TECHNICAL UNIVERSITY OF CRETE

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΟΡΥΚΤΩΝ ΠΟΡΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΑΤΑΣΚΕΥΗ ΠΡΟΓΡΑΜΜΑΤΟΣ Η/Υ ΓΙΑ ΤΟΝ ΣΧΕΔΙΑΣΜΟ ΑΝΑΤΙΝΑΞΕΩΝ ΣΕ ΥΠΟΓΕΙΑ ΜΕΤΩΠΑ ΜΟΡΦΗΣ ΣΤΟΑΣ

ΚΑΤΣΑΝΟΣ ΧΡΗΣΤΟΣ

Εξεταστική Επιτροπή:

Εξαδάκτυλος Γεώργιος, Καθηγητής Πολυτεχνείου Κρήτης (Επιβλέπων καθηγητής)

Λιόλιος Παντελής, Ε.ΔΙ.Π Πολυτεχνείου Κρήτης

Γαλετάκης Μιχαήλ, Καθηγητής Πολυτεχνείου Κρήτης

Χανιά, Φεβρουάριος 2021

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Εξαδάκτυλο Γεώργιο καθώς και τον Δρ. Λιόλιο Παντελή τόσο για τη βοήθεια που μου προσέφεραν στην επιλογή θέματος διπλωματικής εργασίας όσο και για τη μεγάλη καθοδήγηση που μου έδιναν κατά την επεξεργασία της.

Επίσης θα ήθελα να ευχαριστήσω την εξεταστική επιτροπή, τον Δρ. Γαλετάκη Μιχαήλ.

ΠΕΡΙΛΗΨΗ

Σκοπός της διπλωματικής εργασίας είναι η δημιουργία ενός προγράμματος Η/Υ ,σε γλώσσα προγραμματισμού Python για το βέλτιστο σχεδιασμό ανατίναξης υπογείου μετώπου με τη μορφή στοάς με τη μέθοδο Holmberg. Αν αναλογισθεί κανείς ότι στις υπόγειες εκμεταλλεύσεις μεταλλείων το 25% αποτελούν στοές ανάπτυξης συν τις στοές προσπέλασης, μεταφοράς και αερισμού τότε γίνεται αντιληπτό γιατί η αποδοτική όρυξη στοών με την υβριδική τεχνική διάτρησης-ανατίναξης είναι ένα σημαντικό τεχνικό πρόβλημα.

Λαμβάνοντας υπ'όψην όλους τους περιορισμούς του σημαντικού αυτού τεχνικού προβλήματος όπως είναι οι σχεδιαστικοί περιορισμοί και οι περιορισμοί ή στόχοι της απόδοσης της εξόρυξης, εφαρμόστηκε η μέθοδος του Holmberg η οποία και προγραμματίστηκε σε γλώσσα προγραμματισμού Python για την επίλυσή του. Το αποτέλεσμα περιλαμβάνει τον τελικό πίνακα των εκρηκτικών και των υλικών έναυσης, το σχεδιασμό των διατρημάτων και τη σειρά πυροδότησης των υπονόμων για τη βέλτιστη διάτρηση και ανατίναξή της. Το τελικό πρόγραμμα περιλαμβάνει τη χρήση δύο αρχείων. Το πρώτο είναι σε μορφή .json που αφορά τις αρχικές γεωμετρικές παραμέτρους της στοάς και καθορίζεται από τον εκάστοτε χειριστή. Το δεύτερο αρχείο είναι της μορφής .py (Code_Blasting.py) το οποίο περιλαμβάνει τη μέθοδο Holmberg που υλοποιήθηκε σε γλώσσα προγραμματισμού Python. Ο χειριστής καλείται να θέσει τις αρχικές παραμέτρους στο αρχείο config.json και με βάση αυτές, το αρχείο-πρόγραμμα Code_Blasting.py θα υπολογίσει μέσω του κώδικα το βέλτιστο σχεδιασμό της ανατίναξης.

Στη συγκεκριμένη εργασία, για λόγους ασφαλείας προ-επιλέγεται η χρήση εκρηκτικού γαλακτώματος EM-EX σε φυσίγγια διαφορετικής διαμέτρου και μήκους. Στο κεντρικό παράδειγμα εφαρμογής της μεθόδου, η στοά που πρόκειται να διατρηθεί έχει ύψος 4 m και πλάτος 4.5 m με αψίδα ύψους 0.5 m. Κατόπιν η εφαρμοσιμότητα του αλγορίθμου ελέγχεται σε διάφορες γεωμετρίες διατομών στοών. Στο τέλος, γίνεται αξιολόγηση και σχολιασμός των αποτελεσμάτων.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΣΧΗΜΑΤΩΝ-ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

Στο πρώτο κεφάλαιο παρουσιάζεται το πρόβλημα που πρέπει να επιλυθεί.

1.1. Εξόρυξη πετρωμάτων

Από αρχαιοτάτων χρόνων, κρίνεται επιτακτική η ανάγκη του ανθρώπου για εξόρυξη πετρωμάτων τόσο για λόγους εκμετάλλευσης των πετρωμάτων αυτών όσο και για κατασκευαστικούς λόγους δημιουργίας δρόμων, σηράγγων και άλλων τεχνικών έργων. Από την εποχή του Χαλκού (5^{η} χιλιετία π.Χ.) έως και τη σημερινή εποχή, ο άνθρωπος έπρεπε να αναπτύξει την τεχνογνωσία και την τεχνολογία για να διεισδύσει κάτω από την επιφάνεια του εδάφους.

Όπως είναι αντιληπτό, δεν αποτελεί μια απλή διαδικασία. Τόσο για τη διεκπεραίωση μεταλλευτικών έργων όσο και για κατασκευαστικές δραστηριότητες είναι απαραίτητο ο άνθρωπος να διανοίγει υπόγειες στοές ή σήραγγες που θα τον οδηγήσουν στο κατάλληλο βάθος του υπεδάφους πάνω στο οποίο θα εκτελέσει τα έργα, σε συνθήκες που θα το επιτρέπουν. Κάθε τέτοιο τεχνικό έργο απαιτεί εκτενείς έρευνες πριν την υλοποίηση, κατάλληλο μηχανοποιημένο εξοπλισμό και την επιλογή κατάλληλων μεθόδων διάνοιξης. Οι ασφαλιστικές δικλείδες που πρέπει να λαμβάνονται υπ'όψην σε τέτοιας μορφής υπόγεια έργα, σε συνδυασμό με τη μεγαλύτερη αναλογία κέρδους/κόστους δημιούργησαν την ανάγκη στον άνθρωπο να επινοήσει συγκεκριμένες μεθόδους σχεδιασμού ανατινάξεων υπογείων μετώπων με τη μορφή στοάς. Οι μέθοδοι αυτές χαρακτηρίζονται από μεγάλη ακρίβεια τόσο για την ομαλή υλοποίηση του έργου όσο και για την αντιμετώπιση ενδεχομένων αστοχιών.

Η διαδικασία εξόρυξης πετρωμάτων με τη χρήση εκρηκτικών υλών για το θρυμματισμό τους ονομάζεται διαδικασία διάτρησης-ανατίναξης. Διάτρημα ορίζεται ως ένα μικρής διαμέτρου κυλινδρικό άνοιγμα και μπορεί να είναι κατακόρυφο οριζόντιο ή κεκλιμένο. Κατά τη διάτρηση του πετρώματος διανοίγονται διατρήματα μεθοδευμένα σε όλο το πλάτος του υπογείου μετώπου, τα οποία θα γομωθούν με εκρηκτική ύλη και έπειτα θα ανατιναχθούν με σκοπό το θρυμματισμό του πετρώματος. Κάθε διάτρημα για να εναυθεί επιτυχώς απαιτεί την ύπαρξη τουλάχιστον μιας ελεύθερης επιφάνειας με σκοπό την ομαλή εκτόνωση των αερίων της ανατίναξης. Οι υπόγειες ανατινάξεις χωρίζονται σε δύο κατηγορίες ανάλογα με τις ελεύθερες επιφάνειες που υπάρχουν.

Κατά καιρούς έχουν αναπτυχθεί αρκετές μέθοδοι διάτρησης-ανατίναξης υπογείου μετώπου. Αυτό συμβαίνει λόγω των διαφορετικών διατομών της κάθε στοάς, των χαρακτηριστικών των διαφόρων τυπών πετρωμάτων καθώς και λόγω του διαφορετικού διαθέσιμου μηχανοποιημένου εξοπλισμού. Μεταβλητή όλων αυτών των μεθόδων είναι η αρχική διάνοιξη του πετρώματος ή αλλιώς η προεκσκαφή. Βασική αρχή της αρχικής διάνοιξης είναι η εξασφάλιση ελευθέρων επιφανειών για τη βελτίωση του μηχανισμού θραύσης του πετρώματος από την ανατίναξη των υπολοίπων διατρημάτων. Οι πιο διαδεδομένες μέθοδοι αρχικής διάνοιξης είναι οι αρχικές κοπές υπό γωνία και οι παράλληλες αρχικές κοπές.

Αρχικές κοπές υπό γωνία

Επιτυγχάνεται με την όρυξη διατρημάτων σε ζεύγη τα οποία πρέπει να συγκλίνουν σε ένα σημείο στο βάθος του διατρήματος. Ιδανικά, όσο περισσότερο πλησιάζουν οι πυθμένες μεταξύ τους, τόσο καλύτερος είναι και ο θρυμματισμός του πετρώματος. Διακρίνονται σε:

- Πυραμιδοειδής κοπή (pyramid cut)
- Σφηνοειδής κοπή (V cut)
- Κοπή ριπιδίου (fan cut)
- Χαμηλή κοπή (draw cut)

Παράλληλες αρχικές κοπές

Η μέθοδος αυτή αφορά στην όρυξη παραλλήλων διατρημάτων τα οποία είναι διατεταγμένα κατά τέτοιο τρόπο ώστε οι αποστάσεις μεταξύ τους να είναι πολύ μικρές. Κάποια από αυτά θα γομωθούν με εκρηκτική ύλη ενώ κάποια άλλα όχι. Με τη μέθοδο αυτή επιτυγχάνεται μεγαλύτερη προχώρηση από αυτή στη μέθοδο των αρχικών κοπών υπό γωνία. Εκτός των άλλων, η μέθοδος αυτή δεν εξαρτάται από τη γεωμετρία και τη διατομή της στοάς και δεν απαιτεί την αλλαγή κλίσης του διατρητικού (όπως στις κοπές υπό γωνία), πράγμα που σημαίνει την εξοικονόμηση αρκετού χρόνου.

Η επιλογή της κατάλληλης μεθόδου γίνεται με βάση τις δυνατότητες του χειριστή του διατρητικού μηχανήματος, τη διατομή της στοάς και κυρίως με τον διαθέσιμο μηχανοποιημένο εξοπλισμό. Με την ανατίναξη των διατρημάτων της προεκσκαφής δημιουργείται η ελεύθερη επιφάνεια για την ομαλή ανατίναξη των υπολοίπων διατρημάτων.

Η μέθοδος Holmberg αφορά στο βέλτιστο σχεδιασμό διάτρησης ανατίναξης υπογείου μετώπου με τη χρήση παραλλήλων αρχικών κοπών κατά την προεκσκαφή.

