ΣΧΟΛΙΑΣΜΟΣ ΚΩΔΙΚΑ MATLAB ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ 2D ΔΙΚΤΥΩΜΑΤΟΣ ΜΕ ΤΗ ΜΕΘΟΔΟ ΤΩΝ ΠΕΠΕΡΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ

1. Εισαγωγή

Η επίλυση ενός 2D-δικτυώματος με τη Μέθοδο των Πεπερασμένων Στοιχείων (ΜΠΣ, Finite Element Method - FEM) αποτελεί το πρώτο, αλλά ταυτόχρονα πολύ βασικό, βήμα στην περιοχή της Υπολογιστικής Μηχανικής. Όπως φανερώνει ο όρος 'Υπολογιστική', προϋποτίθεται η εφαρμογή κάποιας αριθμητικής μεθόδου, επομένως απαιτείται η σύνταξη ενός κώδικα για την εκτέλεση μίας συγκεκριμένης αριθμητικής διαδικασίας. Στο παρόν κείμενο, σχολιάζεται συνοπτικά ένας κώδικας Matlab για την επίλυση 2D-δικτυωμάτων με τη ΜΠΣ. Πιο συγκεκριμένα, παρουσιάζονται και σχολιάζονται επί μέρους προγραμματιστικά στοιχεία, όπως εντολές και συντακτικές διατυπώσεις. Προφανώς, είναι δυνατόν να χρησιμοποιηθεί οποιοδήποτε άλλο υπολογιστικό ή προγραμματιστικό περιβάλλον (π.χ. Mathematica, MathCad, Fortan, C++, κοκ). Ο προαναφερθέν κώδικας επισυνάπτεται σε Παράρτημα στο τέλος του κειμένου, ενώ η ηλεκτρονική του μορφή μπορεί να βρεθεί στην ιστοσελίδα http://users.ntua.gr/cprovat . Αποτελείται δε, από δύο αρχεία:

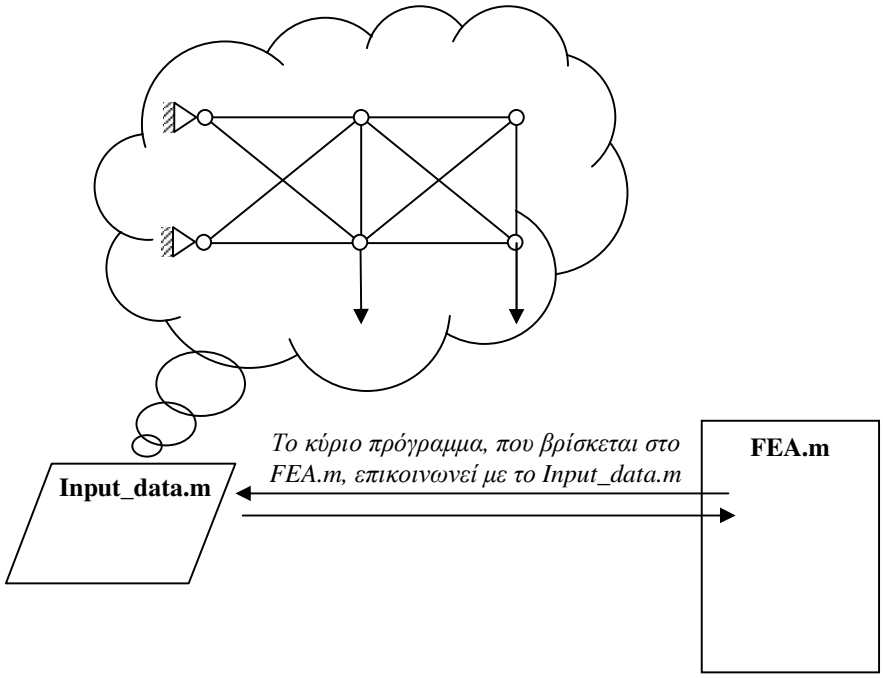
Input_data.m: περιγραφή της, προς ανάλυση, κατασκευής
FEA.m: ανάλυση της κατασκευής

Ακολουθεί η παρουσίαση της δομής του κώδικα καθώς και κάθε αρχείου χωριστά.

2. Η δομή του κώδικα

Ο αντικειμενικός σκοπός είναι η σύνταξη ενός κώδικα επίλυσης 2D δικτυωμάτων με τη ΜΠΣ. Προφανώς, κάθε δικτύωμα αποτελεί μία ξεχωριστή κατασκευή, με τη δική της γεωμετρία, τοπολογία, υλικό κατασκευής, στήριξη και φόρτιση. Ωστόσο, η μεθοδολογία επίλυσης (ΜΠΣ) είναι ακριβώς η ίδια για όλα τα δικτυώματα (και γενικότερα για όλες τις κατασκευές). Επομένως, για λόγους προγραμματιστικής τάξεως, δέον είναι η κατασκευή να περιγράφεται σε ένα αρχείο (έστω Input_data.m), το οποίο ο χρήστης θα αλλάζει κατά περίπτωση, ενώ η ανάλυση της κατασκευής να πραγματοποιείται σε ένα άλλο αρχείο (έστω FEA.m), στο οποίο ο χρήστης δεν θα χρειάζεται να επεμβαίνει. Τα ονόματα των αρχείων, αν και αυθαίρετα, καλόν είναι να προϊδεάζουν για την εργασία στην οποία ανταποκρίνονται. Δεδομένου ότι το αρχείο FEA.m καλεί το αρχείο Input_data.m από το οποίο αντλεί όλες τις απαραίτητες, για την ανάλυση της κατασκευής, πληροφορίες, προκύπτει ότι το FEA.m περιέχει το κύριο πρόγραμμα (main), ενώ το Input_data.m αποτελεί βοηθητικό αρχείο (στο περιβάλλον Matlab δηλώνεται ως function). Η δομή του προτεινόμενου κώδικα καθώς και η επικοινωνία μεταξύ των FEA.m

και Input_data.m απεικονίζεται στο Σχήμα 1. Τα στοιχεία που πρέπει να περιέχει το Input_data.m, καθώς και ο τρόπος υλοποίησης της ΜΠΣ για δικτυώματα που περιέχεται στο FEA.m, σχολιάζονται εκτενώς στο αρχείο AMKI_2D_truss που βρίσκεται στην ιστοσελίδα http://users.ntua.gr/cprovat .



Σχήμα 1: Η δομή του κώδικα

3. Αρχείο Input_data.m[1]

Το αρχείο Input_data.m αποτελεί βοηθητικό αρχείο και δηλώνεται ως function. Ειδικότερα, η πρώτη γραμμή του συγκεκριμένου αρχείου είναι της μορφής:

```
function [NEL,NN,Area,Coor,Dens,Young,Elem,BC,F1]=Input_data;
```

Δεσμευμένη λέξη, δήλωση της συνάρτησης με όνομα Input_data

Ορίσματα εξόδου από τη συνάρτηση με όνομα Input_data

όνομα συνάρτησης

Παρατηρούμε ότι το όνομα της συνάρτησης και το όνομα του αρχείου είναι το ίδιο, με τη μόνη διαφορά ότι το όνομα του αρχείου έχει την κατάληξη των αρχείων Matlab (δηλαδή .m). Μετά από τη δεσμευμένη λέξη **function**, ακολουθεί ένα σύνολο μεταβλητών (ορίσματα)

[1] Το συνημμένο αρχείο στο Παράρτημα αφορά σε μία τυπική βιβλιογραφική περίπτωση δικτυώματος 10 ράβδων.

