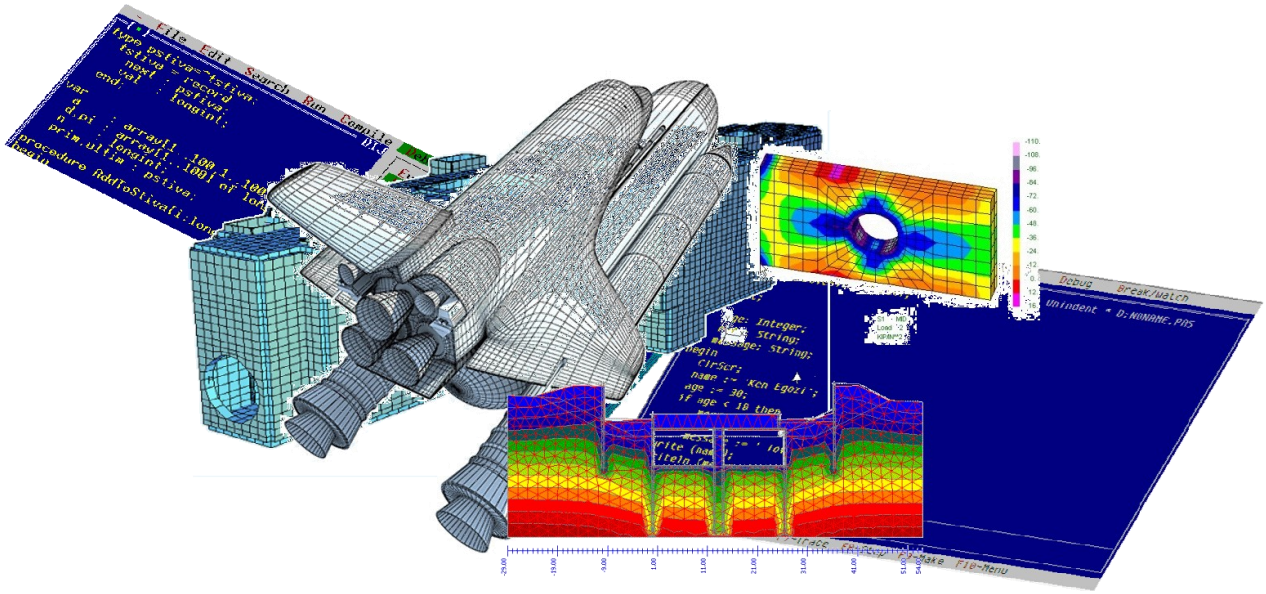
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΣΤΑΤΙΚΗΣ ΚΑΙ ΑΝΤΙΣΕΙΣΜΙΚΩΝ ΕΡΕΥΝΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΡΟΣΘΗΚΗ ΚΩΔΙΚΑ ΣΤΟ FEAP· ΜΗ ΓΡΑΜΜΙΚΕΣ ΕΛΑΣΤΙΚΕΣ ΑΝΑΛΥΣΕΙΣ

ΕΜΜΑΝΟΥΗΛ ΖΕΡΒΑΣ



ΕΠΙΒΛΕΠΩΝ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ
Κ. ΣΠΗΛΙΟΠΟΥΛΟΣ

Αθήνα
Ιανουάριος 2014

Πρόλογος

Με την εργασία αυτή ολοκληρώνεται ο προπτυχιακός κύκλος σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο στη σχολή πολιτικών μηχανικών. Έξι χρόνια σπουδών –ναι, καλά καταλάβατε, άργησα λίγο – ήταν αρκετά για να συνειδητοποιήσω ότι, όσα και να ξέρει κανείς, όση αυτοπεποίθηση και αν νιώθει, τελικά, όσο πιο σίγουρος νιώθει, τόσο λιγότερα ξέρει. Και μπορεί όντως να ξέρει πολλά, αλλά και πάλι λίγα θα είναι διότι αγνοεί το βασικό. Ότι δηλαδή όλες οι γνώσεις του κόσμου δεν είναι τίποτε άλλο από μια σταγόνα στον ωκεανό της σοφίας Εκείνου που έφτιαξε τον κόσμο, έναν κόσμο που μας εμπνέει σαν επιστήμονες, σαν μηχανικούς, σαν ποιητές, σαν ερωτευμένους, ακόμα και σαν τίποτα από όλα αυτά, έναν κόσμο πέρα από κάθε φαντασία που τα μυστήριά του ξεπερνούν ακόμα και την πιο ακραία διανόηση, έναν κόσμο που τελικά μας ξεπερνά και που μας περιβάλλει όχι που τον περιβάλλουμε. Και ως μηχανικός γνωρίζω πολύ καλά ότι για να κατασκευάσεις κάτι, πρέπει να γνωρίζεις ακριβώς όλα τα μέρη που το απαρτίζουν στο βαθμό που αυτό είναι απαραίτητο για να συνεργαστούν κατά το δοκούν. Και μιας και στον κόσμο αυτό βρεθήκαμε και εμείς, είναι εύλογο ότι Εκείνος θα είναι κάπως σαν εμάς και κάτι παραπάνω. Το πιο σωστό θα ήταν βέβαια να πούμε ότι εμείς είμαστε κάτι σαν Αυτόν, για να είμαστε πιο συνεπείς και στην ευλόγως προκύπτουσα ιεραρχία. Επειδή όμως φοβάμαι πως έχω αρχίζει να πλατειάζω επικίνδυνα, ας μείνουμε εδώ.

Ο λόγος που μου άρεσε αυτή η διπλωματική ήταν η σχέση της με τον προγραμματισμό και με αριθμητικές μεθόδους, ήτοι αυτής των πεπερασμένων στοιχείων και μάλιστα, εφαρμοσμένη σε ένα πρόγραμμα ανοιχτού κώδικα που θα μπορούσε χωρίς κανένα πρόβλημα να αποτελεί τον πυρήνα ενός επαγγελματικού προγράμματος πεπερασμένων στοιχείων. Και τα δύο είναι πεδία που με ενδιαφέρουν πολύ επιστημονικά, καθ’ ότι θεωρώ ότι ο μεν προγραμματισμός είναι το απαραίτητο συμπλήρωμα στη φιλοσοφική σκέψη ενός μαθηματικού. Μαθηματικού, γιατί απλούστατα δεν υπάρχει άλλη θετική επιστήμη στον κόσμο τούτο, φιλοσοφική, γιατί απλούστατα δεν υπάρχει άλλη πορεία προς τη σοφία –αντιστέκομαι στον πειρασμό του προφανούς ετυμολογικού σχολίου – προχωρώ όμως στην ερμηνεία του· παρατηρώντας ότι η φιλοσοφία δεν είναι τρόπος, ούτε μεθοδολογία, ούτε αποκύημα σκέψης. Φιλοσοφία είναι ένα και μόνο πράγμα: Αγάπη. Αγάπη για τη σοφία, για τη μία και μοναδική και καθαρή σοφία. Για εκείνη τη σοφία που δεν σε αφήνει να αναρωτιέσαι για την ορθότητά της, ούτε περιμένει να τη βρεις με πολύπλοκους τρόπους, αφού είναι παντελώς απρόσιτη για μας· παρά μόνο αποκαλύπτεται και τότε…. Τότε πείθει μόνο με τη λάμψη της. Και με δεδομένο ότι η αγάπη είναι ταυτόχρονα και λόγος και τρόπος ύπαρξης….

Και τώρα οι ευχαριστίες. Ποιούς να ευχαριστήσω όμως; Μήπως αυτούς που μου έκαναν τα χατίρια, με απάλλαξαν από «περιττό» κόπο, με ευχαρίστησαν με τον ένα ή τον άλλο τρόπο; Μήπως όμως και εκείνους που με στενοχώρησαν, που ήταν σκληροί μαζί μου; Μήπως τελικά και αυτούς που απλώς αδιαφόρησαν; Να ευχαριστήσω τους δασκάλους μου, τον γείτονα που με έμαθε να μην εκνευρίζομαι με τη μουσική του, τους συμφοιτητές μου που μου έμαθαν να ….μην είμαι πάντα ο εαυτός μου; Νιώθω πολύ μικρός για να καταλάβω τελικά το σχέδιο Του Θεού για την αφεντιά μου, οπότε τελικά θα σας ευχαριστήσω όλους και προπαντός Εκείνον. Αλλά μεταξύ μας, ακόμα και αν το καταλάβαινα, πάλι το ίδιο θα έκανα.–

Αθήνα, 8-12-2013

Περιεχόμενα

<u>Κεφάλαιο 1:</u>
Στοιχεία θεωρίας πεπερασμένων στοιχείων

1.1. Εισαγωγή στα πεπερασμένα στοιχεία

Τα φαινόμενα στην φύση τα αντιλαμβανόμαστε αρχικά με την παρατήρηση. Στη συνέχεια ακολουθεί το πείραμα, στο οποίο αναπαράγουμε το φαινόμενο κάτω από ελεγχόμενες συνθήκες και τελικά έπεται η ανάλυση του φαινομένου. Αποτέλεσμα της ανάλυσης είναι η διατύπωση ενός νόμου που διέπει το φαινόμενο με τον οποίο μπορούμε να προβλέψουμε στην ουσία την εξέλιξη του φαινομένου κάθε φορά που αυτό εμφανίζεται, χωρίς να κάνουμε επιπλέον δοκιμές. Ο τελικός στόχος όμως δεν είναι μόνο αυτός, αλλά βρίσκεται ακόμα ένα βήμα πιο πέρα. Την κατανόηση του μηχανισμού που δημιουργεί το φαινόμενο και την τελική διατύπωση μιας θεωρίας που εξηγεί το λόγο για τον οποίο ο νόμος που το διέπει είναι ο συγκεκριμένος και όχι κάποιος άλλος.

Το τελικό αυτό στάδιο εκφράζεται και περιγράφεται στη γλώσσα των μαθηματικών με ένα σύνολο, σύστημα όπως λέγεται, εξισώσεων αλγεβρικών ή διαφορικών. Παρά το γεγονός όμως ότι η διατύπωσή τους δεν είναι ιδιαίτερα δύσκολη, η αναλυτική επίλυσή τους και η παραγωγή κλειστών αλγεβρικών τύπων είναι σε κάποιες περιπτώσεις πολύ δύσκολη, σε κάποιες άλλες όμως ακόμα και αδύνατη. Σε όλες τις περιπτώσεις αυτές καταφεύγουμε σε αριθμητικές επιλύσεις, οι οποίες δεν μας δίνουν την «ακριβή» λύση, μας δίνουν όμως μια λύση «με όση ακρίβεια θέλουμε». Οι αριθμητικές επιλύσεις συνίστανται συνήθως σε επαναλαμβανόμενες, απλές αλγεβρικές πράξεις οι οποίες όμως όσο περισσότερο επαναλαμβάνονται τόσο καλύτερη ακρίβεια δίνουν. Σε τέτοιες περιπτώσεις λοιπόν η χρήση Η/Υ είναι ο καλύτερος τρόπος για να λάβει κανείς τα απαιτούμενα αποτελέσματα.

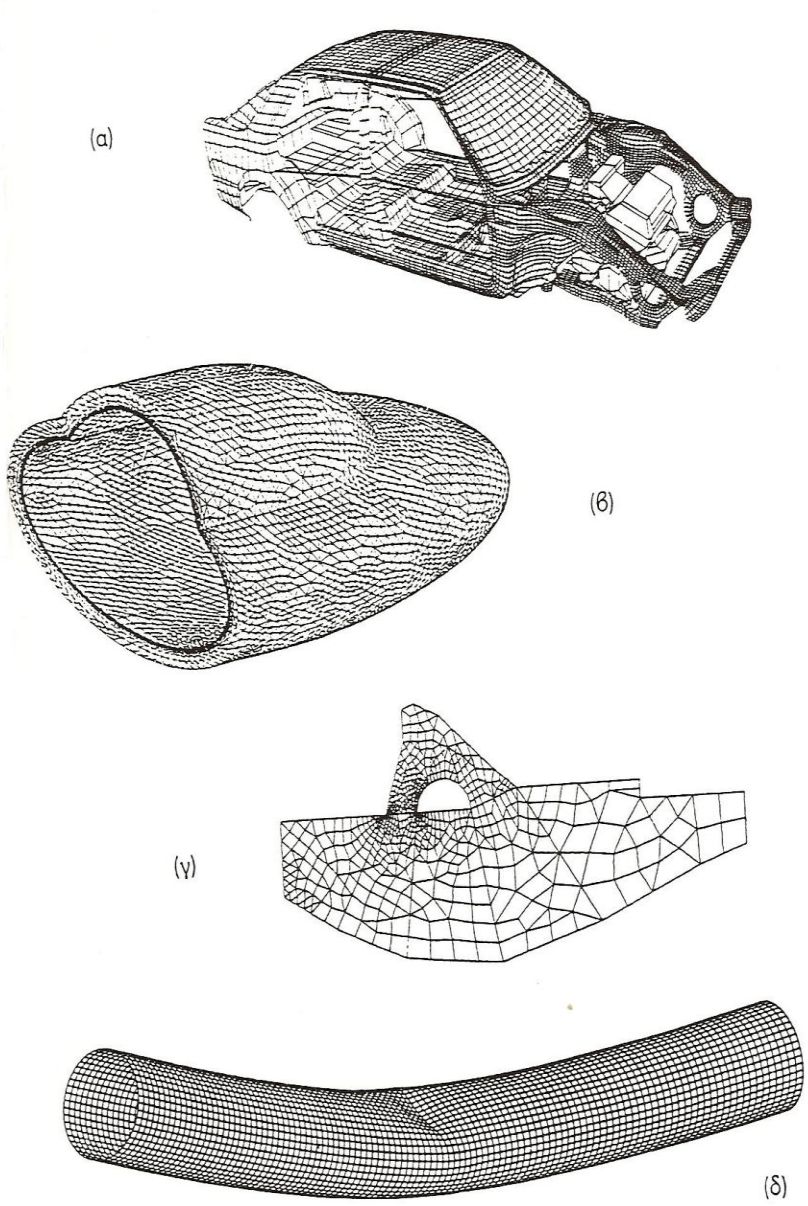
Αυτή η προοπτική που μας δίνουν οι αριθμητικές λύσεις δεν πέρασε απαρατήρητη και αρκετές προσπάθειες έχουν γίνει μέχρι σήμερα για ανάπτυξη και βελτίωση τέτοιων μεθόδων, τόσο από την πλευρά των μαθηματικών, όσο και της επιστήμης των υπολογιστών. Η κριτική μιας αριθμητικής μεθόδου βασίζεται σε ένα και μόνο στοιχείο: Στην ακρίβεια της επίλυσης σε συνδυασμό με το χρόνο στον οποίο αυτή μας δίνεται. Είναι λοιπόν σαφές ότι η βελτίωση μιας αριθμητικής μεθόδου στοχεύει σε δύο πυλώνες που ή θα ικανοποιούνται και οι δύο ταυτόχρονα ή κανένας: α) Στη βελτίωση της παραγόμενης ακρίβειας για τον ίδιο χρόνο εκτέλεσης και β) μείωση του χρόνου εκτέλεσης για παραγωγή λύσης δεδομένης ακρίβειας.

Η πιο σημαντική, διαδεδομένη και αρκετά γενική, ώστε να επεκταθεί η εφαρμογή της σε κάθε πρόβλημα, μέθοδος στην επιστήμη του μηχανικού και όχι μόνο είναι αυτή των πεπερασμένων στοιχείων. Η μέθοδος αυτή χρησιμοποιήθηκε για πρώτη φορά το 1944 από τον Ιωάννη Αργύρη, ο οποίος εργαζόταν ως ερευνητής στο Royal Aeronautical Society της Βρετανίας, για τη βελτιστοποίηση του σχήματος των πτερύγων των μαχητικών αεροσκαφών της εποχής. Σημαντική συμβολή στη μέθοδο αυτή προσέφεραν δώδεκα χρόνια αργότερα οι Turner, Clough, Martin και Topp.

Η μέθοδος βασίζεται σε πολύ διαδεδομένες προσεγγιστικές μεθόδους , όπως αυτές των μεταβολών, στις οποίες ανήκει και η μέθοδος Rayleigh- Ritz και των σταθμικών υπολοίπων, όπως η μέθοδος Galerkin. Η μέθοδος των πεπερασμένων στοιχείων μπορεί να θεωρηθεί ως μια ειδική διατύπωση όλων αυτών, που πλεονεκτεί όμως ως προς την ευκολία προγραμματισμού της στον Η/Υ και χειρισμού πολύπλοκων γεωμετριών. Η μέθοδος βασίζεται στη διακριτοποίηση του πεδίου των λύσεων σε διακριτά στοιχεία τα οποία στο όριό τους διαθέτουν κόμβους. Σε αυτούς τους κόμβους υπολογίζονται αρχικά οι τιμές του μεγέθους που μας ενδιαφέρει και στη συνέχεια υπολογίζονται και εσωτερικά του στοιχείου από παρεμβολή των τιμών στους κόμβους του. Όσο πιο μικρά είναι τα στοιχεία στα οποία χωρίζουμε το πεδίο, τόσο αυξάνεται η ακρίβεια της επίλυσης. Αυτό όμως έχει και ένα άνω όριο, διότι πολύ μικρά στοιχεία μπορεί να προκαλέσουν αριθμητικές αστάθειες λόγω του τρόπου με τον οποίο διαχειρίζεται τους μικρούς αριθμούς το σύστημα του υπολογιστή μας. Στην περίπτωση του πολιτικού μηχανικού τα κύρια μεγέθη που μας ενδιαφέρουν σε μια ανάλυση πεπερασμένων στοιχείων είναι οι τάσεις και οι μετατοπίσεις του φορέα.

1.2. Η μέθοδος των πεπερασμένων στοιχείων με το μοντέλο των μετατοπίσεων

Κατά την ανάπτυξη της μεθόδου των πεπερασμένων στοιχείων έχουμε τη δυνατότητα να επιλέξουμε σαν βασικές μεταβλητές του προβλήματος τις κομβικές μετατοπίσεις ή τις κομβικές δυνάμεις. Έτσι έχουμε δυο μοντέλα ανάπτυξης της μεθόδου, το *μοντέλο των μετατοπίσεων* ή *κινηματικό μοντέλο* και το *μοντέλο των δυνάμεων* ή *στατικό μοντέλο.* Όμως αυτό που κυρίως χρησιμοποιείται είναι αυτό των μετατοπίσεων. Η



Εικόνα 1.1: Παραδείγματα διακριτοποίησης φορέων με πεπερασμένα στοιχεία: (α) αυτοκίνητο, (β) πρόσθιο τμήμα ατράκτου αεροσκάφους, (γ) φράγμα, (δ) παραμορφωμένος σωλήνας

προτίμηση προς αυτή την κατεύθυνση έχει να κάνει με την προγραμματιστική ευκολία της μεθόδου. Στη μέθοδο των μετατοπίσεων κάνουμε χρήση του *μητρώου δυσκαμψίας* του φορέα, σε αντίθεση με τη μέθοδο των δυνάμεων όπου γίνεται χρήση του *μητρώου ευκαμψίας.* Η όλη διαφορά έγκειται στο γεγονός ότι το μητρώο ευκαμψίας είναι γενικά πλήρες, κάτι που οδηγεί σε μεγάλες απαιτήσεις μνήμης κατά την αποθήκευση των δεικτών ευκαμψίας, ενώ το μητρώο δυσκαμψίας έχει αρκετά μηδενικά στοιχεία οπότε και οι απαιτήσεις μνήμης για την αποθήκευση των δεικτών δυσκαμψίας είναι ασήμαντες σε σχέση με τις αντίστοιχες προηγούμενες. Ειδικά, αν αναλογιστεί κανείς πως σε προσομοιώματα αληθινών κατασκευών με πολλούς βαθμούς ελευθερίας τα μηδενικά στοιχεία μπορεί να αντιστοιχούν μέχρι και στο 99% του συνολικού αριθμού των δεικτών δυσκαμψίας του μητρώου. Επίσης η τοπική διάρθρωση του μητρώου δυσκαμψίας ευνοεί τον προγραμματισμό της μεθόδου των μετατοπίσεων και την άμεση γενίκευσή της σε οποιοδήποτε τύπο φορέα. Σε αντιδιαστολή με τη μέθοδο των δυνάμεων όπου, λόγω της καθολικής διάρθρωσης του μητρώου ευκαμψίας, ειδική μέριμνα πρέπει να ληφθεί κατά τη δημιουργία του θεμελιώδη φορέα προκειμένου να εφαρμοστεί η μέθοδος του μοναδιαίου φορτίου.

Κατά την εφαρμογή της μεθόδου των πεπερασμένων στοιχείων, μια κατασκευή μπορεί να διακριτοποιηθεί σε μικρότερα τμήματα πεπερασμένου μεγέθους που αποκαλούνται *πεπερασμένα στοιχεία.* Επομένως, η κατασκευή μπορεί να θεωρηθεί σαν μια σύνθεση αυτών των στοιχείων, μέσω ενός πεπερασμένου αριθμού ενώσεων που ονομάζονται *κόμβοι.* Παραδείγματα διακριτοποιημένων φορέων σε πεπερασμένα στοιχεία δίδονται στην *Εικόνα 1.1*.

Η διαδικασία ανάλυσης μιας κατασκευής με τη μέθοδο των πεπερασμένων στοιχείων αποτελείται από τα ακόλουθα ενδεικτικά στάδια:

1. Επιλογή *συναρτήσεων σχήματος*, ώστε να προσεγγίσουμε τη μεταβολή των μετατοπίσεων μέσα στο στοιχείο και να τη συσχετίσουμε με τις μετατοπίσεις στους κόμβους του στοιχείου.
2. Έκφραση των παραμορφώσεων και τάσεων εντός του στοιχείου σε σχέση με τις επικόμβιες μετατοπίσεις μέσω κατάλληλων μητρώων.
3. Χρήση της αρχής των δυνατών έργων, ώστε να διατυπωθεί η εξίσωση ισορροπίας κάθε στοιχείου (άγνωστες μεταβλητές είναι οι επικόμβιες μετατοπίσεις).
4. Συνδυασμός των επιμέρους εξισώσεων ισορροπίας κάθε στοιχείου με σκοπό να καταλήξουμε στη γενική εξίσωση ισορροπίας για όλο το φορέα, αλλά κατά τέτοιο τρόπο ώστε να διασφαλίζεται η συνέχεια των μετατοπίσεων σε κάθε κόμβο.
5. Επίλυση των εξισώσεων ισορροπίας ώστε να προκύψουν οι άγνωστες επικόμβιες μετατοπίσεις, και να οδηγηθούμε κατόπιν τούτου στον
6. Υπολογισμό των τάσεων και των παραμορφώσεων λαμβάνοντας υπόψη τα χαρακτηριστικά των στοιχείων.

Επομένως, ένα τυπικό πρόγραμμα πεπερασμένων στοιχείων απαρτίζεται από την εξής σειρά εργασιών:

1. Εισαγωγή δεδομένων κόμβων (συντεταγμένες, συνοριακές συνθήκες)
2. Υπολογισμός και αποθήκευση του διανύσματος φόρτισης
3. Μόρφωση καθολικού μητρώου ακαμψίας
4. Υπολογισμός επικόμβιων μετατοπίσεων
5. Υπολογισμός τάσεων
6. Εκτύπωση αποτελεσμάτων

1.3. Διατύπωση των εξισώσεων ισορροπίας με την αρχή των δυνατών έργων

1.3.1. Η αρχή των δυνατών έργων

Η μεθοδολογία που θα αναπτυχθεί στη συνέχεια συνοπτικά, στηρίζεται στην αρχή των δυνατών έργων. Σύμφωνα λοιπόν με την αρχή αυτή, όταν ένας φορέας φορτίζεται με εξωτερικά φορτία και βρίσκεται σε ισορροπία, τότε για οποιαδήποτε μικρή δυνατή παραμόρφωση του φορέα, συμβιβαστή με τις συνθήκες στήριξής του, το δυνατό έργο των εσωτερικών δυνάμεων είναι ίσο με το δυνατό έργο των εξωτερικών δυνάμεων. Η εξίσωση της αρχής των δυνατών έργων γράφεται στη μορφή:

$$W_{\varepsilon\sigma\omega\tau.} = W_{\varepsilon\xi\omega\tau.} \tag{2.1}$$

$$\int_V \{\varepsilon\}^T * \{\sigma\} dV = \int_V \{\overline{U}\}^T * \{f^V\} * dV + \int_S \{\overline{U}^S\}^T * \{f^S\} * dS + \{\overline{D}\}^T * \{R_C\} \tag{2.2}$$

Όπου:

$\{\varepsilon\}$: διάνυσμα των δυνατών ανηγμένων παραμορφώσεων

$\{\sigma\}$: διάνυσμα των τάσεων που ισορροπούν τα εξωτερικά φορτία

$\{\overline{U}\}$: διάνυσμα των δυνατών μετατοπίσεων ενός τυχαίου σημείου $P(X,Y,Z)$ του φορέα

$\{f^V\}$: διάνυσμα των μαζικών δράσεων (δράσεις ανά μονάδα όγκου)

$\{\overline{U}^S\}$: διάνυσμα των δυνατών μετατοπίσεων ενός τυχαίου σημείου της φορτιζόμενης επιφάνειας S

$\{f^S\}$: διάνυσμα των επιφανειακών δράσεων (δράσεις ανά μονάδα επιφανείας)

$\{\overline{D}\}$: διάνυσμα των δυνατών επικόμβιων μετατοπίσεων του φορέα

$\{R_C\}$: διάνυσμα των δυνατών επικόμβιων δράσεων του φορέα

1.3.2. Υπολογισμός των παραμέτρων του στοιχείου

Ιδιαίτερα χρήσιμες για τη μέθοδο των πεπερασμένων στοιχείων είναι και οι τρεις παρακάτω σχέσεις:

1. Σχέση τάσεων – ανηγμένων παραμορφώσεων:

$$\{\sigma_m(X,Y,Z)\} = [E_m] * \{\varepsilon_m(X,Y,Z)\} \tag{2.3}$$

Όπου:

$\{\sigma_m(X,Y,Z)\}$: διάνυσμα των τάσεων ενός σημείου $P(X,Y,Z)$ του πεπερασμένου στοιχείου m

$\{\varepsilon_m(X,Y,Z)\}$: διάνυσμα των ανηγμένων παραμορφώσεων ενός σημείου $P(X,Y,Z)$ του πεπερασμένου στοιχείου m

$[E_m]$: μητρώο ελαστικότητας του πεπερασμένου στοιχείου m

Το διάνυσμα των τάσεων και το διάνυσμα των ανηγμένων παραμορφώσεων στη γενική περίπτωση της τρισδιάστατης ανάλυσης είναι αντίστοιχα:

$$\{\sigma_m(X,Y,Z)\} = \begin{Bmatrix} \sigma_{XX,m}(X,Y,Z) \\ \sigma_{YY,m}(X,Y,Z) \\ \sigma_{ZZ,m}(X,Y,Z) \\ \sigma_{XY,m}(X,Y,Z) \\ \sigma_{YZ,m}(X,Y,Z) \\ \sigma_{ZX,m}(X,Y,Z) \end{Bmatrix} \text{ και } \{\varepsilon_m(X,Y,Z)\} = \begin{Bmatrix} \varepsilon_{XX,m}(X,Y,Z) \\ \varepsilon_{YY,m}(X,Y,Z) \\ \varepsilon_{ZZ,m}(X,Y,Z) \\ \varepsilon_{XY,m}(X,Y,Z) \\ \varepsilon_{YZ,m}(X,Y,Z) \\ \varepsilon_{ZX,m}(X,Y,Z) \end{Bmatrix} \tag{2.4}$$

Το ελαστικό μητρώο $[E_m]$ έχει γενικά 36 μη μηδενικές ελαστικές σταθερές. Ανάλογα με τις απλοποιητικές παραδοχές και την εντατική κατάσταση στην οποία θεωρούμε ότι βρίσκεται ο φορέας, επιλέγεται ο κατάλληλος τύπος πεπερασμένου στοιχείου και τροποποιείται κατάλληλα το διάνυσμα των τάσεων, των παραμορφώσεων και το μητρώο ελαστικότητας.

2. Συνάρτηση σχήματος πεπερασμένου στοιχείου:

$$\{U_m(X,Y,Z)\} = [N_m(X,Y,Z)] * \{d_m\} \tag{2.5}$$

Όπου:

$\{U_m(X,Y,Z)\}$: διάνυσμα των μετατοπίσεων ενός σημείου $P(X,Y,Z)$ στο εσωτερικό του στοιχείου m

$[N_m(X,Y,Z)]$: μητρώο των συναρτήσεων σχήματος ή συναρτήσεων παρεμβολής το οποίο εκφράζει τον τρόπο με τον οποίο συνδέονται οι μετατοπίσεις σε οποιοδήποτε σημείο ενός πεπερασμένου στοιχείου με τις μετατοπίσεις τον κόμβων του στοιχείου. Το μητρώο αυτό εξαρτάται από τον τύπο του στοιχείου και παίζει σημαντικό ρόλο στην ακρίβεια με την οποία υπολογίζεται το μητρώο δυσκαμψίας του στοιχείου

$\{d_m\}$: διάνυσμα των επικόμβιων μετατοπίσεων του στοιχείου m

3. Μητρώο παραμόρφωσης του στοιχείου:

$$\{\varepsilon_m\} = [B_m(X,Y,Z)] * \{d_m\}$$
$$\{d_m\} = [t_m] * \{D\} \quad (2.6)$$

Όπου:

$\{\varepsilon_m\}$: διάνυσμα των ανηγμένων παραμορφώσεων του πεπερασμένου στοιχείου m

$[B_m(X,Y,Z)]$: μητρώο παραμορφώσεως του στοιχείου m και συνδέει το διάνυσμα των ανηγμένων παραμορφώσεων με τις επικόμβιες μετατοπίσεις του στοιχείου

$\{d_m\}$: επικόμβιες μετατοπίσεις του στοιχείου m

$[t_m]$: είναι ένα λογικό μητρώο (boolean), δηλαδή ένα μητρώο με στοιχεία 0 ή 1, και στην περίπτωσή μας χρησιμεύει για να συνδέει τους τοπικούς με τους καθολικούς βαθμούς ελευθερίας των κόμβων του στοιχείου και εκφράζει τη συνθήκη του συμβιβαστού των μετατοπίσεων των κόμβων του στοιχείου με τους κόμβους του φορέα στους οποίους αντιστοιχούν

1.3.3. Εξισώσεις ισορροπίας με επικόμβια φορτία

Η εξίσωση της αρχής των δυνατών έργων (2.2) του φορέα για την περίπτωση του φορέα που φορτίζεται μόνο με επικόμβια φορτία, γράφεται:

$$\int_V \{\varepsilon\}^T * \{\sigma\} * dV = \{\overline{D}\}^T * \{R_C\} \quad (2.7)$$

Το ολοκλήρωμα της σχέσης (2.7) μπορεί να γραφτεί ως άθροισμα του δυνατού έργου των εσωτερικών δυνάμεων όλων των στοιχείων m με τα οποία έχει διακριτοποιηθεί ο φορέας:

$$\sum_m \int_{V_e} \{\bar{\varepsilon}_m\}^T * \{\sigma_m\} * dV_e = \{\overline{D}\}^T * \{R_C\} \quad (2.8)$$

Με V_e τον όγκο του κάθε στοιχείου m.

Με την αντικατάσταση των σχέσεων (2.7) στη σχέση (2.8) προκύπτει η σχέση:

$$\{\overline{D}\}^T * \left(\sum_m \int_{V_e} [t_m]^T * [B_m]^T * [E] * [B_m] * [t_m] * dV_e \right) * \{D\} = \{\overline{D}\}^T * \{R_C\} \quad (2.9)$$

Το διάνυσμα των δυνατών επικόμβιων μετατοπίσεων $\{\overline{D}\}$ του φορέα είναι κοινό για όλα τα στοιχεία και ανεξάρτητο του στοιχείου m. Κατά συνέπεια μπορούν να περάσουν

εκτός του ολοκληρώματος και του αθροίσματος της σχέσης (2.9). Επιπλέον οι δυνατές επικόμβιες μετατοπίσεις του φορέα είναι τυχαίες μη μηδενικές ποσότητες και δύναται να απλοποιηθούν από τη σχέση (2.9). Έτσι έχουμε:

$$\left(\sum_{m}\int_{V_e}[t_m]^T * [B_m]^T * [E] * [B_m] * [t_m] * dV_e\right) * \{D\} = \{R_C\} \Leftrightarrow [K] * \{D\} = \{R_C\} \quad (2.10)$$

Η σχέση (2.10) δίνει την εξίσωση ισορροπίας του φορέα στην οποία το μητρώο $[K]$ εκφράζει το ολικό μητρώο δυσκαμψίας του φορέα το οποίο προκύπτει από τη σύνθεση των επιμέρους μητρώων δυσκαμψίας των στοιχείων του.
Οι επικόμβιες δράσεις του φορέα εκφράζονται ως εξής:

$$\{R_C\} = \sum_{m}[t_m]^T * \{r_{C,m}\} \quad (2.11)$$

Όπου $\{r_{C,m}\}$ είναι οι επικόμβιες δράσεις του στοιχείου.

1.3.4. Ισοδύναμες δράσεις μαζικών και επιφανειακών δυνάμεων

$$\{R_V\} = \sum_{m}[t_m]^T * \int_{V_e}[N_m]^T * \{f^V{}_m\} * dV_e \quad (2.12)$$

$$\{R_S\} = \sum_{m}[t_m]^T * \int_{S_e}[N^S{}_m]^T * \{f^S{}_m\} * dS_e \quad (2.13)$$

Όπου:
$\{R_V\}$: ισοδύναμες μαζικές δράσεις
$\{R_S\}$: ισοδύναμες επιφανειακές δράσεις
$[N_m]$: μητρώο συναρτήσεων σχήματος στοιχείου m
$[N^S{}_m]$: μητρώο συναρτήσεων σχήματος στοιχείου m για επιφανειακές δράσεις
$\{f^V{}_m\}$: κατανεμημένη μαζική φόρτιση του στοιχείου m ανά μονάδα όγκου
$\{f^S{}_m\}$: κατανεμημένη επιφανειακή φόρτιση του στοιχείου m ανά μονάδα επιφάνειας
S_e: φορτιζόμενη επιφάνεια του στοιχείου m

1.3.5. Ισοδύναμες δράσεις αρχικών τάσεων και παραμορφώσεων

$$\{R_{\sigma_0}\} = -\sum_{m}[t_m]^T * \int_{V_e}[B_m]^T * \{\sigma_{0,m}\} * dV_e \quad (2.14)$$

$$\{R_{\varepsilon_0}\} = \sum_{m}[t_m]^T * \int_{V_e}[B_m]^T * [E] * \{\varepsilon_{0,m}\} * dV_e \quad (2.15)$$

όπου
$\{R_{\sigma_0}\}$: δράσεις στους κόμβους του φορέα λόγω αρχικής τάσης $\{\sigma_0\}$
$\{R_{\varepsilon_0}\}$: δράσεις στους κόμβους του φορέα λόγω αρχικής παραμόρφωσης $\{\varepsilon_0\}$

1.3.6. Γενική εξίσωση ισορροπίας φορέα και στοιχείων

Με βάση τις τελικές εκφράσεις των επιμέρους χαρακτηριστικών μεγεθών που υπεισέρχονται στην εξίσωση ισορροπίας του φορέα έχουμε τη σχέση:

$$[K] * \{D\} = \{R\} \tag{2.16}$$

Όπου

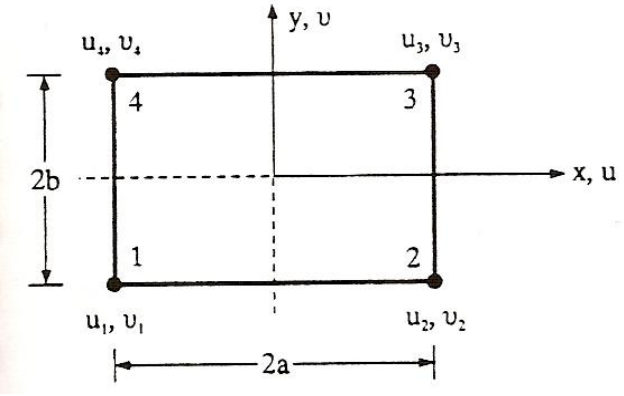
$$[K] = \sum_{m} \int_{V_e} [t_m]^T * [B_m]^T * [E] * [B_m] * [t_m] * dV_e \tag{2.17}$$

$$\{R\} = \{R_C\} + \{R_V\} + \{R_S\} + \{R_{S_0}\} + \{R_{e_0}\} \tag{2.18}$$

1.4. Στοιχεία επίπεδης έντασης – παραμόρφωσης

1.4.1. Ορθογωνικό στοιχείο επίπεδης έντασης τεσσάρων κόμβων

Στην *Εικόνα 1.2* φαίνεται ένα ορθογωνικό στοιχείο τεσσάρων κόμβων στο τοπικό σύστημα συντεταγμένων xy . Ο προσανατολισμός του συστήματος συντεταγμένων ταυτίζεται με τις διευθύνσεις των πλευρών ενώ η αρχή των αξόνων μπορεί να τοποθετηθεί και σε οποιοδήποτε άλλο στοιχείο του επιπέδου xy χωρίς να επηρεαστούν οι ιδιότητες του στοιχείου.



Εικόνα 1.2: Ορθογωνικό στοιχείο επίπεδης έντασης

1. **Συναρτήσεις σχήματος**

   Το πεδίο των μετατοπίσεων ορίζεται από γραμμικά πολυώνυμα ως προς x και y αφού δύο είναι οι επικόμβιοι βαθμοί ελευθερίας για κάθε συνιστώσα της μετατόπισης σε κάθε πλευρά του στοιχείου. Κατά συνέπεια οι πολυωνυμικές σχέσεις που ορίζουν τις μετατοπίσεις u, v σε κάθε σημείο $P(x, y)$ του στοιχείου έχουν τη μορφή:

$$\begin{aligned} u &= a_1 + a_2 * x + a_3 * y + a_4 * x * y \\ v &= a_5 + a_6 * x + a_7 * y + a_8 * x * y \end{aligned} \tag{2.19}$$

   Δηλαδή:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & x & y & x*y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & x*y \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} \tag{2.20}$$

Ο υπολογισμός των γενικευμένων συντεταγμένων $\{a\}$ ακολουθεί την πάγια διαδικασία με την τοποθέτηση του σημείου $P(x,y)$ κυκλικά στους κόμβους 1, 2, 3 και 4. το αποτέλεσμα, σε μητρωική μορφή, δίνεται από τη σχέση:

$$\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{bmatrix} 1 & -a & -b & a*b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -a & -b & a*b \\ 1 & a & -b & -a*b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & a & -b & -a*b \\ 1 & a & b & a*b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & a & b & a*b \\ 1 & -a & b & -a*b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -a & b & -a*b \end{bmatrix} * \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{Bmatrix} \tag{2.21}$$

ή

$$\{d\} = [A] * \{a\} \tag{2.22}$$

Η λύση ως προς $\{a\}$ δίνει:

$$\{a\} = [A]^{-1} * \{d\} \tag{2.23}$$

ή

$$\{a\} = \frac{1}{4*a*b} * \begin{bmatrix} a*b & 0 & a*b & 0 & a*b & 0 & a*b & 0 \\ b & 0 & b & 0 & b & 0 & -b & 0 \\ a & 0 & -a & 0 & a & 0 & a & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & a*b & 0 & a*b & 0 & a & 0 & a*b \\ 0 & -b & 0 & b & 0 & b & 0 & -b \\ 0 & -a & 0 & -a & 0 & a & 0 & a \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \end{bmatrix} * \{d\} \tag{2.24}$$

Με την αντικατάσταση της σχέσης (2.24) στη σχέση (2.20) ορίζεται το πεδίο των μετατοπίσεων συναρτήσει των επικόμβιων μετατοπίσεων και έτσι προκύπτουν οι συναρτήσεις σχήματος του στοιχείου:

$$\{u\} = \begin{bmatrix} 1 & x & y & x*y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & x*y \end{bmatrix} * [A]^{-1} * \{d\} \tag{2.25}$$

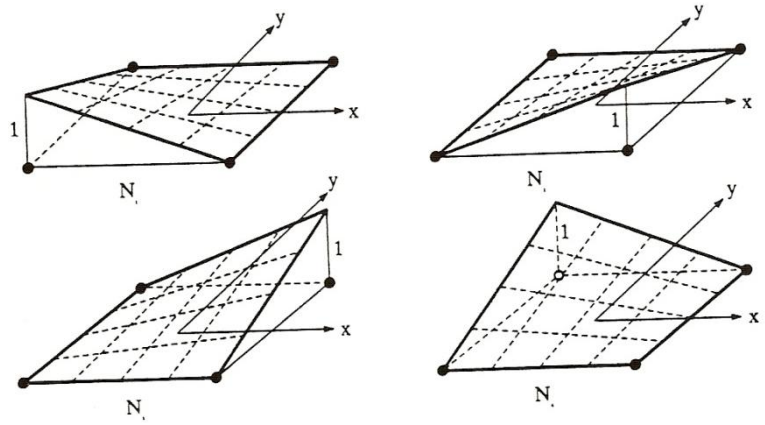
ή

$$\{u\} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} * \{d\} \tag{2.26}$$

Όπου:

$$\begin{aligned} N_1 &= \frac{1}{4} * \left(1 - \frac{x}{a}\right) * \left(1 - \frac{y}{b}\right) \\ N_2 &= \frac{1}{4} * \left(1 + \frac{x}{a}\right) * \left(1 - \frac{y}{b}\right) \\ N_3 &= \frac{1}{4} * \left(1 + \frac{x}{a}\right) * \left(1 + \frac{y}{b}\right) \\ N_4 &= \frac{1}{4} * \left(1 - \frac{x}{a}\right) * \left(1 + \frac{y}{b}\right) \end{aligned} \tag{2.27}$$

Οι γραφικές παραστάσεις των συναρτήσεως σχήματος έχουν σχεδιαστεί στην *Εικόνα 1.3*.



Εικόνα 1.3: Συναρτήσεις σχήματος ορθογωνικού στοιχείου 4 κόμβων

Από την *Εικόνα 1.3* και τη σχέση (2.26) συνεπάγεται ότι η κατανομή της N_i εντός του στοιχείου παριστάνει τη μετατόπιση u ή v εντός του στοιχείου όταν $u_i = 1$ ή $v_i = 1$ και $u_j = 0$, $u_j = 0$ για $i \neq j$. Επίσης ισχύει η σχέση $N_1 + N_2 + N_3 + N_4 = 1$.

Οι συναρτήσεις σχήματος (2.27) μπορούν να προκύψουν απευθείας κάνοντας χρήση των πολυωνύμων Lagrange. Για την περίπτωση της γραμμικής παρεμβολής μεταξύ δυο σημείων $x_1 = -a$, $x_2 = a$ $(n = 2)$ τα πολυώνυμα Lagrange εκφράζονται από τις σχέσεις

$$L_1(x) = \frac{a - x}{2 * a}, L_2(x) = \frac{a + x}{2 * a} \tag{2.28}$$

και μεταξύ των σημείων $y_1 = -b$, $y2 = b$ από τις σχέσεις:

$$L_1(y) = \frac{b - y}{2 * b}, L_2(y) = \frac{b + y}{2 * b} \tag{2.29}$$

2. Μητρώο παραμόρφωσης

Οι σχέσεις των ανηγμένων παραμορφώσεων – μετατοπίσεων για την περίπτωση της επίπεδης έντασης εκφράζονται από τις πιο κάτω σχέσεις:

$$\begin{aligned} \frac{\partial u}{\partial x} &= N_{1,x} * u_1 + N_{2,x} * u_2 + N_{3,x} * u_3 + N_{4,x} * u_4 \\ \frac{dv}{dy} &= N_{1,y} * v_1 + N_{2,y} * v_2 + N_{3,y} * v_3 + N_{4,y} * v_4 \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} &= N_{1,y} * u_1 + N_{1,x} * v_1 + N_{2,y} * u_2 + N_{2,x} * v_2 + \\ &\quad + N_{3,y} * u_3 + N_{3,x} * v_3 + N_{4,y} * u_4 + N_{4,x} * u_4 \end{aligned} \tag{2.30}$$

ή

$$\{\varepsilon\} = \begin{Bmatrix} u_{,x} \\ v_{,y} \\ u_{,y} + v_{,x} \end{Bmatrix} = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 & N_{3,x} & 0 & N_{4,x} & 0 \\ 0 & N_{1,y} & 0 & N_{2,y} & 0 & N_{3,y} & 0 & N_{4,y} \\ N_{1,y} & N_{1,x} & N_{2,y} & N_{2,x} & N_{3,y} & N_{3,x} & N_{4,y} & N_{4,x} \end{bmatrix} * \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix}$$

ή

$$\{\varepsilon\} = [B] * \{d\} \tag{2.31}$$

με

$$\begin{aligned} N_{1,x} &= \ \ y - b \\ N_{2,x} &= -y + b \\ N_{3,x} &= \ \ y + b \end{aligned} \tag{2.32}$$

$$N_{4,x} = -y - b$$
$$N_{1,y} = \ \ x - a$$
$$N_{2,y} = -x - a$$
$$N_{3,y} = \ \ x + a$$
$$N_{4,y} = -x + a$$

3. Μητρώο δυσκαμψίας

Το μητρώο δυσκαμψίας του ορθογωνικού στοιχείου τεσσάρων κόμβων επίπεδης έντασης δίνεται από τη σχέση:

$$\underset{(8\times8)}{[k]} = \int_{V_e} \underset{(8\times3)}{[B]^T} * \underset{(3\times3)}{[E]} * \underset{(3\times8)}{[B]} * dV_e \tag{2.33}$$

ή

$$[k] = t * \int_{y=-b}^{b} \int_{x=-a}^{a} [B]^T * [E] * [B] * dx * dy \tag{2.34}$$

Όπου:

t: είναι το πάχος του στοιχείου το οποίο θεωρήθηκε σταθερό στην επιφάνειά του

$[E]$: μητρώο ελαστικότητας σε συνθήκες επίπεδης έντασης

$$[E] = \frac{E}{1-\nu^2} * \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix} \tag{2.35}$$

4. Μητρώο τάσεων

Το μητρώο τάσεων $\{s\}$ προκύπτει από τη σχέση τάσεων – ανηγμένων παραμορφώσεων $\{s\} = [E] * \{e\}$ και συνδέει το διάνυσμα των τάσεων με τις επικόμβιες μετακινήσεις του στοιχείου. Για την περίπτωση του ορθογωνικού στοιχείου τεσσάρων κόμβων το μητρώο των τάσεων δίνεται από τη σχέση:

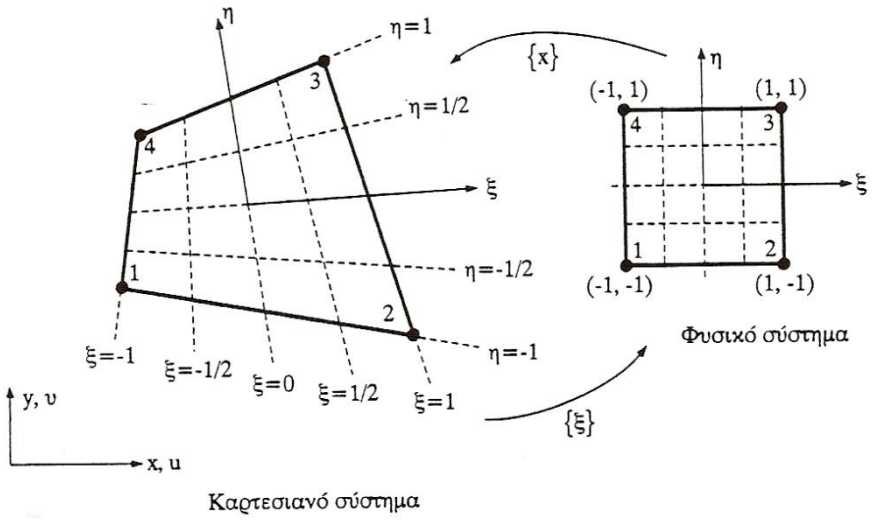
$$\underset{(3\times1)}{\{\sigma\}} = \underset{(3\times3)}{[E]} * \underset{(3\times8)}{[B]} * \underset{(8\times1)}{\{d\}} \tag{2.36}$$

ή

$$\underset{(3\times1)}{\{\sigma\}} = \underset{(3\times8)}{[S]} * \underset{(8\times1)}{\{d\}} \tag{2.37}$$

1.4.2. Τετραπλευρικό ισοπαραμετρικό στοιχείο επίπεδης παραμόρφωσης τεσσάρων κόμβων

Στην *Εικόνα 1.4* φαίνεται το τετραπλευρικό στοιχείο τεσσάρων κόμβων στα δυο συστήματα συντεταγμένων και η αμφιμονοσήμαντη απεικόνιση. Η απεικόνιση του φυσικού συστήματος συντεταγμένων στο καρτεσιανό δεν είναι απαραίτητα ένα ορθογωνικό σύστημα αξόνων. Οι άξονες x, h στο καρτεσιανό σύστημα διέρχονται από τα μέσα των πλευρών και είναι ευθύγραμμοι διότι η απεικόνιση είναι γραμμική.