1.2. Σκοπός Διπλωματικής Εργασίας

Οι μέθοδοι που αφορούν στο σχεδιασμό ανατινάξεων υπογείων μετώπων με τη μορφή στοάς, περιλαμβάνουν πλήθος υπολογισμών με αρκετά μεγάλη απαιτούμενη ακρίβεια για την αποφυγή σφαλμάτων σχεδιασμού που μπορεί να έχουν μεγάλη επίδραση τόσο στην ασφάλεια του έργου όσο και στον οικονομικό προϋπολογισμό του, μιας και μιλάμε για διαδικασίες που γίνονται σε αρκετά μεγάλο βάθος κάτω από την επιφάνεια του εδάφους. Με την συνεχόμενη εξέλιξη της τεχνολογίας ο άνθρωπος έχει αναπτύξει μηχανοποιημένο εξοπλισμό που βοηθά στην εξάλειψη ανθρωπίνων σφαλμάτων για τις διαδικασίες τέτοιων τεχνικών έργων στο υπέδαφος. Παρ'όλα αυτά είναι εκείνος που παίρνει τις αποφάσεις και εκείνος που χειρίζεται τον μηχανοποιημένο εξοπλισμό. Επομένως οι τεχνικές και οι μέθοδοι που θα χρησιμοποιηθούν από τη θεωρία στην πράξη επιλέγονται από τον άνθρωπο.

Σκοπός της παρούσας Διπλωματικής Εργασίας είναι η ένταξη της μεθόδου σχεδιασμού υπογείων ανατινάξεων του Holmberg (1975) σε υπολογιστικό περιβάλλον, σε γλώσσα προγραμματισμού Python. Κύριο μέλημα της εργασίας είναι να παρουσιάσει τον βέλτιστο σχεδιασμό της ανατίναξης με τη μέθοδο του Holmberg ανάλογα με τις παραμέτρους που θέτει ο εκάστοτε χειριστής και αφορούν στις διαφορετικές γεωμετρίες της στοάς και τη χρήση διαφορετικών τύπων εκρηκτικών υλών. Με λίγα λόγια, με την εισαγωγή των γεωμετρικών χαρακτηριστικών της στοάς που απαιτείται να ανατιναχθεί και με τον τύπο της εκρηκτικής ύλης που θα χρησιμοποιηθεί, το πρόγραμμα θα παρουσιάζει την κατάλληλη δομή και διάταξη (διάγραμμα) των διατρημάτων που θα ορυχθούν στο υπόγειο μέτωπο καθώς και τις πυκνότητες εκρηκτικής γόμωσης του κάθε διατρήματος ανάλογα με τη χρήση του. Όλα τα παραπάνω θα περικλείονται σε ένα περιβάλλον υπολογιστή, με δεδομένες τις ασφαλιστικές παραμέτρους βάσει της μεθόδου του Holmberg και με απώτερο σκοπό την ταχύτητα των υπολογισμών και την εξάλειψη οποιουδήποτε ανθρωπίνου λάθους σε αυτούς. Το πρόγραμμα χρησιμοποιείται σε γλώσσα Python λόγω της εύκολης προσβασιμότητάς της και της εύκολης ανάγνωσής της από αρχάριους και μη προγραμματιστές και κυρίως λόγω των απεριόριστων πακέτων υποστήριξής της (ή αλλιώς βιβλιοθήκες).

ΚΕΦΑΛΑΙΟ 2 : ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

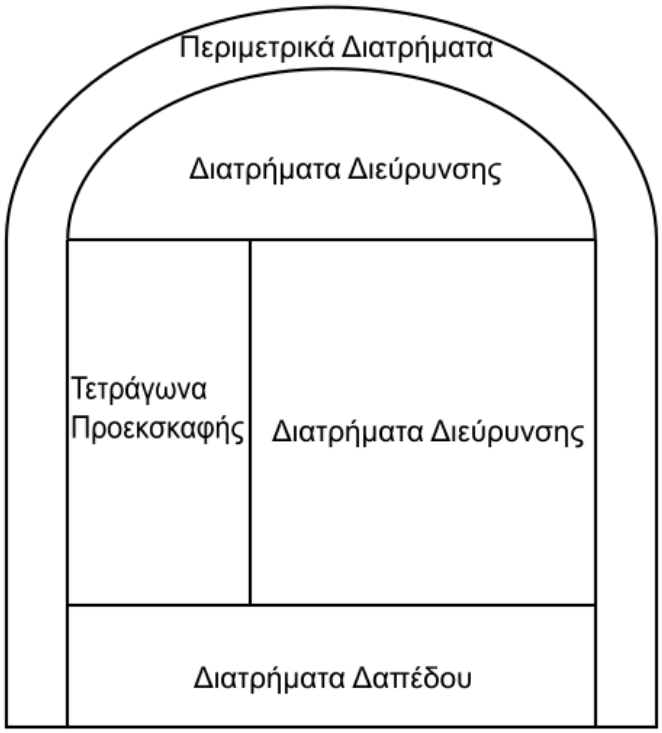
Στο κεφάλαιο αυτό επισημαίνεται το θεωρητικό υπόβαθρο της Διπλωματικής Εργασίας. Όλη η θεωρία της μεθόδου Holmberg καθώς επίσης και οι παράμετροι του προγράμματος με μικρή αναφορά στην εκρηκτική ύλη που χρησιμοποιείται.

2.1. Μέθοδος R. Holmberg (1975)

Η μέθοδος του Σουηδού R. Holmberg (1975) αποτελεί μία μέθοδο σχεδιασμού υπογείων ανατινάξεων με παράλληλη διάταξη διατρημάτων προεκσκαφής. Η βασική αρχή κατά την παράλληλη διάταξη είναι η όρυξη ενός συνόλου παράλληλων διατρημάτων (κάποια από τα οποία περιέχουν εκρηκτική γόμωση και άλλα όχι) τα οποία είναι διατεταγμένα με τέτοιο τρόπο ώστε οι αποστάσεις μεταξύ τους να είναι αρκετά μικρές. Τα διατρήματα της προεκσκαφής σχηματίζουν τα τετράγωνα προεκσκαφής γύρω από ένα ή δύο ή και -σπανιότερα- περισσότερα κενά διατρήματα μεγάλης διαμέτρου (διατρήματα ανακούφισης).

Η μέθοδος αυτή αφορά στον ακριβή υπολογισμό των φορτίων των διατρημάτων για κάθε τομέα της διάτρησης. Φορτίο διατρήματος ορίζεται ως η απόσταση του διατρήματος απο την πλησιέστερη ελεύθερη επιφάνεια. Κατά τη διαδικασία της ανατίναξης θα πρέπει να εξασφαλίζεται μια ελεύθερη επιφάνεια για κάθε διάτρημα που ανατινάσσεται. Με τον τρόπο αυτό τα εκλυόμενα αέρια της ανατίναξης θα μπορούν να εκτονωθούν ομαλά προς την ελεύθερη επιφάνεια και η ανατίναξη του διατρήματος θα είναι επιτυχής. Υπάρχουν δύο ειδών ελεύθερης επιφάνειας. Η στατική ελεύθερη επιφάνεια αποτελεί την ελεύθερη επιφάνεια που υπάρχει στο χώρο πριν τη διαδικασία της ανατίναξης (επιφάνεια του μετώπου), ενώ η δυναμική ελεύθερη επιφάνεια αναφέρεται στην ελεύθερη επιφάνεια που θα δημιουργηθεί από την ανατίναξη του διατρήματος.

Στη μέθοδο του Holmberg (1975), διαχωρίζεται το υπόγειο μέτωπο σε 5 τμήματα. Τα τμήματα αυτά είναι τα τετράγωνα προεκσκαφής *(cut)* , τα ανυψωτικά διατρήματα ή διατρήματα δαπέδου *(lifters)*, τα διατρήματα της οροφής *(roof holes)*, τα διατρήματα του τοίχου *(wall holes)* και τα διατρήματα διεύρυνσης ή κύριας προχώρησης *(stoping holes)*. Στα διατρήματα της οροφής χρησιμοποιούμε τεχνική ελεγχόμενης ανατίναξης *(smooth blasting)* για την αποφυγή «πληγώματος» (damage) του πετρώματος εκτός των ορίων του υπογείου μετώπου, με χαμηλότερη εκρηκτική γόμωση διατρημάτων. Τα διατρήματα της οροφής και των παρειών λέγονται επίσης και περιμετρικά διατρήματα *(contour holes)* που καθορίζουν το σύνορο του μετώπου.



Σχήμα 2.1. Τμήματα σχεδιασμού ανατίναξης υπογείου μετώπου μορφής στοάς

Στο *Σχήμα 2.1.* απεικονίζονται τα τμήματα σχεδιασμού ανατίναξης υπογείου μετώπου με τη μορφή στοάς. Περιμετρικά είναι τα διατρήματα τοίχου και οροφής. Στο αριστερό άκρο του μετώπου αναπτύσσονται τα διατρήματα προεκσκαφής που σχηματίζουν τετράγωνα (*Cut*). Στο χώρο που απομένει (δεξιά και πάνω από τα τετράγωνα προεκσκαφής) ορύσσονται τα διατρήματα διεύρυνσης *(Stoping Holes)* ενώ το τελευταίο τμήμα αφορά στα διατρήματα δαπέδου (*Lifters*).

2.1.1. Σύγκριση εκρηκτικών υλών

Αρχικό στάδιο πριν την έναρξη των υπολογισμών των φορτίων, είναι η επιλογή της κατάλληλης εκρηκτικής ύλης και των ιδιοτήτων της. Η διαδικασία αυτή προϋποθέτει σύγκριση διαφορετικών εκρηκτικών υλών. Διαχρονικά αναπτύχθηκαν πολλές μέθοδοι σύγκρισης εκρηκτικών αλλά τα αποτελέσματα δεν περιείχαν την επιθυμητή ακρίβεια. Αυτό οφείλεται στο γεγονός πως οι δοκιμές θα έπρεπε να περιλαμβάνουν τη χρήση όλων των εκρηκτικών σε όλους τους τύπους πετρωμάτων υπό κάθε πιθανή συνθήκη που επικρατεί. Για το λόγο αυτό, η επικρατέστερη μέθοδος επιλογής του κατάλληλου εκρηκτικού είναι ο υπολογισμός της σχετικής ισχύος του, δηλαδή σύγκριση του εκρηκτικού που μελετάται ως προς ένα άλλο εκρηκτικό που έχει τεθεί σαν σημείο αναφοράς.

Για τον παρακάτω υπολογισμό γίνεται σύγκριση του εκρηκτικού που χρησιμοποιείται ως προς δυναμίτιδα LFB. Η σχέση που υλοποιεί τη σύγκριση μεταξύ των δύο εκρηκτικών είναι η Σουηδική σχέση υπολογισμού σχετικής ισχύος (Langefors, U., and Kihlstrom, B., 1963, *The Modern Technique of Rock Blasting*, Almqvist and Wiskell, Stockholm) :

$$s = \frac{5}{6} * \frac{Q}{Qo} + \frac{1}{6} * \frac{V}{Vo} \qquad \text{[Εξ. 1]}$$

Όπου *s* η σχετική ισχύς του εκρηκτικού ως προς δυναμίτιδα LFB, *Qo* η εκλυόμενη θερμότητα κατά την ανατίναξη 1 kg δυναμίτιδας LFB, *Vo* ο Εκλυόμενος όγκος αερίων κατά την ανατίναξη 1 kg δυναμίτιδας LFB, *Q* η εκλυόμενη θερμότητα κατά την ανατίναξη 1 kg εκρηκτικού που χρησιμοποιείται και *V* ο αντίστοιχος εκλυόμενος όγκος αερίων κατά την ανατίναξή του.