εντός παρενθέσεως. Προσοχή απαιτείται στη χρήση του σωστού τύπου παρένθεσης, ο οποίος για τα ορίσματα εξόδου είναι [] και για τα ορίσματα εισόδου (). Οι μεταβλητές σχολιάζονται στο αρχείο AMKI_2D_truss που βρίσκεται στην ιστοσελίδα http://users.ntua.gr/cprovat .
Δύο καθοριστικής σημασίας ποσότητες είναι το πλήθος NEL των πεπερασμένων στοιχείων και το πληθος NN των κόμβων που χρησιμοποιούνται. Δεδομένου ότι από αυτές τις ποσότητες εξαρτάται η διάσταση διαφόρων πινάκων που θα δημιουργηθούν (π.χ. πίνακες **Area**, **Coor**, **Elem** κ.ο.κ.), επιβάλλεται η δήλωση των NEL και NN στην αρχή του αρχείου.
Μετά από τη δήλωση των NEL και NN ακολουθεί η δημιουργία διαφόρων πινάκων που περιγράφουν πλήρως την προς μελέτη κατασκευή. Ας εξετάσουμε τη δήλωση του πίνακα **Coor**, η οποία είναι:

```
Coor = zeros(NN,2);
```

Η εντολή **zeros(N,M)** δημιουργεί έναν πίνακα διάστασης $N \times M$ και ταυτόχρονα αποδίδει τη μηδενική τιμή σε όλα τα στοιχεία του πίνακα. Διευκρινίζεται ότι το ελληνικό ερωτηματικό στο τέλος της γραμμής δεν επιτρέπει την, επί της οθόνης, παρουσίαση του αποτελέσματος της εκτέλεσης της εντολής της συγκεκριμένης γραμμής του προγράμματος. Άρα, με την ανωτέρω εντολή έχουμε δημιουργήσει (δηλώσει) έναν πίνακα με όνομα **Coor** και διάσταση $NN \times 2$, ενώ ταυτόχρονα αποδίδουμε σε όλα τα στοιχεία του πίνακα τη μηδενική τιμή (μηδενισμός πίνακα). Η απόδοση μίας αρχικής τιμής, συνήθως της μηδενικής, σε μία μεταβλητή αποσκοπεί στην αποτροπή προβλημάτων που, υπό συνθήκες, μπορεί να προκύψουν, ανάλογα με το περιβάλλον στο οποίο εργαζόμαστε και τη διαθέσιμη μνήμη του Η/Υ που χρησιμοποιούμε. Με τον ίδιο τρόπο που δημιουργήσαμε τον πίνακα **Coor**, δημιουργούμε τους πίνακες **Elem**, **Area**, **BC** και **F1**. Στο σημείο αυτό, τονίζεται ότι η συμπλήρωση των πινάκων **Coor**, **Elem**, **Area**, **BC** και **F1** με στοιχεία από την προς επίλυση κατασκευή υλοποιείται σε άλλο σημείο του αρχείου Input_data.m.
Η πυκνότητα του υλικού κατασκευής των ράβδων αποτελεί μία απαραίτητη πληροφορία. Η σχετική δήλωση είναι:

```
Dens = 0.1 * ones(NEL,1);
```

Η εντολή **ones(N,M)** δημιουργεί έναν πίνακα διάστασης $N \times M$ και αποδίδει τη μοναδιαία τιμή σε όλα τα στοιχεία του πίνακα. Πολλαπλασιάζοντας, δε, επί 0.1 με τον τρόπο που φαίνεται στην ανωτέρω σύνταξη, κάθε στοιχείο του πίνακα που δημιούργησε η εντολή **ones** πολλαπλασιάζεται επί 0.1. Ως αποτέλεσμα, διαμορφώνεται ο πίνακας **Dens** διάστασης $NEL \times 1$ (πίνακας-στήλη), όλα τα στοιχεία του οποίου είναι ίσα με 0.1. Η ανωτέρω σύνταξη ισοδυναμεί με την ακόλουθη:

```
Dens(1,1) = 0.1;
Dens(2,1) = 0.1;
..............
Dens(NEL,1) = 0.1;
```

Υπενθυμίζεται ότι εάν υπάρχει ελληνικό ερωτηματικό στο τέλος μίας γραμμής, τότε το αποτέλεσμα της εκτέλεσης της εντολής που αναγράφεται στη συγκεκριμένη γραμμή δεν παρουσιάζεται στην οθόνη. Όπως ακριβώς δημιουργήθηκε ο πίνακας **Dens**, δημιουργείται και ο πίνακας **Young** που περιέχει την πληροφορία για το μέτρο ελαστικότητας του υλικού κατασκευής των ράβδων. Διευκρινίζεται ότι στην περίπτωση όπου για κάθε ράβδο χρησιμοποιείται διαφορετικό υλικό, τότε θα πρέπει να ακολουθηθεί η πορεία δήλωσης του πίνακα **Coor**.
Το επόμενο βήμα αφορά στην συμπλήρωση των πινάκων **Coor**, **Elem**, **Area**, **BC** και **F1** με στοιχεία από την προς επίλυση κατασκευή. Το βήμα αυτό δεν εμφανίζει κάποια ιδιαιτερότητα. Αρκεί η πληκτρολόγηση των τιμών στα αντίστοιχα πεδία των πινάκων. Ως παράδειγμα αναφέρεται η απόδοση τιμών στις διατομές των ράβδων (συμπλήρωση του πίνακα **Area**) με την ακόλουθη (‘κατακόρυφη’) παράθεση δεδομένων:

```
Area = [23.20000;
        30.52200;
         0.10000;
        21.03700;
         7.45720;
        15.22300;
         0.10000;
         0.55135;
         0.10000;
        21.52800];
```

Η ύπαρξη ελληνικού ερωτηματικού μέσα σε έναν πίνακα σημαίνει ‘αλλαγή γραμμής πίνακα’. Η ανωτέρω σύνταξη σημαίνει ότι:

```
Area( 1,1) = 23.20000
Area( 2,1) = 30.52200
……….
Area(10,1) = 21.52800
```

Ισοδύναμα, είναι δυνατόν να χρησιμοποιηθεί η εξής (‘οριζόντια’) παράθεση δεδομένων:

```
Area = [23.20000; 30.52200;  0.10000; 21.03700; 7.45720; 15.22300; 0.10000;
         0.55135;  0.10000; 21.52800];
```