Εικόνα 1.4: Τετραπλευρικό ισοπαραμετρικό στοιχείο επίπεδης παραμόρφωσης τεσσάρων κόμβων

1. Συναρτήσεις σχήματος

Ο μετασχηματισμός των συντεταγμένων ορίζεται από γραμμικά πολυώνυμα ως προς x και h αφού δυο είναι οι κόμβοι σε κάθε πλευρά του στοιχείου. Κατά συνέπεια η πολυωνυμική απεικόνιση των συντεταγμένων x, y θα έχει τη μορφή:

$$x = a_1 + a_2 * \xi + a_3 * \eta + a_4 * \xi * \eta$$
$$y = a_5 + a_6 * \xi + a_7 * \eta + a_8 * \xi * \eta$$

ή

$$x = [1 \quad \xi \quad \eta \quad \xi * \eta] * \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix}$$
$$y = [1 \quad \xi \quad \eta \quad \xi * \eta] * \begin{Bmatrix} a_5 \\ a_6 \\ a_7 \\ a_8 \end{Bmatrix} \tag{2.38}$$

Με ανάλογες σχέσεις, λόγω της ισοπαραμετρικής θεώρησης, εκφράζεται και το πεδίο των μετατοπίσεων του στοιχείου στο καρτεσιανό σύστημα:

$$u = [1 \quad \xi \quad \eta \quad \xi * \eta] * \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{Bmatrix}$$
$$v = [1 \quad \xi \quad \eta \quad \xi * \eta] * \begin{Bmatrix} b_5 \\ b_6 \\ b_7 \\ b_8 \end{Bmatrix} \tag{2.39}$$

Ο υπολογισμός των γενικευμένων συντεταγμένων $\{a\}$ ή $\{b\}$ ακολουθεί την πάγια διαδικασία:

$$\begin{aligned} x_1 &= a_1 - a_2 - a_3 + a_4 \quad (\xi = -1, \eta = -1) \\ x_2 &= a_1 + a_2 - a_3 - a_4 \quad (\xi = \ \ 1, \eta = -1) \\ x_3 &= a_1 + a_2 + a_3 + a_4 \quad (\xi = \ \ 1, \eta = \ \ 1) \\ x_4 &= a_1 - a_2 + a_3 - a_4 \quad (\xi = -1, \eta = \ \ 1) \end{aligned} \tag{2.40}$$

ή

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} * \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} \tag{2.41}$$

Από τη σχέση (1.41) προκύπτει το διάνυσμα των γενικευμένων συντεταγμένων:

$$\begin{Bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \alpha_4\end{Bmatrix} = \frac{1}{4} * \begin{bmatrix} 1 & 1 & 1 & 1\\ -1 & 1 & 1 & -1\\ -1 & -1 & 1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix} * \begin{Bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{Bmatrix} \tag{2.42}$$

Με την αντικατάσταση της σχέσης (1.42) στη σχέση (1.38α) παίρνουμε την έκφραση της συντεταγμένης x και με αντίστοιχο τρόπο της y ενός τυχαίου σημείου του στοιχείου ως προς τις συντεταγμένες $x_i, i = 1, 2, 3, 4$ των τεσσάρων κόμβων του στοιχείου. Έτσι έχουμε:

$$\begin{aligned} x &= [N_1 \quad N_2 \quad N_3 \quad N_4] * \begin{Bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{Bmatrix}\\ y &= [N_1 \quad N_2 \quad N_3 \quad N_4] * \begin{Bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{Bmatrix}\end{aligned} \tag{2.43}$$

Όπου

$$\begin{aligned} N_1 &= \frac{1}{4} * (1-\xi) * (1-\eta)\\ N_2 &= \frac{1}{4} * (1+\xi) * (1-\eta)\\ N_3 &= \frac{1}{4} * (1+\xi) * (1+\eta)\\ N_4 &= \frac{1}{4} * (1-\xi) * (1+\eta)\end{aligned} \tag{2.44}$$

είναι οι συναρτήσεις σχήματος του στοιχείου, των οποίων οι γραφικές παραστάσεις φαίνονται στην *Εικόνα 1.5*.

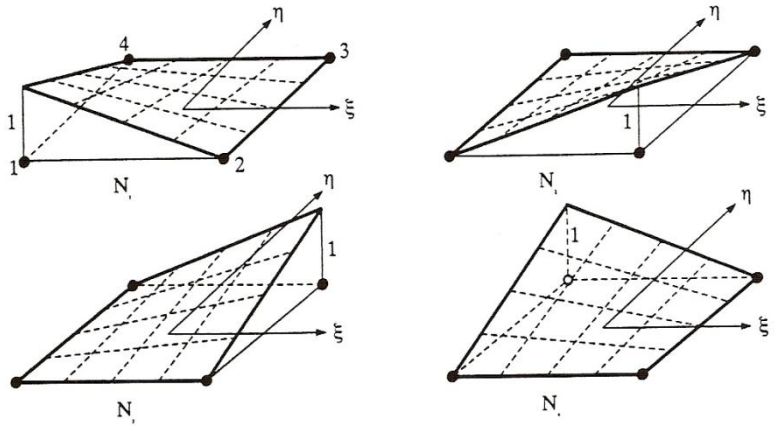
Με αντίστοιχο τρόπο ορίζονται οι συνιστώσες της μετατόπισης u, v στο καρτεσιανό σύστημα:

$$\begin{Bmatrix}u\\ v\end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0\\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4\end{bmatrix} * \begin{Bmatrix}u_1\\ v_1\\ u_2\\ v_2\\ u_3\\ v_3\\ u_4\\ v_4\end{Bmatrix} \tag{2.45}$$

ή

$$\{u\} = [N] * \{d\} \tag{2.46}$$

Για τον υπολογισμό των συντεταγμένων x, y ή των μετατοπίσεων u, v ενός τυχαίου σημείου του στοιχείου δεν έχουμε παρά να αντικαταστήσουμε τις συντεταγμένες x, h του σημείου αυτού στις σχέσεις (2.43) και (2.45) ή (2.46) αντίστοιχα.



Εικόνα 1.5: Συναρτήσεις σχήματος τετραπλευρικού ισοπαραμετρικού στοιχείου τεσσάρων κομβων στο φυσικό σύστημα

2. Ιακωβιανό μητρώο $[J]$

Προκειμένου να υπολογιστεί το μητρώο παραμορφώσεως $[B]$ του στοιχείου απαιτείται ο υπολογισμός του διανύσματος των ανηγμένων παραμορφώσεων – μετατοπίσεων της επίπεδης ελαστικότητας:

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial x} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial x} & 0 \\ 0 & \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} \end{bmatrix} * \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{2.47}$$

Η παραγώγιση των u, v ως προς x, y δε μπορεί να γίνει άμεσα από τις εξισώσεις (2.45) και (2.46) του πεδίου των μετατοπίσεων διότι οι συνιστώσες u και v έχουν εκφραστεί αναλυτικά ως προς τις φυσικές συντεταγμένες ξ, η και όχι ως προς τις καρτεσιανές συντεταγμένες x, y. Για να παρακάμψουμε αυτή τη δυσκολία ακολουθούμε την παρακάτω διαδικασία:

Έστω μια συνάρτηση φ, που ορίζεται στα δυο συστήματα συντεταγμένων $\varphi = \varphi(x, y)$ και $\varphi = \varphi(\xi, \eta)$. Οι παράγωγοι της φ ως προς x, y δίνονται από τις σχέσεις:

$$\left.\begin{aligned} \frac{\partial \varphi}{\partial x} &= \frac{\partial \varphi}{\partial \xi} * \frac{\partial \xi}{\partial \eta} + \frac{\partial \varphi}{\partial \eta} * \frac{\partial \eta}{\partial x} \\ \frac{\partial \varphi}{\partial y} &= \frac{\partial \varphi}{\partial \xi} * \frac{\partial \xi}{\partial y} + \frac{\partial \varphi}{\partial \eta} * \frac{\partial \eta}{\partial y} \end{aligned}\right| \tag{2.48}$$

Εάν οι παράγωγοι της φ ως προς x, y δεν είναι δυνατόν να εκφραστούν με μια αναλυτική σχέση, όπως εκφράζονται οι παράγωγοι της φ ως προς ξ, η, τότε ο υπολογισμός των $\varphi_{,x}$, $\varphi_{,y}$ γίνεται έμμεσα με την παραγώγιση της συνάρτησης φ ως προς ξ, η:

$$\left.\begin{aligned} \frac{\partial \varphi}{\partial \xi} &= \frac{\partial \varphi}{\partial x} * \frac{\partial x}{\partial \xi} + \frac{\partial \varphi}{\partial y} * \frac{\partial y}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} &= \frac{\partial \varphi}{\partial x} * \frac{\partial x}{\partial \eta} + \frac{\partial \varphi}{\partial y} * \frac{\partial y}{\partial \eta} \end{aligned}\right| \Leftrightarrow \begin{Bmatrix} \dfrac{\partial \varphi}{\partial \xi} \\ \dfrac{\partial \varphi}{\partial \eta} \end{Bmatrix}$$

$$= \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} * \begin{Bmatrix} \dfrac{\partial \varphi}{\partial x} \\ \dfrac{\partial \varphi}{\partial y} \end{Bmatrix} \Leftrightarrow \begin{Bmatrix} \dfrac{\partial \varphi}{\partial \xi} \\ \dfrac{\partial \varphi}{\partial \eta} \end{Bmatrix} = [J] * \begin{Bmatrix} \dfrac{\partial \varphi}{\partial x} \\ \dfrac{\partial \varphi}{\partial y} \end{Bmatrix} \tag{2.49}$$

Όπου $[J]$ είναι το Ιακωβιανό μητρώο το οποίο προκύπτει αναλυτικά από την παραγώγιση των σχέσεων (2.43) ως προς ξ, η.
Το Ιακωβιανό μητρώο δίνεται από τη σχέση:

$$[J] = \begin{bmatrix} N_{1,\xi} & N_{2,\xi} & N_{3,\xi} & N_{4,\xi} \\ N_{1,\eta} & N_{2,\eta} & N_{3,\eta} & N_{4,\eta} \end{bmatrix} * \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} = [D_N] * \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \quad (2.50)$$

Όπου:

$$[D_N] = \frac{1}{4} * \begin{bmatrix} -(1-\eta) & 1-\eta & 1+\eta & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & 1+\xi & 1-\xi \end{bmatrix} \quad (2.51)$$

Προκειμένου να υπολογιστούν οι ζητούμενες παράγωγοι της συνάρτησης φ ως προς x, y δεν έχουμε παρά να λύσουμε την εξίσωση (2.49) ως προς τις συναρτήσεις αυτές:

$$\begin{Bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{Bmatrix} = [J]^{-1} * \begin{Bmatrix} \frac{\partial \varphi}{\partial \xi} \\ \frac{\partial \varphi}{\partial \eta} \end{Bmatrix} \quad (2.52)$$

Με το μητρώο $[J]^{-1}$ να δίνεται από τη σχέση:

$$[J]^{-1} = \begin{bmatrix} J_{11}^* & J_{12}^* \\ J_{21}^* & J_{22}^* \end{bmatrix} = \frac{1}{\det[J]} * \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \quad (2.53)$$

Όπου:

$$\det[J] = J_{11} * J_{22} - J_{21} * J_{12} \quad (2.54)$$

Και J_{ij} είναι τα στοιχεία του Ιακωβιανού μητρώου και J_{ij}^* είναι τα στοιχεία του $[J]^{-1}$. Στη γενική περίπτωση τα στοιχεία του Ιακωβιανού μητρώου είναι συναρτήσεις των ξ και η, στην περίπτωση όμως ορθογωνικών ισοπαραμετρικών στοιχείων είναι ανεξάρτητα των ξ και η και δίνονται από σταθερές ποσότητες που σχετίζονται με τη γεωμετρία του στοιχείου.

3. Μητρώο παραμόρφωσης

Από τη σχέση (2.52) προκύπτουν οι παράγωγοι των μετατοπίσεων u, v ως προς x, y με την αντικατάσταση της συνάρτησης φ με τις u, v. Ο συνδυασμός των σχέσεων (2.47) και (2.52), (2.53) δίνει τη σχέση:

$$\{\varepsilon\} = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix} = \frac{1}{\det[J]} * \begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix} * \begin{Bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} \\ \frac{\partial v}{\partial \eta} \end{Bmatrix} \quad (2.55)$$

ή

$$\underset{(3\times1)}{\{\varepsilon\}} = \underset{(3\times4)}{[B_1]} * \underset{(4\times1)}{\{u_{,\xi}\}} \quad (2.56)$$

Οι μερικές παράγωγοι των u, v ως προς x, h προκύπτουν από τις σχέσεις (2.45) και (2.46) και εκφράζονται συναρτήσει των επικόμβιων μετατοπίσεων μέσω της σχέσης:

$$\begin{Bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} \\ \frac{\partial v}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} \end{bmatrix} * \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} \tag{2.57}$$

ή

$$\underset{(4\times1)}{\{u_{,\xi}\}} = \underset{(4\times8)}{[B_2]} * \underset{(8\times1)}{\{d\}} \tag{2.58}$$

Το μητρώο $[B_2]$ σε συνδυασμό με το μητρώο $[D_N]$ της σχέσης (2.51) εκφράζεται ως εξής:

$$[B_2] = \frac{1}{4} * \begin{bmatrix} -(1-\eta) & 0 & 1-\eta & 0 & 1+\eta & 0 & -(1+\eta) & 0 \\ -(1-\xi) & 0 & -(1+\xi) & 0 & 1+\xi & 0 & 1-\xi & 0 \\ 0 & -(1-\eta) & 0 & 1-\eta & 0 & 1+\eta & 0 & -(1+\eta) \\ 0 & -(1-\xi) & 0 & -(1+\xi) & 0 & 1+\xi & 0 & 1-\xi \end{bmatrix} \tag{2.59}$$

Με αντικατάσταση της άνω σχέσης στη σχέση (2.56) προκύπτει το μητρώο παραμορφώσεως του τετραπλευρικού ισοπαραμετρικού στοιχείου επίπεδης έντασης παραμόρφωσης τεσσάρων κόμβων:

$$\underset{(3\times1)}{\{\varepsilon\}} = \underset{(3\times4)}{[B_1]} * \underset{(4\times8)}{[B_2]} * \underset{(8\times1)}{\{d\}} = \underset{(3\times8)}{[B]} * \underset{(8\times1)}{\{d\}} \tag{2.60}$$

4. Μητρώο δυσκαμψίας

Η γενική ολοκληρωτική σχέση του μητρώου δυσκαμψίας για το ισοπαραμετρικό στοιχείο επίπεδης ελαστικότητας τεσσάρων κόμβων γράφεται:

$$\underset{(8\times8)}{[k]} = \int_{A_e} \underset{(8\times3)}{[B]^T} * \underset{(3\times3)}{[E]} * \underset{(3\times8)}{[B]} * t * dA_e \tag{2.61}$$

Όπου $[E]$ είναι το μητρώο ελαστικότητας, ανάλογα με την εντατική κατάσταση, επίπεδη ένταση ή επίπεδη παραμόρφωση, και t είναι το πάχος του στοιχείου που στην περίπτωση αυτή θεωρείται σταθερό. Λαμβάνοντας υπόψη ότι το dA_e ισούται με

$$dA_e = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} * d\xi * d\eta = \det[J] * d\xi * d\eta \tag{2.62}$$

προκύπτει η τελική έκφραση του μητρώου δυσκαμψίας εκφρασμένου ως προς τις φυσικές του συντεταγμένες:

$$[k] = \int_{-1}^{1} \int_{-1}^{1} [B(\xi,\eta)]^T * [E] * [B(\xi,\eta)] * t * \det[J] * d\xi * d\eta \tag{2.63}$$

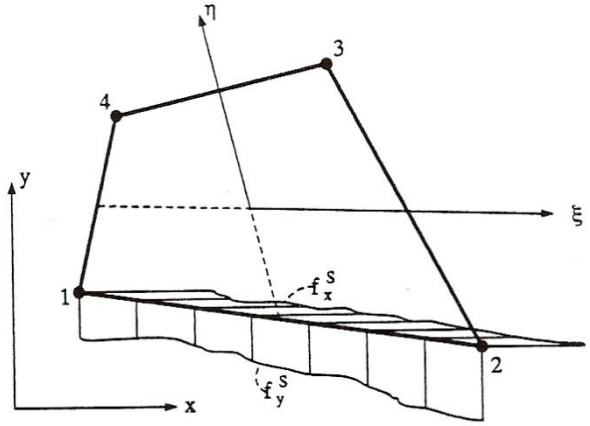
5. Ισοδύναμες δράσεις

Τα διανύσματα των ισοδύναμων δράσεων του στοιχείου δίνονται από το ολοκλήρωμα:

$$\{r\} = \int_{-1}^{1} \int_{-1}^{1} ([B]^T * [E] * \{\varepsilon_0\} - [B]^T * \{\sigma_0\} + [N]^T * \{f^V\}) * \det[J] * t * d\xi * d\eta + \int_{S_e} [N^S]^T * \{f^S\} * dS_e \tag{2.64}$$

Όπου:

$$[N^S] = \frac{1}{2} * \begin{bmatrix} 1-\xi & 0 & 1+\xi & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\xi & 0 & 1+\xi & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.65)$$



Εικόνα 1.6: Κατανεμημένη φόρτιση κατά μήκος μιας πλευράς τετραπλευρικού στοιχείου

Προκειμένου να υπολογιστεί το επιφανειακό ολοκλήρωμα της σχέσης (1.65), απαιτείται η έκφραση της διαφορικής επιφάνειας dS_e ως προς ξ, η. Εάν $t(\xi)$ είναι το πάχος του στοιχείου στη θέση ξ τότε $dS_e = t(\xi) * dl$, όπου dl είναι το στοιχειώδες μήκος στην πλευρά 1-2, σχήμα 1.6, το οποίο ορίζεται από τη σχέση:

$$dl = \det[J^S] * d\xi \quad (2.66)$$

Όπου:

$$\det[J^S] = \sqrt{\left(\frac{\partial x^S}{\partial \xi}\right)^2 + \left(\frac{\partial y^S}{\partial \xi}\right)^2} \quad (2.67)$$

Οι παράγωγοι $\frac{\partial x^S}{\partial \xi}$, $\frac{\partial y^S}{\partial \xi}$ προκύπτουν από τις σχέσεις (2.58) με την αντικατάσταση των u, v με x, y οι οποίες ισχύουν και για τις συντεταγμένες κατά μήκος της πλευράς 1-2. Έτσι έχουμε:

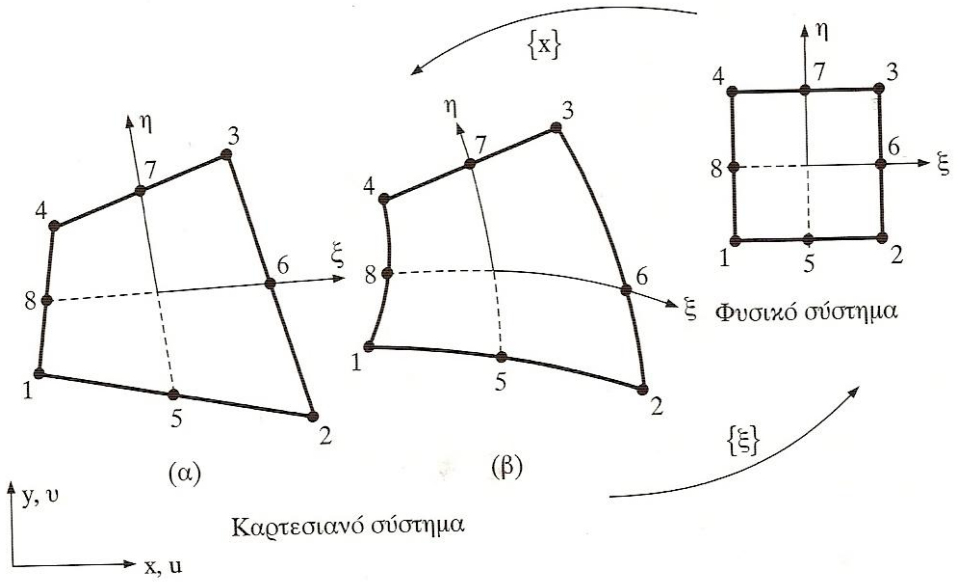
$$\left.\begin{cases} \frac{\partial x^S}{\partial \xi} = \frac{x_2^S - x_1^S}{2} \\ \frac{\partial y^S}{\partial \xi} = \frac{y_2^S - y_1^S}{2} \end{cases}\right| \quad (2.68)$$

Επομένως το επιφανειακό ολοκλήρωμα της σχέσης (1.65) γίνεται:

$$\{r_S\} = \int_{-1}^{1} [N^S]^T * \{f^S\} * t(\xi) * \det[J^S] * d\xi \quad (2.69)$$

1.4.3. Τετραπλευρικό ισοπαραμετρικό στοιχείο επίπεδης παραμόρφωσης οκτώ κόμβων (στοιχείο serendipity)

Στην *Εικόνα 1.7* φαίνεται το τετραπλευρικό στοιχείο οκτώ κόμβων στα δυο συστήματα συντεταγμένων. Οι δυο παραλλαγές (α) και (β) αποδίδουν δυο δυνατότητες της απαραμόρφωτης γεωμετρίας και των θέσεων των ενδιάμεσων κόμβων του στοιχείου. Η περίπτωση (α) αντιστοιχεί σε ένα στοιχείο που στην απαραμόρφωτη κατάσταση έχει τέσσερις πλευρές ευθύγραμμες. Η περίπτωση (β) απεικονίζει ένα στοιχείο που στην απαραμόρφωτη κατάσταση έχει τρεις καμπυλόγραμμες και μια ευθύγραμμη πλευρά, καθώς και έναν ενδιάμεσο κόμβο 6 που βρίσκεται στο μέσον της πλευράς 2-3.



Εικόνα 1.7: Τετραπλευρικά ισοπαραμετρικά στοιχεία οχτώ κόμβων: (α) ευθύγραμμες πλευρές στην απαραμόρφωτη κατάσταση και (β) καμπυλόγραμμες και ευθύγραμμες πλευρές στην απαραμόρφωτη κατάσταση και ενδιάμεσους κομβους που δεν κείνται στο μέσο των πλευρών

1. Συναρτήσεις σχήματος

Οι εκφράσεις των συντεταγμένων ενός τυχαίου σημείου του στοιχείου ορίζονται από πολυώνυμα δευτέρου βαθμού ως προς και αφού τρεις είναι οι κόμβοι σε κάθε πλευρά του στοιχείου. Κατά συνέπεια η πολυωνυμική απεικόνιση θα έχει τη μορφή:

$$\begin{aligned} x &= \alpha_1 + \alpha_2 * \xi + \alpha_3 * \eta + \alpha_4 * \xi * \eta + \alpha_5 * \xi^2 + \alpha_6 * \eta^2 + \alpha_7 * \xi^2 * \eta \\ &\quad + \alpha_8 * \eta^2 * \xi \\ y &= \alpha_9 + \alpha_{10} * \xi + \alpha_{11} * \eta + \alpha_{12} * \xi * \eta + \alpha_{13} * \xi^2 + \alpha_{14} * \eta^2 + \alpha_{15} \\ &\quad * \xi^2 * \eta + \alpha_{16} * \eta^2 * \xi \end{aligned} \tag{2.70}$$

ή

$$\begin{aligned} x &= [1 \quad \xi \quad \eta \quad \xi * \eta \quad \xi^2 \quad \eta^2 \quad \xi^2 * \eta \quad \xi * \eta^2] * \{\alpha_x\} \\ y &= [1 \quad \xi \quad \eta \quad \xi * \eta \quad \xi^2 \quad \eta^2 \quad \xi^2 * \eta \quad \xi * \eta^2] * \{\alpha_y\} \end{aligned} \tag{2.71}$$

Με ανάλογη σχέση, λόγω ισοπαραμετρικής θεώρησης εκφράζεται και το πεδίο των μετατοπίσεων του στοιχείου στο καρτεσιανό σύστημα:

$$\begin{aligned} u &= [1 \quad \xi \quad \eta \quad \xi * \eta \quad \xi^2 \quad \eta^2 \quad \xi^2 * \eta \quad \xi * \eta^2] * \{b_x\} \\ v &= [1 \quad \xi \quad \eta \quad \xi * \eta \quad \xi^2 \quad \eta^2 \quad \xi^2 * \eta \quad \xi * \eta^2] * \{b_y\} \end{aligned} \tag{2.72}$$

Όπου:

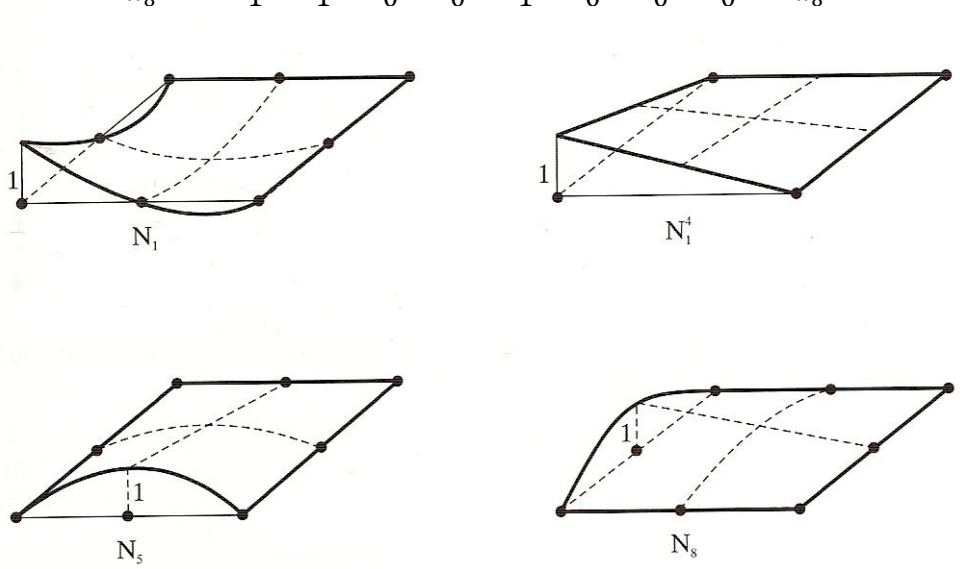
$$\begin{aligned} b_x &= [b_1 \quad b_2 \quad \dots \quad b_8 \ ]^T \\ b_y &= [b_9 \quad b_{10} \quad \dots \quad b_{16}]^T \end{aligned} \tag{2.73}$$

Ο υπολογισμός των γενικευμένων συντεταγμένων $\{a\}$ ή $\{b\}$ ακολουθεί την πάγια διαδικασία:

$$\begin{aligned} x_1 &= a_1 - a_2 - a_3 + a_4 + a_5 + a_6 - a_7 - a_8 && (\xi = -1, \eta = -1) \\ x_2 &= a_1 + a_2 - a_3 - a_4 + a_5 + a_6 - a_7 + a_8 && (\xi = \ \ 1, \eta = -1) \\ x_3 &= a_1 - a_2 + a_3 - a_4 + a_5 + a_6 + a_7 - a_8 && (\xi = \ \ 1, \eta = \ \ 1) \\ x_4 &= a_1 - a_2 + a_3 - a_4 + a_5 + a_6 + a_7 - a_8 && (\xi = -1, \eta = \ \ 1) \\ x_5 &= a_1 \quad - a_3 \quad + a_6 && (\xi = \ \ 0, \eta = -1) \\ x_6 &= a_1 + a_2 \quad + a_5 && (\xi = \ \ 1, \eta = \ \ 0) \\ x_7 &= a_1 \quad + a_3 \quad + a_6 && (\xi = \ \ 0, \eta = \ \ 1) \\ x_4 &= a_1 - a_2 \quad + a_5 && (\xi = -1, \eta = \ \ 0) \end{aligned} \tag{2.74}$$

ή

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{Bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{Bmatrix} \quad (2.75)$$



Εικόνα 1.8: Συναρτήσεις σχήματος τετραπλευρικού ισοπαραμετρικού στοιχείου οχτώ κόμβων (N_1^4 είναι συνάρτηση σχήματος τετρακομβικού στοιχείου).

Με την αντικατάσταση της λύσης ως προς $\{a_x\}$ στη σχέση (2.72) παίρνουμε την έκφραση της συντεταγμένης x ενός τυχαίου σημείου του στοιχείου ως προς τις συντεταγμένες $x_i, i = 1, 2, \dots, 8$ των οκτώ κόμβων του στοιχείου. Μετά την εκτέλεση των σχετικών πράξεων προκύπτει η σχέση:

$$x = [N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \quad N_7 \quad N_8] * \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{Bmatrix} \quad (2.76)$$

Όπου

$$\begin{aligned} N_1 &= \frac{1}{4} * (1-\xi) * (1-\eta) - \frac{1}{2} * (N_8 + N_5) \\ N_2 &= \frac{1}{4} * (1+\xi) * (1-\eta) - \frac{1}{2} * (N_5 + N_6) \\ N_3 &= \frac{1}{4} * (1+\xi) * (1+\eta) - \frac{1}{2} * (N_6 + N_7) \\ N_4 &= \frac{1}{4} * (1-\xi) * (1+\eta) - \frac{1}{2} * (N_7 + N_8) \\ N_5 &= \frac{1}{2} * (1-\xi^2) * (1-\eta) \\ N_6 &= \frac{1}{2} * (1+\xi) * (1-\eta^2) \\ N_7 &= \frac{1}{2} * (1-\xi^2) * (1+\eta) \end{aligned} \quad (2.77)$$

$$N_8 = \frac{1}{2} * (1-\xi) * (1-\eta^2)$$

είναι οι συναρτήσεις σχήματος. Στην *Εικόνα 1.8* απεικονίζονται οι γραφικές παραστάσεις των συναρτήσεων σχήματος N_1 , N_5 , N_8 καθώς και η N_4 του τετρακομβικού στοιχείου.

2. Μητρώο παραμορφώσεως

Το μητρώο παραμορφώσεως $[B]$ προκύπτει με ανάλογη διαδικασία με εκείνη του τετρακομβικού στοιχείου. Η γενική έκφραση του μητρώου παραμορφώσεως δίνεται από τη σχέση (2.60) όπου $[B_1]$ είναι το μητρώο που συνδέει τις ανηγμένες παραμορφώσεις με τις παραγώγους των μετατοπίσεων ως προς ξ, η, ενώ $[B_2]$ είναι το μητρώο που συνδέει τις παραγώγους u, v ως προς ξ, η με τις επικόμβιες μετατοπίσεις. Το μητρώο $[B_1]$ δίνεται από τη σχέση (2.56) και το μητρώο $[B_2]$ από τη σχέση:

$$\begin{Bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial v}{\partial \xi} \\ \frac{\partial v}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & \cdots & N_{8,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & \cdots & N_{8,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & \cdots & 0 & N_{8,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & \cdots & 0 & N_{8,\eta} \end{bmatrix} * \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_8 \\ v_8 \end{Bmatrix}$$

ή

$$\underset{(4\times1)}{\{u_{,\xi}\}} = \underset{(4\times16)}{[B_2]} * \underset{(16\times1)}{\{d\}} \tag{2.78}$$

Κατά συνέπεια το μητρώο παραμορφώσεως του οκτακομβικού στοιχείου επίπεδης έντασης-παραμόρφωσης δίνεται από τη σχέση:

$$\underset{(3\times16)}{[B]} = \underset{(3\times4)}{[B_1]} * \underset{(4\times16)}{[B_2]} \tag{2.79}$$

3. Μητρώο δυσκαμψίας

Η έκφραση του μητρώου στιβαρότητας δίνεται από το ολοκλήρωμα:

$$\underset{(16\times16)}{[k]} = \int_{A_e} \underset{(16\times3)}{[B]^T} * \underset{(3\times3)}{[E]} * \underset{(3\times16)}{[B]} * t * dA_e \tag{2.80}$$

ή

$$[k] = \int_{-1}^{1}\int_{-1}^{1} [B(\xi,\eta)]^T * [E] * [B(\xi,\eta)] * t * \det[J] * d\xi * d\eta \tag{2.81}$$

Όπου $[E]$ είναι το μητρώο ελαστικότητας αναλόγως την περίπτωση του προβλήματος που εξετάζουμε.

1.4.4. Αριθμητική ολοκλήρωση

Προκειμένου να υπολογιστεί το μητρώο δυσκαμψίας ισοπαραμετρικού στοιχείου, η αναλυτική ολοκλήρωση γίνεται εξαιρετικά δυσχερής έως αδύνατη, ιδιαίτερα σε στοιχεία ανωτέρου βαθμού. Για το λόγο αυτό καταφεύγουμε σε αριθμητική ολοκλήρωση.

Τα μητρώα που πρόκειται να ολοκληρωθούν σε δυο διαστάσεις, έχουν τη μορφή $[F(\xi,\eta)]$ και το αντίστοιχο μητρώο δυσκαμψίας δίνεται από το ολοκλήρωμα:

$$\int [F(\xi,\eta)] * d\xi * d\eta \tag{2.82}$$

Ο τύπος αριθμητικής ολοκλήρωσης του πιο πάνω ολοκληρώματος έχει τη μορφή:

$$\int [F(\xi,\eta)] * d\xi * d\eta = \sum_{i,j} \alpha_{ij} * \left[F\left(\xi_i,\eta_j\right)\right] + [e_n] \tag{2.83}$$

Όπου το άθροισμα περιλαμβάνει όλα τα σημεία του στοιχείου που καθορίζονται από τους δείκτες i,j, οι συντελεστές α_{ij} είναι οι συντελεστές βάρους και $\left[F\left(\xi_i,\eta_j\right)\right]$ είναι τα μητρώα $[F(\xi,\eta)]$ υπολογισμένα στις θέσεις i,j του στοιχείου.

Το μητρώο $[e_n]$ είναι το μητρώο σφάλματος το οποίο στην πράξη δεν υπολογίζεται. Έτσι, το ολοκλήρωμα (1.71) προσεγγίζεται από το άθροισμα:

$$\int [F(\xi,\eta)] * d\xi * d\eta \approx \sum_{i,j} \alpha_{ij} * \left[F\left(\xi_i,\eta_j\right)\right] \tag{2.84}$$

Ανάλογα με τον τύπο του στοιχείου και τις χαρακτηριστικές του ιδιότητες επιλέγεται η κατάλληλη μέθοδος αριθμητικής ολοκλήρωσης και το πλήθος n των σημείων ολοκλήρωσης.

1.4.4.1. Αριθμητική ολοκλήρωση Gauss

Μια αποτελεσματική μέθοδος αριθμητικής ολοκλήρωσης, στην οποία τόσο οι θέσεις των σημείων ολοκλήρωσης όσο και οι συντελεστές βάρους έχουν βελτιστοποιηθεί, είναι η ολοκλήρωση Gauss, την οποία και παρουσιάζουμε για την περίπτωση ολοκλήρωσης σε δύο διαστάσεις.

Προκειμένου να ολοκληρώσουμε μια συνάρτηση δυο μεταβλητών επαναλαμβάνουμε τη διαδικασία αριθμητικής ολοκλήρωσης σε κάθε διεύθυνση. Έτσι, για τον υπολογισμό του ολοκληρώματος $\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta) * d\xi * d\eta$, ολοκληρώνουμε πρώτα ως προς τη μια μεταβλητή και μετά ως προς την άλλη:

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta) * d\xi * d\eta = \int_{-1}^{1} g(\eta) * d\eta \tag{2.85}$$

Όπου:

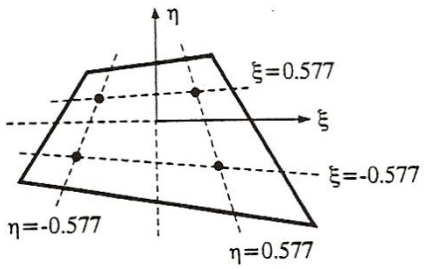
$$g(\eta) = \int_{-1}^{1} f(\xi,\eta) * d\xi = \sum_{i=1}^{n} \alpha_i * f(\xi_i,\eta) \tag{2.86}$$

Με την αντικατάσταση της (1.75) στην (1.74) προκύπτει η σχέση:

$$I = \sum_{j=1}^{n} \alpha_j * \sum_{i=1}^{n} \alpha_i * f\left(\xi_i,\eta_j\right) = \sum_{j=1}^{n}\sum_{i=1}^{n} \alpha_i * \alpha_j * f\left(\xi_i,\eta_j\right) \tag{2.87}$$

Η ολοκλήρωση δεν είναι απαραίτητο να γίνεται με τον ίδιο αριθμό σημείων και στις δυο διευθύνσεις.

Η αριθμητική ολοκλήρωση Gauss εφαρμόζεται άμεσα στα ισοπαραμετρικά ραβδωτά και τετραπλευρικά στοιχεία στα οποία τα όρια ολοκλήρωσης είναι -1 και 1. Έτσι, για τον υπολογισμό του μητρώου δυσκαμψίας και του διανύσματος των ισοδύναμων δράσεων απαιτείται ο υπολογισμός των στοιχείων του μητρώου δυσκαμψίας και του διανύσματος των ισοδύναμων δράσεων στα σημεία Gauss. Στην εργασία αυτή χρησιμοποιήθηκε η παράσταση που φαίνεται στην *Εικόνα 1.9*.



n=2
(βαθμός ακρίβειας 3)

Εικόνα 1.9: Σχηματική παράσταση των θέσεων σημείων Gauss

Η θέση οποιουδήποτε σημείου P Gauss στο καρτεσιανό σύστημα προκύπτει από τις συναρτήσεις σχήματος:

$$x_P = \sum_i N_i * (\xi_P, \eta_P) * x_i$$
$$y_P = \sum_i N_i * (\xi_P, \eta_P) * y_i \quad (2.88)$$

<u>Κεφάλαιο 2:</u>
Προγραμματιστικές παρατηρήσεις

2.1. Εισαγωγικές προγραμματιστικές παρατηρήσεις

Σε κάθε μεγάλη εργασία που αφορά τη δημιουργία ενός μεγάλου προγράμματος είναι λογικό να αναπτύσσονται κώδικες ιδιαίτερα πολύπλοκοι και αρκετά δύσκολοι τόσο στη σύλληψη όσο και στην υλοποίηση. Έχει λοιπόν σημασία να γνωρίζουμε από πριν κάποιες αντιστοιχίες των τμημάτων αυτών του κώδικα με άλλους, ισοδύναμους αλγορίθμους, οι οποίοι έχουν μια πιο προσιτή φυσική σημασία για τα δεδομένα του εκάστοτε προγραμματιστή. Να σημειωθεί ότι αυτή η αντιστοίχιση είναι πιο σημαντική για εκείνον που διαβάζει κάποιον κώδικα που προϋπάρχει, παρά για τον συγγραφέα, καθώς ο δεύτερος γνωρίζει, ή τουλάχιστον πρέπει να γνωρίζει, ακριβώς τι κάνει και το έχει μελετήσει από πριν, έχοντας αφιερώσει σε αυτό αρκετό χρόνο, οπότε δεν μένει, παρά να είναι για αυτόν ο κώδικάς του, κατανοητός.

Γενικά υπάρχουν αρκετές συμβουλές και μέθοδοι που μπορούν να κάνουν τη συγγραφή, αλλά και τη μελέτη ενός κώδικα πιο ευανάγνωστη. Κάποιες από αυτές αναλύονται παρακάτω. Αυτές οι συμβουλές όμως μπορούν να εφαρμοστούν σε project που έχουν υλοποιηθεί με τη μέθοδο του «spaghetti programming». Κατά τη μέθοδο αυτή ένα πρόγραμμα κατασκευάζεται και λειτουργεί open source και επιδέχεται προσθήκες και βελτιώσεις από τους χρήστες του. Κατά καιρούς οι πιο πλήρης εκδόσεις δημοσιεύονται ενσωματωμένες στο αρχικό πρόγραμμα το οποίο ακολουθεί πάλι την ίδια πορεία και εμπλουτίζεται συνεχώς. Το πρόβλημα είναι ότι ο κάθε προγραμματιστής που θα ενσωματώσει κάποια πρόσθετη λειτουργία, δεν θα το κάνει με κριτήριο την συγγραφική ορθότητα του κώδικα, ούτε το κατά πόσο ο πηγαίος κώδικας του εμπλουτισμένου προγράμματος θα είναι ευανάγνωστος. Το βασικό κριτήριο είναι να μπορεί να εκτελεστεί η τρέχουσα εργασία, ακόμα και αν αυτό γίνεται με σκόρπιες προσθήκες κώδικα σε διάφορα σημεία του προγράμματος. Όπως είναι εύκολα αντιληπτό, εάν αυτό συμβεί αρκετές φορές, θα προκύψει ένα τελικό project πλούσιο σε δυνατότητες μεν, δυσνόητο στην ανάλυση του πηγαίου κώδικά του δε. Τα μειονεκτήματα που απορρέουν από αυτή την αλληλουχία είναι δύο. Το μεν πρώτο και προφανές είναι ότι δυσκολεύει συνεχώς η δυνατότητα κατανόησης του προγράμματος από έναν προγραμματιστή, το δε δεύτερο, απόρροια του πρώτου, είναι ότι από κάποιο σημείο και μετά ο εμπλουτισμός του προγράμματος θα σταματήσει και θα χρειαστεί επανασυγγραφή ώστε να μπει σε τάξη ο κώδικας, πράγμα που και πάλι δεν θα είναι αρκετά εύκολο.

Ασφαλώς το FEAP ακολουθεί την πορεία του spaghetti programming έχοντας όλα τα πλεονεκτήματα και μειονεκτήματά της. Στην επόμενη ενότητα θα προσπαθήσουμε να εξηγήσουμε κάποιες χρήσιμες ισοδυναμίες κώδικά και ορισμένες συμβουλές που μπορούν να βοηθήσουν τον επίδοξο αναγνώστη και συγγραφέα του FEAP.

2.2. Μετατροπές

Στο *Παράρτημα Γ* παρουσιάζουμε έναν πίνακα στον οποίο φαίνονται κάποιες μετατροπές κώδικα. Στην ουσία οι μέθοδοι με την οποία μπορεί να τους υλοποιήσει κανείς προγραμματιστικά είναι εναλλακτικές η μία έναντι της άλλης, αυτό όμως δεν ισχύει όταν το πλήθος `N` δεν είναι γνωστό, ή είναι ένας πολύ μεγάλος αριθμός. Στο κεφάλαιο αυτό θα αναλύσουμε τις αντιστοιχίες μία προς μία.

Στη μετατροπή 1 παρατηρούμε απλά `N` συνεχόμενες δομές επιλογής τύπο `IF – END IF`. Ασφαλώς μπορούν αυτές να αντικατασταθούν από μία η οποία να περιέχεται σε ένα βρόχο `DO – END DO` που εκτελείται `N` φορές. Τα ίδιο μοτίβο υπάρχει και στη μετατροπή 2 όπου έχουμε `N` συνεχόμενες δομές επιλογής τύπου `IF – ELSE – END IF`, οι οποίες μπορούν να τοποθετηθούν εντός ενός βρόχου `DO – END DO` ο οποίος απλά να εκτελείται `N` φορές.

Και ενώ μέχρι στιγμής οι μετατροπές 1 και 2 παρουσιάστηκαν για λόγους πληρότητας, στη μετατροπή 3 αρχίζουν τα πράγματα να αποκτούν περισσότερο ενδιαφέρον. Στην περίπτωση αυτή έχουμε μια δομή επιλογής τύπου `IF – ELSE IF – ELSE – END IF` που αποτελείται όμως από ένα `IF`, `N-1 ELSE IF` και ένα `ELSE`. Και το ζητούμενο στην περίπτωση αυτή είναι πώς θα υλοποιήσουμε τα `N-1` φορές επαναλαμβανόμενα `ELSE IF`. Αυτό φαίνεται στον ισοδύναμο κώδικα στη δεύτερη στήλη. Στην ουσία η συγκεκριμένη δομή επιλογής διατρέχει με τη σειρά όλες τις συνθήκες και όταν βρει τη σωστή, δηλαδή την αληθή, εκτελεί τις αντίστοιχες εντολές. Αν δεν βρει καμία αληθή τότε εκτελεί τις εντολές που βρίσκονται στο `ELSE`. Η όλη διαδικασία θυμίσει μοντέλο κώδικα σειριακής αναζήτησης με μοναδικές εγγραφές, που σταματάει δηλαδή όταν βρεθεί η επιθυμητή τιμή. Εάν ο μετρητής, δηλαδή στην περίπτωσή μας το `I` έχει μετά την έξοδο από την επανάληψη τιμές από το `1` έως και το `N` σημαίνει ότι η αντίστοιχη συνθήκη είναι αληθής, ενώ αν έχει τιμή `N+1` σημαίνει ότι καμία συνθήκη δεν είναι αληθής και έτσι εκτελούνται οι εντολές που αντιστοιχούν στο `ELSE`.

Η μετατροπή 4 αφορά `N` δομές επιλογής τύπου `IF – END IF` εμφωλευμένες η εκάστοτε επόμενη στην αμέσως προηγούμενη. Ο ισοδύναμος κώδικας φαίνεται στη δεύτερη στήλη. Αυτό που κάνει στην πράξη αυτή η διάταξη δομών επιλογής είναι να ελέγχει την εκάστοτε συνθήκη `CONDITION(I)` και αν είναι αληθής αν εκτελεί τις αμέσως επόμενες εντολές `COMMANDS1(I)`. Αν και όταν συναντήσει ψευδή συνθήκη δεν εκτελεί τις εντολές από εκεί και κάτω, αλλά προτού περάσει κάθε `END IF` εκτελεί και τις εντολές `COMMANDS2(I)`. Η ίδια εργασία παρουσιάζεται στις επαναλήψεις της δεύτερης στήλης.

Η μετατροπή 5 είναι παρόμοια με την προηγούμενη. Στην περίπτωση αυτή έχουμε `N` δομές επιλογής τύπου `IF – ELSE – END IF` εμφωλευμένες η εκάστοτε επόμενη στην αμέσως προηγούμενη. Η λογική εδώ είναι σαν την προηγούμενη. Όσο οι συνθήκες `CONDITION(I)` είναι αληθείς εκτελούνται οι αντίστοιχες εντολές `COMMANDS1(I)`, αλλά όταν κάποια είναι λάθος εκτελούνται οι αντίστοιχες εντολές `COMMANDS3(I)` του `ELSE`. Μετά βγαίνοντας από τα `IF` με τη σειρά από μέσα προς τα έξω και πριν το κάθε `END IF`, εκτελούνται και οι αντίστοιχες εντολές `COMMANDS2(I)`. Η διαδικασία αυτή φαίνεται επαναληπτικά στη δεύτερη στήλη του πίνακα.

Η μετατροπή 6 είναι η πρώτη που αναφέρεται σε επαναλήψεις επαναλήψεων. Στην ουσία έχουμε `N` συνεχόμενα `DO – END DO` το ένα κάτω από το άλλο. Αυτά μπορούν να υλοποιηθούν όπως φαίνεται στη δεύτερη στήλη με δύο δομές επανάληψης τύπου `DO – END DO` το ένα εμφωλευμένο μέσα στο άλλο.

Οι μετατροπές 7 και 8 είναι από τα πιο ενδιαφέροντα σημεία καθότι λόγω αυτής λύνονται μία μεγάλη γκάμα προβλημάτων όπως του περιοδεύοντος πωλητή με μη καθορισμένο πλήθος πόλεων, διοφαντικές εξισώσεις με μη καθορισμένο πλήθος αθροιστών, κτλ. Αποτελείται από `N` επαναλήψεις τύπου `DO – END DO` εμφωλευμένες η εκάστοτε επόμενη στην αμέσως προηγούμενη. Για να καταρτίσουμε τον ισοδύναμο κώδικα σε αυτόν χρειάζεται να καταλάβουμε πώς λειτουργεί. Οι εμφωλευμένες λοιπόν επαναλήψεις αυτό που κάνουν είναι να εκτελούν επαναλαμβανόμενα τις εντολές `COMMANDS` τόσες φορές όσο είναι το γινόμενο που παράγοντές του είναι τα πλήθη της εκτέλεσης της κάθε επανάληψης. Αν δηλαδή η πρώτη επανάληψη εκτελείται N_1 φορές, η δεύτερη N_2, μέχρι τη Ν-οστή που εκτελείται N_N φορές, τότε οι εντολές `COMMANDS` εκτελούνται $N' = N_1 * N_2 * \ldots * N_N$ φορές. Προφανώς ισχύει ότι $N_i = ακέραιο\_μέρος\left(\frac{L(i)-K(i)}{M(i)}\right) + 1$. Αυτό όμως δεν σημαίνει ότι αρκεί να πάρουμε τις εντολές COMMANDS και να τις εντάξουμε σε ένα βρόχο `DO – END DO` που θα εκτελείται N' φορές, διότι δεν έχει σημασία μόνο το πλήθος των επαναλήψεων, αλλά και οι τιμές των δεικτών των επιμέρους επαναλήψεων. Αυτός άλλωστε είναι και ο λόγος που οι δείκτες σε έναν ή περισσότερους επαναληπτικούς βρόχους `DO – END DO` μπορούν να αυξάνονται

όχι μόνο με βήμα 1, αλλά με όποιο βήμα επιλέξει ο προγραμματιστής. Αρχικά λοιπόν κατασκευάζουμε ένα βρόχο που υπολογίζει το γινόμενο N'. Στη συνέχεια καταρτίζουμε ένα βρόχο `DO – END DO` ο οποίος εκτελείται N' φορές και τώρα το πρόβλημα που αντιμετωπίζουμε είναι πώς θα υπολογίσουμε τους δείκτες που εμφανίζονται στον πρώτο τρόπο υλοποίησης. Ενώ στον πρώτο τρόπο υλοποίησης οι δείκτες ήταν `I1, I2, I3,…. , IN`, στη δεύτερη υλοποίηση είναι `D(1), D(2), D(3),…. , D(N)`. Η τιμή του κάθε δείκτη υπολογίζεται μέσω ακέραιας διαίρεσης του αύξοντα αριθμού της επανάληψης, δηλαδή του δείκτη της επανάληψης στη δεύτερη υλοποίηση, με το γινόμενο του πλήθους των επαναλήψεων που ακολουθούν. Αν το υπόλοιπο της ακέραιας αυτής διαίρεσης είναι `0`, τότε η τιμή του δείκτη είναι το πηλίκο που μόλις υπολογίστηκε, αλλιώς είναι το πηλίκο αυξημένο κατά `1`. Οι υπολογισμοί αυτοί γίνονται σε μια επανάληψη τύπου `DO WHILE – END DO` εμφωλευμένη στην αρχική. Η μετατροπή 7 είναι η πιο απλή εκδοχή αυτού που περιγράφουμε με τα όρια των βρόχων `DO – END DO` να κυμαίνονται από 1 έως `K(I), I=1,2,…., N` με βήμα `1`, ενώ η μετατροπή 7 γίνεται γενικευμένα για με τα όρια των επαναλήψεων να κυμαίνονται από `K(I)` έως `L(I)` με βήμα `M(I), I=1,2,…., N`.

