Τεχνολογικό Εκπαιδευτικό Ίδρυμα Πειραιά
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανολογίας – Κατασκευαστικός

ΓΡΑΜΜΑΤΕΙΑ ΜΗΧΑΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΒΕΛΤΙΩΣΗ ΛΟΓΙΣΜΙΚΟΥ ΕΠΑΝΑΛΗΠΤΙΚΩΝ ΔΙΑΔΙΚΑΣΙΩΝ ΤΟΜΗΣ ΔΟΚΩΝ, ΓΙΑ ΧΡΗΣΗ ΣΕ ΜΕΘΟΔΟ ΣΥΝΟΡΙΑΚΩΝ ΣΤΟΙΧΕΙΩΝ (Β.Ε.Μ)

Φοιτητης : Νικόλας Καρβούνης
Α.Μ. : 34505

Εισηγητής : Δρ. –Μηχ. Κωσταντίνος Στεργίου, Αναπληρωτής Καθηγητής Τμ. Μηχανολογίας

Σύμβουλος : Βασίλειος Σαγιάς MSc, Εργαστηριακός Συνεργάτης Τμ. Μηχανολογίας

Αιγάλεω Ιανουάριος 2012

Βελτίωση Λογισμικού Επαναληπτικών Διαδικασιών Τομής Δοκών, Για Χρήση Σε Μέθοδο Συνοριακών Στοιχείων (Β.Ε.Μ)

Του Νικόλα Καρβούνη του Μάρκου

Μια εργασία που υποβάλλεται για την μερική εκπλήρωση των απαιτήσεων για το πτυχίο του

Μηχανολόγου Μηχανικού ΤΕ

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Πειραιά

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανολογίας

Κατασκευαστικός Τομέας

Αιγάλεω, Ιανουάριος 2012

Επιτροπή Εξέτασης

1.

.....

....

(Ονοματεπώνυμο εξεταστή)

(Υπογραφή Εξεταστή)

2.

.....

....

(Ονοματεπώνυμο εξεταστή)

(Υπογραφή Εξεταστή)

3.

.....

....

(Ονοματεπώνυμο εξεταστή)

(Υπογραφή Εξεταστή)

Ημερομηνία Εξέτασης : 20 Ιανουαρίου 2012

Περιεχόμενα

1)Εισαγωγή

Στα πλαίσια της πτυχιακής εργασίας θα πραγματοποιηθεί η ανάπτυξη λογισμικού (κώδικα) κάνοντας χρήση γλώσσας προγραμματισμού VBA (Visual Basic for Applications). Όλος ο προγραμματισμός θα αναπτυχθεί μέσα στα πλαίσια του προγραμματιστικού περιβάλλοντος (API) του τρισδιάστατου CAD συστήματος Autodesk Inventor.

Στόχος είναι η επαναληπτική διαδικασία τομής σε διαφορετικές θέσεις τρισδιάστατων δοκών οποιασδήποτε μορφής, προερχόμενες απο extrude και loft features, και χρήση της δισδιάστατης τομής για την παραγωγή B.E.M (Boundary Element Method) υπολογιστικών μοντέλων, καθώς και η τρισδιάστατη γραφική απεικόνιση των αποτελεσμάτων πάνω στην δοκό.

Τέλος, θα γίνει συζήτηση σχετικά με τις δυσκολίες-προβλήματα υλοποίησης του προγράμματος και τις τυχόν ελλείψεις του περιβάλλοντος προγραμματισμού, όσον αφορά τα αντικείμενα (Objects).

2)Ιστορική αναδρομή

Η επιστήμη της μηχανολογίας είναι άρρηκτα συνδεδεμένη με το σχέδιο. Το σχέδιο δεν είναι απλά αποτύπωση εικόνων, αλλά μια γραφική γλώσσα επικοινωνίας η οποία μεταφέρει ιδέες και πληροφορίες από τον εκάστοτε μηχανολόγο-σχεδιαστή στον εκάστοτε παραλήπτη.

Με την εξέλιξη της τεχνολογίας και την ανάπτυξη των ηλεκτρονικών υπολογιστών καθώς και τις απαιτήσεις της βιομηχανίας για αύξηση της παραγωγικής διαδικασίας, η χρήση των υπολογιστών για την μείωση του χρόνου και την βελτίωση της ποιότητας του σχεδιασμού με την χρήση των υπολογιστών ήταν μονόδρομος.

Τα πρώτα βήματα πραγματοποιήθηκαν το 1960, από βιομηχανίες που δραστηριοποιούνταν στον χώρο του αυτοκινήτου και της αεροπλοΐας και είχαν ως θέμα την κατασκευή τρισδιάστατων επιφανειών καθώς και προγραμματισμό NC (numerical control). Οι πρωτοστάτες στην αναπαράσταση πολυωνυμικών καμπυλών και επιφανειών ήταν ο Pierre Bézier (Renault), ο Paul de Casteljau (Citroen), ο Steven Anson Coons (MIT, Ford),ο James Ferguson (Boeing), ο Carl de Boor (GM), ο Birkhoff (GM) και ο Garibedian (GM), αλλά και ο W. Gordon (GM) και ο R. Riesenfeld το 1970.

Έτσι στις αρχές του 1970 εισήχθη ο όρος CAD (Computer Aided Design) όπου σύμφωνα με το όνομα του εννοούμε τον σχεδιασμό κατασκευών με την βοήθεια υπολογιστή.

Με την στενή του έννοια ο όρος CAD περιορίζεται στην γεωμετρική μοντελοποίηση (geometric modeling ή συχνά στην αγγλοσαξονική βιβλιογραφία και computer geometry) η οποία περιέχει όλες τις μεθόδους που περιγράφουν με απόλυτη μαθηματική ακρίβεια τη μορφή και το σχήμα αντικειμένων και σωμάτων, αλλά και την προσομοίωση (simulation) δυναμικών διεργασιών. Περιλαμβάνει επίσης όλες τις υπολογιστικές λειτουργίες με τις οποίες αναπαρίσταται, αποθηκεύεται και αναλύετε ένα μοντέλο, ως μοντέλο νοείται η αναπαράσταση στον υπολογιστή ενός πραγματικού αντικειμένου ή μιας διεργασίας.

Ενώ με την ευρεία του έννοια και με την εξέλιξη των συστημάτων CAD περιλαμβάνει και το τμήμα της μελέτης, του υπολογισμού, καθώς και την εκπόνηση σχεδίων.

Από τα 2D (δισδιάστατα) CAD τα οποία μπορούσαν να απεικονίσουν όψεις ενός μοντέλου και όχι το μοντέλο αυτό καθ'αυτό, όπως δηλαδή το σχέδιο στο χαρτί, περάσαμε στην εξέλιξη τους τα 2-1/2D. Τα 2-1/2D συστήματα βασίζονταν στα 2D αλλά μπορούσαν να περιγράψουν ένα αντικείμενο στον χώρο, των οποίων η χαρακτηριστική μορφή εκτείνεται μόνο σε ένα επίπεδο. Η τρίτη διάσταση τους περιέχεται μέσω απλής έκτασης της επιφάνειας-οδηγού γραμμικά είτε περιστροφικά. Τα συστήματα αυτά δεν μπορούν να θεωρηθούν καθαρά σχεδιαστικά διότι ενώ υπάρχει η δυνατότητα εξαγωγής οποιονδήποτε όψεων και τομών από την κύρια όψη, μια αλλαγή σε αυτήν δεν ακολουθείται από αυτόματη αλλαγή στις υπάρχουσες όψεις και τομές.

Η σημερινή μορφή των σχεδιαστικών προγραμμάτων είναι τα 3D CAD τα οποία χρησιμοποιούν τρισδιάστατη μοντελοποίηση για να αναπαραστήσουν το πραγματικό αντικείμενο. Με τα 3D CAD ο χρήστης έχει άμεση πρόσβαση σε πληθώρα πληροφοριών ως προς το αντικείμενο, όπως π.χ ο όγκος του, το κέντρο βάρους του, τη μάζα του, αλλά και σε διάφορες όψεις και τομές του. Το σημαντικότερο όλων όμως είναι ότι ο χρήστης έχει την δυνατότητα να επέμβει στην γεωμετρία του αντικειμένου οποιαδήποτε στιγμή με αυτό να συνεπάγεται και αυτόματη αλλαγή όλων τον παραπάνω παραμέτρων.

Τα 3D CAD χωρίζονται σε τρεις κατηγορίες με κριτίριο τον τρόπο μοντελοποίησης του πραγματικού αντικείμενου:

1) Τα μοντέλα σύρματος (wireframe modelers)

2) Τα μοντέλα επιφανειών (surface modelers)

3) Τα στερεά μοντέλα (solid modelers)

Από τις τρεις παραπάνω έχει επικρατήσει η τρίτη δηλαδή τα συστήματα στερεής μοντελοποίησης οπού με την σειρά τους διακρίνονται μεταξύ τους αναλόγως με τον τρόπο αποθήκευσης των πληροφοριών του υπό κατασκευή αντικειμένου.

1. CSG (Constructive Solid Geometry)

Το μοντέλο δημιουργείται στην βάση δεδομένων με την βοήθεια λειτουργικών συνόλων (boolean operation) απο το συνδυασμό στοιχειωδών στερεών σωμάτων, όπως π.χ ορθογώνιο, κύλινδρος , σφαίρα κλπ. Η δομή των συστημάτων αυτών αποθηκεύεται συνήθως με την μορφή δυαδικού δένδρου.

2. B-Rep (Boundary Representation)

Τα μοντέλα αυτά αποτελούνται από δύο τμήματα, την γεωμετρία τους και από την τοπολογία τους. Η γεωμετρία περιλαμβάνει τις επιφάνειες (surfaces), τις καμπύλες (curves) και τα σημεία (points). Η τοπολογία περιλαμβάνει τις πλευρές (faces), τις ακμές (edges) και τις γωνίες (vertices). Ο συνδυασμός των δύο είναι που διαμορφώνει το τελικό μοντέλο. Μια πλευρά (face) είναι ένα οριοθετημένο τμήμα μίας επιφάνειας (surface) καθώς οι επιφάνειες δεν έχουν όρια. Η ακμή (edge) με την σειρά της σχηματίζεται από την διασταύρωση δύο πλευρών και στην ουσία είναι ένα οριοθετημένο τμήματα μιας καμπύλης (curve). Τέλος μια γωνία (vertex) η οποία προέρχεται από την διασταύρωση δύο ή περισσοτέρων ακμών (edges) είναι ένα σημείο (point) στον χώρο. Η μέθοδος αυτή είναι και η πλέον δημοφιλέστερη και αυτήν χρησιμοποιεί το πρόγραμμα που γίνεται η εργασία.

3) Hybrid systems

Είναι ο συνδυασμός στοιχείων CSG και B-Rep στο ίδιο σύστημα.

Όλα τα σύγχρονα 3D CAD προγράμματα διαθέτουν API (Application Programming Interface) ή στα ελληνικά Διεπαφή Προγραμματισμού Εφαρμογών. Ορίζουμε Διεπαφή Προγραμματισμού Εφαρμογών, τη διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή / και ανταλλαγή δεδομένων.

Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το σύνολο των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή, ένα «συμβόλαιο κλήσης» μεταξύ καλούντος και καλούμενου, διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους.

Π.χ. το ταχυδρομείο παρέχει την υπηρεσία της αποστολής γραμμάτων. Οι κανόνες οι οποίοι πρέπει να ακολουθηθούν για την υποβολή ενός αιτήματος αποστολής (φόρμα διεύθυνσης παραλαβής, γραμματόσημο κτλ) είναι καλώς ορισμένοι, αλλά το πώς θα υλοποιηθεί στην πράξη αυτό το αίτημα αφορά έναν ολόκληρο μηχανισμό υπαλλήλων εν πολλοίς αθέατο στον χρήστη της υπηρεσίας. Στο εν λόγω παράδειγμα διεπαφή είναι οι υπηρεσίες που παρέχονται στους πελάτες οι οποίες συνήθως είναι γραμμένες σε ένα φυλλάδιο, τη διεπαφή του ταχυδρομείου προς τους χρήστες του.

Έτσι λοιπόν το σχεδιαστικό πρόγραμμα έχει τη δική του διεπαφή (κλήσεις συστήματος), η φόρμα της οποίας διατίθεται από την εκάστοτε κατασκευάστρια εταιρεία, η οποία περιγράφει τους τρόπους αξιοποίησης από προγράμματα χρήστη του συνόλου των υπηρεσιών που παρέχει το πρόγραμμα. Το τμήμα του προγράμματος το οποίο υλοποιεί τις υπηρεσίες που περιγράφονται στη διεπαφή, συνήθως στον πυρήνα του, λέμε ότι είναι η υλοποίηση της διεπαφής.

Τα περισσότερα API των σχεδιαστικών προγραμμάτων χρησιμοποιούν ως γλώσσα προγραμματισμού είτε την C/C++ είτε την VBA (Visual Basic for Applications). Στην δική μας περίπτωση η γλώσσα που θα χρησιμοποιήσουμε είναι η VBA.

Η Visual Basic for Application (VBA) είναι μία εξειδικευμένη μορφή της Microsoft Visual Basic®. Δημιουργήθηκε με βάση την γνωστή Basic και χρησιμοποιεί πληθώρα ίδιων ή παρόμοιων χαρακτηριστικών και παρόμοιο σχεδιαστικό περιβάλλον.

Πρόκειται βασικά για μια «Light» έκδοση όσον αφορά τα χαρακτηριστικά λειτουργίας, τον τρόπο διαχείρισης, τις εντολές και τις συναρτήσεις. Σκοπός της δεν είναι η αντικατάσταση της Microsoft VB αλλά η δυνατότητα χρήσης της με σκοπό τον προγραμματισμό των λειτουργιών ενός λογισμικού ξενιστή. Η VBA δεν υπάρχει ως ξεχωριστό λογισμικό. Είναι όμως ενσωματωμένη σε άλλα προγράμματα όπως MsOffice, StarOffice, προγράμματα Cad, MicroStation κτλ. Η χρήση της συνίσταται στην αυτοματοποίηση κάποιων διαδικασιών (ή στην προσθήκη κάποιων νέων) τις οποίες, είτε δεν προέβλεψε ο κατασκευαστής, είτε πρόκειται για λειτουργίες που χρησιμεύουν σε μας, αλλά όχι και σε άλλους χρήστες. Είναι λοιπόν το εργαλείο εκείνο το οποίο θα μας βοηθήσει να προσαρμόσουμε το υπάρχον λογισμικό στις ανάγκες μας.

Ο προγραμματισμός της VBA διαφέρει από τον αντίστοιχο της Ms VB σε αρκετά σημεία. Για παράδειγμα τα εργαλεία που χρησιμοποιούνται στις περισσότερες περιπτώσεις έχουν παραπλήσια ονόματα αλλά όχι ίδια. Επίσης κάθε λογισμικό ξενιστής, πχ το Excel εισάγει τις δικές του εντολές και συναρτήσεις στη διάθεση του χρήστη. Με αυτό τον τρόπο δίνεται η δυνατότητα στο χρήστη να υπολογίσει μια συνάρτηση «τραβώντας» δεδομένα από ορισμένα κελιά του Excel.

3)API Objects

Ακολουθεί η ανάλυση των πιο συμαντικών αντικειμένων (objects) του API του Inventor, τα οποία χρησιμοποιήθηκαν στο πρόγραμμα. Αυτή η ανάλυση έχει ως σκοπό να δείξει στον αναγνώστη τις βασικές αρχές προγραμματισμού της συγκεκριμένης πλατφόρμας, ώστε να υπάρξει μια ομαλή μετάβαση στο κεφάλαιο που αφορά την επεξήγηση του προγράμματος της πτυχιακής.

a) Geometry

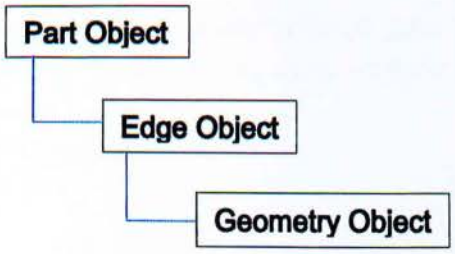
Η κύρια χρήση του Inventor είναι είτε η δημιουργία γεωμετρίας (geometry) είτε η αξιοποίηση ήδη υπάρχουσας γεωμετρίας ως δεδομένο για κάποια άλλη διεργασία, όπως π.χ δημιουργία drawings, ανάλυση ή χρήση για CAM (Computer Aided manufacturing). Γία την δημιουργία ενός μοντέλου, ορίζεται αρχικά η 2D (δισδιάστατη) γεωμετρία μέσα απο ένα sketch και ύστερα χρησιμοποιείται ως input (είσοδος) για τα διάφορα features μέσω των οποίων δημιουργείται το τρισδιάστατο μοντέλο (3D solid model). Εφόσον το Inventor είναι παραμετρικό σύστημα, υπάρχει η δυνατότητα τροποποίησης του μοντέλου αλλάζοντας τις παραμέτρους ή τα inputs.

b)Entities and Geometry

Το API χρησιμοποιεί μια σταθερή ορολογία για την διαφοροποίηση κάποιων εννοιών. Οι όροι entity και geometry συχνά μπορεί να συγχέονται από τον χρήστη, άλλα στο API έχουν διακριτή σημασία.

Ο όρος entity χρησιμοποιείται για οποιοδήποτε object (αντικείμενο) που μπορεί να επιλεχθεί μέσα από το user interface (το περιβάλλον του χρήστη). Ένα τέτοιο object μπορεί να είναι μια διάσταση, ένα Face, ένας κύκλος (Circle) από ένα Sketch.

Ο όρος geometry παραπέμπει στον πραγματικό γεωμετρικό ορισμό ενός entity. Για παράδειγμα, μπορεί να επιλέξει κάποιος ένα Edge (ακμή) από ένα μοντέλο, στην περίπτωση αυτή το entity είναι ένα Edge object και και από το entity αυτό μπορεί να ανατρέξει στο geometry object του, το οποίο μπορεί να είναι διαφόρων τύπων (π.χ Line, Arc, Spline, κτλπ). Στο παρακάτω σχήμα φαίνεται και η ιεραρχία των objects.



Το Inventor κάνει πολλές από τις διεργασίες του αθέατα από τον χρήστη για να μετατρέψει τελικά τις εντολές του χρήστη στην αντίστοιχη γεωμετρία. Με τον όρο transient geometry ονομάζεται μια ομάδα από objects η οποία ορίζει συγκεκριμένα γεωμετρικά σχήματα. Το μέρος «transient» του ονόματος υποδεικνύει ότι αυτά τα objects είναι προσωρινά και δεν αποθηκεύονται από το Inventor. Παράδείγματος χάριν, υποθετικά έχουμε ως μοντέλο έναν στερεό κύβο και επιλέγουμε μια ακμή του, έχουμε επιλέξει δηλαδή ένα entity που στην προκειμένη περίπτωση είναι ένα Edge object. Από το edge object μπορούμε να δούμε την γεωμετρία του η οποία είναι ένα LineSegment object. Το LineSegment object είναι ένα transient geometry object, έχει δηλαδή ιδιότητες η οποίες επιτρέπουν την πρόσβαση στις γεωμετρικές πληροφορίες που είναι σχετικές με αυτό, όπως π.χ το StartPoint και το EndPoint (το αρχικό και το τελικό σημείο της γραμμής στον χώρο).

Μια ενδιαφέρουσα πτυχή των transient geometry objects είναι οτι παρέχονται ως αποσπώμενα objects αυτό σημαίνει ότι η γεωμετρία που λαμβάνεται από ένα entity αντιπροσωπεύει το σχήμα του entity αυτού εκείνη την συγκεκριμένη χρονική στιγμή που έγινε η λήψη και πλέον δεν συνδέεται με το entity. Για παράδειγμα, το LineSegment που πήραμε από το Edge object στο προηγούμενο παράδειγμα, περιέχει το StartPoint και το Endpoint του Edge αλλά αν μετά αλλάξουμε το Edge (π.χ αλλάξουμε το μήκος του extrusion) το geometry object δεν θα αντικατοπτρίζει αυτήν την αλλαγή. Στην ουσία αυτό που κάνει είναι να προβάλει ένα στιγμιότυπο της γεωμετρίας του entity τη χρονική στιγμή που λάβαμε το geometry object. Υπάρχει επίσης η δυνατότητα επεξεργασίας του geometry object π.χ η αλλαγή της τοποθεσίας των StartPoint και EndPoint, αλλά αυτό με την σειρά του δεν θα επηρεάσει το entity.

Τα transient geometry objects μπορούν επίσης να δημιουργηθούν κατευθείαν χωρίς να υπάρχει η ανάγκη λήψης τους από κάποιο entity. Αυτή η λειτουργία υποστηρίζεται από το transient geometry object στο οποίο έχουμε πρόσβαση χρησιμοποιώντας το transient geometry property του Application object. Αυτό το object είναι μια λειτουργία του API και δεν αντιπροσωπεύει τίποτα απ' όσα βλέπουμε στο user interface του Inventor, το μόνο που κάνει είναι να προσφέρει ένα σύνολο μεθόδων για την δημιουργία διαφόρων τύπων γεωμετρίας. Για παράδειγμα, υπάρχει το CreateLineSegment method αυτό λαμβάνει σαν δεδομένα το StartPoint και το Endpoint και επιστρέφει ένα LineSegment object, τίποτα δεν αλλάζει μέσα στο Inventor όταν γίνει αυτό, δεν φαίνεται στην οθόνη κάποια γραμμή στην οθόνη, ακόμα και αν αποθηκεύσουμε το αρχείο η γραμμή αυτή δεν θα αποθηκευτεί, αυτό που δημιουργήσαμε είναι στην ουσία ο γεωμετρικός ορισμός μίας γραμμής (Line) και όχι ένα line entity.

Άλλη μια ομάδα είναι τα transient objects και τα πιο γνωστά και περισσότερο χρησιμοποιημένα objects αυτής της ομάδας είναι αυτά που αφορούν την δημιουργία Collection objects. Τα Collection objects είναι μια συλλογή, όπως λέει και το όνομα τους, από διαφόρων ειδών objects.

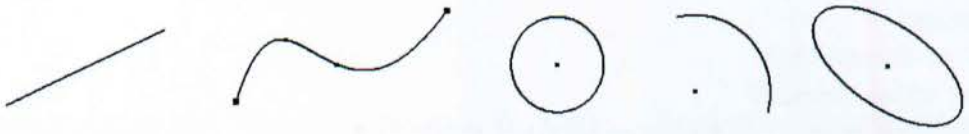
c) Points

Υπάρχουν αρκετά entities στο Inventor τα οποία αντιπροσωπεύουν ένα σημείο (point), όπως π.χ ένα vertex, ένα WorkPoint, ένα 3D SketchPoint και ένα 2D SketchPoint. Το Vertex, το WorkPoint και το 3D SketchPoint πάντα ορίζουν ένα σημείο τριών διαστάσεων(3D) στον χώρο, ενώ ένα 2D SketchPoint ορίζει πάντα ένα δισδιάστατο (2D) σημείο πάνω στο sketch. Απ' όλα αυτά τα entities μπορεί κανείς να πάρει ένα transient geometry point, τα τρισδιάστατα objects επιστρέφουν Point objects ενώ τα δισδιάστατα Point2d objects. Τα point geometry objects προσφέρουν πρόσβαση στις x,y,z ή x,y συντεταγμένες ενός σημείου.

Αντί για ένα Point object το API θα μπορούσε να έχει επιλέξει να χρησιμοποιήσει ένα Array (πίνακα) με τρεις μεταβλητές για να ορίσει το σημείο. Ένα Point object όμως έχει το προνόμιο ότι περιλαμβάνει όλες τις πληροφορίες του μέσα σε ένα μόνο object αλλά το σημαντικότερο είναι ότι υποστηρίζει επιπρόσθετες χρήσιμες μεθόδους (methods). Για παράδειγμα, με την μέθοδο DistanceTo επιστρέφει την απόσταση μεταξύ αυτού του σημείου και ενός δεύτερου επιλεγμένου σημείου.

d) Curves

Curve (καμπύλη) είναι οποιαδήποτε wireframe γεωμετρία, όπως Spline, Circle, Arc, Line. Υπάρχουν πολλά τρισδιάστατα και δισδιάστατα curve entities που υποστηρίζει το Inventor (όπως τα Edge, SketchLine, SketchArc, WorkAxis, SketchSpline κτλπ), όλα αυτά τα entities επιστρέφουν ένα geometry object το οποίο δείχνει το σχήμα που έχει το εκάστοτε entity. Παρακάτω υπάρχουν κάποια παραδείγματα για τα διάφορα geometry objects που αντιστοιχούν σε διάφορα entities.



Το παρακάτω δείγμα κώδικα κρατάει στην μνήμη το εκάστοτε επιλεγμένο Edge. Το δείγμα αυτό θα χρησιμοποιηθεί για τα παραδείγματα που θα ακολουθήσουν.

```
Public Sub CurveGeometry( )

' Get the active document.
Dim oPartDoc As PartDocument
Set oPartDoc = ThisApplication.ActiveDocument

' Get the selected edge.
On Error Resume Next
```

```
Dim oEdge As Edge
Set oEdge = oPartDoc.SelectSet.Item(1)

If Err Then
MsgBox "An edge must be selected."
Exit Sub
End If
On Error GoTo 0

End Sub
```

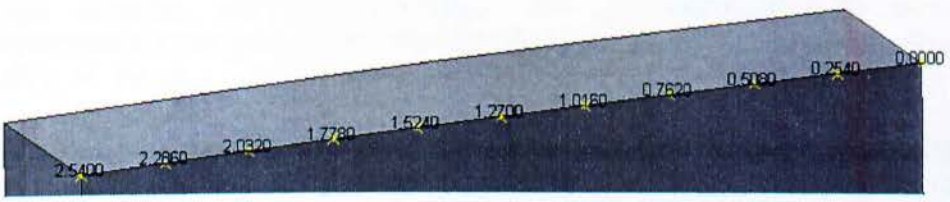
Σε πολλά entities ο τύπος του geometry object που τους αντιστοιχεί είναι γνωστός και μονοσήμαντος. Για παράδειγμα, ένα SketchLine θα επιστρέφει πάντα ένα LineSegment2d geometry object, ένα SketchLine3D object θα επιστρέφει πάντα ένα LineSegment και ένα SketchCircle3D θα επιστρέφει πάντα ένα Circle object. Κάθε Edge object όμως είναι μοναδικό, επειδή μπορεί να έχει διαφορετικό σχήμα. Αυτό σημαίνει ότι ένα Edge object μπορεί να επιστρέψει διαφορετικούς τύπους geometry object που εξαρτώνται από το σχήμα που θα έχει. Ο παρακάτω κώδικας, ο οποίος είναι προέκταση του προηγουμένου κώδικα, χρησιμοποιεί το CurveType property του Edge για να βρει τον τύπο του geometry που θα επιστρέψει και τέλος εκτυπώνει τις πληροφορίες της γεωμετρίας του.

```
' Check the geometry type and print out some geometry specific information.
Select Case oEdge.GeometryType

Case kLineSegmentCurve
Dim oLineSegment As LineSegment
Set oLineSegment = oEdge.Geometry
Debug.Print "Start point: " & PointString(oLineSegment.StartPoint)
Debug.Print "End point: " & PointString(oLineSegment.EndPoint)

Case kCircleCurve
Dim oCircle As Inventor.Circle
Set oCircle = oEdge.Geometry
Debug.Print "Center point: " & PointString(oCircle.Center)
Debug.Print "Radius: " & Format(oCircle.Radius, "0.000000")

Case kCircularArcCurve
Dim dPi As Double
dPi = Atn(1) * 4
Dim oArc As Inventor.Arc3d
Set oArc = oEdge.Geometry
Debug.Print "Center point: " & PointString(oArc.Center)
Debug.Print "Radius: " & Format(oArc.Radius, "0.000000")
Debug.Print "Sweep: " & Format(oArc.SweepAngle * (180 / dPi), "0.000000")

Case Else
   Debug.Print "Unsupported geometry type selected."
End Select
```

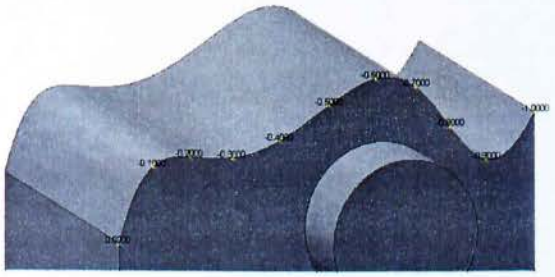
```
End Sub
' Given a point or vector return a string containing the X,Y,Z coordinates.
Private Function PointString(PointOrVector As Object) As String

PointString = Format(PointOrVector.X, "0.000000") & "," & _
Format(PointOrVector.Y, "0.000000") & "," & _
Format(PointOrVector.Z, "0.000000")

End Function
```

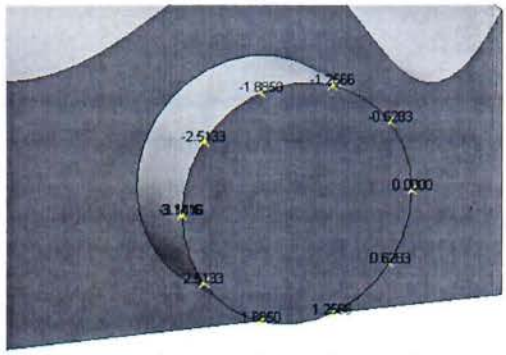
Υπάρχουν και άλλες διεργασίες που μπορούν να γίνουν στα Curves χρησιμοποιώντας το CurveEvaluator object. Κάθε transient geometry Curve object έχει CurveEvaluator, όπως και κάθε Edge object επίσης. Οι τύποι της ανάλυσης που πραγματοποιεί το CurveEvaluator είναι γενικοί και μπορούν να εφαρμοστούν σε κάθε λογής σχήματος Curve.

Για την χρησιμοποίηση του CurveEvaluator είναι σημαντική η κατανόηση της έννοιας του parametric space (παραμετρικού χώρου) ή στην συγκεκριμένη περίπτωση του curve space. Η έννοια αυτή γίνεται αντιληπτή με την βοήθεια της παρακάτω εικόνας. Εάν εκλάβουμε το Curve ως μια αριθμημένη γραμμή μπορούμε να καθορίσουμε οποιοδήποτε σημείο πάνω στο Curve από μια και μόνο τιμή. Η εικόνα το δείχνει με ένα Edge, το Curve space κατά μήκος του Curve αρχίζει με την τιμή 0 και φτάνει στο 2,54. Έτσι μια σειρά από τοποθεσίες ορίζεται κατά μήκος του Edge.



Αυτή η έννοια ισχύει για όλα τα σχήματα ενός Edge . Παρακάτω παρουσιάζεται η ίδια διαδικασία για ένα Spline και ένα Circle. Όπως φαίνεται από τα σχήματα το εύρος των παραμετρικών τιμών διαφέρει σε κάθε Curve. Στο πρώτο σχήμα ήταν από 0 έως 2,54 στο δεύτερο από -1 έως 0 και στο τελευταίο από –π έως π.





Η κατανόηση του παραμετρικού χώρου είναι ζωτικής σημασίας για την χρήση των λειτουργιών του CurveEvaluator object. Τα περισσότερα από τα methods του είτε λαμβάνουν παραμετρικές τιμές ως είσοδο είτε τις επιστρέφουν ως έξοδο. Η λίστα που ακολουθεί περιέχει τα πιο κοινά methods του CurveEvaluator object :

1. **GetParamExtents:** Επιστρέφει την μέγιστη και την ελάχιστη παραμετρική τιμή του Curve.

2. **GetPointAtParam:** Επιστρέφει ένα σημείο στον χώρο που αντιστοιχεί στην δοσμένη παραμετρική τιμή.

3. **GetParamAtPoint:** Εισάγοντας ένα σημείο στον χώρο που βρίσκεται πάνω στο Curve επιστρέφει την αντίστοιχη παραμετρική του τιμή.

4. **GetLengthAtParam:** Επιστρέφει το μήκος ανάμεσα σε δύο παραμετρικές τιμές του Curve. Χρησιμοποιώντας την ελάχιστη και την μέγιστη τιμή από το GetParamExtents έχουμε ως έξοδο το συνολικό μήκος του Curve.

5. **GetParamAtLength:** Επιστρέφει την παραμετρική τιμή απο την δοσμένη απόσταση κατά μήκος του Curve από μια άλλη παραμετρική τιμή.

6. **GetTangent:** Επιστρέφει το εφαπτόμενο διάνυσμα στην καθορισμένη παραμετρική τιμή του Curve.

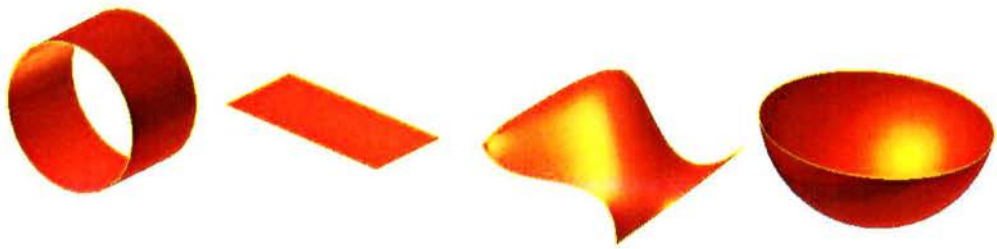
7. **GetCurvature:** Επιστρέφει την καμπυλότητα (curvature) στην δοσμένη παραμετρική τιμή.

Ο παρακάτω κώδικας χρησιμοποιήθηκε για τα προηγούμενα σχήματα και εμπεριέχει και τον προηγούμενο κώδικα για την επιλογή του Edge. Γίνεται χρήση των μεθόδων GetParamExtents και GetPointAtParam.

```
' Get the evaluator from the curve.
Dim oCurveEval As CurveEvaluator
Set oCurveEval = oEdge.Evaluator

' Get the parametric range of the curve.
 Dim dMinParam As Double
 Dim dMaxParam As Double
 Call oCurveEval.GetParamExtents(dMinParam, dMaxParam)

' Set a reference to the TransientGeometry object.
Dim oTG As TransientGeometry
Set oTG = ThisApplication.TransientGeometry

' Iterate 10 steps over the curve length and print the
' parameter values and corresponding model points.
Dim i As Integer
For i = 0 To 10

' Calculate the current parameter to evaluate.
Dim currentParam As Double
currentParam = dMinParam + ((dMaxParam - dMinParam) / 10) * i

' Assign the value to an array since the GetPointAtParam ' takes an array as input.
Dim adParam(0) As Double
adParam(0) = currentParam

' Get the coordinates of the parameter point in model space.
Dim adPoints(2) As Double
Call oCurveEval.GetPointAtParam(adParam, adPoints)

' Print information about this point.
Debug.Print "Parameter: " & Format(currentParam, "0.0000") & _
" Coordinate: " & Format(adPoints(0), "0.000000") & "," & _
 Format(adPoints(1), "0.000000") & "," & _
Format(adPoints(2), "0.000000")

Next

End Sub
```

e) Surfaces

Οι επιφάνειες (surfaces) στο Inventor αντιπροσωπεύονται από τα solid models και τα surface models. Τα WorkPlanes επίσης μπορούν να νοηθούν ως επιφάνειες σε πολλές περιπτώσεις. Τα surface objects μοιάζουν πολύ με τα curve objects στις βασικές τους ιδιότητες, σε κάποιους τομείς μάλιστα είναι και πιο απλά επειδή υπάρχουν μόνο στον τρισδιάστατο χώρο και τα μόνα entities που αντιπροσωπεύουν εκεί μια επιφάνεια είναι τα face objects και τα WorkPlane objects.



