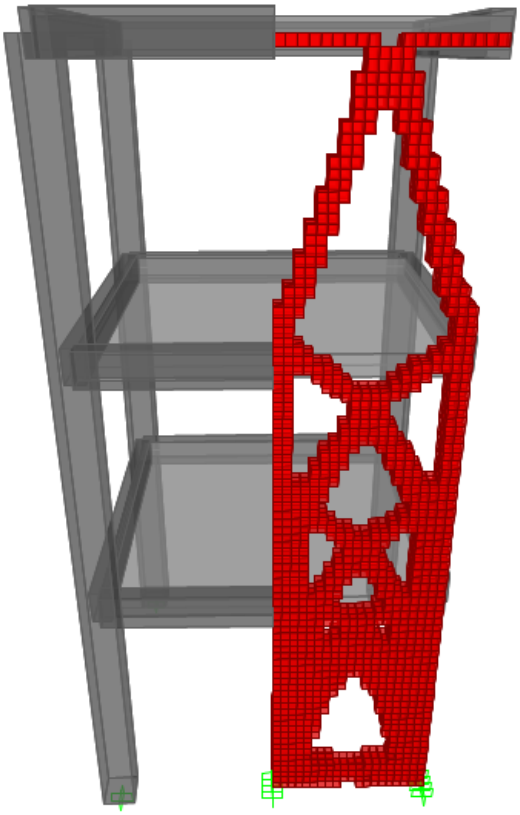
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Πολιτικών Μηχανικών
Τομέας Δομοστατικής
Εργαστήριο Στατικής και Αντισεισμικών Ερευνών

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΤΟΠΟΛΟΓΙΑΣ ΓΙΑ ΤΟ SAP2000 ΣΕ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C#



Διπλωματική Εργασία

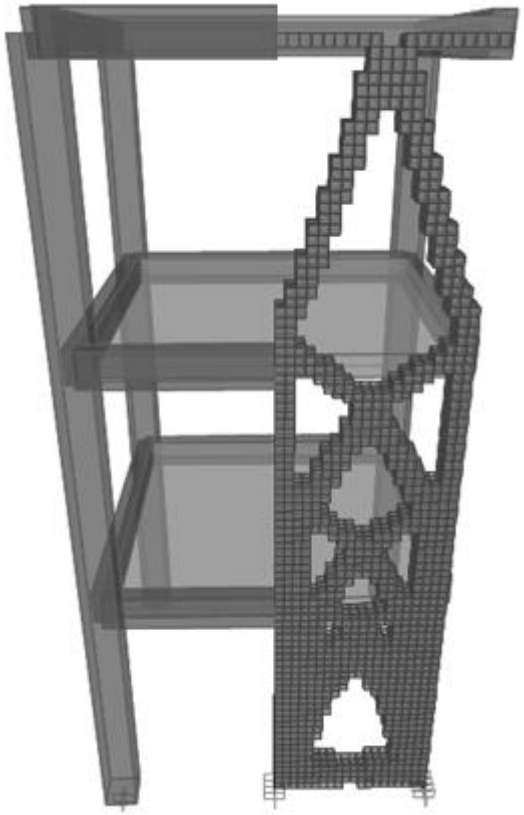
Νικόλαος - Αριστείδης Βασιλείου

Επιβλέπων: Νικόλαος Λαγαρός, Επίκουρος Καθηγητής ΕΜΠ
Συνεπιβλέπων: Γεώργιος Καζάκης, ΥΔ ΕΜΠ

Αθήνα, Οκτώβριος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Πολιτικών Μηχανικών
Τομέας Δομοστατικής
Εργαστήριο Στατικής και Αντισεισμικών Ερευνών

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΤΟΠΟΛΟΓΙΑΣ ΓΙΑ ΤΟ SAP2000 ΣΕ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C#



Διπλωματική Εργασία

Νικόλαος - Αριστείδης Βασιλείου

Επιβλέπων: Νικόλαος Λαγαρός, Επίκουρος Καθηγητής ΕΜΠ
Συνεπιβλέπων: Γεώργιος Καζάκης, ΥΔ ΕΜΠ

Αθήνα, Οκτώβριος 2016

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, Θα ήθελα να ευχαριστήσω τον κ. Νικόλαο Λαγαρό, επιβλέποντα καθηγητή της διπλωματικής εργασίας, για την εξαιρετική συνεργασία που είχαμε και το ειλικρινές του ενδιαφέρον. Η οργάνωση και η καθοδήγησή του σε όλη την διάρκεια, αποτέλεσαν καθοριστικό παράγοντα για την ολοκλήρωση της εργασίας.

Επιπλέον, ευχαριστώ τον υποψήφιο διδάκτορα Γιώργο Καζάκη, για την πολύ σημαντική βοήθειά του τόσο στην κατανόηση των θεωρητικών στοιχείων όσο και στην υλοποίησή τους.

Ακόμα, θα ήθελα να ευχαριστήσω την εταιρεία ACE-HELLAS και ιδιαίτερα τον πρόεδρο κ. Διονύσιο Ιωακείμ που μου έδωσε την δυνατότητα να κάνω την πρακτική μου άσκηση. Κατά τη διάρκεια αυτής, πήρα πολύτιμη βοήθεια και συμβουλές για το πρόγραμμα SAP2000 και την ανάπτυξη του κώδικα.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ στους φίλους και στην οικογένειά μου, για την υποστήριξή τους σε όλη την διάρκεια των σπουδών μου.

Νίκος Βασιλείου

Οκτώβριος 2016

ΠΕΡΙΛΗΨΗ

Αντικείμενο της διπλωματικής εργασίας ήταν η ανάπτυξη ενός λογισμικού βελτιστοποίησης τοπολογίας που να έχει εφαρμογή σε προβλήματα πολιτικού μηχανικού. Ο κώδικας γράφτηκε στην πλατφόρμα Visual Studio σε γλώσσα προγραμματισμού C#. Ως βάση για τις αναλύσεις χρησιμοποιήθηκε το πρόγραμμα SAP2000, το οποίο μπορεί να συνδεθεί με το Visual Studio. Το λογισμικό που παρουσιάζεται, δίνει την δυνατότητα επίλυσης προβλημάτων βελτιστοποίησης τοπολογίας σε συνεχή συστήματα με τις δύο πιο δημοφιλείς μεθοδολογίες, την Optimality Criteria (OC) και την Method of Moving Asymptotes (MMA). Στην αρχή εφαρμόστηκαν απλά παραδείγματα, όπως η βελτιστοποίηση ενός προβόλου ή μιας αμφιέρειστης δοκού. Στην συνέχεια εφαρμόστηκαν πιο σύνθετα παραδείγματα, όπως η βελτιστοποίηση του τοιχίου μιας κατασκευής.

ABSTRACT

The subject of this thesis was the development of a topology optimization software which is applicable to structural problems. The code was written in Visual Studio using C# programming language. SAP2000 was used for the analysis, which can be connected with Visual Studio. The software that is presented solves topology optimization problems of distributed parameter systems using the two most popular algorithms, Optimality Criteria (OC) and Method of Moving Asymptotes (MMA). Initially, simple examples were applied, such as the optimization of a cantilever or the optimization of a simply supported beam. Subsequently, more complicated examples were applied, such as the optimization of a wall.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο και στόχος διπλωματικής εργασίας

Κάθε μηχανικός είτε σχεδιάζει μια μηχανή, είτε ένα εξάρτημα, είτε ένα κτίριο, αποσκοπεί σε μια όσο το δυνατόν καλύτερη λύση. «Βέλτιστος» ονομάζεται ένας σχεδιασμός που ικανοποιεί τις λειτουργικές προδιαγραφές και τους περιορισμούς, ελαχιστοποιώντας ταυτόχρονα συγκεκριμένα κριτήρια όπως το κόστος ή/και το βάρος.

Η βελτιστοποίηση τοπολογίας είναι μια μαθηματική διαδικασία η οποία έχει ως στόχο την εύρεση της βέλτιστης κατανομής υλικού σε μια κατασκευή, για δεδομένες φορτίσεις και συνθήκες στήριξης. Υλοποιείται με χρήση πεπερασμένων στοιχείων για την ανάλυση και διάφορων τεχνικών βελτιστοποίησης, όπως η Method of Moving Asymptotes (MMA), Optimality Criteria (OC), μέθοδοι level set κ.α. Ο συγκεκριμένος τομέας βελτιστοποίησης βρίσκει ευρύ πεδίο εφαρμογών στον τομέα της αυτοκινητοβιομηχανίας, της ναυπηγικής και αεροναυπηγικής. Έτσι, τα περισσότερα λογισμικά επικεντρώνονται σε αυτούς τους τομείς.

Στόχος της διπλωματικής εργασίας ήταν η ανάπτυξη ένος λογισμικού βελτιστοποίησης τοπολογίας που να έχει εφαρμογή σε προβλήματα πολιτικού μηχανικού. Γι αυτό το λόγο χρησιμοποιεί ως βάση για τις αναλύσεις το στατικό πρόγραμμα SAP2000. Το λογισμικό δίνει την δυνατότητα επίλυσης προβλημάτων βελτιστοποίησης τοπολογίας σε συνεχή συστήματα με τις δύο πιο δημοφιλείς μεθοδολογίες, την OC και την MMA.

Η εφαρμογή του βοηθάει στην δημιουργία ενός φορέα που προσεγγίζει σε μεγάλο βαθμό την τελική κατασκευή, έχοντας ως αποτέλεσμα την καλύτερη κατανομή υλικού και την εξοικονόμηση χρόνου σύλληψης.

1.2 Δομή

Παρακάτω παρουσιάζεται η δομή της διπλωματικής εργασίας και περιγράφεται συνοπτικά το κάθε κεφάλαιο.

- ΚΕΦΑΛΑΙΟ 2: Βιβλιογραφική επισκόπηση

  Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της βιβλιογραφικής έρευνας η οποία έγινε κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας. Περιγράφονται συνοπτικά επιστημονικά άρθρα, δημοσιεύσεις και συγγράμματα σχετικά με την βελτιστοποίηση τοπολογίας και παρουσιάζονται τα υπάρχοντα εμπορικά ή ακαδημαϊκά λογισμικά βελτιστοποίησης τοπολογίας.

- ΚΕΦΑΛΑΙΟ 3: Θεωρητικό υπόβαθρο

  Στο κεφάλαιο αυτό αναλύονται βασικές έννοιες που χρειάζονται για να γίνει η κατανόηση του προβλήματος της βελτιστοποίησης τοπολογίας.

- ΚΕΦΑΛΑΙΟ 4: Αντικειμενοστραφής προγραμματισμός με C# - Application Programming Interface του SAP2000

  Στο κεφάλαιο αυτό παρουσιάζονται τα μέσα που χρησιμοποιήθηκαν για την ανάπτυξη του λογισμικού βελτιστοποίησης τοπολογίας, δηλαδή η γλώσσα προγραμματισμού C#, η πλατφόρμα Visual Studio και το πρόγραμμα SAP2000.

- ΚΕΦΑΛΑΙΟ 5: Λογισμικό βελτιστοποίησης τοπολογίας

  Στο κεφάλαιο αυτό ορίζεται το πρόβλημα βελτιστοποίησης τοπολογίας σε συνεχή συστήματα και παρουσιάζονται δυο διαφορετικές μεθοδολογίες επίλυσής του, η Optimality Criteria και η Method of Moving Asymptotes. Επιπλέον, περιγράφεται αναλυτικά και σε μορφή διαγράμματος ροής το λογισμικό που αναπτύχθηκε, το οποίο υλοποιεί τις παραπάνω μεθοδολογίες.

- ΚΕΦΑΛΑΙΟ 6: Αριθμητικές εφαρμογές - Αποτελέσματα

  Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα που προέκυψαν και από τις δύο μεθοδολογίες του λογισμικού. Το πρώτο μέρος αποτελείται από απλά παραδείγματα, όπως πρόβολοι και αμφιέρειστες ενώ το δεύτερο μέρος από πιο σύνθετα, όπως το τοιχίο μια τριώροφης κατασκευής.

- ΚΕΦΑΛΑΙΟ 7: Συμπεράσματα

  Στο κεφάλαιο αυτό καταγράφονται τα συμπεράσματα που προέκυψαν από την εφαρμογή όλων των παραδειγμάτων και προτείνονται λύσεις για βελτίωση του λογισμικού.

ΚΕΦΑΛΑΙΟ 2

ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ

2.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της βιβλιογραφικής έρευνας η οποία έγινε κατά τη διάρκεια εκπόνησης της διπλωματικής εργασίας. Στόχος της ήταν η συλλογή πληροφοριών για την κατανόηση του προβλήματος της βελτιστοποίησης τοπολογίας. Αρχικά, παρουσιάζονται οι κυριότερες μέθοδοι βελτιστοποίησης τοπολογίας και επιπλέον γίνεται αναφορά στους αλγόριθμους που χρησιμοποιούνται για επίλυση προβλημάτων μη-γραμμικού προγραμματισμού. Στην συνέχεια, παρουσιάζονται εμπορικά λογισμικά εταιρειών και ακαδημαϊκά λογισμικά ερευνητικών ομάδων που χρησιμοποιούνται για επίλυση προβλημάτων βελτιστοποίησης τοπολογίας. Τέλος, παρουσιάζονται εφαρμογές βελτιστοποίησης τοπολογίας σε κατασκευές πολιτικού μηχανικού.

2.2 Μέθοδοι για βελτιστοποίηση τοπολογίας

Ο Rosvany [1] στο άρθρο του *«a critical review of established methods of structural topology optimization»* παρουσίασε και σύγκρινε τις δυο καθιερωμένες μεθόδους που χρησιμοποιούνται στην βελτιστοποίηση τοπολογίας, την **SIMP** και την **ESO (ή SERA):**

Η ονομασία **SIMP** σημαίνει **Solid Isotropic Material with Penalization** και είναι η πιο διαδεδομένη μέθοδος πεπερασμένων στοιχείων στην βελτιστοποίηση τοπολογίας. Άλλα ονόματα με τα οποία συναντάται είναι "material interpolation", "artificial material", "power law" ή μέθοδος πυκνότητας. Η SIMP μοντελοποιεί τις μηχανικές ιδιότητες του υλικού ώστε να μην εμφανίζονται πολλά στοιχεία ενδιάμεσων πυκνοτήτων στην βέλτιστη λύση. Η αρχική ιδέα των πεπερασμένων στοιχείων αναφέρθηκε το 1973 από τους Rossow και Taylor [2], οι οποίοι όμως χρησιμοποίησαν γραμμική σχέση πυκνότητας-μητρώου δυσκαμψίας με αποτέλεσμα οι βέλτιστες λύσεις τους να περιέχουν αρκετά γκρι στοιχεία ενδιάμεσων

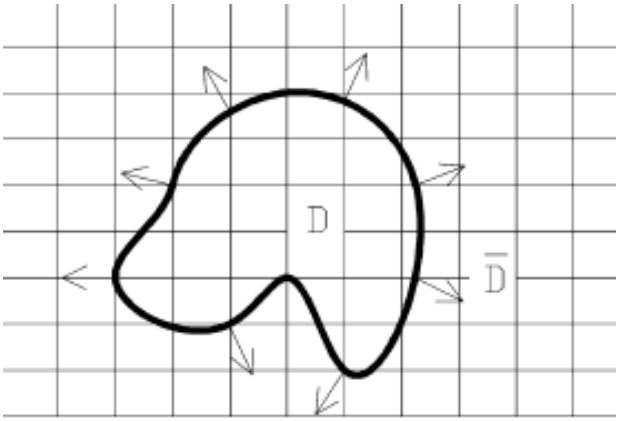
πυκνοτήτων. Σε δημοσίευση τους το 1988 οι Bendsoe and Kikuchi [3] πρότειναν μια μέθοδο ομογενοποίησης με τετραγωνικά ή ορθογωνικά στοιχεία, η οποία έδωσε κάποιο βαθμό ποινικοποίησης των γκρι στοιχείων. Ένα χρόνο αργότερα, το 1989, ο Bendsoe [4] πρότεινε την βασική ιδέα της μεθόδου SIMP, δηλαδή την σχέση $\rho = s^{1/p}$, p>1 η οποία ποινικοποιεί ακόμα περισσότερο τα γκρι στοιχεία. Η ονομασία της μεθόδου καθιερώθηκε το 1992 από τους Rosvany et al [5].

Η **Evolutionary Structural Optimization – ESO** είναι μια μέθοδος που εισάγει αλλαγές στα πεπερασμένα στοιχεία με βάση συγκεκριμένα κριτήρια. Ορίζεται μια συγκεκριμένη συνάρτηση κριτηρίου (π.χ μπορεί να αντιπροσωπεύει τάση von Mises ή πυκνότητα) και υπολογίζεται η τιμή της για κάθε στοιχείο. Σε κάθε επανάληψη, κάποια στοιχεία με την χαμηλότερη τιμή της συνάρτησης κριτηρίου εξαλείφονται , δηλαδή αλλάζουν από μαύρα σε άσπρα. Ο Rosvany και ο Querin [6] υποστήριξαν ότι το όνομα ESO δεν είναι απόλυτα σωστό, καθώς το «optimization» υπονοεί τον υπολογισμό της βέλτιστης λύσης, κάτι το οποίο έχει αποδειχτεί ότι δεν συμβαίνει με την ESO. Έτσι, πρότειναν την ονομασία SERA που σημαίνει Sequential Element Rejections and Admissions.

Η ESO δεν επιτρέπει να επαναφερθούν στοιχεία που έχουν διαγραφεί. Το 1998 οι Yang et al [7] πρότειναν μια παραλλαγή της ESO, την BESO (bi-directional ESO), στην οποία εισάγονται νέα στοιχεία δίπλα στα στοιχεία με την μεγαλύτερη τιμή συνάρτησης κριτηρίου.

Σε πιο πρόσφατες εκδόσεις της ESO χρησιμοποιείται μια διαδικασία δυο σταδίων για καλύτερα αποτελέσματα. Αφού παραχθεί ένας μεγάλος αριθμός αποτελεσμάτων με την διαδικασία της ESO, υπολογίζεται για κάθε λύση η τιμή ενός δείκτη απόδοσης. Τελικά, η βέλτιστη λύση βρίσκεται συγκρίνοντας τα νούμερα των δεικτών απόδοσης και επιλέγοντας το μεγαλύτερο. Ακόμα όμως και αυτή η βελτίωση δέχεται αρκετές κριτικές, όπως ότι μπορεί να μην υπάρχει λογική σχέση μεταξύ της συνάρτησης κριτηρίου και του δείκτη απόδοσης ή ότι δεν είναι πρακτικό να συγκρίνουμε έναν τεράστιο αριθμό αποτελεσμάτων για να καταλήξουμε στην βέλτιστη λύση.

Το 2002 οι Michael Yu Wang, Xiaoming Wang και Dongming Guo [8] παρουσίασαν μια καινούργια πιο προχωρημένη μέθοδο, βασισμένη σε μοντέλα **Level set** για βελτιστοποίηση γραμμικών ελαστικών κατασκευών που ικανοποιούν συγκεκριμένο σχέδιο και δεσμεύσεις. Η κατασκευή που βελτιστοποιείται, αντιπροσωπεύεται από ένα κινούμενο σύνορο ενσωματωμένο σε μία συνάρτηση υψηλότερης διάστασης. Ενώ το σχήμα και η τοπολογία της κατασκευής μπορεί να υποστούν σημαντικές αλλαγές, η level set συνάρτηση παραμένει απλή στην τοπολογία της. Ως εκ τούτου η παρακολούθηση της κίνησης των συνόρων βάσει της σχετικής συνάρτησης οδηγεί στον εντοπισμό των αλλαγών του σχήματος και της τοπολογίας (Εικόνα 2.1). Με τον μαθηματικό αλγόριθμο που πρότειναν, η κίνηση αυτών των συνόρων συγκλίνει στην βέλτιστη λύση. Τα μοντέλα Level set αναφέρονται και ως μοντέλα **Implicit Moving Boundary (IMB)** και είναι ευέλικτα στον χειρισμό σύνθετων τοπολογικών αλλαγών.



Εικόνα 2.1: Η κίνηση των ορίων

2.3 Αλγόριθμοι που χρησιμοποιούνται σε προβλήματα βελτιστοποίησης τοπολογίας

Τα προβλήματα βελτιστοποίησης τοπολογίας είναι συνήθως μη-γραμμικά, δηλαδή η αντικειμενική συνάρτηση εξαρτάται μη-γραμμικά με τις μεταβλητές σχεδιασμού. Αυτά τα προβλήματα λύνονται επαναληπτικά χρησιμοποιώντας αριθμητικές τεχνικές (αλγορίθμους). Οι αλγόριθμοι σε κάθε επανάληψη βρίσκουν νέες αποδεκτές τιμές για τις μεταβλητές σχεδιασμού που να μειώνουν την αντικειμενική συνάρτηση.

Το 1986 οι Fleury και Braibant [9] παρουσίασαν μια μέθοδο μη γραμμικού προγραμματισμού, την **Convex Linearization (CONLIN)**. Η μέθοδος γραμμικοποιεί την αντικειμενική συνάρτηση χωρίζοντας τις μεταβλητές σε δύο διαφορετικές ομάδες και αναπτύσσοντας τους δύο πρώτους όρους της σειράς Taylor. Έτσι, το αρχικό πρόβλημα βελτιστοποίησης αντικαθίσταται με μια ακολουθία προβλημάτων που έχουν απλή αλγεβρική δομή. Τα υποπροβλήματα είναι κυρτά και μπορούν να διαχωριστούν και να λυθούν χρησιμοποιώντας δυαδική μέθοδο.

Ο Krister Svanberg [10] παρουσίασε το 1987 την **Method of Moving Asymptotes (MMA).** Η μέθοδος σε κάθε βήμα της επαναληπτικής διαδικασίας, δημιουργεί και λύνει μια αυστηρά κυρτή προσέγγιση. Η δημιουργία των υποπροβλημάτων ελέγχεται από αυτό που ονομάζει «κινούμενες ασύμπτωτες», οι οποίες σταθεροποιούν και επιταχύνουν την σύγκλιση της διαδικασίας.

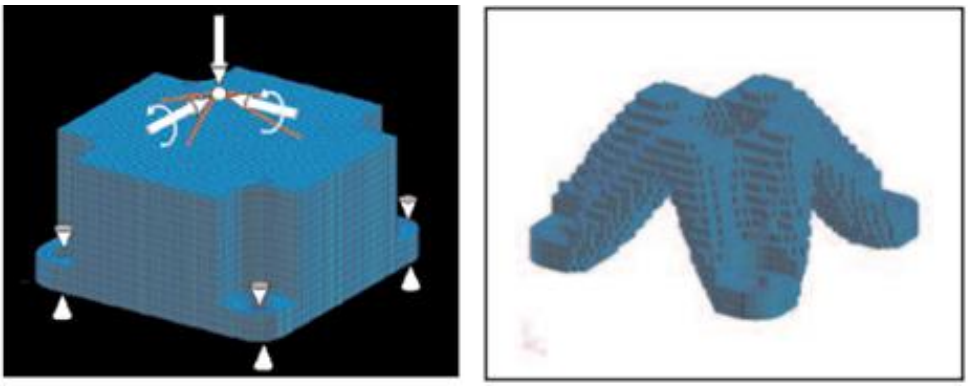
Άλλοι δυο γνωστοί αλγόριθμοι είναι η **Sequential Linear Programming (SLP)** και η **Sequential Quadratic Programming (SQP),** οι οποίες μαζί με τις δύο προηγούμενες περιγράφονται στο βιβλίο *«An Introduction to structural optimization»* [11]. Η SLP χρησιμοποιεί τους δύο πρώτους όρους της σειράς Taylor για να γραμμικοποιήσει την αντικειμενική συνάρτηση και τους περιορισμούς με την μεταβλητή σχεδιασμού, ενώ η SQP χρησιμοποιεί και τον τρίτο όρο της σειράς Taylor για την γραμμικοποίηση.

2.4 Λογισμικά για βελτιστοποίηση τοπολογίας

2.4.1 Εμπορικά λογισμικά πακέτα

MSC Nastran

Το πρόγραμμα MSC Nastran, της εταιρείας MSC Software, προσφέρει δυνατότητες ανάλυσης για στατική, δυναμική φόρτιση και θερμική ανάλυση γραμμικών και μη-γραμμικών φορέων, χρησιμοποιώντας κυρίως μεθόδους πεπερασμένων στοιχείων. Επιπλέον, προσφέρεται επέκταση (Design Optimization) για επίλυση διαφόρων προβλημάτων βελτιστοποίησης. Μέσω αυτής, υπάρχει δυνατότητα επίλυσης προβλημάτων βελτιστοποίησης τοπολογίας, με την μέθοδο πυκνότητας και μαθηματικό προγραμματισμό. Απευθύνεται κυρίως σε Μηχανολόγους Μηχανικούς. Στην εικόνα 2.2 παρουσιάζεται παράδειγμα εφαρμογής βελτιστοποίησης τοπολογίας.



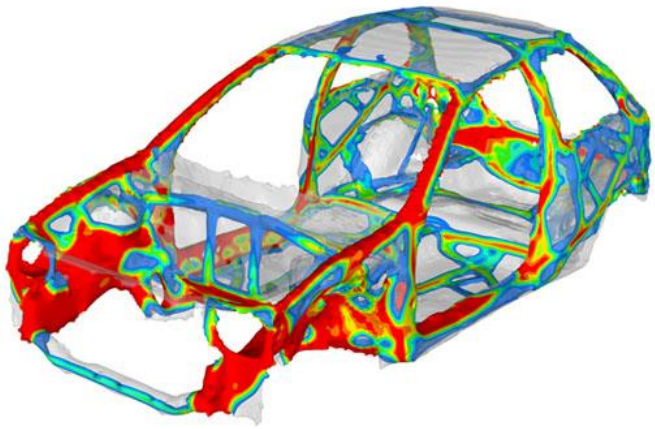
Εικόνα 2.2: Βάση ρεζέρβας αυτοκινήτου

Genesis

Το Genesis (Vanderplaats Research & Development, Inc.) είναι ένα λογισμικό πακέτο που προσφέρει δυνατότητα ανάλυσης πεπερασμένων στοιχείων και βελτιστοποίησης. Μερικές από τις διαθέσιμες επιλογές του για βελτιστοποίηση είναι: μορφής/σχήματος, τοπογραφίας, τοπομετρίας και τοπολογίας. Η βελτιστοποίηση τοπολογίας αφαιρεί υλικό από μια συγκεκριμένη ποσότητα που του δίνεται μέχρι να καταλήξει στην βέλτιστη λύση, ενώ η βελτιστοποίηση τοπομετρίας έχει και την δυνατότητα να προσθέσει υλικό όπου κρίνει ότι χρειάζεται. Το συγκεκριμένο πρόγραμμα επίσης απευθύνεται κυρίως σε Μηχανολόγους Μηχανικούς.

OptiStruct

Το OptiStruct της εταιρείας Altair προσφέρει δυνατότητες ανάλυσης για στατική και δυναμική φόρτιση γραμμικών και μη-γραμμικών φορέων. Χρησιμοποιείται κυρίως από Μηχανολόγους Μηχανικούς (πιο συχνά στον τομέα της αυτοκινητοβιομηχανίας) για ανάλυση και βελτιστοποίηση κατασκευών. Στην εικόνα 2.3 παρουσιάζεται το πλαίσιο αυτοκινήτου που προέκυψε από βελτιστοποίηση τοπολογίας:



Εικόνα 2.3: Πλαίσιο αυτοκινήτου

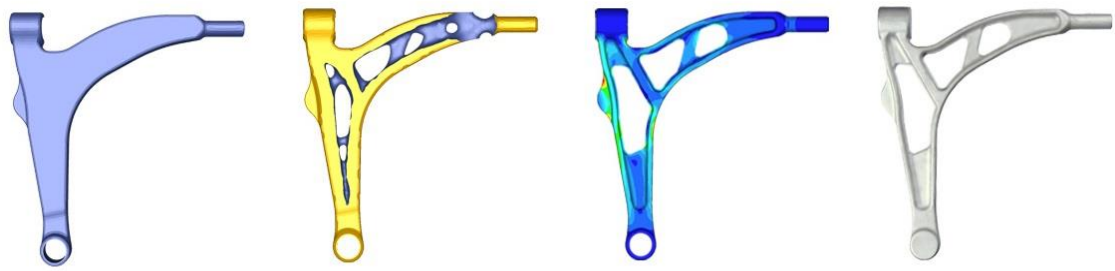
SIMULIA Tosca Structure

Το Tosca Structure (Dassault Systemes) είναι λογισμικό βελτιστοποίησης που χρησιμοποιεί τα συνήθη προγράμματα επίλυσης πεπερασμένων στοιχείων, όπως το ANSYS, το Abaqus ή το MSC Nastran. Όπως και τα άλλα προγράμματα, έτσι και αυτό προσφέρει λύσεις για διάφορα προβλήματα βελτιστοποίησης. Το συγκεκριμένο πρόγραμμα επίσης απευθύνεται κυρίως σε Μηχανολόγους Μηχανικούς. Μερικά παραδείγματα του φαίνονται παρακάτω:

- Κύριο πλαίσιο τουρμπίνας ανέμου: Η χρησιμοποίηση του προγράμματος για τον σχεδιασμό του πλαισίου οδήγησε σε ελαφρύτερη δομή με μάζα μειωμένη κατά 40% που ικανοποιεί τις στατικές και δυναμικές απαιτήσεις σχεδιασμού (Εικόνα 2.4).

Εικόνα 2.4: Κύριο πλαίσιο τουρμπίνας ανέμου

- Φάσεις βελτιστοποίησης μηχανολογικού εξαρτήματος (ψαλίδι ανάρτησης αυτοκινήτου)



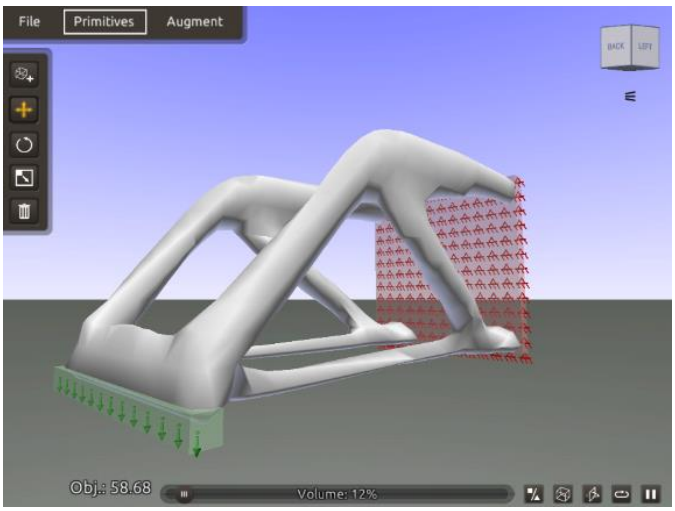
Εικόνα 2.5: Ψαλίδι ανάρτησης αυτοκινήτου

2.4.2 Ακαδημαϊκά λογισμικά πακέτα

TopOpt research group

Πρόκειται για μια διατμηματική ερευνητική ομάδα του DTU (Technical University of Denmark) και πιο συγκεκριμένα του τμήματος μηχανολόγων μηχανικών και του τμήματος εφαρμοσμένων μαθηματικών. Το πρόγραμμα ηγείται ο δανός καθηγητής Ole Sigmund από το τμήμα των μηχανολόγων μηχανικών, γνωστός για τις ερευνητικές εργασίες του στον τομέα της βελτιστοποίησης τοπολογίας. Αυτή η ερευνητική ομάδα έχει δημιουργήσει αρκετούς κώδικες και λογισμικά μερικά από τα οποία φαίνονται παρακάτω:

- Το 3D TopOpt App (Εικόνα 2.6) και το 2D TopOpt App είναι εφαρμογές που λύνουν ένα απλό πρόβλημα βελτιστοποίησης τοπολογίας τριών ή δύο διαστάσεων αντίστοιχα ελαχιστοποιώντας το έργο των δυνάμεων. Οι εφαρμογές επιτρέπουν στον χρήστη να αλλάξει τα φορτία, τις στηρίξεις και τον συνολικό όγκο (ή επιφάνεια αντίστοιχα) που θέλει να έχει η κατασκευή.



Εικόνα 2.6: 3D TopOpt App

- TopOpt Game: Είναι ένα παιχνίδι που επιτρέπει στον χρήστη να δοκιμάσει τις γνώσεις του λύνοντας προβλήματα δύο διαστάσεων. Ο χρήστης σχεδιάζει με το χέρι ή το ποντίκι την βέλτιστη λύση που θεωρεί και στην συνέχεια αυτή συγκρίνεται με το αποτέλεσμα που προκύπτει από τον υπολογιστή.

