ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟΣ ΤΟΜΕΑΣ ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ
ΧΑΝΙΑ 2014

Μεταπτυχιακή Εργασία με θέμα: «Αλγόριθμοι για τον βέλτιστο σχεδιασμό δικτυωμάτων μεγάλου μεγέθους σε συνδυασμό με τη μέθοδο των πεπερασμένων στοιχείων»

(Algorithms for the optimal design of large scale trusses in combination with the finite elements method)

Πουλή Ιωάννα

2011019027

Επιτροπή Αξιολόγησης:
1. Σταυρουλάκης Γεώργιος
2. Δουλάμης Αναστάσιος
3. Μαρινάκης Ιωάννης

Ευχαριστίες

Από τη θέση αυτή επιθυμώ να ευχαριστήσω θερμά τον μέχρι πρότινος επιβλέποντα Καθηγητή του Πολυτεχνείου Κρήτης, Γεώργιο Ε. Σταυρουλάκη για την συμβολή του και καθοδήγηση του στην εκπόνηση αυτής της εργασίας, καθώς και για την επιλογή του θέματος. Πρόκειται για έναν εξαίρετο απόκτημα-παράδειγμα του Πολυτεχνείου Κρήτης, με πολλές γνώσεις και δημοσιεύσεις στον Παγκόσμιο Τύπο.

Ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω στην οικογένεια μου, τον πατέρα μου Φώτη, τη μητέρα μου Αντιγόνη ,και τον αδερφό μου Παρασκευά , για την ηθική , ψυχολογική , πνευματική και υλική υποστήριξή τους και κυρίως την υπομονή που επέδειξαν κατά τη διάρκεια της φοίτησης μου. Τους ευχαριστώ για τις αξίες που μου μετέδωσαν και για τον άνθρωπο που με έκαναν. Ελπίζω και εύχομαι να είναι περήφανοι για μένα, και υπόσχομαι να τους τιμάω πάντα.

Και ένα τελευταίο και πολύ σημαντικό, τεράστιο ευχαριστώ στους παιδικούς μου φίλους και φίλες από την ιδιαίτερη πατρίδα μου την Πάτρα, που έμειναν δίπλα μου σαν να μην έφυγα ποτέ.

Σας ευχαριστώ όλους από την καρδιά μου!

Η επιστήμη πρέπει να προήλθε από την αίσθηση ότι κάτι δεν πήγαινε καλά.

Thomas Carlyle

Περιεχόμενα

Περίληψη

Κατά την εκπόνηση της ερευνητικής αυτής εργασίας, σκοπός είναι ο συνδυασμός του Βέλτιστου σχεδιασμού Δικτυωμάτων μεγάλου μεγέθους με τη χρήση αλγορίθμων βελτιστοποίησης. Ο βέλτιστος σχεδιασμός έχει να κάνει με τη μείωση του βάρους- και κατά συνέπεια και του κόστους- της κατασκευής, με σκοπό να αποφευχθεί η περιοχή συχνοτήτων συντονισμού, και την επίτευξη μέγιστης ακαμψίας της κατασκευής με περιορισμούς για τις παραμορφώσεις και τις τάσεις.

Η μέθοδος πεπερασμένων στοιχείων είναι μια αριθμητική μέθοδος (δηλ. μέθοδος υπολογισμού με χρήση Η/Υ) για τον υπολογισμό προσεγγιστικών λύσεων μερικών διαφορικών εξισώσεων. Αυτή η μέθοδος είναι μεν προσεγγιστική, αλλά μπορεί να δώσει αξιόπιστα αποτελέσματα και έχει το πλεονέκτημα ότι μπορεί να εφαρμοστεί σε όλα τα προβλήματα

Οι αλγόριθμοι βελτιστοποίησης διακρίνονται σε δύο βασικές κατηγορίες:

- **Αιτιοκρατικές Μέθοδοι** όπου ο αλγόριθμος κινείται προς συνεχώς καλύτερες λύσεις με αυστηρά ντετερμινιστικό τρόπο αξιοποιώντας πληροφορία από την με την γενική έννοια παράγωγο της αντικειμενικής συνάρτησης. Χαρακτηριστική και πολύ απλή περίπτωση αιτιοκρατικού αλγορίθμου αποτελεί η μέθοδος *Απότομης Καθόδου (Steepest Descent)*.
- **Στοχαστικές μέθοδοι** των οποίων η συμπεριφορά παρουσιάζει συγκροτημένη τυχαιότητα και δεν είναι απόλυτα προβλέψιμη. Γνωστότερο παράδειγμα αποτελούν ίσως οι *Γενετικοί Αλγόριθμοι* (GA) και οι παραλλαγές τους.

1.Εισαγωγή

Σε αυτό το κεφάλαιο θα προσπαθήσουμε να ορίσουμε τις σχετικές με το θέμα μας έννοιες, δηλαδή αυτή του Βέλτιστου Σχεδιασμού, των Αλγορίθμων Βελτιστοποίησης, και τέλος της θεωρίας των πεπερασμένων στοιχείων.

1.1Βέλτιστος Σχεδιασμός Κατασκευών

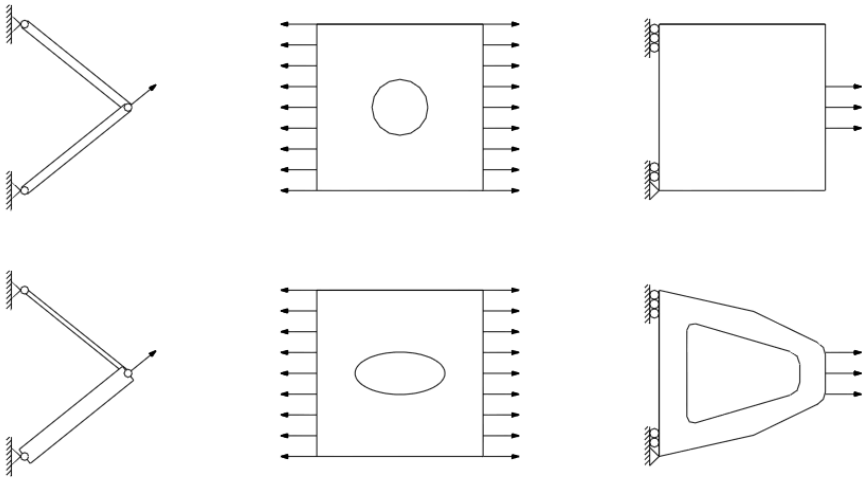
Βέλτιστος ονομάζεται γενικά ο σχεδιασμός μίας κατασκευής ο οποίος ικανοποιεί τις κατασκευαστικές προδιαγραφές και τις λειτουργικές απαιτήσεις, ενώ ταυτόχρονα ελαχιστοποιεί συγκεκριμένα κριτήρια, όπως (συνηθέστερα) το κόστος και το βάρος της κατασκευής. Ως "κατασκευή", βλέποντάς την από τη σκοπιά της μηχανικής των κατασκευών, ορίζεται εκείνη η διευθέτηση των υλικών που εξυπηρετεί στην ανάληψη μηχανικών φορτίων. Η λέξη "εξυπηρετεί" στον παραπάνω ορισμό, έχει την έννοια της αντοχής και της λειτουργικότητας. Ο ορισμός αυτός είναι ιδιαίτερα γενικός, συμπεριλαμβάνει δε όλα τα γνωστά μας ήδη κατασκευών, όπως κτίρια, γέφυρες, αεροσκάφη, σκελετούς ζώντων οργανισμών και φυτών κ.α. Ο όρος "διευθέτηση των υλικών", υπονοεί ότι στο εξής δεν υφίσταται διαχωρισμός των υλικών από τη μία και των κατασκευών από την άλλη, καθώς τα ίδια τα υλικά, διευθετημένα κατά τον κατάλληλο τρόπο (σχεδιασμός), αποτελούν την κατασκευή στο σύνολό της.

Στην πράξη, και με πολύ απλά λόγια, μία κατασκευή μπορεί να θεωρηθεί σχεδιασμένη κατά βέλτιστο τρόπο, αν ικανοποιεί όλες τις κατασκευαστικές απαιτήσεις (στέκει, αποκρίνεται ικανοποιητικά στις δυσμενέστερες φορτίσεις σχεδιασμού), και ταυτόχρονα είναι κατά το δυνατόν οικονομική, δεν έχουν γίνει επομένως άσκοπες σπατάλες. Η βελτιστοποίηση είναι έτσι ο κύριος στόχος του καλού σχεδιαστή – μηχανικού, ο οποίος καλείται να συμβιβάσει με τον καλύτερο τρόπο αυτές τις αντικρουόμενες απαιτήσεις: Αντοχή –λειτουργικότητα και οικονομία. Σκοπός του μηχανικού λοιπόν ήταν ανέκαθεν η αυστηρή τήρηση των παραπάνω περιορισμών, σε συνδυασμό όμως με την ικανοποίηση – κατά το δυνατό – και άλλων, κυρίως οικονομικής φύσεως απαιτήσεων με τελικό στόχο τη μέγιστη δυνατή απόδοση του έργου.[(1)]

Η βελτιστοποίηση κατασκευών χωρίζεται σε τρεις μεγάλες κατηγορίες:

- Τη βελτιστοποίηση των μεγεθών των **διατομών** (*Sizing optimization*),
- τη βελτιστοποίηση του **σχήματος** (*Shape optimization*) και
- τη βελτιστοποίηση της **τοπολογίας** (*Topology optimization*).

Σήμερα υπάρχει η τάση ένα γενικό πρόβλημα βελτιστοποίησης να εντοπίζεται αρχικά ως πρόβλημα εύρεσης της βέλτιστης τοπολογίας και στη συνέχεια ως πρόβλημα εύρεσης του βέλτιστου σχήματος ή εύρεσης των βέλτιστων διατομών. Στο σχήμα 1.1 που ακολουθεί παρουσιάζονται οι τρεις διαφορετικοί τύποι βελτιστοποίησης. Για την κάθε περίπτωση, παρουσιάζεται ο αρχικός φορέας και ο φορέας μετά το πέρας της διαδικασίας βελτιστοποίησης. Στις επόμενες παραγράφους, θα γίνει μια σύντομη αναφορά στις τρεις κατηγορίες βελτιστοποίησης. [(2)]



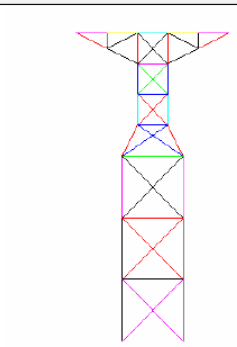
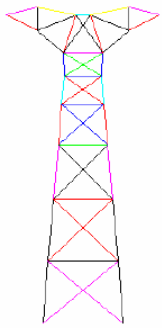
Σχήμα 1.1 Είδη βελτιστοποίησης κατασκευών:
(α) βαθμωτών μεγεθών (διατομών) , (β) σχήματος, (γ) τοπολογίας

1.1.1Βελτιστοποίηση τοπολογίας κατασκευών (Topology optimization)

Στον σχεδιασμό των κατασκευών είναι απαραίτητο να καθορισθεί μία όσο το δυνατόν καλύτερη τοπολογία (Topology) ή "δομή" (layout) της κατασκευής, έτσι ώστε να γίνει αποτελεσματικότερη στην ανάληψη φορτίων και οικονομικότερη για τα υλικά. Στον όρο "δομή" της κατασκευής συμπεριλαμβάνεται κάθε είδους πληροφορία που αφορά την τοπολογία, το σχήμα και το μέγεθός της.

Η τοπολογία μίας κατασκευής συνήθως είναι προκαθορισμένη από τις απαιτήσεις του προβλήματος και τους κατασκευαστικούς περιορισμούς, ή προέρχεται από κάποιο αρχικό σχεδιασμό του μηχανικού. Οι αλγόριθμοι βελτιστοποίησης της τοπολογίας, οι οποίοι άρχισαν να αναπτύσσονται μόλις τα τελευταία δεκαπέντε χρόνια, είναι εργαλεία που βοηθάνε τον σχεδιαστή μηχανικό να επιτύχει την αποτελεσματική χρήση των υλικών για να καλύψει τις ανάγκες του προβλήματος. [(3)]

Αρχικά ορίζεται ο χώρος σχεδιασμού ή αναφοράς (reference domain), ο τύπος υλικού, οι συνθήκες στήριξης και οι φορτίσεις και έπειτα γίνεται η ανάλυση του φορέα και η επαναληπτική διαδικασία που θα οδηγήσει στη βέλτιστη τοπολογία, δηλαδή στην καλύτερη δυνατή κατανομή υλικού στην κατασκευή. Έχοντας επιτύχει τη βέλτιστη τοπολογία το επόμενο βήμα προς τον βέλτιστο σχεδιασμό είναι να "καθαριστεί" το σχήμα και το μέγεθος του φορέα με κατάλληλους αλγόριθμους βελτιστοποίησης. Οι αλγόριθμοι αυτοί, που έχουν να κάνουν με τη βελτιστοποίηση του σχήματος πλέον της κατασκευής, έρχονται να δράσουν συμπληρωματικά με τη βελτιστοποίηση της τοπολογίας, ώστε να επιτευχθεί τελικά η βέλτιστη δομή της κατασκευής.

	Αρχικό στάδιο	Τελικό στάδιο
Combined sizing and shape optimization		

Η Τοπολογική Βελτιστοποίηση κατασκευών διαφέρει από την απλή βελτιστοποίηση όσον αφορά την πολυπλοκότητα της. Υπάρχουν δύο πιθανά προβλήματα που δικαιολογούν αυτό το γεγονός. Το πρώτο είναι ότι τα μοντέλα από μόνα τους διαφοροποιούνται κατά τη διάρκεια της σχεδιαστικής διαδικασίας και δεύτερο είναι ότι ο αριθμός των συνδέσεων των στοιχείων αυξάνεται με μεγάλο ρυθμό καθώς αυξάνουμε του κόμβους σύνδεσης.

Ύστερα από μελέτες, τα πλαίσια με μεταλλικές δοκούς είναι τα πλέον κατάλληλα για την εφαρμογή της Τοπολογικής βελτιστοποίησης.[1]

1.1.2 Βελτιστοποίηση σχήματος κατασκευών (Shape optimization)

Η βελτιστοποίηση σχήματος (Shape optimization) είναι ένα σχετικά καινούργιο πεδίο έρευνας στον χώρο του βέλτιστου σχεδιασμού κατασκευών. Με τη βελτιστοποίηση σχήματος, μεταβάλλονται το εσωτερικό και το εξωτερικό περίγραμμα της κατασκευής ώστε να επιτευχθεί η καλύτερη δυνατή κατανομή των τάσεων στο εσωτερικό της. Οι πρώτες μεθοδολογίες καταγράφονται σποραδικά στη δεκαετία του 1970 παίρνοντας στοιχεία από τη βελτιστοποίηση βαθμωτών μεγεθών κατασκευών.

Οι πρώτες δομημένες τεχνικές πάνω στο αντικείμενο άρχισαν να υλοποιούνται στη δεκαετία του 1980, οπότε έγιναν και οι πρώτες απόπειρες καταγραφής της υπάρχουσας βιβλιογραφίας στον τομέα αυτόν. Τα τελευταία χρόνια και με τη βοήθεια που παρέχουν τα εξελιγμένα πακέτα λογισμικού γεωμετρικής μοντελοποίησης και γραφικής αναπαράστασης έχουν αναπτυχθεί αρκετά προχωρημένα αυτόματα συστήματα βελτιστοποίησης σχήματος που καλύπτουν μια ευρεία γκάμα κατασκευαστικών προβλημάτων.

Το πρόβλημα της βελτιστοποίησης του σχήματος των κατασκευών είναι ιδιαίτερα δύσκολο, καθώς η γεωμετρία και το μοντέλο ανάλυσης μεταβάλλονται συνεχώς, σε αντίθεση με το πρόβλημα της βελτιστοποίησης διατομών και ιδιοτήτων υλικών, όπου στόχος είναι η ελαχιστοποίηση του βάρους κάποιου φορέα συγκεκριμένου σχήματος και τοπολογίας. Η βελτιστοποίηση σχήματος όμως, παρ' όλη την πολυπλοκότητά της, είναι πολύ πιο αποτελεσματική, ειδικά σε προβλήματα βελτιστοποίησης συνεχών κατασκευών.

Ας θεωρήσουμε, λόγου χάρη, μία πλάκα με μία οπή στο κέντρο της, στα χείλη της οποίας αναμένεται συγκέντρωση τάσεων. Βελτιστοποίηση της πλάκας μόνο ως προς το πάχος (βαθμωτό μέγεθος) απλά θα επιφέρει ένα αυξομειούμενο πάχος, ιδιαίτερα μεγάλο γύρω από την οπή. Μια τέτοια λύση είναι προβληματική τόσο κατασκευαστικά όσο και από άποψη λειτουργικότητας. Αντίθετα, η αντιμετώπιση του ίδιου προβλήματος με τεχνικές βελτιστοποίησης σχήματος θα επηρεάσει μόνο το σύνορο της οπής κάνοντάς το ομαλό με μορφή έλλειψης (σχήμα 1.1β).

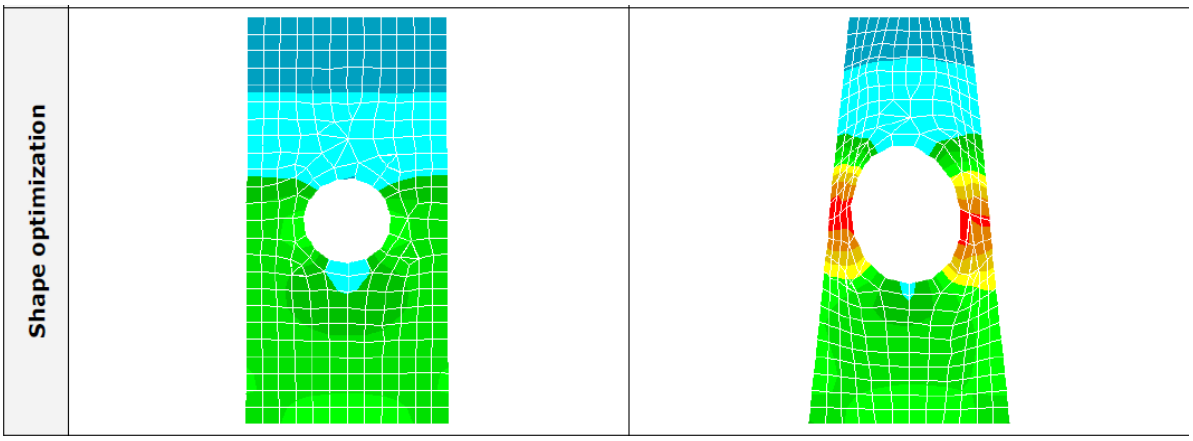
Συνήθως η βελτιστοποίηση του σχήματος μίας κατασκευής είναι το δεύτερο στάδιο του βέλτιστου σχεδιασμού της, λαμβάνοντας ως δεδομένο ότι έχει προηγηθεί ένα πρώτο στάδιο όπου έχει ευρεθεί η τοπολογία της κατασκευής, είτε αυτόματα μέσω κάποιου αλγόριθμου βελτιστοποίησης, είτε εμπειρικά από τον σχεδιαστή. Υπάρχουν ωστόσο και μέθοδοι οι οποίες βελτιώνουν ταυτόχρονα και την τοπολογία και το σχήμα της κατασκευής.

Στόχος των μεθόδων βελτιστοποίησης σχήματος είναι να βελτιώσουν το σχήμα της κατασκευής και κατά συνέπεια να γίνει πιο οικονομική και λειτουργική η χρήση της,

λαμβάνοντας φυσικά υπόψη διάφορους περιορισμούς που τίθενται για το συγκεκριμένο πρόβλημα. Συνήθως αυτό γίνεται σε τρεις φάσεις σε κάθε βήμα της διαδικασίας βελτιστοποίησης του σχήματος:

i. Καθορισμός και έλεγχος του σχήματος,
ii. Ανάλυση της απόκρισης της κατασκευής (συμπεριλαμβανομένης και της ανάλυσης ευαισθησίας αυτής, αν ο χρησιμοποιούμενος αλγόριθμος βελτιστοποίησης το απαιτεί),
iii. Επίλυση του μαθηματικού προβλήματος της βελτιστοποίησης με τη χρήση κάποιου ντετερμινιστικού μαθηματικού ή πιθανοτικού αλγορίθμου βελτιστοποίησης.

Η αυτοματοποιημένη βελτιστοποίηση σχήματος παρέχει στο σχεδιαστή ένα αρκετά γενικό και ευέλικτο πλαίσιο, έτσι ώστε διαλέγοντας κατάλληλα τις παραμέτρους σχεδιασμού, την αντικειμενική συνάρτηση και τις συναρτήσεις περιορισμών, να μορφώσει το καλύτερο δυνατό μοντέλο για τη βελτιστοποίηση οποιουδήποτε σχήματος.[2]



1.1.3 Βελτιστοποίηση των μεγεθών των διατομών (Sizing optimization)

Το πρόβλημα της βελτιστοποίησης των μεγεθών των διατομών ενός φορέα συγκεκριμένης τοπολογίας και σχήματος είναι το πρώτο πρόβλημα βελτιστοποίησης με το οποίο ασχολήθηκαν εκτενώς οι ερευνητές. Στην αρχή ως παράμετροι σχεδιασμού χρησιμοποιούνταν οι διαστάσεις των διατομών δικτυωτών κατασκευών ή πλαισίων και αργότερα τα πάχη πλακών και κελυφών. Η συνηθέστερη αντικειμενική συνάρτηση για τέτοια προβλήματα ήταν και παραμένει το βάρος της κατασκευής, το οποίο ζητείται να ελαχιστοποιηθεί υπό κάποιους περιορισμούς (π.χ. στις τάσεις κάποιων μελών, στις μετατοπίσεις κάποιων κόμβων, κ.α.).

Η διαδικασία βελτιστοποίησης των βαθμωτών μεγεθών, ακολουθεί γενικά τα ακόλουθα βήματα:

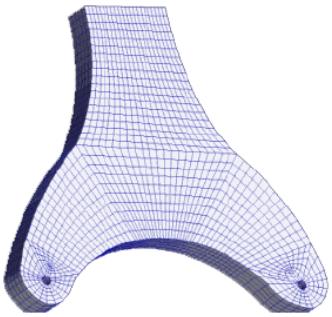
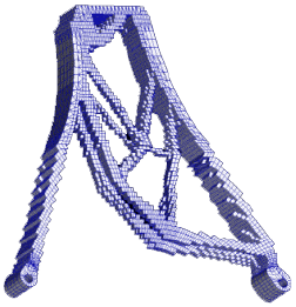
(i) Ορίζεται το δομοστατικό μοντέλο, η γεωμετρία και τα φορτία σχεδιασμού της κατασκευής.

(ii) Ορίζεται το μοντέλο βελτιστοποίησης, δηλαδή γίνεται η επιλογή των παραμέτρων σχεδιασμού (οι οποίες ενδέχεται να είναι εξαρτώμενες η μία από την άλλη και η επιλογή των συναρτήσεων περιορισμών.

(iii) Υπολογίζονται οι τάσεις και οι μετατοπίσεις στα μέλη της κατασκευής για τον εκάστοτε σχεδιασμό με τη μέθοδο των πεπερασμένων στοιχείων, ώστε να γίνει ο έλεγχος των περιορισμών

(iv) Στην περίπτωση που χρησιμοποιείται μαθηματικός βελτιστοποιητής (SQP), διενεργείται ανάλυση ευαισθησίας των περιορισμών και της αντικειμενικής συνάρτησης για μικρές μεταβολές των παραμέτρων σχεδιασμού.

(v) Αν τα κριτήρια τερματισμού που έχουν τεθεί ικανοποιούνται, τότε θεωρείται ότι έχει εντοπιστεί το βέλτιστο και η διαδικασία τερματίζεται, διαφορετικά ο βελτιστοποιητής επιλέγει νέους υποψήφιους βέλτιστους σχεδιασμούς και η διαδικασία επιστρέφει στο βήμα (iii).

Topology optimization		

1.2 Μέθοδοι βέλτιστου σχεδιασμού

Οι αλγόριθμοι βελτιστοποίησης μπορούν να χωριστούν γενικά σε δύο μεγάλες κατηγορίες:

- Τις μαθηματικές ή αιτιοκρατικές μεθόδους, και
- Τις εξελικτικές ή δαρβίνειες μεθόδους.

1.2.1 Μαθηματικές μέθοδοι βελτιστοποίησης

Οι μαθηματικές μέθοδοι βελτιστοποίησης προέρχονται από τις επιστημονικές περιοχές των οικονομικών μαθηματικών και της επιχειρησιακής έρευνας και ήταν οι πρώτες που εφαρμόστηκαν σε προβλήματα βέλτιστου σχεδιασμού κατασκευών.

Ο κλάδος της επιστήμης των μαθηματικών που έχει ως αντικείμενο τη βελτιστοποίηση της απόδοσης ενός συστήματος που υπόκειται σε ορισμένους περιορισμούς, ονομάζεται *Μαθηματικός Προγραμματισμός* (Mathematical Programming). Φυσικά απαραίτητη προϋπόθεση για να λυθεί ένα πρόβλημα με μεθόδους μαθηματικού προγραμματισμού είναι να μπορεί το πρόβλημα αυτό να μετατραπεί σε αυστηρά μαθηματική μορφή. Ο όρος μαθηματικός προγραμματισμός, που δεν πρέπει να συγχέεται με την ανάπτυξη προγραμμάτων λογισμικού, χρησιμοποιήθηκε πολύ πριν την ευρεία εφαρμογή των ηλεκτρονικών υπολογιστών. Ο όρος αυτός αφορά τη μελέτη προβλημάτων βελτιστοποίησης, τις μαθηματικές ιδιότητες και την ανάπτυξη αλγορίθμων βέλτιστου σχεδιασμού.