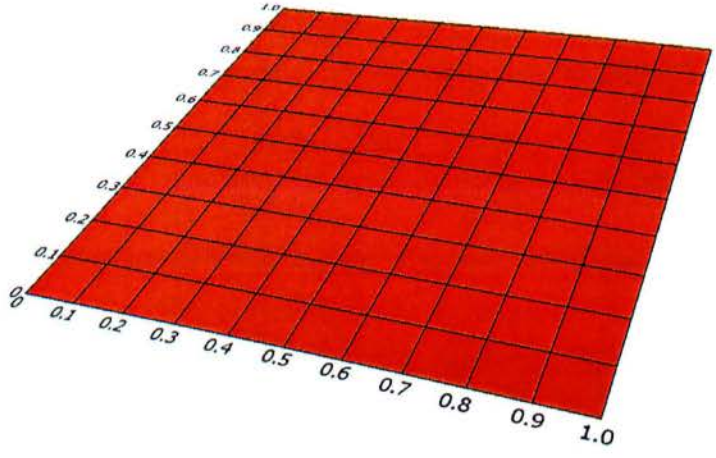
Όπως και στα curves το entity επιστρέφει ένα geometry object το οποίο δείχνει το σχήμα του entity, έτσι και τα WorkPlanes επιστρέφουν πάντα plane (επίπεδο) object ως geometry object. Ενώ τα face objects μπορούν να επιστρέψουν διαφορετικούς τύπους geometry object αναλόγως το σχήμα του face. Έτσι το SurfaceType property του face object μας καθορίζει τι τύπος geometry object είναι το εκάστοτε face. Το παρακάτω πρόγραμμα είναι μια εναλλαγή του προηγούμενου κώδικα για τα curves, ώστε να δουλεύει με faces.

```
Public Sub SurfaceGeometry()
  ' Get the active document.
Dim oPartDoc As PartDocument
Set oPartDoc = ThisApplication.ActiveDocument

' Get the selected face.
On Error Resume Next
Dim oFace As Face
Set oFace = oPartDoc.SelectSet.Item(1)
If Err Then
    MsgBox "A face must be selected."
    Exit Sub
End If
On Error GoTo 0



' Check the geometry type and print out some geometry specific information.
Select Case oFace.SurfaceType
  Case kPlaneSurface
    Dim oPlane As Plane
    Set oPlane = oFace.Geometry
```

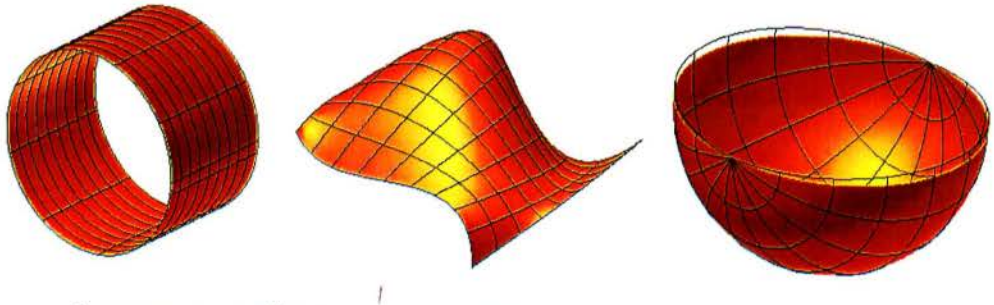
```
    Debug.Print "Planar face"
    Debug.Print " Root point: " & PointString(oPlane.RootPoint)
    Debug.Print " Normal vector: " & PointString(oPlane.Normal)
  Case kCylinderSurface
    Dim oCylinder As Cylinder
    Set oCylinder = oFace.Geometry
    Debug.Print "Cylindrical face"
    Debug.Print " Base point: " & PointString(oCylinder.BasePoint)
    Debug.Print " Axis vector: " & PointString(oCylinder.AxisVector)
    Debug.Print " Radius: " & Format(oCylinder.Radius, "0.000000")
  Case kSphereSurface
    Dim oSphere As Sphere
    Set oSphere = oFace.Geometry
    Debug.Print "Spherical face"
    Debug.Print " Center point: " & PointString(oSphere.CenterPoint)
    Debug.Print " Radius: " & Format(oSphere.Radius, "0.000000")
Case Else
    Debug.Print "Unsupported geometry selected: " & TypeName(oFace.Geometry)
End Select

End Sub
```

Ένας σημαντικός παράγοντας που πρέπει να κατανοηθεί είναι ότι τα περισσότερα από τα transient geometry surfaces δεν έχουν όρια. Για παράδειγμα, ένα plane μπορεί να ορισθεί από μόνο ένα σημείο και ένα vector (διάνυσμα), με λίγα λόγια δεν έχει μέγεθος. Επίσης, ένας κύλινδρος (cylinder) ορίζεται από ένα origin point, έναν άξονα (axis) και μία ακτίνα (radius), δηλαδή δεν έχει μήκος. Η παρακάτω εικόνα δείχνει ένα κύλινδρο όπως θα μπορούσε να υπάρχει μέσα στο Inventor, προφανώς αυτός ο κύλινδρος δεν έχει άπειρο μήκος, επιπρόσθετα έχει και τρύπες και διαμορφωμένα edges. Αυτό γίνεται διότι δεν είναι ο κύλινδρος ο οποίος ορίζει τα edges, αλλά είναι τα edges του face που ορίζουν τα όρια του κυλίνδρου και αυτό ισχύει για όλα τα surface objects.

Ένα ακόμα κοινό χαρακτηριστικό των surfaces με τα curves είναι ότι και τα δύο χρησιμοποιούν τον παραμετρικό χώρο, όμως αντίθετα με τα curves όπου ο παραμετρικός τους χώρος έχει μια διάσταση, ο παραμετρικός χώρος των surfaces είναι δισδιάστατος. Έτσι για να ορίσεις οποιαδήποτε τοποθεσία πάνω στο surface χρειάζονται δύο παράμετρικές τιμές. Ο παραμετρικός χώρος αυτός μπορεί να νοηθεί καλύτερα ώς ένα τετράγωνο επίπεδο . Η παρακάτω εικόνα απεικονίζει τον παραμετρικό χώρο με κάθετες και οριζόντιες γραμμές όπως ακριβώς ένα δισδιάστατο σύστημα συντεταγμένων. Ορίζει κανείς ένα σημείο πάνω του δίνοντας τις x και y συντεταγμένες, μόνο που στον παραμετρικό χώρο αυτές αναφέρονται ως u και v αντίστοιχα.



Παρακάτω απεικονίζονται κάποιες επιφάνειες με το παραμετρικό πλέγμα σχεδιασμένο πάνω τους, αν και οι επιφάνειες αυτές δεν είναι επίπεδες υπάρχει ακόμα η δυνατότητα να προσδιοριστεί οποιαδήποτε τοποθεσία πάνω τους βάση των συντεταγμένων u και v.



Άλλη μια ομοιότητα των surfaces με τα curves είναι ότι και τα surfaces υποστηρίζουν το evaluator object το οποίο στην περίπτωση τους ονομάζεται SurfaceEvaluator object και μπορεί να λειτουργίσει με τα face objects αλλά και με οποιοδήποτε από τα surface geometry objects. Επιθυμητό όμως είναι να χρησιμοποιείται με face objects διότι λαμβάνει και άλλες παραμέτρους υπόψιν του σχετικά με το σώμα που εξετάζει. Για παράδειγμα, όταν λαμβάνει μια κάθετο ως προς το face object ενός στερεού πάντα θα είναι από την εξωτερική πλευρά του στερεού σώματος.

Η παρακάτω λίστα αναφέρει τις πιο κοινές λειτουργίες του SurfaceEvaluator:

1. **ParamRangeRect:** Επιστρέφει τις μέγιστες και ελάχιστες τιμές για τις u και v παραμέτρους της επιφάνειας.

2. **GetPointAtParam:** Επιστρέφει το σημείο στον χώρο το οποίο αντιστοιχεί στις δοσμένες u και v παραμέτρους.

3. **GetParamAtPoint:** Επιστρέφει τις u-v παραμέτρους ενός δοσμένου σημείου στον χώρο.

4. **IsParamOnFace:** Υποδεικνύει αν οι δοσμένες u-v παράμετροι βρίσκονται πάνω στο face. Αυτή η λειτουργία είναι εξαιρετικά χρήσιμη στην περίπτωση πού υπάρχουν τρύπες στο face, διότι μα αναφέρει αν το σημείο αυτό είναι πάνω σε στερεά περιοχή του face ή όχι.

5. **GetNormal:** Επιστρέφει τα κάθετα διανύσματα ως προς την επιφάνεια στο σημείο που ορίζουν οι δοσμένες u-v παράμετροι.

Παρακάτω υπάρχει ένα μικρό πρόγραμμα το οποίο επιστρέφει το κάθετο διάνυσμα (normal vector) για ένα σημείο σε ένα επιλεγμένο face. Το σημείο που χρησιμοποιείται είναι στο κέντρο του παραμετρικού χώρου της επιφάνειας.

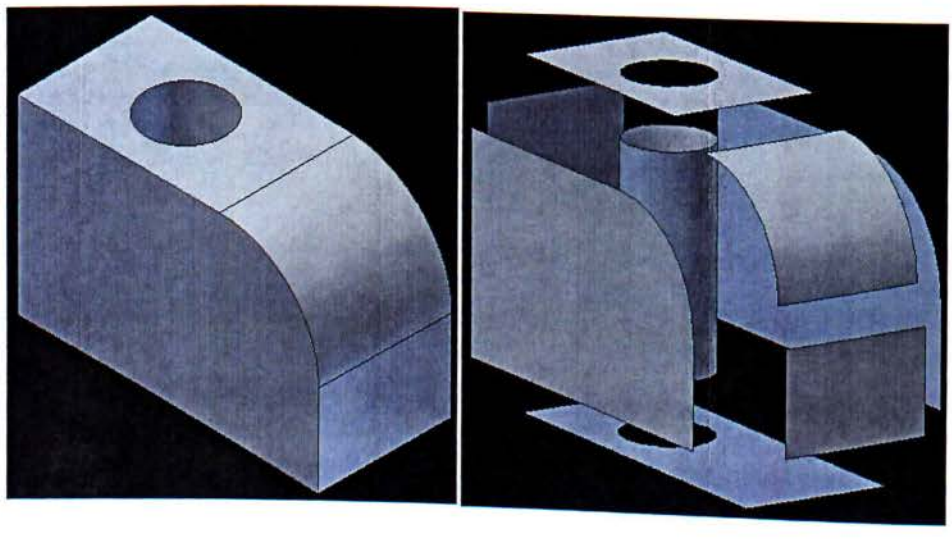
```
' Get the surface evaluator from the face.
Dim oSurfEval As SurfaceEvaluator
Set oSurfEval = oFace.Evaluator

' Get the parametric range of the surface.
Dim oParamRange As Box2d
Set oParamRange = oSurfEval.ParamRangeRect

' Calculate the u-v values at the parametric center of the surface.
' (This code is bigger than it should be to work around a VBA issue.)
Dim adParamCenter(1) As Double
Dim U As Double, V As Double
U = oParamRange.MinPoint.X
V = oParamRange.MaxPoint.X
adParamCenter(0) = (U + V) / 2
U = oParamRange.MinPoint.X
V = oParamRange.MaxPoint.X adParamCenter(1) = (U + V) / 2

' Get the normal at the u-v parameter.
Dim adNormal(2) As Double
Call oSurfEval.GetNormal(adParamCenter, adNormal)

' Print the normal vector.
```

```
Debug.Print " Normal: " & Format(adNormal(0), "0.000000") & "," &
                          Format(adNormal(1), "0.000000") & "," &
                          Format(adNormal(2), "0.000000")

' Get the model space coordinate of the parameter so we
' know where in space the normal was calculated.
Dim adPoint(2) As Double
Call oSurfEval.GetPointAtParam(adParamCenter, adPoint)

' Print the coordinate.
Debug.Print " Normal coordinate: " & Format(adPoint(0), "0.000000") & "," &
                                    Format(adPoint(1), "0.000000") & "," &
                                    Format(adPoint(2), "0.000000")

End Sub
```

f) Solids

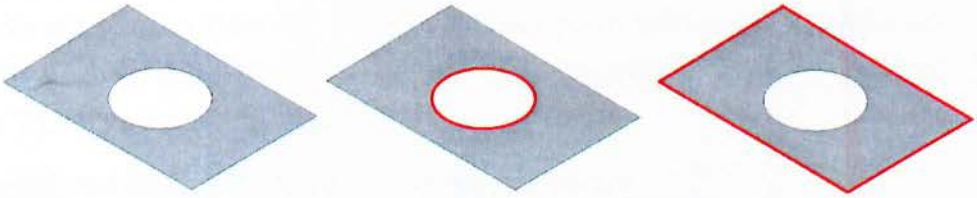
Τα εκάστοτε σχεδιασμένα μοντέλα αντιπροσωπεύονται από τα λεγόμενα solids (στερεά σώματα), τα οποία ακολουθούν την λογική του B-rep (Boundary Representation). Αυτό σημαίνει ότι μια σειρά από ενωμένες επιφάνειες οι οποίες ορίζουν τα εξωτερικά όρια ενός όγκου. Αν και τα μοντέλα φαίνονται στέρεα, οπτικά αλλά και λόγο των ενεργειών που μπορούν να πραγματοποιηθούν πάνω τους, όπως η μέτρηση της μάζας τους κτλπ, στην πραγματικότητα είναι μόνο επιφάνειες. Ο τρόπος σύνδεσης των επιφανειών ανήκει στην τοπολογία του μοντέλου.

Στο API τα solids αντιπροσωπεύονται από το SurfaceBody object και κάθε επιφάνεια απο αυτές που αποτελούν το μοντέλο αντιπροσωπεύεται απο ένα face object ανεξαρτήτως σχήματος. Όπως φαίνεται στην παρακάτω εικόνα το solid αυτό αποτελείται από 8 faces.



Βέβαια μια ομάδα από faces που ορίζουν έναν όγκο δεν είναι αρκετή για να ορίσει ένα solid, αυτά τα faces πρέπει να είναι και ενωμένα συγχρόνως. Η ένωση μεταξύ των faces ορίζεται από τα Edge objects και αντιπροσωπεύεται γεωμετρικά από τα curve objects. Κάθε face γνωρίζει τα edges που ορίζουν τα όρια του και κάθε Edge γνωρίζει ποία faces συνδέει. Ένας ακόμα παράγοντας της τοπολογίας που μπορεί να μας φανεί χρήσιμος είναι τα loops. Στο API τα loops αντιπροσωπεύονται από το EdgeLoop object. Τα loops είναι μια ομάδα συνδεδεμένων edges.

Η παρακάτω εικόνα δείχνει ένα face με δύο loops. Το εσωτερικό loop αποτελείται από ένα Edge ενώ το εξωτερικό loop αποτελείται από 4 Edges. Τα loops ορίζουν τα σύνορα του face.



Ένα ακόμα B-Rep entity είναι το vertex object, το οποίο ορίζει το σημείο που συναντιούνται τουλάχιστον 3 edges. Κάθε Edge γνωρίζει το vertex της κάθε άκρης του και κάθε vertex γνωρίζει ποία Edges ενώνει.

g) Math utility objects

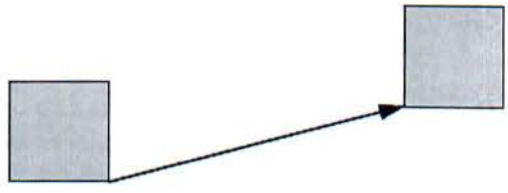
i. Transient geometry object

Ως τώρα έχουμε δει πώς μπορούμε να ανακτήσουμε geometry objects από διάφορα entities, επίσης πώς μπορούμε να δημιουργήσουμε geometry objects απευθείας μέσω του TransientGeometry object. Εκτός από τα geometry objects το TransientGeometry object υποστηρίζει και κάποια μαθηματικά objects, δύο από τα πιο χρήσιμα είναι τα Vector (διανύσματα) objects και τα Matrix(πίνακες) objects και επειδή τα συγκεκριμένα δεν έχουν γραφικό-ορατό αντίκτυπο είναι και πιο δύσκολο να κατανοηθούν.

ii. Vector object

Ένα διάνυσμα αποτελεί έναν εύκολο τρόπο να καθορίσει κανείς μια διεύθυνση και ένα μέτρο. Τα vector αποτελούνται απο τρεις τιμές x,y,z (τα 2D vectors αποτελούνται από x και y τιμές). Παρόλο που τα δεδομένα αυτά είναι απλά τα vector objects υποστηρίζουν ένα μεγάλο αριθμό methods και properties που επιτρέπουν την λήψη πληροφοριών καθώς και την εύκολη διαχείριση τους.

Μια κοινή χρήση των vectors είναι ο ορισμός της κίνησης ενός object (η ορολογία γι' αυτό είναι translation ενός object) στον χώρο χωρίς καμία περιστροφή γύρο από τον άξονα του. Για παράδειγμα ένα vector (3,1,0) ορίζει την κίνηση ενός object, 3 μονάδες προς την διεύθυνση του άξονα x, 1 μονάδα προς τον άξονα y και 0 μονάδες προς τον άξονα των z, με αποτέλεσμα την συνολική κίνηση των 3,162 μονάδων η οποία απεικονίζεται παρακάτω.



Μια ακόμα κοινή χρήση των vectors είναι ο καθορισμός της διεύθυνσης. Στο προηγούμενο παράδειγμα το vector χρησιμοποιήθηκε για να ορίσει την διεύθυνση και το μέτρο που θα μετακινηθεί ένα αντικείμενο, αλλά όταν ένα vector χρησιμοποιείται για τον καθορισμό μίας διεύθυνσης τότε χρησιμοποιούμε συνήθως το UnitVector object. Το μήκος του UnitVector είναι πάντα 1 μονάδα. Ένα παράδειγμα είναι όταν λαμβάνουμε προσανατολιστικές πληροφορίες από ένα cylinder object που υποστηρίζει το AxisVector property το οποίο επιστρέφει ένα UnitVector. Αυτό το UnitVector ορίζει την διεύθυνση του άξονα του κυλίνδρου.

Παρακάτω αναφέρονται κάποιες λειτουργίες που χρησιμεύουν στην σύγκριση μεταξύ δύο vectors:

1. **AngleTo:** Επιστρέφει την γωνία μεταξύ δύο vectors .
2. **IsParallelTo:** Επιστρέφει αν είναι η όχι δύο vectors παράλληλα.
3. **IsPerpendicularTo:** Επιστρέφει αν δύο vectors είναι κάθετα.

Οι επόμενες λειτουργίες επιτρέπουν τον συνδυασμό δύο vectors:

1. **AddVector:** Προσθέτει δύο vectors.
2. **SubtractVector:** Αφαιρεί ένα vector από το άλλο.
3. **CrossProduct:** Επιστρέφει ένα τρίτο vector κάθετο στο επίπεδο που ορίζουν τα δύο δοσμένα vectors. Αυτή η λειτουργία είναι χρήσιμη για τον ορισμό αξόνων συντεταγμένων.
4. **DotProduct:** Επιστρέφει το εσωτερικό γινόμενο των δύο διανυσμάτων.

Το vector object εν αντιθέσει με το UnitVector object υποστηρίζει κάποιες λειτουργίες σχετικά με το μήκος του:

1. **Length:** Επιστρέφει την τιμή του μήκους του vector.
2. **Normalize:** Μετατρέπει το μήκος του vector σε 1 μονάδα.
3. **ScaleBy:** Μετατρέπει την τιμή του μήκους ανάλογα με την εισερχόμενη τιμή.

Τέλος, υπάρχει η δυνατότητα μετατροπής του vector σε UnitVector και το αντίστροφο από τις λειτουργίες AsUnitVector και AsVector αντίστοιχα.

iii. Matrix objects

Τα matrix objects είναι πίνακες τιμών ακριβώς όπως γνωρίζουμε απο τα μαθηματικά και χρησιμεύουν είτε για να ορίσουμε ένα ορθοκανονικό σύστημα συντεταγμένων, είτε για να κάνουμε αλλαγές στην θέση και τον προσανατολισμό ενός αντικειμένου. Στην συνέχεια θα ασχοληθούμε με την δεύτερη χρήση των matrices, η οποία εφαρμόζεται και στο πρόγραμμα. Έτσι λοιπόν για να αλλάξουμε την θέση ενός αντικειμένου δημιουργούμε ένα matrix object, το οποίο ορίζει την αλλαγή που θέλουμε να γίνει και ύστερα εφαρμόζουμε το matrix στο αντικείμενο.

Στο παράδειγμα που ακολουθεί, υποθέτουμε ότι θέλουμε να μετακινήσουμε απευθείας με ένα matrix κάποια αντικείμενα από ένα assembly κατά 5 cm προς την διεύθυνση του άξονα x.

```
' Iterate through the occurrences in the assembly.
Dim oOcc As ComponentOccurrence
For Each oOcc In oAsmDoc.ComponentDefinition.Occurrences
   ' Get the matrix from the current occurrence.
      Dim oMatrix As Matrix
```

```
    Set oMatrix = oOcc.Transformation

    ' Edit the cell that defines the X component of the origin.
     oMatrix.Cell(1, 4) = oMatrix.Cell(1, 4) + 5

  ' Set the transformation of the occurrence using the modified matrix.
oOcc.Transformation = oMatrix
Next
```

Ενώ στο παράδειγμα που ακολουθεί, η ίδια διαδικασία γίνεται μέσω ενός matrix που ορίζει την αλλαγή και εφαρμόζεται στο υπάρχον matrix του αντικειμένου (το οποίο ορίζει την τοποθεσία του) μέσω της λειτουργίας TransformBy.

```
' Create the matrix that defines the transform.
Dim oTransMatrix As Matrix
Set oTransMatrix = ThisApplication.TransientGeometry.CreateMatrix oTransMatrix.Cell(1, 4) = 5

' Iterate through the occurrences in the assembly.
Dim oOcc As ComponentOccurrence
For Each oOcc In oAsmDoc.ComponentDefinition.Occurrences
    ' Get the matrix from the current occurrence.
      Dim oMatrix As Matrix
      Set oMatrix = oOcc.Transformation

    ' Apply the transform to the matrix.
     Call oMatrix.TransformBy(oTransMatrix)

  ' Set the transformation of the occurrence using the modified matrix.    oOcc.Transformation = oMatrix
Next
```

Και οι δύο κώδικες έχουν το ίδιο αποτέλεσμα, στη πρώτη περίπτωση η απευθείας επεξεργασία του matrix φαινομενικά είναι πιο εύκολη, αλλά αν η αλλαγή που πρέπει να εφαρμοστεί είναι πιο πολύπλοκη η ασφαλέστερη λύση είναι η δεύτερη μέθοδος. Για πολύπλοκες αλλαγές υπάρχει η δυνατότητα του συνδιασμού πολλών matrix, που ορίζουν το καθένα μία αλλαγή, σε ένα μόνο matrix.

Το παράδειγμα που ακολουθεί δείχνει αυτήν την διαδικασία. Ας υποθέσουμε λοιπόν ότι θέλουμε να περιστρέψουμε ένα ακτικείμενο κατά 45 μοίρες γύρο από τον άξονα των x, ύστερα 30 μοίρες γύρο απο τον άξονα των y και τέλος να το μετακινήσουμε 5 cm προς την διεύθυνση του x άξονα και 3 cm προς την διεύθυνση του z άξονα. Έτσι δημιουργούμε ένα matrix για κάθε μια από τις παραπάνω αλλαγές και πολλαπλασιάζοντας τα matrices μεταξύ τους δημιουργείται ένα νέο matrix που περιέχει όλες τις αλλαγές μαζί. Η σειρά με την οποία θα πολλαπλασιαστούν τα matrices κάνει διαφορά, διότι η αλλαγές εφαρμόζονται με την σειρά που είχαν όταν γινόταν ο πολλαπλασιασμός. Η εντολή του πολλαπλασιασμού είναι το TransformBy.

```
Dim dPi As Double
dPi = Atn(1) * 4

Dim oTG As TransientGeometry
Set oTG = ThisApplication.TransientGeometry
' Create a matrix that defines a 45° rotation around the x-axis.
Dim oTransMatrix As Matrix
Set oTransMatrix = oTG.CreateMatrix
Call oTransMatrix.SetToRotation(dPi / 4, oTG.CreateVector(1, 0, 0), _      oTG.CreatePoint(0, 0, 0))
' Create a matrix that defines a 30° rotation around the y-axis and apply it.
Dim oTempMatrix As Matrix
Set oTempMatrix = oTG.CreateMatrix
Call oTempMatrix.SetToRotation(dPi / 6, oTG.CreateVector(0, 1, 0), _ oTG.CreatePoint(0, 0, 0))
Call oTransMatrix.TransformBy(oTempMatrix)
' Create a matrix that defines a 5 cm X and 3 cm Z move and applies it. oTempMatrix.SetToIdentity
Call oTempMatrix.SetTranslation(oTG.CreateVector(5, 0, 3))
Call oTransMatrix.TransformBy(oTempMatrix)

' Iterate through the occurrences in the assembly.
Dim oOcc As ComponentOccurrence
For Each oOcc In oAsmDoc.ComponentDefinition.Occurrences
   ' Get the matrix from the current occurrence.
      Dim oMatrix As Matrix
      Set oMatrix = oOcc.Transformation
    ' Apply the transform to the matrix.
    Call oMatrix.TransformBy(oTransMatrix)

' Set the transformation of the occurrence using the modified matrix.          oOcc.Transformation =
oMatrix

Next
```

Το παράδειγμα αυτό δεν διαφέρει από τα προηγούμενα, απλά εκμεταλλεύεται κάποιες λειτουργίες του matrix object ώστε να διευκολύνει τον χρήστη. Χαρακτηριστικές λειτουργίες είναι:

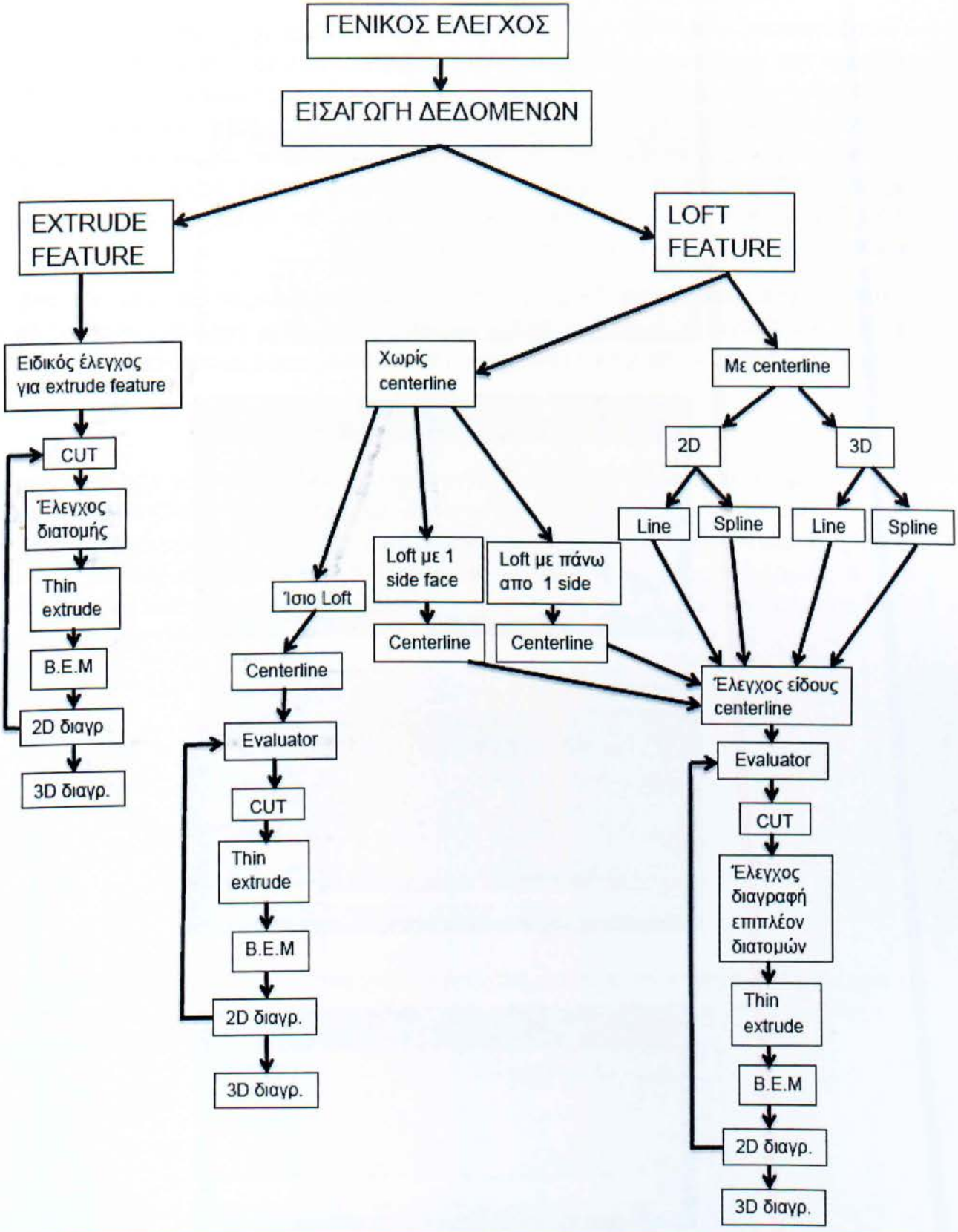
1. **SetToRotation:** που ορίζει την περιστροφή που θα εφαρμόσει το matrix.

2. **Invert:** Αντιστρέφει την αλλαγή του matrix.

4)Ανάλυση Προγράμματος

Ακολουθεί η αναλυτική περιγραφή του τρόπου με τον οποίο γράφτηκε το πρόγραμμα της πτυχιακής, των μεθόδων που χρησιμοποιήθηκαν, καθώς επίσης και της λογικής που κρύβεται πίσω απο αυτές. Ακόμα αναφέρονται τα προβλήματα που αντιμετωπίστηκαν.

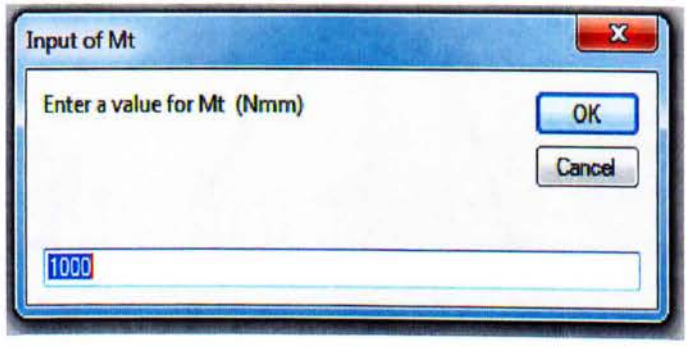
a) Διάγραμμα Ροής

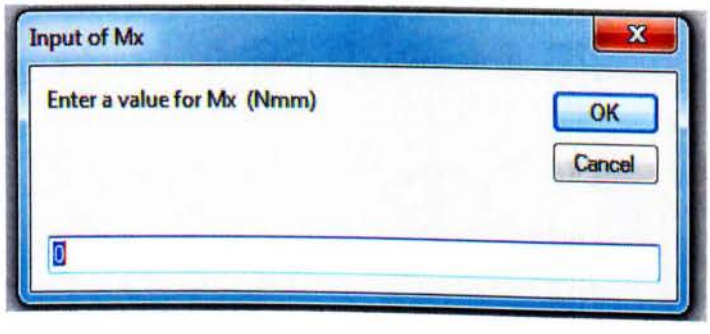


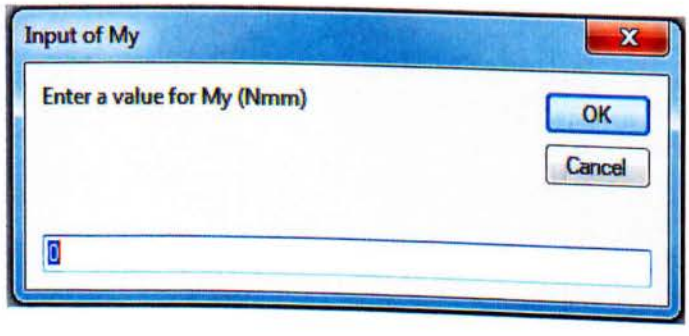
b)Αρχή - Γενικός έλεγχος- Εισαγωγή δεδομένων

Το πρόγραμμα ξεκινώντας πραγματοποιεί κάποιους ελέγχους για να διασφαλίσει ότι το μοντέλο που θα εξετάσει προέρχεται είτε από loft είτε απο extrude features και όχι από άλλα features ή συνδυασμούς τους. Ο εντοπισμός της όποιας παρατυπίας συνοδεύεται με το αντίστοιχο μήνυμα σφάλματος, το οποίο ενημερώνει τον χρήστη για το είδος του λάθος του και το πρόγραμμα τερματίζει την λειτουργία του, ώστε να δώσει την ευκαιρία στον χρήστη να διορθώσει το πρόβλημα και να επανεκκινήσει το πρόγραμμα. Στην περίπτωση που δεν γινόταν αυτος ο προκαταρκτικός έλεγχος θα υπήρχε μεγάλη πιθανότητα σφάλματος στην μετέπειτα εκτέλεση του προγράμματος.

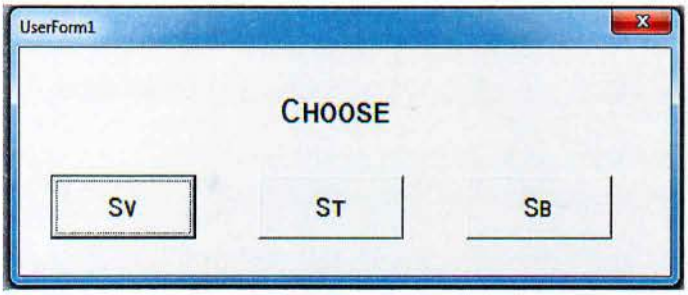
Στην συνέχεια εμφανίζονται 3 μηνύματα εισαγωγής δεδομένων, τα οποία ζητούν από τον χρήστη να εισάγει τις τιμές των ροπών κάμψης Mbx, Mby, και ροπή στρέψης Mt σε (Nmm). Σαν δεδομένο θεωρείται επίσης και η ίδια η δοκός που έχει σχεδιαστεί.



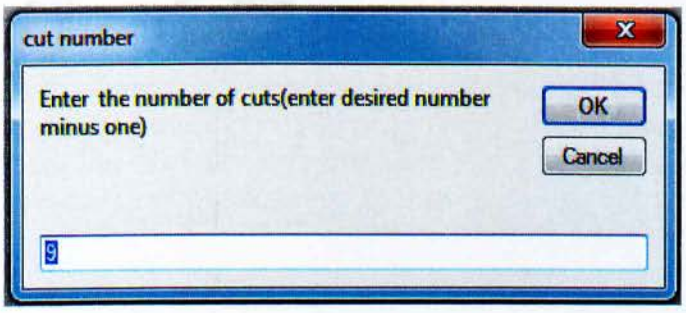




Το επόμενο μήνυμα που εμφανίζεται ζητάει από τον χρήστη να επιλέξει για το είδος των τάσεων που θα απεικονίζουν τα 2D και 3D διαγράματα, οι επιλογές είναι ανάμεσα σε: Sv, St, Sb.



Το τελευταίο μήνυμα που εμφανίζεται είναι για τον ορισμό του αριθμού των τομών που θα γίνουν στο μοντέλο. Αυτή η τελευταία επιλογή δόθηκε επειδή όσο περισσότερες τομές κάνει το πρόγραμμα τόσο πιο πολύ επιβαρύνεται η μνήμη ram του υπολογιστή και η υπολογιστική του ισχύς, καθώς το πρόγραμμα είναι αρκετά μεγάλο και «βαρυ». Έτσι ο χρήστης έχει την επιλογή του αριθμού των τομών ανάλογα με την ακρίβεια των αποτελεσμάτων που επιθυμεί.



Σε μια άλλη πτυχιακή θα μπορούσε να αναπτυχθεί μια αυτοματοποιημένη διαδικασία, η οποία να καθορίζει την απόσταση ανάμεσα στις τομές, αναλόγως με την μεταβολή των αποτελεσμάτων τις συγκριτικής τάσης ανάμεσα σε δυο διαδοχικές τομές.

c) Loft Feature - Χωρίς centerline (κεντροβαρικός άξονας)

Η περίπτωση που έχουμε ένα μοντέλο κατασκευασμένο με loft χωρίς centerline, χωρίζεται σε δύο κατηγορίες:

1. Τα μοντέλα των οποίων όλα τα SideFaces έχουν γεωμετρία BsplineSurface. Αυτή η κατηγορία χωρίζεται σε δύο υποκατηγορίες:

a. Τα μοντέλα που έχουν περισσότερα απο ένα (1) SideFaces.

b. Τα μοντέλα που έχουν (1) ένα μόνο SideFace.