Η προτίμηση σχετικά με την ‘οριζόντια’ ή την ‘κατακόρυφη’ παράθεση δεδομένων αποτελεί προσωπική επιλογή του προγραμματιστή, εξυπηρετεί κυρίως εποπτικούς λόγους και δεν επηρεάζει δίολου την υπολογιστική διαδικασία. Με αντίστοιχο τρόπο ενημερώνονται οι πίνακες **Coor**, **Elem**, **BC** και **F1**.
Για τη συμπλήρωση του πίνακα **BC**, χρησιμοποιούνται μόνον δύο τιμές: 0 ή 1. Με 0 δηλώνεται ένας μη-δεσμευμένος βαθμός ελευθερίας (ΒΕ) και με 1 δηλώνεται ένας δεσμευμένος ΒΕ. Οι δεσμευμένοι ΒΕ, δηλαδή οι, σχετιζόμενοι με στηρίξεις, ΒΕ, είναι συνήθως λιγότεροι σε πλήθος από τους μη-δεσμευμένους. Ως εκ τούτου, ο σχηματισμός του πίνακα **BC** υλοποιείται εύκολα σε δύο βήματα:

- δημιουργία ενός πίνακας **BC** με μηδενικά στοιχεία (0) και
- κατόπιν αντικατάσταση των 0 με 1 μόνο για εκείνα τα στοιχεία του πίνακα που αντιστοιχούν σε δεσμευμένους ΒΕ.

Με παρόμοιο τρόπο σχηματίζεται και ο πίνακας **F1**. Πιο συγκεκριμένα:

- αρχικά κατασκευάζεται ένας πίνακας με μηδενικά στοιχεία και
- κατόπιν αντικαθίστανται τα μηδενικά στοιχεία με τις αντίστοιχες μη-μηδενικές συνιστώσες των εξωτερικά επιβαλλομένων κομβικών δυνάμεων.

Σχετικά με τις δυνάμεις στήριξης ισχύουν τα εξής:

- ως προς την κατασκευή, αποτελούν εξωτερικά ασκούμενες δυνάμεις, άρα πρέπει να εμφανίζονται στον πίνακα **F1**
- είναι, αρχικώς, άγνωστες, άρα ο πίνακας **F1**, σε αντίθεση με τους πίνακες **Coor**, **Elem**, **Area** και **BC**, δεν είναι δυνατόν να συμπληρωθεί πλήρως από απλή εποπτεία της, προς επίλυση, κατασκευής

Εκ πρώτης όψεως, φαίνεται ότι υπάρχει πρόβλημα με τη συμπλήρωση του πίνακα **F1**, κάτι που τελικά δεν ισχύει, όπως αποδεικνύεται στην επόμενη παράγραφο.

Η εφαρμογή της ΜΠΣ οδηγεί στην κατάστρωση του ακόλουθου γραμμικού συστήματος εξισώσεων:

$$[K]_{glob,free\_dofs} \cdot \{U\}_{free\_dofs} = \{F\}_{free\_dofs}$$

Η λύση του συστήματος ισούται με:

$$\{U\}_{free\_dofs} = \left([K]_{glob,free\_dofs}\right)^{-1} \cdot \{F\}_{free\_dofs}$$

όπου:

$[K]_{glob}$	:	το καθολικό μητρώο δυσκαμψίας της κατασκευής
$\{U\}$	:	διάνυσμα όλων των κομβικών μετατοπίσεων της κατασκευής
$\{F\}$	:	διάνυσμα όλων των εξωτερικών δυνάμεων που ασκούνται στην κατασκευή
$[K]_{glob,free\_dofs}$	:	το τμήμα του $[K]_{glob}$ που αντιστοιχεί σε μη-δεσμευμένους ΒΕ
$\{U\}_{free\_dofs}$	:	το τμήμα του $\{U\}$ που αντιστοιχεί σε μη-δεσμευμένους ΒΕ
$\{F\}_{free\_dofs}$	:	το τμήμα του $\{F\}$ που αντιστοιχεί σε μη-δεσμευμένους ΒΕ

Είναι προφανές ότι στην επίλυση του ανωτέρω συστήματος συμμετέχουν ΜΟΝΟΝ εκείνες οι συνιστώσες δύναμης που σχετίζονται με μη-δεσμευμένους ΒΕ. *Συνεπώς, οι δυνάμεις στήριξης, ως σχετιζόμενες με δεσμευμένους ΒΕ, ΔΕΝ συμμετέχουν στην επίλυση του συστήματος*, άρα, αρχικώς, είναι δυνατόν να θεωρηθούν ΑΥΘΑΙΡΕΤΑ ως μηδενικές. Μετά την επίλυση του συστήματος, οι δυνάμεις στήριξης υπολογίζονται ως εξής:

$$\{F\}_{fixed\_dofs} = [K]_{glob,fixed\_dofs} \cdot \{U\}$$

όπου:

$\{F\}_{fixed\_dofs}$: οι δυνάμεις στήριξης

$[K]_{glob,fixed\_dofs}$: το τμήμα του $[K]_{glob}$ που σχετίζεται με δεσμευμένους ΒΕ

$\{U\}$: το διάνυσμα των κομβικών μετατοπίσεων

Διευκρινίζεται ότι ισχύει:

$$\{U\} = \{U\}_{free\_dofs} \cup \{U\}_{fixed\_dofs}$$

και

$$\{U\}_{fixed\_dofs} = \{0\}$$

4. Αρχείο FEA.m

Στο αρχείο FEA.m υλοποιείται η ΜΠΣ. Οι πρώτες γραμμές, δεδομένου ότι ξεκινούν με τον χαρακτήρα **%**, αποτελούν σχόλια και όχι εκτελέσιμες γραμμές κώδικα. Σε αυτές τις γραμμές αναφέρεται η ονομασία και το περιεχόμενο των μεταβλητών που χρησιμοποιούνται. Η πρώτη εκτελέσιμη εντολή αφορά στη διαγραφή από τη μνήμη (**clear all**) όλων των μεταβλητών και από την οθόνη (**clc**) όλων των μηνυμάτων που ενδεχομένως έχουν προκύψει λόγω προηγηθείσας εκτέλεσης κάποιου προγράμματος.

```
% clear memory / clear screen
clear all
clc
```

Το επόμενο βήμα είναι η κλήση του αρχείου με τα δεδομένα της, προς ανάλυση, κατασκευής, δηλαδή η κλήση του αρχείου Input_data.m.

```
% Call file Input_data.m
[NEL,NN,Area,Coor,Dens,Young,Elem,BC,F1]=Input_data;
```

Παρατηρούμε ότι η γραμμή κλήσης προκύπτει από την πρώτη γραμμή του αρχείου Input_data.m με διαγραφή της δεσμευμένης λέξεως **function**. Με τον τρόπο αυτό εξασφαλίζεται η, πρωτίστης σημασίας, επικοινωνία μεταξύ των αρχείων FEA.m (καλούν αρχείο) και Input_data.m (καλούμενο αρχείο). Η ορθή επικοινωνία εξασφαλίζεται όταν το πλήθος των ορισμάτων και η σειρά με την οποία αυτά αναγράφονται, είναι η ίδια τόσο στο καλούν αρχείο όσο και στο καλούμενο αρχείο. Σε διαφορετική περίπτωση, κάτι που συχνά