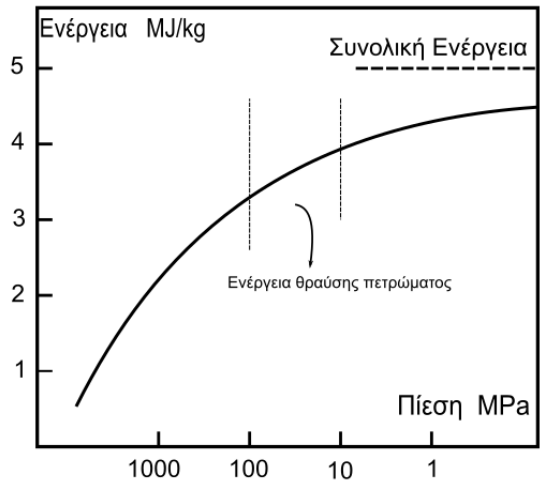
Η παραπάνω εξίσωση ουσιαστικά έχει σαν βάση το γεγονός πως η θραύση του πετρώματος που προκαλείται από την εκρηκτική ύλη (Ε/Υ) εξαρτάται κυρίως από την εκλυόμενη ενέργεια κατά την ανατίναξη και σε μικρότερο βαθμό από τον όγκο των αερίων που απελευθερώνονται. Οι εμπειρικές σχέσεις 5/6 και 1/6 προκύπτουν από δοκιμές σε πεδία υπαίθριων εκμεταλλεύσεων με χρήση διαφόρων εκρηκτικών που συγκρίθηκαν και μελετήθηκαν βάσει της δυναμίτιδας LFB. Παρ'όλα αυτά, πλέον, έχει καθιερωθεί η ανάλυση εκκρητκικών υλών βάσει του πετραμμωνίτη ANFO. Σε αυτή την περίπτωση, η σχετική ισχύς της εκρηκτικής ύλης (Ε/Υ) προσδιορίζεται πρωτίστως ως προς τη δυναμίτιδα LFB και μετέπειτα ως προς τον πετραμμωνίτη βάσει της εξίσωσης $S_{ANFO}=S_{LFB}/0.84$.

Στον *Πίνακα 2.1.* παρουσιάζεται ενδεικτικά η σχετική ισχύς κατά βάρος ορισμένων εκρηκτικών υλών ως προς τη δυναμίτιδα LFB.

Πίνακας 2.1. Σχετική ισχύς ορισμένων εκρηκτικών υλών ως προς τη δυναμίτιδα LFB

Explosive	Q, (MJ/kg)	V, (m^3/kg)	S_{LFB}	$S_{DX\ B}$	S_{ANFO}	Density, (kg/m^3)
LFB Dynamite	5.00	0.850	1.00	1.09	1.19	-
Dynamex B	4.6	0.765	0.92	1.00	1.10	1450
ANFO	3.92	0.973	0.84	0.91	1.00	900
TNT	4.1	0.690	0.82	0.89	0.98	1500
PETN	6.12	0.780	1.17	1.27	1.39	-
NABIT	4.1	0.892	0.86	0.93	1.02	1000
GURIT	3.73	0.425	0.71	0.77	0.85	1000

Η σχετική ισχύς κατά βάρος περιγράφει το θρυμματισμό του πετρώματος που προκαλείται από την ανατίναξη της εκρηκτικής ύλης (Ε/Υ). Η εκλυόμενη ενέργεια από την ανατίναξη αυτή δεν χρησιμοποιείται όλη για το θρυμματισμό του πετρώματος. Για να γίνει αυτό, θα έπρεπε ο όγκος των αερίων της ανατίναξης να απελευθερώνεται σε πολύ μικρή πίεση. Η απόδοση της ανατίναξης οφείλεται και στα συστατικά της εκρηκτικής ύλης (Ε/Υ). Για παράδειγμα, οι εκρηκτικές ύλες που αποτελούνται από αλουμίνιο έχουν μεγάλη συνολική εκλυόμενη ενέργεια όμως ο θρυμματισμός του πετρώματος λαμβάνει μέρος στο σημείο που η πίεση είναι χαμηλή. Για το λόγο αυτό η απόδοση της ανατίναξης μιας τέτοιας εκρηκτικής ύλης είναι πολύ μειωμένη. Στο παρακάτω *Σχήμα 2.2.*, παρουσιάζεται το διάγραμμα ενέργειας – πίεσης για εκρηκτικό γαλάκτωμα αλουμινίου.



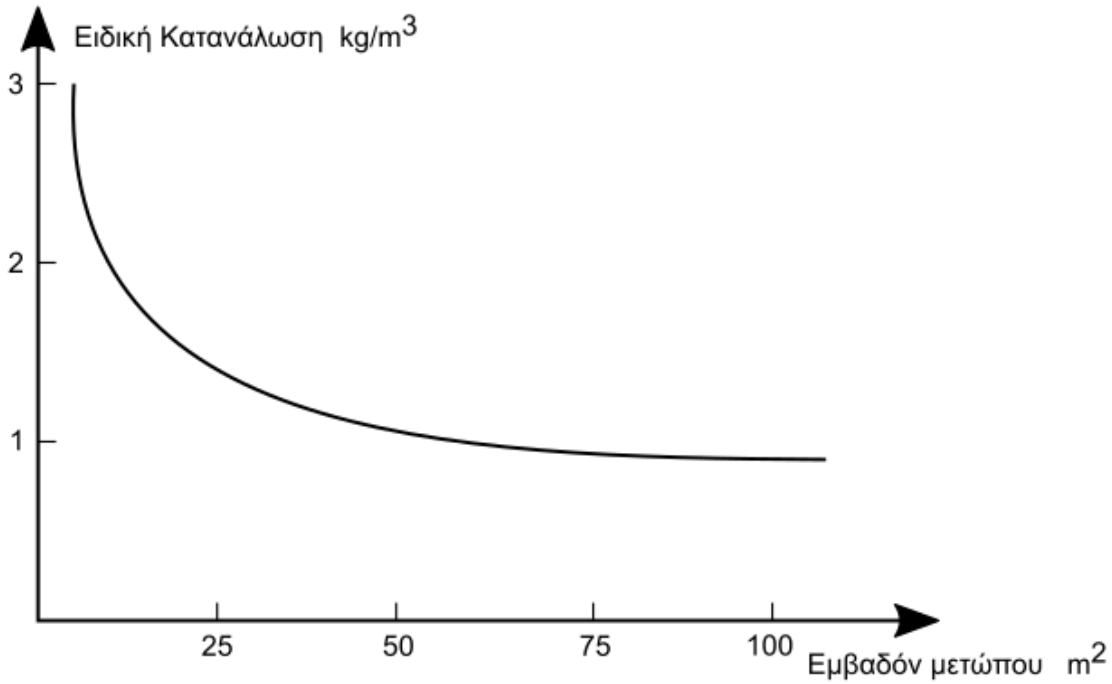
Σχήμα 2.2. Ενέργεια θραύσης πετρώματος σαν συνάρτηση της πίεσης για εκρηκτικό γαλάκτωμα αλουμινίου

Όπως φαίνεται και στο *Σχήμα 2.2.*, η θραύση του πετρώματος γίνεται σε περιοχή μικρής πίεσης και η ενέργεια θραύσης είναι πολύ μικρότερη από την συνολική εκλυόμενη ενέργεια.

2.1.2. Επιφάνεια ανατίναξης

Για την επιτυχή έκβαση μιας υπόγειας ανατίναξης κάθε διάτρημα πρέπει να βρίσκεται πλησίον μιας ελεύθερης επιφάνειας. Έτσι η κρουστική ενέργεια να μπορεί να δημιουργήσει τις αρχικές ακτινικές ρωγμές που διαδίδονται προς αυτήν και μετά τις ρωγμές αποφλοίωσης παράλληλα με την ελεύθερη επιφάνεια για το θρυμματισμό του πετρώματος μεταξύ διατρήματος και ελεύθερης επιφάνειας. Ο σχεδιασμός ανατινάξεων υπογείων μετώπων είναι μια δύσκολη διαδικασία εξόρυξης για το λόγο του ότι η μοναδική ελεύθερη επιφάνεια που παρέχεται είναι η επιφάνεια του μετώπου. Επομένως, θα πρέπει το αρχικό στάδιο της διάτρησης να περιλαμβάνει διατρήματα που όταν πυροδοτηθούν, θα δημιουργήσουν επιπλέον ελεύθερη επιφάνεια και χώρο για τα υπόλοιπα. Στο πλαίσιο αυτό σχεδιάζονται τα διατρήματα προεκσκαφής. Μπορούν να τοποθετηθούν σε διάταξη με μορφή σφήνας *(V-cut),* με μορφή ριπιδίου *(fan cut)* ή σε παράλληλη γεωμετρία με τη χρήση αρχικού βοηθητικού κενού διατρήματος *(parallel cut).* Όπως έχει προαναφερθεί, στη συγκεκριμένη μέθοδο (R. Holmberg, 1975) χρησιμοποιείται η παράλληλη διάταξη. Η διάταξη αυτή αφορά σε τοποθέτηση διατρημάτων σε μικρές αποστάσεις παράλληλα μεταξύ τους σχηματίζοντας τετράγωνα. Είναι τα τετράγωνα προεκσκαφής. Η διάταξη αυτή δεν εξαρτάται από το πλάτος του μετώπου και δεν απαιτεί τη χρήση κλίσης στην όρυξη διατρημάτων. Αρχικά ορύσσεται διάτρημα με διάμετρο μεγαλύτερη των υπολοίπων, το οποίο δεν θα γομωθεί με εκρηκτική ύλη, και θα αποτελέσει ελεύθερη επιφάνεια για την έναυση των μικρότερων διατρημάτων. Με την σταδιακή ανατίναξη των διατρημάτων αυτών, δημιουργείται επαρκής χώρος για την ομαλή έναυση των διατρημάτων διεύρυνσης.