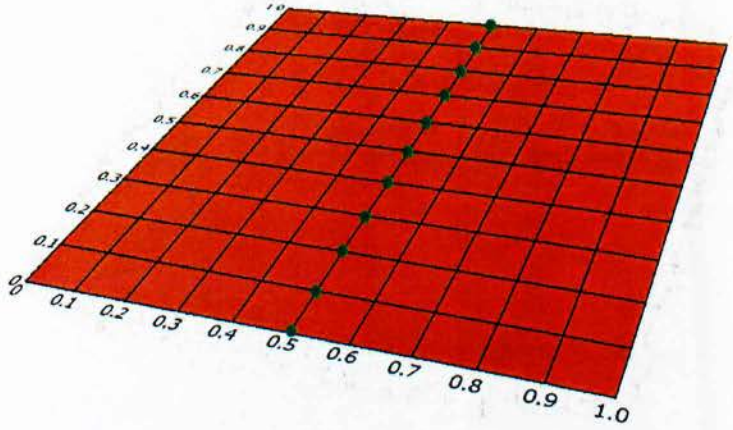
2. Τα μοντέλα που είτε έχουν έστω μία πλευρά που έχει γεωμετρία PlaneSurface, είτε αποτελούνται από ένα SideFace που έχει γεωμετρία cone ή EllipticalCone.

i. Περίπτωση 1.a

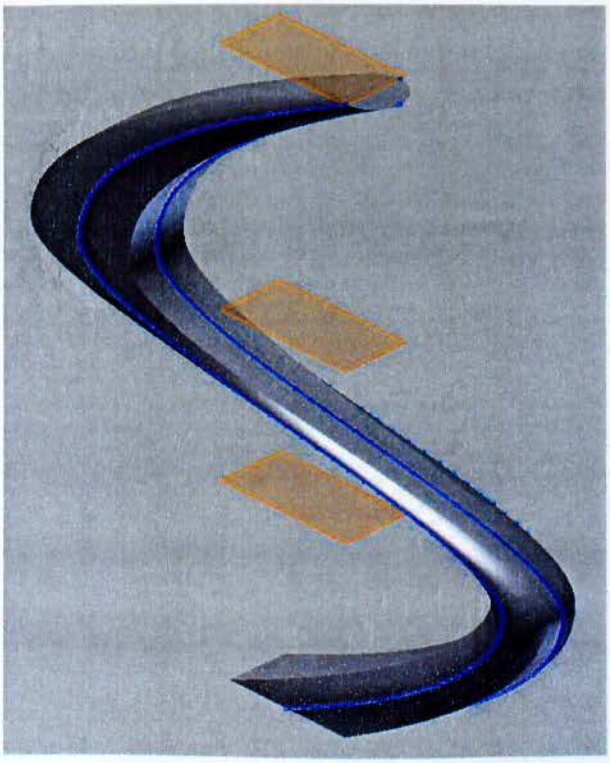
Είναι πιο συγκεκριμένα όταν το μοντέλο έχει φτιαχτεί με την χρήση του loft feature χωρίς centerline και ένα τουλάχιστον section του loft είναι φτιαγμένο με lines και όχι splines ή circles. Αυτό έχει ως αποτέλεσμα το inventor να δημιουργήσει sidefaces και edges για να ομαλοποιήσει το loft και το τελικό αποτέλεσμα. Πάνω σε αυτό το γεγονός βασίζεται και η υπορουτίνα που υπολογίζει την centerline και χρησιμοποιεί τα sidefaces ως μέσω.Η υπορουτίνα θα μπορούσε να χρησιμοποιεί τα edges και τον curve evaluator για τον ίδιο σκοπό, δεν τα επιλέγουμε όμως, διότι σε πολλές περιπτώσεις λόγο μεγάλης καμπυλότητας ένα edge μπορεί να αποτελείται από περισσότερα του ενός splines, έτσι θα χρειαζόταν το «ράψιμο» τους σε ένα spline, για να λειτουργήσει ομαλά ο curve evaluator με αποτέλεσμα το πρόγραμμα να γίνει αργό και αναξιόπιστο.

ii. Centerline (1.a)

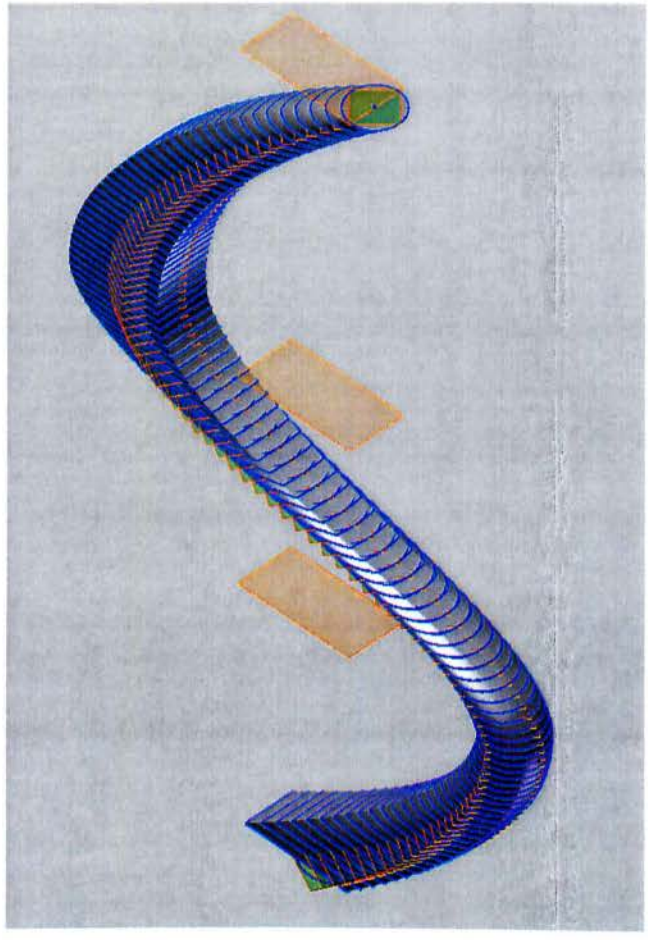
Η υπορουτίνα με μια επαναληπτική μέθοδο χρησιμοποιεί τον SurfaceEvaluator για να ορίσει έναν αριθμό από σημεία στο μέσο κάθε sideface, με την βοήθεια των παραμετρικών τιμών u-v, τα οποία αποθηκεύει σε ένα collection.

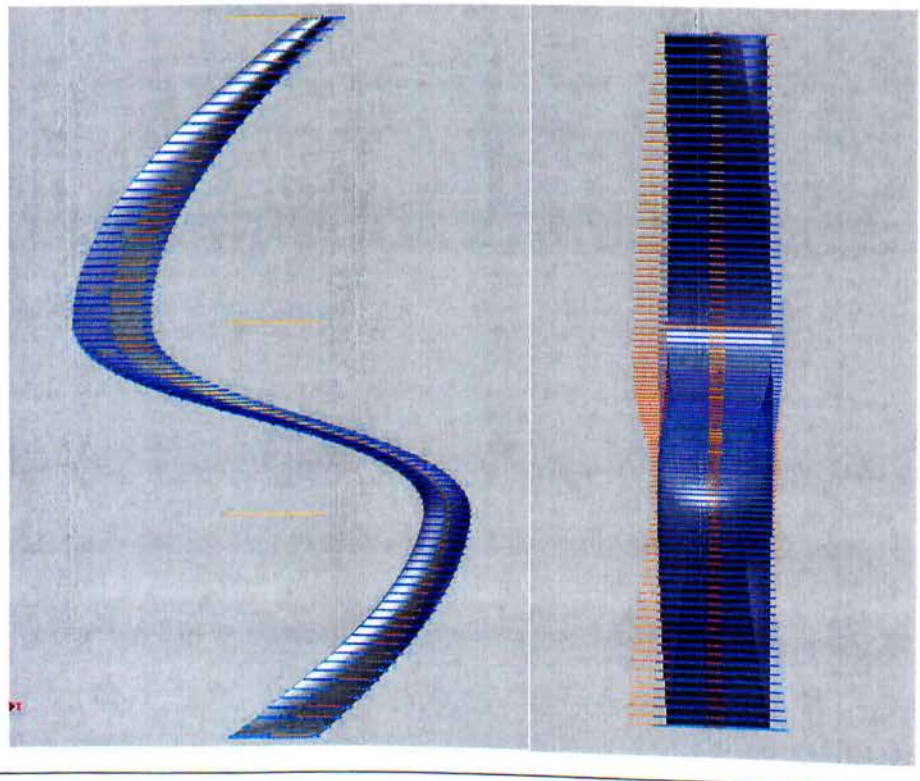


Στην συνέχεια δημιουργεί ένα 3D spline επάνω σε κάθε sideface με το collection των σημείων του κάθε sideface. Η επιλογή της 3D spline έγινε διότι το SketchPoints3D.add δέχεται μονάχα ως είσοδο Point objects και όχι collection με points, με αυτόν τον τρόπο τοποθετούμε με μία μόνο εντολή όλα τα σημεία στο sketch αποφεύγοντας ακόμα μια επαναληπτική μέθοδο, η οποία θα πέρναγε ένα προς ένα τα points και θα επιβάρυνε την απόδοση του προγράμματος.

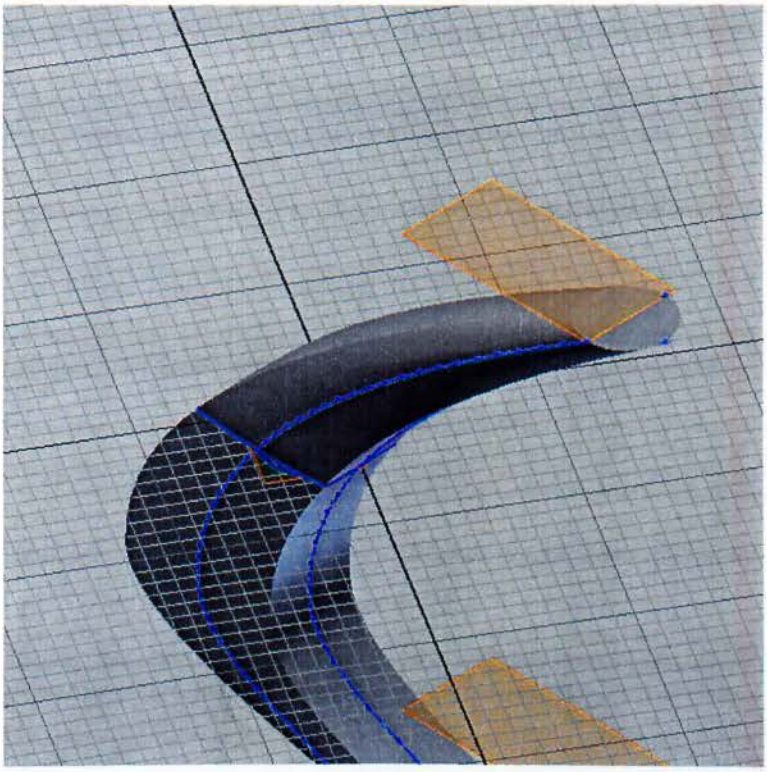


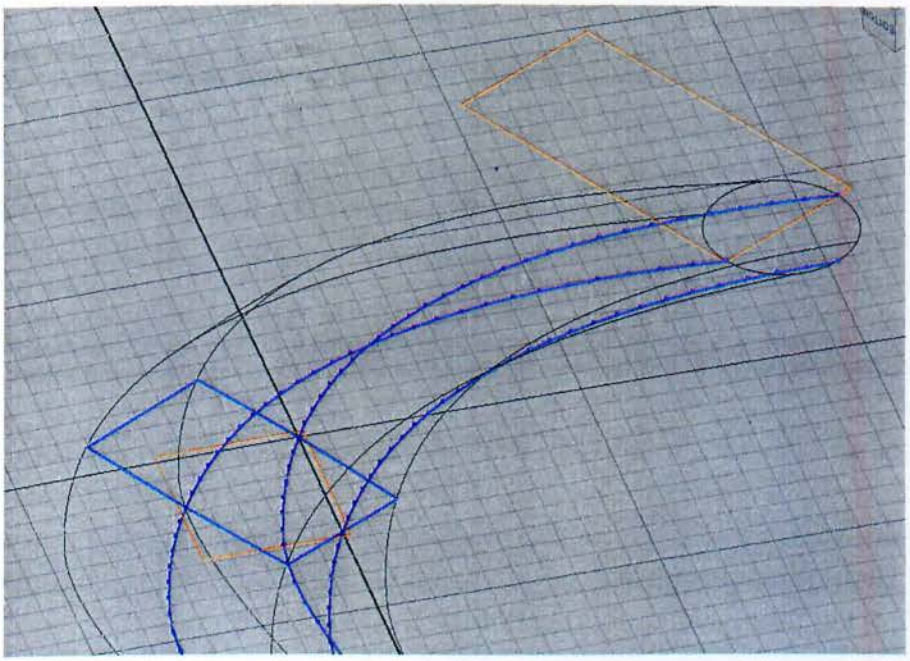
Το επόμενο βήμα, βασίζεται στο γεγονός ότι τρεία σημεία στον χώρο ορίζουν ένα επίπεδο. Σαρώνοντας τα σημεία των τριών πρώτων 3D sketch με την μέθοδο SetByThreePoints η υπορουτίνα δημιουργεί επίπεδα παράλληλα μεταξύ τους και κάθετα στο μοντέλο.



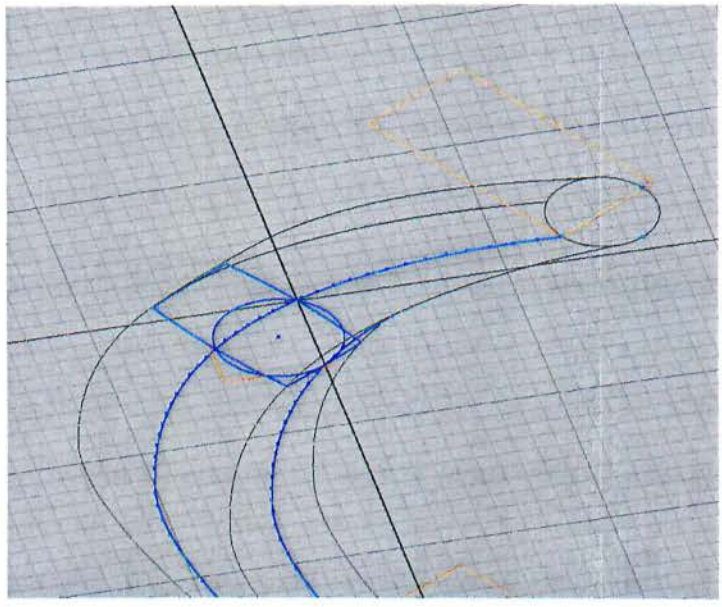


Σε κάθε επίπεδο δημιουργείται ένα sketch και με την εντολή ProjectCutEdges λαμβάνουμε τις διατομές του μοντέλου στα σημεία που το τέμνει το επίπεδο.





Επειδή όμως λόγο μεγάλης καμπυλότητας του μοντέλου το επίπεδο μπορεί να τέμνει το μοντέλο σε παραπάνω από μία διατομή πραγματοποιείται η παρακάτω διαδικασία. Στο sketch δημιουργείται επίσης ένα sketchcircle χρησιμοποιώντας τα τρία σημεία που όρισαν το επίπεδο, το οποίο μας χρησιμεύει στην συνέχεια.

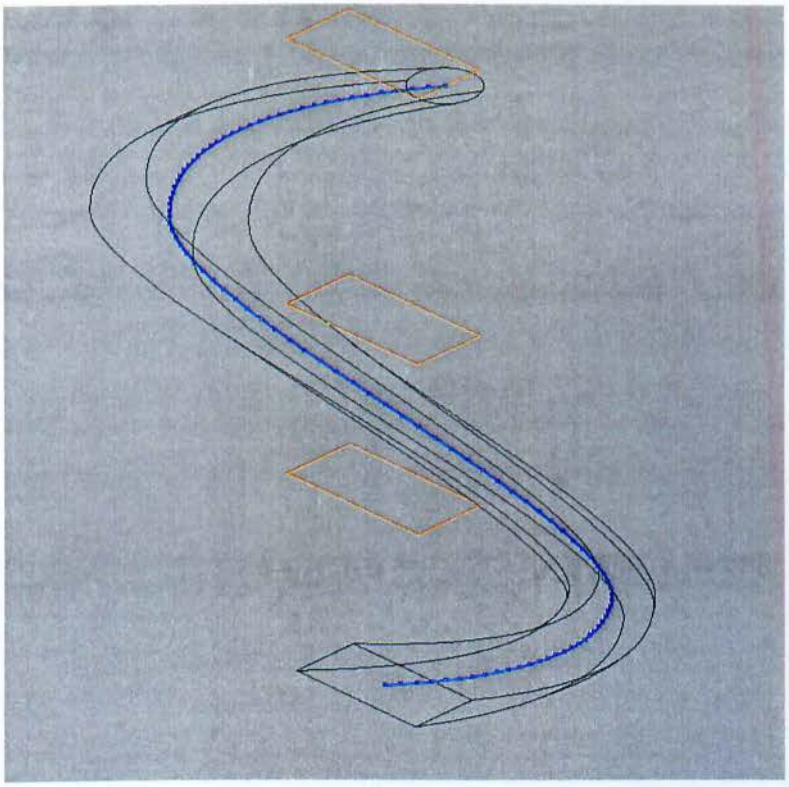


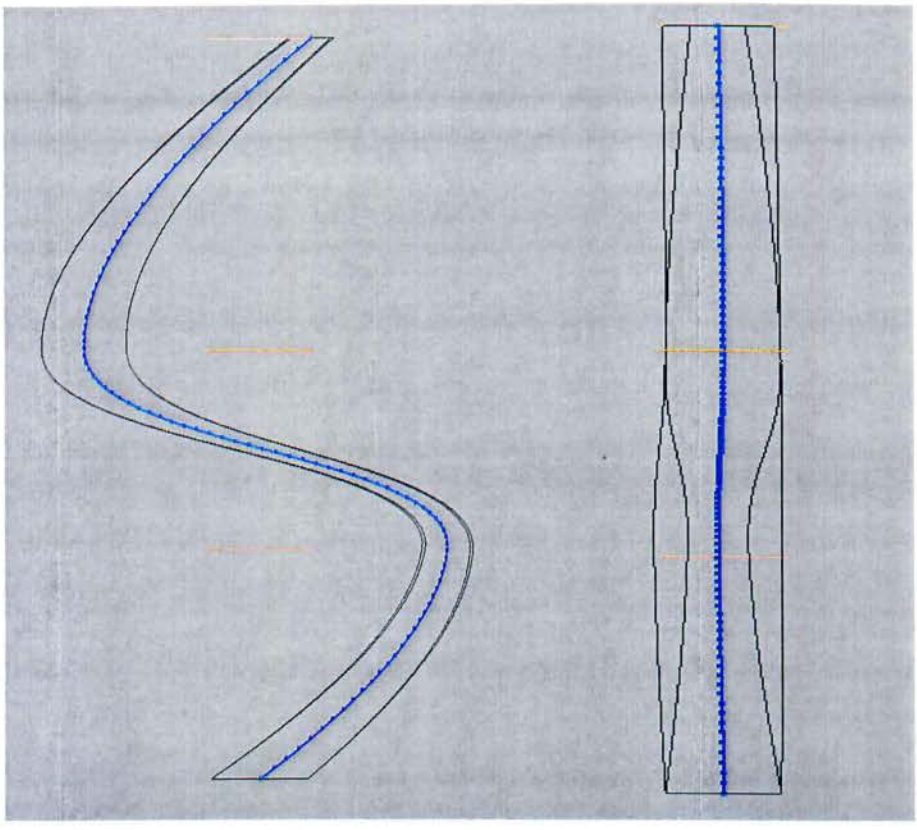
Τα entities που αποτελούν την κάθε διατομή μέσα από αυτήν την διαδικασία (ProjectCutEdges) είναι πάντα είτε τύπου sketchSpline είτε τύπου sketchLine, για να τα ξεχωρίσουμε από τα υπόλοιπα entities του sketch τα βάζουμε σε ένα collection.

Το collection αυτό το χρησιμοποιούμε σε μια επαναληπτική μέθοδο, η οποία αρχικά καταχωρεί στην μνήμη της το πρώτο entity του collection ως σημείο αναφοράς. Στην συνέχεια συγκρίνει αν συμπίπτει το end point του με το start point του επόμενου entity, αν ναι τότε καταχωρείται το δεύτερο σε ένα άλλο collection και η διαδικασία προχωράει στο επόμενο ζευγάρι entities. Παράλληλα ελέγχει αν το end point του εξεταζόμενου entity συμπίπτει με το start point του entity που έχει κρατηθεί στην μνήμη ώς σημείο αναφοράς, όταν συμβεί αυτό τότε σημαίνει ότι έχουμε μια πλήρη διατομή, τότε η υπορουτίνα καταχωρεί και το entity που ήταν το σημείο αναφοράς στο collection και με αυτό δημιουργεί ένα profile με αυτές τις πλευρές και προχωράει με τον ίδιο τρόπο για να ορίσει την επόμενη διατομή αν υπάρχει. Τα profiles αυτά στην ουσία ορίζουν τις διατομές που «έκοψε» το επίπεδο και είναι αυτά που πρέπει να συγκρίνουμε ώστε να βρούμε ποίο αντιστοιχεί στην σωστή διατομή και αυτό γίνεται με τη μεθόδο IsDisJoint του rangebox.

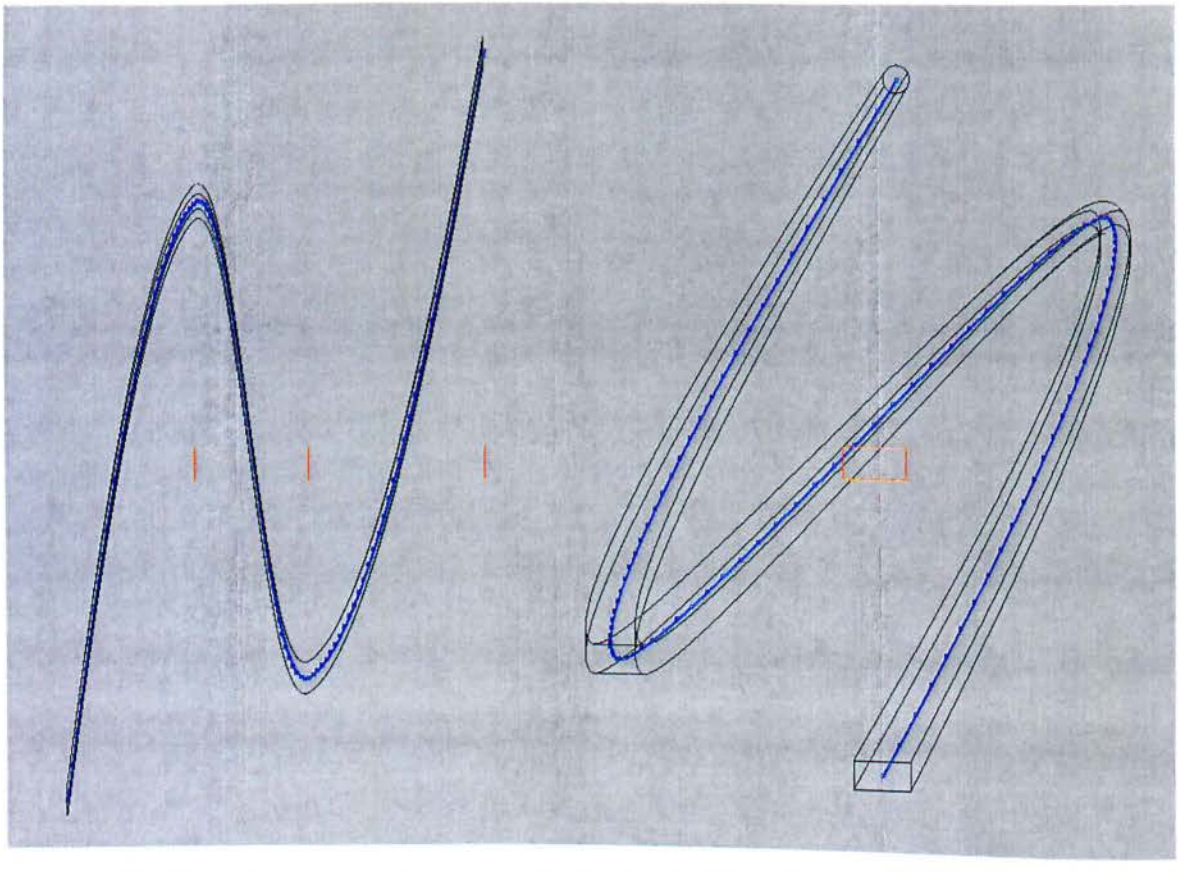
Το **πρόβλημα** όμως είναι ότι τα profiles δεν διαθέτουν rangebox ! γι' αυτό τον λόγο με την βοήθεια μίας επαναληπτικής μεθόδου δημιουργούμε rangebox για το κάθε profile. Η μέθοδος αυτή ορίζει το rangebox του πρώτου entity του profile ως το κύριο rangebox και το συγκρίνει με τα rangebox των υπολοίπων entities του profile. Αν το minpoint του rangebox δεν περιέχεται μέσα στο κύριο rangebox τότε με την εντολή extend, μεγαλώνει τόσο ώστε να περιέχει το minpoint, αντίστοιχα η διαδικασία γίνεται και για το maxpoint, μέχρις ότου το κύριο rangebox να περιέχει όλα τα υπόλοιπα rangebox και στην ουσία να έχει γίνει πλέον το rangebox του profile.

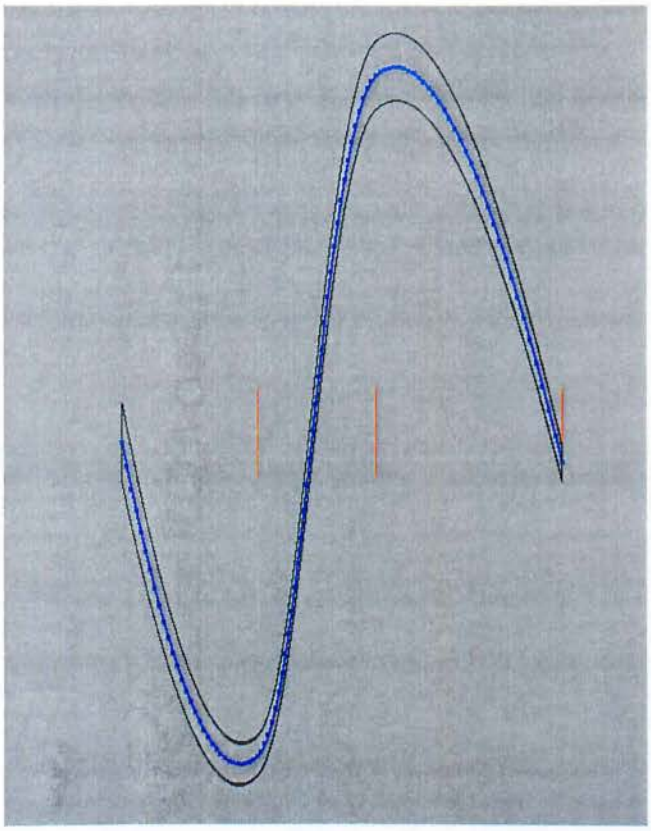
Στην συνέχεια η υπορουτίνα συγκρίνει το rangebox του sketchcircle με τα rangebox των profiles και καταλήγει στην επιλογή του σωστού. Από το profile εκείνο αποθηκεύουμε το centroid του, το οποίο είναι το κέντρο βάρους της διατομής σε ένα collection. Τέλος με όλα τα centroids από το collection δημιουργούμε σε ένα νέο 3D sketch ένα 3D spline το οποίο αποτελεί την centerline του μοντέλου.



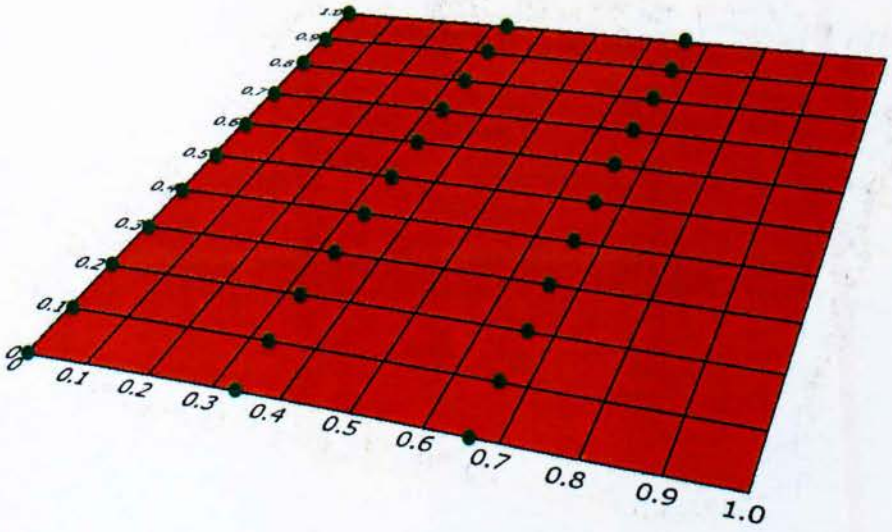


Αυτή η διαδικασία μπορεί να εφαρμοστεί ακόμα και στις πιο ακραίες εκδοχές όπου η καμπυλότητα της δοκού δεν μεταβάλλεται ως προς έναν μόνο άξονα όπως στο παραπάνω παράδειγμα.

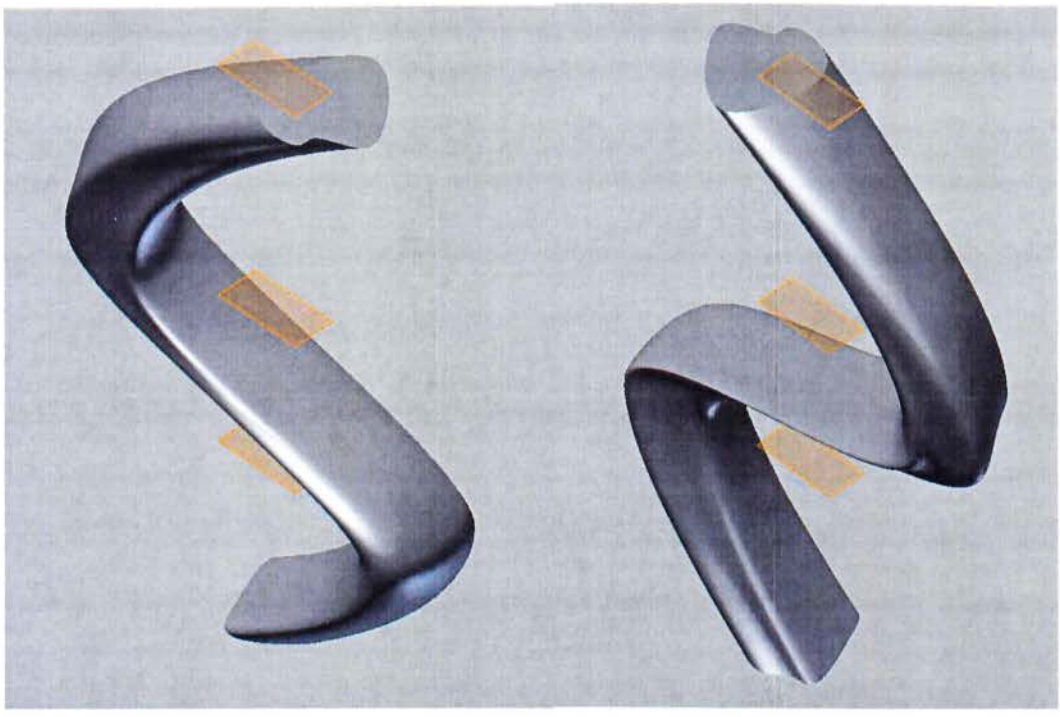


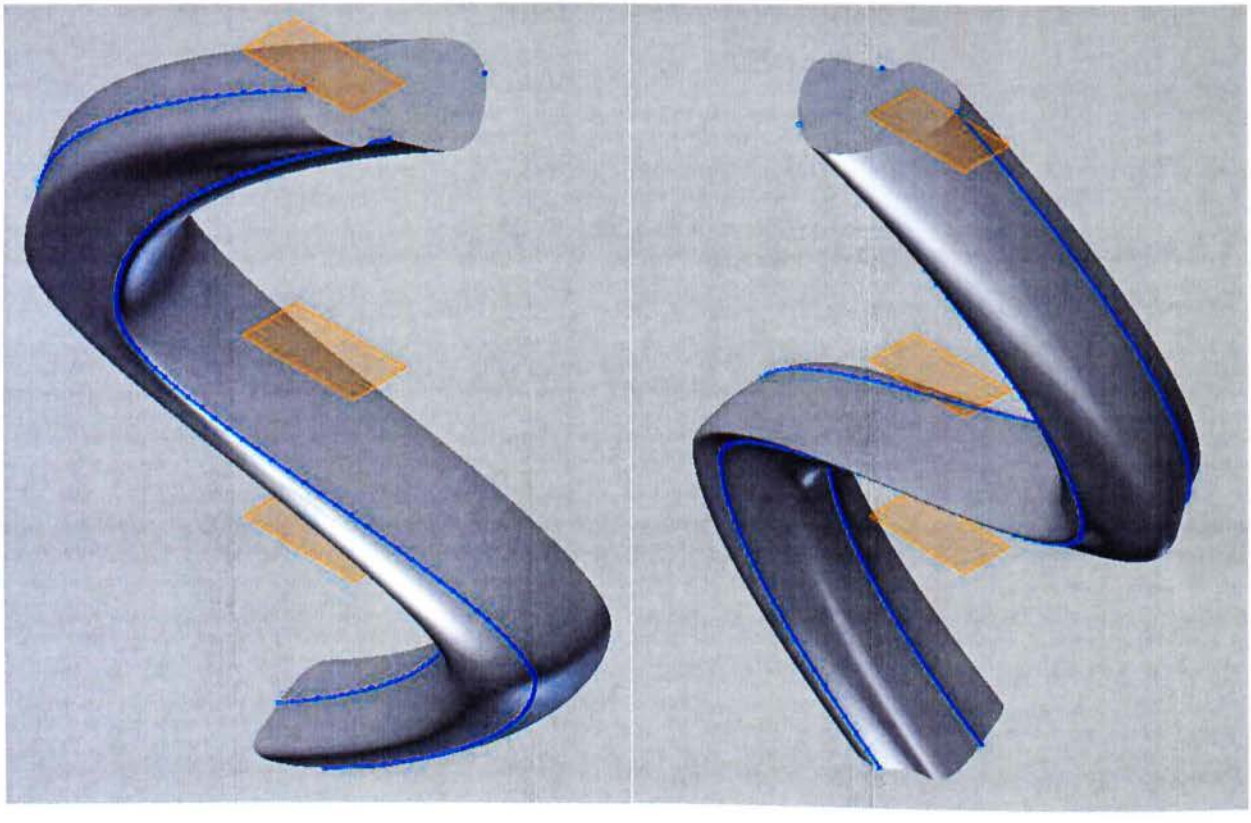


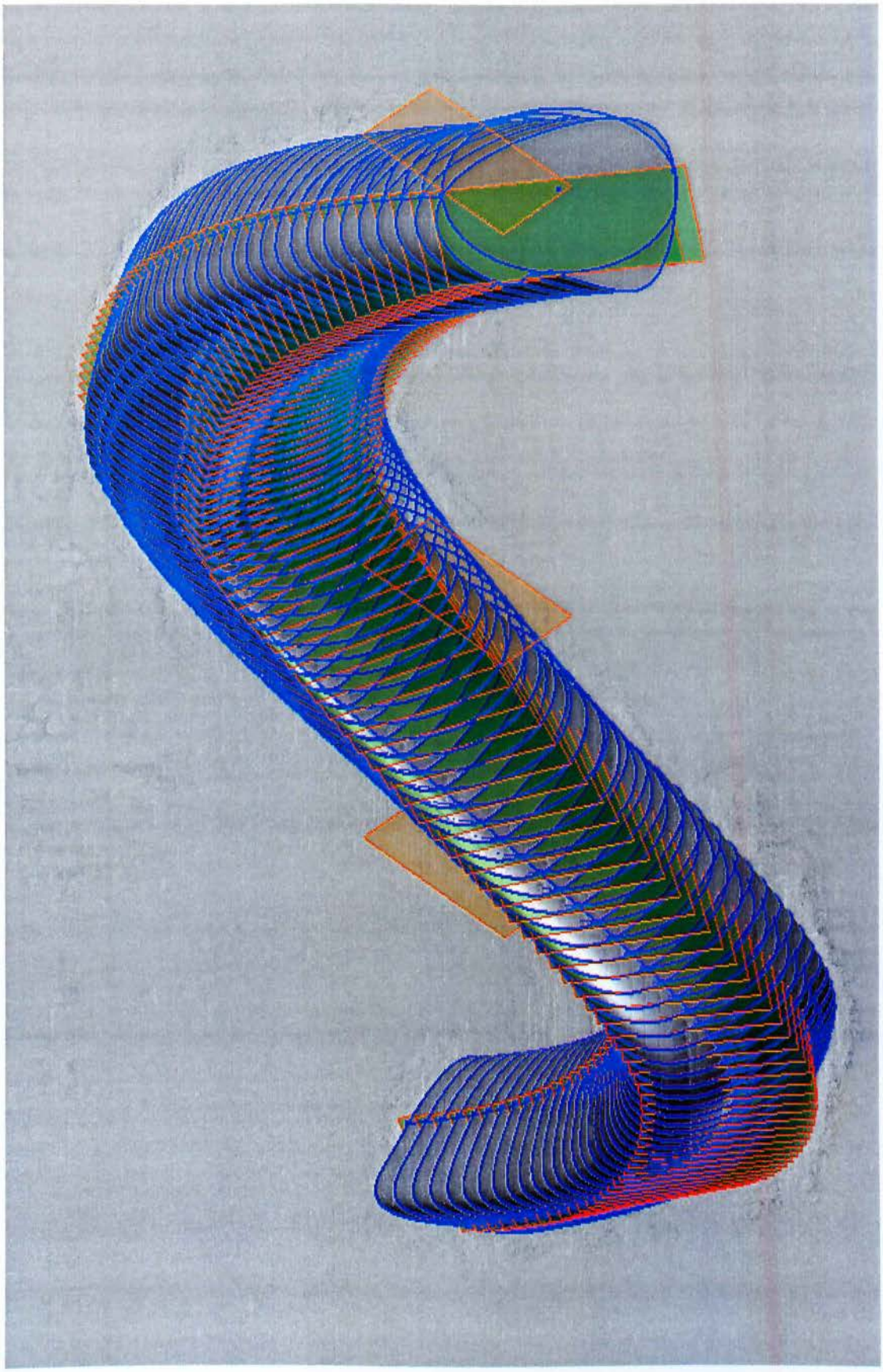
Η περίπτωση 1.b είναι όταν ένα μοντέλο είναι φτιαγμένο με το loft feature, χωρίς centerline και σε όλα του τα section έχει circle ή/και splines. Η διαδικασία που ακολουθείται είναι σχεδόν ίδια με την 1.a, το μόνο που αλλάζει είναι ότι αντί η υπορουτίνα να φτιάξει ένα spline στο κέντρο κάθε sideface, φτιάχνει 3 spline στο ένα (και μοναδικό) sideface.