Οι τεχνικές βελτιστοποίησης που βασίζονται στις αρχές του μαθηματικού προγραμματισμού μπορούν γενικά να ταξινομηθούν σε πέντε μεγάλες κατηγορίες:

i. **Γραμμικός Προγραμματισμός (Linear Programming – LP).**

Οι μέθοδοι γραμμικού προγραμματισμού αντιμετωπίζουν προβλήματα στα οποία τόσο η αντικειμενική συνάρτηση όσο και οι συναρτήσεις περιορισμών είναι γραμμικές συναρτήσεις των μεταβλητών σχεδιασμού. Σε αυτήν την περίπτωση η βέλτιστη λύση βρίσκεται επί του συνόρου μίας ή περισσοτέρων συναρτήσεων περιορισμού. Σε προβλήματα αυτού του είδους (κυρτά) ένα τοπικό ελάχιστο είναι οπωσδήποτε και καθολικό ελάχιστο του προβλήματος

ii. **Μη Γραμμικός Προγραμματισμός Non Linear Programming – NLP).**

Είναι οι πιο διαδεδομένες τεχνικές μαθηματικού προγραμματισμού, ιδιαίτερα σε προβλήματα βελτιστοποίησης κατασκευών, αφού αντιμετωπίζουν γενικά όλες τις περιπτώσεις όπου η αντικειμενική συνάρτηση αλλά και οι συναρτήσεις περιορισμών

είναι μη γραμμικές συναρτήσεις των μεταβλητών σχεδιασμού. Σε αυτή την περίπτωση (μη κυρτό πρόβλημα) η εύρεση ενός τοπικού ελαχίστου δεν πιστοποιεί την εύρεση ενός καθολικού ελαχίστου.

iii. **Ακέραιος Προγραμματισμός (Integer Programming – IP).**

Οι μέθοδοι αυτές αντιμετωπίζουν προβλήματα στα οποία οι παράμετροι σχεδιασμού δεν είναι συνεχείς, αλλά παίρνουν διακριτές τιμές από κάποιο συγκεκριμένο σύνολο τιμών. Συνήθως οι τιμές αυτές είναι ακέραιες, εξ' ου και το όνομα "Ακέραιος Προγραμματισμός". Επίσης υπάρχουν και περιπτώσεις "Μικτού Ακέραιου Προγραμματισμού" (Mixed Integer Programming), όπου κάποιες από τις παραμέτρους σχεδιασμού είναι διακριτές και άλλες είναι συνεχείς.

iv. **Δυναμικός προγραμματισμός (Dynamic Programming – DP).**

Κύριος στόχος αυτών των μεθόδων είναι να διασπασθεί ένα σχετικά μεγάλο πρόβλημα βελτιστοποίησης σε μικρότερα τα οποία μπορούν να αντιμετωπισθούν ως ξεχωριστά προβλήματα βέλτιστου σχεδιασμού. Κάθε υποπρόβλημα περιέχει μέρος από τα στοιχεία του καθολικού προβλήματος και μπορεί να επιλυθεί με κάποια από τις προαναφερθείσες μεθοδολογίες. Έτσι στην ουσία ο δυναμικός προγραμματισμός δεν αποτελεί μια ξεχωριστή τεχνική μαθηματικής βελτιστοποίησης αλλά μάλλον μία τεχνική διαμερισμού μεγάλων προβλημάτων βελτιστοποίησης σε μικρότερα.

v. **Γεωμετρικός προγραμματισμός (Geometric Programming – GP).**

Αναφέρεται σε μία ειδική κατηγορία προβλημάτων όπου οι συναρτήσεις περιορισμών αλλά και η αντικειμενική συνάρτηση είναι πολυωνυμικής μορφής συναρτήσεις των παραμέτρων σχεδιασμού. Σε προβλήματα αυτού του είδους πρέπει να ισχύει πάντα η αυστηρή προϋπόθεση ότι οι παράμετροι σχεδιασμού λαμβάνουν πάντα θετικές τιμές.

1.2.2 Εξελικτικές μέθοδοι βελτιστοποίησης

Ήδη από τη δεκαετία του 1960 είχαν αρχίσει να εφαρμόζονται τεχνικές βασιζόμενες σε τυχηματικές διαδικασίες δειγματοληψίας που προσομοίαζαν γενετικές διεργασίες και πρότυπα για την επίλυση προβλημάτων επιχειρησιακής έρευνας, οικονομικών μαθηματικών καθώς και υπολογιστικής μηχανικής. Οι πιο διαδεδομένες από τις μεθόδους αυτές είναι οι επονομαζόμενες *Δαρβίνειες Μέθοδοι* οι οποίες οφείλουν την ονομασία τους στο γεγονός ότι *μιμούνται* τη διαδικασία της εξέλιξης των ειδών στη φύση, όπως την παρουσίασε πρώτος ο Κάρολος Δαρβίνος.

Οι Δαρβίνειες μέθοδοι ή διαφορετικά *Μέθοδοι Εξελικτικών Αλγορίθμων* (Evolutionary Algorithms) σε αντίθεση με τις μεθόδους μαθηματικού προγραμματισμού, χρησιμοποιούν έναν "πληθυσμό" πιθανών λύσεων του προβλήματος. Οι λύσεις αυτές είναι ανεξάρτητες μεταξύ τους και μπορούν να αντιμετωπιστούν επίσης ανεξάρτητα η μία από την άλλη, πράγμα που καθιστά ιδιαίτερα αποδοτική την εφαρμογή των μεθόδων αυτών σε παράλληλο υπολογιστικό περιβάλλον. Ο αρχικός πληθυσμός επιλέγεται με τυχαίο τρόπο και στη συνέχεια με τη χρήση κατάλληλων γενετικών μηχανισμών όπως είναι η *Επιλογή* (Selection), ο *Ανασυνδυασμός* (Recombination) και η *Μετάλλαξη* (Mutation), ο πληθυσμός εξελίσσεται κινούμενος σε όλο και καλύτερες περιοχές του χώρου αναζήτησης και μέσω της αρχής της επικράτησης του ισχυρότερου (Survival of the fittest) επιτυγχάνεται ο εντοπισμός της βέλτιστης λύσης.

Οι μέθοδοι αυτές εφαρμόζονται σήμερα συχνά στον βέλτιστο σχεδιασμό των κατασκευών, με ιδιαίτερα ενθαρρυντικά αποτελέσματα. Το βασικό τους πλεονέκτημα είναι ότι λόγω της τυχαιότητας στην διερεύνηση του χώρου αναζήτησης, έχουν μεγαλύτερες πιθανότητες για την εύρεση του απόλυτα βέλτιστου σχεδιασμού σε σχέση με τις μαθηματικές μεθόδους οι οποίες χρησιμοποιούν νομοτελειακά καθορισμένες σχέσεις. Από την άλλη, το κυριότερο μειονέκτημα των μεθόδων αυτών είναι ο μεγάλος αριθμός των επαναλήψεων που απαιτούνται κατά τη διαδικασία βελτιστοποίησης, παρόλο που το υπολογιστικό κόστος της κάθε μίας από αυτές είναι σχετικά μικρό, αφού δεν εμπεριέχεται ανάλυση ευαισθησίας.

Οι σημαντικότερες από τις μεθόδους αυτές είναι:

(i) **Η μέθοδος των *Γενετικών αλγορίθμων* (Genetic Algorithms – GA).**

Η μέθοδος των ΓΑ αναπτύχθηκε από τον J. Holland και τους συνεργάτες του στο πανεπιστήμιο του Michigan. Πρόκειται για μία από τις κυριότερες εξελικτικές μεθόδους, που λόγω της απλής δομής και της ευχρηστίας της έχει βρει εφαρμογή στη βελτιστοποίηση κατασκευών. Οι ΓΑ βασίζονται στην τυχαία ανασυγκρότηση ενός πληθυσμού και την αξιολόγηση των μελών του πληθυσμού μέσω μίας συνάρτησης

ποιότητας (Fitness ή Quality function), που σε προβλήματα βελτιστοποίησης ισούται με το άθροισμα της αντικειμενικής συνάρτησης και των παραβιάσεων των συναρτήσεων περιορισμού. Τα μέλη του πληθυσμού των ΓΑ κωδικοποιούνται σε δυαδική (binary) μορφή, ώστε να προσομοιάζουν καλύτερα τα βιολογικά χρωμοσώματα (strings). Με τη χρήση των γενετικών τελεστών του συνδυασμού και της μετάλλαξης, οι ΓΑ επιλέγουν το καλύτερο μέλος του πληθυσμού, δηλαδή αυτό που δίνει την καλύτερη τιμή στη συνάρτηση ποιότητας.

(ii) Η μέθοδος των *Στρατηγικών Εξέλιξης* (Evolution Strategies – ES).

Η μέθοδος των ΣΕ αναπτύχθηκε αρχικά από τον I. Rechenberg και εν συνεχεία από τον H.P. Schwefel και την ερευνητική του ομάδα.

(iii) Η μέθοδος του *Εξελικτικού Προγραμματισμού* (Evolutionary Programming – EP).

Η μέθοδος του ΕΠ αναπτύχθηκε από τους L.J. Fogel et al. και βασίζεται αποκλειστικά την αρχή της επιλογής του καλύτερου και όχι στη διαδικασία της αναπαραγωγής των ειδών. Για τον λόγο αυτό έχει μειωμένες δυνατότητες και περιορισμένη αποτελεσματικότητα.

(iv) Η μέθοδος του *Γενετικού Προγραμματισμού* (Genetic Programming – GP).

Η μέθοδος του ΓΠ, η οποία αναπτύχθηκε από τον J.R. Koza, διαφέρει κατά πολύ από τις προαναφερθείσες εξελικτικές μεθοδολογίες και απλά έχει ως στόχο την αυτόματη παραγωγή προγραμμάτων Η/Υ για την αντιμετώπιση συγκεκριμένων προβλημάτων (π.χ. γεννήτρια τυχαίων αριθμών).

Μία επίσης τυχηματική αλλά διαφορετικής δομής τεχνική είναι η μέθοδος της *Προσομοίωσης Ανόπτησης* (Simulated Annealing). Η μέθοδος αυτή ακολουθεί τους τυχηματικούς νόμους που διέπουν την κίνηση των μορίων στο εσωτερικό ενός μεταλλικού σώματος κατά τη σταδιακή ψύξη αυτού. Τα μόρια κινούνται ώστε να εντοπιστεί η κατάσταση με την ελάχιστη δυνατή δυναμική ενέργεια για την τρέχουσα θερμοκρασία, ώστε τελικά στην χαμηλότερη θερμοκρασία να επιτευχθεί η ελάχιστη όλων των δυνατών ενεργειακά καταστάσεων.[4]

1.3 Μέθοδος πεπερασμένων στοιχείων

Στη μηχανική, η σχεδίαση μέσω Η/Υ (Computer Aided Design-CAD) χρησιμοποιείται στη σχεδίαση των προηγμένων κατασκευαστικών συστημάτων. Οι τεχνικές προσομοίωσης μέσω ηλεκτρονικών υπολογιστών χρησιμοποιούνται συχνά για να υπολογιστούν οι μετατοπίσεις, οι τάσεις, οι παραμορφώσεις, οι φυσικές συχνότητες κλπ. στις διάφορες κατασκευές για δεδομένες αρχικές συνθήκες, τοπολογίες, ιδιότητες υλικών και καταστάσεις φόρτισης. Αυτού του είδους τα προβλήματα καλούνται ευθύ προβλήματα και συχνά καθορίζονται από συνήθεις ή μερικές διαφορικές εξισώσεις (Ordinary Differential Equations – ODE ή Partial Differential Equations - PDE) με άγνωστες τις μεταβλητές του πεδίου (field variables).

Για τα κατασκευαστικά προβλήματα, οι μεταβλητές πεδίου είναι κυρίως οι μετατοπίσεις, ενώ οι σταθερές των διαφορικών και το πεδίο ορισμού του προβλήματος είναι γνωστά a priori. Η πηγή ή η αιτία του προβλήματος ή του φαινομένου που ορίζουν οι διαφορικές και σχετικές αρχικές και οριακές συνθήκες είναι επίσης γνωστές. Για την επίλυση ενός ευθέως προβλήματος, χρειάζεται ουσιαστικά η επίλυση των διαφορικών εξισώσεων που ορίζονται από το πρόβλημα.

Πολλές τεχνικές επίλυσης, κυρίως υπολογιστικές, έχουν αναπτυχθεί όπως:

- Μέθοδος των πεπερασμένων διαφορών (Finite Deference Method -FDM)
- Μέθοδος των πεπερασμένων στοιχείων (Finite Element Method - FEM)
- Μέθοδος στοιχείων λωρίδας (Strip Element Method - SEM)
- Μέθοδος των συνοριακών στοιχείων (Boundary Element Method - BEM)
- Συνδυασμός FEM/BEM
- Μέθοδοι χωρίς πλέγμα (Mesh- free methods)
- Επιλύτες διάδοσης κύματος (Wave propagation solvers)

Οι μέθοδοι αυτές, αποτελούν πλέον την καθιερωμένη προσέγγιση για την επίλυση ευθέων προβλημάτων. Ωστόσο, οι μέθοδοι επίλυσης χωρίς πλέγμα είναι ακόμα σε πρωταρχικό στάδιο και χρήζουν περαιτέρω έρευνας. Χρησιμοποιώντας αυτές τις μεθόδους, λαμβάνονται ως έξοδοι οι μετατοπίσεις της κατασκευής και εν συνεχεία οι τάσεις και οι παραμορφώσεις της, μόνο εάν είναι γνωστές οι ιδιότητες του υλικού, οι φορτίσεις, οι αρχικές και οριακές συνθήκες και η γεωμετρία της κατασκευής (είσοδοι του προβλήματος).

Η μέθοδος των Πεπερασμένων Στοιχείων είναι μια εξέλιξη των μητρωϊκων μεθόδων που έγινε από επιστήμονες όπως ο Αργύρης Ι., ο Clough, ο Ritz και άλλοι. Οι βασικές ιδέες προήλθαν στις αρχές της δεκαετίας του 40, από εξελίξεις στην δομική ανάλυση αεροσκαφών. Αρχικά ο Hrenikoff χρησιμοποίησε τη "Μέθοδο των δικτυωμάτων" , αργότερα Ο Turner δημιούργησαν μητρώα ακαμψίας για δικτυώματα, δοκούς και άλλα στοιχεία.

Ο όρος Πεπερασμένα στοιχεία χρησιμοποιήθηκε το 1960. Οι μαθηματικές, βέβαια, βάσεις για την σημερινή μορφή της μεθόδου μπήκαν την δεκαετία του 70. Πλέον αποτελεί ένα ισχυρό εργαλείο για την αριθμητική επίλυση ενός μεγάλου φάσματος προβλημάτων μηχανικού. Οι εφαρμογές εκτείνονται από την παραμόρφωση και ανάλυση τάσεων σε αυτοκίνητα, αεροπλάνα, κτίρια και γέφυρες, μέχρι την ανάλυση πεδίων ροής θερμότητας, ροής υγρών, μαγνητικής ροής, κ.α. Με την εξέλιξη των υπολογιστικών συστημάτων και των συστημάτων CAD, σύνθετα προβλήματα μπορούν να μοντελοποιηθούν πολύ εύκολα. Με αυτή τη μέθοδο μια πολύπλοκη περιοχή, διακριτοποιείται σε απλά γεωμετρικά σχήματα, τα οποία ονομάζονται Πεπερασμένα Στοιχεία (Finite Elements). Μια διαδικασία σύνθεσης, η οποία θεωρεί φορτία και περιορισμούς, έχει ως αποτέλεσμα ένα σύνολο εξισώσεων. Η επίλυση αυτών, δίνει κατά προσέγγιση(με αρκετά μεγάλη ακρίβεια) τη συμπεριφορά του αρχικού πολύπλοκου μοντέλου.

Μια άλλη κατηγορία πρακτικών προβλημάτων, που παρουσιάζεται αρκετά συχνά είναι τα αντίστροφα προβλήματα. Στα αντίστροφα προβλήματα, τα αποτελέσματα ή οι έξοδοι (μετατόπιση, ταχύτητα, επιτάχυνση, φυσική συχνότητα κτλ.) μπορεί να είναι γνωστά μεγέθη (π.χ. μέσω πειραμάτων). Όμως, οι ιδιότητες του υλικού, οι φορτίσεις, οι αρχικές και οριακές συνθήκες, η γεωμετρία της κατασκευής (είσοδοι) ή συνδυασμός αυτών πρέπει να καθοριστούν. Η επίλυση τέτοιου είδους προβλημάτων είναι παρά πολύ χρήσιμη για πολλές εφαρμογές της μηχανικής.

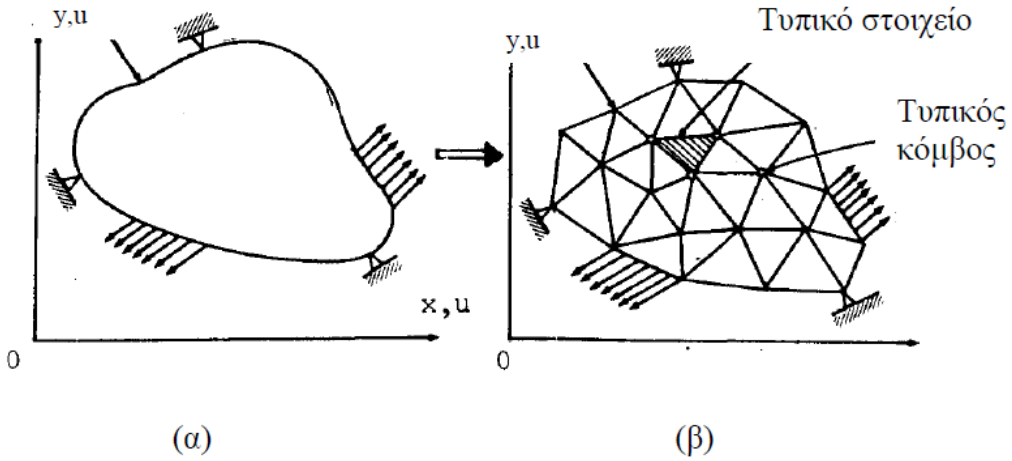
Η δυναμικότητα των προσεγγιστικών μεθόδων που βασίζονται στα ενεργειακά θεωρήματα είναι πολύ μεγάλη και δίνει λύσεις σε σημεία που οι υπόλοιπες μέθοδοι δεν μπορούν να δώσουν. Κύριο χαρακτηριστικό των προσεγγιστικών μεθόδων είναι η ανάγκη προσδιορισμού ενός παραδεκτού πεδίου (π.χ. μετατοπίσεων) με τη βοήθεια συναρτήσεων, που ορίζονται σε όλο το σώμα και ικανοποιούν ορισμένες συνθήκες στα σύνορα. Με τον τρόπο αυτό το αρχικό πρόβλημα του προσδιορισμού του πεδίου των μετατοπίσεων σε κάθε σημείο του σώματος (άπειρος βαθμός ελευθερίας κίνησης) μετασχηματίζεται σε ένα υποκατάστατο πρόβλημα προσδιορισμού πεπερασμένου αριθμού άγνωστων συντελεστών (πεπερασμένος βαθμός ελευθερίας).

Το υποκατάστατο πρόβλημα δίνει μια προσεγγιστική λύση στο αρχικό πρόβλημα. Η κατασκευή των συναρτήσεων, που ικανοποιούν τις συνθήκες στα σύνορα, είναι μια σχετικά εύκολη υπόθεση όταν το σύνορο του σώματος είναι απλό. Όταν όμως

το σύνορο του σώματος γίνεται πολύπλοκο, η ικανοποίηση των οριακών συνθηκών γίνεται σχεδόν αδύνατη. Αυτές ακριβώς τις δυσκολίες αποφεύγουμε με τη Μέθοδο των Πεπερασμένων Στοιχείων. Σύμφωνα με αυτή, το σώμα (Σχήμα 2.1(α)) χωρίζεται σ' έναν αριθμό περιοχών που λέγονται στοιχεία (Σχήμα 2.1(β)). Οι συναρτήσεις που χρησιμοποιούνται μέσα σε κάθε στοιχείο είναι απλές (συνήθως κάποιες πολυωνυμικές εκφράσεις).

Χρησιμοποιώντας όμως ένα μεγάλο αριθμό στοιχείων γίνεται δυνατό να αποκτήσουμε μια καλή προσέγγιση της πραγματικής κατάστασης. Το τελικό αποτέλεσμα είναι να αναχθεί πάλι το αρχικό πρόβλημα σ' ένα πρόβλημα με πεπερασμένο αριθμό βαθμών ελευθερίας κίνησης. Η μεθοδολογία αυτή, αν και μπορεί να υπαχθεί στις μεθοδολογίες Ritz ή Galerkin.

i. Οι συναρτήσεις είναι απλούστερες και δεν απαιτείται η ικανοποίηση κάποιων συνθηκών στα σύνορα του σώματος.
ii. Οι ολοκληρώσεις γίνονται σε κάθε στοιχείο χωριστά (οι συναρτήσεις είναι μηδέν έξω από κάθε στοιχείο). Έτσι, στις απλές περιπτώσεις το αποτέλεσμα προκύπτει εύκολα με αναλυτικό τρόπο, ενώ στις πιο σύνθετες μπορεί να χρησιμοποιηθεί αριθμητική ολοκλήρωση.



Σχήμα 2.1 Διαμέριση σώματος.
(α) Αρχικό σώμα. (β) Διαμέριση σώματος σε ένα πεπερασμένο αριθμό περιοχών.

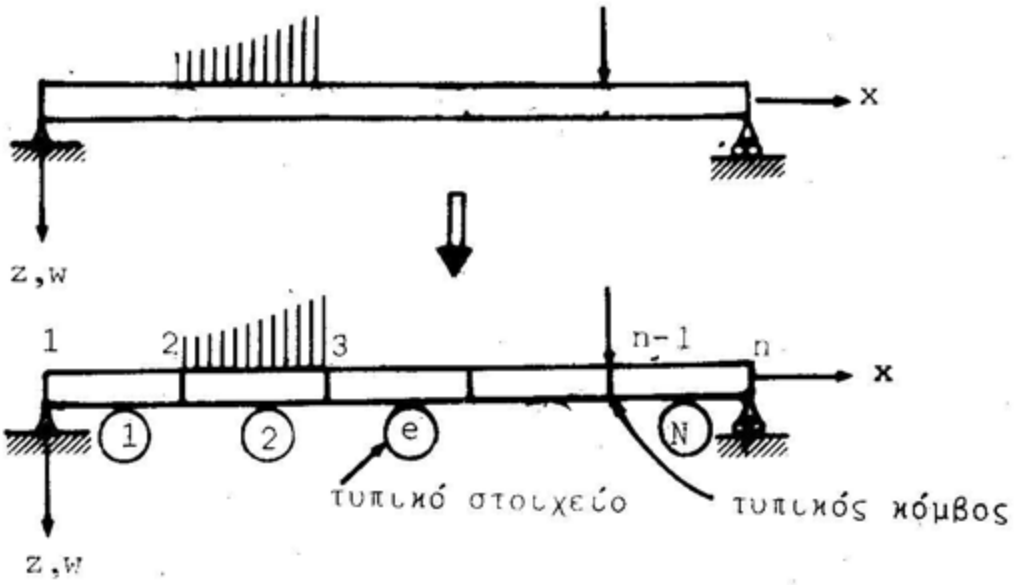
Από τα παραπάνω γίνεται αντιληπτό, ότι η μέθοδος των πεπερασμένων στοιχείων αποτελεί μια από τις πιο ισχυρές μεθόδους της αριθμητικής ανάλυσης για την επίλυση οριακών προβλημάτων, θα μπορούσε λοιπόν να γίνει μια γενική περιγραφή, που θα ίσχυε για κάθε διαφορική εξίσωση. Όμως μια τέτοια περιγραφή αφενός ξεφεύγει εντελώς από τα όρια αυτής της εργασίας, αφετέρου δεν είναι ιστορικά ακριβής.

Πραγματικά, η μέθοδος των πεπερασμένων στοιχείων αναπτύχθηκε από τους μηχανικούς στην προσπάθεια τους να επιλύσουν δύσκολα προβλήματα κατασκευών. Ο στόχος τους ήταν να χρησιμοποιήσουν μια τεχνική ανάλογη με αυτή που

χρησιμοποιούσαν στις ραβδωτές κατασκευές. Αυτό σημαίνει, ότι η μέθοδος των πεπερασμένων στοιχείων προέκυψε σαν εξέλιξη της μητρωϊκής ανάλυσης των κατασκευών. Σήμερα όμως, η μητρωϊκή ανάλυση των κατασκευών περιλαμβάνεται στις ειδικές εφαρμογές της μεθόδου των πεπερασμένων στοιχείων. Στην παράγραφο αυτή έχουμε σαν στόχο να δώσουμε μια μικρή εισαγωγή στο αντικείμενο.[2,5]

1.3.1 Ορισμός του καννάβου και γενικότητες

Έστω ένα ελαστικό σώμα που καταλαμβάνει τον χώρο *D* (Σχήμα 2.1(α)). Ο χώρος αυτός υποτίθεται ότι χωρίζεται με έναν κάνναβο ιδεατών επιφανειών ή γραμμών σε έναν αριθμό περιοχών (Σχήμα 2.1(β)), τα πεπερασμένα στοιχεία (α) και (β)). Στην περίπτωση που ο D είναι τρισδιάστατος, τα πεπερασμένα στοιχεία είναι τρισδιάστατα, ενώ όταν το σώμα είναι δισδιάστατο τα πεπερασμένα στοιχεία είναι δισδιάστατα. Στην περίπτωση των ραβδωτών κατασκευών έχουμε τα μονοδιάστατα στοιχεία. Σε μια κατασκευή είναι δυνατόν να χρειαστούν και τα τρία είδη στοιχείων.