Στην ίδια φιλοσοφία κινείται και η μετατροπή 9, η οποία αποτελείται από `N` επαναλήψεις τύπου `DO – END DO` εμφωλευμένη η εκάστοτε επόμενη στην αμέσως προηγούμενη, αλλά πριν και μετά από κάθε μία υπάρχουν και ομάδες εντολών `COMMANDS1(I)` και `COMMANDS2(I)`. Αρχικά υπολογίζονται οι δείκτες όπως και πριν και στη συνέχεια ελέγχεται από τις τιμές των δεικτών ποιες από τις εντολές `COMMANDS1(I)` πρέπει να εκτελεστούν. Στη συνέχεια το ίδιο γίνεται και για τις εντολές `COMMANDS2(I)`.

Η μετατροπή 10 αναφέρεται σε `N` εμφωλευμένες επαναλήψεις τύπου `DO WHILE – END DO` εμφωλευμένες η εκάστοτε επόμενη στην αμέσως προηγούμενη. Η υλοποίηση φαίνεται στη δεύτερη στήλη. Ο αριθμός της συνθήκης `CONDITION(I)` που ελέγχεται κάθε φορά είναι στη μεταβλητή `I`. Όταν ο έλεγχος μιας συνθήκης είναι αληθής η μεταβλητή αυξάνεται κατά `1` για να ελεγχθεί και η επόμενη και ούτω καθεξής. Όταν η μεταβλητή `I` γίνει `0`, σημαίνει πως όλες οι επαναλήψεις τελείωσαν.

Η μετατροπή 11 είναι η ίδια με την προηγούμενη, μόνο που εδώ παρεμβάλλονται και εντολές πριν και μετά από κάθε σύνολο επαναλήψεων `DO WHILE – END DO`. Οι εντολές που βρίσκονται επάνω είναι οι `COMMANDS1(I)`, ενώ αυτές που βρίσκονται κάτω είναι οι `COMMANDS2(I)`. Η υλοποίηση είναι αυτή που φαίνεται στη δεύτερη στήλη και ακολουθεί την ίδια φιλοσοφία με πριν. Ελέγχεται η συνθήκη `CONDITION (I)` και αν είναι αληθής τότε εκτελούνται οι εντολές `COMMANDS(I)` και η μεταβλητή `I` αυξάνεται κατά `1` για να ελεγχθεί η επόμενη συνθήκη. Αλλιώς εκτελούνται οι εντολές `COMMANDS2(I)` και η μεταβλητή `I` μειώνεται κατά `1` για να ελεγχθεί ξανά η προηγούμενη συνθήκη. Όταν η μεταβλητή `I` πάρει τιμή `0` τότε σημαίνει ότι το σύνολο των επαναλήψεων έχει τελειώσει.

Στην επόμενη ενότητα θα αναλύσουμε τα είδη των σφαλμάτων τα οποία αντιμετωπίζουμε κατά τη συγγραφή ενός κώδικα.

2.3. Σφάλματα

Τα σφάλματα που συναντούμε σε κάθε αριθμητική υπολογιστική εργασία είναι πέντε ειδών:

1. Round- off
2. Overflow
3. Underflow
4. Λόγω ταχύτητας σύγκλισης της αριθμητικής μεθόδου
5. Λόγω εγγενούς σφάλματος της θεωρίας ή μοντελοποίησης που χρησιμοποιούμε

Κάθε ένα από αυτά έχει τη δική του σημασία και χρειάζεται να γνωρίζουμε την παρουσία ή απουσία του στον κώδικά μας, άσχετα αν το αντιμετωπίζουμε ή όχι.

Τα round- off σφάλματα είναι τα πιο σημαντικά στον προγραμματισμό και για τον προγραμματιστή, αλλά και για τον χρήστη ενός προγράμματος και δεν μπορούν να αποφευχθούν. Αυτό σημαίνει ότι οι μεταβλητές που χρησιμοποιούμε δεν έχουν ακριβώς τις τιμές που θέλουμε, ακόμα και όταν τους τις έχουμε δώσει με εκχώρηση. Αυτό συμβαίνει γιατί στην πραγματικότητα ο τρόπος αποθήκευσης των μεταβλητών στη μνήμη του υπολογιστή γίνεται με εκθετικό τρόπο και το τελικό αποτέλεσμα είναι ότι η τελική τιμή που αποθηκεύεται είναι ελαφρώς διαφοροποιημένη από αυτή που δώσαμε στην αρχή. Αυτό έχει να κάνει και με τον τύπο των μεταβλητών που χρησιμοποιούμε. Για παράδειγμα ο τύπος double precision είναι πιο ακριβής από τον τύπο real, αλλά απαιτεί περισσότερη μνήμη για αποθήκευση. Αυτή η διαφορά στην ακρίβεια είναι και ένας λόγος που υπάρχουν τόσοι διαφορετικοί τύποι μεταβλητών. Στις περισσότερες κλασικές εφαρμογές δεν συνυπολογίζονται καθόλου, καθώς τα τελικά σφάλματα που προξενούν είναι συνήθως μικρά. Ο ρόλος τους όμως φαίνεται σε προβλήματα όπως μελέτες ευαισθησίας συστημάτων, ή προβλήματα διακριτών επιλογών όπου ακόμα και πολύ μικρά σφάλματα μπορεί να έχουν καταλυτικό ρόλο στο τελικό αποτέλεσμα. Για παράδειγμα, αν μας ενδιαφέρει να βρούμε αν μια εξίσωση $2^{ου}$ βαθμού έχει πραγματική λύση περισσότερο από τη λύση καθαυτή και η διακρίνουσα είναι πολύ κοντά στο μηδέν, τότε αντιλαμβανόμαστε ότι η λύση στο πρόβλημα δεν είναι η προφανής. Συνήθως τέτοια προβλήματα λύνονται μόνο με θεωρητικά μαθηματικά, ή με μετασχηματισμούς που ξεφεύγουν όμως από το σκοπό της εργασίας αυτής.

Εκτός από τα round- off errors υπάρχουν και τα overflow και underflow. Τα πρώτα συμβαίνουν όταν προσπαθούμε να αποθηκεύσουμε στη μνήμη έναν πολύ μεγάλο κατ' απόλυτη τιμή αριθμό, θετικό ή αρνητικό. Τα δεύτερα προκύπτουν όταν κάνουμε το ίδιο, αλλά για έναν πολύ μικρό κατ' απόλυτη τιμή αριθμό θετικό ή αρνητικό. Για παράδειγμα αν δοκιμάσουμε να αποθηκεύσουμε στη μνήμη τον αριθμό $\pm 8 \cdot 10^{400}$ τότε ο αριθμός είναι πολύ μεγάλος κατ' απόλυτη τιμή και θα προκαλέσει σφάλμα στη μνήμη (overflow), ενώ ο αριθμός $\pm 8 \cdot 10^{-400}$ θα προκαλούσε σφάλμα λόγω του ότι είναι κατ' απόλυτη τιμή πολύ μικρός (underflow). Αυτά τα όρια εντός των οποίων μια μεταβλητή μπορεί να πάρει τιμή ποικίλουν και αυτά αναλόγως τον τύπο μεταβλητής που χρησιμοποιείται και είναι ο δεύτερος λόγος για τον οποίον μπορεί να χρησιμοποιούνται πολλοί τύποι μεταβλητών για να εκφράσουν φαινομενικά το ίδιο πράγμα.

Τα σφάλματα που προκύπτουν από τη σύγκλιση της αριθμητικής μεθόδου που χρησιμοποιούμε υπάρχουν και αυτά πλέον των προηγούμενων. Είναι γνωστό ότι μία αριθμητική- επαναληπτική διαδικασία συγκλίνει προς το επιθυμητό αποτέλεσμα. Αυτό σημαίνει ότι το αποτέλεσμα δεν είναι ακριβές, αλλά μπορεί να προσδιοριστεί με όση ακρίβεια επιθυμούμε. Προφανώς μεγάλη ακρίβεια σημαίνει και περισσότερες επαναλήψεις.

Τα εγγενή σφάλματα της θεωρίας είναι και τα πιο χονδροειδή. Προκύπτουν από το ίδιο το σφάλμα που εμπεριέχουν οι θεωρητικοί υπολογισμοί, είτε λόγω μη ακριβούς μοντελοποίησης του συστήματος είτε λόγω μαθηματικών απλοποιήσεων ή παραδοχών. Συνήθως σε υπολογισμούς μηχανικού το σφάλμα κυμαίνεται περίπου στο 10%. Δεν έχει λοιπόν κανένα νόημα να προσπαθούμε να μειώσουμε σφάλματα, από οποιεσδήποτε αιτίες και αν προέρχονται αυτά, που είναι αρκετά μικρότερα του ποσοστού αυτού, αφού η μείωσή τους δεν θα μας οδηγήσει τελικά σε ακριβέστερα αποτελέσματα.

<u>Κεφάλαιο 3:</u>
Το πρόγραμμα FEAP.

3.1. Ο τρόπος προγραμματισμού του FEAP

3.1.1. Γενικές αρχές λειτουργίας

Το πρόγραμμα FEAP είναι ένα πρόγραμμα γραμμένο σε fortran το οποίο περιέχει διάφορες υπορουτίνες που, συνεργαζόμενες μεταξύ τους, κάνουν ανάλυση φορέων με τη μέθοδο των πεπερασμένων στοιχείων.

Οι υπορουτίνες λαμβάνουν και αποδίδουν δεδομένα λαμβάνοντάς τα είτε ως ορίσματα, είτε μέσω της δήλωσης common. Οι δηλώσεις των μεταβλητών γίνονται κανονικά μέσα στο πρόγραμμα, αλλά κυρίως είναι γραμμένες σε header files[1], και τοποθετούνται μέσω της δήλωσης include στην υπορουτίνα που χρειάζεται. Με αυτό τον τρόπο δεν χρειάζεται να γράφουμε πολλές φορές το ίδιο κομμάτι δηλώσεων μεταβλητών, αλλά μία μόνο φορά μέσα στο header file το οποίο το συμπεριλαμβάνουμε όπου θέλουμε.

Το FEAP, ως επαγγελματικό πρόγραμμα που στοχεύει σε ταχύτητα και αξιοπιστία, χρησιμοποιεί μόνο μονοδιάστατους πίνακες έχοντας προσαρμόσει τους κλασσικούς από τα μαθηματικά αλγορίθμους χειρισμού πολυδιάστατων μητρών, σε αλγορίθμους χειρισμού πολυδιάστατων μητρών αποθηκευμένων σε μονοδιάστατες, δηλαδή διανύσματα, χρησιμοποιώντας τις μεθόδους του **διανυσματικού προγραμματισμού**[2]. Η φιλοσοφία είναι να αποθηκεύονται όλα τα δεδομένα που απαιτούνται για την επίλυση σε ένα μεγάλο μητρώο διάνυσμα, το οποίο να υπάρχει μέσω ενός common στις περισσότερες υπορουτίνες, ώστε να μην απαιτείται να εισέρχεται ή να εξέρχεται από αυτές ως όρισμα. Το μητρώο αυτό ονομάζεται hr για τις πραγματικές μεταβλητές και mr για τις ακέραιες. Τα δύο αυτά μητρώα είναι και η κύρια μνήμη στην οποία αποθηκεύονται όλα σχεδόν εκείνα τα στοιχεία που χρησιμοποιεί το FEAP και οι περισσότερες κύριες εργασίες που εκτελούνται σε αυτό γίνονται πάνω στα στοιχεία του hr και του mr στα οποία η αναφορά γίνεται με τη μέθοδο των indexes[3].

Το θέμα λοιπόν το οποίο υπάρχει είναι σε ποιες θέσεις των διανυσμάτων hr και mr είναι αποθηκευμένη κάθε φορά ή τιμή της μεταβλητής ή του μητρώου που μας ενδιαφέρει. Το πρόβλημα αυτό επιτείνεται αν σκεφτούμε ότι κατά τη διάρκεια των υπολογισμών το μέγεθος μιας αποθηκευμένης μήτρας μπορεί να αλλάξει με τις αντίστοιχες αλλαγές στις θέσεις αποθήκευσης των μεταβλητών που βρίσκονται αποθηκευμένες μετά από αυτή. Το πρόβλημα αυτό αντιμετωπίζεται από το FEAP μέσω της χρήσης ενός άλλου διανύσματος, του np, που περιέχει τη θέση στο διάνυσμα hr ή mr της μεταβλητής που μας ενδιαφέρει.

[1] Είναι αρχεία κειμένου με κατάληξη «.h». Σε αυτά γράφουμε το τμήμα δηλώσεων των μεταβλητών και τα common, όπως ακριβώς θα τα γράφαμε στον κώδικα. Χρησιμοποιούνται για συντομία προς αποφυγή επαναλήψεων.

[2] **Διανυσματικός προγραμματισμός** είναι ένα είδος προγραμματισμού που εκτελεί εργασίες σε πολυδιάστατους πίνακες χρησιμοποιώντας μόνο μονοδιάστατους (διανύσματα) στους οποίους αποθηκεύει όλα τα στοιχεία που απαιτούνται. Για παράδειγμα αποθηκεύονται δύο δισδιάστατοι πίνακες M x N και N x K σε έναν μονοδιάστατο κατά στήλη με τις στήλες τους σε σειρά τη μία μετά την άλλη και για γνωστά M, N, K και φτιάχνεται ένας αλγόριθμος που βρίσκει το γινόμενό τους και το αποθηκεύει με τον ίδιο τρόπο σε ένα διάνυσμα. Ο τρόπος κατασκευής του αλγορίθμου αυτού από μετατροπή ενός κλασσικού αλγορίθμου πολλαπλασιασμού δισδιάστατων μητρών από τα μαθηματικά λέγεται διανυσματικός προγραμματισμός.

[3] Όταν λέμε μέθοδο των indexes εννοούμε ότι μια υπορουτίνα έρχεται και «κουμπώνει» πάνω στις θέσεις μνήμης των μεταβλητών που λαμβάνει ως όρισμα και εκτελεί τις διεργασίες που προβλέπονται επάνω σε αυτή, χωρίς να αντιγράφει τις μεταβλητές σε άλλες δικές της θέσεις μνήμης.

Αν δεν μας ενδιαφέρει απλά μία μεμονωμένη μεταβλητή, αλλά ένας πίνακας τότε το np περιέχει τη θέση του πρώτου στοιχείου του, και από στοιχεία όπως το πλήθος των γραμμών και των στηλών του και το αν έχει αποθηκευτεί κατά γραμμή ή κατά στήλη μπορούμε να βρούμε τη θέση του τελευταίου του στοιχείου, καθώς και τον τρόπο με τον οποίο διαβάζεται και εμφανίζεται[4]. Είναι λοιπόν προφανές τώρα ότι ο δείκτης του μητρώου np είναι σταθερός για κάθε μεταβλητή που μας ενδιαφέρει, αλλά το np(δείκτη) μεταβάλλεται, όπως μεταβάλλεται και η θέση μνήμης στο hr ή το mr της μεταβλητής που μας ενδιαφέρει. Φυσικά αυτό δε γίνεται από μόνο του, αλλά από κώδικα που έχει γραφεί χειρόγραφα από τους προγραμματιστές του FEAP. Έτσι λοιπόν για να έχουμε πρόσβαση στην τιμή μίας μεταβλητής που ξέρουμε ότι αντιστοιχεί για το διάνυσμα np στη θέση i, δεν έχουμε παρά να γράψουμε hr(np(i)), αν η μεταβλητή είναι πραγματική, ή mr(np(i)) αν η μεταβλητή είναι ακέραια.

Έτσι πλέον μας μένει να γνωρίζουμε το σταθερό i στο οποίο αντιστοιχεί η μεταβλητή ή ο πίνακας που θέλουμε να έχουμε πρόσβαση στις τιμές του. Αυτό είναι γραμμένο περιληπτικά στο programmer 's manual του FEAP, υπάρχει όμως πλήρης κατάλογος στα σχόλια της υπορουτίνας palloc. Εάν λοιπόν θέλουμε να γράψουμε μια δική μας υπορουτίνα που να χρησιμοποιεί κάποιες μεταβλητές που είναι αποθηκευμένες στα διανύσματα hr και mr, δεν έχουμε παρά να δώσουμε πρόσβαση στην υπορουτίνα μας στα διανύσματα hr, mr και np, είτε μέσω ορισμάτων, είτε μέσω common που είναι και ο πιο εύκολος τρόπος. Στην πραγματικότητα βέβαια δεν θα δηλώσουμε πάλι τα διανύσματα αυτά για να τα χρησιμοποιήσουμε στο common, αλλά θα κάνουμε include τα header files που περιέχουν αυτές τις δηλώσεις. Όσο για το ποια header files περιέχουν γενικά τις δηλώσεις που θέλουμε, μπορούμε να το βρούμε από το programmer 's manual, ή με αναζήτηση στο περιεχόμενο των header files και ανοίγοντας κάθε ένα που πιθανώς να περιέχει τις δηλώσεις που θέλουμε[5]. Αν δεν υπάρχει κάποιο header file που να περιέχει τις δηλώσεις που θέλουμε, τότε μπορούμε να φτιάξουμε δικό μας, ή να κάνουμε τις δηλώσεις με το χέρι στην υπορουτίνα μας.

Στο σημείο αυτό αντιλαμβανόμαστε ένα πολύ βασικό και θεμελιώδες προγραμματιστικό πρόβλημα που έχουμε πάντα όταν χρησιμοποιούμε **δυναμικές μνήμες**[6]. Αφού ορισμένα από τα μητρώα που είναι αποθηκευμένα στα κύρια διανύσματα του FEAP αλλάζουν μέγεθος, τότε για να μειώνεται η συνολική απαιτούμενη μνήμη, χρειάζεται τα στοιχεία που είναι αποθηκευμένα μετά από αυτά να μετακινούνται. Αυτό πρέπει υποχρεωτικά να συμβαίνει όταν τα μητρώα αυξάνονται, ενώ όταν μειώνονται δεν είναι απαραίτητο, απλά θα μένουν κάποιες αχρησιμοποίητες θέσεις στα κύρια διανύσματα. Η μετατόπιση (shift) όμως όλων των στοιχείων είναι μια αρκετά χρονοβόρα διαδικασία, ειδικά όταν συμβαίνει επαναλαμβανόμενα. Για το σκοπό αυτό χρειάζεται να ελαχιστοποιήσουμε το πλήθος των shifts που θα χρειαστούν κατά την εκτέλεση ενός προγράμματος και να τοποθετήσουμε προς το τέλος τα μητρώα που αλλάζουν συχνά

[4] Αποθήκευση δισδιάστατου μητρώου σε διάνυσμα κατά γραμμή (row- wise) ή κατά στήλη (column-wise) σημαίνει ότι οι γραμμές ή οι στήλες αντίστοιχα του μητρώου είναι αποθηκευμένες στο διάνυσμα η μία μετά την άλλη. Αυτό ασφαλώς χρειάζεται να το ξέρουμε για να μπορούμε να διαβάσουμε σωστά το μητρώο από το διάνυσμα, γιατί αλλιώς υπάρχει κίνδυνος να μπερδέψουμε τον πίνακα που διαβάσαμε με τον ανάστροφό του.

[5] Στα πλαίσια αυτής της διπλωματικής έχει αναπτυχθεί πρόγραμμα σε περιβάλλον matlab που του δίνουμε εμείς τι κείμενο αναζητούμε σε κάποιο header file και μας εμφανίζει και μας ανοίγει όλα όσα περιέχουν το κείμενο αυτό.

[6] **Δυναμική μνήμη**, σε αντιδιαστολή με τη **στατική μνήμη** είναι ότι η δεύτερη έχει εξ αρχής προαποφασισμένο μέγεθος το οποίο δεν υπερβαίνεται απλά δεν είναι απαραίτητο να το χρησιμοποιήσουμε όλο, ενώ η πρώτη έχει μεταβλητό μέγεθος που προσαρμόζεται από τον προγραμματιστή κατά το δοκούν με απλές εκτελέσιμες εντολές κατά τη διάρκεια της εκτέλεσης του προγράμματος.

διαστάσεις, ακόμα και να αφήσουμε εξ αρχής εσκεμμένα κάποιες θέσεις αχρησιμοποίητες για να χρησιμοποιηθούν στη συνέχεια. Αυτό λέγεται διαχείριση μνήμης (memory management) και η βελτιστοποίηση αυτού του είδους της διαχείρισης είναι από τα πιο δύσκολα και δημοφιλή θέματα στην κατασκευή προγραμμάτων, ειδικά όταν αναζητούμε το βέλτιστο τρόπο και όχι έναν απλά «καλό» τρόπο. Να σημειώσουμε επίσης ότι, ακόμα και αν δεν γίνει από τον προγραμματιστή, η διαχείριση αυτή γίνεται ούτως ή άλλως από τον compiler κατά τη μετάφραση[7] του προγράμματος, αλλά συνήθως όχι κατά το βέλτιστο για το συγκεκριμένο πρόβλημα τρόπο.

Στην περίπτωση όμως του προγράμματος FEAP εκτός από τα διανύσματα hr, mr και np, υπάρχουν και άλλες μεταβλητές που μας είναι χρήσιμες σε όλη την εξέλιξη του προγράμματος και δεν είναι αποθηκευμένες σε αυτά τα διανύσματα. Τέτοιες μεταβλητές είναι η numnp που είναι το πλήθος των κόμβων στο πρόβλημα, η numel που είναι ο αριθμός των πεπερασμένων στοιχείων που χρησιμοποιούνται, η nummat που είναι το πλήθος των υλικών που έχουμε χρησιμοποιήσει, η ndf που είναι ο μέγιστος αριθμός των αγνώστων ανά κόμβο, nen που είναι ο μέγιστος αριθμός κόμβων ανά στοιχείο και άλλες που πιθανό να χρειαστεί ο κάθε προγραμματιστής αναλόγως τη δουλειά που κάνει. Η πρόσβαση σε αυτές γίνεται με τον ίδιο τρόπο που έχουμε περιγράψει και για τα κύρια διανύσματα, αλλά συνήθως προτιμάται το include του header file που τις περιέχει όλες σαν σε πακέτο.

Σε επόμενα κεφάλαια γίνεται λεπτομερής αναφορά στις μεταβλητές που χρειαζόμαστε και σε ποιά header files μπορούμε να τις βρούμε.

3.1.2. Εισαγωγή δεδομένων και εξαγωγή αποτελεσμάτων

Το FEAP δέχεται ορισμένα στοιχεία ως δεδομένα και με βάση αυτά διαμορφώνει την τελική λύση. Τα στοιχεία που μπορούμε να του εισάγουμε ως δεδομένα είναι στοιχεία για τον κάναβο των πεπερασμένων στοιχείων, την επιλογή των υλικών, τις συνοριακές συνθήκες, τις συνθήκες φόρτισης στατικής ή δυναμικής, πολλαπλές διαδοχικές φορτίσεις στατικές ή δυναμικές και εντολές που αφορούν τους τρόπους εμφάνισης των αποτελεσμάτων, όλα δηλαδή τα στοιχεία που είναι απαραίτητα για τον πλήρη καθορισμό ενός προβλήματος πεπερασμένων στοιχείων. Επειδή θα ήταν δύσχρηστο όλα αυτά τα στοιχεία να δίνονται από το πληκτρολόγιο συνέχεια και κάθε φορά, χρησιμοποιούνται txt αρχεία (input files) που ο χρήστης τα διαμορφώνει κατάλληλα και περιέχουν τα απαραίτητα δεδομένα για την επίλυση. Έτσι το ίδιο πρόβλημα μπορεί να επιλύεται πολλές φορές χωρίς να χρειάζεται να δίνουμε συνέχεια τα ίδια δεδομένα. Τα αποτελέσματα αποθηκεύονται αντίστοιχα σε txt αρχεία (output files), από τα οποία διαβάζουμε τα αποτελέσματα που θέλουμε.

Τα αρχεία που χρησιμοποιούνται προτείνεται να έχουν συγκεκριμένο format ονόματος. Το όνομα ενός input file καλό είναι να ξεκινάει με το χαρακτήρα “I”. Τα output files καθορίζονται από το ίδιο το FEAP και το όνομά τους είναι ίδιο με του input file μόνο που ο πρώτος του χαρακτήρας “I” αντικαθίσταται με άλλους, οι οποίοι βοηθούν στο να τα ξεχωρίζουμε. Το τελικό output file που περιέχει όλα τα αποτελέσματα που θέλουμε ξεκινάει με το χαρακτήρα “O”.

Η δομή ενός input file περιγράφεται στο user ‘s manual του προγράμματος και η ακριβής περιγραφή της ξεφεύγει από τους στόχους αυτής της εργασίας. Ενδεικτικά όμως θα αναφέρουμε κάποια βασικά σημεία:

[7] Μετάφραση ενός προγράμματος είναι η μετατροπή της γλώσσας προγραμματισμού που χρησιμοποιούμε (πχ fortran) σε γλώσσα μηχανής (δηλαδή 0 και 1) από τον compiler που χρησιμοποιούμε (πχ visual studio, force, intel compaq fortran κτλ).

1. Αρχικές εντολές.
   Ορίζουν την αρχή ενός input file και κάποιες λειτουργίες που χρειάζεται να καθοριστούν εξ' αρχής όπως το όνομα του προβλήματος, το πλήθος των κόμβων, των στοιχείων, των υλικών, τη μέγιστη απόσταση των κόμβων του καννάβου, το μεγαλύτερο πλήθος των αγνώστων από όλους τους κόμβους, το μεγαλύτερο πλήθος των κόμβων από όλα τα στοιχεία, το μέγιστο πλήθος παραμέτρων για καθορισμό των ιδιοτήτων των υλικών και το μέγιστο πλήθος παραμέτρων για καθορισμό των ιδιοτήτων των υλικών που έχει ορίσει ο χρήστης.
2. Εντολές καθορισμού των υλικών.
   Στην ουσία πρόκειται για ομάδες χαρακτηριστικών των στοιχείων που χρησιμοποιούμε. Καθορίζουμε δηλαδή το μοντέλο συμπεριφοράς τους και τις παραμέτρους που το διέπουν, για παράδειγμα αν το υλικό είναι ελαστικό και με τι μέτρο ελαστικότητας, αν είναι πλαστικό οπότε δίνουμε μέτρο ελαστικότητας και τάση διαρροής και άλλα πιο σύνθετα μοντέλα που υπάρχουν εξ' αρχής στο FEAP ή που έχει δημιουργήσει ακόμα και ο χρήστης από μόνος του με κατάλληλες παρεμβάσεις στον κώδικα.
3. Εντολές εισαγωγής γεωμετρίας φορέα και παραγωγής του καννάβου πεπερασμένων στοιχείων.
   Υπάρχει η επιλογή να δώσουμε εμείς κατευθείαν τις συντεταγμένες του καννάβου. Αυτό είναι ιδιαίτερα βολικό όταν το input file κατασκευάζεται αυτοματοποιημένα από άλλα προγράμματα με δυνατότητα να παράγουν το δικό τους κάνναβο, τον οποίο και θέλουμε να χρησιμοποιήσουμε στο FEAP.
4. Εντολές προσδιορισμού των συνοριακών συνθηκών που αφορούν στηρίξεις του φορέα.
   Στήριξη του φορέα και μετακινησιακές δεσμεύσεις σε κόμβους. Δεν χρειάζεται να προσδιορίσουμε ακριβώς σε κάθε κόμβο τι γίνεται, αλλά υπάρχουν αυτοματοποιημένες εντολές που μπορούμε να αναφερθούμε σε μια ολόκληρη ομάδα κόμβων για παράδειγμα μιας πλευράς.
5. Εντολές προσδιορισμού της χωρικής συνάρτησης της φόρτισης.
   Περιλαμβάνουν κάθε είδος φορτίου που μπορεί να χρειαστεί, επικόμβια, επιφανειακά, μαζικά και με την επιθυμητή **χωρική**[8] κατανομή. Συγκεκριμένα υπάρχει ένα μοντέλο φορτίου με διάφορες παραμέτρους που από τις τιμές που τους δίνουμε μπορεί να προκύψει φορτίο γραμμικό, ημιτονοειδές και ημιτονοειδές εκθετικό. Εάν όμως δεν μας καλύπτει αυτός ο τρόπος υπάρχει και άλλη εναλλακτική στην οποία δίνουμε το φορτίο που θέλουμε ως ζεύγη τιμών θέσης- φορτίου. Είναι προφανές ότι όσο περισσότερα σημεία δώσουμε, τόσο πιο ακριβής είναι η χωρική κατανομή του φορτίου που επιθυμούμε.
6. Εντολές προσδιορισμού της χρονικής συνάρτησης της φόρτισης[9].

[8] Χρειάζεται να αντιδιαστείλουμε τη **χωρική** από τη **χρονική** κατανομή ενός φορτίου. Η πρώτη έχει να κάνει με την κατανομή της φόρτισης επάνω στο φορέα. Η δεύτερη έχει να κάνει με το πώς μεταβάλλεται η φόρτιση με το χρόνο. Προφανώς μια πλήρης φόρτιση είναι συνάρτηση μιας χωρικής μεταβλητής, διανυσματικής ή μονόμετρης, που αντιστοιχεί μονοσήμαντα σε κάθε σημείο του φορέα και μιας χρονικής, μονόμετρης προφανώς (….), που αντιστοιχεί στη χρονική στιγμή που αναφερόμαστε. Έτσι λοιπόν γενικά σε ένα φορέα έχουμε πολλές (άπειρες) χωρικές φορτίσεις που λέγονται και **στιγμιότυπα φόρτισης**, με κάθε ένα εξ' αυτών να αντιστοιχεί μονοσήμαντα σε μια χρονική στιγμή.

[9] Να παρατηρήσουμε ότι και στατικά φορτία μπορεί να είναι χρονικά μεταβαλλόμενα. Η στατική ή δυναμική ανάλυση έχει να κάνει κυρίως με τον τρόπο που επιλέγουμε να αναλύσουμε το φορέα και όχι με το είδος της φόρτισης. Η έννοια του στατικού αλλά μεταβλητού με το χρόνο φορτίου είναι ότι για κάθε ένα από τα στιγμιότυπα φόρτισης –άπειρα ή όχι – που επιβάλλει στο φορέα, δίνει μία και μοναδική εντατική κατάσταση σε αυτόν ανεξάρτητη από το χρόνο, αφού η ανάλυση του φορέα είναι στατική, άρα συνολικά μια ακολουθία στιγμιότυπων εντατικών καταστάσεων. Η μέθοδος είναι

Οι εντολές αυτές δίνονται εντός ενός βρόχου BATCH- END, όπως περιγράφεται στο user 's manual και στην ουσία καθορίζουν με ζεύγη τιμών της χρονικής συνάρτησης με την οποία πολλαπλασιάζεται η χωρική συνάρτηση της φόρτισης για να προκύψει η τελική φόρτιση ως προς χώρο και χρόνο.

7. Εντολές καθορισμού των αποτελεσμάτων που θέλουμε να λάβουμε.
   Δίνουμε στο πρόγραμμα τι είδους αποτελέσματα θέλουμε να μας δώσει. Για παράδειγμα αν θέλουμε μετατοπίσεις, τάσεις και ποιες τάσεις, σε ποιους κόμβους ή βαθμούς ελευθερίας θέλουμε τα αποτελέσματα κτλ..

Είναι γεγονός, ότι το FEAP δεν διαθέτει κάποιο είδος interface για την εισαγωγή του προβλήματος, αλλά χρειάζεται να κατασκευάσουμε εμείς το input file που θέλουμε, είτε με το χέρι, είτε με χρήση άλλου προγράμματος. Στην ουσία στο input file δίνουμε εντολές προς εκτέλεση, όπως ακριβώς θα τις δίναμε και σε κάποια γλώσσα προγραμματισμού. Όπως και εκεί έτσι και εδώ έχουμε συγκεκριμένο τρόπο σύνταξης των εντολών, οι οποίες βέβαια δεν εκτελούνται από compiler, αλλά από το πρόγραμμα FEAP.

Εκτός όμως από input files το FEAP προσφέρει και μια πιο διαδραστική με το χρήστη μέθοδο για εισαγωγή του προβλήματος και του τι είδους αποτελέσματα θέλουμε να λάβουμε. Αυτή η δυνατότητα ονομάζεται interactive mode. Συγκεκριμένα κατά την εκτέλεση του προγράμματος ανοίγει ένα παράθυρο στο οποίο εμείς μπορούμε να δίνουμε μία- μία τις εντολές που θα δίναμε σε ένα input file, με μικρές διαφορές. Οι εντολές εκλαμβάνονται ως ακολουθίες χαρακτήρων και μέσα στο πρόγραμμα αντιστοιχίζονται στις απαραίτητε ενέργειες που χρειάζεται να γίνουν. Έτσι μπορούμε να βλέπουμε διαδραστικά τα δεδομένα που δίνουμε να χρησιμοποιούνται από το FEAP, για παράδειγμα να παρακολουθήσουμε βήμα- βήμα τον ορισμό του φορέα, τη δημιουργία καννάβου πεπερασμένων στοιχείων, την επιβολή φόρτισης και τέλος τον υπολογισμό και εμφάνιση του είδους των αποτελεσμάτων που θέλουμε. Επίσης επιτρέπεται και συνδυασμός input file στην αρχή και διαδραστικού περιβάλλοντος μετά. Αυτό είναι χρήσιμο όταν πάρουμε, με οποιονδήποτε τρόπο, τα αποτελέσματα που θέλουμε και στη συνέχεια θέλουμε να τα επεξεργαστούμε περαιτέρω, για παράδειγμα να κάνουμε κάπου zoom ή να κάνουμε κάποιες τομές.

Η έξοδος των αποτελεσμάτων δεν γίνεται όμως μόνο μέσω του output file, αλλά γίνεται και γραφικά με χρήση γραφικών **OpenGL**[10]. Συγκεκριμένα μπορούμε να επιλέξουμε να εμφανιστούν ισοϋψείς για τάσεις, ή να δούμε τις επικόμβιες μετατοπίσεις με διανύσματα. Τα γραφικά αυτά παράγονται στο ίδιο παράθυρο που λειτουργεί το interactive mode ή σε ξεχωριστό παράθυρο, αναλόγως τις ρυθμίσεις που έχουμε κάνει κατά την εγκατάσταση[11]. Δεν είναι δυνατή η εμφάνιση πολλαπλών γραφημάτων ταυτόχρονα.

3.1.3. Μεταβλητές στα header files

ιδιαίτερα χρήσιμη όταν μελετάμε στην πλαστικότητα μεταβολές φορτίσεων και κυρίως αποφορτίσεις, οπότε υπάρχουν στο φορέα και παραμένουσες τάσεις οι οποίες με αυτόν τον τρόπο συνεκτιμώνται στους υπολογισμούς.

[10] Τα γραφικά **OpenGL** είναι μια βιβλιοθήκη με εντολές γραφικών των windows η οποία μπορεί να εγκατασταθεί ξεχωριστά στον υπολογιστή και είναι συμβατή με πολλές γλώσσες προγραμματισμού, στις οποίες μπορεί να κληθεί κανονικά σαν βιβλιοθήκη και να χρησιμοποιηθούν οι εντολές της. Είναι ιδιαίτερα χρήσιμη σε γλώσσες προγραμματισμού που δεν έχουν δική τους δυνατότητα για δημιουργία γραφικών, αλλά λόγω της ταχύτητας, της αποτελεσματικότητας και της ευκολίας που προσφέρει με τις προηγμένες εντολές της χρησιμοποιείται πολύ ακόμα και αν η γλώσσα που δουλεύουμε έχει δικές της δυνατότητες χειρισμού γραφικών. Τα γραφικά που χειρίζεται ξεκινούν από απλή χάραξη έγχρωμων γραμμών και φτάνουν σε προηγμένες λειτουργίες animation.

[11] Δες στο installation manual του FEAP υποσημείωση στη σελίδα 8.

Στο σημείο αυτό θα περιγράψουμε κάποιες σημαντικές μεταβλητές που δεν βρίσκονται στα διανύσματα της κύριας μνήμης, δηλαδή τα hr και mr, και χρησιμοποιούνται στο FEAP καθώς και τα header files στα οποία βρίσκονται, τόσο αυτές, όσο και τα διανύσματα της κύριας μνήμης.

1. setups.h
   Περιέχει όλες τις μεταβλητές που έχουν να κάνουν με αρχικές ρυθμίσεις του FEAP και αποτελείται από ακέραιες και λογικές μεταβλητές. Ορισμένες σημαντικές μεταβλητές που περιέχει:
   - *processor:* Είναι το πλήθος των επεξεργαστών ου χρησιμοποιεί ο compiler.
   - *soltyp:* Παίρνει τιμές 1 ή δύο αναλόγως το είδος της λύσης που χρησιμοποιείται.
   - *solver:* Του δίνουμε τιμή .true. αν θέλουμε να χρησιμοποιηθεί ο επιλύτης του FEAP ή τιμή .false. αν θέλουμε να χρησιμοποιηθεί του χρήστη.
   - *perflg:* Αν είναι .true. σημαίνει ότι υπάρχει περιοδικότητα στη διαμόρφωση των στοιχείων του καννάβου.
2. cdata.h
   Πολύ σημαντικό header file που περιέχει μεταβλητές που έχουν να κάνουν με στοιχεία του φορέα. Μεταβλητές που περιέχει:
   - *numnp:* Πλήθος κόμβων.
   - *numel:* Πλήθος στοιχείων.
   - *nummat:* Πλήθος υλικών.
   - *nen:* Πλήθος κόμβων στα στοιχεία. Αν διαφέρει από στοιχείο σε στοιχείο το μέγιστο.
   - *neq:* Πλήθος ενεργών εξισώσεων.
   - *ipr:* Αν έχει τιμή 1 οι μεταβλητές τύπου real που χρησιμοποιούνται είναι single precision, ενώ αν έχει τιμή 2 double precision.
3. sdata.h
   - *ndf:* Το μέγιστο πλήθος βαθμών ελευθερίας ανά κόμβο.
   - *ndm:* Είναι το πλήθος των διαστάσεων τους χώρου. Για μονοδιάστατα προβλήματα έχει τιμή 1, για δισδιάστατα 2 και για τρισδιάστατα 3.
4. comblk.h
   Πολύ σημαντικό header file που περιέχει τα διανύσματα της κύριας μνήμης:
   - *hr:* Το διάνυσμα της κύριας μνήμης που περιέχει πραγματικούς αριθμούς.
   - *mr:* Το διάνυσμα της κύριας μνήμης που περιέχει ακέραιους αριθμούς.

   Περισσότερες πληροφορίες για το περιεχόμενο της κύριας μνήμης θα δώσουμε σε επόμενο κεφάλαιο.
5. pointer.h
   Περιέχει όλους του απαιτούμενους δείκτες που χρειάζονται για το χειρισμό της κύρια μνήμης.
   - *np:* Περιέχει τους δείκτες στους οποίους υπάρχει ένα συγκεκριμένο μέγεθος ορισμένο από το πρόγραμμα στην κύρια μνήμη, την ακέραια, ή την πραγματική.
   - *up:* Περιέχει τους δείκτες στους οποίους υπάρχει ένα συγκεκριμένο μέγεθος ορισμένο από το χρήστη στην κύρια μνήμη, την ακέραια, ή την πραγματική.

   Είναι λογικό ότι για να χειριστούμε την κύρια μνήμη χρειαζόμαστε τους δείκτες που βρίσκονται αποθηκευμένοι στις μεταβλητές αυτού του header file. Άρα δεν έχει νόημα να χρησιμοποιούμε κάπου την κύρια μνήμη χωρίς να χρησιμοποιούμε τους δείκτες αυτούς. Επίσης για τον πλήρη ορισμό ενός μητρώου στην κύρια μνήμη χρειαζόμαστε ακόμα και τις μεταβλητές του cdata.h. Αυτός είναι και ο λόγος που αυτά τα τρία header files γίνονται συνήθως όλα μαζί include και όχι χωριστά.

3.1.4. Περιεχόμενα της κύριας μνήμης του FEAP

Η κύρια μνήμη του FEAP είναι αποθηκευμένη όπως έχουμε πει σε δύο μεγάλα διανύσματα μεταβλητού μεγέθους, το hr και το mr και περιλαμβάνονται στο header file comblk.h. Στο κεφάλαιο αυτό θα δούμε λεπτομερέστερα τα περιεχόμενά τους.

Όπως έχουμε πει οι μνήμες αυτές έχουν αποθηκεύσει τα κύρια μητρώα τα οποία χρησιμοποιούμε όπως οι συντεταγμένες του καννάβου, οι ιδιότητες των υλικών, τα μητρώα δυσκαμψίας κτλ. με τη μορφή διανύσματος. Ως εκ τούτου λοιπόν κρίνεται αναγκαία η εξήγηση της μετατροπής ενός μητρώου σε διάνυσμα και το αντίστροφο.

Στο σημείο όμως αυτό χρειάζεται να κάνουμε μία παρατήρηση. Τα μητρώα δεν έχουν συγκεκριμένο όνομα στο FEAP. Αυτό συμβαίνει διότι η επεξεργασία τους γίνεται κατευθείαν πάνω στην κύρια μνήμη που είναι αποθηκευμένα από υπορουτίνες, οι οποίες ως γνωστόν μπορούν να χρησιμοποιούν όποιο όνομα θέλουν, αρκεί αυτό να αντιστοιχίζεται σωστά στη θέση μνήμης την οποία επεξεργάζονται. Όμως, παρ' όλ' αυτά τα μητρώα έχουν συμβατικά ένα όνομα το οποίο είναι αποθηκευμένο με τη μορφή συμβολοσειράς στο ευρετήριο του FEAP[12] και χρησιμοποιείται και στα manuals του προγράμματος ώστε να είναι η πιο εύκολη η κατανόηση των περιεχομένων της κύριας μνήμης και η προσπέλασή τους. Στο σημείο αυτό θα δούμε μερικά από τα πιο σημαντικά μητρώα που μπορεί να χρειαστεί κάποιος, πώς είναι αποθηκευμένα, σε ποια θέση και το πώς μπορούμε να έχουμε πρόσβαση σε αυτά. Χαρακτηριστικά μητρώα στην κύρια μνήμη που μας ενδιαφέρουν είναι:

1. <u>Το μητρώο X:</u>
   Βρίσκεται στη θέση np(43). Αποτελείται από mumnp στήλες με την i- στήλη να περιέχει τις συντεταγμένες του i- κόμβου, κάθε μία συνιστώσα των οποίων είναι αποθηκευμένη στην αντίστοιχη γραμμή. Για παράδειγμα η τετμημένη του 3^{ου} κόμβου είναι αποθηκευμένη στη θέση (1, 3), ενώ η τεταγμένη του στη θέση (2, 3) του μητρώου. Αν έχουμε και τρίτη διάσταση τα πράγματα λειτουργούν αντίστοιχα. Το δισδιάστατο μητρώο αυτό είναι αποθηκευμένο στην κύρια μνήμη κατά στήλη.
2. <u>Το μητρώο F:</u>
   Βρίσκεται στη θέση np(27). Είναι ένα τρισδιάστατα μητρώο που αποτελείται από δύο επίπεδα. Το πρώτο περιέχει τις επικόμβιες δρώσες δυνάμεις και το δεύτερο τις μετατοπίσεις. Κάθε επίπεδο είναι ένας δυσδιάστατος πίνακας που αποτελείται από ndf γραμμές και numnp στήλες. Στην i- στήλη περιέχει για παράδειγμα το πρώτο επίπεδο τις δυνάμεις στον συγκεκριμένο κόμβο με τη δύναμη κατά τον j βαθμό ελευθερίας να είναι αποθηκευμένη στη j γραμμή.
3. <u>Το μητρώο S:</u>
   Βρίσκεται στη θέση np(36). Είναι δυσδιάστατο μητρώο, με την κάθε μία του διάσταση να είναι συνήθως το γινόμενο των βαθμών ελευθερίας ανά στοιχείο επί το πλήθος των στοιχείων. Περιέχει το καθολικό μητρώο ακαμψίας του φορέα.
4. <u>Το μητρώο D:</u>
   Βρίσκεται στη θέση np(25) της κύριας μνήμης και είναι ndd x nummat. Αποτελείται δηλαδή από ndd γραμμές οι οποίες σε κάθε στήλη περιέχουν τα στοιχεία[13] για κάθε

[12] Το FEAP περιέχει ένα ευρετήριο στο οποίο αποθηκεύει ως συμβολοσειρές τα ονόματα των μητρώων που χρησιμοποιεί και ορισμένα στοιχεία για αυτά, όπως σε ποια θέση βρίσκεται το πρώτο στοιχείο τους στην κύρια μνήμη, ο οποίος είναι και ο βασικός λόγος για να χρησιμοποιήσει κάποιος το ευρετήριο. Η χρήση του ευρετηρίου γίνεται μέσω της υπορουτίνας pgetd που λαμβάνει διάφορα ορίσματα συμπεριλαμβανομένου και του ονόματος του μητρώου ή της μεταβλητής που θέλει. Για περισσότερες πληροφορίες για τη χρήση του ευρετηρίου και της συνάρτησης δες το programmer manual του FEAP στη σελίδα 13.

[13] Δες στο programmer manual σελίδα 35.

υλικό που χρησιμοποιείται, όπως το μέτρο ελαστικότητας, ο λόγος Poisson, ο συντελεστής θερμικής διαστολής κτλ. Αυτό επαναλαμβάνεται σε κάθε στήλη αντίστοιχα για κάθε υλικό.

5. Το μητρώο ANG:
   Βρίσκεται στη θέση np(45) της κύριας μνήμης και είναι διάστασης numnp. Είναι δηλαδή ένα μονοδιάστατο μητρώο που περιέχει όλες τις γωνίες στροφής για κάθε κόμβο και αφορούν τροποποίηση της γωνίας των βαθμών ελευθερίας του, για παράδειγμα στην περίπτωση κεκλιμένης στήριξης.
6. Το μητρώο T:
   Βρίσκεται στη θέση np(38) της κύριας μνήμης και είναι διάστασης numnp. Είναι δηλαδή ένα μονοδιάστατο μητρώο που περιέχει τις διαφορές θερμοκρασίας των κόμβων σε περίπτωση που θέλουμε να συμπεριλάβουμε και θερμοκρασίες στην ανάλυσή μας.
7. Το μητρώο U:
   Βρίσκεται στη θέση np(40) της κύριας μνήμης και είναι ndf x numnp x 3. Αποτελείται δηλαδή από 3 επίπεδα τα οποία περιέχουν τις τελικές μετατοπίσεις ανά κόμβο, για κάθε βαθμό ελευθερίας του. Έτσι, αν στη θέση i είναι αποθηκευμένη η μετατόπιση για τον πρώτο βαθμό ελευθερίας ενός κόμβου, στη θέση i+1 είναι αποθηκευμένη η μετατόπιση κατά τον επόμενο βαθμό ελευθερίας κοκ. Και τα τρία επίπεδα περιέχουν τα ίδια πράγματα.

3.1.5. Εσωτερική προγραμματιστική δομή του FEAP

Το FEAP ελέγχεται από μία μεγάλη ρουτίνα που ονομάζεται feap83, η οποία και περιέχει όλες τις άλλες. Αυτή αποτελείται συνολικά από τρεις μεγάλες υπορουτίνες, την pstart, την pmessage και την pcontr οι οποίες περιέχουν με τη σειρά τους άλλες μέσω των οποίων διατυπώνεται και λύνεται το πρόβλημα. Συνολικά η κύρια πολυπλοκότητα στον κώδικα προκύπτει από τον έλεγχο των εισαγόμενων δεδομένων και εντολών που δίνουμε και την αντιστοίχιση των εντολών στις κατάλληλες ρουτίνες. Θα προσπαθήσουμε να περιγράψουμε τη δομή του προγράμματος επεξηγώντας κάθε υπορουτίνα χωριστά από τις άλλες και μόνη της, ώστε να δώσουμε βάση στην γενική οργανωτική δομή του FEAP και όχι στις λεπτομέρειες υπορουτίνων που δεν μας αφορούν.