συμβαίνει στην πράξη λόγω αβλεψίας του προγραμματιστή κατά τη φάση σύνταξης του κώδικα, υπάρχουν οι εξής δυνατότητες:

- Κατά την εκτέλεση του προγράμματος εμφανίζεται κάποιο σφάλμα. Πρόκειται για μία εκνευριστική μεν αλλά καλή εξέλιξη διότι ο προγραμματιστής πληροφορείται σχετικά, μέσω κάποιου μηνύματος, για την πηγή του σφάλματος και έχει τη δυνατότητα διορθωτικής επέμβασης.
- Το πρόγραμμα εκτελείται χωρίς σφάλματα. Πρόκειται για μία ανεπιθύμητη περίπτωση διότι το τελικό αποτελέσμα είναι λογιστικά ορθό αλλά ουσιαστικά εσφαλμένο. Όταν δε, ένας κώδικας περιέχει μεγάλο πλήθος υπορουτινών / συναρτήσεων, ο εντοπισμός και αποκατάσταση μίας επικοινωνίας μεταξύ δύο τμημάτων του κώδικα, η οποία, αν και εσφαλμένη, δεν προκαλεί διακοπή κατά την εκτέλεση του κώδικα, αποτελεί μία δύσκολη, επίπονη και χρονοβόρο διαδικασία.

Μετά την κλήση του αρχείου Input_data.m ακολουθεί η δήλωση των χρησιμοποιούμενων πινάκων. Η διάσταση αυτών των πινάκων εξαρτάται από τις ποσότητες NEL και NN, οι οποίες δηλώνονται στο αρχείου Input_data.m, επομένως η κλήση του εν λόγω αρχείου πρέπει να προηγηθεί της δηλώσεων των πινάκων. Έτσι, ισχύει:

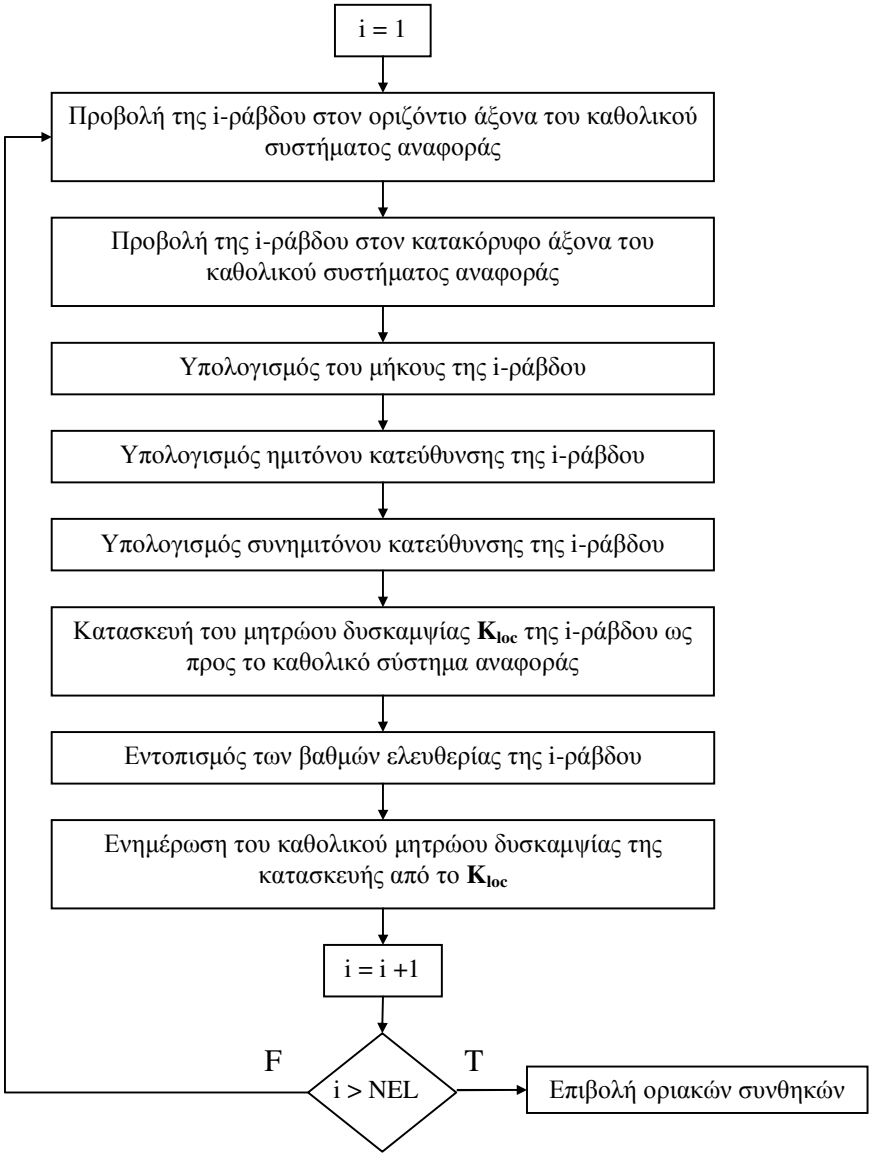
```
% Matrix definition / initialization
         L = zeros(NEL,1);
     Kglob = sparse(2*NN,2*NN);
      Kloc = zeros(4,4);
        U1 = zeros(2*NN,1);
       DUX = zeros(2*NN,1);
       DUY = zeros(2*NN,1);
   Strain1 = zeros(NEL,1);
   Stress1 = zeros(NEL,1);
 free_dofs = [];
fixed_dofs = [];
```

Στις παραπάνω δηλώσεις παρατηρούμε τα εξής:

- Χρήση της εντολής **zeros** (η εντολή αυτή έχει ήδη σχολιασθεί)
- Χρήση της εντολής **sparse**.
  Έστω ο πίνακας $\left[K\right]_{glob,free\_dofs}$ διάστασης $N_{free\_dofs} \times N_{free\_dofs}$. Προφανώς, αυτός διαθέτει $N_{free\_dofs} \times N_{free\_dofs}$ στοιχεία. Με την εντολή **sparse** δεν αποθηκεύεται ολόκληρος ο πίνακας αλλά μόνον τα μη-μηδενικά στοιχεία του καθώς και οι θέσεις αυτών των στοιχείων. Υπάρχουν ειδικές τεχνικές αριθμητικής ανάλυσης για την εκτέλεση μητρωϊκών διαδικασιών (π.χ. αντιστροφή ενός πίνακα), στις οποίες αξιοποιούνται μόνον τα μη-μηδενικά στοιχεία, μειώνοντας τον απαιτούμενο χρόνο υπολογισμού. Εκ κατασκευής, ο πίνακας $\left[K\right]_{glob,free\_dofs}$ διαθέτει πολλά μηδενικά στοιχεία, οπότε η **sparse** δήλωσή του οδηγεί σε σημαντικό υπολογιστικό κέρδος.