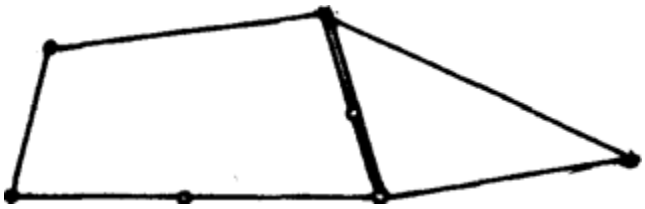


Σχήμα 2.2: Διαμέριση σώματος.
(α) Αρχικό σώμα. (β) Διαμέριση σώματος σε ένα πεπερασμένο αριθμό περιοχών.

Τα στοιχεία υποτίθεται ότι συνδέονται σ' ένα πεπερασμένο αριθμό σημείων, τους κόμβους (Σχήμα 2.1, , Σχήμα 2.3), (κόμβοι βέβαια μπορεί να υπάρχουν και στο εσωτερικό κάποιου στοιχείου, που δεν «συνδέονται» με τα γειτονικά στοιχεία). Κάθε κόμβος, ανάλογα με το είδος της κατασκευής, έχει κάποιες δυνατότητες μετατόπισης, κάποιους δηλαδή βαθμούς ελευθερίας (β.ε.) κίνησης. Είναι κατανοητό, ότι σε μια τρισδιάστατη κατασκευή κάθε σημείο (άρα και κάθε κόμβος) έχει τρεις β.ε., στη δισδιάστατη δύο και στα δικτυώματα μια. Εμπλοκή εμφανίζεται στις καμπτόμενες κατασκευές, όπου για να προσδιορισθεί η θέση του στοιχείου δεν επαρκεί το βέλος αλλά χρειάζεται και η κλίση. Έτσι, για παράδειγμα, στις επίπεδες πλαισιωτές κατασκευές κάθε κόμβος έχει τρεις βαθμούς ελευθερίας, που είναι οι μετατοπίσεις κατά τους άξονες

x και y και η στροφή του κόμβου γύρω από τον άξονα z, στις πλάκες χρειάζεται τουλάχιστον το βέλος καθώς και οι κλίσεις σε δύο κάθετες διευθύνσεις κ.λπ.

Γενικά, υπάρχουν κάποιες ιδιομορφίες στις καμπτόμενες κατασκευές. Ονομάζουμε *N* τον συνολικό αριθμό των στοιχείων στον οποίον υποδιαιρείται η κατασκευή, n τον συνολικό αριθμό των κόμβων της κατασκευής, k τον αριθμό των β.ε. κάθε κόμβου, ενώ γίνεται η υπόθεση ότι κάθε στοιχείο έχει p κόμβους.



Σχήμα 2.3: Σύνδεση στοιχείων.

Κάθε πεπερασμένο στοιχείο δεν παύει να έχει την ίδια ελαστική συμπεριφορά με το αρχικό σώμα. Το κέρδος από αυτή την υποδιαίρεση είναι, ότι το στοιχείο έχει πεπερασμένο μέγεθος και απλούστερη μορφή. Αυτές του οι ιδιότητες επιτρέπουν τη μελέτη της έντασης του προσεγγιστικά. Έτσι, γνωρίζοντας τις μετατοπίσεις των κόμβων ενός στοιχείου είναι εφικτή η παρεμβολή, για τον υπολογισμό των μετατοπίσεων κάθε σημείου του στοιχείου. Το επόμενο βήμα είναι βέβαια ο υπολογισμός των παραμορφώσεων και τέλος των τάσεων. Έχοντας αυτά τα στοιχεία είναι δυνατή η εφαρμογή των ενεργειακών θεωρημάτων, απ' όπου θα προκύψουν kn εξισώσεις, που προσδιορίζουν τις kn άγνωστες μετατοπίσεις των κόμβων.

Με την υπόθεση ότι οι τάσεις στους κόμβους είναι γνωστές, προκύπτει το στατικό μοντέλο. Στη συνέχεια θα γίνει περιγραφή του κινηματικού μοντέλου ή μοντέλο μετατοπίσεων, που είναι το περισσότερο εν χρήσει. Ο κατάλογος των δυνατών μοντέλων συμπληρώνεται από το υβριδικό και το μικτό μοντέλο.[22]

1.3.2 Προσδιορισμός των μετατοπίσεων σε κάθε στοιχείο

Όπως αναφέρθηκε και προηγουμένως, οι μετατοπίσεις σε κάθε σημείο του στοιχείου εκφράζονται συναρτήσει των μετατοπίσεων των κόμβων με την βοήθεια κάποιου παρεμβολικού τύπου. Συνήθως, για την παρεμβολή χρησιμοποιούνται πολυώνυμα και το πεδίο των μετατοπίσεων ορίζεται ξεχωριστά σε κάθε στοιχείο. Τα παραπάνω συνεπάγονται κάποια ελευθερία, δεν αποκλείουν όμως την ύπαρξη κάποιων προϋποθέσεων. Έτσι π.χ., δεν μπορεί να είναι οι μετατοπίσεις των στοιχείων τέτοιες που τα στοιχεία μετά την παραμόρφωση να αλληλεπικαλύπτονται.

Άρα, πρώτος κανόνας είναι:

1. *Οι μετατοπίσεις των στοιχείων πρέπει να ορισθούν έτσι ώστε να υπάρχει συνέχεια μετατοπίσεων μεταξύ τους.* Στην περίπτωση καμπτόμενων κατασκευών δεν μπορούν τα στοιχεία στα σημεία σύνδεσης τους να εμφανίζουν, μετά την παραμόρφωση, γωνίες διαφορετικές από τις γωνίες που εμφάνιζαν πριν την παραμόρφωση. Άρα, υπάρχει και αυτή η πρόσθετη απαίτηση για τις καμπτόμενες κατασκευές. Είναι επίσης αυτονόητο ότι:

2. *Οι μετατοπίσεις σε κάθε στοιχείο πρέπει να είναι συνεχείς συναρτήσεις ώστε να δίνουν συνεχείς παραμορφώσεις*.

Ακόμα υπάρχει και η ακόλουθη απαίτηση:

3. *Το πεδίο των μετατοπίσεων σε κάθε στοιχείο πρέπει να είναι τέτοιο ώστε να μπορεί να περιγράψει την μετακίνηση, στερεού σώματος καθώς και την σταθερή, παραμόρφωση, της κατασκευής.*

Άρα, είναι αναγκαίο να υπάρχει στο πολυώνυμο της παρεμβολής σταθερός όρος καθώς και πρωτοβάθμιοι όροι. Για να γίνει πιο κατανοητή η παραπάνω διαδικασία, παρουσιάζεται η γενική μορφή του πολυωνύμου της παρεμβολής, που δίνει τη μετατόπιση q(x)=u(x) σ' ένα μονοδιάστατο στοιχείο *e* () (ράβδο δικτυώματος ή δοκό σε κάμψη)

$$q(x) = u(x) = a_1 + a_2 x + a_3 x^2 + \ldots + a_{m+1} x^m$$

του γράφεται :

$$q(x) = [1 \quad x \quad x^2 \quad \ldots x^m] \begin{bmatrix} a_1 \\ a_2 \\ \ldots \\ a_{m+1} \end{bmatrix}$$ ή $q(x)=M(x)a$.

όπου a είναι το μητρώο των συντελεστών $a_1,\ldots,a_{m+1}$.

Από τα παραπάνω, αλλά όπως θα γίνει κατανοητό και στη συνέχεια, ο αριθμός των όρων (m+1) πρέπει να είναι ίσος με τον συνολικό αριθμό των βαθμών ελευθερίας του στοιχείου. Στην περίπτωση της επίπεδης έντασης ή επίπεδης παραμόρφωσης, υπάρχουν δύο συνιστώσες, *υ(x,y)* και *υ(x,y)* της μετατόπισης. Το πολυώνυμο της παρεμβολής για κάθε μετατόπιση θα είναι

$$u(x,y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 + \ldots$$
$$\upsilon(x,y) = b_1 + b_2 x + b_3 y + b_4 x^2 + b_5 xy + b_6 y^2 + \ldots$$

Ή αλλιώς

$$q(x,y) = \begin{bmatrix} u \\ \upsilon \end{bmatrix} = \begin{bmatrix} 1 & x & y & \ldots & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & \ldots & 1 & x & y & \ldots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \ldots \\ b_1 \\ b_2 \\ \ldots \end{bmatrix}$$

Ή $$q(x,y) = \begin{bmatrix} G(x,y) & 0 \\ 0 & G(x,y) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$.

Όσον αφορά στην τρισδιάστατη ανάλυση

$$q(x,y,z)=\begin{bmatrix} u(x,y,z) \\ \upsilon(x,y,z) \\ w(x,y,z) \end{bmatrix}=\begin{bmatrix} G(x,y,z) & 0 & 0 \\ 0 & G(x,y,z) & 0 \\ 0 & 0 & G(x,y,z) \end{bmatrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Στο σημείο αυτό να σημειωθεί, ότι το πλήθος των συντελεστών a_i,b_i *και* c_i στις παραπάνω σχέσεις είναι ίσος με τον αριθμό των κόμβων του στοιχείου. Οι σχέσεις αυτές μπορούν να γραφτούν και με ενιαία μορφή ως εξής

$$q(x)=M(x)a$$

όπου a είναι το μητρώο στήλη των συντελεστών a_i, b_i (και c_i στην τρισδιάστατη κατάσταση) και x είναι οι συντελεστές ενός σημείου του στοιχείου. Η τελευταία σχέση συμπίπτει με τη δεύτερη σχέση για τα μονοδιάστατα στοιχεία.

Η τελευταία σχέση που δίνει τις μετατοπίσεις κάθε σημείου ενός στοιχείου *e* ((β)), θα δίνει και τις μετατοπίσεις q_i,q_j,q_m,… των κόμβων *i,j,m,…* του στοιχείου, αν εφαρμοστεί για τις τιμές του x ίσες με τις συντεταγμένες x_i,x_j,x_m,… των κόμβων.

Προκύπτουν λοιπόν οι γραμμικές σχέσεις

$q_i=M(x_i)a$

$q_j=M(x_j)a$

$q_m=M(x_m)a$

. . .

οι οποίες μπορούν να γραφτούν με μητρωϊκή μορφή ως εξής:

$$\begin{bmatrix} q_i \\ q_j \\ q_m \\ \dots \end{bmatrix}=\begin{bmatrix} M(x_i) \\ M(x_j) \\ M(x_m) \\ \dots \end{bmatrix}a$$

ή, σε πιο συμπυκνωμένη γραφή $q_e=Aa$.

Στη γενική περίπτωση (όχι στις καμπτόμενες κατασκευές) η σχέση $\mathbf{q_e=Aa}$ μπορεί να αντιστραφεί, οπότε $\mathbf{a=A^{-1}q_e}$. Αντικαθιστώντας τη σχέση τελευταία στη σχέση **q(x)=M(x)a** βρίσκουμε:

$$q(x)=N(x)q_e.$$

$$q(x)=\begin{bmatrix} N_i(x) & N_j(x) & N_m(x) & \dots \end{bmatrix}\begin{bmatrix} q_i \\ q_j \\ q_m \\ \dots \end{bmatrix}$$

Οπού $N(x)=M(x)A^{-1}$.

Οι συναρτήσεις $N_l(x)$ (l=i,j,m...) ονομάζονται συναρτήσεις σχήματος και ικανοποιούν τις σχέσεις $N_i(x)=I$ (μοναδιαίο μητρώο) , $N_j(x_i)=N_m(x_i)=...=0$. Τα ίδια ισχύουν και για όσες προκύπτουν από κυκλική εναλλαγή των δεικτών. Με τον τρόπο αυτό εκφράζονται οι μετατοπίσεις μέσα στο στοιχείο e συναρτήσει των μετατοπίσεων των κόμβων του στοιχείου. Το επόμενο βήμα είναι να υπολογισμός των τροπών και των τάσεων.[22]

1.3.3 Τροπές – Τάσεις

Στη δισδιάστατη ελαστικότητα οι τροπές δίνονται από τη σχέση

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{bmatrix} = \begin{bmatrix} \partial u / \partial x \\ \partial \upsilon / \partial y \\ \partial u / \partial x + \partial \upsilon / \partial y \end{bmatrix} = \begin{bmatrix} \partial / \partial x & 0 \\ 0 & \partial / \partial y \\ \partial / \partial y & \partial / \partial x \end{bmatrix} \begin{bmatrix} u \\ \upsilon \end{bmatrix}$$

Ή **ε=Xq** , όπου X είναι το μητρώο τελεστής που φαίνεται στην τελευταία από τις σχέσεις . Αντίστοιχα ορίζονται οι τροπές και στην τρισδιάστατη κατάσταση ή και στη μονοδιάστατη. Άρα, η τελευταία σχέση είναι μια γενική έκφραση, οπότε αντικαθιστώντας την **q(x)=N(x)q_e** στην **ε=Xq** προκύπτει **ε=XNq=Bq_e**. Η τελευταία σχέση εκφράζει τις τροπές συναρτήσει των μετατοπίσεων των κόμβων q_e.

Όμως οι τάσεις συνδέονται με τις τροπές με τη σχέση **σ = Dε** όπου D=C είναι το μητρώο των ελαστικών σταθερών, δηλαδή το μητρώο που έχει το μέτρο ελαστικότητας προς όλες τις κατευθύνσεις. Έτσι, χρησιμοποιώντας και την σ = Dε προκύπτει : **σ = DBq_e**.

1.3.4 Αρχή των δυνατών έργων

Έστω, F={Fx ,*Fy* ,*Fz* }$_T$ και t={tx,ty,tz }$_T$ τα διανύσματα των καθολικών και επιφανειακών δυνάμεων. Τότε η αρχή των δυνατών έργων γράφεται με μητρωϊκή γραφή

$$\int_V \sigma^T \varepsilon^* \mathrm{dV} = \int_V F^T q^* \mathrm{dV} + \int_S t^T q^* \mathrm{dS}$$

Οι δυνατές μετατοπίσεις q* και οι δυνατές τροπές ε* εκφράζονται με τη βοήθεια σχέσεων αντίστοιχων προς τις σχέσεις **q(x)=N(x)q$_e$** και **ε=XNq=Bq$_e$** είναι

q*= Nq*$_e$, ε*= Bq*$_e$

Υποθέτοντας, ότι η αρχή των δυνατών έργων μπορεί να εκφραστεί σαν άθροισμα των επί μέρους δυνατών έργων σε κάθε στοιχείο και λαμβάνοντας υπόψη τα προηγούμενα προκύπτει:

$$\sum_{e=1}^{N} (q^{*e})^T \left[\int_{V^e} B^T DBdV - \int_{V^e} N^T FdV - \int_{S^e} T^T tdS \right] = 0$$

όπου η τελευταία σχέση προέκυψε από το ανάστροφο της σχέσης $\int_V \sigma^T \varepsilon^* \mathrm{dV} = \int_V F^T q^* \mathrm{dV} + \int_S t^T q^* \mathrm{dS}$ και V_e, S_e είναι αντίστοιχα ο όγκος και η επιφάνεια του στοιχείου *e*.

Η τελευταία σχέση γράφεται επίσης $\sum_{e=1}^{N} (q^{*e})^T \left[k^e q^e - F^e \right] = 0$ όπου $F^e = F_F^e + F_t^e$ οι κομβικές δυνάμεις, δηλαδή η αναγωγή των καθολικών φορτίσεων *F* και των επιφανειακών *t*, σε ισοδύναμα φορτία στους κόμβους. Οι κομβικές δυνάμεις F_F^e και F_t^e δίνονται από τις σχέσεις

$$F_F^e = \int_{V^e} B^T DBdV \quad , \quad F_t^e = \int_{S^e} N^T tdS \; .$$

Τέλος k_e είναι το μητρώο ακαμψίας του στοιχείου e και δίνεται από τη σχέση

$$k^e = \int_{V^e} B^T DBdV$$

Εισάγονται τα μητρώα δ(kn x l) και δ*(kn x l) των μετατοπίσεων και των δυνατών μετατοπίσεων όλων των κόμβων της κατασκευής. Η σχέση (2.25) μπορεί να γραφεί με τη βοήθεια του δ αντί των μητρώων των μετατοπίσεων, q_e, των κόμβων του στοιχείου e. Για το λόγο αυτό χρησιμοποιείται το μητρώο στήλη R_e με διαστάσεις (kn x l), που περιλαμβάνει τα στοιχεία του F_e σε αυτές τις θέσεις που αντιστοιχούν και τα στοιχεία του q_e μέσα στο μητρώο S. Τα υπόλοιπα στοιχεία του R_e είναι μηδέν. Όμοια εισάγεται το μητρώο K. Έτσι, η σχέση $\sum_{e=1}^{N}(q^{*e})^T\left[k^e q^e - F^e\right]=0$ γράφεται:

$$\sum_{e=1}^{N}(\delta^*)^T\left[k^e\delta - \mathrm{R}^e\right]=0$$

ή τελικά, επειδή δ* είναι τυχαίο διάνυσμα,

$$\left(\sum_{e=1}^{N}k^e\right)\delta-\sum_{e=1}^{N}\mathrm{R}^e=0$$

.

Αυτή, λόγω Kδ = R μπορεί να γραφτεί ως εξής: $K=\sum_{e=1}^{N}k^e$ που είναι το μητρώο ακαμψίας όλης της κατασκευής, με διαστάσεις (kn x kn). Στην πραγματικότητα το μητρώο K σχηματίζεται με τοποθέτηση των στοιχείων του k_e σε γραμμή και στήλη που αντιστοιχεί το q_e μέσα στο δ. Ακόμα, $R=\sum_{e=1}^{N}R^e$ είναι το μητρώο των εξωτερικών φορτίων στους κόμβους.

Η αναγωγή των κατανεμημένων επιφανειακών δυνάμεων και των μαζικών δυνάμεων σε ισοδύναμες κομβικές δυνάμεις γίνεται με τη βοήθεια των σχέσεων $\mathrm{F}_{\mathrm{F}}^{\mathrm{e}}=\int_{V^e}B^T DBdV$, $\mathrm{F}_{\mathrm{t}}^{\mathrm{e}}=\int_{S^e}N^T tdS$. Η σχέση Kδ = R αποτελεί ένα γραμμικό σύστημα εξισώσεων που προσδιορίζει τις μετατοπίσεις των κόμβων. Γνωρίζοντας τις μετατοπίσεις είναι εφικτός ο υπολογισμός των τάσεων και των τροπών από τις σχέσεις ε=XNq=Bq^e και σ = DBq^e.

Στην περίπτωση που η αρχή των δυνατών έργων περιορισθεί μόνο στο στοιχείο *e*, από τη σχέση $\sum_{e=1}^{N}(q^{*e})^T\left[k^e q^e - F^e\right]=0$ ή την σχέση Kδ = R προκύπτει: $k^e q^e = F^e$.

2. Αλγόριθμοι Βελτιστοποίησης

2.1 Αλγόριθμοι: Γενικά

Ως **αλγόριθμος** ορίζεται μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος. Πιο απλά **αλγόριθμο** ονομάζουμε μία σειρά από εντολές που έχουν αρχή και τέλος, είναι σαφείς και εκτελέσιμες που σκοπό έχουν την επίλυση κάποιου προβλήματος. [20]

2.1.1 Δημιουργία αλγορίθμου

Τα βήματα δημιουργίας αλγόριθμου είναι:

- Διατύπωση του προβλήματος
- Κατανόηση του προβλήματος
- Λύση του προβλήματος
- Διατύπωση του αλγόριθμου
- Έλεγχος της λύσης

2.1.2 Κριτήρια αλγορίθμου

Οι αλγόριθμοι θα πρέπει να πληρούν κάποια πρότυπα και να διατυπώνονται με συγκεκριμένο τρόπο. Έτσι ένας αλγόριθμος πρέπει να ικανοποιεί τα επόμενα κριτήρια:

- Καθοριστικότητα
- Περατότητα
- Αποτελεσματικότητα
- Επεκτασιμότητα
- Να έχει είσοδο δεδομένων, επεξεργασία και έξοδο αποτελεσμάτων

Αναλυτικότερα,

- **Καθοριστικότητα - Definiteness**

Κάθε κανόνας του ορίζεται επακριβώς και η αντίστοιχη διεργασία είναι συγκεκριμένη. Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της.

Π.χ. Σε μία διαίρεση να λαμβάνεται υπόψη και η περίπτωση όπου ο διαιρετέος λαμβάνει μηδενική τιμή. Τυπικές περιπτώσεις η διαίρεση με το μηδέν, υπόριζος ποσότητα αρνητική, κλπ. Προβλήματα καθοριστικότητας αντιμετωπίζονται συχνά με τη λογική της επιλογής, δηλ. Αν α>0 τότε ..... αλλιώς .....

- **Περατότητα - Finiteness**

Κάθε εκτέλεση είναι *πεπερασμένη*, δηλαδή τελειώνει ύστερα από έναν πεπερασμένο αριθμό διεργασιών ή βημάτων. Μία διαδικασία που δεν τελειώνει μετά από συγκεκριμένο/πεπερασμένο αριθμό βημάτων λέγεται απλά υπολογιστική διαδικασία.

- **Αποτελεσματικότητα - Effectiveness**

Είναι *μηχανιστικά* αποτελεσματικός, δηλαδή όλες οι διαδικασίες που περιλαμβάνει μπορούν να πραγματοποιηθούν με ακρίβεια και σε πεπερασμένο χρόνο "με μολύβι και χαρτί". Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή (και όχι σύνθετη). Δηλαδή μία εντολή δεν αρκεί να έχει ορισθεί αλλά πρέπει να είναι και εκτελέσιμη.

- **Είσοδος δεδομένων - Input**

Κατά την εκκίνηση εκτέλεσης του αλγορίθμου καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δε δίνονται τιμές δεδομένων εμφανίζεται όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες

πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.

- **Έξοδος αποτελεσμάτων - Output**

Δίδει τουλάχιστον ένα μέγεθος ως αποτέλεσμα που εξαρτάται κατά κάποιο τρόπο από τις αρχικές εισόδους. Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον *μία* τιμή (δεδομένων) ως αποτέλεσμα προς το χρήστη ή προς ένα άλλο αλγόριθμο.[19]

2.1.3 Περιγραφή και αναπαράσταση

Τέσσερις είναι οι βασικοί τρόποι αναπαράστασης ενός αλγορίθμου:

Ελεύθερο κείμενο, που αποτελεί τον πιο αδόμητο τρόπο παρουσίασης αλγορίθμου. Ελλοχεύει η δημιουργία μιας μη εκτελέσιμης κατάστασης παραβιάζοντας έτσι το κριτήριο της αποτελεσματικότητας.

Διάγραμμα ροής, που συνιστά έναν πιο γραφικό τρόπο παρουσίασης του αλγορίθμου. Η χρήση διαγραμμάτων ροής δεν είναι η πλέον ενδεδειγμένη λύση για ένα πρόβλημα και για αυτό χρησιμοποιούνται σπάνια στη βιβλιογραφία.

Φυσική γλώσσα που εκτελείται κατά βήματα. Σε αυτή την περίπτωση μπορεί να παραβιαστεί το κριτήριο του καθορισμού μεταξύ των βημάτων.

Κωδικοποίηση του αλγορίθμου σε ψευδογλώσσα ή γλώσσα προγραμματισμού. Έτσι ο αλγόριθμος παρουσιάζεται πιο συνοπτικός, συμπαγής ενώ πληροί και τις προϋποθέσεις του Δομημένου προγραμματισμού.[20]

2.2 Τυποποιημένοι αλγόριθμοι

Οι αλγόριθμοι είναι σημαντικοί γιατί σχετίζονται άμεσα με τον τρόπο με τον οποίο οι υπολογιστές επεξεργάζονται δεδομένα και παράγουν πληροφορίες . Ένα πρόγραμμα υπολογιστών είναι ουσιαστικά ένας αλγόριθμος που λέει στον υπολογιστή ποια συγκεκριμένα βήματα να εκτελέσει (σε ποια συγκεκριμένη σειρά) προκειμένου να επιτευχθεί ένας συγκεκριμένος στόχος, όπως π.χ. ο υπολογισμός των μισθών των υπαλλήλων ή η εκτύπωση των έλεγχων των μαθητών. Κατά συνέπεια, ένας αλγόριθμος μπορεί να θεωρηθεί οποιαδήποτε ακολουθία εντολών που μπορεί να εκτελεσθεί από μια υπολογιστική μηχανή (turing).

Χαρακτηριστικά, όταν ένας αλγόριθμος συνδέεται με την επεξεργασία πληροφοριών, τα δεδομένα διαβάζονται από μια συσκευή εισόδου, γράφονται σε μια συσκευή εξόδου, και / ή αποθηκεύονται για την περαιτέρω χρήση. Τα αποθηκευμένα στοιχεία θεωρούνται ως τμήμα της εσωτερικής κατάστασης του συστήματος που εκτελεί τον αλγόριθμο.

Για οποιαδήποτε τέτοια υπολογιστική διαδικασία, ο αλγόριθμος πρέπει να οριστεί αυστηρά: να είναι ορισμένος για όλες τις πιθανές περιστάσεις που θα μπορούσαν να προκύψουν. Δηλαδή οποιαδήποτε υπό όρους βήματα πρέπει να εξεταστούν συστηματικά, και σε κάθε περίπτωση τα κριτήρια πρέπει να είναι σαφή (και υπολογίσιμα).

Επειδή ένας αλγόριθμος είναι ένας ακριβής κατάλογος βημάτων ακριβείας, η σειρά του υπολογισμού θα είναι σχεδόν πάντα κρίσιμη για τη λειτουργία του αλγόριθμου. Οι εντολές συνήθως απαριθμούνται ρητά, και περιγράφονται σαν να ξεκινούν "από την κορυφή" και πηγαίνοντας "προς στο κατώτατο σημείο", μια ιδέα που περιγράφεται τυπικά με τον όρο της "ροής ελέγχου".

