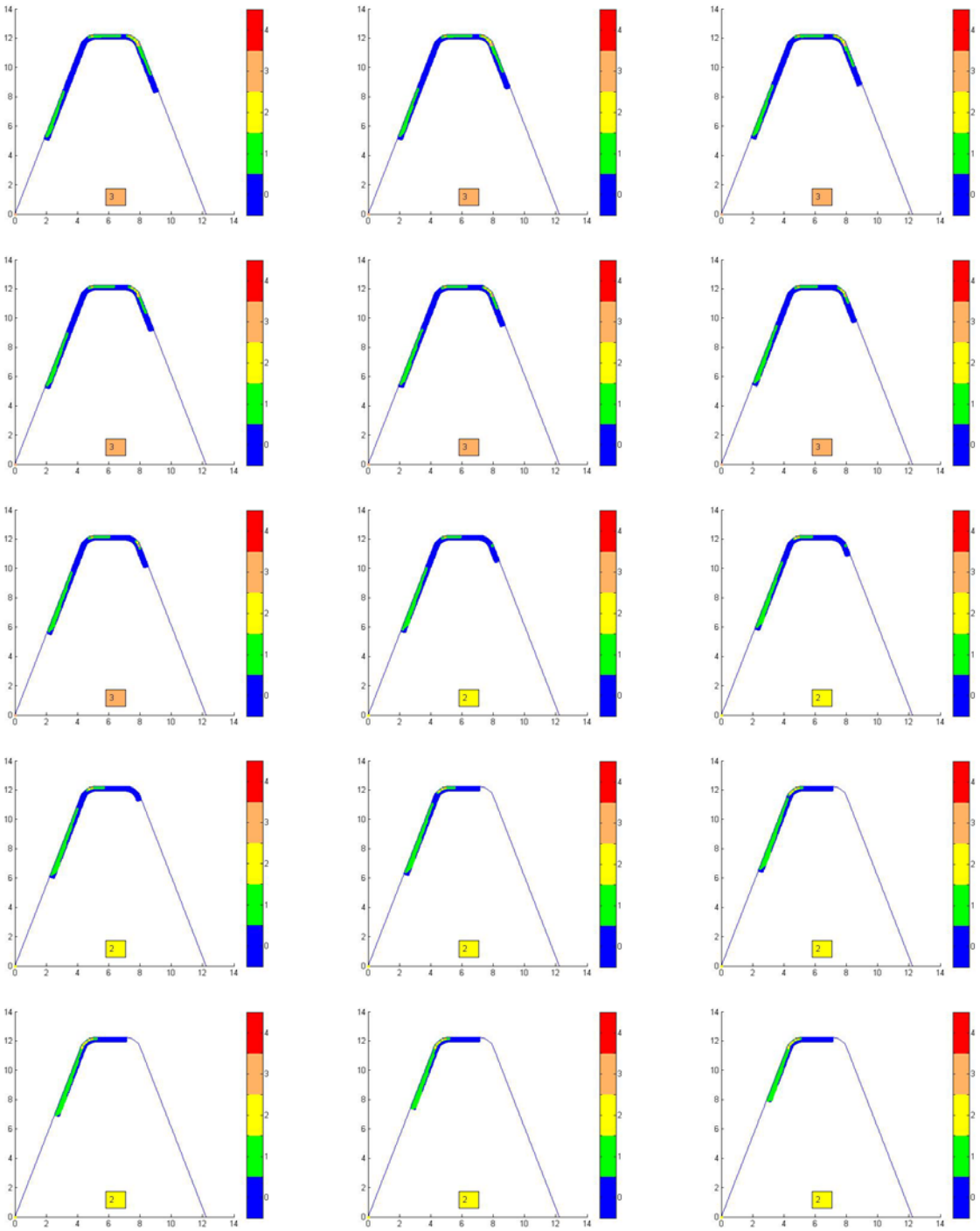


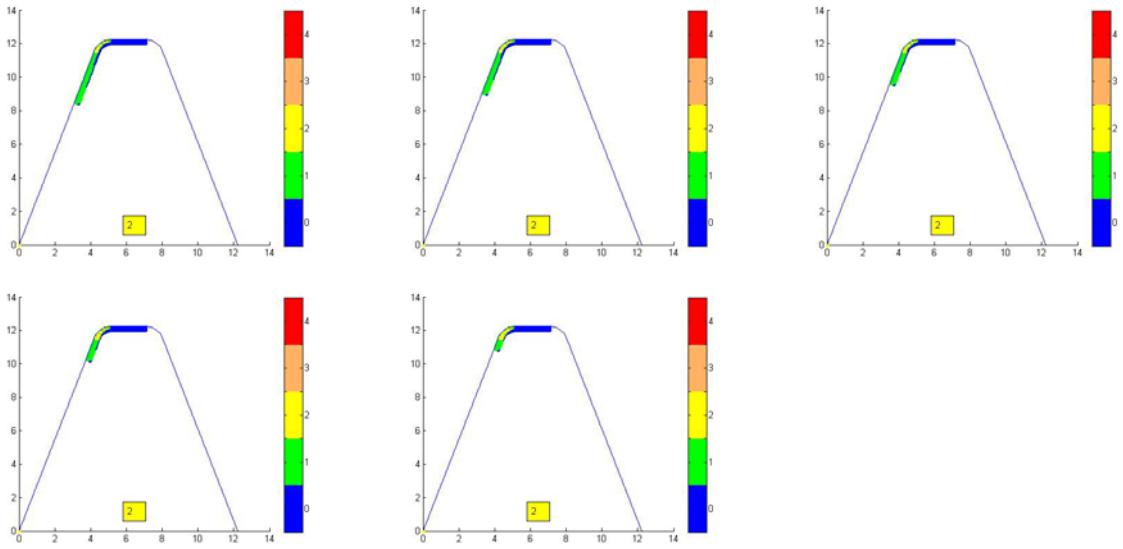
Σχήμα 4.5 Θέσεις περιστροφής με το μεγαλύτερο βαθμό σύγκρουσης για τις θέσεις κύλισης -5 (α) και 1 (β)

m=5mm , d_H=125mm, n_i=12, h_a=0^o, z_1=1, z_2=30, t=11mm, f_a=-4mm/wrev

Τα αποτελέσματα για όλες τις θέσεις περιστροφής του αποβλίττου στη θέση κύλισης -2.