Στη συνέχεια γίνεται επιλογή της κατάλληλης εκρηκτικής ύλης (Ε/Υ) που θα χρησιμοποιηθεί για τη γόμωση των διατρημάτων. Βασική προϋπόθεση στην επιλογή εκρηκτικής ύλης είναι η αποφυγή υψηλής συγκέντρωσης τοξικών αερίων κατά την ανατίναξη τόσο για περιβαλλοντικούς λόγους όσο και για λόγους ασφαλείας. Επίσης, η εκρηκτική ύλη που θα χρησιμοποιηθεί θα πρέπει να χαρακτηρίζεται από υψηλή ταχύτητα πυροδότησης για να αποτραπεί η εμφάνιση του φαινομένου του καναλιού (*channel effect)*. Το φαινόμενο αυτό εμφανίζεται στην περίπτωση που η εκρηκτική ύλη δεν εφάπτεται με τα τοιχώματα του διατρήματος. Αν η ταχύτητα πυροδότησης δεν είναι αρκετά υψηλή (μικρότερη των 3000 m/s), τα αέρια που απελευθερώνονται κατά την ανατίναξη οδηγούν στο μπροστινό τμήμα του διατρήματος συμπιεσμένο αέρα δημιουργώντας μια διαχωριστική επιφάνεια υψηλής θερμοκρασίας και πίεσης. Η δόνηση αυτή έχει σαν αποτέλεσμα η διαχωριστική επιφάνεια που δημιουργείται να συμπιέζει τη στήλη γόμωσης μπροστά από το σημείο πυροδότησης. Αυτό έχει σαν συνέπειες την εξάλειψη τυχόν σημείων έναυσης ή την αύξηση της πυκνότητας σε σημείο που σταματά η πυροδότηση ή τη χαμηλή απελευθερώμενη ενέργεια κατά την ανατίναξη που σημαίνει μειωμένη απόδοση εκρηκτικής ύλης. Τέλος, η εκρηκτική ύλη (Ε/Υ) που θα χρησιμοποιηθεί για τη γόμωση των ανυψωτικών διατρημάτων *(Lifters)* θα πρέπει να είναι ανθεκτική στο νερό. Στο παρακάτω *Σχήμα 2.3.* παρουσιάζεται η σχέση ειδικής κατανάλωσης εκρηκτικής ύλης (Ε/Υ) ως συνάρτηση του εμβαδού του μετώπου.



Σχήμα 2.3. Ειδική κατανάλωση (Ε/Υ) ως προς το εμβαδόν του υπογείου μετώπου

Όπως φαίνεται και στο *Σχήμα 2.3.*, η ειδική κατανάλωση εκρηκτικής ύλης μειώνεται όσο μεγαλύτερη είναι η επιφάνεια του υπογείου μετώπου.

2.1.3. Προχώρηση Μετώπου

Η διαδικασία διάτρησης του μετώπου προϋποθέτει τον υπολογισμό της απαιτούμενης προχώρησής. Η προχώρηση εξαρτάται πλήρως από το αρχικό κενό διάτρημα καθώς επίσης και από το βάθος των διατρημάτων. Για οικονομικούς λόγους, είναι πολύ σημαντικό να ισούται με το μέγιστο βάθος των διατρημάτων. Με το σκεπτικό αυτό, η προχώρηση του μετώπου θα πρέπει να καλύπτει το 95% του βάθους των διατρημάτων. Οποιαδήποτε άλλη τιμή μικρότερη, είναι οικονομικά απαγορευτική. Η εμπειρική σχέση που χρησιμοποιείται για τον υπολογισμό του βάθους διατρημάτων είναι:

$$H = 0.15 + 34.1 * \varphi - 39.4 * \varphi^2 \qquad \text{(m)} \quad [\text{Εξ. 2}]$$

Όπου φ η διάμετρος του αρχικού κενού διατρήματος σε μέτρα.

Η προχώρηση του μετώπου δίνεται από την εμπειρική σχέση:

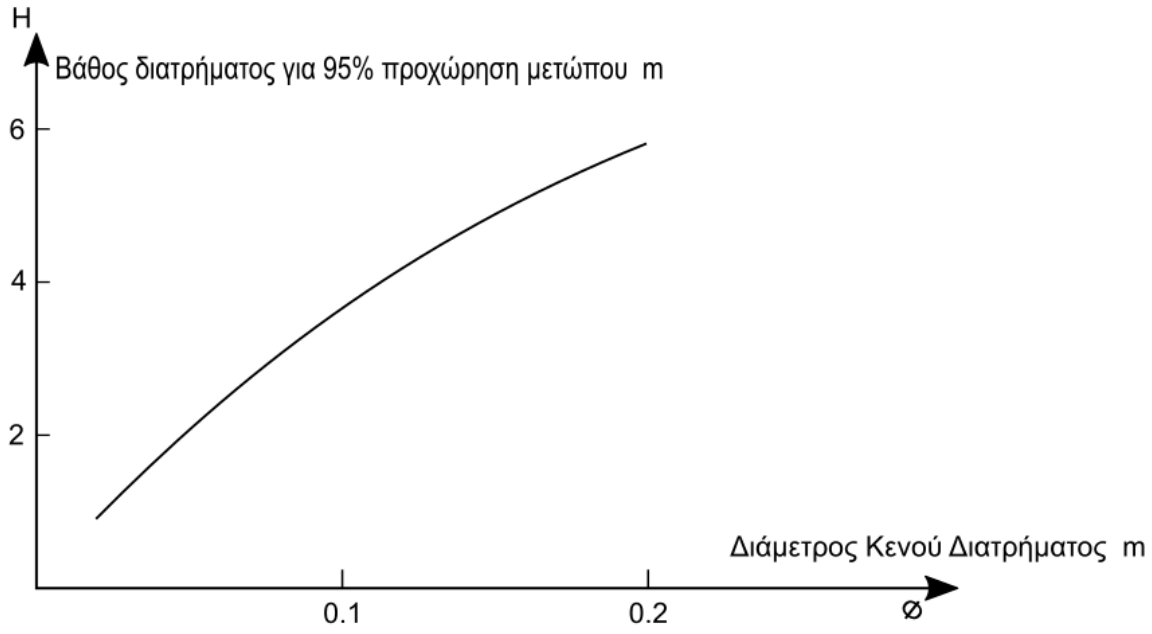
$$I = 0.95 * H \qquad \text{(m)} \quad [\text{Εξ. 3}]$$

Το αρχικό κενό διάτρημα μπορεί να διανοιχθεί εναλλακτικά σαν δύο διατρήματα που εφάπτονται ή αλληλεπικαλύπτονται μερικώς. Πολλές φορές ο διαθέσιμος μηχανοποιημένος εξοπλισμός δεν έχει τη δυνατότητα όρυξης διατρήματος μεγάλης διαμέτρου. Για το λόγο αυτό, επιλέγεται η όρυξη δύο κενών διατρημάτων με μικρότερη διάμετρο που εφάπτονται μεταξύ τους. Μάλιστα, στη συγκεκριμένη εργασία εφαρμόζεται η μέθοδος αυτή. Η εμπερική σχέση που αποτυπώνει τη διάμετρο του ενιαίου κενού διατρήματος σαν συνάρτηση των των δύο επιμέρους διατρημάτων είναι:

$$\varphi = d_0 * \sqrt{2} \qquad \text{(m)} \quad [\text{Εξ. 4}]$$

Όπου d_0 η διάμετρος των δύο διατρημάτων.

Στο παρακάτω *Σχήμα 2.4.* απεικονίζεται η σχέση μεταξύ βάθους διατρήματος για προχώρηση 95% ως προς τη διάμετρο αρχικού κενού διατρήματος.



Σχήμα 2.4. Σχέση βάθους διατρήματος για προχώρηση 95% σε συνάρτηση με τη διάμετρο του αρχικού κενού διατρήματος

2.1.4. Πρώτο Τετράγωνο Προεκσκαφής

Η βασική αρχή της μεθόδου για κάθε τμήμα της διάτρησης είναι ο υπολογισμός του φορτίου των διατρημάτων. Υπολογίζοντας αρχικά το μέγιστο φορτίο, σε συνδυασμό με την απόκλιση διάτρησης οδηγεί στο πρακτικό (πραγματικό) φορτίο των διατρημάτων. Έπειτα υπολογίζεται η πυκνότητα γόμωσης που απαιτείται για απαιτούμενη θραύση και συνεπώς ο τύπος του φυσιγγίου της εκρηκτικής ύλης (Ε/Υ) που θα χρησιμοποιηθεί. Τέλος, με τον υπολογισμό της απόστασης των διατρημάτων μεταξύ τους καθώς και της επιγόμωσής (όπου χρειάζεται), εκτιμάται ο αριθμός των διατρημάτων και των φυσιγγίων που χρησιμοποιηθούν.

Τα διατρήματα του πρώτου τετραγώνου προεκσκαφής χρησιμοποιούν σαν ελεύθερη επιφάνεια το αρχικό κενό διάτρημα. Για το λόγο αυτό το φορτίο τους, η απόστασή τους δηλαδή από το αρχικό διάτρημα, δεν θα πρέπει να ξεπερνάει τη διάμετρο του αρχικού διατρήματος κατά 1.7 φορές. Οποιαδήποτε τιμή μεγαλύτερη από αυτή, θα έχει σαν αποτέλεσμα πλαστική παραμόρφωση κατά την ανατίναξη του πετρώματος. Όλα αυτά αφορούν στο μέγιστο επιτρεπόμενο φορτίο. Όπως είναι αντιληπτό, το πρακτικό φορτίο που θα εφαρμοστεί θα είναι εμφανώς μειωμένο λόγω της απόκλισης διατρημάτων. Αν η απόκλιση διατρημάτων είναι της τάξης του 0.5% έως και 1%, τότε το πρακτικό φορτίο που θα προκύψει δίνεται από την εμπειρική σχέση:

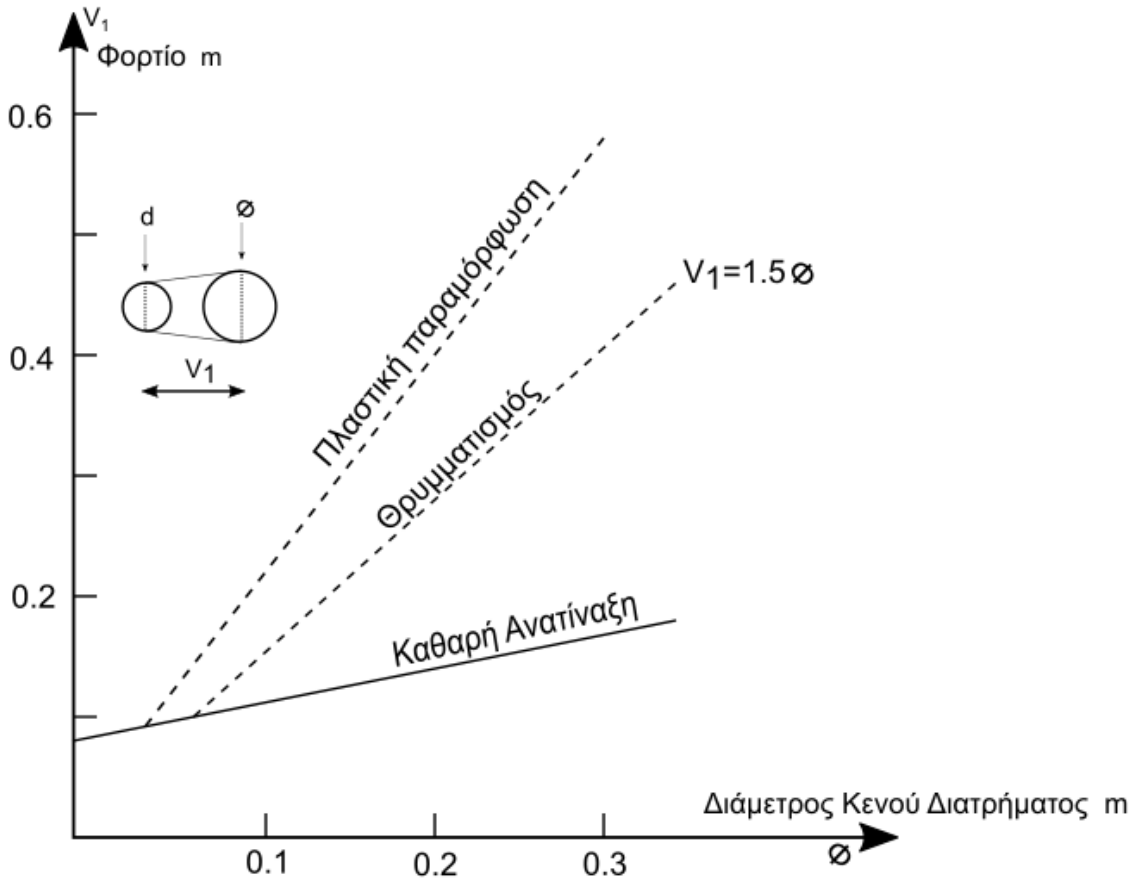
$$V_1 = 1.5 * \varphi \qquad \text{(m)} \quad [\text{Εξ. 5}]$$

Στην περίπτωση που η απόκλιση διατρημάτων ξεπερνά το 1%, τότε το πρακτικό φορτίο θα προκύψει από τη μείωση του μεγίστου φορτίου κατά την απόκλιση, όπως αναφέρει η εμπειρική σχέση:

$$V_1 = 1.7 * \varphi - (a * H + \beta) \qquad \text{(m)} \quad [\text{Εξ. 6}]$$

Όπου *(α*H+β)* η απόκλιση διατρημάτων F, *α* το σφάλμα κλίσης (παρέκκλιση) σε m/m και *β* το σφάλμα τοποθέτησης ή κολάρου σε m.