- Δήλωση πίνακα ως []
  Η αντίστοιχη μεταβλητή θα χρησιμοποιηθεί ως πίνακας, με δυνατότητα ορισμού ή/και μεταβολής της διάστασης του πίνακα κατά την εκτέλεση του κώδικα.

Μετά τη δήλωση των πινάκων, ακολουθεί η υλοποίηση της ΜΠΣ (περισσότερες λεπτομέρειες στο αρχείο AMKI_2D_truss που βρίσκεται στην ιστοσελίδα http://users.ntua.gr/cprovat). Το διάγραμμα ροής της σύνθεσης του καθολικού μητρώου δυσκαμψίας της κατασκευής της διαδικασίας φαίνεται στο Σχήμα 2.



Σχήμα 2: Σύνθεση του καθολικού μητρώου δυσκαμψίας της κατασκευής

Στη διαδικασία αυτή, ενδιαφέρον παρουσιάζει η ενημέρωση του καθολικού μητρώου δυσκαμψίας K_{glob} της κατασκευής από το μητρώο δυσκαμψίας K_{loc} κάθε ράβδου. Εάν ως **edof** ορισθεί ένας πίνακας που περιέχει τους ΒΕ της εκάστοτε εξεταζομένης ράβδου, δηλαδή:

```
edof=[2*Elem(i,1)-1; 2*Elem(i,1); 2*Elem(i,2)-1; 2*Elem(i,2)];
```

τότε η αντίστοιχη σύνταξη για την ενημέρωση του K_{glob} είναι:

```
Kglob(edof,edof) = Kglob(edof,edof) + Kloc;
```

Πρόκειται για μία εξαιρετικά εύκολη και ταυτόχρονα συμπυκωμένη σύνταξη, η οποία μεταφράζεται ως εξής:

Πρόσθεσε τον πίνακα K_{loc} *στον υποπίνακα* $K_{glob}\left(edof, edof\right)$ *του πίνακα* K_{glob} *.*

Εάν, για λόγους ευκολίας στη γραφή, θεωρήσουμε ότι:

$$edof = [\text{dof1; dof2; dof3; dof4}]$$

τότε ο υποπίνακας $K_{glob}\left(edof, edof\right)$ ισούται με:

$$K_{glob}(edof, edof) = \begin{bmatrix} K_{glob}(dof1, dof1) & K_{glob}(dof1, dof2) & K_{glob}(dof1, dof3) & K_{glob}(dof1, dof4) \\ K_{glob}(dof2, dof1) & K_{glob}(dof2, dof2) & K_{glob}(dof2, dof3) & K_{glob}(dof2, dof4) \\ K_{glob}(dof3, dof1) & K_{glob}(dof3, dof2) & K_{glob}(dof3, dof3) & K_{glob}(dof3, dof4) \\ K_{glob}(dof4, dof1) & K_{glob}(dof4, dof2) & K_{glob}(dof4, dof3) & K_{glob}(dof4, dof4) \end{bmatrix}$$

Μετά τη σύνθεση του καθολικού μητρώου δυσκαμψίας K_{glob} της κατασκευής, ακολουθεί η απόσπαση από αυτό εκείνου του τμήματος που σχετίζεται με μη-δεσμευμένους ΒΕ, οι οποίοι και πρέπει να εντοπισθούν μέσα από τον πίνακα **BC**. Η εντολή **find** αναζητεί ένα συγκεκριμένο στοιχείο μέσα σε έναν πίνακα. Ως παράδειγμα, με τη σύνταξη

```
data_1 = find(Table1==13);
```

αναζητούμε όλα τα στοιχεία του πίνακα Table1 τα οποία είναι ίσα (προσοχή! ==) με 13. Επομένως, είναι δυνατόν να χρησιμοποιήσουμε την εντολή **find** προκειμένου να εντοπίσουμε όλα τα 0 του πίνακα **BC**, τα οποία αντιστοιχούν σε μη-δεσμευμένους ΒΕ. Ωστόσο, η συγκεκριμένη εντολή εφαρμόζεται σε έναν πίνακα-στήλη. Επομένως, θα πρέπει να μετατρέψουμε τον πίνακα **BC** σε μία ισοδύναμη μορφή πίνακα-στήλη. Αυτό επιτυγχάνεται με την εντολή **reshape**. Έστω ότι ο πίνακας **Table1** είναι διάστασης 4×2. Με τη σύνταξη:

```
Table1 = reshape(Table1,8,1);
```

αλλάζουμε τη διάσταση του πίνακα σε 8×1. Το πρώτο σημείο που πρέπει να προσεχθεί ιδιαίτερα είναι το γεγονός ότι το γινόμενο 4×2 της αρχικής μορφής πρέπει να ισούται με το γινόμενο 8×1 της τελικής μορφής. Σε διαφορετική περίπωση θα εμφανισθεί μήνυμα σφάλματος. Το δεύτερο σημείο που πρέπει να προσεχθεί είναι το γεγονός ότι η αλλαγή στη διάσταση του πίνακα γίνεται κατά την 'κατακόρυφη' έννοια. Πιο συγκεκριμένα, έστω ο πίνακας:

```
Table1 = [1 2; 3 4; 5 6; 7 8];
```

Μία πιο εποπτική σύνταξη του ίδιου πίνακα:

```
Table1 = [ 1 2;
           3 4;
           5 6;
           7 8];
```

Η εντολή

```
reshape(Table1,8,1)
```

δίδει

```
ans =
     1
     3
     5
     7
     2
     4
     6
     8
```

δηλαδή πρώτα τοποθετήθηκαν τα στοιχεία της 1ης στήλης και από κάτω τα στοιχεία της 2ης στήλης ('κατακόρυφη' έννοια). Με βάση τη σύμβαση για την αρίθμηση των ΒΕ (βλ. αρχείο AMKI_2D_truss, σελ.9 & 17), ο πίνακας **BC** είναι της μορφής:

$$BC = \begin{bmatrix} dof_{N_1,x} & dof_{N_1,y} \\ dof_{N_1,x} & dof_{N_2,y} \\ \cdots & \cdots \\ dof_{NN,x} & dof_{NN,y} \end{bmatrix}$$

άρα ο ισοδύναμος πίνακας-στήλη είναι της μορφής:

$$BC = \begin{bmatrix} dof_{N_1,x} \\ dof_{N_1,y} \\ dof_{N_2,x} \\ dof_{N_2,y} \\ \cdots \\ dof_{NN,x} \\ dof_{NN,y} \end{bmatrix}$$

Επομένως, η αλλαγή στη διάσταση του πίνακα πρέπει να γίνει κατά την 'οριζόντια' έννοια, κάτι που επιτυγχάνεται όταν η εντολή **reshape** εφαρμοσθεί στον ανάστροφο (**transpose**) του πίνακα **BC**:

```
BC = reshape(transpose(BC),2*NN,1);
```