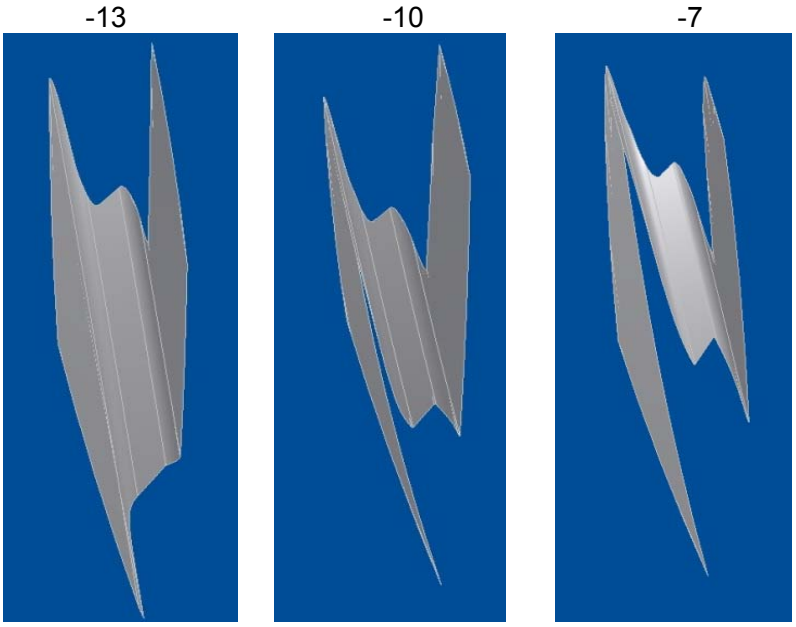



3
3
3
3
3
3
3
3
3
2
2
2
2
2
2

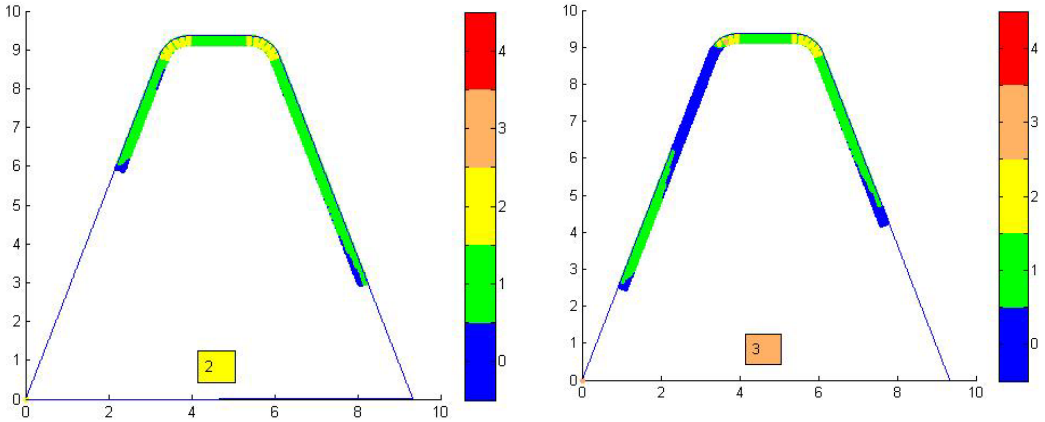