Centre for Innovative Structures and Materials (CISM)

Το Centre for Innovative Structures and Materials (CISM) είναι ένα ερευνητικό πρόγραμμα που εδράζεται στο πανεπιστήμιο Royal Melbourne Institute of Technology (RMIT). Το πρόγραμμα ηγείται ο καθηγητής Mike Xie, ειδικός σε θέματα υπολογιστικής μηχανικής και δομικής βελτιστοποίησης. Οι ερευνητές του προγράμματος έχουν αναπτύξει τα παρακάτω λογισμικά για βελτιστοποίηση τοπολογίας:

- BESO2D: Πρόκειται για ένα ανεξάρτητο πρόγραμμα βελτιστοποίησης τοπολογίας για κατασκευές δύο διαστάσεων χρησιμοποιώντας τους τελευταίους αλγόριθμους της μεθοδολογίας BESO. Το BESO2D δίνεται με ελεύθερη άδεια.

- BESO3D για Abaqus: Είναι ένα πρόγραμμα που χρησιμοποιεί το Abaqus για επίλυση προβλημάτων δύο και τριών διαστάσεων. Χρησιμοποιεί και αυτό τους τελευταίους αλγόριθμους της μεθοδολογίας BESO. Το CISM χορηγεί δωρεάν άδεια για έναν χρόνο.

- BESO3D για Rhinoceros: Αντίστοιχη λογική με το προηγούμενο πρόγραμμα μόνο που χρησιμοποεί το Rhinoceros για την επίλυση των προβλημάτων.

Οι αλγόριθμοι που χρησιμοποιούν τα προγράμματα παρουσιάζονται στο βιβλίο «*Evolutionary Topology Optimization of Continuum Structures: Methods and Applications* [12]».

Shape optimization group at CMAP

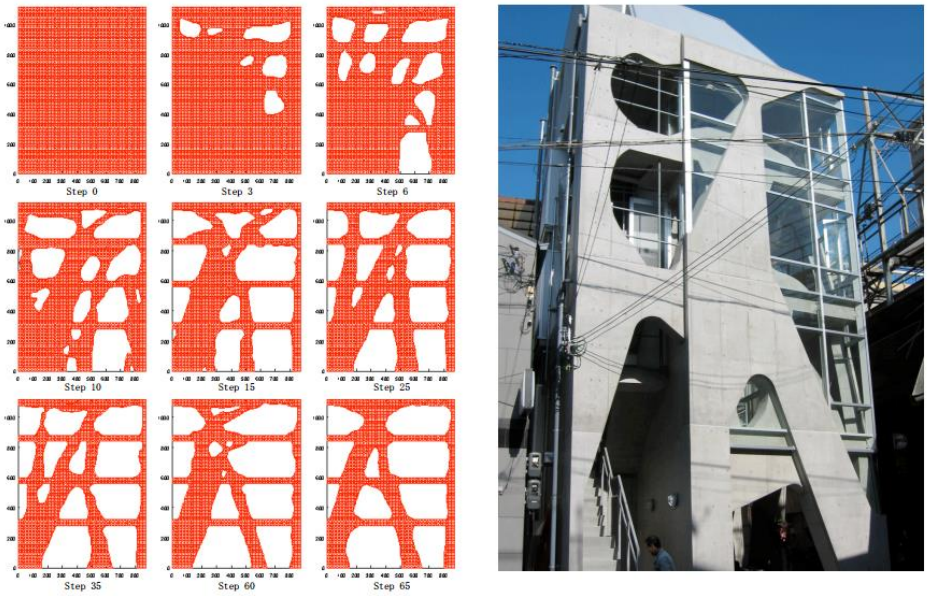
Πρόκειται για μια ερευνητική ομάδα του τμήματος εφαρμοσμένων μαθηματικών στο l'École Polytechnique που ασχολείται με προβλήματα βελτιστοποίησης σχήματος και τοπολογίας. Της προσπάθειας ηγείται ο καθηγητής Grégoire Allaire. Η ομάδα αυτή έχει αναπτύξει το λογισμικό FreeFem++ toolbox (γραμμένο με C++) για επίλυση προβλημάτων δύο διαστάσεων, χρησιμοποιώντας ρουτίνες που έχουν γράψει οι G.Allaire, B. Boutin, C. Dousset, O.Pantz. Επιπλέον, έχει αναπτύξει κώδικα στο scilab για προβλήματα δύο διαστάσεων βασισμένο στην μέθοδο level set.

2.5 Εφαρμογή βελτιστοποίησης τοπολογίας σε κατασκευές πολιτικού μηχανικού

Εικόνα 2.7: Qatar National Convention Centre (QNCC)

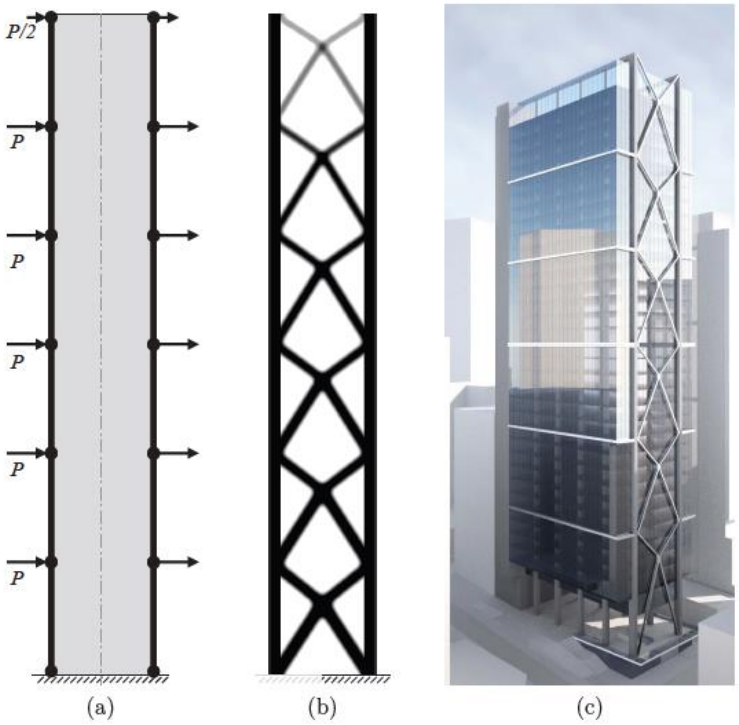
Η πιο γνωστή εφαρμογή βελτιστοποίησης τοπολογίας σε κατασκευές πολιτικού μηχανικού είναι το Qatar National Convention Centre (QNCC) που βρίκεται στην πόλη Ντόχα (Εικόνα 2.7). Σχεδιάστηκε από τον Ιάπωνα αρχιτέκτονα Arata Isozaki σε συνεργασία με τους αρχιτέκτονες RHWL σύμφωνα με τα «χρυσά» πρότυπα του U.S. Green Building Council's Leadership in Energy and Environment Design (LEED). Αυτό που παρουσιάζει μεγαλύτερο ενδιαφέρον είναι η στήριξη της οροφής από τεράστια μέλη που έχουν δενδροειδή μορφή, εικόνα που παραπέμπει στο ιερό Ισλαμικό δέντρο Sidrat al-Muntaha. Ο σχεδιασμός της τελικής λύσης ήταν ένα σύνθετο πρόβλημα βελτιστοποίησης τοπολογίας, αναζητώντας την βέλτιστη λύση που θα ικανοποιούσε και τα αισθητικά κριτήρια. Η βελτιστοποίηση αυτή έγινε από τον πολιτικό μηχανικό Buro Happold και την ομάδα του SMART (Smart Modeling Analysis Research Technologies).

Οι Ohmori et al [13] παρουσίασαν ένα άλλο παράδειγμα εφαρμογής βελτιστοποίησης τοπολογίας, που χρησιμοποιήθηκε για τον σχεδιασμό των τοίχων ενός κτιρίου στην Ιαπωνία. Στην εικόνα 2.8 παρουσιάζεται η διαδικασία βελτιστοποίησης που έγινε με την μέθοδο ESO και το τελικό αποτέλεσμα.



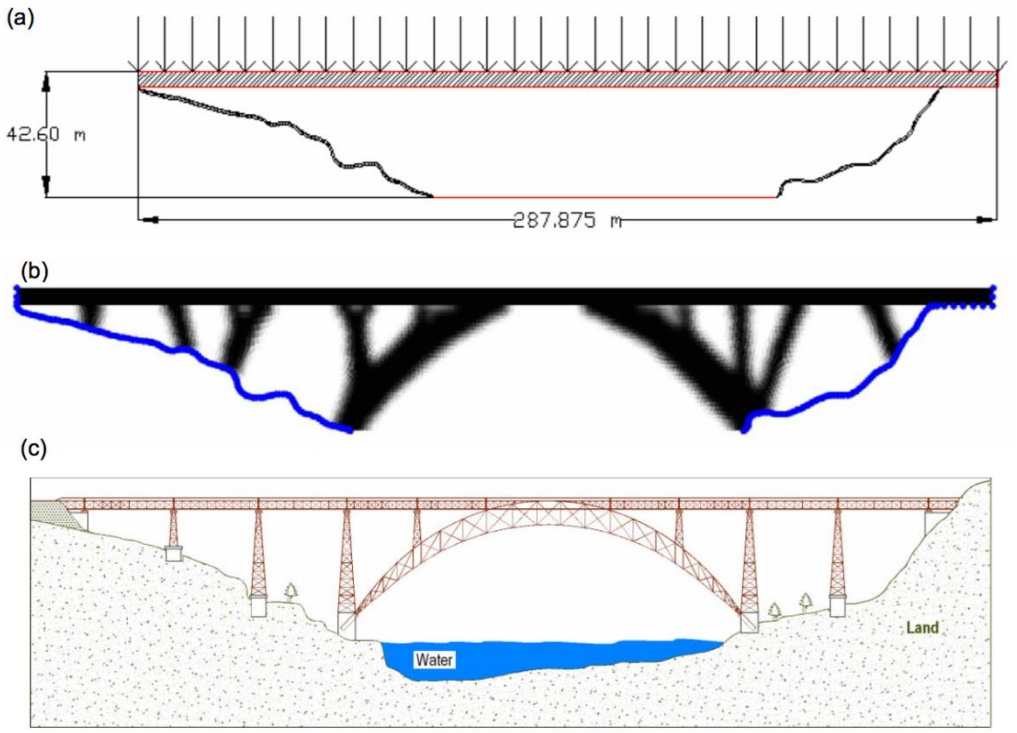
Εικόνα 2.8: Τοίχος κτιρίου στην Ιαπωνία

Η βελτιστοποίηση τοπολογίας μπορεί να εφαρμοστεί σε πολύ ψηλά κτίρια ώστε να δώσει μια πρώτη προσέγγιση που να οδηγήσει σε καλύτερο αρχιτεκτονικό και στατικό σχεδιασμό των κτιρίων. Με αυτό το θέμα έχει ασχοληθεί εκτενώς ο καθηγητής Glaucio H. Paulino. Η Lauren Lynne Beghini [14] στο διδακτορικό που έκανε υπό την επίβλεψη του Paulino, παρουσίασε διάφορα παραδείγματα. Στην εικόνα 2.9 φαίνεται ένα εξ' αυτών:



Εικόνα 2.9: (a) Προσομοίωση - (b) Βέλτιστη λύση - (c) Σχεδιασμός

Τέλος, αξίζει να αναφερθεί το άρθρο των M Meenakshi Sundaram and G K Ananthasuresh [15] για τον διάσημο αρχιτέκτονα Alexandre Gustave Eiffel .Ο Eiffel σχεδίασε κατασκευές με σκοπό να ικανοποιούν όχι μόνο τα αισθητικά κριτήρια αλλά και οικονομικά, χρησιμοποιώντας όσο το δυνατόν καλύτερα το υλικό. Έτσι, ουσιασικά πραγματοποιούσε μια βελτιστοποίηση τοπολογίας χωρίς να χρησιμοποιεί τις αυστηρά μαθηματικές μεθόδους που υπάρχουν σήμερα. Πιο συγκεκριμένα, οι αρθρογράφοι παρουσίασαν δύο παραδείγματα στα οποία ο Eiffel είχε προτείνει σχεδόν την βέλτιστη λύση, τον φημισμένο πύργο του Eiffel και την σιδηροδρομική γέφυρα Maria Pia πάνω από τον ποταμό Ντούρο στην Πορτογαλία. Μάλιστα, για την τελευταία δημιούργησαν μια προσομοίωση δύο διαστάσεων (Εικόνα 2.10.a) και κατέληξαν μέσω μαθηματικών αλγορίθμων στο βελτιστοποιημένο μοντέλο που φαίνεται στην εικόνα 2.10.b. Στην εικόνα 2.10.c φαίνονται οι ομοιότητες που υπάρχουν με το σχέδιο του Eiffel.



Εικόνα 2.10: (a) Προσομοίωση - (b) Βέλτιστη λύση - (c) Σχέδιο Eiffel

2.6 Αναφορές

[1] Rozvany G I N (2008) A critical review of established methods of structural topology optimization. Struct Optim.

[2] Rossow M P, Taylor J E (1973) A finite element method for the optimal design of variable thickness sheets. AIAA J 11:1566–1569.

[3] Bendsoe M P, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71:197–224.

[4] Bendsoe M P (1989) Optimal shape design as a material distribution problem. Struct Optim 1:193–202.

[5] Rozvany G I N, Zhou M, Birker T (1992) Generalized shape optimization without homogenization. Struct Optim 4:250–254.

[6] Rozvany G I N, Querin O M (2002a) Combining ESO with rigorous optimality criteria. Int J Veh Des 28:294–299.

[6] Rozvany G I N, Querin O M (2002b) Theoretical foundations of sequential element rejections and admissions (SERA) methods and their computational implementations in topology optimization. Proceedings of the 9th AIAA/ISSMO Symposium on Multidisc. Anal and Optim. (held in Atlanta, Georgia), AIAA, Reston, VA.

[7] Yang X Y, Xie Y M, Steven G P, Querin O M (1998) Bi-directional evolutionary structural optimization. Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium Multidisc Anal. Optim (St. Louis), pp 1449–1457.

[8] Wang M Y, Wang X , Guo D (2003) A level set method for structural topology optimization. Comput. Methods Appl. Mech. Eng. 192: 227–246.

[9] Fleury C Braibant V (1986) Structural optimization: a new dual method using mixed variables. International Journal for Numerical Methods in Engineering 23:409-428.

[10] Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. International Journal for Numerical Methods in Engineering 24:359-373.

[11] Christensen P W, Klarbing A (2009) Solid Mechanics and its Applications. An Introduction to Structural Optimization. Springer.

[12] Huang X, Xie Y M (2010) Evolutionary Topology Optimization of Continuum Structures: Methods and Applications. Wiley.

[13] Ohmori H, Futai H, Ijima T, Muto A and Hasagawa H (2005). “Application of Morphogenesis to Structural Design”. Proc. of Frontiers of Computational Science Symposium. Nagoya, Japan.

[14] Beghini L L (2013) Building science through topology optimization.

[15] Sundaram M M, Ananthasuresh G K (2009) Gustave Eiffel and his Optimal Structures

ΚΕΦΑΛΑΙΟ 3

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

3.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται το θεωρητικό υπόβαθρο που χρειάζεται για να γίνει η κατανόηση του προβλήματος της βελτιστοποίησης τοπολογίας. Στο πρώτο μέρος του κεφαλαίου παρουσιάζονται βασικές έννοιες της βελτιστοποίησης, όπως οι κατηγορίες προβλημάτων, μεθοδολογίες επίλυσης και το μαθηματικό μοντέλο. Στην συνέχεια, παρουσιάζονται βασικοί ορισμοί και αρχές του κυρτού προγραμματισμού, όπως οι συνθήκες ΚΚΤ και η Lagrangian Duality. Το θεωρητικό υπόβαθρο που παρουσιάζεται έγινε κατανοητό από την διδακτορική διατριβή του επιβλέποντα καθηγητή Ν. Λαγαρού [1] και από την διπλωματική εργασία του Γ.Καζάκη [2]. Πολύ σημαντική βοήθεια έδωσαν το βιβλίο που έγραψε ο Ν. Λαγαρός με τον Μ. Καρλαύτη «*Επιχειρησιακή έρευνα και βελτιστοποίηση για μηχανικούς*» [3], το βιβλίο «*An Introduction to structural optimization*» [4] αλλά και η διπλωματική εργασία του Σ.Μαργαρίτη [5].

3.2 Βασικές έννοιες της βελτιστοποίησης

3.2.1 Κατηγορίες γενικών προβλημάτων βελτιστοποίησης

Τα προβλήματα και οι αντίστοιχοι αλγόριθμοι βελτιστοποίησης μπορούν να χωριστούν σε κατηγορίες με βάση διάφορα κριτήρια:

- Ως προς την ύπαρξη περιορισμών: με περιορισμούς και χωρίς περιορισμούς.
- Ως προς την φύση των μεταβλητών: συνεχείς και διακριτές.

- Ως προς τον τρόπο αναζήτησης της πιθανής λύσης: ντετερμινιστικές, ημιστοχαστικές και στοχαστικές μέθοδοι.

3.2.2 Μαθηματικό μοντέλο βελτιστοποίησης

Σε ένα πρόβλημα βελτιστοποίησης κατασκευών συναντώνται πάντα οι παρακάτω όροι:

- Μεταβλητές Σχεδιασμού (s): Ονομάζονται οι παράμετροι οι οποίες όταν λάβουν συγκεκριμένη τιμή καθορίζουν πλήρως έναν σχεδιασμό. Κατά την διάρκεια της βελτιστοποίησης συνεχώς αλλάζουν. Συνήθως ως μεταβλητές σχεδιασμού θεωρούνται γεωμετρικά χαρακτηριστικά της κατασκευής, όπως το πάχος ενός στοιχείου, η διάμετρος μιας ράβδου κτλ.
- Αντικειμενική συνάρτηση F(s): Είναι μια συνάρτηση που εξαρτάται από τις μεταβλητές σχεδιασμού και για κάθε πιθανή τιμή τους (δηλαδή για κάθε πιθανό σχεδιασμό) επιστρέφει μια τιμή. Ως αντικειμενική συνάρτηση τίθεται το βασικό αντικείμενο της βελτιστοποίησης το οποίο μεγιστοποιείται ή ελαχιστοποιείται. Συνήθη παραδείγματα αντικειμενικών συναρτήσεων είναι το βάρος, οι μετακινήσεις σε κάποια διεύθυνση, οι απώλειες ενέργειας και το κόστος.
- Περιορισμοί $g(s)\geq 0$, $h(s)=0$: Κάθε απαίτηση του μηχανικού εισάγεται στο μαθηματικό μοντέλο βελτιστοποίησης με τη μορφή ανισοτήτων και ισοτήτων, οι οποίες ονομάζονται περιορισμοί και βάσει αυτών πραγματοποιείται ο έλεγχος περί του εφικτού ή μη του τρέχοντος σχεδιασμού. Οι περιορισμοί που συνήθως επιβάλλονται σε προβλήματα κατασκευών είναι των τάσεων και των μετατοπίσεων, των οποίων οι τιμές δεν επιτρέπεται να υπερβαίνουν κάποια καθορισμένα όρια.

Γενικά ένα πρόβλημα βελτιστοποίησης κατασκευών ενδέχεται να είναι συνεχές ή διακριτό, ανάλογα με το είδος του πεδίου τιμών των παραμέτρων σχεδιασμού. Λόγοι τυποποίησης πολλές φορές επιβάλλουν το πεδίο τιμών να είναι διακριτό. Το μαθηματικό μοντέλο ενός συνεχούς προβλήματος βέλτιστου σχεδιασμού μπορεί να διατυπωθεί ως εξής:

$$F(s)\rightarrow \min$$

$$s=\{s_1,s_2,...,s_n\}^T$$

s.t

$$l_i<s_i<U_i \text{ , i=1,2,…,n}$$

$$g_j(s)\geq 0 \text{ , j=1,2,..,m}$$

$$h_j(s)=0 \text{ , j=m+1,m+2,…,t}$$

Σε αντιστοιχία με το συνεχές πρόβλημα που διατυπώνεται παραπάνω, ένα διακριτό πρόβλημα βέλτιστου σχεδιασμού γράφεται:

$$F(s) \to \min$$
$$s = \{s_1, s_2, ..., s_n\}^T$$

s.t

$$l_i < s_i < U_i \text{ , i=1,2,…,n}$$

$$s_i \in \mathbb{R}^d \text{ , i=1,2,…,n}$$

$$g_j(s) \geq 0 \text{ , j=1,2,..,m}$$

$$h_j(s) = 0 \text{ , j=m+1,m+2,…,t}$$

Όπου $\mathbb{R}^d$ είναι το πεδίο τιμών των διακριτών μεταβλητών *s*. Οι μεταβλητές σχεδιασμού $s = \{s_1, s_2, ..., s_n\}^T$ μπορούν να λάβουν τιμές μόνο από το σύνολο τιμών $\mathbb{R}^d$

3.2.3 Μεθοδολογίες βελτιστοποίησης

Τα τελευταία χρόνια έχει αναπτυχθεί πλήθος μεθοδολογιών βελτιστοποίησης. Οι πρώτοι αλγόριθμοι βελτιστοποίησης που εφαρμόστηκαν σε δομικά προβλήματα ήταν δανεισμένοι από τους τομείς των οικονομικών, των μαθηματικών και της επιχειρησιακής έρευνας και βασίζονταν στον **μαθηματικό προγραμματισμό**. Οι τεχνικές βελτιστοποίησης που βασίζονται στις αρχές του μαθηματικού προγραμματισμού μπορούν γενικά να ταξινομηθούν σε πέντε μεγάλες κατηγορίες:

- Γραμμικός προγραμματισμός (Linear Programming – LP). Αντιμετωπίζει προβλήματα στα οποία τόσο η αντικειμενική συνάρτηση όσο και οι συναρτήσεις περιορισμών είναι γραμμικές συναρτήσεις των μεταβλητών σχεδιασμού. Σε προβλήματα αυτού του είδους ένα τοπικό ελάχιστο είναι οπωσδήποτε και καθολικό ελάχιστο του προβλήματος.
- Μη-Γραμμικός προγραμματισμός (Non Linear Programming – NLP). Είναι οι πιο διαδεδομένες τεχνικές μαθηματικού προγραμματισμού. Αντιμετωπίζουν γενικώς όλες τις περιπτώσεις όπου η αντικειμενική συνάρτηση αλλά και οι συναρτήσεις περιορισμού είναι μη-γραμμικές συναρτήσεις των μεταβλητών σχεδιασμού . Σε αυτή την περίπτωση η εύρεση ενός τοπικού ελαχίστου δεν πιστοποιεί την εύρεση ενός καθολικού ελαχίστου.
- Ακέραιος προγραμματισμός (Integer Programming – IP). Σε αυτή την περίπτωση οι μεταβλητές σχεδιασμού δεν είναι συνεχείς, αλλά παίρνουν διακριτές τιμές από κάποιο συγκεκριμένο σύνολο τιμών. Συνήθως οι μεταβλητές αυτές είναι ακέραιες, για αυτό και λέγεται ακέραιος προγραμματισμός. Επίσης υπάρχουν οι περιπτώσεις «μεικτού

ακέραιου προγραμματισμού» (mixed-integer programming), όπου κάποιες εκ των μεταβλητών σχεδιασμού είναι συνεχείς και οι υπόλοιπες διακριτές.

- Γεωμετρικός προγραμματισμός (Geometric Programming – GP). Ειδική περίπτωση όπου οι συναρτήσεις περιορισμού αλλά και η αντικειμενική συνάρτηση είναι πολυωνυμικής μορφής συναρτήσεις των μεταβλητών σχεδιασμού. Προϋπόθεση είναι οι μεταβλητές σχεδιασμού να λαμβάνουν πάντα θετικές τιμές.
- Δυναμικός προγραμματισμός (Dynamic Programming – DP). Κύριος στόχος αυτών των μεθόδων είναι να διασπαστεί ένα σχετικά μεγάλο πρόβλημα βελτιστοποίησης σε μικρότερα τα οποία μπορούν να αντιμετωπιστούν ως ξεχωριστά προβλήματα βέλτιστου σχεδιασμού. Κάθε υποπρόβλημα περιέχει μέρος από τα στοιχεία του καθολικού προβλήματος και μπορεί να επιλυθεί με κάποια από τις προαναφερθείσες μεθοδολογίες.

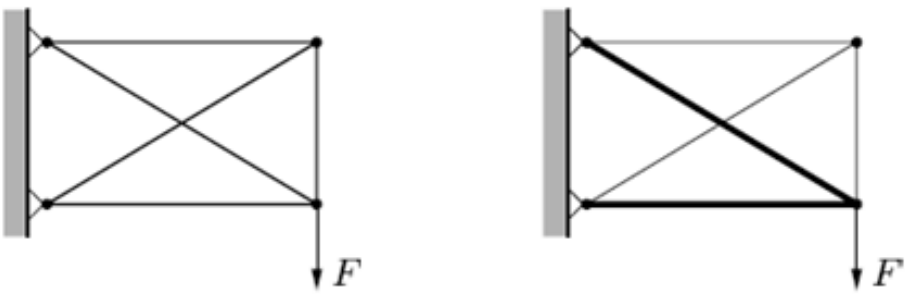
Εκτός των μαθηματικών μεθόδων βελτιστοποίησης, υπάρχουν και οι **μεταευριστικές (metaheuristics)** μέθοδοι, όπως είναι η μέθοδος των στατηγικών εξέλιξης (evolution strategies), των γενετικών αλγορίθμων (genetic algorithms), η μέθοδος της προσομοίωσης ανόπτησης (simulated annealing) κ.α.. Οι μεταευριστικές ή δαρβίνειες μέθοδοι οφείλουν την ονομασία τους στο γεγονός ότι μιμούνται τη διαδικασία της εξέλιξης των ειδών στη φύση, όπως την παρουσίασε πρώτος ο Κάρολος Δαρβίνος. Η εφαρμογή των μεθόδων αυτών αποδεικνύεται ιδιαίτερα αποτελεσματική σε ένα ευρύτερο πεδίο προβλημάτων σε σχέση με το πεδίο εφαρμογής των μεθόδων μαθηματικού προγραμματισμού.

3.2.4 Κατηγορίες προβλημάτων βελτιστοποίησης κατασκευών

Όσον αφορά τον τομέα των κατασκευών τα προβλήματα βελτιστοποίησης κατανείμονται σε 3 μεγάλες κατηγορίες:

Βελτιστοποίηση διατομών κατασκευής (sizing optimization)

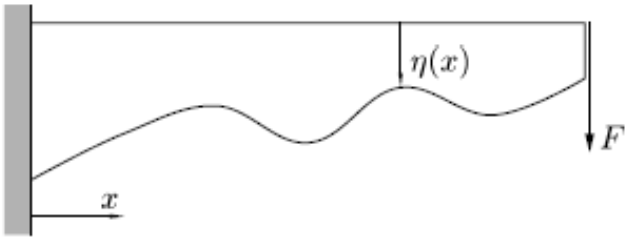
Στόχος των μεθόδων βελτιστοποίησης διατομών είναι η ελαχιστοποίηση ενός μεγέθους (π.χ βάρος, κόστος κτλ) σε κατασκευή συγκεκριμένου σχήματος και τοπολογίας λαμβάνοντας ως μεταβλητές σχεδιασμού διαστάσεις διατομών, πάχη, ιδιότητες υλικών.



Εικόνα 3.1: Παράδειγμα βελτιστοποίησης διατομών κατασκευής

Βελτιστοποίηση σχήματος (Shape optimization)

Στόχος των μεθόδων βέλτιστου σχεδιασμού σχήματος κατασκευών είναι ο καθορισμός του σχήματος διατομών ράβδων ή ολόσωμων κατασκευών για πιο οικονομική και λειτουργική χρήση λαμβάνοντας υπόψη διάφορους περιορισμούς που τίθενται για το συγκεκριμένο πρόβλημα.

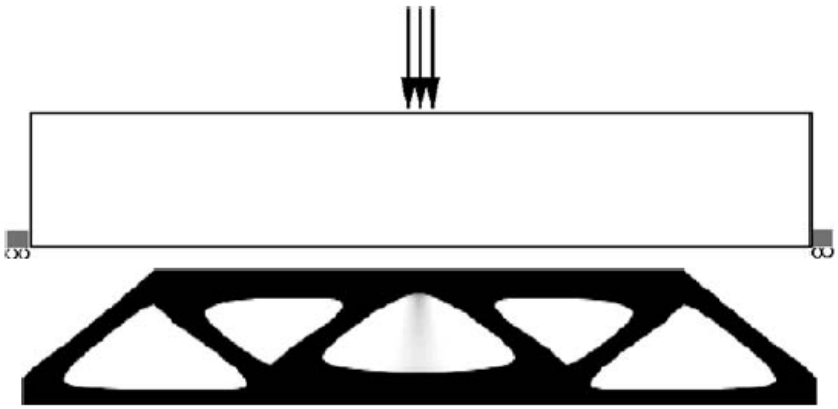


Εικόνα 3.2: Παράδειγμα βελτιστοποίησης σχήματος

Βελτιστοποίηση τοπολογίας (topology optimization)

Στον σχεδιασμό των κατασκευών είναι απαραίτητο να καθοριστεί μία όσον το δυνατόν καλύτερη "τοπολογία" (topology) ή "δομή" (layout) της κατασκευής, έτσι ώστε να είναι αποτελεσματικότερη στην ανάληψη των φορτίων και να δίνει οικονομικότερο σχεδιασμό. Στον όρο "δομή" της κατασκευής συμπεριλαμβάνεται κάθε είδους πληροφορία που αφορά στην τοπολογία, το σχήμα και το μέγεθος της κατασκευής. Ο βέλτιστος σχεδιασμός της δομής των κατασκευών αποσκοπεί στην εύρεση των δομών εκείνων στις οποίες η αντοχή και η ακαμψία της κατασκευής μεγιστοποιείται με το ελάχιστο δυνατό υλικό.

Οι αλγόριθμοι βελτιστοποίησης της τοπολογίας είναι εργαλεία που βοηθάνε τον μηχανικό να επιτύχει την αποτελεσματική χρήση των υλικών για να επιτύχει τον στόχο του. Αρχικά ορίζεται ο χώρος σχεδιασμού ή αναφοράς (reference domain), ο τύπος υλικού, οι συνθήκες στήριξης και οι φορτίσεις, έπειτα γίνεται η ανάλυση του φορέα και ακολουθεί η επαναληπτική διαδικασία που θα οδηγήσει στη βέλτιστη τοπολογία, δηλαδή στην καλύτερη δυνατή κατανομή υλικού στην κατασκευή.



Εικόνα 3.3: Παράδειγμα βελτιστοποίησης τοπολογίας

3.3 Κυρτός προγραμματισμός

3.3.1 Τοπικά και ολικά ελάχιστα

Ο μαθηματικός ορισμός του τοπικού και ολικού ελάχιστου δίνεται ως εξής:

Ένα σημείο s* στον χώρο σχεδιασμού θεωρείται **τοπικό ή σχετικό ελάχιστο** όταν πληροί τις συναρτήσεις περιορισμού και ισχύει η σχέση $F(s^*) \leq F(s)$ σε κάθε σημείο εφικτού σχεδιασμού σε μικρή ακτίνα γύρω από το σημείο s*. Εάν ισχύει μόνο η ανισότητα $F(s^*) < F(s)$, τότε το σημείο s* καλείται αυστηρό (strict) ή μοναδικό (unique) ή δυνατό (strong) τοπικό ελάχιστο.

Ένα σημείο s* στον χώρο σχεδιασμού θεωρείται **καθολικό ή απόλυτο ελάχιστο** όταν πληροί τις συναρτήσεις περιορισμού και ισχύει η σχέση $F(s^*) \leq F(s)$ σε κάθε σημείο εφικτού σχεδιασμού. Εάν ισχύει μόνο η ανισότητα $F(s^*) < F(s)$, τότε το σημείο s* καλείται αυστηρό (strict) ή μοναδικό (unique) ή δυνατό (strong) καθολικό ελάχιστο.

Στην περίπτωση που δεν υπάρχουν συναρτήσεις περιορισμού τότε ισχύουν οι ίδιοι ορισμοί, αλλά σε ολόκληρο τον χώρο σχεδιασμού και όχι μόνο στην περιοχή των εφικτών σχεδιασμών.