Για τη μελέτη της προγραμματιστικής δομής του FEAP αρχικά χρειάζεται να γνωρίζουμε τις ακόλουθες ρουτίνες και συναρτήσεις:

1. pstrip
   Είναι υπορουτίνα. Λαμβάνει μία ακολουθία κειμένου yyy και αφαιρεί από αυτή τα κενά στην αρχή του και τα σχόλια που το συνοδεύουν, δηλαδή ό,τι έχει γραφτεί μετά από ένα χαρακτήρα "!" και αποδίδει το αποτέλεσμα xxx.
2. pcomp
   Είναι συνάρτηση. Λαμβάνει δύο ακολουθίες κειμένου a και b και συγκρίνει τους n πρώτους χαρακτήρες τους χωρίς διάκριση πεζών κεφαλαίων. Αν είναι, ίδιοι επιστρέφει .true., αλλιώς επιστρέφει .false..
3. palloc
   Είναι συνάρτηση. Λαμβάνει το όνομα name μιας μεταβλητής που θέλουμε να αποθηκευτεί σε ένα από τα δύο κύρια διανύσματα και το δείκτη num του διανύσματος np που θα περιέχει τη θέση του, το μήκος length της μεταβλητής και μια τιμή precis που είναι 1 για ακέραια μεταβλητή και 2 για πραγματική. Δεσμεύει το χώρο που χρειάζεται σε ένα από τα κύρια διανύσματα και επιστρέφει .true. αν η

δέσμευση έγινε επιτυχώς, αλλιώς .false.. Πολύ χρήσιμα είναι τα σχόλιά της που λένε σε ποιο δείκτη του μητρώου np αντιστοιχεί κάποιο μέγεθος ή μητρώο.

4. getcon
   Είναι υπορουτίνα. Υπολογίζει την μικρότερη ποσότητα κατά την οποία μπορεί δύο αριθμοί να διαφέρουν στον επεξεργαστή.
5. pvalues
   Είναι υπορουτίνα. Υπολογίζει κάποιες μαθηματικές παραμέτρους ώστε να έχει έτοιμες τις τιμές τους και να μη χρειάζεται να τις υπολογίσει ξανά. Τις τιμές τις μεταφέρει στο απαραίτητο σημείο του προγράμματος μέσω του include file pconstant.h.
6. acheck
   Είναι υπορρουτίνα. Λαμβάνει με τη μορφή χαρακτήρων ένα πεδίο, εντοπίζει τα κόμματα και ξεχωρίζει τα δεδομένα που χωρίζονται από αυτά και τα παραθέτει με τη σειρά σε έναν πίνακα. Και τα δεδομένα και τα αποτελέσματα είναι τύπου character.
7. vinput
   Είναι λογική συνάρτηση. Λαμβάνει έναν αριθμό ως χαρακτήρα και τον μετατρέπει σε πραγματική μεταβλητή. Αν η μετατροπή είναι επιτυχής, τότε παίρνει την τιμή .true., αλλιώς την τιμή .false..
8. tinput
   Είναι μία από τις δύο συναρτήσεις του FEAP εισόδου δεδομένων από αρχείο. Εισάγει δεδομένα τύπου double precision, αλλά επιτρέπει και στα δεδομένα αυτά να υπάρχουν χαρακτήρες που είναι συνήθως ονόματα μεταβλητών που αντιπροσωπεύουν κάποια τιμή μεταβλητής.
9. pinput
   Είναι μία από τις δύο συναρτήσεις του FEAP εισόδου δεδομένων από αρχείο. Εισάγει τα δεδομένα σε έναν μονοδιάστατο πίνακα τύπου double precision.

Εκτός από αυτά εκτελούνται και άλλες υπορουτίνες, είτε κατ' απαίτηση του χρήστη, είτε ούτως ή άλλως για τη λύση του προβλήματος. Αρχικά θα ξεκινήσουμε με τις εισαγωγικές.

1. pstart
   Σκοπός της υπορουτίνας pstart είναι να λάβει τη διαδρομή του αρχείου εισόδου δεδομένων, να αναγνωρίσει το μέγεθος της οθόνης του υπολογιστή ώστε να καθορίσει τα γραφικά, να καθορίσει κάποιες ρυθμίσεις εγκατάστασης και να καθορίσει τα ονόματα των αρχείων.
2. pmessage
   Η υπορουτίνα pmessage εμφανίζει κάποια αρχικά μηνύματα στη οθόνη και τίποτα άλλο.
3. pcontr
   Είναι υπορουτίνα. Είναι η τελευταία υπορουτίνα από τις βασικές του FEAP και είναι αυτή που στην ουσία λύνει το πρόβλημα. Αυτή καλεί όλες τις υπορουτίνες που καλούνται και κάνουν όλους τους υπολογισμούς για την επίλυση του προβλήματος.

3.1.5.1. Η υπορουτίνα pcontr

Η υπορουτίνα pcontr είναι η πρώτη βασική που εκτελείται και καλεί όλες τις υπορουτίνες που κάνουν τους απαιτούμενους υπολογισμούς. Αρχικά ελέγχει τα ονόματα των αρχείων στα οποία θα διαβαστούν ή θα γραφούν δεδομένα. Στη συνέχεια ελέγχει αν ο χρήστης έχει γράψει δικό του κώδικα στις υπορουτίνες που περιέχουν στοιχεία για κάναβο, τρόπο επίλυσης, μοντέλα υλικών, τρόπους εκτύπωσης και καινούριες εντολές από αυτόν.

Στις υπορουτίνες αυτές γίνεται αναφορά από τις πιο μεγάλες υπορουτίνες umshlib, umaclib, uconst, upltlib και usetib αντίστοιχα για κάθε περίπτωση.

Στη συνέχεια υπάρχει ένας μεγάλος βρόχος υλοποιούμενος με go to που ξεκινάει από την εντολή με ετικέτα 1 και τελειώνει σε κάθε σημείο που υπάρχει η εντολή «go to 1». Στόχος του βρόχου είναι να διαβάζει τις εντολές του αρχείου εισαγωγής δεδομένων ή τις εντολές μέσω interactive mode. Αυτός ο διαχωρισμός γίνεται από τη λογική μεταβλητή intx. Αν η τιμή της είναι .true., τότε η ανάγνωση των εντολών γίνεται από το interactive mode, αλλιώς από το input file. Στην περίπτωση του input file, η εντολή που δόθηκε καθαρίζεται από κενά ή σχόλια μέσω της pstrip και μετά αναλόγως την εντολή εκτελούνται οι κατάλληλες εντολές ή υπορουτίνες.

Οι εντολές όμως που διαβάζει η pcontr και κάνει την αντιστοίχιση δεν είναι όλες οι εντολές που περιέχονται στο input file, αλλά μόνο οι βασικές εντολές, αυτές δηλαδή που στόχος τους δεν είναι να εκτελέσουν κάποια συγκεκριμένη ενέργεια, αλλά να προδηλώσουν ότι ακολουθεί κάποια συγκεκριμένη ομάδα εντολών που αναφέρονται όλες στο ίδιο πράγμα. Τέτοιες εντολές είναι για παράδειγμα η πρώτη εντολή που τοποθετείται στο πάνω μέρος του input file και προδηλώνει το τι ακολουθεί, όπως η εντολή «feap» δείχνει ότι ακολουθούν εντολές σχετικές με γεωμετρική δημιουργία φορέα και φόρτισης, η εντολή «tie» που σημαίνει ότι ολοκληρώθηκε η εισαγωγή του φορέα και χρειάζεται να γίνει έλεγχος για διαγραφή διπλών κόμβων κτλ, η εντολή «batch» που στην ουσία προδηλώνει οι εντολές που ακολουθούν αναφέρονται στην επίλυση του φορέα, η εντολή «stop» που κλείνει όποια ανοιχτά παράθυρα ή αρχεία υπάρχουν και τερματίζει το πρόγραμμα κ.α.. Συνολικά οι εντολές που διαβάζονται από την pcontr είναι οι «feap», «noco», «coun», «keep», «noke», «match», «clus», «bina», «ufea», «prin», «nopr», «incl», «init», «titl», «inte», «batc», «macr», «manu», «link», «elin», «clin», «tie», «para», «cons», «rigi», «join», «rloa», «rbou», «rdis», «mast», «opti», «dict», «loop», «next», «file», «tri2», «unit», «*rea», «*com», «cont», «part», «orde», «lagr», «rema», «debu», «doma», «stop» και «cntn».

Ενδιαφέρον παρουσιάζει η συμπεριφορά της υπορουτίνας pcontr όταν συναντά την εντολή «batc» που δείχνει ότι ακολουθεί η λύση. Κατ' αρχάς έχει εκτελεστεί πρώτα η εντολή «tie» η οποία χρησιμοποιεί την υπορουτίνα «tienod» και ενώνει τους κόμβους που έχουν τις ίδιες συντεταγμένες και ως εκ τούτου μειώνει τους αρχικούς κόμβους. Στη συνέχεια εκτελούνται οι υπορουτίνες pmerbc, pecmes, pidset, pextnd από την pcontr που, με βάση τις συγχωνεύεις που έγιναν στην tienod, συγχωνεύουν δυνάμεις, μετατοπίσεις και συνοριακές συνθήκες και παράγουν το τελικό φορέα. Τέλος εκτελείται η formfe που παράγει κάποια προκαταρκτικά μητρώα απαιτούμενα για την επίλυση και τελικά η pmacr που περιέχει τελικά και τις υπορουτίνες που κάνουν τη λύση.

3.1.5.2. Η υπορουτίνα pnewprob

Η υπορουτίνα pnewprob καλείται από την pcontr σε δύο περιπτώσεις. Με την εντολή «feap» και με την εντολή «bina». Στην πρώτη περίπτωση καλείται με όρισμα «1» και στη δεύτερη περίπτωση με όρισμα «2». Η διαφορά των ορισμάτων σημαίνει ότι εκτελούνται διαφορετικές ομάδες εντολών σε αυτή.

Στην περίπτωση που καλείται από την εντολή «feap», οπότε έχει και όρισμα «1», αυτό που κάνει είναι να ξεκινάει από την αρχή ένα καινούριο πρόβλημα. Διαγράφει αρχεία, καθαρίζει τη μνήμη και ξεκινά ένα καινούριο πρόβλημα. Επίσης αποδίδει τιμές στα numnp, numel, nummat, ndm, ndf, nen, nad, npd, nud, είτε από τον καθορισμό τους στον τίτλο, είτε μετρώντας τα από τα δεδομένα του καννάβου χρησιμοποιώντας την υπορουτίνα pnums. Στη συνέχεια καθορίζει τις λογικές μεταβλητές που θα χρησιμοποιηθούν παρακάτω. Επίσης τρέχει τις υπορουτίνες rgidb, pmesh, meshck για καθορισμό των ιδιοτήτων του στερεού, ανάγνωση των δεδομένων καννάβου, φόρτισης και συνοριακών συνθηκών και έλεγχό τους

αν έχουν διαβαστεί σωστά και είναι τα δεδομένα επαρκή για να προχωρήσει το πρόβλημα παρακάτω. Τέλος καλεί τη ρουτίνα pextnd για να μετρήσει το clip range των δεδομένων, το πλήθος των στοιχείων που συνδέονται σε κάθε κόμβο και αν υπάρχουν κόμβοι που δεν συνδέονται σε στοιχεία.

3.1.5.3. Η υπορουτίνα pmesh

Η υπορουτίνα pmesh διαβάζει τα δεδομένα που αφορούν τη γεωμετρία του φορέα, τις συνοριακές συνθήκες και τη φόρτισή του. Καλείται από την pnewprob και διαβάζει το κομμάτι του input file που έχει να κάνει με τη γεωμετρική εισαγωγή του φορέα. Για κάθε εντολή που διαβάζει καλεί άλλες ρουτίνες να συνεχίσουν το διάβασμα του αρχείου για το συγκεκριμένο τμήμα του στο οποίο αυτές αναφέρονται. Οι εντολές που αναγνωρίζει είναι οι «coor», «elem», «mate», «boun», «forc», «temp», «end », «prin», «nopr», «titl», «bloc», «pola», «ebou», «angl», «sloa», «cons», «sphe», «btem», «icon», «pars», «nopa», «trib», «para», «efor», «eang», «cbou», «cfor», «cang», «foll», «slav», «rese», «sblo», «curv», «rota», «setn», «setr», «btra», «fpro», «cpro», «regi», «tran», «damp», «mass», «stif», «csur», «ereg», «reac», «manu», «body», «glob», «shif», «disp», «edis», «cdis», «debu», «side», «face», «snod», «blen», «move», «rigi», «moda», «flex», «base», «epro», «mpro», «loop», «next», «file», «cdam», «cmas», «csti», «ebas», «cbas», «eule», «ceul», «rfor», «lfor», «load», «swee», «spin», «peri», «*nod», «*ele», «mes1», «mes2», «mes3», «mes4», «mes5», «mes6», «mes7», «mes8», «mes9», «mes0».

Συγκεκριμένα τα διάφορα τμήματα του input file διαβάζονται από τις ακόλουθες ρουτίνες:

1. pconst
   Διαβάζει από το input file όλα τα blocks εντολών που έχουν να κάνουν με καθορισμό σταθερών και αποθήκευση των τιμών τους. Ενεργοποιείται με την εντολή «para».
2. pmatin
   Διαβάσει από το input file τα blocks εντολών που έχουν να κάνουν με εισαγωγή δεδομένων των υλικών και αποθήκευση των τιμών τους. Ενεργοποιείται με την εντολή «mate».
3. pnodes
   Διαβάσει από το input file τα blocks εντολών που έχουν να κάνουν με εισαγωγή δεδομένων των supernodes και αποθήκευση των τιμών τους. Ενεργοποιείται με την εντολή «snod».
4. psides
   Διαβάσει από το input file τα blocks εντολών που βρίσκονται κάτω από την εντολή side και καθορίζουν την παραγωγή των κόμβων κατά μήκος μιας πλευράς. Ενεργοποιείται με την εντολή «side».
5. pblend
   Διαβάσει από το input file τα blocks εντολών που βρίσκονται κάτω από την εντολή «side» και καθορίζουν την παραγωγή των κόμβων κατά μήκος μιας πλευράς. Ενεργοποιείται με την εντολή «side».
6. plinka
   Διαβάσει από το input file τα blocks εντολών που έχουν να κάνουν με συνοριακές συνθήκες. Δηλαδή ενεργοποιείται με τις εντολές «ebou» και «csur».
7. pblendm
   Παράγει από παρεμβολή τις τελικές συντεταγμένες των κόμβων των πεπερασμένων στοιχείων που καθορίζονται συνήθως με την εντολή «blen».

3.1.5.4. Η υπορουτίνα formfe

Η υπορουτίνα αυτή κατασκευάζει τα μητρώα, όπως αυτά χρειάζονται για την επίλυση του προβλήματος πεπερασμένων στοιχείων. Σε αυτήν υπολογίζονται τα μητρώα δυσκαμψίας και παραμορφώσεων του φορέα, καθώς και συνεκτιμώνται όσοι συγκεντρωμένοι επικόμβιοι όροι υπάρχουν.

Η formfe καλεί την υπορουτίνα pform η οποία κατασκευάζει τα κύρια μητρώα που απαιτούνται για την επίλυση. Στη συνέχεια ακολουθεί η pnodal που προσθέτει συγκεντρωμένους ή επικόμβιους όρους.

Η υπορουτίνα pform κατασκευάζει τα τοπικά μητρώα που απαιτούνται, κάνει τις απαραίτητες τροποποιήσεις λόγω συνοριακών συνθηκών και τελικά τα συνενώνει στο μεγάλο μητρώο ακαμψίας του φορέα. Αρχικά καλεί την υπορουτίνα plocal που δεν παράγει κάτι καινούριο, αλλά τακτοποιεί τα στοιχεία που έχουμε δώσει ήδη, όπως συντεταγμένες κόμβων, φορτία στο φορέα, θερμοκρασίες, κλπ, τοποθετώντας τα σε πίνακες από τους οποίους μπορούμε να τα αναζητήσουμε, αλλά με βάση τον αύξοντα αριθμό του στοιχείου. Ακολούθως καλεί την elmlib που, από το είδος του στοιχείου που έχουμε ορίσει ότι έχουμε και τις συναρτήσεις σχήματός του, παράγει τα τοπικά μητρώα ελαστικότητας, παραμόρφωσης και ακαμψίας του στοιχείου. Στη συνέχεια καλεί την ptlocal που τροποποιεί τα μητρώα αν υπάρχουν εστραμμένοι βαθμοί ελευθερίας και την modify που εφαρμόζει στα τοπικά μητρώα των στοιχείων τις συνοριακές συνθήκες λόγω των στηρίξεων. Τέλος καλεί την passble που κάνει την τελική συνένωση των τοπικών μητρώων και παράγει το τελικό μητρώο ακαμψίας του φορέα.

Η υπορουτίνα elmlib αρχικά μας παραπέμπει στο στοιχείο κατάλληλης διάστασης, ξεχωρίζει δηλαδή αν είμαστε σε μία, δύο, ή τρεις διαστάσεις και μας παραπέμπει στο κατάλληλο στοιχείο. Στο παράδειγμα που εξετάζουμε μας παρέπεμψε στην solid2d. Η τελευταία, αφού ελέγξει για σφάλματα στα δεδομένα που έχουμε εισάγει για το στοιχείο, προχωράει στην κατασκευή του μητρώου ακαμψίας του. Στην περίπτωσή μας το κατάλληλο μοντέλο που χρησιμοποιήσαμε είναι το displacement model στο οποίο η συνάρτηση που παρήγαγε τα μητρώα του είναι η sld2d1.

Τέλος, το τελικό μητρώο δυσκαμψίας τροποποιείται ώστε να μπορέσουν να ενσωματωθούν σε αυτό οι συνθήκες στήριξης. Πριν την τροποποίηση η ορίζουσά του ήταν μηδενική, που σημαίνει ότι το πρόβλημα δεν είχε λύση. Το πρόβλημα όμως ολοκληρώνεται όταν συνεκτιμώνται και οι στηρίξεις και αυτό γίνεται με την κατάλληλη τροποποίηση. Τις εργασίες αυτές τις αναλαμβάνει η υπορουτίνα modify.

3.1.5.5. Η υπορουτίνα sld2d1

Είναι η υπορουτίνα στην οποία υπολογίζεται το μητρώο ακαμψίας με αριθμητική ολοκλήρωση του μητρώου παραμόρφωσης. Αρχικά υπολογίζονται τα σημεία ολοκλήρωσης κατά Gauss και το πλήθος τους μέσω της υπορουτίνας quadr2d. Και στη συνέχεια αρχίζει η ολοκλήρωση, η οποία υλοποιείται με δύο εμφωλευμένους βρόχους do, ο μεν εξωτερικός σαρώνει όλα τα σημεία Gauss, ο δε εσωτερικός όλους τους κόμβους του στοιχείου. Να σημειωθεί ότι για την αριθμητική ολοκλήρωση δε μας ενδιαφέρουν οι αναλυτικές εκφράσεις των συναρτήσεων σχήματος, αλλά οι τιμές τους.

Αρχικά λοιπόν εντός του εξωτερικού βρόχου και πριν τον εσωτερικό καλείται η υπορουτίνα interp2d που στην ουσία υπολογίζει την τιμή των συναρτήσεων σχήματος για το δεδομένο σημείο. Οι τιμές των συναρτήσεων σχήματος αποθηκεύονται σε ένα τρισδιάστατο μητρώο στο οποίο κάθε επίπεδο αναφέρεται και σε ξεχωριστό σημείο. Κάθε επίπεδο είναι ένας δυσδιάστατος πίνακας που στην πρώτη γραμμή έχει για το δεδομένο σημείο στο οποίο αναφέρεται, τις τιμές της παραγώγου των συναρτήσεων σχήματος ως

προς την πρώτη διάσταση (ως προς x δηλαδή), στη δεύτερη γραμμή ως προς την άλλη (ως προς y) και στην τρίτη τις κανονικές τιμές της μη παραγωγισμένης συνάρτησης σχήματος[14].

Στη συνέχεια καλείται η υπορουτίνα modsld, η οποία υπολογίζει το μητρώο ελαστικότητας του στοιχείου. Στην πραγματικότητα το μητρώο αυτό δεν είναι ένα απλό 6x6 μητρώο, όπως θα περίμενε κανείς, αλλά ένα 6x6x5 μητρώο του οποίου κάθε επίπεδο αποτελεί και ξεχωριστό μητρώο ελαστικότητας ανάλογα με την περίσταση που χρησιμοποιείται.

Τέλος υλοποιούνται οι απαιτούμενοι πολλαπλασιασμοί μητρώων για την παραγωγή του τελικού μητρώου ακαμψίας. Να παρατηρηθεί ότι επειδή τα μητρώα των συναρτήσεων σχήματος δεν είναι ακριβώς αποθηκευμένα όπως στη θεωρία, οι πολλαπλασιασμοί δεν γίνονται με τον κλασσικό μαθηματικό τρόπο πολλαπλασιασμού πινάκων, αλλά εν μέρει με απλές πράξεις άθροισης επιλεκτικών γινομένων.

3.1.5.6. Η υπορουτίνα pmacr

Η υπορουτίνα pmacr καλείται από την pcontr και περιέχει τις υπορουτίνες που κάνουν την τελική λύση και εμφάνιση των αποτελεσμάτων, είτε γραφικά, είτε με απλή τύπωση σε αρχείο. Σαν όρισμα δέχεται τη λογική μεταβλητή initf που αν είναι .true. η λύση αρχικοποιείται, αλλιώς όχι. Χρησιμοποιεί τις εξής υπορουτίνες:

1. pmacio
   Χρησιμοποιείται για να διαβάζει από το input file τις εντολές για επίλυση. Δηλαδή ποια μεγέθη θα υπολογίσει και αν χρειάζεται να χρησιμοποιήσει κάποια ιδιαίτερη μέθοδο.
2. pmacr1
   Η βασική της λειτουργία είναι ότι κάνει την επίλυση μέσω της υπορουτίνας psolve η οποία και επιλύει τα μητρώα που απαρτίζουν το τελικό γραμμικό σύστημα. Πριν από αυτό υπολογίζει και τις δράσεις παγίωσης με την υπορουτίνα pload.
3. pmacr2
   Ασχολείται κυρίως με την αναγνώριση και υλοποίηση των βρόχων «loop» στο input file και την προσαύξηση των βημάτων στην περίπτωση που έχουμε proportional loading.
4. pmacr3
   Ασχολείται με την τύπωση των αποτελεσμάτων είτε σε αρχείο, είτε στην οθόνη με γραφικό τρόπο.
5. pmacr4
   Ασχολείται με την υλοποίηση εντολών επίλυσης που έχει ορίσει ο χρήστης. Για το σκοπό αυτό καλεί την υπορουτίνα umaclib που είναι η βιβλιοθήκη των εντολών που έχει ορίσει ο χρήστης.

3.1.6. Προσθήκη νόμων υλικού από το χρήστη

Το FEAP δίνει τη δυνατότητα στο χρήστη να χρησιμοποιεί νόμους υλικών πέρα από αυτά που έχει το ίδιο περασμένα μέσα. Αυτό υλοποιείται σχετικά εύκολα, γιατί το FEAP για

[14] Υπενθυμίζουμε ότι στο συγκεκριμένο παράδειγμα το στοιχείο είναι δυσδιάστατο. Αν σε άλλο παράδειγμα είχαμε και τρισδιάστατα στοιχεία, τότε προφανώς θα είχαμε και παράγωγο ως προς την τρίτη διάσταση (ως προς z) και το πλήθος των γραμμών του κάθε επιπέδου δεν θα ήταν 3 όπως τώρα, αλλά 4.

κάθε υλικό έχει και μία υπορουτίνα που λαμβάνει το διάνυσμα της παραμόρφωσης και επιστρέφει το διάνυσμα των τάσεων και το μητρώο ελαστικότητας. Εκτός λοιπόν από τις υπορουτίνες που έχει και χρησιμοποιούνται για τους νόμους που έχει από μόνο του περασμένους μέσα, έχει και υπορουτίνες που είναι κενές και αναφέρονται σε νόμους που θέλει να ορίσει ο χρήστης.

Η διαδικασία που κάνουμε για να ορίσουμε ένα υλικό είναι η εξής: Από την ομάδα των υποπρογραμμάτων με όνομα umati1, umati2, …., umati9, umati0 επιλέγουμε αυτό που μας ενδιαφέρει. Το FEAP μας δίνει δυνατότητα να ορίσουμε μέχρι 10 υλικά. Ασφαλώς με μία πιο ευρείας κλίμακας παρέμβαση στον κώδικα μπορούμε να τον μετατρέψουμε να δέχεται όσα θέλουμε, αλλά αυτό ξεφεύγει από τους στόχους μας αυτή τη στιγμή. Η υπορουτίνα αυτή λαμβάνει τρία ορίσματα. Το ένα είναι το type και είναι η ονομασία του μοντέλου του υλικού, το vv που περιέχει σε διαδοχικά κελιά τους αριθμούς που εμείς ορίσαμε ως παραμέτρους του από το input file και το d το οποίο περιέχει τις παραμέτρους που κρατάει από μόνο του τα FEAP για κάθε υλικό. Από τις τιμές του vv υπολογίζουμε τις παραμέτρους που χρειαζόμαστε για το μοντέλο μας και τις αποθηκεύουμε στο μητρώο ud, το οποίο και αποδίδεται από την υπορουτίνα. Συνήθως αυτός ο υπολογισμός δεν είναι παρά μια απλή ανάθεση, γιατί συνήθως δίνουμε τις παραμέτρους που θέλουμε έτοιμες. Εκτός από τη ud η υπορουτίνα αποδίδει και τις τιμές των μεταβλητών n1 και n3, οι οποίες όμως έχουν να κάνουν με το πόσες από τις παλιές τιμές ορισμένων μεταβλητών θα διατηρηθούν σαν ιστορικό, αντί να διαγραφούν. Δεν επηρεάζει όμως αυτό τους υπολογισμούς μας.

Στη συνέχεια εντοπίζουμε την ομάδα των υποπρογραμμάτων umatl1, umatl2, …., umatl9, umatl0 και επιλέγουμε το αντίστοιχο με αυτό της προηγούμενης ομάδας που έχουμε ήδη επιλέξει από αυτή. Η δουλειά της υπορουτίνας αυτής είναι να λαμβάνει το διάνυσμα των παραμορφώσεων και να επιστρέφει το διάνυσμα των τάσεων που αντιστοιχεί σε αυτό, καθώς και το μητρώο ελαστικότητας για την τρέχουσα φόρτιση. Αναλόγως το νόμο που χρησιμοποιούμε, είναι στη δική μας ευχέρεια να κατασκευάσουμε τον κώδικα από τον οποίο θα υπολογίζονται οι τάσεις από τις παραμορφώσεις έχοντας με αυτόν τον τρόπο τη δυνατότητα να υλοποιήσουμε ακόμα και πολύπλοκους νόμους υλικών[15]. Ενδιαφέρον παρουσιάζει το γεγονός ότι ο ορισμός ενός νόμου υλικών έχει να κάνει με το μητρώο ελαστικότητάς του, οπότε δεν χρειάζεται να τον τροποποιούμε κάθε φορά για δυσδιάστατες και τρισδιάστατες αναλύσεις, ούτε για χρήση άλλων στοιχείων.

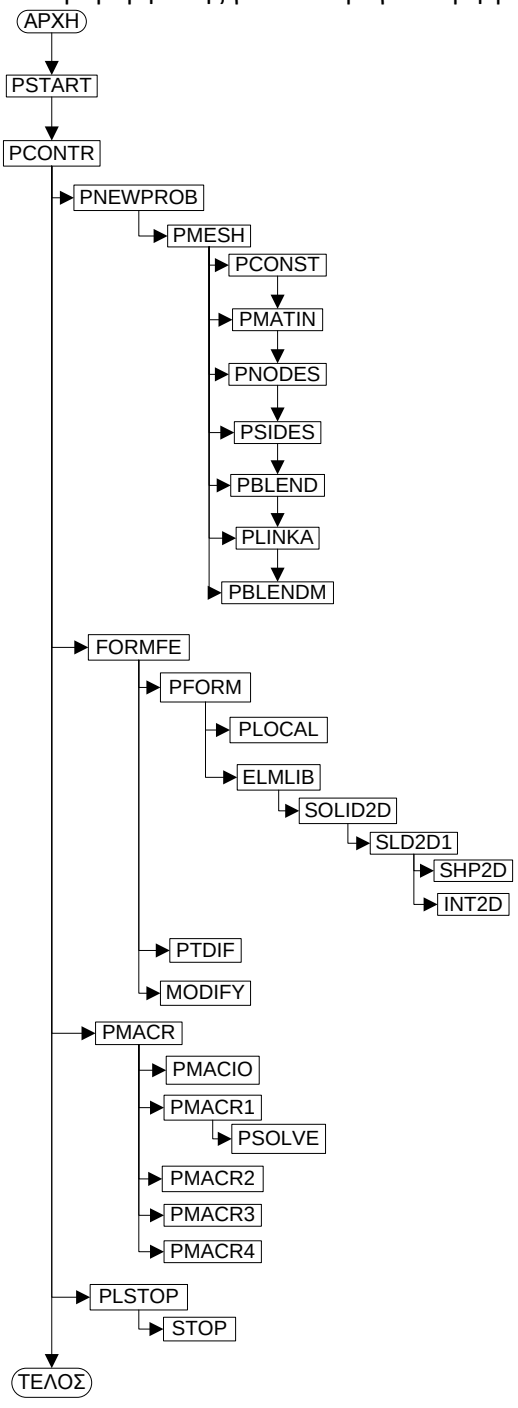
3.2. Συμπεράσματα

Έχοντας λοιπόν την προηγούμενη εικόνα για τα υποπρογράμματα που χρησιμοποιεί το FEAP μπορούμε να τα σχηματοποιήσουμε στο διάγραμμα που φαίνεται στην *Εικόνα 3.10*. Παρατηρούμε τον τρόπο με τον οποίο είναι δομημένο συνολικά το πρόγραμμα, ασφαλώς πολύ επιγραμματικά. Στόχος μας είναι να δείξουμε ότι και το FEAP δεν ξεφεύγει από την κλασική δομή ενός τυπικού προγράμματος πεπερασμένων στοιχείων. Πάρα πολλές υπορουτίνες έχουν παραλειφθεί, αλλά η προσθήκη τους θα αύξανε πολύ το βαθμό πολυπλοκότητας, απομακρύνοντάς μας από την ουσία. Σαφώς και το FEAP διαθέτει προηγμένες λειτουργίες, όπως ανάγνωση από input file ή interactive mode, επίλυση δυναμικών προβλημάτων, έλεγχοι δεδομένων και απαλοιφή εσωτερικών αριθμητικών σφαλμάτων και άλλα που ασφαλώς ξεφεύγουν από την προσέγγιση που επιθυμούμε.

Για λόγους πληρότητας να συμπληρώσουμε κάποια υποπρογράμματα που φαίνονται στο διάγραμμα, αλλά δεν έχουν εξηγηθεί μέχρι τώρα. Αναφερόμαστε στην υπορουτίνα elmlib. Η υπορουτίνα αυτή είναι στην ουσία μια βιβλιοθήκη στοιχείων, για την

[15] Για την περίπτωση του δικού μας κώδικα δες και παράρτημα Ι, σελίδα 79.

ακρίβεια βιβλιοθήκη συναρτήσεων σχήματος για κάθε τύπο στοιχείου, μονοδιάστατο (solid1d), δυσδιάστατο (solid2d) και τρισδιάστατο (solid3d). Το πρόβλημά μας είναι δυσδιάστατο, άρα ενεργοποιείται η solid2d που αντιστοιχεί σε αυτό. Στη συνέχεια η υπορουτίνα solid2d που έχει διάφορους τύπους πεπερασμένων στοιχείων ενεργοποιεί την sld2d1 που αντιστοιχεί στον τύπο του στοιχείου που έχουμε και εκτελεί τις απαραίτητες διεργασίες που απαιτούνται για αυτό όπως αυτές έχουν ήδη περιγραφεί. Οι τιμές των κατάλληλων συναρτήσεων σχήματος υπολογίζονται από την shp2d που αντιστοιχεί στο στοιχείο που έχουμε στο παράδειγμά μας και τέλος η υπορουτίνα int2d εκτελεί την ολοκλήρωση του μητρώου παραμόρφωσης για να παράγει το μητρώο δυσκαμψίας.

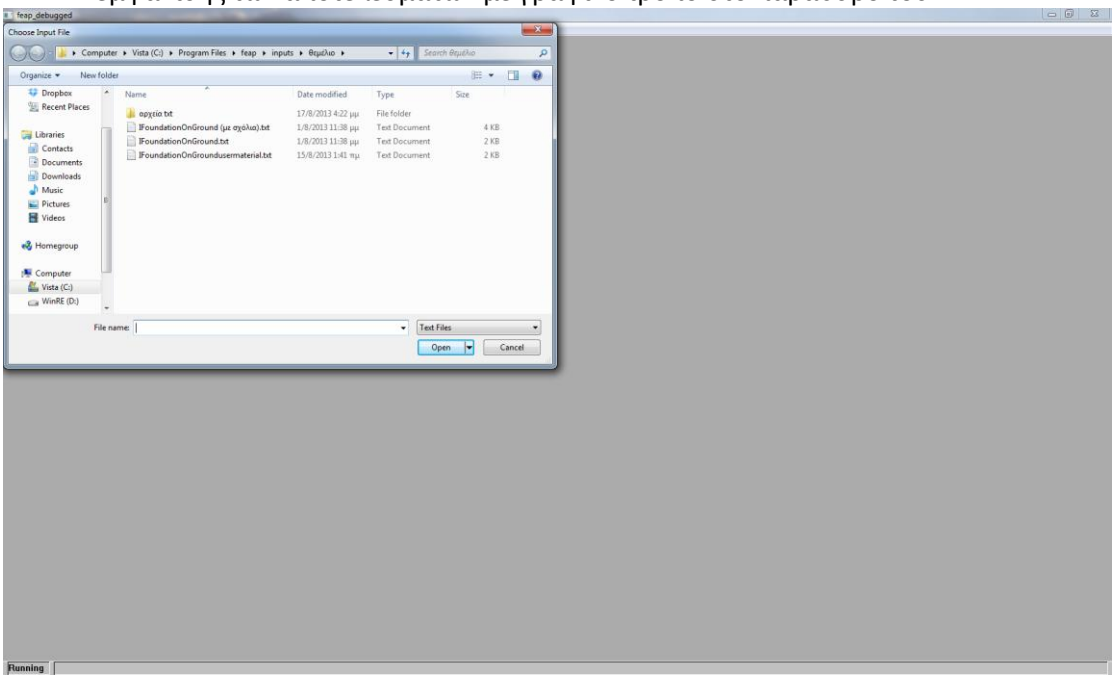


Εικόνα 3.10: Συνοπτικό διάγραμμα ροής FEAP.

3.3. Χρήση του FEAP

Το πρόγραμμα όπως έχουμε πει λαμβάνει τα δεδομένα από ένα αρχείο εισόδου δεδομένων το οποίο έχουμε δημιουργήσει εμείς με το χέρι, είτε με κάποιο άλλο πρόγραμμα. Η επιλογή του γίνεται από μια αρχική οθόνη όπως φαίνεται στην *Εικόνα 3.11* που μας καλεί να το επιλέξουμε ανάμεσα στα άλλα. Το αρχείο αυτό περιέχει όλες τις εντολές που εμείς θέλουμε να εκτελεστούν και είναι 6 κατηγοριών[16]:

1. Εντολές έναρξης καινούριου προβλήματος.
2. Εντολές καταχώρησης των υλικών που θα χρησιμοποιηθούν.
3. Εντολές δημιουργίας της γεωμετρίας του καννάβου των πεπερασμένων στοιχείων και της διακριτοποίησής τους.
4. Εντολές επιβολής της φόρτισης και των συνοριακών συνθηκών στήριξης.
5. Εντολές επίλυσης του μοντέλου.
6. Εντολές τύπωσης των αριθμητικών αποτελεσμάτων σε κάποιο αρχείο και εντολές εμφάνισης των αποτελεσμάτων με γραφικό τρόπο στο παράθυρο του FEAP.

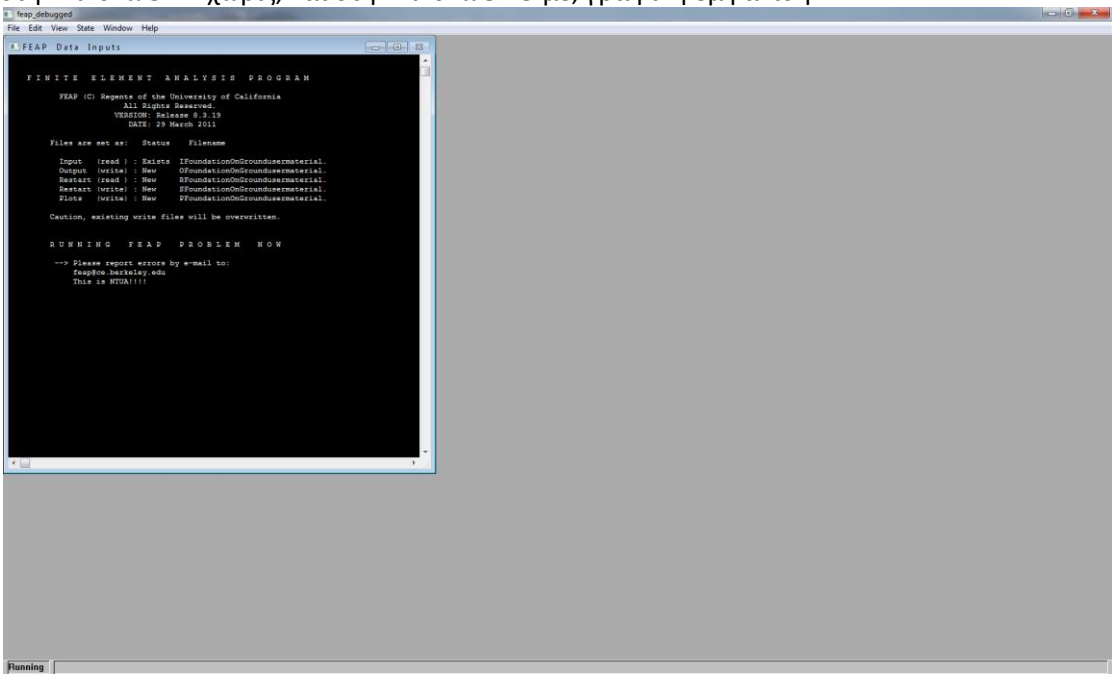


Εικόνα 3.11: Παράθυρο επιλογής αρχείου εισόδου δεδομένων.

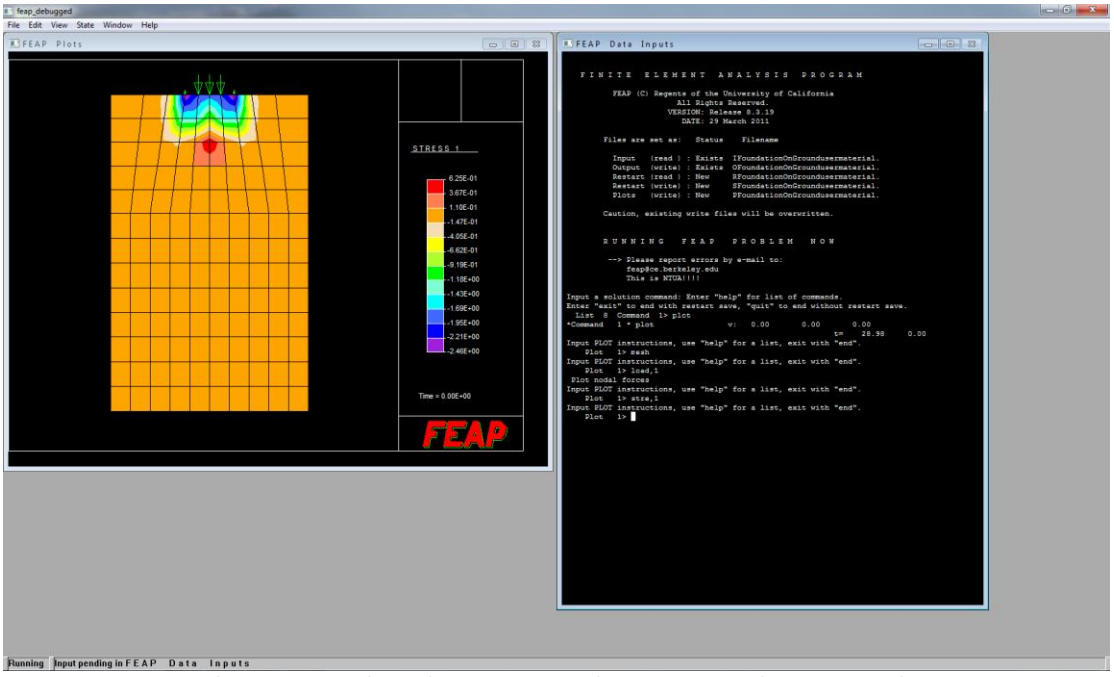
Δεν είναι απαραίτητο όμως να δώσουμε όλες τις εντολές που θέλουμε να εκτελεστούν στο αρχείο εισόδου δεδομένων. Το FEAP μας δίνει τη δυνατότητα να του δίνουμε τις εντολές που θέλουμε να εκτελέσει σε πραγματικό χρόνο μέσω του «interactive mode». Όταν αυτό είναι ενεργοποιημένο τότε μετά την εκτέλεση των εντολών του αρχείου εισόδου δεδομένων υπάρχει ένα παράθυρο το οποίο δέχεται τις εντολές που θέλουμε να δώσουμε σε μορφή γραμμής εντολών και εκεί μπορούμε να δούμε κάθε εντολή να υλοποιείται. Αυτό είναι ιδιαίτερα χρήσιμο, όταν οι εντολές που θέλουμε να δώσουμε εξαρτώνται από τα αποτελέσματα που θα πάρουμε. Παρ' όλ' αυτά στο αρχείο εισόδου δεδομένων θα πρέπει τουλάχιστον να ορίσουμε τη γεωμετρία του καννάβου, τη φόρτιση και τις συνοριακές συνθήκες στήριξης και στη συνέχεια, εάν θέλουμε, τις εντολές επίλυσης μέσω interactive mode. Εάν θέλουμε όμως να εμφανίσουμε και γραφικά τα αποτελέσματα τότε, τα γραφικά εμφανίζονται σε ξεχωριστό παράθυρο δίπλα από το παράθυρο των εντολών του προγράμματος, ή στο ίδιο ανάλογα με τη ρύθμιση που έχουμε κάνει στην

[16] Για περισσότερες πληροφορίες και λεπτομέρειες δες το user manual του προγράμματος.

εγκατάσταση του προγράμματος[17]. Έτσι η τελική εμφάνιση είναι περίπου όπως φαίνεται στην *Εικόνα 3.12* χωρίς, και στην *Εικόνα 3.13* με, γραφική εμφάνιση.



Εικόνα 3.12: Τελική εμφάνιση του προγράμματος χωρίς εμφάνιση γραφικών.



Εικόνα 3.13: Τελική εμφάνιση του προγράμματος με εμφάνιση γραφικών.

[17] Δες στο installation manual του FEAP υποσημείωση στη σελίδα 8. Σε κάθε περίπτωση εμείς προτείνουμε τα γραφικά να εμφανίζονται στο δικό τους παράθυρο.

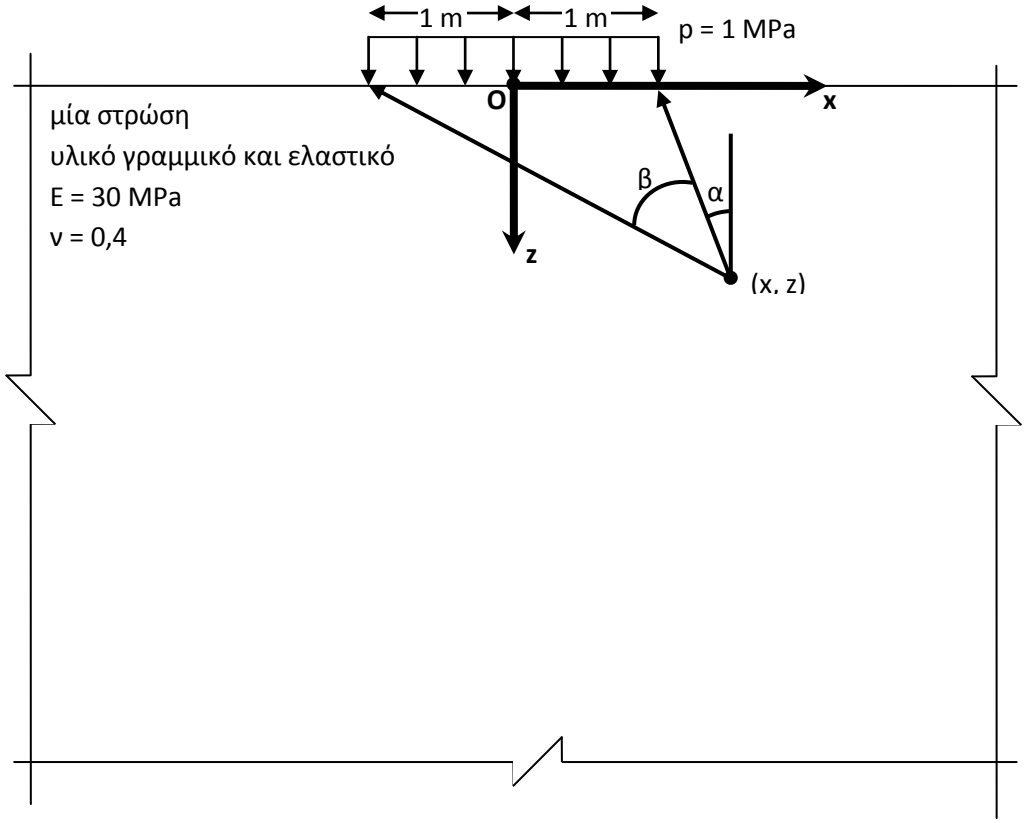
<u>Κεφάλαιο 4:</u>
Παραδείγματα που επιλύθηκαν.

4.1. Σύγκριση αριθμητικών και αναλυτικών λύσεων

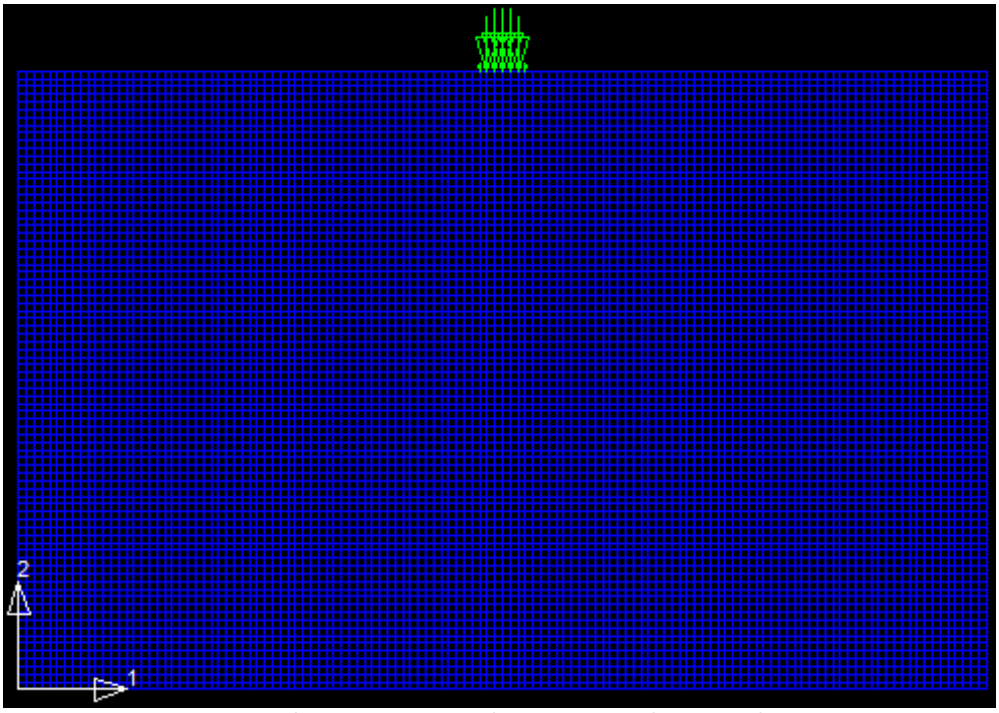
4.1.1. Εισαγωγή

Πριν λύσουμε κάποιο παράδειγμα, ιδιαίτερη σημασία έχει το να μελετήσουμε τα αποτελέσματα που παράγει το FEAP και να τα συγκρίνουμε με αναλυτικές λύσεις που είναι διαθέσιμες. Αυτό είναι εφικτό μόνο στην περίπτωση του ελαστικού ημίχωρου που υπάρχουν κλειστές αναλυτικές λύσεις. Για τη σύγκριση θα λύσουμε τον ελαστικό ημίχωρο φορτιζόμενο με ομοιόμορφο κατανεμημένο φορτίο αριθμητικά χρησιμοποιώντας το FEAP και εν συνεχεία αναλυτικά εφαρμόζοντας τους τύπους του Boussinesq.

Να σημειωθεί ότι, επειδή το FEAP δεν δέχεται απειρομήκη στοιχεία, έχουν χρησιμοποιηθεί πεπερασμένες διαστάσεις καννάβου, αρκετά μεγάλες όμως ώστε να μην προκαλούν σημαντικές αποκλίσεις. Συγκεκριμένα έχει χρησιμοποιηθεί ημιπλάτος λωριδωτού φορτίου $R = 1\ m$, ύψος $H = 32 * R = 32\ m$ και οριζόντιο πλάτος $L = 50 * R = 50\ m$. Η διακριτοποίηση που έχουμε κάνει είναι 126 στοιχεία κατά την οριζόντια διεύθυνση, και 80 κατά την κατακόρυφη, δηλαδή οι διαστάσεις του πεπερασμένου στοιχείου είναι $0{,}4 \times 0{,}4\ m^2$. Το πρόβλημα που επιλύουμε και η αντίστοιχη διακριτοποίησή του από το FEAP παρουσιάζονται στις εικόνες *4.1* και *4.2*.