Λαμβάνοντας υπ' όψιν όλα τα παραπάνω, η κατάλληλη σύνταξη για τον εντοπισμό τόσο των δεσμευμένων όσο και των μη-δεσμευμένων ΒΕ είναι:

```
BC = reshape(transpose(BC),2*NN,1);
free_dofs  = find(BC==0);
fixed_dofs = find(BC==1);
```

Αλλαγή διάστασης απαιτείται και για τον πίνακα **F1**, διότι η εφαρμογή της ΜΠΣ καταλήγει στο σχηματισμό του γραμμικού συστήματος:

$$[K]_{glob,free\_dofs} \cdot \{U\}_{free\_dofs} = \{F\}_{free\_dofs}$$

όπου η ποσότητα $\{F\}_{free\_dofs}$ είναι πίνακας-στήλη (τμήμα του πίνακα **F1**). Επομένως, πρέπει και ο πίνακας **F1** να αποκτήσει τη μορφή ενός πίνακα-στήλη. Η αντίστοιχη σύνταξη είναι:

```
F1 = reshape(transpose(F1),2*NN,1);
```

Διευκρινίζεται ότι θα ήταν δυνατόν στο αρχείο Input_data.m να συνταχθούν εξ αρχής οι πίνακες **BC** και **F1** ως πίνακες-στήλη. Ωστόσο, με τον τρόπο που έχει επιλεγεί (βλ. αρχείο Input_data.m), ο αύξων αριθμός της γραμμής του πίνακα, είτε του **BC** είτε του **F1**, ταυτίζεται με τον αύξοντα αριθμό του κόμβου στον οποίο επιβάλλονται είτε οι οριακές συνθήκες είτε η εξωτερική φόρτιση, κάτι που γενικά βολεύει.

Το επόμενο βήμα είναι η επίλυση του ανωτέρω γραμμικού συστήματος:

```
U1(free_dofs,1) = Kglob(free_dofs,free_dofs) \ F1(free_dofs,1);
```

Πρόκειται για μία ακόμα συμπυκωμένη γραφή, η οποία ερμηνεύεται ως εξής:

- *από τον πίνακα* K_{glob} *αποσπάται ο υποπίνακας* $[K]_{glob,free\_dofs}$ *που σχετίζεται με όλους τους μη-δεσμευμένους ΒΕ*
- *από τον πίνακα* F_1 *αποσπάται ο υποπίνακας* $\{F\}_{free\_dofs}$ *που σχετίζεται με όλους τους μη-δεσμευμένους ΒΕ*
- *εκτελείται η διαίρεση (left division)* $[K]_{glob,free\_dofs} \backslash \{F_1\}_{free\_dofs}$, και το αποτέλεσμα τοποθετείται στον πίνακα $\{U\}_{free\_dofs}$

Στο σημείο αυτό διευκρινίζεται ότι η 'αριστερή διαίρεση' (left division), αν και δεν ορίζεται από μαθηματικής απόψεως, αποτελεί μία δυνατότητα της Matlab για γρηγορότερη επίλυση ενός γραμμικού συστήματος. Εναλλακτικά, μπορεί να χρησιμοποιηθεί ο 'κλασσικός' δρόμος, δηλαδή:

$$\{U\}_{free\_dofs} = \left([K]_{glob,free\_dofs}\right)^{-1} \cdot \{F\}_{free\_dofs}$$

η διατύπωση του οποίου είναι:

```
U1(free_dofs,1) = inv(Kglob(free_dofs,free_dofs)) / F1(free_dofs,1);
```

Η εντολή **inv(A)** εφαρμόζεται επί ενός τετραγωνικού πίνακα και επιστρέφει τον αντίστροφό του.

Με τον υπολογισμό των, σχετιζομένων με μη-δεσμευμένους ΒΕ, κομβικών μετατοπίσεων $\{U\}_{free\_dofs}$ ολοκληρώνεται η πρώτη φάση της ανάλυσης της κατασκευής. Όσον αφορά στις, σχετιζόμενες με δεσμευμένους ΒΕ, κομβικές μετατοπίσεις, αυτές έχουν ήδη αντιμετωπισθεί σιωπηρώς. Πιο συγκεκριμένα, στη δήλωση/μηδενισμό των πινάκων στην αρχή του FEA.m, ο πίνακας **U1** έχει αποκτήσει μηδενικές τιμές:

```
U1 = zeros(2*NN,1);
```
$\rightarrow \quad \{U\} = \{0\}$

Εξ ορισμού ισχύει:

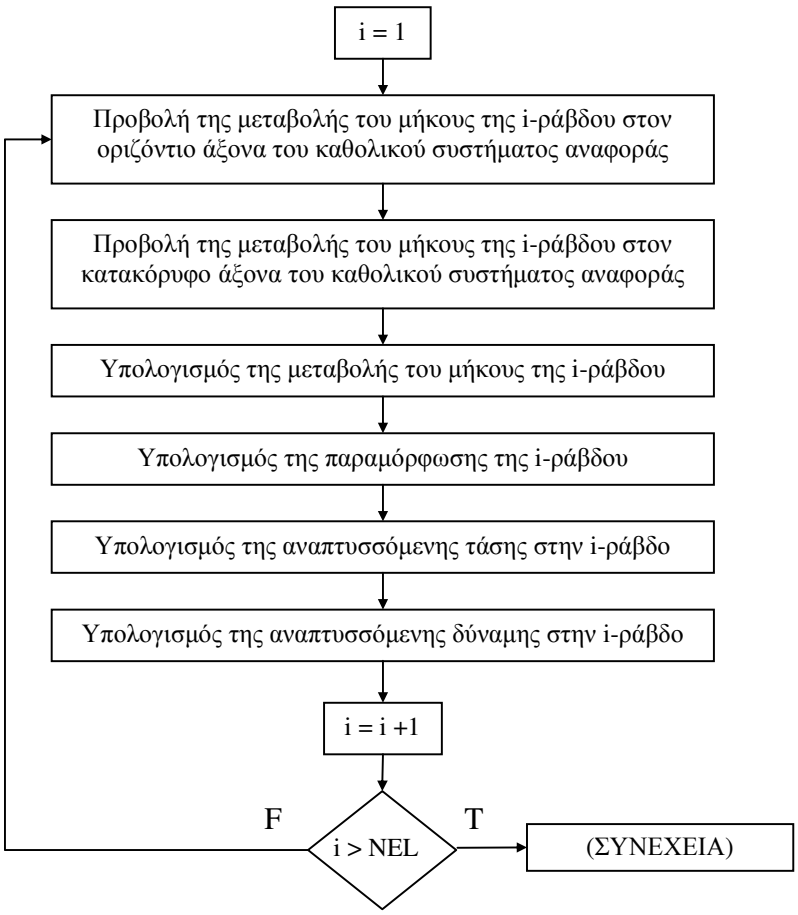
$$\{U\} = \{U\}_{free\_dofs} \cup \{U\}_{fixed\_dofs} \text{ και } \{U\}_{free\_dofs} \cap \{U\}_{fixed\_dofs} = \varnothing$$

Από τον συνδυασμό των ανωτέρω σχέσεων προκύπτει ότι:

$$\{U\}_{fixed\_dofs} = \{0\}$$

Δεδομένου ότι οι, σχετιζόμενες με δεσμευμένους ΒΕ, κομβικές μετατοπίσεις είναι εξ ορισμού μηδενικές, έπεται ότι ο προσδιορισμός του διανύσματος $\{U\}_{fixed\_dofs}$ καλύπτεται από τον αρχικό μηδενισμό του $\{U\}$.