Για μια συνάρτηση f(x), όπου το x μπορεί να είναι ένας πίνακας με μεταβλητές, το διάνυσμα κλίσης της είναι:

$$\nabla f(x) = \begin{pmatrix} \dfrac{\partial f(x)}{dx_1} \\ \dfrac{\partial f(x)}{dx_2} \\ \vdots \\ \dfrac{\partial f(x)}{dx_n} \end{pmatrix} = \left[\dfrac{\partial f(x)}{dx_1} \; \dfrac{\partial f(x)}{dx_2} \; \cdots \; \dfrac{\partial f(x)}{dx_n} \right]^{\mathrm{T}}$$

Τα σημεία στα οποία η παράγωγος της συνάρτησης μηδενίζεται ονομάζονται στάσιμα σημεία (stationary points). Σε προβλήματα βελτιστοποίησης χωρίς περιορισμούς τα τοπικά ή ολικά ελάχιστα βρίσκοντα στα στάσιμα σημεία. Σε προβλήματα με περιορισμούς τα στάσιμα σημεία μπορεί να βρίσκονται εκτός των περιορισμών οπότε το ελάχιστο να είναι στο όριο των περιορισμών. Αντίστοιχα το ίδιο ισχύει και για τα μέγιστα.

3.3.2 Κυρτότητα

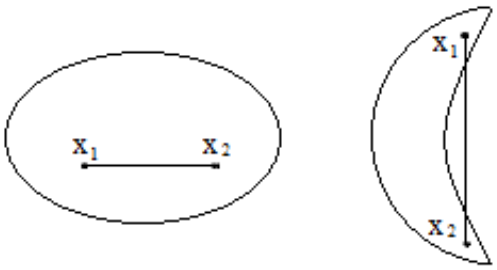
Όταν προσπαθούμε να εντοπίσουμε ένα βέλτιστο σημείο, πολλές φορές είναι χρήσιμο να γνωρίζουμε την μορφή της συνάρτησης. Παρακάτω παρουσιάζονται οι ορισμοί του κυρτού συνόλου και της κυρτής συνάρτησης.

Κυρτό Σύνολο

Ένα σύνολο Α είναι κυρτό αν για κάθε $\chi_1,\chi_2 \in A$ και για κάθε $\lambda\in(0,1)$ ισχύει:

$$\lambda\chi_1+(1-\lambda)\chi_2 \in A$$

δηλαδή ένα σύνολο είναι κυρτό αν όλα τα σημεία της γραμμής που ενώνει τα χ_1,χ_2, βρίσκονται εντός του συνόλου. Το σύνολο στην εικόνα 3.4.a είναι κυρτό ενώ το σύνολο στην εικόνα 3.4.b όχι.



Εικόνα 3.4: (a) Κυρτό - (b) Μη κυρτό σύνολο

Κυρτή Συνάρτηση

Μια συνάρτηση f: A→R είναι κυρτή (στο κυρτό σύνολο Α) αν για κάθε $\chi_1,\chi_2 \in A$ και για κάθε $\lambda\in(0,1)$ ισχύει:

$$f(\lambda\chi_1+(1-\lambda)\chi_2) \le f(\chi_1)+(1-\lambda)f(\chi_2)$$

Αν ισχύει μόνο η ανίσωση < τότε η f ονομάζεται αυστηρά κυρτή (strictly convex). Δηλαδή μια συνάρτηση είναι αυστηρά κυρτή αν μια ευθεία γραμμή μεταξύ δύο σημείων της συνάρτησης υπερεκτιμά παντού την τιμή της συνάρτησης. Στην εικόνα 3.5 παρουσιάζονται μια αυστηρά κυρτή, μια κυρτή και μια μη κυρτή συνάρτηση.

Εικόνα 3.5: (a) Αυστηρά κυρτή– (b) Κυρτή – (c) Μη κυρτή συνάρτηση

Πρόταση

Όταν μια συνάρτηση είναι κυρτή και ορίζεται σε κυρτό σύνολο τότε το τοπικό ελάχιστο αποτελεί και το ολικό ελάχιστο.

3.3.3 Υπολογισμός ελαχίστων κυρτών συναρτήσεων με περιορισμούς

Έστω το παρακάτω πρόβλημα υπολογισμού του ελάχιστου μιας συνάρτησης f μέσα στους περιορισμούς:

$$\begin{cases} \min\limits_{x} f(x) \\ \text{s.t} \quad g_i(x) \le 0 \qquad \text{i=0, … , l} \\ \qquad x_j^{\min} \le x_j \le x_j^{\max} \quad \text{j=0, … , n} \end{cases}$$

Αρχικά καθορίζουμε την συνάρτηση Λαγκράνζ (Lagrangian function), όπου το ελάχιστό της είναι το ελάχιστο της συνάρτησης f μέσα στους περιορισμούς:

$$L(x,\lambda) = f(x) + \sum_{i=0}^{l} \lambda_i g_i(x)$$

όπου τα λ_i ονομάζονται πολλαπλασιαστές Λαγκράνζ (Lagrange multipliers).

Το ελάχιστο της συνάρτησης Λαγκράνζ μπορεί να υπολογιστεί είτε με την βοήθεια των συνθηκών ΚΚΤ είτε κάνοντας χρήση της Lagrangian Duality.

KKT conditions

Το ελάχιστο της συνάρτησης του Λαγκράνζ είναι ο συνδυασμός των (x,λ) που ικανοποιούν τις συνθήκες των Karush-Kuhn-Tucker (KKT conditions), οι οποίες είναι:

$$\frac{\partial L(x,\lambda)}{\partial x_j} \le 0 \qquad \alpha\nu \quad x_j = x_j^{\max}$$

$$\frac{\partial L(x,\lambda)}{\partial x_j} \ge 0 \qquad \alpha\nu \quad x_j = x_j^{\min}$$

$$\frac{\partial L(x,\lambda)}{\partial x_j} = 0 \qquad \alpha\nu \quad x_j^{\min} \le x_j \le x_j^{\max}$$

$$\lambda_i g_i(x) = 0$$

$$g_i(x) \le 0$$

$$\lambda_i \ge 0$$

$$x_j^{\min} \le x_j \le x_j^{\max} \qquad \text{Όπου j=0, … , n και i=0, … , l}$$

Lagrangian Duality

Πολλές φορές είναι δύσκολο να βρεθεί λύση που να ικανοποιεί τις συνθήκες ΚΚΤ, ειδικά για πιο σύνθετα προβλήματα. Μια άλλη πιο βολική μέθοδος για την εύρεση του ελαχίστου της συνάρτησης Λαγκράνζ είναι η Lagrangian Duality:

$$\min_{x\in X}\max_{\lambda\geq 0} L(x,\lambda) \;=\; \min_{x\in X}\max_{\lambda\geq 0}\left\{ f(x)+\sum_{i=1}^{l}\lambda_i g_i(x)\right\}$$

Δηλαδή, αρχικά μεγιστοποιείται η συνάρτηση L ως προς λ με σταθερό x και στην συνέχεια ελαχιστοποιείται ως προς x.

3.3.4 Κυρτές προσεγγίσεις για υπολογισμό ελαχίστων μη κυρτών συναρτήσεων

Οι προηγούμενες μέθοδοι ισχύουν για κυρτές συναρτήσεις ορισμένες σε κυρτά σύνολα. Στην πράξη όμως, τα περισσότερα προβλήματα βελτιστοποίησης κατασκευών είναι μη κυρτά και επομένως δεν μπορούν επιλυθούν με αυτόν τον τρόπο. Τέτοια προβλήματα αντιμετωπίζονται αντικαθιστώντας το αρχικό σύνθετο πρόβλημα με μια ακολουθία απλών υποπροβλημάτων. Επειδή τα υποπροβλήματα που παράγονται είναι κυρτά και η λύση τους προσεγγίζει την λύση του αρχικού προβλήματος, τα ονομάζουμε κυρτές προσεγγίσεις. Οι πιο γνωστές μεθοδολογίες που παράγουν κυρτές προσεγγίσεις είναι η Convex Linearization (CONLIN), η Method of Moving Asymptotes (MMA), η Sequential Linear Programming (SLP) και η Sequential Quadratic Programming (SQP), οι οποίες παρουσιάζονται στο δεύτερο κεφάλαιο της παρούσας διπλωματικής.

Ο γενικός αλγόριθμος επίλυσης ενός μη κυρτού προβλήματος βελτιστοποίησης κατασκευών ακολουθεί τα εξής βήματα:

1. Αρχικοποιούνται οι μεταβλητές σχεδιασμού, δηλαδή ορίζεται ένας αρχικός σχεδιασμός x_0 για να ξεκινήσει η πρώτη επανάληψη.
2. Υπολογίζονται οι μετατοπίσεις $u(x_0)$ από την σχέση $K(x_0)u(x_0)=F(x_0)$
3. Υπολογίζεται η αντικειμενική συνάρτηση, οι συναρτήσεις περιορισμών και οι παράγωγοί τους για τον αρχικό σχεδιασμό x_0.
4. Με χρήση μιας εκ των μεθοδολογιών (CONLIN, MMA, SQP, SLP) το αρχικό πρόβλημα αντικαθίσταται από υποπροβλήματα - κυρτές προσεγγίσεις.
5. Επιλύονται τα κυρτά υποπροβλήματα με χρήση των συνθηκών ΚΚΤ ή της Lagrangian Duality και έτσι η διαδικασία καταλήγει σε έναν νέο σχεδιασμό x_1.
6. Η επαναληπτική διαδικασία συνεχίζεται μέχρι να ικανοποιηθεί το κριτήριο τερματισμού που έχει επιλεγεί.

3.4 Αναφορές

[1] Λαγαρός Ν (2000) Βελτιστοποίηση κατασκευών με χρήση εξελικτικών αλγορίθμων και νευρωνικών δικτύων, Αθήνα: ΕΜΠ

[2] Καζάκης Γ (2016) Βελτιστοποίηση τοπολογίας με χρήση μονάδων επεξεργασίας γραφικών σε προβλήματα πολιτικού μηχανικού: Θεώρηση μαζικών και θερμικών δράσεων σε δύο και τρεις διαστάσεις, Αθήνα: ΕΜΠ

[3] Καρλαύτης Μ, Λαγαρός Ν (2010) Επιχειρησιακή έρευνα και βελτιστοποίηση για μηχανικούς. Αθήνα: Συμμετρία

[4] Christensen P W, Klarbing A (2009) Solid Mechanics and its Applications. An Introduction to Structural Optimization. Springer

[5] Μαργαρίτης Σ (2015) Βελτιστοποίηση τοπολογίας σε κατασκευές πολιτικού μηχανικού: Ερμηνεία με μικτούς φορείς, Αθήνα: ΕΜΠ

ΚΕΦΑΛΑΙΟ 4

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ C# - APPLICATION PROGRAMMING INTERFACE ΤΟΥ SAP2000

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται τα μέσα που χρησιμοποιήθηκαν για την ανάπτυξη του λογισμικού βελτιστοποίησης τοπολογίας. Το πρώτο μέρος του κεφαλαίου περιέχει κάποιες γενικές έννοιες της γλώσσας προγραμματισμού C#. Στην συνέχεια αναλύονται πιο συγκεκριμένα εργαλεία της, όπως οι μεταβλητές, οι τελεστές, οι δομές επιλογής, οι δομές επανάληψης κτλ. Τα παραπάνω έγιναν κατανοητά από το βιβλίο «*Sams Teach Yourself the C# Language in 21 Days*» [1] καθώς και από το tutorial της ιστοσελίδας tutorialspoint [2] από το οποίο προέρχονται τα περισσότερα παραδείγματα και οι πίνακες. Στο δεύτερο μέρος του κεφαλαίου παρουσιάζεται το πρόγραμμα Visual Studio που χρησιμοποιήθηκε για να γραφεί ο κώδικας σε C# καθώς και ο τρόπος που συνδέεται αυτό με το στατικό πρόγραμμα SAP2000, που χρησιμοποιήθηκε για τις αναλύσεις. Περισσότερες πληροφορίες για το Visual Studio υπάρχουν στην ιστοσελίδα της Visual Studio [3] ενώ για το SAP2000 στην ιστοσελίδα της εταιρείας CSi America [4] αλλά και στην ιστοσελίδα της ACE-HELLAS [5], εκπρόσωπο της CSi America στην Ελλάδα.

4.2 Γλώσσα προγραμματισμού C#

4.2.1 Βασικές έννοιες της C#

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, δηλαδή οργανώνεται γύρω από αντικείμενα και όχι γύρω από διαδικασίες. Οι πιο σημαντικές έννοιες του αντικειμενοστραφή προγραμματισμού είναι η Ενθυλάκωση (Encapsulation), η Κληρονομικότητα (Inheritance) και ο Πολυμορφισμός (Polymorphism).

Ένθυλάκωση (Encapsulation)

Η ενθυλάκωση είναι η λογική δημιουργίας «πακέτων» που περιέχουν ότι χρειάζεται ο χρήστης. Η C# δίνει την δυνατότητα δημιουργίας κλάσεων (βλέπε 4.2.11 για καλύτερη κατανόηση των εννοιών κλάση και αντικείμενο) στις οποίες αποθηκεύονται όλα τα δεδομένα καθώς και οι ρουτίνες που διαχειρίζονται αυτά. Για παράδειγμα μπορεί να δημιουργηθεί μια κλάση με την ονομασία κύκλος, η οποία να δέχεται ως δεδομένα την ακτίνα και το κέντρο του κύκλου. Εκτός όμως από τα δεδομένα η κλάση έχει και όλες τις ρουτίνες - διαδικασίες που μπορεί να χρειαστούν σε έναν κύκλο όπως ο υπολογισμός της περιμέτρου, του εμβαδού κτλ. Έτσι δημιουργώντας αυτό το πακέτο απλοποιούνται τα πράγματα για τον χρήστη καθώς αυτός είναι υποχρεωμένος να δώσει μόνο τα δεδομένα και όχι να γνωρίζει το πως υπολογίζεται κάθε διαδικασία του κύκλου. Τέλος, η κλάση μπορεί να χρησιμοποιηθεί πολλές φορές για να δημιουργήσει διαφορετικά αντικείμενα. Δηλαδή χρησιμοποιώντας την κλάση κύκλος μπορούν να δημιουργηθούν πολλοί κύκλοι.

Πολυμορφισμός (Polymorphism)

Ο πολυμορφισμός είναι η ικανότητα ενός αντικειμένου να παίρνει πολλές μορφές. Αυτό, σημαίνει οτι ο χρήστης μπορεί να καλέσει ένα αντικέμενο ή μια διαδικασία με παραπάνω από έναν τρόπους. Στο παράδειγμα του κύκλου, ο χρήστης για να πάρει το εμβαδό μπορεί να καλέσει την αντίστοιχη ρουτίνα δίνοντας είτε το κέντρο και την ακτίνα του είτε τρία σημεία της περιφέρειάς του. Σε μια γλώσσα που δεν είναι αντικειμενοστραφής η παραπάνω διαδικασία χρειάζεται δύο διαφορετικές ρουτίνες. Στην C# χρειάζεται πάλι δύο ρουτίνες με την διαφορά ότι μπορούν να έχουν το ίδιο όνομα. Το πρόγραμμα αποφασίζει ποιά ρουτίνα από τις δύο θα χρησιμοποιήσει ανάλογα με τα δεδομένα που θα δεχτεί.

Κληρονομικότητα (Inheritance)

Η κληρονομικότητα είναι πιο πολύπλοκη έννοια. Προηγουμένως παρουσιάστηκε η δυνατότητα δημιουργίας μιας κλάσης με το όνομα κύκλος η οποία περιέχει όλες τις απαραίτητες υπολογιστικές ρουτίνες. Μία άλλη κλάση με όνομα σφαίρα έχει όλα τα χαρακτηριστικά του κύκλου συν την τρίτη διάσταση. Η σφαίρα για παράδειγμα μπορεί να κληρονομήσει τις διαδικασίες του κύκλου. Αυτή η δυνατότητα ονομάζεται κληρονομικότητα.

4.2.2 Μεταβλητές (Variables)

Οι μεταβλητές είναι ονομασίες που δίνονται σε αποθηκευτικούς χώρους. Κάθε μεταβλητή έχει συγκεκριμένο τύπο (βλέπε 4.2.4 Τύποι δεδομένων), ο οποίος καθορίζει το μέγεθος της μνήμης που δεσμεύεται, το εύρος τιμών που μπορεί να πάρει η μεταβλητή και το είδος των διεργασιών που μπορούν να εφαρμοστούν σε αυτή.

Ο καθορισμός μεταβλητών (Defining variables) έχει την παρακάτω σύνταξη:

```
<data type> <variable_name>;

Examples:     int counter;
              double grade;
              char name;
              bool test_bool;
```

Η αρχικοποίηση μεταβλητών (Initializing variables) γίνεται χρησιμοποιώντας το σύμβολο της ισότητας (=)

```
<variable_name> = value;

Examples:     counter = 0;
              grade = 19.5;
              name = "Nikos"
              bool = true;
```

Ακόμα, δίνεται η δυνατότητα η αρχικοποίηση να γίνει κατά την δήλωση:

```
<data type> <variable_name> = value;

Examples:     int counter = 0;
              double grade = 19.5;
              char name = "Nikos";
              bool test_bool = true;
```

Η ονομασία των μεταβλητών πρέπει να ακολουθεί τους παρακάτω κανόνες:

- Το όνομα μπορεί να περιέχει γράμματα, αριθμούς και την κάτω πάυλα (_).
- Ο πρώτος χαρακτήρας του ονόματος πρέπει να είναι γράμμα. Μπορεί να είναι και κάτω παύλα αλλά δεν συνίσταται.
- Η C# έχει ευαισθησία στα κεφαλαία γράμματα, δηλαδή τα ονόματα count και Count αναφέρονται σε διαφορετικές μεταβλητές.
- Υπάρχουν συγκεκριμένες λέξεις keywords που δεν μπορούν να χρησιμοποιηθούν ως μεταβλητές (π.χ do , if , else κτλ).

4.2.3 Σταθερές (Constants)

Οι σταθερές αντιμετωπίζονται ακριβώς όπως οι κανονικές μεταβλητές με την διαφορά ότι η τιμή τους δεν μπορεί να αλλάξει μετά την δήλωσή τους. Οι σταθερές όπως και οι μεταβλητές μπορούν να είναι οποιουδήποτε τύπου δεδομένων.

4.2.4 Τύποι δεδομένων (Data Types)

Στην C# υπάρχουν οι παρακάτω τύποι δεδομένων:

Type	Represents	Range
bool	Boolean value	True or False
byte	8-bit unsigned integer	0 to 255
char	16-bit Unicode character	U+0000 to U+ffff
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0 \text{ to } 28}$
double	64-bit double – precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647
long	64-bit signed integer type	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
sbyte	8-bit signed integer type	-128 to 127
short	16-bit signed integer type	-32,768 to 32,767
uint	32-bit unsigned integer type	0 to 4,294,967,295
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615
ushort	16-bit unsigned integer type	0 to 65,535

4.2.5 Τελεστές (Operators)

Οι τελεστές (operators) είναι σύμβολα που χρησιμοποιούνται για να καταλάβει ο υπολογιστής ποια μαθηματική ή λογική πράξη θα εκτελέσει. Χωρίζονται στις εξής κατηγορίες:

- Αριθμητικοί τελεστές (Arithmetic operators)
- Σχεσιακοί τελεστές (Relational operators)
- Λογικοί τελεστές (Logical operators)

Αριθμητικοί τελεστές (Arithmetic operators)

Έστω δύο ακέραιες μεταβλητές A=10 και B=5.

Operator	Description	Example
+	Adds two operands	A + B = 15
-	Subtracts second operand from the first	A – B = 5
*	Multiplies both operands	A * B = 50
/	Divides numerator by de-numerator	A / B = 2
%	Modulus Operator and remainder of after an integer division	A % B = 0
++	Increment operator increases integer value by one	A++ = 11
--	Decrement operator decreases integer value by one	A-- = 9

Σχεσιακοί τελεστές (Relational Operators)

Έστω δύο ακέραιες μεταβλητές A=10 και B=5.

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	A == B is false
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	A != B is true
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true	A > B is true
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	A < B is false
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	A >= B is true
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	A <= B is false

Λογικοί τελεστές (Logical Operators)

Έστω δύο λογικές μεταβλητές C=True και D=False.

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	C && D is false
||	Called Logical OR Operator. If any of the two operands is non zero then condition becomes true.	C || D is true
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	! (C && D) is true

4.2.6 Δομές Επιλογής (Decision making / Selection statements)

if statement:

Η δομή if επιτρέπει να εκτελεστεί μια εντολή ή ένα κομμάτι εντολών, αναλόγως με το αν ικανοποιείται μια συνθήκη. Ως συνθήκη μπορούμε να ορίσουμε μια οποιαδήποτε λογική έκφραση. Η σύνταξή της είναι η παρακάτω:

```
If   (boolean_expression)
{
   /* statement(s) will execute if the expression is true */
}
```

If…else if…else statement

Εάν ο χρήστης θέλει να ελέγξει παραπάνω από μία συνθήκες τότε μπορεί να χρησιμοποιήσει συνεχόμενα if-else-if. Σύνταξη:

```
If (boolean_expression 1)
{
   /* Executes when the Boolean expression 1 is true */
}
else if (boolean_expression 2)
{
   /* Executes when the Boolean expression 2 is true */
}
else
{
   /* Executes when none of the above conditions are true */
}
```

Switch statement

Η δομή switch επιτρέπει στο πρόγραμμα να εκτελέσει διαφορετικά κομμάτια κώδικα, ανάλογα με την τιμή μιας μεταβλητής. Η σύνταξή της είναι η εξής:

```
switch (expression)
{
   case constant-expression:
       statement(s);
       break;

   case constant-expression:
       statement(s);
       break;

   case constant-expression:
       statement(s);
       break;

   /*you can have as any number of case statements */

   default: /* Optional */
       statement(s);
       break;
}
```

- Στην δομή switch μια έκφραση (συνήθως απλά μεταβλητή) ελέγχεται αν είναι ίση με μια λίστα από τιμές.
- Η έκφραση πρέπει να έχει ακέραιο ή αριθμητικό τύπο και οι τιμές με τις οποίες ελέγχεται πρέπει να είναι ίδιου τύπου με αυτή.
- Όταν ικανοποιείται σε κάποια περίπτωση η ισότητα, τότε εκτελείται το αντίστοιχο κομμάτι εντολών.
- Όταν το πρόγραμμα φτάσει σε εντολή «break» η δομή switch τερματίζεται και το πρόγραμμα συνεχίζει με την αμέσως επόμενη εντολή μετά την δομή switch.
- Μπορούν να υπάρχουν πολλές περιπτώσεις.
- Μπορεί να υπάρχει στο τέλος η περίπτωση «default», έτσι ώστε να εκτελεστεί ένα κομμάτι κώδικα αν καμία άλλη περίπτωση δεν αληθεύει.

4.2.7 Δομές Επανάληψης (Loops / Iteration statements)

While loop

Η δομή αυτή επαναλαμβάνει ορισμένες εντολές όσο μία συνθήκη είναι αληθής. Πρώτα εξετάζεται η συνθήκη. Αν είναι αληθής εκτελούνται οι εντολές μέσα στον βρόγχο και στην συνέχεια επανεξετάζεται η συνθήκη. Η διαδικασία τερματίζεται όταν η συνθήκη γίνει λανθασμένη. Σύνταξη:

```
while (condition)
{
   statement(s);
}
```

Do…While Loop

Έχει αντίστοιχη λειτουργία με την προηγούμενη δομή με την διαφορά ότι εδώ ο έλεγχος της συνθήκης βρίσκεται στο τέλος του βρόγχου οπότε οι εντολές μέσα στον βρόγχο εκτελούνται τουλάχιστον μια φορά. Σύνταξη:

```
do
{
   statement(s);
} while (condition)
```

For loop

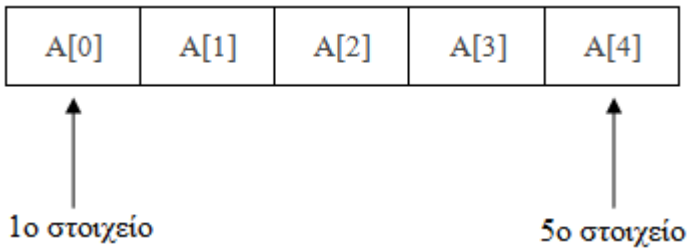
Η for loop είναι μια δομή που επαναλαμβάνει ένα κομμάτι εντολών συγκεκριμένες φορές.

```
for ( init; condition; increment )
{
   statement(s);
}
```

- Αρχικά στο βήμα «init» δηλώνεται η μεταβλητή και γίνεται η αρχικοποίησή της.
- Στην συνέχεια ελέγχεται η συνθήκη (condition): Αν είναι αληθής εκτελούνται οι εντολές μέσα στον βρόγχο. Αν είναι ψευδής ο βρόγχος τερματίζεται και η ροή του προγράμματος συνεχίζει στην αμέσως επόμενη εντολή μετά τον βρόγχο.
- Αν εκτελεστούν οι εντολές στο κυρίως σώμα, τότε η ροή επιστρέφει στο βήμα increment όπου αυξάνεται η τιμή της μεταβλητής κατά το βήμα. Στην συνέχεια ελέγχεται η συνθήκη.
- Η διαδικασία επαναλαμβάνεται για όσο η συνθήκη είναι αληθής.

4.2.8 Πίνακες (Arrays)

Ο πίνακας είναι μια στατική δομή δεδομένων που αποθηκεύει μέσα του δεδομένα του ίδιου τύπου. Όλοι οι πίνακες αποτελούνται από συνεχόμενες θέσεις μνήμης, οι οποίες δίνουν την δυνατότητα στον υπολογιστή να διαχειριστεί καλύτερα τα δεδομένα του πίνακα. Η μικρότερη διεύθυνση αντιστοιχεί στο πρώτο στοιχείο και η μεγαλύτερη στο τελευταίο. Έστω ένας πίνακας Α, πέντε στοιχείων:



Δήλωση πινάκων (Declaring arrays)

Ένας πίνακας δηλώνεται χρησιμοποιώντας την παρακάτω σύνταξη:

```
datatype [] ArrayName;

example: int [] A;
```

Αρχικοποίηση πινάκων (Initializing arrays)

Η δήλωση ενός πίνακα δεν σημαίνει ότι αρχικοποιείται στην μνήμη. Οι πίνακες είναι τύποι αναφοράς (reference type), οπότε χρησιμοποιείται η λέξη new για να δημιουργηθεί ο πίνακας.

```
datatype [] ArrayName = new datatype [fixed number];

example: int [] A = new int [5];
```

Τοποθέτηση τιμών σε πίνακες

Υπάρχουν διάφοροι τρόποι να τοποθετηθούν τιμές σε έναν πίνακα. Κατ' αρχάς, ο χρήστης μπορεί να τοποθετήσει τιμές σε κάθε θέση ξεχωριστά μετά την αρχικοποίηση του πίνακα.

```
int [] A = new int [6];

A[0]=13;
A[1]=2;
A[2]=5;
A[3]=29;
A[4]=18;
```

Ακόμα, ο χρήστης μπορεί να τοποθετήσει τιμές κατά τη διάρκεια της δήλωσης ή της αρχικοποίησης.

```
int [] A {13,2,5,29,18};

int [] A = new int [6] {13,2,5,29,18};
```

Πρόσβαση στα στοιχεία των πινάκων

Ο χρήστης μπορεί να έχει πρόσβαση στα στοιχεία ενός πίνακα χρησιμοποιώντας το όνομα του πίνακα και τον αριθμό της θέσης του στοιχείου μέσα στις αγκύλες [ ]. Η παρακάτω γραμμή κώδικα προσθέτει όλα τα στοιχεία ενός πίνακα.

```
int sumA = A[0] + A[1] + A[2] + A[3] + A[4];
```

Δισδιάστατοι πίνακες (two-dimensional arrays)

Ένας δισδιάστατος πίνακας είναι ουσιαστικά μια λίστα από πολλούς μονοδιάστατους. Μπορεί να θεωρηθεί ως ένας πίνακας όπου έχει x σειρές και y στήλες: Έστω ένας πίνακας B διαστάσεων 3x5:

	στήλη 0	στήλη 1	στήλη 2	στήλη 3	στήλη 4
σειρά 0	B[0,0]	B[0,1]	B[0,2]	B[0,3]	B[0,4]
σειρά 1	B[1,0]	B[1,1]	B[1,2]	B[1,3]	B[1,4]
σειρά 2	B[2,0]	B[2,1]	B[2,2]	B[2,3]	B[2,4]

Η δήλωση και αρχικοποίηση των δισδιάστατων πινάκων καθώς και η πρόσβαση στα στοιχεία τους γίνεται με αντίστοιχο τρόπο με τους μονοδιάστατους πίνακες:

```
datatype [,] ArrayName;

example: int [,] B;

datatype [,] ArrayName = new datatype [fixed number,fixed number];

example: int [,] B = new int [3,5];


int [,] B = new int [3,5]

{{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15}}
```

4.2.9 Μέθοδοι (Methods)

Η μέθοδος είναι ένα ξεχωριστό κομμάτι κώδικα που περιέχει εντολές για την επίτευξη κάποιας διαδικασίας. Η μέθοδος έχει δικό της όνομα και μπορεί να χρησιμοποιηθεί πολλές φορές. Για να χρησιμοποιηθεί μια μέθοδος πρέπει αρχικά να έχει καθοριστεί και στην συνέχεια να την καλεί το πρόγραμμα.

Καθορισμός μεθόδων (Defining methods)

```
<Access Specifier> <Return Type> <Method Name> (Parameter list)
{
        Statement(s);
}
```

Access Specifier: Καθορίζει την δυνατότητα πρόσβασης στην μέθοδο από μια άλλη κλάση ή από ένα άλλο πρόγραμμα.

Return type: Οι μέθοδοι συνήθως επιστρέφουν τιμές χρησιμοποιώντας στον κώδικά τους την λέξη κλειδί «return» ακολουθούμενη από μια μεταβλητή. Στο κομμάτι <Return Type> ορίζεται ο τύπος δεδομένου που θα επιστρέφει η μέθοδος. Αν μια μέθοδος δεν επιστρέφει καμία τιμή, τότε ο τύπος ορίζεται ως «void».

Method Name: Το όνομα της μεθόδου πρέπει να είναι ξεχωριστό και προτείνεται να έχει κάποια σχέση με την διαδικασία που εκτελεί η μέθοδος. Για παράδειγμα στην κλάση κύκλος μπορεί να υπάρχει μια μέθοδος που να υπολογίζει το εμβαδόν του και να ονομαστεί «CalculateArea».

Parameter list: Στην parameter list δηλώνονται οι παράμετροι που χρησιμοποιούνται για να περαστούν δεδομένα από και προς την μέθοδο. Υπάρχουν τρεις διαφορετικοί τρόποι με τους οποίους μπορεί να επιτευχθεί αυτό:

- By value (π.χ int a): Όταν καλείται η μέθοδος, δημιουργείται καινούργια θέση μνήμης για κάθε τέτοια παράμετρο. Οι τιμές των πραγματικών μεταβλητών του προγράμματος αντιγράφονται στις παραμέτρους. Έτσι, όποια αλλαγή γίνει στις παραμέτρους της μεθόδου δεν επηρεάζει τις αρχικές μεταβλητές.
- By reference (π.χ ref int a): Μια τέτοια παράμετρος αναφέρεται σε μια υπάρχουσα θέση μνήμης. Έτσι, όταν καλείται η μέθοδος δεν δημιουργείται καινούργια θέση μνήμης για κάθε παράμετρο. Οι παράμετροι αντιπροσωπεύουν τις αρχικές μεταβλητές και οποιαδήποτε αλλαγή γίνει στις παραμέτρους επηρεάζει και τις μεταβλητές.
- By Output (π.χ out int a): Σε μια μέθοδο η λέξη κλειδί «return» χρησιμοποιείται για να επιστραφεί μόνο μία τιμή. Ο χρήστης μπορεί να επιστρέψει παραπάνω από μία τιμές χρησιμοποιώντας τις παραμέτρους «output».

4.2.10 Κλάσεις (Classes)

Οι κλάσεις είναι το πιο σημαντικό στοιχείο μιας αντικειμενοστραφούς γλώσσας προγραμματισμού, όπως η C#. Μια κλάση αποτελεί προσχέδιο για την δημιουργία αντικειμένων, καθορίζει δηλαδή από τι θα αποτελούνται τα αντικείμενα που φτιάχνονται από αυτή την κλάση και τι διαδικασίες μπορούν να γίνουν σε αυτά. Προσοχή, όταν καθορίζεται η κλάση δεν δημιουργούνται αντικείμενα αλλά μόνο το πρότυπο σύμφωνα με το οποίο θα δημιουργηθούν τα αντικείμενα.