Στο *Σχήμα 2.5.* απεικονίζεται ακριβώς το αποτέλεσμα της ανατίναξης για διαφορετικές σχέσεις πρακτικού φορτίου και αρχικού κενού διατρήματος. Η θραύση του πετρώματος απαιτεί τη χρήση πρακτικού φορτίου 1.5 φορές μεγαλύτερο της διαμέτρου του αρχικού κενού διατρήματος. Αν η τιμή αυτή υπερβεί την τιμή του μέγιστου φορτίου, τότε θα υπάρξει πλαστική παραμόρφωση του πετρώματος.



Σχήμα 2.5. Αποτελέσματα ανατίναξης για διαφορετικές σχέσεις πρακτικού φορτίου-διαμέτρου αρχικού διατρήματος με απόκλιση μικρότερη του 1%

Με τον υπολογισμό του φορτίου για τα διατρήματα του πρώτου τετραγώνου προεκσκαφής, σειρά έχει η εκτίμηση της πυκνότητας εκρηκτικής γόμωσης που θα εφαρμοστεί σε αυτά. Οι Langefors και Kihlstrom (1963) καθιέρωσαν μια εμπειρική σχέση για τον υπολογισμό της πυκνότητας γόμωσης ως συνάρτηση του μεγίστου φορτίου και της διαμέτρου του αρχικού κενού διατρήματος. Η σχέση αυτή αφορά μόνο σε διατρήματα διαμέτρου 0.032 m και με την προϋπόθεση ότι η απόκλιση διατρημάτων είναι μικρότερη από 1% και κατά συνέπεια το πρακτικό φορτίο ισούται με 1.5 φορές τη διάμετρο του κενού διατρήματος. Η εμπειρική αυτή σχέση είναι:

$$I = 1.5 * (\frac{V}{\varphi})^{1.5} * (V - \frac{\varphi}{2}) \qquad \text{(kg/m)} \quad [\text{Εξ. 7}]$$

Για διατρήματα διαφορετικής διαμέτρου, ο υπολογισμός της πυκνότητας γόμωσης μπορεί να γίνει μέσω της ακόλουθης σχέσης:

$$I = \frac{d}{d_1} * I_1 \qquad \text{(kg/m)} \quad [\text{Εξ. 8}]$$

Όπου I_1 η πυκνότητα γόμωσης για διατρήματα διαμέτρου d_1=0.032 m όπως καθιερώθηκε από τους Langefors και Kihlstrom (1963).

Συνυπολογίζοντας τη σταθερά του εκάστοτε πετρώματος καθώς και τον τύπο της εκρηκτικής ύλης (Ε/Υ), η [Εξ. 7] αναδιατυπώνεται για τη χρήση διατρημάτων οποιασδήποτε διαμέτρου. Η εμπερική αυτή σχέση είναι:

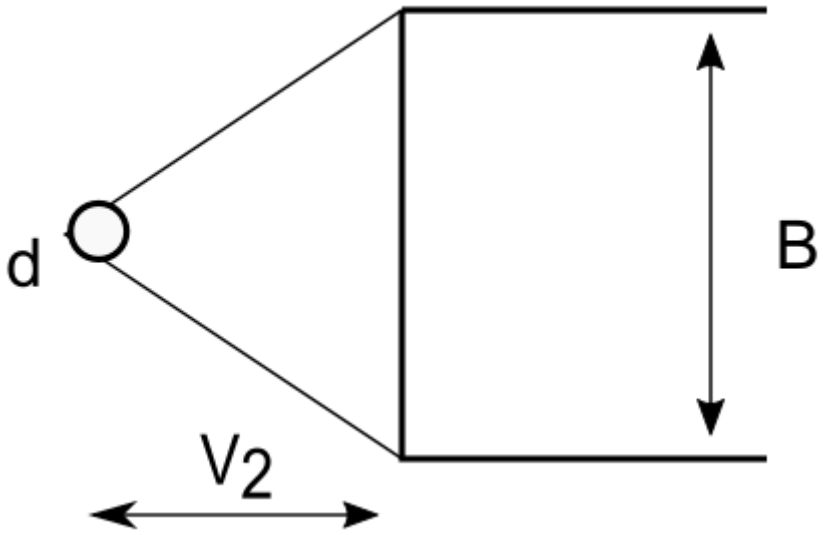
$$I = \frac{55 * d * (\frac{V}{\varphi})^{1.5} * \left(V - \frac{\varphi}{2}\right) * (\frac{c}{0.4})}{S_{ANFO}} \qquad \text{(kg/m)} \quad [\text{Εξ. 9}]$$

Όπου c η σταθερά πετρώματος και S_{ANFO} η σχετική ισχύς κατά βάρος της εκρηκτικής ύλης (Ε/Υ) ως προς πετραμμωνίτη.

Η σταθερά πετρώματος c, αφορά στην ποσότητα εκρηκτικής ύλης (Ε/Υ) που απαιτείται για τη θραύση ενός κυβικού μέτρου πετρώματος. Είναι διαφορετική για κάθε πέτρωμα και έχει εκτιμηθεί από δοκιμές σε πεδία υπαιθρίων εκμεταλλεύσεων. Παρ'όλα αυτά, οι τιμές της σταθεράς πετρώματος κυμαίνονται από 0.3 έως και 0.4 kg/m^3. Στη συγκεκριμένη μέθοδο σχεδιασμού υπογείων ανατινάξεων του Holmberg, η σταθερά πετρώματος λαμβάνεται ως 0.4 kg/m^3.

2.1.5. Δεύτερο Τετράγωνο Προεκσκαφής

Με την όρυξη των διατρημάτων του πρώτου τετραγώνου προεκσκαφής, δημιουργούνται νέες συνθήκες στον υπολογισμό των υπολοίπων τετραγώνων. Η ανατίναξη των διατρημάτων αυτών θα δημιουργήσει μία τετραγωνική-κάθετη ελεύθερη επιφάνεια, σαφώς μεγαλύτερη. Υπό αυτές τις συνθήκες, κύριο μέλημα για τον υπολογισμό των διατρημάτων του δευτέρου τετραγώνου είναι η εφαρμογή μεγαλύτερης γραμμικής πυκνότητας εκρηκτικής γόμωσης. Αυτό έχει σαν σκοπό την αύξηση του πρακτικού φορτίου τους και κατά συνέπεια τη χρήση λιγότερων διατρημάτων. Στο *Σχήμα 2.6.* απεικονίζεται η γεωμετρία σχεδιασμού ανατίναξης σε κάθετη ελεύθερη επιφάνεια.



Σχήμα 2.6. Γεωμετρία σχεδιασμού ανατίναξης σε κάθετη ελεύθερη επιφάνεια

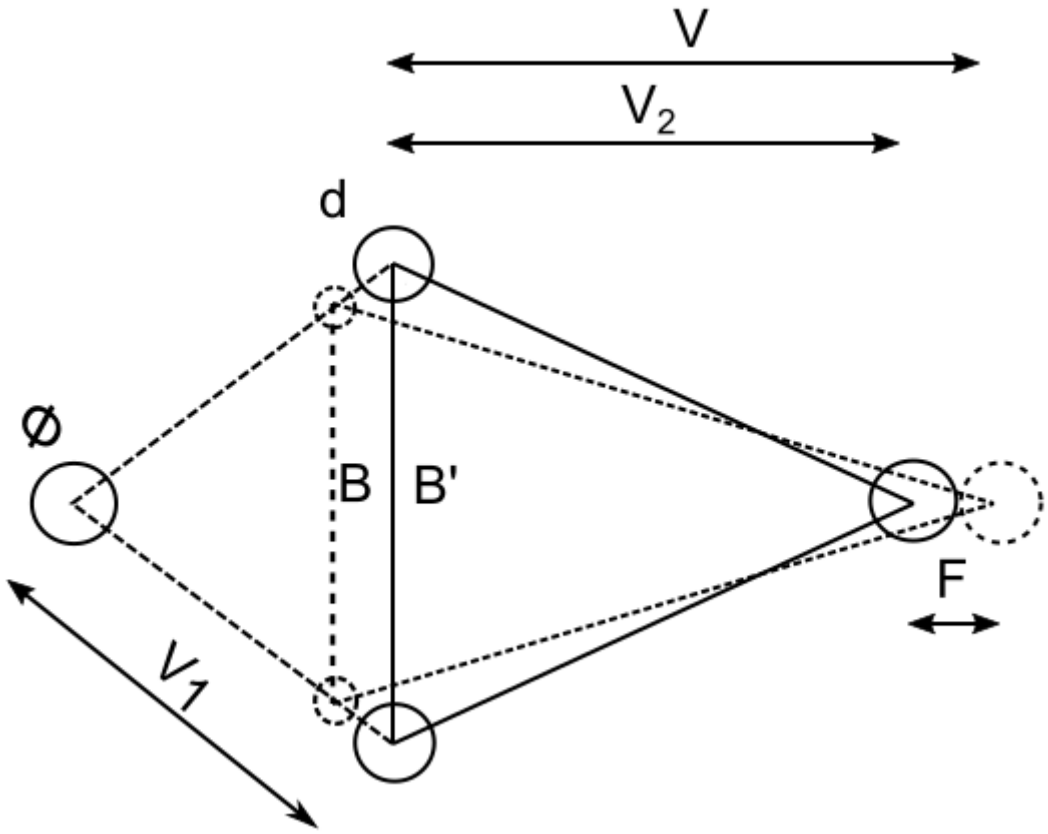
Στην περίπτωση που το πλάτος του ανοίγματος (ελεύθερης επιφάνειας) και το μέγιστο φορτίο είναι γνωστά, η γραμμική πυκνότητα γόμωσης υπολογίζεται από την παρακάτω εμπερική σχέση:

$$I = \frac{32.3 * d * c * V}{S_{ANFO} * [\sin\left(atn\left(\frac{B}{2 * V}\right)\right)]^{1.5}} \quad \text{(kg/m)} \quad [\text{Εξ. 10}]$$

Στην αντίθετη όμως περίπτωση όπου θεωρείται γνωστή η γραμμική πυκνότητα γόμωσης καθώς και το πλάτος του ανοίγματος, τότε το μέγιστο φορτίο υπολογίζεται από την παρακάτω σχέση:

$$V = 8.8 * 10^{-2} * \sqrt{\frac{B * I * S_{ANFO}}{d * c}} \quad \text{(m)} \quad [\text{Εξ. 11}]$$

Στο *Σχήμα 2.7.* απεικονίζεται η επίδραση της απόκλισης διατρημάτων στον υπολογισμό των πρακτικών φορτίων τους. Όπως φαίνεται και στο σχήμα η ελεύθερη επιφάνεια πλάτους *B* που σχηματίζεται από το πρώτο τετράγωνο, διαφέρει από την απόσταση των διατρημάτων του πρώτου τετραγώνου προεκσκαφής *B΄*. Αυτό οφείλεται στην απόκλιση των διατρημάτων *F* του προηγούμενου τετραγώνου.



