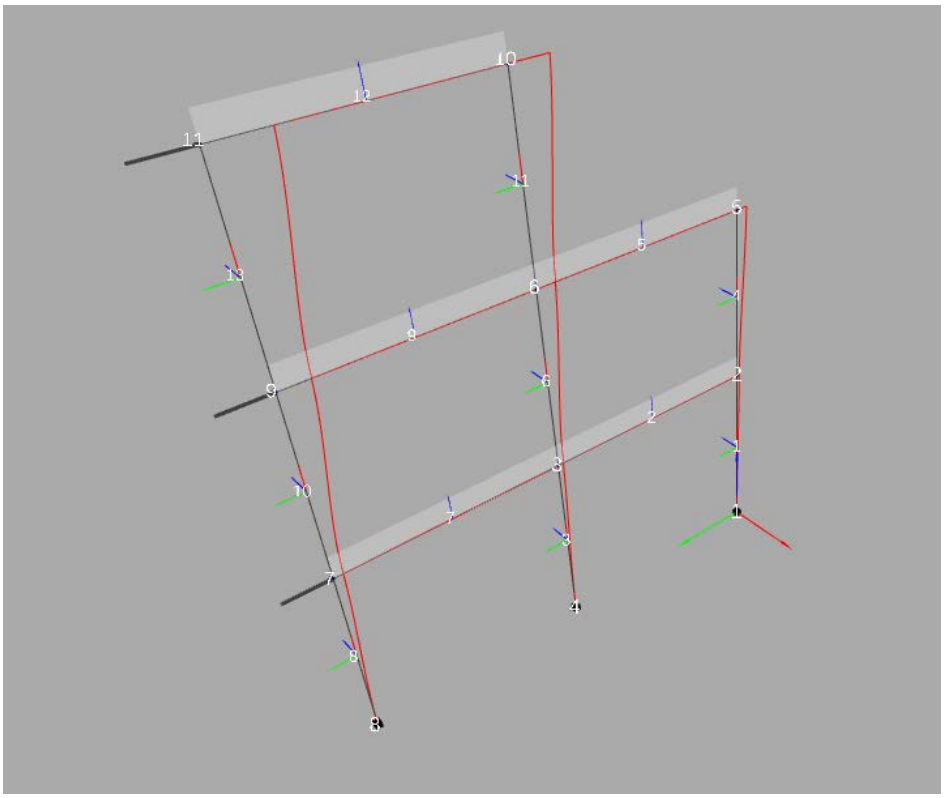
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Πολιτικών Μηχανικών

Ανάπτυξη εφαρμογής στατικής ανάλυσης φορέων στο διαδίκτυο



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παντελεήμων Α. Τσάκαλης

Επιβλέπων: Νικόλαος Λαγαρός

Αθήνα, Οκτώβριος 2019

ΕΜΚ ΔΕ 2019/23

Τσάκαλης Π. (2019).
Ανάπτυξη εφαρμογής στατικής ανάλυσης φορέων στο διαδίκτυο
Διπλωματική Εργασία ΕΜΚ ΔΕ 2019/23
, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα.

Tsakalis P. (2019).
Development of structural analysis application on the web
Diploma Thesis EMK ΔΕ 2019/23
Institute of Steel Structures, National Technical University of Athens, Greece
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΕΜΚ ΔΕ 2018/23

Ανάπτυξη εφαρμογής στατικής ανάλυσης φορέων στο διαδίκτυο

Τσάκαλης Π. (Επιβλέπων: Λαγαρός Ν.)

Περίληψη

Αναπτύχθηκε εφαρμογή στατικής ανάλυσης προκειμένου να εξυπηρετηθούν οι ανάγκες φοιτητών σε επίλυση βασικών φορέων σε στατικά φορτία. Προκειμένου να δοθεί πλήρης ευελιξία στους χρήστες η εφαρμογή αναπτύχθηκε στο πλαίσια διαδικτυακής εφαρμογής αίροντας την ανάγκη για διανομή κάποιου προγράμματος προς εγκατάσταση. Το πρόγραμμα διαθέτει εύχρηστο και απλοποιημένο GUI το οποίο επιτρέπει τον πλήρη ορισμό του μοντέλου. Η ανάλυση που εκτελείται είναι γραμμική ανάλυση βασισμένη στη μέθοδο άμεσης δυσκαμψίας. Μπορούν να αντιμετωπιστούν ελευθερώσεις και φορτία εντός μέλους (σημειακά και κατανεμημένα). Στο χρήστη επιστρέφονται αποτελέσματα σε μορφή πινάκων και διαγραμμάτων. Ο χρήστης θα έχει τη δυνατότητα να κάνει λήψη των αποτελεσμάτων σε μορφή αρχείων κειμένου.

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
FACULTY OF CIVIL ENGINEERING

DIPLOMA THESIS
EMK ΔΕ 2018/23

Development of structural analysis application on the web

Tsakalis P. (supervised by Lagaros N.)

Abstract

The aim of this diploma thesis was the development of a structural analysis application for the purposes of assisting students to various projects. In order to give flexibility to the users the application was developed on the cloud removing the need for distribution of files for installation. The application is complemented with a user friendly GUI through which a user can define the desired model. The analysis is based on the direct stiffness method. The application can handle end member releases and loads within the member(point or distributed). The user gets back the results of the analysis in the form of tables and diagrams. The user will be able to download text files of the results.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένειά μου, που στάθηκε δίπλα μου καθ' όλη τη διάρκεια των σπουδών μου. Τον κ. Νικόλαο Λαγαρό για την πολύτιμη βοήθειά του κατά την επίβλεψη της εργασίας και τέλος, τους φίλους μου που με τη στήριξή τους συνέβαλαν σημαντικά στην πορεία μου τα τελευταία χρόνια.

This research has been supported by the OptArch project: "Optimization Driven Architectural Design of Structures" (No: 689983) belonging to the Marie Skłodowska-Curie Actions (MSCA) Research and Innovation Staff Exchange (RISE) H2020-MSCA-RISE-2015.

Πίνακας περιεχομένων

1 Εισαγωγή

1.1 Γενικά Στοιχεία

Θέμα της παρούσας διπλωματικής είναι η ανάπτυξη εφαρμογής ανάλυσης κατασκευών με γραφικό περιβάλλον διεπαφής στο νέφος(cloud). Αυτό σημαίνει ότι η διαχείριση τον δεδομένων και οι υπολογισμοί εκτελούνται σε έναν απομακρυσμένο server. Ο χρήστης μπορεί να χρησιμοποιεί το πρόγραμμα από οποιαδήποτε συσκευή μέσω browser με μόνη απαίτηση σύνδεση στο διαδίκτυο. Στο υπόλοιπο της διπλωματικής οι διεργασίες που γίνονται στον server θα αποκαλούνται server-side operations ενώ στον χρήστη(πελάτη) client-side operations.

Υπολογιστικό νέφος(Cloud Computing)

Το μοντέλο υπολογιστικού νέφους που υλοποιήθηκε ειναι Software-as-a-Service (SaaS) όπου αντί να εγκατασταθεί λογισμικό στο μηχάνημα και στον υπολογιστή του πελάτη επιβαρύνοντας τον με τακτικές επιδιορθώσεις, συχνές εκδόσεις κτλ., εφαρμογές όπως το Word, CRM (Διαχείριση Σχέσεων Πελατών), ERP (Enterprise Resource) Προγραμματισμός) διατίθενται (φιλοξενούνται) μέσω του διαδικτύου για την κατανάλωση του τελικού χρήστη.