Καθορισμός κλάσεων

Ο καθορισμός μιας κλάσης ξεκινά με την λέξη κλειδί «class» ακολουθούμενη από το όνομα της κλάσης. Ο κώδικας μέσα στις αγκύλες αποτελεί το κύριο μέρος και περιέχει τα μέλη της κλάσης (μέθοδοι και μεταβλητές).

```
<access specifier> class  class_name
{
   // member variables
   <access specifier> <data type> variable1;
   ...
   <access specifier> <data type> variableN;
   // member methods
   <access specifier> <return type> method1(parameter_list)
   {
       // method body
   }
   ...
   <access specifier> <return type> methodN(parameter_list)
   {
       // method body
   }
}
```

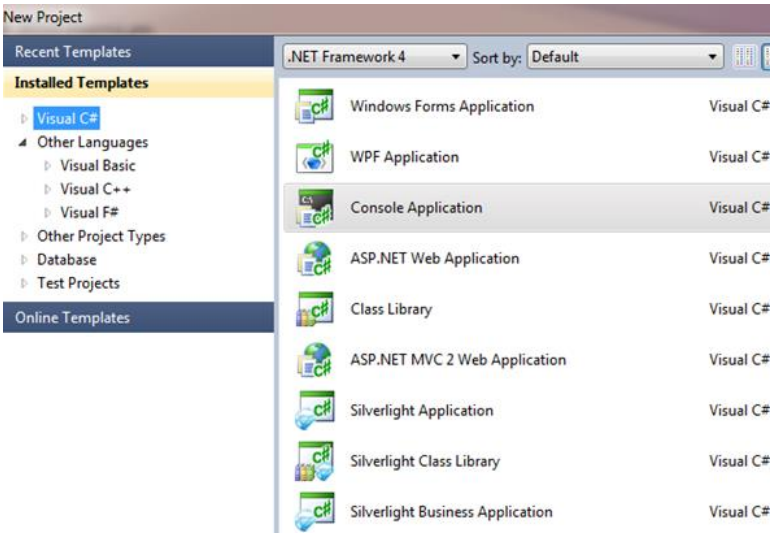
- Access Specifier: Καθορίζει την δυνατότητα πρόσβασης στις μεθόδους της κλάσης αλλά και στην ίδια την κλάση από μία άλλη κλάση ή από ένα άλλο πρόγραμμα.
- Η πρόσβαση στα μέλη της κλάσης γίνεται χρησιμοποιώντας τον τελεστή τελεία (.)
- Η τελεία συνδέει το όνομα του αντικειμένου με το όνομα του μέλους.
- Return type: ορίζει τον τύπο δεδομένου που θα επιστρέφει η μέθοδος.
- Data type: ορίζει τον τύπο δεδομένου της μεταβλητής.

Παρακάτω παρατίθεται ένα ολοκληρωμένο παράδειγμα για το πως καθορίζεται μια κλάση και τα μέλη της, πως δημιουργούνται αντικείμενα της κλάσης και πως γίνεται η πρόσβαση στα μέλη της κλάσης.

```
using System;
namespace BoxApplication
{
   class Box  //Defining class Box
   {
      //variables
      private double length;
      private double breadth;
      private double height;
      //Methods
      public void setLength( double len )
      {
         length = len;
      }

      public void setBreadth( double bre )
      {
         breadth = bre;
      }

      public void setHeight( double hei )
      {
         height = hei;
      }

      public double getVolume()
      {
         return length * breadth * height;
      }
   }
   class Boxtester
   {
      static void Main(string[] args)
      {
         Box Box1 = new Box();   // Declare Box1 of type Box
         double volume;

         // box 1 specification
         Box1.setLength(6.0);
         Box1.setBreadth(7.0);
         Box1.setHeight(5.0);

         // volume of box 1
         volume = Box1.getVolume();
         Console.WriteLine("Volume of Box1 : {0}" ,volume);
      }
   }
}
```

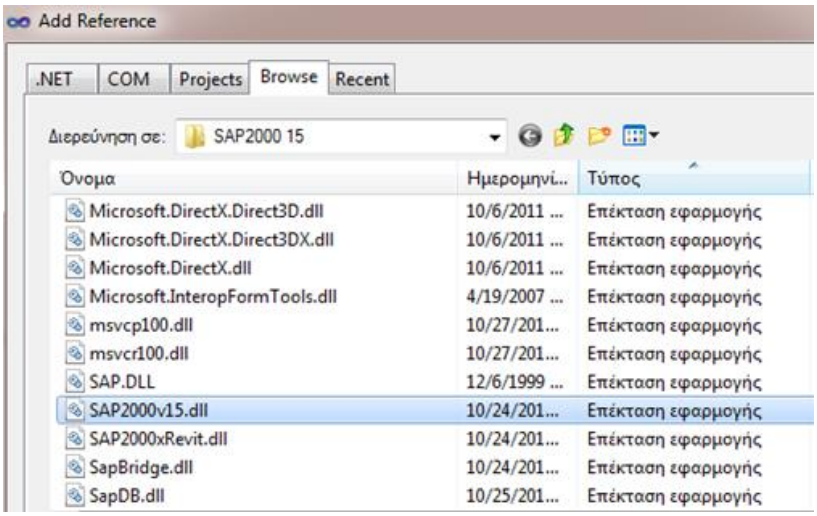
4.3 Visual Studio / SAP2000 OAPI

Η πλατφόρμα που χρησιμοποιήθηκε για να γραφεί ο κώδικας είναι το Visual Studio 2010 της εταιρείας Microsoft. Το Visual Studio παρέχει την δυνατότητα δημιουργίας διαφόρων ειδών εφαρμογών και υποστηρίζει πολλές γλώσσες προγραμματισμού όπως C#, Visual Basic, C++, F#, TypeScript, Python κ.α. Στην παρούσα διπλωματική δημιουργήθηκε εφαρμογή «Console Application» σε γλώσσα προγραμματισμού C#, όπως φαίνεται στην εικόνα 4.1:



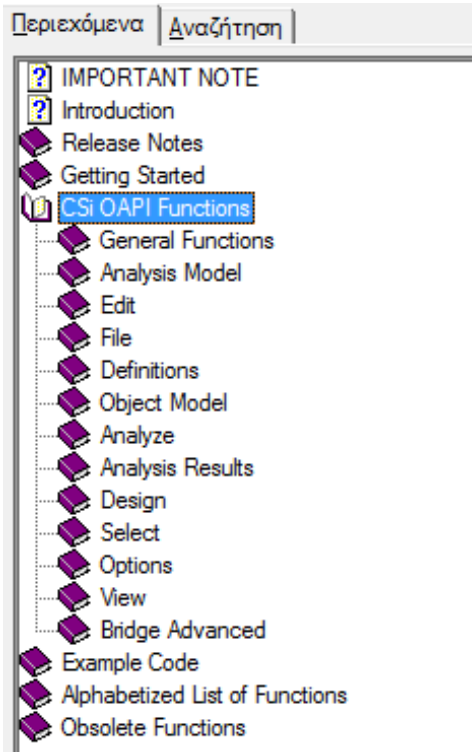
Εικόνα 4.1: Δημιουργία εφαρμογής στο Visual Studio

Το στατικό πρόγραμμα που χρησιμοποιήθηκε για τις αναλύσεις είναι το SAP2000. Το **Open Application Programming Interface (OAPI)** επιτρέπει στο SAP2000 να συνδεθεί με προγράμματα όπως το Visual Studio για την δημιουργία σύνθετων εφαρμογών. Αυτή η σύνδεση επιτυγχάνεται προσθέτοντας ως αναφορά στο Visual Studio το αρχείο SAP2000v15.dll με τον εξής τρόπο: Project→Add Reference→Εικόνα 4.2.

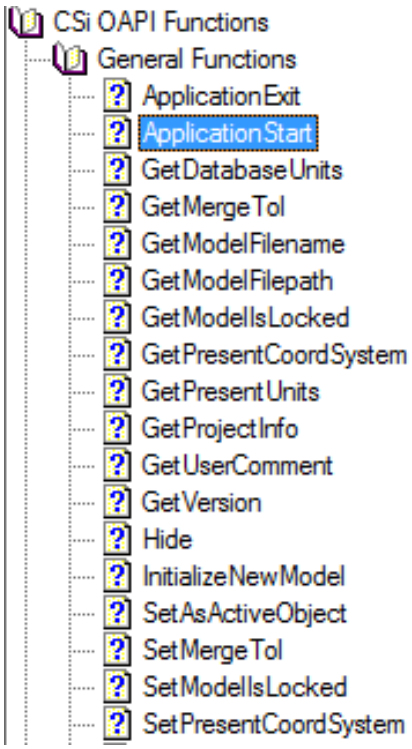


Εικόνα 4.2: Παράθυρο που εμφανίζεται για το add reference

Από την στιγμή που προστίθεται η παραπάνω αναφορά, η επικοινωνία του Visual Studio με το SAP2000 γίνεται μέσω ειδικών συναρτήσεων που προσφέρει το SAP2000 OAPI. Ο χρήστης μέσω των συναρτήσεων δίνει εντολή στο SAP2000 να εκτελέσει μια συγκεκριμένη διαδικασία όπως ακριβώς μπορεί να κάνει και χειροκίνητα. Το πλεονέκτημα που αποκτά με αυτόν τον τρόπο είναι ότι μπορεί να προγραμματίσει μια διαδικασία να γίνει πολλές φορές, ή να γίνει εφόσον ικανοποιούνται ορισμένες συνθήκες και να μην χρειάζεται κάθε φορά ένας άνθρωπος να χειρίζεται το πρόγραμμα χειροκίνητα. Οι συναρτήσεις αυτές είναι συγκεντρωμένες σε βιβλιοθήκες όπως φαίνεται στην εικόνα 4.3.



Εικόνα 4.3: Βιβλιοθήκες με συναρτήσεις



Εικόνα 4.4: Παράδειγμα για την ApplicationStart

Για παράδειγμα για να ξεκινήσει μια εφαρμογή SAP2000 ο χρήστης ψάχνει στις γενικές συναρτήσεις και βρίσκει την συνάρτηση ApplicatonStart (Εικόνα 4.4). Πατώντας πάνω στην συνάρτηση εμφανίζονται πληροφορίες για το πώς συντάσσεται (Εικόνα 4.5) και για το τι κάνει (Εικόνα 4.6).

Syntax

SapObject.ApplicationStart

VB6 Procedure

Function ApplicationStart(Optional ByVal Units As eUnits = kip_in_F, Optional ByVal Visible As Boolean = True, Optional ByVal FileName As String = "")

Εικόνα 4.5: Σύνταξη της συνάρτησης ApplicationStart

Remarks

This function starts the Sap2000 application.

When the model is not visible it does not appear on screen and it does not appear in the Windows task bar.

If no filename is specified, you can later open a model or create a model through the API.

Εικόνα 4.6: χρησιμότητα της συνάρτησης ApplicationStart

4.4 Αναφορές

[1] Bradley L. Jones (2004) Sams Teach Yourself the C# Language in 21 Days, USA: Sams Publishing

[2] http://www.tutorialspoint.com/csharp/

[3] https://www.visualstudio.com/

[4] https://www.csiamerica.com/

[5] http://www.ace-hellas.gr/

ΚΕΦΑΛΑΙΟ 5

ΛΟΓΙΣΜΙΚΟ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΤΟΠΟΛΟΓΙΑΣ

5.1 Εισαγωγή

Τα προβλήματα βελτιστοποίησης τοπολογίας που αποτέλεσαν αντικείμενο αυτής της διπλωματικής είναι σε συνεχή συστήματα (Distributed Parameter Systems). Στο πρώτο μέρος του κεφαλαίου παρουσιάζεται η μορφή των παραπάνω προβλημάτων, δηλαδή ποια είναι η αντικειμενική συνάρτηση, ποιες οι μεταβλητές σχεδιασμού και ποιοι οι περιορισμοί. Το **λογισμικό βελτιστοποίησης τοπολογίας** που αναπτύχθηκε δίνει την δυνατότητα επίλυσης αυτών των προβλημάτων με δύο διαφορετικές μεθοδολογίες. Στο υποκεφάλαιο 5.3 παρουσιάζονται τα θεωρητικά στοιχεία καθώς και ο τρόπος υλοποίησης της μεθοδολογίας «Optimality Criteria», ενώ στο υποκεφάλαιο 5.4 παρουσιάζονται τα αντίστοιχα στοιχεία της μεθοδολογίας «Method of Moving Asymptotes». Στην συνέχεια, περιγράφεται αναλυτικά σε βήματα το λογισμικό που αναπτύχθηκε και παρουσιάζεται σε μορφή διαγράμματος ροής.

5.2 Προβλήματα βελτιστοποίησης κατασκευών - συνεχή συστήματα

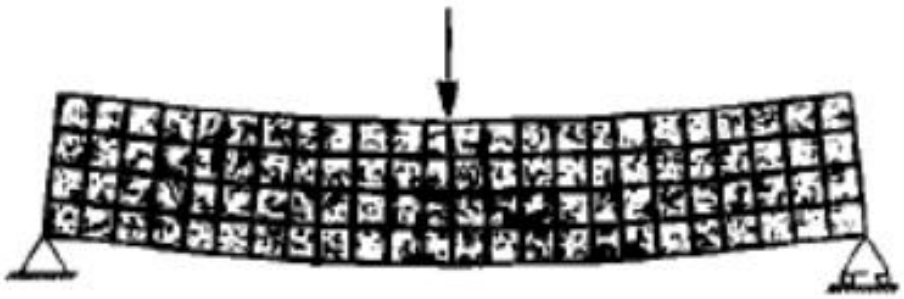
Σε ένα πρόβλημα βελτιστοποίησης τοπολογίας σε συνεχές σύστημα ορίζεται ένας χώρος, τα σημεία στήριξης και οι ασκούμενες δυνάμεις και ζητείται για ένα δεδομένο ποσοστό του χώρου, να καθοριστεί το σχήμα που πρέπει να πάρει η κατασκευή ώστε να παραλαμβάνει τα φορτία με τον καλύτερο δυνατό τρόπο. Στην εικόνα 5.1 φαίνεται ένα παράδειγμα συνεχούς συστήματος:

Εικόνα 5.1: Πρόβλημα συνεχούς συστήματος

Για την αντιμετώπιση του προβλήματος πρέπει πρώτα να γίνει διακριτοποίηση της κατασκευής με τη μέθοδο των πεπερασμένων στοιχείων. Σε κάθε στοιχείο ορίζεται ένα μέγεθος που δείχνει αν το στοιχείο έχει υλικό ή όχι. Το μέγεθος αυτό ονομάζεται πυκνότητα και συμβολίζεται με x. Η μεταβλητή x κάθε στοιχείου μπορεί να πάρει τιμές από μηδέν μέχρι ένα:

$$0 \le x_e \le 1 \text{για κάθε } x_e \in \Omega$$

Μηδέν σημαίνει ότι το στοιχείο πρακτικά δεν έχει καθόλου υλικό, ενώ ένα σημαίνει ότι το στοιχείο είναι πλήρες. Οι μεταβλητές x_e ανάλογα με την τιμή τους ορίζουν έναν σχεδιασμό και αποτελούν τις **μεταβλητές σχεδιασμού του προβλήματος βελτιστοποίησης.**



Εικόνα 5.2: Διακριτοποίηση

Η επιβολή των φορτίων στους φορείς δημιουργεί παραμορφώσεις με συνέπεια τα φορτία να παράγουν έργο. Η **αντικειμενική συνάρτηση f** που θα ελαχιστοποιηθεί είναι το έργο από τις δυνάμεις που ασκούνται στην κατασκευή και συμβολίζεται με C.

$$C=F^T u$$

όπου F^T είναι το ανάστροφο μητρώο δυνάμεων και U είναι το μητρώο των μετακινήσεων.

Επειδή F=K U η προηγούμενη σχέση μπορεί να γραφεί: $C=u^T K u$

Οι **συναρτήσεις περιορισμού** παίρνουν την μορφή:

$$\int_{\Omega} x\, d\Omega = \sum_{e=1}^{n} x_e a_e = V$$

όπου Ω είναι ο χώρος που δίνεται, a_e είναι ο όγκος ενός πεπερασμένου στοιχείου και V είναι το συνολικό επιτρεπόμενο ποσοστό του όγκου που έχει δοθεί ως δεδομένο.

Επομένως το πρόβλημα έχει την μορφή:

$$\begin{cases} \min C=F^T u \\ \text{s.t} \begin{cases} \sum_{e=1}^{n} x_e\, a_e = V \\ 0 \le x_e \le 1 \quad \gamma\iota\alpha \quad e=1,\ldots,n \end{cases} \end{cases}$$

Solid Isotropic Material with Penalization (SIMP)

Όταν η δυσκαμψία του υλικού μοντελοποιείται να εξαρτάται γραμμικά από την πυκνότητα σύμφωνα με τη σχέση ρ=s, οι βέλτιστες λύσεις που προκύπτουν αποτελούνται από πολλά γκρίζα στοιχεία, δηλαδή στοιχεία με τιμή πυκνότητας μεταξύ μηδέν και ένα. Τέτοιες λύσεις δεν βοηθάνε σε προβλήματα βελτιστοποίησης τοπολογίας, όπου είναι επιθυμητό κάθε στοιχείο είτε να έχει πλήρες υλικό (1), είτε καθόλου (0). Λύσεις που να πλησίαζουν πιο κοντά στα 0 και 1, μπορούν να επιτευχθούν αν ποινικοποιηθούν τα γκρι στοιχεία σύμφωνα με την σχέση:

$$\rho = s^{1/p} \quad p>1$$

Η παραπάνω σχέση εφαρμόζεται αλλάζοντας το μέτρο ελαστικότητας του υλικού του κάθε πεπερασμένου στοιχείου σύμφωνα με την σχέση:

$$E_e(x_e) = x_e^P E_e$$

Έτσι το μητρώο δυσκαμψίας του κάθε στοιχείου γίνεται:

$$K_e = x_e^P K_e^0$$

όπου K_e^0 είναι το μητρώο δυσκαμψίας του στοιχείου για πυκνότητα x=1. Στην παρούσα διπλωματική εργασία εφαρμόστηκε η προτεινόμενη τιμή p=3.

5.3 Επίλυση του προβλήματος με την μέθοδο Optimality Criteria (OC)

5.3.1 Θεωρητικά στοιχεία της OC

Το πρόβλημα επιλύεται με επαναληπτική διαδικασία. Αρχικά, επιλέγονται κάποιες πυκνότητες x^k. Στην συνέχεια, περιγράφεται η διαδικασία με την οποία επιλέγονται οι επόμενες πυκνότητες:

Για να υπολογιστεί η παράγωγος της αντικειμενικής συνάρτησης εφαρμόζεται η εξής σχέση:

$$C(x) = F^T u(x) - \lambda(K(x)u(x) - F)$$

Έτσι η παράγωγος ως προς x_e είναι:

$$\frac{\partial C(x)}{\partial x_e} = F^{\mathrm{T}} \frac{\partial u_e(x)}{\partial x_e} - \lambda \frac{\partial K_e(x)}{\partial x_e} u_e(x) - \lambda K_e(x) \frac{\partial u_e(x)}{\partial x_e}$$

$$\frac{\partial C(x)}{\partial x_e} = -\lambda \frac{\partial K_e(x)}{\partial x_e} u_e(x) + (F^{\mathrm{T}} - \lambda K_e(x)) \frac{\partial u_e(x)}{\partial x_e}$$

Για $\lambda=u(x)^{\mathrm{T}}$ η παράσταση $F - \lambda K_e(x)=0$, επομένως η παράγωγος του έργου ως προς x_e παίρνει την μορφή:

$$\frac{\partial C(x)}{\partial x_e} = -u_e(x)^T \frac{\partial K_e(x)}{\partial x_e} u_e(x) \ <0$$

Στην συνέχεια γραμμικοποιείται η αντικειμενική συνάρτηση C και χρησιμοποιώντας την αλλαγή μεταβλητής $y_e = x_e^{-a}$ ($\alpha >0$) καταλήγουμε στην εξής σχέση:

$$C(x) = C(x^k) + \sum_{e=1}^{n} \left. \frac{\partial C(x^k)}{\partial y_e} \right|_{x=x^k} (y_e - y_e{}^k) = C(x^k) + y_e \sum_{e=1}^{n} \left. \frac{\partial C(x^k)}{\partial y_e} \right|_{x=x^k} - y_e{}^k \sum_{e=1}^{n} \left. \frac{\partial C(x^k)}{\partial y_e} \right|_{x=x^k}$$

Όμως: $$\frac{\partial C}{\partial y_e} = \frac{\partial C}{\partial x_e} \frac{\partial x_e}{\partial y_e} = \frac{\partial C}{\partial x_e} \frac{1}{\frac{\partial y_e}{\partial x_e}} = \frac{\partial C}{\partial x_e} \frac{1}{\frac{\partial x_e^{-a}}{\partial x_e}} = -\frac{x_e^{1+a}}{a} \frac{\partial C}{\partial x_e}$$

Άρα η συνάρτηση C(x) παίρνει την μορφή:

$$C(x) = C(x^k) + \sum_{e=1}^{n} b_e{}^k x_e^{-a} + y_e{}^k \left(\frac{x_e^{1+a}}{a} \frac{\partial C}{\partial x_e}\right) \qquad \text{όπου} \quad b_e{}^k = -\frac{x_e^{1+a}}{a} \frac{\partial C}{\partial x_e} > 0$$

Ο πρώτος και ο τρίτος όρος είναι σταθερές ποσότητες που εξαρτώνται από τον πίνακα x^k των προηγούμενων πυκνοτήτων. Ο δεύτερος όρος είναι μεταβλητή ποσότητα που παίρνει πάντα θετικές τιμές. Κάθε βήμα επανάληψης πρέπει να οδηγεί σε μικρότερο έργο επομένως αρκεί η ελαχιστοποίηση του δεύτερου όρου. Αυτή είναι και η ουσία της μεθόδου optimality criteria καθώς τελικά επιλέγεται να ελαχιστοποιηθεί ένα μέρος της αντικειμενικής συνάρτησης σύμφωνα με κάποιο κριτήριο. Έτσι ορίζεται το υποπρόβλημα:

$$\begin{cases} \min \sum_{e=1}^{n} b_e^{\,k} x_e^{-a} \\ \text{s.t} \begin{cases} \sum_{e=1}^{n} x_e\, a_e = V \\ 0 \le x_e \le 1 \quad \gamma\alpha \quad e = 1,\ldots,n \end{cases} \end{cases}$$

Για να ελαχιστοποιηθεί η $\sum_{e=1}^{n} b_e^{\,k} x_e^{-a}$ αρκεί να ελαχιστοποιηθεί η Lagrangian function:

$$L(x,\lambda) = \sum_{e=1}^{n} b_e^{\,k} x_e^{-a} + \lambda(x^T \alpha_e - V)$$

Το πρόβλημα είναι κυρτό και μπορεί να εφαρμοστεί η Lagrangian duality: Αρκεί να βρεθεί το ελάχιστο ως προς x και μετά το μέγιστο ως προς λ.

Το ελάχιστο ως προς x προκύπτει στο σημείο μηδενισμού της παραγώγου:

$$\frac{\partial L}{\partial x_e} = -a b_e^{\,k} x_e^{-a-1} + \lambda a_e = 0 \Leftrightarrow x_e = \left(\frac{a b_e^{\,k}}{\lambda a_e}\right)^{\frac{1}{1+a}}$$

Το x_e πρέπει να βρίσκεται μέσα στα όρια: $0 \le x_e \le 1$. Επομένως αν η λύση ξεπερνά τις τιμές των άκρων το x_e παίρνει την ακριανή τιμή.

$$x_e = \begin{cases} 0 & \alpha\nu \quad \left(\frac{a b_e^{\,k}}{\lambda a_e}\right)^{\frac{1}{1+a}} \le 0 \\ \left(\frac{a b_e^{\,k}}{\lambda a_e}\right)^{\frac{1}{1+a}} & \alpha\nu \quad 0 < \left(\frac{a b_e^{\,k}}{\lambda a_e}\right)^{\frac{1}{1+a}} < 1 \\ 1 & \alpha\nu \quad \left(\frac{a b_e^{\,k}}{\lambda a_e}\right)^{\frac{1}{1+a}} \ge 1 \end{cases}$$

Το μέγιστο ως προς λ βρίσκεται παραγωγίζοντας την $L(x,\lambda)$ ως προς λ για δεδομένο x_e και εξισώνοντας την με μηδέν.

$$\frac{\partial L}{\partial \lambda} = \sum_{e=1}^{n} \left(x_e \alpha_e - V \right) = 0$$

Αυτή η διαδικασία αποτελεί μια εσωτερική επανάληψη για τον υπολογισμό του κατάλληλου λ. Υπολογίζονται τα x_e για κάθε λ και ελέγχεται αν ισχύει η σχέση:

$$\sum_{e=1}^{n} \left(x_e \alpha_e \right) = V$$

Η εσωτερική επανάληψη συνεχίζεται μέχρι να βρεθεί το λ για το οποίο ισχύει η παραπάνω σχέση. Τα x_e που προέκυψαν για αυτό το λ θα αποτελέσουν τις νέες πυκνότητες της επαναληπτικής διαδικασίας που λύνει το αρχικό πρόβλημα βελτιστοποίησης τοπολογίας. Η διαδικασία θα τερματιστεί όταν ικανοποιηθεί το κριτήριο τερματισμού που έχει επιλεγεί.

5.3.2 Υλοποίηση της OC

Παρακάτω παρουσιάζονται κάποια επιπλέον στοιχεία για το πως υλοποιήθηκε η μέθοδος OC στον κώδικα.

Ο υπολογισμός της παραγώγου της αντικειμενικής συνάρτησης ως προς κάθε στοιχείο x_e σύμφωνα με τα θεωρητικά στοιχεία γίνεται σύμφωνα με την σχέση:

$$\frac{\partial C(x)}{\partial x_e} = -u_e(x)^T \frac{\partial K_e(x)}{\partial x_e} u_e(x)$$

Το SAP2000 δεν δίνει τον πίνακα δυσκαμψιών $K_e(x)$ οπότε η σχέση πρέπει να μετασχηματιστεί. Σύμφωνα με την SIMP αλλάζοντας το μέτρο ελαστικότητας του υλικού του κάθε πεπερασμένου στοιχείου το μητρώο δυσκαμψίας του κάθε στοιχείου γίνεται:

$$K_e(x_e) = x_e^3 K_e^0 \quad \Leftrightarrow \quad K_e^0 = \frac{K_e(x_e)}{x_e^3}$$

$$\frac{\partial K_e(x)}{\partial x_e} = 3x_e^2 K_e^0 = 3x_e^2 \frac{K_e(x_e)}{x_e^3} = 3\frac{K_e(x_e)}{x_e}$$

Και επομένως η σχέση μετασχηματίζεται:

$$\frac{\partial C(x)}{\partial x_e} = -u_e(x)^T \frac{\partial K_e(x)}{\partial x_e} u_e(x) = -u_e(x)^T \, 3\frac{K_e(x_e)}{x_e} u_e(x) = -3\frac{F^T u_e(x)}{x_e} = -\frac{3C_e}{x_e}$$

Αφού υπολογιστούν οι παράγωγοι, στην συνέχεια εφαρμόζεται μια εσωτερική επανάληψη κατά την οποία υπολογίζονται τα καινούργια x_e για κάθε λ και ελέγχεται αν ισχύει η σχέση:

$$\sum_{e=1}^{n}(x_e)=V$$

Σύμφωνα με τα θεωρητικά στοιχεία και για α_e=1, α=1, η σχέση που προκύπτει για την εύρεση της καινούργιας πυκνότητας κάθε στοιχείου, είναι η εξής:

$$x_e=\begin{cases}0 & \alpha\nu \quad (\frac{b_e^{\,k}}{\lambda})^{\frac{1}{2}}\le 0\\ (\frac{b_e^{\,k}}{\lambda})^{\frac{1}{2}} & \alpha\nu \quad 0<(\frac{b_e^{\,k}}{\lambda})^{\frac{1}{2}}<1\\ 1 & \alpha\nu \quad (\frac{b_e^{\,k}}{\lambda})^{\frac{1}{2}}\ge 1\end{cases} \qquad \text{όπου} \quad b_e^{\,k}=-x_e^{\,2}\frac{\partial C}{\partial x_e}$$

Για να υπολογιστεί πιο γρήγορα το λ για το οποίο ισχύει η σχέση $\sum_{e=1}^{n}(x_e)=V$, εφαρμόζεται η μέθοδος της διχοτόμησης διαστήματος. Ορίζεται ένα διάστημα (a,b) με τις τιμές που μπορεί να πάρει το λ και υπολογίζονται τα x_e για $\lambda=\frac{a+b}{2}$. Στην συνέχεια ελέγχεται η σχέση. Αν το άθροισμα των x_e είναι μεγαλύτερο από τον όγκο V, αυτό σημαίνει ότι τα x_e πρέπει να μειωθούν, άρα το λ να αυξηθεί. Έτσι προκύπτει ότι το λ βρίσκεται στο διάστημα $(\frac{a+b}{2},b)$. Αντίστοιχα, αν το άθροισμα των x_e είναι μικρότερο του όγκου V, τα x_e πρέπει να αυξηθούν άρα το λ βρίσκεται στο διάστημα $(a,\frac{a+b}{2})$. Η διχοτόμηση συνεχίζεται μέχρι να βρεθεί το κατάλληλο λ.

Τα x_e που προκύπτουν τελικά από την προηγούμενη επαναληπτική διαδικασία , αποτελούν τις νέες πυκνότητες με τις οποίες θα ξεκινήσει η επόμενη επανάληψη. Η διαδικασία τερματίζεται όταν ικανοποιηθεί το εξής κριτήριο: Υπολογίζεται για κάθε στοιχείο η διαφορά της καινούργιας πυκνότητας x_e με την παλιά. Όταν η μέγιστη διαφορά όλων των στοιχείων είναι μικρότερη του 0.001, δηλαδή όταν όλα τα x_e αλλάζουν ελάχιστα, τότε η διαδικασία τερματίζεται. Επειδή το παραπάνω κριτήριο μπορεί να αργήσει να ικανοποιηθεί ,η διαδικασία μπορεί να τερματιστεί και με κριτήριο τον αριθμό των επαναλήψεων.

5.4 Επίλυση του προβλήματος με την Method of Moving Asymptotes (MMA)

5.4.1 Θεωρητικά στοιχεία της MMA

Έστω ένα γενικό πρόβλημα βελτιστοποίησης:

$$\mathrm{P:}\begin{cases}\min f_o(x) \qquad (x\in\mathbb{R}^n)\\ \mathrm{S.t}\begin{cases} f_i(x)\le \bar{f}_i \quad \gamma\iota\alpha\ i=0,\ldots,m\\ \underline{x}_j\le x_j\le \bar{x}_j \quad \gamma\iota\alpha\ j=1,\ldots,n\end{cases}\end{cases}$$

Η MMA ακολουθεί την παρακάτω γενική προσέγγιση για επίλυση τέτοιου είδους προβλημάτων:

- Βήμα 0: Η διαδικασία αρχίζει επιλέγοντας ένα αρχικό $x^{(0)}$ και ξεκινώντας την επανάληψη k=0.
- Βήμα 1: Στην k επανάληψη γίνεται υπολογισμός των $f_i(x^{(k)})$ και των παραγώγων $\nabla f_i(x^{(k)})$ για i=0,…,n.
- Βήμα 2: Δημιουργία του υποπροβλήματος $P^{(k)}$ αντικαθιστώντας στο πρόβλημα P, τις συναρτήσεις f_i με τις προσεγγιστικές συναρτήσεις $f_i^{(k)}$ οι οποίες βασίζονται στους υπολογισμούς του βήματος 1.
- Βήμα 3: Επίλυση του $P^{(k)}$ και υπολογισμός του $x^{(k+1)}$. Επιστροφή στο βήμα 1.
- Η επαναληπτική διαδικασία τερματίζεται όταν ικανοποιηθεί το κριτήριο τερματισμού που έχει επιλεγεί.

Για να οριστεί όμως, πρέπει να περιγραφούν **πως καθορίζονται οι συναρτήσεις** $\boldsymbol{f_i^{(k)}}$

Στην k επανάληψη δίνεται ο πίνακας $x^{(k)}$ των μεταβλητών σχεδιασμού. Επιλέγονται οι παράμετροι $L_j^{(k)}$ και $U_j^{(k)}$ για κάθε j=1,…,n ώστε:

$$L_j^{(k)}\le x_j^{(k)}\le U_j^{(k)}$$

Η διαδικασία επιλογής των παραμέτρων θα αναλυθεί στην συνέχεια.