Σχήμα 4.6 Αποτελέσματα

4.4 Παρουσίαση αποτελεσμάτων για υπό γωνία ομόρροπη διαμόρφωση οδοντώσεων



Σχήμα 4.7 Απόβλιττο για τρεις διαφορετικές θέσεις κύλισης

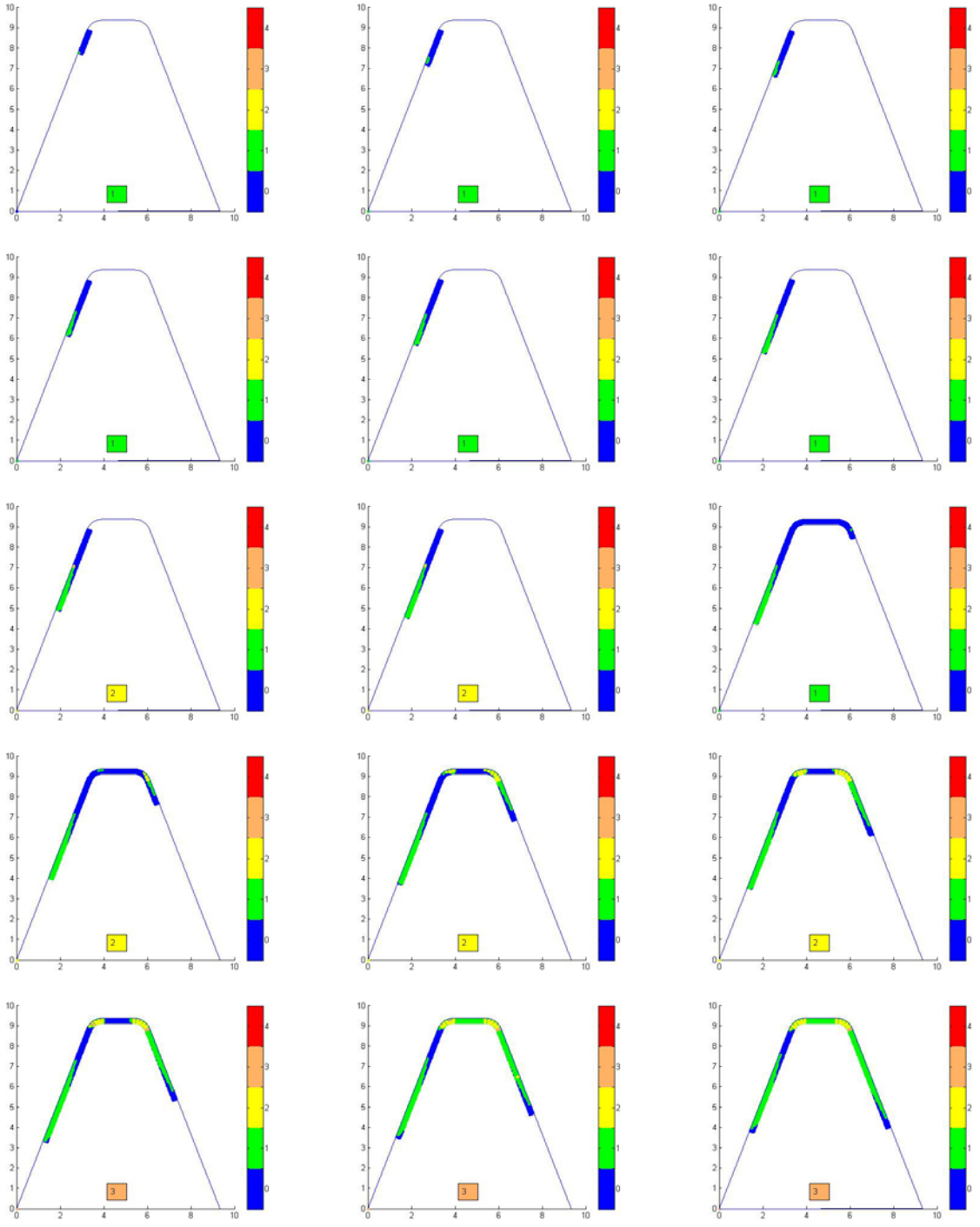


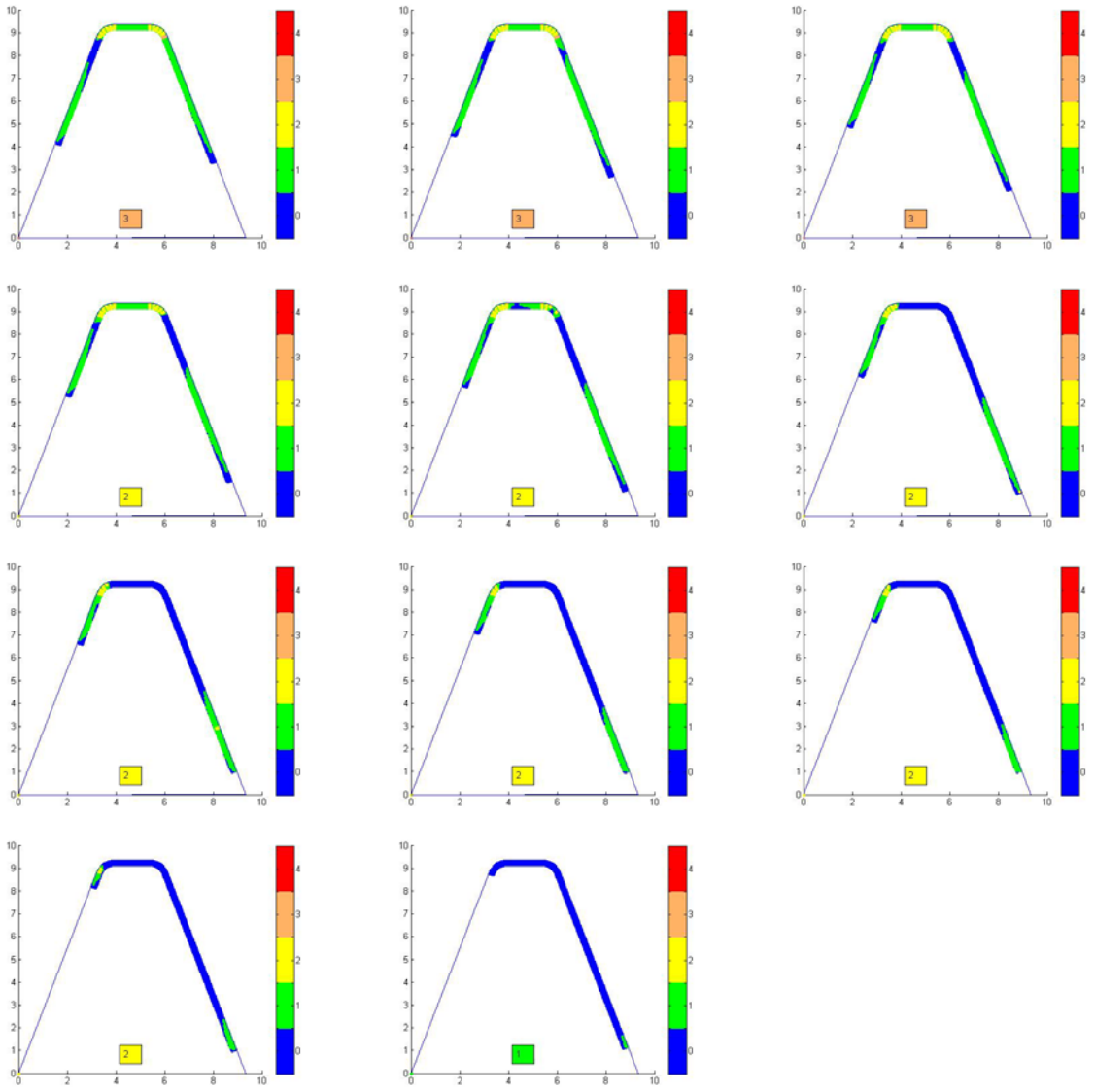
(α) (β)