Οι λόγοι που επιλέχθηκε η ανάπτυξη της εφαρμογής στο νέφος είναι οι εξής:

- **Κόστος απόδοσης** - Το Cloud computing είναι πιθανότατα η πιο αποδοτική μέθοδος για τη χρήση, διατήρηση και αναβάθμιση λογισμικού.
- **Αυτόματη ενσωμάτωση λογισμικού** -Στο σύννεφο, η ενσωμάτωση λογισμικού γίνεται αυτόματα. Δεν απαιτείται εγκατάσταση.
- **Εύκολη χρήση**- Μόλις οι χρήστες εγγραφούν στο Υπολογιστικό Νέφος, μπορούν να έχουν πρόσβαση στις πληροφορίες από οπουδήποτε, όπου υπάρχει σύνδεση στο Διαδίκτυο.
- **Γρήγορη ανάπτυξη** - Οταν ο χρήστης αποφασίζει να χρησιμοποιήσει το Cloud computing, το σύστημα είναι πλήρως λειτουργικό σε μερικά λεπτά.
- **Χαμηλές απαιτήσεις σε υπολογιστική ισχύ του χρήστη** - Εφόσον οι υπολογισμοί εκτελούνται στον διακομιστή(server) δεν απαιτείται ιδιαίτερη υπολογιστική ισχύ από τον χρήστη.

1.2 Σκοπός

Σκοπός της παρούσας διπλωματικής εργασίας είναι η παροχή ενός προγράμματος ανάλυσης στους φοιτητές για το οποίο δεν θα χρειάζονται ούτε υψηλού επιπέδου υπολογιστές ούτε άδειες χρήσης. Επίσης δίνεται το έναυσμα σε άλλους φοιτητές να ασχοληθουν στις

διπλωματικές τους με τον προγραμματισμο προκειμένου να επεκτείνουν τις δυνατότητες αυτού του προγράμματος.

1.3 Οργάνωση περιεχομένων

Στο δεύτερο κεφάλαιο γίνεται η περιγραφή του προγράμματος. Αναλύεται η βασική δομή και εργαλεία που χρησιμοποιήθηκαν. Στο τρίτο κεφάλαιο αναφέρεται αναλυτικά η δομή και σχεδιασμός του front end(σχεδιαστικό περιβάλλον και GUI), ενώ στο τέταρτο κεφάλαιο αναπτύσσεται η δομή και ο κώδικας του back end. Στο πέμπτο κεφάλαιο γίνεται παρουσίαση του αλγορίθμου επίλυσης και του κωδικα υλοποιησης. Τέλος, στο έκτο κεφάλαιο γίνεται υποδειγματική επίλυση φορέα.

2 Περιγραφή του προγράμματος

2.1 Δυνατότητες

Ο χρήστης μπορεί να δημιουργήσει το δικό του μοντέλο μέσω του γραφικού περιβάλλοντος ή να εισάγει αρχείο μορφής DXF και στη συνέχεια να το επεξεργαστεί. Το πρόγραμμα δίνει τη δυνατότητα στατικής γραμμικής ανάλυσης σε τρισδιάστατους φορείς. Υποστηρίζει ελευθερώσεις άκρων μέλους και φορτίσεις κατά μήκος μέλους. Μετά την ανάλυση το πρόγραμμα επιστρέφει σε μορφή διαγραμμάτων εντατικά μεγέθη και μετακινήσεις κάθε μέλους καθώς και τις αντιδράσεις του φορέα. Υπάρχει η δυνατότητα αποθήκευσης των αποτελεσμάτων σε μορφή csv αρχείων. Το μοντέλο του χρήστη αποθηκεύεται στη βάση δεδομένων για μελλοντική χρήση.

2.2 Δομή προγράμματος

Τα κύρια τμήματα του προγράμματος είναι τα εξής:

1. Front End(Client Side)
2. Back End(Server Side)
3. Post-processing(Client and Server Side)

2.2.1 Front End(Client Side)

Αποτελείται από το γραφικό περιβάλλον χρήστη(GUI) και από το γραφικό περιβάλλον απεικόνισης του μοντέλου.

Για το Front End(Client Side) χρησιμοποιήθηκαν οι γλώσσες HTML5 και JavaScript με τα εξής πακέτα:

1. Jquery - για διάφορες συναρτήσεις client side
2. Threejs - βασισμένο στο webGL API για σχεδιασμό 3D γραφικών.

2.2.2 Back End(Server Side)

Αποτελείται από τις εργασίες διαχείρισης του Front End, από τη βάση δεδομένων και απο του υπολογισμούς επίλυσης(υπολογιστικός πυρήνας).

Για το Back End(Server Side) χρησιμοποιήθηκε η γλώσσα Python με τα εξής πακέτα:

1. Flask - web microframework και οι επεκτάσεις του(Flask-Security, Admin)
2. Sqlalchemy - SQL toolkit και object-relational mapper για την διεπαφή με τη βάση δεδομένων
3. Numpy- προσθέτει υποστήριξη πινάκων στην Python
4. Pandas - χρησιμοποιήθηκε εκτενώς για την ευέλικτη διαχείριση των δεδομένων

2.2.3 Post Processing(Client Side και Server Side)

Αποτελείται από τις εργασίες συγκέντρωσης και παρουσίασης των αποτελεσμάτων. Αφορά διεργασίες που εκτελούνται και στη πλευρά του χρήστη(διαγράμματα, πίνακες) αλλά και στον server.

Για το Post-processing χρησιμοποιήθηκε η γλώσσα Python με τα εξής πακέτα:

- Plotly - για τα διαγράμματα τον αποτελεσμάτων, δημιουργεί αρχεία σε HTML και JavaScript.

3 Front End

Το κομμάτι του Front End αφορά τις διεργασίες που εκτελούνται στη πλευρά του χρήστη(Client Side) και πλαισιώνουν την εμπειρία χρήσης του προγράμματος.

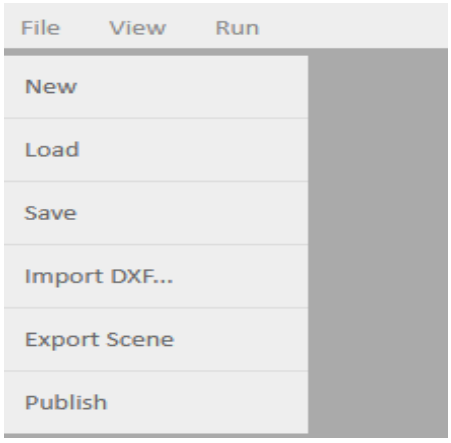
Στο συγκεκριμένο πρόγραμμα το Front End αποτελείται από δύο κύρια τμήματα:

1. Περιβάλλον Διεπαφής Χρήστη(GUI)
2. Γραφικό Περιβάλλον Σχεδιασμού

3.1 Περιβάλλον Διεπαφής Χρήστη(GUI)

Κύρια απαίτηση κατα την εκπόνηση της διπλωματικής ήταν η δημιουργία ενός εύχρηστου και οπτικά ικανοποιητικού προγράμματος. Αυτό επιτυγχάνεται με την ανάπτυξη ενός σωστού και έξυπνου γραφικού περιβάλλοντος διεπαφής χρήστη, στο οποίο στη συνέχεια θα αναφερόμαστε ως GUI. Μελετήθηκαν προγράμματα ανάλυσης, επαγγελματικά και μη, προκειμένου να σχηματιστεί μία εικόνα για τις ανάγκες χρήστη. Μετά από εκτενής έρευνα και πολλές δοκιμές αποφασίστηκε να χρησιμοποιηθεί ένα σχεδιαστικό περιβάλλον ανοικτού κώδικα, το οποίο τροποποιήθηκε εκτενώς προκειμένου να καλυφθούν οι απαιτήσεις.

Οι λειτουργίες που δίνονται στον χρήστη μέσω του GUI είναι οι εξής:



- **Import DXF** - Εισαγωγή αρχείου μορφής DXF

- **New** - εκκαθάριση όλων των δεδομένων και στη πλευρά του χρήστη αλλά και στη βάση δεδομένων
- **Ορισμός στοιχείων** - μέσω των φορμών εισαγωγής ο χρήστης ορίζει κόμβους, μέλη, διατομές και φορτία
- **Save** - το μοντέλο που έχει δημιουργήσει ο χρήστης αποθηκεύεται στη βάση δεδομένων
- **Load from DB** - φορτώνεται από τη βάση δεδομένων και σχέδιαζεται μοντέλο που ήδη έχει αποθηκεύσει ο χρήστης
- **Run** - γίνεται έλεγχος του μοντέλου και εκτελείται η ανάλυση.
- **Results** - εμφάνιση των αποτελεσμάτων

Κάθε μία από αυτές τις λειτουργίες εκτός από τον ορισμό στοιχείων, εκτελούνται στον διακομιστή(Server). Η επικοινωνία μεταξύ διακομιστή-πελάτη γίνεται μέσω AJAX calls(Asynchronous Javascript and XML) τα οποία δεν επηρεάζουν την εμπειρία του χρήστη και εκτελούνται στο παρασκήνιο. Κάθε AJAX call εκτελείται στο δικό του endpoint στο οποίο αντιστοιχεί η εκάστοτε συνάρτηση, οπου για λόγους ευκολίας έχουν ονομαστεί σύμφωνα με τη λειτουργία που εκτελείται.