Για κάθε i=0,….,m η συνάρτηση $f_i^{(k)}$ καθορίζεται ως εξής:

$$f_i^{(k)}(x) = r_i^{(k)} + \sum_{j=1}^{n}\left(\frac{p_{ij}^{(k)}}{(U_j^{(k)} - x_j)} + \frac{q_{ij}^{(k)}}{(x_j - L_j^{(k)})}\right)$$

$$\text{Όπου } p_{ij}^{(k)} = \begin{cases} (U_j^{(k)} - x_j^{(k)})^2 \dfrac{\partial f_i}{\partial x_j} & \alpha\nu \quad \dfrac{\partial f_i}{\partial x_j}\Big|_{x=x_t} > 0 \\ 0 & \alpha\nu \quad \dfrac{\partial f_i}{\partial x_j}\Big|_{x=x_t} \le 0 \end{cases}$$

$$q_{ij}^{(k)} = \begin{cases} 0 & \alpha\nu \quad \dfrac{\partial f_i}{\partial x_j}\Big|_{x=x_t} > 0 \\ -(x_j^{(k)} - L_j^{(k)})^2 \dfrac{\partial f_i}{\partial x_j} & \alpha\nu \quad \dfrac{\partial f_i}{\partial x_j}\Big|_{x=x_t} \le 0 \end{cases}$$

Αφού προσδιοριστούν οι προσεγγιστικές συναρτήσεις $f_i^{(k)}$ το πρόβλημα P μετατρέπεται στο:

$$P^{(k)}: \begin{cases} \min \; r_0^{(k)} + \sum_{j=1}^{n}\left(\dfrac{p_{0j}^{(k)}}{U_j^{(k)} - x_j} + \dfrac{q_{0j}^{(k)}}{x_j - L_j^{(k)}}\right) \\ \text{s.t} \begin{cases} r_i^{(k)} + \sum_{j=1}^{n}\left(\dfrac{p_{ij}^{(k)}}{U_j^{(k)} - x_j} + \dfrac{q_{ij}^{(k)}}{x_j - L_j^{(k)}}\right) \le \hat{f}_i \quad \gamma\iota\alpha \quad i = 1,\ldots,m \\ \max\{\underline{x}_j, a_j^{(k)}\} \le x_j \le \max\{\overline{x}_j, \beta_j^{(k)}\} \quad \gamma\iota\alpha \quad j = 1,\ldots,n \end{cases} \end{cases}$$

Για να αποφευχθεί η διαίρεση με το μηδέν οι παράμετροι $a_j^{(k)}$ και $\beta_j^{(k)}$ επιλέγονται ώστε:

$$L_j^{(k)} \le a_j^{(k)} \le x_j^{(k)} \le \beta_j^{(k)} \le U_j^{(k)}$$

Μια επιλογή που μπορεί να γίνει είναι η εξής:

$a_j^{(k)} = 0.9L_j^{(k)} + 0.1\,x_j^{(k)}$ και

$\beta_j^{(k)} = 0.9U_j^{(k)} + 0.1x_j^{(k)}$

Δεδομένου ότι τα κατώτερα και τα ανώτερα όρια των μεταβλητών σχεδιασμού είναι φυσικώς λογικά, μία επιλογή για τις παραμέτρους $L_j^{(k)}$ και $U_j^{(k)}$ είναι σύμφωνα με τον Svanberg [1]:

$$L_j^{(k)} = \underline{x}_j - s(\overline{x}_j - \underline{x}_j)$$

$$U_j^{(k)} = \overline{x}_j + s(\overline{x}_j - \underline{x}_j)$$

όπου s μια σταθερή τιμή. Αυτές οι παράμετροι όμως δεν εξαρτώνται από την επανάληψη k, οπότε καλούνται «σταθερές ασύμπτωτες» και όχι «κινούμενες ασύμπτωτες».

Ένας γενικός κανόνας για επιλογή των παραμέτρων $L_j^{(k)}$ και $U_j^{(k)}$ είναι ο εξής:

a) Αν η διαδικασία τείνει να ταλαντεύεται, πρέπει να σταθεροποιηθεί. Η σταθεροποίηση μπορεί να γίνει μετακινώντας τις ασύμπτωτες πιο κοντά στο $x_j^{(k)}$

b) Αν αντίθετα η διαδικασία είναι μονότονη και αργή, χρειάζεται να «χαλαρώσει». Αυτό μπορεί να γίνει μετακινώντας τις ασύμπτωτες πιο μακριά από το $x_j^{(k)}$

για k=0, k=1 σύμφωνα με το άρθρο των Liu K, Tovar A [2] οι παράμετροι επιλέγονται:

$$\left.\begin{array}{l} U_j^{(k)} + L_j^{(k)} = 2x_j^{(k)} \\ U_j^{(k)} - L_j^{(k)} = 1 \end{array}\right\} \quad \begin{array}{l} L_j^{(k)} = x_j^{(k)} - 0.5 \\ U_j^{(k)} = x_j^{(k)} + 0.5 \end{array}$$

Ενώ για k≥2:

$$\left.\begin{array}{l} U_j^{(k)} + L_j^{(k)} = 2x_j^{(k)} \\ U_j^{(k)} - L_j^{(k)} = \gamma_j^{(k)} \end{array}\right\} \quad \begin{array}{l} L_j^{(k)} = x_j^{(k)} - 0.5\,\gamma_j^{(k)} \\ U_j^{(k)} = x_j^{(k)} + 0.5\,\gamma_j^{(k)} \end{array}$$

$$\text{όπου} \quad \gamma_j^{(k)} = \begin{cases} 0.7 & \alpha\nu \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) < 0 \\ 1.2 & \alpha\nu \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) > 0 \\ 1 & \alpha\nu \quad (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) = 0 \end{cases}$$

5.4.2 Υλοποίηση της MMA

Έστω τώρα το πρόβλημα βελτιστοποίησης σε ένα συνεχές σύστημα:

$$\begin{cases} \min\ C=F^T u \\ \text{s.t} \begin{cases} \sum_{e=1}^{n} x_e\, a_e = V \\ 0 \le x_e \le 1 \quad \gamma\iota\alpha \quad e=1,\ldots,n \end{cases} \end{cases}$$

Υπολογίζονται για κάθε $x_e^{(k)}$ οι «κινούμενες ασύμπτωτες» $L_e^{(k)}$ και $U_e^{(k)}$ καθώς και οι παράμετροι $a_e^{(k)}$ και $\beta_e^{(k)}$ σύμφωνα με την διαδικασία που περιγράφηκε στα θεωρητικά στοιχεία.

Στην συνέχεια, υπολογίζεται για κάθε $x_e^{(k)}$ η παράγωγος του έργου ως προς x_e, σύμφωνα με τον τύπο:

$$\frac{\partial C(x)}{\partial x_e} = -u_e(x)^T \frac{\partial K_e(x)}{\partial x_e} u_e(x) < 0$$

Ο παραπάνω τύπος μετασχηματίζεται όπως ακριβώς περιγράφηκε στην υλοποίηση της OC. Επειδή η παράγωγος είναι πάντα αρνητική, ισχύει:

$$p_e^{(k)} = 0$$

$$q_e^{(k)} = -(x_e^{(k)} - L_e^{(k)})^2 \frac{\partial C}{\partial x_e}$$

Έτσι, το πρόβλημα $P^{(k)}$ που παρουσιάστηκε στα θεωρητικά στοιχεία γίνεται:

$$P^{(k)}: \begin{cases} g(x) = \min \sum_{e=1}^{n} \left(\frac{q_{0e}^{(k)}}{x_e - L_e^{(k)}} \right) \\ \text{s.t} \begin{cases} \sum_{e=1}^{n} x_e\, a_e = V \\ 0 \le x_e \le 1 \quad \gamma\iota\alpha\ e=1,\ldots,n \end{cases} \end{cases}$$

Η Λαγκρανζιανή συνάρτηση του $P^{(k)}$ είναι:

$$l(x,\lambda)=\sum_{e=1}^{n}\left(\frac{q_e^{(k)}}{x_e-L_e^{(k)}}\right)+\lambda\,(\sum_{e=1}^{n}x_e\,a_e-V)$$

Σύμφωνα με την Lagrangian duality, αρκεί να βρεθεί το ελάχιστο ως προς x και μετά το μέγιστο ως προς λ. Για το ελάχιστο ως προς x κάθε στοιχείου ισχύει:

$$\frac{\partial l(x,\lambda)}{\partial x_e}=0 \quad \Leftrightarrow -\frac{q_e^{(k)}}{(x_e-L_e^{(k)})^2}+\lambda=0 \Leftrightarrow x_e=\sqrt{\frac{q_e^{(k)}}{\lambda}}+L_e^{(k)}$$

Το x_e πρέπει να βρίσκεται μέσα στα όρια: $0\le x_e\le 1$ και $a_e^{(k)}\le x_e^{(k)}\le \beta_e^{(k)}$. Επομένως αν η λύση ξεπερνά τις τιμές των άκρων το x_e παίρνει την ακραία τιμή:

$$x_e=\begin{cases}\max(0,a_e^{(k)}) & \alpha\nu \quad \sqrt{\frac{q_e^{(k)}}{\lambda}}+L_e^{(k)}\le \max(0,a_e^{(k)})\\ \sqrt{\frac{q_e^{(k)}}{\lambda}}+L_e^{(k)} & \alpha\nu \quad \min(1,\beta_e^{(k)})\le \sqrt{\frac{q_e^{(k)}}{\lambda}}+L_e^{(k)}\le \max(0,a_e^{(k)})\\ \min(1,\beta_e^{(k)}) & \alpha\nu \quad \sqrt{\frac{q_e^{(k)}}{\lambda}}\ge \min(1,\beta_e^{(k)})\end{cases}$$

Το μέγιστο ως προς λ βρίσκεται παραγωγίζοντας την $L(x,\lambda)$ ως προς λ για δεδομένα x_e και εξισώνοντάς την με μηδέν.

$$\frac{\partial L}{\partial \lambda}=\sum_{e=1}^{n}\left(x_e\alpha_e-V\right)=0$$

Αυτή η διαδικασία αποτελεί μια εσωτερική επανάληψη για τον υπολογισμό του κατάλληλου λ. Υπολογίζονται τα x_e για κάθε λ και ελέγχεται αν ισχύει η σχέση:

$$\sum_{e=1}^{n}\left(x_e\alpha_e\right)=V$$

Η εσωτερική επανάληψη συνεχίζεται μέχρι να βρεθεί το λ για το οποίο ισχύει η παραπάνω σχέση. Για την εύρεση του λ πιο γρήγορα εφαρμόζεται και εδώ η μέθοδος της διχοτόμησης, όπως περιγράφηκε στην ΟC. Τα x_e που προέκυψαν για αυτό το λ, θα αποτελέσουν τις νέες πυκνότητες της επαναληπτικής διαδικασίας που λύνει το αρχικό πρόβλημα βελτιστοποίησης τοπολογίας.

5.5 Περιγραφή του λογισμικού

5.5.1 Αναλυτική περιγραφή

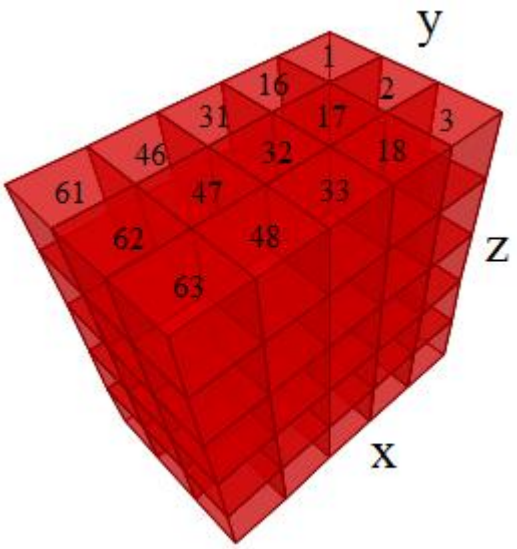
Το λογισμικό ξεκινάει ανοίγοντας το αρχείο SDB που έχει δημιουργηθεί στο SAP2000.

1. Το πρώτο πράγμα που κάνει, είναι να διαβάσει το μοντέλο με την μέθοδο **«ReadModel»**. Η μέθοδος αυτή επιστρέφει σε πίνακες τα ονόματα όλων των στοιχείων solid, area και frame του μοντέλου, καθώς και τα ονόματα όλων των σημείων (points) και όλων των loadcases. Για όλους αυτούς τους πίνακες επιστρέφει με την βοήθεια μεταβλητών και τον αριθμό στοιχείων του κάθε πίνακα. Επιπλέον, μέσα στην μέθοδο ReadModel ορίζονται οι τοπικοί άξονες όλων των στοιχείων στην ίδια κατεύθυνση με τους καθολικούς, έτσι ώστε όλα τα στοιχεία να έχουν κοινούς άξονες αναφοράς.

2. Το κομμάτι της κατασκευής που βελτιστοποιείται αποτελείται από πολλά πεπερασμένα στοιχεία, τα οποία στο SAP2000 προσομοιώνονται με στοιχεία Solid. Σε κάθε πεπερασμένο στοιχείο ορίζεται η μεταβλητή σχεδιασμού x_e η οποία αντιπροσωπεύει την πυκνότητα του στοιχείου. Η γενική διαδικασία αντιμετώπισης αυτών των προβλημάτων είναι η εξής: Αρχικά, επιλέγονται κάποιες πυκνότητες x^k και στην συνέχεια μέσω μιας επαναληπτικής διαδικασίας επιλέγονται οι επόμενες πυκνότητες. Ανάλογα με την τιμή της μεταβλητής x_e, το κάθε στοιχείο πρέπει κάθε φορά να προσομοιωθεί με τον κατάλληλο τρόπο. Αυτό που διαφοροποιείται ουσιαστικά ανάλογα με την πυκνότητα (βλέπε θεωρητικά στοιχεία SIMP), είναι το μέτρο ελαστικότητας του υλικού του κάθε πεπερασμένου στοιχείου, σύμφωνα με την σχέση $E_e(x_e) = x_e^3 E_e$. Έτσι, προκειμένου να γίνει η κατάλληλη προσομοίωση των στοιχείων που περιγράφηκε παραπάνω, το δεύτερο βήμα αποτελείται από την μέθοδο **«CreateMatSolid»**, η οποία δημιουργεί 101 διαφορετικά υλικά με μέτρο ελαστικότητας που αυξάνει ανά 0.01*E (όπου E το αρχικό μέτρο ελαστικότητας). Για το πρώτο υλικό επειδή δεν μπορεί να έχει μέτρο ελαστικότητας 0, επιλέχθηκε τιμή πολύ κοντά σε αυτό. Το δεύτερο υλικό έχει μέτρο ελαστικότητας 0.01*E, το τρίτο 0.02*E κτλ. Για κάθε τέτοιο υλικό δημιουργείται η αντίστοιχη διατομή Solid, η οποία πετυχαίνει την απαιτούμενη προσομοίωση.

3. Στο πρώτο βήμα η μέθοδος ReadModel επιστρέφει πίνακες με όλα στα στοιχεία της κατασκευής. Όμως, προκειμένου να γίνει η βελτιστοποίηση, το λογισμικό χρειάζεται να έχει και έναν πίνακα μόνο με τα ονόματα των στοιχείων Solid που πρόκειται να βελτιστοποιηθούν. Για να γίνει αυτό, καλείται η μέθοδος **«CreateOptiSolidName»**, η οποία δημιουργεί τον απαιτούμενο πίνακα ως εξής: Ο χρήστης δίνει τις συντεταγμένες του όγκου της κατασκευής που βελτιστοποιείται, καθώς και τις διαστάσεις του κάθε πεπερασμένου στοιχείου. Έτσι, το λογισμικό μπορεί να υπολογίσει τις συντεταγμένες του κάθε πεπερασμένου στοιχείου που βελτιστοποιείται και χρησιμοποιώντας την συνάρτηση του OAPI «Model.SelectObj.CoordinateRange», μπορεί να επιλέγει κάθε φορά το στοιχείο, να βρίσκει το όνομά του και να το τοποθετεί στην κατάλληλη θέση του νέου πίνακα. Ο τρόπος με τον οποίο ταξινομούνται τα ονόματα στον πίνακα OptiSolidName έχει να κάνει με την αρίθμηση των στοιχείων που θα περιγραφεί στο επόμενο βήμα.

4. Κάθε φορά που προκύπτουν καινούργιες πυκνότητες ή καινούργιες παράγωγοι, πρέπει να γίνει ένα φιλτράρισμα για να αποφευχθεί το πρόβλημα της σκακιέρας. Το φίλτρο εφαρμόζεται σε κάθε στοιχείο ξεχωριστά με βάση τα γειτονικά του στοιχεία. Για να γνωρίζει το λογισμικό για κάθε στοιχείο ποια είναι τα γειτονικά του, εφαρμόστηκε η λογική αρίθμησης που υπάρχει και στον κώδικα 88 γραμμών της Matlab [3]. Στις εικόνες 5.3 και 5.4 φαίνεται η αρίθμηση ενός παραδείγματος τριών στοιχείων κατά τον άξονα y, πέντε κατά τον x και πέντε κατά τον z.



Εικόνα 5.3: Όψη 3d

3	2	1
6	5	4
9	8	7
12	11	10
15	14	13

Εικόνα 5.4: Όψη yz

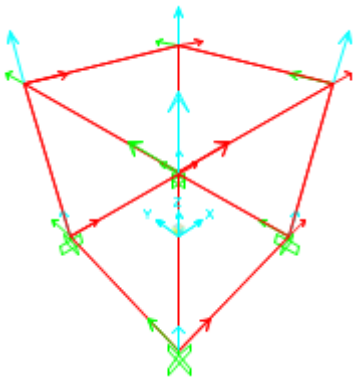
Με την παραπάνω αρίθμηση είναι εύκολο για το λογισμικό να υπολογίσει για κάθε στοιχείο ποια είναι τα γειτονικά του. Αυτή η διαδικασία γίνεται με την βοήθεια της μεθόδου **«CreateMapDist»**. Σε αυτή τη μέθοδο δημιουργείται ένας πίνακας με το όνομα Map, όπου για κάθε στοιχείο περιέχει τους αριθμούς των γειτονικών του στοιχείων. Επιπλέον, δημιουργείται και ο πίνακας Dist στον οποίο υπάρχουν οι αντίστοιχες αποστάσεις μεταξύ των στοιχείων. Τα δεδομένα αυτά χρησιμοποιούνται στην συνέχεια σε κάθε επανάληψη για το φιλτράρισμα.

Στην συνέχεια, αρχίζει η επαναληπτική διαδικασία για τον υπολογισμό των καινούργιων πυκνοτήτων.

5. Πρώτο βήμα της επαναληπτικής διαδικασίας είναι η ανανέωση του μοντέλου με βάση τον πίνακα των τελευταίων φιλτραρισμένων πυκνοτήτων xkfil. Όταν το πρόγραμμα βρίσκεται στην $1^{η}$ επανάληψη, τότε όλες οι τιμές του πίνακα είναι αρχικοποιημένες και ίσες με το συνολικό επιτρεπόμενο ποσοστό όγκου V. Η μέθοδος **«UpdateModel»** δέχεται ως δεδομένο τον πίνακα xkfil και τοποθετεί την κατάλληλη διατομή Solid σε κάθε πεπερασμένο στοιχείο ανάλογα με την τιμή της πυκνότητάς του.

6. Αφού γίνει η ανανέωση του μοντέλου, το επόμενο βήμα είναι το τρέξιμό του με την συνάρτηση RunAnalysis.

7. Ακόμα, γίνεται ο υπολογισμός του έργου από τον τύπο: $C=F^T U$. Το SAP2000 μπορεί να δώσει πίνακες με τις μετατοπίσεις και τις δυνάμεις κατά τους τοπικούς άξονες στα 8 σημεία κάθε στοιχείου solid. Πολλαπλασιάζοντας τις δυνάμεις με τις αντίστοιχες μετατοπίσεις σε κάθε κατεύθυνση, υπολογίζεται το έργο του στοιχείου solid. Το συνολικό έργο της κατασκευής υπολογίζεται προσθέτοντας τα έργα για κάθε στοιχείο. Η παραπάνω διαδικασία υλοποιείται καλώντας την μέθοδο **«CalcCompliance»**.



Εικόνα 5.5: στοιχείο solid

Το συνολικό έργο της κατασκευής δεν λαμβάνεται υπόψη στην διαδικασία εύρεσης των καινούργιων πυκνοτήτων αλλά χρησιμοποιείται για να διαπιστωθεί αν η διαδικασία βελτιστοποίησης είναι σωστή, δηλαδή αν μετά από κάθε μία επανάληψη, οι νέες πυκνότητες έχουν ως αποτέλεσμα να παραχθεί μικρότερο έργο.

8. Επόμενο βήμα είναι ο υπολογισμός των παραγώγων του έργου ως προς κάθε στοιχείο x_e, καλώντας την μέθοδο **«CalcDerivatives»**. Η μέθοδος δέχεται ως δεδομένο τον πίνακα xkfil. Όπως αναφέρθηκε στο κεφάλαιο 5.3.2, ο τελικός τύπος για τον υπολογισμό της κάθε παραγώγου είναι:

$$\frac{\partial C(x)}{\partial x_e} = -\frac{3C_e}{x_e}$$

Έτσι λοιπόν, η μέθοδος βρίσκει για κάθε στοιχείο solid το έργο του με την διαδικασία που περιγράφηκε στο βήμα 7 και υπολογίζει την παράγωγο. Η μέθοδος στο τέλος επιστρέφει τον πίνακα der με όλες τις παραγώγους σε αντίστοιχη διάταξη με τον πίνακα OptiSolidName που περιέχει τα ονόματα των στοιχείων που βελτιστοποιούνται.

9. Αφού υπολογιστούν οι παράγωγοι πρέπει να φιλτραριστούν για να αποφευχθεί το φαινόμενο της σκακιέρας. Η μέθοδος **«FilterDer»** δέχεται ως δεδομένα τους πίνακες der και xkfil αλλά και τους πίνακες Map και Dist που δημιουργήθηκαν στο 4ο βήμα. Φιλτράρει την παράγωγο κάθε στοιχείου ξεχωριστά χρησιμοποιώντας τον παρακάτω τύπο, που παρουσιάστηκε στον κώδικα των 88 γραμμών της Matlab [3]:

$$\frac{\partial C(x)}{\partial x_e} = \frac{1}{\max(10^{-3}, x_e)\sum_{i\in N_e} H_{ei}} \sum_{i\in N_e} H_{ei}\, x_i \frac{\partial C}{\partial x_i}$$

Για κάθε στοιχείο παίρνει από τον πίνακα Map τα γειτονικά του. Για κάθε γειτονικό του στοιχείο, πολλαπλασιάζει την πυκνότητα του (x_i) με την παράγωγό του και με την απόστασή του από το αρχικό στοιχείο (H_{ei}). Την απόσταση αυτή την παίρνει από τον πίνακα Dist. Υπολογίζει το άθροισμα των παραπάνω γινομένων για κάθε γειτονικό στοιχείο και το διαιρεί με το άθροισμα των αποστάσεων και με την πυκνότητα του αρχικού στοιχείου x_e. Στην περίπτωση όπου η πυκνότητα του στοιχείου είναι 0, η διαίρεση γίνεται με το 10^{-3}. Έτσι λοιπόν, προκύπτει για κάθε στοιχείο η νέα του φιλτραρισμένη παράγωγος. Η μέθοδος επιστρέφει τον πίνακα derfil με όλες τις φιλτραρισμένες παραγώγους.

10. Μέχρι εδώ η διαδικασία είναι ουσιαστικά ίδια και για τις δύο μεθοδολογίες. Στην συνέχεια, ανάλογα με το ποια μεθοδολογία έχει επιλεγεί, το πρόγραμμα καλεί είτε την μέθοδο **«Optimality Criteria»** είτε την μέθοδο **«MMA»**:

    - Η μέθοδος Optimality Criteria υλοποιεί την διαδικασία εύρεσης των νέων πυκνοτήτων που περιγράφηκε στο κεφάλαιο 5.3.2
    - Η μέθοδος MMA υλοποιεί την διαδικασία εύρεσης των νέων πυκνοτήτων που περιγράφηκε στο κεφάλαιο 5.4.2

    Θέλει προσοχή ότι και οι δύο μέθοδοι δέχονται ως δεδομένο τον πίνακα αφιλτράριστων πυκνοτήτων xk και όχι των φιλτραρισμένων xkfil. Στην συνέχεια σε κάθε επανάληψη βρίσκουν τις νέες πυκνότητες (πίνακας xnew), τις φιλτράρουν (πίνακας xnewfil) και μετά ελέγχουν αν ισχύει η σχέση:

    $$\sum_{e=1}^{n}(x_e) = V$$

    Το φίλτρο που χρησιμοποιείται για τις πυκνότητες δίνεται από την σχέση:

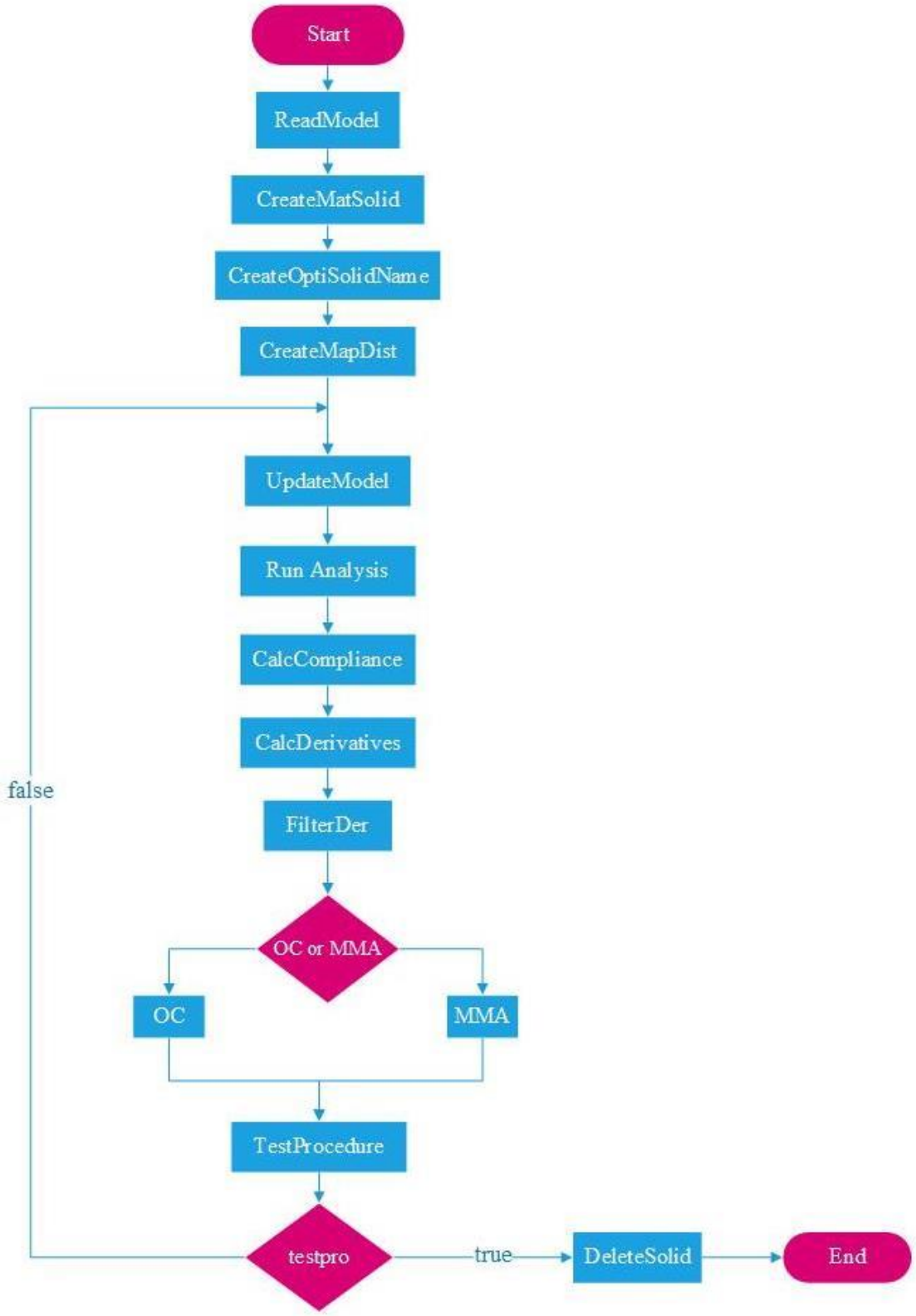
    $$x_e = \frac{1}{\sum_{i \in N_e} H_{ei}} \sum_{i \in N_e} H_{ei}\, x_i$$

    Είναι προφανές ότι για να εφαρμόσουν το φίλτρο, οι μέθοδοι δέχονται ως δεδομένα τους πίνακες Map και Dist που είχαν δημιουργηθεί στο $4^{ο}$ βήμα.

11. Η μέθοδος **«TestProcedure»** δέχεται τους πίνακες των αφιλτράριστων πυκνοτήτων xk και xnew, ελέγχει αν ικανοποιείται το κριτήριο τερματισμού και επιστρέφει το αποτέλεσμα με την λογική μεταβλητή testpro. Αν η μέθοδος επιστρέψει την τιμή true, η επαναληπτική διαδικασία τερματίζεται. Αν η μέθοδος επιστρέψει την τιμή false, τότε η διαδικασία γυρίζει ξανά στο $5^{ο}$ βήμα. Πριν γυρίσει, οι τιμές του πίνακα xnew τοποθετούνται στον xk και οι τιμές του πίνακα xnewfil στον xkfil.

12. Στο τέλος, καλείται η μέθοδος **«DeleteSolid»** για να διαγράψει όλα τα στοιχεία solid που έχουν πυκνότητα κάτω από μία τιμή που ορίζει ο χρήστης. Όταν διαγραφούν αυτά τα στοιχεία, η κατασκευή έχει πάρει την βελτιστοποιημένη της μορφή και ο χρήστης μπορεί να την δεί κατευθείαν με ένα refresh στο περιβάλλον του SAP2000.

5.5.2 Διάγραμμα ροής



5.6 Αναφορές

[1] Svanberg K (1987) The method of moving asymptotes—a new method for structural optimization. International Journal for Numerical Methods in Engineering 24:359-373

[2] Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. Struct Multidisc Optim

[3] Andreassen E, Clausen A, Schevenels M, Lazarov B.S, Sigmund O (2011) Efficient topology optimization in MATLAB using 88 lines of code. Struct Multidisc Optim 43:1-16

ΚΕΦΑΛΑΙΟ 6

ΑΡΙΘΜΗΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ - ΑΠΟΤΕΛΕΣΜΑΤΑ

6.1 Εισαγωγή

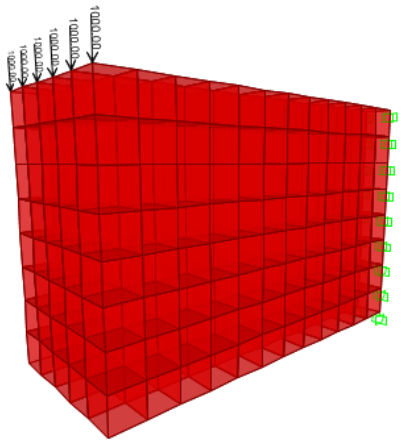
Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα που προέκυψαν από το λογισμικό βελτιστοποίησης τοπολογίας. Το λογισμικό αναπτύχθηκε ώστε να έχει εφαρμογή σε προβλήματα πολιτικού μηχανικού, επομένως εφαρμόστηκαν ανάλογα παραδείγματα. Στο πρώτο μέρος παρουσιάζονται πιο απλά παραδείγματα, όπως η βελτιστοποίηση προβόλων και αμφιερείστων με διάφορες φορτίσεις. Στο δεύτερος μέρος παρουσιάζονται πιο σύνθετα παραδείγματα, όπως η βελτιστοποίηση του τοιχίου μιας τριώροφης κατασκευής.

6.2 Απλά παραδείγματα

Για κάθε παράδειγμα παρουσιάζεται το αρχικό του σχήμα και στην συνέχεια τα σχήματα που προέκυψαν μετά την εφαρμογή και των δύο μεθόδων βελτιστοποίησης τοπολογίας του λογισμικού (OC και MMA). Τα πρώτα αποτελέσματα συγκρίθηκαν με τα αποτελέσματα που προέκυψαν από τον αλγόριθμο βελτιστοποίησης 88 γραμμών της Matlab ώστε να διαπιστωθεί η ορθότητά τους. Για τα επόμενα παραδείγματα παρουσιάζονται μόνο τα αποτελέσματα του λογισμικού.

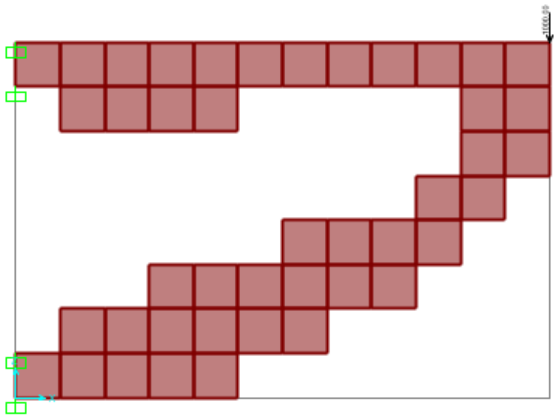
Παράδειγμα 1

Το πρώτο παράδειγμα είναι ένας πρόβολος διαστάσεων 1.2m κατά τον διαμήκη άξονα x, 0.5m κατά τον εγκάρσιο άξονα y και 0.8m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=12, nly=5, nlz=8) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 480 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m. Ορίζονται δεσμεύσεις σε όλα τα σημεία στην δεξιά πλευρά του προβόλου, ενώ εφαρμόζονται συγκεντρωμένες δυνάμεις με φορά προς τα κάτω, στα άνω σημεία της αριστερής πλευράς.