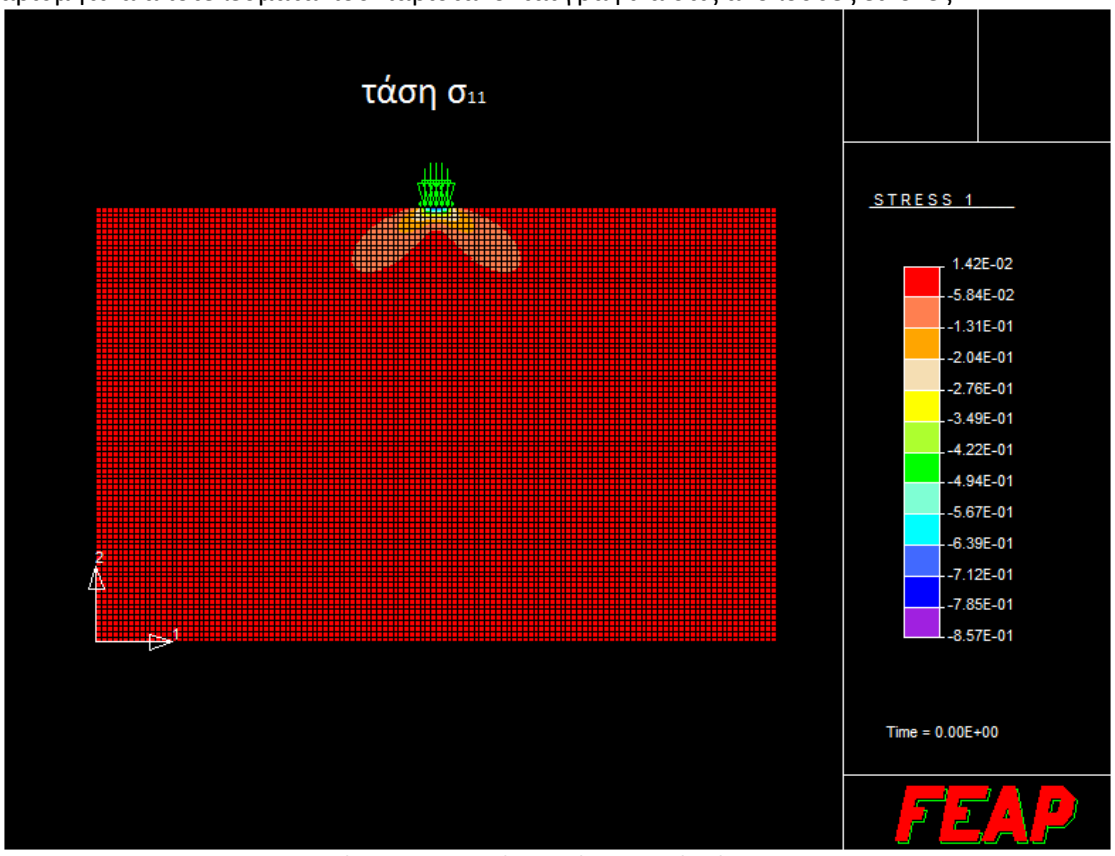


Εικόνα 4.1: Παρουσίαση προβλήματος

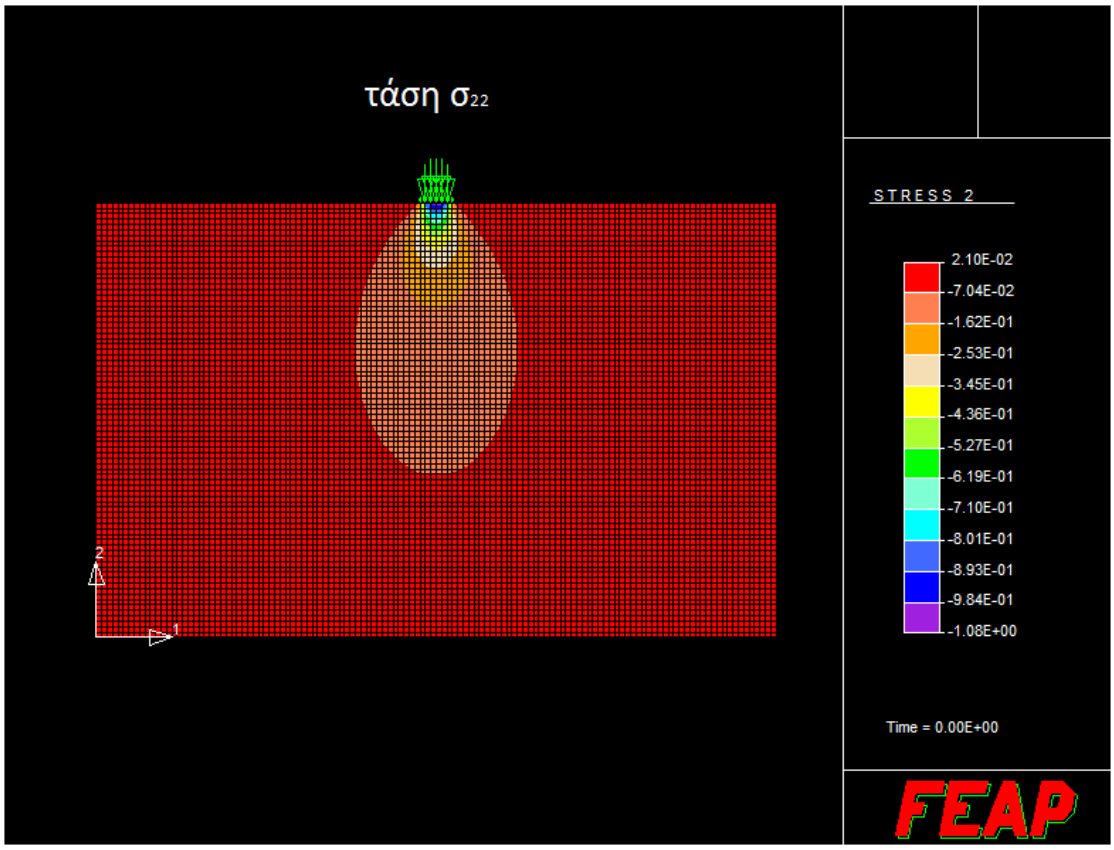


Εικόνα 4.2: Διακριτοποίηση του προβλήματος από το FEAP

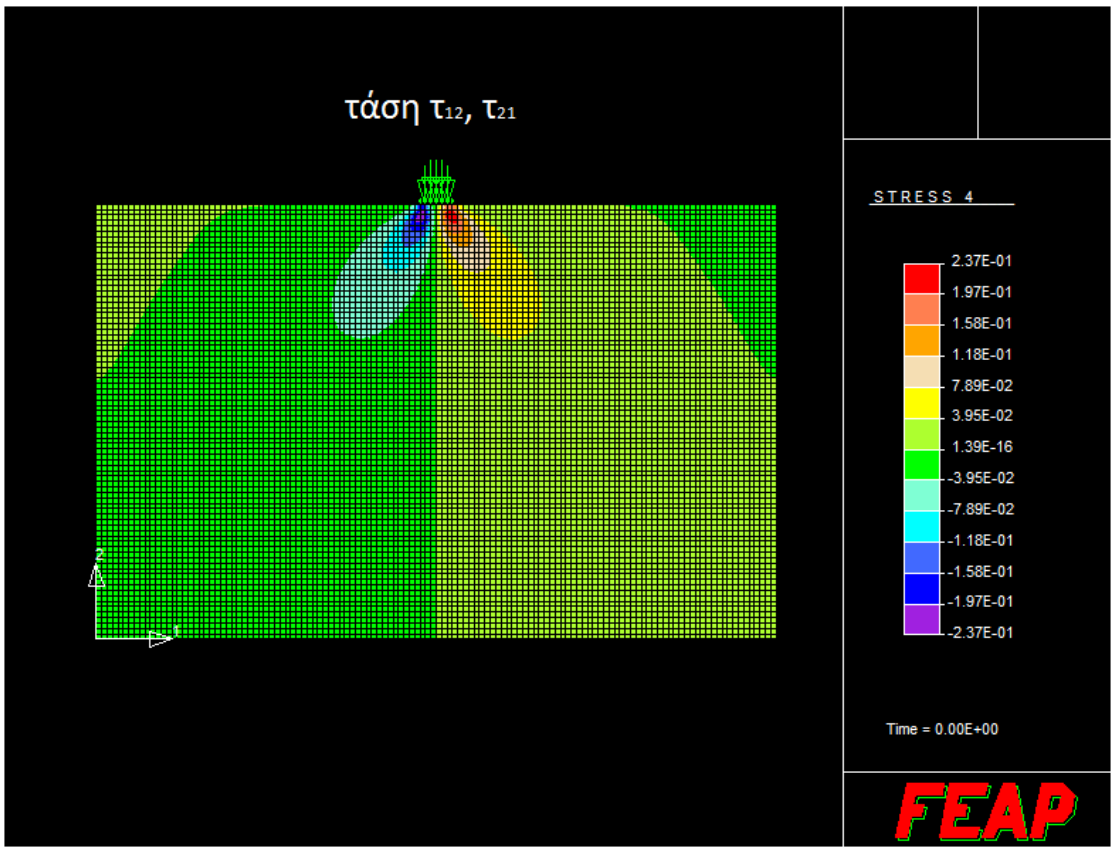
Στη συνέχεια εισάγουμε το φορέα στο πρόγραμμα FEAP και λαμβάνουμε τα αριθμητικά αποτελέσματα που παριστάνονται γραφικά στις ακόλουθες εικόνες:



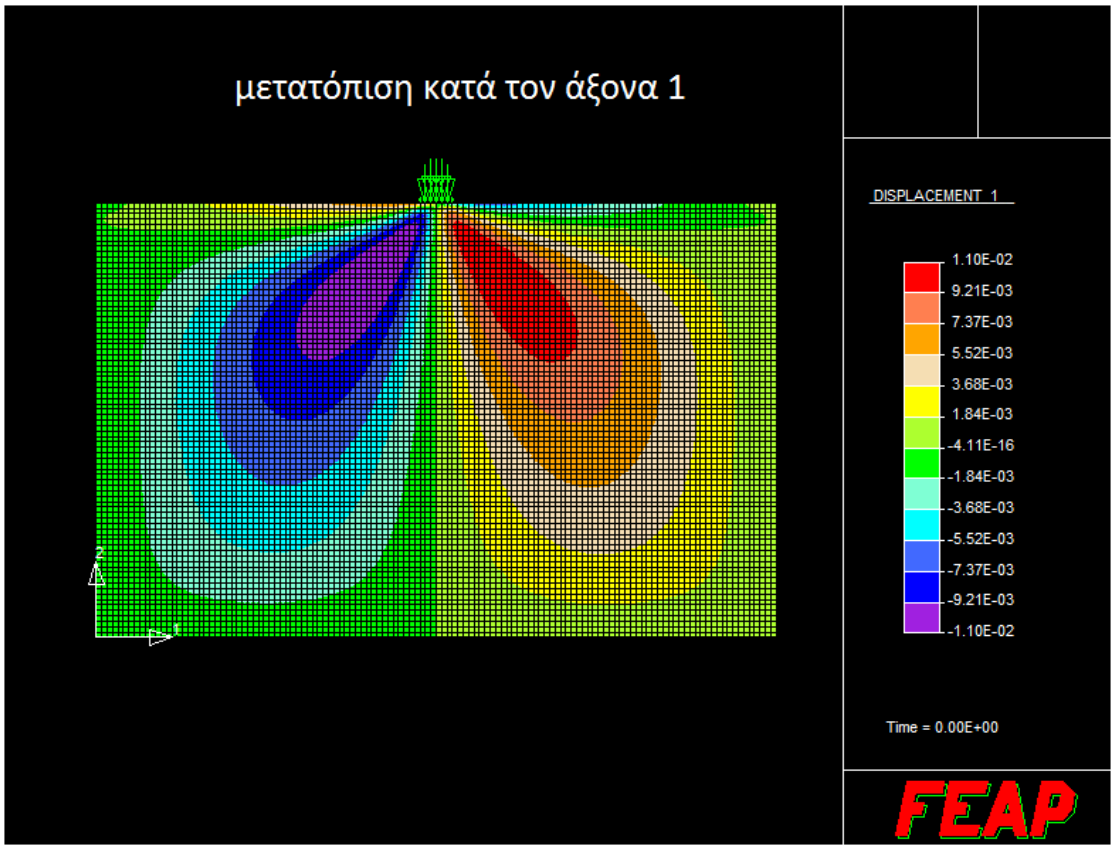
Εικόνα 4.3: Γραφική απεικόνιση ορθής τάσης σ_{11}



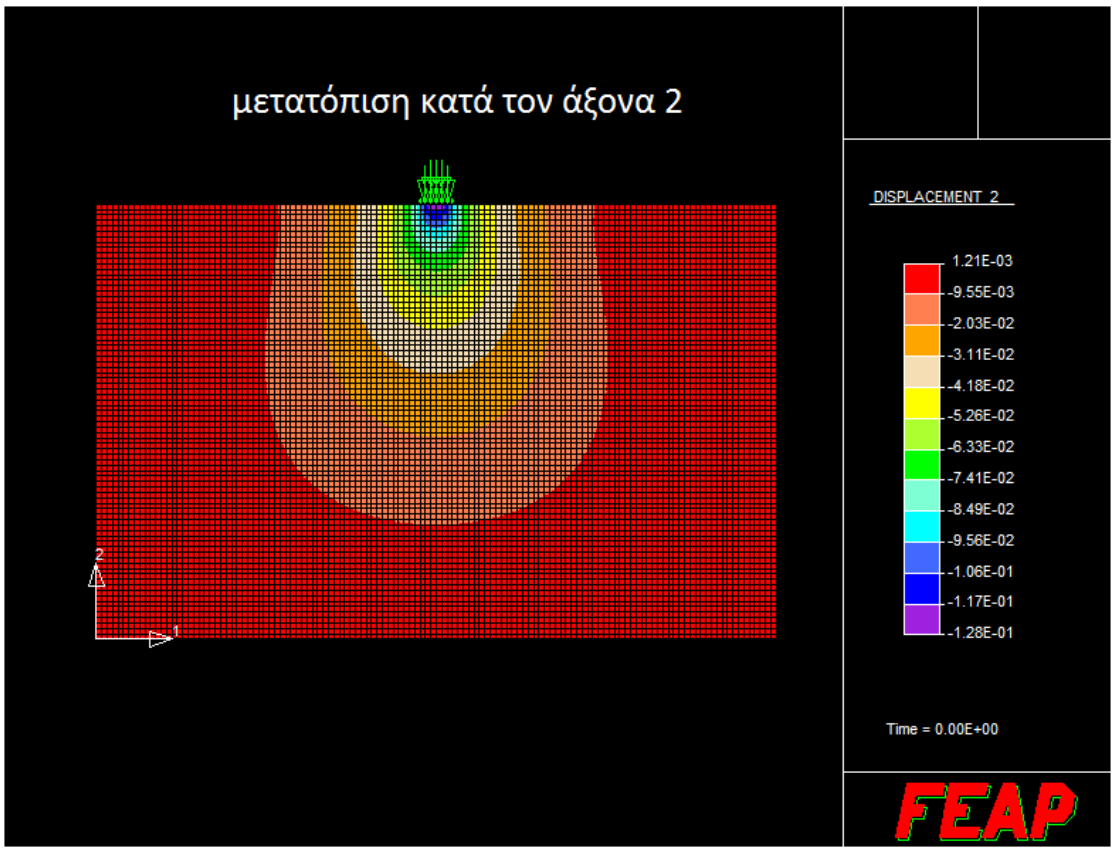
Εικόνα 4.4: Γραφική απεικόνιση ορθής τάσης σ_{22}



Εικόνα 4.5: Γραφική απεικόνιση διατμητικών τάσεων τ_{12} και τ_{21} ($\tau_{12} = \tau_{21}$)



Εικόνα 4.6: Γραφική απεικόνιση των μετατοπίσεων κατά τον οριζόντιο άξονα



Εικόνα 4.7: Γραφική απεικόνιση των μετατοπίσεων κατά τον κατακόρυφο άξονα

4.1.2. Σύγκριση με αναλυτικές λύσεις

Για να γίνει η σύγκριση με τις αναλυτικές λύσεις μπορούμε να χρησιμοποιήσουμε τα αριθμητικά αποτελέσματα που παράγονται και βρίσκονται στο output file του παραδείγματός μας. Τα αποτελέσματα αυτά αναφέρονται στα Gauss points κάθε πεπερασμένου στοιχείου, που είναι τέσσερα για κάθε στοιχείο. Για τα ίδια σημεία θα υπολογίσουμε τις λύσεις που μας δίνουν οι αναλυτικές σχέσεις και θα συγκρίνουμε πόσο κοντά βρίσκονται οι λύσεις μας. Η σύγκριση θα γίνει μόνο για τάσεις, οι οποίες υπολογίζονται αναλυτικά από τους τύπους του Boussinesq, ύστερα από κατάλληλο μετασχηματισμό των αξόνων μεταξύ αυτών που χρησιμοποιεί το FEAP και η θεωρία.
Ο μετασχηματισμός που απαιτείται για τους άξονες είναι:

$$y = x_1 - \frac{L}{2}$$
$$z = H - x_2$$

Και αυτές οι συντεταγμένες εισάγονται στους τύπους του Boussinesq:

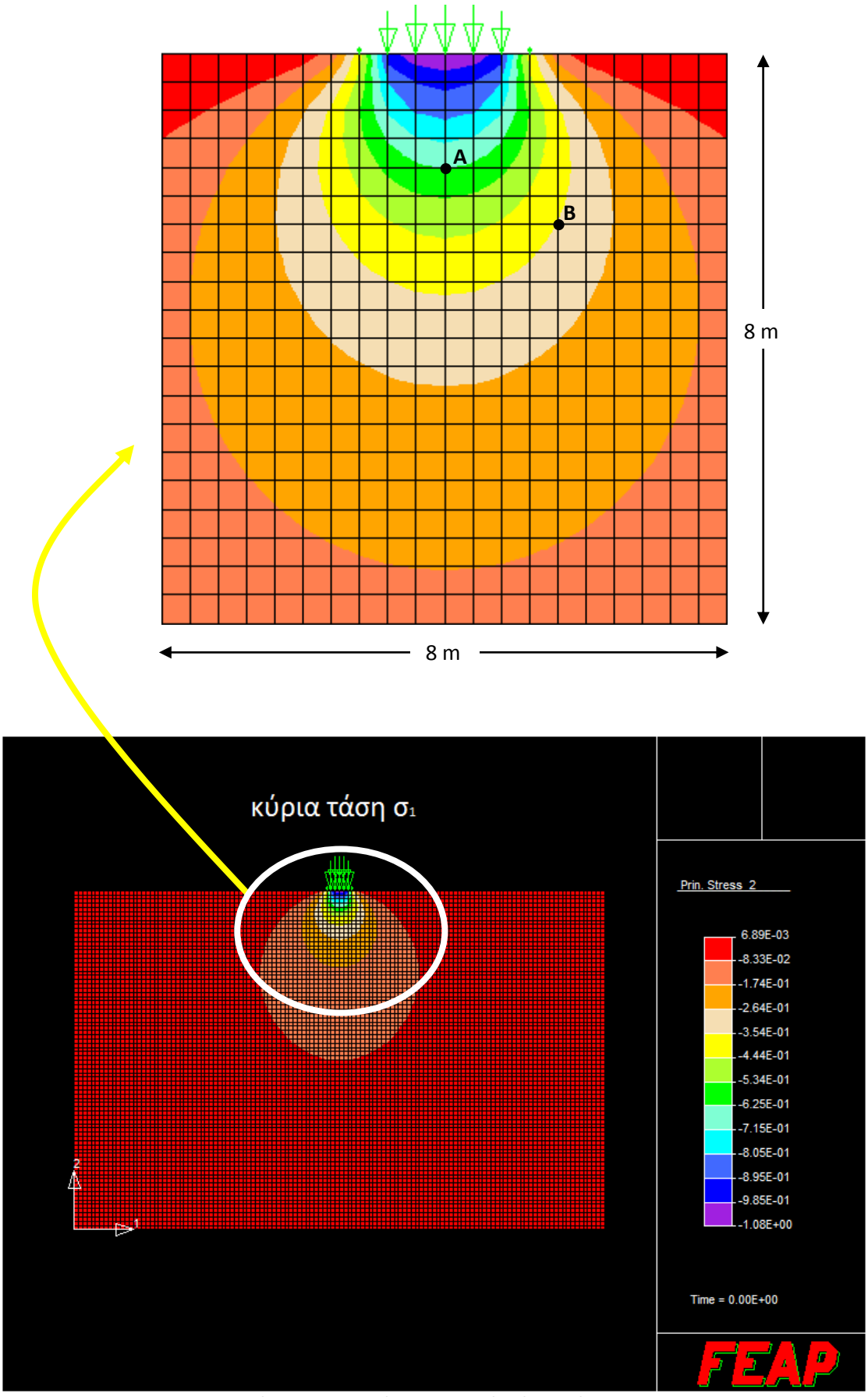
$$\alpha = \operatorname{atan}\left(\frac{x - R}{y}\right)$$
$$\beta = \operatorname{atan}\left(\frac{x + R}{y}\right) - \alpha$$
$$\sigma_{11} = -\frac{P}{\pi} * (\beta - \sin(\beta) * \cos(2 * \alpha + \beta))$$
$$\sigma_{22} = \frac{P}{\pi} * (\beta + \sin(\beta) * \cos(2 * \alpha + \beta))$$
$$\tau_{12} = -\frac{P}{\pi} * \sin(\beta) * \sin(2 * \alpha + \beta)$$

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε τα διαθέσιμα νομογραφήματα των κυρίων τάσεων και να τα συγκρίνουμε με τα παραγόμενα αποτελέσματα από το FEAP. Τα νομογραφήματα περιέχουν στον οριζόντιο και κατακόρυφο άξονα τις αδιαστατοποιημένες παραμέτρους $^{y}/_{R}$ και $^{z}/_{R}$ αντίστοιχα και δίνουν την τιμή του αδιαστατοποιημένου παράγοντα $^{\sigma_i}/_{P}$, για $i = 1, 3$. Στην περίπτωσή μας βέβαια που $R = 1\ m$ και $P = 1\ MPa$, τα αποτελέσματα προκύπτουν απευθείας.

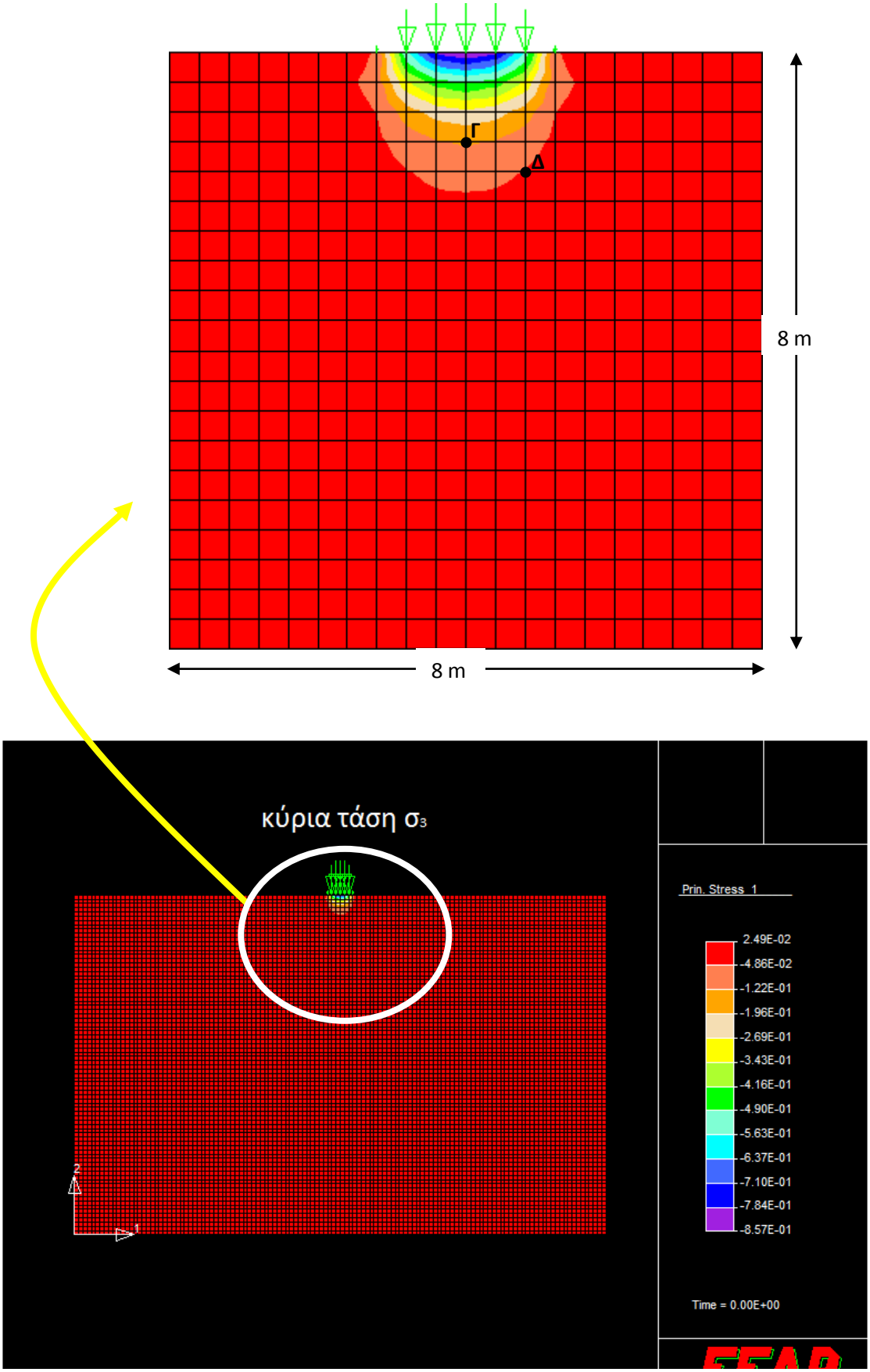
Στην *Εικόνα 4.8* παρατηρούμε τα αποτελέσματα του FEAP για τη μέγιστη κύρια τάση σ_1. Να σημειωθεί ότι, σε αντίθεση με την κλασσική σύμβαση της εδαφομηχανικής που για ευκολία θεωρούμε θετικές τις θλιπτικές τάσεις, στο FEAP θετικές θεωρούνται οι εφελκυστικές τάσεις. Έτσι, ενώ για μας η μέγιστη κατ' απόλυτη τιμή θλιπτική τάση θεωρείται ως η σ_1 και η ελάχιστη ως η σ_2, για το FEAP θεωρούνται αντίστροφα, αφού όσο πιο μεγάλη απόλυτη τιμή έχει ένας αρνητικός αριθμός, τόσο πιο μικρός αλγεβρικά είναι. Σε αυτό οφείλεται και η ασυμφωνία των ονομάτων των κυρίων τάσεων που τους δίνουμε εμείς σε σχέση με αυτά που δίνονται από το FEAP. Σε κάθε περίπτωση, εμείς θα ακολουθήσουμε τη σύμβαση της εδαφομηχανικής. Για τη σύγκριση θα θεωρήσουμε δύο σημεία για κάθε περίπτωση.

Για την περίπτωση ελέγχου της τάσης σ_1 θα χρησιμοποιήσουμε δυο σημεία $A(y = 0, z = 1{,}6)$ και $B(y = 1{,}6, z = 2{,}4)$ για τα οποία παρατηρούμε ότι ${\sigma_1}^A = 0{,}625\ MPa$ και ${\sigma_1}^B = 0{,}354\ MPa$ (*Εικόνα 4.8*). Οι τιμές αυτές επαληθεύονται από το νομογράφημα.

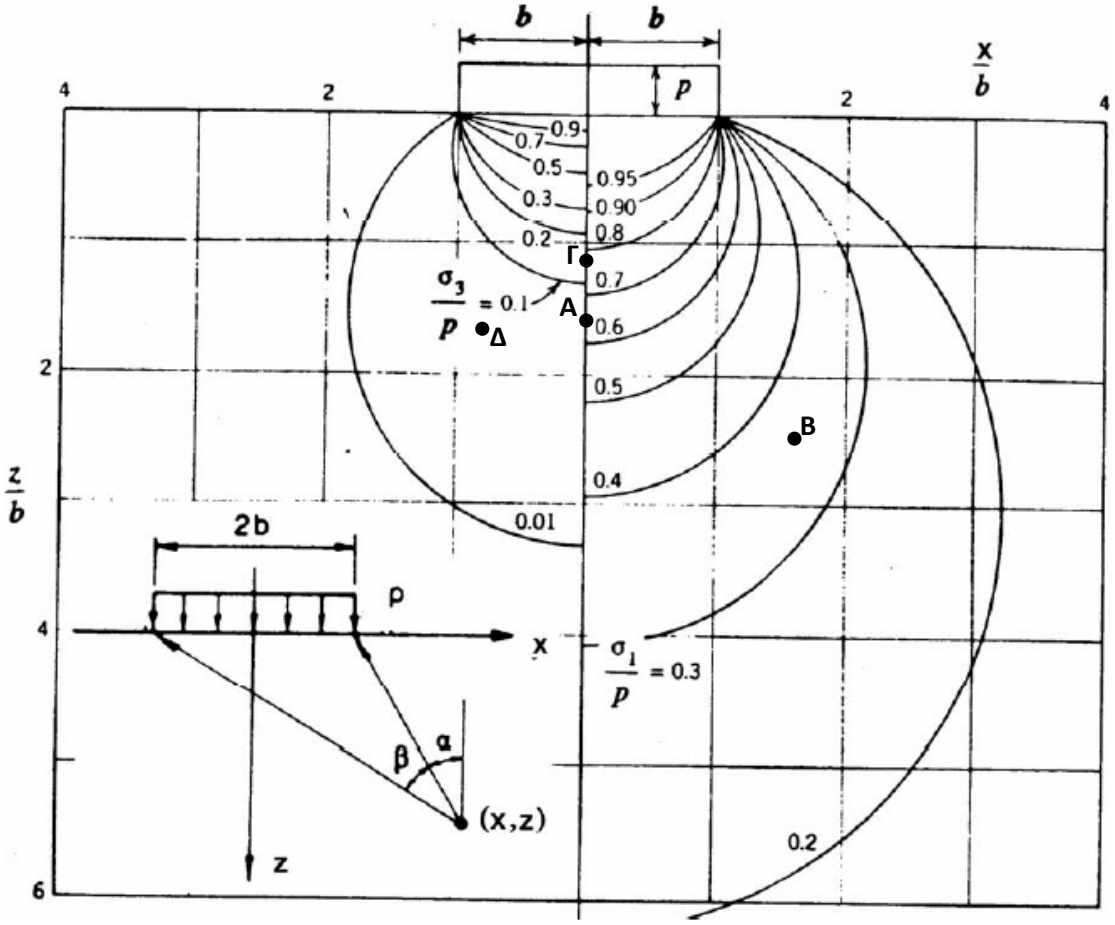
Για την περίπτωση τώρα ελέγχου της τάσης σ_3 θα χρησιμοποιήσουμε δύο σημεία $\Gamma(y = 0, z = 1{,}2)$ και $\Delta(y = 0{,}8, z = 1{,}6)$ για τα οποία παρατηρούμε ότι ${\sigma_3}^\Gamma = 0{,}122\ MPa$ και ${\sigma_3}^\Delta = 0{,}0486\ MPa$ (*Εικόνα 4.9*). Και πάλι οι δύο τιμές επαληθεύονται από το νομογράφημα (*Εικόνα 4.10*).



Εικόνα 4.8: Γραφικός υπολογισμός κύριας τάσης σ_1



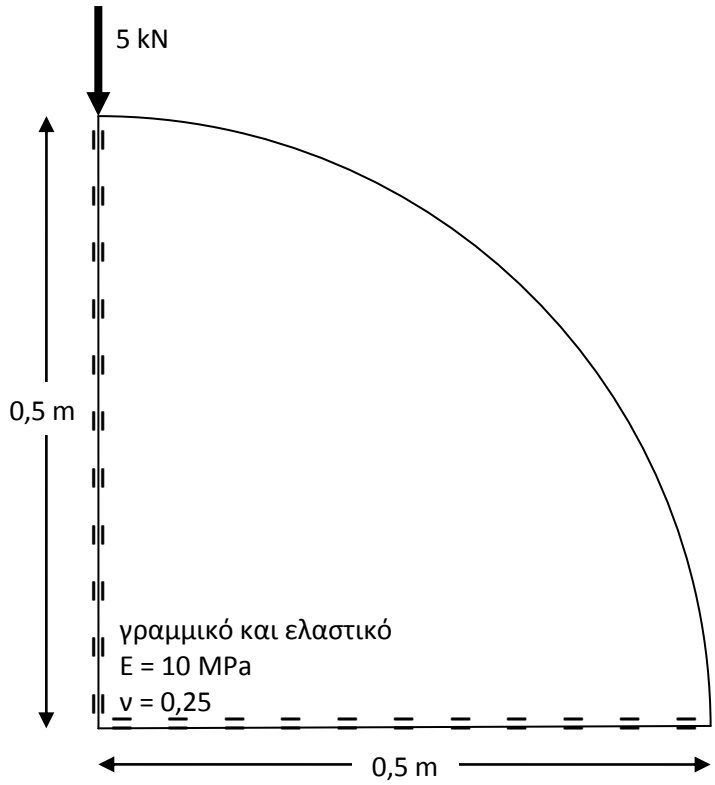
Εικόνα 4.9: Γραφικός υπολογισμός κύριας τάσης σ_3



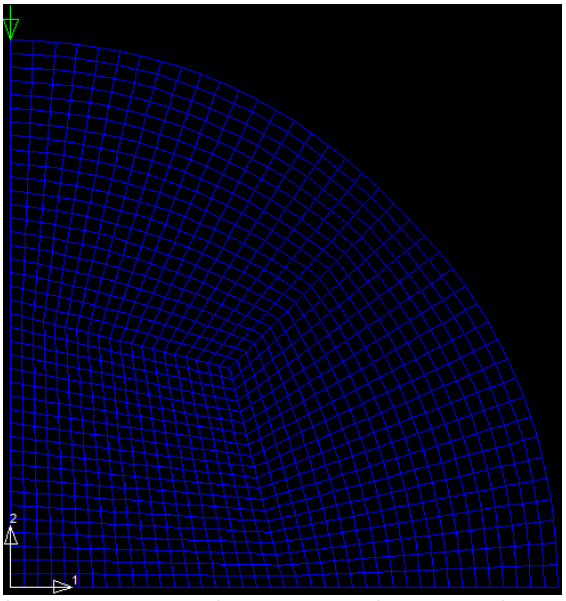
Εικόνα 4.10: Νομογράφημα για υπολογισμό κυρίων τάσεων σ_1 και σ_3

4.2. <u>Πρόβλημα 1:</u> Τεταρτοκύκλιο δίσκου με σημειακό φορτίο σε γραμμική ελαστική ανάλυση

Το πρόβλημα αποτελείται από έναν επίπεδο φορέα σχήματος τεταρτοκυκλίου, ο οποίους φορτίζεται με ένα συγκεντρωμένο φορτίο στην ακραία αριστερή και πάνω πλευρά του. Η αριστερή ακμή του έχει δεσμευτεί έναντι οριζόντιας μετακίνησης, ενώ η κάτω έναντι κατακόρυφης. Στις εικόνες *4.11* και *4.12* παρουσιάζεται ο φορέας και ο κάνναβος στον οποίο έχει διακριτοποιηθεί για να γίνει η ανάλυσή του.

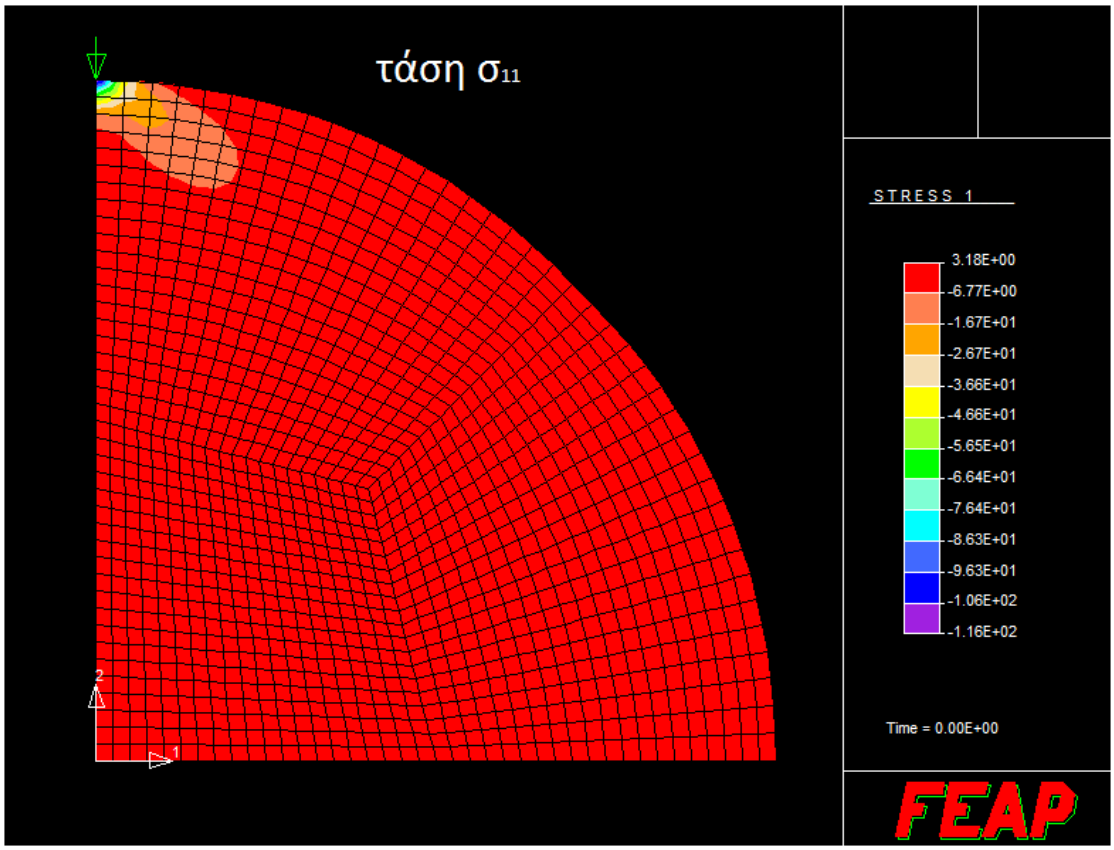


Εικόνα 4.11: Παρουσίαση προβλήματος 1

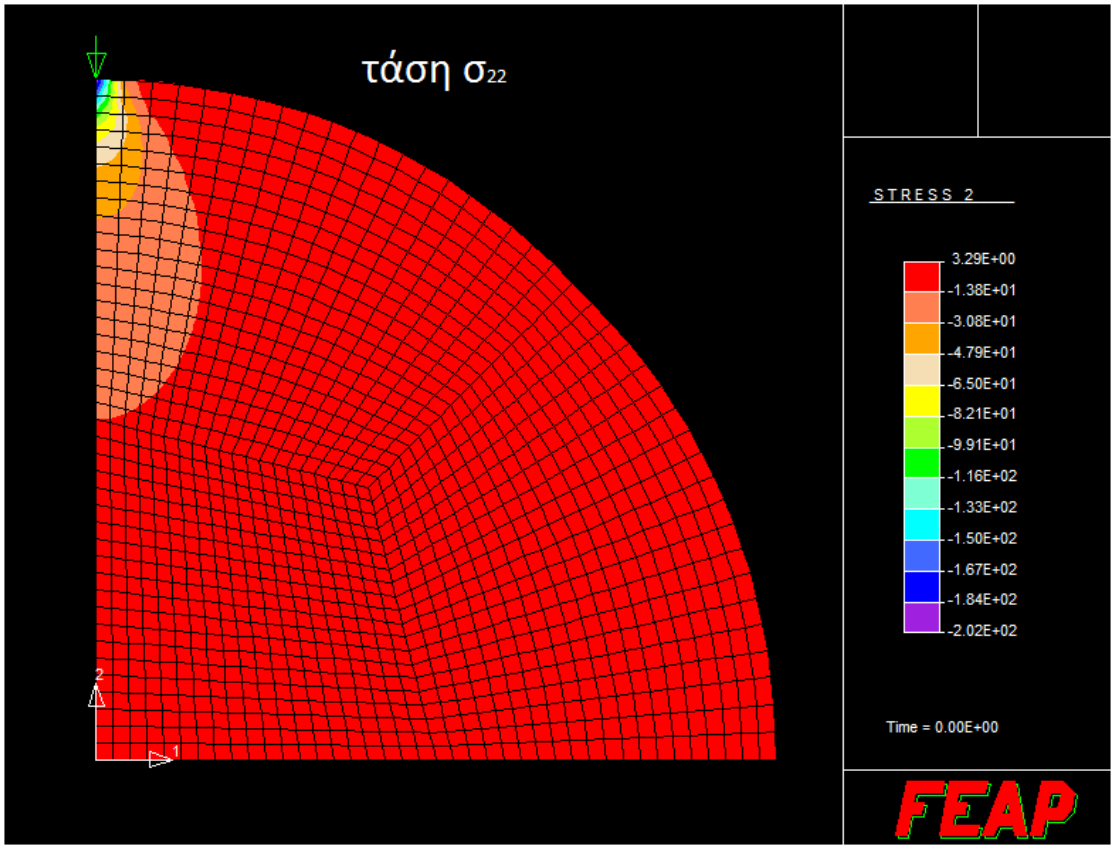


Εικόνα 4.12: Διακριτοποίηση του προβλήματος 1 από το FEAP

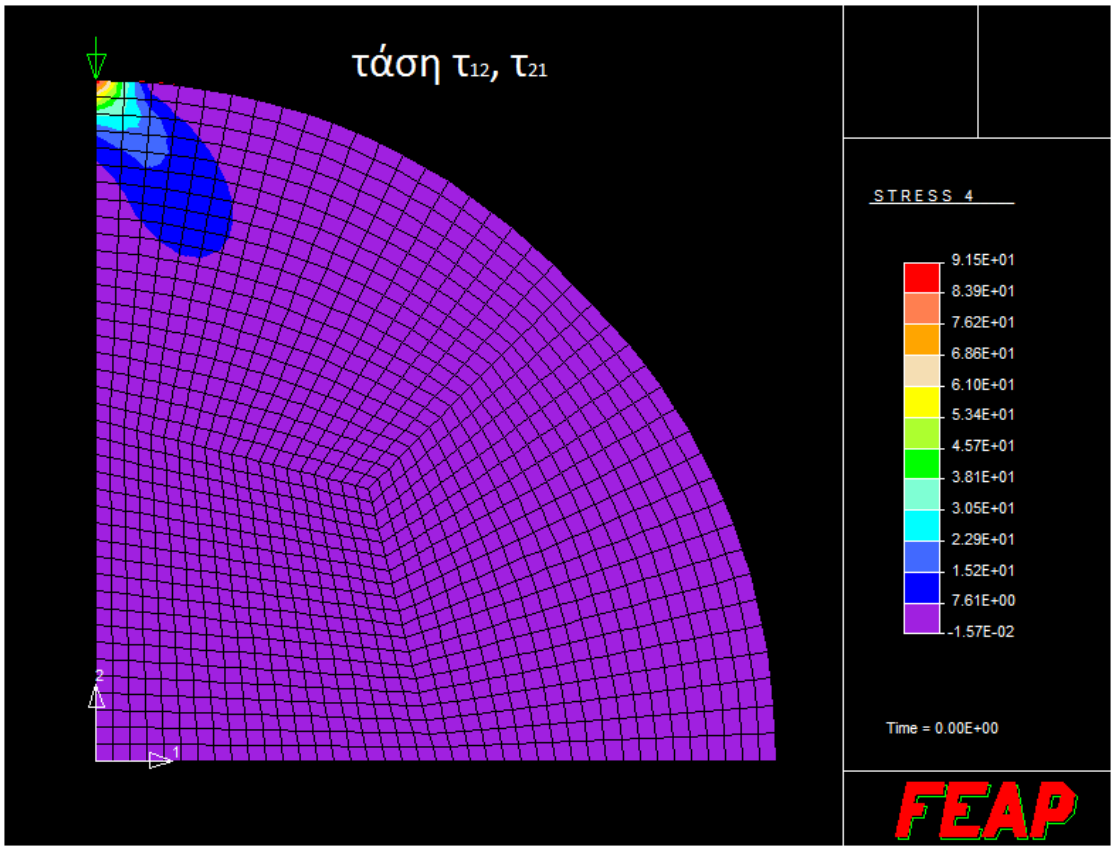
Στη συνέχεια παρουσιάζουμε τα αποτελέσματα που πήραμε για τον συγκεκριμένο φορέα με χρήση του FEAP:



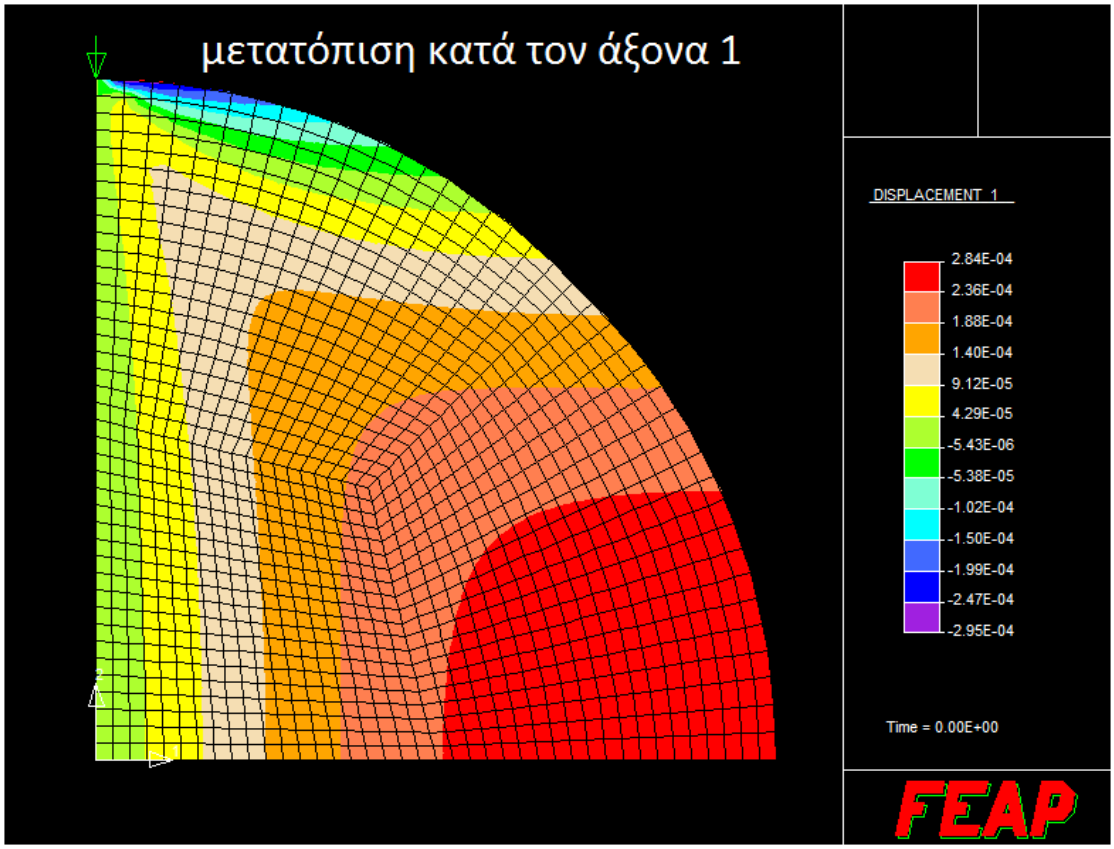
Εικόνα 4.13: Γραφική απεικόνιση ορθής τάσης σ_{11}



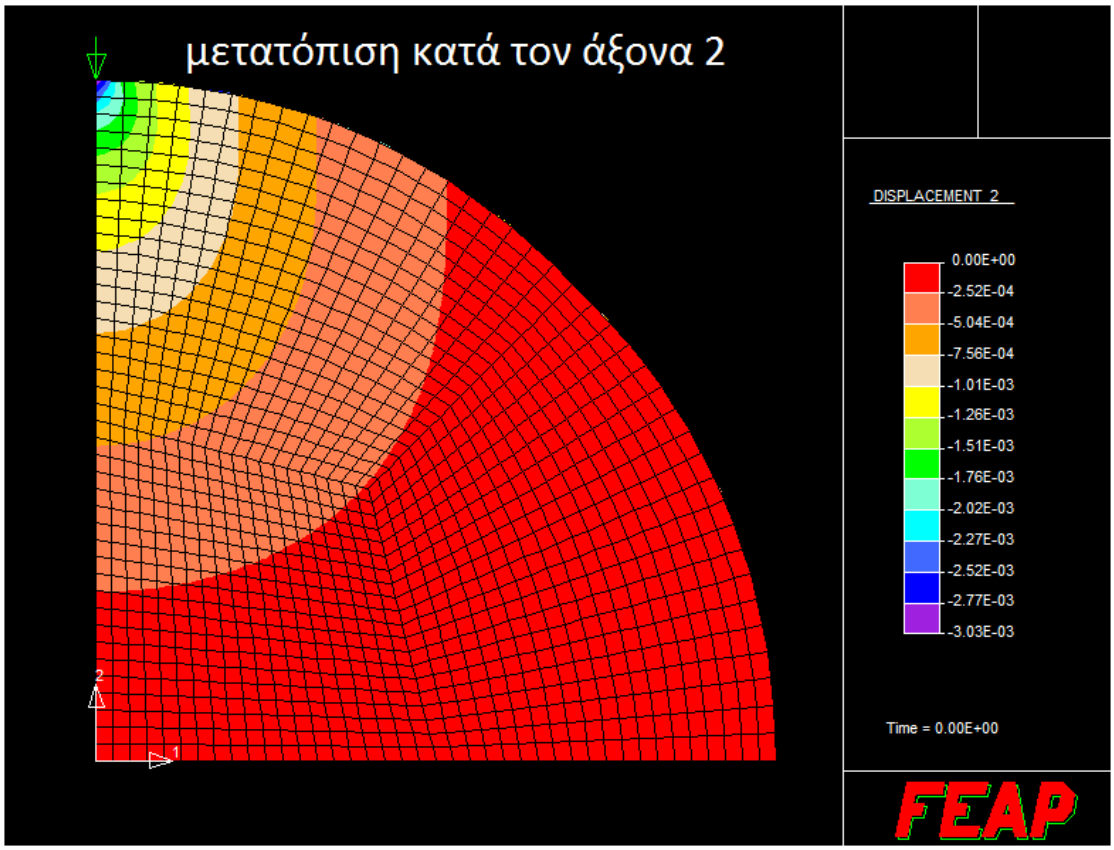
Εικόνα 4.14: Γραφική απεικόνιση ορθής τάσης σ_{22}



Εικόνα 4.15: Γραφική απεικόνιση διατμητικών τάσεων τ_{12} και τ_{21} ($\tau_{12} = \tau_{21}$)



Εικόνα 4.16: Γραφική απεικόνιση των μετατοπίσεων κατά τον οριζόντιο άξονα



Εικόνα 4.17: Γραφική απεικόνιση των μετατοπίσεων κατά τον κατακόρυφο άξονα

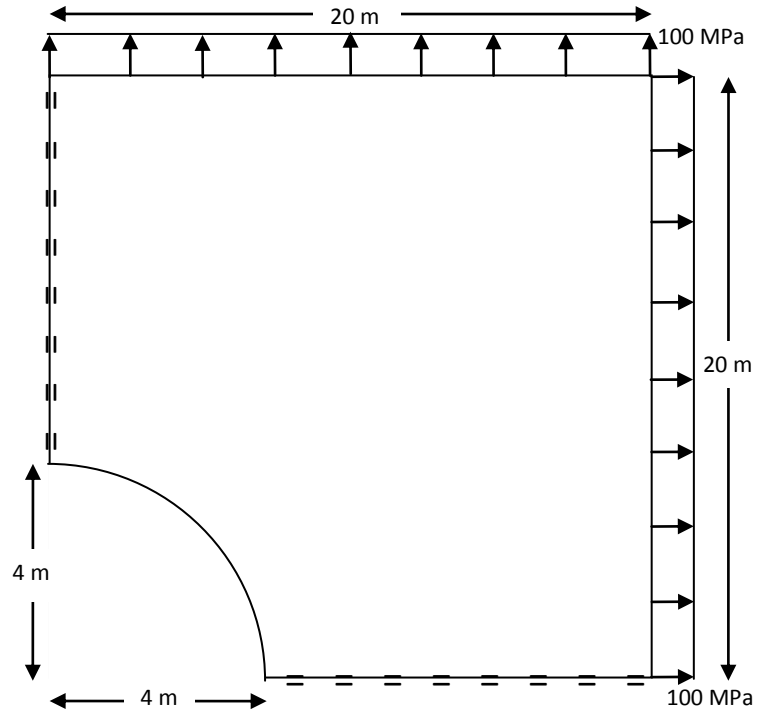
Στις εικόνες *4.13* και *4.14* βλέπουμε τα αποτελέσματα που λαμβάνουμε για τις ορθές τάσεις. Παρατηρούμε και στις δύο περιπτώσεις το «βολβό» τάσεων που σχηματίζεται στο εσωτερικό του φορέα. Στην περίπτωση της κατακόρυφης ορθής τάσης ο βολβός διαφέρει από την περίπτωση της οριζόντιας, κάτι που άλλωστε είναι γνωστό και προβλέπεται από τη θεωρία. Εάν ο φορέας μας ήταν συμμετρικός και προς τα αριστερά με άξονα συμμετρίας το αριστερό όριο που έχει τώρα, θα μπορούσαμε να δούμε ολόκληρο το βολβό στην περίπτωση της σ_{22} και ακριβώς τον συμμετρικό του βολβού της σ_{11}. Επίσης παρατηρούμε και στις δύο περιπτώσεις διαπιστώνουμε ότι τα αποτελέσματα είναι συμβατά με τις συνοριακές συνθήκες.