Η διαδικασίες των λειτουργιών που εκτελούνται στον server αναπτύσσονται αναλυτικά στο κεφάλαιο Back End.

3.1.1 Load

Γίνεται ανάκτηση αποθηκευμένου μοντέλου για περαιτέρω επεξεργασία. Μέσω AJAX Call στο url /load αποστέλλονται από τον διακομιστή οι κόμβοι, τα μέλη, τα φορτία και οι διατομές που είχε ορίσει ο χρήστης σε προηγούμενη συνεδρία. Στη συνέχεια καλούνται οι συναρτήσεις drawNodes, drawElements, drawPointLoads ο κώδικας των οποίων βρίσκεται στις αντίστοιχες λειτουργίες

```
$.ajax({
     type: "POST",
     url: "/load",
     data: {},
     dataType: 'text',
     success: function (e) {
         var data = e.split("|");
         var nodes = JSON.parse(data[0]);
         var elements = JSON.parse(data[1]);
         var pointLoads = JSON.parse(data[2]);
         drawNodes( editor, nodes.data );
         drawElements( editor, elements.data );
         drawPointLoads( editor, pointLoads.data );
     },
     error: function(xhr, status, error) {
         console.log(xhr, status, error);
     }
 });
```

AJAX Call για φόρτωση του μοντέλου (Menubar.File.js)

3.1.2 Save

Ο χρήστης μπορεί να αποθηκεύσει το μοντέλο στη βάση δεδομένων προκειμένου να μπορεί να ανακτηθεί στο μέλλον. Η αποθήκευση είναι προϋπόθεση προκειμένου να τρέξει η ανάλυση. Μέσω AJAX Call αποστέλλονται από τον πελάτη οι κόμβοι, τα μέλη, τα φορτία και οι διατομές στον διακομιστή. Η διαχείριση των δεδομένων αναπτύσσεται στη αντίστοιχη λειτουργία του Back End.

```
        data1 = []
        nodes = [];
        elements = [];
        loads = [];
        for (i=0; i<editor.scene.children.length; i++){
            obj = editor.scene.children[i]
            if (obj.userData.type == 'element') {
                elements.push( obj.userData );
            } else if (obj.userData.type == 'node') {
                nodes.push( obj.userData );
            } else if (obj.userData.type == 'p_load' || obj.userData.type == 'p_moment')
{
                loads.push( obj.userData );
            }
        }
        data1.push(elements);
        data1.push(nodes);
        data1.push(loads);
        var sections = editor.sections.sections;
        data1.push(sections )
        data1 = JSON.stringify( data1 );
        $.ajax({
            type: "POST",
            url: '/save',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            data: data1,

        });
```

AJAX Call για αποθήκευση του μοντέλου (Menubar.File.js)

3.1.3 Import DXF

Η συγκεκριμένη διαδικασία ξεκινάει από τη πλευρά του χρήστη μέσω ενός AJAX call και ολοκληρώνεται στο back end. Το κομμάτι της διαδικασίας που εκτελείται στο server θα αναλυθεί στο αντίστοιχο κεφάλαιο.

Η εκτέλεση της συγκεκριμένης διαδικασίας χρειάζεται επιβεβαίωση από τον χρήστη διότι αντικαθιστά και τα δεδομένα στη βάση αλλά και το ορισμένο μοντέλο στη πλευρά του χρήστη. Μετά την επιβεβαίωση ζητείται από τον χρήστη να

ανεβάσει το αρχείο μορφής DXF. Το αρχείο πρέπει να αποτελείται από line objects. Σχεδιάζεται στο γραφικό περιβάλλον το εισαγόμενο μοντέλο.

```
editor.clear();

$.ajax({
    type: 'POST',
    timeout: 20000,
    url: '/read DXF',
    data: fileInput.files[0],
    processData: false,
    contentType: 'application/dxf',
    dataType: 'text',
    success: function(e) {
        var nodelm = e.split("|");
        var nodes = JSON.parse(nodelm[0]);
        var elements = JSON.parse(nodelm[1]);
        drawNodes( editor, nodes.data );
        drawElements( editor, elements.data );
    },
    error: function(xhr, status, error) {
        console.log(xhr, status, error);
    }
});
```

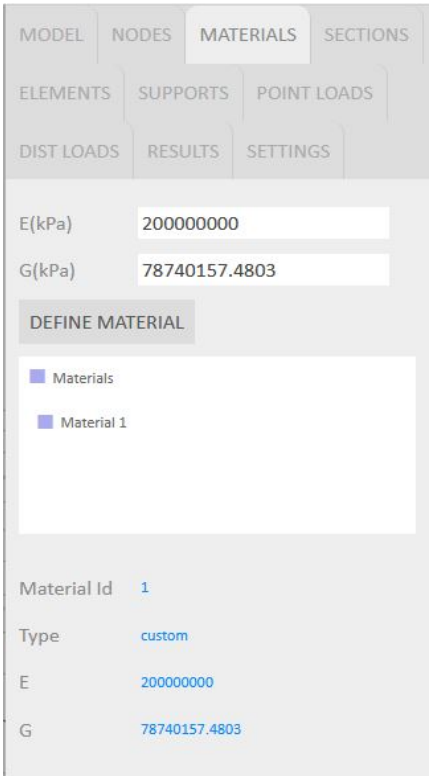
AJAX Call για εισαγωγή αρχείου DXF (Menubar.File.js)

3.1.4 Clear

Η εκτέλεση της συγκεκριμένης διαδικασίας χρειάζεται επιβεβαίωση από τον χρήστη διότι αντικαθιστά και τα δεδομένα στη βάση αλλά και το ορισμένο μοντέλο στη πλευρά του χρήστη.

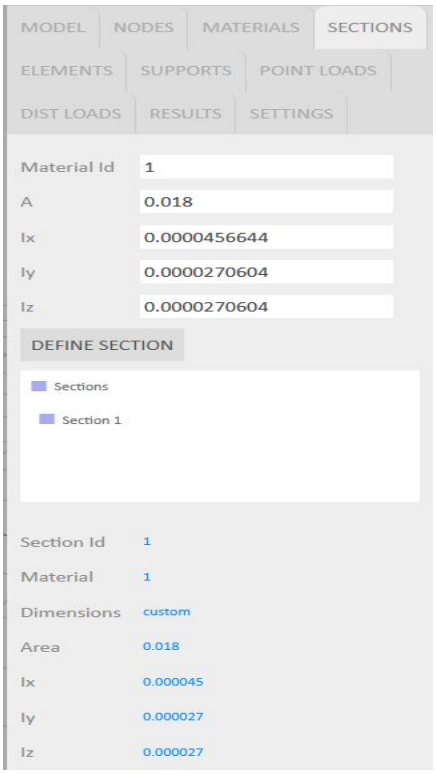
3.1.5 Ορισμός υλικών(Materials)

Γίνεται ο ορισμός υλικού της μέσω λίστας επιλογής ανάμεσα σε τυποποιημένα υλικά.



3.1.6 Ορισμός διατομών(Sections)

Στη ορίζεται η γεωμετρία της διατομής, ανάλογα με το υλικό που επιλέχθηκε και ορίζεται ο αντίστοιχος κωδικός της διατομής.