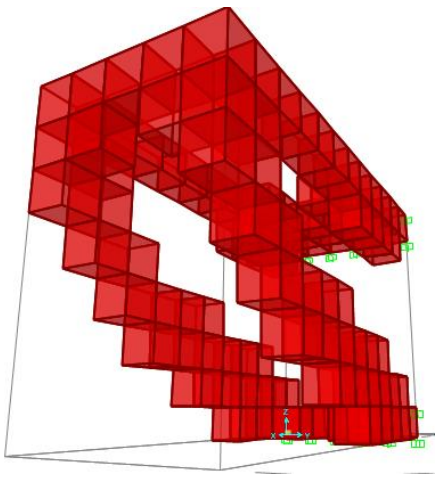


Εικόνα 6.1: Αρχικό σχήμα

Εφαρμόστηκαν και οι δυο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 30%. Η μέθοδος OC σταμάτησε στο όριο των 150 επαναλήψεων που είχε οριστεί και έδωσε το εξής αποτέλεσμα:

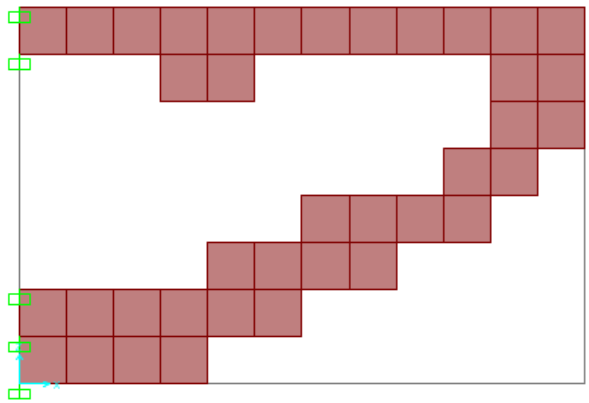


Εικόνα 6.2: Αποτέλεσμα OC - Όψη

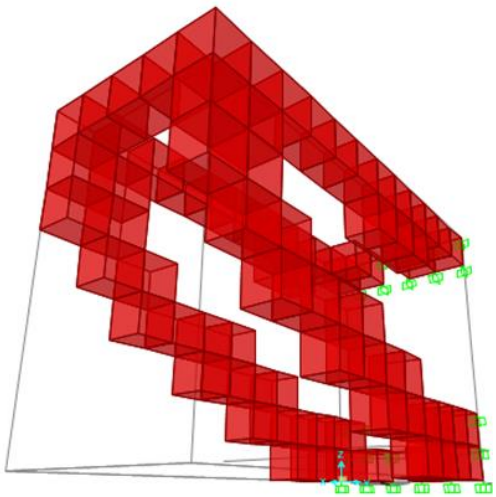


Εικόνα 6.3: Αποτέλεσμα OC – Όψη 3D

Και η μέθοδος MMA σταμάτησε στο όριο των 150 επαναλήψεων και έδωσε το αποτελέσμα που φαίνεται στις εικόνες 6.4 και 6.5:

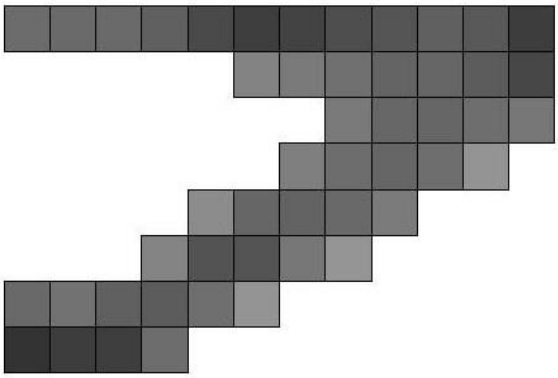


Εικόνα 6.4: Αποτέλεσμα MMA - Όψη

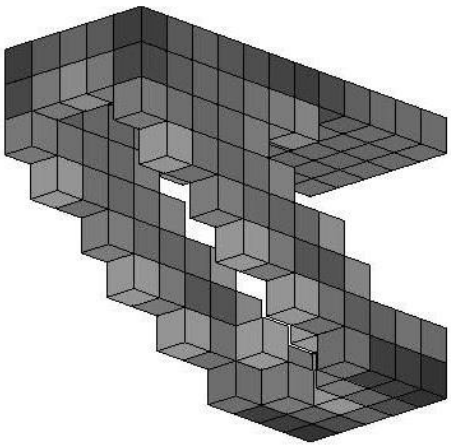


Εικόνα 6.5: Αποτέλεσμα MMA – Όψη 3D

Παρατηρήθηκε οτι οι λύσεις που προέκυψαν από το πρόγραμμα και με τις δύο μεθοδολογίες OC και MMA είναι σχεδόν ίδιες. Προκειμένου να ελεγχθεί το αποτέλεσμα, εφαρμόστηκε το ίδιο παράδειγμα στον κώδικα της Matlab. Έδωσε το παρακάτω αποτέλεσμα:



Εικόνα 6.6: Αποτέλεσμα Matlab - Όψη

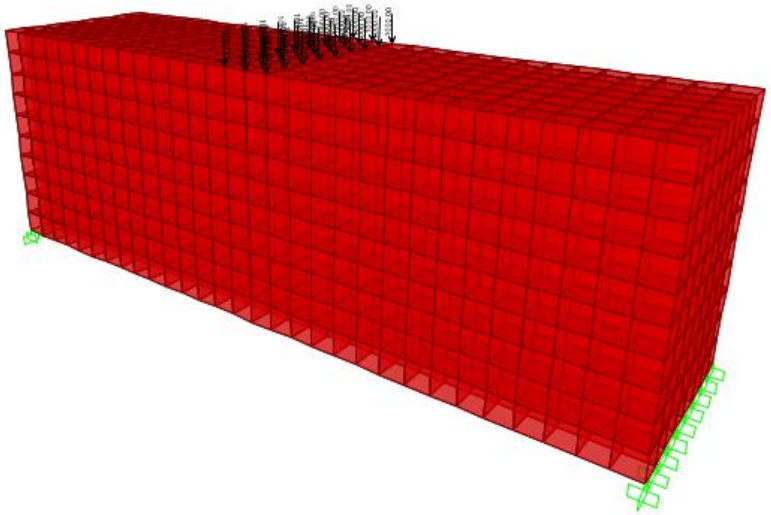


Εικόνα 6.7: Αποτέλεσμα Matlab – Όψη 3D

Όπως φαίνεται στις εικόνες 6.6 και 6.7 η Matlab έδωσε παρόμοιο σχήμα με το πρόγραμμα, γεμίζοντας όμως περισσότερα στοιχεία κοντά στην πλευρά που εφαρμόζονται οι δυνάμεις. Ο πιο πιθανός λόγος που συνέβη αυτό, είναι ότι η Matlab διαγράφει τα στοιχεία με πυκνότητα κάτω του 0.5, ενώ στο λογισμικό επιλέχτηκε η διαγραφή των στοιχείων κάτω του 0.6.

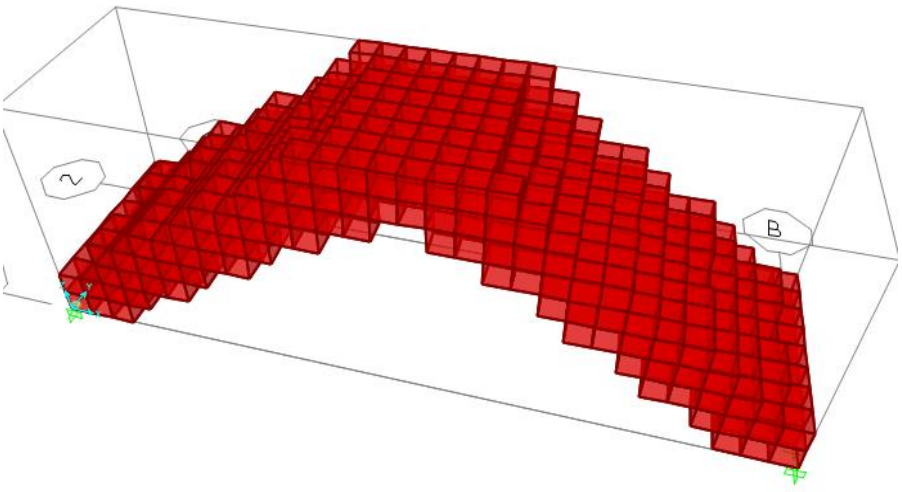
Παράδειγμα 2

Το δεύτερο παράδειγμα είναι μια αμφιέρειστη δοκός διαστάσεων 3.0m κατά τον διαμήκη άξονα x, 0.8m κατά τον εγκάρσιο άξονα y και 1.0m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=30, nly=8, nlz=10) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 2400 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m Ορίζονται δεσμεύσεις στα κάτω σημεία κάθε πλευράς, ενώ εφαρμόζονται συγκεντρωμένες δυνάμεις με φορά προς τα κάτω στο κέντρο της δοκού, όπως φαίνεται στην εικόνα 6.8:

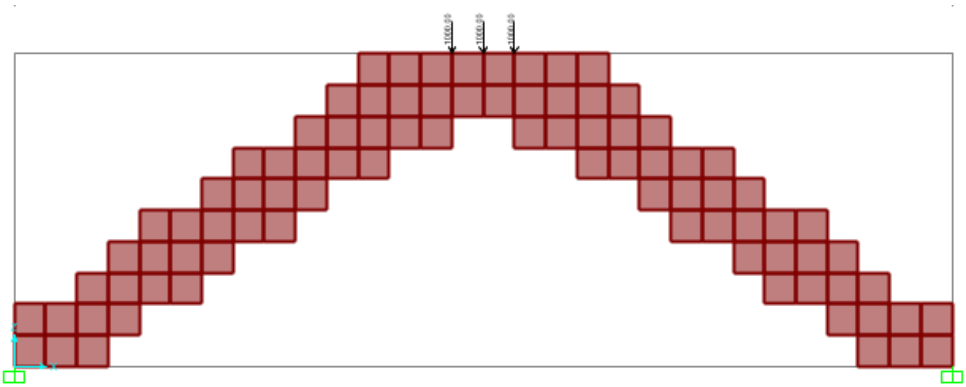


Εικόνα 6.8: Αρχικό σχήμα

Εφαρμόστηκαν και οι δυο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 30%. Η μέθοδος OC ικανοποίησε το κριτήριο τερματισμού και σταμάτησε στις 90 επαναλήψεις. Στις εικόνες 6.9 και 6.10 φαίνονται τα αποτελέσματα της μεθόδου:

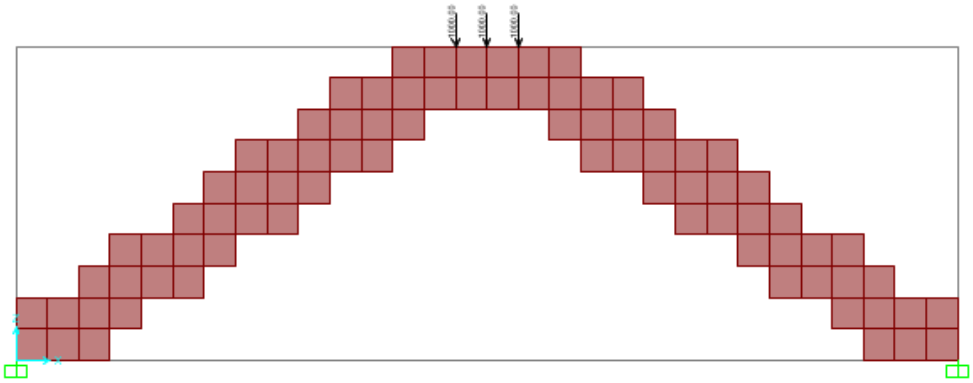


Εικόνα 6.9: Αποτέλεσμα OC – Όψη 3D



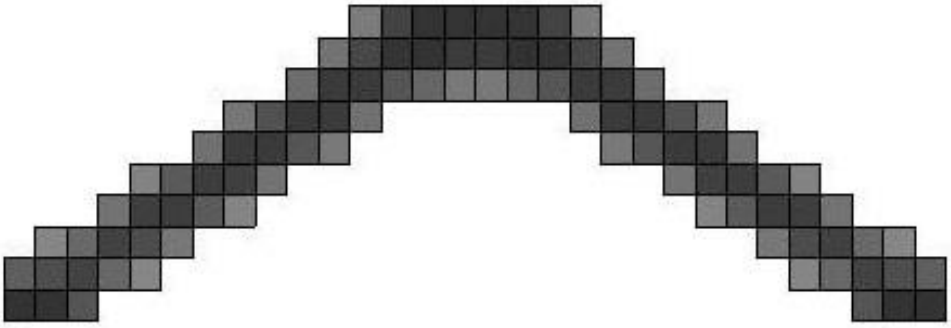
Εικόνα 6.10: Αποτέλεσμα OC – Όψη

Η μέθοδος MMA σταμάτησε στο όριο των 150 επαναλήψεων και έδωσε το αποτέλεσμα:

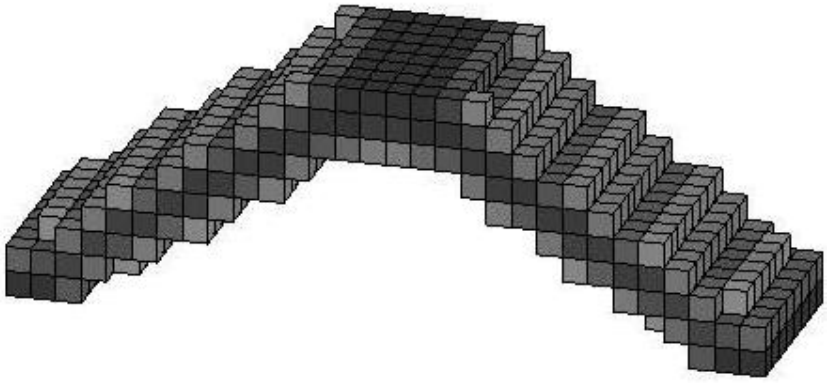


Εικόνα 6.11: Αποτέλεσμα MMA – Όψη

Παρατηρήθηκε ότι το αποτέλεσμα που προέκυψε από την μέθοδο OC, είναι σχεδόν ίδιο με αυτό που προέκυψε από την μέθοδο MMA. Προκειμένου να ελεγχθούν τα παραπάνω αποτελέσματα εφαρμόστηκε το ίδιο παράδειγμα στον κώδικα της Matlab:

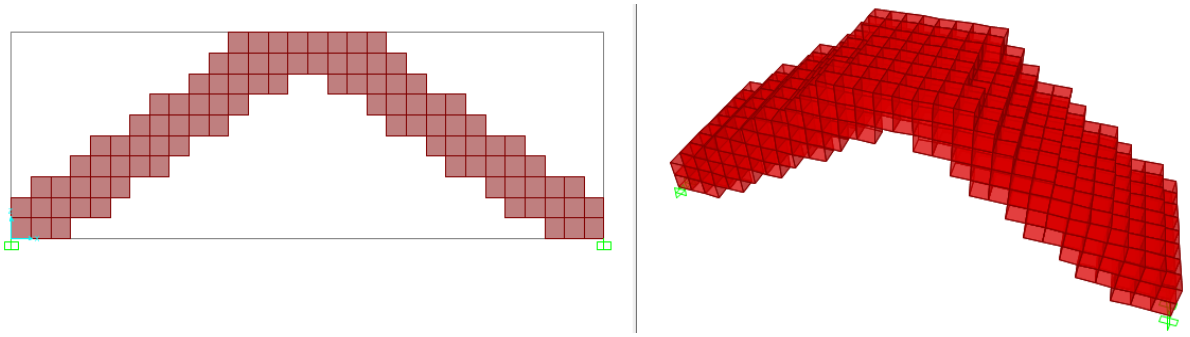


Εικόνα 6.12: Αποτέλεσμα Matlab – Όψη



Εικόνα 6.13: Αποτέλεσμα Matlab – Όψη 3D

Όπως φαίνεται στα σχήματα 6.12 και 6.13, τα αποτελέσμτα της Matlab επαλήθευσαν αυτά του προγράμματος. Τέλος, στο συγκεκριμένο παράδειγμα εφαρμόστηκαν ξανά οι μέθοδοι OC και MMA, αυτή τη φορά με όριο επαναλήψεων τις 20.

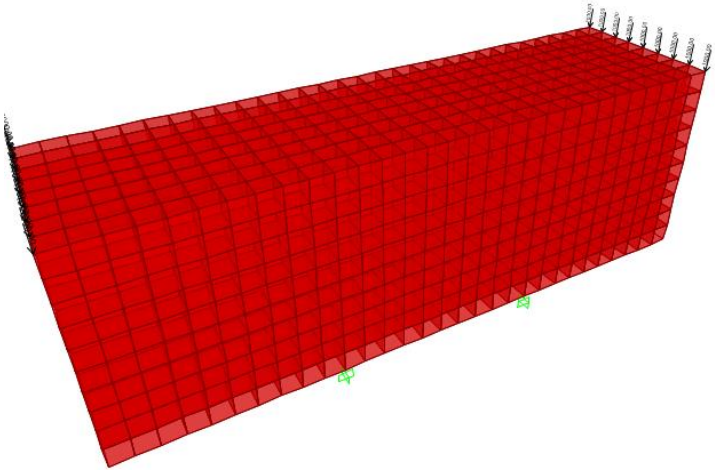


Εικόνα 6.14: Αποτέλεσμα MMA 20 επαναλήψεων

Τα αποτελέσματα που προέκυψαν από 20 επαναλήψεις της MMA είναι ίδια με τα προηγούμενα. Αυτό σημαίνει. ότι στο συγκεκριμένο παράδειγμα αρκούσαν μόνο 20 επαναλήψεις, ώστε να πάρει το βελτιστοποιημένο σχήμα που θα έπερνε αν γινόντουσαν 150! Αυτή η παρατήρηση ήταν ιδιαίτερα σημαντική, καθώς οι 20 επαναλήψεις χρειάζονται πολύ λιγότερο χρόνο.

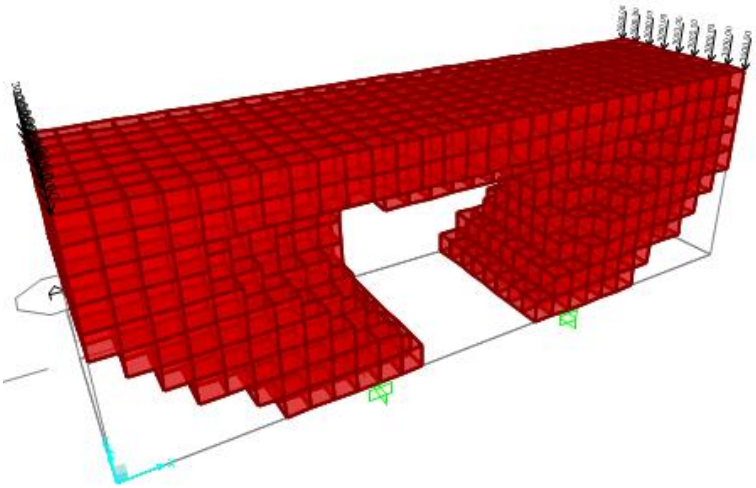
Παράδειγμα 3

Το τρίτο παράδειγμα είναι μια δοκός όπως η προηγούμενη στο παράδειγμα 2, διαστάσεων 3.0m κατά τον διαμήκη άξονα x, 0.8m κατά τον εγκάρσιο άξονα y και 1.0m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=30, nly=8, nlz=10) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 2400 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m. Αυτή τη φορά οι δεσμεύσεις δεν ορίζονται στα κάτω σημεία των εξωτερικές πλευρών, αλλά σε απόσταση 1m από αυτές, όπως φαίνεται στην εικόνα 6.15. Επιπλέον, εφαρμόζονται δυνάμεις στα άνω σημεία των εξωτερικών πλευρών.

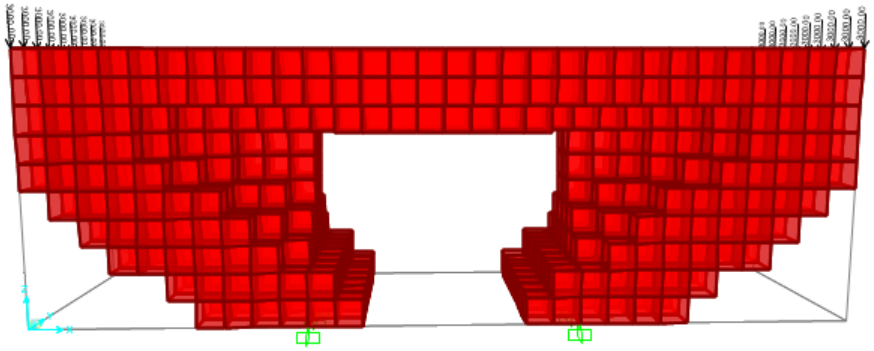


Εικόνα 6.15: Αρχικό σχήμα

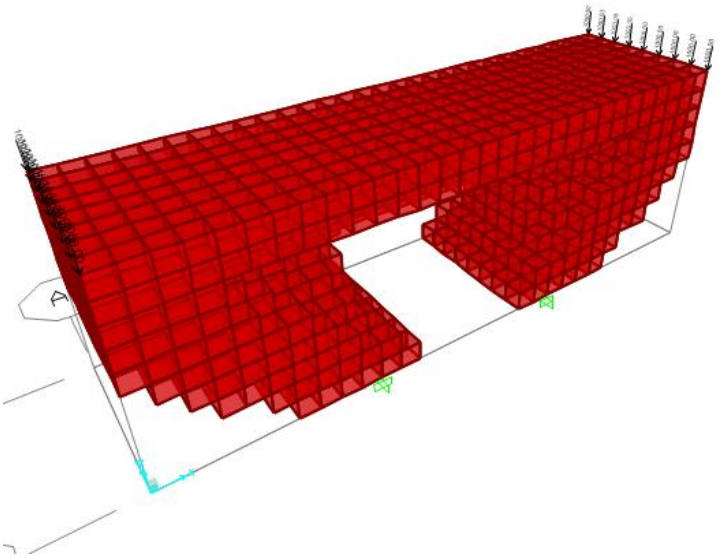
Εφαρμόστηκαν και οι δυο μέθοδοι για ποσοστό όγκου 60%. Η μέθοδος OC σταμάτησε στις 140 επαναλήψεις, ενώ η MMA στο όριο των 150. Τα αποτελέσματα παρατίθενται παρακάτω:



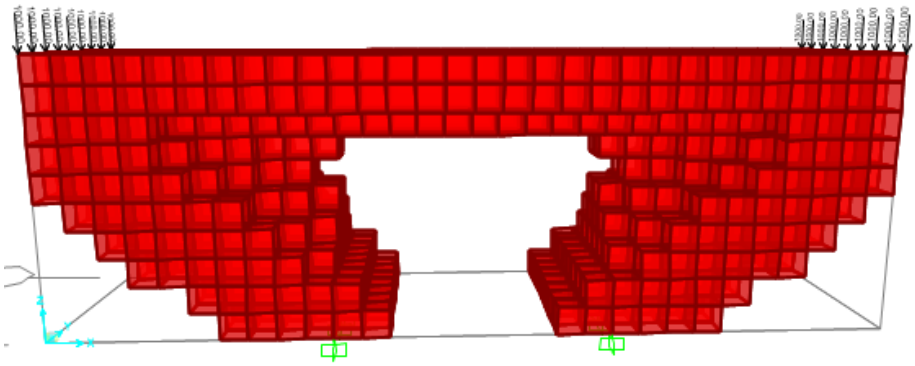
Εικόνα 6.16: Αποτέλεσμα OC – Όψη 3D



Εικόνα 6.17: Αποτέλεσμα OC – Όψη

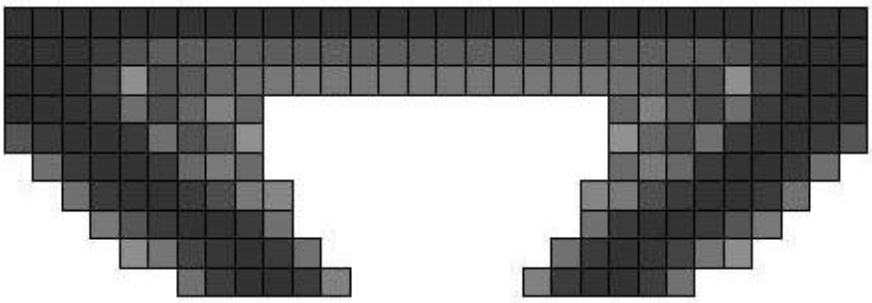


Εικόνα 6.18: Αποτέλεσμα MMA – Όψη 3D

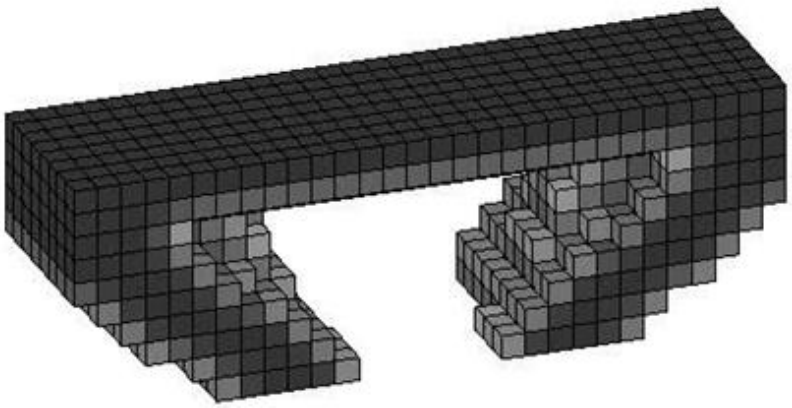


Εικόνα 6.19: Αποτέλεσμα MMA – Όψη

Για να ελεγχθούν τα αποτελέσματα εφαρμόστηκε το παράδειγμα στον κώδικα της Matlab:



Εικόνα 6.20: Αποτέλεσμα Matlab – Όψη

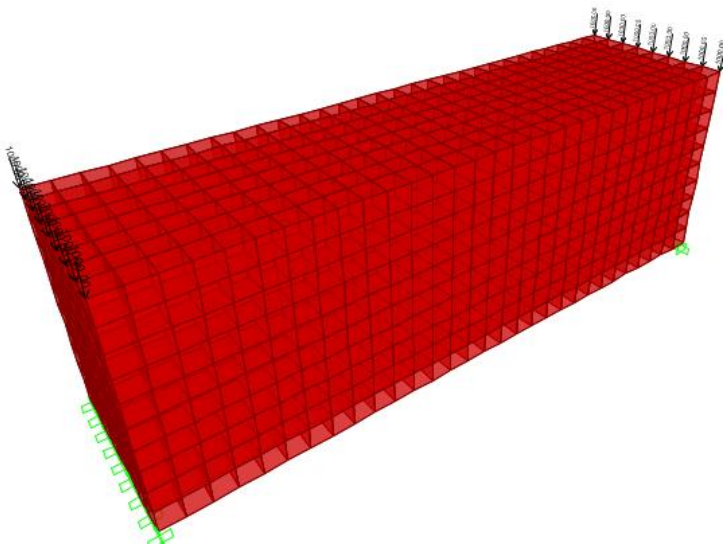


Εικόνα 6.21: Αποτέλεσμα Matlab – Όψη 3D

Έτσι, και αυτά τα αποτελέσματα επαληθεύτηκαν. Εφόσον διαπιστώθηκε ότι τα αποτελέσματα που προέκυψαν από το λογισμικό των τριών πρώτων παραδειγμάτων έχουν ουσιαστικά ίδιο σχήμα με τα αποτελέσματα που έδωσε η Matlab, για τα επόμενα παραδείγματα εφαρμόστηκαν μόνο οι μέθοδοι του λογισμικού και βγήκαν συμπεράσματα επί αυτών.

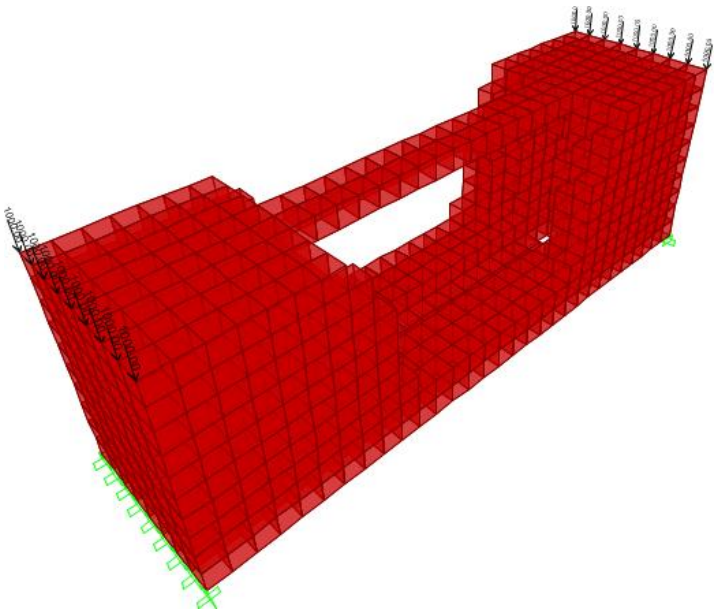
Παράδειγμα 4

Το τέταρτο παράδειγμα είναι η αμφιέρειστη δοκός όπως η προηγούμενη στο παράδειγμα 2. Αυτή τη φορά δεν εφαρμόζονται δυνάμεις στο κέντρο, αλλά στα άνω σημεία των εξωτερικών πλευρών, όπως φαίνεται στην εικόνα 6.22

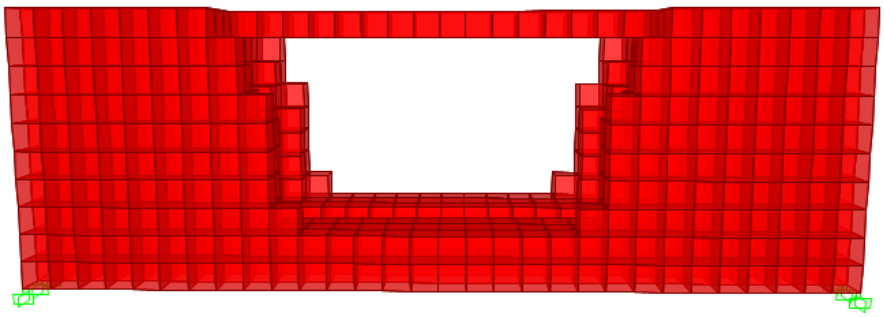


Εικόνα 6.22: Αρχικό σχήμα

Εφαρμόστηκαν και οι δυο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 60%. Η μέθοδος OC σταμάτησε στο όριο των 150 και έδωσε το εξής αποτέλεσμα:

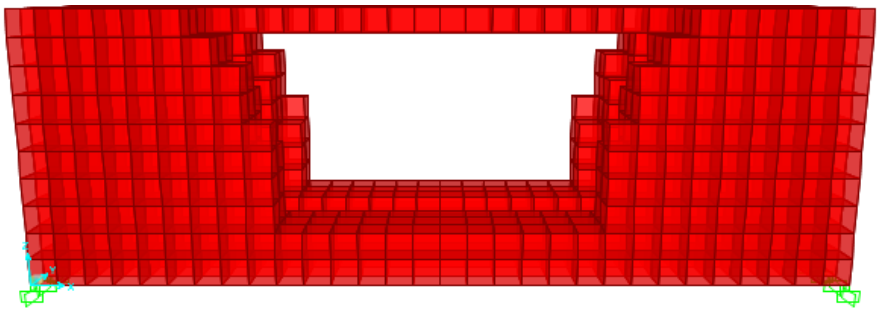


Εικόνα 6.23: Αποτέλεσμα ΟC – Όψη 3D

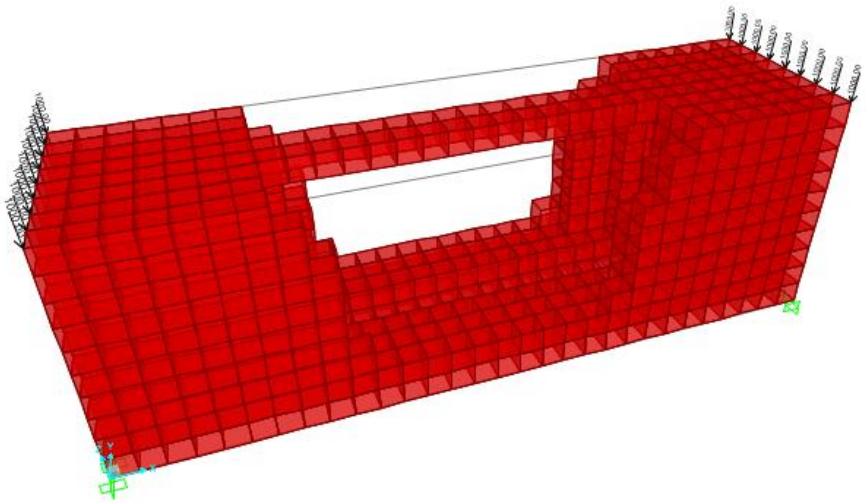


Εικόνα 6.24: Αποτέλεσμα ΟC – Όψη

Και η μέθοδος MMA σταμάτησε στο όριο των 150 επαναλήψεων και έδωσε το αποτέλεσμα:



Εικόνα 6.25: Αποτέλεσμα MMA – Όψη

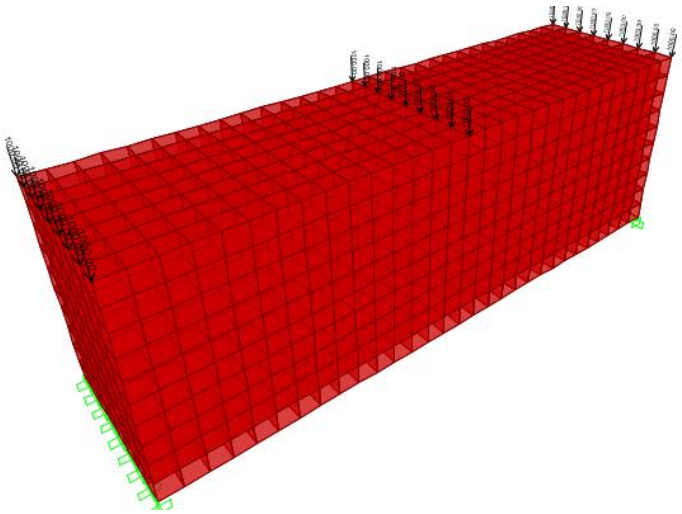


Εικόνα 6.26: Αποτέλεσμα MMA – Όψη 3D

Παρατηρήθηκε ότι οι λύσεις είναι ίδιες. Τέλος, εφαρμόστηκε ξανά στο ίδιο παράδειγμα η μέθοδος OC, αυτή τη φορά με όριο επαναλήψεων τις 20 και προέκυψε ίδιο αποτέλεσμα.