Ο βολβός τάσεων για τη διάτμηση που παρουσιάζεται στην *Εικόνα 4.15* μοιάζει με αυτόν των οριζόντιων τάσεων ελαφρώς αλλοιωμένο. Η ομοιότητα εξηγείται από το γεγονός ότι η διάτμηση εξαρτάται κυρίως από την οριζόντια ορθή τάση λόγω τριβής, αλλά η τελική αλλοίωση προκύπτει και από τη συμβολή της κατακόρυφης.

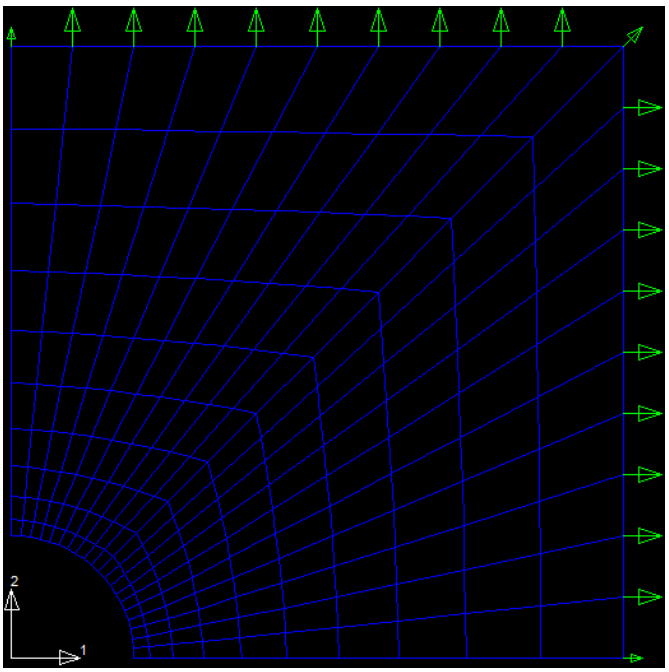
Τα αποτελέσματα των μετατοπίσεων, οριζόντιας και κατακόρυφης, παρουσιάζονται στις εικόνες *4.16* και *4.17* και επαληθεύονται και αυτές από τη θεωρία. Οι οριζόντιες μετατοπίσεις είναι κατά βάση θετικές, δηλαδή ο φορέας «φουσκώνει» προς τα έξω υπό τη φόρτιση, εκτός όμως από κάποιες μικρές περιοχές στην επιφάνειά του και κοντά στη φόρτιση που μετατοπίζονται προς αυτή, διότι οι περιοχές αυτές επηρεάζονται περισσότερο από την κατακόρυφη μετατόπιση που είναι αρκετά σημαντική στις περιοχές αυτές. Στις κατακόρυφες μετατοπίσεις τα πράγματα είναι πιο απλά. Όλος ο φορέας συνθλίβεται και οι κατακόρυφες μετατοπίσεις είναι προς τα κάτω, δηλαδή αρνητικές, που όλο και μειώνονται με το βάθος. Και στις δύο περιπτώσεις παρατηρούμε ότι ικανοποιούνται οι συνοριακές συνθήκες.

4.3. <u>Πρόβλημα 2:</u> Πλάκα με οπή τεταρτοκυκλίου με κατανεμημένα φορτία στις ακραίες ακμές σε γραμμική ελαστική ανάλυση

Το πρόβλημα αποτελείται από έναν επίπεδο φορέα ορθογωνικού σχήματος με μία κυκλική οπή στο κάτω αριστερά μέρος του. Στην αριστερή ακμή είναι δεσμευμένος οριζόντια και στην κάτω δεσμευμένος κατακόρυφα, ενώ στις άλλες δύο φορτίζεται με κατανεμημένο φορτίο κάθετο σε αυτές. Η μοντελοποίηση του προβλήματος, καθώς και ο τρόπος διακριτοποίησής του με πεπερασμένα στοιχεία φαίνεται στις εικόνες *4.18* και *4.19*.

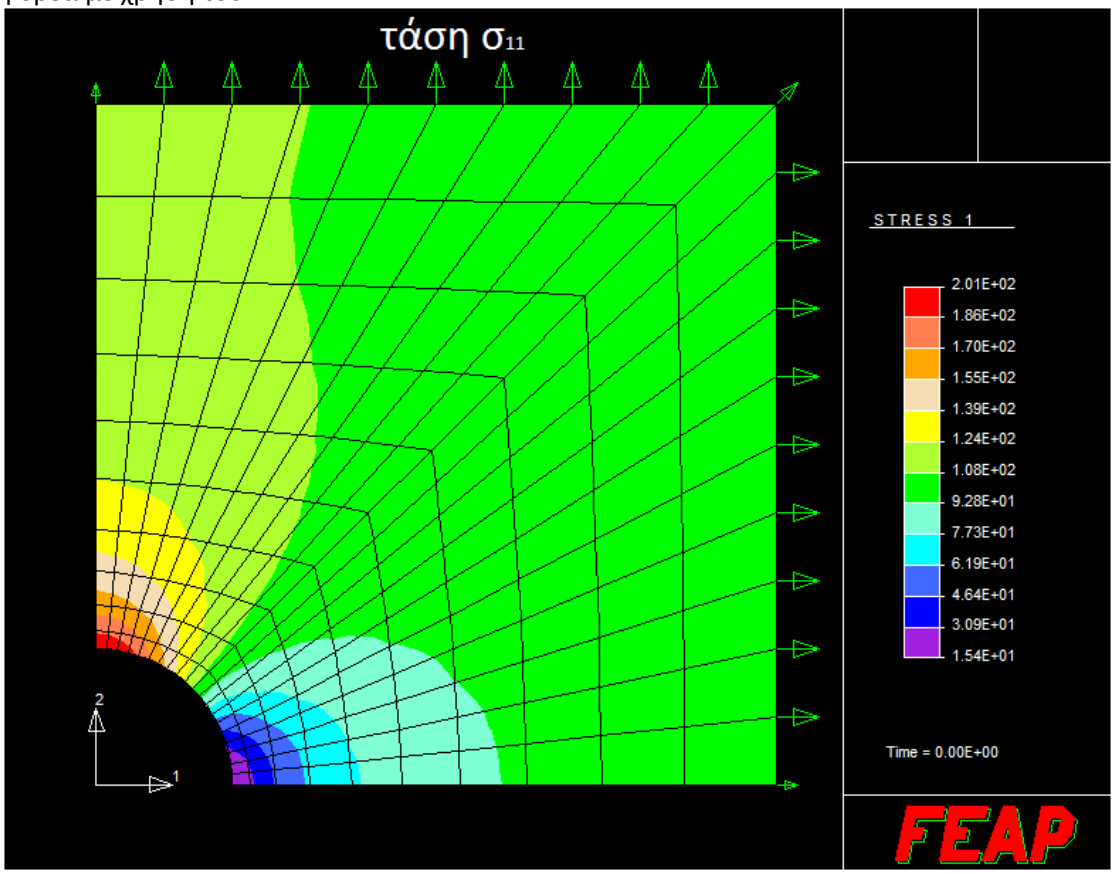


Εικόνα 4.18: Παρουσίαση προβλήματος 2

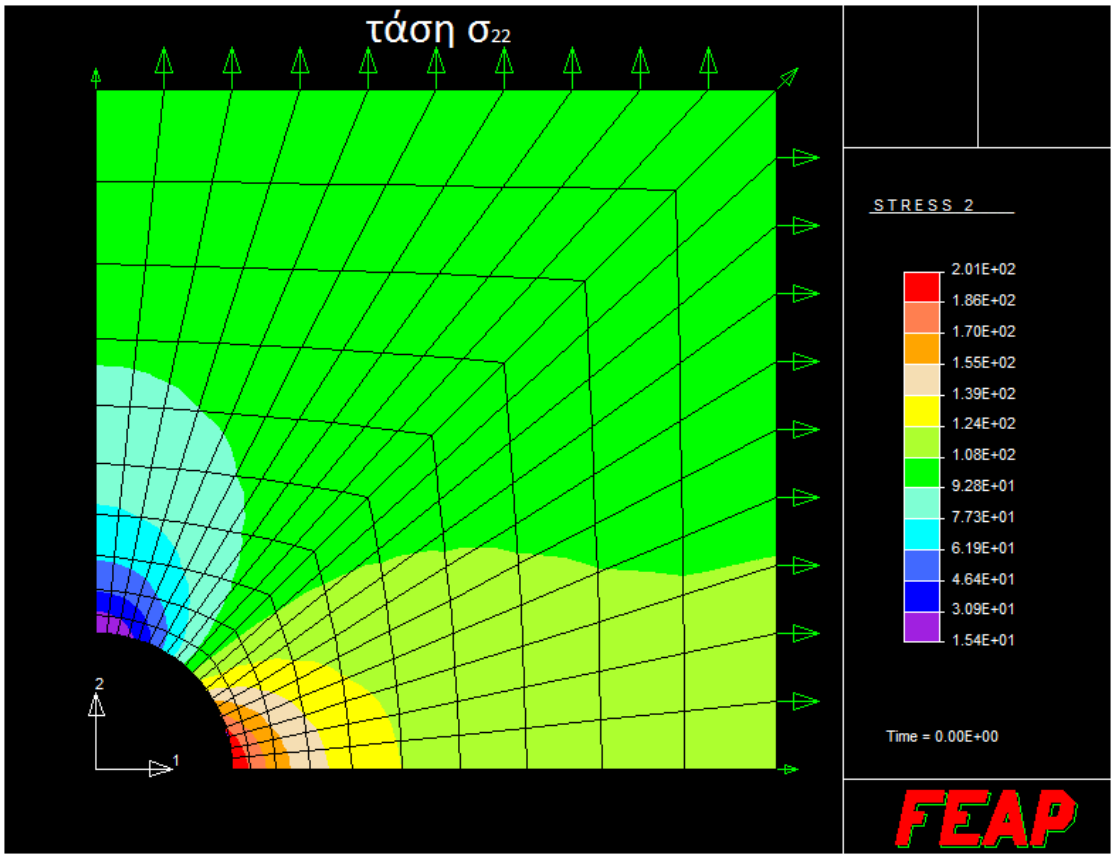


Εικόνα 4.19: Διακριτοποίηση του προβλήματος 2 από το FEAP

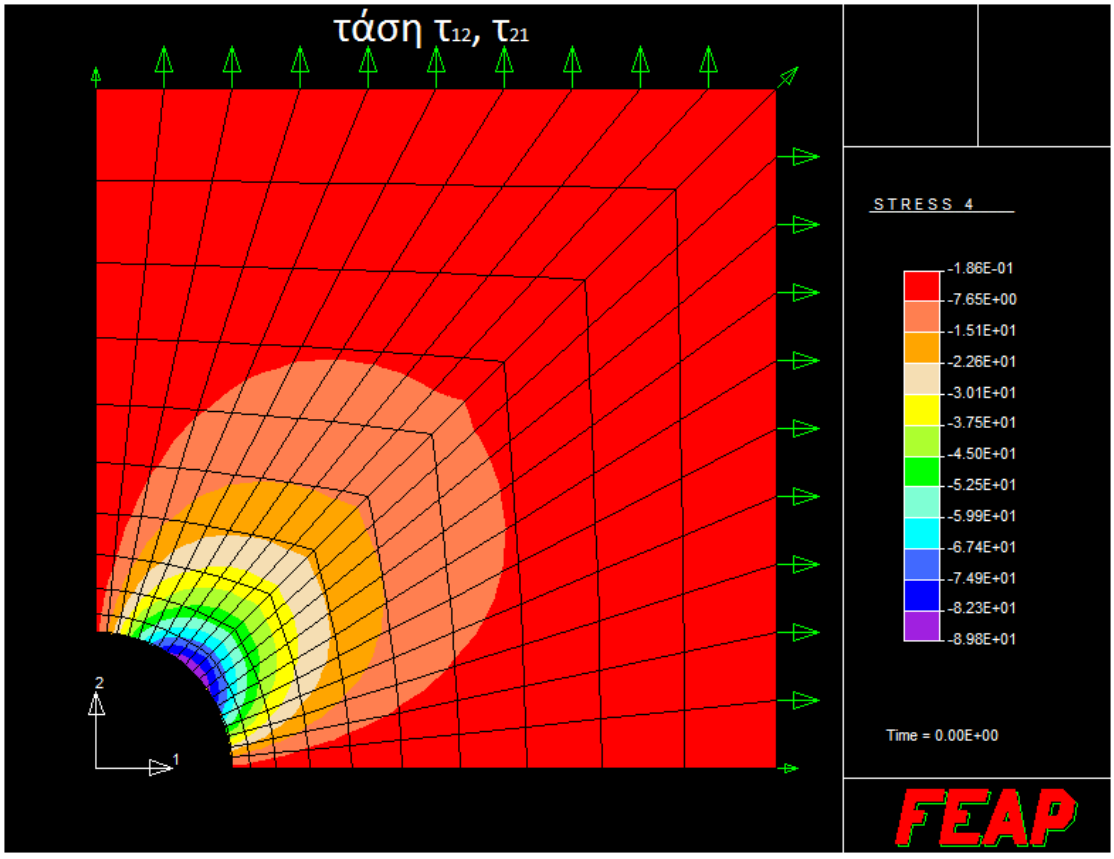
Στη συνέχεια παρουσιάζουμε τα αποτελέσματα που πήραμε για τον συγκεκριμένο φορέα με χρήση του FEAP:



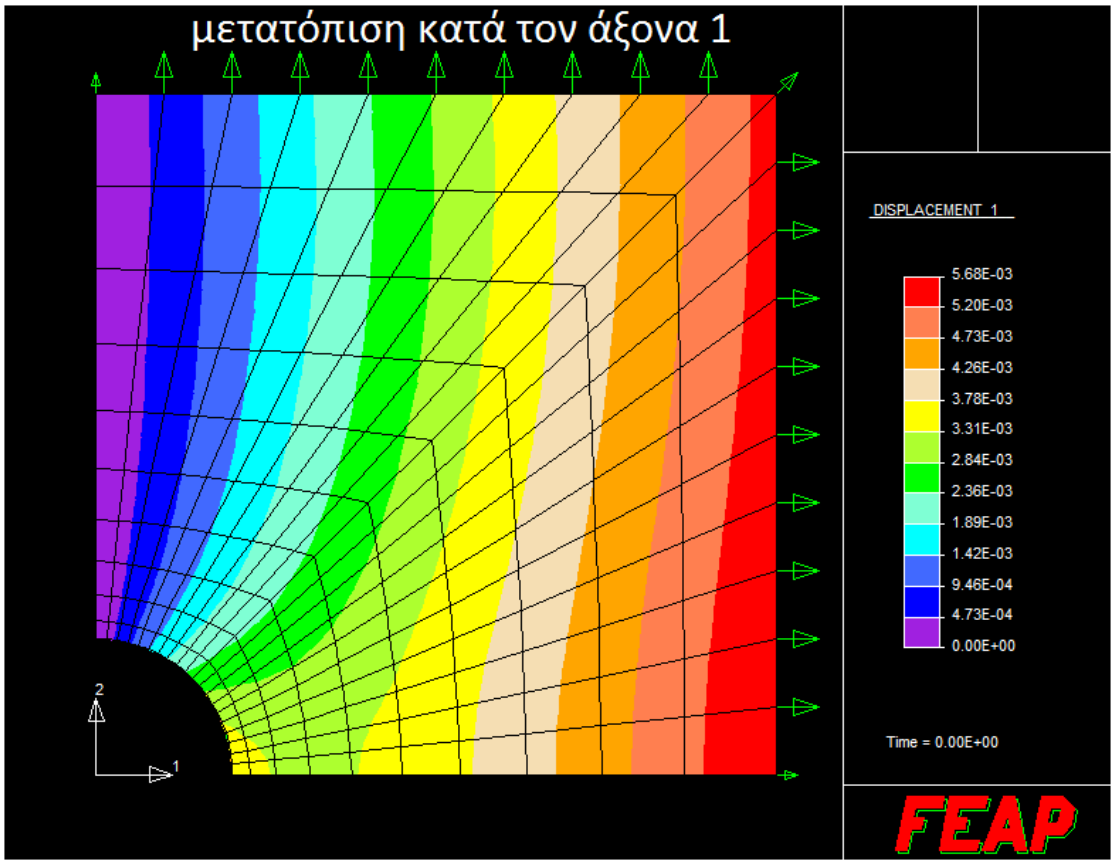
Εικόνα 4.20: Γραφική απεικόνιση ορθής τάσης σ_{11}



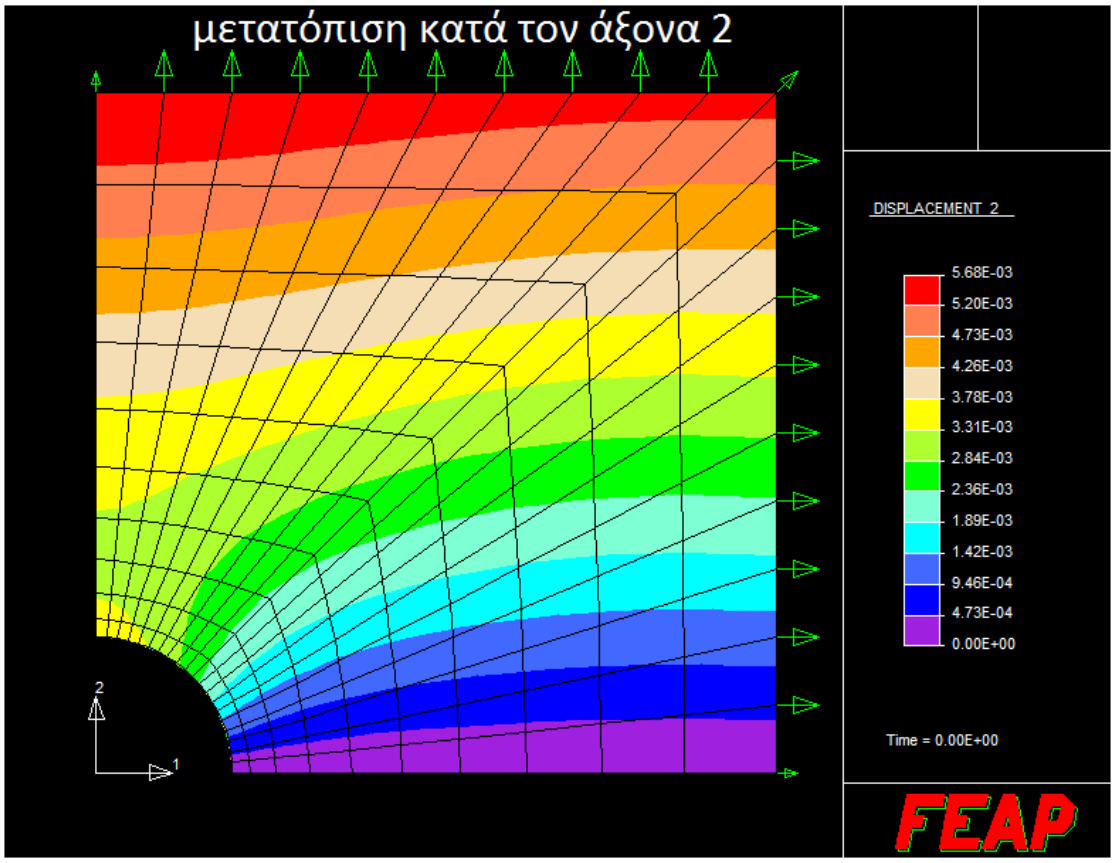
Εικόνα 4.21: Γραφική απεικόνιση ορθής τάσης σ_{22}



Εικόνα 4.22: Γραφική απεικόνιση διατμητικών τάσεων τ_{12} και τ_{21} ($\tau_{21} = \tau_{21}$)



Εικόνα 4.23: Γραφική απεικόνιση των μετατοπίσεων κατά τον οριζόντιο άξονα



Εικόνα 4.24: Γραφική απεικόνιση των μετατοπίσεων κατά τον κατακόρυφο άξονα

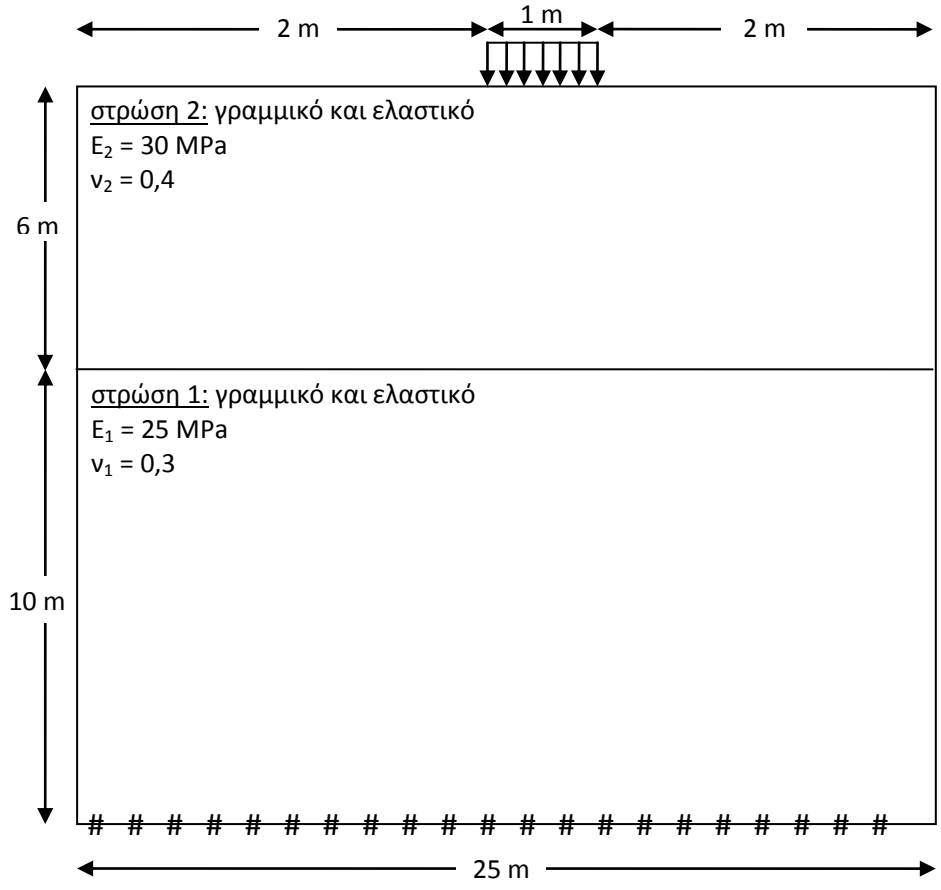
Στις εικόνες *4.20* και *4.21* έχουμε τις ορθές τάσεις σ_{11} και σ_{22} αντίστοιχα που αναπτύσσονται. Το αξιοσημείωτο είναι η συμμετρικότητα που παρουσιάζουν, κάτι λογικό αφού η φόρτιση είναι συμμετρική ως προς τη διαγώνιο της πλάκας και οι ορθές τάσεις ασκούνται σε επίπεδα κάθετα μεταξύ τους. Παρατηρούμε ότι και πάλι αναπτύσσονται βολβοί τάσεων γύρω από την περιοχή της οπής.

Στην *Εικόνα 4.22* έχουμε τις αναπτυσσόμενες διατμητικές τάσεις στο φορέα. Παρατηρούμε και εδώ ξεκάθαρα τον βολβό τάσεων που ξεκινά από την οπή και αναπτύσσεται συμμετρικά ως προς τη διαγώνιο της πλάκας.

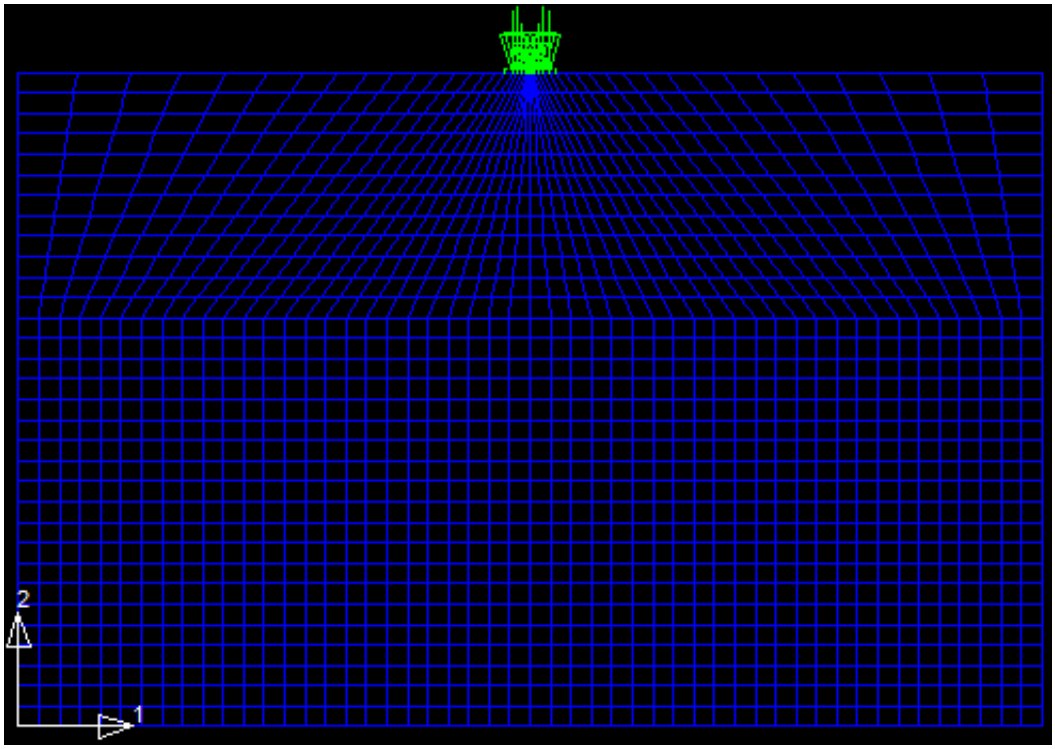
Στις εικόνες *4.23* και *4.24* έχουμε τη γραφική απεικόνιση των οριζόντιων και κατακόρυφων αντίστοιχα μετατοπίσεων. Όπως αναμενόταν και οι δύο είναι συμμετρικές ως προς το γεωμετρικό κέντρο της πλήρους πλάκας και είναι προς την κατεύθυνση των φορτίων παρουσιάζοντας αύξηση όταν βρισκόμαστε πάνω από την οπή.

4.4. <u>Πρόβλημα 3:</u> Προσομοίωση εδάφους και εύκαμπτου πέδιλου εδραζόμενου επ' αυτού σε γραμμική ελαστική ανάλυση

Το πρόβλημα περιλαμβάνει ένα δίστρωτο σχηματισμό εδάφους στον οποίο εφαρμόζουμε ένα κατανεμημένο φορτίο, όπως είναι και η φόρτιση από ένα εύκαμπτο πέδιλο στη επιφάνειά του. Το έδαφος προσομοιώνεται ως ένας φορέας με αρκετά μεγαλύτερες διαστάσεις από αυτές του θεμελίου και το θεμέλιο όπως είπαμε με κατανεμημένο φορτίο. Ο φορέας είναι πακτωμένος μόνο στην κάτω του ακμή. Το μοντέλο του προβλήματος παρουσιάζεται στις εικόνες *4.25* και *4.26*.

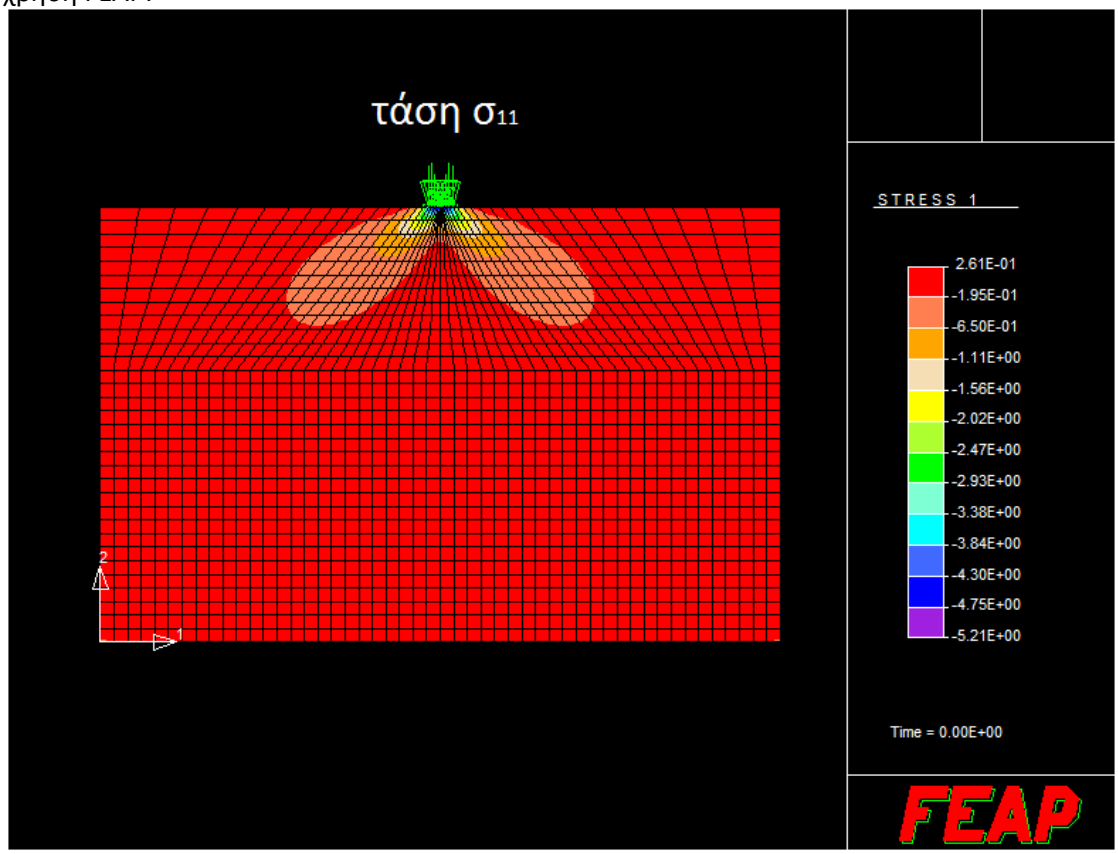


Εικόνα 4.25: Παρουσίαση προβλήματος 3

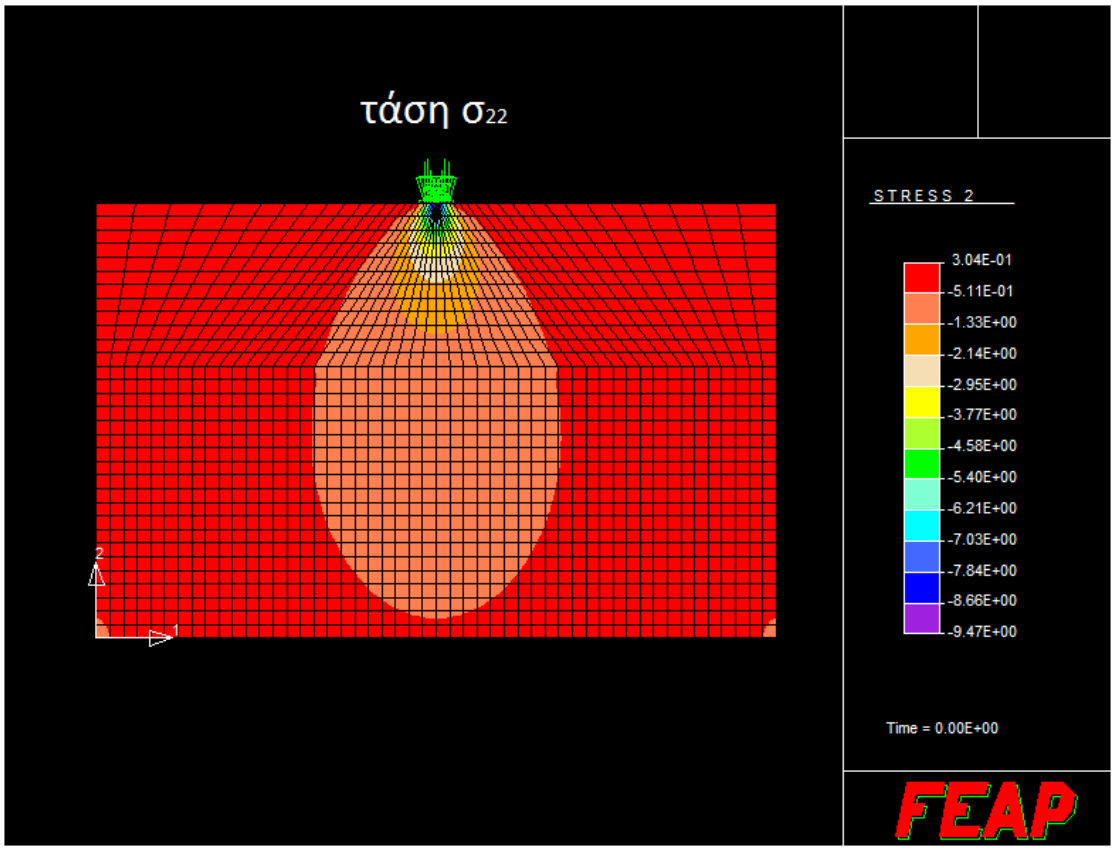


Εικόνα 4.26: Διακριτοποίηση του προβλήματος 3 από το FEAP

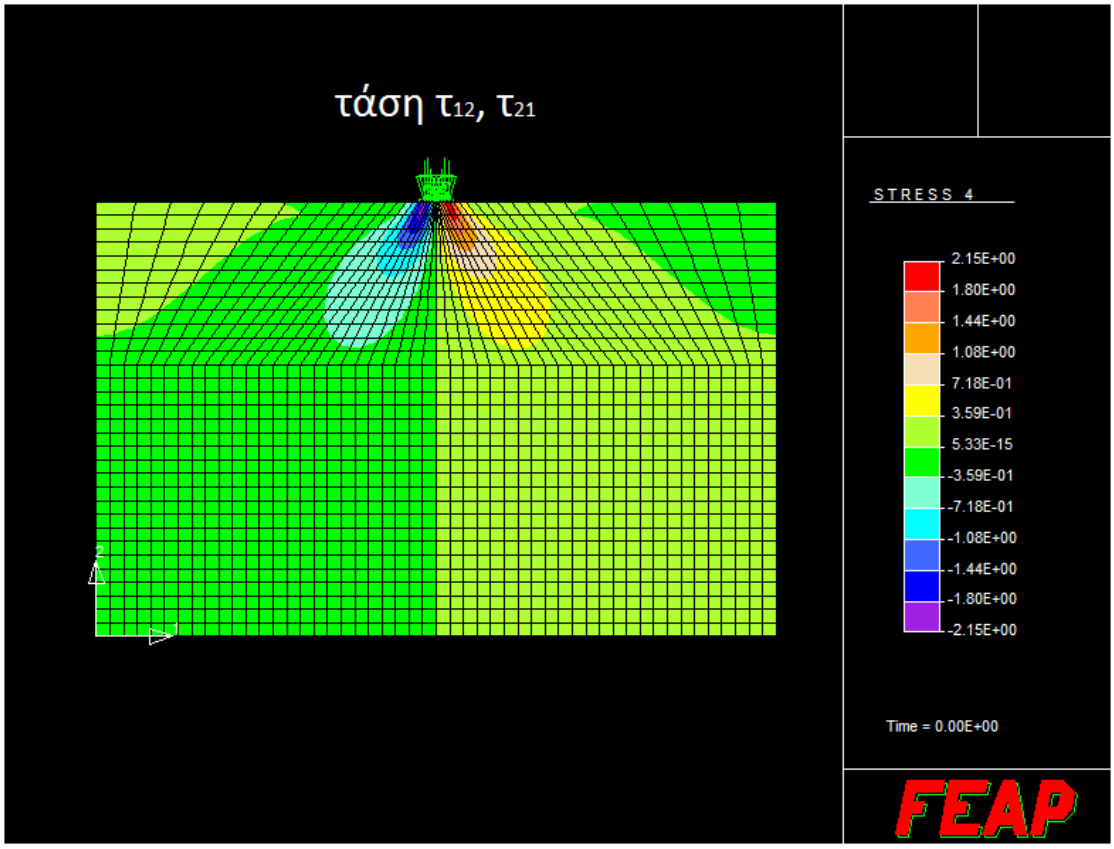
Παρουσιάζουμε τα αποτελέσματα που πήραμε για την επίλυση του φορέα με χρήση FEAP:



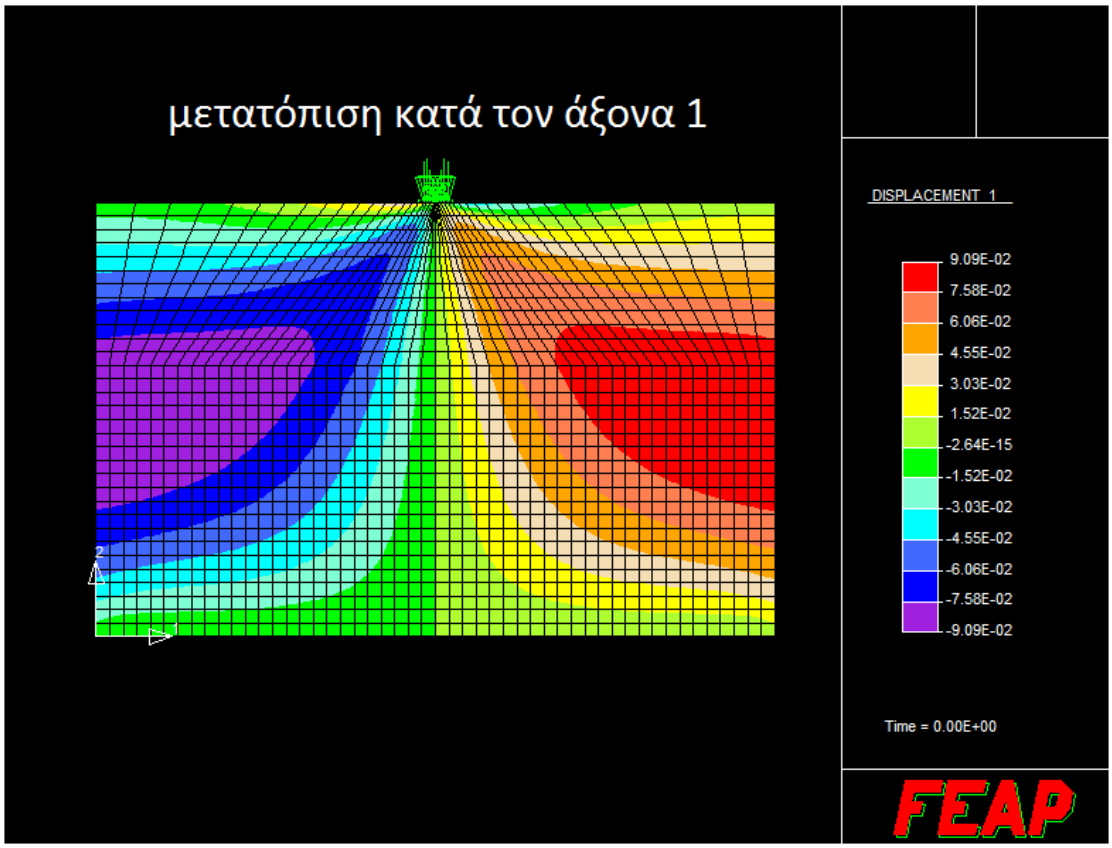
Εικόνα 4.27: Γραφική απεικόνιση ορθής τάσης σ_{11}



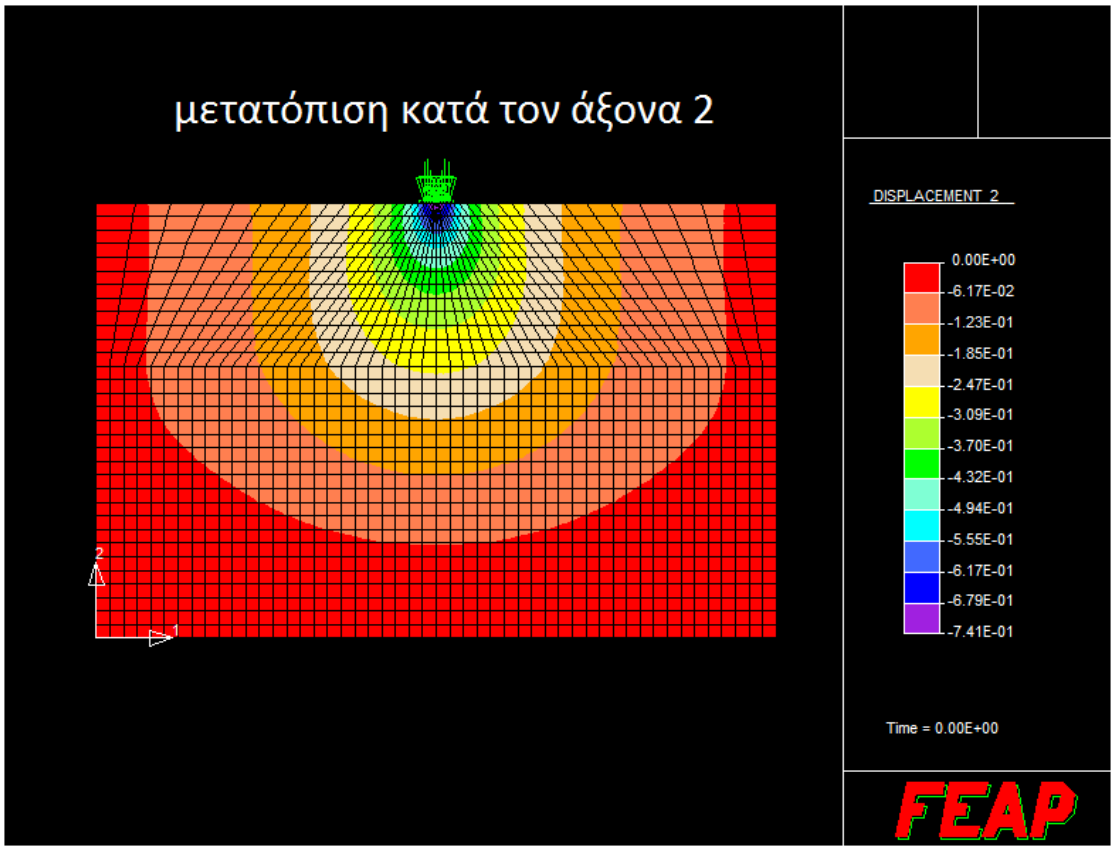
Εικόνα 4.28: Γραφική απεικόνιση ορθής τάσης σ_{22}



Εικόνα 4.29: Γραφική απεικόνιση διατμητικών τάσεων τ_{12} και τ_{21} ($\tau_{12} = \tau_{21}$)



Εικόνα 4.30: Γραφική απεικόνιση των μετατοπίσεων κατά τον οριζόντιο άξονα



Εικόνα 4.31: Γραφική απεικόνιση των μετατοπίσεων κατά τον κατακόρυφο άξονα

Στις εικόνες *4.27* και *4.28* φαίνονται οι ορθές τάσεις που αναπτύσσονται. Για την σ_{11} βλέπουμε το σχηματισμό δύο βολβών τάσεων συμμετρικών ως προς τον άξονα συμμετρίας του φορέα, ενώ για την σ_{22} σχηματίζεται ένας βολβός τάσεων που είναι και πάλι συμμετρικός ως προς τον άξονα συμμετρίας του φορέα. Όλα τα αποτελέσματα είναι αναμενόμενα, με το δίστρωτο έδαφος να μην έχει επηρεάσει κατά πολύ την ποιοτική κατανομή των τάσεων σε σχέση με αυτή που θα παίρναμε αν είχαμε μία μόνο στρώση. Στην *Εικόνα 4.29* παρατηρούμε την ανάπτυξη των διατμητικών τάσεων. Και εδώ έχουμε δύο βολβούς συμμετρικούς ως προς τον άξονα συμμετρίας του φορέα, αλλά με αντίθετα πρόσημα, όπως αναμενόταν. Οι συνοριακές συνθήκες ικανοποιούνται σε κάθε περίπτωση.

Στις εικόνες *4.30* και *4.31* έχουμε τις οριζόντιες και κατακόρυφες μετατοπίσεις αντίστοιχα. Παρατηρούμε ότι κοντά στην επιφάνεια του εδάφους οι οριζόντιες μετατοπίσεις συμβαίνουν προς την πλευρά που είναι η φόρτιση, ενώ όσο πηγαίνουμε πιο βαθιά κάποια στιγμή μηδενίζονται και μετά αρχίζουν να εκδηλώνονται προς την αντίθετη από πριν κατεύθυνση. Με τις κατακόρυφες μετατοπίσεις τα πράγματα είναι πιο απλά. Το έδαφος συνθλίβεται και δεν παρουσιάζει φαινόμενα διόγκωσης και έτσι όλες οι μετατοπίσεις είναι αρνητικές. Σε όλες τις περιπτώσεις τα αποτελέσματα είναι αναμενόμενα και οι συνοριακές συνθήκες του προβλήματος ικανοποιούνται χωρίς αντιφάσεις.