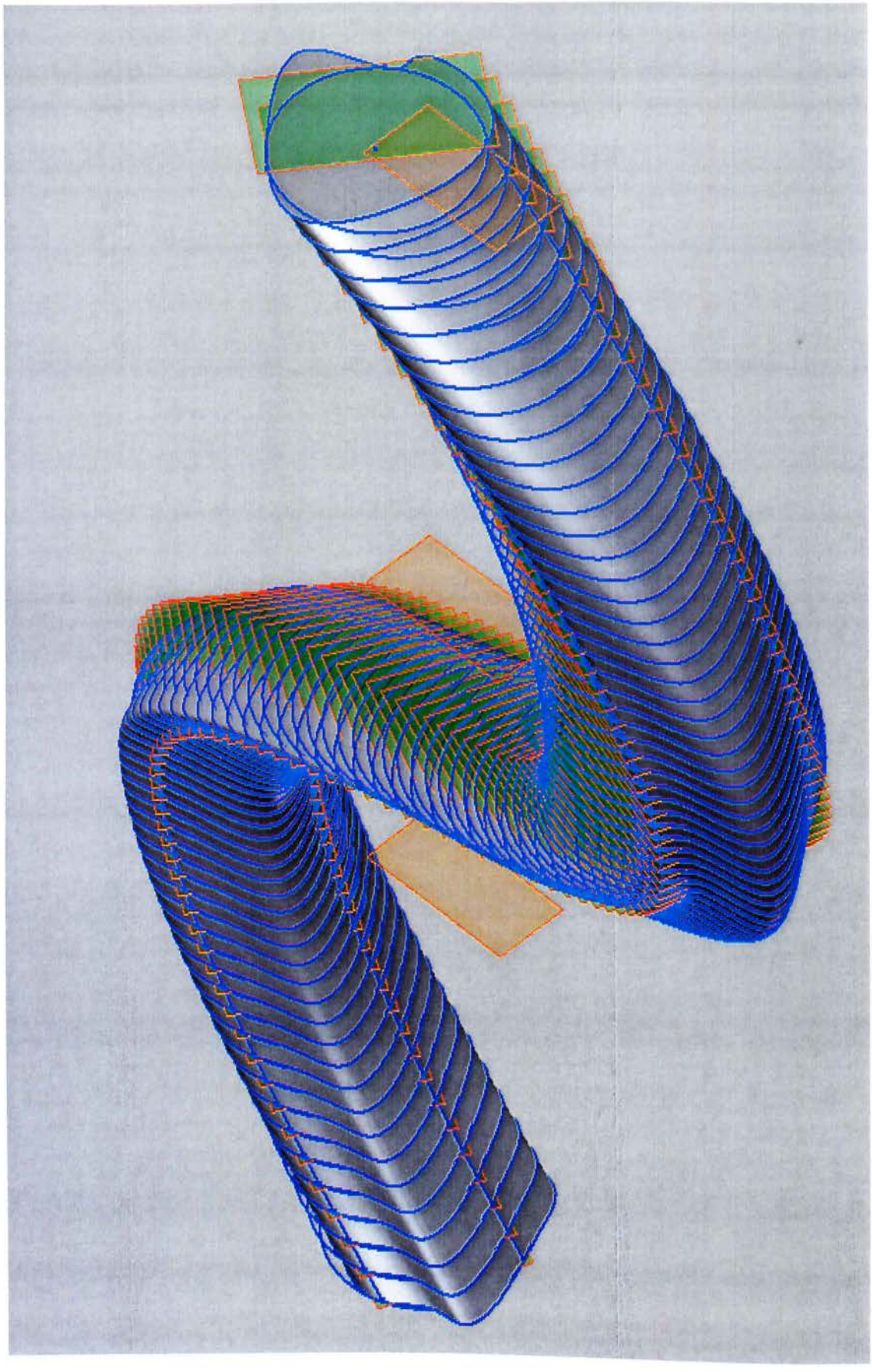


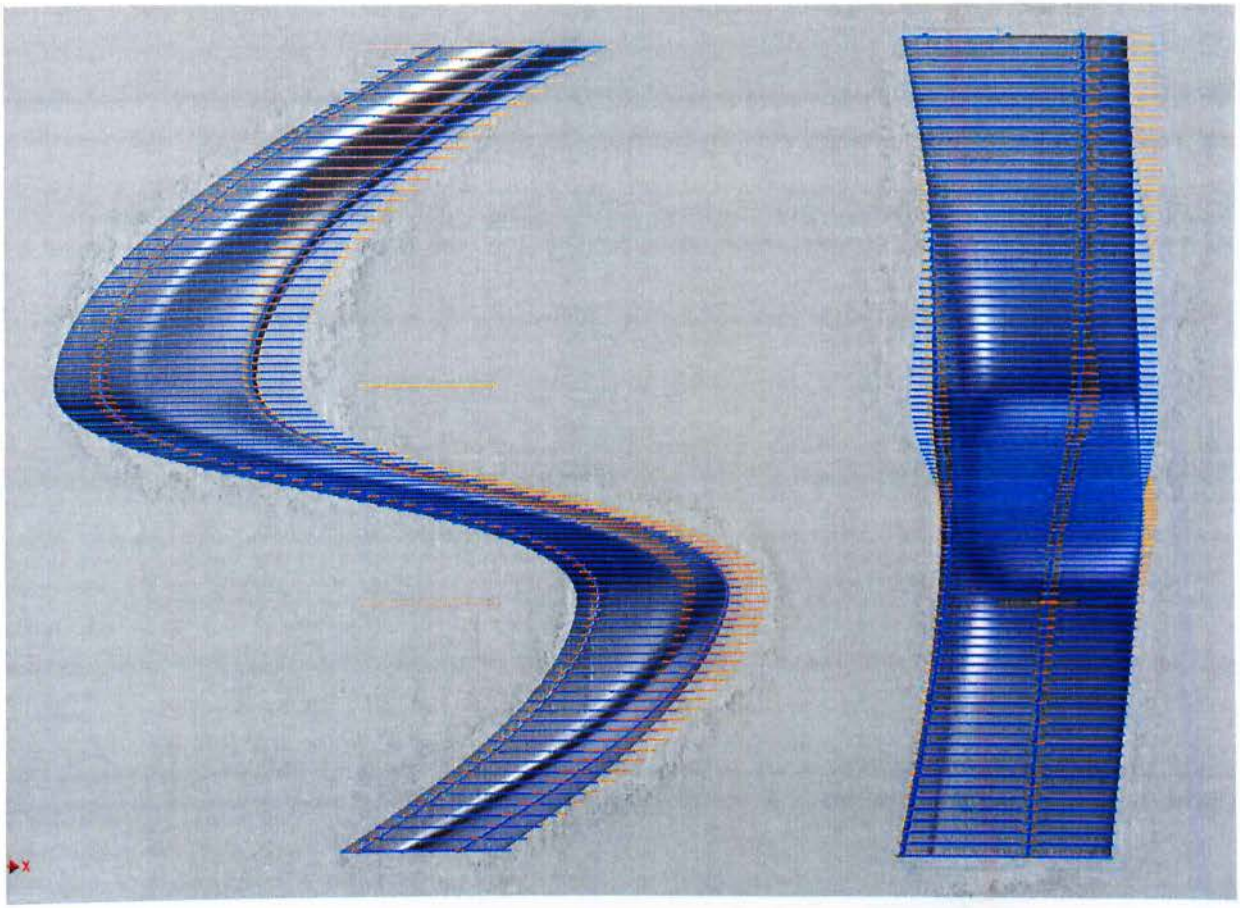
Δεν φτιάχνουμε spline στο u=1 διότι όταν το sideface είναι διπλωμένο όπως π.χ ένας κύκλος τότε το u=1 και το u=0 συμπίπτουν.Ακολουθεί παράδειγμα με εικόνες (βήμα – βήμα):

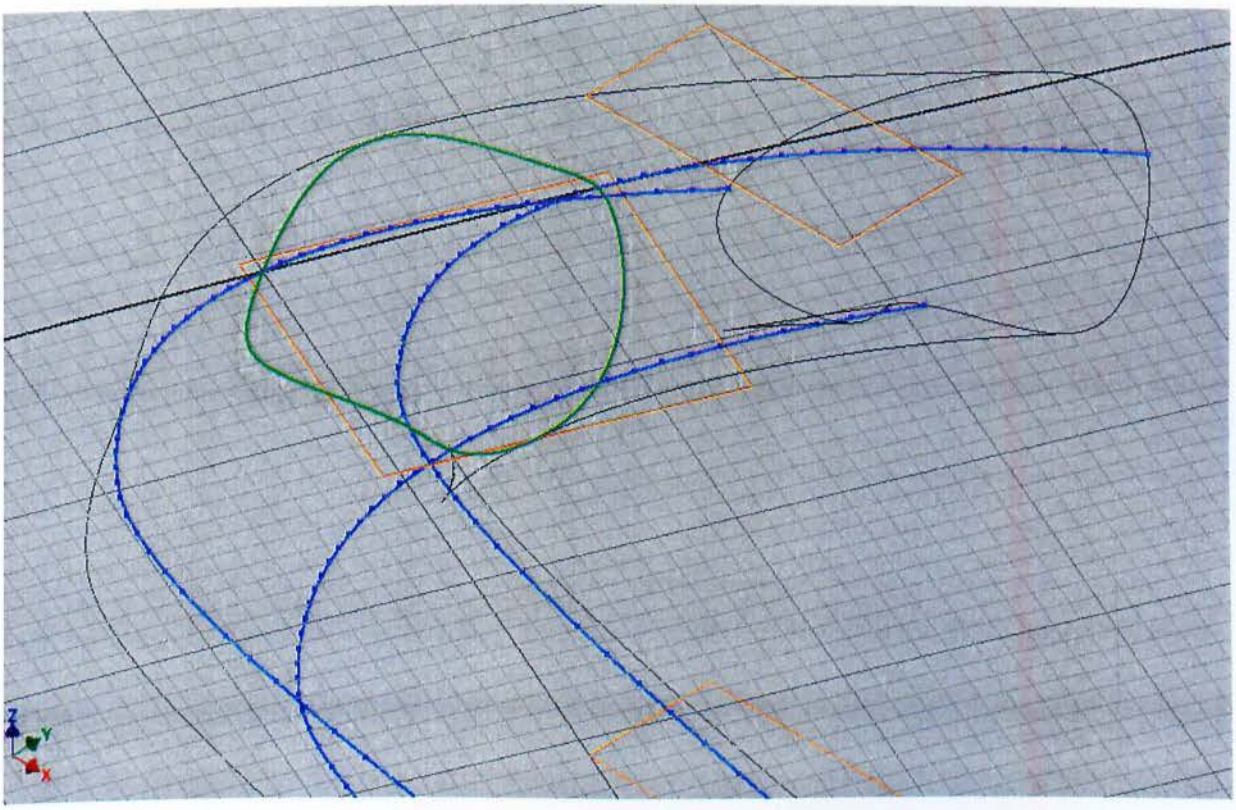


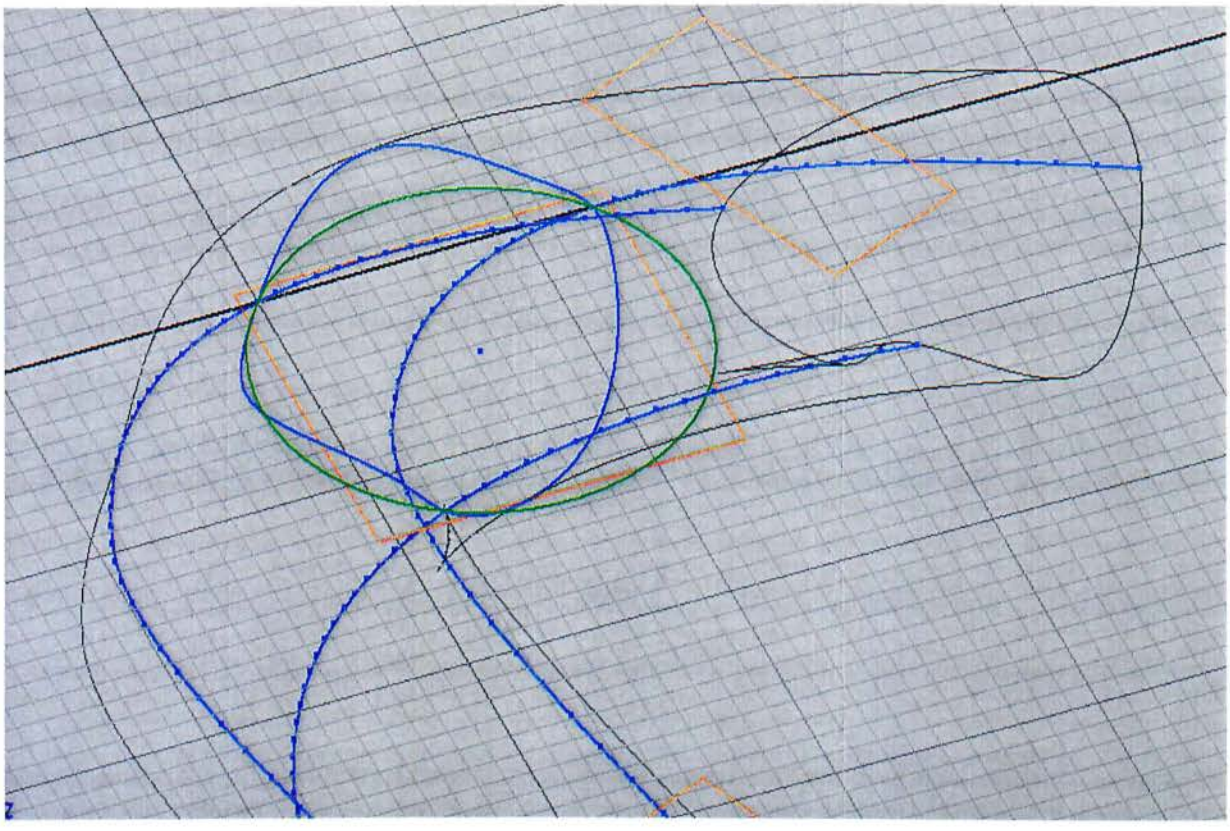


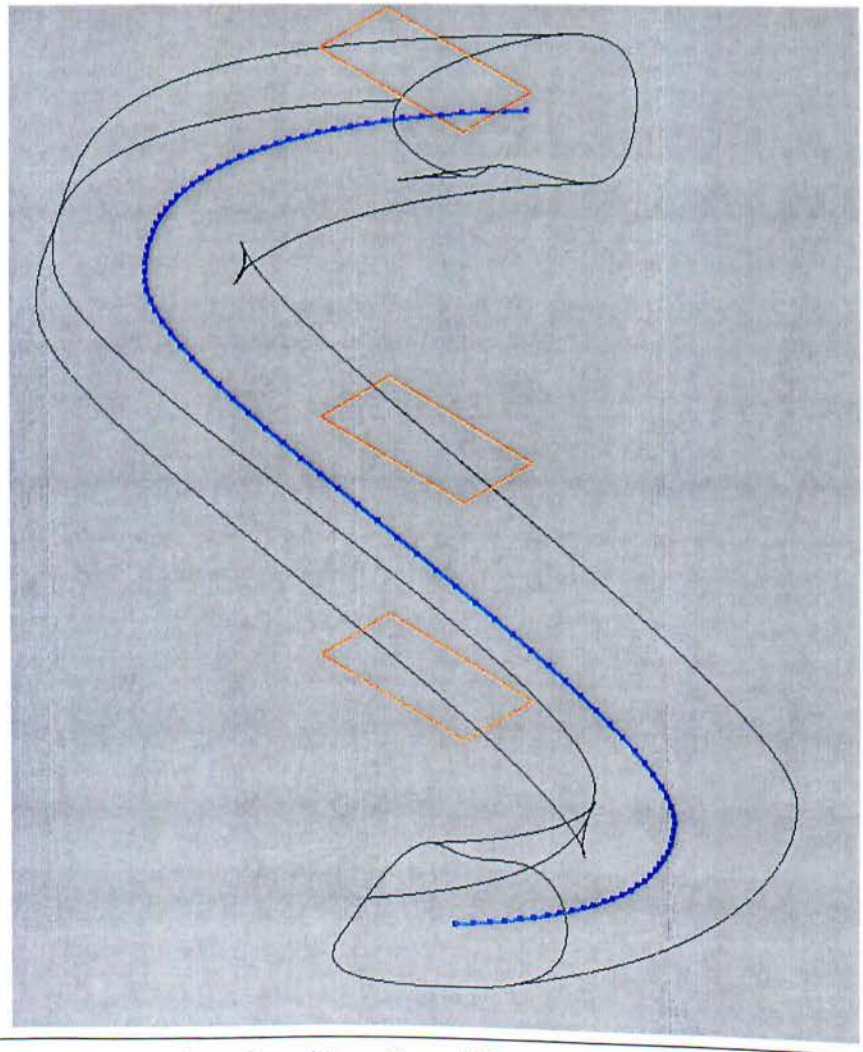


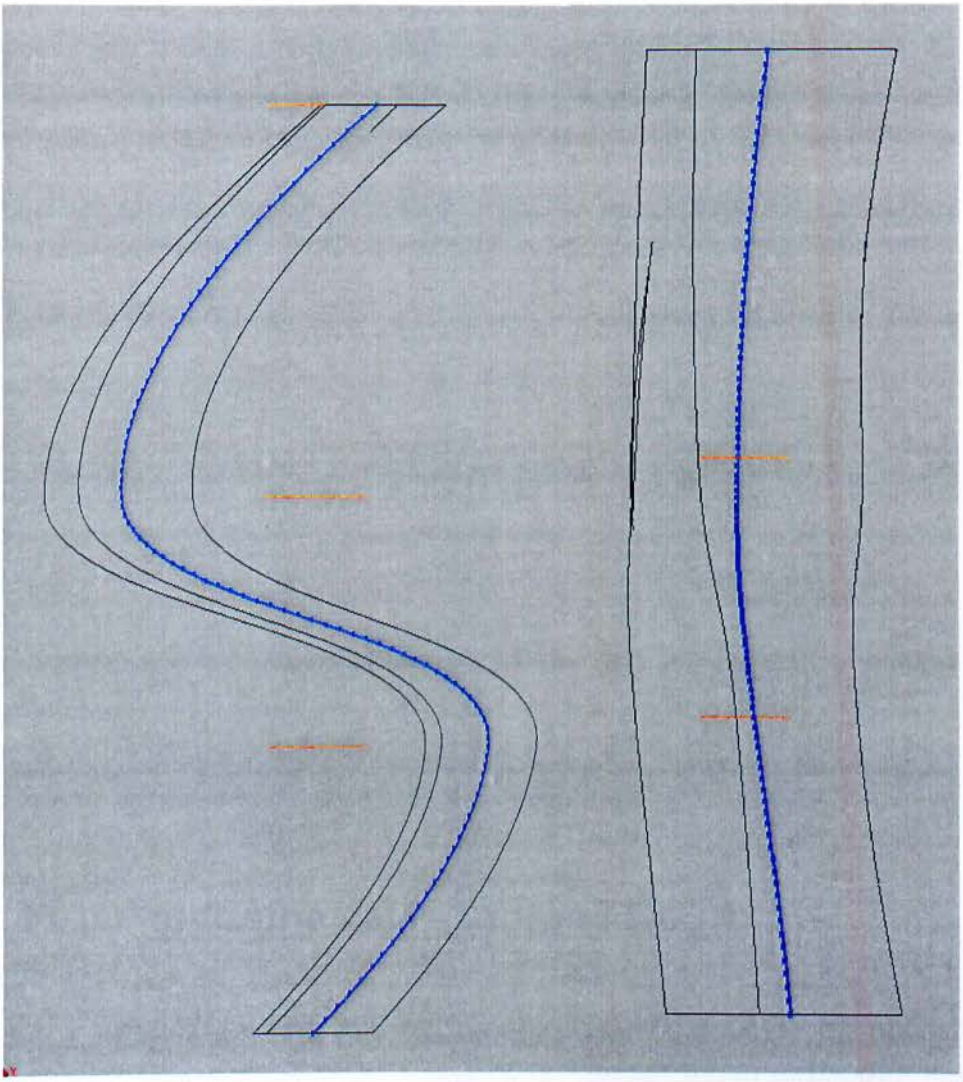






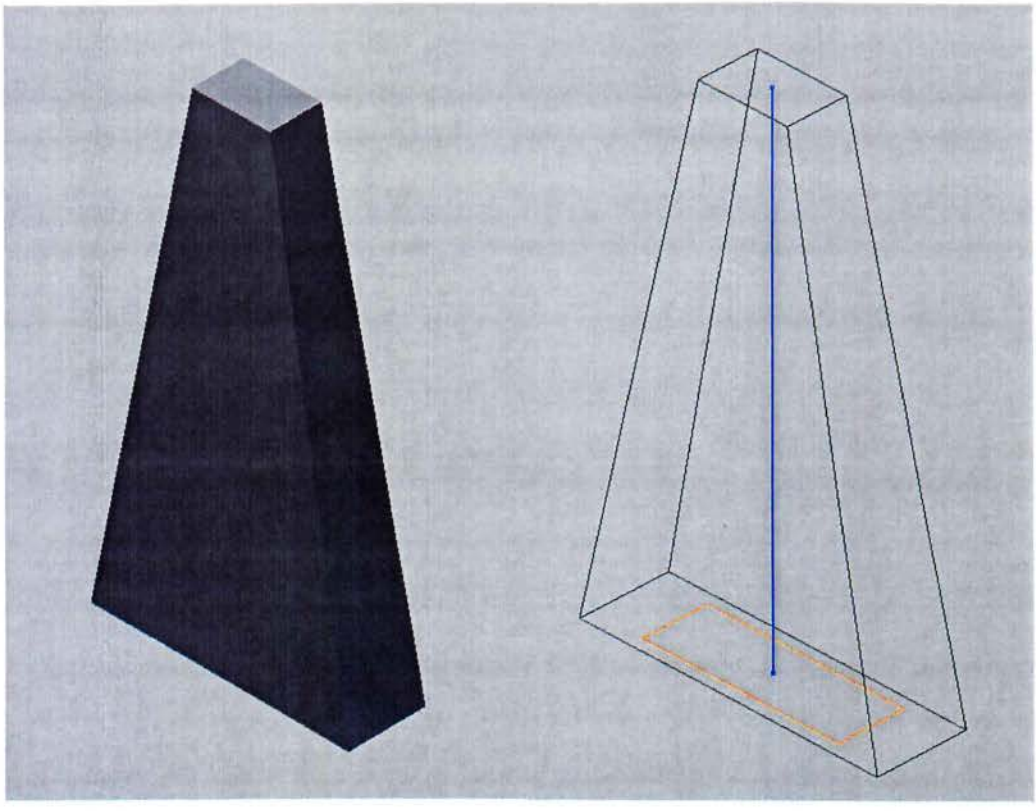






iii. Περίπτωση 2- Centerline

Αφορά μοντέλα που έχουν σχεδιαστεί με τέτοιο τρόπο ώστε τα sidefaces που δημιουργούνται δεν είναι τύπου BsplineSurface αλλά PlaneSurface. Σε αυτήν εδώ την περίπτωση ακολουθείται διαφορετική προσέγγιση, καθώς το **πρόβλημα** είναι ότι το SurfaceEvaluator δεν λειτουργεί στα PlaneSurfaces. Έτσι ορίζουμε μια γραμμή με τα centroids του πρώτου και του τελευταίου section που χρησιμοποίησε το loft feature για να δημιουργήσει το μοντέλο. Η περίπτωση αυτή είναι πολύ πιο απλή από τις δύο προηγούμενες, καθώς η δημιουργία του centerline γίνεται με δύο σημεία και τέλος δεν υπάρχει η πιθανότητα κάποιο WorkPlane να τέμνει σε παραπάνω από ένα σημεία το μοντέλο. Για τον λόγο αυτό η μετέπειτα επαναληπτική διαδικασία τομής του μοντέλου γίνεται ξεχωριστά από την κοινή διαδικασία που πραγματοποιείται για τα υπόλοιπα είδη loft feature, ώστε να μην επιβαρύνεται η επίδοση του προγράμματος με άσκοπους ελέγχους.



d)Loft Feature - Με centerline

Στην περίπτωση που έχουμε μοντέλο κατασκευασμένο με loft feature με χρήση centerline υπάρχουν οι παρακάτω περιπτώσεις:

1. Η centerline να είναι κατασκευασμένη απο 2D sketch και να είναι είτε Line2d είτε BSpline2d.

2. Η centerline να είναι κατασκευασμένη από 3D sketch και να είναι είτε Line είτε BSpline

Είναι είτε line είτε spline, διότι δεν γίνεται να είναι συνδυασμός τους καθώς δεν είναι ομαλή η μετάβαση από το ένα στο άλλο και γι' αυτό δεν το δέχεται το Inventror. Έτσι ο έλεγχος γίνεται διακριτά γι' αυτά τα δύο είδη.

e) Loft Feature – Επαναληπτική διαδικασία τομών

Σε όλες τις περιπτώσεις μοντέλων κατασκευασμένων με loft feature με μοναδική εξαίρεση την περίπτωση (2), Το εκάστοτε centerline που είτε υπάρχει ήδη είτε δημιουργείται με τις παραπάνω διαδικασίες, εφοδιάζεται με μία ταυτότητα. Η ταυτότητα αυτή παίζει σημαντικό ρόλο, διότι χρησιμεύει για να ελέγχει και να καθορίζει το είδος του Curve Evaluator που θα χρησιμοποιηθεί στην επαναληπτική διαδικασία. Το είδος του evaluator που χρησιμοποιείται είναι κοινό για κάποιες ομάδες centerline, οι ομάδες αυτές χωρίζονται ως εξής :

1. 3D spline centerlines : αυτή η ομάδα περιέχει τις centerlines απο τα μοντέλα που σχεδιάστηκαν χωρίς centerline με ένα (1) ή και παραπάνω side faces, καθώς και την centerline από μοντέλα που σχεδιάστηκαν με centerline τύπου Bspline3d.

2. 3D line centerline : περιέχει την περίπτωση μοντέλου σχεδιασμένου με centerline τύπου Line.

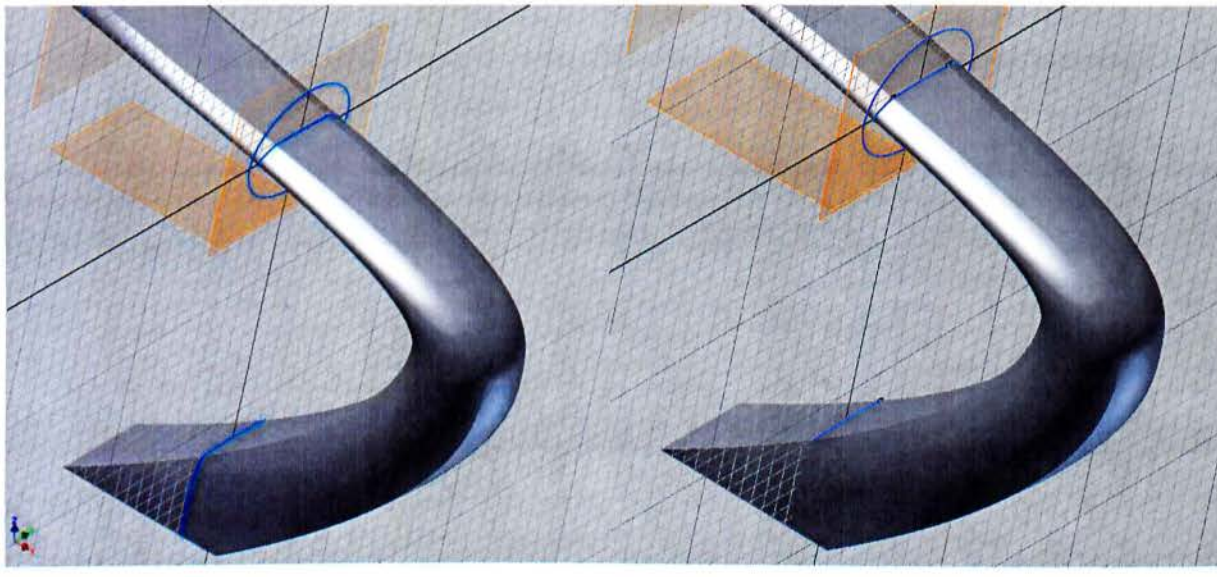
3. 2D line centerline : περιέχει την περίπτωση μοντέλου σχεδιασμένου με centerline τύπου Line2d.

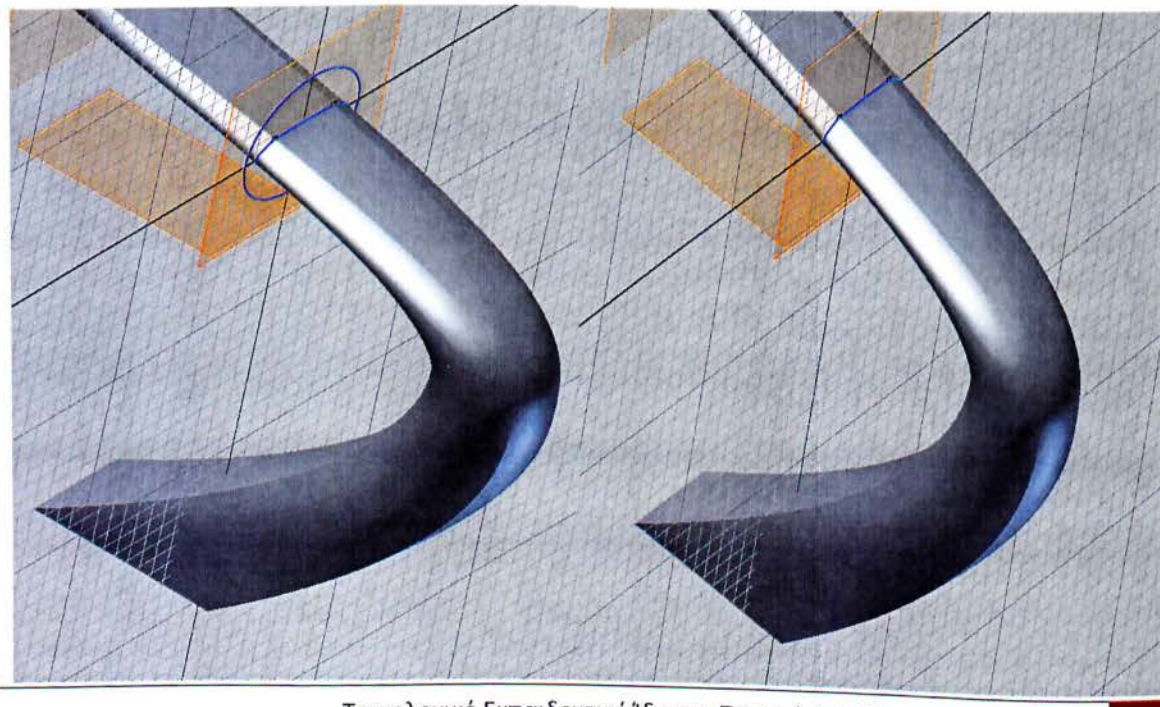
4. 2D spline centerline : περιέχει την περίπτωση μοντέλου σχεδιασμένου με centerline τύπου Bspline2d.

