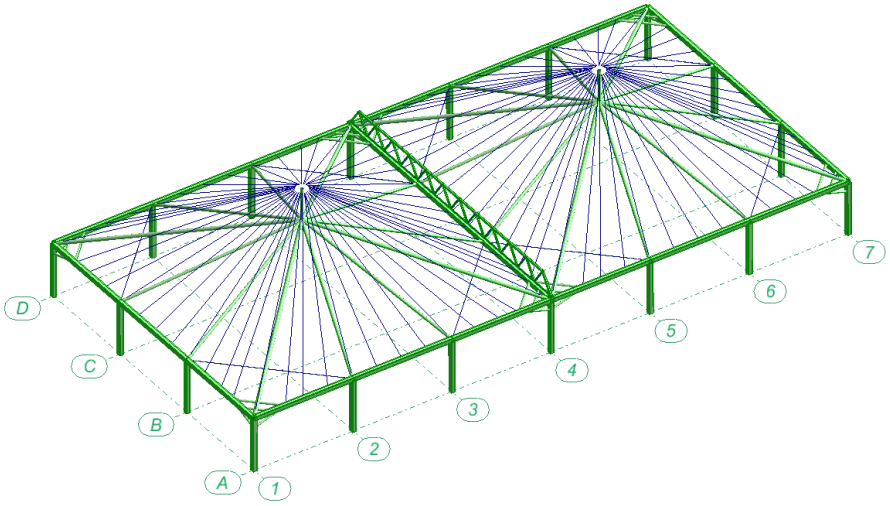
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΙΑΣ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΓΓΡΑΦΗ ΚΩΔΙΚΑ ΣΤΑΤΙΚΗΣ ΑΝΑΛΥΣΗΣ ΔΙΚΤΥΩΜΑΤΟΣ ΜΕ ΤΗΝ ΜΕΘΟΔΟ ΤΩΝ ΠΕΠΕΡΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ

ΣΠΟΥΔΑΣΤΗΣ

ΦΕΤΟΚΑΚΗΣ ΕΜΜΑΝΟΥΗΛ

Α.Μ 4743

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ

ΜΑΝΙΑΤΗ ΜΑΡΙ-ΜΙΣΕΛ

ΗΡΑΚΛΕΙΟ 2013-2014

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την συγγραφή κώδικα για την επίλυση επίπεδων δικτυωμάτων με την μέθοδο των πεπερασμένων στοιχείων. Η συγγραφή κώδικα γίνεται στην γλώσσα προγραμματισμού matlab.

Αρχικά θα γίνει μια περιγραφή για τα πεπερασμένα στοιχεία και διάφορες μεθόδους που χρησιμοποιούνται στην επίλυση προβλημάτων. Επίσης θα γίνει μία αναφορά στην μέθοδο που χρησιμοποιήθηκε για την επίλυση των δικτυωμάτων στην παρούσα πτυχιακή εργασία.

Έπειτα ακλουθεί μία γενική πληροφόρηση πάνω στην γλώσσα προγραμματισμού της Matlab, καθώς και επεξήγηση των εντολών που χρησιμοποιήθηκαν για την υλοποίηση του προγράμματος.

Στην συνέχεια γίνεται ανάλυση του κώδικα, σε συνδυασμό με παραδείγματα επίλυσης επίπεδων δικτυωμάτων.

Τέλος έχουμε κάποια συμπεράσματα της πτυχιακής εργασίας και κάποιες ιδέες για την βελτίωση ή περεταίρω ανάπτυξη του κώδικα.

Λέξεις κλειδιά

Πεπερασμένα στοιχεία, γλώσσας προγραμματισμού matlab, μέθοδος δυσκαμψίας

Abstract

The scope of this thesis is the writing of a code for the solution of plane nettings with finite elements method. The code has been developed in matlab program.

First of all, there will be a description of finite elements and the variety of methods that can be used for the solution of a problem. In addition there will be a report at the method that has been used for the solution of plane nettings in the current thesis.

Then, we have some general information about matlab application and the explanation of commands that have been used for the developing of the code.

After, there is analysis of the code and some examples of plane nettings settlement.

In the end, we have some conclusions of this thesis and some ideas for improvement or development of this code.

Keywords

Finite elements, matlab programming, stiffness method

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου εργασίας θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν στην περάτωση αυτής της εργασίας.

Κατά κύριο λόγο θα ήθελα να ευχαριστήσω την καθηγήτρια μου κ. Μανιάτη Μαρί-Μισέλ για την εμπιστοσύνη που μου έδειξε, και την υπομονή που έκανε κατά την διάρκεια υλοποίησης της πτυχιακής εργασίας. Όπως επίσης για την πολύτιμη βοήθεια και καθοδήγηση της, για την επίλυση διάφορων θεμάτων που προέκυψαν.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου για την υπομονή που έδειξαν καθ' όλη την διάρκεια των σπουδών μου, και για την οικονομική υποστήριξη που μου παρείχαν φροντίζοντας για την καλύτερη δυνατή μόρφωση μου.

Τέλος θα ήθελα να ευχαριστήσω τους Δαμιανάκη Αστρινό, Μακαρώνα Κωνσταντίνο και Μπιράκη Νικόλαο για την ψυχολογική υποστήριξη που μου παρείχαν όποτε αυτή χρειάστηκε, καθότι έπαιξε σημαντικό ρόλο στην ολοκλήρωση της πτυχιακής μου εργασίας.

Περιεχόμενα

Κεφάλαιο 1

1 Εισαγωγή

1.1 Σκοπός της πτυχιακής εργασίας

Η πτυχιακή έχει ως σκοπό την δημιουργία ενός προγράμματος στο υπολογιστικό περιβάλλον Matlab, το οποίο θα μπορεί να επιλύει διδιάστατα δικτυώματα με την μέθοδο των πεπερασμένων στοιχείων .

Ο χρήστης θα μπορεί να ορίσει στο πρόγραμμα το δικτύωμα δίνοντας :

- Περιγραφή των κόμβων
- Περιγραφή των ράβδων
- Διατομές ράβδων
- Περιγραφή φορτίσεων
- Περιγραφή στηρίξεων

Το πρόγραμμα θα πρέπει να έχει την δυνατότητα να :

- Υπολογίζει τις μετατοπίσεις των κόμβων
- Υπολογίζει τις τάσεις στις ράβδους
- Υπολογίζει τις αντιδράσεις στηρίξεων
- Εμφανίζει σχεδιάγραμμα πριν και μετά τις μετατοπίσεις των κόμβων

Κεφάλαιο 2

2 Πεπερασμένα στοιχεία

2.1 Η μέθοδος των πεπερασμένων στοιχείων

Η μέθοδος πεπερασμένων στοιχείων είναι μια προσεγγιστική μέθοδος επίλυσης διαφορικών εξισώσεων που χρησιμοποιούνται κανόνες μητρωικού λογισμού.

Η αναλυτική λύση των εξισώσεων με τις οποίες περιγράφονται τα διάφορα τεχνικά προβλήματα είναι δυνατή μόνο σε ειδικές περιπτώσεις, όπου οι καταπονήσεις και τα γεωμετρικά σχήματα είναι πάρα πολύ απλά. Όμως, υπήρχε η ανάγκη να λυθούν και πιο σύνθετα προβλήματα και γι' αυτό το λόγο αναπτύχθηκαν διάφορες προσεγγιστικές μέθοδοι.

Μία τέτοια μέθοδος είναι και η μέθοδος των πεπερασμένων στοιχείων. Αυτή η μέθοδος είναι μεν προσεγγιστική, αλλά μπορεί να δώσει αξιόπιστα αποτελέσματα και έχει το πλεονέκτημα ότι μπορεί να εφαρμοστεί σε όλα τα προβλήματα. Το μειονέκτημά της είναι οι αυξημένες απαιτήσεις σε υπολογιστική ισχύ, ιδίως όταν εφαρμόζεται σε σύνθετα μοντέλα. Αυτό όμως το μειονέκτημα ξεπεράστηκε τα τελευταία χρόνια χάρη στη ραγδαία ανάπτυξη των υπολογιστών. Η επιτυχία αυτής της μεθόδου ήταν τόσο μεγάλη, που ακόμα και σήμερα χρησιμοποιείται στην έρευνα και στην βιομηχανία για τον υπολογισμό και τη μελέτη διάφορων κατασκευών.

Για να εφαρμοστεί η μέθοδος των πεπερασμένων στοιχείων απαιτούνται τα εξής στάδια:

i. Εισάγεται η γεωμετρία της κατασκευής σε ένα πρόγραμμα στο οποίο δημιουργείται το μοντέλο.
ii. Γίνεται η διακριτοποίηση του μοντέλου σε πεπερασμένα στοιχεία και αφού ετοιμαστεί το πλέγμα επιλέγεται το είδος της επίλυσης και εισάγονται τα επιπλέον δεδομένα που απαιτούνται στο πρόγραμμα. Για παράδειγμα, αν επιλεγεί να λυθεί το μοντέλο σε στατική καταπόνηση θα πρέπει να δοθούν τα δεδομένα για τις δυνάμεις και τις στηρίξεις.
iii. Όταν ετοιμαστούν τα δεδομένα για επίλυση, το πρόγραμμα θα κάνει την επίλυση του προβλήματος.
iv. Όταν τελειώσει η επίλυση το πρόγραμμα θα προβάλει τα αποτελέσματα στον μελετητή.

2.2 Προσαρμοστικά πεπερασμένα στοιχεία

Με την μέθοδο των πεπερασμένων στοιχείων επιτυγχάνεται μια προσεγγιστική λύση του μαθηματικού προσομοιώματος του προβλήματος. Η ακρίβεια της προσέγγισης εξαρτάται από τον τύπο των πεπερασμένων στοιχείων που χρησιμοποιούνται και την πυκνότητα του δικτύου. Με την πύκνωση του δικτύου και την αύξηση του βαθμού της πολυωνυμικής συνάρτησης που εκφράζει το πεδίο των μετατοπίσεων είναι δυνατόν να επιτευχθεί σύγκλιση στην ακριβή λύση. Η αποτελεσματική εφαρμογή της μεθόδου των πεπερασμένων στοιχείων εστιάζεται στην αναζήτηση της βέλτιστης διακριτοποίησης του φορέα. Δηλαδή στον προσδιορισμό ενός δικτύου με τα κατάλληλα στοιχεία και τους ελάχιστους βαθμούς ελευθερίας ώστε να μειωθεί το σφάλμα διακριτοποίησης στο επιθυμητό επίπεδο. Η βέλτιστη διακριτοποίηση δεν αναφέρεται μόνο στο πλήθος και στον τύπο των στοιχείων αλλά και στην κατανομή της πυκνότητας του δικτύου στην επιφάνεια ή τον όγκο του φορέα. Περιοχές με υψηλή συγκέντρωση τάσεων απαιτούν πυκνότερη διακριτοποίηση ή στοιχεία ανωτέρας τάξεως από περιοχές με ομαλή κατανομή των τάσεων. Οι περιοχές αυτές ενδέχεται να μην είναι γνωστές εκ των προτέρων. Γι' αυτό τον λόγο ακολουθείται μια σταδιακή βελτίωση του δικτύου μέσα από την διαδικασία προσαρμογής και διαδοχικών επιλύσεων.

Η διαδικασία προσαρμογής του δικτύου των πεπερασμένων στοιχείων περιλαμβάνει σε κάθε βήμα προσαρμογής δυο στάδια:

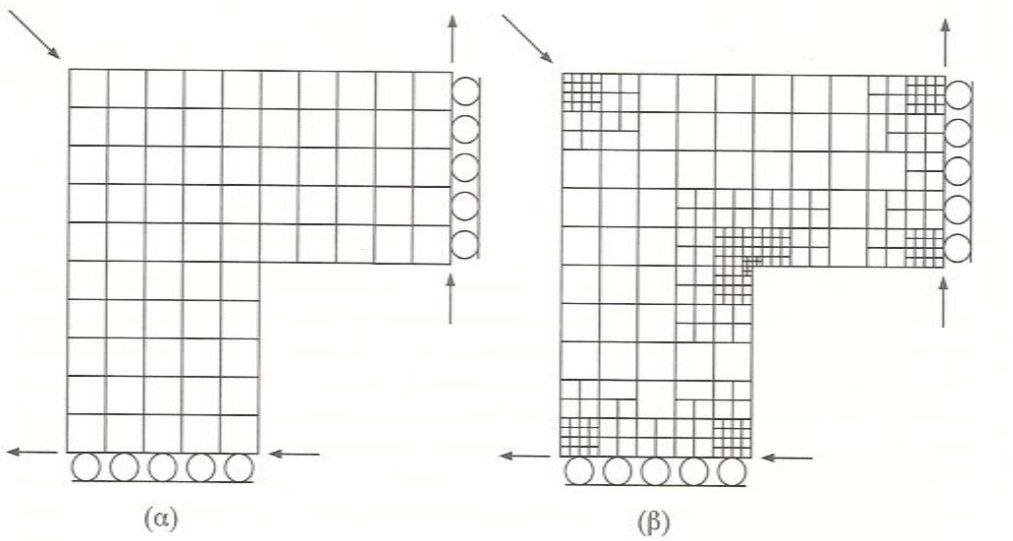
i. Την εκτίμηση του σφάλματος της προσεγγιστικής λύσης για το δεδομένο δίκτυο.
ii. Τη βελτίωση του δικτύου σύμφωνα με την κατανομή στο σφάλματος διακριτοποίησης χρησιμοποιώντας τους ελάχιστους επιπρόσθετους βαθμούς ελευθερίας.

Τα στάδια αυτά επαναλαμβάνονται μέχρι να επιτευχθεί η μείωση του σφάλματος διακριτοποίησης στο επιθυμητό επίπεδο. Η βελτίωση του δικτύου επιτυγχάνεται κυρίως με τρεις τρόπους:

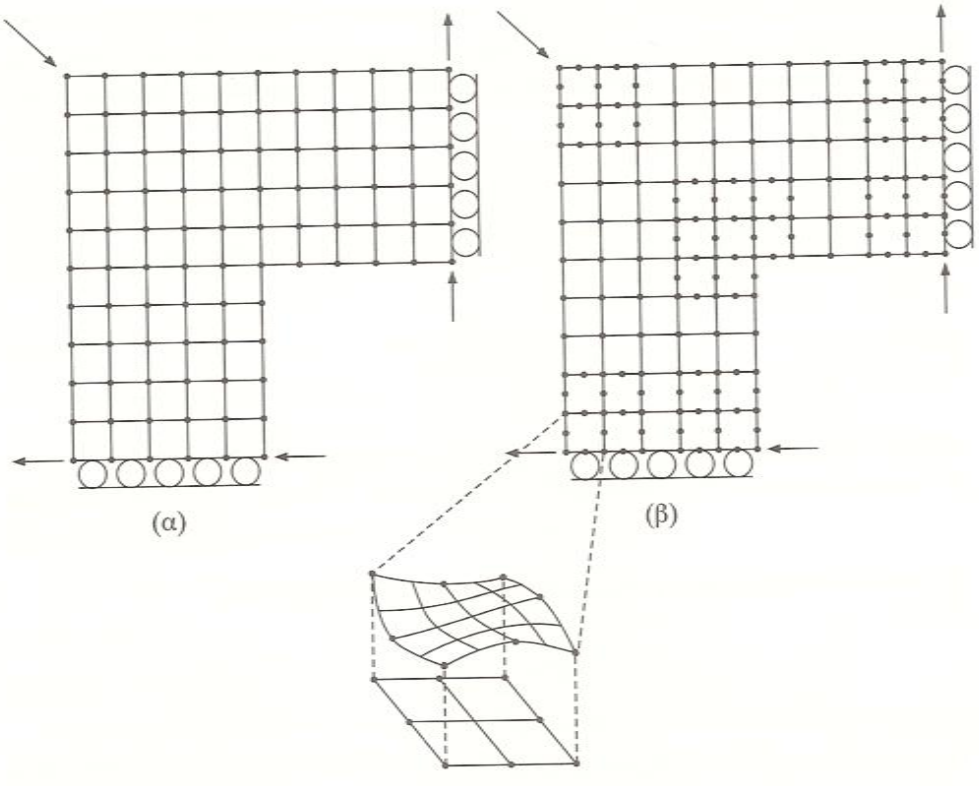
i. Αύξηση της πυκνότητας του δικτύου με χρήση στοιχείων μικρότερου μεγέθους – **προσαρμογή τύπου h.**
ii. Διατήρηση του αρχικού δικτύου και αύξηση του βαθμού της πολυωνυμικής συνάρτησης του πεδίου των μετατοπίσεων των στοιχείων – **προσαρμογή τύπου p.**
iii. Συνδυασμός των μεθόδων h και p, όπου μετά τη βελτίωση του αρχικού δικτύου με τη μέθοδο h ακολουθεί η προσαρμογή τύπου p της οποίας η αποτελεσματικότητα βελτιώνεται σημαντικά.

Στα παρακάτω σχήματα φαίνονται δύο παραδείγματα εφαρμογής πεπερασμένων στοιχείων τύπου h και p, αντίστοιχα, σε προβλήματα επίπεδης έντασης.