Σχήμα 4.8 Θέσεις περιστροφής με το μεγαλύτερο βαθμό σύγκρουσης για τις θέσεις κύλισης -13 (α) και -7 (β)

m=3,82mm , d_H=147mm, n_i=6, h_a=30^o, z_1=1, z_2=24, t= 9,70mm, f_a=4mm/wrev

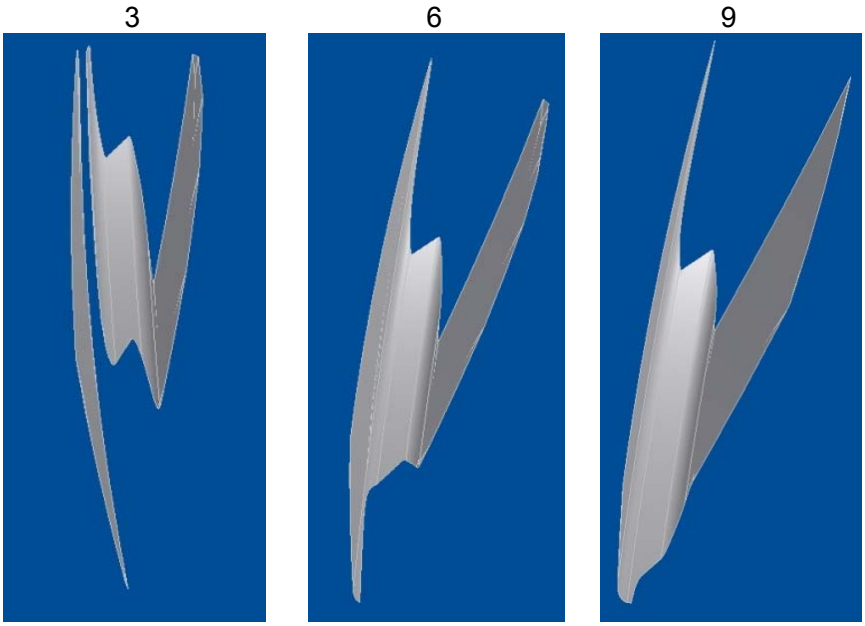
Τα αποτελέσματα για όλες τις θέσεις περιστροφής του αποβλίττου στη θέση κύλισης -10.