3.1.7 Ορισμός κόμβων(Nodes)

Απαιτούνται οι συντεταγμένες του κόμβου(x, y, z) και οι βαθμοί ελευθερίας του. Δημιουργείται στο γραφικό περιβάλλον σφαίρα(για ελεύθερο κόμβο) και κύβος(για δεσμευμένο κόμβο). Το αντικείμενο κάθε κόμβου έχει την δομή userData στην οποία ο χρήστης μπορεί να αποθηκεύσει κάθε είδους πληροφορία σε μορφή JSON. Εκεί αποθηκεύονται όλα τα απαραίτητα στοιχεία για την ανάλυση.

3.1.8 Ορισμός μελών(Elements)

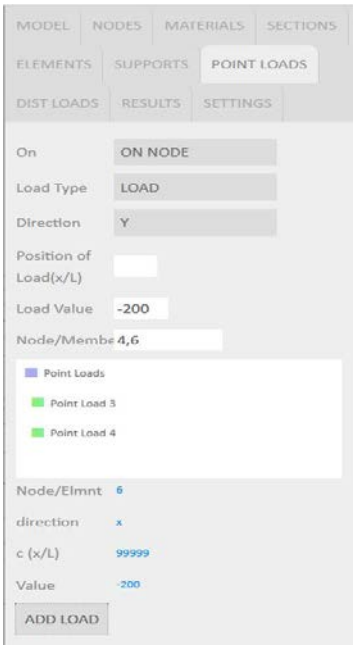
Απαιτείται ο αριθμός της διατομής(έχει οριστεί προηγουμένως) οι κόμβοι αρχής και τέλους. Υπάρχει δυνατότητα επιλογής των κόμβων γραφικά(Pick Nodes). Ο χρήστης επιλέγει τους κόμβους αρχής και τέλους.

3.1.9 Ορισμός σημειακών φορτίων(Point Loads)

Ορίζεται ο κόμβος/μέλος πάνω στον οποίο εφαρμόζεται το φορτίο, η απόσταση από τον κόμβο αρχής σε ποσοστό του μήκους(αν το φορτίο εφαρμοστεί σε κόμβο αφήνετε κενό). Η διεύθυνση του φορτίου καθώς και ο τύπος, δύναμη ή ροπή.

3.1.10 Ορισμός κατανεμημένων φορτίων(Dist Loads)

Ορίζεται ο κόμβος/μέλος πάνω στον οποίο εφαρμόζεται το φορτίο, η απόσταση από τον κόμβο αρχής σε ποσοστό του μήκους(αν το φορτίο εφαρμοστεί σε κόμβο αφήνετε κενό). Η διεύθυνση του φορτίου καθώς και ο τύπος, δύναμη ή ροπή.



3.2 Γραφικό Περιβάλλον Σχεδιασμού

Σκοπος ήταν να παρέχεται στο χρήστη ένα εύχρηστο περιβάλλον σχεδιασμού για τον βασικό ορισμό του μοντέλου. Δίνεται η δυνατότητα εισαγωγής αρχείου DXF, το οποίο μπορεί να επεξεργαστεί στη συνέχεια ο χρήστης.

Λόγω της εξαιρετικής δυσκολίας του να δημιουργηθεί ένα γραφικό περιβάλλον σχεδιασμού από το μηδέν αποφασίστηκε να τροποποιηθεί ένα γραφικό περιβάλλον ανοικτού κώδικα. Συγκεκριμένα επιλέχθηκε ένα demo του πακέτου γραφικών που θα χρησιμοποιούνταν εξαρχής, το THREE.js.

3.2.1 Three.js Editor

Αξίζει να γίνει αναφορά στο λογισμικού ανοικτού κώδικα διότι χωρίς αυτό, η ολοκλήρωση του προγράμματος θα ήταν εξαιρετικά δύσκολη. Το λογισμικό Three.js Editor έχει δημιουργηθεί με σκοπό να παρουσιάσει τις δυνατότητες της βιβλιοθήκης 3D γραφικών, Three.js. Ουσιαστικά πρόκειται για ένα σχεδιαστικό περιβάλλον το οποίο μεταξύ άλλων επιτρέπει τη δημιουργία και επεξεργασία διαφόρων γεωμετρικών σχημάτων.

Το συγκεκριμένο demo: https://threejs.org/editor/ παρείχε έτοιμες πολλές βασικές λειτουργίες όπως undo, redo, δημιουργία και διαγραφή στοιχείων και την πολύ βολική client side βάση δεδομένων.

Αυτό δεν αναιρεί το γεγονός ότι έπρεπε να διαβαστεί προσεκτικά ο πηγαίος κώδικας και να γίνουν πολλές μετατροπές και προσθήκες προκειμένου να ικανοποιηθούν οι απαιτήσεις σχεδιασμού.

3.2.2 Βασική Δομή και Συναρτήσεις

Προκειμένου κάποιος να επεκτείνει τις δυνατότητες του σχεδιαστικού περιβάλλοντος πρέπει να κατανοήσει τη βασική δομή της συγκεκριμένης υλοποίησης.

Το βασικότερο αντικείμενο είναι ο editor. Ουσιαστικά είναι γονέας όλων των πληροφοριών, μεθόδων και συναρτήσεων. Ο ορισμός όλων των υπολοίπων ξεκινάει από την εκτέλεση του της συνάρτησης Editor στο Editor.js.

Βασικη δομή του γραφικού περιβάλλοντος και του GUI είναι τα εξής:

1. Scene
    a. editor.scene
- Είναι ο γονέας οποιουδήποτε στοιχείου σχεδιάζεται.

2. Menubar
3. Sidebar
4. editor.scene

    Αποτελεί τη βάση του Threejs. Είναι ο γονέας οποιουδήποτε στοιχείου. Εκεί τοποθετούνται όλα τα στοιχεία με τα οποία ο χρήστης μπορεί να αλληλεπιδρά.

    Προκειμένου η προσθήκη του στοιχείου να συμπεριληφθεί στο history των undo, redo και για να υπάρχει στην client sided βάση δεδομένων πρέπει να γίνει με τη εξής εντολή:

    editor.execute(new AddObjectCommand)

    Διαφέρει από το editor.sceneHelpers.
5. editor.sceneHelpers

    Ίδιες δυνατότητες με editor.scene. Σε αυτήν τοποθετούνται στοιχεία με τα οποία ο χρήστης δεν χρειάζεται να αλληλεπιδράσει όπως οι άξονες.

3.2.3 Μετατροπές/Προσθήκες

Είναι σημαντικό να αναφερθούν οι αλλαγές και οι προσθήκες που έχουν γίνει στον πηγαίο κώδικα καθώς και οι συναρτήσεις που είναι χρήσιμες για πιθανές επεκτάσεις του προγράμματος.

4 Back End

4.1 Flask Framework

Ο όρος Back End αναφέρεται σε όλες τις διεργασίες που εκτελούνται και σχετίζονται με τη διαχείριση των δεδομένων και την εκτέλεση υπολογισμών. Στο παρόν κεφάλαιο θα αναλυθούν η διαχείριση και αποθήκευση των δεδομένων. Οι υπολογισμοί θα αναλυθούν σε επόμενο κεφάλαιο με πλήρη περιγραφή της διαδικασίας αλλα και του αλγορίθμου υλοποίησης.

Η δρομολόγηση της εφαρμογής και η επικοινωνία με τον πελάτη(client side) υλοποιήθηκε με χρήση του Flask micro-Framework. Η συγκεκριμένη βιβλιοθήκη επιλέχθηκε έναντι άλλων διότι επιτρέπει χωρίς την παροχή υψηλού επιπέδου συναρτήσεις, την άμεση και απλή δρομολόγηση και διαχείριση της εφαρμογής. Περαιτέρω λειτουργίες παρέχονται μέσω την προεκτάσεων που διατίθενται από την κοινότητα.

4.1.1 Διακομιστής (Server)

Όλες οι κύριες διαδικασίες της διαχείρισης μιας εφαρμογής στο ίντερνετ συμβαίνουν στον server. Στη παρούσα εφαρμογή αυτές έχουν υλοποιηθεί μέσω του Flask και βρίσκονται στο αρχείο app.py.