Βελτίωση δικτύου πεπερασμένων στοιχείων με προσαρμογή τύπου h: (α) αρχικό δίκτυο, (β) βελτιωμένο δίκτυο



Βελτίωση δικτύου πεπερασμένων στοιχείων με προσαρμογή τύπου p: (α) αρχικό δίκτυο, (β) βελτιωμένο δίκτυο



2.2.1 Εκτίμηση του σφάλματος διακριτοποίησης

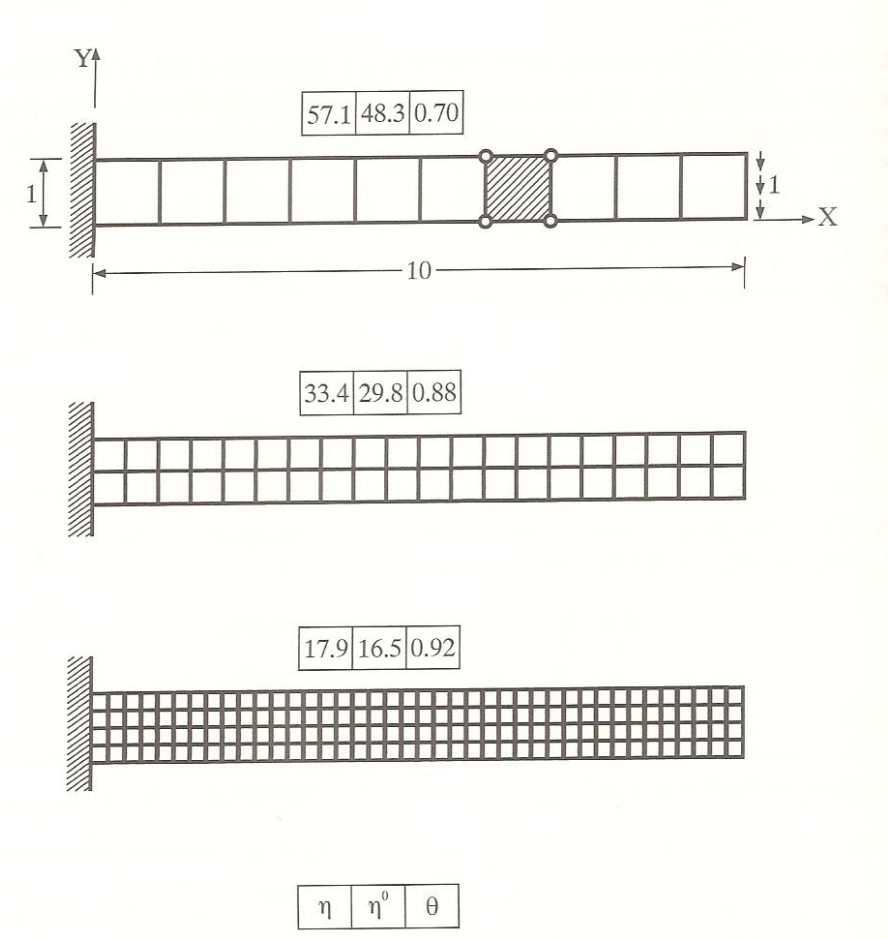
Η εκτίμηση του σφάλματος σε μια περιορισμένη περιοχή του φορέα μπορεί να οδηγήσει σε παραπλανητικά αποτελέσματα ως προς την ποιότητα του δικτύου των πεπερασμένων στοιχείων. Για παράδειγμα στην εγγύτερη περιοχή ενός σημειακού φορτίου τα σφάλματα των μετατοπίσεων και των τάσεων είναι σημαντικά, στο δε σημείο εφαρμογής απειρίζονται. Αυτό δεν εμποδίζει όμως την συνολική λύση των πεπερασμένων στοιχείων να είναι αποδεκτή.

Η αξιοπιστία της μεθόδου εκτίμησης του σφάλματος προσδιορίζεται από τον δείκτη αποτελεσματικότητας θ, ο οποίος ορίζεται ως εξής :

$\theta = |e| / |e^*|$ όπου:

- θ: δείκτης αποτελεσματικότητας
- e: εκτιμώμενο σφάλμα
- e^*: πραγματικό σφάλμα

Εδώ φαίνεται μια προσομοίωση προβόλου με ορθογωνικά στοιχεία τεσσάρων κόμβων. Εκτίμηση σφάλματος για ομοιόμορφη πύκνωση δικτύου.



Όπου:

- η: Ακριβής % τιμή του σφάλματος
- η^0: Εκτιμώμενη % τιμή του σφάλματος
- θ: Δείκτης αποτελεσματικότητας

Η βελτιστοποίηση του δείκτη αποτελεσματικότητας σταματάει σύμφωνα με την κρίση των μηχανικών στο πρόβλημα που έχουν.

2.2.2 Μέθοδοι προσαρμογής τύπου h

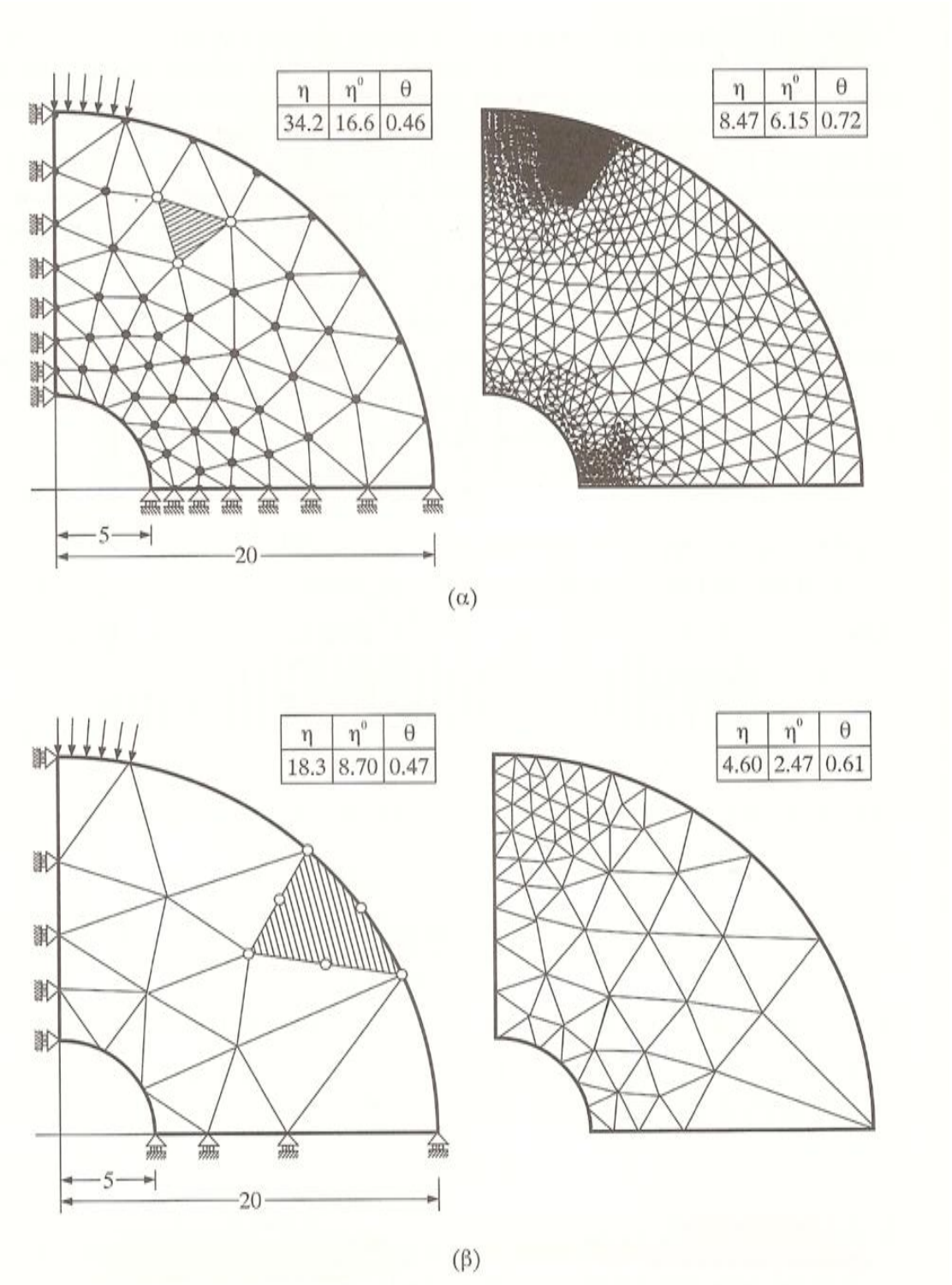
Οι τεχνικές προσαρμογής ενός δικτύου πεπερασμένων στοιχείων με μείωση του μεγέθους των στοιχείων διακρίνονται σε τέσσερις κατηγορίες:

i. Επανασχεδιασμός όλου του δικτύου: Μετά την εκτίμηση του σφάλματος το παλιό δίκτυο καταργείται και δημιουργείται ένα καινούργιο δίκτυο σε όλο τον φορέα με την απαιτούμενη πυκνότητα.
ii. Πύκνωση του αρχικού δικτύου: Η δημιουργία του νέου δικτύου βασίζεται στο αρχικό δίκτυο όπου ορισμένες ομάδες στοιχείων επανασχεδιάζονται με υποδιαίρεση των στοιχείων τους.
iii. Μετάθεση των κόμβων δικτύου: Η αρχική τοπολογία του δικτύου διατηρείται σε όλα τα βήματα προσαρμογής, αλλάζει όμως η θέση των κόμβων. Αυτή η μέθοδος προσαρμογής ονομάζεται και **προσαρμογή τύπου r**.
iv. Ομοιόμορφη πύκνωση δικτύου: Η πύκνωση του δικτύου γίνεται ομοιόμορφα προς ρο αρχικό δίκτυο. Η αποτελεσματικότητα της μεθόδου αυτής είναι μειωμένη αφού κληρονομεί τις αδυναμίες του αρχικού δικτύου και αυξάνει σημαντικά τους βαθμούς ελευθερίας του προβλήματος.

Η αποτελεσματικότητα των τεχνικών προσαρμογής, ως προς την ταχύτητα συγκλίσεως και το πλήθος των βαθμών ελευθερίας, αποτελεί συνάρτηση πολλών παραμέτρων του προβλήματος και για τον λόγο αυτό δεν υπάρχουν συγκεκριμένα κριτήρια επιλογής.

Η βασική αρχή που ακολουθείται στις μεθόδους προσαρμογής είναι η επίτευξη μίας κατά το δυνατόν ομοιόμορφης κατανομής του καθολικού σφάλματος εντός του δικτύου και η μείωση στη συνέχεια του καθολικού σφάλματος σε αποδεκτά επίπεδα.

Εδώ φαίνεται η βελτίωση δικτύου πεπερασμένων στοιχείων με προσαρμογή τύπου h: (α) τριγωνικά στοιχεία τριών κόμβων, (β) τριγωνικά στοιχεία έξι κόμβων.

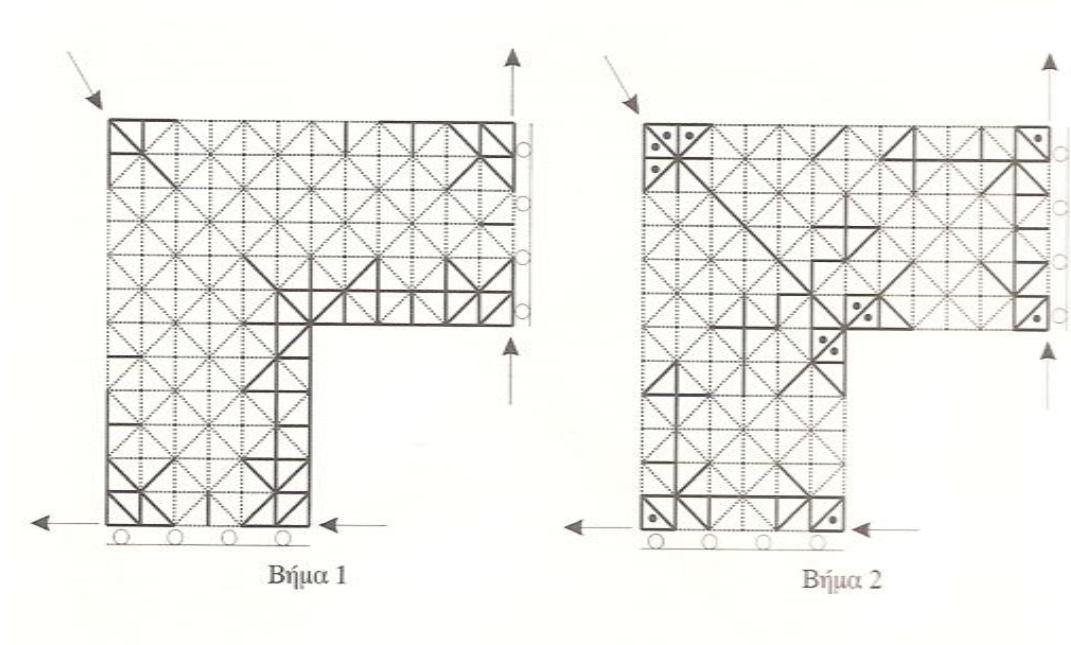


2.2.3 Μέθοδος προσαρμογής τύπου p

Στη μέθοδο προσαρμογής τύπου p διατηρείται το αρχικό δίκτυο των πεπερασμένων στοιχείων και αυξάνεται ο βαθμός της πολυωνυμικής συνάρτησης που εκφράζει το πεδίο των μετατοπίσεων. Η μέθοδος αυτή συνδυάζεται συνήθως με μια ιεραρχική δομή των συναρτήσεων σχήματος έτσι ώστε με την αύξηση του βαθμού του

πολυωνύμου ωα μην χρειάζεται να επαναϋπολογιστεί από την αρχή όλο το μητρώο στιβαρότητας των στοιχείων.

Εδώ φαίνεται η βελτίωση δικτύου με προσαρμοστικά ιεραρχικά πεπερασμένα στοιχεία τύπου p σε δύο βήματα προσαρμογής.



2.3 Η μέθοδος των μετατοπίσεων

Η μέθοδος των μετατοπίσεων είναι η μέθοδος που χρησιμοποιήθηκε στην παρούσα πτυχιακή εργασία για την επίλυση επίπεδων δικτυωμάτων.

Γενικά, τα περισσότερα προγράμματα Η/Υ έχουν ως θεμελιώδη βάση τους την μέθοδο των μετατοπίσεων. Σε αυτή την μέθοδο, όπως και στις άλλες μεθόδους μετακινήσεων, οι εξισώσεις ισορροπίας γράφονται ως προς τις άγνωστες μετατοπίσεις των κόμβων.

Αφού προσδιορίσουμε τους κινηματικά ελεύθερους κόμβους και ορίσουμε τους βαθμούς ελευθερίας κίνησης τους, το πρόγραμμα επιβάλει αντίστοιχες μοναδιαίες μετακινήσεις και υπολογίζει τους συντελεστές ακαμψίας, καταστρώνει και επιλύει τις εξισώσεις ισορροπίας, και υπολογίζει τις αντιδράσεις, μετακινήσεις των κόμβων και δυνάμεις μελών του φορέα.

Η μέθοδος των μετατοπίσεων βασίζεται στα μητρώα δυσκαμψίας των επιμέρους μελών της κατασκευής βάσει τον οποίων σχηματίζεται το συνολικό μητρώο δυσκαμψίας Κ της κατασκευής. Οι άγνωστοι στις σχηματιζόμενες εξισώσεις είναι οι μετακινήσεις των ελεύθερων κόμβων της κατασκευής.

Επιλύνοντας το σύστημα εξισώσεων που σχηματίζεται, υπολογίζονται οι μετακινήσεις των βαθμών ελευθερίας των κόμβων της κατασκευής. Έπειτα, χρησιμοποιώντας τα επιμέρους μητρώα δυσκαμψίας του κάθε μέλους, υπολογίζονται τα εντατικά μεγέθη στα άκρα του κάθε μέλους.

Η μέθοδος συνοψίζεται στα εξής στάδια:

- Καθορισμός σχέσεων εντατικών μεγεθών και των αντίστοιχων μετακινήσεων των μελών ενός φορέα, βάσει των μητρώων δυσκαμψίας των επιμέρους μελών κατασκευής
- Κατάλληλοι μετασχηματισμοί από τοπικό σε απόλυτο σύστημα συντεταγμένων
- Εφαρμογή εξισώσεων ισορροπίας στους κόμβους
- Σχηματισμός μητρώου δυσκαμψίας της κατασκευής
- Εφαρμογή συνοριακών συνθηκών
- Επίλυση σχηματιζόμενου συστήματος εξισώσεων
- Υπολογισμός μετατοπίσεων ελεύθερων κόμβων κατασκευής

F= K*Q όπου:

- F: Τα ασκούμενα φορτία
- K: Το ολικό μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις κατά τους βαθμούς ελευθερίας

- Υπολογισμός αντιδράσεων στις στηρίξεις

$R=K_R*Q$ όπου:

- R: Οι αντιδράσεις στους περιορισμούς του δικτυώματος
- K_R: Το προσαρμοσμένο μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις των κόμβων του δικτυώματος

- Υπολογισμός εντατικών μεγεθών στα άκρα του κάθε μέλους

σ=k*q

- σ: Η ζητούμενη τάση στην ράβδο
- k: Το μητρώο δυσκαμψίας της ράβδου
- q: Οι κομβικές μετατοπίσεις της ράβδου

Κεφάλαιο 3

3 Προγραμματίζοντας στην Matlab

3.1 Η γλώσσα προγραμματισμού της Matlab

Το Matlab είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού που επιτρέπει την ολοκλήρωση υπολογιστικά απαιτητικών εργασιών ταχύτερα και ευκολότερα απ' ότι οι παραδοσιακές γλώσσες προγραμματισμού, όπως C, C++ και Fortran. Το όνομα του προέρχεται από τα αρχικά των λέξεων MATrix LABoratory που σημαίνει 'εργαστήριο πινάκων' επειδή η βασική του δομή είναι ο πίνακας. Όλες οι μεταβλητές αποθηκεύονται ως πίνακες. Ακόμα και ένας αριθμός θεωρείται ότι είναι ένας πίνακας 1x1.