Στη συνέχεια, υπολογίζονται διάφορα χρήσιμα μεγέθη (βλ. αρχείο AMKI_2D_truss, σελ.9 & 17) σύμφωνα με το διάγραμμα ροής που φαίνεται στο Σχήμα 3.



Σχήμα 3: Υπολογισμός χρήσιμων μεγεθών

Στους υπολογισμούς των χρήσιμων μεγεθών χρησιμοποιούνται συμπυκνωμένες εκφράσεις, δεδομένου ότι αυτές αποτελούν το ισχυρό σημείο για το οποίο προτιμάται ο προγραμματισμός σε Matlab. Πιο συγκεκριμένα, γίνεται χρήση του πολλαπλασιασμού τύπου .* και της εντολής **sum**.

Ο πολλαπλασιασμός A.*B (προσοχή! .* → τελεία άστρο, δύο σύμβολα) εφαρμόζεται επί δύο πινάκων A και B της αυτής διάστασης και συνεπάγεται μόνο τον στοιχείο-προς-στοιχείο πολλαπλασιασμό των πινάκων (χωρίς άθροιση των επί μέρους γινομένων). Ως παράδειγμα:

$$\text{εάν } A=\begin{bmatrix}1\\2\\3\end{bmatrix} \text{ και } B=\begin{bmatrix}10\\11\\12\end{bmatrix} \text{ τότε } A.*B=\begin{bmatrix}1\\2\\3\end{bmatrix}\cdot*\begin{bmatrix}10\\11\\12\end{bmatrix}=\begin{bmatrix}1\cdot 10\\2\cdot 11\\3\cdot 12\end{bmatrix}=\begin{bmatrix}10\\22\\33\end{bmatrix}$$

Με αυτόν τον τύπο του πολλαπλασιασμού συμπυκνώνεται ένα βρόχος (loop) σε μία γραμμή, όπως χαρακτηριστικά φαίνεται από την γραμμή:

```
Stress1 = Young .* Strain1;
```

Η εντολή **sum(A)** εφαρμόζεται επί ενός διανύσματος Α και επιστρέφει το άθροισμα των στοιχείων του διανύσματος. Υπενθυμίζεται ότι ένα διάνυσμα ταυτίζεται με έναν πίνακα-στήλη. Με χρήση αυτής της εντολής υπολογίζεται το βάρος της κατασκευής:

```
Weight = sum(Dens .* L .* Area);
```

Η ανωτέρω συμπυκνωμένη έκφραση μεταφράζεται ως εξής:

- πραγματοποιείται πολλαπλασιασμός τύπου .* μεταξύ των πινάκων **Dens** (πίνακας-στήλη με την πυκνότητα του υλικού για κάθε ράβδο), **L** (πίνακας-στήλη με το μήκος κάθε ράβδου) και **Area** (πίνακας-στήλη με τη διατομή κάθε ράβου) και
- κατόπιν αθροίζονται τα επί μέρους γινόμενα.

Τέλος, υπενθυμίζεται ότι εάν Α ένας πίνακας, τότε είναι δυνατόν να αποσπάσουμε ένα οποιοδήποτε τμήμα Asub αυτού χρησιμοποιώντας τη σύνταξη:

```
Asub = A(row_ini:step_row:row_final, col_ini:step_col:col_final,);
```

Η ανωτέρω σύνταξη μεταφράζεται ως εξής:

αποσπάται το τμήμα του πίνακα Α από τη γραμμή row_ini έως και τη γραμμή row_final λαμβάνοντας τις γραμμές ανά step_row και από τη στήλη col_ini έως και τη στήλη col_final λαμβάνοντας τις στήλες ανά step_col

Ως παράδειγμα, έστω ότι

$$A=\begin{bmatrix}11&12&13&14&15&16&17&18&19\\21&22&23&24&25&26&27&28&29\\31&32&33&34&35&36&37&38&39\\41&42&43&44&45&46&47&48&49\end{bmatrix}$$

τότε θα είναι

$$A(2:2:4,1:3:9)=\begin{bmatrix}21&24&27\\41&44&47\end{bmatrix}$$

Επίσης, δυνατόν να παραληφθούν κάποια από τα ορίσματα:

$A(3,:) = [31 \quad 32 \quad 33 \quad 34 \quad 35 \quad 36 \quad 37 \quad 38 \quad 39]$ σημαίνει
απόσπαση της 3ης γραμμής και όλων των στηλών δηλαδή $A(3:1:3,1:1:9)$

Ένα ακόμα πολύ χρήσιμο στοιχείο είναι ο υπολογισμός των δυνάμεων στήριξης:

```
F1(fixed_dofs,1) = Kglob(fixed_dofs,:) * U1;
```

Τέλος, ένας ικανός αλλά όχι και αναγκαίος έλεγχος για την ορθότητα του κώδικα είναι ο υπολογισμός της συνισταμένης δύναμης κατά την οριζόντια και την κατακόρυφη διεύθυνση του καθολικού συστήματος αναφοράς. Δεδομένου ότι η κατασκευή ισορροπεί, θα πρέπει το άθροισμα των, εφαρμοζομένων στην κατασκευή, δυνάμεων κατά την κάθε διεύθυνση να είναι μηδενικό. Το άθροισμα των x-συνιστωσών των εξωτερικά ασκουμένων δυνάμεων προκύπτει εάν αθροισθούν τα στοιχεία του πίνακα-στήλη **F1** ανά δύο, ξεκινώντας από την 1η γραμμή:

```
Fx = sum(F1(1:2:2*NN));
```

Αντίστοιχα, το άθροισμα των y-συνιστωσών των εξωτερικά ασκουμένων δυνάμεων προκύπτει εάν αθροισθούν τα στοιχεία του πίνακα-στήλη **F1** ανά δύο, ξεκινώντας από τη 2η γραμμή:

```
Fy = sum(F1(2:2:2*NN));
```