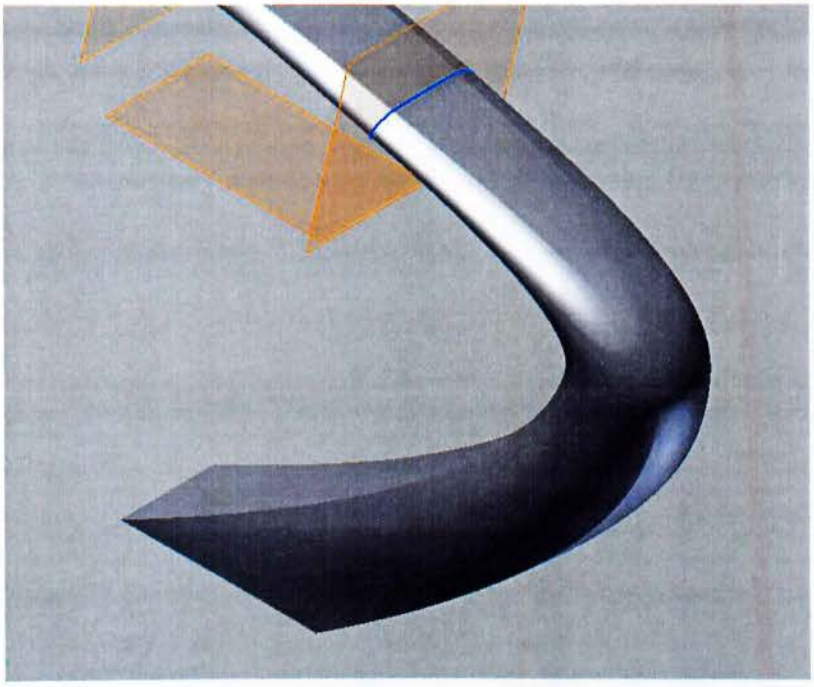
Έτσι αρχίζοντας η επαναληπτική διαδικασία ελέγχει την ταυτότητα της centerline και καταλήγει σε ποίο Evaluator θα χρησιμοποιήσει. Στην συνέχεια με τις λειτουργίες GetParamExtents και GetPointAtParam του Evaluator και αναλόγως με τον αριθμό των τομών που έχει εισάγει ο χρήστης το πρόγραμμα βρίσκει σημεία πάνω στην centerline. Με την βοήθεια της λειτουργίας των workplanes AddByNormalToCurve που χρησιμοποιεί ως δεδομένα τα σημεία αυτά και την ίδια την centerline το πρόγραμμα προσθέτει workplanes (επίπεδα) κάθετα στην centerline στο εκάστοτε σημείο. Σε κάθε επίπεδο δημιουργείται ένα sketch και με την εντολή ProjectCutEdges λαμβάνουμε τη διατομή/διατομές του μοντέλου στα σημεία που τέμνει το επίπεδο.

i. Ελεγχος-διαγραφή επιπλέον διατομών

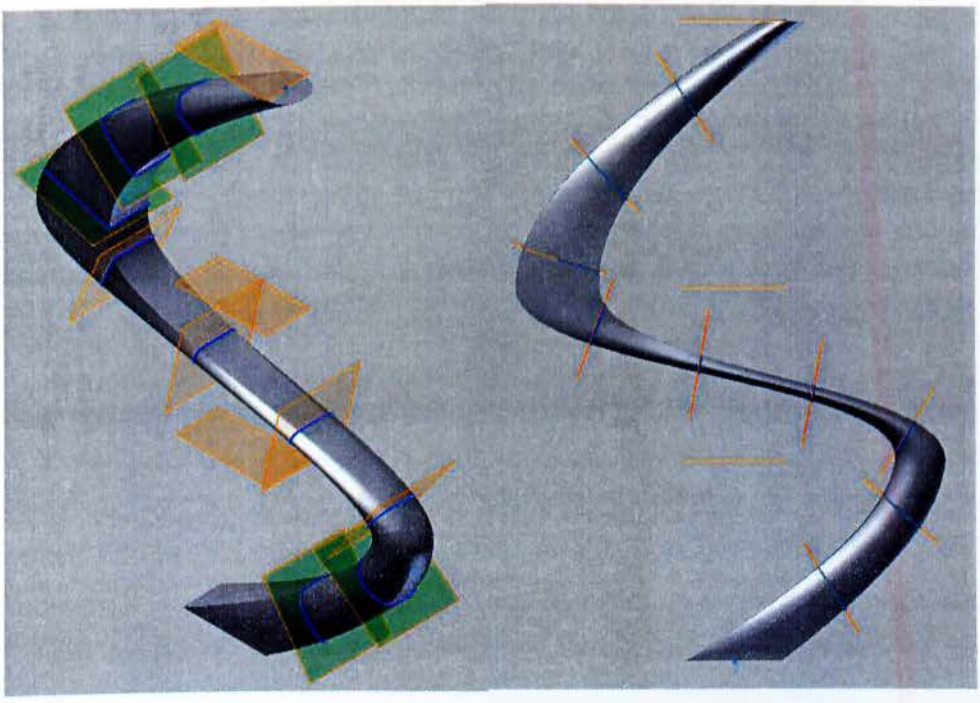
Επειδή όμως λόγο μεγάλης καμπυλότητας του μοντέλου το επίπεδο μπορεί να τέμνει το μοντέλο σε παραπάνω του ενός σημείου πραγματοποιείται η ίδια διαδικασία που αναφέρθηκε παραπάνω για την εύρεση της centerline. Με την διαφορά ότι η υποροutίνα εφόσον εντοπίσει το profile της «σωστής» διατομής και δημιουργήσει το rangebox του, χρησιμοποιεί το rangebox αυτό ως κριτήριο για το αν ανήκουν ή όχι στην διατομή αυτή όλα τα lines, splines και points του sketch. Ο έλεγχος γίνεται μόνο για αυτά τα τρία είδη entities επειδή σε αυτού του τύπου τα μοντέλα, η εντολή projectedcuts παράγει μόνο lines, splines και τα σημεία που τους ανήκουν. Στην περίπτωση που κάποιο από αυτά τα entities δεν ανήκει στην περιοχή του rangebox τότε διαγράφεται. Όταν η διαδικασία της διαγραφής έχει τελειώσει, έχει μείνει η «σωστή» διατομή που πρέπει να εξεταστεί. Ο λόγος είναι ότι το Β.Ε.Μ δεν μπορεί να επεξεργαστεί παραπάνω από μία διατομή την φορά.



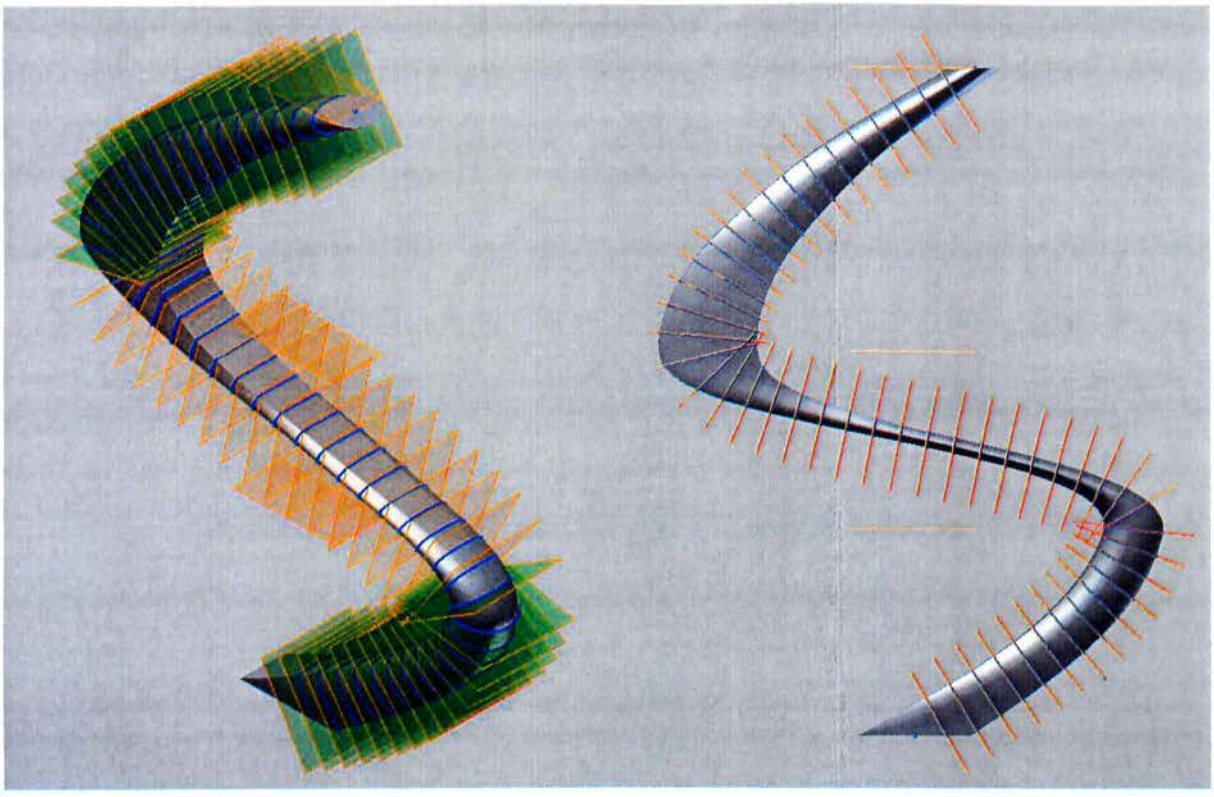




Μοντέλο με 10 τομές



Μοντέλο με 50 τομές



Παρατηρούμε στο τελευταίο σχήμα (με τις 50 τομές-δεξιά εικόνα) ότι οι τομές που έχουν γίνει στις βάσεις της δοκού δεν αντιστοιχούν με αυτές που πρέπει να εξεταστούν πραγματικά. Αυτό το **πρόβλημα** προκύπτει επειδή η centerline δημιουργήθηκε μετέπειτα της δοκού, καθώς υπό κανονικές συνθήκες αν κατασκευάζαμε πρώτα την centerline, το start face και το end face αναγκαστικά μέσω του inventor θα ήταν κάθετα στα σημεία που εφάπτονταν στην centerline (περίπτωση loft με centerline), ενώ σε σε αυτήν την περιπτωση δεν είναι. Με την παρουσία αυτου του προβλήματος τα αρχικά και τα τελικά αποτελέσαματα που προκύπτουν για την δοκό είναι λανθασμένα. Έγινε μια προσπάθεια επίλυσης, με μια μέθοδο σύγκρισης του επιπέδου των start και end faces με την γωνία του επιπέδου της κάθε τομής, καθώς και πια άλλα επίπεδα τέμνει αλλά απέτυχε, λόγω έλειψης objects σύγκρισης των planes και workplanes. Είναι μια παραδοχή που γίνεται, αφού η ευρέση λύσης για αυτό το πρόβλημα αποτελεί ένα κεφάλαιο μόνη της και δεν αφορά την εργασία αυτή.

ii. Thin extrude

Το πρόγραμμα Β.Ε.Μ είναι σχεδιασμένο να έχει ως δεδομένα εισόδου τα edges από loft και extrude features. Καθώς ως έξοδο από το δικό μου πρόγραμμα ήταν ένα sketch έπρεπε να βρεθεί ένας αξιόπιστος και αποτελεσματικός τρόπος σύνδεσης μεταξύ των δύο προγραμμάτων. Επίσης μία ακόμα απαίτηση ήταν, η αποθήκευση ξεχωριστά κάθε διατομής και των σχεδιαγραμμάτων που θα δημιουργούνταν πάνω σε αυτήν. Η λύση προήλθε με την δημιουργία ενός πολύ λεπτού extrude προερχόμενο από το sketch της εκάστοτε διατομής. Το sketch της διατομής αντιγράφεται στην μνήμη, ύστερα το πρόγραμμα δημιουργεί ένα νέο Part Document και επικολλά το sketch εκεί. Από το sketch δημιουργεί ένα profile, με το οποίο φτιάχνει το thin extrude (extrude πολύ μικρού πάχους) το οποίο και χρησιμοποιείται τελικά ως είσοδος για το πρόγραμμα Β.Ε.Μ.

iii. B.E.M

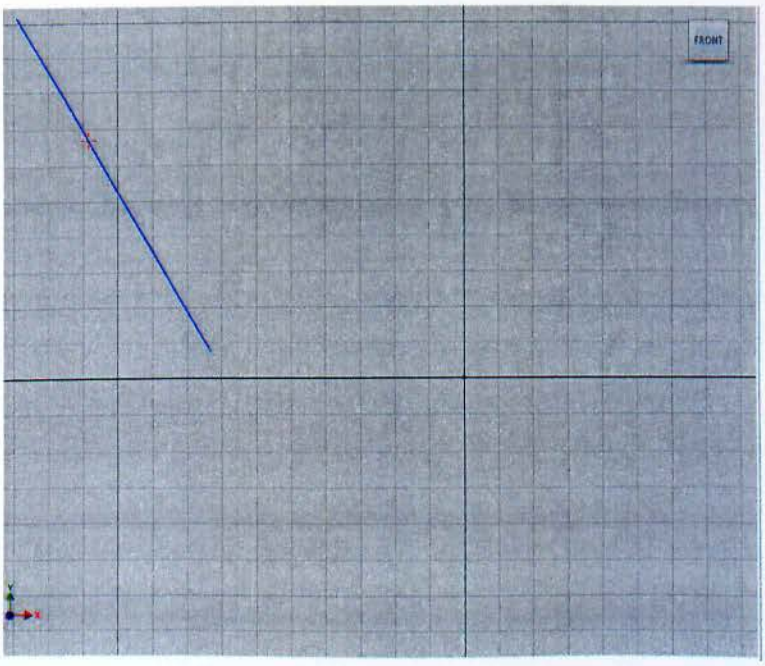
Ως είσοδο λοιπόν το πρόγραμμα B.E.M λαμβάνει το Part Document που περιέχει το thinextrude, το thinextrude και τις τιμές για τις ροπές και το είδος των τάσεων που θα απεικονίζονται στο σχέδιο, που έχει εισάγει ο χρήστης. Ως έξοδο έχει arrays (συστοιχίες αριθμών) χιλίων (1000) θέσεων με τις τιμές των X, των Y μεταβλητών, καθώς και τις τιμές των τάσεων Sv, St, St.

iv. 2D – 3D Διαγράμματα

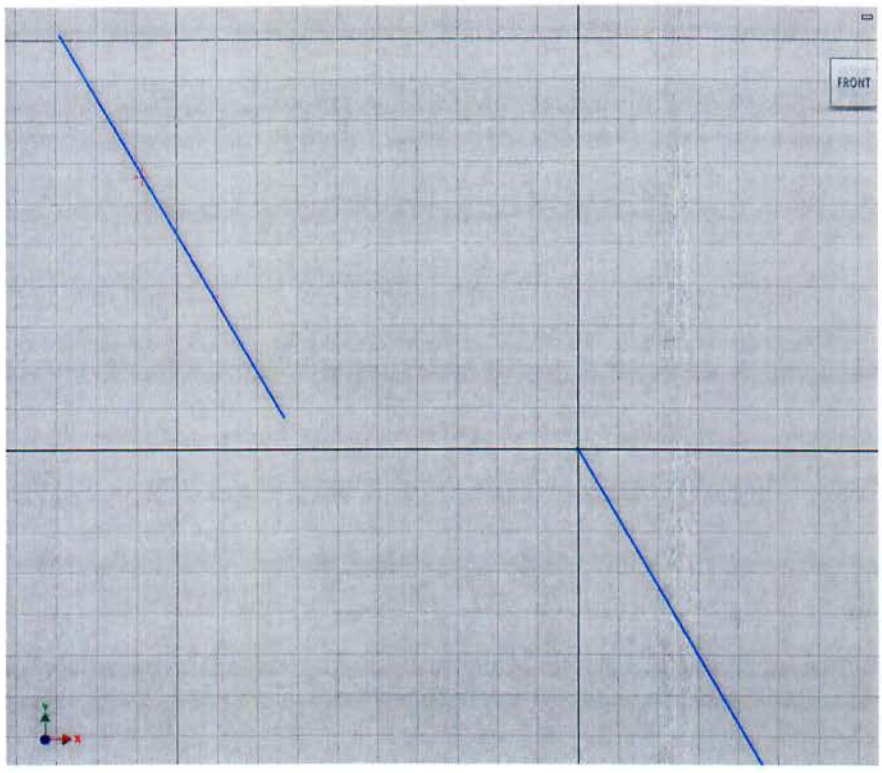
Η σχεδίαση των διαγραμμάτων γίνεται από την υπορουτίνα με το όνομα metafora. Η υπορουτίνα αυτή χρησιμοποιεί επαναληπτική μέθοδο. Σε κάθε επανάληψη δημιουργεί ένα sketch point από τις εκάστοτε τιμές των arrays των X και Y, ώστε να υπάρξει μια απτή αναφορά του σημείου αυτού πάνω στο sketch. Ύστερα σαρώνει όλα τα sketch entities (μόνο τα sketchlines, sketchsplines, sketchcircles) εξετάζοντας εάν το σημείο περιέχεται μέσα στο rangebox κάποιου entity. Όταν εντοπιστεί μία τέτοια περίπτωση τότε προσθέτει ένα geometric constraint (γεωμετρικό περιορισμό) μεταξύ του sketch point και του entity, με την εντολή AddCoincident η οποία στην ουσία τα κάνει να συμπίπτουν. Αυτή η διαδικασία γίνεται διότι το inventor έχει **πρόβλημα** με την ακρίβεια πολλών δεκαδικών. Εισάγοντας τα σημεία που είχε υπολογίσει το B.E.M είχε ως αποτέλεσμα τα σημεία να φαίνονται οτι συμπίπτουν (γραφικά) π.χ με μια γραμμή αλλά υπολογιστικά το inventor να μην το λαμβάνει υπόψιν του. Έτσι ήταν αδύνατη η μετέπειτα διαδικασία της δημιουργίας των διαγραμμάτων που ακολουθεί.

Όταν πλέον το σημείο είχε μία αναφορά σε κάποιο sketch entity τότε με την βοήθεια του evaluator του και την εντολή GetTangent βρίσκει την εφαπτομένη του entity στο συγκεκριμένο σημείο. Ακολουθεί ένα γραφικό παράδειγμα της διαδικασίας.

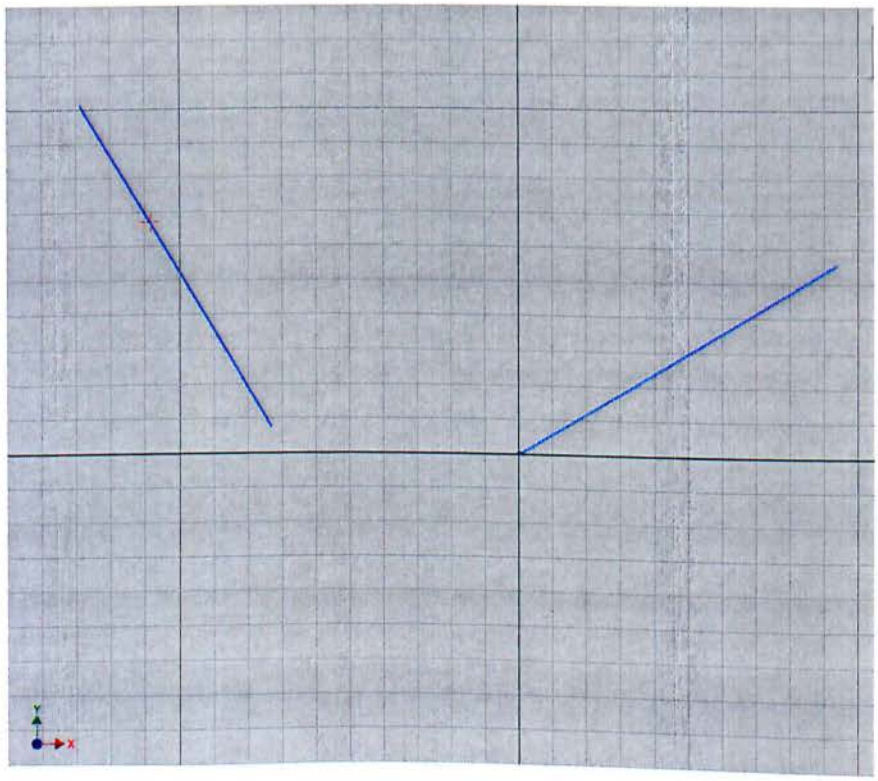
Εδώ φαίνεται μία γραμμή και πάνω της ένα σημείο.



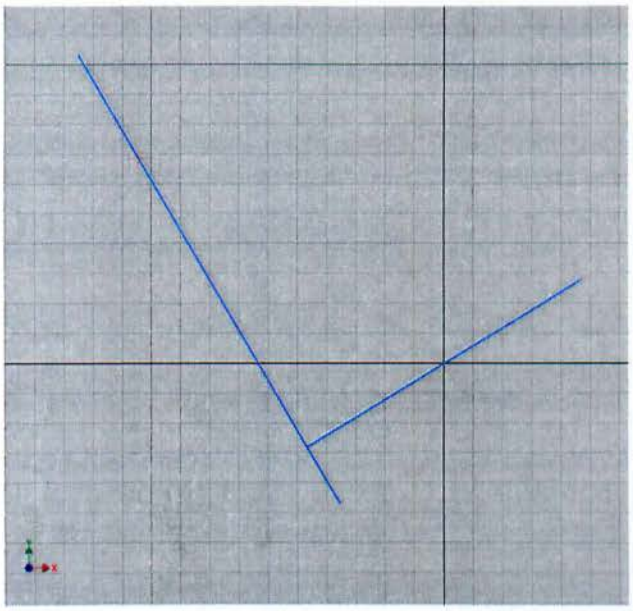
Εδώ διακρίνουμε το διάνυσμα που δημιουργήθηκε και αντιπροσωπεύει την εφαπτομένη της γραμμής στο κόκκινο σημείο.



Στην συνέχεια η υπορουτίνα με την βοήθεια ενός matrix προγραμματισμένου να κάνει περιστροφή 90 μοιρών ως προς το κέντρο της αρχής των αξόνων, περιστρέφει το διάνυσμα ώστε να είναι πλέον κάθετο ως προς την γραμμή στο συγκεκριμένο σημείο.

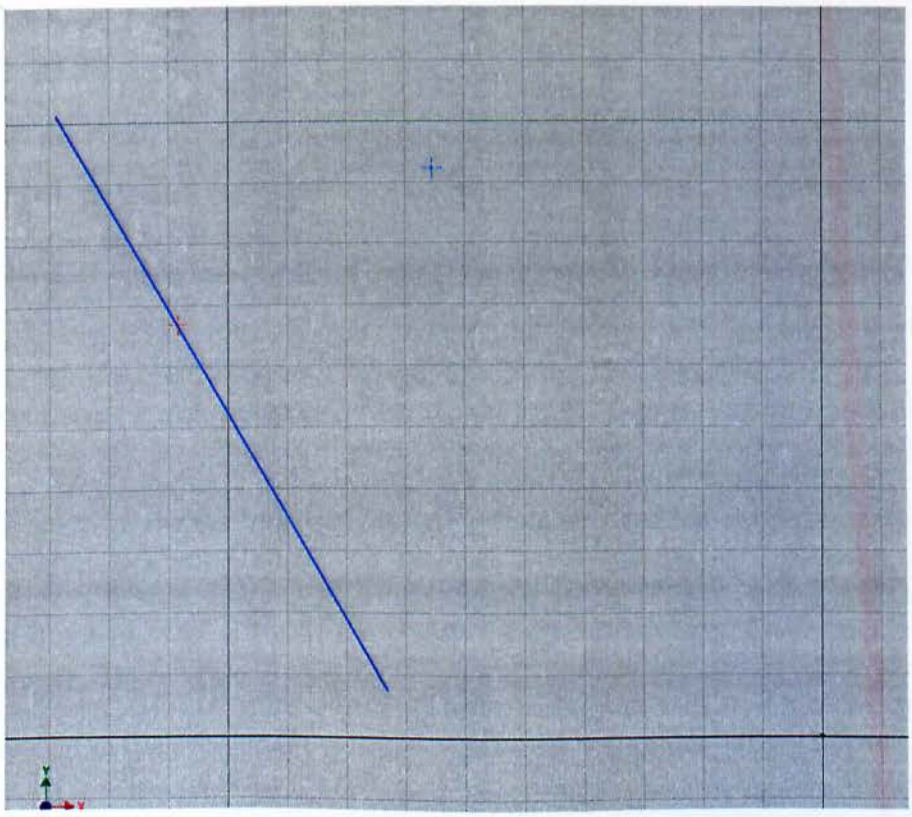


Αυτό φαίνεται πιο εύκολα αν προεκτείνουμε τις δύο ευθείες.

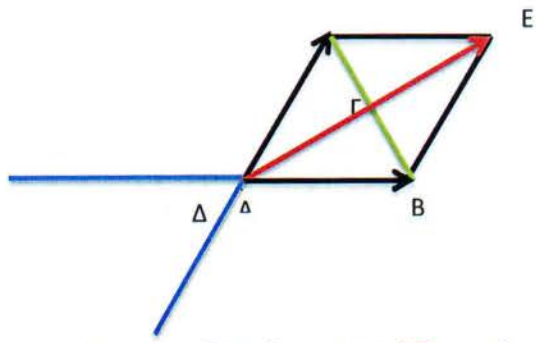


Παρατηρούμε ότι η κάθετη ευθεία που αντιπροσωπευεί το διάνυσμα δεν συμπίπτει με το σημείο. Αυτό γίνεται επειδή η υποροutίνα χρησιμοποιεί το διάνυσμα για να ορίσει μια κατεύθυνση και ένα μέτρο και γι' αυτό δεν έχει σημασία απο που ξεκινάει το διάνυσμα.

Ύστερα το μέτρο του διανύσματος με την εντολή normalize γίνεται μονάδα, ώστε στην συνέχεια πολλαπλασιάζοντας με την τιμή της τάσης που έχει επιλέξει ο χρήστης (Sv, Sb, St) από το αντίστοιχο array το διάνυσμα παίρνει το τελικό του μέτρο. Τέλος με την εντολή TranslateBy του σημείου η αλλαγή εφαρμόζεται ώστε να φτιαχτεί ένα νέο σημείο με αυτές τις αλλαγές.



Στην περίπτωση που το σημείο βρίσκεται πάνω σε γωνία τότε ανήκει σε δύο γραμμές, έτσι δημιουργούνται δύο διανύσματα τα οποία αυτόματα συνδυάζονται σε ένα (πρόσθεση διανυσμάτων). Το μέτρο αυτού το διανύσματος για να είναι μονάδα πρέπει να προσαρμοστούν τα μέτρα των δύο διανυσμάτων. Έχοντας ως δεδομένο ότι τα δύο διανίσματα έχουνε το ίδιο μέτρο και γνωρίζοντας την γωνία τους.
Έχουμε:



Εφόσον θέλουμε το μέτρο του διανύσματος ΑΕ να είναι μονάδα (ΑΕ=1) χρησιμοποιούμε το τρίγωνο ΑΒΓ για να βρούμε την πλευρά ΑΓ η οποία είναι το μισό της ζητούμενης ΑΕ. Η γωνία Α είναι το μισό της γνωστής γωνίας Δ. Έτσι χρησιμοποιούμε την παρακάτω σχέση :

$$A\Gamma = \frac{AE}{2}$$

$$\cos(A) = \frac{A\Gamma}{AB} \Rightarrow \cos(A) = \frac{AE}{2 \cdot AB} \Rightarrow AE = 2 \cdot AB \cdot \cos(A) \Rightarrow AB = \frac{1}{2 \cdot \cos(A)}$$

Έτσι με αυτόν τον τύπο η υπορουτίνα μεταβάλει το μέτρο των δύο διανυσμάτων ώστε το τελικό διάνυσμα να έχει μέτρο την μονάδα και να μπορεί με ακρίβεια να πολλαπλασιαστεί με την επιλεγμένη τάση.

Ένα **πρόβλημα,** είναι περίπτωση που υπάρχουν δύο γραμμές και είναι συνδυασμός line με spline είτε spline με spline τότε η μέθοδος getangle (για object) δεν μπορεί να εφαρμοστεί διότ δεν μπορεί να υπολογίσει την γωνία μεταξύ ευθείας και καμπύλης είτε γωνία μεταξύ καμπυλών. Έτσι για να υπάρχει μια λύση για όλες τις περιπτώσεις, μια υπορουτίνα βρίσκει τις εφαπτόμενες των δύο entities (lines ή splines) σε εκείνο το σημείο και διμιουργεί τα αντίστοιχα διανύσματα (vectors) που τις αντιπροσωπεύουν (ίδια διαδικασία που εφαρμόζεται παραπάνω) και με την εντολη getangle (για vectors) βρήσκουμε την γωνία.

Τα τελικά σημεια της διατομής αποθηκεύονται σε ένα collection, επιστρέφοντας από το document του thin extrude στο κύριο document που βρίσκεται το sketch της τομής, μεταφέρεται από το δισδιάστατο επίπεδο του sketch στο τρισδιάστατο επίπεδο, με την εντολή sketchTomodelspace του sketch. Ύστερα με μια επαναληπτική διαδικασία τα σημεία αυτά γίνονται 3D sketch points σε ένα 3d sketch που έχει δημιουργηθεί πρίν αρχίσουν οι τομές. Τα σημεία αυτά απεικονίζουν την κατανομή των τάσεων σε κάθε διατομή. Σε συνεργασία με μία εσωτερική υπορουτίνα του Β.Ε.Μ το πρόγραμμα βρίσκει τις μέγιστες τιμές στα arrays των Sv, St και Sb. Έτσι το σημείο με την μέγιστη τιμή αποθηκεύεται και αυτό σε ένα άλλο collection το οποίο μετά το τέλος της διαδικασίας των τομών δημιουργεί ένα 3D spline στο ίδιο 3D sketch που απεικονίζει τις μέγιστες τιμές κατά μήκος της δοκού.

v. Ίσιο Loft Feature

Η διαδικασία τομών του loft feature με ίσιες επιφάνειες είναι ακριβώς ή ίδια με την διαδικασία που περιγράφηκε για τα υπόλοιπα loft features. Η μόνη διαφορά είναι ότι επειδή δεν υπάρχει πιθανότητα κάποιο WorkPlane να τέμνει σε παραπάνω από ένα σημεία το μοντέλο δεν πραγματοποιείται κανένας έλεγχος. Για τον λόγο αυτό η μετέπειτα επαναληπτική διαδικασία τομής του μοντέλου γίνεται ξεχωριστά από την κοινή διαδικασία που πραγματοποιείται για τα υπόλοιπα είδη loft feature, ώστε να μην επιβαρύνεται η επίδοση του προγράμματος με άσκοπους ελέγχους.

f) Extrude Feature

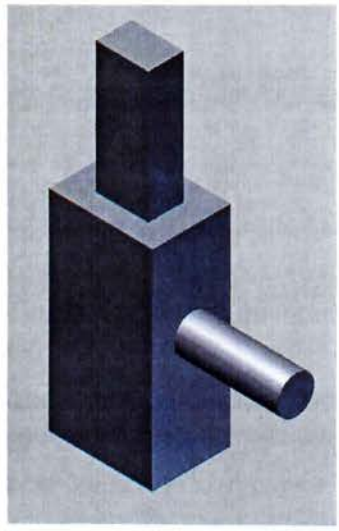
Αρχικά, στην περίπτωση του extrude feature δεν χρησιμοποιείται κάποιο centerline όπως στην περίπτωση των loft features. Η βασική ιδέα είναι ότι η υπορουτίνα ξεκινά να «κόβει» το μοντέλο ορίζοντας τα work planes με την εντολή AddByPlaneAndOffset. Αυτή η εντολή απαιτεί ένα επίπεδο αναφοράς και ένα offset (μετατόπιση). Το επίπεδο αναφοράς που χρησιμοποιούμε είναι το end face διότι αν υπάρχουν παραπάνω από ένα extrude τότε τα start faces που βρίσκονται ανάμεσα στα δύο ή παραπάνω extrude δεν υφίστανται πλέον (λόγο ότι πλέον το μοντέλο θεωρείται ένα σώμα έτσι έχει μια αρχή και ένα τέλος). Ένας άλλος λόγος που χρησιμοποιούμε το end face είναι ότι η μέθοδος αυτή είναι πιο εύκολη και ασφαλής σε θέματα σφαλμάτων όταν το μοντέλο αποτελείται από πολλά extrude. Το offset ορίζεται με βάση το μήκος του μοντέλου και των τομών που πρέπει να γίνουν. Και αυξάνεται μέσω της επαναληπτικής διαδικασίας με σταθερό βήμα ανάλογο τον παραπάνω δεδομένων .

i. Ειδικός έλεγχος για extrude feature

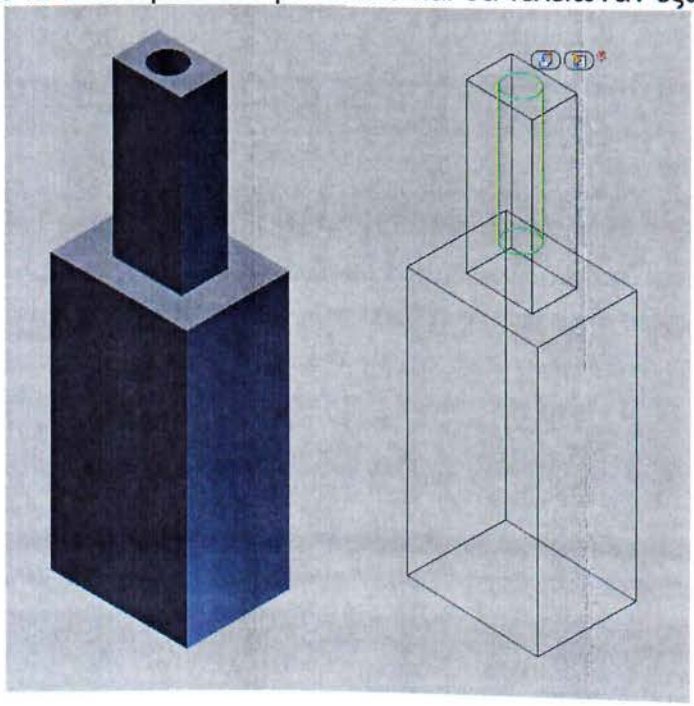
Ο ειδικός έλεγχος πριν αρχίσει η διαδικασία τομών αποτελείται απο τρια βήματα:

1. Αρχικά ελέγχεται αν το μοντέλο έχει δημιουργηθεί με την μέθοδο intersect του extrude feature, το οποίο περιπλέκει την κατάσταση. Ο λόγος είναι ότι η μέθοδος αυτή δημιουργεί έναν όγκο από το συνδυασμό ενος επιλεγμένου profile και ενός ήδη υπάρχοντος μοντέλου, ότι «υλικό» βρίσκεται εκτός το ορισμένου όγκου αφαιρείται. Το θέμα όμως είναι ότι το τελικό μοντέλο που μένει δεν έχει ούτε start face ούτε end face, επειδή όπως αναφέρεται και στην εισαγωγή, το inventor στην ουσία δεν γνωρίζει σε ποίο σημείο μέσα σε αυτόν τον όγκο αρχίζει να υπάρχει υλικό και σε ποίο σταματάει να υπάρχει, έτσι το solid αυτό μένει ακαθόριστο. Έτσι στην περίπτωση που έχει εφαρμοστεί η μέθοδος του intersect εμφανίζεται μήνυμα σφάλματος και το πρόγραμμα σταματάει.

2. Στην συνέχεια, στην περίπτωση που υπάρχουν περισσότερα από ένα extrude ελέγχεται αν έχουν την ίδια διεύθυνση ώστε το σχήμα της δοκού να μην έχει ακαθόριστο σχήμα. Με αποτέλεσμα να μην ξεκινήσει την διαδικασία τομών το πρόγραμμα από λάθος face. Ο έλεγχος γίνεται με την σύγκριση του πρώτου extrude με τα υπόλοιπα που έγιναν. Στη παρακάτω εικόνα παρατηρείται ότι η κυκλική διατομή που προεξέχει στα πλάγια δεν έχει τη ίδια διεύθυνση με το extrude τις βάσης το οποίο κατασκευάστηκε πρώτο. Σε αυτό το μοντέλο το πρόγραμμα θα εμφανίσει μήνυμα σφάλματος για να αλλάξει τον τρόπο σχεδίασης ο χρήστης.

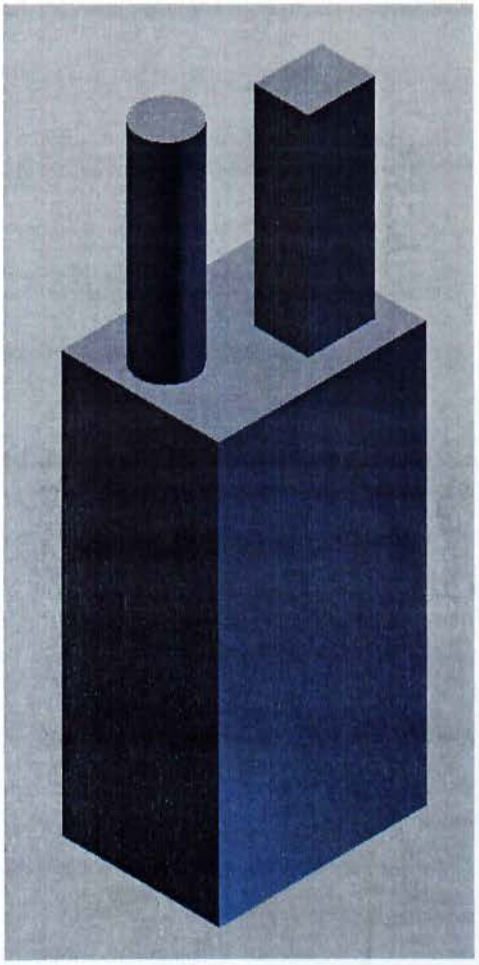


3. Τέλος γίνεται έλεγχος για την περίπτωση που το τελευταίο extrude έγινε με την μέθοδο cut, καθώς το end face θα βρίσκεται μέσα στην δοκό και θα υπάρχει σφάλμα. Παρατηρούμε την κυκλική διατομή (το τελευταίο extrude που έγινε) με φορά προς το εσωτερικό του μοντέλου με την μέθοδο cut. Χωρίς αυτόν τον έλεγχο η τομές θα άρχιζαν από το εσωτερικό του μοντέλου και θα τελείωναν έξω από αυτό.



ii. Extrude feature – έλεγχος διατομής

Ο έλεγχος διατομή γίνεται ακριβός με τον ίδιο τρόπο με τον έλεγχο που πραγματοποιείται για την εύρεση του centerline σε loft features χωρίς centerline. Η μόνη διαφορά είναι ότι όταν η υπορουτίνα ανακαλύψει την ύπαρξη παραπάνω του ενός profile τότε το πρόγραμμα σταματάει και εμφανίζεται μήνυμα σφάλματος. Ένα παράδειγμα φαίνεται στην παρακάτω εικόνα.