3.2 Ιστορία του Matlab

Το Matlab αποτελεί την φυσική εξέλιξη προγραμμάτων σε Fortran για την επίλυση γραμμικών προβλημάτων (πρόγραμμα Linpack) και προβλημάτων εύρεσης ιδιοτιμών (Eispack).

Αναπτύχθηκε αρχικά από τον Cleve Moler τη δεκαετία του 1970 για την επίλυση των παραπάνω προβλημάτων χωρίς την γνώση Fortran από τον χρήστη.

Επίσης, διαπιστώθηκε η ανάγκη ύπαρξης γραφικού περιβάλλοντος και σχεδιασμού διαγραμμάτων . Ως αποτέλεσμα, την δεκαετία του 1980 το πρόγραμμα γράφτηκε ξανά σε γλώσσα C με περισσότερες λειτουργίες, βιβλιοθήκες, δυνατότητα γραφικών παραστάσεων και απεικονίσεων.

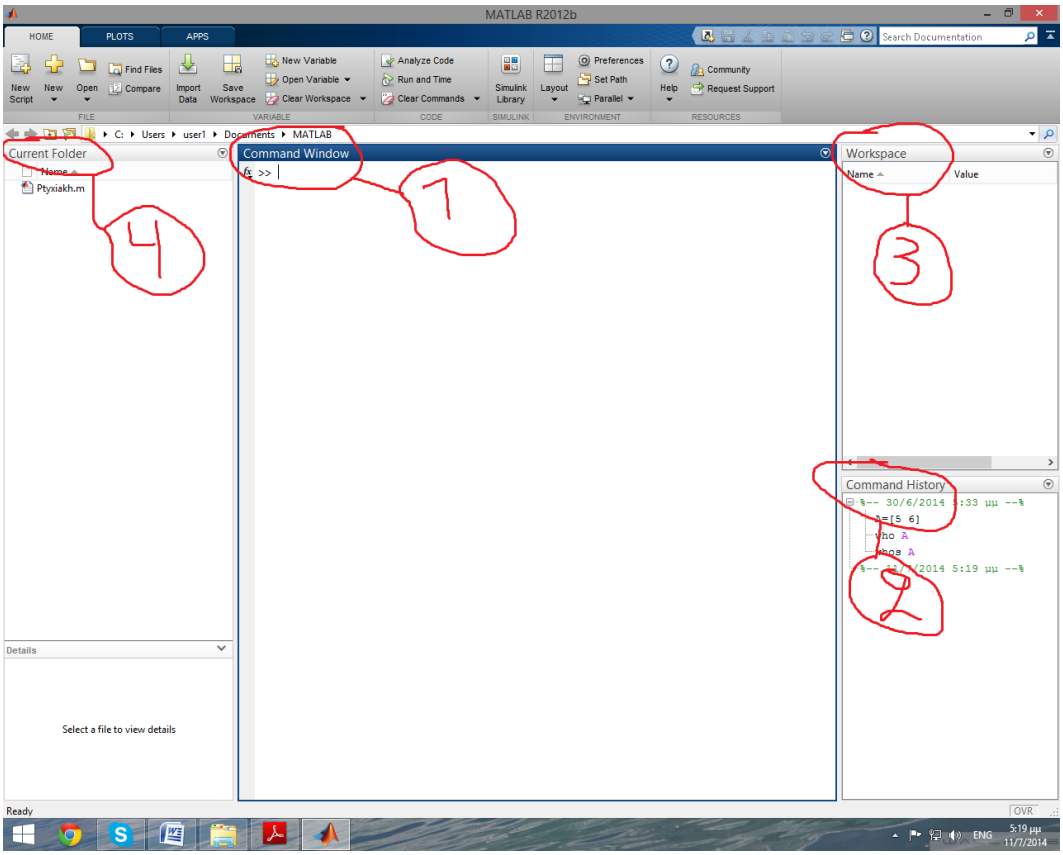
Το 1984 ιδρύεται η εταιρία MathWorks η οποία είναι υπεύθυνη για τη διάθεση στο εμπόριο, την εξέλιξη και την υποστήριξη του λογισμικού μέχρι σήμερα.
http://www.mathworks.com

3.3 Το περιβάλλον του Matlab

Το περιβάλλον του προορίζεται για ανάπτυξη αλγορίθμων, επεξεργασία, ανάλυση και απεικόνιση δεδομένων, αριθμητικές μεθόδους επίλυσης προβλημάτων και μοντελοποίηση συστημάτων. Τα βασικότερα μέρη του περιβάλλοντος του Matlab είναι τα εξής :

- Command Window (παράθυρο εντολών): Στο παράθυρο αυτό, εισάγονται οι εντολές της Matlab, καλούνται συναρτήσεις και M-files, καθώς τυπώνονται και τα αποτελέσματα.
- Command History (ιστορικό εντολών): Στο Command History αποθηκεύονται οι εντολές που εισάγουμε στο Command Window.

- Workspace (χώρος εργασίας): Στο Workspace αποθηκεύονται οι μεταβλητές και οι πίνακες που έχουμε δημιουργήσει. Ο χώρος αυτός βρίσκεται στην μνήμη του υπολογιστή, και μέσα σ' αυτόν εκτελούνται οι πράξεις και ταυτόχρονα διατηρούνται όλα τα αποτελέσματα τους, με σκοπό να μπορούν να χρησιμοποιηθούν σε επόμενες πράξεις.
- Current Folder (τρέχων κατάλογος): Εδώ είναι το σημείο αναφοράς των αρχείων του Matlab, αν ο χρήστης θέλει να καλέσει μια συνάρτηση ή ένα M-file θα πρέπει να βρίσκεται στον τρέχοντα κατάλογο.



1. Command Window
2. Command History
3. Workspace
4. Current Folder

3.4 Πλεονεκτήματα γλώσσας προγραμματισμού Matlab

- Η εκμάθηση της γλώσσας προγραμματισμού Matlab είναι σχετικά ευκολότερη από μια άλλη γλώσσα προγραμμάτισμου.
- Ο κώδικας του Matlab είναι σχεδιασμένος να δίνει γρήγορα αποτελέσματα όταν πραγματοποιούνται πράξεις πινάκων.
- Γλώσσα προγραμματισμού για ανάπτυξη εφαρμογών και ταυτόχρονα λογισμικό υλοποίησης επιστημονικών υπολογισμών.
- Εύκολος εντοπισμός και διόρθωση λαθών.
- Φιλικό περιβάλλον επικοινωνίας με τον χρήστη.

3.5 Μειονεκτήματα γλώσσας προγραμματισμού Matlab

- Η Matlab δεν αποτελεί μια γλώσσα προγραμματισμού γενικής χρήσης.
- Το Matlab είναι σχεδιασμένο για επιστημονικούς υπολογισμούς και δεν είναι κατάλληλο για κάποιες άλλες λειτουργίες (π.χ. , ανάγνωση κειμένου).
- Οι αναπτυσσόμενες εφαρμογές υστερούν σε απόδοση από την άποψη του χρόνου εκτέλεσης σε σχέση με αντίστοιχες που αναπτύσσονται στις κλασσικές γλώσσες προγραμματισμού (C,C++).
- Το Matlab είναι μια μεταφραζόμενη προγραμματιστική γλώσσα που απαιτεί τον μεταφραστή (το Matlab δηλαδή) για να εκτελέσει κάποιο πρόγραμμα. Δεν παράγει εκτελέσιμα αρχεία (π.χ. .exe).

3.6 Εντολές που χρησιμοποιήθηκαν

Οι εντολές που χρησιμοποιήθηκαν για την δημιουργία του προγράμματος είναι οι εξής:

- clear all: Καθαρίζει το Workspace
- clc: Καθαρίζει το Command Window
- format compact: Εξοικονόμηση χώρου στο Command Window

Πριν το format compact:

```
a

a =

    2     3
```

Μετά το format compact:

```
a
a =
    2     3
```

- format short: Εμφανίζει μέχρι 4 δεκαδικά ψηφία
- for: Εντολή επανάληψης, η οποία δομείται ως εξής:

```
for i=1:5
      j(i)=i+1;
end
j
j =
    2     3     4     5     6
```

- if-else: Εντολή ελέγχου, η οποία δομείται ως εξής:

```
if a==2
    disp('True')
else
  disp('False')
end
```

- while: Επαναλαμβάνει μια ή περισσότερες εντολές όσο ισχύει μια συνθήκη

```
While συνθήκη
       Εντολές
end
```

- input: Εμφανίζει στην οθόνη μια δήλωση και περιμένει να λάβει μια τιμή από το πληκτρολόγιο

```
AGE = input ('Πόσο χρονών είσαι? \n');
Πόσο χρονών είσαι?
```

- round: Στρογγυλοποίηση στον πλησιέστερο ακέραιο
- disp: Εμφανίζει ένα κείμενο ή μια μεταβλητή στην οθόνη

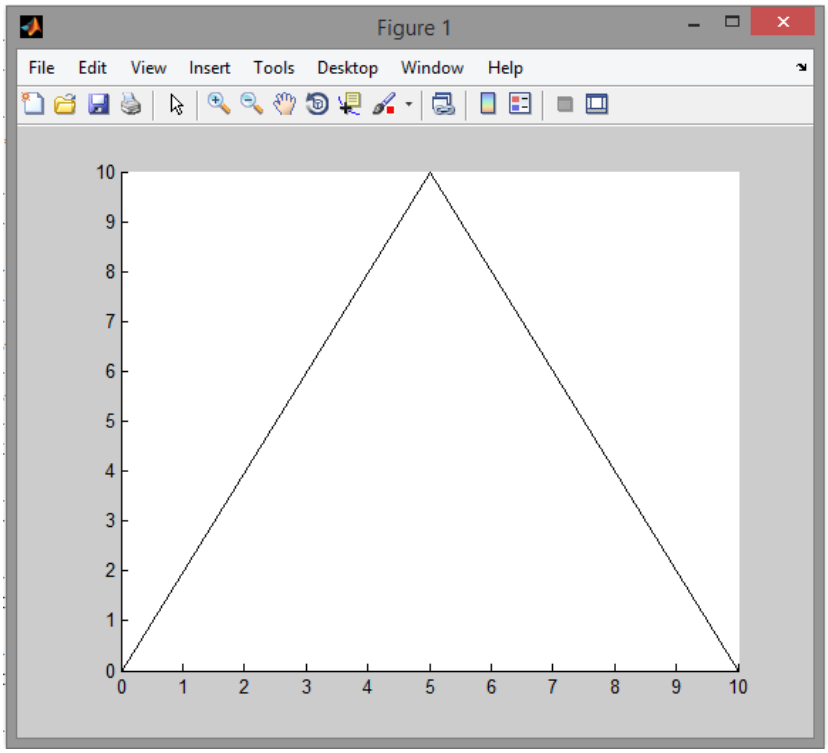
```
disp('Wake me up when its all over')
Wake me up when its all over
```

- num2str: Μετατρέπει ένα νούμερο σε γραμματοσειρά, ως αποτέλεσμα να μπορεί να αναγνωρίζει το πρόγραμμα το νούμερο σαν γράμμα

```
disp(['Κόμβος : ' ,1])
Κόμβος :
>> disp(['Κόμβος : ' ,num2str(2)])
Κόμβος : 2
```

- sqrt: Τετραγωνική ρίζα
- patch: Ενώνει συντεταγμένες x με y με μια χρωματιστή γραμμή

```
SX=[0 10 5]
SX =
    0    10     5
>> SY=[0 0 10]
SY =
    0     0    10
>> AKR=[1 2 3]
AKR =
    1     2     3
>> TKR=[2 3 1]
TKR =
    2     3     1
>> xdata=[SX(AKR);SX(TKR)];
ydata=[SY(AKR);SY(TKR)];
patch(xdata,ydata,'w')
```



- find: Εντοπίζει ένα νούμερο μέσα σε έναν πίνακα

```
>> A=[2 5 3 5 5];
>> B=find(A==5)
B =
    2     4     5
```

Δηλαδή , το 2 4 5 συμβολίζει ότι το νούμερο 5 βρίσκεται στις θέσεις 2,4,5 του πίνακα Α.

- ismember: Ελέγχει αν μια μεταβλητή είναι μέρος ενός πίνακα

```
A=[1 2 3 4 5];
if ismember(5,A)
   disp('Το 5 είναι μέρος του πίνακα Α');
else
   disp('Το 5 δεν είναι μέρος του πίνακα Α');
end
Το 5 είναι μέρος του πίνακα Α
```

- sum: Προσθέτει όλες τις μεταβλητές ενός πίνακα

```
A=[2 -3 5];
B=[-2 3];
C=sum(A)+sum(B)
C =
    5
```

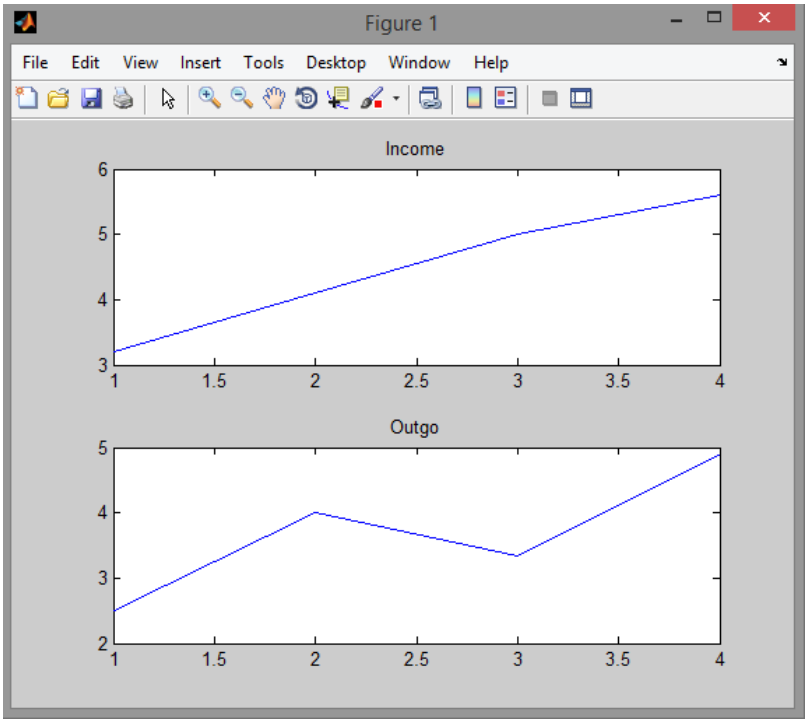
- floor: Στρογγυλοποιεί έναν δεκαδικό αριθμό προς το μείον άπειρο

```
A=5.9;
B=floor(A)
B =
    5
>> C=3.01;
D=floor(C)
D =
    3
```

- mod: Το υπόλοιπο μιας διαίρεσης

```
C=mod(10,3)
C =
    1
```

- subplot: Δημιουργία πολλαπλών αξόνων στο ίδιο παράθυρο



- title: τοποθετεί όνομα στους άξονες
- axis: διάφορες επιλογές και προτιμήσεις για τους άξονες
- text: δημιουργία ενός κειμένου σε ένα σχεδιάγραμμα

Κεφάλαιο 4

4 Ανάλυση του κώδικα επίλυσης επίπεδου δικτυώματος

4.1 Εισαγωγή

Η συγγραφή κώδικα έγινε όσο το δυνατόν πιο ευανάγνωστη και κατανοητή με σκοπό την περεταίρω ανάπτυξη του προγράμματος.

4.2 Επεξήγηση μεταβλητών

Εδώ θα γίνει μια σύντομη περιγραφή της κάθε μεταβλητής για διευκόλυνση στους αναγνώστες του προγράμματος. Βέβαια υπάρχουν και κάποιες μεταβλητές που αλλάζουν συνέχεια τιμή ή χρησιμοποιηθήκαν για βοηθητικούς λόγους, οι οποίες εντοπίζονται στο πρόγραμμα με μικρά γράμματα και δεν αναφέρονται παρακάτω.