Η δημιουργία μιας εφαρμογής μέσω Flask γίνεται με πολύ απλά βηματα:

Δρομολόγηση

Ορίζονται τα url endpoints και οι διαδικασίες που εκτελούνται σε κάθε ένα από αυτά. Σε μερικά απλά επιστρέφεται στον χρήστη η ιστοσελίδα, σε άλλα γίνεται αμφίδρομη αποστολή δεδομένων από τον server στον client και το αντίστροφο. Η ονομασία των urls λόγω ευκολίας αντιστοιχεί πλήρως στην λειτουργία που εκτελούν.

Url Endpoints

1. ‘/’
   Η αρχική σελίδα της εφαρμογής. Ο χρήστης μπορεί να κάνει login ή να δημιουργήσει λογαριασμό.

```
@app.route('/')
@app.route('/index')
def index():
   return render_template('home.html')
```

Url Endpoint (‘/’) (app.py)

2. ‘/login’
   Η σελίδα στην οποία ανακατευθύνεται ο χρήστης αν επιλέξει να συνδεθεί

```
@app.route('/login', methods=['GET', 'POST'])
def login():
   if request.method == 'POST':
       # Get Form Fields
       username = request.form['username']
       password_candidate = request.form['password']

       # Get user by username
       result =  models.User.query.filter_by(username=username).first()

       if result is not None:
           # Get stored hash
           session['logged_in'] = True
           session['username'] = username
           password = result.password

           # Compare Passwords
           if sha256_crypt.verify(password_candidate, password):
               # Passed
               session['logged_in'] = True
               session['username'] = username
               flash('You are now logged in', 'success')
               return redirect(url_for('editor'))
           else:
               error = 'Invalid login'
               return render_template('login.html', error=error)
           # Close connection
           cursor.close()
       else:
           error = 'Username not found'
           return render_template('login.html', error=error)

   return render_template('login.html')
```

Url Endpoint (‘/register’) (app.py)

3. ‘/register’
   Η σελίδα στην οποία ανακατευθύνεται ο χρήστης αν επιλέξει να δημιουργήσει λογαριασμό

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm(request.form)
    if request.method == 'POST' and form.validate():
        name = form.name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))

        usr = models.User(username=username, password=password)
        db.session.add(usr)
        db.session.commit()

        flash('You are now registered and can log in', 'success')

        return redirect(url_for('login'))
    return render_template('register.html', form=form)
```

Url Endpoint (‘/register’) (app.py)

4. ‘/editor’
   Με την επιτυχή σύνδεση ο χρήστης ανακατευθύνεται στον editor, το κύριο κομμάτι της εφαρμογης

```
@app.route('/editor')
def editor():
    if 'logged_in' in session:
        return render_template('editor.html')
    else:
        flash('Unauthorized, Please login', 'danger')
        return redirect(url_for('login'))
```

Url Endpoint (‘/editor’) (app.py)

5. ‘/save’
   Πραγματοποιείται στο παρασκήνιο μέσω AJAX Call. Τα δεδομένα του μοντέλου αποθηκεύονται στη βάση δεδομένων όποτε θελήσει ο χρήστης

```
@app.route('/save', methods=["GET", "POST"])
def save():
    if request.method == 'POST':
        data = request.get_json()
        user_id = session['username']
        elements, nodes, point_loads, sections = parser(user_id, data, engine)
        save_db(user_id, engine, elements=elements, nodes=nodes,
point_loads=point_loads, sections=sections)
    return render_template('editor.html')
```

Url Endpoint (‘/save) (app.py)

6. ‘/load’
   Πραγματοποιείται στο παρασκήνιο μέσω AJAX Call. Τα δεδομένα του μοντέλου που εχουν αποθηκευτεί στη βάση δεδομένων ανακτούνται και επανασχεδιάζονται

```
@app.route('/load', methods=['GET', 'POST'])
def load():
   if request.method == 'POST':
       user = session['username']
       nodes = pd.read_sql("SELECT * from nodes WHERE user_id='" + user + "'", engine)
       del nodes['id'], nodes['user_id']
       elements = pd.read_sql("SELECT * from elements WHERE user_id='" + user + "'",
engine)
       del elements['id'], elements['user_id']
       point_loads = pd.read_sql("SELECT * from point_loads WHERE user_id='" + user +
"'", engine)
       del point_loads['id'], point_loads['user_id']
       return nodes.to_json(orient='table', index=False) + '|' +
elements.to_json(orient='table', index=False)+ '|' +
point_loads.to_json(orient='table', index=False)
```

Url Endpoint (‘/save) (app.py)

7. ‘/autosave’
   Πραγματοποιείται στο παρασκήνιο μέσω AJAX Call. Τα δεδομένα του μοντέλου αποθηκεύονται στη βάση δεδομένων αυτομάτως προκειμένου να αποφευχθούν απώλειες δεδομένων

```
@app.route('/autosave', methods=["GET", "POST"])
def save():
   if request.method == 'POST':
       data = request.get_json()
       user_id = session['username']
       elements, nodes, point_loads, sections = parser(user_id, data, engine)
       save_db(user_id, engine, elements=elements, nodes=nodes,
point_loads=point_loads, sections=sections)
   return render_template('editor.html')
```

Url Endpoint (‘/autosave) (app.py)

8. ‘/loadsections’
   Πραγματοποιείται στο παρασκήνιο μέσω AJAX Call. Επιστρέφει στο load τις ορισμένες από τον χρήστη διατομές

```
@app.route('/loadsections', methods=['POST'])
def load_sections():
   user_id = session['username']
   sect = pd.read_sql("SELECT material, sect_type, section_id from sections WHERE
user_id='" + user_id + "'", engine)
   return sect.to_json(orient='table', index=False)
```

Url Endpoint (‘/autosave) (app.py)

9. ‘/yellow’
   Πραγματοποιείται στο παρασκήνιο μέσω AJAX Call. Ξεκινάει την διαδικασία ανάλυσης

```
@app.route('/yellow', methods=["GET", "POST"])
def run_analysis():
    if request.method == 'POST':
        data = request.get_json()
        user_id = session['username']
        main(user_id, engine)
    return render_template('editor.html')
```

Url Endpoint (‘/autosave) (app.py)

4.1.2 Βάση Δεδομένων

Η διαχείριση πάσης φύσεως δεδομένων(χρήστες, μοντέλο, αποτελέσματα) ξεκινούν και καταλήγουν στην βάση δεδομένων. Επιλέχθηκε βάση δεδομένων(MySQL) λόγω της εξαιρετικής ευελιξίας που προσφέρουν στη διαχείριση τους.

Για την υλοποίηση και διεπαφή με τη βάση δεδομένων χρησιμοποιήθηκαν και η εξειδικευμένη γλώσσα SQL αλλά και η βιβλιοθήκη Sqlalchemy της Python

4.2 Διαχείριση Δεδομένων-Βάση Δεδομένων

4.2.1 Δημιουργία Βάσης Δεδομένων

Η δημιουργία των πινάκων αυτοματοποιήθηκε για διευκόλυνση εγκατάστασης και ανάπτυξης. Στο αρχείο models.py, υπάρχουν οι κλάσεις της αντιστοίχισης των πινάκων της βάσης με αντικείμενα της γλώσσας Python.

Για την υλοποίηση και διεπαφή με τη βάση δεδομένων χρησιμοποιήθηκαν και η εξειδικευμένη γλώσσα SQL αλλά και η βιβλιοθήκη Sqlalchemy της Python.

Η παρούσα υλοποίηση της βάσης δεδομένων αποτελείται από εννέα πίνακες:

- Users
- Nodes

- Elements
- Point Loads
- MQN
- Steel Sections
- Materials
- Displacements
- Sections

Η χρήση και αρχιτεκτονική των πινάκων θα αναλυθούν στο επόμενο κεφάλαιο.