Μέχρι τώρα, σε αυτήν η συζήτηση για την τυποποίηση του αλγορίθμου, έχουμε δεχθεί ως βάση τον διαδικαστικό προγραμματισμό. Αυτή είναι και η πιο κοινή αντίληψη, η οποία προσπαθεί να περιγράψει ένα έργο με διακεκριμένα, "μηχανικά" μέσα. Μοναδικός σε αυτήν την αντίληψη των αλγόριθμων είναι ο ρόλος της λειτουργίας ανάθεσης (ο καθορισμός της τιμής μιας μεταβλητής) ο οποίος προέρχεται από τη ιδέα "της μνήμης" σαν πρόχειρο τετράδιο. Δείτε ακόμα το λειτουργικό προγραμματισμό και τον λογικό προγραμματισμό για εναλλακτικές αντιλήψεις για το τι αποτελεί έναν αλγόριθμο.

2.3 Αλγόριθμοι Βελτιστοποίησης

Ένας Αλγόριθμος βελτιστοποίησης αποτελεί μια αριθμητική μέθοδο για την εύρεση μια τιμής x^* , ώστε το αποτέλεσμα μιας αντικειμενικής συνάντησης $f(x)$ να είναι βέλτιστο (ελάχιστο ή μέγιστο). Πιθανό είναι να υπάρχουν και κάποιοι περιορισμοί όσον αφορά τις τιμές του x .

Η μαθηματική έκφραση αυτού είναι η παρακάτω:

Minimize $f(x)$

με περιορισμούς

$\mathbf{h(x) = 0}$

$\mathbf{g(x) = 0}$ με $x \in X \subseteq R_n$,

όπου X ένα υποσύνολο του n-διάστατου χώρου R_n.

2.3.1 Ο αλγόριθμος Quasi – Newton.

Στην βελτιστοποίηση οι μέθοδοι Quasi – Newton (γνωστή και ως variable metric μέθοδος) είναι αλγόριθμοι για την εύρεση τοπικών ελάχιστων και τοπικών μέγιστων σε μια συνάρτηση. Βασίζονται στη μέθοδο του Newton, η οποία προσπαθεί να προσδιορίσει ένα στάσιμο σημείο, όπου η πρώτη παράγωγος είναι ίση με μηδέν. Αυτή η μέθοδος υποθέτει ότι μια συνάρτηση, σε μια περιοχή γύρω από το βέλτιστο, μπορεί, τοπικά, να προσεγγιστεί σαν τετραγωνική (quadratic). Έτσι με την βοήθεια της πρώτης και της δεύτερης παραγώγου, μπορεί να βρεθεί το στάσιμο σημείο.

Σε μεγαλύτερες διαστάσεις, η μέθοδος Quasi – Newton χρησιμοποιεί την παράγωγο και τον Χεσσιανό πίνακα (μήτρα) της δεύτερης παραγώγου της συνάρτησης για να ελαχιστοποιηθεί.

Στις μεθόδους Quasi-Newton η Χεσσιανή μήτρα δεν χρειάζεται να υπολογιστεί. Ο Χεσσιανός πίνακας ενημερώνεται αναλύοντας διαδοχικά διανύσματα κλίσης αντ 'αυτού. Οι Μέθοδοι Quasi-Newton είναι μια γενίκευση της μεθόδου των τεμνουσών για να βρούμε τη ρίζα της πρώτης παραγώγου για πολυδιάστατα προβλήματα. Σε πολλαπλές διαστάσεις η τέμνουσα εξίσωση είναι υπό-προσδιοριζόμενη, και οι μέθοδοι Quasi -Newton διαφοροποιούνται στον τρόπο που περιορίζουν τη λύση, συνήθως με την προσθήκη μιας απλής ελαχίστου τιμής στην τρέχουσα εκτίμηση της Χεσσιανής μήτρας.

Ένα από τα κύρια πλεονεκτήματα των μεθόδων Quasi -Newton σε σχέση με τη μέθοδο του Νεύτωνα είναι ότι η Χεσσιανή μήτρα (ή, στην περίπτωση των μεθόδων Quasi -Newton, η προσέγγισή της) B δεν χρειάζεται να αναστραφεί. Η μέθοδος του Νεύτωνα, και τα παράγωγά της, όπως η μέθοδος του εσωτερικού σημείου , απαιτούν την αναστροφή του Χεσσιανού, η οποία τυπικά εφαρμόζεται από την επίλυση ενός συστήματος γραμμικών εξισώσεων και συχνά είναι αρκετά δαπανηρή. Αντίθετα, οι μέθοδοι Quasi-Newton παράγουν συνήθως μια εκτίμηση της B^{-1} άμεσα.

Ας προσπαθήσουμε να περιγράψουμε την μέθοδο. Αν ένας πραγματικός αριθμός x^* είναι στάσιμο για μια συνάρτηση $f(x)$, τότε ο x^* είναι ρίζα της $f'(x)$. Το ανάπτυγμα του Taylor για την $f(x)$ θα δίνεται από τον παρακάτω τύπο,

$$f(x_k + \Delta x) \approx f(x_k) + \nabla f(x_k)^T \Delta x + \frac{1}{2} \Delta x^T B \, \Delta x,$$

και επιτυγχάνει το ακρότατο όταν το Δx επιλύει την γραμμική εξίσωση,

$\nabla f(x_k + \Delta x) \approx \nabla f(x_k) + B \, \Delta x$. και η $f''(x)$ είναι θετική.

Και θέτοντας την παραγώγο ίση με 0 (που είναι η αντικειμενική της βελτιστοποίησης) μας δίνει το βήμα Newton :

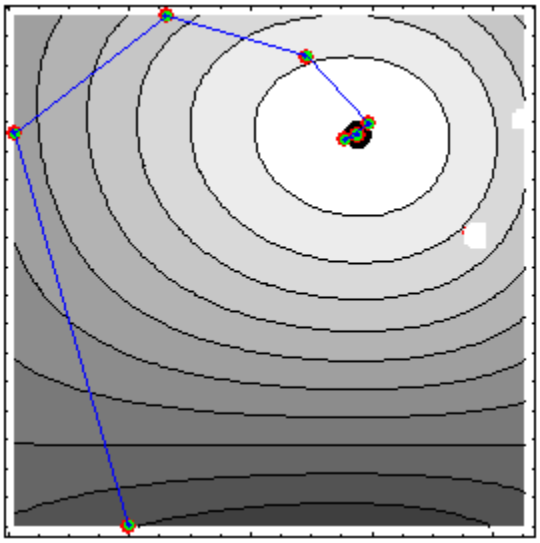
$$\Delta x = -B^{-1} \nabla f(x_k),$$

Γι' αυτό το λόγο, εφόσον η $f(x)$ είναι διπλά διαφορήσιμη, και η αρχική υπόθεση x_0 (τιμή από την οποία ξεκινά ο αλγόριθμος), έχει επιλεχθεί κοντά στο x^* , η ακολουθία x_n ορίζεται από τον παρακάτω τύπο,

$$X_{n+1} = X_n - \frac{f'(X_n)}{f''(X_n)}, n \geq 0$$

και συγκλίνει στο x^* .

Παρακάτω φαίνεται, γραφικά, πως ο αλγόριθμος αυτός συγκλίνει στο x^* .



Αυτή η επαναληπτική ιδέα μπορεί να γενικευθεί σε πολλές διαστάσεις, αντικαθιστώντας την πρώτη παράγωγο με βαθμωτή μεταβολή (gradient), $\nabla f(x)$ και την δεύτερη παράγωγο με τον αντίστροφο του Χεσσιανού πίνακα, $Hf(x)$. Έτσι έχουμε,

$$x_{n+1} = x_n - [Hf(x_n)]^{-1} \nabla f(x_n), n \geq 0 .$$

Συνηθίζεται η μέθοδος Quasi-Newton να περιλαμβάνει ένα μικρό βήμα, μεγαλύτερο του μηδενός, αντί για γ =1,

$$X_{n+1} = X_n - \gamma [Hf(x_n)]^{-1} \nabla f(x_n), n \geq 0 .$$

Αυτό γίνεται για να εξασφαλισθεί ότι ισχύουν οι συνθήκες Wolfe, για κάθε βήμα $X_n \rightarrow X_{n+1}$ της επανάληψης.

Ο πρώτος αλγόριθμος Quasi-Newton προτάθηκε από τον Davidon W. C. Το 1959, η DFP ενημερωτική φόρμουλα (DFP updating formula). Σήμερα οι γνωστότεροι και συχνότερα χρησιμοποιούμενοι αλγόριθμοι Quasi-Newton είναι η φόρμουλα SR1 και η μέθοδος BFGS.[7]

Στο παρακάτω πινακάκι βλέπουμε συνοπτικά τις πιο δημοφιλείς μεθόδους

Method	$B_{k+1} =$	$H_{k+1} = B_{k+1}^{-1} =$
DFP	$\left(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k}\right) B_k \left(I - \frac{\Delta x_k y_k^T}{y_k^T \Delta x_k}\right) + \frac{y_k y_k^T}{y_k^T \Delta x_k}$	$H_k + \frac{\Delta x_k \Delta x_k^T}{y_k^T \Delta x_k} - \frac{H_k y_k y_k^T H_k^T}{y_k^T H_k y_k}$
BFGS	$B_k + \frac{y_k y_k^T}{y_k^T \Delta x_k} - \frac{B_k \Delta x_k (B_k \Delta x_k)^T}{\Delta x_k^T B_k \Delta x_k}$	$\left(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k}\right)^T H_k \left(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k}\right) + \frac{\Delta x_k \Delta x_k^T}{y_k^T \Delta x_k}$
Broyden	$B_k + \frac{y_k - B_k \Delta x_k}{\Delta x_k^T \Delta x_k} \Delta x_k^T$	$H_k + \frac{(\Delta x_k - H_k y_k) \Delta x_k^T H_k}{\Delta x_k^T H_k y_k}$
Broyden family	$(1 - \varphi_k) B_{k+1}^{BFGS} + \varphi_k B_{k+1}^{DFP}, \quad \varphi \in [0,1]$	
SR1	$B_k + \frac{(y_k - B_k \Delta x_k)(y_k - B_k \Delta x_k)^T}{(y_k - B_k \Delta x_k)^T \Delta x_k}$	$H_k + \frac{(\Delta x_k - H_k y_k)(\Delta x_k - H_k y_k)^T}{(\Delta x_k - H_k y_k)^T y_k}$

Όπου DFP είναι η μέθοδος Davidon–Fletcher–Powell η οποία βρίσκει τη λύση της εξίσωσης τέμνουσας που είναι πιο κοντά στην τρέχουσα εκτίμηση και πληροί τις συνθήκες κυρτότητας. Ήταν η πρώτη Quasi-Newton μέθοδος που γενικεύει την μέθοδο τέμνουσας σε ένα πολυδιάστατο πρόβλημα. Αυτή η ενημερωμένη έκδοση διατηρεί τη συμμετρία και τη θετική οριστικότητα της Χεσσιανής μήτρας . Η BFGS είναι η μέθοδος των Broyden–Fletcher–Goldfarb–Shanno, είναι μια επαναληπτική μέθοδος για την επίλυση χωρίς περιορισμούς μη γραμμικών προβλημάτων βελτιστοποίησης.[20]

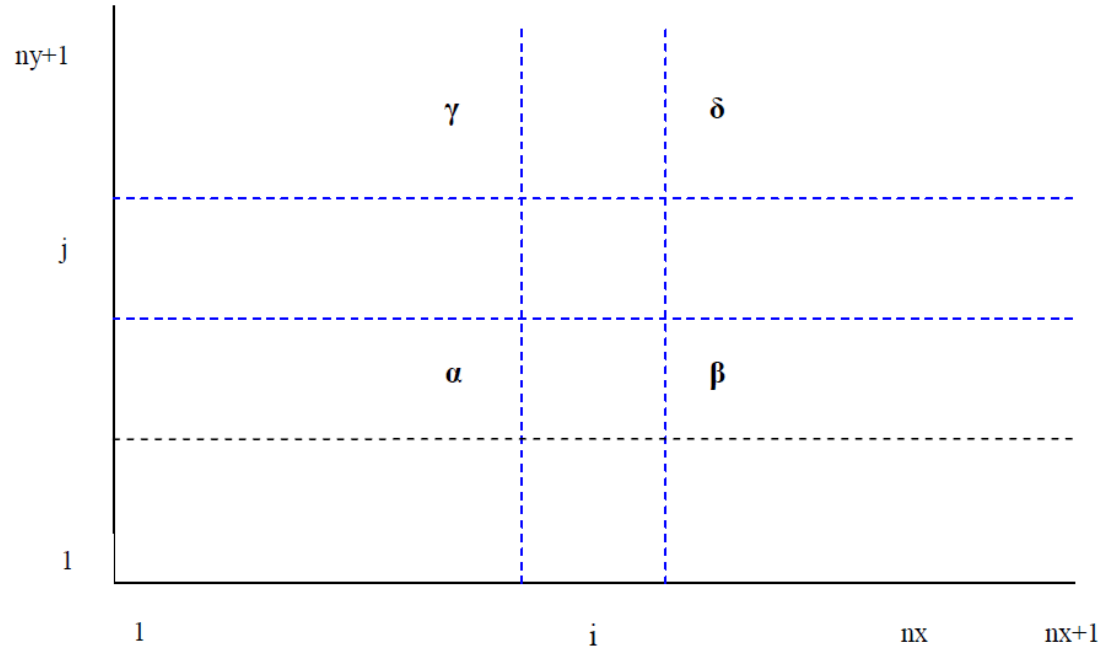
3. Το πρόβλημα των δυο διαστάσεων

3.1 Ορισμός του προβλήματος

Για την διαχείριση προβλημάτων δυο διαστάσεων χρησιμοποιήθηκε η διπλωματική εργασία του κ.Ν.Σκάρου, με τίτλο «Βέλτιστος σχεδιασμός δικτυωτών φορέων με την βοήθεια της μεθόδου των πεπερασμένων στοιχείων και αλγορίθμων βελτιστοποίησης». Σε αυτή την εργασία λοιπόν, επιλύθηκαν προβλήματα δυο διαστάσεων με την μέθοδο των Πεπερασμένων, σε συνδυασμό με την χρήση αλγορίθμου βελτιστοποίησης ,όπου στη συνέχεια συνέκρινε τα αποτελέσματα του κώδικα σε σχέση με άλλα αποτελέσματα προβλημάτων της βιογραφίας.

Το γενικό σχέδιο ήταν να κατασκευαστεί το μηχανικό πρόβλημα, να προσδιοριστούν οι συναρτήσεις στόχους και τέλος να βελτιστοποιηθούν αυτές και τις παραμέτρους τους. Πάνω από όλα τρέχει ένα σχετικά απλό πρόγραμμα βελτιστοποίησης, με περιορισμούς, που χρησιμοποιεί Quasi-Newton μέθοδο. Αυτό παίρνει σαν είσοδο του τιμές διατομών της κατασκευή μας και σαν έξοδο έχει τιμές διατομών που δίνουν την βέλτιστη αντικειμενική συνάρτηση , καθώς και την τιμή της. Πίσω από αυτό σχεδιάστηκαν κώδικες που , αρχικά, σχεδιάζουν δυσδιάστατα την κατασκευή, και στη συνέχεια εισέρχονται τα Πεπερασμένα στοιχεία και επιλύεται το σύστημα των εξισώσεων $K*u = p$.

Η αρίθμηση των κόμβων γίνεται όπως φαίνεται στο παρακάτω σχήμα. Ξεκινάμε από τον κόμβο με αριθμό 1, που βρίσκεται στο κάτω αριστερό άκρο και κατευθυνόμαστε προς τα δεξιά. Στη συνέχεια ανεβαίνουμε μια γραμμή πάνω και συνεχίζουμε την αρίθμηση.



Γενικά έστω ότι έχουμε το τυχαίο τετράπλευρο που δημιουργείται από τους κόμβους α,β,γ,δ. Η αρίθμηση αυτών θα δίνεται από τις παρακάτω σχέσεις, όπου,

$\alpha = (j-1)*(nx+1)+i,$

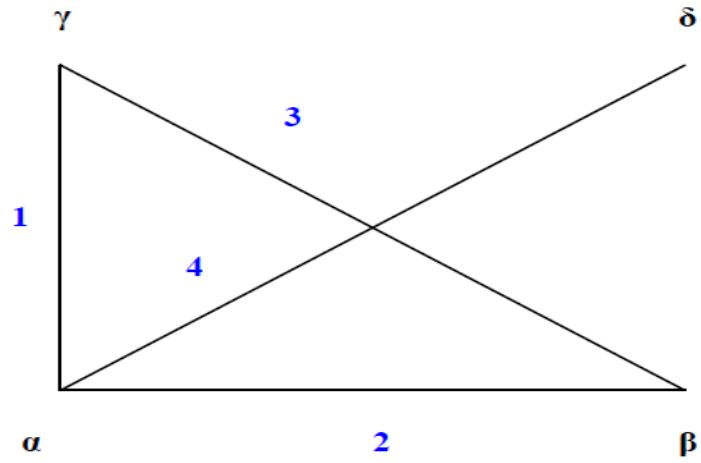
$\beta = \alpha+1 = (j-1)*(nx+1)+i+1,$

$\gamma = j*(nx+1)+i,$

$\delta = \gamma+1 = j*(nx+1)+i+1,$

όπου $1 \le i \le nx+1$ και $1 \le j \le ny+1$

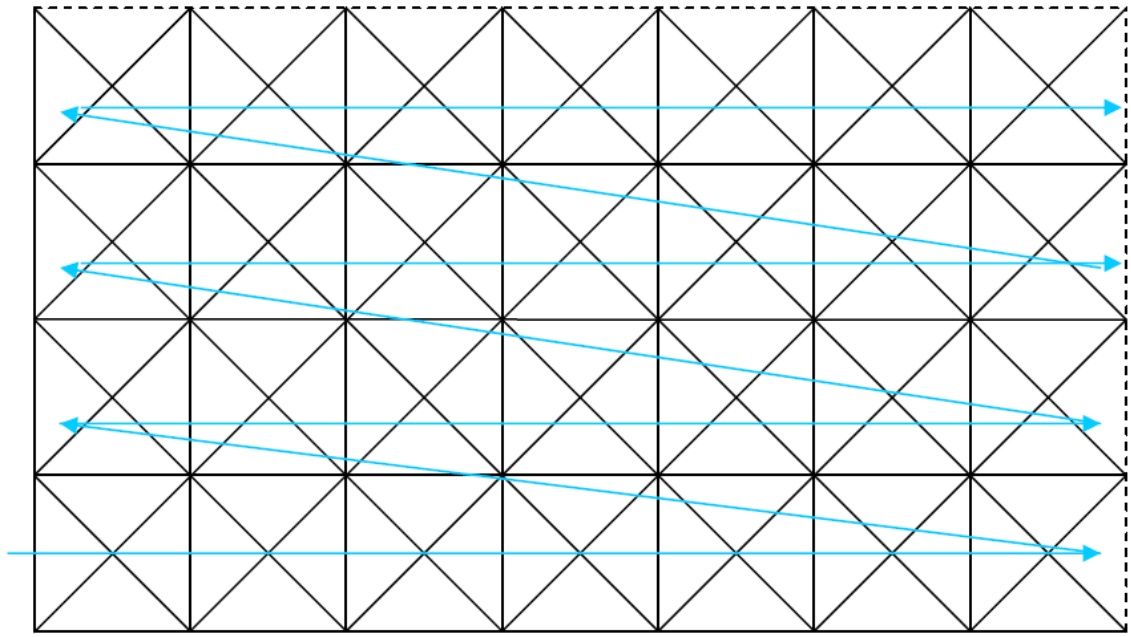
Όσον αφορά, τώρα την αρίθμηση των ράβδων που αποτελούν το πλαίσιό μας, εργαζόμαστε ως εξής στο παρακάτω σχήμα,



Ο παρακάτω πίνακας μας δίνει την αρχή και τέλος κάθε ράβδου του παραπάνω σχήματος.

Ραβδος	Αρχή (κόμβος)	Τέλος (κομβος)
1	γ	α
2	α	β
3	γ	β
4	α	δ

Έπειτα χωρίζουμε το πλαίσιο μας σε τέτοια μικρά σχήματα και συνεχίζουμε την αρίθμηση προς τα δεξιά και πάνω. Όταν τελειώσει αυτή διαδικασία έχουμε αριθμήσει τις παρακάτω ράβδους του πλαισίου.

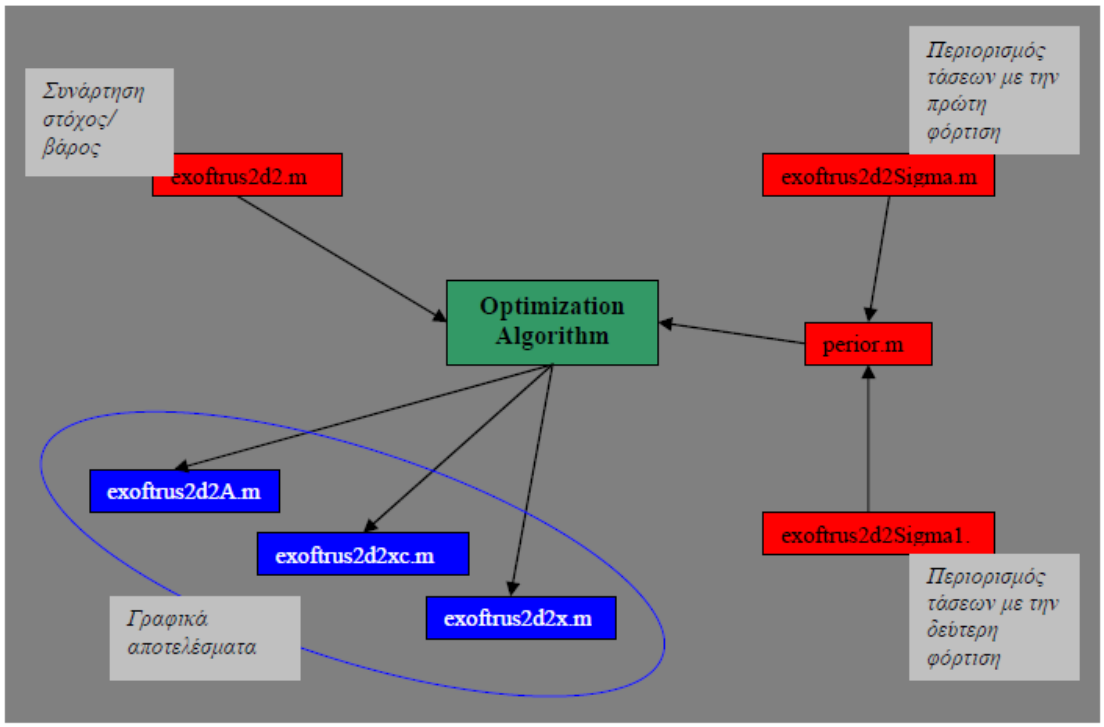


Αφού καθοριστεί η συνδεσμολογία, στη συνέχεια καθορίζονται οι συνοριακές συνθήκες στήριξης, οι παράμετροι του υλικού και θέτουμε φορτίσεις. Έτσι προσομοιώνεται μια κατασκευή. Κάθε φορά ο αλγόριθμος βελτιστοποίησης επεξεργάζεται την εκάστοτε αντικειμενική συνάρτηση (συνάρτηση βάρους ή συνάρτηση δυσκαμψίας) και προσπαθεί να βγάλει τα βέλτιστα αποτελέσματα κάτω από περιορισμούς που του δίνουμε (**Βέλτιστος Σχεδιασμός**).

Τα αποτελέσματα αυτά είναι οι τιμές των διατομών που βελτιστοποιούν κάθε φορά την αντικειμενική συνάρτηση, καθώς και γραφικά αποτελέσματα (**Τοπολογική Βελτιστοποίηση**).

Παρακάτω θα παραθέσουμε και το διάγραμμα του Ν.Σκάρου με σκοπό την εμπέδωση και κατανόηση των συνδεόμενων προγραμμάτων και συναρτήσεων. Για την υλοποίηση του παραπάνω προβλήματος αναπτύξαμε τα υποπρογράμματα ,exoftrus2d2.m, exoftrus2d2Sigma.m, exoftrus2d2Utotal.m, perior.m, καθώς και την υπορουτίνα truss2d.m που βρίσκονται στο φάκελο με τους κώδικες. Ο αλγόριθμος βελτιστοποίησής μας παίρνει την αντικειμενική συνάρτηση func1 (συνάρτηση βάρους), το Sigma, το Utotal από τα τρία πρώτα, αντίστοιχα και βελτιστοποιεί την func1 αφού λάβει τους περιορισμούς του τέταρτου, όπως

φαίνεται στο παρακάτω σχήμα.



Όπου `nx,ny` είναι ο αριθμός της υποδιαίρεσης του μήκους κάθε πλευράς του πλαισίου μας. Ο συνολικός αριθμός των ράβδων (nelements) καθώς και τα υπόλοιπα δεδομένα (αριθμός κόμβων, κ.α.) υπολογίζονται αυτόματα στο πρόγραμμά μας. Αξίζει να σημειωθεί ότι θέσαμε περιορισμούς στις τιμές των διατομών των ράβδων 0.1 μονάδες (lb). Δεν μπορεί δηλαδή να υπάρξει μη-θετική διατομή. Εφαρμόζουμε φόρτιση στον κόμβο (nx+2)/2 , κάθε φορά, κάθετη με κατεύθυνση προς τα κάτω και μέτρο 50 μονάδες. Ως πρότυπη κατασκευή χρησιμοποιούμε ένα δυσδιάστατο πλαίσιο, το οποίο το φορτίζουμε με μονόπλευρη δύναμη και δουλεύουμε πάνω σε αυτό.[8]

3.1. Βελτιστοποίηση ακαμψίας για συγκεκριμένο βάρος πλαισίου

3.1.1. Πρόβλημα 1

Το πρόβλημα εδώ είναι με δεδομένο βάρος να φτιάξουμε την πιο δύσκαμπτη κατασκευή.