- **E**: Σταθερά 210 Gpa
- **NN**: Αριθμός κόμβων δικτυώματος
- **NB**: Αριθμός ράβδων δικτυώματος
- **NF**: Αριθμός κόμβων που δέχονται δύναμη
- **NS**: Αριθμός κόμβων που δέχονται δύναμη
- **SX**: Συντεταγμένες κόμβων x
- **SY**: Συντεταγμένες κόμβων y
- **P1**:Πίνακας με τους κόμβους του δικτυώματος και τις συντεταγμένες αυτών
- **P2**: Πίνακας με τις ράβδους του δικτυώματος, τον αρχικό, τελικό κόμβο και την διατομή της κάθε ράβδου.
- **P3**:Πίνακας με τους κόμβους που δέχονται δύναμη, σε ποιόν άξονα δέχονται δύναμη και πόση είναι αυτή η δύναμη
- **P4**:Πίνακας με τους κόμβους που δέχονται στήριξη και τι στήριξη είναι αυτή
- **AKR**: Αρχικός κόμβος της κάθε ράβδου
- **TKR**: Τελικός κόμβος της κάθε ράβδου
- **ASX**: Συντεταγμένες x των αρχικών κόμβων της κάθε ράβδου
- **ASY**: Συντεταγμένες y των αρχικών κόμβων της κάθε ράβδου
- **TSX**: Συντεταγμένες x των τελικών κόμβων της κάθε ράβδου
- **TSY**: Συντεταγμένες y των τελικών κόμβων της κάθε ράβδου
- **D**: Διατομές των ράβδων
- **L**: Μήκη των ράβδων
- **KD**: Οι κόμβοι που δέχονται δύναμη
- **KDX**: Οι δυνάμεις που δέχονται οι κόμβοι στον άξονα x
- **KDY**: Οι δυνάμεις που δέχονται οι κόμβοι στον άξονα y
- **KS**: Οι κόμβοι που δέχονται στήριξη

- **KSX**: Οι κόμβοι που μπορούν να κινηθούν στον άξονα των x
- **KSY**: Οι κόμβοι που μπορούν να κινηθούν στον άξονα των y
- **SIN:** Πίνακας με τα ημίτονα τις κάθε ράβδου
- **COS**: Πίνακας με τα συνημίτονα της κάθε ράβδου
- **MS**: Μητρώο συστροφής
- **I**: (E*D)/L
- **KEL**: Μητρώο δυσκαμψίας της κάθε ράβδου
- **AR**: Δείχνει ποιες ράβδοι έχουν αρχικό τον κάθε κόμβο
- **TR**: Δείχνει ποιες ράβδοι έχουν τελικό τον κάθε κόμβο
- **NODES**: NN*2
- **OMD**: Ολικό μητρώο δυσκαμψίας
- **ARXDIAGX**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους αρχικούς κόμβους για F(x) ως προς q(x)
- **TELDIAGX***:* Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους τελικούς κόμβους για F(x) ως προς q(x)
- **ARXDIAGY**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους αρχικούς κόμβους για F(y) ως προς q(y)
- **TELDIAGY**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους αρχικούς κόμβους για F(y) ως προς q(y)
- **ARXUPX**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους αρχικούς κόμβους για F(x) ως προς q(y)
- **TELUPX**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους τελικούς κόμβους για F(x) ως προς q(y)
- **ARXDOWNY**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους αρχικούς κόμβους για F(y) ως προς q(x)
- **TELDOWNY**: Τα στοιχεία των μητρώων δυσκαμψίας της κάθε ράβδου τα οποία επηρεάζουν τους ανάλογους τελικούς κόμβους για F(y) ως προς q(x)

(Σημείωση: παρακάτω, σε αυτό το κεφάλαιο, θα δούμε σε παράδειγμα το ολικό μητρώο δυσκαμψίας ενός δικτυώματος όπου και θα γίνουν ξεκάθαρες οι έννοιες F(x), F(y), q(x), q(y)).

- **HELPDIAGX**: Το άθροισμα των στοιχείων της κάθε γραμμής του πίνακα ARXDIAGX με των στοιχείων της αντίστοιχης γραμμής του TELDIAGX
- **HELPDIAGY**: Το άθροισμα των στοιχείων της κάθε γραμμής του πίνακα ARXDIAGY με των στοιχείων της αντίστοιχης γραμμής του TELDIAGY
- **HELPKELX**: Το άθροισμα των στοιχείων της κάθε γραμμής του πίνακα ARXUPX με των στοιχείων της αντίστοιχης γραμμής του TELUPX
- **HELPKELY**: Το άθροισμα των στοιχείων της κάθε γραμμής του πίνακα ARXDOWNY με των στοιχείων της αντίστοιχης γραμμής του TELDOWNY
- **DYN:** Το ποσό της δύναμης που δέχεται ο κάθε κόμβος
- **STIRIKSEIS:** Τους κόμβους που δέχονται στήριξη και το είδος της στήριξης

(Το 1 δηλώνει ότι ο κόμβος δεν μπορεί να κινηθεί σε αυτόν τον άξονα, το 0 δηλώνει ότι ο κόμβος μπορεί να κινηθεί σε αυτόν τον άξονα)

- **HOLD:** Το ολικό μητρώο δυσκαμψίας μαζί με τις στηρίξεις των κόμβων.
- **Q:** Οι μετατοπίσεις του κάθε κόμβου μετά την επίλυση του δικτυώματος.
- **QX:** Οι μετατοπίσεις του κάθε κόμβου στον άξονα x
- **QY:** Οι μετατοπίσεις του κάθε κόμβου στον άξονα y
- **QARXX:** Την μετατόπιση των αρχικών κόμβων της κάθε ράβδου στον άξονα x
- **QARXY:** Την μετατόπιση των αρχικών κόμβων της κάθε ράβδου στον άξονα y
- **QTELX:**Την μετατόπιση των τελικών κόμβων της κάθε ράβδου στον άξονα x
- **QTELY:** Την μετατόπιση των τελικών κόμβων της κάθε ράβδου στον άξονα y
- **HELPTASEIS:** Τις κομβικές μετατοπίσεις της κάθε ράβδου
- **TASEIS:** Οι τάσεις στις ράβδους
- **OMDR:** Το προσαρμοσμένο μητρώο δυσκαμψίας του δικτυώματος
- **R:** Αντιδράσεις στηρίξεων
- **HOPEX:** Οι νέες συντεταγμένες x των κόμβων μετά την μετατόπιση τους
- **HOPEY:** Οι νέες συντεταγμένες y των κόμβων μετά την μετατόπιση τους

4.3 Δομή Κώδικα

Θα μπορούσαμε να πούμε ότι ο κώδικας χωρίζεται σε τρία βασικά μέρη:

1. Στην εισαγωγή δεδομένων από τον χρήστη
2. Στην επίλυση του δικτυώματος με την μέθοδο των πεπερασμένων στοιχείων
3. Εμφάνιση των αποτελεσμάτων και διαγραμμάτων

Ας δούμε το πρώτο μέρος όπου ο χρήστης εισάγει τα δεδομένα για την περιγραφή του δικτυώματος. Εδώ το πρόγραμμα στην ουσία, περιμένει την περιγραφή του δικτυώματος κάνοντας σύγχρονος κάποιους λογικούς ελέγχους.

Για αρχή θα πρέπει να δηλωθούν οι κόμβοι του δικτυώματος.

```
NN = input ('Εισάγετε τον αριθμό κόμβων του δικτυώματος. \n');
    NN = round(NN);        %Στρογγυλοποίηση αριθμού

while NN~=1:100            %Έλεγχος θετικού αριθμού και μέχρι 100
    NN = input ('Παρακαλώ εισάγετε αριθμό κόμβων ανάμεσα στο 1 εως
100.\n');
    NN = round(NN);        %Στρογγυλοποίηση στον πλησιέστερο ακέραιο
end
```

Εδώ διακρίνεται ότι ο αριθμός κόμβων του δικτυώματος που μπορεί να οριστεί είναι από 1 μέχρι 100 κόμβους. Σε περίπτωση αρνητικού ή μηδενικού αριθμού κόμβων, ζητείται από τον χρήστη να ξανά εισάγει τον αριθμό των κόμβων.

Με τις ράβδους βλέπουμε ακριβώς την ίδια λειτουργία, με την διαφορά ότι μπορούν να εισαχθούν μέχρι 150 ράβδοι.

```
NB = input ('Εισάγετε τον αριθμό ράβδων του δικτυώματος. \n');
    NB = round(NB);
while NB~=1:150           %Έλεγχος θετικού αριθμού και μέχρι 150
    NB = input ('Παρακαλώ εισάγετε αριθμό ράβδων ανάμεσα στο 1 έως
150.\n');
    NB = round(NB);
end
```

Για την εισαγωγή του αριθμού κόμβων που δέχονται δυνάμεις και στηρίξεις γίνετε ο έλεγχος του κόμβου που δηλώθηκε να υπάρχει στο δικτύωμα.

```
NF = input ('Πόσοι κόμβοι δέχονται δύναμη? \n');
while NF~=0:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    NF = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
end

NS = input ('Πόσοι κόμβοι δέχονται στήριξη? \n');
while NS~=0:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    NS = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
end
```

Τώρα θα οριστούν οι συντεταγμένες του κάθε κόμβου και θα εμφανιστούν στο τέλος σε έναν πίνακα για επιβεβαίωση από τον χρήστη.

```
                    %Συντεταγμένες κόμβων
for j=1:NN

    disp(['Κόμβος : ' ,num2str(j)])  %Μετατρέπω το νούμερο j σαν
γράμμα ώστε να μπορεί να το εμφανίσει στην οθόνη
    SX(j) = input ('Ποιές είναι οι συντεταγμένες x?\n');
    SY(j) = input ('Ποιές είναι οι συντεταγμένες y?\n');

end

P1=[1:NN ; SX ; SY];
disp('Συντεταγμένες κόμβων : ')
disp('   Κόμβος  x     y')
disp((P1'))
```

Στην οθόνη για παράδειγμα θα εμφανιστεί:

```
Συντεταγμένες κόμβων :
  Κόμβος  x    y
     1    5    5
     2   15    5
     3   10   15
```

Έπειτα ο χρήστης θα πρέπει να περιγράψει την κάθε ράβδο, δηλώνοντας τον αρχικό και τελικό κόμβο της ράβδου και την διατομή αυτής.

```
                    %Ενωμένες ράβδοι και ιδιότητες
for i=1:NB

    disp(['Ράβδος : ' ,num2str(i)])
    AKR(i) = input ('Ποιός είναι ο αρχικός κόμβος της ράβδου?\n');
    while AKR(i)~=1:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    AKR(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end
    ASX(i)=P1(2,AKR(i));         %Παίρνω τις συντεταγμένες χ του
αρχικού κόμβου της αντίστοιχη ράβδου
    ASY(i)=P1(3,AKR(i));         %Παίρνω τις συντεταγμένες y του
αρχικού κόμβου της αντίστοιχη ράβδου

    TKR(i) = input ('Ποιός είναι ο τελικός κόμβος της ράβδου?\n');
    while TKR(i)~=1:NN
    TKR(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end
    TSX(i)=P1(2,TKR(i));         %Παίρνω τις συντεταγμένες χ του
τελικού κόμβου της αντίστοιχη ράβδου
    TSY(i)=P1(3,TKR(i));         %Παίρνω τις συντεταγμένες y του
τελικού κόμβου της αντίστοιχη ράβδου

    D(i) = input ('Ποιά είναι η διατομή της ράβδου σε mm²?\n');
    while D(i)<=0            %Έλεγχος θετικού αριθμού
    D(i) = input ('Παρακαλώ εισάγετε θετικό αριθμό διατομής.\n');
    end

                        %Υπολογισμός μήκους ράβδων
    L(i)=sqrt((TSX(i)-ASX(i))^2+(TSY(i)-ASY(i))^2);

end

L;
P2=[1:NB ; AKR ; TKR ; D];
disp('Ενωμένες ράβδοι και ιδιότητες :')
disp('Ράβδος Αρχικ. Τελικ. Διατ.')
disp((P2'))
```

Μετά την δήλωση των τιμών, στην οθόνη θα εμφανιστεί:

```
Ενωμένες ράβδοι και ιδιότητες :
Ράβδος Αρχικ. Τελικ. Διατ.
   1      1      2     30
   2      2      3     30
   3      1      3     30
```

Εδώ θα οριστούν οι δυνάμεις που δέχονται οι κόμβοι

```
                        %Φορτίσεις στους κόμβους
for i=1:NF

    KD(i) = input ('Ποιός είναι ο κόμβος που δέχεται δύναμη?\n');
    while KD(i)~=1:NN          %Έλεγχος δεδομένων που εισάχθηκαν
    KD(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end

    disp(['Κόμβος : ' ,num2str(KD(i))])
    KDX(i) = input ('Πόση είναι η δύναμη που δέχεται ο κόμβος στον
άξονα x σε KN?\n');

    KDY(i) = input ('Πόση είναι η δύναμη που δέχεται ο κόμβος στον
άξονα y σε KN?\n');


end


P3=[KD ; KDX ; KDY];
disp('Φορτίσεις στους κόμβους : ')
disp('  Κόμβος  Px    Py')
disp((P3'))
```

Εδώ βλέπουμε ότι πάλι γίνεται ένας έλεγχος για να δηλωθεί ένας κόμβος που να έχει το δικτύωμα. Τα αποτελέσματα θα φάνουν στην οθόνη ως εξης:

```
Φορτίσεις στους κόμβους :
 Κόμβος  Px   Py
    3     0   10
```

Το τελευταίο στοιχείο που θα δηλώσει ο χρήστης είναι οι στηρίξεις στους κόμβους

```
                        %Στηρίξεις στους κόμβους
for i=1:NS

    KS(i) = input ('Εισάγεται τον κόμβο που δέχεται στήριξη.\n');
    while KS(i)~=1:NN            %Έλεγχος δεδομένων που εισάχθηκαν
    KS(i) = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
    end
```

```
    disp(['Κόμβος : ' ,num2str(KS(i))])
    KSX(i) = input ('Μπορεί να κινηθεί ο κόμβος στον άξονα των
X?[Y=0/N=1]\n');
    while KSX(i)~=0:1
        KSX(i) = input ('Παρακαλώ εισάγετε "0" αν ο κόμβος μπορεί να
κινηθεί στον άξονα των X και "1" αν όχι\n');
    end

    KSY(i) = input ('Μπορεί να κινηθεί ο κόμβος στον άξονα των
Y?[Y=0/N=1]\n');
     while KSY(i)~=0:1
        KSY(i) = input ('Παρακαλώ εισάγετε "0" αν ο κόμβος μπορεί να
κινηθεί στον άξονα των Y και "1" αν όχι\n');
    end


end

P4=[KS ; KSX; KSY ];
disp('Στηρίξεις στους κόμβους : ')
disp('Κόμβος     X     Y')
disp((P4'))
```

Τώρα εκτός του συνηθισμένο έλεγχο για τους κόμβους, γίνεται και ένας έλεγχος ότι ο χρήστης δήλωσε τις τιμές 0 ή 1. Όπου το 0 δηλώνει ότι ο κόμβος μπορεί να κινηθεί ενώ το 1 δηλώνει ότι δεν μπορεί να κινηθεί στον συγκεκριμένο άξονα. Στην οθόνη θα εμφανιστεί:

```
Στηρίξεις στους κόμβους :
Κόμβος    X    Y
   1      0    1
   3      1    1
```

Η εισαγωγή των δεδομένων και ουσιαστικά η περιγραφή του δικτυώματος τελείωσε.

Στην συνέχει το πρόγραμμα θα κάνει τις απαραίτητες πράξεις για την επίλυση του δικτυώματος με την μέθοδο των πεπερασμένων στοιχείων.

Για αρχή έχουμε τον υπολογισμό του μητρώου δυσκαμψίας της κάθε ράβδου

```
for i=1:NB
                          %Υπολογισμός ημιτόνου
      z=(TSX(i)-ASX(i))/L(i);
      SIN(i)=cosd(z);
                          %Υπολογισμός συνημίτονου
      z=(TSY(i)-ASY(i))/L(i);
      COS(i)=sind(z);
```

```
                        %Μητρώο συστροφής
      MS=[cosd(z)^2  sind(z)*cosd(z)  -cosd(z)^2  -sind(z)*cosd(z);
         sind(z)*cosd(z)  sind(z)^2  -sind(z)*cosd(z)  -sind(z)^2;
         -cosd(z)^2  -sind(z)*cosd(z)  cosd(z)^2  sind(z)*cosd(z);
         -sind(z)*cosd(z)  -sind(z)^2  sind(z)*cosd(z)  sind(z)^2;];

      I(i)=E*D(i)/L(i);

    Kel(1,1)=I(i)*cosd(z);
    Kel(1,2)=I(i)*sind(z)*cosd(z);
    Kel(1,3)=I(i)*(-cosd(z)^2);
    Kel(1,4)=I(i)*(-sind(z)*cosd(z));
    Kel(2,1)=Kel(1,2);
    Kel(2,2)=I(i)*sind(z)^2;
    Kel(2,3)=I(i)*(-sind(z)*cosd(z));
    Kel(2,4)=I(i)*(-sind(z)^2);
    Kel(3,1)=Kel(1,3);
    Kel(3,2)=Kel(2,3);
    Kel(3,3)=I(i)*cosd(z)^2;
    Kel(3,4)=I(i)*sind(z)*cosd(z);
    Kel(4,1)=Kel(1,4);
    Kel(4,2)=Kel(2,4);
    Kel(4,3)=Kel(3,4);
    Kel(4,4)=I(i)*sind(z)^2;

    KEL{i}=Kel;
end

KEL;          %Μητρώο δυσκαμψίας της κάθε ράβδου
SIN;          %Βοηθητικός πίνακας με ημίτονα
COS;          %Βοηθητικός πίνακας με συνημίτονα
I;            %Βοηθητικός πίνακας που υπολογίζει το E*D/L της κάθε
ράβδου
```

Το μητρώο δυσκαμψίας της κάθε ράβδου μπορεί να εμφανισθεί στην οθόνη καλώντας τον πίνακα KEL και τον αριθμό της ράβδου που θέλουμε. Για παράδειγμα, το μητρώο δυσκαμψίας της ράβδου 1 είναι :

```
KEL{1}
ans =
  420   0 -420   0
    0   0    0   0
 -420   0  420   0
    0   0    0   0
```

Τώρα ο σκοπός είναι η δημιουργία του ολικού μητρώου δυσκαμψίας. Η δημιουργία του ολικού μητρώου δυσκαμψίας χωρίζεται σε τρία στάδια τα οποία είναι τα εξής:

- Δημιουργία των στοιχείων της διαγωνίου
- Δημιουργία των στοιχείων δίπλα από την διαγώνιο
- Συμπλήρωση του υπόλοιπου πίνακα

Για την δημιουργία των στοιχείων της διαγωνίου του ολικού μητρώου δυσκαμψίας έχουμε τα παρακάτω.