Σχήμα 2.7. Επίδραση της απόκλισης διατρημάτων F στον υπολογισμό του φορτίου

Με αυτά τα δεδομένα, το πλάτος της ελεύθερης επιφάνειας που σχηματίζεται θα αναδιατυπωθεί συνυπολογίζοντας την απόκλιση αυτή:

$$B = \sqrt{2} * (V_1 - F) \quad \text{(m)} \quad \text{[Εξ. 12]}$$

Επομένως, η σχέση υπολογισμού του μέγιστου φορτίου θα διαμορφωθεί ως εξής:

$$V = 10.5 * 10^{-2} * \sqrt{\frac{(V_1 - F) * I * S_{ANFO}}{d * c}} \quad \text{(m)} \quad \text{[Εξ. 13]}$$

Σαφώς αυτή η τιμή θα πρέπει να μειωθεί κατά την απόκλιση των διατρημάτων για να εκτιμηθεί το πρακτικό φορτίο των διατρημάτων του δευτέρου τετραγώνου:

$$V_2 = V - F \quad \text{(m)} \quad \text{[Εξ. 14]}$$

Ο υπολογισμός του πρακτικού φορτίου περιέχει κάποιους περιορισμούς που πρέπει να ληφθούν υπ'όψην. Με την προϋπόθεση να μην υπάρξει πλαστική παραμόρφωση του πετρώματος, δίνεται η παρακάτω σχέση:

$$V_2 \leq 2 * B \quad \text{(m)} \quad \text{[Εξ. 15]}$$

Αν ο παραπάνω περιορισμός δεν μπορεί να πραγματοποιηθεί, τότε για την αποφυγή της πλαστικής παραμόρφωσης θα πρέπει να μειωθεί η γραμμική πυκνότητα γόμωσης κατά τις παρακάτω σχέσεις:

$$I = \frac{32.3 * d * c * 2 * B}{S_{ANFO} * [\sin(atn(1/4))]^{1.5}} \quad \text{(kg/m)} \quad \text{[Εξ. 16]}$$

Ή

$$I = \frac{540 * d * c * B}{S_{ANFO}} \quad \text{(kg/m)} \quad \text{[Εξ. 17]}$$

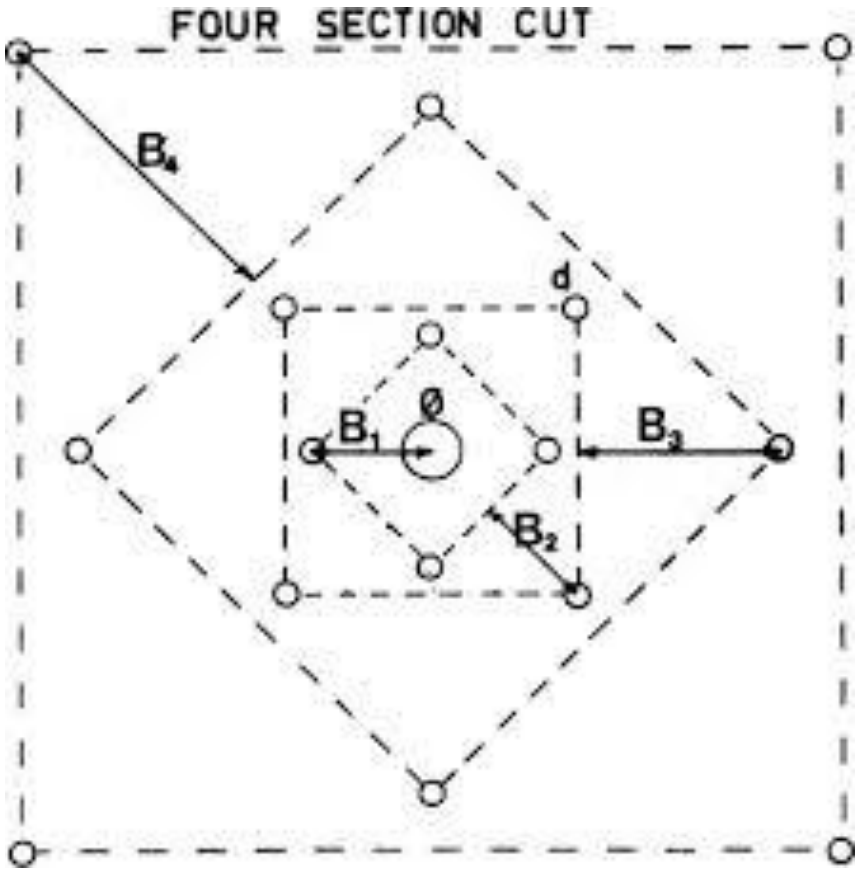
Επίσης, η γωνία του ανοίγματος θα πρέπει να είναι μικρότερη από 90^o για να διατηρείται το σχήμα του τετραγώνου. Έτσι, λοιπόν:

$$V_2 > 0.5 * B \quad \text{(m)} \quad \text{[Εξ. 18]}$$

Σε αυτό, όπως και σε όλα τα τετράγωνα προεκσκαφής είναι σημαντικό να τονιστεί ότι κάθε διάτρημα γομώνεται με εκρηκτικά μέχρι κάποιο σημείο. Το μήκος του εκρηκτικού που θα παραμείνει χωρίς εκρηκτική γόμωση, δίνεται από την παρακάτω σχέση:

$$h = 10 * d \quad \text{(m)} \quad \text{[Εξ. 19]}$$

Στη συγκεκριμένη εργασία ακολουθήθηκε η παρακάτω διαδικασία για τον υπολογισμό των διατρημάτων του δευτέρου τετραγώνου. Αρχικά υπολογίστηκε το πλάτος του ανοίγματος, με δεδομένη την απόκλιση διατρημάτων. Έπειτα, με γνώση του πλάτους ανοίγματος υπολογίστηκε το μέγιστο φορτίο για κάθε τύπο φυσιγγίου εκρηκτικής ύλης (E/Y). Επιλέγεται το φυσίγγιο εκείνο με το μέγιστο δυνατό φορτίο που ικανοποιεί παράλληλα τους περιορισμούς της μεθόδου Holmberg όπως αναγράφηκαν παραπάνω. Με αυτό τον τρόπο, αυξάνεται όσο το δυνατόν περισσότερο η γραμμική πυκνότητα γόμωσης των διατρημάτων που συνεπάγεται τη χρήση λιγότερων διατρημάτων. Οι παραπάνω υπολογισμοί αφορούν στους υπολογισμούς και των υπολοίπων τετραγώνων προεκσκαφής. Στο *Σχήμα 2.8.* απεικονίζονται τα τετράγωνα προεκσκαφής, όπου V τα πρακτικά φορτία των διατρημάτων (απόστασή τους από ελεύθερη επιφάνεια) και B οι αποστάσεις μεταξύ τους.



Σχήμα 2.8. Τετράγωνα προεκσκαφής (Holmberg, R., 1975, "Computer Calculations of Drilling Patterns for Surface and Underground Blastings", Design Methods in Rock Mechanics, C. Fairhurst and S. Crouch, eds., 16th Symposium on Rock Mechanics, University of Minnesota, Minneapolis.)

2.1.6. Διατρήματα Δαπέδου

Τα διατρήματα δαπέδου είναι αυτά που καθορίζουν την κλίση της νέας στοάς που θα δημιουργηθεί με το πέρας της ανατίναξης. Για τα διατρήματα αυτά, όπως έχει προαναφερθεί απαιτείται εκρηκτική ύλη (Ε/Υ) με ανοχή στο νερό. Αφου υπολογιστεί το μέγιστο δυνατό φορτίο τους, υπολογίζεται το πρακτικό φορτίο δηλαδή η απόστασή τους από την ελεύθερη επιφάνεια. Έπειτα καθορίζεται η μεταξύ τους απόσταση με τη διαφορά ότι τα δύο διατρήματα στα άκρα του δαπέδου έχουν μικρότερη απόσταση με τα επόμενα, λόγω της γωνιακής απόκλισης των περιμετρικών διατρημάτων.

Αρχικά, το μέγιστο φορτίο δίνεται από την εμπειρική σχέση:

$$V = 0.9 * \sqrt{\frac{I * S_{ANFO}}{\bar{c} * f * \frac{E}{V}}} \quad \text{(m)} \quad \text{[Εξ. 20]}$$

Όπου f ο συντελεστής περιορισμού που ισούται με 1.45, *E/V* η σχέση απόστασης διατρημάτων-φορτίου που ισούται με 1 και $\bar{c}$ η διορθωμένη σταθερά του πετρώματος. Αυτές οι τιμές ισχύουν για τα διατρήματα δαπέδου.