```
engine = create_engine('mysql+pymysql://root:password@localhost/yellow')
Base.metadata.create_all(bind=engine)
```

Κώδικας δημιουργίας πινάκων(models.py)

4.2.2 Πίνακες

Πίνακας χρηστών(user)

Αποθηκεύονται οι χρήστες της εφαρμογής. Αποτελείται από τρεις στήλες: id, όνομα χρήστη και τον κωδικό. Ο κωδικός αποθηκεύεται κρυπτογραφημένος στη βάση.

id	username	password
1	cv13116	5rounds=535000$Sg9WyH8hbZPCchnL$U2G...
2	admin	5rounds=535000$52S5EGzpAJcKB/HW$Wpf...
NULL	NULL	NULL

Πίνακας χρηστών(user)

Πίνακας κόμβων(nodes)

Αποθηκεύονται οι κόμβοι των ορισμένων μοντέλων. Αποτελείται από δώδεκα στήλες: id, όνομα χρήστη, συντεταγμένες κόμβου και τους βαθμούς ελευθερίας του κόμβου. Ο αριθμός 1 υποδηλώνει ελεύθερο βαθμό ελευθερίας ενω το 0 δεσμευμένο.

id	user_id	nn	coord_x	coord_y	coord_z	dof_dx	dof_dy	dof_dz	dof_rx	dof_ry	dof_rz
1	cv13116	1	0	0	0	0	0	0	0	0	0
2	cv13116	2	3	3	3	1	1	1	1	1	1
3	cv13116	3	3	4	4	1	1	1	1	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Πίνακας κόμβων(nodes)

Πίνακας μελών(elements)

Αποθηκεύονται τα κόμβοι των ορισμένων μοντέλων. Αποτελείται από οκτώ στήλες: id, όνομα χρήστη, αριθμός μέλους, κόμβοι αρχής και τέλους, το μήκος, ο τύπος του μέλος και ο αριθμός της διατομής.

id	user_id	en	nodei	nodej	length	elem_type	section_id
1	cv13116	1	1	2	5.19615	beam	1
2	cv13116	2	2	3	1.41421	beam	1
4	cv13116	1	1	2	5.19615	beam	1
5	cv13116	2	2	3	1.41421	beam	1
10	admin	1	1	2	5.19615	beam	1
11	admin	2	2	3	1.41421	beam	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Πίνακας μελών(elements)

Πίνακας διατομών(sections)

Αποθηκεύονται τα διατομές που έχουν ορίσει οι χρήστες. Αποτελείται από δώδεκα στήλες: id, όνομα χρήστη, ο αριθμός της διατομής, το υλικο και οι γεωμετρικές ιδιότητες της διατομής.

id	user_id	section_id	material	sect_type	E	G	A	Ix	Iy	Iz
7	cv13116	1	S255	HEA120	210000	136500	0.00438	0.00000369	0.0000134	0.000000149
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Πίνακας διατομών(sections)

Πίνακας σημειακών φορτίων(point_loads)

Αποθηκεύονται τα σημειακά φορτία εκάστοτε μοντέλου. Αποτελείται από δώδεκα στήλες: id, ο αριθμός της διατομής, το υλικο, του μέλος και η ταυτότητα της διατομής.

id	user_id	nn	c	p_x	p_y	p_z	m_x	m_y	m_z
1	cv13116	1	NULL	100	100	0	0	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Πίνακας σημειακών φορτίων(point_loads)

Η διαχείριση πάσης φύσεως δεδομένων(χρήστες, μοντέλο, αποτελέσματα) ξεκινούν και καταλήγουν στην βάση δεδομένων. Επιλέχθηκε βάση δεδομένων(MySQL) λόγω της εξαιρετικής ευελιξίας που προσφέρουν στη διαχείριση τους.

5

Υπολογισμοί-Ανάλυση

5.1 Παρουσίαση αλγορίθμου επίλυσης

Θα παρουσιαστούν παράλληλα και το θεωρητικό υπόβαθρο και η υλοποίηση των υπολογισμών.

Οι υπολογισμοί βασίστηκαν στην μέθοδο Άμεσης Δυσκαμψίας. Η μέθοδος βασίζεται στην εξίσωση ισορροπίας:

$$P = K * \Delta$$

όπου:

K: ολικό μητρώο κατασκευής
P: επικόμβιες δυνάμεις
Δ: επικόμβιες μετακινήσεις

Παρακάτω αναφέρονται τα βήματα επίλυσης:

5.1.1 Φόρτωση δεδομένων

Καλείται η συνάρτηση load_data(user_id, engine). Η συνάρτηση φορτώνει από τη βάση δεδομένων το μοντέλο που έχει ορίσει ο χρήστης στα αντίστοιχα objects: Nodes, Elements, PointLoads, DistLoads, Sections, Materials.

Κατά τη διαδικασία ορισμού υπολογίζονται απευθείας επιμέρους στοιχεία κάθε αντικειμένου.

5.1.2 Αρίθμηση βαθμών ελευθερίας

Σε κάθε κόμβο απαιτείται η αρίθμηση των αντίστοιχων βαθμων ελευθερίας του. Αυτό επιτυγχάνεται στον αλγόριθμό κατα τον ορισμό κάθε αντικειμένου κόμβου στο class Nodes.

5.1.3 Υπολογισμός μητρώων μετασχηματισμού

Προκειμένου όλα τα μεγέθη της κατασκευής να έχουν κοινό σύστημα αναφοράς απαιτείται ο μετασχηματισμός των μελών από τα αντίστοιχα τοπικά συστήματα στο καθολικό σύστημα της κατασκευής. Αυτό επιτυγχάνεται από το τα μητρώα μετασχηματισμού. Αυτά δημιουργούνται απο την ακόλουθη διαδικασία:

5.1.4 Υπολογισμός μητρώων στιβαρότητας(τοπικών και ολικών)

Για κάθε μέλος υπολογίζεται το τοπικό μητρώο στιβαρότητας με βάση το μητρώο χωρικού στοιχείου με έξι βαθμού ελευθερίας σε κάθε άκρο.