Αρχικά δημιουργούνται οι παρακάτω πινάκες ως μηδενικοί πίνακες (στην παράγραφο 4.2 εξηγείται τι περιέχει ο κάθε πίνακας).

```
NODES=NN*2;
OMD(NODES,NODES)=0;
ARXDIAGX(NN,NB)=0;
TELDIAGX(NN,NB)=0;
ARXDIAGY(NN,NB)=0;
TELDIAGY(NN,NB)=0;
HELPDIAGX(NN,NN)=0;
HELPDIAGY(NN,NN)=0;
```

Έπειτα θα χρειαστούν δύο πίνακες που θα δείχνουν για τον κάθε κόμβο σε ποίες ράβδους είναι αρχικός και σε ποίες τελικός.

```
for i=1:NN

     AR{i}=find(AKR==i);      %Βρίσκω ο κάθε κόμβος σε ποιες ράβδους
είναι αρχικός,
     TR{i}=find(TKR==i);      %Βρίσκω ο κάθε κόμβος σε ποιες ράβδους
είναι τελικός
end
```

Για παράδειγμα αν ο χρήστης θέλει να δει ο κόμβος δύο, σε ποίες ράβδους είναι τελικός θα έχουμε

```
AR{2}
ans =
   2   4
```

Δηλαδή ο κόμβος δύο στην παρούσα περίπτωση είναι αρχικός στην ράβδο δύο και τέσσερα.

Τώρα θα χρειαστούν οι αρχικοί, τελικοί κόμβοι της κάθε ράβδου για F(x) ως προς q(x) και για F(y) ως προς q(y). Θα αποθηκευτούν στους πίνακες που δημιουργήθηκαν πριν.

```
                  %Παίρνω αρχικούς κόμβους για F(x) ως προς το q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i}) %Ψάχνει αν η ράβδος j ανήκει στον AK{i}

            ARXDIAGX(i,j)=KEL{j}(1,1); %Αν ανήκει τότε παίρνει από το
KEL της ράβδου το στοιχείο που επηρεάζει τον αρχικό κόμβο για F(i)(x)
ως προς q(i)(x) ,δηλαδή το (1,1)
        else
```

```
            ARXDIAGX(i,j)=0;
        end
    end
end

                %Παίρνω τελικούς κόμβους για F(x) ως προς το q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i}) %Ψάχνει αν η ράβδος j ανήκει στον TK{i}

            TELDIAGX(i,j)=KEL{j}(3,3); %Αν ανήκει τότε παίρνει από το
KEL της ράβδου το στοιχείο που επηρεάζει τον τελικό κόμβο για F(i)(x)
ως προς q(i)(x) ,δηλαδή το (3,3)
        else
            TELDIAGX(i,j)=0;
        end
    end
end

                    %Παίρνω αρχικούς κόμβους για F(y) ως προς το
q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})     %Ψάχνει αν η ράβδος j ανήκει στον
AK{i}

            ARXDIAGY(i,j)=KEL{j}(2,2);  %Αν ανήκει τότε παίρνει από
το KEL της ράβδου το στοιχείο που επηρεάζει τον αρχικό κόμβο για
F(i)(y) ως προς q(i)(y) ,δηλαδή το (2,2)
        else
            ARXDIAGY(i,j)=0;
        end
    end
end

                    %Παίρνω τελικούς κόμβους για F(y) ως προς το
q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i})    %Ψάχνει αν η ράβδος j ανήκει στον
TK{i}

            TELDIAGY(i,j)=KEL{j}(4,4);  %Αν ανήκει τότε παίρνει από
το KEL της ράβδου το στοιχείο που επηρεάζει τον τελικό κόμβο για
F(i)(y) ως προς q(i)(y) ,δηλαδή το (4,4)
        else
            TELDIAGY(i,j)=0;
        end
    end
end
```

Το επόμενο βήμα είναι να αποθηκευτεί σε έναν πίνακα το άθροισμα των αρχικών, τελικών κόμβων της κάθε ράβδου για F(x) ως προς q(x) και σε έναν άλλο το άθροισμα των αρχικών, τελικών κόμβων της κάθε ράβδου για F(y) ως προς q(y).

```
          %Άθροισμα αρχικών και τελικών κόμβων για F(x)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))  %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i


            HELPDIAGX(i,j)=sum(ARXDIAGX(i,:),2) +
sum(TELDIAGX(i,:),2);      %Το sum(x,1) προσθέτει στήλες ενώ το
sum(x,2) γραμμές

        elseif ismember(j,TKR(TR{i}))   %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i

            HELPDIAGX(i,j)=sum(ARXDIAGX(i,:),2) +
sum(TELDIAGX(i,:),2);


        end
    end
end

                %Άθροισμα αρχικών και τελικών κόμβων για F(y)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))  %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i


            HELPDIAGY(i,j)=sum(ARXDIAGY(i,:),2) +
sum(TELDIAGY(i,:),2);

        elseif ismember(j,TKR(TR{i}))   %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i

            HELPDIAGY(i,j)=sum(ARXDIAGY(i,:),2) +
sum(TELDIAGY(i,:),2);


        end
    end
end
```

Τέλος, έχοντας τα απαραίτητα στοιχεία, το συμπλήρωμα της διαγωνίου γίνεται ως εξής:

```
                %Συμπλήρωμα της διαγωνίου

for i=1:NODES

    div=(i+1)/2;           %Προσθέτω στο i +1 και έπειτα το διαιρώ με
2 έτσι ώστε να μπορεί να παίρνει στοιχεία από τους πίνακες
HELPDIAGX,HELPDIAGY που είναι 5x5 και έπειτα τα τοποθετώ στον OMD που
είναι 10x10
    v=floor(div);

    ypol=mod((i+1),2);


    if ypol==0

        OMD(i,i)=HELPDIAGX(v,v);

    elseif ypol==1

        OMD(i,i)=HELPDIAGY(v,v);

    end
end
```

Τώρα ξεκινάει το δεύτερο στάδιο, το οποίο είναι η δημιουργία των στοιχείων δίπλα από την διαγώνιο. Η διαδικασία που ακολουθείτε είναι σχεδόν η ίδια.

Αρχικά δημιουργούνται έξι μηδενικοί πίνακες

```
ARXUPX(NN,NB)=0;
TELUPX(NN,NB)=0;
ARXDOWNY(NN,NB)=0;
TELDOWNY(NN,NB)=0;
HELPKELX(NN,NN)=0;
HELPKELY(NN,NN)=0;
```

Έπειτα θα χρειαστούν οι αρχικοί, τελικοί κόμβοι της κάθε ράβδου για F(x) ως προς q(y) και για F(y) ως προς q(x).

```
                %Παίρνω αρχικούς κόμβους για F(x) ως προς q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})

            ARXUPX(i,j)=KEL{j}(1,2);
        else
            ARXUPX(i,j)=0;
        end
    end
end


                    %Παίρνω τελικούς κόμβους για F(x)ως προς q(y)
```

```
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i})

            TELUPX(i,j)=KEL{j}(3,4);
        else
            TELUPX(i,j)=0;
        end
    end
end



                    %Παίρνω αρχικούς κόμβους για F(y) ως προς q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})

            ARXDOWNY(i,j)=KEL{j}(2,1);
        else
            ARXDOWNY(i,j)=0;
        end
    end
end

                    %Παίρνω τελικούς κόμβους για F(y) ως προς q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i})

            TELDOWNY(i,j)=KEL{j}(4,3);
        else
            TELDOWNY(i,j)=0;
        end
    end
end
```

Το επόμενο βήμα πάλι, είναι να αποθηκευτεί σε έναν πίνακα το άθροισμα των αρχικών, τελικών κόμβων της κάθε ράβδου για F(x) ως προς q(y) και σε έναν άλλο το άθροισμα των αρχικών, τελικών κόμβων της κάθε ράβδου για F(y) ως προς q(x).

```
                    %Άθροισμα αρχικών και τελικών κόμβων για F(x)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))


            HELPKELX(i,j)=sum(ARXUPX(i,:),2) + sum(TELUPX(i,:),2);

        elseif ismember(j,TKR(TR{i}))
```

```
            HELPKELX(i,j)=sum(ARXUPX(i,:),2) + sum(TELUPX(i,:),2);


        end
    end
end

                    %Άθροισμα αρχικών και τελικών κόμβων για F(y)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))


            HELPKELY(i,j)=sum(ARXDOWNY(i,:),2) +
sum(TELDOWNY(i,:),2);

        elseif ismember(j,TKR(TR{i}))

            HELPKELY(i,j)=sum(ARXDOWNY(i,:),2) +
sum(TELDOWNY(i,:),2);


        end
    end
end
```

Το συμπλήρωμα των στοιχείων δίπλα από την διαγώνιο γίνεται ως εξής:

```
            %Συμπλήρωμα στοιχείων δίπλα από τα F(x),F(y)

for i=1:NODES

    div=(i+1)/2;
    v=floor(div);

    ypol=mod((i+1),2);

    if ypol==0

        OMD(i,i+1)=HELPKELX(v,v);

     elseif ypol==1

        OMD(i,i-1)=HELPKELY(v,v);

    end
end
```

Το τελευταίο βήμα είναι να ολοκληρωθεί η συμπλήρωση του ολικού μητρώου δυσκαμψίας.

```
                    %ΣΤΟΙΧΕΙΑ ΕΚΤΟΣ ΔΙΑΓΩΝΙΟΥ

                  %Συμπλήρωμα του υπόλοιπου πίνακα

for i=1:NODES

    div=(i+1)/2;
    v=floor(div);

    ypol=mod((i+1),2);

                      %Συμπληρώνει γραμμές F(x)
   if ypol==0
     for j=1:NB
         if v==P2(2,j)

            OMD(i,P2(3,j)*2-1)=KEL{P2(1,j)}(1,3);   %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(3,j)*2)  =KEL{P2(1,j)}(1,4);   %Συμπληρώνει
στήλες q(y)

         elseif v==P2(3,j)

            OMD(i,P2(2,j)*2-1)=KEL{P2(1,j)}(3,1);   %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(2,j)*2)  =KEL{P2(1,j)}(3,2);   %Συμπληρώνει
στήλες q(y)

         end
     end

                      %Συμπληρώνει γραμμές F(y)
   elseif ypol==1
       for j=1:NB
          if v==P2(2,j)

            OMD(i,P2(3,j)*2-1)=KEL{P2(1,j)}(2,3);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(3,j)*2)  =KEL{P2(1,j)}(2,4);    %Συμπληρώνει
στήλες q(y)

          elseif v==P2(3,j)

            OMD(i,P2(2,j)*2-1)=KEL{P2(1,j)}(4,1);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(2,j)*2)  =KEL{P2(1,j)}(4,2);    %Συμπληρώνει
στήλες q(y)

          end
       end
    end
end
```

Μετά την συμπλήρωση του ολικού μητρώου δυσκαμψίας τοποθετούνται σε ένα πίνακα οι φορτίσεις του κάθε κόμβου.

```
                    %ΦΟΡΤΙΣΕΙΣ ΣΤΟΥΣ ΚΟΜΒΟΥΣ

DYN(NODES,1)=0;
fortiseis(2,NN)=0;


for j=1:NF
 for i =1:NN
      if ismember(i,KD)
    fortiseis(1,KD(j))=KDX(j);
    fortiseis(2,KD(j))=KDY(j);
      end
 end
end



 for i=1:NODES
    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);

    if ypol==0
        if ismember(v,KD)

         DYN(i)=fortiseis(1,v);

        end
    elseif ypol==1
        if ismember(v,KD)

          DYN(i)=fortiseis(2,v);

        end
    end
 end
```

Ο πίνακας DYN θα εμφανίζει σε κάθε γραμμή αν κάποιος κόμβος δέχεται δύναμη στον άξονα x ή y, για παράδειγμα αν ο κόμβος 3 δέχεται δύναμη 5KN στον άξονα x, θα έχουμε:

```
DYN =
   0
   0
   0
   0
   5
   0
```

Έπειτα τοποθετούνται στο ολικό μητρώο δυσκαμψίας οι στηρίξεις των κόμβων. Το πρόγραμμα εδώ περιλαμβάνει τις πιθανές περιπτώσεις για κύλιση ή άρθρωση σε κάθε κόμβο.

```
STIRIKSEIS(3,NN)=0;
 HOLD=OMD;

for j=1:NS
    for i=1:NN
        if ismember(i,KS)
            STIRIKSEIS(1,KS(j))=KS(j);
            STIRIKSEIS(2,KS(j))=KSX(j);
            STIRIKSEIS(3,KS(j))=KSY(j);
        end
    end
end



for i=1:NODES
    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);
                                        %0 = kinhtai  1 = den kinhtai
    if ypol==0
        if ismember(v,KS)&& STIRIKSEIS(2,v)==1 && STIRIKSEIS(3,v)==1
%'Αρθρωση
            HOLD(i,:)=0;
            HOLD(:,i)=0;
            HOLD(i,i)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==0 &&
STIRIKSEIS(3,v)==1  %Κύλιση στον άξονα x
            HOLD(i+1,:)=0;
            HOLD(:,i+1)=0;
            HOLD(i,i)=OMD(i,i);
            HOLD(i+1,i+1)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==1 &&
STIRIKSEIS(3,v)==0  %Κύλιση στον άξονα y
           HOLD(i,:)=0;
           HOLD(:,i)=0;
           HOLD(i,i)=OMD(i,i);
        end

    elseif ypol==1
         if ismember(v,KS)&& STIRIKSEIS(2,v)==1 && STIRIKSEIS(3,v)==1
%'Αρθρωση
             HOLD(i,:)=0;
             HOLD(:,i)=0;
             HOLD(i,i)=1;

          elseif ismember(v,KS)&& STIRIKSEIS(2,v)==0 &&
STIRIKSEIS(3,v)==1  %Κύλιση στον άξονα x
              HOLD(i,:)=0;
              HOLD(:,i)=0;
              HOLD(i,i)=1;
```

```
          elseif ismember(v,KS)&& STIRIKSEIS(2,v)==1 &&
STIRIKSEIS(3,v)==0  %Κύλιση τον άξονα y
              HOLD(i-1,:)=0;
              HOLD(:,i-1)=0;
              HOLD(i,i)=OMD(i,i);
              HOLD(i-1,i-1)=1;
         end
    end
end


Q=HOLD\DYN;
```

Πλέον το πρόγραμμα έχει υπολογίσει τις τελικές μετατοπίσεις του κάθε κόμβου αποθηκευμένες στον πίνακα Q.

Τώρα θα υπολογιστούν οι τάσεις στις ράβδους καθώς και οι αντιδράσεις στηρίξεων.

```
for i=1:NODES

    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);

    if ypol==0
        QX(v)=Q(i);

    elseif ypol==1
        QY(v)=Q(i);
    end
end

for i=1:NB
    for j=1:NN
        if P2(2,i)==j
            QARXX(i)=QX(j);
            QARXY(i)=QY(j);

        elseif P2(3,i)==j
            QTELX(i)=QX(j);
            QTELY(i)=QY(j);

        end
    end
end


for i=1:NB
    HELPTASEIS{i}(1,1)=QARXX(i);
    HELPTASEIS{i}(2,1)=QARXY(i);
    HELPTASEIS{i}(3,1)=QTELX(i);
    HELPTASEIS{i}(4,1)=QTELY(i);
end
```

```
for i=1:NB
    TASEIS{i}=(KEL{i}/D(i))*HELPTASEIS{i};
end



for i =1:NODES
    for j=1:NODES

    r1=sum(HOLD(i,:));

    if r1==1
        OMDR(i,j)=OMD(i,j);

    elseif r1~=1
       OMDR(i,j)=0;

    end
    end
end
```

Εδώ τελειώνει το δεύτερο μέρος του προγράμματος και τώρα ακολουθεί η εμφάνιση των αποτελεσμάτων και διαγραμμάτων.

Τα αποτελέσματα εμφανίζονται εδώ:

```
Q
TASEIS
R=OMDR*Q;
```

Η εμφάνιση των διαγραμμάτων χωρίζεται σε δυο μέρη. Το δικτύωμα στην αρχική του μορφή και το δικτύωμα στην τελική του μορφή μετά τις μετατοπίσεις των κόμβων.

Για το αρχικό δικτύωμα έχουμε τα εξής:

```
maxx=SX(1);
 for i=1:NN
  if SX(i)>maxx
  maxx=SX(i);
  end
 end

 maxy=SY(1);
 for i=1:NN
     if SY(i)>maxy
         maxy=SY(i);
     end
 end

 minx=SX(1);
 for i=1:NN
     if SX(i)<minx
    minx=SX(i);
```

```
    end
end

miny=SY(1);
for i=1:NN
    if SY(i)<miny
        miny=SY(i);
    end
end
```

Εδώ εντοπίζεται η πιο χαμηλή συντεταγμένη στους άξονες x,y και η πιο ψηλή συντεταγμένη στους άξονες x,y. Αυτό γίνεται για την ρύθμιση του κάθε άξονα.

Η δημιουργία του αρχικού δικτυώματος ολοκληρώνεται παρακάτω

```
xdata=[SX(AKR);SX(TKR)];
ydata=[SY(AKR);SY(TKR)];

subplot(2,1,1); patch(xdata,ydata,'r')
title('Αρχικό δικτύωμα')
axis([(minx-10) (maxx+10) (miny-10) (maxy+10)])
for i=1:NS
    if P4(2,i)==1 && P4(3,i)==1
        text(SX(P4(1,i))-1,SY(P4(1,i))-
1,'Άρθρωση','EdgeColor','blue');
    else
        text(SX(P4(1,i))-1,SY(P4(1,i))-
1,'Κύλιση','EdgeColor','blue');
    end
end

for i=1:NF
    if P3(2,i)>0
        text(SX(P3(1,i)),SY(P3(1,i)),'\leftarrow');
    elseif P3(2,i)<0
        text(SX(P3(1,i)),SY(P3(1,i)),'\rightarrow');
    elseif P3(3,i)>0
        text(SX(P3(1,i)),SY(P3(1,i)),'\downarrow');
    elseif P3(3,i)<0
        text(SX(P3(1,i)),SY(P3(1,i)),'\uparrow');
    end
end
```

Μαζί με το δικτύωμα εμφανίζονται οι δυνάμεις και οι στηρίξεις των κόμβων.