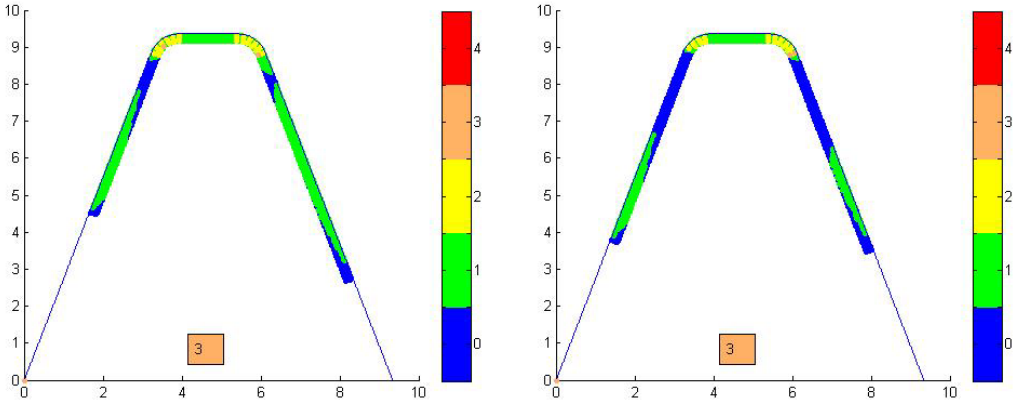


Σχήμα 4.9 Αποτελέσματα

4.4 Παρουσίαση αποτελεσμάτων για υπό γωνία αντίρροπη διαμόρφωση οδοντώσεων



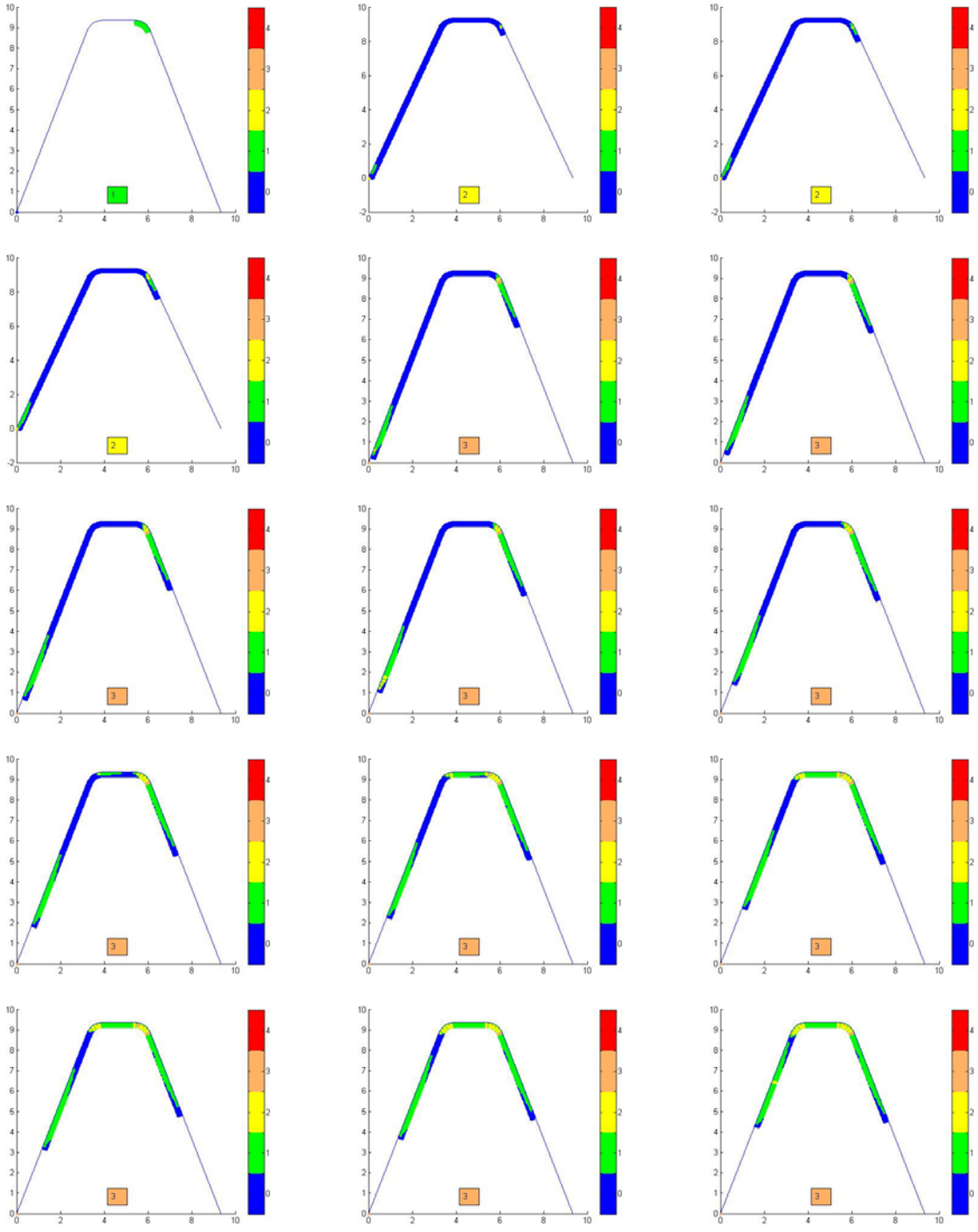
Σχήμα 4.10 Απόβλιττο για τρεις διαφορετικές θέσεις κύλισης