5. Επίλογος

Από όλα τα παραπάνω, καθίσταται φανερό ότι η χρήση του περιβάλλοντος Matlab παρέχει σημαντικές διευκολύνσεις όσον αφορά στη δυνατότητα συμπυκνωμένης σύνταξης και εκτέλεσης πράξεων μητρωϊκού λογισμού. Δύο επιπρόσθετα χρήσιμα στοιχεία είναι η ανάπτυξη ενός γραφικού περιβάλλοντος για την απεικόνιση της προς ανάλυση κατασκευής καθώς και η η δυνατότητα δημιουργίας ενός ASCII αρχείου στο οποίο θα αποθήκευονται όλα τα στοιχεία (δεδομένα εισαγωγής και αποτελέσματα). Με τη γραφική απεικόνιση, ο χρήστης, πριν από το στάδιο της ανάλυσης, έχει τη δυνατότητα του εποπτικού ελέγχου της κατασκευής που έχει περιγράψει στο αρχείο Input_data.m, με σκοπό τον εντοπισμό τυχόν λαθών που πολύ εύκολα μπορεί να σημειωθούν κατά την πληκτρολόγηση των διαφόρων τιμών, όπως για παράδειγμα κατά την πληκτρολόγηση της περιγραφής των ράβδων (πίνακας **Elem**). Με το ASCII αρχείο, ουσιαστικά αποθηκεύεται ηλεκτρονικά μία κατασκευή μαζί με την ανάλυσή της, οπότε εάν μελλοντικά παραστεί ανάγκη ανασκόπησης της εν λόγω μελέτης, να μην χρειάζεται μία εκ νέου ανάλυση.

ΠΑΡΑΡΤΗΜΑ: Κώδικας Matlab για την επίλυση 2Δ-δικτυωμάτων

Ο συνημμένος κώδικας Matlab αποτελείται από δύο αρχεία:

Αρχείο Input_data.m: *απαραίτητα στοιχεία για την ανάλυση της κατασκευής*

```
function [NEL,NN,Area,Coor,Dens,Young,Elem,BC,F1]=Input_data;
NEL  = 10;              % Number of ELements
NN   = 6;               % Number of Nodes
% Matrix definitions / initializations
Coor = zeros(NN,2);        % nodal coordinates
Elem = zeros(NEL,2);       % element connectivity
Area = zeros(NEL,1);       % cross-sectional area
BC   = zeros(NN,2);        % boundary conditions
F1   = zeros(NN,2);        % externally applied forces
% Material data
Dens  = 0.1 * ones(NEL,1);       % material density
Young = 10000000 * ones(NEL,1);  % material Elastic modulus
% Cross-sectional areas
Area = [23.20000;
        30.52200;
         0.10000;
        21.03700;
         7.45720;
        15.22300;
         0.10000;
         0.55135;
         0.10000;
        21.52800];
% Nodal Coordinates
Coor = [  0    0;
        360    0;
        720    0;
          0  360;
        360  360;
        720  360];
% Element Connectivity
Elem = [1  2;
        4  5;
        2  5;
        1  5;
        4  2;
        2  3;
        5  6;
        3  6;
        2  6;
        5  3];
% Boundary Conditions
% default: all dofs are free - define only the restrained (fixed) dofs
% BC(nn,1)=1 -> the x-translation of the nn node is restrained
% BC(nn,2)=1 -> the y-translation of the nn node is restrained
BC(1,1)=1;
% Externally Applied Forces
% default: all applied forces are zero - define only non-zero components
F1(2,2) = -100000;
```

Αρχείο FEA.m: *ανάλυση της κατασκευής*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FINITE ELEMENT ANALYSIS - 2D Truss structures   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright: DeathValley & C.G. Provatidis, 2005
%            National Technical University of Athens
%            School of Mechanical Engineering
%            Mechanical Design and Control Systems Section
%            Laboratory of Dynamics and Constructions
% Strain1         : vector of all member strains
% Stress1         : vector of all member stresses
% Force1          : vector of all member forces
% fixed_dofs      : restrained dofs
% free_dofs       : non-restrained dofs
% F1              : vector of all nodal forces
% F1(fixed_dofs) : vector of the support forces
% Fx              : the sum of all nodal forces along the x-axis
% Fy              : the sum of all nodal forces along the y-axis
% Weight          : structural weight

% clear memory / clear screen
clear all
clc

% Call file Input_data.m
[NEL,NN,Area,Coor,Dens,Young,Elem,BC,F1]=Input_data;

% Matrix definition / initialization
         L = zeros(NEL,1);
     Kglob = sparse(2*NN,2*NN);
      Kloc = zeros(4,4);
        U1 = zeros(2*NN,1);
       DUX = zeros(2*NN,1);
       DUY = zeros(2*NN,1);
   Strain1 = zeros(NEL,1);
   Stress1 = zeros(NEL,1);
 free_dofs = [];
fixed_dofs = [];

% FEA
for i=1:NEL
 % calculate the length L(i) of the i-th element
    dx = Coor(Elem(i,2),1) - Coor(Elem(i,1),1);
    dy = Coor(Elem(i,2),2) - Coor(Elem(i,1),2);
  L(i) = sqrt(dx*dx + dy*dy);

 % define the orientation (rs=sin(thita), rc=cos(thita)) of the i-th element
 % with respect to the global coordinate system
   rc(i) = dx / L(i);
   rs(i) = dy / L(i);

 % define the local stiffness matrix Kloc of the i-th element
 % with respect to the global coordinate system
  k1 = rc(i) * rc(i);
  k2 = rs(i) * rs(i);
  k3 = rc(i) * rs(i);

  Kloc = Area(i)* Young(i) / L(i) * [ k1  k3 -k1 -k3;
                                      k3  k2 -k3 -k2;
                                     -k1 -k3  k1  k3;
                                     -k3 -k2  k3  k2];

  % define the global numbering of the involved degrees of freedom (dofs)
  edof=[2*Elem(i,1)-1; 2*Elem(i,1); 2*Elem(i,2)-1; 2*Elem(i,2)];

  % update the global stiffness matrix Kg with the stiffness matrix of the i-th element
  Kglob(edof,edof) = Kglob(edof,edof) + Kloc;

end
```

```
% trace free dofs and fixed dofs
        BC = reshape(transpose(BC),2*NN,1);
free_dofs  = find(BC==0);
fixed_dofs = find(BC==1);

% calculate diplacements
F1 = reshape(transpose(F1),2*NN,1);
U1(free_dofs,1) = Kglob(free_dofs,free_dofs) \ F1(free_dofs,1);

% calculate strains, stresses, member forces
DUX = U1(2*Elem(:,2)-1) - U1(2*Elem(:,1)-1);
DUY = U1(2*Elem(:,2)  ) - U1(2*Elem(:,1)  );
Strain1 = 1 ./ L .* (transpose(rc) .* DUX + transpose(rs) .* DUY); % member strain
Stress1 = Young .* Strain1;                                        % member stress

Force1  = Stress1 .* Area;                                         % member force

% calculate support forces
F1(fixed_dofs,1) = Kglob(fixed_dofs,:) * U1;

% calculate weight
Weight = sum(Dens .* L .* Area);

% check: Sum of Fx = Sum of Fy = 0
Fx = sum(F1(1:2:2*NN));
Fy = sum(F1(2:2:2*NN));

% End of program
```