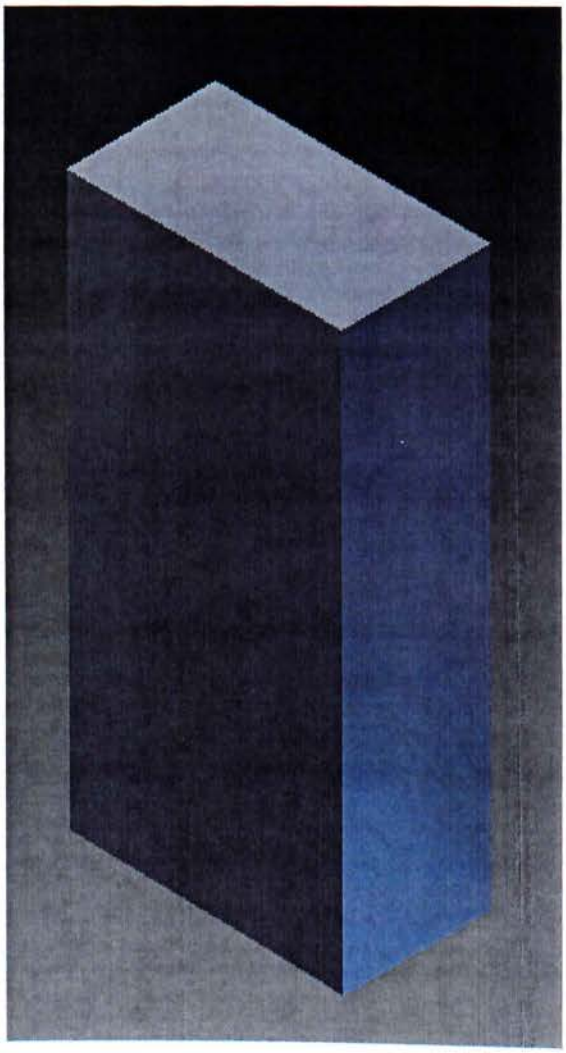


5)Παραδείγματα

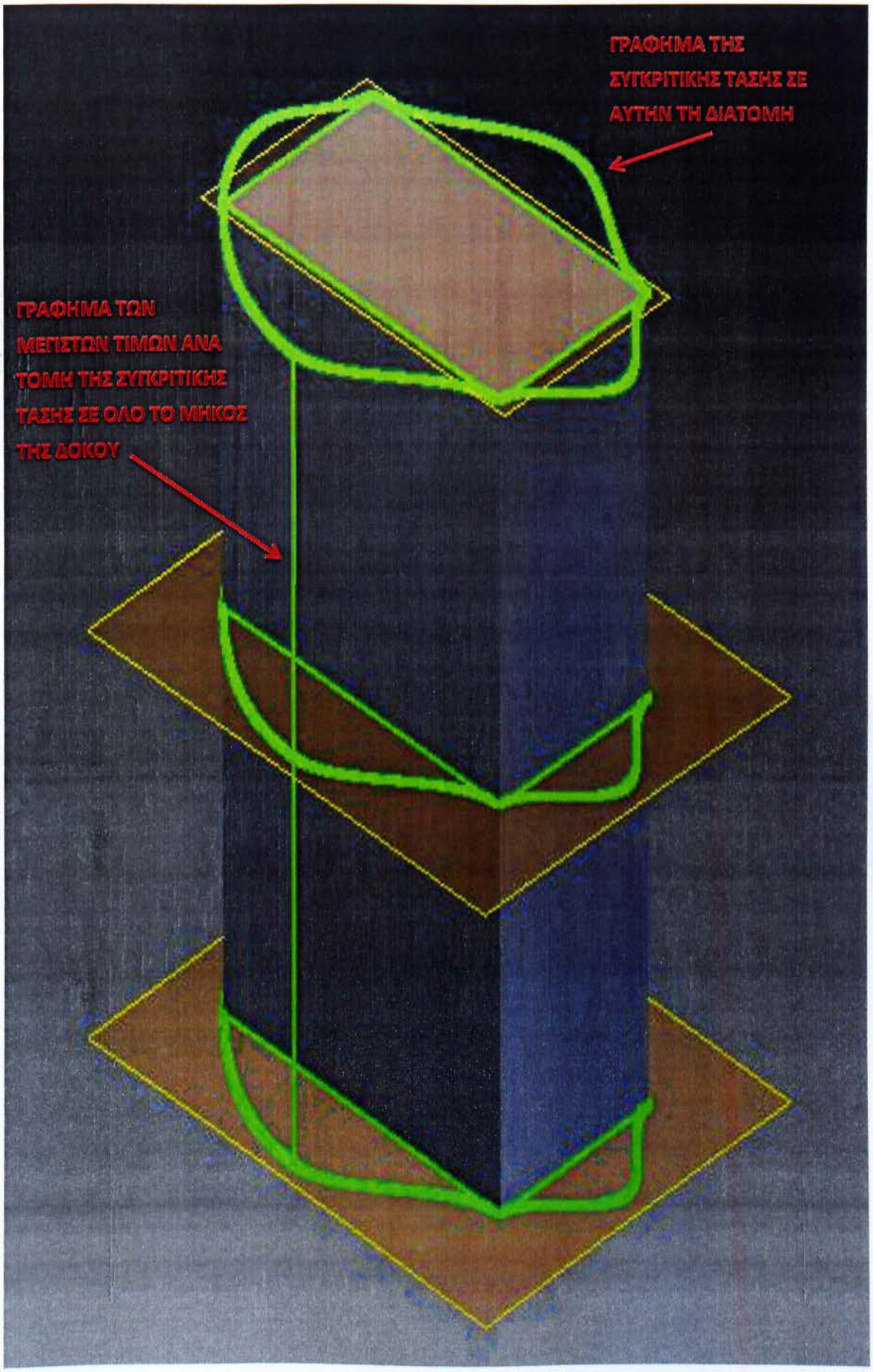
ΠΑΡΑΤΗΡΗΣΗ:

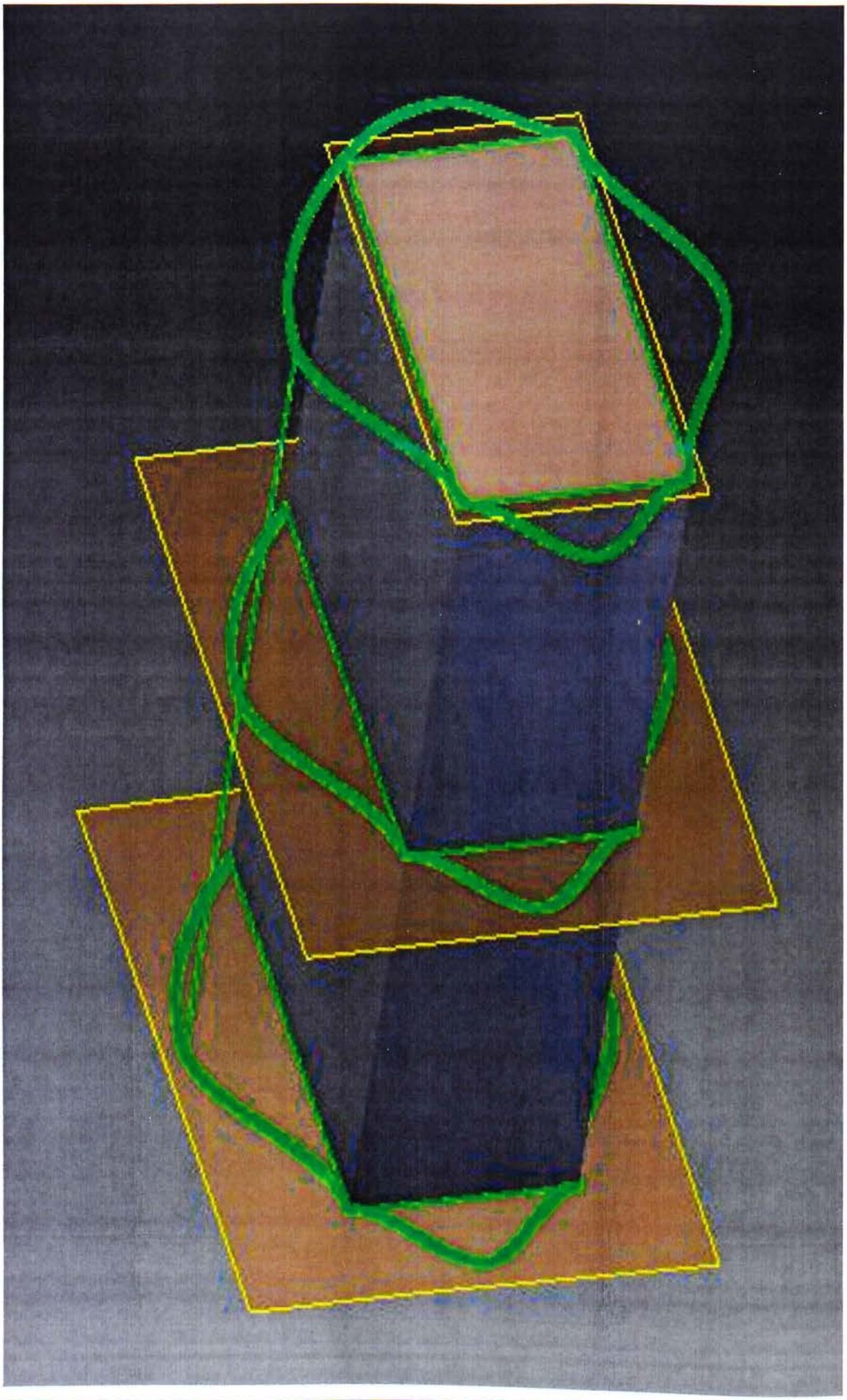
Λόγω του ότι το πρόγραμμα είναι πολύ βαρύ στα παραδείγματα αποφύγαμε την χρήση splines καθώς και πιο περίπλοκες μορφές διατομών, τα παρακάτω παραδείγματα προέρχονται μονάχα απο extrude feature. Η ελάφρυνση του προγράμματος μπορεί να αποτελέσει την βάση για μια νέα πτυχιακή.

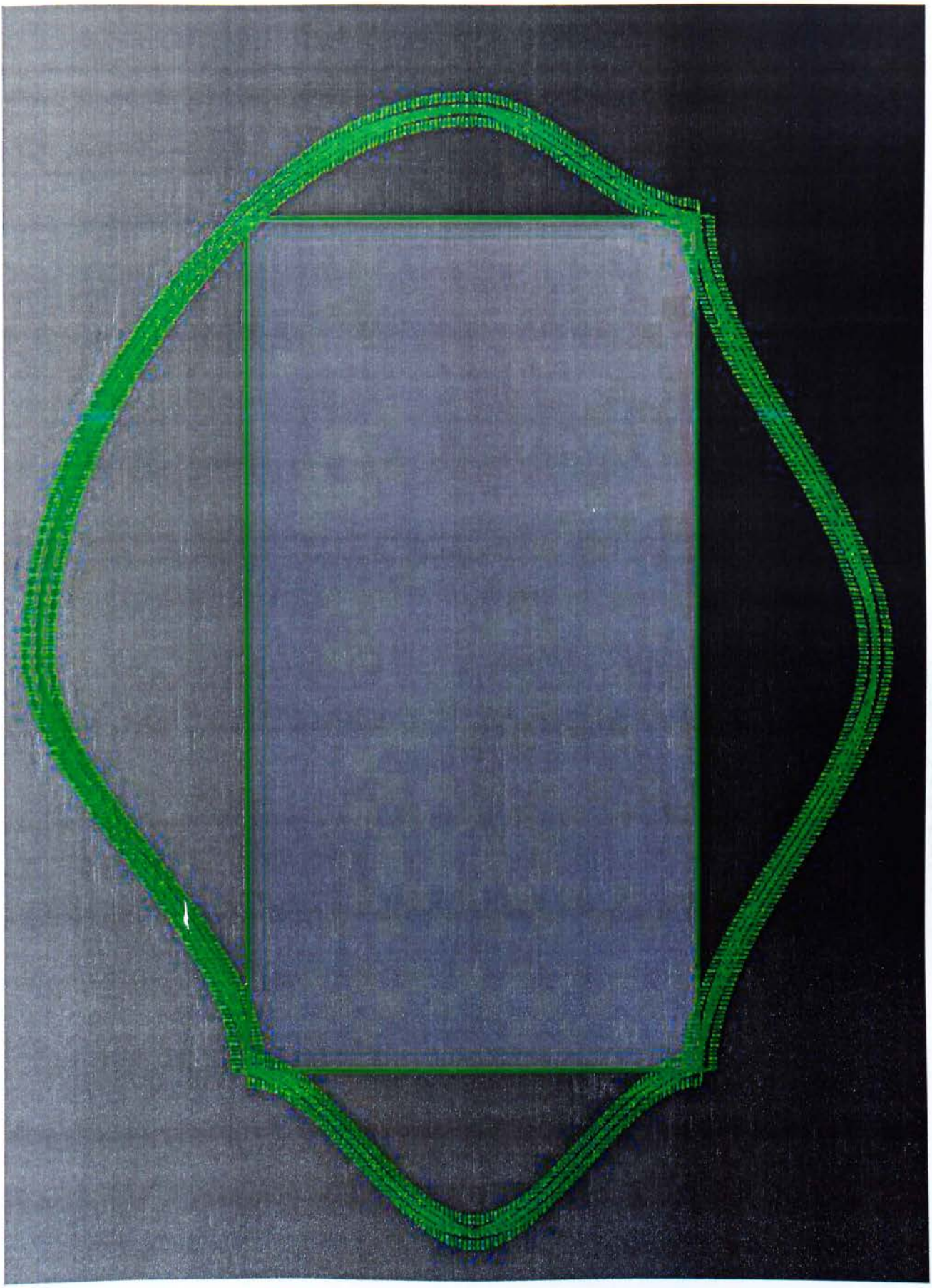
Πρώτο παράδειγμα, με δοκό απλής τετράπλευρης ορθογωνικής διατομής. Με Mt =1000 (Nmm), Mx=0 (Nmm), My=0 (Nmm) επιλογή απεικόνισης Sv και Τομές = 3.



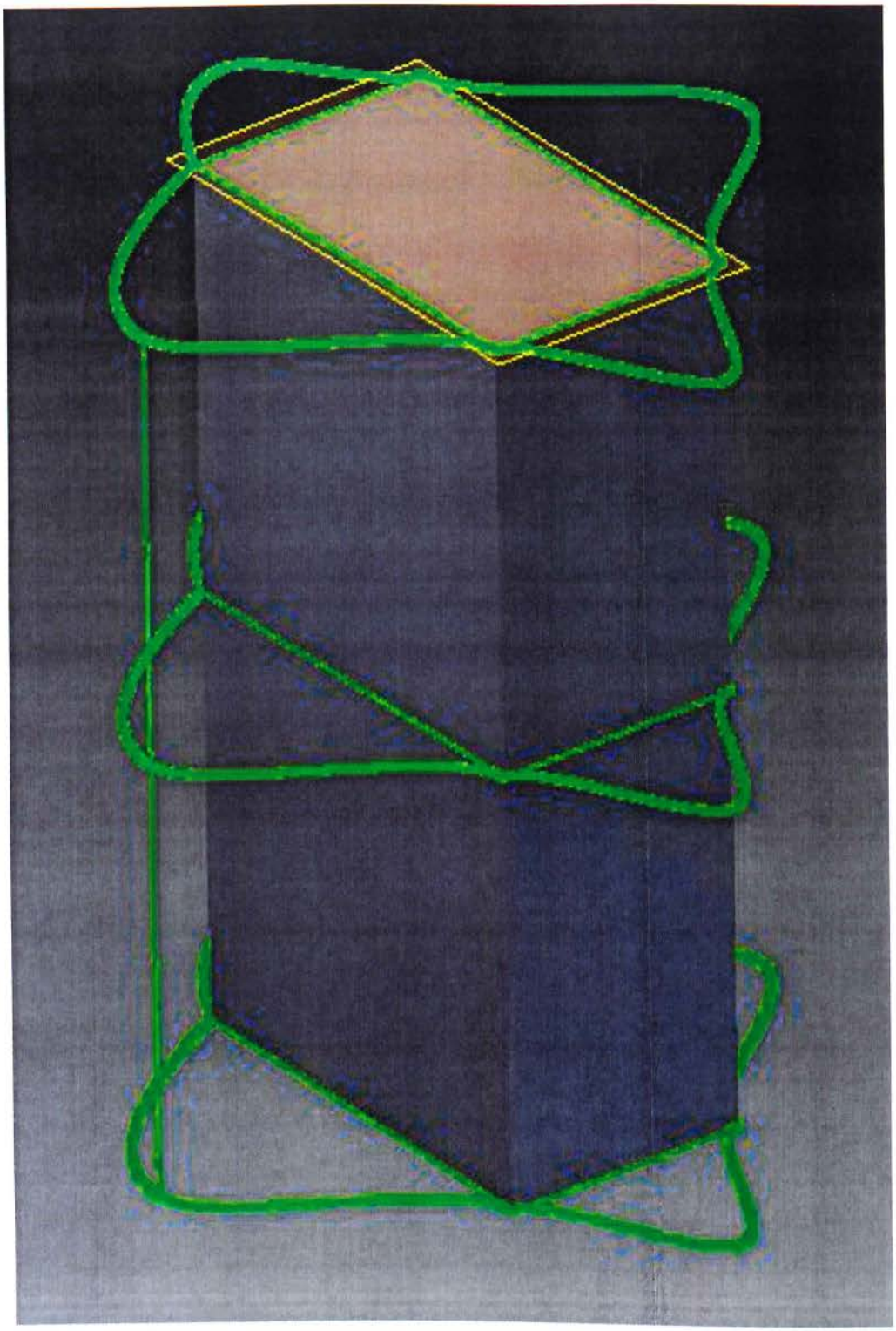
Στα παρακάτω παραδείγματα βλέπουμε την γραφική απεικόνιση των τάσεων που είναι αποτέλεσμα της εκτέλεσης του προγράμματος της πτυχιακής εργασίας. Η γραμμές (που στην ουσία είναι ένα πλήθος σημείων) γύρο απο την διατομή απεικονίζουν την κατανομή των τάσεων στην εκάστοτε διατομή και η γραμμή στην μέση απεικονίζει τις μέγιστες τάσεις κατά μήκος της δοκού.

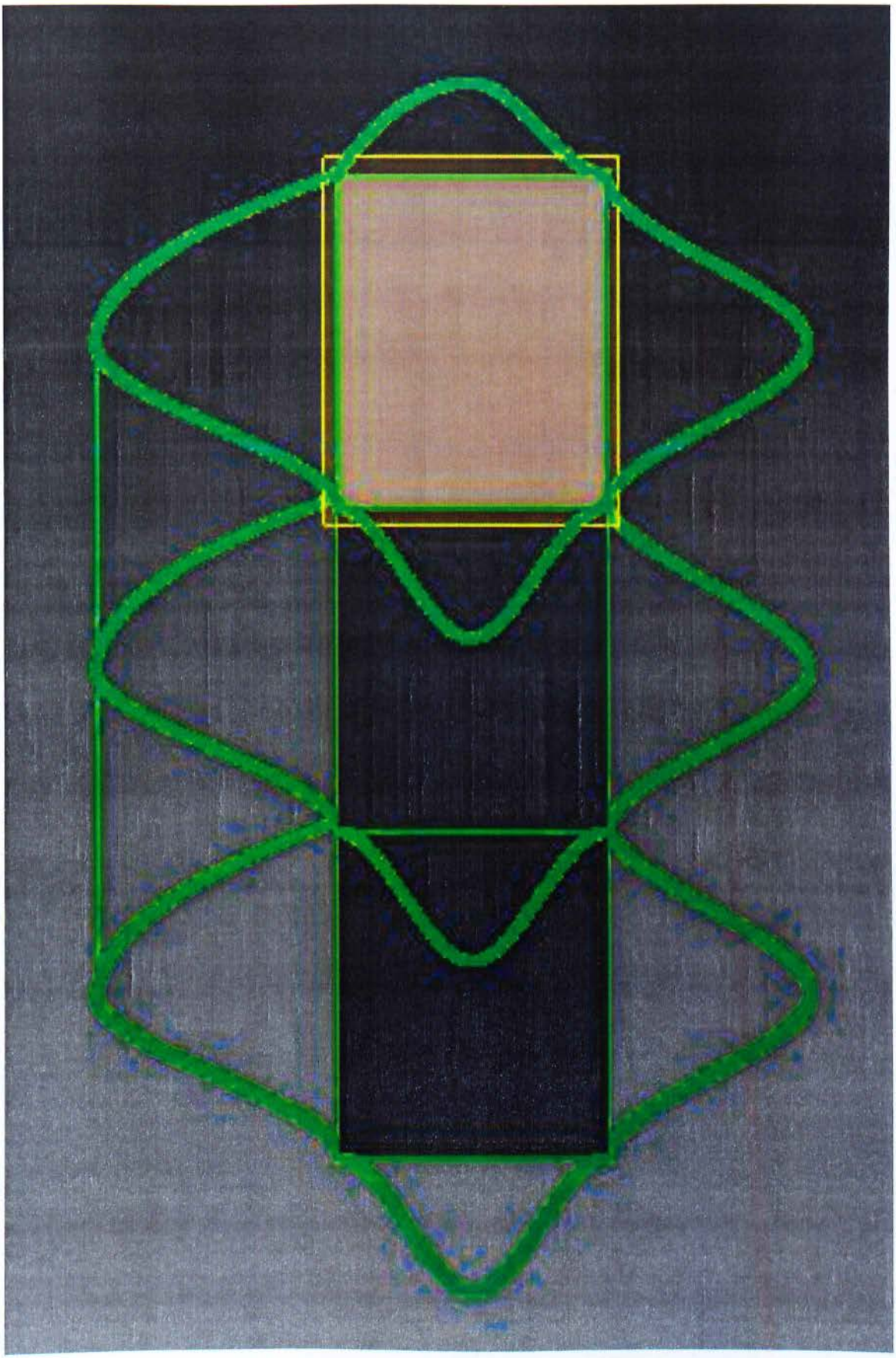

ΓΡΑΦΗΜΑ ΤΗΣ ΣΥΓΚΡΙΤΙΚΗΣ ΤΑΣΗΣ ΣΕ ΑΥΤΗΝ ΤΗ ΔΙΑΤΟΜΗ
ΓΡΑΦΗΜΑ ΤΩΝ ΜΕΓΙΣΤΩΝ ΤΙΜΩΝ ΑΝΑ ΤΟΜΗ ΤΗΣ ΣΥΓΚΡΙΤΙΚΗΣ ΤΑΣΗΣ ΣΕ ΟΛΟ ΤΟ ΜΗΚΟΣ ΤΗΣ ΔΟΚΟΥ

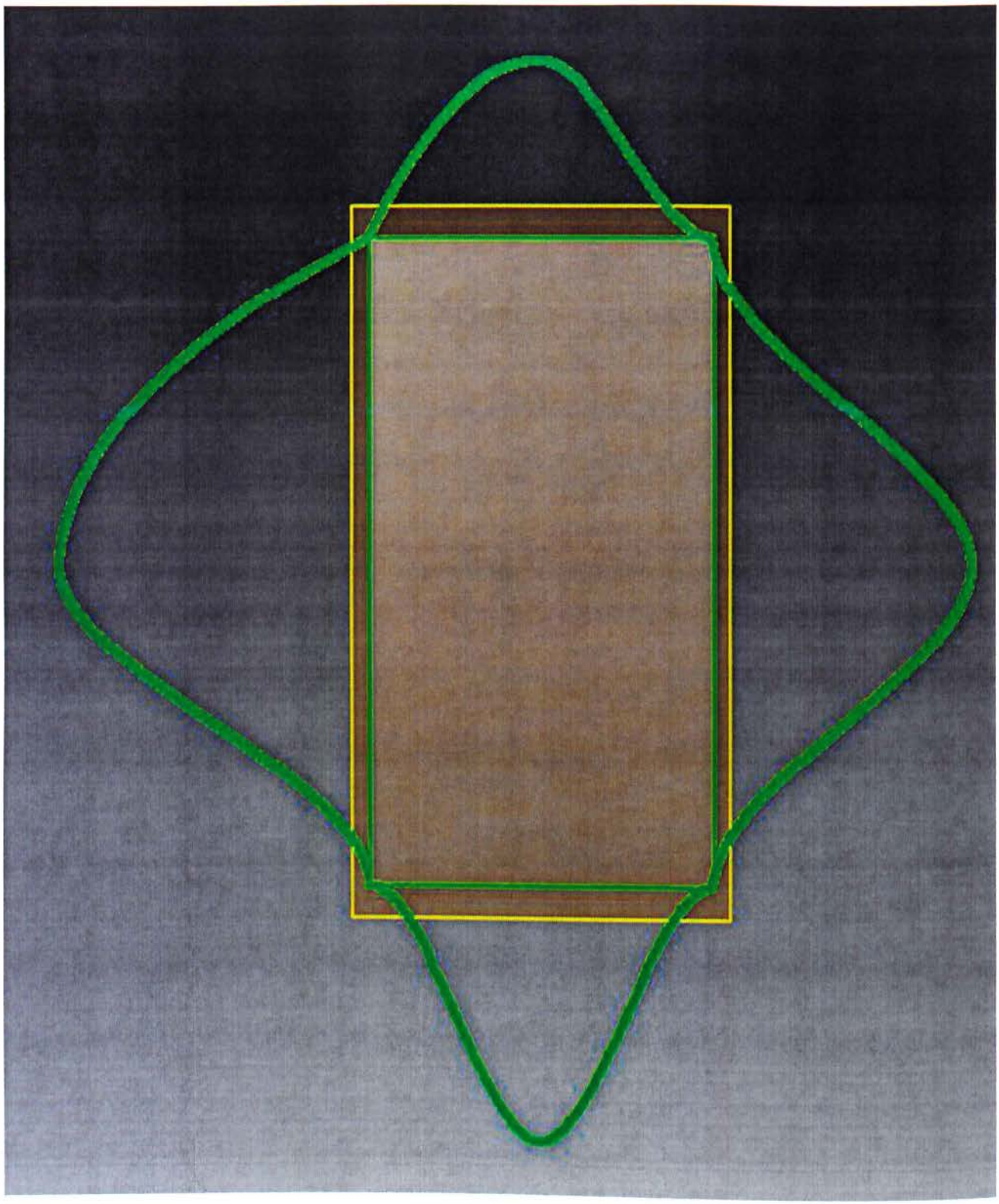




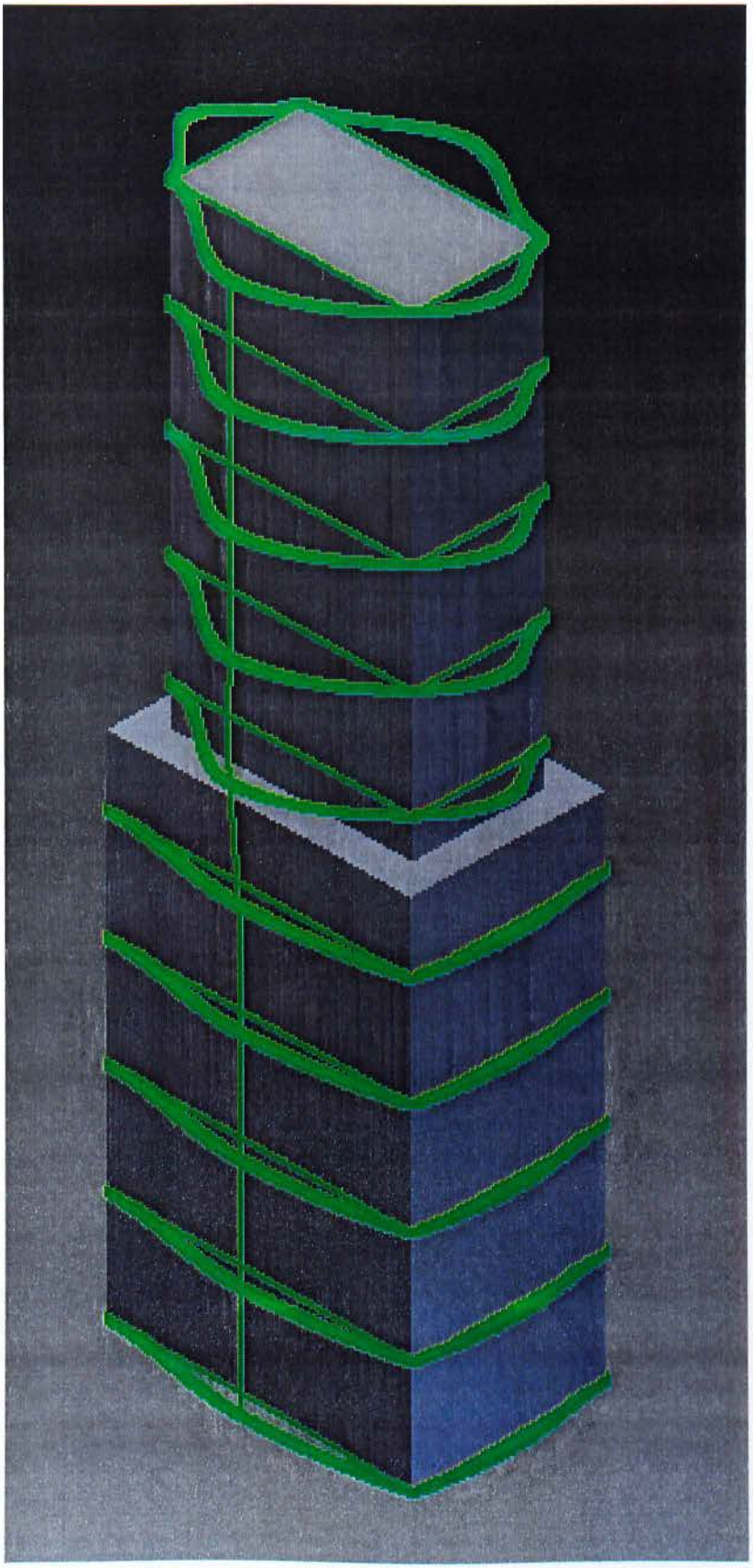
Δεύτερο παράδειγμα, με δοκό απλής τετράπλευρης ορθογωνικής διατομής. Με Mt =3000 (Nmm), Mx=0 (Nmm), My=0 (Nmm) επιλογή απεικόνισης Sv και Τομές = 3.