Για το τελικό δικτύωμα γίνεται η ίδια διαδικασία με έναν υπολογισμό συντεταγμένων στην αρχή. Μέτα τον υπολογισμό μετατοπίσεων των κόμβων σύμφωνα με την αρχική τους θέση έχουμε πάλι υπολογισμό των ορίων του κάθε άξονα για το τελικό δικτύωμα.

```
hopex=SX-(QX*-1);     % Ο πολλαπλασιασμός με -1 γίνεται για να
hopey=SY-(QY*-1);     % απεικονίζεται η θετική φορά προς τα δεξιά

 maxx1=hopex(1);
 for i=1:NN
  if hopex(i)>maxx1
  maxx1=hopex(i);
  end
 end

 maxy1=hopey(1);
 for i=1:NN
     if hopey(i)>maxy1
         maxy1=hopey(i);
     end
 end

 minx1=hopex(1);
 for i=1:NN
     if hopex(i)<minx1
    minx1=hopex(i);
     end
 end

 miny1=hopey(1);
 for i=1:NN
     if hopey(i)<miny1
         miny1=hopey(i);
     end
 end
```

Το τελικό δικτύωμα ολοκληρώνεται παρακάτω

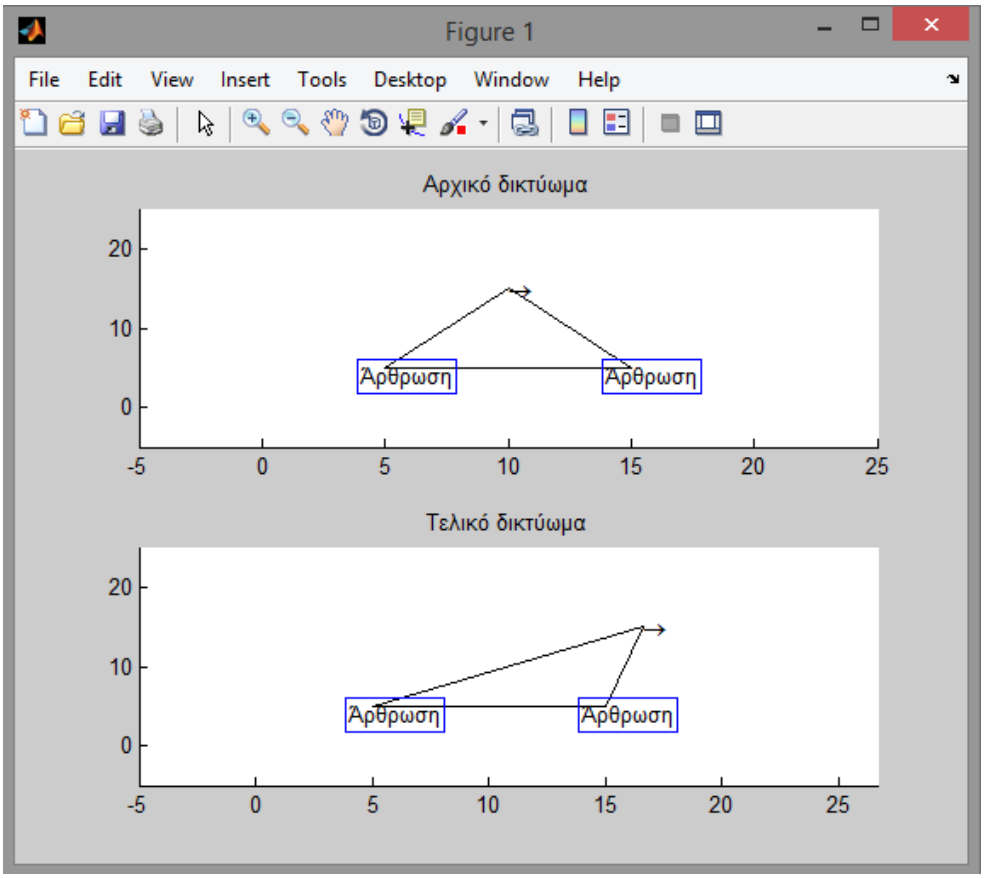
```
xdata1=[hopex(AKR);hopex(TKR)];
ydata1=[hopey(AKR);hopey(TKR)];

subplot(2,1,2); patch(xdata1,ydata1,'r')
title('Τελικό δικτύωμα')
axis([(minx1-10) (maxx1+10) (miny1-10) (maxy1+10)])
for i=1:NS
    if P4(2,i)==1 && P4(3,i)==1
        text(hopex(P4(1,i))-1,hopey(P4(1,i))-
1,'Άρθρωση','EdgeColor','blue');
    else
        text(hopex(P4(1,i))-1,hopey(P4(1,i))-
1,'Κύλιση','EdgeColor','blue');
   end
end

for i=1:NF
    if P3(2,i)>0
        text(hopex(P3(1,i)),hopey(P3(1,i)),'\leftarrow');
    elseif P3(2,i)<0
        text(hopex(P3(1,i)),hopey(P3(1,i)),'\rightarrow');
    elseif P3(3,i)>0
```

```
        text(hopex(P3(1,i)),hopey(P3(1,i)),'\downarrow');
    elseif P3(3,i)<0
        text(hopex(P3(1,i)),hopey(P3(1,i)),'\uparrow');
    end
end
```

Το διάγραμμα του αρχικού και τελικού δικτυώματος εμφανίζεται στο ίδιο παράθυρο με την παρακάτω μορφή:



Εδώ είναι και η ολοκλήρωση συγγραφής του κώδικα.

4.4 Παράδειγμα 1

Έστω ότι θέλουμε την επίλυση του παρακάτω δικτυώματος:

- E=210 GPa
- Αριθμός κόμβων: 4
- Αριθμός ράβδων: 5
- Κόμβοι που δέχονται δύναμη: 1
- Κόμβοι που δέχονται στήριξη: 2
- Συντεταγμένες κόμβων:

Κόμβος	X(mm)	Y(mm)
1	5	5
2	15	5
3	10	15
4	20	15

- Ενωμένοι ράβδοι και ιδιότητες:

Ράβδος	Αρχικός Κόμβος	Τελικός Κόμβος	Διατομή (mm^2)
1	1	2	20
2	2	3	20
3	3	1	20
4	2	4	20
5	3	4	20

- Φορτίσεις στους κόμβους:

Κόμβος	Px(KN)	Py(KN)
4	0	0.5

- Στηρίξεις στους κόμβους:

Κόμβος	X	Y
1	1	1
2	1	1

Το 0 δηλώνει ότι ο κόμβος μπορεί να κινηθεί στον συγκεκριμένο άξονα ενώ το 1 δηλώνει ότι ο κόμβος δεν μπορεί να κινηθεί στον συγκεκριμένο άξονα. Στην παρούσα περίπτωση ο κόμβος 1,2 έχουν άρθρωση.

Αυτά είναι όλα τα απαραίτητα στοιχεία που χρειάζεται το πρόγραμμα για την επίλυση του δικτυώματος.

Το ολικό μητρώο δυσκαμψίας του δικτυώματος είναι (OMD) :

Ολικό Μητρώο Δυσκαμψίας								
	q1(x)	q1(y)	q2(x)	q2(y)	q3(x)	q3(y)	q4(x)	q4(y)
F1(x)	795,5679	-5,86335	-420	0	-375,568	5,863354	0	0
F1(y)	-5,86335	0,091538	0	0	5,863354	-0,09154	0	0
F2(x)	-420	0	1171,227	11,72671	-375,568	-5,86335	-375,568	-5,86335
F2(y)	0	0	11,72671	0,183077	-5,86335	-0,09154	-5,86335	-0,09154
F3(x)	-375,568	5,863354	-375,568	-5,86335	1171,182	0	-420	0
F3(y)	5,863354	-0,09154	-5,86335	-0,09154	0	0,183077	0	0
F4(x)	0	0	-375,568	-5,86335	-420	0	795,5679	5,863354
F4(y)	0	0	-5,86335	-0,09154	0	0	5,863354	0,091538

Το ολικό μητρώο δυσκαμψίας του δικτυώματος μαζί με τις στηρίξεις στους κόμβους είναι (HOLD) :

Ολικό Μητρώο Δυσκαμψίας								
	q1(x)	q1(y)	q2(x)	q2(y)	q3(x)	q3(y)	q4(x)	q4(y)
F1(x)	1	0	0	0	0	0	0	0
F1(y)	0	1	0	0	0	0	0	0
F2(x)	0	0	1	0	0	0	0	0
F2(y)	0	0	0	1	0	0	0	0
F3(x)	0	0	0	0	1171,182	0	-420	0
F3(y)	0	0	0	0	0	0,183077	0	0
F4(x)	0	0	0	0	-420	0	795,5679	5,863354
F4(y)	0	0	0	0	0	0	5,863354	0,091538

Μετατοπίσεις κόμβων:

[DYN]=[HOLD]*[Q] όπου:

- DYN: Οι φορτίσεις των κόμβων
- HOLD: Το γενικό μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις των κόμβων

Q (mm):

q1(x)	0
q1(y)	0
q2(x)	0
q2(y)	0
q3(x)	-0,0426
q3(y)	0
q4(x)	-0,1188
q4(y)	13,0774

Τάσεις στις ράβδους:

{TASEIS}= {KEL}/D*{HELPTASEIS} όπου:

- TASEIS: Οι τάσεις στις ράβδους
- KEL: Το μητρώο δυσκαμψίας της κάθε ράβδου
- D= Η διατομή της κάθε ράβδου
- HELPTASEIS: Τις κομβικές μετατοπίσεις της κάθε ράβδου

TASEIS (GPa):

Ράβδος : 1	
σ1(x)	0
σ1(y)	0
σ2(x)	0
σ2(y)	0

Ράβδος : 2	
σ2(x)	0.8006
σ2(y)	0.0124
σ3(x)	-0.8006
σ3(y)	-0.0124

Ράβδος : 3	
σ3(x)	-0.8007
σ3(y)	0.0124
σ1(x)	0.8006
σ1(y)	-0.0124

Ράβδος : 4	
σ2(x)	-1.6013
σ2(y)	-0.025
σ4(x)	1.6013
σ4(y)	0.025

Ράβδος : 5	
σ3(x)	1.6013
σ3(y)	0
σ4(x)	-1.6013
σ4(y)	0

Αντιδράσεις στηρίξεων:

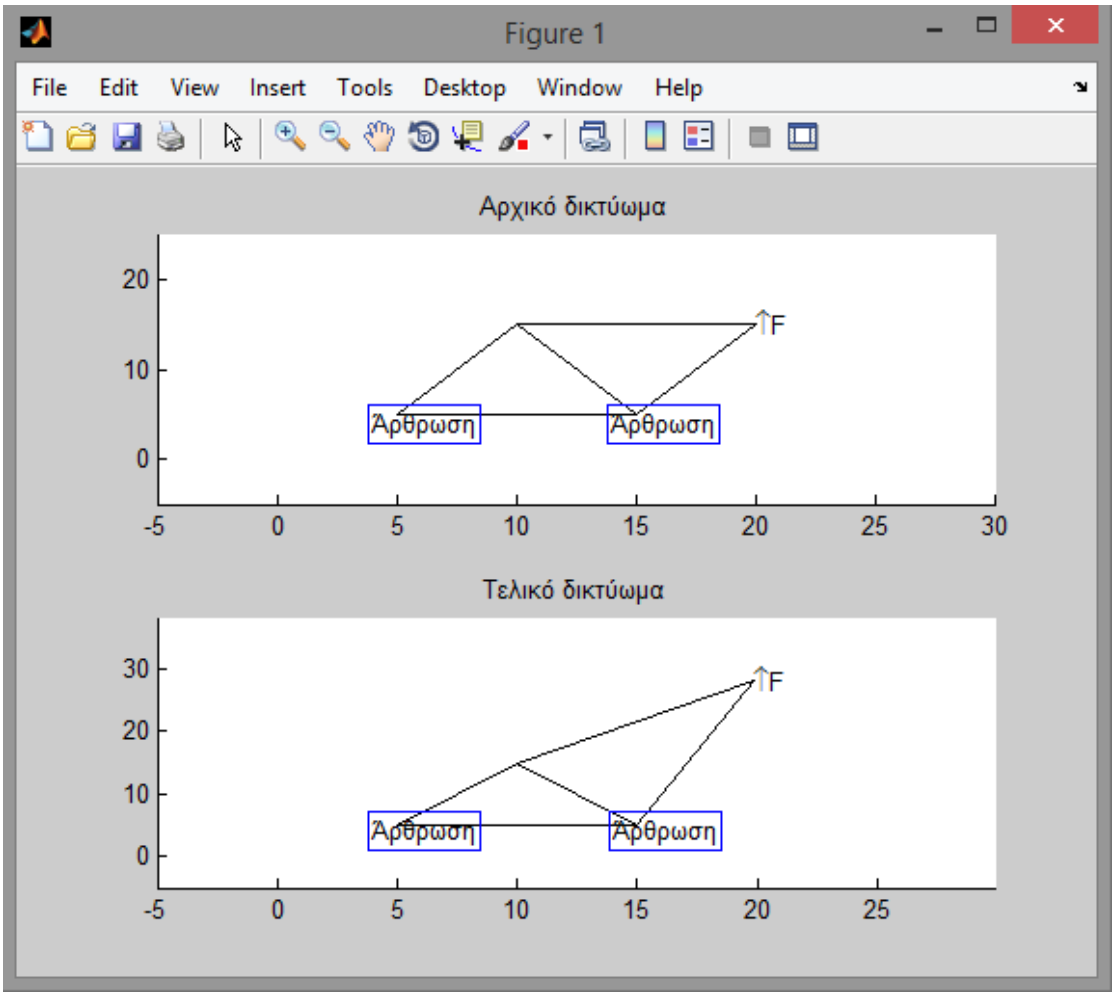
[R]=[HOLD]*[Q] οπού:

- R: Αντιδράσεις στηρίξεων
- HOLD: Το γενικό μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις των κόμβων

R (KN):

R1(x)	16.012
R1(y)	-0.24998
R2(x)	-16.014
R2(y)	-0.25002
R3(x)	0
R3(y)	0
R4(x)	0
R4(y)	0

Σχεδιάγραμμα δικτυώματος:



4.5 Παράδειγμα 2

Έχουμε το παρακάτω δικτύωμα:

- E=210 GPa
- Αριθμός κόμβων: 3
- Αριθμός ράβδων: 3
- Κόμβοι που δέχονται δύναμη: 1
- Κόμβοι που δέχονται στήριξη: 2
- Συντεταγμένες κόμβων:

Κόμβος	X(mm)	Y(mm)
1	5	5
2	15	5
3	10	200

- Ενωμένοι ράβδοι και ιδιότητες:

Ράβδος	Αρχικός Κόμβος	Τελικός Κόμβος	Διατομή (mm^2)
1	1	2	2
2	2	3	2
3	3	1	2

- Φορτίσεις στους κόμβους:

Κόμβος	Px(KN)	Py(KN)
3	50	0

- Στηρίξεις στους κόμβους:

Κόμβος	X	Y
1	1	1
2	0	1

Το ολικό μητρώο δυσκαμψίας του δικτυώματος είναι (OMD) :

Ολικό Μητρώο Δυσκαμψίας						
	q1(x)	q1(y)	q2(x)	q2(y)	q3(x)	q3(y)
F1(x)	44,15248	-0,03756	-42	0	-2,15248	0,037559
F1(y)	-0,03756	0,000655	0	0	0,037559	-0,00066
F2(x)	-42	0	44,15281	0,037559	-2,15248	-0,03756
F2(y)	0	0	0,037559	0,000655	-0,03756	-0,00066
F3(x)	-2,15248	0,037559	-2,15248	-0,03756	4,305294	0
F3(y)	0,037559	-0,00066	-0,03756	-0,00066	0	0,001311

Το ολικό μητρώο δυσκαμψίας του δικτυώματος μαζί με τις στηρίξεις στους κόμβους είναι (HOLD) :

Ολικό Μητρώο Δυσκαμψίας						
	q1(x)	q1(y)	q2(x)	q2(y)	q3(x)	q3(y)
F1(x)	1	0	0	0	0	0
F1(y)	0	1	0	0	0	0
F2(x)	0	0	44,15281	0	-2,15248	-0,03756
F2(y)	0	0	0	1	0	0
F3(x)	0	0	-2,15248	0	4,305294	0
F3(y)	0	0	-0,03756	0	0	0,001311

Μετατοπίσεις κόμβων:

[DYN]=[HOLD]*[Q] όπου:

- DYN: Οι φορτίσεις των κόμβων
- HOLD: Το γενικό μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις των κόμβων

Q (mm):

q1(x)	0
q1(y)	0
q2(x)	0.5951
q2(y)	0
q3(x)	11.911
q3(y)	17.055

Τάσεις στις ράβδους:

{TASEIS}= {KEL}/D*{HELPTASEIS} όπου:

- TASEIS: Οι τάσεις στις ράβδους
- KEL: Το μητρώο δυσκαμψίας της κάθε ράβδου
- D: Η διατομή της κάθε ράβδου
- HELPTASEIS: Τις κομβικές μετατοπίσεις της κάθε ράβδου

TASEIS (GPa):