Μαθηματικά, το πρόβλημα αυτό διατυπώνεται ως εξής:

$$\min \quad F = \sum_{j=1}^{nnodes} U_j * F_j,$$

$$K(x_i) * U_j = p$$

$$\text{st} \quad W \le W_{\max},$$

$$\text{με} \quad x_{\min} \le x_i \le x_{\max}.$$

Ο στόχος είναι να σχεδιαστεί η πιο άκαμπτη κατασκευή με περιορισμούς για το βάρος και τη διατομή που δεν μπορεί να είναι μικρότερη από 0.1 μονάδες. Η πυκνότητα του υλικού κατασκευής είναι 0,1 lb/in^3, ο συντελεστής Young E είναι 10^4.

Στο παράδειγμα που εφαρμόσαμε θέσαμε lx= 1000, ly= 1000, nx=5, ny=5, με ασκούμενο φορτίο στον κόμβο (1,2) = (nx+2)/2 και στον κόμβο (1,3) = -200 . Όσον αφορά στις συνοριακές συνθήκες έχουμε δύο, ενώ όλοι οι κόμβοι έχουν στηρίξεις. Οι ιδιότητες του υλικού περιλαμβάνονται στον κώδικα αλλά δεν αλλάζουν ουσιαστικά το αριθμητικό αποτέλεσμα.

Ο αλγόριθμος βελτιστοποίησης είναι ο εξής

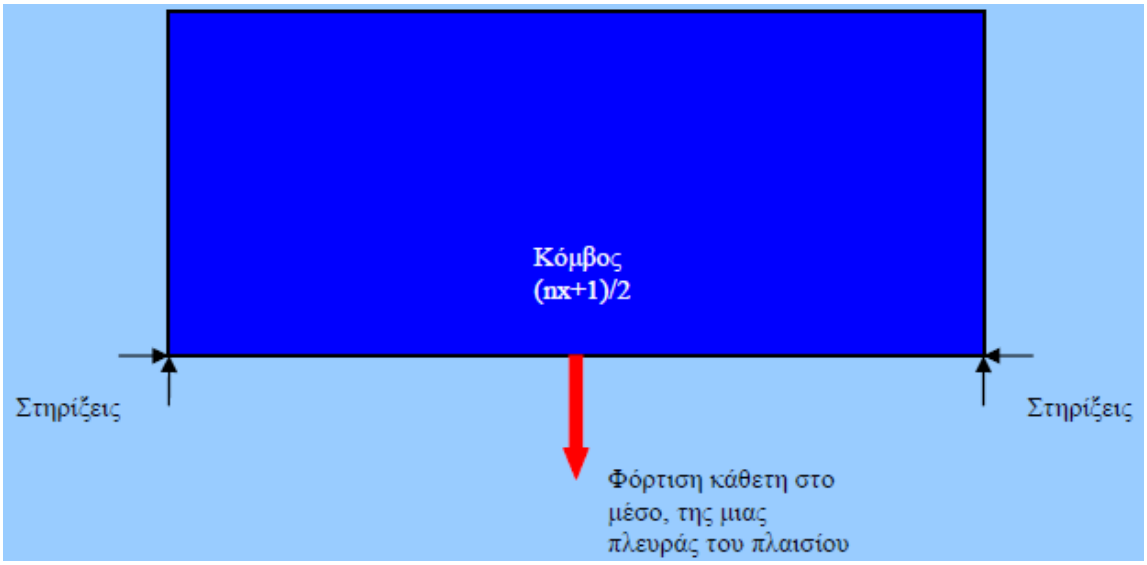
```
nx=5;

ny=5;

nelements = (nx*ny)*4+ny+nx;

x0=ones(nelements,1);

lb=[0.1*ones(nelements,1)];

ub=[101*ones(nelements,1)];

options =optimset('LargeScale', 'off');
```

```
[x,fval,exitflag,output]=fmincon(@exoftrus2d2,x0,[],[],[],[],lb
,ub,@perior,options);
```

Παίρνει δηλαδή την αντικειμενική συνάρτηση func2 (συνάρτηση κάμψης), το βάρος από τα exoftrus2d2.m, exotrus2d2func1.m, perior.m, truss2d.m που βρίσκονται στο φάκελο ΚΕΦΑΛΑΙΟ 3.

Όπου nx,ny=5 η υποδιαίρεση του μήκους κάθε πλευράς στο πλαίσιο, ο συνολικός αριθμός ράβδων υπολογίζεται αυτόματα, και επίσης δεν μπορεί να υπάρξει μη θετική διατομή. Αξίζει να σημειωθεί ότι θέσαμε περιορισμούς στις τιμές των διατομών των ράβδων από 0.1 μονάδες, έως 101 μονάδες(lb, ub), ενώ δεν μπορεί να υπάρξει μη-θετική διατομή.

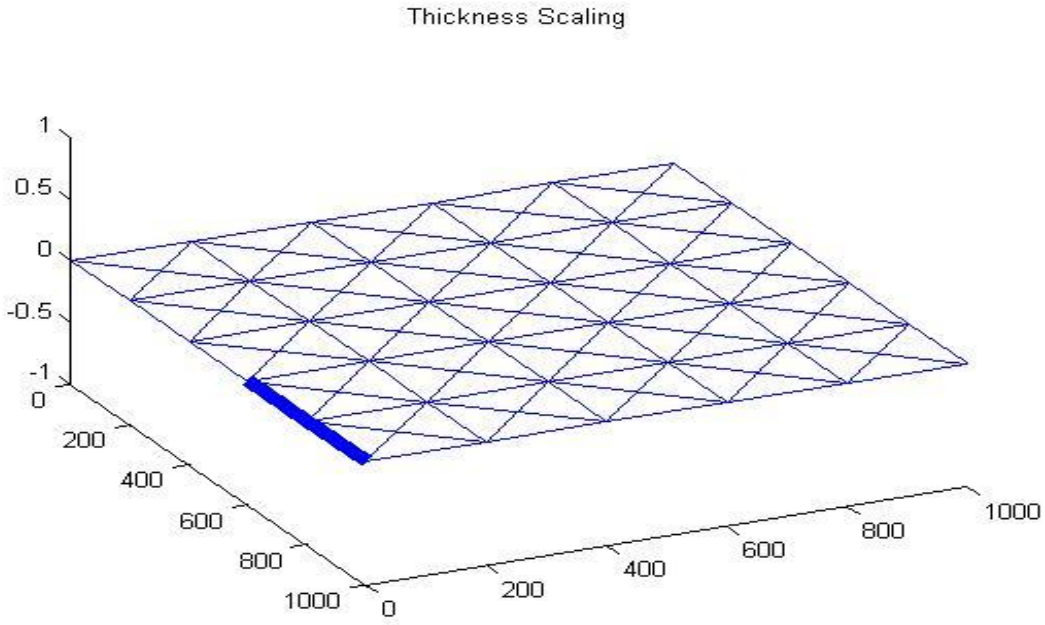
Εφαρμόζουμε φόρτιση στον κόμβο (nx+2)/2 , κάθε φορά, κάθετη με κατεύθυνση προς τα κάτω και μέτρο -200 μονάδες. ως πρότυπη κατασκευή χρησιμοποιούμε ένα δυσδιάστατο πλαίσιο, το οποίο το φορτίζουμε με μονόπλευρη δύναμη και δουλεύουμε πάνω σε αυτό.



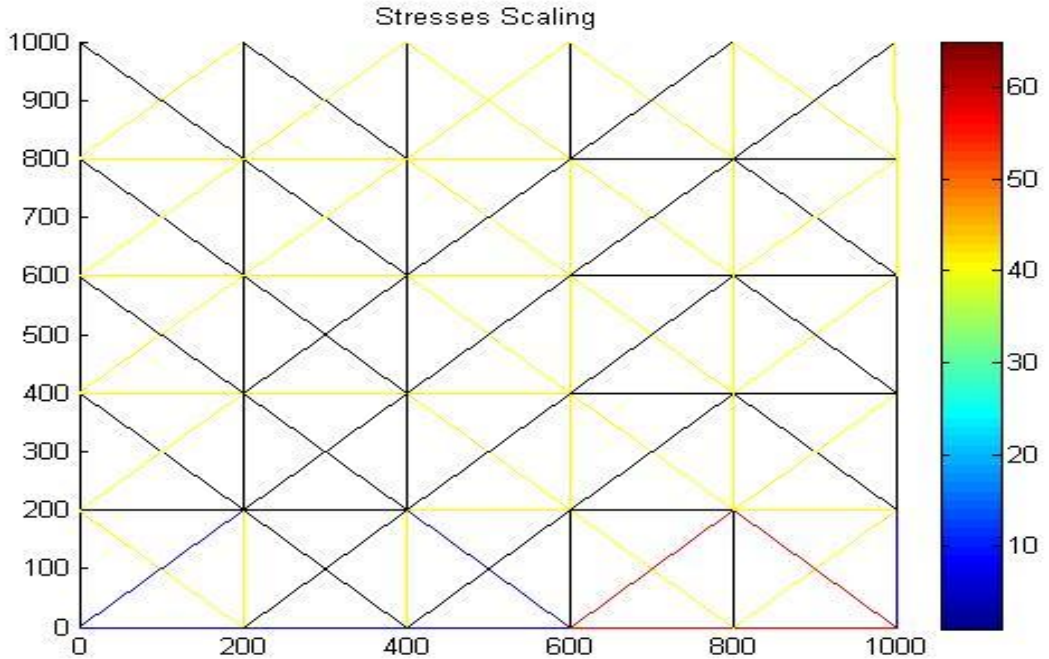
Αφού υπολογιστούν οι βέλτιστες διατομές των ράβδων και με την βοήθεια των υποπρογραμμάτων exoftrus2d2x.m, exoftrus2d2xc.m, exoftrus2d2A.m, παρατηρούμε γραφικά τα αποτελέσματα της φόρτισης του πλαισίου μας, τις τάσεις που αναπτύσσονται στις ράβδους, καθώς και το πάχος που έχουν οι ράβδοι με καθορισμένη κλίμακα.

Φαίνεται η διαβάθμιση των τάσεων του πλαισίου μας. Ειδικά από –00□ έως 0, έχουμε θλίψη και από 0 έως +00 έχουμε εφελκυσμό. Όταν οι τάσεις σχεδιάζονται με μαύρο χρώμα, αυτό αποτελεί πολύ καλό αποτέλεσμα για την ακαμψία της κατασκευής. Με τάσεις στο χρώμα μπλε ή κίτρινο σχετικά καλό αποτέλεσμα και τέλος με τάση στο χρώμα κόκκινο, «άσχημο αποτέλεσμα».

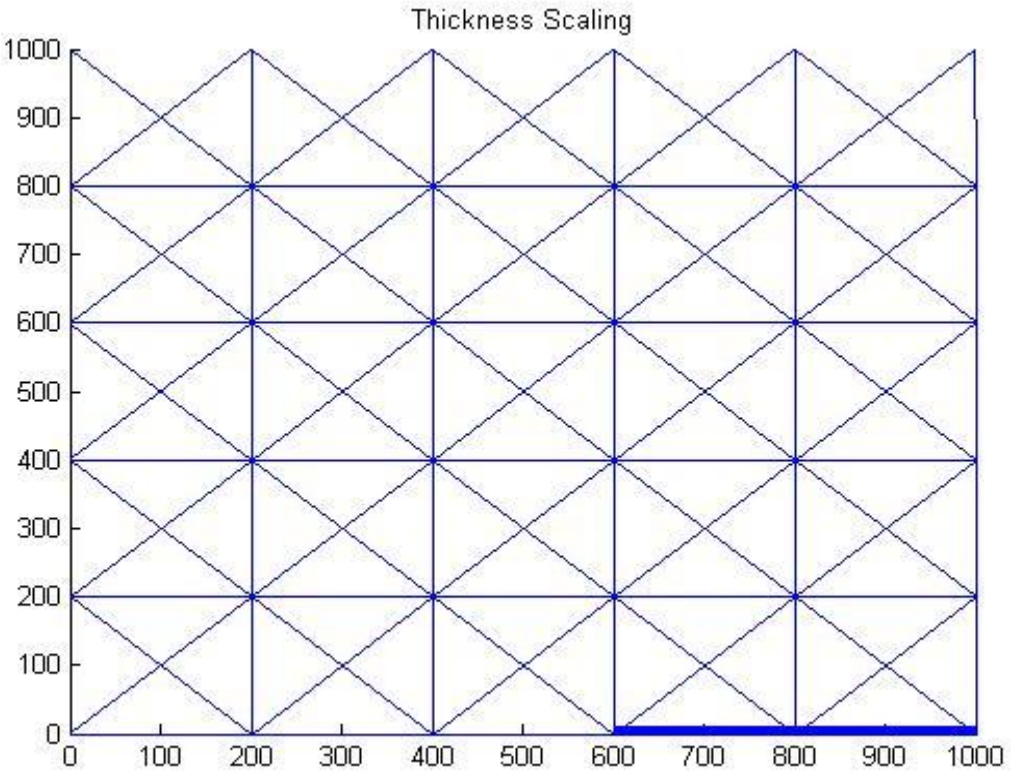
Μετά από επίλυση το προβλήματος προκύπτει η τιμή της αντικειμενικής συνάρτησης είναι fval = 1.4981e+005. Τα γραφικά που προκύπτουν έχουν ως εξής:



Αφού υπολογιστούν οι βέλτιστες διατομές των ράβδων και με την βοήθεια των υποπρογραμμάτων exoftrus2d2x.m, exoftrus2d2xc.m, exoftrus2d2A.m, παρατηρούμε γραφικά τα αποτελέσματα της φόρτισης του πλαισίου μας, τις τάσεις που αναπτύσσονται στις ράβδους (με καθορισμένη χρωματική κλίμακα), καθώς και το πάχος που έχουν οι ράβδοι (με καθορισμένη κλίμακα).



Στο παραπάνω διάγραμμα φαίνεται η κατανομή των τάσεων και παρακάτω,



το βέλτιστο πάχος των διατομών.

Η δε κλίμακα που χρησιμοποιήθηκε για το πάχος των ράβδων είναι η παρακάτω.

Για διατομή μικρότερη των :

- **Alpha_min+2*L/30** μονάδων, σχεδιάζει με **LineWidth = 1,**
- **Alpha_min+2*L/6** μονάδων, σχεδιάζει με **LineWidth = 2,**
- **Alpha_min+3*L/5** μονάδων, σχεδιάζει με **LineWidth = 3,**
- **Alpha_min+4*L/5** μονάδων, σχεδιάζει με **LineWidth = 7,**
- **αλλιώς** σχεδίαζει με **LineWidth = 9.**

Όπου ***L= Alpha_max-Alpha_min*** με Alpha_max, Alpha_min να καθορίζονται από εμάς. Π.χ. στην συγκεκριμένη περίπτωση έχουμε Alpha_max=1.5 και Alpha_min=0.

Χρόνος βελτιστοποίησης σε αυτό το παράδειγμα ήταν 22 λεπτά, σε υπολογιστικό σύστημα Windows 7, με διπύρηνο επεξεργαστή 1GHz και 6GB RAM, Matlab 2012a.

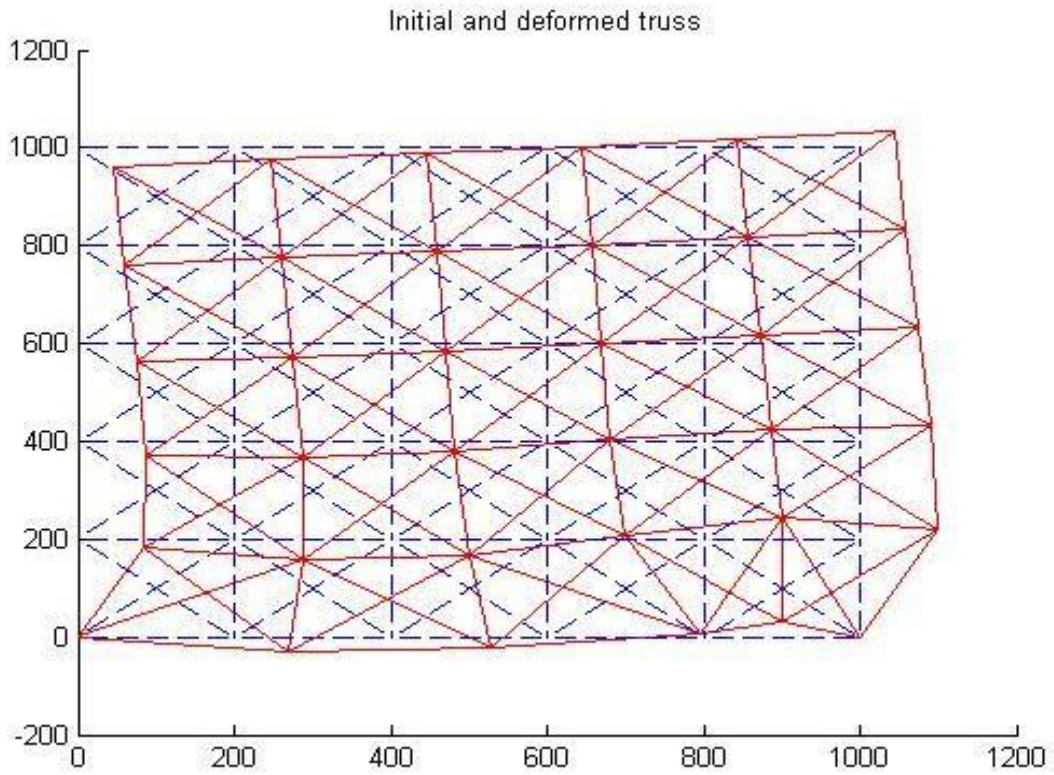
3.1.2 Πρόβλημα 2

Σε αυτό το σημείο θα επιλύσουμε ένα ακόμα παράδειγμα λίγο διαφοροποιημένο από το προηγούμενο.

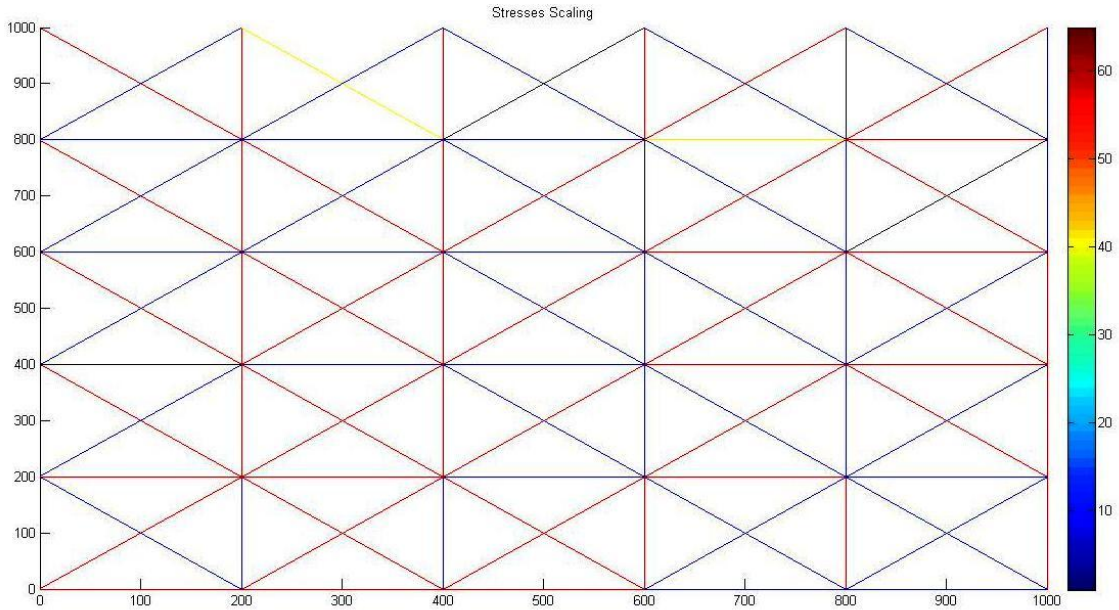
Στο παράδειγμα που εφαρμόσαμε θέσαμε lx= 1000, ly= 1000, nx=5, ny=5, με ασκούμενο φορτίο στον κόμβο (1,1) = (nx+2)/2 και στον κόμβο (1,3) = 200. Όσον αφορά στις συνοριακές συνθήκες έχουμε δύο, ενώ όλοι οι κόμβοι έχουν στηρίξεις. Οι ιδιότητες του υλικού περιλαμβάνονται στον κώδικα αλλά δεν αλλάζουν ουσιαστικά το αριθμητικό αποτέλεσμα. Ο αλγόριθμος συγκλίνει πολύ γρήγορα και εδώ και ικανοποιούνται οι οριακές συνθήκες που είχαμε θέσει στον κώδικα.

```
%%Algorithmos Veltistopoiisis

 nx=5;

 ny=5;

 nelements = (nx*ny)*4+ny+nx;

 x0=ones (nelements,1);

 lb=[0.1*ones(nelements,1)];

 ub=[101*ones(nelements,1)];

 options =optimset('LargeScale', 'on');

[x,fval,exitflag,output]=fmincon(@exoftrus2d2,x0,[],[],[],[
],lb,ub,@perior,options);
```

Η τιμή της αντικειμενικής συνάρτησης εδώ είναι fval =1.4981e+005. Η τιμή για την func2 = 4.3390e+006. Παρατηρούμε με κόκκινη γραμμή το παραμορφωμένο δικτύωμα ενώ με μπλε σκούρο είναι το αρχικό.



Και εδώ βλέπουμε και το διάγραμμα των τάσεων έτσι όπως αυτές διαμορφώθηκαν.



Χρόνος βελτιστοποίησης σε αυτό το παράδειγμα ήταν 23 λεπτά, σε υπολογιστικό σύστημα Windows XP, και 16GB RAM, Matlab 2008a.

4. Το πρόβλημα των τριών διαστάσεων

4.1 Ορισμός του προβλήματος

Όπως προκύπτει και από την τίτλο της εργασίας, σκοπός μας είναι η επίλυση ενός προβλήματος τριών διαστάσεων με βελτιστοποίηση του δικτυώματος. Η πολυπλοκότητα του προβλήματος καθιστά πολλές φορές αδύνατη την επίλυση μεγάλων προβλημάτων λόγω του υπερβολικού φόρτου που απαιτούν αυτά.

Στα παραδείγματα που ακολουθούν, γίνεται πάλι επίλυση με χρήση της συνάρτησης fmincon.

4.1.1. Πρόβλημα 1

Αρχικά, επιλύουμε ένα απλό πρόβλημα με διαστάσεις:

```
nx=2;  Lx=40; loads(inod1,1) = inod+inod1;  bcs(inod,1) = inod;

ny=2;  Ly=40; loads(inod1,2) = 0;           bcs(inod,2) = 1;
nz=1;  Lz=40; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 100;         bcs(inod,4) = 1;
```

Δηλαδή, το μήκος του κύβου που θα δημιουργηθεί είναι 40 μονάδες σε κάθε διάσταση, η διακριτοποίηση 2,2,1 , υπάρχουν φορτίσεις στον κόμβο (inod1,4) 100 μονάδων και έχουμε συνοριακές συνθήκες για τον κόμβο (inod,1), το σημείο εφαρμογής αυτών και στηρίξεις στους άλλους κόμβους.

Ο αλγόριθμος βελτιστοποίησης διαμορφώνεται εδώ ως εξής :

```
nx=2;
ny=2;
nz=1;
nnodes=(nx+1)*(ny+1)*(nz+1)
nelementsmax=25*nnodes;

 x0=ones(nelementsmax,1);
 lb=[0.1*ones(nelementsmax,1)];
 ub=[];

 options =optimset('LargeScale', 'on','Display','iter')

[x,fval,exitflag,output]=fmincon(@teliko,x0,[],[],[],[],lb,
ub,[],options).
```

Λόγω της χαμηλής πολυπλοκότητας του παραδείγματος αυτού, η επίλυση δεν καθυστερεί πολύ και το πρόβλημα λύνεται σε σχετικά χαμηλούς χρόνους, περίπου στα 20 λεπτά. Τα αρχεία για την επίλυση αυτή βρίσκονται στο φάκελο ΚΩΔΙΚΕΣ ΔΙΑΤΡΙΒΗΣ-ΚΕΦΑΛΑΙΟ 4-ΠΑΡΑΔΕΙΓΜΑ 1.