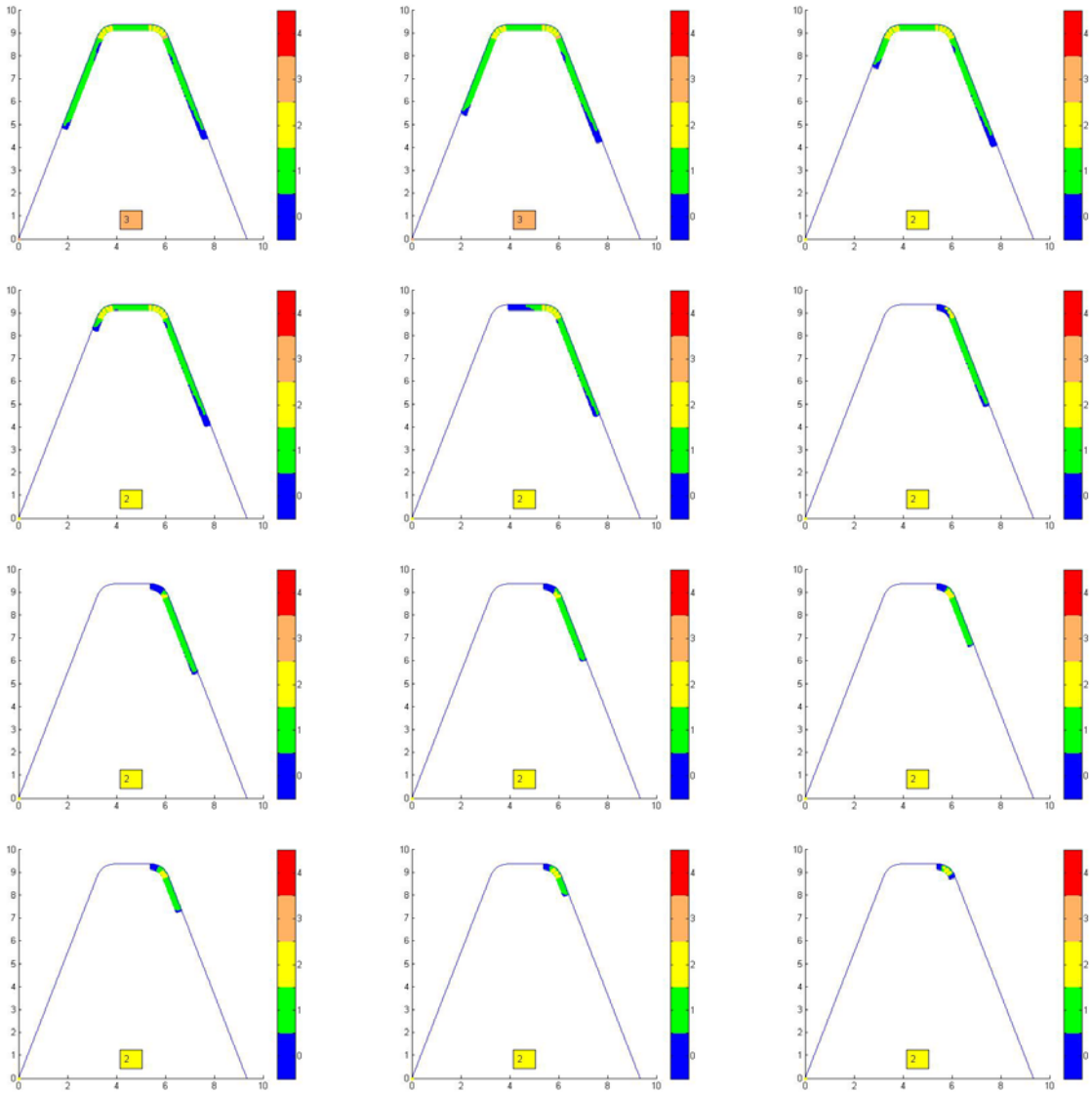


Σχήμα 4.11 Θέσεις περιστροφής με το μεγαλύτερο βαθμό σύγκρουσης για τις θέσεις κύλισης 3 (α) και 9 (β)

m=3,82mm , d_H=147mm, n_i=6, h_a=30^o, z_1=1, z_2=24, t= 9,70mm, f_a=-4mm/wrev

Τα αποτελέσματα για όλες τις θέσεις περιστροφής του αποβλίττου στη θέση κύλισης -10.





Σχήμα 4.12 Αποτελέσματα

5. ΣΥΝΟΨΗ

Με το πέρας αυτής της εργασίας έγινε εφικτή η παροχή μιας επιπλέον πληροφορίας για τον υπολογισμό της φθοράς του κοπτικού εργαλείου κατά τη διαδικασία διαμόρφωσης οδοντώσεων με κοπή.

Αρχικά παρατέθηκαν γενικότερες πληροφορίες για τη διαδικασία κοπής με φραιζάρισμα με κύλιση οδοντώσεων, για τη φθορά των μετάλλων και πιο συγκεκριμένα για τη φθορά του κοπτικού εργαλείου.

Το κύριο μέρος της εργασίας έγινε σε περιβάλλον CAD του προγράμματος Autodesk Inventor, σε προγραμματιστικό περιβάλλον Visual Basic for Applications και σε περιβάλλον του προγράμματος Matlab. Αυτός ο συνδυασμός επέφερε αποτελέσματα ακριβή και ευνόητα. Η ακρίβεια των αποτελεσμάτων εξαρτάται από κάποιες μεταβλητές και μπορεί να αυξηθεί με την κατάλληλη προσαρμογή τους όπως αναλύθηκε κατά τη διάρκεια της εργασίας.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

C. Claudin, J. Rech, 2009. Development of a new rapid characterization method of hob's wear resistance in gear manufacturing—Application to the evaluation of various cutting edge preparations in high speed dry gear hobbing. Journal of Materials Processing Technology 209 (2009) 5152–5160

J. Rech, M.A. Djouadi, J. Picot, 2001. Wear resistance of coatings in high speed gear hobbing. Journal of Wear 250 (2001) 45–53

K.-D. Bouzakis, S. Kombogiannis, O. Friderikos, J. Anastopoulos. Cutting performance increasing in gear hobbing by means of HSS hobs, coated with effective PVD films. Laboratory for Machine Tools and Manufacturing Engineering, Mechanical Engineering Department, Aristoteles University of Thessaloniki

Σ. Γρηγοριάδη, 1999. Μελέτη του μηχανισμού της φθοράς των κοπτικών εργαλείων. Τμήμα μηχανολογίας – Εργαστήριο Εργαλειομηχανών, Τ.Ε.Ι. Κρήτης.