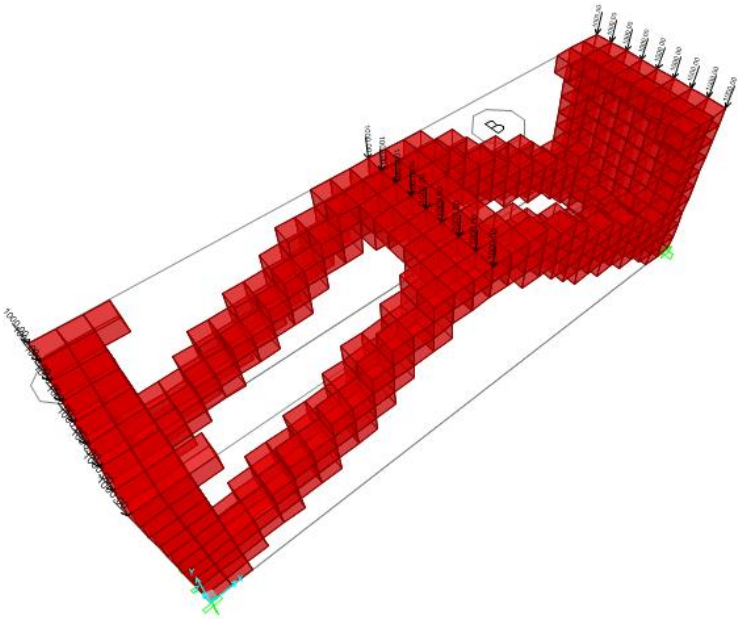
Παράδειγμα 5

Το πέμπτο παράδειγμα είναι ίδιο με το παράδειγμα 4, με την διαφορά ότι εφαρμόζονται επιπλέον συγκεντρωμένες δυνάμεις στα άνω στοιχεία του κέντρου.

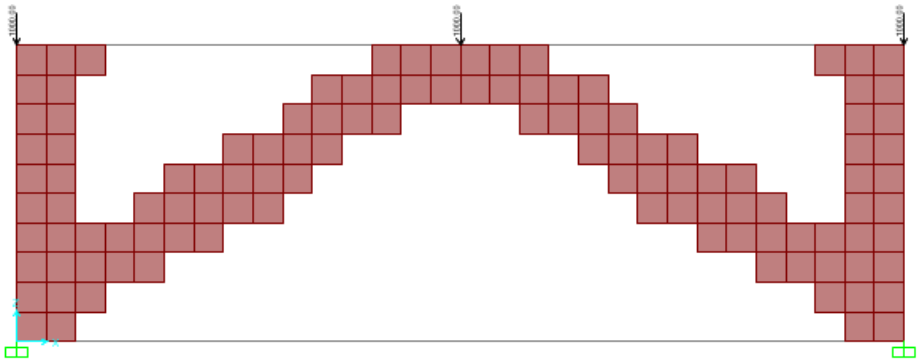


Εικόνα 6.27: Αρχικό σχήμα

Εφαρμόστηκαν και οι δυο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 30%. Η μέθοδος MMA σταμάτησε στο όριο των 150 επαναλήψεων και έδωσε το αποτέλεσμα που φαίνεται στις εικόνες 6.28 και 6.29:

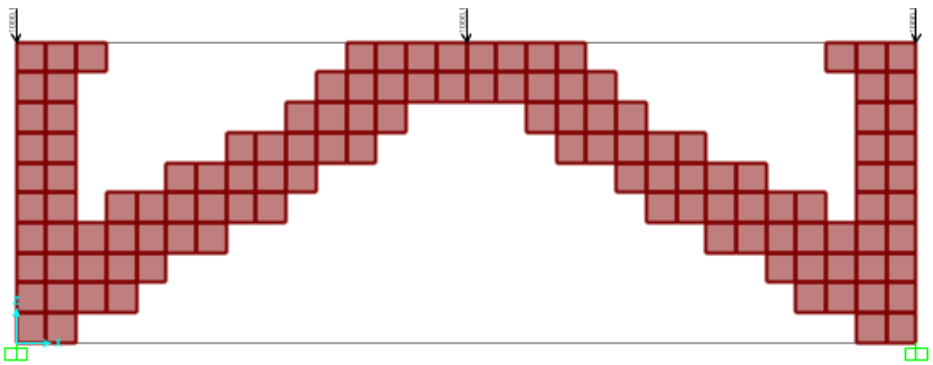


Εικόνα 6.28: Αποτέλεσμα ΜΜΑ – Όψη 3D



Εικόνα 6.29: Αποτέλεσμα ΜΜΑ – Όψη

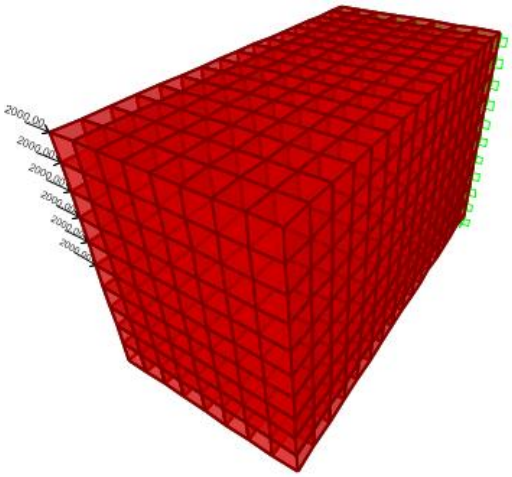
Η μέθοδος ΟC σταμάτησε επίσης στο όριο των 150 επαναλήψεων και έδωσε το ίδιο αποτέλεσμα με την ΜΜΑ:



Εικόνα 6.30: Αποτέλεσμα ΟC – Όψη

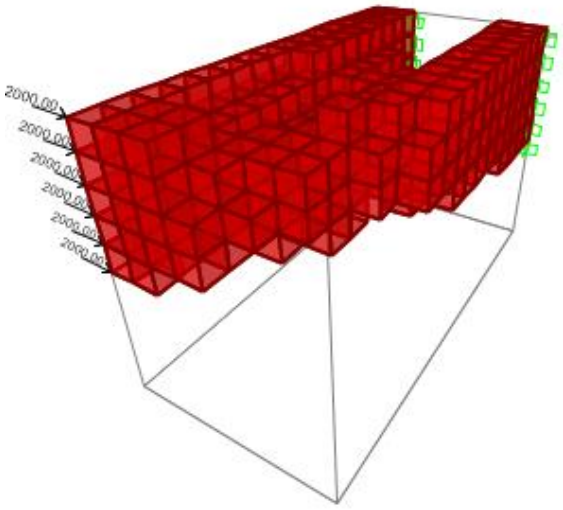
Παράδειγμα 6

Το έκτο παράδειγμα είναι ένας πρόβολος διαστάσεων 1.6m κατά τον διαμήκη άξονα x, 0.8m κατά τον εγκάρσιο άξονα y και 1.0m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=16, nly=8, nlz=10) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 1280 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m. Ορίζονται δεσμεύσεις σε όλα τα σημεία στην δεξιά πλευρά του προβόλου, ενώ εφαρμόζονται συγκεντρωμένες δυνάμεις κατά τον εγκάρσιο άξονα, όπως φαίνεται στην εικόνα 6.31:

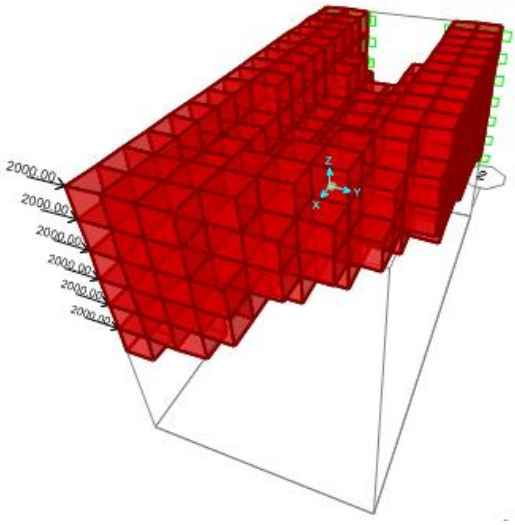


Εικόνα 6.31: Αρχικό σχήμα

Εφαρμόστηκαν και οι δύο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 30%. Και οι δύο μέθοδοι σταμάτησαν στο όριο των 150 επαναλήψεων που είχε οριστεί και έδωσαν τα εξής αποτελέσματα:



Εικόνα 6.32: Αποτέλεσμα MMA – Όψη 3D

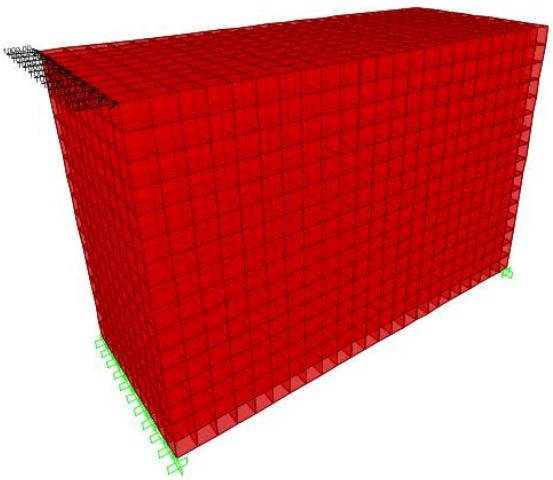


Εικόνα 6.33: Αποτέλεσμα ΟC– Όψη 3D

Όπως φαίνεται στις εικόνες 6.32 και 6.33 τα αποτελέσματα ήταν ίδια και σε αυτό το παράδειγμα.

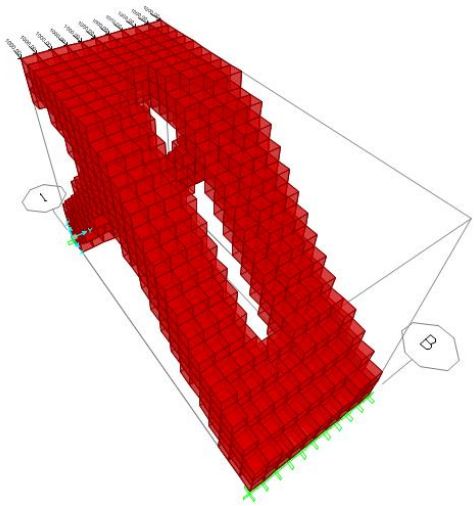
Παράδειγμα 7

Το έβδομο παράδειγμα είναι μια αμφιέρειστη δοκός διαστάσεων 1.2m κατά τον διαμήκη άξονα x, 0.5m κατά τον εγκάρσιο άξονα y και 0.8m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=24, nly=10, nlz=16) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 3840 τετραγωνικά στοιχεία με κάθε πλευρά 0.05m. Ορίζονται δεσμεύσεις στα κάτω σημεία κάθε πλευράς, ενώ εφαρμόζονται συγκεντρωμένες δυνάμεις στα άνω στοιχεία της εξωτερικής πλευράς κατά τον διαμήκη άξονα, όπως φαίνεται στην εικόνα 6.34:

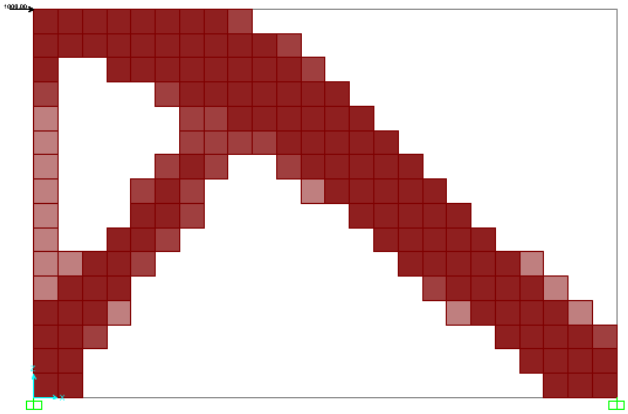


Εικόνα 6.34: Αρχικό σχήμα

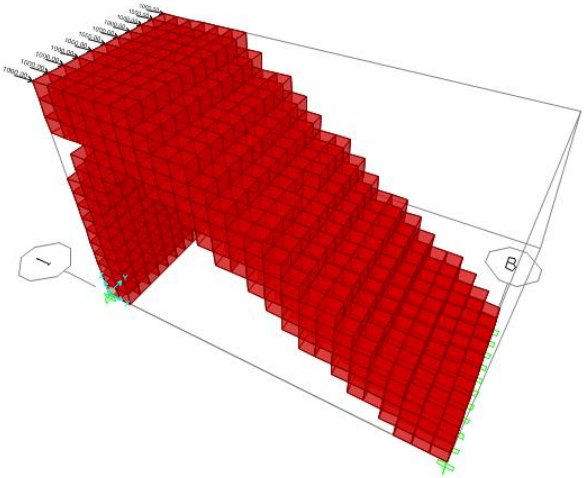
Εφαρμόστηκαν και οι δυο μέθοδοι βελτιστοποίησης τοπολογίας για ποσοστό όγκου 30%. Και οι δυο σταμάτησαν στο όριο των 150 επαναλήψεων και έδωσαν τα παρακάτω αποτελέσματα:



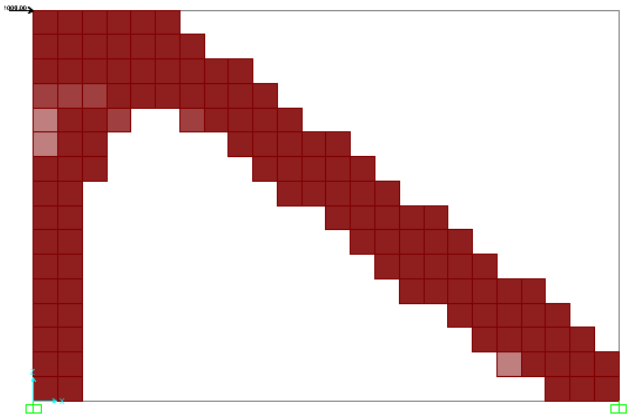
Εικόνα 6.35: Αποτέλεσμα OC– Όψη 3D



Εικόνα 6.36: Αποτέλεσμα OC– Όψη



Εικόνα 6.37: Αποτέλεσμα MMA – Όψη 3D



Εικόνα 6.38: Αποτέλεσμα MMA – Όψη

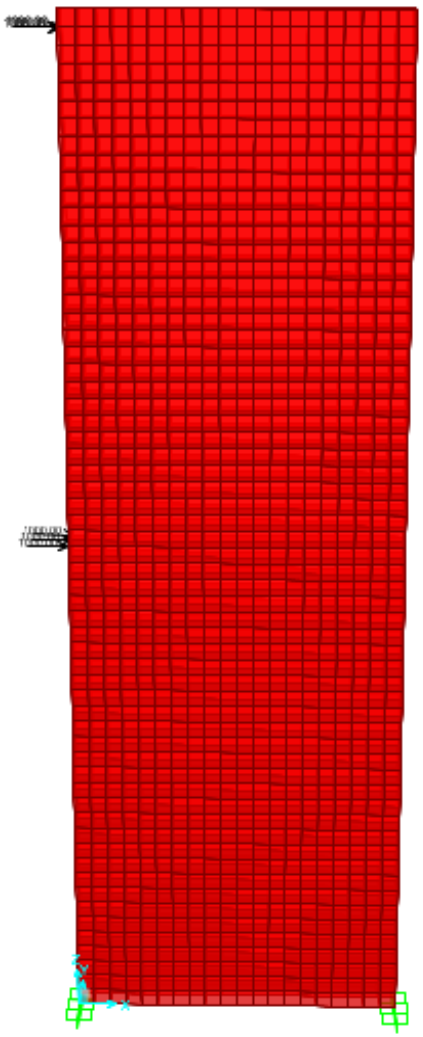
Το παράδειγμα αυτό ήταν το πρώτο στο οποίο παρατηρήθηκε ουσιαστική διαφορά μεταξύ των δύο μεθόδων. Συγκρίνοντας τα αποτελέσματα, αυτό που προέκυψε από την MMA μοιάζει ορθότερο. Αυτό δεν σημαίνει ότι το αποτέλεσμα που προέκυψε από την OC είναι λάθος καθώς και αυτό είναι λογικό.

6.3 Σύνθετα παραδείγματα

Όπως αναφέρθηκε στην αρχή, στόχος της διπλωματικής εργασίας ήταν η ανάπτυξη ενός λογισμικού βελτιστοποίησης τοπολογίας που θα έχει εφαρμογή σε προβλήματα πολιτικού μηχανικού. Έτσι, αφού αναπτύχθηκε το λογισμικό και ελέγχθηκε ότι λειτουργεί σωστά στα απλά παραδείγματα, το επόμενο βήμα ήταν να δοκιμαστεί σε ένα πραγματικό πρόβλημα, όπως η βελτιστοποίηση ενός τοιχίου.

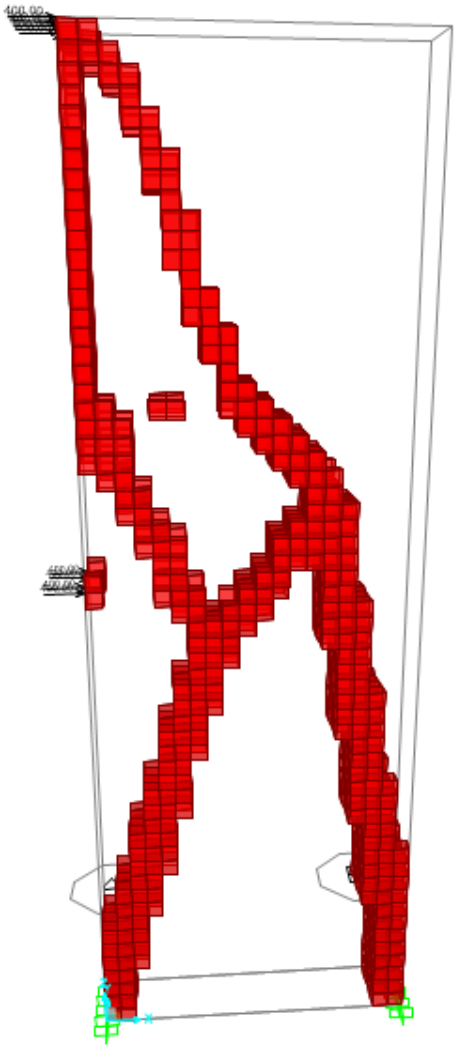
Παράδειγμα 1

Αρχικά, το παράδειγμα που εφαρμόστηκε ήταν ένα τοιχίο διαστάσεων 2m κατά τον διαμήκη άξονα x, 0.3m κατά τον εγκάρσιο άξονα y και 6m κατά τον κατακόρυφο άξονα z. Γίνεται διακριτοποίηση (nlx=20, nly=3, nlz=60) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 3600 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m. Οι δυνάμεις που ασκούνται και οι στηρίξεις φαίνονται στην εικόνα 6.39:

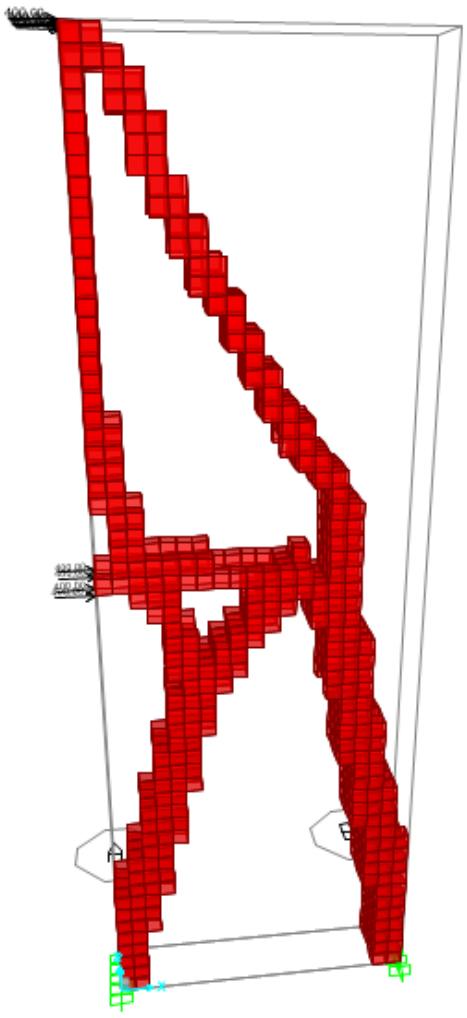


Εικόνα 6.39: Αρχικό σχήμα

Στην αρχή, εφαρμόστηκε η μέθοδος OC για 30% του όγκου, 75 επαναλήψεις και προέκυψε το σχήμα της εικόνας 6.40. Το σχήμα είναι λάθος καθώς δεν παραλαμβάνονται οι δυνάμεις που βρίσκονται στην μέση της κατασκευής. Παρ' όλα αυτά, το υπόλοιπο σχήμα έχει μια λογική, επομένως το επόμενο βήμα ήταν η εφαρμογή του ίδιου παραδείγματος για περισσότερες επαναλήψεις. Στην εικόνα 6.41 φαίνεται το αποτέλεσμα της μεθόδου OC για 250 επαναλήψεις. Το σχήμα που προέκυψε ήταν παρόμοιο με το προηγούμενο, αλλά αυτή τη φορά είχε διορθωθεί το πρόβλημα με τις δυνάμεις που ασκούνται στην μέση. Αυτό σημαίνει ότι το παράδειγμα αυτό χρειαζόταν περισσότερες επαναλήψεις για να βελτιστοποιηθεί σε σχέση με τα πιο απλά παραδείγματα, όπου σε κάποια αρκούσαν μόνο 20 επαναλήψεις.

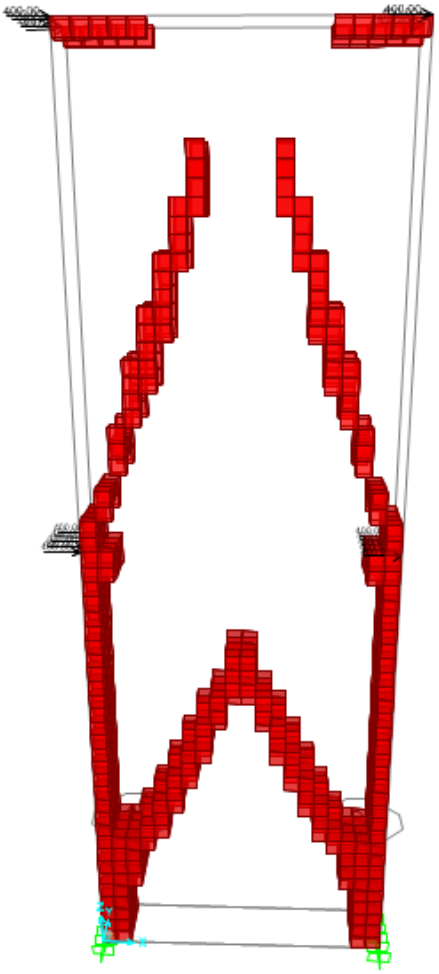


Εικόνα 6.40: Αποτέλεσμα OC - 75 επαναλήψεις

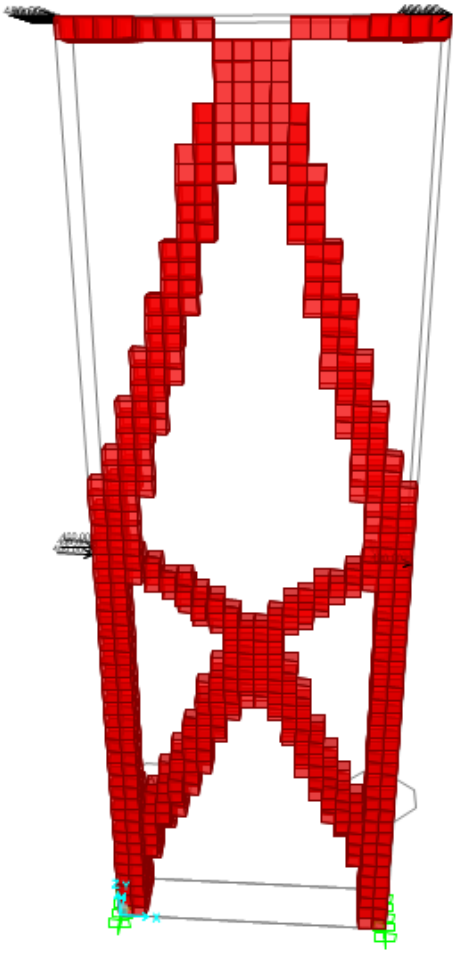


Εικόνα 6.41: Αποτέλεσμα OC - 250 επαναλήψεις

Χωρίς να να γίνουν αλλαγές στις διαστάσεις και στις συνθήκες στήριξης του τοιχίου, δοκιμάστηκε μια διαφορετική φόρτιση. Πλέον εκτός από τις θλιπτικές δυνάμεις που εφαρμόζονται στην μέση και στο πάνω μέρος της αριστερής πλευράς του τοιχίου, εφαρμόζονται και εφελκυστικές δυνάμεις στις αντίστοιχες θέσεις από την δεξιά πλευρά. Το αποτέλεσμα της μεθόδου OC για 30% του όγκου και 75 επαναλήψεις φαίνεται στην εικόνα 6.42, ενώ για 30% και 250 επαναλήψεις στην εικόνα 6.43.



Εικόνα 6.42: Αποτέλεσμα OC – 30%V, όριο διαγραφής 0.6, 75 επαναλήψεις

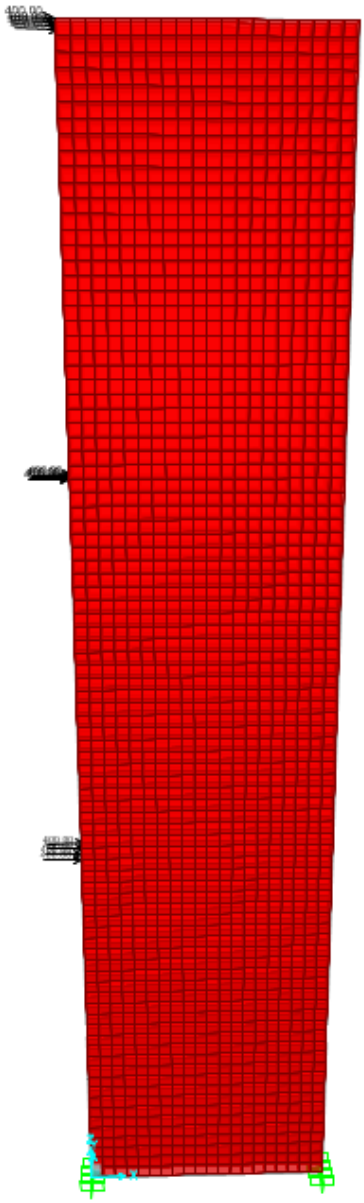


Εικόνα 6.43: Αποτέλεσμα OC – 30% V, όριο διαγραφής 0.6, 250 επαναλήψεις

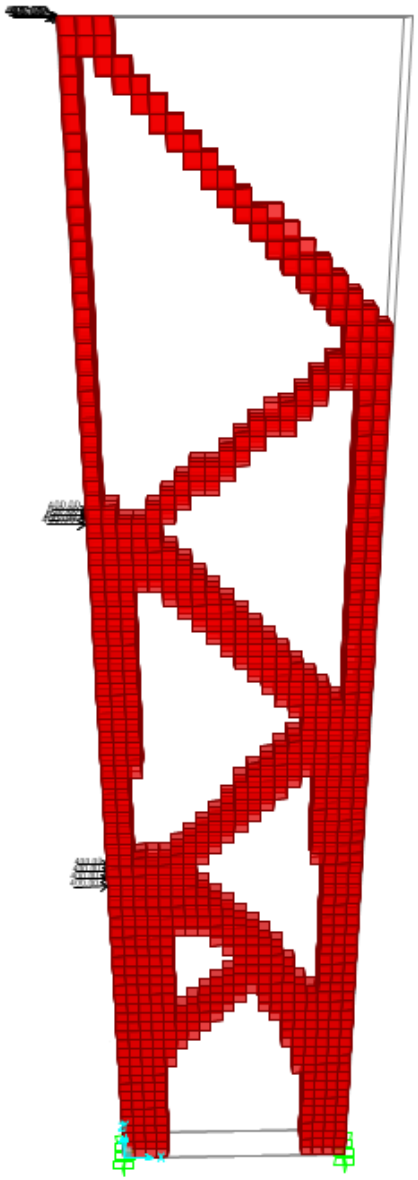
Εδώ, είναι εμφανές ότι για 75 επαναλήψεις η διαδικασία είναι ημιτελής και θέλει πολύ περισσότερες για να ολοκληρωθεί. Μάλιστα, στο σχήμα που προέκυψε από τις 250 επαναλήψεις υπάρχει ένα λάθος στο πάνω μέρος του τοιχίου, όπου λείπουν τέσσερα στοιχεία. Τα στοιχεία αυτά θα μπορούσαν να υπάρχουν, αν η μέθοδος λειτουργούσε για ακόμα περισσότερες επαναλήψεις ή αν μειωνόταν λίγο το όριο διαγραφής των στοιχείων. Δηλαδή αντί να διαγράφονται όλα τα στοιχεία με πυκνότητα κάτω του 0.6, να διαγράφονται αυτά με πυκνότητα κάτω του 0.5.

Παράδειγμα 2

Το δεύτερο παράδειγμα που εφαρμόστηκε ήταν ένα τοιχίο διαστάσεων 2m κατά τον διαμήκη άξονα x, 0.3m κατά τον εγκάρσιο άξονα y και 9m κατά τον κατακόρυφο άξονα z. Δηλαδή, σε σχέση με το προηγούμενο είχε ύψος 3m παραπάνω, ώστε να παραπέμπει σε τριώροφη κατασκευή. Γίνεται διακριτοποίηση (nlx=20, nly=3, nlz=90) στο αρχικό στοιχείο solid και δημιουργούνται συνολικά 5400 τετραγωνικά στοιχεία με κάθε πλευρά 0.1m. Οι δυνάμεις που ασκούνται και οι στηρίξεις φαίνονται στην εικόνα 6.44. Εφαρμόστηκε η μέθοδος OC για 40% του όγκου, 300 επαναλήψεις και προέκυψε το σχήμα της εικόνας 6.45.