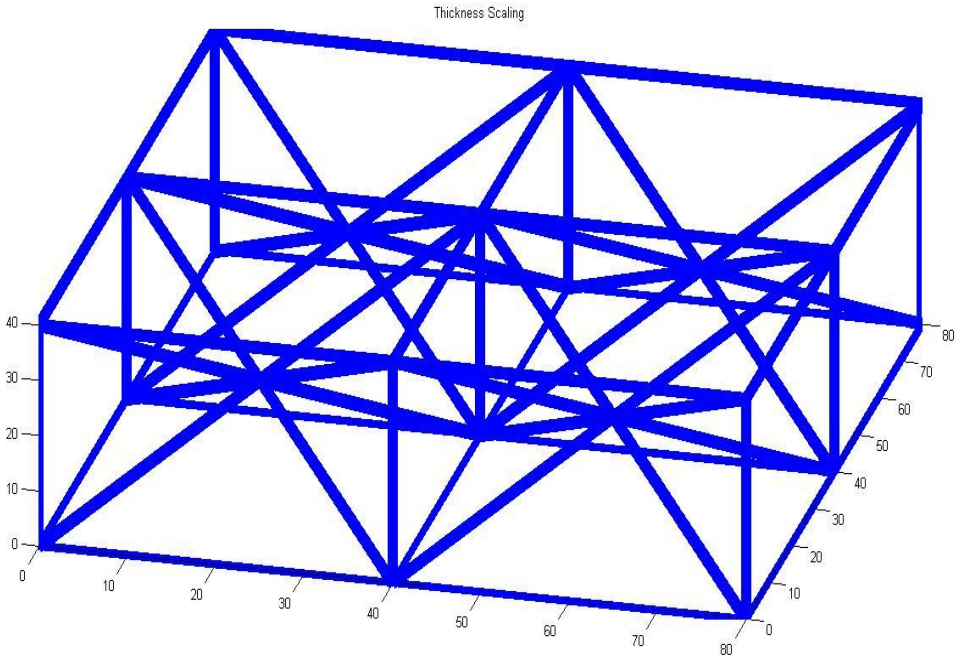
Ο αλγόριθμος τερματίζει στις 18επαναλήψεις, τις εξής:

Iter	F-count	f(x)	Max constraint	Line search steplength	Directional derivative	First-order optimality	Procedure
0	676	14125.9	-0.9				
1	1352	14.2347	-0.9	1	-3.95e+03	0.00378	
2	2028	14.2347	-0.9	1	-0.00512	0.00378	
3	2704	12.3389	-0.9	1	-0.00512	0.00149	Hessian modified
4	3380	10.8543	-0.9	1	-0.00287	0.000838	
5	4056	9.00981	-0.9	1	-0.0019	0.00057	
6	4732	7.22066	-0.9	1	-0.00116	0.000384	
7	5408	5.60139	-0.9	1	-0.000702	0.000256	
8	6084	4.29278	-0.9	1	-0.000409	0.000163	
9	6760	3.26509	-0.9	1	-0.000236	0.0001	
10	7436	2.47534	-0.9	1	-0.000135	6.01e-05	
11	8112	1.87284	-0.9	1	-7.73e-05	3.56e-05	
12	8788	1.41559	-0.9	1	-4.41e-05	2.08e-05	
13	9464	1.06936	-0.9	1	-2.52e-05	1.21e-05	
14	10140	0.807556	-0.9	1	-1.43e-05	6.99e-06	
15	10816	0.609745	-0.9	1	-8.18e-06	4.02e-06	
16	11492	0.46034	-0.9	1	-4.66e-06	2.31e-06	
17	12168	0.347526	-0.9	1	-2.66e-06	1.32e-06	
18	12844	0.26235	-0.9	1	-1.51e-06	7.58e-07	

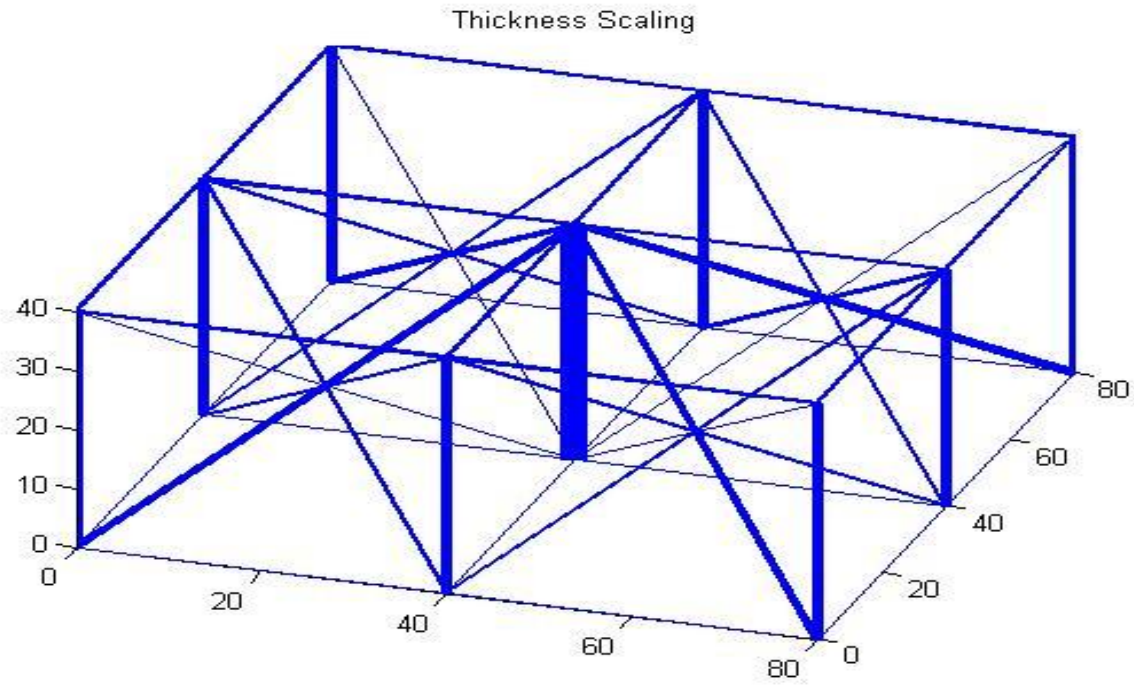
Φτάνοντας στη βέλτιστη τιμή της αντικειμενικής συνάρτησης fval = 0.2624, και μέγεθος βήματος 7.4562e+04.

Το δικτύωμα πλέον διαμορφώνεται ως εξής:

Αρχικό δικτύωμα:



Τελικό δικτύωμα:



Όπως παρατηρούμε, υπάρχουν κάποια πάχη που είναι μεγαλύτερα κ κάποιες διατομές που λόγω περιορισμών διαγράφονται με σκοπό την απλοποίηση του δικτυώματος και κατά συνέπεια την βελτιστοποίηση του.

Η διαφοροποίηση για τα πάχη των δικτυωμάτων έγινε ως εξής:

```
if x(i)<=Alpha_min+L/32*10^2
    Thickness = Thickness_matrix(1,1);
 elseif x(i)<=Alpha_min+2*L/30*10^2
    Thickness = Thickness_matrix(1,2);
 elseif x(i)<=Alpha_min+2*L/6*10^2
    Thickness = Thickness_matrix(1,3);
 elseif x(i)<=Alpha_min+3*L/5*10^2
    Thickness = Thickness_matrix(1,4);
 elseif x(i)<=Alpha_min+4*L/5*10^2
    Thickness = Thickness_matrix(1,5);
    else
    Thickness = Thickness_matrix(1,6);
 end
```

Ο αλγόριθμος δηλαδή, διαβάζει το x(i) στην εκάστοτε επανάληψη και ανάλογα δημιουργεί τα πάχη. Ο χρόνος που απαιτείται για να τρέξει η βελτιστοποίηση είναι 36 λεπτά.

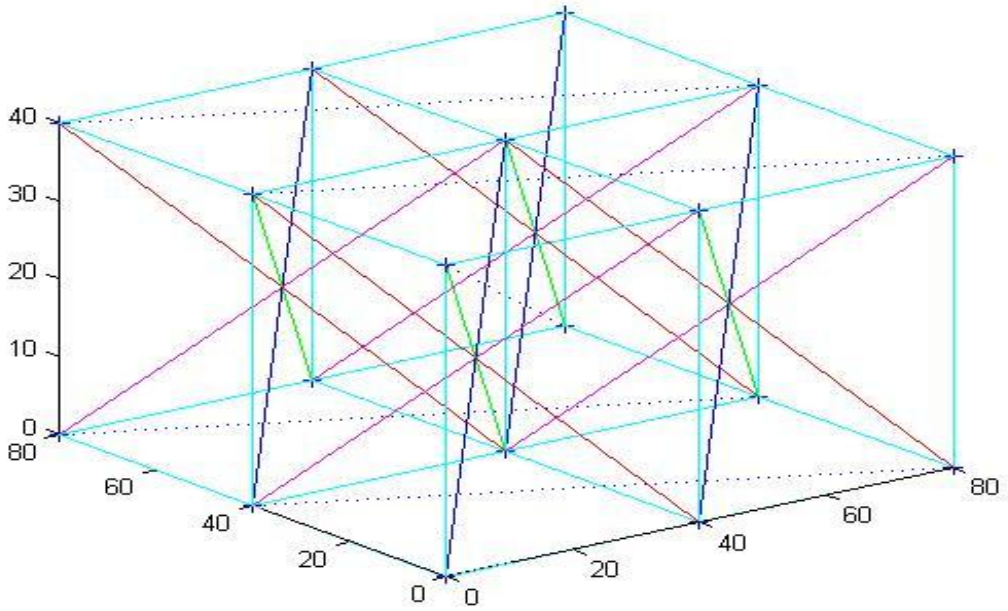
Χρόνος βελτιστοποίησης σε αυτό το παράδειγμα ήταν 37 λεπτά, σε υπολογιστικό σύστημα Windows 7, με διπύρηνο επεξεργαστή 1GHz και 6GB RAM, Matlab 2012a.

4.2.2 Πρόβλημα 2

```
nx=2;  Lx=40; loads(inod1,1) = inod+inod1;  bcs(inod,1) = inod;

ny=2;  Ly=40; loads(inod1,2) = 0;           bcs(inod,2) = 1;
nz=1;  Lz=40; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 100;         bcs(inod,4) = 1;
```

Δηλαδή, το μήκος του κύβου που θα δημιουργηθεί είναι 40 μονάδες σε κάθε διάσταση, η διακριτοποίηση 2,2 1 , υπάρχουν φορτίσεις στον κόμβο (inod1,1) και έχουμε συνοριακές συνθήκες για τον κόμβο (inod,1), το σημείο εφαρμογής αυτών και στηρίξεις στους άλλους κόμβους.

Το αρχικό κέλυφος που δημιουργείται έχει ως εξής.



Όπως παρατηρούμε, η επίλυση του προβλήματος είναι ακριβώς η ίδια. Λόγω της χαμηλής πολυπλοκότητας του παραδείγματος αυτού, η επίλυση δεν καθυστερεί πολύ και το πρόβλημα λύνεται σε σχετικά χαμηλούς χρόνους, περίπου στα 35 λεπτά. Τα αρχεία για την επίλυση αυτή βρίσκονται στο φάκελο ΚΩΔΙΚΕΣ ΔΙΑΤΡΙΒΗΣ-

ΚΕΦΑΛΑΙΟ 4-ΠΑΡΑΔΕΙΓΜΑ 2.

Iter	F-count	f(x)	Max constraint	Line search steplength	Directional derivative	First-order optimality	Procedure
0	451	14125.9	-0.9				
1	902	14.2347	-0.9	1	-3.95e+03	0.00378	
2	1353	14.2347	-0.9	1	-0.00512	0.00378	
3	1804	12.3389	-0.9	1	-0.00512	0.00149	Hessian modified
4	2255	10.8543	-0.9	1	-0.00287	0.000838	
5	2706	9.00978	-0.9	1	-0.0019	0.00057	
6	3157	7.22066	-0.9	1	-0.00116	0.000384	
7	3608	5.60136	-0.9	1	-0.000702	0.000256	
8	4059	4.29279	-0.9	1	-0.000409	0.000163	
9	4510	3.26509	-0.9	1	-0.000236	0.0001	
10	4961	2.47534	-0.9	1	-0.000135	6.01e-05	
11	5412	1.87284	-0.9	1	-7.73e-05	3.56e-05	
12	5863	1.4156	-0.9	1	-4.41e-05	2.08e-05	
13	6314	1.06936	-0.9	1	-2.52e-05	1.21e-05	
14	6765	0.807558	-0.9	1	-1.43e-05	6.99e-06	
15	7216	0.609744	-0.9	1	-8.18e-06	4.02e-06	
16	7667	0.460342	-0.9	1	-4.66e-06	2.31e-06	
17	8118	0.347526	-0.9	1	-2.66e-06	1.32e-06	
18	8569	0.262352	-0.9	1	-1.51e-06	7.58e-07	

Προφανώς και fval = 0.2624

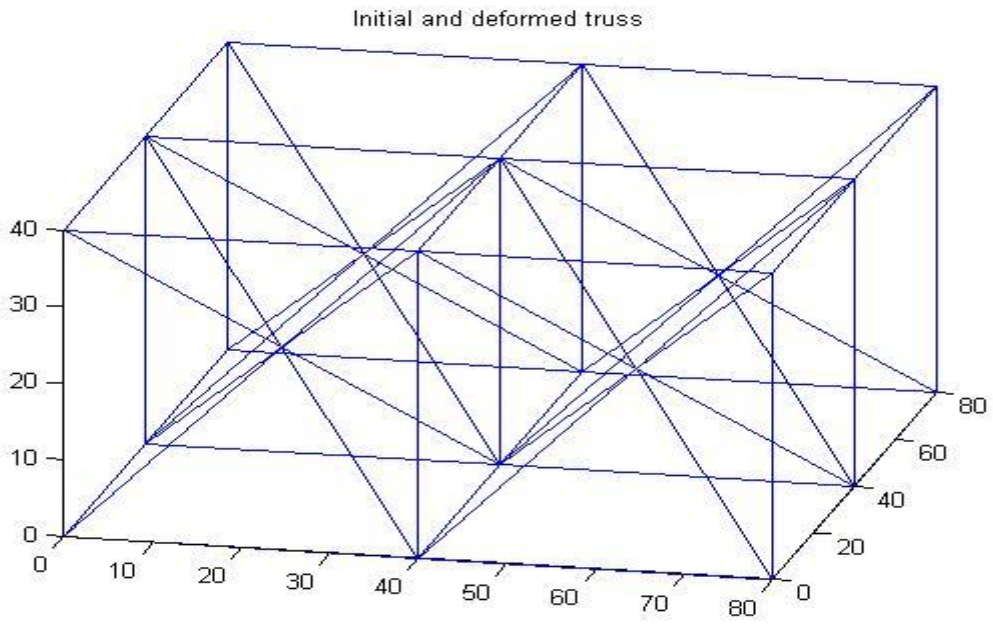
Ο αλγόριθμος βελτιστοποίησης δίνεται ως εξης:

```
%%Algorithmos Veltistopoiisis 3d

nx=2;
ny=2;
nz=1;
nnodes=(nx+1)*(ny+1)*(nz+1)
nelementsmax=18*nnodes;

 x0=ones(nelementsmax,1);

 %Oria pou mporei na kinithoun oi times tou A

 lb=[0.1*ones(nelementsmax,1)];
 ub=[];

 %options =optimset('Display','iter')
 options =optimset('LargeScale', 'on','Display','iter')
```

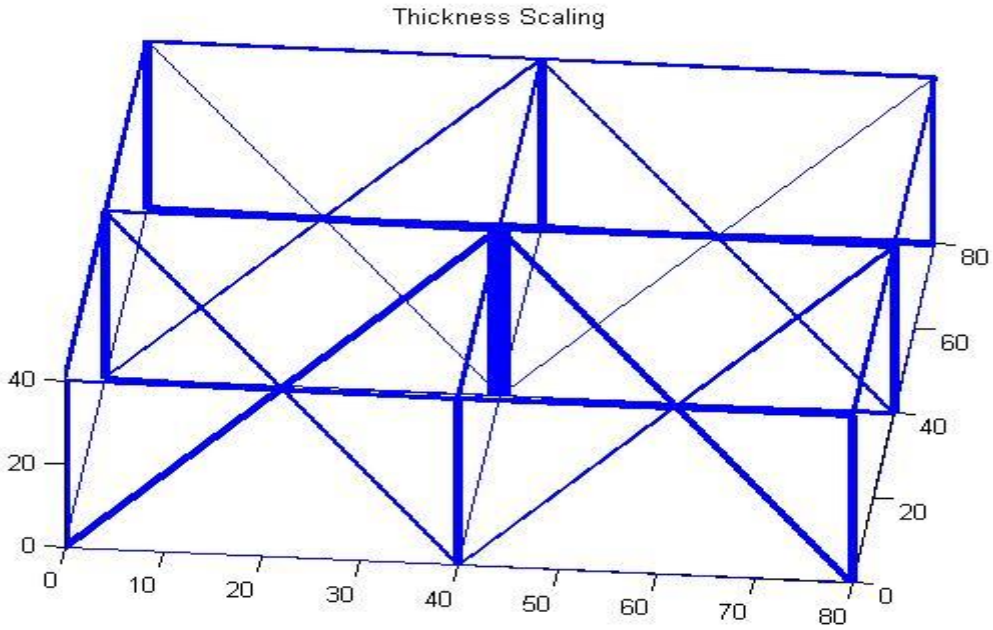
```
[x,fval,exitflag,output]=fmincon(@ground3truss2,x0,[],[],[]
,[],lb,ub,[],options)
```

Τα γραφικά αποτελέσματα είναι παρόμοια.

Αρχικό γράφημα:



Τελικό γράφημα:



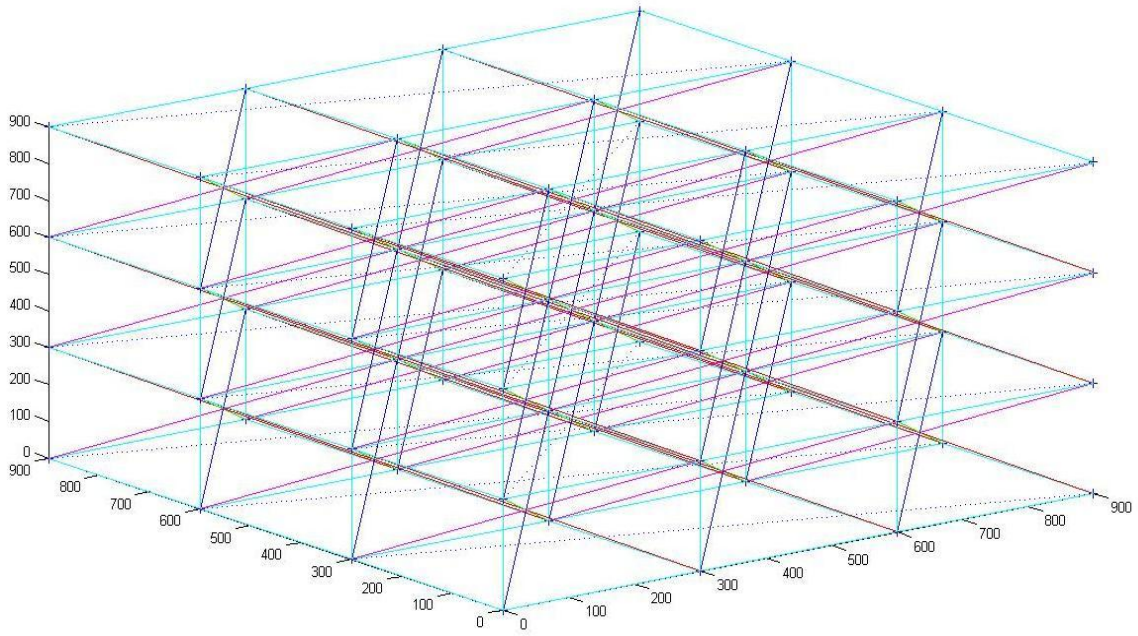
Χρόνος βελτιστοποίησης σε αυτό το παράδειγμα ήταν 37 λεπτά, σε υπολογιστικό σύστημα Windows 7, με διπύρηνο επεξεργαστή 1GHz και 6GB RAM, Matlab 2012a. ‘Ιδιο χρόνο έκανε και στον Windows Server 2008 του Πολυτεχνείου Κρήτης που υποστηρίζεται από 16GB RAM.

4.2.3 Πρόβλημα 3

Στο τρίτο πρόβλημα που θα επιλύσουμε με τον αλγόριθμο, θα έχουμε τα εξής ορίσματα:

```
nx=3; Lx=300; loads(inod1,1) = inod+inod1;  bcs(inod,1) = inod;
ny=3; Ly=300; loads(inod1,2) = 0;           bcs(inod,2) = 1;
nz=3; Lz=300; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 100;         bcs(inod,4) = 1;
```

Το αρχικό κέλυφος που δημιουργεί ο κώδικας με τα συνδεόμενα στοιχεία είναι το εξής:

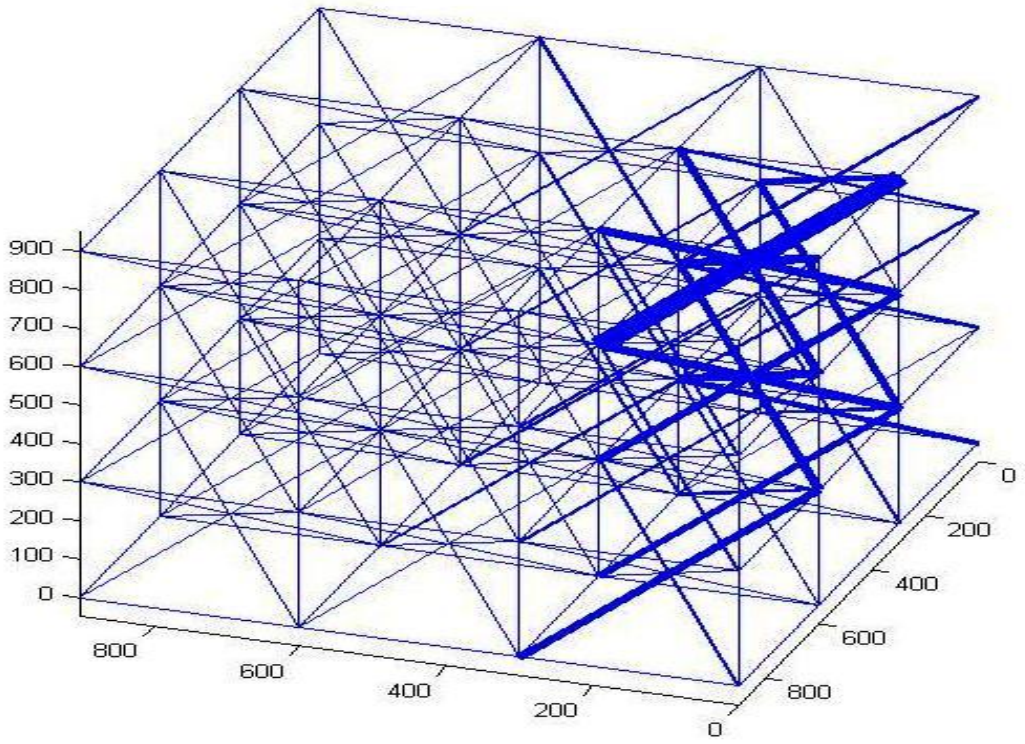


Παρατίθενται ενδεικτικά κάποιες τιμές για το X

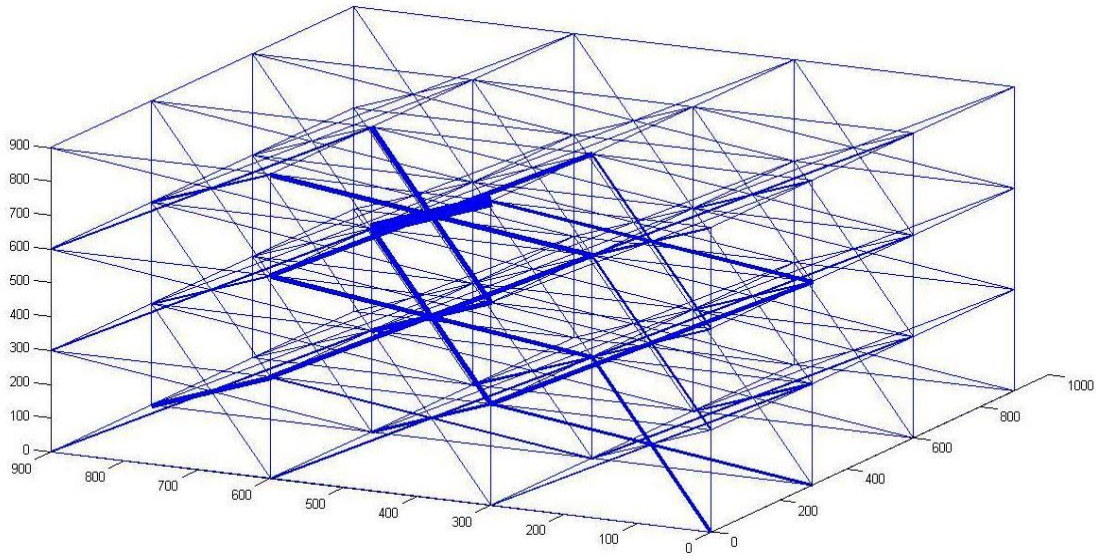
x = 1.2407
42.3341
6.5410
6.5410
6.5398
6.5415
1.2408
42.3341
6.5406
6.5398

42.3354
1.2408
42.3353
1.2410
6.5410
6.5387
50.6390
76.1220

Ο αλγόριθμος έκανε 18 επαναλήψεις για να φτάσει στο τελικό σημείο, όπου fval= 0.2624. Το νέο πλοτάρισμα που προκύπτει έχει ως εξής:



Και σε μια διαφορετική γωνία του:



Όπως ήταν αναμενόμενο, στα σημεία εφαρμογής της δύναμης οι ράβδοι έχουν μεγαλύτερο πάχος. Ο χρόνος που χρειάστηκε το παράδειγμα αυτό για να τρέξει ήταν 1 ώρα και 46 λεπτά, σε υπολογιστικό σύστημα Windows 7, με διπύρηνο επεξεργαστή 1GHz και 6GB RAM, Matlab 2012a.