$$k_{local} = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EIz}{L^3} & 0 & 0 & 0 & \frac{6EIz}{L^2} & 0 & -\frac{12EIz}{L^3} & 0 & 0 & 0 & \frac{6EIz}{L^2} \\
0 & 0 & \frac{12Ey}{L^3} & 0 & -\frac{6EIy}{L^2} & 0 & 0 & 0 & -\frac{12Ey}{L^3} & 0 & -\frac{6EIy}{L^2} & 0 \\
0 & 0 & 0 & \frac{GIx}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GIx}{L} & 0 & 0 \\
0 & 0 & -\frac{6EIy}{L^2} & 0 & \frac{4EIy}{L} & 0 & 0 & 0 & \frac{6EIy}{L^2} & 0 & \frac{2EIy}{L} & 0 \\
0 & \frac{6EIz}{L^2} & 0 & 0 & 0 & \frac{4EIz}{L} & 0 & -\frac{6EIz}{L^2} & 0 & 0 & 0 & \frac{2EIz}{L} \\
-\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EIz}{L^3} & 0 & 0 & 0 & -\frac{6EIz}{L^2} & 0 & \frac{12EIz}{L^3} & 0 & 0 & 0 & -\frac{6EIz}{L^2} \\
0 & 0 & -\frac{12EIy}{L^3} & 0 & \frac{6EIy}{L^2} & 0 & 0 & 0 & \frac{12EIy}{L^3} & 0 & \frac{6EIy}{L^2} & 0 \\
0 & 0 & 0 & -\frac{GIx}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GIx}{L} & 0 & 0 \\
0 & 0 & -\frac{6EIy}{L^2} & 0 & \frac{2EIy}{L} & 0 & 0 & 0 & \frac{6EIy}{L^2} & 0 & \frac{4EIy}{L} & 0 \\
0 & \frac{6EIz}{L^2} & 0 & 0 & 0 & \frac{2EIz}{L} & 0 & -\frac{6EIz}{L^2} & 0 & 0 & 0 & \frac{4EIz}{L}
\end{bmatrix}$$

Στην συνέχεια ορίζεται το αντίστοιχο καθολικό μητρώο στιβαρότητας του μέλους.

$$k_{global} = L * k_{local}$$

όπου:

k_{global}: καθολικό μητρώο μέλους
Λ: μητρώο μετασχηματισμού
k_{local}: τοπικό μητρώο μέλους

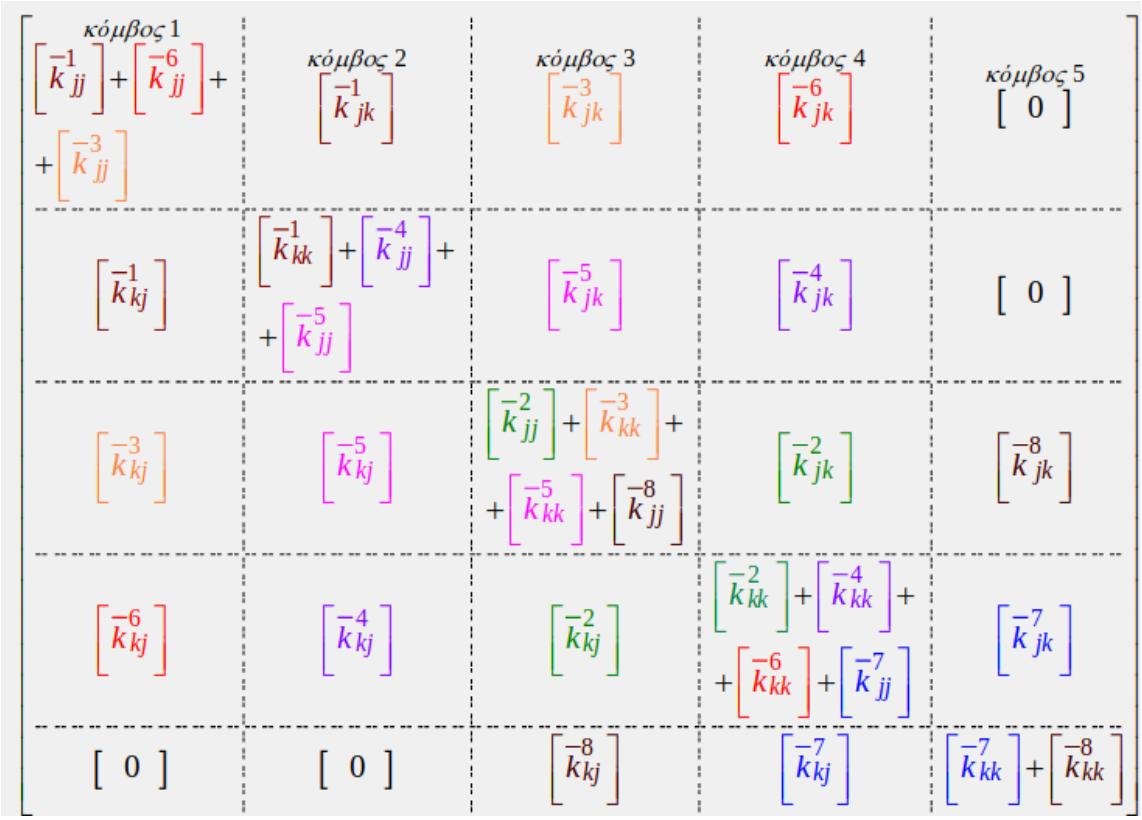
5.1.5 Ορισμός αντικειμένου model

Το αντικείμενο model αποτελεί instance της κλάσης Model το οποίο έχει ως attributes όλα τα instances των στοιχείων που όρισε ο χρήστης.

Οι μέθοδοι του της κλάσης Model αντιπροσωπεύουν τις απαραίτητες διαδικασίες επίλυσης η οποίες παρουσιάζονται στη συνέχεια.

5.1.6 Υπολογισμός ολικού μητρώου δυσκαμψίας κατασκευής

Γίνεται η σύνθεση του ολικού μητρώου της κατασκευής. Αυτό επιτυγχάνεται χρησιμοποιώντας τον αρχικό και τελευταίο βαθμό ελευθερίας κάθε άκρου του μέλους και προσθέτοντας τους κόμβους με κοινούς βαθμούς ελευθερίας.



5.1.7 Δημιουργία διανύσματος επικόμβιων δυνάμεων

Τα φορτία που δρουν στους κόμβους προστίθενται στο διάνυσμα επικόμβιων δυνάμεων. Αυτό επιτυγχάνεται χρησιμοποιώντας τον αρχικό και τελευταίο βαθμό

ελευθερίας κάθε άκρου του μέλους και προσθέτοντας τα φορτία τα οποία δρουν σε κοινούς βαθμούς ελευθερίας.

$$\left\{\overline{P}\right\} = \left[\begin{array}{c} \left\{\overline{P}^{(1)}\right\} \\ \hline \left\{\overline{P}^{(2)}\right\} \\ \hline \vdots \\ \hline \left\{\overline{P}^{(N)}\right\} \end{array}\right]$$

5.1.8 Υπολογισμός δράσεων παγίωσης

Ο υπολογισμός αντιδράσεων του παγιωμένου φορέα αφορά την αντιμετώπιση των φορτίων εντός του μέλους είτε σημειακού, είτε κατανεμημένου. Υπολογίζονται οι αντιδράσεις του αντιστοιχου αμφίπακτου μέλους με την ίδια φόρτιση. Οι αντιδράσεις τροποποιούνται με το μητρώο μετασχηματισμού του μέλους προκειμένου να αναφέρονται στο καθολικό σύστημα. Οι δράσεις παγίωσης τοποθετούνται στο διάνυσμα S.

5.1.9 Αναδιάταξη βαθμών ελευθερίας

Η μεθοδολογία διαχωρισμού των γνωστών από τα άγνωστα μεγέθη στα διανύσματα ολικών επικόμβιων δράσεων και ολικών επικόμβιων μετατοπίσεων επιτυγχάνεται με τη μετάθεση στοιχείων στα διανύσματα αυτά ή όπως αλλιώς καλείται αναδιάταξη των διανυσμάτων αυτών. Πιο συγκεκριμένα επιδιώκεται η αναδιάταξη των στοιχείων των διανυσμάτων επικόμβιων δράσεων και επικόμβιων μετατοπίσεων , έτσι ώστε και στα δύο διανύσματα να προηγούνται οι ελεύθεροι βαθμοί ελευθερίας (γνωστές επικόμβιες δράσεις και άγνωστες επικόμβιες μετατοπίσεις) και να έπονται οι δεσμευμένοι βαθμοί ελευθερίας (άγνωστες επικόμβιες δράσεις και γνωστές επικόμβιες μετατοπίσεις). Είναι προφανές ότι η προαναφερθείσα αναδιάταξη θα έχει ως άμεσο αποτέλεσμα και την απαιτούμενη τροποποίηση του καθολικού μητρώου στιβαρότητας του δικτυώματος.

5.1.10 Μητρώο αναδιάταξης και αναδιάταξη μητρώων/διανυσμάτων

Το μητρώο αναδιάταξης χρησιμοποιείται προκειμένου να ανταποκρίνονται τα μητρώα στιβαρότητας και διανύσματα στην αναδιάταξη των βαθμών ελευθερίας.