Ράβδος : 1	
σ1(x)	-12.499
σ1(y)	0
σ2(x)	12.499
σ2(y)	0

Ράβδος : 2	
σ2(x)	-12.499
σ2(y)	-0.2181
σ3(x)	12.499
σ3(y)	0.2181

Ράβδος : 3	
σ3(x)	12.501
σ3(y)	-0.2181
σ1(x)	-12.499
σ1(y)	0.2181

Αντιδράσεις στηρίξεων:

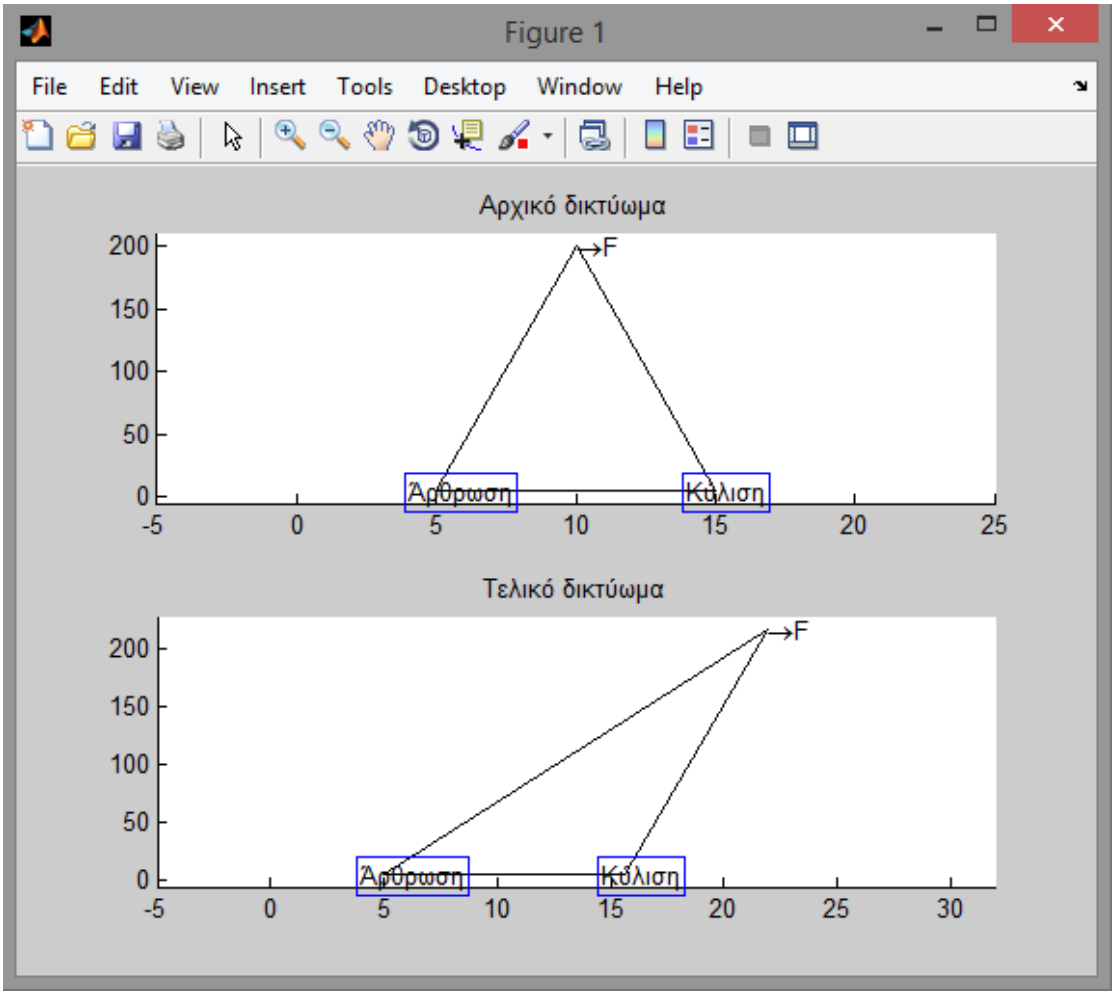
[R]=[HOLD]*[Q] οπού:

- R: Αντιδράσεις στηρίξεων
- HOLD: Το γενικό μητρώο δυσκαμψίας του δικτυώματος
- Q: Οι μετατοπίσεις των κόμβων

R (KN):

R1(x)	-49.996
R1(y)	0.4362
R2(x)	0
R2(y)	-0.4362
R3(x)	0
R3(y)	0

Σχεδιάγραμμα δικτυώματος:



Κεφάλαιο 5

5 Συμπεράσματα

5.1 Προσωπική άποψη

Τελειώνοντας την παρούσα πτυχιακή εργασία νιώθω χαρούμενος για την επιλογή του θέματος, καθώς και πολύ ικανοποιημένος για τις γνώσεις που μου προσέφερε κατά την διάρκεια της δημιουργίας της.

Ο προγραμματισμός είναι ένα κεφάλαιο που με ενδιέφερε από παλιά και μέσα από αυτή την πτυχιακή κλήθηκα να τον γνωρίσω καλύτερα. Πέρα από τις γνώσεις που απέκτησα στην διάρκεια των σπουδών μου ως μηχανολόγος μηχανικός δεν είχα ξανά ασχοληθεί με τον προγραμματισμό. Το να προγραμματίσω στην γλώσσα προγραμματισμού της matlab μου φάνηκε πολύ ενδιαφέρον παρότι συνάντησα αρκετές δυσκολίες μέχρι να την κατανοήσω καλύτερα.

Παρόλα αυτά το σημείο που συνάντησα τις περισσότερες δυσκολίες ήταν η κατανόηση των πεπερασμένων στοιχείων και η εφαρμογή τους μέσα στον προγραμματισμό. Καθοριστικός παράγοντας στο να ξεπεραστούν αυτές οι δυσκολίες ήταν η καθηγήτρια μου κ. Μανιάτη Μαρί-Μισέλ.

5.2 Αποτελέσματα πτυχιακής εργασίας

Η δημιουργία κώδικα για την επίλυση επίπεδων δικτυωμάτων επιτεύχθηκε σε αρκετά ικανοποιητικό βαθμό.

Στο πρόγραμμα μπορεί να περιγραφεί το δικτύωμα, οι φορτίσεις, οι στηρίξεις από τον χρήστη και το πρόγραμμα να κάνει επίλυση του δικτυώματος, καθώς και γραφική αναπαράσταση του. Το δικτύωμα μπορεί να έχει μέχρι 100 κόμβους και έως 150 ράβδους. Αυτό έγινε για καθαρούς λογούς μνήμης του υπολογιστή. Η αλλαγή των ορίων στους κόμβους και τους ράβδους δεν θα επηρεάσει την ορθή λειτουργία του προγράμματος.

Επιπλέον, έγινε επίλυση απλού δικτυώματος στο χαρτί σε σύγκριση με το ίδιο δικτύωμα στο πρόγραμμα για την εξέταση των αποτελεσμάτων.

5.3 Περιορισμοί προγράμματος

Το πρόγραμμα έχει αρκετούς περιορισμούς από τους οποίους οι σημαντικότεροι είναι:

- Δεν μπορούν να εισαχθούν φορτίσεις υπό γωνία
- Δεν υπάρχει δυνατότητα αλλαγής κάποιας μεταβλητής. Για παράδειγμα αν ο χρήστης θέλει να αλλάξει κάποια φόρτιση ή κάποιο στοιχείο του δικτυώματος θα πρέπει να τρέξει το πρόγραμμα από την αρχή

- Το περιβάλλον εισαγωγής δεδομένων θα μπορούσε να γίνει μέσα από εικονικά παράθυρα έτσι ώστε να είναι πιο φιλικό για τον χρήστη
- Η εμφάνιση των αποτελεσμάτων θα μπορούσε να είναι πιο ξεκάθαρη με επεξηγήσεις, για την καλύτερη κατανόηση τους από τον χρήστη
- Η γραφική απεικόνιση του δικτυώματος θα μπορούσε να έχει πιο ξεκάθαρες τις φορτίσεις και τις στηρίξεις των κόμβων

5.4 Βελτίωση – Περαιτέρω ανάπτυξη προγράμματος

Το πρόγραμμα θα μπορούσε να βελτιωθεί αρκετά. Μερικές ιδέες βελτιστοποίησης του προγράμματος είναι παρακάτω:

- Πιο ξεκάθαρος ορισμός μεταβλητών
- Δυνατότητα ορισμού θετικής φοράς των αξόνων στην γραφική αναπαράσταση του δικτυώματος
- Περισσότερα επεξηγηματικά σχόλια
- Δυνατότητα ορισμού μονάδων μέτρησης
- Δημιουργία του κώδικα σε function.m μορφή αντί σε script.m με σκοπό την δυνατότητα εισαγωγής βιβλιοθηκών
- Επίλυση των περιορισμών που αναφέρθηκαν στην παράγραφο 5.3

Επίσης, το πρόγραμμα είναι φτιαγμένο για την επίλυση διδιάστατων δικτυωμάτων. Μια περαιτέρω ανάπτυξη του προγράμματος θα μπορούσε να του δώσει την δυνατότητα να επιλύει τρισδιάστατα δικτυώματα και να δέχεται πακτώσεις.

Τέλος, το πρόγραμμα θα μπορούσε να εξελιχθεί έτσι ώστε να μπορεί να επιλύει πλαίσια με στοιχεία δοκού.

Βιβλιογραφία

1. Μ. Παπαδρακάκης, Ανάλυση Φορέων με την Μέθοδο των Πεπερασμένων Στοιχείων, εκδόσεις Παπασωτηρίου
2. Ανδρέας Ε. Κανάρχος – Χριστόφορος Γ. Προβατίδης, Πεπερασμένα Στοιχεία στη Μηχανολογία, εκδόσεις Παπασωτηρίου
3. http://users.auth.gr/~arvanit/Eisagogi_sto_Matlab.pdf
4. http://www.mathworks.com/products/matlab/
5. http://www.mathworks.com/help/pdf_doc/matlab/matlab_prog.pdf
6. http://www.eng.ucy.ac.cy/roussis/Courses/CEE220/CEE220_L18.pdf
7. http://www.eng.ucy.ac.cy/roussis/Courses/CEE220/CEE220_L14.pdf
8. http://www.eng.ucy.ac.cy/CEE325/CEE325_Spring2006/CEE221_Stiff_Review.pdf