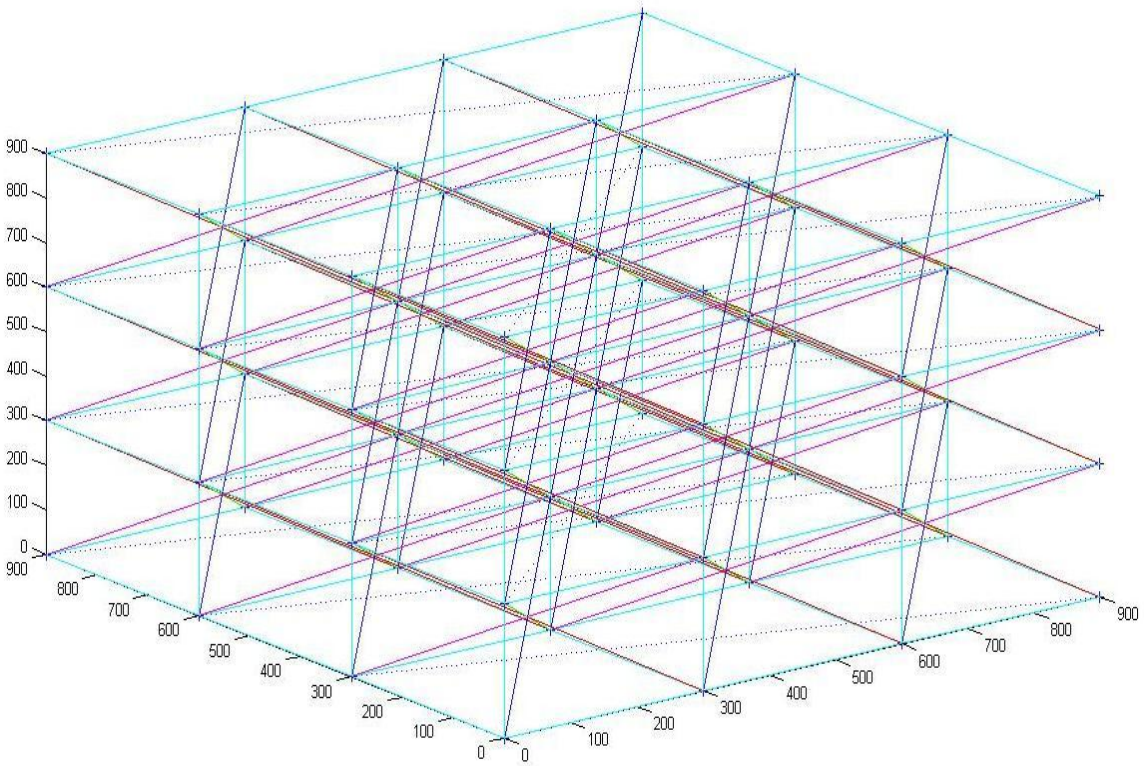
4.2.4 Πρόβλημα 4

Στο τέταρτο πρόβλημα που θα επιλύσουμε με τον αλγόριθμο, θα έχουμε τα εξής ορίσματα:

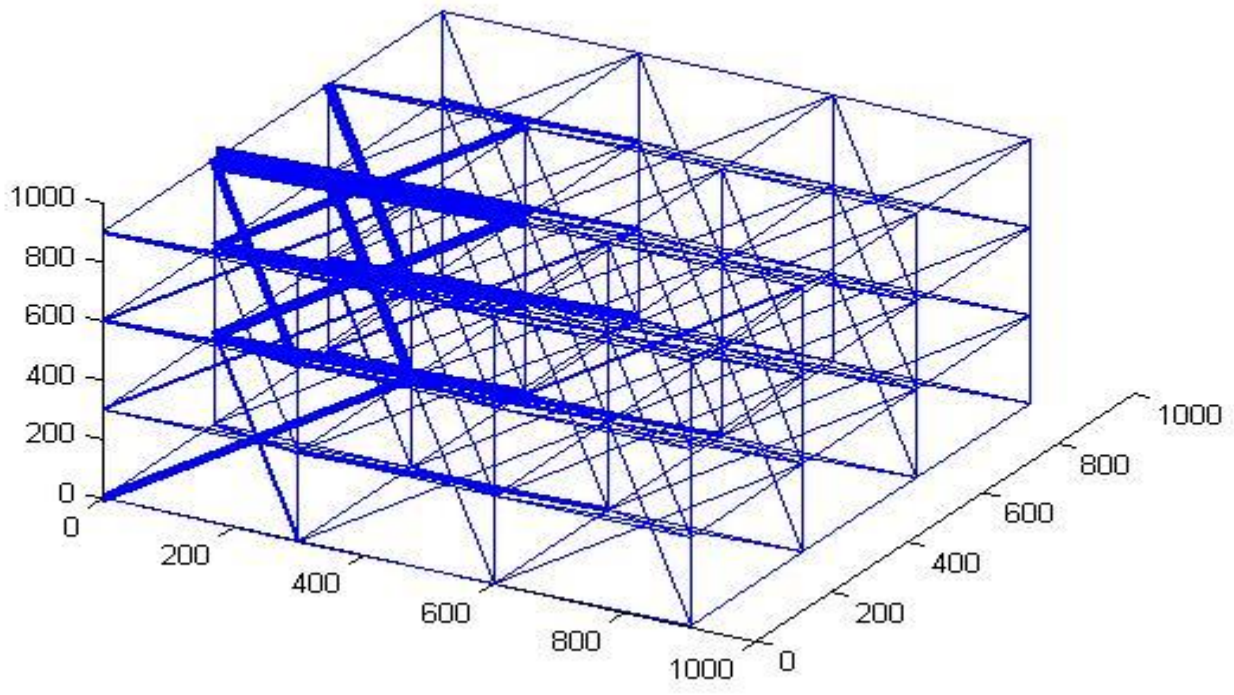
```
nx=3; Lx=300; loads(inod1,1) = inod+inod1;  bcs(inod,1) = inod;
ny=3; Ly=300; loads(inod1,2) = 100;         bcs(inod,2) = 1;
nz=3; Lz=300; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 0;           bcs(inod,4) = 1;
```

Αλλάζουμε δηλαδή το σημείο εφαρμογής της δύναμης σε σχέση με το προηγούμενο πρόβλημα για να δούμε τις αλλαγές που παρατηρούνται στο δικτύωμα. Και εδώ, ο αλγόριθμος συγκλίνει στις 18 επαναλήψεις.

Το αρχικό κέλυφος που δημιουργείται από τον κώδικα φαίνεται εδώ



Το νέο δικτύωμα διαμορφώθηκε ως εξής:



Όπως ήταν αναμενόμενο τα πάχη των γραμμών είναι μεγαλύτερα στα σημεία εφαρμογής της δύναμης που ορίσαμε. Ο χρόνος που χρειάστηκε σε αυτή την βελτιστοποίηση ήταν 1 ώρες και 50 λεπτά, Χρόνος βελτιστοποίησης σε αυτό το παράδειγμα ήταν 37 λεπτά, σε υπολογιστικό σύστημα Windows 7, με διπύρηνο επεξεργαστή 1GHz και 6GB RAM, Matlab 2012a.

4.2.5 Πρόβλημα 5

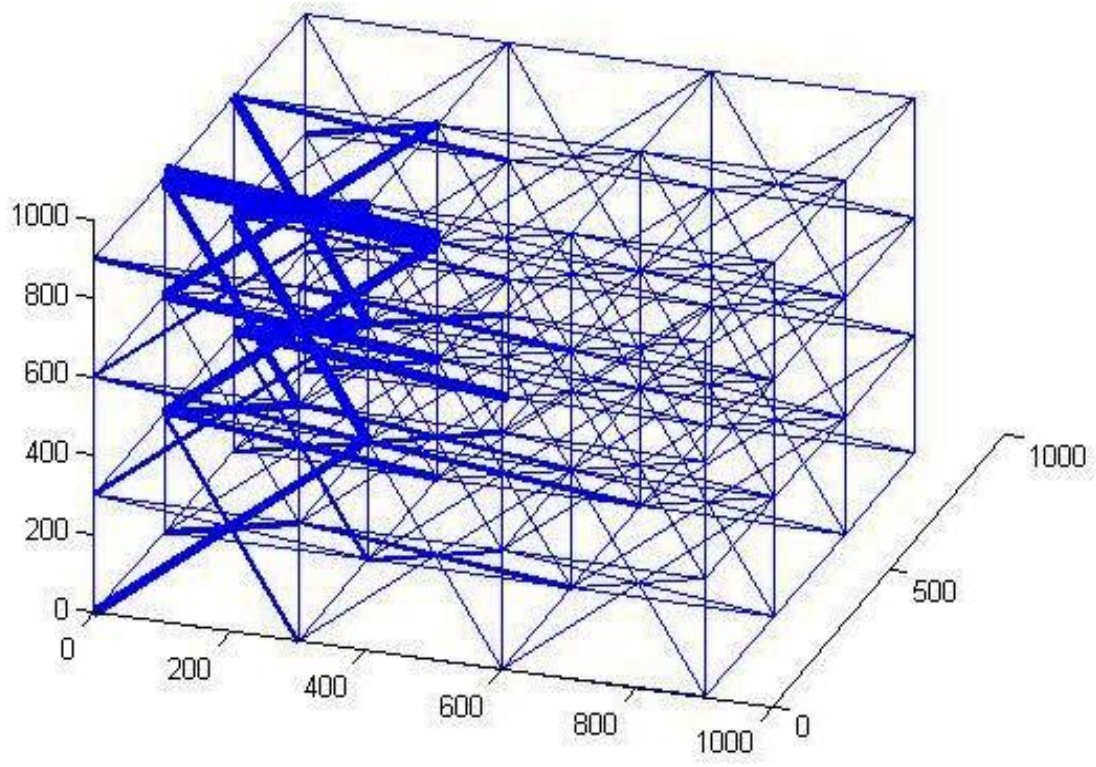
Στο πέμπτο πρόβλημα που θα επιλύσουμε με τον αλγόριθμο, θα έχουμε τα εξής ορίσματα:

```
nx=3; Lx=300; loads(inod1,1) = inod+inod1;  bcs(inod,1) = 1;
ny=3; Ly=300; loads(inod1,2) = 100;         bcs(inod,2) = 1;
nz=3; Lz=300; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 0;           bcs(inod,4) = 0;
```

Αλλάζουμε δηλαδή τις στηρίξεις για να δούμε πώς θα τροποποιηθεί το νέο δικτύωμα. Παραθέτουμε ενδεικτικά κάποιες τιμές από το X.

x =
1.2407
42.3341
6.5410
6.5410
6.5398
6.5415
1.2408
42.3341
6.5406
6.5398
42.3354
1.2408
42.3353
1.2410
6.5410
6.5387

Το νέο δικτύωμα διαμορφώθηκε ως εξής



Ο χρόνος που πήρε αυτό το παράδειγμα για να τρέξει η βελτιστοποίηση ήταν 2ώρες και 15 λεπτά. Το παράδειγμα έτρεξε σε Windows Server 2008 του Πολυτεχνείου Κρήτης που υποστηρίζεται απο16GB RAM.

4.2.6 Πρόβλημα 6

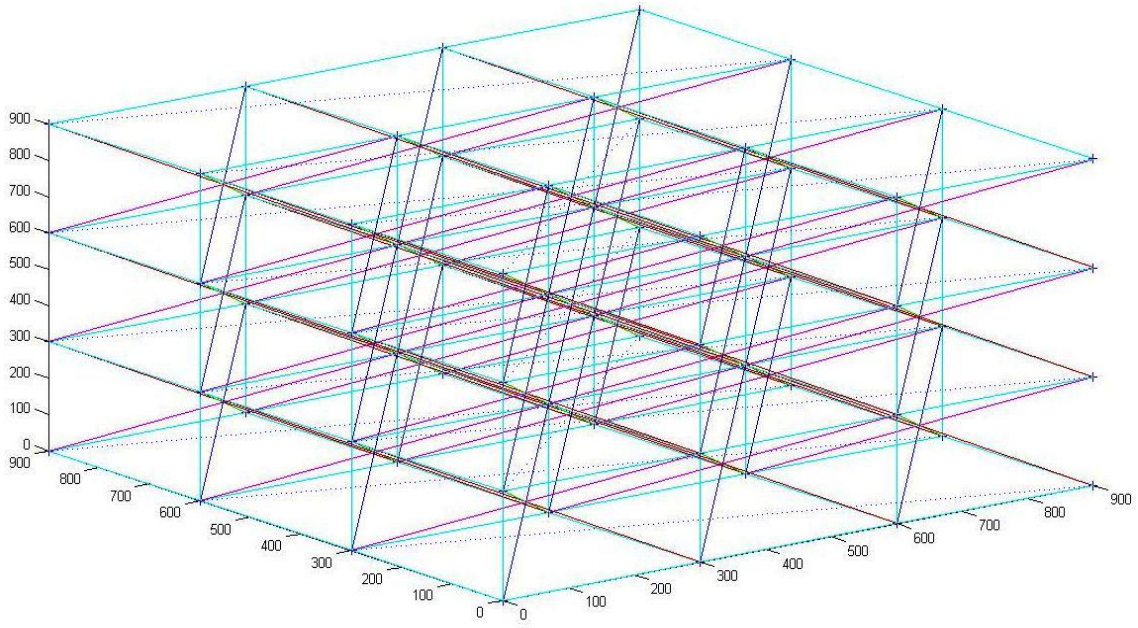
Σε αυτό το σημείο θα επιλύσουμε ένα ακόμα πρόβλημα με τις εξής αρχικές τιμές:

```
nx=3; Lx=300; loads(inod1,1) = inod+inod1;  bcs(inod,1) = inod;
ny=3; Ly=300; loads(inod1,2) = 0;           bcs(inod,2) = 1;
nz=3; Lz=300; loads(inod1,3) = 0;           bcs(inod,3) = 1;
              loads(inod1,4) = 100;         bcs(inod,4) = 1;
```

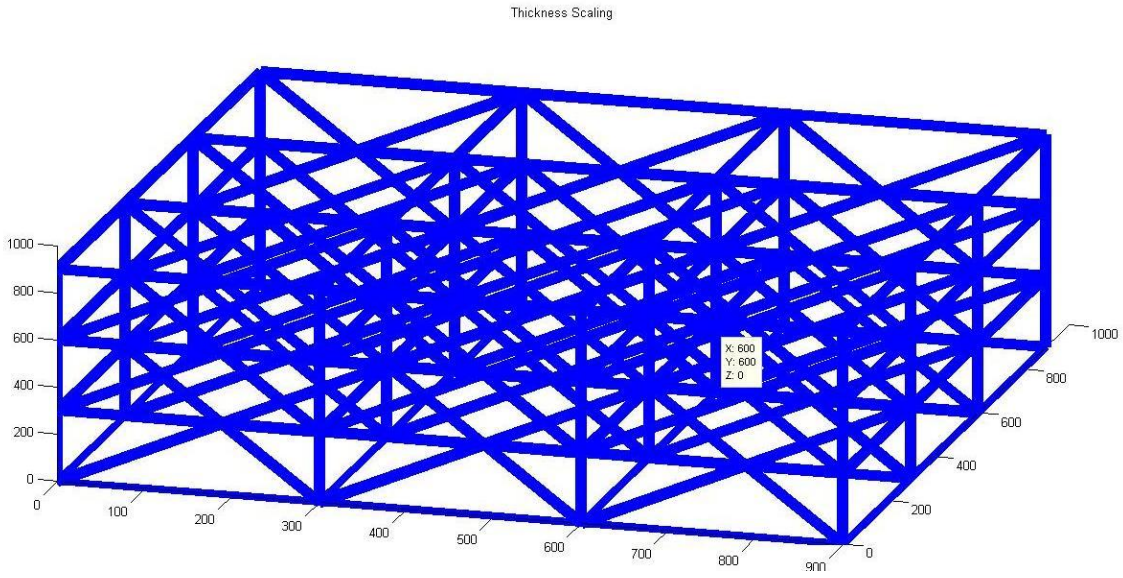
Ο αλγόριθμος βελτιστοποίησης δίνεται ως εξής:

```
%%Algorithmos Veltistopoiisis 3d
nx=3;
ny=3;
nz=3;
nnodes=(nx+1)*(ny+1)*(nz+1)
nelementsmax=25*nnodes;

 x0=ones(nelementsmax,1);

 lb=[0.1*ones(nelementsmax,1)];
 ub=[];
options =optimset('LargeScale', 'on','Display','iter')

[x,fval,exitflag,output]=fmincon(@ground3dtruss2,x0,[],[],[
],[],lb,ub,[],options)
```

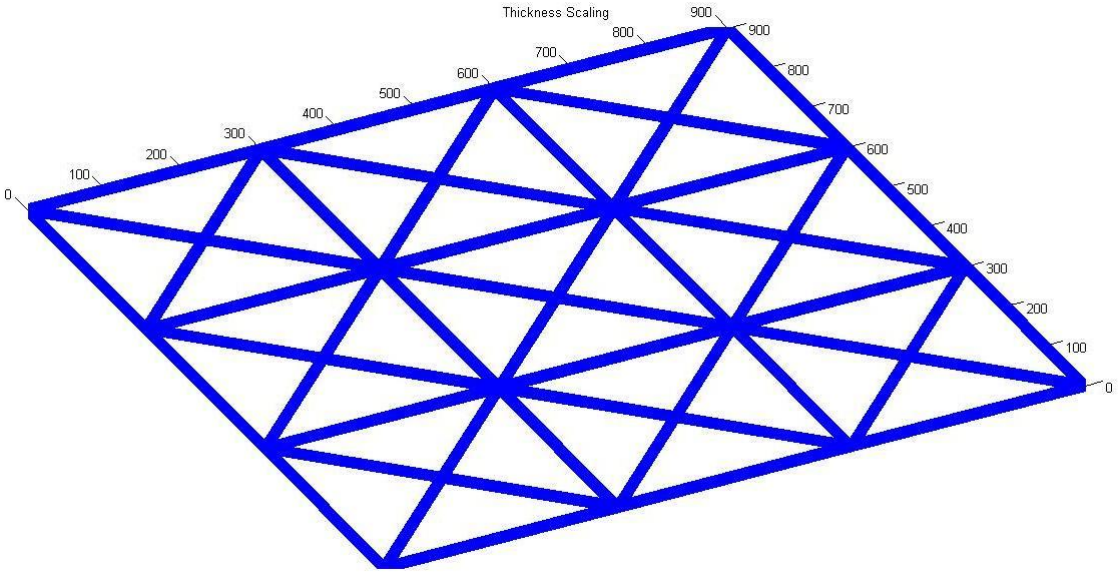
Ο αλγόριθμος εδώ κάνει μόνο δύο επαναλήψεις και ικανοποιούνται οι οριακές συνθήκες που έχουμε θέσει. Το αρχικό κέλυφος που δημιουργείται από τον κώδικα έχει την παρακάτω μορφή.



Το δικτύωμα που δημιουργείται από τον αλγόριθμο έχει τν παρακάτω μορφή (πλάγια όψη)



Κάτοψη



Η τιμή της αντικειμενικής συνάρτησης είναι fval = 72.9118, και αλγόριθμος συγκλίνει αμέσως, στα βήματα:

```
Max     Line search  Directional  First-order

 Iter F-count         f(x)    constraint    steplength    derivative    optimality
Procedure

    0    1601 2.92154e+006         -0.9

    1    3202      72.9118         -0.9             1    -3.46e+005    4.28e-005
```

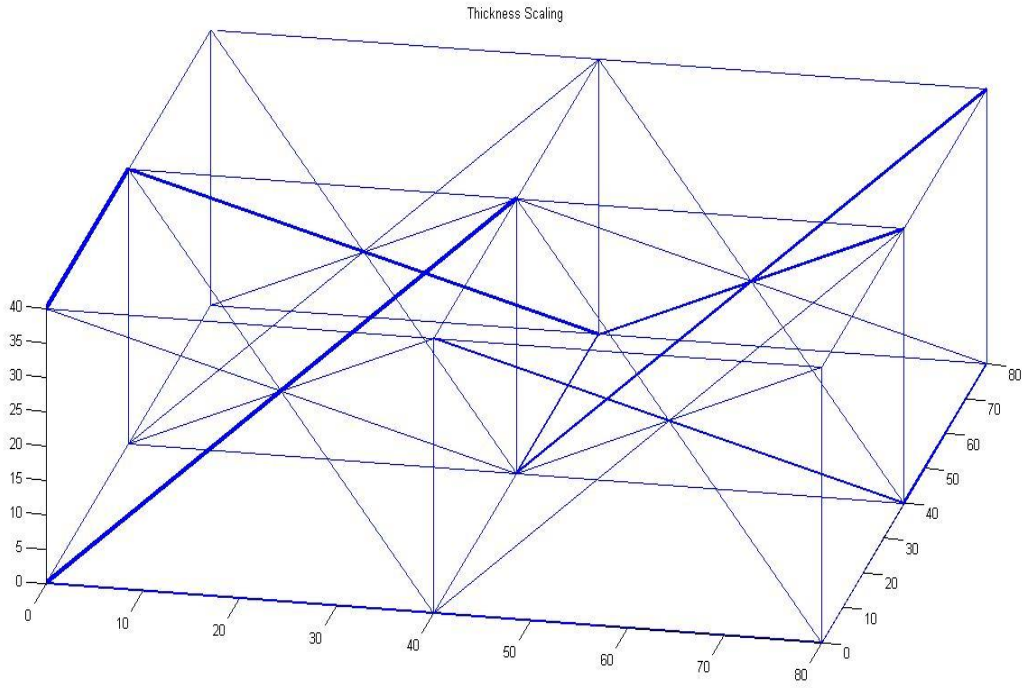
Κάποιες τιμές για το χ δίνονται παρακάτω :

x =

1.9434
11.2380
3.3970
3.3969
5.0381
3.0804
0.9837
0.9837
7.3545
1.4960

Το βήμα εδώ είναι 2.2115e-04.

Το νέο πλοτάρισμα που προκύπτει είναι το παρακάτω:



Ο χρόνος βελτιστοποίησης ήταν 43 λεπτά, το πρόβλημα αυτό επιλύθηκε στον Windows Server 2008, και έγινε με το πρόγραμμα της Matlab αυτή τη φορά στην έκδοση R 2014a. Οι συνοριακές συνθήκες ικανοποιήθηκαν εύκολα εδώ, και αυτό και έχουμε μόνο 2 επαναλήψεις και ο αλγόριθμος τερμάτισε.

5. Συμπεράσματα

Όπως παρατηρήθηκε, τόσο ο αλγόριθμος του Ν. Σκάρου τόσο και ο αλγόριθμος για το τρισδιάστατο δικτύωμα που μελετήθηκε εδώ, λειτουργούν σωστά και παράγουν καλά αποτελέσματα.

Ο βαθμός δυσκολίας τόσο στην κατανόηση, όσο και στην επίλυση των 3-Δ δικτυωμάτων και γενικότερα προβλημάτων , έγκειται στην περιπλοκότητα που προκύπτει λόγω της επιπλέον αυτής διάστασης. Η επίλυση του 3-Δ προβλήματος απαιτεί αρχικά την εμπέδωση του χώρου, και των μετακινήσεων που λαμβάνουν χώρα μέσα σε αυτόν.

Η χρήση πολύπλοκων μαθηματικών σχέσεων και εξισώσεων βοηθά μεν στην επίλυση του προβλήματος, ωστόσο δεν λύνει το πρόβλημα της εμπέδωσης του. Το «πλοτάρισμα» των δικτυωμάτων και κατ' επέκταση η απεικόνιση τους είναι αυτή που μας βοηθά να αντιληφθούμε το πρόβλημα και την λύση του.

Το πρόβλημα που είχαμε να λύσουμε εδώ ήταν η τοπολογική βελτιστοποίηση των διατομών σε ένα δικτύωμα για συγκεκριμένους περιορισμούς. Όταν έχουμε περιορισμούς, το σύστημα δεν είναι ελεύθερο, και η όλη επίδραση των περιορισμών υλοποιείται μέσα τους πολλαπλασιαστές Lagrange. Αν υπάρχει τρόπος να υπολογιστούν αυτά τα λ τότε μπορώ να παραμερίσω το A*u=b. Όσον αφορά στις στηρίξεις, είναι ένα ιδιάζον πρόβλημα με μηδενικές ιδιοτιμές.

Γενικά, προσπαθήσαμε να αντιστοιχήσουμε κάθε παράμετρο σχεδιασμού με ένα πεπερασμένο στοιχείο και τα κριτήρια βέλτιστου να τα ανάγουμε σε πεπερασμένα στοιχεία. Θέλουμε να επιλέξουμε διατομές που να επιτύχουν βέλτιστη αξιοποίηση υλικού και οι τάσεις στο όριο σχεδιασμού. Αν η τάση < όριο-επιτρεπτό, τότε μειώνω την διατομή. Αν μεγαλώσουμε τη διατομή μειώνονται οι τάσεις.[6]

Σε περιπτώσεις που έχουμε μόνο μια παράπλευρη συνθήκη, την τοποθετώ όπου θέλω και μπορώ να κάνω παραλλαγές. Σε τέτοια προβλήματα μπορούμε να «ποινικοποιήσουμε» την θλίψη ή τον εφελκυσμό για να μην έχουμε φαινόμενα λυγισμού.

Το μεγάλο πρόβλημα που πρέπει να αντιμετωπίσουμε δυστυχώς σε αυτά τα προβλήματα, είναι ο μεγάλος αριθμός των βαθμών ελευθερίας των συστημάτων, καθώς και η χρονοτριβή που προκαλείται από αυτόν. Τα μικρά προβλήματα μπορούν να επιλυθούν σε περίπου μισή με μια ώρα, χωρίς αυτό να είναι απόλυτο. Μεγαλύτερα προβλήματα όπου nx,ny,nz> 5 η επίλυση είναι σχεδόν ανέφικτη σε συμβατικά pc, και απαιτεί ώρες μπορεί και μέρες.

Το σημαντικότερο όλων ωστόσο είναι ότι ο κώδικας βγάζει σωστά αποτελέσματα, με πολύ μικρή τροποποίηση, ακόμα κ για μεγάλα προβλήματα, και προβλήματα διαφορετικών υλικών ή κατασκευών.