Η διορθωμένη σταθερά του πετρώματος δίνεται από τους ακόλουθους περιορισμούς:

$$\bar{c} = \begin{cases} c + 0.05 & V \geq 1.4\ m \\ c + \frac{0.07}{V} & V < 1.4\ m \end{cases} \quad \text{(m)} \quad \text{[Εξ. 21]}$$

Η απόσταση των διατρημάτων δαπέδου θα έπρεπε να ισούται με το μέγιστο φορτίο παρ'όλα αυτά εξαρτάται από το πλάτος του μετώπου. Ο αριθμός των διατρημάτων δαπέδου δίνεται από την παρακάτω σχέση:

$$N = int(\frac{tunnel\ width + 2 * H * sin\gamma}{V} + 2) \quad \text{[Εξ. 22]}$$

Η απόσταση των ενδιάμεσων διατρημάτων δαπέδου δίνεται από τη σχέση:

$$E_L = \frac{tunnel\ width + 2 * H * sin\gamma}{N - 1} \quad \text{(m)} \quad \text{[Εξ. 23]}$$

Ενώ η απόσταση των δύο γωνιακών διατρημάτων δαπέδου δίνεται από τη σχέση:

$$E_{L'} = E_L - H * sin\gamma \quad \text{(m)} \quad \text{[Εξ. 24]}$$

Και το πρακτικό φορτίο υπολογίζεται σα συνάρτηση του μεγίστου φορτίου, της γωνιακής απόκλισης περιμετρικών διατρημάτων και της απόκλισης διατρημάτων ως εξής:

$$V_L = V - H * sin\gamma - F \quad \text{(m)} \quad \text{[Εξ. 25]}$$

Στα διατρήματα δαπέδου εμφανίζεται και η επιγόμωση πυθμένα και στήλης. Αυτό σημαίνει ότι η στήλη του διατρήματος θα γομωθεί εκρηκτικά με διαφορετική γραμμική πυκνότητα γόμωσης απ'ότι ο πυθμένας του. Το μήκος γόμωσης πυθμένα που απαιτείται, υπολογίζεται από τη σχέση:

$$h_b = 1.25 * V_L \quad \text{(m)} \quad \text{[Εξ. 26]}$$

Η γραμμική πυκνότητα γόμωσης της στήλης είναι κατά 70% μικρότερη από αυτή που εφαρμόζεται στον πυθμένα και η οποία έχει ήδη υπολογιστεί, ενώ το μήκος γόμωσης στήλης καθορίζεται από τον τύπο:

$$h_c = H - h_b - 10 * d \quad \text{(m)} \quad \text{[Εξ. 27]}$$

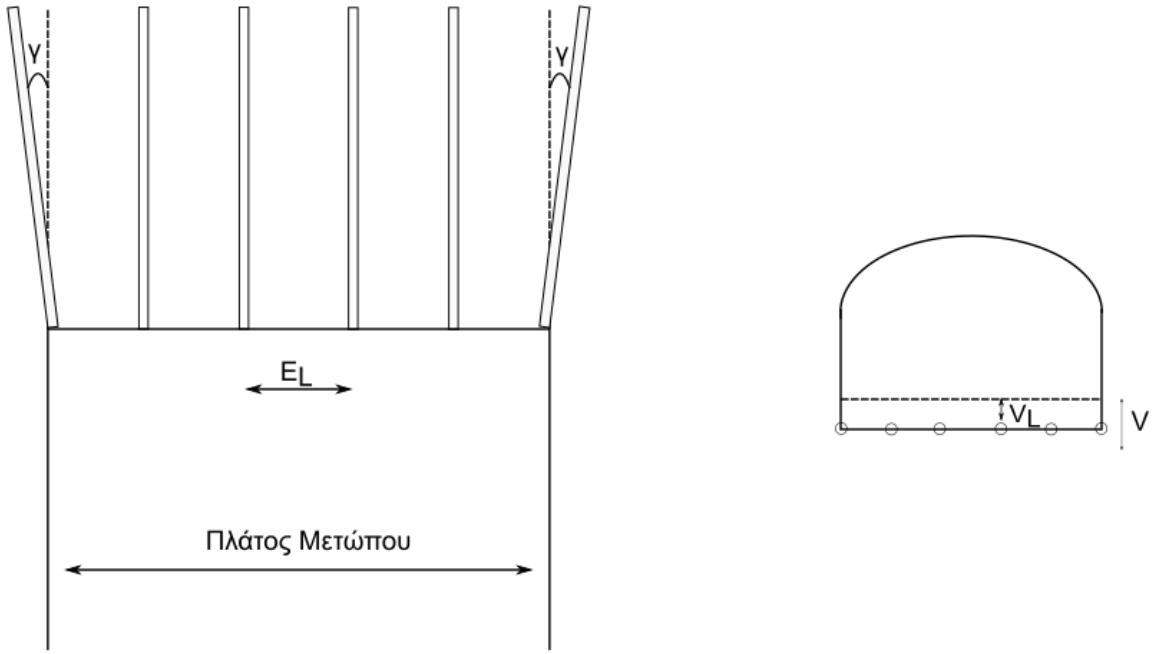
Γνωρίζοντας πλέον τα μήκη επιγόμωσης, τις γραμμικές πυκνότητες γόμωσης για πυθμένα και στήλη, καθώς και τα φυσίγγια εκρηκτικής ύλης (Ε/Υ) που θα χρησιμοποιηθούν εκατέρωθεν, υπολογίζεται εύκολα ο αριθμός των φυσιγγίων που θα χρησιμοποιηθούν σε κάθε διάτρημα.

Σημαντικός περιορισμός στη χρήση της [Εξ. 20] για τον υπολογισμό του μέγιστου φορτίου των διατρημάτων δαπέδου είναι ο εξής:

$$V \leq 0.6 * H \quad \text{(m)} \quad \text{[Εξ. 28]}$$

Αν δεν μπορεί να πραγματοποιηθεί ο παραπάνω περιορισμός, τότε το μέγιστο φορτίο μπορεί να μειωθεί με ενδεχόμενη μείωση της γραμμικής πυκνότητας γόμωσης.

Στο *Σχήμα 2.9.* απεικονίζεται η γεωμετρία σχεδιασμού των διατρημάτων δαπέδου όπου E_L η μεταξύ τους απόσταση και *γ* η γωνιακή απόκλιση των περιμετρικών διατρημάτων.



Σχήμα 2.9. Γεωμετρία σχεδιασμού διατρημάτων δαπέδου σε υπόγειο μέτωπο

Ο συντελεστής περιορισμού που αναφέρθηκε στη σχέση *[Εξ. 20]* διαφέρει ανάλογα την εκάστοτε περίσταση. Αν τα διατρήματα είναι κεκλιμένα αυτό διευκολύνει τη χαλάρωση του πετρώματος στις άκρες. Συνυπολογίζοντας αυτό, θα χρησιμοποιηθεί ένας χαμηλότερος συντελεστής περιορισμού $(f < 1)$ για κεκλιμένα διατρήματα και θα έχει σαν αποτέλεσμα μεγαλύτερο φορτίο. Επομένως, οι τιμές των συντελεστών περιορισμού θα ποικίλουν και θα μεταβάλλονται αναλόγως τις αληλλεπιδράσεις των διατρημάτων και της βαρύτητας.

2.1.7. Διατρήματα Διεύρυνσης (stoping holes)

Τα διατρήματα διεύρυνσης αφορούν στα διατρήματα που ακολουθούν το τετράγωνο της προεκσκαφής. Στη συγκεκριμένη εργασία τα διατρήματα διεύρυνσης τοποθετούνται δεξιά της προεκσκαφής και πάνω από αυτήν. Είναι τα διατρήματα που θα πυροδοτηθούν αμέσως μετά τα διατρήματα της προεκσκαφής και επομένως θα χρησιμοποιήσουν σαν ελεύθερη επιφάνεια, το άνοιγμα που δημιουργήθηκε. Ο υπολογισμός τους γίνεται με βάση τη μέθοδο που ακολουθήθηκε στα διατρήματα δαπέδου. Τα διατρήματα διεύρυνσης χωρίζονται σε δύο κατηγορίες, αυτά που θα δράσουν οριζόντια με κλίση προς τα πάνω (δεξιά της προεκσκαφής) και αυτά που θα δράσουν κάθετα με κλίση προς τα κάτω (πάνω από την προεκσκαφή). Για τον υπολογισμό τους χρησιμοποιείται η σχέση *[Εξ. 20]* με κάποια διαφορά στις παραμέτρους. Για τα κάθετα διατρήματα διεύρυνσης χρησιμοποιείται συντελεστής περιορισμού *f=1.2* και

σχέση απόστασης διατρημάτων-φορτίου *E/V=1.25*, ενώ για τα οριζόντια διατρήματα διεύρυνσης ο συντελεστής περιορισμού παίρνει την τιμή *f=1.45* και η σχέση απόστασης-φορτίου *E/V=1.25*. Σε αυτό το τμήμα της διάτρησης υπάρχει επίσης επιγόμωση κατά την οποία η γραμμική πυκνότητα γόμωσης στήλης είναι κατά 50% μικρότερη από τη γραμμική πυκνότητα γόμωσης πυθμένα.

Ο αριθμός των διατρημάτων διεύρυνσης δεν υπόλογίζεται από κάποια εμπειρική σχέση. Βάσει του πρακτικού φορτίου τους και του διαθέσιμου χώρου που απομένει τόσο από την προεκσκαφή όσο και από τα περιμετρικά διατρήματα, εκτιμάται ο αριθμός των διατρημάτων διεύρυνσης καθώς και η απόσταση μεταξύ τους. Για τα διατρήματα διεύρυνσης της περιοχής πάνω από την προεκσκαφή κριτήριο αποτελεί ο χώρος που απομένει μεταξύ διατρημάτων οροφής και προεκσκαφής, ενώ τα διατρήματα διεύρυνσης δεξιά από την προεκσκαφή κριτήριο είναι ο χώρος μεταξύ προεκσκαφής και διατρημάτων τοίχου.

2.1.8. Διατρήματα Οροφής-Τοιχωμάτων (παρειών της στοάς)

Τα διατρήματα της οροφής και του τοίχου ονομάζονται και περιμετρικά διατρήματα και αφορούν στο σύνορο του μετώπου. Όσο αναφορά τα διατρήματα τοίχου, ο υπολογισμός τους είναι ίδιος με των διατρημάτων δαπέδου. Χρησιμοποιείται το μέγιστο δυνατό φορτίο βάσει των περιορισμών, για τη χρήση όσο το δυνατόν λιγότερων διατρημάτων. Στα διατρήματα οροφής εφαρμόζεται τεχνική ελεγχόμενης ανατίναξης *(Smooth-blasting)*. Η τεχνική αυτή αφορά σε περιμετρικά διατρήματα ή αλλιώς διατρήματα συνόρου και αποσκοπεί στην διαφύλαξη των ορίων της στοάς και στην ελαχιστοποίηση της ζώνης κατακερματισμού του πετρώματος. Σαν βασική ιδεά της μεθόδου αυτής είναι η ασθενέστερη εκρηκτική γόμωση των περιμετρικών διατρημάτων σε σχέση με τα υπόλοιπα. Η παρακάτω εμπειρική σχέση αφορά στην απόσταση των διατρημάτων οροφής σε τεχνική ελεγχόμενης ανατίναξης:

$$E = k * d \qquad \text{(m)} \quad [\text{Εξ. } 29]$$

Όπου d η διάμετρος των διατρημάτων, ενώ η σταθερά k παίρνει τυχαία τιμή στο εύρος [15,16]. Στην τεχνική ελεγχόμενης ανατίναξης, το διάτρημα γομώνεται με εκρηκτική ύλη (Ε/Υ) σε όλο το μήκος του για την αποφυγή καταστροφής. Επομένως δεν υπάρχει επιγόμωση. Η ελάχιστη επιτρεπόμενη γραμμική πυκνότητα γόμωσης είναι συνάρτηση της διαμέτρου των διατρημάτων. Για διαμέτρους διατρημάτων έως και 0.15 m ,η ελάχιστη επιτρεπόμενη γραμμική πυκνότητα γόμωσης δίνεται από τη σχέση:

$$I = 90 * d^2 \qquad \text{(kg/m)} \quad [\text{Εξ. } 30]$$

Στην περίπτωση που δεν γίνει χρήση τεχνικής ελεγχόμενης ανατίναξης, ο υπολογισμός του φορτίου διατρημάτων οροφής υπολογίζεται από τη σχέση των διατρημάτων δαπέδου [Εξ. 20], με συντελεστή περιορισμού f=1.2, σχέση απόστασης-φορτίου E/V=1.25 και γραμμική πυκνότητα γόμωσης στήλης κατά 50% μικρότερη από την αντίστοιχη του πυθμένα.