$$V = \begin{cases} & \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{matrix} \\ free & \\ & \vdots \\ & \begin{matrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix} \\ sup & \end{cases}$$

$$\Delta_m = V \cdot \Delta = \frac{\Delta_f}{\Delta_s}$$

$$P_m = V \cdot P = \frac{P_f}{P_s}$$

5.1.11 Επίλυση εξίσωσης ισορροπίας

Η τελική επίλυση γίνεται με τα αναδιατεταγμένα μητρώα και υπολογίζονται τα αντίστοιχα μεγέθη.

$$P_m = K_m \cdot \Delta_m \Rightarrow \begin{bmatrix} P_f \\ P_s \end{bmatrix} = \begin{bmatrix} K_{ff} & K_{fs} \\ K_{sf} & K_{ss} \end{bmatrix} \cdot \begin{bmatrix} \Delta_f \\ \Delta_s \end{bmatrix} \quad (5.1.11.1)$$

$$\Delta_f = K_{ff}^{-1} \cdot \left(P_f - K_{fs} \cdot \Delta_s \right) \quad (5.1.11.2)$$

$$P_s = K_{sf} \cdot \Delta_f + K_{ss} \cdot \Delta_s \quad (5.1.11.3)$$

Δ_f : επικόμβιες μετακινήσεις ελεύθερων κόμβων
P_s: αντιδράσεις φορέα

5.1.12 Υπολογισμός παραμορφωμένων μελών

Έχοντας υπολογίσει τις επικόμβιες μετακινήσεις μετατρέπουμε αυτές στο τοπικό σύστημα του εκάστοτε μέλους μπορούμε να υπολογίσουμε τα τις τοπικές μετακινήσεις των άκρων του μέλους

$$\Delta_{local} = \Lambda \cdot \Delta_{global}$$

Για την κατανομή των παραμορφώσεων κατά μήκος του μέλους χρησιμοποιούνται οι παρακάτω εξισώσεις σχήματος γραμμικού στοιχείου δύο κόμβων.

$$q = x/L$$

$$N_1 = 1 - 3q^2 + 2q^3$$
$$N_2 = Lq(1 - 2q + q^2)$$
$$N_3 = q^2(3 - 2q)$$
$$N_4 = Lq^2(q - 1)$$

$$d = N_1 * v1 + N_2 * theta1 + N_3 * v2 + N_4 * theta2$$

5.1.13 Υπολογισμός εντατικών μεγεθων

Τα εντατικά μεγέθη άκρων υπολογίζονται απο τον πολλαπλασιασμό του τοπικού μητρώου και των τοπικών μετακινήσεων.

$$MQN_{nodal} = k_{local} \cdot \Delta_{local}$$

Κατά μήκος του μέλους τα διαγράμματα υπολογίζονται απευθείας με επανάληψη των φορτίων κατα μήκος του μέλους.

6

Αποτελέσματα Ανάλυσης (Post Processing)

6.1 Παρουσίαση αποτελεσμάτων

Μετά την ολοκλήρωση της ανάλυσης τα αποτελέσματα αποθηκεύονται στη βάση δεδομένων. Η εφαρμογή έχει την δυνατότητα εκτενούς παρουσίασης των αποτελεσμάτων. Αυτά εμφανίζονται σε μορφή πινάκων και διαγραμμάτων. Επίσης υπάρχει η δυνατότητα αποθήκευσης των αποτελεσμάτων σε αρχεία csv

6.1.1 Αντιδράσεις στηρίξεων

Οι αντιδράσεις του φορέα υπολογίζονται από την εξίσωση 5.11.1.3. Οι αντιδράσεις αντιστοιχούνται στους δεσμευμένους κόμβους. Ο χρήστης έχει εποπτεία των αντιδράσεων μέσω πίνακα από την καρτέλα RESULTS.

Node	Fx	Fy	Fz	Mx	My	Mz
1	5.228	112.646	133.619	-249.929	20.498	-3.984
3	19.673	92.899	-13.331	-223.684	48.765	0.316
5	16.087	194.455	41.688	-351.212	56.499	9.342

Πίνακας αντιδράσεων (kN)

6.1.2 Επικομβιες μετακινήσεις

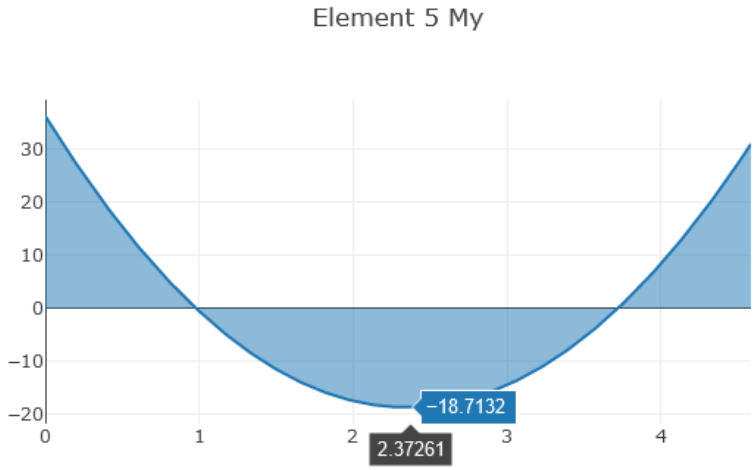
Οι μετακινήσεις των ελεύθερων κόμβων του φορέα υπολογίζονται από την εξίσωση 5.11.1.3. Ο χρήστης έχει εποπτεία των μετακινήσεων μέσω πίνακα από την καρτέλα RESULTS.

Node	ux	uy	uz	rx	ry	rz
1	0	0	0	0	0	0
2	-0.02	-0.14742	-0.00015	0.01821	-0.00742	0.00443
3	0	0	0	0	0	0
4	-0.03331	-0.14755	0.00001	0.028	-0.00696	-0.00035
5	0	0	0	0	0	0
6	-0.01692	-0.08188	-0.00002	0.05793	-0.01493	-0.0052

Πίνακας μετακινήσεων(m)

6.1.3 Εντατικά μεγέθη

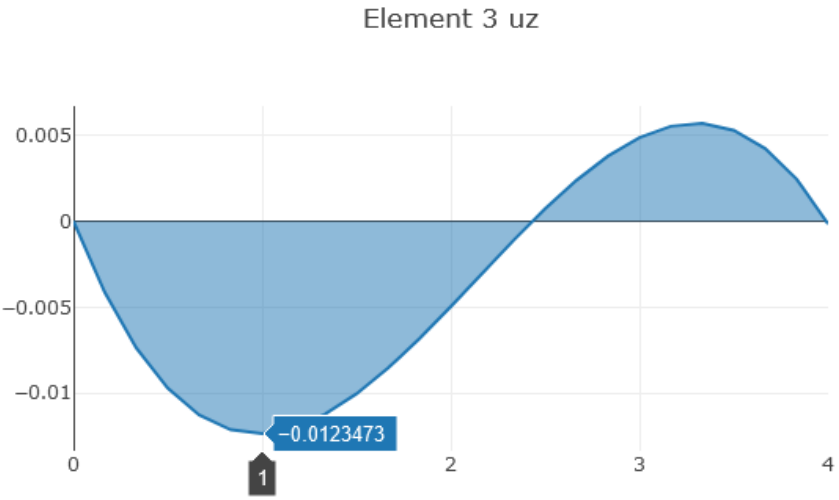
Έχοντας οι υπολογίσει τις επικόμβιες μετακινήσεις μπορούμε να υπολογίσουμε και τα εντατικά μεγέθη στους κόμβους κατά τη διεύθυνση του κάθε μέλους. Αυτό απαιτεί την μετατροπή των καθολικών μετακινήσεων στο τοπικό σύστημα κάθε μέλους.



Διάγραμμα Ροπής(kN)

6.1.4 Παραμορφωσεις μελών

Όπως αναφέρθηκε σε παραπάνω κεφάλαιο μέσω των εξισώσεων σχήματος υπολογίζονται τα παραμορφωμένα μέλη στους τοπικούς άξονες στις διευθύνσεις X, Y.



Διάγραμμα Ροπής(kN)

Βιβλιογραφία

1. Βαλιάσης Θ. (1999). «ΣΤΑΤΙΚΗ ΤΩΝ ΓΡΑΜΜΙΚΩΝ ΦΟΡΕΩΝ». Εκδόσεις ΖΗΤΗ, Αθήνα.
2. Σπυράκος Κ. (1996). «Finite Element Modeling: In Engineering Practice», Εκδόσεις Algor Inc
3. A. Ghali and A.M Neville. (2009). «Structural Analysis A unified classical and matrix approach». CRC Press
4. Παπαδρακάκης Μ., Σαπουντζάκης Ε.Ι. (2016) "Ανάλυση Ραβδωτών Φορέων με Μητρωικές Μεθόδους – Μέθοδος Άμεσης Στιβαρότητας", Αθήνα, Εκδόσεις Τσότρας
5. https://docs.python.org/3.6/
6. https://pythonprogramming.net/practical-flask-introduction/