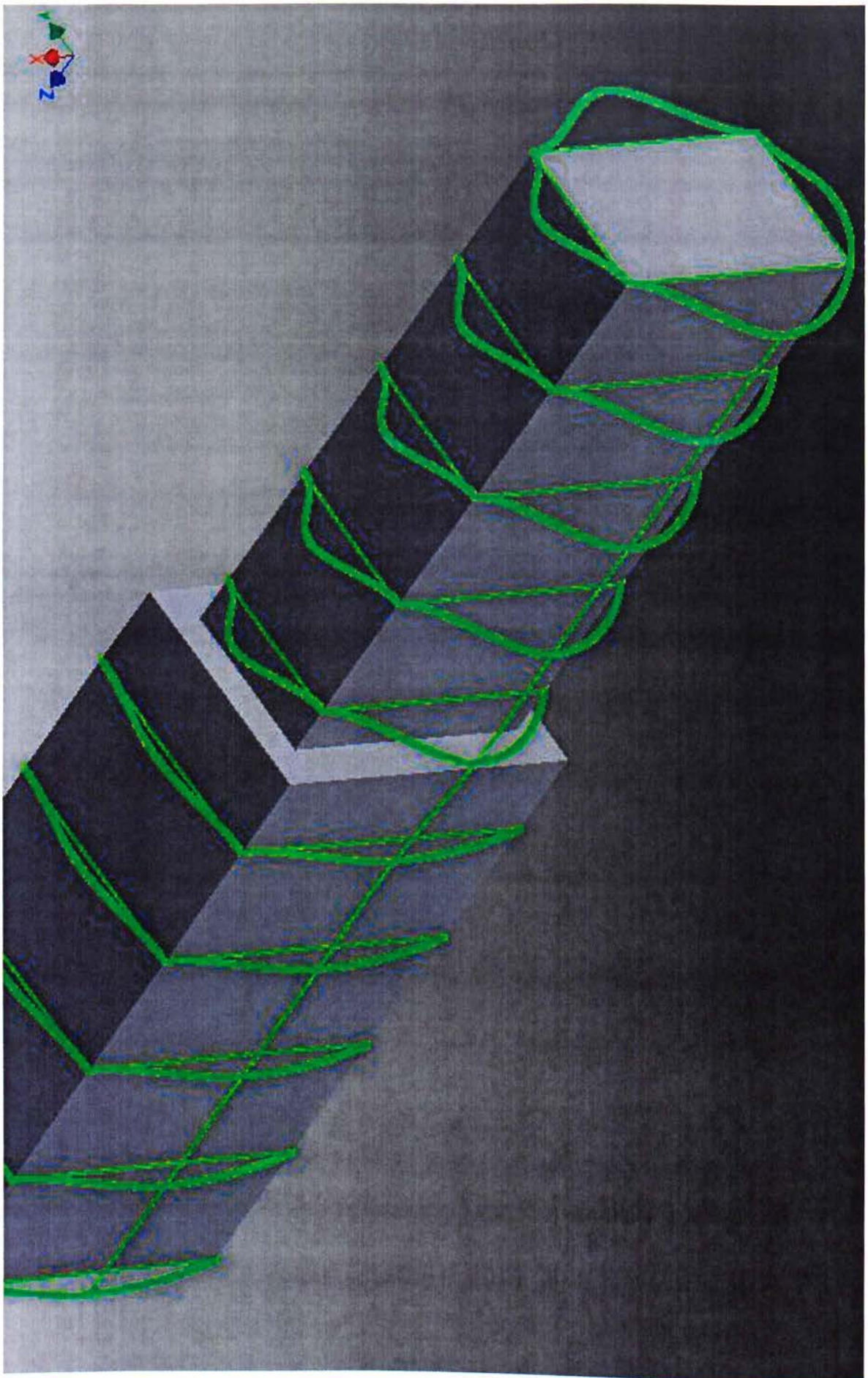


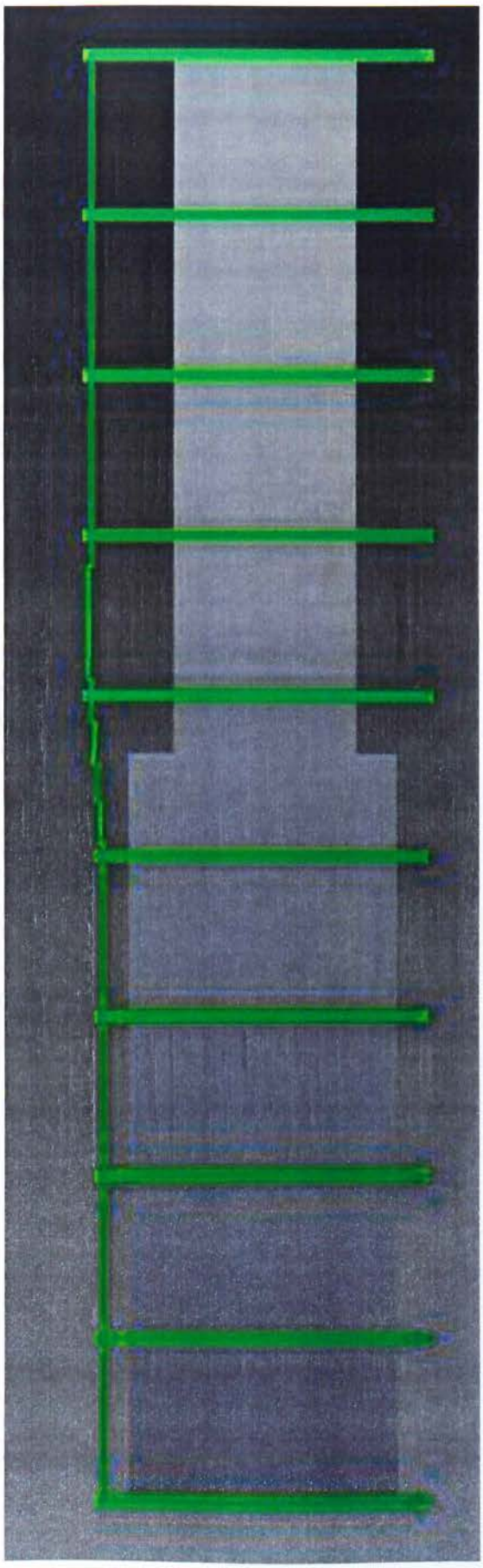


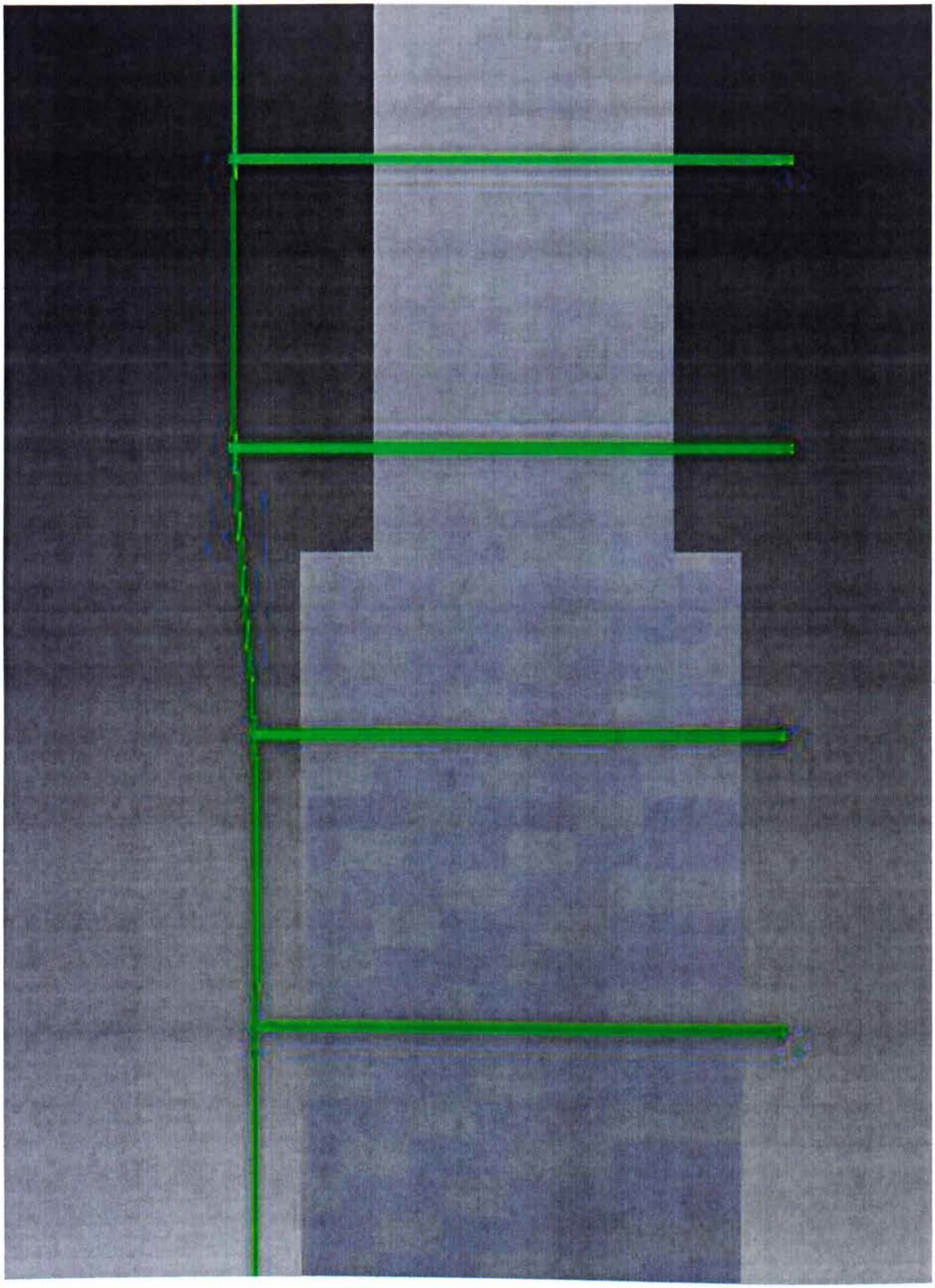


Τρίτο παράδειγμα, με δοκό διπλης τετράπλευρης ορθογωνικής διατομής. Με Mt =1000 (Nmm), Mx=0 (Nmm), My=0 (Nmm) επιλογή απεικόνισης Sv και Τομές = 10.

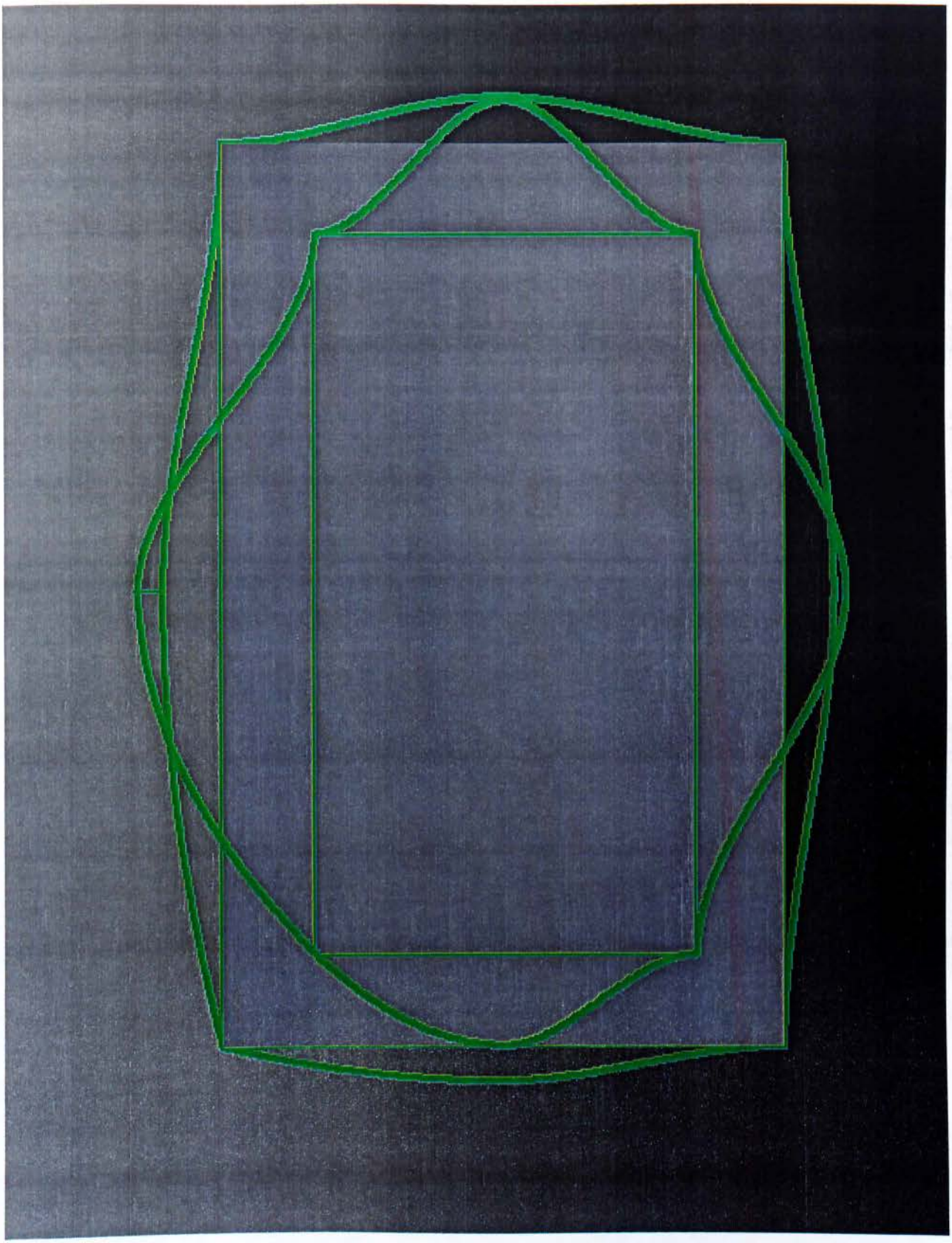


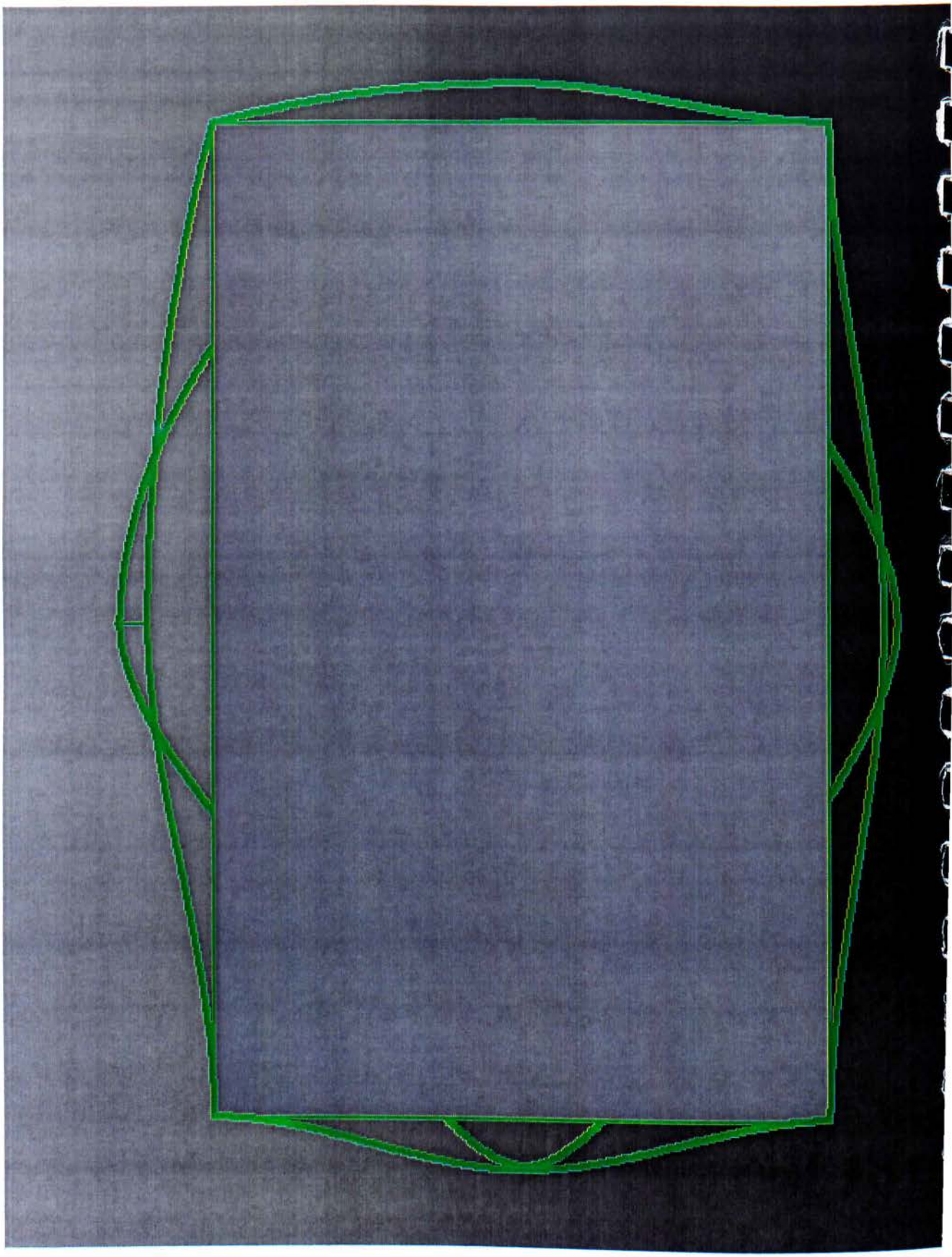






Παρατηρούμε την καμπύλη που κάνει η γραμμη που αναπαριστά τις μέγιστες τιμές των τάσεων. Αν είχαν γίνει περισσότερες τομές η καμπύλη της γραμμής θα ήταν πολύ πιο ομαλή.





6)Συμπεράσματα

Το API του inventor είναι ένα πολύ ισχυρό εργαλείο στα χέρια κάποιου με γνώσεις προγραμματισμού καθως και των εσωτερικών λειτουργιών του συστήματος. Το API δίνει στον χρήστη την δυνατότητα αυτοματοποίησης διαδικασιών με σύνολα εντολών για την διευκόλυνση του. Επίσης μπορεί να μειώσει τον χρόνο σχεδίασης, καθώς και να ελαχιστοποιήσει των λάθοι που ενδέχεται να γίνουν έφοσον η επαναληπτικές μεθόδοι εκτελούνται πάντα με τον ίδιο τρόπο.

Τα περισσότερα απο τα προβλήματα που εμφανίστηκαν κατά την διάρκεια της εκπόνησης της πτυχιακής επιλύθηκαν με επιτυχία. Το πρόβλημα των profiles τα οποία δεν υποστηρίζουν την ιδιότητα του rangebox object θα μπορούσε να επιλυθεί απο την ίδια την autodesk με την προσθήκη ενος τέτοιου object, εφόσον είναι απολύτος εφικτό και θα γλύτωνε χρόνο, μνήμη και υπολογιστική δύναμη απο το παρόν πρόγραμμα. Ένα ακόμα πρόβλημα ήταν η αδυναμία του surface evaluator να επεξεργαστεί surfaces τύπου plane surface, cone surface, elliptical cone surface (για την εύρεση centerline σε ίσιο loft). Η χρησιμοποίηση μιας άλλης μεθόδου έδωσε την λύση, παρόλ' αυτά θα ήταν χρήσιμη η αναβάθμιση του παρόντος surface evaluator και για αυτούς τους τύπους επιφανειών.

Στο inventor επίσης παρατηρήθηκε ανακρίβεια σε νούμερα πολλών δεκαδικών. Για παράδειγμα ενα σημείο και μια γραμμή ενώ ουσιαστικά συμπίπτουν δεν μπορούν να θεωρηθουν απο το inventor οτί συμπιπτουν αν μετά το 7 δεκαδικό ψηφίο του σημείου δεν είναι το ίδιο με το σημείο της γραμμής. Ο λόγος του προβλήματος αυτού παραμένει άγνωστος. Το πρόβλημα αυτό αντιμετωπίστηκε με την χρήση constraints (περιορισμών).

Το πρόβλημα με το άθροισμα των διανύσματων στις περιπτώσεις σημείων που ενώνουν line με spline ή spline με spline ήταν η εύρεση της γωνίας που σχημάτιζαν, εφόσον το getangle των measuretools δεν υποστήριζε αυτή τη λειτουργία. Το πρόβλημα λύθηκε ανάγοντας τις εφαπτομένες τους σε αυτο το σημείο σε διανύσματα και μετέπειτα συγκρίνοντας την γωνία που δημιουργούσαν με την εντολή getangle των vectors. Η αναβάθμιση του object getangle των measuretools να υποστηρίζει αυτή την λειτουργία θα βοηθούσε πολύ, πολλούς χρήστες.

Το τελευταίο πρόβλημα ήταν οι λάθος αρχικές τομές στα άκρα των loft features χωρίς centerline. Οι λόγοι που δημιουργούν αυτό το πρόβλημα αναφέρθηκαν στο αντίστοιχο κεφάλαιο, αλλά ο λόγος που δεν γινόταν να βρεθεί μια λύση είναι η αδυναμια του inventor να ορίσει στην 3D Bsplinecurve που αντιπροσώπευε την centerline ένα constrain που να ορίζει ότι θα κατασκευαστεί με τέτοιο τρόπο ώστε να είναι κάθετη στα οριακά workplane.

Η παρούσα πτυχιακή εργασία θα μπορούσε να αποτελέσει βάση για θέματα άλλων πτυχιακών, Ένα τέτοιο θέμα είναι αυτό της διόρθωσης των αρχικών τομών. Ένδιαφέρον θέμα θα ήταν επίσης όπως προανέφερα η αυτοματοποίηση της διαδικασίας τομών της δοκού και ένταξη μεθόδων optimization σε αυτήν. Έτσι ως αποτέλεσμα θα είχαμε την βέλτιστη δυνατή λύση τόσο σε πλήθος όσο και σε (μεταβαλώμενη) απόσταση μεταξύ των τομών κατα μήκος της δοκού.

Τέλος, αυτή η εργασία μου έδωσε την ευκαιρία να αναπτύξω τις γνώσεις μου στην προγραμματιστική γλώσσα Visual Basic for Application (V.B.A), όπως επίσης να μάθω για την λειτουργία και την δομή των σχεδιαστικών προγραμμάτων όπως το Autodesk Inventor. Ακόμα, έμαθα να οργανώνω καλύτερα την μελέτη μου και να εξελίξω την μεθοδολογία μου στην επίλυση προβλημάτων. Ο συνδυασμός αυτών διαμορφώνει ένα ισχυρό υπόβαθρο για περαιτέρω μελέτη και βελτίωση στον τομέα της ηλεκτρονικής σχεδίασης, αλλά και γενικότερα στην μηχανολογία. Επίσης, θεωρώ ότι η παιδεία ενός μηχανολόγου λόγω των γνώσεων του πάνω στην μεθοδολογία επίλυσης περίπλοκων μηχανολογικών προβλημάτων είχε ως αποτέλεσμα την ευκολότερη προσέγγιση μου στα προγραμματιστικά προβλήματα.

.

7)Βιβλιογραφία

1. Examing the matrix and other Inventor Math and Geometry Objects by Brian Ekins – Autodesk.

2. Συστήματα CAD απο Dr.-Ing Κωσταντίνο Στεργίου.