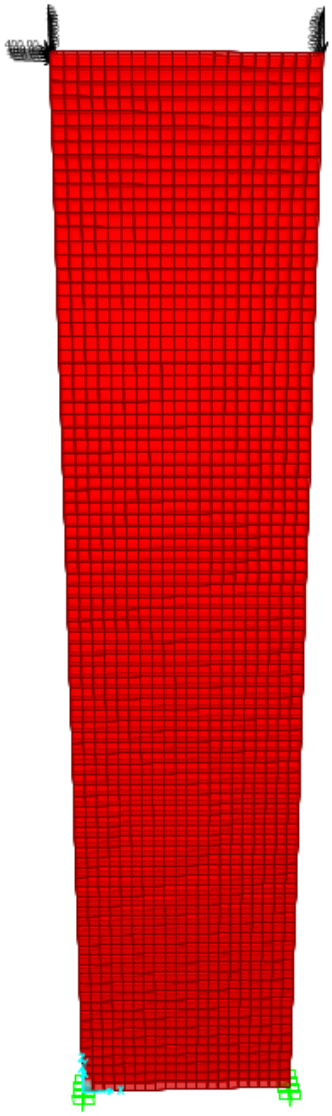


Εικόνα 6.44: Αρχικό σχήμα

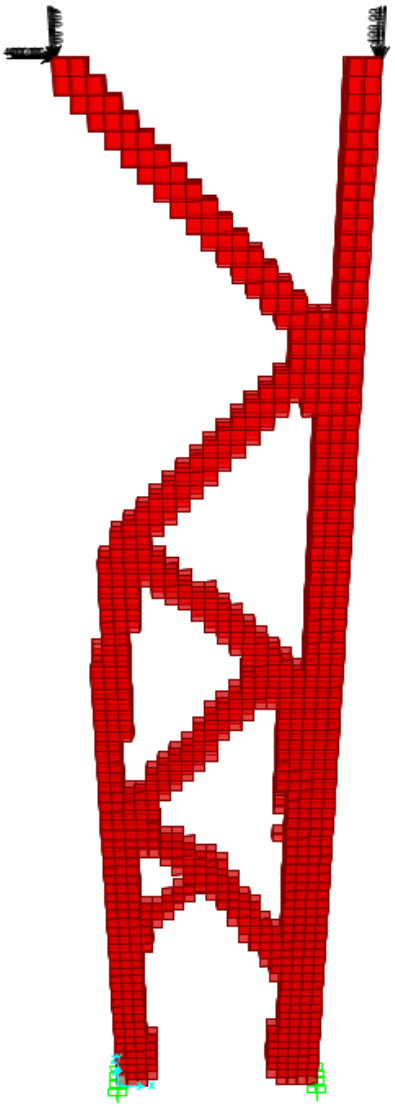


Εικόνα 6.45: Αποτέλεσμα OC – 40% V, όριο διαγραφής 0.6, 300 επαναλήψεις

Στην συνέχεια, χωρίς να γίνουν αλλαγές στις διαστάσεις και στις συνθήκες στήριξης του τοιχίου, δοκιμάστηκε μια διαφορετική φόρτιση που φαίνεται στην εικόνα 6.46. Αυτή τη φορά εφαρμόστηκε η μέθοδος MMA για 30% του όγκου, 300 επαναλήψεις και προέκυψε το σχήμα της εικόνας 6.47.



Εικόνα 6.46: Αρχικό σχήμα

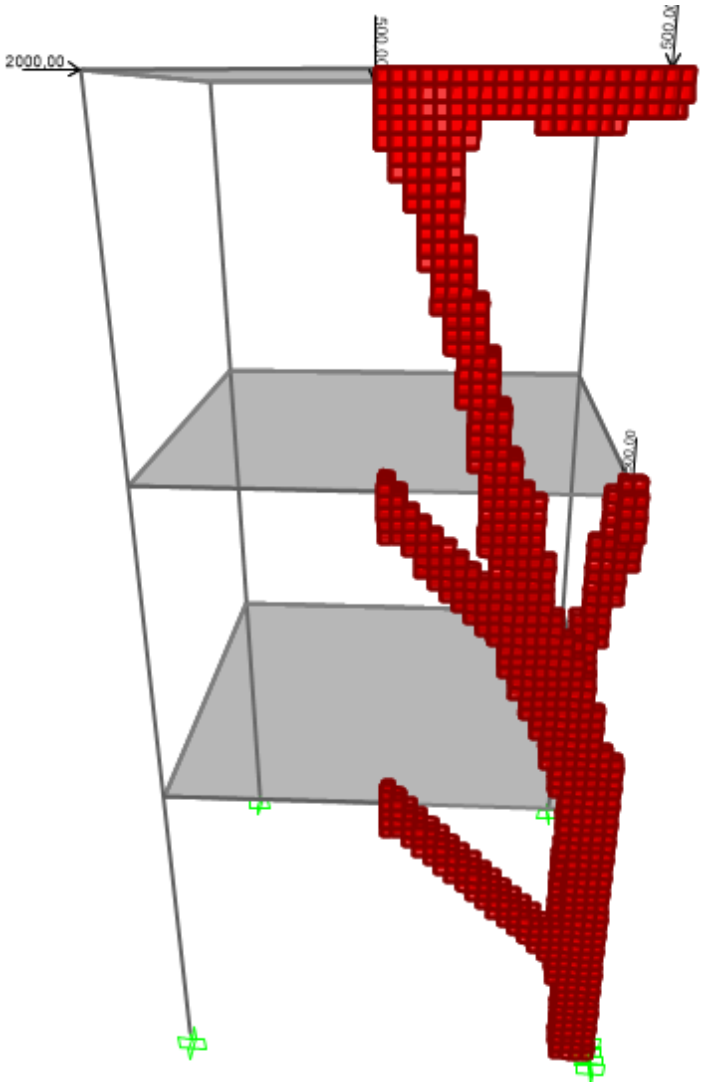


Εικόνα 6.47: Αποτέλεσμα MMA – 30% V όριο διαγραφής 0.6, 300 επαναλήψεις

Παρατηρήθηκε ότι το βελτιστοποιημένο σχήμα είναι παρόμοιο με αυτό έδωσε η προηγούμενη φόρτιση εκτός από τον «3^{ο} όροφο». Εκεί, ναι μεν παραλαμβάνει σωστά την συνισταμένη των δύο δυνάμεων με τον λοξό θλιπτήρα, αλλά θα ήταν καλύτερα αν τα κατακόρυφα στοιχεία της αριστερής πλευράς συνέχιζαν μέχρι πάνω για να ολοκληρωθεί το τρίγωνο, όπως συνέβη στην προηγούμενη φόρτιση. Το «λάθος» αυτό συνέβη διότι δεν λαμβάνεται υπόψιν το βάρος των στοιχείων. Γι' αυτό το λόγο, το λογισμικό μπορεί να χρησιμοποιηθεί για να δώσει ένα αρχικό σχήμα της βέλτιστης κατανομής του υλικού, αλλά

πρέπει πάντα να ελέγχεται από τον επιβλέποντα μηχανικό και να διορθώνεται κατά την κρίση του.

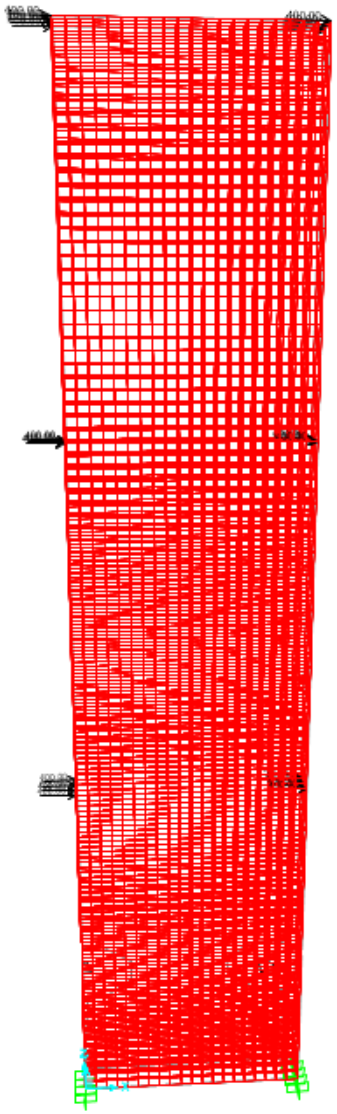
Πολύ ενδιαφέρον είναι το αποτέλεσμα του ίδιου τοιχίου, με την ίδια φόρτιση, αν ενσωματωθεί σε μια τριώροφη κατασκευή, όπου το συνολικό μήκος κάθε πλευράς είναι 4m. Στην κατασκευή αυτή εφαρμόστηκε πάλι η μέθοδος MMA για 30% του όγκου, 150 επαναλήψεις και το αποτέλεσμά της παρουσιάζεται στην εικόνα 6.48.



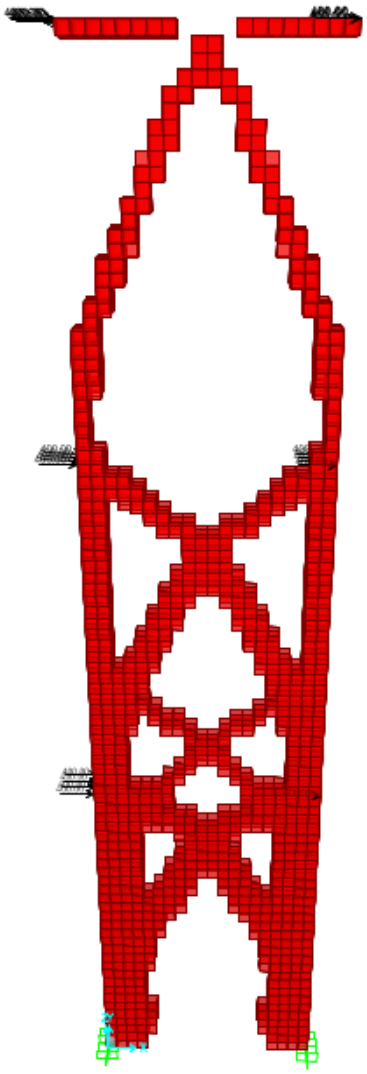
Εικόνα 6.48: Αποτέλεσμα MMA – 30% V,
όριο διαγραφής 0.6, 300 επαναλήψεις

Το λογισμικό θεώρησε ως στηρίξεις τις συνδέσεις δοκαριών-τοιχίου, με αποτέλεσμα το σχήμα να προκύψει τελείως διαφορετικό από το προηγούμενο. Αυτό δεν είναι σωστό, καθώς το τοιχίο πρέπει να στηρίζει τα δοκάρια και όχι τα δοκάρια να στηρίζουν το τοιχίο. Παρ' όλα αυτά το σχήμα μορφής κλαδιού δέντρου είναι ένα πολύ ενδιαφέρον αποτέλεσμα από το οποίο βγαίνει το συμπέρασμα ότι η βελτιστοποίηση τοπολογίας έχει την τάση να ακολουθεί τους κανόνες της φύσης.

Η επόμενη φόρτιση που δοκιμάστηκε στο δεύτερο παράδειγμα τοιχίου ήταν οριζόντιες δυνάμεις εκατέρωθεν σε κάθε «όροφο», όπως έγινε και στο πρώτο παράδειγμα. Το αρχικό σχήμα με την φόρτιση φαίνεται στην εικόνα 6.49. Η φόρτιση αυτή ήταν πιο απαιτητική για το λογισμικό και χρειάστηκε αρκετές προσπάθειες για να καταλήξει σε σωστό αποτέλεσμα. Παρακάτω θα παρουσιαστούν και θα σχολιαστούν τα αποτελέσματα της μεθόδου OC που έφτασαν πιο κοντά σε σωστή λύση. Αρχικά, στην εικόνα 6.50 φαίνεται το αποτέλεσμα που προέκυψε για 45% του όγκου και 400 επαναλήψεις.



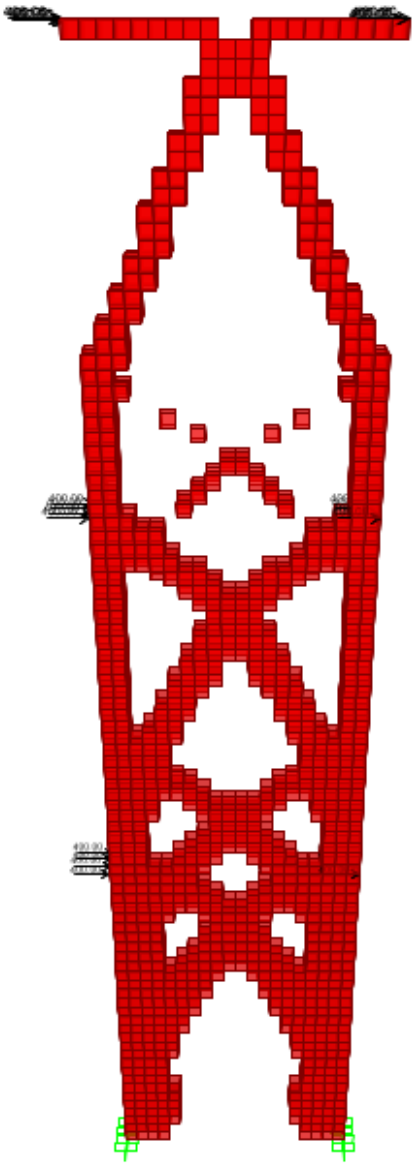
Εικόνα 6.49: Αρχικό σχήμα



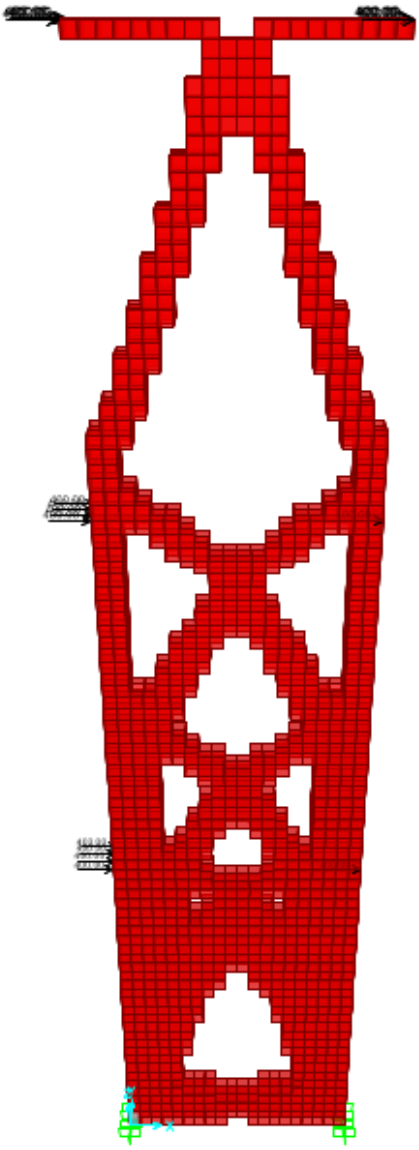
Εικόνα 6.50: Αποτέλεσμα OC - 45% V, όριο διαγραφής 0.6, 400 επαναλήψεις

Υπάρχει σημαντικό λάθος στο πάνω μέρος, όπου δεν ενώθηκαν τα οριζόντια στοιχεία με την υπόλοιπη κατασκευή. Εφόσον η μέθοδος εφαρμόστηκε για αρκετές επαναλήψεις , αυτό που οδήγησε στο λάθος κατά πάσα πιθανότητα, ήταν το όριο διαγραφής των στοιχείων. Μέχρι τώρα, σε όλα τα παραδείγματα διαγράφονταν τα στοιχεία με πυκνότητα κάτω του 0.6. Η επόμενη δοκιμή έγινε για 45% του όγκου, 400 επαναλήψεις και όριο διαγραφής 0.55. Στο σχήμα της εικόνας 6.51 φαίνεται ότι διορθώθηκε το λάθος στο πάνω μέρος της κατασκευής,

αλλά εμφανίστηκαν επιπλέον στοιχεία στο κέντρο. Αυτά τα στοιχεία είχαν πυκνότητα μεταξύ 0.55 και 0.6 και προσπαθούν να σχηματίσουν άλλο ένα «x». Τέλος, η δοκιμή που έδωσε το πιο σωστό αποτέλεσμα ήταν για 55% του όγκου, 400 επαναλήψεις και όριο διαγραφής 0.6. (Εικόνα 6.52). Τελικά, η αύξηση του όγκου δεν οδήγησε στην ολοκήρωση του «x» στο κέντρο, αλλά γέμισε στοιχεία στο κάτω μέρος της κατασκευής.

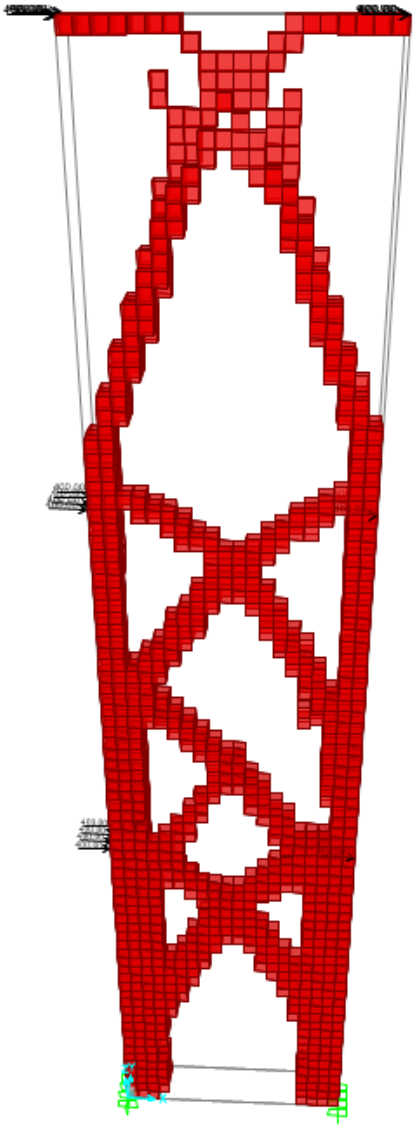


Εικόνα 6.51: Αποτέλεσμα OC - 45% V, όριο διαγραφής 0.5, 400 επαναλήψεις

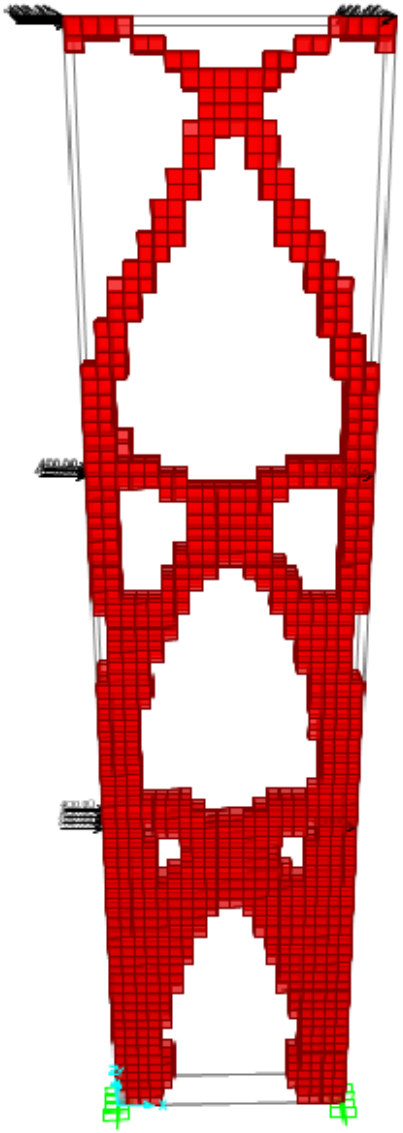


Εικόνα 6.52: Αποτέλεσμα OC - 55% V, όριο διαγραφής 0.6, 400 επαναλήψεις

Τέλος, εφαρμόστηκε και η μέθοδος MMA για την προηγούμενη φόρτιση του τοιχίου. Χρειάστηκαν πάλι αρκετές δοκιμές ώστε το λογισμικό να καταλήξει σε μια ολόσωστη λύση. Παρακάτω παρουσιάζονται τα καλύτερα αποτελέσματα που έδωσε η MMA για αυτή τη φόρτιση:



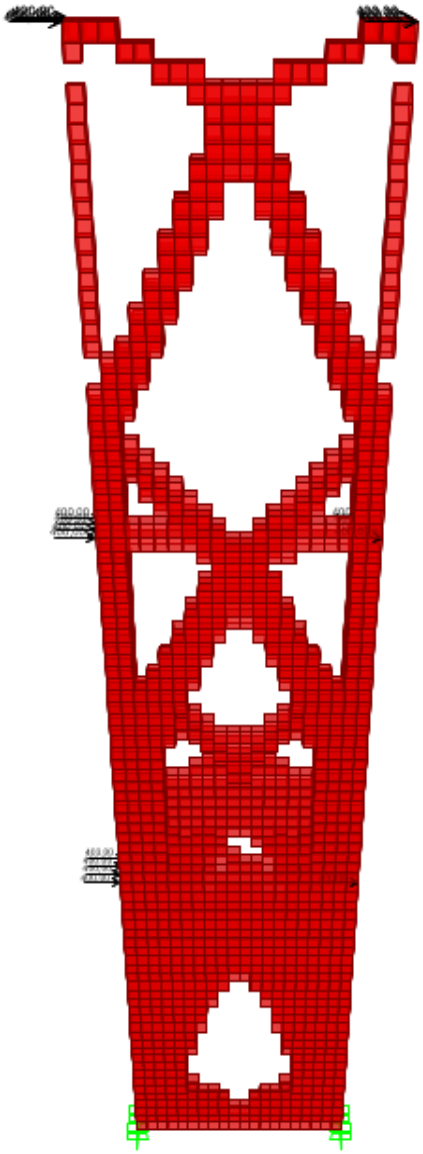
Εικόνα 6.53: Αποτέλεσμα MMA - 40% V, όριο διαγραφής 0.6, 400 επαναλήψεις



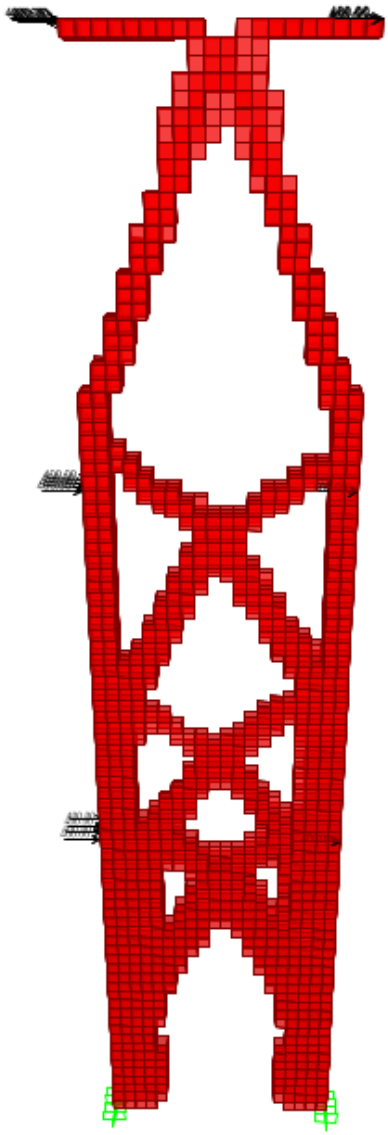
Εικόνα 6.54: Αποτέλεσμα MMA - 45% V, όριο διαγραφής 0.6, 400 επαναλήψεις

Στην εικόνα 6.53 φαίνεται το αποτέλεσμα της μεθόδου για 40% του όγκου και 400 επαναλήψεις. Υπάρχουν αρκετά λάθος στοιχεία στο πάνω μέρος, ενώ πιο κάτω δεν ολοκλήρωσε το μεσαίο «x». Η επόμενη δοκιμή ήταν για 45% του όγκου και 400 επαναλήψεις. Όπως δείχνει η εικόνα 6.54, άλλαξε η μορφή στο κέντρο, ενώ στο πάνω μέρος έχει λάθος «σκακιέρας». Από αυτά τα δύο σχήματα βγαίνει το συμπέρασμα ότι το μεσαίο «x», δεν είναι τόσο απαραίτητο όσο τα άλλα δύο, γι αυτό και στο πρώτο σχήμα δεν ολοκληρώθηκε, ενώ στο δεύτερο δεν δημιουργήθηκε καν.

Εφόσον η OC έδωσε το πιο σωστό αποτέλεσμα για 55% του όγκου, δοκιμάστηκε το ίδιο και με την μέθοδο MMA. Το αποτέλεσμα παρουσιάζεται στην εικόνα 6.55. Ενώ στο κάτω μέρος το σχήμα θυμίζει αυτό που έδωσε η OC, το πάνω μέρος είναι διαφορετικό, αφού το λογισμικό προσπαθεί να τοποθετήσει εκεί κατακόρυφα στοιχεία. Το τελικό αποτέλεσμα μπορεί να μην είναι καλό, αλλά δείχνει ποιο είναι το αμέσως επόμενο σημείο που χρειάζεται υλικό. Η μέθοδος MMA έδωσε το πιο σωστό αποτέλεσμα για 47% του όγκου και 400 επαναλήψεις (Εικόνα 6.56).



Εικόνα 6.55: Αποτέλεσμα MMA - 55% V, όριο διαγραφής 0.6, 400 επαναλήψεις



Εικόνα 6.56: Αποτέλεσμα MMA - 47% V, όριο διαγραφής 0.6, 400 επαναλήψεις

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα διπλωματική εργασία αναπτύχθηκε ένα λογισμικό βελτιστοποίησης τοπολογίας που να έχει εφαρμογή σε προβλήματα πολιτικού μηχανικού. Στο $6^{ο}$ κεφάλαιο παρουσιάστηκαν τα αποτελέσματα των αριθμητικών εφαρμογών και σε εδώ θα παρουσιαστούν τα συμπεράσματα που διαπιστώθηκαν αλλά και προτάσεις για περαιτέρω έρευνα και βελτίωση του λογισμικού.

Κατ' αρχάς το λογισμικό είναι αρκετά αργό καθώς το SAP2000 χρειάζεται πολύ χρόνο για να κάνει την κάθε ανάλυση. Μάλιστα, το πρόγραμμα δείχνει ότι χάνει τον περισσότερο χρόνο στην κατασκευή του μοντέλου ανάλυσης (creating model) και όχι στην ανάλυση (analyzing). Κατά την διάρκεια του «creating model» το SAP2000 μπορεί να εκτελεί οποιαδήποτε διαδικασία έχουν προγραμματίσει οι δημιουργοί του. Σε κάθε περίπτωση, μετά την ανάλυση το μοντέλο κλειδώνει και όταν στην συνέχεια ξεκλειδώσει για να αλλάξουν οι διατομές solid στα πεπερασμένα στοιχεία, όλα τα δεδομένα χάνονται. Οι διατομές solid όμως, δεν είναι κάθε φορά καινούργιες για να χρειάζεται για παράδειγμα σε κάθε επανάληψη ο υπολογισμός του τοπικού μητρώου δυσκαμψίας τους. Υπάρχουν από την αρχή 101 διαφορετικές διατομές οι οποίες εναλλάσσονται. Έτσι, σίγουρα θα υπάρχουν κάποιες διαδικασίες που θα μπορούσαν να γίνουν μία φορά και όχι σε κάθε επανάληψη.

Όσον αφορά τις επαναλήψεις διαπιστώθηκε ότι το λογισμικό χρειάζεται λίγες επαναλήψεις στα πιο απλά παραδείγματα ενώ στα πιο σύνθετα ήθελε σαφώς περισσότερες. Ως απλά, χαρακτηρίζονται τα παραδείγματα στα οποία υπήρχαν αρκετά στοιχεία και στις τρεις διαστάσεις έτσι ώστε το υλικό να μπορεί να κατανεμηθεί σε όποια κατεύθυνση «θέλει». Ένας δεύτερος παράγοντας που επηρεάζει την διαδικασία εύρεσης βέλτιστης λύσης, είναι οι δυνάμεις που ασκούνται. Πιο δύσκολες φορτίσεις, όπως η φόρτιση του τοιχίου εκατέρωθεν απαιτούσαν περισσότερες επαναλήψεις για να οδηγήσουν σε αποτέλεσμα.

Στα απλά παραδείγματα το λογισμικό όχι μόνο χρειάστηκε λίγες επαναλήψεις αλλά έδωσε πάντα σωστό αποτέλεσμα και με τις δυο μεθοδολογίες. Όμως, στα πιο δύσκολα παραδείγματα το αποτέλεσμα πολλές φορές δεν ήταν ολόσωστο. Όπως φάνηκε στο τοιχίο των 9 μέτρων, χρειάστηκαν πολλές δοκιμές στο ποσοστό όγκου ή και στο όριο διαγραφής των στοιχείων για να βρεθεί ένα αποτέλεσμα χωρίς κανένα λάθος. Αυτό το πρόβλημα οφείλεται σε τρεις πιθανούς λόγους:

- Πρώτον, μπορεί να αυξήθηκαν οι επαναλήψεις στα δύσκολα παραδείγματα, όμως το λογισμικό ίσως να χρειαζόταν ακόμα περισσότερες για να φτάσει σε ολόσωστο αποτέλεσμα.
- Δεύτερον, ακόμα και στα δύσκολα παραδείγματα σχηματικά η λύση ήταν πάντα σωστή, εκτός από μερικά στοιχεία τα οποία έλειπαν και δημιουργούσαν πρόβλημα σκακιέρας. Αυτό το πρόβλημα μπορεί να λυθεί με βελτίωση του φίλτρου. Το πιο εύκολο είναι να μεγαλώσει η ακτίνα r_{min} και έτσι κάθε στοιχείο να φιλτράρεται με βάση περισσότερα γειτονικά του στοιχεία.
- Τρίτο και σημαντικότερο, παραδείγματα όπως αυτά των τοιχίων στα οποία δεν υπάρχουν πολλά στοιχεία σε κάποια διάσταση, θα μπορούσαν να απλουστευθούν με μεγαλύτερη διακριτοποίηση. Έτσι, θα δημιουργηθούν πολλά περισσότερα στοιχεία και θα είναι πιο εύκολο για το λογισμικό να κατανείμει βέλτιστα το υλικό. Αν για παράδειγμα στο τοιχίο που είχε διαστάσεις 2x0.3x9 γινόταν διακριτοποίηση (nlx=40,nly=6,nlz=180), θα δημιουργούσε 43.200 στοιχεία, πολύ περισσότερα σε σχέση με τα 5400 που δημιουργήθηκαν για την διακριτοποίηση που έγινε (nlx=20,nly=3,nlz=90). Αυτό σίγουρα θα είχε καλύτερη συμπεριφορά και θα έδινε καλύτερα αποτελέσματα, όμως θα χρειαζόταν και πολύ περισσότερο χρόνο, ίσως σε απαγορευτικό βαθμό. Αν λυθεί το πρόβλημα του χρόνου που αναλύθηκε στην πρώτη παράγραφο, τότε θα μπορεί να γίνει μεγαλύτερη διακριτοποίηση που θα βελτιώσει τα αποτελέσματα.

Τέλος, σχετικά με τις δύο μεθοδολογίες, διαπιστώθηκε ότι η OC είχε καλύτερη συμπεριφορά και παρουσίασε λιγότερα προβλήματα. Αυτό κατ' αρχάς φάνηκε κατά την διάρκεια εποπτείας του έργου: Στην OC το έργο τις περισσότερες φορές είχε μια συνεχόμενα πτωτική πορεία και σταθεροποιόταν αρκετά γρηγορότερα σε σχέση με την MMA, όπου το έργο ταλαντευόταν περισσότερο. Επιπλέον, κατά την διάρκεια ελέγχου του κριτήριου τερματισμού, η μέγιστη διαφορά μεταξύ των πυκνοτήτων του κάθε στοιχείου, είχε και αυτή πτωτική πορεία στην OC. Αντίθετα, στην MMA παρατηρήθηκε περισσότερες φορές το ανώτατο όριο διαφοράς 0.2. Τα παραπάνω είχαν αντίκτυπο και στα αποτελέσματα καθώς μόνο η OC σταμάτησε κάποιες φορές ικανοποιώντας το κριτήριο τερματισμού και όχι το κριτήριο επαναλήψεων. Ακόμα, όπως φάνηκε στο τοιχίο των 9 μέτρων, η OC ήταν πιο σταθερή στη μορφή των αποτελεσμάτων της, ενώ η MMA αμφιταλαντεύτηκε περισσότερο, δίνοντας λίγο διαφορετικά σχήματα σε κάθε δοκιμή.