<u>Παράρτημα Ι:</u>
Κώδικες σε fortran που κατασκευάστηκαν.

I.1. Μετασχηματισμοί πινάκων πολυδιάστατων σε μονοδιάστατους και το αντίστροφο

I.1.1. Μονοδιάστατα μητρώα

Η μετατροπή αυτή στην περίπτωση του μονοδιάστατου διανύσματος είναι πολύ απλή και δεν έχουμε παρά να τοποθετήσουμε το μονοδιάστατο διάνυσμα στο διάνυσμα της κύριας μνήμης με πρώτο στοιχείο όχι πλέον το δείκτη 1, αλλά το δείκτη της θέσης στο διάνυσμα της κύριας μνήμης όπου έχει αποθηκευτεί το πρώτο στοιχείο του διανύσματος. Έτσι, έχουμε τον εξής μετασχηματισμό του δείκτη ενός τυχόντος στοιχείου του διανύσματος στον δείκτη του ίδιου στοιχείου στο διάνυσμα της κύριας μνήμης:

$$i \to i + a - 1$$

Όπου:

i: Ο δείκτης του αρχικού διανύσματος.
a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

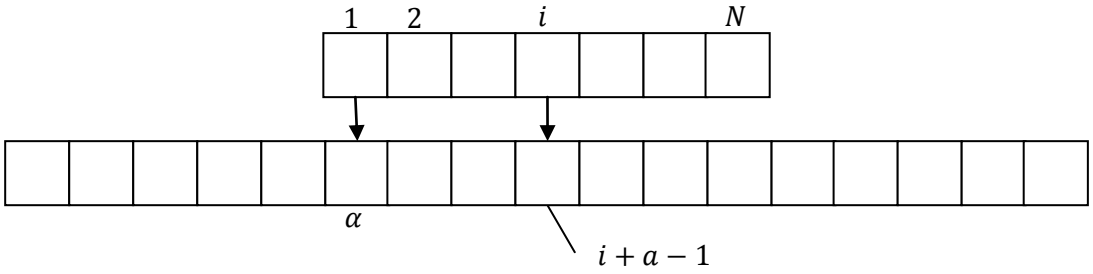
Και αντίστροφα από το δείκτη του διανύσματος της κύριας μνήμης στο δείκτη του αρχικού διανύσματος:

$$x \to x - a + 1$$

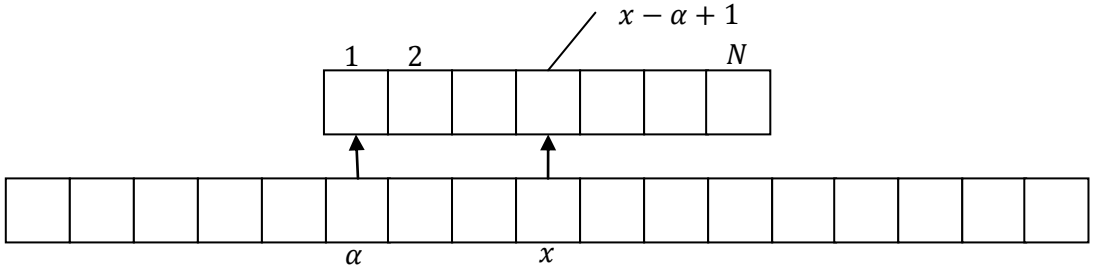
Όπου:

x: Ο δείκτης στο διάνυσμα της κύριας μνήμης.
a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

Μια σχηματική απεικόνιση της μετατροπής αυτής μπορεί κανείς να δει στις *εικόνες I.38* και *I.39*.



Εικόνα I.32: Από τοπικό μονοδιάστατο πίνακα σε μονοδιάστατο πίνακα κύριας μνήμης.



Εικόνα I.33: Από μονοδιάστατο πίνακα κύριας μνήμης σε τοπικό μονοδιάστατο πίνακα.

I.1.2. Δυσδιάστατα μητρώα

Στην περίπτωση του δυσδιάστατου μητρώου τα πράγματα δεν είναι τόσο απλά. Αρχικά να παρατηρήσουμε ότι υπάρχουν δύο τρόποι εντάξεως του αρχικού δυσδιάστατου μητρώου. Ο ένας είναι ανά γραμμή και ο άλλος ανά στήλη. Στον πρώτο τρόπο οι γραμμές του μητρώου αποθηκεύονται στο διάνυσμα της κύριας μνήμης η μία μετά την άλλη, ενώ στον δεύτερο γίνεται το ίδιο με τις στήλες. Έτσι έχουμε τον εξής μετασχηματισμό του δείκτη ενός τυχόντος στοιχείου του μητρώου στον δείκτη του ίδιου στοιχείου στο διάνυσμα της κύριας μνήμης:

Κατά γραμμή:

$$(i,j) \to (i-1) * N + j + a - 1$$

Κατά στήλη:

$$(i,j) \to (j-1) * M + i + a - 1$$

Όπου:

- i: Ο δείκτης που εκφράζει τις γραμμές στο αρχικό μητρώο.
- j: Ο δείκτης που εκφράζει τις στήλες στο αρχικό μητρώο.
- M: Το συνολικό πλήθος των γραμμών του αρχικού πίνακα.
- N: Το συνολικό πλήθος των στηλών του αρχικού πίνακα.
- a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

Και αντίστροφα από το δείκτη του διανύσματος της κύριας μνήμης στο δείκτη του αρχικού μητρώου:

Κατά γραμμή:

$$x \to \left(A.M.\left(\frac{x+1-a}{N}\right) + 1, x - N * A.M.\left(\frac{x+1-a}{N}\right) \right)$$

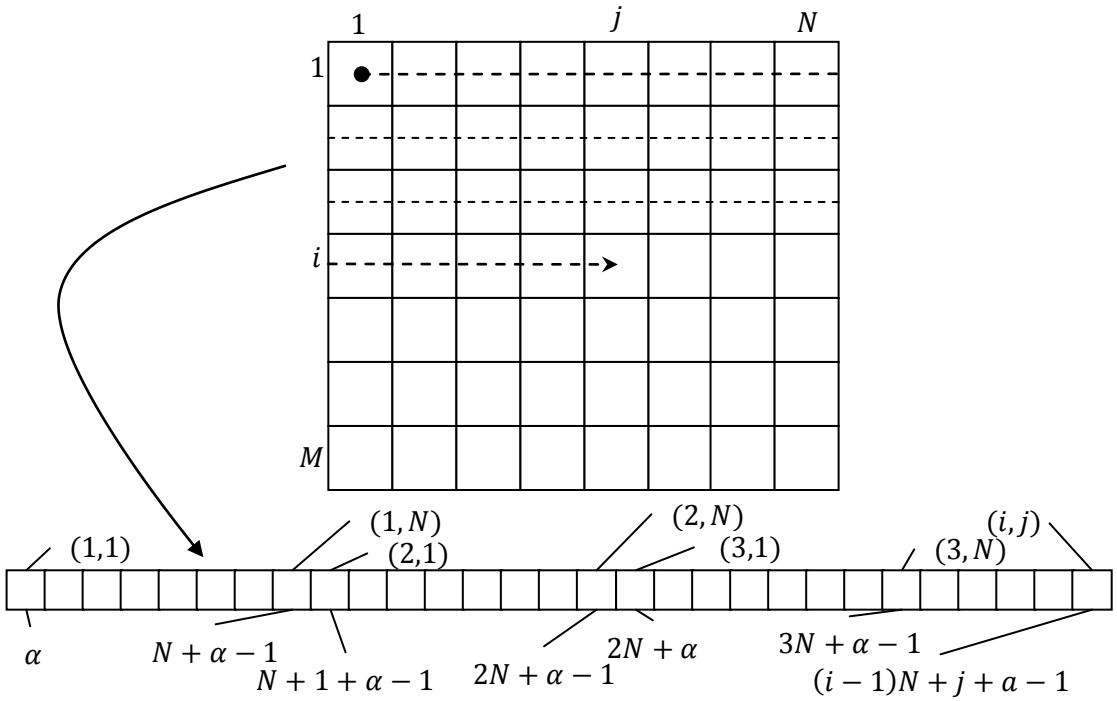
Κατά στήλη:

$$x \to \left(x + 1 - a - M * A.M.\left(\frac{x+1-a}{M}\right), A.M.\left(\frac{x+1-a}{M}\right) + 1 \right)$$

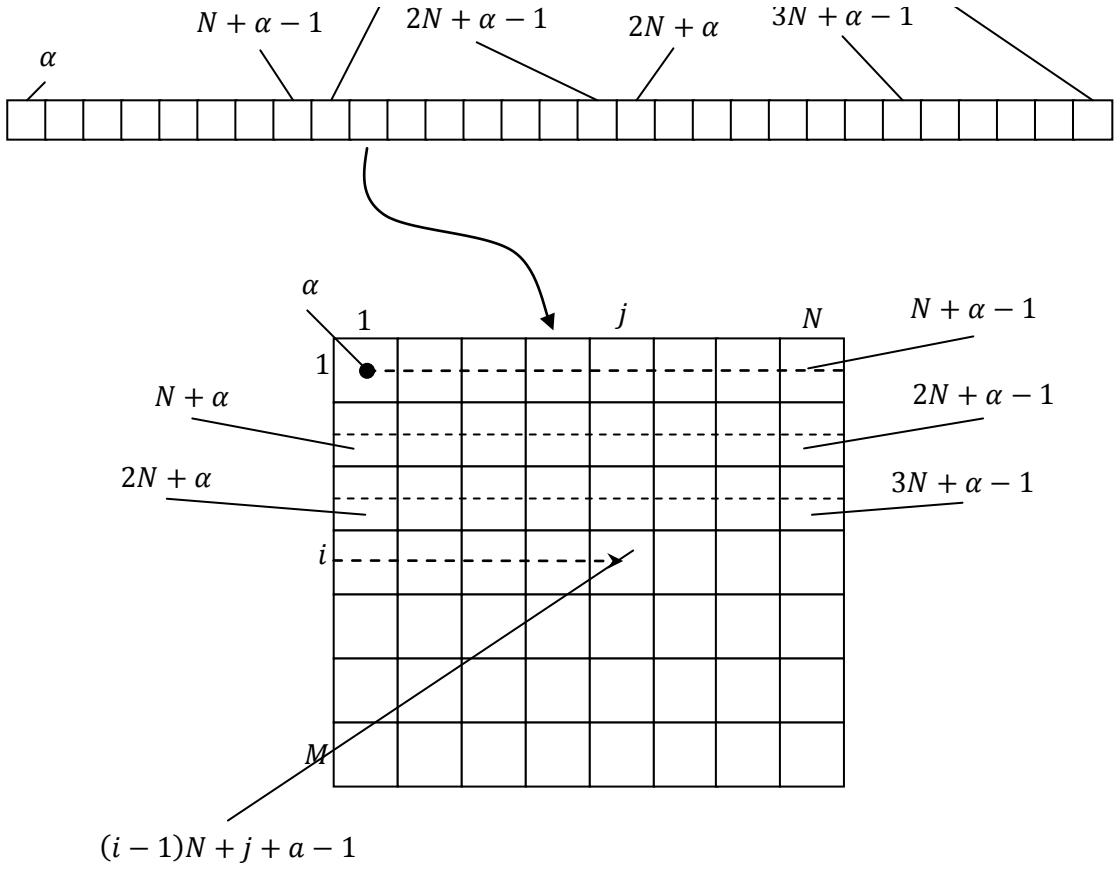
Όπου:

- x: Ο δείκτης που εκφράζει τη θέση στο διάνυσμα της κύριας μνήμης.
- M: Το συνολικό πλήθος των γραμμών του αρχικού πίνακα.
- N: Το συνολικό πλήθος των στηλών του αρχικού πίνακα.
- a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

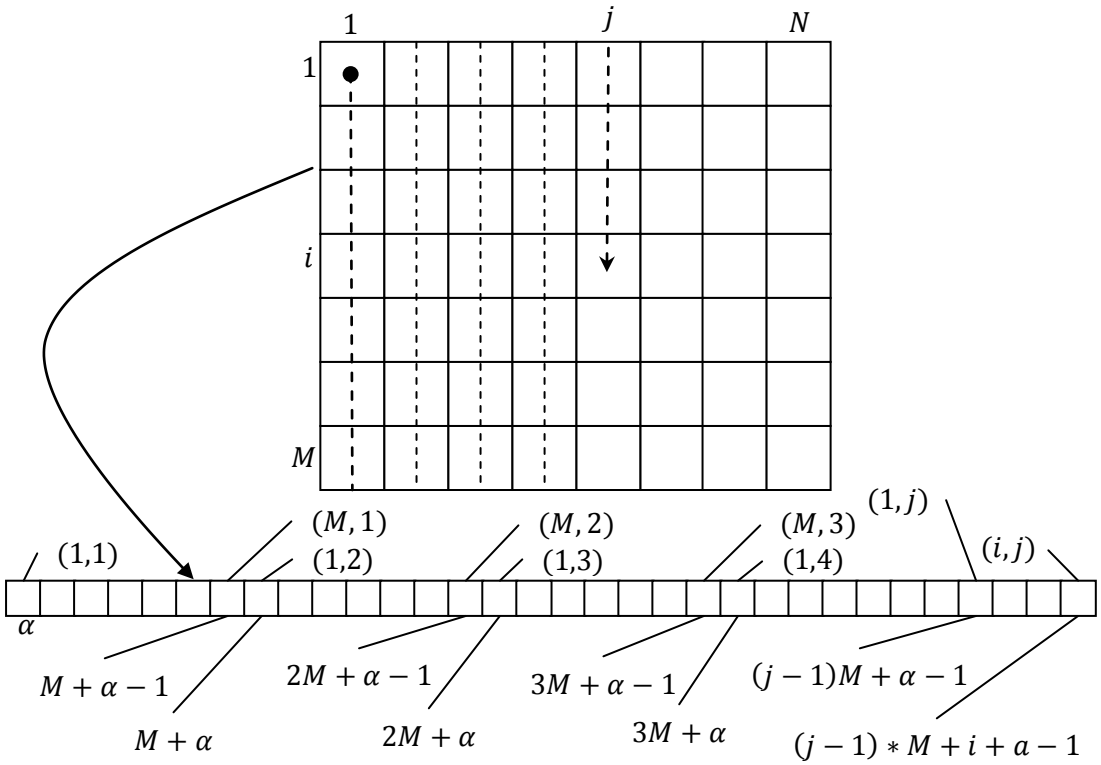
Σχηματικά οι κατά γραμμή απεικονίσεις για δισδιάστατους πίνακες παριστάνονται στις *εικόνες Ι.40* και *Ι.41*. Και οι αντίστοιχες απεικονίσεις ανά στήλη στις *εικόνες Ι.42* και *Ι.43*.



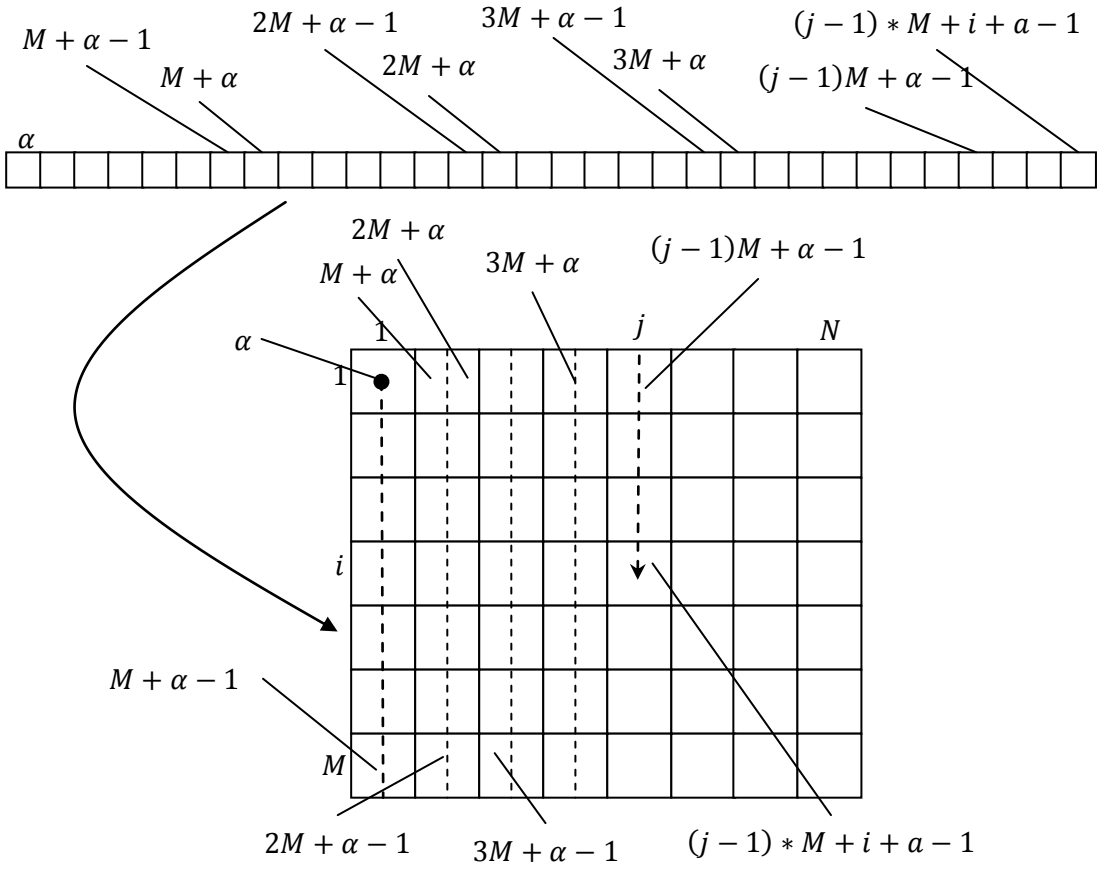
Εικόνα I.34: Μετασχηματισμός δεικτών κατά γραμμή από τοπικό δυσδιάστατο πίνακα σε μονοδιάστατο κύριας μνήμης.



Εικόνα I.35: Μετασχηματισμός δεικτών κατά γραμμή από μονοδιάστατο πίνακα κύριας μνήμης σε τοπικό δυσδιάστατο.



Εικόνα Ι.36: Μετασχηματισμός δεικτών κατά στήλη από τοπικό δυσδιάστατο πίνακα σε μονοδιάστατο κύριας μνήμης.



Εικόνα Ι.37: Μετασχηματισμός δεικτών κατά στήλη από μονοδιάστατο πίνακα κύριας μνήμης σε τοπικό δυσδιάστατο.

Ι.1.3. Τρισδιάστατα μητρώα

Τέλος θα αναφερθούμε στην περίπτωση του τρισδιάστατου μητρώου. Στην περίπτωση αυτή υπάρχουν τέσσερις τρόποι αποθήκευσής του σε ένα διάνυσμα. Οι πρώτοι είναι ανά γραμμή και ανά στήλη με κατεύθυνση κάθετη στην τρίτη διάσταση, για κάθε επίπεδο (τρίτη διάσταση) και οι άλλοι είναι ανά γραμμή με κατεύθυνση κάθετη στις στήλες, για κάθε στήλη και ανά στήλη με κατεύθυνση κάθετη στις γραμμές, για κάθε γραμμή. Εμείς θα αναφερθούμε μόνο στους δύο πρώτους τρόπους που είναι και οι πιο συνηθισμένοι και που του χρειαζόμαστε και στην περίπτωσή μας. Θα αναφερόμαστε απλά σε αυτούς ονομάζοντάς τους «κατά γραμμή» και «κατά στήλη». Έτσι έχουμε τον εξής μετασχηματισμό του δείκτη ενός τυχόντος στοιχείου του μητρώου στον δείκτη του ίδιου στοιχείου στο διάνυσμα της κύριας μνήμης:

Κατά γραμμή:

$$(i,j,k) \to (k-1)*M*N + (i-1)*N + j + a - 1$$

Κατά στήλη:

$$(i,j,k) \to (k-1)*M*N + (j-1)*M + i + a - 1$$

Όπου:

- i: Ο δείκτης που εκφράζει τις γραμμές στο αρχικό μητρώο.
- j: Ο δείκτης που εκφράζει τις στήλες στο αρχικό μητρώο.
- k: Ο δείκτης που εκφράζει τα επίπεδα στο αρχικό μητρώο.
- M: Το συνολικό πλήθος των γραμμών του αρχικού πίνακα.
- N: Το συνολικό πλήθος των στηλών του αρχικού πίνακα.
- K: Το συνολικό πλήθος των επιπέδων του αρχικού πίνακα.
- a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

Και αντίστροφα από το δείκτη του διανύσματος της κύριας μνήμης στο δείκτη του αρχικού μητρώου:

Κατά γραμμή:

$$x \to \left(A.M.\left(\frac{x+1-a-M*N*A.M.\left(\frac{x+1-a}{M*N}\right)}{N} \right) + 1, x+1-a-M*N \right.$$
$$* A.M.\left(\frac{x+1-a}{M*N}\right) - N$$
$$\left. * A.M.\left(\frac{x+1-a-M*N*A.M.\left(\frac{x+1-a}{M*N}\right)}{N} \right), A.M.\left(\frac{x+1-a}{M*N}\right) + 1 \right)$$

Κατά στήλη:

$$x$$
$$\to \left(x+1-a-M*N*A.M.\left(\frac{x+1-a}{M*N}\right) - M \right.$$
$$* A.M.\left(\frac{x+1-a-M*N*A.M.\left(\frac{x+1-a}{M*N}\right)}{M} \right), A.M.\left(\frac{x+1-a-M*N*A.M.\left(\frac{x+1-a}{M*N}\right)}{M} \right)$$
$$\left. + 1, A.M.\left(\frac{x+1-a}{M*N}\right) + 1 \right)$$

Όπου:

- x: Ο δείκτης που εκφράζει τη θέση στο διάνυσμα της κύριας μνήμης.
- M: Το συνολικό πλήθος των γραμμών του αρχικού πίνακα.

N: Το συνολικό πλήθος των στηλών του αρχικού πίνακα.
K: Το συνολικό πλήθος των επιπέδων του αρχικού πίνακα.
a: Η θέση (δείκτης) που έχει στην κύρια μνήμη το πρώτο στοιχείο του αρχικού διανύσματος.

Τώρα, έχοντας πλέον καλύψει το θεωρητικό υπόβαθρο μπορούμε να φτιάξουμε μια υπορουτίνα που να της δίνουμε τη διάσταση του πίνακα που έχουμε, το στοιχείο στο οποίο θέλουμε να αναφερθούμε και το στοιχείο a και αυτή να μας επιστρέφει τον κατάλληλο δείκτη που πρέπει να χρησιμοποιήσουμε για το διάνυσμα της κύριας μνήμης. Η ρουτίνα αυτή είναι η ακόλουθη:

```
c-----[--.----+----.----+----.---------------------------------------]

      subroutine MatrixesAndVectors(what2what, way, i, j, k, M, N, L,
     2                              a, x, flag)
c     ****************************************************************
c     Είναι μια υπορουτίνα που μετατρέπει τους δείκτες από μητρώο σε
c     διάνυσμα και αντίστροφα αναγνωρίζοντας τι θέλουμε να κάνουμε από
c     τα ορίσματα, τα οποία σε κάποιες περιπτώσεις μπορεί να είναι εισό-
c     δου, αλλά σε άλλες εξόδου και το αντίστροφο. Μεταβλητές που χρησι-
c     μοποιούνται:
c        Εισόδου:
c           what2what: Λαμβάνει τιμή 'm2v' αν θέλουμε να μετατρέψουμε
c                      δείκτες από μητρώο σε διάνυσμα και 'v2m' από διά-
c                      νυσμα σε μητρώο.
c           way:       Λαμβάνει τιμή 'row' αν θέλουμε να γίνει η μετα-
c                      τροπή ανά γραμμή και 'col' ανά στήλη.
c           M:         Το πλήθος των γραμμών.
c           N:         Το πλήθος των στηλών.
c           L:         Το πλήθος των επιπέδων.
c           a:         Η θέση του πρώτου στοιχείου του μητρώου στο διά-
c                      νυσμα.
c        Εισόδου- εξόδου:
c           i:         Η αντίστοιχη γραμμή στο μητρώο.
c           j:         Η αντίστοιχη στήλη στο μητρώο.
c           k:         Το αντίστοιχο επίπεδο στο μητρώο.
c           x:         Ο αντίστοιχος δείκτης στο διάνυσμα.
c        Εξόδου:
c           flag:      Λογική μεταβλητή. Αν η αντιστοίχιση είναι πετυχη-
c                      μένη είναι .true., αλλιώς είναι .false..
c     ****************************************************************

      implicit none

      integer      :: i, j, k, M, N, L, a, x
      logical      :: flag
      character(*) :: way, what2what

      if (what2what.eq.'m2v') then
        call Matrix2Vector(way, i, j, k, M, N, L, a, x, flag)
      else if (what2what.eq.'v2m') then
        call Vector2Matrix(way, i, j, k, M, N, L, a, x, flag)
      else
        flag = .false.
      end if

      end subroutine MatrixesAndVectors

c-----[--.----+----.----+----.---------------------------------------]

      subroutine Matrix2Vector(way, i, j, k, M, N, L, a, x, flag)
      implicit none
```

```
      integer      :: i, j, k, M, N, L, a, x
      logical      :: flag
      character(*) :: way

      if (way.eq.'row')then
        call Matrix2VectorRowWise(i, j, k, M, N, L, a, x, flag)
      else if (way.eq.'col') then
        call Matrix2VectorColumnWise(i, j, k, M, N, L, a, x, flag)
      else
        flag = .false.
      end if

      end subroutine Matrix2Vector

c-----[--.----+----.----+----.----------------------------------------]

      subroutine Matrix2VectorColumnWise(i, j, k, M, N, L, a, x, flag)

      implicit none

      integer     :: i, j, k, M, N, L, a, x
      logical     :: flag

      flag = .true.
      if (M.eq.1.and.N.eq.1.and.L.eq.1) then
        x = a
      else if (M.gt.1.and.N.eq.1.and.L.eq.1) then
        x = i+a-1
      else if (M.gt.1.and.N.gt.1.and.L.eq.1) then
        x = (j-1)*M+i+a-1
      else if (M.gt.1.and.N.gt.1.and.L.gt.1) then
        x = (k-1)*M*N+(j-1)*M+i+a-1
      else
        flag = .false.
      end if

      end subroutine Matrix2VectorColumnWise

c-----[--.----+----.----+----.----------------------------------------]

      subroutine Matrix2VectorRowWise(i, j, k, M, N, L, a, x, flag)

      implicit none

      integer :: i, j, k, M, N, L, a, x
      logical :: flag

      flag = .true.
      if (M.eq.1.and.N.eq.1.and.L.eq.1) then
        x = a
      else if (M.gt.1.and.N.eq.1.and.L.eq.1) then
        x = i+a-1
      else if (M.gt.1.and.N.gt.1.and.L.eq.1) then
        x = (i-1)*N+j+a-1
      else if (M.gt.1.and.N.gt.1.and.L.gt.1) then
        x = (k-1)*M*N+(i-1)*N+j+a-1
      else
        flag = .false.
      end if

      end subroutine Matrix2VectorRowWise
```

```
c-----[--.----+----.----+----.----------------------------------------]

      subroutine Vector2Matrix(way, i, j, k, M, N, L, a, x, flag)

      implicit none

      integer      :: i, j, k, M, N, L, a, x
      logical      :: flag
      character(*) :: way

      if (way.eq.'row')then
        call Vector2MatrixRowWise(i, j, k, M, N, L, a, x, flag)
      else if (way.eq.'col') then
        call Vector2MatrixColumnWise(i, j, k, M, N, L, a, x, flag)
      else
        flag = .false.
      end if

      end subroutine Vector2Matrix

c-----[--.----+----.----+----.----------------------------------------]

      subroutine Vector2MatrixRowWise(i, j, k, M, N, L, a, x, flag)

      implicit none

      integer :: i, j, k, M, N, L, a, x
      logical :: flag

      flag = .true.
      if (M.eq.1.and.N.eq.1.and.L.eq.1) then
        i = 1
        j = 1
        k = 1
      else if (M.gt.1.and.N.eq.1.and.L.eq.1) then
        i = x-a+1
        j = 1
        k = 1
      else if (M.gt.1.and.N.gt.1.and.L.eq.1) then
        i = (x+1-a)/N+1
        j = x+1-a-N*(i-1)
        k = 1
      else if (M.gt.1.and.N.gt.1.and.L.gt.1) then
        k = (x+1-a)/(M*N)+1
        i = (x+1-a-M*N*(k-1))/N+1
        j = x+1-a-M*N*(k-1)-N*(i-1)
      else
        flag = .false.
      end if

      end subroutine Vector2matrixRowWise

c-----[--.----+----.----+----.----------------------------------------]

      subroutine Vector2MatrixColumnWise(i, j, k, M, N, L, a, x, flag)

      implicit none

      integer :: i, j, k, M, N, L, a, x
      logical :: flag
```

```
      flag = .true.
      if (M.eq.1.and.N.eq.1.and.L.eq.1) then
        i = 1
        j = 1
        k = 1
      else if (M.gt.1.and.N.eq.1.and.L.eq.1) then
        i = x-a+1
        j = 1
        k = 1
      else if (M.gt.1.and.N.gt.1.and.L.eq.1) then
        j = (x+1-a)/M+1
        i = x+1-a-M*(j-1)
        k = 1
      else if (M.gt.1.and.N.gt.1.and.L.gt.1) then
        k = (x+1-a)/(M*N)+1
        j = (x+1-a-M*N*(k-1))/M+1
        i = x+1-a-M*N*(k-1)-M*(j-1)
      else
        flag = .false.
      end if


      end subroutine Vector2MatrixColumnWise

c-----[--.----+----.----+----.---------------------------------------]
```

Τώρα, έχοντας αυτοματοποιήσει τη διαδικασία μετασχηματισμού των δεικτών είμαστε σε θέση να χρησιμοποιήσουμε τους μετασχηματισμούς αυτούς για να πάρουμε από το διάνυσμα της κύριας μνήμης τα περιεχόμενά του και για να γράψουμε σε αυτό. Αρκεί να γνωρίζουμε τη θέση στην οποία αποθηκεύεται το πρώτο στοιχείο του μητρώου μας στο διάνυσμα της κύριας μνήμης και τον τρόπο με τον οποίο είναι αποθηκευμένο το μητρώο σε αυτό, ανά γραμμή δηλαδή ή ανά στήλη. Στην παρούσα φάση ενδιαφερόμαστε για να πάρουμε τα αποτελέσματα από την κύρια μνήμη και να τα τοποθετήσουμε σε ένα αρχείο. Για το σκοπό αυτό μπορούμε να φτιάξουμε μερικές ρουτίνες για κάθε περίπτωση που μας ενδιαφέρει ή να φτιάξουμε μία που να λαμβάνει ως όρισμα το μητρώο που μας ενδιαφέρει με τη μορφή ακολουθίας χαρακτήρων μιας και οι περιπτώσεις των μητρώων που μας ενδιαφέρουν είναι συγκεκριμένες. Γενικά θα προτιμήσουμε τη δεύτερη περίπτωση, γιατί μας απαλλάσσει από τον κόπο του να προγραμματίζουμε ξανά και ξανά.

I.2. Προγραμματισμός ενός νέου νόμου υλικού

I.2.1. Γενικά

Ο νόμος υλικού που μας ενδιαφέρει εκφράζεται συνήθως με κάποια έκφραση του μέτρου ελαστικότητας ή σε πιο γενική μορφή του μητρώου ελαστικότητας συναρτήσει των τάσεων που έχουμε. Στο FEAP αυτό υλοποιείται στη γενική του μορφή προσθέτοντας στο κατάλληλο υποπρόγραμμα τον κώδικα ο οποίος λαμβάνει το μητρώο των παραμορφώσεων και αποδίδει για αυτές τις παραμορφώσεις το μητρώο των τάσεων και το τρέχον μητρώο ελαστικότητας.

Ο προγραμματισμός ενός τέτοιου κώδικα μπορεί να είναι από απλός έως αρκετά σύνθετος. Ο λόγος είναι ότι αν το μέτρο ελαστικότητας είναι συνάρτηση των τάσεων που προκύπτουν, δημιουργείται πεπλεγμένο πρόβλημα, αφού στον κώδικα εισέρχονται οι παραμορφώσεις και οι τάσεις δεν έχουν υπολογιστεί ακόμα. Για να υπολογιστούν όμως απαιτείται το μέτρο ή το μητρώο ελαστικότητας, το οποίο όμως για να υπολογιστεί

χρειάζεται τις τάσεις. Και έτσι, από τη στιγμή που οι υπολογισμοί δεν μπορούν να γίνουν ο ένας μετά τον άλλο, δημιουργείται μοιραία ένα σύστημα εξισώσεων που αποτελείται από τις κλασσικές εξισώσεις που συνδέουν τις τάσεις και τις παραμορφώσεις και ο επιπλέον άγνωστος που είναι το μέτρο ελαστικότητας, αντισταθμίζεται από την επιπλέον εξίσωση στο σύστημα του νόμου του υλικού. Η φύση και η δυσκολία του συστήματος εξαρτάται προφανώς από τις εξισώσεις που το απαρτίζουν, κυρίως δηλαδή από την εξίσωση του νόμου του υλικού, αφού οι άλλες είναι συγκεκριμένες. Σε περίπτωση που ο νόμος του υλικού δεν αναφέρεται μόνο στο μέτρο ελαστικότητας, που είναι μία μόνο παράμετρος, αλλά γενικά στο μητρώο ελαστικότητας, τότε οι άγνωστες παράμετροι είναι περισσότερες. Κάθε μία όμως από αυτές αντισταθμίζεται από την εξίσωση του νόμου του υλικού που την περιέχει.

Η επίλυση ενός τέτοιου συστήματος γίνεται συνήθως με αριθμητικές μεθόδους, ενώ ελάχιστες είναι οι περιπτώσεις που μπορεί να επιλυθεί αναλυτικά. Κλασσικές μέθοδοι είναι του Bolzano, των Newton- Raphson και από εκεί και πέρα ξεκινάει μεγάλη ποικιλία αριθμητικών και προσεγγιστικών μεθόδων και παραλλαγών προσαρμοσμένων κάθε φορά στην ειδική περίπτωση του προβλήματος που αντιμετωπίζουμε. Η επιλογή της κατάλληλης αριθμητικής μεθόδου είναι κομβικό σημείο, γιατί η υπορουτίνα του νόμου υλικού θα εκτελεστεί πολλές φορές για κάθε πεπερασμένο στοιχείο στην ανάλυση, οπότε έχει μεγάλη σημασία η ελαχιστοποίηση του χρόνου εκτέλεσής της. Παρ' όλ' αυτά όμως η επιλογή της πλέον καταλληλότερης αριθμητικής μεθόδου ξεφεύγει από τους στόχους της εργασίας αυτής.

I.2.2. Προγραμματισμός νόμου υλικού $\mathbf{E = k * (\sigma_1 + \sigma_2 + \sigma_3)^n}$

Ο κώδικας που υλοποιεί τη λύση με βάση αυτό το νόμο υλικού και που παρεμβαίνει στις υπορουτίνες umati0 και umatl0 είναι ο ακόλουθος:
Για την παρέμβαση στην υπορουτίνα umati0:

```
c$Id:$
      subroutine umati0(type,vv, d, ud, n1,n3)

c      * * F E A P * * A Finite Element Analysis Program

c....  Copyright (c) 1984-2011: Regents of the University of California
c                               All rights reserved

c-----[--.----+----.----+----.-----------------------------------------]
c     Modification log                                 Date (dd/mm/year)
c       Original version                                   01/11/2006
c-----[--.----+----.----+----.-----------------------------------------]
c      Purpose: Dummy user material model routine

c      Inputs:
c         type   - Name of material model
c         vv(5)  - Command line real data
c         d(*)   - Program material parameter data

c      Outputs:
c         ud(*)  - Material parameter data for model
c         n1     - Number of history items/point (time   dependent)
c         n3     - Number of history items/point (time independent)
c-----[--.----+----.----+----.-----------------------------------------]
      implicit  none

      logical   pcomp
      character type*15
```

```
      integer   n1,n3
      real*8    vv(5),d(*),ud(*)

c     Set command name

      if(pcomp(type,'mat0',4)) then     ! Default  form DO NOT CHANGE
c       type = 'name'                   ! Specify 'name'
        type = 'mattest1'

c     Input user data and save in ud(*) array

      else                              ! Perform input for user data
        n1 = 3
        ud(1) = vv(1) ! k
        ud(2) = vv(2) ! n
        ud(3) = vv(3) ! v
      endif

      end
```

Για την παρέμβαση στην υπορουτίνα umatl0:

```
c$Id:$
      subroutine umatl0(eps,theta,td,d,ud,hn,h1,nh,ii,istrt, sig,dd,isw)

c      * * F E A P * * A Finite Element Analysis Program

c....  Copyright (c) 1984-2011: Regents of the University of California
c                               All rights reserved

c-----[--.----+----.----+----.-----------------------------------------]
c     Modification log                                Date (dd/mm/year)
c       Original version                                  01/11/2006
c-----[--.----+----.----+----.-----------------------------------------]
c     Purpose: User Constitutive Model 1

c     Input:
c          eps(*)  -  Current strains at point      (small deformation)
c                  -  Deformation gradient at point (finite deformation)
c          theta   -  Trace of strain at point
c                  -  Determinant of deformation gradient
c          td      -  Temperature change
c          d(*)    -  Program material parameters (ndd)
c          ud(*)   -  User material parameters (nud)
c          hn(nh)  -  History terms at point: t_n
c          h1(nh)  -  History terms at point: t_n+1
c          nh      -  Number of history terms
c          ii      -  Current point number
c          istrt   -  Start state: 0 = elastic; 1 = last solution
c          isw     -  Solution option from element

c     Output:
c          sig(*)  -  Stresses at point.
c                     N.B. 1-d models use only sig(1)
c          dd(6,*) -  Current material tangent moduli
c                     N.B. 1-d models use only dd(1,1) and dd(2,1)
c-----[--.----+----.----+----.-----------------------------------------]
      implicit none

      integer  nh,istrt,isw, ii
      real*8   td
      real*8   eps(*),theta(*),d(*),ud(*),hn(nh),h1(nh), sig(*),dd(6,*)

      real*8   nu, k, n
```

```
      integer  command, i, l
      double precision E, En, E1, E2, a(4),f1, f2, s(3)

c     Compute and output stress (sig) and (moduli)

      k = ud(1)
      n = ud(2)
      nu = ud(3)
      go to 125
  100 sig(1) = E/(1+nu)/(1-2*nu)*((1-nu)*eps(1)+nu*eps(2)+nu*eps(3))
      sig(2) = E/(1+nu)/(1-2*nu)*(nu*eps(1)+(1-nu)*eps(2)+nu*eps(3))
      sig(3) = E/(1+nu)/(1-2*nu)*(nu*eps(1)+nu*eps(2)+(1-nu)*eps(3))
      sig(4) = 2*E/(1+nu)*eps(4)
      sig(5) = 2*E/(1+nu)*eps(5)
      sig(6) = 2*E/(1+nu)*eps(6)
      a(4) = 1
      a(3) = -(sig(1)+sig(2)+sig(3))
      a(2) = sig(1)*sig(2)+sig(2)*sig(3)+sig(1)*sig(3)-(sig(4)**2+
     2sig(5)**2+sig(6)**2)
      a(1) = -(sig(1)*sig(2)*sig(3)+2*sig(4)*sig(5)*sig(6)-
     2sig(1)*sig(5)**2-sig(2)*sig(6)**2-sig(3)*sig(4)**2)
      call Aplopoihsh(a(1),a(2),a(3),a(4))
      call cubic(a,s,l)
      En = k*(s(1)+s(2)+s(3))**n
      go to (200, 300, 400) command

  125 E1 = 0
      E2 = 2
  150 E = E1
      command = 1
      go to 100
  200 f1 = E-En
      E = E2
      command = 2
      go to 100
  300 f2 = E-En
      if (f1*f2>0) then
        E2 = 2*E2
        go to 150
      end if
  350 E = (E1+E2)/2
      command = 3
      go to 100
  400 if (f1*(E-En)<0) then
        f2 = E-En
        E2 = E
      else
        f1 = E-En
        E1 = E
      end if
      if (E2-E1>1e-4) then
        go to 350
      end if
      E = (E1+E2)/2
      sig(1) = E/(1+nu)/(1-2*nu)*((1-nu)*eps(1)+nu*eps(2)+nu*eps(3))
      sig(2) = E/(1+nu)/(1-2*nu)*(nu*eps(1)+(1-nu)*eps(2)+nu*eps(3))
      sig(3) = E/(1+nu)/(1-2*nu)*(nu*eps(1)+nu*eps(2)+(1-nu)*eps(3))
      sig(4) = 2*E/(1+nu)*eps(4)
      sig(5) = 2*E/(1+nu)*eps(5)
      sig(6) = 2*E/(1+nu)*eps(6)
      a(4) = 1
      a(3) = -(sig(1)+sig(2)+sig(3))
```

```
      a(2) = sig(1)*sig(2)+sig(2)*sig(3)+sig(1)*sig(3)-(sig(4)**2+
     2sig(5)**2+sig(6)**2)
      a(1) = -(sig(1)*sig(2)*sig(3)+2*sig(4)*sig(5)*sig(6)-
     2sig(1)*sig(5)**2-sig(2)*sig(6)**2-sig(3)*sig(4)**2)
      call Aplopoihsh(a(1),a(2),a(3),a(4))
      call cubic(a,s,l)
      dd(1,1) = E*(1-nu)/(1+nu)/(1-2*nu)
      dd(1,2) = E*nu/(1+nu)/(1-2*nu)
      dd(1,3) = E*nu/(1+nu)/(1-2*nu)
      dd(2,1) = E*nu/(1+nu)/(1-2*nu)
      dd(2,2) = E*(1-nu)/(1+nu)/(1-2*nu)
      dd(2,3) = E*nu/(1+nu)/(1-2*nu)
      dd(3,1) = E*nu/(1+nu)/(1-2*nu)
      dd(3,2) = E*nu/(1+nu)/(1-2*nu)
      dd(3,3) = E*(1-nu)/(1+nu)/(1-2*nu)
      dd(4,4) = 2*E/(1+nu)
      dd(5,5) = 2*E/(1+nu)
      dd(6,6) = 2*E/(1+nu)

      end

c-----[--.----+----.----+----.---------------------------------------]

      subroutine cubic(a,x,l)
c     ***cubic*************************************************08.11.1986
c     solution of a cubic equation
c     Equations of lesser degree are solved by the appropriate formulas.
c     The solutions are arranged in ascending order.
c     No warranty, always test this subroutine after downloading.
c     ********************************************************************
c     a(0:3) : (i)  vector containing the polynomial coefficients
c     x(1:l) : (o)  results
c     l      : (o)  number of valid solutions (beginning with x(1))
c     ********************************************************************

      implicit double precision(a-h,o-z)
      dimension a(0:3),x(3),u(3)
      parameter(pi=3.1415926535897932d+0,third=1.d+0/3.d+0)
      intrinsic min,max,acos
c
c     define cubic root as statement function
      cbrt(z)=sign(abs(z)**third,z)
c
c     ==== determine the degree of the polynomial ====
c
      if (a(3).ne.0.d+0) then
c
c       cubic problem
       w=a(2)/a(3)*third
       p=(a(1)/a(3)*third-w**2)**3
       q=-.5d+0*(2.d+0*w**3-(a(1)*w-a(0))/a(3))
       dis=q**2+p
       if (dis.lt.0.d+0) then
c         three real solutions!
c         confine the argument of acos to the interval [-1;1]!
        phi=acos(min(1.d+0,max(-1.d+0,q/sqrt(-p))))
        p=2.d+0*(-p)**(5.d-1*third)
        do 100 i=1,3
  100       u(i)=p*cos((phi+dble(2*i)*pi)*third)-w
        x(1)=min(u(1),u(2),u(3))
        x(2)=max(min(u(1),u(2)),min(u(1),u(3)),min(u(2),u(3)))
        x(3)=max(u(1),u(2),u(3))
```

```
        l=3
       else
c         only one real solution!
        dis=sqrt(dis)
        x(1)=cbrt(q+dis)+cbrt(q-dis)-w
        l=1
       end if
c
      else if (a(2).ne.0.d+0) then
c
c       quadratic problem
       p=5.d-1*a(1)/a(2)
       dis=p**2-a(0)/a(2)
       if (dis.ge.0.d+0) then
c         two real solutions!
        x(1)=-p-sqrt(dis)
        x(2)=-p+sqrt(dis)
        l=2
       else
c         no real solution!
        l=0
       end if
c
      else if (a(1).ne.0.d+0) then
c
c       linear equation
       x(1)=-a(0)/a(1)
       l=1
c
      else
c       no equation
       l=0
      end if
c
c     ==== perform one step of a newton iteration in order to minimize
c          round-off errors ====
      do i=1,l
       x(i)=x(i)-(a(0)+x(i)*(a(1)+x(i)*(a(2)+x(i)*a(3))))
     *  /(a(1)+x(i)*(2.d+0*a(2)+x(i)*3.d+0*a(3)))
      end do
      if (l==1) then
        x(2) = x(1)
        x(3) = x(1)
      else if (l==2) then
        x(3) = x(2)
      end if
      return
      end

c-----[--.----+----.----+----.----------------------------------------]

      subroutine Aplopoihsh(a1,a2,a3,a4)

      implicit none

      double precision :: a1, a2, a3, a4, a
      integer          :: i, command, min, max

      a = a1
   10 i = 0
      if (abs(a)>1) then
        do while (abs(a)>1)
```

```
          a = a/10
          i = i+1
        end do
        i = i-1
      else if (abs(a)<1.and.a/=0) then
        do while (abs(a)<1)
          a = a*10
          i = i-1
        end do
      end if
      go to (20, 30, 40) command
      min = i
      max = i
      a = a2
      command = 1
      go to 10
   20 if (i<min) min = i
      if (i>max) max = i
      a = a3
      command = 2
      go to 10
   30 if (i<min) min = i
      if (i>max) max = i
      a = a4
      command = 3
      go to 10
   40 if (i<min) min = i
      if (i>max) max = i
      i = (min+max)/2
      a1 = a1/10**i
      a2 = a2/10**i
      a3 = a3/10**i
      a4 = a4/10**i

      end subroutine Aplopoihsh

c-----[--.----+----.----+----.----------------------------------------]
```

Στο σημείο αυτό να παρατηρήσουμε κάτι που δεν γίνεται εύκολα αντιληπτό. Η έννοια της συνάρτησης αποκτά μια ελαφρώς διαφοροποιημένη έννοια από αυτή που έχουμε συνηθίσει από τα μαθηματικά. Μέχρι στιγμής είχαμε συνηθίσει ότι η συνάρτηση συνοδεύεται από έναν τύπο ο οποίος παίρνει μία ή περισσότερες ανεξάρτητες μεταβλητές και από αυτές παράγει μία εξαρτημένη. Στην πραγματικότητα όμως οι συναρτήσεις είναι κάτι περισσότερο. Είναι διαδικασίες που, ναι μεν λαμβάνουν κάποιες ανεξάρτητες μεταβλητές, αλλά η τελική εξαρτημένη δεν προκύπτει πάντα από κάποιον μαθηματικό τύπο. Μπορεί να προκύπτει και από μία γενικευμένη διαδικασία που περιλαμβάνει επαναλήψεις, συνθήκες και οτιδήποτε άλλο, όπως δηλαδή είναι και ο γενικός μαθηματικός της ορισμός. Στο παράδειγμά μας η ανεξάρτητη μεταβλητή ήταν το E και η εξαρτημένη το E_n, το οποίο όμως προέκυπτε μέσα από επαναληπτικές διαδικασίες προσέγγισης της πρώτης ρίζας, επίλυσης τριωνύμων κτλ, κάθε άλλο παρά από έναν κλειστό μαθηματικό τύπο δηλαδή. Και όλα αυτά εμπεριέχονται στη συνάρτηση f όταν την ορίζουμε δηλώνοντας $E_n = f(E)$. Ένα άλλο παράδειγμα μιας τέτοιας συνάρτησης θα μπορούσε να είναι μία με ανεξάρτητη μεταβλητή το συντελεστή του δευτεροβάθμιου όρου ενός πολυωνύμου που να αποδίδει ως εξαρτημένη τη μικρότερη από τις ρίζες του.

Η παραπάνω διαδικασία μέσω της υλοποίησης του θεωρήματος Bolzano για συνεχείς συναρτήσεις στο πεδίο αναζήτησης της ρίζας τους είναι αρκετά γενική και μπορεί να χρησιμοποιηθεί για οποιονδήποτε νόμο υλικού με ελάχιστη τροποποίηση στον παραπάνω κώδικα, συνήθως εκεί που υπολογίζεται η τιμή του E_n. Ιδιαίτερη προσοχή

χρειάζεται στην τελική συνάρτηση g στην οποία θα εφαρμοστεί το θεώρημα, καθότι χρειάζεται η συνάρτηση αυτή να είναι συνεχής στο τελικό διάστημα $[E_1, E_2]$ (στην περίπτωσή μας $E_1 = 0$) στο οποίο θα ξεκινήσουμε τις επαναλήψεις αναζήτησης της ρίζας. Εάν δεν είναι, τότε μια λύση είναι να χωρίσουμε το διάστημα αυτό σε υποδιαστήματα στα οποία η g να είναι συνεχής και να αναζητήσουμε λύση σε κάθε ένα από αυτά. Εάν η g είναι αρκετά πολύπλοκη ώστε να γίνει κάτι τέτοιο, τότε απαιτείται να γίνει πρώτα μελέτη της με αριθμητικό ή αναλυτικό τρόπο προτού αναζητηθούν οι ρίζες της. Ένας απλός τρόπος για να γίνει αυτό είναι με τη σχεδίαση γραφικής παράστασής της, ώστε να εντοπίσουμε και πιο εύκολα το διάστημα στο οποίο να τις αναζητήσουμε.