3. Moving Up to Autodesk Inventor Add-Ins by Neil Munro - C – Cubed Technologies Ltd.

4. http://forums.autodesk.com/t5/Autodesk-Inventor-Customization/bd-p/120

5. http://en.wikipedia.org/wiki/Computer-aided_design

8)Παράρτημα-Κώδικας

Ακολουθεί ο πηγαίος κώδικας της πτυχιακής.

```
Public obutton As Integer

Dim ocollection As ObjectCollection


Dim osvm As Point2d

Public Sub nik()

Dim oLoop As Long
oLoop = 0

Dim opartcompdef As PartComponentDefinition
Set opartcompdef = ThisApplication.ActiveDocument.ComponentDefinition

Dim otransgeom As TransientGeometry
Set otransgeom = ThisApplication.TransientGeometry

Dim otransobj As TransientObjects
Set otransobj = ThisApplication.TransientObjects

'orizo to p to gnosto 3,14...'

Dim o3dspline2 As SketchSpline3D

Dim dpi As Double
dpi = Atn(1) * 4

Dim ovcoll As ObjectCollection
Set ovcoll = otransobj.CreateObjectCollection

Dim ee As Integer
Dim omax3d As Sketch3D
Dim omax3dp As SketchPoint3D
Dim omax3dspline As SketchSpline3D


'orizo ola ta features oste na elenxo kai na exasfaliso oti to part mou'
'einai ftiaxmeno eite me extrutions eite me ena loft kai tipota allo'

Dim ofeatures As Object
Set ofeatures = opartcompdef.Features

Dim orevolve As Object
Set orevolve = opartcompdef.Features.RevolveFeatures
```

```
Dim osweep As Object
Set osweep = opartcompdef.Features.SweepFeatures

Dim ocoil As Object
Set ocoil = opartcompdef.Features.CoilFeatures

Dim oExtrude As Object
Set oExtrude = opartcompdef.Features.ExtrudeFeatures

Dim oLoft As Object
Set oLoft = opartcompdef.Features.LoftFeatures


If (orevolve.Count > 0) Or (osweep.Count > 0) Or (ocoil.Count > 0) Then

MsgBox "please use extrude or loft features "
End

ElseIf (oExtrude.Count > 0) And (oLoft.Count > 0) Then

MsgBox " parts with extrude and loft features cannot be examined "
End

ElseIf oLoft.Count > 1 Then

MsgBox "parts with only one loft feature can be examined "
End

ElseIf ofeatures.Count = 0 Then

MsgBox "please insert a feature"
End

End If

Dim oMt As Double      'The Torsional Moment
Dim oMx As Double      'The counteraction on the X axis
Dim oMy As Double      'The counteraction on the Y axis


'Default Values
oMt = 1000     'in N*mm    (50000)
oMx = 0        'in N*mm    (-560000)
oMy = 0        'in N*mm    (280000)


Dim Message, Title, Default
Dim MtValue, MxValue, MyValue
Dim ok As Integer
```

```
'Input of Mt
Message = "Enter a value for Mt  (Nmm)"
Title = "Input of Mt"
Default = "1000"
'1.12.02.00
MtValue = InputBox(Message, Title, Default)
oMt = MtValue


'Input of Mx
Message = "Enter a value for Mx  (Nmm)"
Title = "Input of Mx"
Default = "0"
'1.12.03.00
MxValue = InputBox(Message, Title, Default)
oMx = MxValue


'Input of My
Message = "Enter a value for My (Nmm)"
Title = "Input of My"
Default = "0"
'1.12.04.00
MyValue = InputBox(Message, Title, Default)
oMy = MyValue


Call UserForm1.Show

'edo exetazo tin periptosi pou to part mou exei geinei me loft'

If oLoft.Count = 1 Then

'afto tha xrisimopoiithei san taftotita gia kathe periptosi centerline oste sto telos to programa'
'na gnorizei ti input curve tha exei me tin methodo case, dioti ena centerline borei na einai bspline,dspline2d,line,line2d'

Dim nu As Integer

'i periptosi afti einia otan iparxei etoimi centerline sto loft'

Dim oceli As Object
Set oceli = opartcompdef.Features.LoftFeatures.Item(1).Centerline

'kai elenxoume na doume an einia'
```

```
If Not oceli Is Nothing Then

'apo 2d sketch i apo 3dsketch mias kai iparxoune kai oi dio dinatoties'

'gia 2dsketch'

If oceli.Parent.Type = kPlanarSketchObject Then

'edo tsekaroume an to centerline einai ftiaxmeno apo line i apo spline (se 2d sketch) den iparxei'
'i dianatotita sindiasmou line me spline gia tin dimiourgia centerline giati to inventor den to dexete'
'i centerline den einai smooth'

If oceli.Parent.SketchLines.Count > 0 Then

Dim olinecurve2d As LineSegment2d

Set olinecurve2d = oceli.Parent.SketchLines.Item(1).Geometry
nu = 1


ElseIf oceli.Parent.SketchSplines.Count > 0 Then

Dim osplinecurve2d As BSplineCurve2d

Set osplinecurve2d = oceli.Parent.SketchSplines.Item(1).Geometry
nu = 2


Else

MsgBox "error"

End If

'gia 3d sketch'

ElseIf oceli.Parent.Type = kSketch3DObject Then

If oceli.Parent.SketchLines3D.Count > 0 Then

Dim olinecurve3d As LineSegment
```

```
Set olinecurve3d = oceli.Parent.SketchLines3D.Item(1).Geometry
nu = 3


ElseIf oceli.Parent.SketchSplines3D.Count > 0 Then

Dim osplinecurve3d As BSplineCurve

Set o3dspline2 = oceli.Parent.SketchSplines3D.Item(1)

Set osplinecurve3d = oceli.Parent.SketchSplines3D.Item(1).Geometry
nu = 4


Else

MsgBox "error"

End If

End If

End If


Dim q As Integer
Dim oboolean1 As Boolean
oboolean1 = False

'edo exo xorisei tis katigories xoris centerline stis exis 2: 1)aftes pou ola ta sidefaces'
'einai bsplinesurface kai afti i katigoria exei alles dio ipokatigories :1)afti me parapano apo mia plevres'
'kai 2) afti me mia plevra. i defteri katigoria (2) einai afti pou esto mia plevra einai cone,ellipticalcone '
'(stis parapano einai panta mia plevra etsi kialios) kai telos planesurface'


Dim osidefaces As Object
Set osidefaces = opartcompdef.Features.LoftFeatures.Item(1).SideFaces

Dim osurfacetype As Object

'edo elenxo gia ta side faces kai ti type einai, an kapoia einai (plane,cone..) tote i timi boolean pernei value true'
'kai leitourgei san kleidi gia tis epomenes katigories'
```

```
For q = 1 To osidefaces.Count

Set osurfacetype = osidefaces.Item(q)

If osurfacetype.SurfaceType = kPlaneSurface Or osurfacetype.SurfaceType = kConeSurface Or osurfacetype.SurfaceType = kEllipticalConeSurface Then

oboolean1 = True

End If

Next


If (oboolean1 = False) And oceli Is Nothing Then


If osidefaces.Count > 1 Then

'afti i periptosi einai otan to loft den exei ftiaxtei me centerline (normal loft)'
'episis ena toulaxiston section tou einai ftiaxmeno me lines kai oxi splines i circle'
'afto exei os apotelesma to inventor gia na omalopoihsei to loft na ftiaxei plevres kai edges'
'se afti tin periptosi ekmetalevomaste to gegonos afto gia na vroume tin centerline'
'den epilegoume edges giati se polles periptoseis logo megalis kampilotitas ena edge borei na apoteleite'
'apo peritotera tou enos splines kai afto tha ekane diskolo kai argo to rapsimo tous se ena spline'
'kathos kai anaxiopisto. oi plevres diladi ta surfaces (ta opoia einai spline surfaces) omos einai eniaia'
'einai san planes se diafores morfes etsi boreis na vreis me sidetagmenes pano tous ta simeia pou thes i to antitheto'
'perisotera apo simeioseis pou exo...'
'stin ousia orizo enan arithmo simeion pano sta side faces (mesa apo evaluator) pou einai sto meso tis kai meta dimiourgo 3dspline apo afta ta simeia'
'ta simeia ton 3dspline einai pane paralila ston xoro etsi p.x ta prota simeia apo kathe 3dspline kai plevras boroun na orisoun ena epipedo (xriazonte 3 simeia) katheto sto feature'
'sto epipedo afto dimiourgoume ena 2dsketch kai me tin entoli projectedcut lamvanoume tin diatomi/tomes se afto to epipedo kai me mia diadikasia filtrarismatos kataligoume se
'se ena profile...apo to profile afto tha xrisimopoihsoume to centroid gia na ftiaxoume tin centerline'


Set osidefaces = opartcompdef.Features.LoftFeatures.Item(1).SideFaces

Dim j As Integer

'gia ta treia prota sidefaces kano afti tin diadikasia'
```

```
For j = 1 To 3

Dim osideface As Face
Set osideface = opartcompdef.Features.LoftFeatures.Item(1).SideFaces.Item(j)

Dim osurfeval As SurfaceEvaluator
Set osurfeval = osideface.Evaluator

Dim X As Integer
Dim Y As Double

Y = 0
X = 0

Dim adpoint(2) As Double
Dim oparam(1) As Double

Dim oFitPoints As ObjectCollection
Set oFitPoints = otransobj.CreateObjectCollection

'edo vriskei ta simeia'

Do

oparam(0) = 0.5
oparam(1) = Y

Call osurfeval.GetPointAtParam(oparam, adpoint)

Dim opoint As Point
Set opoint = otransgeom.CreatePoint(adpoint(0), adpoint(1), adpoint(2))

Call oFitPoints.Add(opoint)

Y = Y + 0.01
X = X + 1
'evala 100 oste na exei kali akriveia alla kai na min varenei to sistima'

Loop Until X = 101

'edo ftiaxnei tin 3dspline tis plevras'

Dim o3dsketch As Sketch3D
Set o3dsketch = opartcompdef.Sketches3D.Add

Dim o3dspline As SketchSpline3D

Set o3dspline = o3dsketch.SketchSplines3D.Add(oFitPoints)

Next
```

```
Dim w As Integer
Dim k As Integer


Dim ofitpoints2 As ObjectCollection
Set ofitpoints2 = otransobj.CreateObjectCollection

'gia kathe simeio'

For w = 1 To 101

Dim pp1 As SketchPoint3D
Dim pp2 As SketchPoint3D
Dim pp3 As SketchPoint3D

'me tin seira ta simeia apo tis treis protes 3dsplines ton plevron'

Set pp1 = opartcompdef.Sketches3D.Item(1).SketchPoints3D.Item(w)
Set pp2 = opartcompdef.Sketches3D.Item(2).SketchPoints3D.Item(w)
Set pp3 = opartcompdef.Sketches3D.Item(3).SketchPoints3D.Item(w)

'orizo workplane gia to 2dsketch'

Dim owplane As WorkPlane
Set owplane = opartcompdef.WorkPlanes.AddByThreePoints(pp1, pp2, pp3, False)

Dim oplanarsketch As PlanarSketch
Set oplanarsketch = opartcompdef.Sketches.Add(owplane)

oplanarsketch.Edit

Dim oDef As ControlDefinition
  Set                                    oDef                                    =
ThisApplication.CommandManager.ControlDefinitions.Item("SketchProjectCutEdgesCmd")

  oDef.Execute

Dim oprojectcut As ProjectedCut
Set oprojectcut = oplanarsketch.ProjectedCuts.Item(1)

oprojectcut.BreakLink

Dim ocircle As SketchCircle
Set                                    ocircle                                    =
oplanarsketch.SketchCircles.AddByThreePoints(oplanarsketch.ModelToSketchSpace(pp1.
Geometry),                                    oplanarsketch.ModelToSketchSpace(pp2.Geometry),
oplanarsketch.ModelToSketchSpace(pp3.Geometry))

Dim obox As Box2d
```

```
Set obox = ocircle.RangeBox


Dim ocol As ObjectCollection
Set ocol = otransobj.CreateObjectCollection


Dim ostp1 As Point2d
Dim oendp1 As Point2d
Dim ostp2 As Point2d
Dim oendp2 As Point2d
Dim oflagpoint As Point2d

Dim oSketchEntities As Object
Set oSketchEntities = oplanarsketch.SketchEntities


Dim c As Integer

For c = 1 To oSketchEntities.Count

Dim otype As ObjectTypeEnum

otype = oplanarsketch.SketchEntities.Item(c).Type

If otype = kSketchSplineObject Or otype = kSketchLineObject Then

Dim oSketchEnt As Object
Set oSketchEnt = oplanarsketch.SketchEntities.Item(c)

ocol.Add oSketchEnt

End If

Next


Dim ocol2 As Object
Set ocol2 = otransobj.CreateObjectCollection

Dim L As Integer
L = 1

Dim g As Integer
g = 1

Dim obool As Boolean
obool = False
```

```
Do While obool = False


Set oflagpoint = ocol.Item(g).StartSketchPoint.Geometry

Set ostp1 = ocol.Item(L).StartSketchPoint.Geometry
Set oendp1 = ocol.Item(L).EndSketchPoint.Geometry

If ostp1.IsEqualTo(oendp1) Then

ocol2.Add ocol.Item(L)

Set oProfile = oplanarsketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 1
g = L

Else


Set ostp2 = ocol.Item(L + 1).StartSketchPoint.Geometry
Set oendp2 = ocol.Item(L + 1).EndSketchPoint.Geometry

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = False) Then

ocol2.Add ocol.Item(L + 1)

L = L + 1

End If

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = True) Then

ocol2.Add ocol.Item(L + 1)

ocol2.Add ocol.Item(g)

Set oProfile = oplanarsketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 2

g = L

End If

End If
```

```
If L = ocol.Count + 1 Then

obool = True

End If

Loop


Dim oboxcol As ObjectCollection
Set oboxcol = otransobj.CreateObjectCollection


Dim oprof As Profiles
Set oprof = oplanarsketch.Profiles

Dim oprofile2 As Profile

For Each oprofile2 In oprof

Dim orange As Box2d
Set orange = oprofile2.Item(1).Item(1).Curve.Evaluator.RangeBox

Dim oPath As ProfilePath

For Each oPath In oprofile2

Dim oEntity As ProfileEntity

For Each oEntity In oPath

Dim oEval2 As Curve2dEvaluator
Set oEval2 = oEntity.Curve.Evaluator


Dim ocurverange As Box2d
Set ocurverange = oEval2.RangeBox

Dim ominpoint As Point2d
Dim omaxpoint As Point2d

Set ominpoint = ocurverange.MinPoint
Set omaxpoint = ocurverange.MaxPoint


If (orange.Contains(ominpoint) = False) Then

orange.Extend ominpoint

End If
```

```
If (orange.Contains(omaxpoint) = False) Then

orange.Extend omaxpoint

End If


Next

Next

If orange.IsDisjoint(obox) = True Then

oprofile2.Delete

Else

oboxcol.Add orange

End If

Next


Dim ocentroid As Point2d
Set ocentroid = oplanarsketch.Profiles.Item(1).RegionProperties.Centroid

'to metatrepo se point apo point2d gia na to xrisimopoihso gia tin 3dspine (centerline)'

Dim ocentroid3d As Point
Set ocentroid3d = oplanarsketch.SketchToModelSpace(ocentroid)

'ta vazo mesa se collection'

ofitpoints2.Add ocentroid3d

oplanarsketch.ExitEdit

oplanarsketch.Delete

owplane.Delete

Next

Dim o3dsketches As Object
Set o3dsketches = opartcompdef.Sketches3D
```

```
'svino ta proigoume 3dsketches gia na min pianoun xoro'

Do Until o3dsketches.Count = 0

o3dsketches.Item(1).Delete

Loop

'telos dimourgo to teliko 3dsketch kai ftiaxno tin center line apo tin collection me ta centroids'


Dim o3dsketch2 As Sketch3D
Set o3dsketch2 = opartcompdef.Sketches3D.Add



Set o3dspline2 = o3dsketch2.SketchSplines3D.Add(ofitpoints2)
Set osplinecurve3d = o3dspline2.Geometry
nu = 5


ElseIf osidefaces.Count = 1 Then


'afti einai i periptosi opou to loft den einai ftiaxmeno me centerline kai ola tou ta section einia ftiaxena'
'eite apo circle eite apo spline eite kai apo ta dio. se afti tin periptosi to inventor dimiourgei monaxa mia plevra'
'kai ena edge (sta plagia milame panta) opou einai iarxi kai to telos tis plevras...'
'se aftin tin periptosi kanoume to idio me tin periptosi me tis plevres me mia mono diafora'
'anti na ftiaxoume ta splines apo ta simeia pou orisame apo kathe plevra tha ta friaxoune apo mia plevra allazontas kathe fora'
'tis sintetagmenes..xorizontas tin epifaneia se tesera idia komatia '
'i ipolipi diadikasia einai idia me afti me tis plevres'


Set osideface = opartcompdef.Features.LoftFeatures.Item(1).SideFaces.Item(1)

Set osurfeval = osideface.Evaluator

Y = 0
X = 0


Set oFitPoints = otransobj.CreateObjectCollection
```

```
Do

oparam(0) = 0
oparam(1) = Y

Call osurfeval.GetPointAtParam(oparam, adpoint)

Set opoint = otransgeom.CreatePoint(adpoint(0), adpoint(1), adpoint(2))

Call oFitPoints.Add(opoint)

Y = Y + 0.01
X = X + 1


Loop Until X = 101

Set o3dsketch = opartcompdef.Sketches3D.Add

Set o3dslpine = o3dsketch.SketchSplines3D.Add(oFitPoints)

'svino tin idia collection oste n amin fiaxo alli'

oFitPoints.Clear

'midenizo tous tis metavlites oste na tis xanaxrisimopoihso , gia na min oriso kialous kai varinei to programa'

X = 0
Y = 0


Do

oparam(0) = 0.33333
oparam(1) = Y

Call osurfeval.GetPointAtParam(oparam, adpoint)

Set opoint = otransgeom.CreatePoint(adpoint(0), adpoint(1), adpoint(2))

Call oFitPoints.Add(opoint)

Y = Y + 0.01
X = X + 1
```

```
Loop Until X = 101

Set o3dsketch = opartcompdef.Sketches3D.Add

Set o3dslpine = o3dsketch.SketchSplines3D.Add(oFitPoints)

oFitPoints.Clear


X = 0
Y = 0


Do

oparam(0) = 0.66666
oparam(1) = Y

Call osurfeval.GetPointAtParam(oparam, adpoint)

Set opoint = otransgeom.CreatePoint(adpoint(0), adpoint(1), adpoint(2))

Call oFitPoints.Add(opoint)

Y = Y + 0.01
X = X + 1


Loop Until X = 101

Set o3dsketch = opartcompdef.Sketches3D.Add

Set o3dslpine = o3dsketch.SketchSplines3D.Add(oFitPoints)

oFitPoints.Clear

X = 0
Y = 0


'den kano kai gia to 1 (sto x) giati ekei arxizei kai ekei telionei i plevra ara to 0 eiani to idio'


'apo edo kai kato einai idio'


Set ofitpoints2 = otransobj.CreateObjectCollection
```

```
For w = 1 To 101

Set pp1 = opartcompdef.Sketches3D.Item(1).SketchPoints3D.Item(w)
Set pp2 = opartcompdef.Sketches3D.Item(2).SketchPoints3D.Item(w)
Set pp3 = opartcompdef.Sketches3D.Item(3).SketchPoints3D.Item(w)

Set owplane = opartcompdef.WorkPlanes.AddByThreePoints(pp1, pp2, pp3, False)

Set oplanarsketch = opartcompdef.Sketches.Add(owplane)


oplanarsketch.Edit

  Set oDef =
ThisApplication.CommandManager.ControlDefinitions.Item("SketchProjectCutEdgesCmd")

  oDef.Execute


Set oprojectcut = oplanarsketch.ProjectedCuts.Item(1)

oprojectcut.BreakLink


Set ocircle =
oplanarsketch.SketchCircles.AddByThreePoints(oplanarsketch.ModelToSketchSpace(pp1.
Geometry), oplanarsketch.ModelToSketchSpace(pp2.Geometry),
oplanarsketch.ModelToSketchSpace(pp3.Geometry))

Set obox = ocircle.RangeBox

Set ocol = otransobj.CreateObjectCollection


Set oSketchEntities = oplanarsketch.SketchEntities


For c = 1 To oSketchEntities.Count

otype = oplanarsketch.SketchEntities.Item(c).Type

If otype = kSketchSplineObject Or otype = kSketchLineObject Then

Set oSketchEnt = oplanarsketch.SketchEntities.Item(c)

ocol.Add oSketchEnt

End If

Next
```

```
Set ocol2 = otransobj.CreateObjectCollection

L = 1

g = 1

obool = False

Do While obool = False


Set oflagpoint = ocol.Item(g).StartSketchPoint.Geometry

Set ostp1 = ocol.Item(L).StartSketchPoint.Geometry
Set oendp1 = ocol.Item(L).EndSketchPoint.Geometry

If ostp1.IsEqualTo(oendp1) Then

ocol2.Add ocol.Item(L)

Set oProfile = oplanarsketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 1
g = L

Else


Set ostp2 = ocol.Item(L + 1).StartSketchPoint.Geometry
Set oendp2 = ocol.Item(L + 1).EndSketchPoint.Geometry

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = False) Then

ocol2.Add ocol.Item(L + 1)

L = L + 1

End If

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = True) Then

ocol2.Add ocol.Item(L + 1)

ocol2.Add ocol.Item(g)
```

```
Set oProfile = oplanarsketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 2

g = L

End If

End If

If L = ocol.Count + 1 Then

obool = True

End If

Loop

Set oboxcol = otransobj.CreateObjectCollection

Set oprof = oplanarsketch.Profiles

For Each oprofile2 In oprof

Set orange = oprofile2.Item(1).Item(1).Curve.Evaluator.RangeBox

For Each oPath In oprofile2

For Each oEntity In oPath

Set oEval2 = oEntity.Curve.Evaluator

Set ocurverange = oEval2.RangeBox

Set ominpoint = ocurverange.MinPoint
Set omaxpoint = ocurverange.MaxPoint

If (orange.Contains(ominpoint) = False) Then

orange.Extend ominpoint

End If
```

```
If (orange.Contains(omaxpoint) = False) Then
orange.Extend omaxpoint
End If

Next
Next
If orange.IsDisjoint(obox) = True Then
oprofile2.Delete
Else
oboxcol.Add orange
End If
Next

Set ocentroid = oplanarsketch.Profiles.Item(1).RegionProperties.Centroid
Set ocentroid3d = oplanarsketch.SketchToModelSpace(ocentroid)
ofitpoints2.Add ocentroid3d
oplanarsketch.ExitEdit
oplanarsketch.Delete
owplane.Delete

Next

Set o3dsketches = opartcompdef.Sketches3D

Do Until o3dsketches.Count = 0
o3dsketches.Item(1).Delete
Loop
```

```
Set o3dsketch2 = opartcompdef.Sketches3D.Add


Set o3dspline2 = o3dsketch2.SketchSplines3D.Add(ofitpoints2)
Set osplinecurve3d = o3dspline2.Geometry
nu = 6

End If

ElseIf (oboolean1 = True) And oceli Is Nothing Then


'afti i periptosi eiani gia loft to opoio exei sxediastei me tetoio tropo oste oi epifaneies pou dimiourgounte'
'einai isies kai oxi splines. se afti tin periptos exoume diaforetiki prosegisi dioti stis isies epifaneies den'
'borei na leitourgisei to surface evaluator. etsi orizoume mia eftheia grami apo ta centroids to sections'
'me afti ti grammi kai to evaluator tis tin xorizoume se simeia opou tha xrisimopoihsoume gia na dosoume tin thesi'
'ton epipedon pou tha kanoun cut sto feature mas'


Dim oStartPoint As Point2d
Dim oEndPoint As Point2d

Dim osections As Object
Set osections = opartcompdef.Features.LoftFeatures.Item(1).Sections

'orizoume to start kai to end point apo to proto kai to telefteo section'

Set oStartPoint = osections.Item(1).RegionProperties.Centroid
Set oEndPoint = osections.Item(osections.Count).RegionProperties.Centroid

Dim oplanarsketch1 As PlanarSketch
Set oplanarsketch1 = osections.Item(1).Parent

Dim oplanarsketch2 As PlanarSketch
Set oplanarsketch2 = osections.Item(osections.Count).Parent

'dimiourgoume ena 3dsketch'

Set o3dsketch = opartcompdef.Sketches3D.Add

'metatrepo ta dio simeia apo 2d se 3d kai me afta friaxno mia 3dline'

Dim oline3d As SketchLine3D
```

```
Set oline3d = o3dsketch.SketchLines3D.AddByTwoPoints(oplanarsketch1.SketchToModelSpace(oStart Point), oplanarsketch2.SketchToModelSpace(oEndPoint))

Dim oevaluator As CurveEvaluator
Set oevaluator = oline3d.Geometry.Evaluator

Dim omin As Double
Dim oMax As Double

'me to evaluator vrisko tin megisti kai tin elaxisti timi ton parametron tis line'

Call oevaluator.GetParamExtents(omin, oMax)


Dim cparam As Double

'exontas tin megisti kai tin elaxisti parametro xorizo se idia komatia tin line'
'o arithmos tou w exartate apo to posa kommatia thelo kai kata sinepeia poso megali akriveia thelo'
'gia na min ftiaxnei workplane stin arxi kai to telos tou feature'


Message = "Enter  the number of cuts(enter desired number minus one)"    ' Set prompt.
Title = "cut number"    ' Set title.
Default = "9"    ' Set default.
' Display message, title, and default value.
MyValue = InputBox(Message, Title, Default)
ok = MyValue



Set omax3d = opartcompdef.Sketches3D.Add

For w = 0 To ok

cparam = omin + ((oMax - omin) / ok) * w

Dim adparam(0) As Double

adparam(0) = cparam


'metatrepo tin kathe parametro se simeio ston xoro'

Call oevaluator.GetPointAtParam(adparam, adpoint)

Set opoint = otransgeom.CreatePoint(adpoint(0), adpoint(1), adpoint(2))

'dimiourgo kiallo 3dsketch '
```

```
Set o3dsketch2 = opartcompdef.Sketches3D.Add

'kai sto sketch afto prostheto ta points pou vrika'

Dim o3dpoint As SketchPoint3D
Set o3dpoint = o3dsketch2.SketchPoints3D.Add(opoint)

Dim oplana As Face
Set oplana = opartcompdef.Features.LoftFeatures.Item(1).StartFace

'edo me ta points pou exo kai tin 3dline prosetho workplanes ta opoia xrisimevoune san
vasi gia tin dimiourgia'
'ton metetpeita cut'

'edo iparxei kai i enalaktiki me addnormaltocurve alla den exo apofasisei akoma'
'se afti ti fasi apla to workplane einai paralilo me to plane tou start face'

Set owplane = opartcompdef.WorkPlanes.AddByPlaneAndPoint(oplana, o3dpoint, False)

Dim oSketch As PlanarSketch
Set oSketch = opartcompdef.Sketches.Add(owplane)

oSketch.Edit

Set oDef =
ThisApplication.CommandManager.ControlDefinitions.Item("SketchProjectCutEdgesCmd")

oDef.Execute

Set oprojectcut = oSketch.ProjectedCuts.Item(1)

oprojectcut.BreakLink

oSketch.ExitEdit



Dim odocument As PartDocument
Set odocument = ThisApplication.ActiveDocument

Call odocument.SelectSet.Clear
Call odocument.SelectSet.Select(oSketch)

Dim oCopyControlDef As ControlDefinition
Set oCopyControlDef =
ThisApplication.CommandManager.ControlDefinitions.Item("AppCopyCmd")
```

```
oCopyControlDef.Execute

Dim odocument2 As PartDocument
Set odocument2 = ThisApplication.Documents.Add(kPartDocumentObject,
ThisApplication.FileManager.GetTemplateFile(kPartDocumentObject))

Dim oDefinition As PartComponentDefinition
Set oDefinition = odocument2.ComponentDefinition


Dim othinsketch As PlanarSketch
Set othinsketch = oDefinition.Sketches.Add(oDefinition.WorkPlanes.Item(3))



othinsketch.Edit


   Dim oPasteControlDef As ControlDefinition
   Set oPasteControlDef =
ThisApplication.CommandManager.ControlDefinitions.Item("AppPasteCmd")

   oPasteControlDef.Execute

   Set oProfile = othinsketch.Profiles.AddForSolid


   Dim othinExtrude As ExtrudeFeature
   Set othinExtrude = oDefinition.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile,
0.00001, kPositiveExtentDirection, kNewBodyOperation)


othinsketch.ExitEdit

Call SvCalc(odocument2, othinExtrude, oMt, oMx, oMy, oLoop, osvm, ocollection)

Call odocument2.Close(True)


For ee = 1 To ocollection.Count

Set omax3dp =
omax3d.SketchPoints3D.Add(oSketch.SketchToModelSpace(ocollection.Item(ee)), False)

Next


Set ovov = oSketch.SketchToModelSpace(osvm)
```

```
ovcoll.Add ovov

oLoop = oLoop + 1

Next

Set omax3dspline = omax3d.SketchSplines3D.Add(ovcoll)

End

End If

Dim oeval As CurveEvaluator
Dim oeval2d As Curve2dEvaluator


If nu = 1 Then

Set oeval2d = olinecurve2d.Evaluator

Call oeval2d.GetParamExtents(omin, oMax)

ElseIf nu = 2 Then

Set oeval2d = osplinecurve2d.Evaluator

Call oeval2d.GetParamExtents(omin, oMax)

ElseIf nu = 3 Then

Set oeval = olinecurve3d.Evaluator

Call oeval.GetParamExtents(omin, oMax)

ElseIf nu = 4 Or nu = 5 Or nu = 6 Then

Set oeval = osplinecurve3d.Evaluator

Call oeval.GetParamExtents(omin, oMax)

End If

Dim o3dsketch3 As Sketch3D
Set o3dsketch3 = opartcompdef.Sketches3D.Add
```

```
Dim d As Integer

Dim adpoint2(2) As Double

Dim opoint2 As Point

Dim opoint2d As Point2d

Dim oskepi As SketchPoint


Dim v As Integer
Dim t As Double

Dim oa As Object
t = 0


'edo ipologizoume to meso oro ton emvadon olon ton section'
'tha mas xriastei argotera'

Set osections = opartcompdef.Features.LoftFeatures.Item(1).Sections

For v = 1 To osections.Count

Set oa = opartcompdef.Features.LoftFeatures.Item(1).Sections.Item(v).RegionProperties

t = t + oa.Area

Next

t = t / osections.Count

Dim osketchcollection As ObjectCollection
Set osketchcollection = otransobj.CreateObjectCollection


Message = "Enter  the number of cuts(enter desired number minus one)"    ' Set prompt.
Title = "cut number"    ' Set title.
Default = "9"    ' Set default.
' Display message, title, and default value.
MyValue = InputBox(Message, Title, Default)
ok = MyValue

Set omax3d = opartcompdef.Sketches3D.Add
```

```
For d = 0 To ok

cparam = omin + ((oMax - omin) / ok) * d

adparam(0) = cparam

'dialego kai pali analogos tin periptosi poio evaluator tha xrisimopoihso'


If nu = 1 Then

Call oeval2d.GetPointAtParam(adparam, adpoint2)

Set opoint2d = otransgeom.CreatePoint2d(adpoint2(0), adpoint2(1))

Set oskepi = oceli.Parent.SketchPoints.Add(opoint2d)

Set oline = oceli.Parent.SketchLines.Item(1)

Set owplane = opartcompdef.WorkPlanes.AddByNormalToCurve(oline, oskepi, False)


ElseIf nu = 2 Then

Call oeval2d.GetPointAtParam(adparam, adpoint2)

Set opoint2d = otransgeom.CreatePoint2d(adpoint2(0), adpoint2(1))

Set oskepi = oceli.Parent.SketchPoints.Add(opoint2d)

Dim oskspline2d As SketchSpline

Set oskspline2d = oceli.Parent.SketchSplines.Item(1)

Set owplane = opartcompdef.WorkPlanes.AddByNormalToCurve(oskspline2d, oskepi,
False)


ElseIf nu = 3 Then

Call oeval.GetPointAtParam(adparam, adpoint2)

Set opoint2 = otransgeom.CreatePoint(adpoint2(0), adpoint2(1), adpoint2(2))

Set o3dpoint = o3dsketch3.SketchPoints3D.Add(opoint2)
```

```
Set oline3d = oceli.Parent.SketchLines3D.Item(1)

Set owplane = opartcompdef.WorkPlanes.AddByNormalToCurve(oline3d, o3dpoint, False)

ElseIf nu = 5 Or nu = 6 Or nu = 4 Then

Call oeval.GetPointAtParam(adparam, adpoint2)

Set opoint2 = otransgeom.CreatePoint(adpoint2(0), adpoint2(1), adpoint2(2))

Set o3dpoint = o3dsketch3.SketchPoints3D.Add(opoint2)

Set owplane = opartcompdef.WorkPlanes.AddByNormalToCurve(o3dspline2, o3dpoint,
False)


End If

Set oSketch = opartcompdef.Sketches.Add(owplane)

oSketch.Edit

Dim oRadius As Double


oRadius = Sqr((t / dpi))

If nu = 1 Or nu = 2 Then

Set opoint2 = oceli.Parent.SketchToModelSpace(opoint2d)

End If


Dim skpoint2d As Point2d
Set skpoint2d = oSketch.ModelToSketchSpace(opoint2)

Set ocircle = oSketch.SketchCircles.AddByCenterRadius(skpoint2d, oRadius)


  Set                                    oDef                                    =
ThisApplication.CommandManager.ControlDefinitions.Item("SketchProjectCutEdgesCmd")

oDef.Execute


Set oprojectcut = oSketch.ProjectedCuts.Item(1)

oprojectcut.BreakLink
```

```
Set obox = oSketch.SketchCircles.Item(1).RangeBox

Set ocol = otransobj.CreateObjectCollection

Set oSketchEntities = oSketch.SketchEntities

For c = 1 To oSketchEntities.Count

otype = oSketch.SketchEntities.Item(c).Type

If otype = kSketchSplineObject Or otype = kSketchLineObject Then


Set oSketchEnt = oSketch.SketchEntities.Item(c)

ocol.Add oSketchEnt

End If

Next

Set ocol2 = otransobj.CreateObjectCollection

L = 1

g = 1

obool = False

Do While obool = False


Set oflagpoint = ocol.Item(g).StartSketchPoint.Geometry

Set ostp1 = ocol.Item(L).StartSketchPoint.Geometry
Set oendp1 = ocol.Item(L).EndSketchPoint.Geometry

If ostp1.IsEqualTo(oendp1) Then

ocol2.Add ocol.Item(L)

Set oProfile = oSketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 1
g = L

Else
```

```
Set ostp2 = ocol.Item(L + 1).StartSketchPoint.Geometry
Set oendp2 = ocol.Item(L + 1).EndSketchPoint.Geometry

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = False) Then

ocol2.Add ocol.Item(L + 1)

L = L + 1

End If

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = True) Then

ocol2.Add ocol.Item(L + 1)

ocol2.Add ocol.Item(g)

Set oProfile = oSketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 2

g = L

End If

End If

If L = ocol.Count + 1 Then

obool = True

End If

Loop

Set oboxcol = otransobj.CreateObjectCollection


Set oprof = oSketch.Profiles
```

```
For Each oprofile2 In oprof

Set orange = oprofile2.Item(1).Item(1).Curve.Evaluator.RangeBox

For Each oPath In oprofile2

For Each oEntity In oPath

Set oEval2 = oEntity.Curve.Evaluator

Set ocurverange = oEval2.RangeBox

Set ominpoint = ocurverange.MinPoint
Set omaxpoint = ocurverange.MaxPoint

If (orange.Contains(ominpoint) = False) Then

orange.Extend ominpoint

End If

If (orange.Contains(omaxpoint) = False) Then

orange.Extend omaxpoint

End If

Next

Next

If orange.IsDisjoint(obox) = True Then

oprofile2.Delete

Else

oboxcol.Add orange

End If

Next
```

```
Dim f As Integer

Dim oEnd As Boolean
oEnd = False

f = 1

Dim osketchsplines As Object
Set osketchsplines = oSketch.SketchSplines

If osketchsplines.Count > 0 Then

Do While oEnd = False

Dim osksp As SketchSpline

Dim obox2 As Box2d

Set obox2 = oSketch.SketchSplines.Item(f).RangeBox

If (oboxcol.Item(1).IsDisjoint(obox2) = True) Then

Set osksp = oSketch.SketchSplines.Item(f)

osksp.Delete

If osketchsplines.Count = 0 Then

oEnd = True

Else

f = 1

End If

ElseIf f = osketchsplines.Count Then

oEnd = True

Else

f = f + 1

End If

Loop


End If
```

```
Dim n As Integer
Dim oEnd2 As Boolean
n = 1
oEnd2 = False


Dim osketchlines As Object
Set osketchlines = oSketch.SketchLines

If osketchlines.Count > 0 Then


Dim oskl As SketchLine
Dim obox3 As Box2d


Do While oEnd2 = False


Set obox3 = oSketch.SketchLines.Item(n).RangeBox

If (oboxcol.Item(1).IsDisjoint(obox3) = True) Then

Set oskl = oSketch.SketchLines.Item(n)

oskl.Delete

If osketchlines.Count = 0 Then

oEnd2 = True

Else

n = 1

End If

ElseIf n = osketchlines.Count Then

oEnd2 = True

Else

n = n + 1

End If

Loop

End If
```

```
Set osketchpoints = oSketch.SketchPoints

n = 1
oEnd2 = False

If osketchpoints.Count > 0 Then

Do While oEnd2 = False

Set obox3 = oSketch.SketchPoints.Item(n).RangeBox

If (oboxcol.Item(1).IsDisjoint(obox3) = True) Then

Set op2d = oSketch.SketchPoints.Item(n)

op2d.Delete

If osketchpoints.Count = 0 Then

oEnd2 = True

Else

n = 1

End If

ElseIf n = osketchpoints.Count Then

oEnd2 = True

Else

n = n + 1

End If

Loop

End If

ocircle.Delete

oSketch.ExitEdit
```

```
Set odocument = ThisApplication.ActiveDocument

Call odocument.SelectSet.Clear
Call odocument.SelectSet.Select(oSketch)

Set oCopyControlDef =
ThisApplication.CommandManager.ControlDefinitions.Item("AppCopyCmd")

oCopyControlDef.Execute

Set odocument2 = ThisApplication.Documents.Add(kPartDocumentObject,
ThisApplication.FileManager.GetTemplateFile(kPartDocumentObject))


Set oDefinition = odocument2.ComponentDefinition


Set othinsketch = oDefinition.Sketches.Add(oDefinition.WorkPlanes.Item(3))



othinsketch.Edit



   Set oPasteControlDef =
ThisApplication.CommandManager.ControlDefinitions.Item("AppPasteCmd")

   oPasteControlDef.Execute

   Set oProfile = othinsketch.Profiles.AddForSolid

   Set othinExtrude = oDefinition.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile,
0.00001, kPositiveExtentDirection, kNewBodyOperation)


othinsketch.ExitEdit

Call SvCalc(odocument2, othinExtrude, oMt, oMx, oMy, oLoop, osvm, ocollection)

Call odocument2.Close(True)


For ee = 1 To ocollection.Count

Set omax3dp =
omax3d.SketchPoints3D.Add(oSketch.SketchToModelSpace(ocollection.Item(ee)), False)

Next
```

```
Set ovov = oSketch.SketchToModelSpace(osvm)

ovcoll.Add ovov


oLoop = oLoop + 1


Next


Set omax3dspline = omax3d.SketchSplines3D.Add(ovcoll)


End


End If

If oExtrude.Count > 0 Then

' panta (apo to telefteo extrude) to end face giati otan to ena einai pano sto allo den iparxei startface !!
'episi einia pio efkolo na vrei apo pou tha arxisei an exei polla extrude...'

'edo elenxoume an to kommati exei geinei me intersect operation to opoio periplekei polu ta pragmata kathos'
'san leitourgia dimiourgei enan ogo apo to epilexmeno profile kai krataei oti iliko iparxei mesa se afton ton ogo'
'kai to ipoleipo to diagrafei...to thema einai omos oti afto pou menei den exei oute start face oute end face'
'epeidi orizeis enan ogo kai stin ousia den xerei to inventor se poio simeio mesa se afton ton ogo arxizei na iparxei'
'uliko kai pote teleionei etsi to solid pou mas menei einai akathoristo'

Dim p As Integer

For p = 1 To oExtrude.Count

If oExtrude.Item(p).Operation = kIntersectOperation Then

MsgBox "error ,dont use intersect method"

End

End If

Next
```

```
'edo tsekaro se periptosi pou iparxoune parapano apo ena extrude na einai ola pros tin idia
katefthinsi
'oste to sxima tis dokou na min exei akathoristo sxima '
'oste na min aliothoune ta apotelesmata tou bem alla kirios oste na kanei sosta
'cut to programa (na min xekinisei apo allo face kai pros alli fora'
'stin periptosi pou den einai tote vgenei minima lathous oste o xristis na allaxei ton tropo
sxediasis
'tis dokou kai na epixeirisei ek neou'


If oExtrude.Count > 1 Then


Dim oplane1 As Plane
Dim oplane2 As Plane

Set                                    oplane1                                    =
opartcompdef.Features.ExtrudeFeatures.Item(1).Profile.Parent.PlanarEntityGeometry

Dim h As Integer

For h = 2 To oExtrude.Count

Set                                    oplane2                                    =
opartcompdef.Features.ExtrudeFeatures.Item(h).Profile.Parent.PlanarEntityGeometry

If Not oplane1.IsParallelTo(oplane2) Then

MsgBox "error"

End

End If

Next

Dim oface As Face

Dim oboolean As Boolean
oboolean = False

v = oExtrude.Count
```

```
'edo elenxoume an to telefteo extrude exei cut operation dioti an einai cut to
'end face einai mesa stin doko kai iparxei sfalma..'

Do Until oboolean = True

If oExtrude.Item(v).Operation = kJoinOperation Then

Set oface = oExtrude.Item(v).EndFaces.Item(1)

oboolean = True

Else

v = v - 1

End If

Loop

Dim r As Integer
Dim odist As Object
Dim oleng As Double
oleng = 0

For r = 1 To oExtrude.Count

Set odist = oExtrude.Item(r).Extent.Distance

oleng = oleng + odist.Value

Next



Else

Set oface = opartcompdef.Features.ExtrudeFeatures.Item(1).EndFaces.Item(1)

Set odist = oExtrude.Item(1).Extent.Distance

oleng = odist.Value

End If
```

```
Message = "Enter  the number of cuts"    ' Set prompt.
Title = "cut number"    ' Set title.
Default = "10"    ' Set default.
' Display message, title, and default value.
MyValue = InputBox(Message, Title, Default)
ok = MyValue

Dim i As Double


Set omax3d = opartcompdef.Sketches3D.Add

For i = 0 To oleng Step (oleng / (ok - 1))


Set owplane = opartcompdef.WorkPlanes.AddByPlaneAndOffset(oface, -i)

Set oSketch = opartcompdef.Sketches.Add(owplane)

oSketch.Edit

  Set                                    oDef                                    =
ThisApplication.CommandManager.ControlDefinitions.Item("SketchProjectCutEdgesCmd")

oDef.Execute

Set oprojectcut = oSketch.ProjectedCuts.Item(1)

oprojectcut.BreakLink

Set ocol = otransobj.CreateObjectCollection

Set oSketchEntities = oSketch.SketchEntities

For c = 1 To oSketchEntities.Count

otype = oSketch.SketchEntities.Item(c).Type

If Not otype = kSketchPointObject Then


Set oSketchEnt = oSketch.SketchEntities.Item(c)

ocol.Add oSketchEnt

End If

Next
```

```
Set ocol2 = otransobj.CreateObjectCollection

L = 1

g = 1

obool = False

Do While obool = False

If ocol.Item(L).Type = kSketchCircleObject Then

ocol2.Add ocol.Item(g)

Set oProfile = oSketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 1
g = L

Else

Set oflagpoint = ocol.Item(g).StartSketchPoint.Geometry

Set ostp1 = ocol.Item(L).StartSketchPoint.Geometry
Set oendp1 = ocol.Item(L).EndSketchPoint.Geometry

If ostp1.IsEqualTo(oendp1) Then

ocol2.Add ocol.Item(L)

Set oProfile = oSketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 1
g = L

Else

Set ostp2 = ocol.Item(L + 1).StartSketchPoint.Geometry
Set oendp2 = ocol.Item(L + 1).EndSketchPoint.Geometry

If (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = False) Then

ocol2.Add ocol.Item(L + 1)

L = L + 1
```

```
ElseIf (oendp1.IsEqualTo(ostp2) = True) And (oendp2.IsEqualTo(oflagpoint) = True) Then

ocol2.Add ocol.Item(L + 1)

ocol2.Add ocol.Item(g)

Set oProfile = oSketch.Profiles.AddForSolid(False, ocol2)

ocol2.Clear

L = L + 2

g = L

End If

End If

End If

If L = ocol.Count + 1 Then

obool = True

End If

If oSketch.Profiles.Count > 1 Then

oSketch.ExitEdit

MsgBox "error"

End

End If

Loop

oSketch.ExitEdit

Set odocument = ThisApplication.ActiveDocument

Call odocument.SelectSet.Clear
Call odocument.SelectSet.Select(oSketch)

Set                                    oCopyControlDef                                    =
ThisApplication.CommandManager.ControlDefinitions.Item("AppCopyCmd")

oCopyControlDef.Execute
```

```
Set odocument2 = ThisApplication.Documents.Add(kPartDocumentObject, ThisApplication.FileManager.GetTemplateFile(kPartDocumentObject))

Set oDefinition = odocument2.ComponentDefinition


Set othinsketch = oDefinition.Sketches.Add(oDefinition.WorkPlanes.Item(3))

othinsketch.Edit

  Set oPasteControlDef = ThisApplication.CommandManager.ControlDefinitions.Item("AppPasteCmd")

  oPasteControlDef.Execute

  Set oProfile = othinsketch.Profiles.AddForSolid


  Set othinExtrude = oDefinition.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, 0.00001, kPositiveExtentDirection, kNewBodyOperation)


othinsketch.ExitEdit

Call SvCalc(odocument2, othinExtrude, oMt, oMx, oMy, oLoop, osvm, ocollection)

Call odocument2.Close(True)

For ee = 1 To ocollection.Count

Set omax3dp = omax3d.SketchPoints3D.Add(oSketch.SketchToModelSpace(ocollection.Item(ee)), False)

Next

Set ovov = oSketch.SketchToModelSpace(osvm)

ovcoll.Add ovov


oLoop = oLoop + 1

Next

Set omax3dspline = omax3d.SketchSplines3D.Add(ovcoll)

End
End If
End Sub
```

```
Public Sub metafora(oNewSketch, Section_SN, X, Y, oXYCen, ud, Sv, Tt, Sb, It, Xm, Ym,
Um, Svm, osvm, ocollection)

Dim oSvmax, oTtmax, oSbmax As Double

Dim oSvmaxSN, oTtmaxSN, oSbmaxSN As Long

If obutton = 1 Then

Call FindMax(Sv, oSvmax, oSvmaxSN)  '1.11.01.00

ElseIf obutton = 2 Then


Call FindMax(Tt, oTtmax, oTtmaxSN)  '1.11.01.00

ElseIf obutton = 3 Then

Call FindMax(Sb, oSbmax, oSbmaxSN)  '1.11.01.00

End If


Dim dpi As Double
dpi = Atn(1) * 4


Set ocollection = ThisApplication.TransientObjects.CreateObjectCollection


Dim j As Integer

For j = 1 To 1000

Dim opoint2dd As Point2d
Set opoint2dd = ThisApplication.TransientGeometry.CreatePoint2d((X(j) / 10), (Y(j) / 10))

Dim opoint2d As SketchPoint
Set opoint2d = oNewSketch.SketchPoints.Add(opoint2dd)

Dim k As Integer

Dim osskentites As Object
Set osskentites = oNewSketch.SketchEntities

Dim oboxx As Box2d
Dim olline As SketchLine
Dim osspline As SketchSpline
Dim occircle As SketchCircle
```

```
For k = 1 To osskentites.Count

Set obbject = osskentites.Item(k)


If obbject.Type = kSketchLineObject Then

Set oboxx = osskentites.Item(k).RangeBox

If oboxx.Contains(opoint2d.Geometry) = True Then

Set olline = osskentites.Item(k)

Call oNewSketch.GeometricConstraints.AddCoincident(opoint2d, olline)

End If

ElseIf obbject.Type = kSketchSplineObject Then

Set oboxx = osskentites.Item(k).RangeBox

If oboxx.Contains(opoint2d.Geometry) = True Then

Set osspline = osskentites.Item(k)

Call oNewSketch.GeometricConstraints.AddCoincident(opoint2d, osspline)

End If

ElseIf obbject.Type = kSketchCircleObject Then

Set oboxx = osskentites.Item(k).RangeBox

If oboxx.Contains(opoint2d.Geometry) = True Then

Set occircle = osskentites.Item(k)
Call oNewSketch.GeometricConstraints.AddCoincident(opoint2d, occircle)

End If
End If

Next

Dim opoint22 As Point2d
Set opoint22 = opoint2d.Geometry

Dim op() As Double
opoint22.GetPointData op()
```

```
Dim oeval As Curve2dEvaluator

Dim oline As LineSegment2d
Dim ospline As BSplineCurve2d
Dim ocircle As Circle2d


Dim oattached As Object
Set oattached = opoint2d.AttachedEntities

Dim kk As Integer


For kk = 1 To oattached.Count

If oattached.Item(kk).Type = kSketchLineObject Then

Set oline = oattached.Item(kk).Geometry

Set oeval = oline.Evaluator

ElseIf oattached.Item(kk).Type = kSketchSplineObject Then

Set ospline = oattached.Item(kk).Geometry

Set oeval = ospline.Evaluator

ElseIf oattached.Item(kk).Type = kSketchCircleObject Then

Set ocircle = oattached.Item(kk).Geometry

Set oeval = ocircle.Evaluator

End If


Dim oparam() As Double
Dim ovectors() As Double


Dim oguessparams() As Double
Dim omaxdeviations() As Double
Dim osoltypes() As SolutionNatureEnum


Call oeval.GetParamAtPoint(op, oguessparams, omaxdeviations, oparam, osoltypes)

Dim otangents() As Double

Call oeval.GetTangent(oparam, otangents)
```

```
Dim ovector As Vector2d
Set ovector = ThisApplication.TransientGeometry.CreateVector2d(otangents(0), otangents(1))


Dim omatrix As Matrix2d
Set omatrix = ThisApplication.TransientGeometry.CreateMatrix2d

Dim ocenter As Point2d
Set ocenter = ThisApplication.TransientGeometry.CreatePoint2d(0, 0)

If oattached.Item(kk).Type = kSketchLineObject Or oattached.Item(kk).Type = kSketchSplineObject Then

Call omatrix.SetToRotation((dpi / 2), ocenter)

ElseIf oattached.Item(kk).Type = kSketchCircleObject Then

Call omatrix.SetToRotation(-(dpi / 2), ocenter)

End If

Call ovector.Normalize

Call ovector.TransformBy(omatrix)

Dim jj As Double

If obutton = 1 Then

jj = Sv(j)

ElseIf obutton = 2 Then

jj = Tt(j)

ElseIf obutton = 3 Then

jj = Sb(j)

End If


If jj = 0 Then

GoTo 666

End If
```

```
If oattached.Count > 1 Then

Set oeval = oattached.Item(1).Geometry.Evaluator


Dim oparam1() As Double
Dim ovectors1() As Double


Dim oguessparams1() As Double
Dim omaxdeviations1() As Double
Dim osoltypes1() As SolutionNatureEnum


Call oeval.GetParamAtPoint(op, oguessparams1, omaxdeviations1, oparam1, osoltypes1)

Dim otangents1() As Double

Call oeval.GetTangent(oparam1, otangents1)


Dim ovector1 As Vector2d
Set ovector1 = ThisApplication.TransientGeometry.CreateVector2d(otangents1(0),
otangents1(1))


Set oeval = oattached.Item(2).Geometry.Evaluator


Dim oparam2() As Double
Dim ovectors2() As Double


Dim oguessparams2() As Double
Dim omaxdeviations2() As Double
Dim osoltypes2() As SolutionNatureEnum


Call oeval.GetParamAtPoint(op, oguessparams2, omaxdeviations2, oparam2, osoltypes2)

Dim otangents2() As Double

Call oeval.GetTangent(oparam2, otangents2)

Dim ovector2 As Vector2d
Set ovector2 = ThisApplication.TransientGeometry.CreateVector2d(otangents2(0),
otangents2(1))
  Dim oangle As Double
```

```
oangle = ovector1.AngleTo(ovector2)
 Dim ll As Double
 ll = 1 / (2 * Cos(oangle / 2))
 jj = jj * ll
 Call ovector.ScaleBy(jj)
Else
Call ovector.ScaleBy(jj)
End If
Call opoint22.TranslateBy(ovector)
666
ocollection.Add opoint22
Next
opoint2d.Delete
Next
Dim oske As SketchPoint
If obutton = 1 Then
Set osvm = ocollection.Item(oSvmaxSN)
ElseIf obutton = 2 Then
Set osvm = ocollection.Item(oTtmaxSN)
ElseIf obutton = 3 Then
Set osvm = ocollection.Item(oSbmaxSN)
End If
End Sub

Private Sub CommandButton1_Click()
obutton = 1
Call UserForm1.Hide
```

```
End Sub
Private Sub CommandButton2_Click()
obutton = 2
Call UserForm1.Hide
End Sub
Private Sub CommandButton3_Click()
obutton = 3
Call UserForm1.Hide
End Sub
```