Παράρτημα

Ο κώδικας Matlab

```
clear all
clc

format compact  %Εξοικονόμηση χωρου
format short g  %Βλέπω 4 δεκαδικά ψηφία
E=210;


                % 1)ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ

NN = input ('Εισάγετε τον αριθμό κόμβων του δικτυώματος. \n');
    NN = round(NN);       %Στρογγυλοποίηση αριθμού

while NN~=1:100           %Έλεγχος θετικού αριθμού και μέχρι 100
    NN = input ('Παρακαλώ εισάγετε αριθμό κόμβων ανάμεσα στο 1 εως
100.\n');
    NN = round(NN);       %Στρογγυλοποίηση στον πλησιέστερο ακέραιο
end

NB = input ('Εισάγετε τον αριθμό ράβδων του δικτυώματος. \n');
    NB = round(NB);
while NB~=1:150           %Έλεγχος θετικού αριθμού και μέχρι 150
    NB = input ('Παρακαλώ εισάγετε αριθμό ράβδων αναμεσα στο 1 εως
150.\n');
    NB = round(NB);
end

NF = input ('Πόσοι κόμβοι δέχονται δύναμη? \n');
while NF~=0:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    NF = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
end

NS = input ('Πόσοι κόμβοι δέχονται στήριξη? \n');
while NS~=0:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    NS = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
end

                    %Συντεταγμένες κόμβων
for j=1:NN

    disp(['Κόμβος : ' ,num2str(j)])  %Μετατρέπω το νούμερο j σαν
γράμμα ώστε να μπορεί να το εμφανίσει στην οθόνη
    SX(j) = input ('Ποιές είναι οι συντεταγμένες x?\n');
    SY(j) = input ('Ποιές είναι οι συντεταγμένες y?\n');

end

P1=[1:NN ; SX ; SY];
disp('Συντεταγμένες κόμβων : ')
disp('   Κόμβος  x     y')
```

```
disp((P1'))


                    %Ενωμένες ράβδοι και ιδιότητες
for i=1:NB

    disp(['Ράβδος : ' ,num2str(i)])
    AKR(i) = input ('Ποιός είναι ο αρχικός κόμβος της ράβδου?\n');
    while AKR(i)~=1:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    AKR(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end
    ASX(i)=P1(2,AKR(i));         %Παίρνω τις συντεταγμένες χ του
αρχικού κόμβου της αντίστοιχη ράβδου
    ASY(i)=P1(3,AKR(i));         %Παίρνω τις συντεταγμένες y του
αρχικού κόμβου της αντίστοιχη ράβδου

    TKR(i) = input ('Ποιός είναι ο τελικός κόμβος της ράβδου?\n');
    while TKR(i)~=1:NN
    TKR(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end
    TSX(i)=P1(2,TKR(i));         %Παίρνω τις συντεταγμένες χ του
τελικού κόμβου της αντίστοιχη ράβδου
    TSY(i)=P1(3,TKR(i));         %Παίρνω τις συντεταγμένες y του
τελικού κόμβου της αντίστοιχη ράβδου

    D(i) = input ('Ποιά είναι η διατομή της ράβδου σε mm²?\n');
    while D(i)<=0            %Έλεγχος θετικού αριθμού
    D(i) = input ('Παρακαλώ εισάγετε θετικό αριθμό διατομής.\n');
    end

                        %Υπολογισμός μήκους ράβδων
    L(i)=sqrt((TSX(i)-ASX(i))^2+(TSY(i)-ASY(i))^2);

end

L;
P2=[1:NB ; AKR ; TKR ; D];
disp('Ενωμένες ράβδοι και ιδιότητες :')
disp('Ράβδος Αρχικ. Τελικ. Διατ.')
disp((P2'))


                      %Φορτίσεις στους κόμβους
for i=1:NF

    KD(i) = input ('Ποιός είναι ο κόμβος που δέχεται δύναμη?\n');
    while KD(i)~=1:NN          %Έλεγχος δεδομένων που εισάχθηκαν
    KD(i) = input ('Παρακαλώ εισάγετε ένα κόμβο που έχει το
δικτύωμα.\n');
    end

    disp(['Κόμβος : ' ,num2str(KD(i))])
    KDX(i) = input ('Πόση είναι η δύναμη που δέχεται ο κόμβος στον
άξονα x σε KN?\n');
```

```
    KDY(i) = input ('Πόση είναι η δύναμη που δέχεται ο κόμβος στον
άξονα y σε KN?\n');


end


P3=[KD ; KDX ; KDY];
disp('Φορτίσεις στους κόμβους : ')
disp('  Κόμβος   Px     Py')
disp((P3'))




                        %Στηρίξεις στους κόμβους
for i=1:NS

    KS(i) = input ('Εισάγεται τον κόμβο που δέχεται στήριξη.\n');
    while KS(i)~=1:NN           %Έλεγχος δεδομένων που εισάχθηκαν
    KS(i) = input ('Παρακαλώ εισάγετε αριθμό κόμβων που έχει το
δικτύωμα.\n');
    end

    disp(['Κόμβος : ' ,num2str(KS(i))])
    KSX(i) = input ('Μπορει να κινηθεί ο κόμβος στον άξονα των
X?[Y=0/N=1]\n');
    while KSX(i)~=0:1
        KSX(i) = input ('Παρακαλώ εισάγετε "0" αν ο κόμβος μπορεί να
κινηθεί στον άξονα των X και "1" αν όχι.\n');
    end

    KSY(i) = input ('Μπορεί να κινηθεί ο κόμβος στον άξονα των
Y?[Y=0/N=1]\n');
     while KSY(i)~=0:1
        KSY(i) = input ('Παρακαλώ εισάγετε "0" αν ο κόμβος μπορεί να
κινηθεί στον άξονα των Y και "1" αν όχι.\n');
    end


end

P4=[KS ; KSX; KSY ];
disp('Στηρίξεις στους κόμβους : ')
disp('Κόμβος     X     Y')
disp((P4'))

                            % 2)ΕΠΙΛΥΣΗ ΔΙΚΤΥΩΜΑΤΟΣ

for i=1:NB
                            %Υπολογισμός ημιτόνου
      z=(TSX(i)-ASX(i))/L(i);
      SIN(i)=cosd(z);
                            %Υπολογισμός συνημίτονου
      z=(TSY(i)-ASY(i))/L(i);
      COS(i)=sind(z);
```

```
                        %Μητρώο συστροφής
      MS=[cosd(z)^2  sind(z)*cosd(z)  -cosd(z)^2  -sind(z)*cosd(z);
         sind(z)*cosd(z)  sind(z)^2  -sind(z)*cosd(z)  -sind(z)^2;
         -cosd(z)^2  -sind(z)*cosd(z)  cosd(z)^2  sind(z)*cosd(z);
         -sind(z)*cosd(z)  -sind(z)^2  sind(z)*cosd(z)  sind(z)^2;];

      I(i)=E*D(i)/L(i);

    Kel(1,1)=I(i)*cosd(z);
    Kel(1,2)=I(i)*sind(z)*cosd(z);
    Kel(1,3)=I(i)*(-cosd(z)^2);
    Kel(1,4)=I(i)*(-sind(z)*cosd(z));
    Kel(2,1)=Kel(1,2);
    Kel(2,2)=I(i)*sind(z)^2;
    Kel(2,3)=I(i)*(-sind(z)*cosd(z));
    Kel(2,4)=I(i)*(-sind(z)^2);
    Kel(3,1)=Kel(1,3);
    Kel(3,2)=Kel(2,3);
    Kel(3,3)=I(i)*cosd(z)^2;
    Kel(3,4)=I(i)*sind(z)*cosd(z);
    Kel(4,1)=Kel(1,4);
    Kel(4,2)=Kel(2,4);
    Kel(4,3)=Kel(3,4);
    Kel(4,4)=I(i)*sind(z)^2;

    KEL{i}=Kel;
end
KEL;          %Μητρώο δυσκαμψίας της κάθε ράβδου
SIN;          %Βοηθητικός πίνακας με ημίτονα
COS;          %Βοηθητικός πίνακας με συνημίτονα
I;            %Βοηθητικός πίνακας που υπολογίζει το E*D/L της κάθε
ράβδου


                    %ΔΙΑΓΩΝΙΟΣ ΟΛΙΚΟΥ ΜΗΤΡΩΟΥ ΔΥΣΚΑΜΨΙΑΣ



NODES=NN*2;
OMD(NODES,NODES)=0;
ARXDIAGX(NN,NB)=0;
TELDIAGX(NN,NB)=0;
ARXDIAGY(NN,NB)=0;
TELDIAGY(NN,NB)=0;
HELPDIAGX(NN,NN)=0;
HELPDIAGY(NN,NN)=0;


for i=1:NN

     AR{i}=find(AKR==i);     %Βρίσκω ο κάθε κόμβος σε ποιες ράβδους
είναι αρχικός,
                               %δηλαδή μου δείχνει σε ποιές ράβδους
είναι αρχικός ο κόμβος που τρέχει στην for
     TR{i}=find(TKR==i);     %Βρίσκω ο κάθε κόμβος σε ποιες ράβδους
είναι τελικός
```

```
                                %δηλαδή μου δείχνει σε ποιές ράβδους
είναι τελικός ο κόμβος που τρέχει στην for
end


                    %Παίρνω αρχικούς κόμβους για F(x) ως προς το
q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i}) %Ψάχνει αν η ράβδος j ανήκει στον AK{i}

            ARXDIAGX(i,j)=KEL{j}(1,1); %Αν ανήκει τότε παίρνει από το
KEL της ράβδου το στοιχείο που επηρεάζει τον αρχικό κόμβο για F(i)(x)
ως προς q(i)(x) ,δηλαδή το (1,1)
        else
            ARXDIAGX(i,j)=0;
        end
    end
end

                      %Παίρνω τελικούς κόμβους για F(x) ως προς το
q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i}) %Ψάχνει αν η ράβδος j ανήκει στον TK{i}

            TELDIAGX(i,j)=KEL{j}(3,3); %Αν ανήκει τότε παίρνει από το
KEL της ράβδου το στοιχείο που επηρεάζει τον τελικό κόμβο για F(i)(x)
ως προς q(i)(x) ,δηλαδή το (3,3)
        else
            TELDIAGX(i,j)=0;
        end
    end
end

                     %Παίρνω αρχικούς κόμβους για F(y) ως προς το
q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})     %Ψάχνει αν η ράβδος j ανήκει στον
AK{i}

            ARXDIAGY(i,j)=KEL{j}(2,2);  %Αν ανήκει τότε παίρνει από
το KEL της ράβδου το στοιχείο που επηρεάζει τον αρχικό κόμβο για
F(i)(y) ως προς q(i)(y) ,δηλαδή το (2,2)
        else
            ARXDIAGY(i,j)=0;
        end
    end
end

                     %Παίρνω τελικούς κόμβους για F(y) ως προς το
q(y)
for i=1:NN
    for j=1:NB
```

```
        if ismember(j,TR{i})    %Ψάχνει αν η ράβδος j ανήκει στον
TK{i}

            TELDIAGY(i,j)=KEL{j}(4,4);  %Αν ανήκει τότε παίρνει από
το KEL της ράβδου το στοιχείο που επηρεάζει τον τελικό κόμβο για
F(i)(y) ως προς q(i)(y) ,δηλαδή το (4,4)
        else
            TELDIAGY(i,j)=0;
        end
    end
end

                %Άθροισμα αρχικών και τελικών κόμβων για F(x)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))  %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i


            HELPDIAGX(i,j)=sum(ARXDIAGX(i,:),2) +
sum(TELDIAGX(i,:),2);      %Το sum(x,1) προσθέτει στήλες ενω το
sum(x,2) γραμμές

        elseif ismember(j,TKR(TR{i}))   %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i

            HELPDIAGX(i,j)=sum(ARXDIAGX(i,:),2) +
sum(TELDIAGX(i,:),2);



        end
    end
end

                %Άθροισμα αρχικών και τελικών κόμβων για F(y)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))  %Ψάχνει αν η ράβδος j είναι η
ράβδος που έχει αρχικό κόμβο τον i


            HELPDIAGY(i,j)=sum(ARXDIAGY(i,:),2) +
sum(TELDIAGY(i,:),2);

        elseif ismember(j,TKR(TR{i}))   %Ψάχνει αν η ράβδος j ειναι η
ράβδος που έχει αρχικό κόμβο τον i

            HELPDIAGY(i,j)=sum(ARXDIAGY(i,:),2) +
sum(TELDIAGY(i,:),2);



        end
    end
end
```

```
                %Συμπλήρωμα της διαγωνίου

for i=1:NODES

    div=(i+1)/2;           %Προσθέτω στο i +1 και έπειτα το διαιρώ με
2 έτσι ώστε να μπορεί να πέρνει στοιχεία απο τους πίνακες
HELPDIAGX,HELPDIAGY που ειναι 5x5 και επειτα τα τοποθετώ στον OMD που
είναι 10x10
    v=floor(div);              %floor=στρογγυλοποίηση προς τα κατώ

    ypol=mod((i+1),2);          %mod=υπόλοιπο


    if ypol==0

        OMD(i,i)=HELPDIAGX(v,v);

    elseif ypol==1

        OMD(i,i)=HELPDIAGY(v,v);

    end
end


               %ΣΤΟΙΧΕΙΑ ΜΕ ΑΘΡΟΙΣΜΑ ΔΙΠΛΑ ΑΠΟ ΤΗΝ ΔΙΑΓΩΝΙΟ

ARXUPX(NN,NB)=0;
TELUPX(NN,NB)=0;
ARXDOWNY(NN,NB)=0;
TELDOWNY(NN,NB)=0;
HELPKELX(NN,NN)=0;
HELPKELY(NN,NN)=0;

                       %Παίρνω αρχικούς κόμβους για F(x) ως προς q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})

            ARXUPX(i,j)=KEL{j}(1,2);
        else
            ARXUPX(i,j)=0;
        end
    end
end

                          %Παίρνω τελικούς κόμβους για F(x)ως προς
q(y)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i})

            TELUPX(i,j)=KEL{j}(3,4);
```

```
        else
            TELUPX(i,j)=0;
        end
    end
end


                        %Παίρνω αρχικούς κόμβους για F(y) ως προς q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,AR{i})

            ARXDOWNY(i,j)=KEL{j}(2,1);
        else
            ARXDOWNY(i,j)=0;
        end
    end
end

                        %Παίρνω τελικούς κόμβους για F(y) ως προς q(x)
for i=1:NN
    for j=1:NB

        if ismember(j,TR{i})

            TELDOWNY(i,j)=KEL{j}(4,3);
        else
            TELDOWNY(i,j)=0;
        end
    end
end

                    %Άθροισμα αρχικών και τελικών κόμβων για F(x)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))


            HELPKELX(i,j)=sum(ARXUPX(i,:),2) + sum(TELUPX(i,:),2);

        elseif ismember(j,TKR(TR{i}))

            HELPKELX(i,j)=sum(ARXUPX(i,:),2) + sum(TELUPX(i,:),2);


        end
    end
end

                    %Άθροισμα αρχικών και τελικών κόμβων για F(y)
for i=1:NN

    for j=1:NN
        if ismember(j,AKR(AR{i}))
```

```
            HELPKELY(i,j)=sum(ARXDOWNY(i,:),2) +
sum(TELDOWNY(i,:),2);

        elseif ismember(j,TKR(TR{i}))

            HELPKELY(i,j)=sum(ARXDOWNY(i,:),2) +
sum(TELDOWNY(i,:),2);


        end
    end
end

             %Συμπλήρωμα στοιχείων δίπλα από τα F(x),F(y)

for i=1:NODES

    div=(i+1)/2;
    v=floor(div);

    ypol=mod((i+1),2);

    if ypol==0

        OMD(i,i+1)=HELPKELX(v,v);

     elseif ypol==1

        OMD(i,i-1)=HELPKELY(v,v);

    end
end


                %ΣΤΟΙΧΕΙΑ ΕΚΤΟΣ ΔΙΑΓΩΝΙΟΥ

                %Συμπλήρωμα του υπόλοιπου πίνακα

for i=1:NODES

    div=(i+1)/2;
    v=floor(div);

    ypol=mod((i+1),2);

                  %Συμπληρώνει γραμμές F(x)
   if ypol==0
     for j=1:NB
         if v==P2(2,j)

            OMD(i,P2(3,j)*2-1)=KEL{P2(1,j)}(1,3);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(3,j)*2)  =KEL{P2(1,j)}(1,4);    %Συμπληρώνει
στήλες q(y)
```

```
         elseif v==P2(3,j)

            OMD(i,P2(2,j)*2-1)=KEL{P2(1,j)}(3,1);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(2,j)*2)  =KEL{P2(1,j)}(3,2);    %Συμπληρώνει
στήλες q(y)

         end
     end

                        %Συμπληρώνει γραμμές F(y)
   elseif ypol==1
       for j=1:NB
          if v==P2(2,j)

            OMD(i,P2(3,j)*2-1)=KEL{P2(1,j)}(2,3);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(3,j)*2)  =KEL{P2(1,j)}(2,4);    %Συμπληρώνει
στήλες q(y)

          elseif v==P2(3,j)

            OMD(i,P2(2,j)*2-1)=KEL{P2(1,j)}(4,1);    %Συμπληρώνει
στήλες q(x)
            OMD(i,P2(2,j)*2)  =KEL{P2(1,j)}(4,2);    %Συμπληρώνει
στήλες q(y)

          end
       end
    end
end

                  %ΦΟΡΤΙΣΕΙΣ ΣΤΟΥΣ ΚΟΜΒΟΥΣ

DYN(NODES,1)=0;
fortiseis(2,NN)=0;


for j=1:NF
 for i =1:NN
      if ismember(i,KD)
    fortiseis(1,KD(j))=KDX(j);
    fortiseis(2,KD(j))=KDY(j);
      end
 end
end



 for i=1:NODES
    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);

    if ypol==0
        if ismember(v,KD)

         DYN(i)=fortiseis(1,v);
```

```
        end
    elseif ypol==1
        if ismember(v,KD)

         DYN(i)=fortiseis(2,v);

        end
    end
 end

                              %ΣΤΗΡΙΞΕΙΣ ΣΤΟΥΣ ΚΟΜΒΟΥΣ

 STIRIKSEIS(3,NN)=0;
 HOLD=OMD;

for j=1:NS
    for i=1:NN
        if ismember(i,KS)
            STIRIKSEIS(1,KS(j))=KS(j);
            STIRIKSEIS(2,KS(j))=KSX(j);
            STIRIKSEIS(3,KS(j))=KSY(j);
        end
    end
end



for i=1:NODES
    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);
                                          %0 = kinhtai  1 = den kinhtai
    if ypol==0
         if ismember(v,KS)&& STIRIKSEIS(2,v)==1 && STIRIKSEIS(3,v)==1
%'Αρθρωση
            HOLD(i,:)=0;
            HOLD(:,i)=0;
            HOLD(i,i)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==0 &&
STIRIKSEIS(3,v)==1  %Κύλιση στον άξονα x
            HOLD(i+1,:)=0;
            HOLD(:,i+1)=0;
            HOLD(i,i)=OMD(i,i);
            HOLD(i+1,i+1)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==1 &&
STIRIKSEIS(3,v)==0  %Κύλιση στον άξονα y
           HOLD(i,:)=0;
           HOLD(:,i)=0;
           HOLD(i,i)=OMD(i,i);
        end

    elseif ypol==1
         if ismember(v,KS)&& STIRIKSEIS(2,v)==1 && STIRIKSEIS(3,v)==1
%'Αρθρωση
             HOLD(i,:)=0;
```

```
            HOLD(:,i)=0;
            HOLD(i,i)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==0 &&
STIRIKSEIS(3,v)==1  %Κύλιση στον άξονα x
            HOLD(i,:)=0;
            HOLD(:,i)=0;
            HOLD(i,i)=1;

        elseif ismember(v,KS)&& STIRIKSEIS(2,v)==1 &&
STIRIKSEIS(3,v)==0  %Κύλιση τον άξονα y
            HOLD(i-1,:)=0;
            HOLD(:,i-1)=0;
            HOLD(i,i)=OMD(i,i);
            HOLD(i-1,i-1)=1;
        end
    end
end

Q=HOLD\DYN;
%Q=Q*-1;

for i=1:NODES

    div=(i+1)/2;
    v=floor(div);
    ypol=mod((i+1),2);

    if ypol==0
        QX(v)=Q(i);

    elseif ypol==1
        QY(v)=Q(i);
    end
end

for i=1:NB
    for j=1:NN
        if P2(2,i)==j
            QARXX(i)=QX(j);
            QARXY(i)=QY(j);

        elseif P2(3,i)==j
            QTELX(i)=QX(j);
            QTELY(i)=QY(j);

        end
    end
end


for i=1:NB
    HELPTASEIS{i}(1,1)=QARXX(i);
    HELPTASEIS{i}(2,1)=QARXY(i);
    HELPTASEIS{i}(3,1)=QTELX(i);
    HELPTASEIS{i}(4,1)=QTELY(i);
end
```

```
for i=1:NB
    TASEIS{i}=(KEL{i}/D(i))*HELPTASEIS{i};
end

for i =1:NODES
    for j=1:NODES

    r1=sum(HOLD(i,:));

    if r1==1
        OMDR(i,j)=OMD(i,j);

    elseif r1~=1
       OMDR(i,j)=0;

    end
    end
end
```

```
%3)Εμφάνιση αποτελεσμάτων και διαγραμμάτων

Q
TASEIS
R=OMDR*Q;


maxx=SX(1);
 for i=1:NN
  if SX(i)>maxx
  maxx=SX(i);
  end
 end

 maxy=SY(1);
 for i=1:NN
     if SY(i)>maxy
         maxy=SY(i);
     end
 end

 minx=SX(1);
 for i=1:NN
     if SX(i)<minx
    minx=SX(i);
     end
 end

 miny=SY(1);
 for i=1:NN
     if SY(i)<miny
         miny=SY(i);
     end
```

```
 end


xdata=[SX(AKR);SX(TKR)];
ydata=[SY(AKR);SY(TKR)];

subplot(2,1,1); patch(xdata,ydata,'r')
title('Αρχικό δικτύωμα')
axis([(minx-10) (maxx+10) (miny-10) (maxy+10)])
for i=1:NS
    if P4(2,i)==1 && P4(3,i)==1
        text(SX(P4(1,i))-1,SY(P4(1,i))-
1,'Άρθρωση','EdgeColor','blue');
    else
        text(SX(P4(1,i))-1,SY(P4(1,i))-
1,'Κύλιση','EdgeColor','blue');
    end
end

for i=1:NF
    if P3(2,i)>0
        text(SX(P3(1,i)),SY(P3(1,i)),'\rightarrowF');
    elseif P3(2,i)<0
        text(SX(P3(1,i)),SY(P3(1,i)),'\leftarrowF');
    elseif P3(3,i)>0
        text(SX(P3(1,i)),SY(P3(1,i)),'\uparrowF');
    elseif P3(3,i)<0
        text(SX(P3(1,i)),SY(P3(1,i)),'\downarrowF');
    end
end


HOPEX=SX-(QX*-1);          % Ο πολλαπλασιασμός με -1 γίνεται για να
HOPEY=SY-(QY*-1);          % απεικονίζεται η θετική φορά προς τα δεξιά

 maxx1=HOPEX(1);
 for i=1:NN
  if HOPEX(i)>maxx1
  maxx1=HOPEX(i);
  end
 end

 maxy1=HOPEY(1);
 for i=1:NN
     if HOPEY(i)>maxy1
         maxy1=HOPEY(i);
     end
 end

 minx1=HOPEX(1);
 for i=1:NN
     if HOPEX(i)<minx1
    minx1=HOPEX(i);
     end
 end

 miny1=HOPEY(1);
 for i=1:NN
     if HOPEY(i)<miny1
```

```
        miny1=HOPEY(i);
    end
end



xdata1=[HOPEX(AKR);HOPEX(TKR)];
ydata1=[HOPEY(AKR);HOPEY(TKR)];

subplot(2,1,2); patch(xdata1,ydata1,'r')
title('Τελικό δικτύωμα')
axis([(minx1-10) (maxx1+10) (miny1-10) (maxy1+10)])
for i=1:NS
    if P4(2,i)==1 && P4(3,i)==1
        text(HOPEX(P4(1,i))-1,HOPEY(P4(1,i))-
1,'Άρθρωση','EdgeColor','blue');
    else
        text(HOPEX(P4(1,i))-1,HOPEY(P4(1,i))-
1,'Κύλιση','EdgeColor','blue');
   end
end

for i=1:NF
    if P3(2,i)>0
        text(HOPEX(P3(1,i)),HOPEY(P3(1,i)),'\rightarrowF');
    elseif P3(2,i)<0
        text(HOPEX(P3(1,i)),HOPEY(P3(1,i)),'\leftarrowF');
    elseif P3(3,i)>0
        text(HOPEX(P3(1,i)),HOPEY(P3(1,i)),'\uparrowF');
    elseif P3(3,i)<0
        text(HOPEX(P3(1,i)),HOPEY(P3(1,i)),'\downarrowF');
    end
end
```