<u>Παράρτημα ΙΙ:</u>
Κώδικες σε VBA που κατασκευάστηκαν.

II.1. Ανάγκη χρήσης μακροεντολών

Για τη μελέτη ενός κώδικα χρειάζεται να κάνουμε ένα σύνολο ενεργειών που μπορεί να είναι από πολύ απλές έως και αρκετά πολύπλοκες. Στις απλές περιπτώσεις είναι αρκετό το να διατρέξουμε τον κώδικα γραμμή- γραμμή και να παρατηρήσουμε τις εντολές που εκτελούνται, υπάρχουν όμως περιπτώσεις που, είτε λόγω της μεγάλης έκτασης του κώδικα είτε λόγω πολυπλοκότητας αυτό δεν είναι εφικτό. Στις περιπτώσεις αυτές οι απαιτούμενες ενέργειες δεν έχουν παρά να γίνουν με τη χρήση άλλων κωδίκων που θα έχουν ως στόχο τη μελέτη του κώδικα που μας ενδιαφέρει. Στο σημείο αυτό, οι δρόμοι που μπορεί να επιλέξει κανείς είναι τρεις: Να χρησιμοποιήσει όποια γλώσσα προγραμματισμού θέλει και να φτιάξει πρόγραμμα με το οποία θα διαβάσει τον υφιστάμενο κώδικα από τα αρχεία στα όποια είναι αποθηκευμένος στο δίσκο και από εκεί και πέρα να κάνει ό,τι ενέργειες του χρειάζονται για να συλλέξει τα δεδομένα που θέλει, να χρησιμοποιήσει το περιβάλλον μακροεντολών που του προσφέρει ο compiler στον οποίο εργάζεται και να κατασκευάσει μακροεντολές που να κάνουν την ίδια εργασία και τέλος να τροποποιήσει τον υπάρχοντα κώδικα με τέτοιο τρόπο ώστε σε συγκεκριμένα σημεία να δημιουργεί αναφορές συνήθως σε αρχεία txt με τις απαραίτητες πληροφορίες. Και οι τρεις μέθοδοι έχουν τα πλεονεκτήματά του και τα μειονεκτήματά τους, τα οποία εκτίθενται στη συνέχεια και είναι στην κρίση του εκάστοτε προγραμματιστή να αποφασίσει ποια θα επιλέξει.

Η μέθοδος με τις μακροεντολές είναι σίγουρα πιο διαδεδομένη και πιο εύκολη προγραμματιστικά, αφού χρησιμοποιεί όλα τα ενσωματωμένα χαρακτηριστικά που έχει ούτως ή άλλως ο compiler και κάτω από άλλες συνθήκες ο προγραμματιστή θα χρειαζόταν να τα προγραμματίσει ο ίδιος. Είναι σίγουρα πιο φιλική προς τον προγραμματιστή και το χρήστη και συνήθως απαιτεί πολύ λιγότερες γραμμές κώδικα για να υλοποιηθεί. Το μειονέκτημά της είναι ότι χρειάζεται πολύ καλή γνώση των ειδικών εντολών που διαθέτει ο compiler, ώστε αυτές να χρησιμοποιηθούν κατ' ευθείαν και να απαλλάξουν τον προγραμματιστή από περιττό κόπο προγραμματισμού. Η εύρεση όμως των επιθυμητών εντολών δεν είναι πάντα εύκολη. Ειδικά σε compilers υψηλού επιπέδου υπάρχει τεράστιος όγκος διαθέσιμων εντολών και πολλές φορές ο χρόνος που απαιτείται για την εύρεση της κατάλληλης εντολής είναι συγκρίσιμος με αυτόν που θα χρειαζόταν για να προγραμματιστεί εκ νέου. Η μεγάλη υπεροχή των μακροεντολών βρίσκεται σε περιπτώσεις που χρειάζεται να αναφερθούμε σε τρέχουσες τιμές μεταβλητών ή να εντοπίσουμε ιεραρχίες στοιβών, λειτουργίες που είναι πολύ δύσκολο να προγραμματιστούν από την αρχή (from scratch), αλλά υπάρχουν σε έτοιμες εντολές σε όλους σχεδόν τους compilers. Υπάρχουν όμως ακόμα δύο σημαντικά μειονεκτήματα. Επειδή οι μακροεντολές για να υλοποιηθούν χρησιμοποιούν πιο υψηλού επιπέδου γλώσσες προγραμματισμού, ο χρόνος εκτέλεσής τους είναι αρκετά μεγάλος. Επίσης καταναλώνουν πάρα πολλούς πόρους τόσο σε μνήμη όσο σε επεξεργαστική ισχύ, οπότε δεν είναι σπάνιο να μην μπορούν να ολοκληρωθούν λόγω έλλειψης υπολογιστικής ισχύος, ειδικά στις περιπτώσεις που χρειάζεται να εκτελούνται παράλληλα με τον κώδικα. Χρειάζεται λοιπόν προσοχή με τη διαχείριση πόρων. Επίσης οι μακροεντολές προγραμματίζονται συνήθως με μία μόνο γλώσσα προγραμματισμού, οπότε είναι ανάγκη να γνωρίζουμε τη γλώσσα αυτή ή να μπορέσουμε να τη μάθουμε γρήγορα. Στην περίπτωσή μας χρησιμοποιήσαμε visual basic.

Η μέθοδος επεξεργασίας των αρχείων στα οποία είναι αποθηκευμένος ο κώδικας είναι σίγουρα η πιο δύσκολη από όλες. Στην ουσία, αναλόγως βέβαια και το βάθος που θέλει να φτάσει, ο προγραμματιστής καλείται να επαναπρογραμματίσει έναν compiler από την αρχή, που είναι από τις πιο δύσκολες προγραμματιστικές εργασίες που υπάρχουν. Σαφέστατα έχει το πλεονέκτημα ότι ο προγραμματιστής έχει ελευθερία επιλογής της γλώσσας προγραμματισμού που επιθυμεί, αρκεί αυτή να έχει δυνατότητα επεξεργασίας αρχείων, δυνατότητα που υπάρχει σε όλες σχεδόν τις γλώσσες. Έτσι παρακάμπτεται το πρόβλημα της ταχύτητας και ανεπάρκειας υπολογιστικών πόρων. Με τον τρόπο αυτό έχει

τη δυνατότητα να συλλέγει όποιες πληροφορίες θέλει και να επεξεργάζεται τον κώδικα που έχει με μοναδικό περιορισμό τις ικανότητές του.

Υπάρχει και η δυνατότητα τροποποίησης του υπάρχοντος κώδικα. Μπορεί για παράδειγμα να εισάγει σε κάποια σημεία του κώδικα εντολές εμφάνισης των τιμών κάποιων μεταβλητών, ή να εισάγει μετρητές σε προγράμματα που δείχνουν πόδες φορές έχουν κληθεί, κτλ. Εδώ υπάρχει το πλεονέκτημα της εξοικείωσης με τον κώδικα πάνω στον οποίο δουλεύει, χρειάζεται όμως προσοχή μήπως αλλοιωθεί με τις παρεμβάσεις αυτές η λειτουργικότητα και αποτελεσματικότητά του και μετά δεν μπορούμε να αναιρέσουμε τις αλλαγές που έχουμε κάνει.

Ο πιο συνηθισμένος τρόπος είναι, όπως και στις περισσότερες φορές, ο συνδυαστικός. Χρησιμοποιούμε μακροεντολές για να αποφύγουμε δύσκολα προγραμματιστικά μονοπάτια και προσπέλαση των αρχείων του κώδικα για να εντοπίσουμε απλά δεδομένα, όπως πόσες φορές εμφανίζεται μια εντολή και σε ποιες γραμμές κώδικα, ή τι μεταβλητές ορίζονται κτλ και τροποποιούμε τον αρχικό κώδικα για να αποφύγουμε διαδικασίες που για να γίνουν με μακροεντολές, χρειάζεται αυτές να εκτελούνται παράλληλα με την εκτέλεση του κώδικα και είναι χρονοβόρες ή περιέχουν τον κίνδυνο να μην εκτελεστούν μέχρι το τέλος.

Παρακάτω παρουσιάζουμε τους τρόπους και τις μεθόδους που εμείς χρησιμοποιήσαμε για τη μελέτη του κώδικα που FEAP.

ΙΙ.2. Μακροεντολές για προσθήκη breakpoints (visual basic)

Σε έναν κώδικα μεγάλο για τη μελέτη του χρειάζεται σε κάποια σημεία που παρουσιάζουν ιδιαίτερο ενδιαφέρον να τοποθετούνται breakpoints, σημεία δηλαδή στα οποία η εκτέλεση του κώδικα σταματάει και μπορούμε να εξετάσουμε έτσι ορισμένα πράγματα που μας ενδιαφέρουν όπως οι τρέχουσες τιμές των μεταβλητών, ή να διαπιστώσουμε αν η εκτέλεση του κώδικα διέρχεται από κάποιο σημείο που μας ενδιαφέρει. Υπάρχουν περιπτώσεις που αυτό γίνεται με το χέρι, αφού τα σημεία αυτά είναι λίγα κάθε φορά και αυτή είναι και η συνήθης περίπτωση, αλλά υπάρχουν και φορές που το πλήθος των breakpoints είναι μεγάλο και δεν μπορούν να τοποθετηθούν έτσι. Στις περιπτώσεις αυτές κρίνεται απαραίτητη η σύνταξη μακροεντολών που αναλαμβάνουν την εργασία αυτή. Τέτοιες περιπτώσεις είναι όταν για παράδειγμα θέλουμε να τοποθετήσουμε breakpoint σε όλες τις εντολές read που υπάρχουν στον κώδικα, να μετρήσουμε πόσες φορές εκτελείται η κάθε μία και να λάβουμε τα αποτελέσματα σε διάγραμμα. Στην περίπτωση του visual studio μπορούμε να αναζητήσουμε μια ακολουθία χαρακτήρων και να μας εμφανίσει σε ξεχωριστό παράθυρο σε ποιες γραμμές του κώδικα εμφανίζεται. Σε όλες αυτές τις γραμμές μπορούμε με μία μακροεντολή να προσθέσουμε breakpoints. Ο κώδικας για αυτή την εργασία είναι παρουσιάζεται παρακάτω ως υπορουτίνα με όνομα «`SetBreakpointsToFindResults1`».

Επόμενη εργασία που θα θέλαμε να κάνουμε είναι να βάλουμε breakpoint στην αρχή και στο τέλος κάθε συνάρτησης ή υπορουτίνας που χρησιμοποιείται. Αυτό μας εξυπηρετεί ώστε κάθε φορά που θα περνάει η εκτέλεση του προγράμματος από το breakpoint στην αρχή μπορούμε να προγραμματίσουμε να εκτελείται μία άλλη μακροεντολή που θα καταγράφει το όνομα της συνάρτησης ή υπορουτίνας σε ένα αρχείο txt με αποτέλεσμα να έχουμε μετά το πέρας της εκτέλεσης σε αυτό το αρχείο κειμένου όλα τα ονόματα των συναρτήσεων ή υπορουτίνων από τις οποίες έχει περάσει το πρόγραμμα, κάτι δηλαδή σαν ένα απλό διάγραμμα ροής. Το breakpoint στο τέλος χρησιμεύει για τον καθορισμό των εσοχών στο αρχείο κειμένου, από τις οποίες θα φαίνεται η ιεραρχία του τι καλεί τι, ώστε να υπάρχει μια πιο ξεκάθαρη εικόνα των μεγάλων και καθοριστικών υποπρογραμμάτων και των απλών. Στη συνέχεια το αρχείο αυτό μπορεί να υποστεί

επεξεργασία ώστε, αν κάποιο υποπρόγραμμα έχει κληθεί πολλές συνεχόμενες φορές, να μην είναι γράφεται στο αρχείο συνέχεια, αλλά μία φορά και το πλήθος των φορών που έχει εκτελεστεί να ακολουθεί μέσα σε παρένθεση. Το ίδιο μπορεί να γίνει και με ομάδες υποπρογραμμάτων. Για αυτό χρειαζόμαστε κώδικα που να ανιχνεύει τέτοιου είδους επαναλήψεις μέσα στο αρχικό αρχείο κειμένου και να παράγει ένα καινούριο, πιο καλογραμμένο και εύληπτο. Όμως αυτό ξεφεύγει από τους στόχους αυτής της εργασίας, οπότε δεν φτάσαμε μέχρι εκεί. Η διαδικασία τοποθέτησης των breakpoint στην αρχή και το τέλος ενός αρχείου γίνεται αντίστοιχα με τις υπορουτίνες που παρουσιάζονται παρακάτω με τα ονόματα «SetBreakpointsOnSubroutine» και «SetBreakpointsOnEndSubroutine» και για κάθε αρχείο στην αρχή και το τέλος με τις υπορουτίνες που παρουσιάζονται αντίστοιχα με τα όνοματα «SetBreakpointsOnSubroutineAllFiles» και «SetBreakpointsOnEndSubroutineAllFiles».

Ο κώδικας για τον οποίο γίνεται λόγος ανήκει σε μία μεγάλη module (έτσι οργανώνονται οι μακροεντολές στο visual studio) και είναι ο ακόλουθος:

```
Imports System
Imports EnvDTE
Imports EnvDTE80
Imports EnvDTE90
Imports EnvDTE90a
Imports EnvDTE100
Imports System.Diagnostics
Imports System.Text.RegularExpressions
Imports System.IO

Public Module Breakpoints

    Sub SetBreakpointsToFindResults1()

        Dim findResultsWindow As Window = _
DTE.Windows.Item(Constants.vsWindowKindFindResults1)
        Dim selection As TextSelection

        selection = findResultsWindow.Selection
        selection.SelectAll()

        Dim findResultsReader As New StringReader(selection.Text)
        Dim findResult As String = findResultsReader.ReadLine()_
        Dim findResultRegex As New
Regex("(?<Path>.*?)\((?<LineNumber>\d+)\):")

        While Not findResult Is Nothing
            Dim findResultMatch As Match = _
findResultRegex.Match(findResult)

            If findResultMatch.Success Then
                Dim path As String = _
findResultMatch.Groups.Item("Path").Value
                Dim lineNumber As Integer = _
Integer.Parse(findResultMatch.Groups.Item("LineNumber").Value)

                Try
                    DTE.Debugger.Breakpoints.Add("", path, _
lineNumber)
                Catch ex As Exception
                    ' breakpoints can't be added everywhere
                End Try
            End If
```

```
            findResult = findResultsReader.ReadLine()
        End While
        MsgBox("Breakpoints are set.")

    End Sub

    Sub SetBreakpointsOnSubroutineAllFiles()


        Dim FS As FileStream = New FileStream("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeapSubroutines.txt",_
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)
        Dim Path As String

        Path = SR.ReadLine
        Do Until Path = DBNull.Value.ToString
            SetBreakpointsOnSubroutine(Path)
            Path = SR.ReadLine
        Loop
        SR.Close()
        FS.Close()
        MsgBox("Breakpoints are set.")

    End Sub

    Sub SetBreakpointsOnEndSubroutineAllFiles()

        Dim FS As FileStream = New FileStream("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeapSubroutines.txt",_
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)
        Dim Path As String

        Path = SR.ReadLine
        Do Until Path = DBNull.Value.ToString
            SetBreakpointsOnEndSubroutine(Path)
            Path = SR.ReadLine
        Loop
        SR.Close()
        FS.Close()
        MsgBox("Breakpoints are set.")

    End Sub

    Function SetBreakpointsOnSubroutine(ByVal PathOfSubroutine As _
String)

        Dim FileLine As String
        Dim FS As FileStream = New FileStream(PathOfSubroutine, _
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)
        Dim LineNumber As Integer = 0
        Dim CountBlankContinuousLines As Integer
        Dim LinesToSetBreakpoints() As Integer
        Dim i As Integer = -1

        Do
            CountBlankContinuousLines = 0
            FileLine = DBNull.Value.ToString
```

```
            Do While CountBlankContinuousLines < 30 And FileLine = _
DBNull.Value.ToString
                FileLine = SR.ReadLine
                LineNumber = LineNumber + 1
                CountBlankContinuousLines = _
CountBlankContinuousLines + 1
            Loop
            If FileLine <> DBNull.Value.ToString Then
                CountBlankContinuousLines = _
CountBlankContinuousLines - 1
                If (FileLine.Contains("subroutine") = True Or _
                    FileLine.Contains("function") = True) And _
                    FileLine.Chars(0) <> "c" Then
                    i = i + 1
                    ReDim Preserve LinesToSetBreakpoints(i)
                    LinesToSetBreakpoints(i) = LineNumber
                End If
            End If
        Loop Until CountBlankContinuousLines = 30
        SR.Close()
        FS.Close()
        If i > -1 Then
            For Each LineNumber In LinesToSetBreakpoints
                DTE.Debugger.Breakpoints.Add("", PathOfSubroutine, _
LineNumber)
            Next
        End If

    End Function

    Function SetBreakpointsOnEndSubroutine(ByVal PathOfSubroutine As_
String)

        Dim FileLine As String
        Dim FS As FileStream = New FileStream(PathOfSubroutine, _
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)
        Dim CurrentLine, NumberOfWantedLine, _
CountBlankContinuousLines, LinesToSetBreakpoints() As Integer
        Dim i As Integer = -1

        CurrentLine = 0
        Do
            CountBlankContinuousLines = 0
            FileLine = SR.ReadLine
            CurrentLine = CurrentLine + 1
            Do While (CountBlankContinuousLines < 30 And FileLine = _
DBNull.Value.ToString)
                FileLine = SR.ReadLine
                CurrentLine = CurrentLine + 1
                CountBlankContinuousLines = _
CountBlankContinuousLines + 1
            Loop
            If FileLine <> DBNull.Value.ToString Then
                CountBlankContinuousLines = _
CountBlankContinuousLines - 1
                If IsStringFound(FileLine, "end") = True Then
                    NumberOfWantedLine = CurrentLine
                ElseIf (FileLine.Contains("subroutine") = True Or _
                        FileLine.Contains("function") = True) And _
                        FileLine.Chars(0) <> "c" And _
```

```
                        NumberOfWantedLine > 0 Then
                    i = i + 1
                    ReDim Preserve LinesToSetBreakpoints(i)
                    LinesToSetBreakpoints(i) = NumberOfWantedLine
                End If
            ElseIf NumberOfWantedLine > 0 Then
                i = i + 1
                ReDim Preserve LinesToSetBreakpoints(i)
                LinesToSetBreakpoints(i) = NumberOfWantedLine
            End If
        Loop Until CountBlankContinuousLines = 30
        SR.Close()
        FS.Close()
        If i > -1 Then
            For Each NumberOfWantedLine In LinesToSetBreakpoints
                DTE.Debugger.Breakpoints.Add("", PathOfSubroutine, _
NumberOfWantedLine)
            Next
        End If

    End Function

End Module
```

ΙΙ.3. Μακροεντολές δημιουργίας διαγράμματος ροής (visual basic)

Όπως είπαμε και πριν, ο στόχος της προσθήκης στην αρχή και στο τέλος ενός υποπρογράμματος ή υπορουτίνας, στη μεν αρχή στη γραμμή δήλωσής του, στο δε τέλος στην τελευταία γραμμή του εξαιρουμένων των σχολίων, breakpoint είναι να καταφέρουμε να καταγράψουμε σε ένα αρχείο κειμένου την πορεία που ακολουθεί η εκτέλεση του προγράμματος διαμέσου των υποπρογραμμάτων του, να καταρτίσουμε δηλαδή κάτι σαν ένα διάγραμμα ροής του. Η τεχνική που χρησιμοποιούμε για το σκοπό εαυτό είναι η εξής: Αρχικά βρίσκουμε την εντολή που μας δίνει το όνομα της τρέχουσας υπορουτίνας ή συνάρτησης. Στη συνέχεια ορίζουμε σε κάθε breakpoint που υπάρχει στην αρχή να εκτελείται κάθε φορά που το πρόγραμμα περνάει από εκεί μία μακροεντολή που αυξάνει την προπορευόμενη του ονόματός του εσοχή κατά ένα κενό. Στη συνέχεια εμφανίζει στο αρχείο κειμένου το όνομα της τρέχουσας υπορουτίνας ή συνάρτησης τοποθετώντας πριν από αυτό το απαιτούμενο πλήθος κενών, ώστε οι εσοχές να είναι τέτοιες που να ξεχωρίζει η ιεραρχία. Κάθε φορά που εκτελείται breakpoint που βρίσκεται στην αρχή το πλήθος των προπορευόμενων κενών αυξάνεται κατά ένα και τυπώνεται αμέσως μετά το όνομα της τρέχουσας υπορουτίνας ή συνάρτησης και κάθε φορά που η εκτέλεση διέρχεται από breakpoint που βρίσκεται στο τέλος, απλά μειώνεται το πλήθος των προπορευόμενων κενών κατά ένα. Σε αντίθεση με το όνομα του τρέχοντος υποπρογράμματος ή συνάρτησης που με το που μας το δίνει η κατάλληλη για το σκοπό αυτό εντολή το γράφουμε στο αρχείο και δεν το χρειαζόμαστε μετά σε κάτι, το τρέχον πλήθος των κενών χρειάζεται να μένει αποθηκευμένο και να μεταφέρεται από μακροεντολή σε μακροεντολή. Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε είτε καθολικές μεταβλητές για όλες αυτές τις μακροεντολές, είτε να το αποθηκεύουμε στο δίσκο σε αρχείο. Ασφαλώς θα περίμενε κανείς ο δεύτερος τρόπο να είναι πιο αργός σε σχέση με τον πρώτο, αλλά επειδή ο πρώτος καταλήγει να κολλάει πολλές φορές, ο δεύτερος είναι πιο αξιόπιστος. Να σημειωθεί ότι η εξαγωγή του διαγράμματος ροής με αυτόν τον τρόπο είναι σε κάθε περίπτωση ιδιαίτερα χρονοβόρα και για μεγάλους κώδικες υπάρχει σοβαρή πιθανότητα να εξαντληθεί η υπολογιστική ισχύς προτού το πρόγραμμα λυθεί. Στην περίπτωσή μας χρειαστήκαμε κάτι

περισσότερο από τρία εικοσιτετράωρα για να λάβουμε αποτελέσματα για διάγραμμα ροής χωρίς εσοχές.

Ο κώδικάς μας απαρτίζεται από δύο υπορουτίνες. Η πρώτη ονομάζεται «CreateFlowchartUp» και ορίζεται να εκτελείται όταν η εκτέλεση του προγράμματος διέρχεται από breakpoint στην αρχη και η δεύτερη ονομάζεται «CreateFlowchartDown» και ορίζεται να εκτελείται όταν η εκτέλεση του προγράμματος διέλθει από breakpoint στο τέλος. Ο κώδικας, ενταγμένος πάλι μέσα σε module όπως χρειάζεται να είναι οι μακροεντολές στο visual studio, είναι ο εξής:

```
Imports System
Imports EnvDTE
Imports EnvDTE80
Imports EnvDTE90
Imports EnvDTE90a
Imports EnvDTE100
Imports System.Diagnostics
Imports System.IO

Public Module Flowchart

    Sub CreateFlowchartUp()

        Dim path As String = "C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\flowchart_Idisk2b.txt"
        Dim pathSpacesToGo As String = "C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\SpacesToGo.txt"
        Dim pathName As String = "C:\Users\Εμμανουήλ
Ζέρ\Documents\διπλωματική\αρχεία txt\Name.txt"
        Dim NameOfCurrentSubroutine = _
DTE.Debugger.CurrentStackFrame.FunctionName.ToLower + ".f"
        Dim Spaces As String = ""
        Dim NameOfPreviousSubroutine As String
        Dim SpacesIn As Integer

        NameOfPreviousSubroutine = _
My.Computer.FileSystem.ReadAllText(pathName)
        SpacesIn = _
Convert.ToInt16(My.Computer.FileSystem.ReadAllText(pathSpacesToGo))
        If NameOfCurrentSubroutine <> NameOfPreviousSubroutine Then
            Try
                DTE.Windows.Item(NameOfPreviousSubroutine).Close()
            Catch ex As Exception
                'Can' t be closed.
            End Try
        End If
        SpacesIn = SpacesIn + 1
        For i = 1 To SpacesIn
            Spaces = Spaces + " "
        Next
        My.Computer.FileSystem.WriteAllText(path, Spaces + _
NameOfCurrentSubroutine + vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(pathSpacesToGo, _
SpacesIn.ToString, False)
        My.Computer.FileSystem.WriteAllText(pathName, _
NameOfCurrentSubroutine, False)

    End Sub

    Sub CreateFlowchartDown()
```

```
        Dim path As String = "C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\flowchart_Idisk2b.txt"
        Dim pathSpacesToGo As String = "C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\SpacesToGo.txt"
        Dim pathName As String = "C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\Name.txt"
        Dim NameOfCurrentSubroutine = _
DTE.Debugger.CurrentStackFrame.FunctionName.ToLower + ".f"
        Dim Spaces As String = ""
        Dim NameOfPreviousSubroutine As String
        Dim SpacesIn As Integer

        NameOfPreviousSubroutine = _
My.Computer.FileSystem.ReadAllText(pathName)
        SpacesIn = _
Convert.ToInt16(My.Computer.FileSystem.ReadAllText(pathSpacesToGo))
        If NameOfCurrentSubroutine <> NameOfPreviousSubroutine Then
            Try
                DTE.Windows.Item(NameOfPreviousSubroutine).Close()
            Catch ex As Exception
                'Can' t be closed.
            End Try
        End If
        SpacesIn = SpacesIn - 1
        For i = 1 To SpacesIn
            Spaces = Spaces + " "
        Next
        My.Computer.FileSystem.WriteAllText(pathSpacesToGo, _
SpacesIn.ToString, False)
        My.Computer.FileSystem.WriteAllText(pathName, _
NameOfCurrentSubroutine, False)

    End Sub

End Module
```

ΙΙ.4. Μακροεντολές διαχείρισης αρχείων (visual basic)

Για να μπορέσουμε να τοποθετήσουμε όμως breakpoint στην επιθυμητή θέση χρειαζόμαστε τη γραμμή του κώδικα στην οποία θα την τοποθετήσουμε και τη διαδρομή του αρχείου στο οποίο είναι αποθηκευμένος ο κώδικας με τη μορφή αρχείου κειμένου. Έτσι χρειαζόμαστε να αποθηκεύσουμε κάπου όλες τις διαδρομές των αρχείων του FEAP. Αυτό είναι δύσκολο να γίνει με το χέρι, γιατί τα αρχεία είναι πολλά, γίνεται όμως με μακροεντολή. Αρχικά εντοπίζουμε τις διαδρομές των φακέλων μέσα στους οποίους βρίσκονται τα αρχεία οι οποίες είναι 21 και κατασκευάζουμε μία μακροεντολή που καταγράφει σε ένα αρχείο τη διαδρομή κάθε αρχείου που περιέχεται σε κάθε έναν από αυτούς τους φακέλους. Ο κώδικας παρουσιάζεται τη συνέχεια ως υπορουτίνα με όνομα «`FindAdressesOfFeapFiles`».

Στη συνέχεια όμως χρειαζόμαστε μια μακροεντολή που να μπορεί να ξεχωρίζει από αυτά τα αρχεία ποια περιέχουν υπορουτίνα και ποια συνάρτηση. Για το σκοπό αυτό ελέγχουμε τις πρώτες γραμμές του κάθε αρχείου αν περιέχουν τις λέξεις «subroutine» ή «function» και αντιστοίχως τα ξεχωρίζουμε γράφοντας σε κάθε περίπτωση τις διαδρομές τους σε ξεχωριστά αρχεία κειμένου. Ο κώδικας που αναγνωρίζει αν ένα αρχείο είναι συνάρτηση ή υπορουτίνα παρουσιάζεται ως συνάρτηση με το όνομα «`FunctionOrSubroutine`» και ο κώδικας που γράφει σε διαφορετικά νέα αρχεία κατά περίπτωση τη διαδρομή του αρχείου αναλόγως παρουσιάζεται ως υπορουτίνα με όνομα

«FindAdressesOfFeapSubroutinesOrFunctions». Και πάλι παρουσιάζονται ενταγμένα σε module:

```
Imports System
Imports EnvDTE
Imports EnvDTE80
Imports EnvDTE90
Imports EnvDTE90a
Imports EnvDTE100
Imports System.Diagnostics
Imports System.IO

Public Module PrepareFiles

    Sub FindAdressesOfFeapFiles()

        Dim PathAdresses(20) As String
        Dim FileAdresses() As String
        Dim Path, FileAdress As String

        PathAdresses(0) = "C:\Program Files\feap\ver83\contact\main\"
        PathAdresses(1) = "C:\Program _
Files\feap\ver83\contact\ntrnd\"
        PathAdresses(2) = "C:\Program _
Files\feap\ver83\contact\nts2d\"
        PathAdresses(3) = "C:\Program _
Files\feap\ver83\contact\nts3d\"
        PathAdresses(4) = "C:\Program _
Files\feap\ver83\contact\ptpnd\"
        PathAdresses(5) = "C:\Program _
Files\feap\ver83\contact\tie2d\"
        PathAdresses(6) = "C:\Program Files\feap\ver83\contact\util\"
        PathAdresses(7) = "C:\Program _
Files\feap\ver83\elements\couple3d\"
        PathAdresses(8) = "C:\Program _
Files\feap\ver83\elements\frame\"
        PathAdresses(9) = "C:\Program _
Files\feap\ver83\elements\material\finite\"
        PathAdresses(10) = "C:\Program _
Files\feap\ver83\elements\material\small\"
        PathAdresses(11) = "C:\Program _
Files\feap\ver83\elements\shells\"
        PathAdresses(12) = "C:\Program _
Files\feap\ver83\elements\solid1d\"
        PathAdresses(13) = "C:\Program _
Files\feap\ver83\elements\solid2d\"
        PathAdresses(14) = "C:\Program _
Files\feap\ver83\elements\solid3d\"
        PathAdresses(15) = "C:\Program _
Files\feap\ver83\elements\thermal\"
        PathAdresses(16) = "C:\Program Files\feap\ver83\plot\"
        PathAdresses(17) = "C:\Program Files\feap\ver83\program\"
        PathAdresses(18) = "C:\Program Files\feap\ver83\user\"
        PathAdresses(19) = "C:\Program Files\feap\ver83\windows\"
        PathAdresses(20) = "C:\Program Files\feap\ver83\window2\"

        For Each Path In PathAdresses
            FileAdresses = Directory.GetFiles(Path)
            For Each FileAdress In FileAdresses

My.Computer.FileSystem.WriteAllText("C:\Users\Εμμανουήλ _
```

```
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeap.txt", FileAdress_
& vbCrLf, True)
            Next
        Next
        FileAdress = "C:\Program _
Files\feap\ver83\elements\material\inmate.f"
        My.Computer.FileSystem.WriteAllText("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeap.txt", FileAdress_
& vbCrLf, True)
        MsgBox("The file <C:\Users\Εμμανουήλ
_Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeap.txt> is ready.")

    End Sub

    Sub FindAdressesOfFeapSubroutinesOrFunctions()

        Dim FilePath, what As String
        Dim FS As FileStream = New FileStream("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeap.txt", _
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)

        FilePath = SR.ReadLine
        Do While FilePath <> DBNull.Value.ToString
            what = FunctionOrSubroutine(FilePath)
            If what = "function" Then

My.Computer.FileSystem.WriteAllText("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeapFunctions.txt", _
FilePath & vbCrLf, True)
            ElseIf what = "subroutine" Then

My.Computer.FileSystem.WriteAllText("C:\Users\Εμμανουήλ _
Ζέρ\Documents\διπλωματική\αρχεία txt\AdressesOfFeapSubroutines.txt",_
FilePath & vbCrLf, True)
            End If
            FilePath = SR.ReadLine
        Loop
        SR.Close()
        FS.Close()

    End Sub

    Function FunctionOrSubroutine(ByVal FilePath As String) As String

        Dim FileLine As String
        Dim FS As FileStream = New FileStream(FilePath, _
FileMode.Open)
        Dim SR As StreamReader = New StreamReader(FS)
        Dim CountBlankContinuousLines As Integer

        Do
            CountBlankContinuousLines = 0
            FileLine = DBNull.Value.ToString
            Do While CountBlankContinuousLines < 30 And FileLine = _
DBNull.Value.ToString
                FileLine = SR.ReadLine
                CountBlankContinuousLines = _
CountBlankContinuousLines + 1
            Loop
            If FileLine <> DBNull.Value.ToString Then
```

```
                CountBlankContinuousLines = _
CountBlankContinuousLines - 1
                If FileLine.Contains("subroutine") = True And _
                    FileLine.Chars(0) <> "c" Then
                    Return ("subroutine")
                ElseIf FileLine.Contains("function") = True And _
                       FileLine.Chars(0) <> "c" Then
                    Return ("function")
                End If
            End If
        Loop Until CountBlankContinuousLines = 30
        SR.Close()
        FS.Close()
    End Function

End Module
```

<u>Παράρτημα ΙΙΙ:</u>
Μετατροπές και αντιστοιχίες

III.1. Μετατροπές και αντιστοιχίες

α/α	κώδικας	ισοδύναμος
1	IF (CONDITION(1)) THEN COMMANDS(1) END IF IF (CONDITION(2)) THEN COMMANDS(2) END IF ... IF (CONDITION(N)) THEN COMMANDS(N) END IF	DO I=1,N IF (CONDITION(I)) THEN COMMANDS(I) END IF END DO
2	IF (CONDITION(1)) THEN COMMANDS1(1) ELSE COMMANDS2(1) END IF IF (CONDITION(2)) THEN COMMANDS1(2) ELSE COMMANDS2(2) END IF ... IF (CONDITION(N)) THEN COMMANDS1(N) ELSE COMMANDS2(N) END IF	DO I=1,N IF (CONDITION(I)) THEN COMMANDS1(I) ELSE COMMANDS2(I) END IF END DO
3	IF (CONDITION(1)) THEN	I = 1

	COMMANDS(1) ELSE IF (CONDITION(2)) THEN COMMANDS(2) ... ELSE IF (CONDITION(N)) THEN COMMANDS(N) ELSE COMMANDS_ELSE END IF	DO WHILE (I.LE.N.AND.(.NOT.CONDITION(I))) I = I+1 END DO IF (I.LE.N) THEN COMMANDS(I) ELSE COMMANDS_ELSE END IF
4	IF (CONDITION(1)) THEN COMMANDS1(1) IF (CONDITION(2)) THEN COMMANDS1(2) IF (CONDITION(3)) THEN COMMANDS1(3) ... IF (CONDITION(N)) THEN COMMANDS1(N) COMMANDS2(N) END IF ... COMMANDS2(3) END IF COMMANDS2(2) END IF COMMANDS2(1) END IF	I = 1 DO WHILE (I.LE.N.AND.CONDITION(I)) COMMANDS1(I) I = I+1 END DO I = I-1 DO J=I,1,-1 COMMANDS2(J) END DO
5	IF (CONDITION(1)) THEN COMMANDS1(1) IF (CONDITION(2)) THEN COMMANDS1(2) IF (CONDITION(3)) THEN	I = 1 DO WHILE (I.LE.N.AND.CONDITION(I)) COMMANDS1(I) I = I+1 END DO

	COMMANDS1(3) ... IF (CONDITION(N)) THEN COMMANDS1(N) COMMANDS2(N) ELSE COMMANDS3(N) END IF ... COMMANDS2(3) ELSE COMANDS3(3) END IF COMMANDS2(2) ELSE COMMANDS3(2) END IF COMMANDS2(1) ELSE COMMANDS3(1) END IF	COMMANDS3(I) I = I-1 DO J=I,1,-1 COMMANDS2(J) END DO
6	DO I=K(1),L(1),M(1) COMMANDS(1) END DO DO I=K(2),L(2),M(2) COMMANDS(2) END DO ... DO I=K(N),L(N),M(N) COMMANDS(N) END DO	DO I=1,N DO J=K(I),L(I),M(I) COMMANDS(I) END DO END DO
7	DO I1=1,K(1)	NPROD = 1

	DO I2=1,K(2) DO I3=1,K(3) ... DO IN=1,K(N) COMMANDS.....[I1,I2,...,IN] END DO ... END DO END DO END DO	DO I=1,N NPROD = NPROD*K(I) END DO DO I=1,NPROD PROD=NPROD J = 1 IY = I FLAG = .TRUE. DO WHILE (FLAG) PROD = NPROD/K(J) IP = IY/PROD IY = IY-IY/PROD*PROD IF (IY.EQ.0) THEN D(J) = IP DO K=J+1,N D(K) = N(K) END DO FLAG = .FALSE. ELSE D(J) = IP+1 J = J+1 END IF END DO COMMANDS.....[D(1),D(2),...,D(N)] END DO
8	DO I1=K(1),L(1),M(1) DO I2=K(2),L(2),M(2) DO I3=K(3),L(3),M(3) ... DO IN=K(N),L(N),M(N) COMMANDS.....[I1,I2,...,IN] END DO	NPROD = 1 DO I=1,N NPROD = NPROD*((L(I)-K(I))/M(I)+1) END DO DO I=1,NPROD PROD = NPROD J = 1

	... END DO END DO END DO	IY = I FLAG = .TRUE. DO WHILE (FLAG) PROD = PROD/((L(J)-K(J))/M(J)+1) IP = IY/PROD IY = IY-IY/PROD*PROD IF (IY.EQ.0) THEN D1 = IP D(J) = K(J)+(D1-1)*M(J) DO P=J+1,N D1 = (L(P)-K(P))/M(P)+1 D(P) = K(P)+(D1-1)*M(P) END DO FLAG = .FALSE. ELSE D1 = IP+1 D(J) = K(J)+(D1-1)*M(J) J = J+1 END IF END DO COMMANDS.......[D(1),D(2),...,D(N)] END DO
9	DO I1=K(1),L(1),M(1) COMMANDS1(1) ....[I1] DO I2=K(2),L(2),M(2) COMMANDS1(2) ....[I1,I2] DO I3=K(3),L(3),M(3) COMMANDS1(3) ....[I1,I2,I3] ... COMMANDS1(N-1) ....[I1,I2,...,IN-1] DO IN=K(N),L(N),M(N) COMMANDS1(N).......[I1,I2,...,IN]	NPROD = 1 DO I=1,N NPROD = NPROD*((L(I)-K(I))/M(I)+1) END DO DO I=1,NPROD PROD = NPROD J = 1 IY = I FLAG = .TRUE. DO WHILE (FLAG)

```
        COMMANDS2(N).......[I1,I2,...,IN]
      END DO
      COMMANDS2(N-1) .......[I1,I2,...,IN-1]
    ...
      COMMANDS2(3) .......[I1,I2,I3]
    END DO
    COMMANDS2(2) .......[I1,I2]
  END DO
  COMMANDS2(1) .......[I1]
END DO
```

```
    PROD = PROD/((L(J)-K(J))/M(J)+1)
    IP = IY/PROD
    IY = IY-IY/PROD*PROD
    IF (IY.EQ.0) THEN
      D1 = IP
      D(J) = K(J)+(D1-1)*M(J)
      DO P=J+1,N
        D1 = (L(P)-K(P))/M(P)+1
        D(P) = K(P)+(D1-1)*M(P)
      END DO
      FLAG = .FALSE.
    ELSE
      D1 = IP+1
      D(J) = K(J)+(D1-1)*M(J)
      J = J+1
    END IF
  END DO
  PROD = NPROD
  DO P=1,N-1
    PROD = PROD/((L(P)-K(P))/M(P)+1)
    IF (I-I/PROD*PROD.EQ.0) THEN
      COMMANDS1(P).......[D(1),D(2),...,D(P)]
    END IF
  END DO
  COMMANDS1(N).......(D(1),D(2),...,D(N))
  COMMANDS2(N).......(D(1),D(2),...,D(N))
  PROD = 1
  DO P=N-1,1,-1
    PROD = PROD*((L(P)-K(P))/M(P)+1)
    IF (I-I/PROD*PROD.EQ.0) THEN
      COMMANDS2(P).......[D(1),D(2),...D(P)]
    END IF
```

		END DO END DO
10	DO WHILE (CONDITION(1)) DO WHILE (CONDITION(2)) DO WHILE (CONDITION(3)) ... DO WHILE (CONDITION(N)) COMMANDS END DO ... END DO END DO END DO	I = 1 DO WHILE (I.GE.1.AND.I.LE.N) IF (CONDITION(I)) THEN I = I+1 ELSE I = I-1 END IF END DO DO WHILE (I.NE.0) COMMANDS I = I-1 DO WHILE (I.GE.1.AND.I.LE.N) IF (CONDITION(I)) THEN I = I+1 ELSE I = I-1 END IF END DO END DO
11	DO WHILE (CONDITION(1)) COMMANDS1(1) DO WHILE (CONDITION(2)) COMMANDS1(2) DO WHILE (CONDITION(3)) COMMANDS1(3) ... DO WHILE (CONDITION(N)) COMMANDS1(N) COMMANDS2(N) END DO	I = 1 DO WHILE (I.GE.1.AND.I.LE.N) IF (CONDITION(I)) THEN COMMANDS1(I) I = I+1 ELSE I = I-1 COMMANDS2(I) END IF END DO DO WHILE (I.NE.0)

	`` ... COMMANDS2(3) END DO COMMANDS2(2) END DO COMMANDS2(1) END DO ```	``` COMMANDS1(N) COMMANDS2(N) I = I-1 DO WHILE (I.GE.1.AND.I.LE.N) IF (CONDITION(I)) THEN COMMANDS1(I) I = I+1 ELSE I = I-1 COMMANDS2(I) END IF END DO END DO ```

<u>Παράρτημα IV:</u>
Αρχεία εισόδου δεδομένων

IV.1. <u>Πρόβλημα 1:</u> Τεταρτοκύκλιο δίσκου με σημειακό φορτίο σε γραμμική ελαστική ανάλυση

Το αρχείο εισόδου δεδομένων που χρησιμοποιήσαμε για την ανάλυση είναι το ακόλουθο:

```
FEAP
0 0 0 2 2 4

NOPRint

PARAmeter
  m = 20
  n = 20
                  ! End of parameters
BLOCk 1
  CARTesian,m,n,1,1,1
    1  0.0  0.0
    2  0.5  0.0
    3  0.4  0.4
    4  0.0  0.5
                  ! Blank termination record

PARAmeters
  s = sind(22.5)
  c = cosd(22.5)
                  ! Blank termination record
BLOCk 2
  CARTesian,n,n
    1 0.5   0.0
    2 1.0   0.0
    3 0.701 0.701
    4 0.4   0.4
    6 c     s
                  ! Blank termination record
BLOCk 3
  CARTesian,n,n
    1 0.4   0.4
    2 0.701 0.701
    3 0.0   1.0
    4 0.0   0.5
    6 s     c
                ! Blank termination record
EBOUndary       ! Edge boundary restraints
  1 0.0 1 0
  2 0.0 0 1
                ! Blank termination record
CFORce          ! Coordinate specified forces
  NODE 0.0 1.0  0.0 -5.0
                ! Blank termination record
MATErial,1
  SOLId
  ELAStic ISOTropic 10000 0.25
  DENSity data      0.10
  QUADrature data 2  2
                ! Blank termination record
```

```
END
TIE              ! Tie nodes with same coordinates.

BATCh
  TANGent,,1
  DISPlacement,,1,100
  STRESS,,1,20
  REACtion,,1,100
  PLOT,MESH
  PLOT,LOAD,,-1
  PLOT,CONT,2
END               ! End of batch execution

INTEractive

STOP
```

IV.2. <u>Πρόβλημα 2:</u> Πλάκα με οπή τεταρτοκυκλίου με κατανεμημένα φορτία στις ακραίες ακμές σε γραμμική ελαστική ανάλυση

Παρουσιάζουμε το αρχείο εισόδου δεδομένων που χρησιμοποιήσαμε για την επίλυση του προβλήματος:

```
FEAP
  0  0  0  2  2  4

PARAmeters     ! μονάδες
  nu = 0.3
  e  = 210000. ! MPa
  y  = 360.    ! MPa

MATErial 1
  solid
    elastic isotropic e nu

PARAmeters       ! μονάδες
  r  = 4         ! m
  h  = 20        ! m
  f  = 0.45
  d  = 0.4
  m  = 5         ! στοιχεία
  n  = 2*m       ! στοιχεία
  pi = 4*atan(1)

SNODes
  1 0 0
  2 r 0
  3 r*sind(45) r*cosd(45)
  4 0 r
  5 h 0
  6 h h
  7 0 h
  8 f*h 0
```

```
  9 0   f*h
 10 d*h d*h

SIDE
  polar 2 3 1
  polar 3 4 1
  cart  2 5 8
  cart  3 6 10
  cart  4 7 9

BLENd
  surf 10 10 0 0 1
   2 5 6 3

BLENd
  surf 10 10 0 0 1
   3 6 7 4

EBOUn
 1 0 1 0
 2 0 0 1

LOAD
  CSURface ! users manual σελίδα 62
    LINEar
    1 h h 100.
    2 0 h 100.
LOAD END

LOAD
  CSURface ! users manual σελίδα 62
    LINEar
    1 h 0 100.
    2 h h 100.
LOAD END

END ! MESH
TIE

BATCh
  CHECk
  TANG,,1
  DISPlacements,all
  FORCe,all
  REACtions,all
  STREss,all
END

INTEractive
STOP
```

IV.3. <u>Πρόβλημα 3:</u> Προσομοίωση εδάφους και εύκαμπτου πέδιλου εδραζόμενου επ' αυτού σε γραμμική ελαστική ανάλυση

Παρουσιάζουμε το αρχείο εισόδου δεδομένων που χρειάζεται να δώσουμε στο πρόγραμμα FEAP ώστε να γίνει η ανάλυση:

```
FEAP
  0 0 2 2 2 4

PARAmeters !Στοιχεία υλικών.
  E1 = 25.0 !MPa
  P1 = 0.3
  E2 = 30.0 !MPa
  P2 = 0.4

MATErial 1
  SOLId
    ELAStic ISOTropic E1 P1

MATErial 2
  SOLId
    ELAStic ISOTropic E2 P2

PARAmeters !Στοιχεία γεωμετρίας.
  R  = 0.5     !m
  H1 = 6.0     !m
  H2 = 10.0    !m
  H  = H1+H2   !m
  L  = 50*R    !m

SNODes
   1 0     0
   2 L     0
   3 L     H2
   4 0     H2
   5 0     H
   6 L     H
   7 L/2-R H
   8 L/2+R H
   9 0     H1/2+H2
  10 L     H1/2+H2
  11 L/2   H2
  12 0     H2/2
  13 L     H2/2
  14 L/2   0

SIDE
  CARTesian 1 4 12
  CARTesian 4 5 9
  CARTesian 1 2 14
  CARTesian 2 3 13
  CARTesian 3 6 10
  CARTesian 4 3 11
  CARTesian 5 6 7 8
```

```
PARAmeters ! Στοιχεία διακριτοποίησης.
  NL  = 60 !στοιχεία
  NH1 = 30 !στοιχεία
  NH2 = 30 !στοιχεία

BLENd
  SURFace 50 20 0 0 1
    1 2 3 4

BLENd
  SURFace 50 12 0 0 2
    4 3 6 5

EBOUndary
  2 0 1 1

PARAmeters !Στοιχεία φόρτισης.
  P = -10 !MPa

LOAD
  CSURface ! users manual σελίδα 62
    LINEar
    1 L/2+R H P
    2 L/2-R H P
LOAD END

END !mesh

TIE

BATCh
  CHECk
  TANGent
  FORM
  SOLVe
  DISPlacement,all
  STREss,all
  REACtions,all
END

INTEractive

STOP
```

Βιβλιογραφία

Bathe, Klaus- Jürgen. 1996. *Finite element procedures.* New Jersey : Prentice- Hall, Inc., 1996.

Hillman, Jeff. stackoverflow. [Ηλεκτρονικό] http://stackoverflow.com/users/3950/jeff-hillman.

Nyhoff, Larry R. και Leestma, Sanford C. 1996. *Fortran 90 for engineers & scientists.* s.l. : Prentice Hall, 1996.

Petroutsos, Evangelos. 2010. *Mastering Microsoft visual basic 2010.* Canada : Sybex, 2010.

Spiliopoulos, Konstantinos V. και Panagiotou, Konstantinos D. 2012. A direct method to predict steady states of elastoplastic structures. *ELSEVIER.* 13 March 2012, σ. 13.

Zienkiewicz, C. O. και Taylor, R. L. 2000. *The finite element method.* California : BUTTERWORTH HEINEMANN, 2000.

Καββαδάς, Μιχάλης Ι. 2009. *Στοιχεία εδαφομηχανικής.* Αθήνα : Συμεών, 2009.

Παναγιώτου, Κωνσταντίνος. 2007. *Διπλωματική εργασία: Αλγόριθμοι ελαστοπλαστικής ανάλυσης & εφαρμογές τους.* Αθήνα : s.n., 2007.

Παπαδρακάκης, Μανόλης. 2001. *Ανάλυση φορέων με τη μέθοδο των πεπερασμένων στοιχείων.* Αθήνα : Παπασωτηρίου, 2001.