5. Βιβλιογραφία

(1) http://users.civil.ntua.gr/papadrakakis/files/theses/abstracts/Abs_Plevris.pdf

(2) Μεταπτυχιακή εργασία με τίτλο *«Βέλτιστος σχεδιασμός κατασκευών με πολλαπλά κριτήρια με χρήση στρατηγικών εξέλιξης»*, Ευάγγελου Ε. Πλεύρη

(3) Σημειώσεις μαθήματος «*Βέλτιστος Σχεδιασμός Κατασκευών με Περιορισμούς Αξιοπιστίας και χρήση Ανταγωνιστικών ΓΑ*», Χρήστος Κ. Δήμου

(4) Μεταπτυχιακή εργασία με τίτλο *«Μεθοδολογία Βελτιστοποίησης πολλαπλών στόχων για την κατανομή πόρων στα τεχνικά έργα»* , Σοφία Π.Καϊφα

(5) Διπλωματική εργασία με τίτλο *«Δυναμική διερεύνηση πιεζοηλεκτρικών κατασκευών με χρήση πεπερασμένων στοιχείων»*, Σπυρίδων Γαλάνης

(6) Σημειώσεις μαθήματος «*Βέλτιστος Δομικός Σχεδιασμός Υλικών και Κατασκευών Βέλτιστος Δομικός Σχεδιασμός*», Γ.Σταυρουλάκη

(7) Σημειώσεις μαθήματος «*Προχωρημένα θέματα μη γραμμικού προγραμματισμού*», Ι. Παπαμιχαήλ

(8) Διπλωματική εργασία με τίτλο «*Βέλτιστος σχεδιασμός δικτυωτών φορέων με την βοήθεια της μεθόδου των πεπερασμένων στοιχείων και αλγόριθμος βελτιστοποίησης*», Ν.Σκάρος

(9) «*Improved genetic operators for structural engineering optimization*», J. P. B. Leite & B. H. V. Topping

(10) Διδακτορική Διατριβή με τίτλο *«Διερεύνηση της μεθόδου των πεπερασμένων όγκων στην ελαστικότητα και στην επίλυση ροϊκών προβλημάτων για την προσομοίωση διεργασιών παραγωγής αεροναυπηγικών συνθέτων υλικών με μεθόδους αναρρόφησης και μεταφοράς.»* Χρήστου Δ. Βρεττού

(11) «*Optimization of practical trusses with constraints on eigenfrequencies, displacements, stresses, and buckling*», N.L. Pedersen and A.K. Nielsen

(12) «*Algorithms in structural optimization: Looking ahead (back)* », M.P. Bendsøe

(13) «*Εφαρμογή της μεθόδου βελτιστοποίησης με αποικίες μυρμηγκιών σε τρισδιάστατα προβλήματα αντίστροφης σχεδίασης πτερυγίου στροβιλομηχανής*», Γιάννης Ρούσσος

(14) «*Βέλτιστος σχεδιασμός των κατασκευών», Α.Πλεύρης*

(15) «*Εντοπισμός ατελειών σε επίπεδη πλάκα ομογενούς και σύνθετου υλικού με χρήση νευρωνικών δικτύων και μεθόδου πεπερασμένων στοιχείων*», Ε.Ανδριανάκης

(16) «*Βέλτιστος σχεδιασμός των κατασκευών*», Β.Κουμούσης

(17) «*Ανάπτυξη μοντέλου πεπερασμένων στοιχείων για τη μελέτη της οστικής ανακατασκευής βραχέων οστών*», Γ. Σάββας

(18) «*Finite element model updating of a truss model using incomplete modal data*», Y.X. Zhang,S.H. Sim & B.F. Spencer, Jr

(19) ftp://teiser.gr/pliroforiki/Algorithmoi-DomesDedomenon/theNotesADS.pdf
(20) www.wikipedia.com
(21) http://ebooks.edu.gr/modules/ebook/show.php/DSGL-C101/36/198,1063/
(22) http://users.ntua.gr/caridis/methodoi/keimena/chap%2003/Chapter%2003.pdf

ΠΑΡΑΡΤΗΜΑ
ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Σε αυτό το κομμάτι θα εξηγήσουμε τον κώδικα που χρησιμοποιήσαμε για την επίλυση του τρισδιάστατου δικτυώματος.

Ανοίγοντας το αρχείο **groundtruss2d.m** βλέπουμε τον βασικό κώδικα που χρησιμοποιήσαμε και διαμορφώσαμε ανάλογα τις περιπτώσεις.

```
-----------------------------------------------------
function [func2]= ground3dtruss2(x)
```

Εδώ γίνεται ο ορισμός της συνάρτησης στόχου

```
-----------------------------------------------------
% creating 3d ground structure and solving it
% number of rows in x y z
nx=3;
ny=3;
nz=3;
% length of rows in x y z
lx=300;
ly=300;
lz=300;
% number of nodes
nnodes=(nx+1)*(ny+1)*(nz+1);
% coordinates
xcoo(nnodes)=0;
ycoo(nnodes)=0;
zcoo(nnodes)=0;
%
```

Σε αυτό το κομμάτι δίνουμε τον βαθμό της διακριτοποίησης σε κάθε διάσταση, το μήκος του δικτυώματος, τον αριθμό τον κόμβων του πλαισίου και ορίζονται οι αρχικές συντεταγμένες (0,0,0).

```
-----------------------------------------------------
for i=1:(nx+1)
    for j=1:(ny+1)
        for k=1:(nz+1)
            %
```

```
        inumber=(k-1)*(nx+1)*(ny+1)+(j-1)*(nx+1)+i;
        i;
        j;
        k;
        inumber;
        xcoo(inumber) = (i-1)*lx;
        ycoo(inumber) = (j-1)*ly;
        zcoo(inumber) = (k-1)*lz;
%        pause
        end
    end
end
 %
```

Εδώ φαίνεται το πώς θα τοποθετηθούν οι κόμβοι στο εσωτερικό του δικτυώματος, και η αρίθμησή τους.

```
 plot3(xcoo,ycoo,zcoo,'b+:')
 hold on
% pause
 % creating elements
 nelementsmax=25*nnodes;
 cnct(nelementsmax,2)=0;
 %
 nelem=0;
```

```
-----------------------------------------------------------
 outer shell
 for i=1:(nx)
     for j=0:(ny-1)
         for k=0:(nz-1)
             %
         a1=(i)+(j)*(nx+1)   +(k)*(nx+1)*(ny+1);
         b1=(i+1)   +(j)*(nx+1)   +(k)*(nx+1)*(ny+1);
         c1=(i)+(j)*(nx+1)   +(k+1)*(nx+1)*(ny+1);
         d1=(i+1)   +(j)*(nx+1)   +(k+1)*(nx+1)*(ny+1);
         a2=(i)+(j+1)*(nx+1)     +(k)*(nx+1)*(ny+1);
         b2=(i+1)   +(j+1)*(nx+1)     +(k)*(nx+1)*(ny+1);
         c2=(i)+(j+1)*(nx+1)     +(k+1)*(nx+1)*(ny+1);
         d2=(i+1)   +(j+1)*(nx+1)     +(k+1)*(nx+1)*(ny+1);
         a1;b1;c1;d1;
         a2;b2;c2;d2;
         nelem=nelem+1;
         cnct(nelem,1)=c1;
         cnct(nelem,2)=b2;
```

```
            %
%            pause
        %    a = [xcoo(arxi) xcoo(telos)]'
        %    b = [ycoo(arxi) ycoo(telos)]'
        %    c = [zcoo(arxi) zcoo(telos)]'
        %    plot3(a,b,c,'g-');
        %    pause
            %
            nelem=nelem+1;
            cnct(nelem,1)=a1;
            cnct(nelem,2)=d2;
            %
        %    a = [xcoo(cnct(nelem,2)} xcoo(cnct(nelem,3))]';
        %    b = [ycoo(cnct(nelem,2)) ycoo(cnct(nelem,3))]';
        %    c = [zcoo(cnct(nelem,2)) zcoo(cnct(nelem,3))]';
        %    plot3(a,b,c,'b-');
            %
            nelem=nelem+1;
            cnct(nelem,1)=b1;
            cnct(nelem,2)=c2;
            %
        %    a = [xcoo(cnct(nelem,2)} xcoo(cnct(nelem,3))]';
        %    b = [ycoo(cnct(nelem,2)) ycoo(cnct(nelem,3))]';
        %    c = [zcoo(cnct(nelem,2)) zcoo(cnct(nelem,3))]';
        %    plot3(a,b,c,'g-');
            %
            nelem=nelem+1;
            cnct(nelem,1)=d1;
            cnct(nelem,2)=a2;
            %
        %    a = [xcoo(cnct(nelem,2)} xcoo(cnct(nelem,3))]';
        %    b = [ycoo(cnct(nelem,2)) ycoo(cnct(nelem,3))]';
        %    c = [zcoo(cnct(nelem,2)) zcoo(cnct(nelem,3))]';
        %    plot3(a,b,c,'b-');
            %
            end
    end
end
%
nelem1=0;
% along axes z
for i=1:(nx+1)
    for j=0:(ny)
        a1=j*(nx+1)+i;
        for k=0:(nz-1)
            %
            nelem1=nelem1+1;
            nelem=nelem+1;
```

```
        a11= a1+ (k)*(nx+1)*(ny+1);
        a21= a1 + (k+1)*(nx+1)*(ny+1);
        cnct(nelem,1)=a11;
        cnct(nelem,2)=a21;
        end
    end
end
%
 % along axes y
for i=1:(nx+1)
    for k=0:(nz)
        a1=k*(nx+1)*(ny+1)+i;
        for j=0:(ny-1)
            nelem1=nelem1+1;
            nelem=nelem+1;
        a11= a1+ (j)*(nx+1);
        a21= a1 + (j+1)*(nx+1);
        cnct(nelem,1)=a11;
        cnct(nelem,2)=a21;
        end
    end
end
 % along axes x
for j=0:ny
    for k=0:(nz)
            a1=k*(nx+1)*(ny+1)+j*(nx+1);
        for i=1:(nx)
            nelem1=nelem1+1;
            nelem=nelem+1;
        a11= a1+ i;
        a21= a1 + i+1;
        cnct(nelem,1)=a11;
        cnct(nelem,2)=a21;
        end
    end
end
%
%nelem1
%nelem
%pause
%
nelem=0;
for i=1:(nx)
    for j=1:(ny)
        for k=1:(nz)
            %
        nelem=nelem+1;
        arxi=cnct(nelem,1);
```

```
            telos=cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
            b = [ycoo(arxi) ycoo(telos)]';
            c = [zcoo(arxi) zcoo(telos)]';
            plot3(a,b,c,'g-');
        %    pause
            %
            nelem=nelem+1;
            %
            arxi=cnct(nelem,1);
            telos=cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
            b = [ycoo(arxi) ycoo(telos)]';
            c = [zcoo(arxi) zcoo(telos)]';
            plot3(a,b,c,'b-');
            %
            nelem=nelem+1;          %
            arxi=cnct(nelem,1);
            telos=cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
            b = [ycoo(arxi) ycoo(telos)]';
            c = [zcoo(arxi) zcoo(telos)]';
            plot3(a,b,c,'r-');
            %
            nelem=nelem+1;
            %
            arxi=cnct(nelem,1);
            telos=cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
            b = [ycoo(arxi) ycoo(telos)]';
            c = [zcoo(arxi) zcoo(telos)]';
            plot3(a,b,c,'m-');
            %
            end
        end
    end
     % along axes z
    for i=1:(nx+1)
        for j=1:(ny+1)
            for k=0:(nz-1)
            nelem=nelem+1;
            arxi = cnct(nelem,1);
            telos = cnct(nelem,2);
                a = [xcoo(arxi) xcoo(telos)]';
            b = [ycoo(arxi) ycoo(telos)]';
            c = [zcoo(arxi) zcoo(telos)]';
            plot3(a,b,c,'c');
            end
```

```
    end
end
% along axes y
for i=1:(nx+1)
    for k=1:(nz+1)
        for j=0:(ny-1)
        nelem=nelem+1;
        arxi = cnct(nelem,1);
        telos = cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
        b = [ycoo(arxi) ycoo(telos)]';
        c = [zcoo(arxi) zcoo(telos)]';
        plot3(a,b,c,'c');
        end
    end
end
%
% along axes x
for j=1:(ny+1)
    for k=1:(nz+1)
        for i=0:(nx-1)
        nelem=nelem+1;
        arxi = cnct(nelem,1);
        telos = cnct(nelem,2);
            a = [xcoo(arxi) xcoo(telos)]';
        b = [ycoo(arxi) ycoo(telos)]';
        c = [zcoo(arxi) zcoo(telos)]';
        plot3(a,b,c,'c');
        end
    end
end
hold off
nelements = nelem;
%
```

Εδώ αφού δημιουργήθηκε το εξωτερικό κέλυφος του δικτυώματος, υπολογίζουμε τον αριθμό των στοιχείων που ενώνουν τους κόμβους, καθώς και τον τρόπο διασύνδεσής τους, και για τις τρεις διαστάσεις.

```
-----------------------------------------------------------
 %% material constants
E = zeros(nelementsmax,1);
A = zeros(nelementsmax,1);
for i=1:nelementsmax
    E(i) = 10000;
```

```
    A(i) = x(i);
end
A;
```

Εδώ δίνονται οι ιδιότητες του υλικού, το συντελεστής Young δηλαδή έχει υπολογιστεί ίσος με 10000.

```
-----------------------------------------------------------
 % loading boundary conditions etc

%% loading
% number of loads
nuload = (nx+1)*(ny+1);
% number of node, x - y loadings
loads = zeros(5000,4);
inod=nz*(nx+1)*(ny+1);
inod1=0;
for i=1:(nx+1)
    for j=1:(ny+1)
        inod1=inod1+1;
        loads(inod1,1) = inod+inod1;
        loads(inod1,2) = 0;
        loads(inod1,3) = 0;
        loads(inod1,4) = 100;
    end
end
%% boundary conditions
% number of boundary conditions
nubcs = (nx+1)*(ny+1);
% number of node, x - y displs (code 1 = 0, code 0 = free)
bcs = zeros(5000,4);
for i=1:(nx+1)
    for j=1:(ny+1)
        inod=(j-1)*(nx+1)+i;
bcs(inod,1) = inod;
bcs(inod,2) = 1;
bcs(inod,3) = 1;
bcs(inod,4) = 1;
    end
end
```

Εδώ δίνουμε τον αριθμό των φορτίσεων, το σημείο εφαρμογής τους, την κατεύθυνσής τους και το μέτρο τους. Στη συνέχεια, γίνεται ο ορισμός των οριακών συνθηκών, το πού έχουμε στηρίξεις δηλαδή, και το είδος της στήριξης, δηλαδή 0= ελεύθερη στήριξη.

```
-----------------------------------------------------------
%% preparation
% stiffnes matrix
Ktotal = zeros(3*nnodes,3*nnodes);
% loading vector
Ftotal = zeros(3*nnodes,1);
% space for the solution
Utotal = zeros(3*nnodes,1);
% space for local stiffness matrix
Kelm = zeros(6,6);
%
% assemply of the stiffness matrix
for i=1:nelements
    % for element i
    % first node number cnct(i,1)
    % first node coordinates xnode(cnct(i,1)),
ynode(cnct(i,1))
    P1 = [ xcoo(cnct(i,1)), ycoo(cnct(i,1)),
zcoo(cnct(i,1)) ];
    P2 = [ xcoo(cnct(i,2)), ycoo(cnct(i,2)),
zcoo(cnct(i,2)) ];
    % local 4x4 stiffness matrix
    Kelm = truss3d(A(i),E(i),P1,P2);
    % assemply in global Ktotal stiffness matrix
    Ktotal((cnct(i,1)-1)*3+1:(cnct(i,1)-1)*3+3,(cnct(i,1)-
1)*3+1:(cnct(i,1)-1)*3+3)= ...
    Ktotal((cnct(i,1)-1)*3+1:(cnct(i,1)-1)*3+3,(cnct(i,1)-
1)*3+1:(cnct(i,1)-1)*3+3)+Kelm(1:3,1:3);
    Ktotal((cnct(i,1)-1)*3+1:(cnct(i,1)-1)*3+3,(cnct(i,2)-
1)*3+1:(cnct(i,2)-1)*3+3)= ...
    Ktotal((cnct(i,1)-1)*3+1:(cnct(i,1)-1)*3+3,(cnct(i,2)-
1)*3+1:(cnct(i,2)-1)*3+3)+Kelm(1:3,4:6);
    Ktotal((cnct(i,2)-1)*3+1:(cnct(i,2)-1)*3+3,(cnct(i,1)-
1)*3+1:(cnct(i,1)-1)*3+3)= ...
    Ktotal((cnct(i,2)-1)*3+1:(cnct(i,2)-1)*3+3,(cnct(i,1)-
1)*3+1:(cnct(i,1)-1)*3+3)+Kelm(4:6,1:3);
    Ktotal((cnct(i,2)-1)*3+1:(cnct(i,2)-1)*3+3,(cnct(i,2)-
1)*3+1:(cnct(i,2)-1)*3+3)= ...
    Ktotal((cnct(i,2)-1)*3+1:(cnct(i,2)-1)*3+3,(cnct(i,2)-
1)*3+1:(cnct(i,2)-1)*3+3)+Kelm(4:6,4:6);
End

% loading vector
%
for i=1:nuload
    % load at node loads(i,1) with x-y contribution equal
to loads(i,2), loads(i,3)
    Ftotal(3*(loads(i,1)-1)+1)=loads(i,2);
```

```
    Ftotal(3*(loads(i,1)-1)+2)=loads(i,3);
      Ftotal(3*(loads(i,1)-1)+3)=loads(i,4);
end
%% imposing the boundary conditions
% simply by setting 1*u=0, scaling 1 to be at the same
level of diagonal
% elements at Ktotal
level=max(diag(Ktotal));
oneone=100*level;
for i=1:nubcs
    % for node bcs(i,1) check x-y supports
    % if bcs(i,2)=1 then x displacement is fixed equal to
zero
    % if bcs(i,3)=1 then y displacement is fixed equal to
zero
    if bcs(i,2)==1
        Ktotal(3*(bcs(i,1)-1)+1,:)=0;
        Ktotal(:,3*(bcs(i,1)-1)+1)=0;
        Ktotal(3*(bcs(i,1)-1)+1,3*(bcs(i,1)-1)+1)=oneone;
    end
    if bcs(i,3)==1
        Ktotal(3*(bcs(i,1)-1)+2,:)=0;
        Ktotal(:,3*(bcs(i,1)-1)+2)=0;
        Ktotal(3*(bcs(i,1)-1)+2,3*(bcs(i,1)-1)+2)=oneone;
    end
        if bcs(i,4)==1
        Ktotal(3*(bcs(i,1)-1)+3,:)=0;
        Ktotal(:,3*(bcs(i,1)-1)+3)=0;
        Ktotal(3*(bcs(i,1)-1)+3,3*(bcs(i,1)-1)+3)=oneone;
        end
end
%
```

Εδώ γίνεται η κατασκευή του μητρώου δυσκαμψίας,με μη μηδενικά στοιχεία στην διαγώνιο, με σκοπό την επίλυση του συστήματος εξισώσεων Utotal=Ktotal/Ftotal.

```
-----------------------------------------------------------
%% solving the system of equations
Utotal=Ktotal\Ftotal;
```

Εδώ επιλύεται το σύστημα των εξισώσεων μας.

```
-----------------------------------------------------------
%% printing the solution
scale1=max(Utotal);
scale2=max(lx,ly);
```

```
scale3=max(scale2,lz);
scale= 0.8*scale3/scale1;
newplot
hold on
for i=1:nelements
x = [xcoo(cnct(i,1)) xcoo(cnct(i,2)) ]';
y=[ycoo(cnct(i,1)) ycoo(cnct(i,2)) ]';
z = [zcoo(cnct(i,1)) zcoo(cnct(i,2)) ]';
    plot3(x,y,z,'b-')
    % displacements of nodes
%     xdispl = [Utotal(3*(cnct(i,1)-1)+1)
Utotal(3*(cnct(i,2)-1)+1) ]';
%     ydispl = [Utotal(3*(cnct(i,1)-1)+2)
Utotal(3*(cnct(i,2)-1)+2) ]';
%        zdispl = [Utotal(3*(cnct(i,1)-1)+3)
Utotal(3*(cnct(i,2)-1)+3) ]';
%
plot3(x+scale*xdispl,y+scale*ydispl,z+scale*zdispl,'r')
end
title('Initial and deformed truss')
hold off
```

Εδώ τυπώνεται η αρχική λύση.

```
-----------------------------------------------------------
%timi = norm(Utotal);
%timi = Utotal' * Ftotal
%% calculation of optimization functions
%func1 = 0;
func2 = 0;
rho = 0.1;
NormSigma=0;
tem=0;
for i=1:nelements
    % for element i
    % first node number cnct(i,1)
    iarxis = cnct(i,1);
    itelous = cnct(i,2);
% first node coordinates xnode(cnct(i,1)), ynode(cnct(i,1))
   P1 = [ xcoo(cnct(i,1)), ycoo(cnct(i,1)), zcoo(cnct(i,1))
];
   P2 = [ xcoo(cnct(i,2)), ycoo(cnct(i,2)), zcoo(cnct(i,2))
];
  l = norm(P2-P1);
alpha = atan2(P2(2)-P1(2),P2(1)-P1(1));
lamb1 = [-cos(alpha) -sin(alpha) cos(alpha) sin(alpha)];
```

```
%
displnodes = [ Utotal(2*(iarxis-1)+1) Utotal(2*(iarxis-
1)+2) Utotal(2*(itelous-1)+1) Utotal(2*(itelous-1)+2) ]';
sigma(i) = (E(i)/l)*lamb1*displnodes;
force(i) = sigma(i)*A(i);
%func1 = func1+rho*A(i)*l;

for j=1:3*nnodes
    func2=func2+Utotal(j)*Ftotal(j);
    %for maximization
    %func2 = -func2;
end
tem=tem+1;
func2;
end
func2;
%% printing the solution
%
newplot
 hold on
for i=1:nelements
 x = [xcoo(cnct(i,1)) xcoo(cnct(i,2)) ];
 y = [ycoo(cnct(i,1)) ycoo(cnct(i,2)) ];
 z = [zcoo(cnct(i,1)) zcoo(cnct(i,2)) ]';
%     plot3(x,y,z,'b-')

 Alpha_max = 1.5;
 Alpha_min = 0.1;
 L= Alpha_max-Alpha_min;

 Thickness_matrix =[0.5 2 3 4 6 9];

 if A(i)<=Alpha_min+L/32
    Thickness = Thickness_matrix(1,1);
 elseif A(i)<=Alpha_min+2*L/30
    Thickness = Thickness_matrix(1,2);
 elseif A(i)<=Alpha_min+2*L/6
    Thickness = Thickness_matrix(1,3);
 elseif A(i)<=Alpha_min+3*L/5
    Thickness = Thickness_matrix(1,4);
 elseif A(i)<=Alpha_min+4*L/5
    Thickness = Thickness_matrix(1,5);
    else
    Thickness = Thickness_matrix(1,6);
 end
 plot3(x,y,z,'LineWidth',Thickness)

end
```

```
 title('Thickness Scaling')
 hold off
```

εδώ δίνονται οι συναρτήσεις ως προς βελτιστοποίηση και υπολογίζονται οι τάσεις(sigma).

Από το αρχείο **truss3d.m:**

```
function K = truss3d(A,E,P1,P2)
%   TRUSS3D
%
%   This command generates the 6 x 6 stiffness matrix for a three
%   dimensional truss element in global coordinates. The syntax is:
%               K = truss3d(A,E,P1,P2)
%   where: A is the cross-sectional area;
%          E is the Young's modulus;
%          P1 and P2 are vectors of the {x,y,z} coordinates of the
%              endpoints.

L = norm(P2-P1);
lpq = (P2(1)-P1(1))/L;
mpq = (P2(2)-P1(2))/L;
npq = (P2(3)-P1(3))/L;
lamb = [lpq mpq npq -lpq -mpq -npq]';
K = A*E/L*(lamb)*(lamb');
```

Με αυτό το αρχείο δημιουργείται το τρισδιάστατο πλέγμα, σύμφωνα με την νόρμα το P2-P1, όπου αυτά είναι τα διανύσματα των (x,y,z), δηλαδή τέλος-αρχή.

Από το αρχείο **plotarisma.m:**

```
-------------------------------------------------------
nelements=675;

%  x=load('xmat.mat');
 x=x/1000
% for i=1:nelements
%
%     A(i) = x(i);
% end

newplot
 hold on
for i=1:nelements
    i
 f = [xcoo(cnct(i,1)) xcoo(cnct(i,2)) ];
 y = [ycoo(cnct(i,1)) ycoo(cnct(i,2)) ];
```

```
 z = [zcoo(cnct(i,1)) zcoo(cnct(i,2)) ]';
%     plot3(x,y,z,'b-')

 Alpha_max = 0.5;
 Alpha_min = 0.1;
 L= Alpha_max-Alpha_min;

 Thickness_matrix =[0.5 2 3 4 6 9];

if x(i)<=Alpha_min+L/32*10^2
    Thickness = Thickness_matrix(1,1);
 elseif x(i)<=Alpha_min+2*L/30*10^2
    Thickness = Thickness_matrix(1,2);
 elseif x(i)<=Alpha_min+2*L/6*10^2
    Thickness = Thickness_matrix(1,3);
 elseif x(i)<=Alpha_min+3*L/5*10^2
    Thickness = Thickness_matrix(1,4);
 elseif x(i)<=Alpha_min+4*L/5*10^2
    Thickness = Thickness_matrix(1,5);
    else
    Thickness = Thickness_matrix(1,6);
 end
 plot3(f,y,z,'LineWidth',Thickness)

end
 title('Thickness Scaling')
 hold off
```

Εδώ ουσιαστικά έχουμε ξεχωρίσει τον κώδικα της εκτύπωσης της λύσης για δική μας διευκόλυνση. Τα πάχη των γραμμών δίνονται κατά περίπτωση.