2.2. Παράμετροι προγράμματος

2.2.1. Γεωμετρία στοάς

Για το σχεδιασμό ανατίναξης σε υπόγειο μέτωπο χρησιμοποιείται ένα πρότυπο υπογείου μετώπου με διαστάσεις 4.5 m x 4 m όπου 4 μέτρα το ύψος της στοάς *(abutment height)* και 4.5 μέτρα το πλάτος της (*width*). Αρχικό κενό διάτρημα σαν ελέθευρη επιφάνεια τοποθετείται διάτρημα διαμέτρου 0.127 μέτρα (δύο διατρήματα των 0,089 m) και διάμετρος κανονικού διατρήματος 0.045 μέτρα για τα υπόλοιπα *(hole diameter).*

Η οροφή αποτελεί τμήμα κυκλικού δίσκου με ύψος αψίδας 0.5 m *(height of arch).* Σφάλμα κλίσης (παρέκκλιση) ισούται με 10 mm/m *(angular deviation, a),* ενώ το σφάλμα τοποθέτησης ή κολάρου λαμβάνει την τιμή 20 mm *(collar deviation, b).* Τέλος, η γωνιακή απόκλιση για τα περιμετρικά διατρήματα ανέρχεται στις 3 μοίρες ή αλλίως 0.05 rad *(lookout for contour holes, g).*

2.2.2. Εκρηκτικό γαλάκτωμα EM-EX

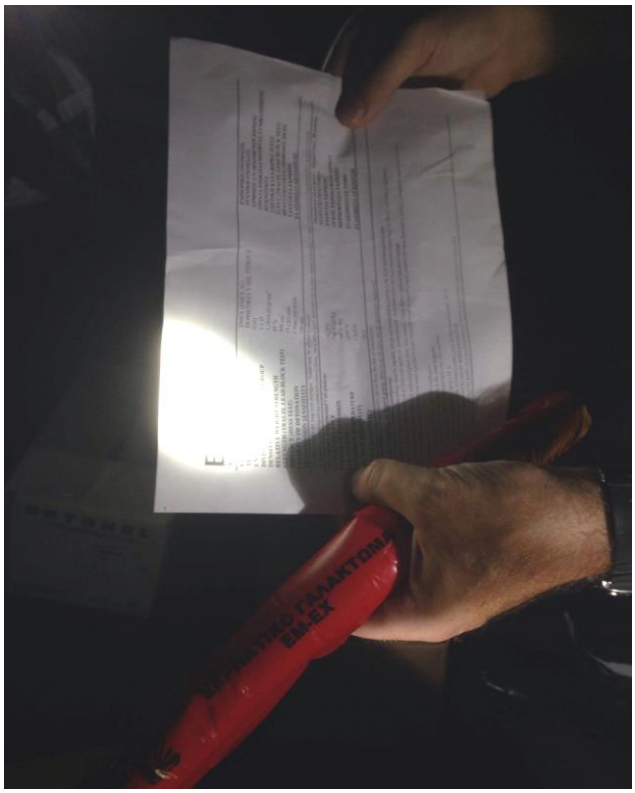
Το εκρηκτικό που θα χρησιμοποιηθεί για την ανατίναξη του συγκεκριμένου μετώπου είναι τύπου γαλακτώματος. Επιλέγεται η συγκεκριμένη εκρηκτική ύλη πρωτίστως για την ιδιότητά της να μην απελευθερώνει τοξικά αέρια κατά την ανατίναξη, που αποτελεί βασική προϋπόθεση στο σχεδιασμό υπογείων ανατινάξεων τόσο για περιβαλλοντικούς όσο και για λόγους ασφαλείας. Σημαντική ιδιότητα του εκρηκτικού γαλακτώματος EM-EX είναι η αντοχή στο νερό κάτι που απαιτείται στο σχεδιασμό των διατρημάτων δαπέδου. Επίσης, τα συστατικά του συγκεκριμένου γαλακτώματος δεν προκαλούν πονοκέφαλο ή άλλες διαταραχές σε αντίθεση με τα νιτρογλυκερινούχα εκρηκτικά. Τέλος, τα εκρηκτικά γαλακτώματα έχουν υψηλότερες ταχύτητες έκρηξης από τις υπόλοιπες εκρηκτικές ύλες (https://www.extraco.gr/index.php).

Το εκρηκτικό γαλάκτωμα (*Εικόνα 2.1.)* χρησιμοποιείται σε φυσίγγια καθορισμένων διαστάσεων. Σκοπός του βέλτιστου σχεδιασμού ανατίναξης, πέραν του ποσού και της θέσης των διατρημάτων, είναι και η επιλογή και ποσότητα του καταλληλότερου τύπου φυσιγγίου βάσει της διαμέτρου, του βάρους του και της γραμμικής πυκνότητας γόμωσής του. Είναι σημαντικό να τονιστεί ότι οι διαστάσεις των φυσιγγίων που θα χρησιμοποιηθούν θα πρέπει να συμβαδίζουν με τη γεωμετρία της στοάς. Σε διάτρημα διαμέτρου 45 mm όπως στη συγκεκριμένη εργασία, η μέγιστη επιτρεπόμενη διάμετρος φυσιγγίου είναι στα 38 mm.

Οι διαθέσιμοι τύποι φυσιγγίων που θα χρησιμοποιηθούν για το σχεδιασμό της διάτρησης απεικονίζονται στον *Πίνακα 2.2.*

Πίνακας 2.2. Διαθέσιμοι τύποι φυσιγγίων εκρηκτικού γαλακτώματος EM-EX

28mm x 255mm
32mm x 420mm
38mm x 385mm



Εικόνα 2.1. Εκρηκτικό γαλάκτωμα σε φυσίγγιο και έντυπο αναφοράς χαρακτηριστικών σε υπόγειο μεταλλείο στην περιοχή της Ολυμπιάδας Χαλκιδικής.

ΚΕΦΑΛΑΙΟ 3 : ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ Η/Υ

Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η διαδικασία υλοποίησης του κώδικα.

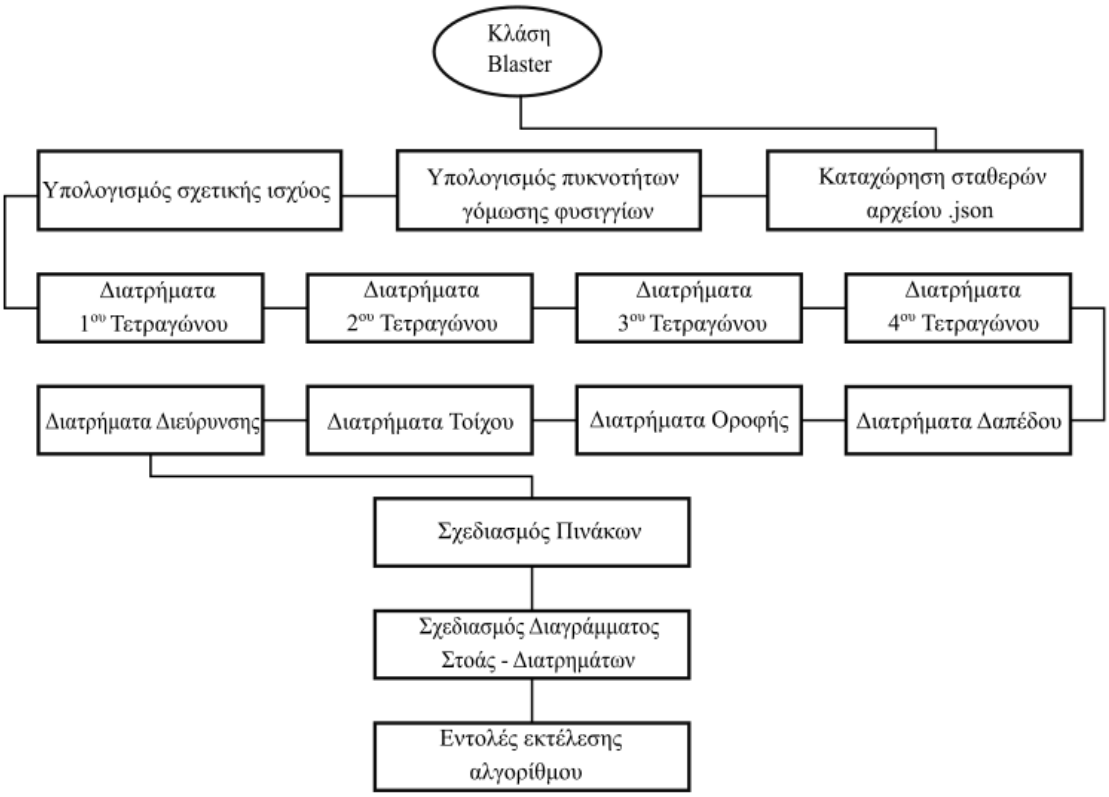
3.1. Γλώσσα προγραμματισμού Python

Η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που δημιουργήθηκε από τον Guido Van Rossum το 1991. Είναι η γλώσσα που σύμφωνα με τον δημιουργό της έχει φτιαχτεί για να είναι ευχάριστη και να παρέχει μηχανισμούς που διευκολύνουν τον προγραμματισμό, αλλά ταυτόχρονα να είναι και αναγνώσιμη. Πρόκειται για μια γλώσσα ανοικτού κώδικα, γενικής χρήσης και ανήκει στις υψηλού επιπέδου γλώσσες

προγραμματισμού. Στα πλεονεκτήματα της χρήσης Python συγκαταλέγεται το απλό συντακτικό που απαιτεί καθώς και η εξαιρετική αναγνωσιμότητα του κώδικα. Είναι κατάλληλη για αρχάριους και έμπειρους προγραμματιστές ταυτόχρονα. Εκτός των άλλων, σημαντικό πλεονέκτημα της είναι τα αρκετά πακέτα υποστήριξης της γλώσσας ή αλλιώς βιβλιοθήκες κάτι που απαιτείται για τη συγκεκριμένη εργασία λόγω των αρκετών υπολογισμών, εξισώσεων καθώς και τη δημιουργία διαγραμμάτων και πινάκων . Η Python ενδείκνυται ως γλώσσα εισαγωγική στον προγραμματισμό και στην επιστήμη των υπολογιστών γενικότερα, έχοντας σαν δεδομένη την απλή σύνταξη που απαιτεί.

3.2. Σχεδιασμός προγράμματος

Το συγκεκριμένο πρόγραμμα απαιτεί τη χρήση δύο αρχείων. Το πρώτο αρχείο αποτελεί ένα λεξικό σε μορφή .json *(config.json)* στο οποίο καταγράφονται οι παράμετροι του προβλήματος τις οποίες καθορίζει ο προγραμματιστής. Τέτοιες τιμές αφορούν τα χαρακτηριστικά της ΕΥ που χρησιμοποιείται, τη διάμετρο των διατρημάτων τις γεωμετρικές παραμέτρους της στοάς. Όπως είναι κατανοητό, αυτές οι τιμές αποτελούν «σταθερές» για το σχεδιασμό της διάτρησης του μετώπου. Επομένως, στο αρχικό στάδιο του προγράμματος γίνεται καταχώρηση των τιμών αυτών στο αρχείο config.json .



Σχήμα 3.1. Διάγραμμα ροής αλγορίθμου

Το δεύτερο αρχείο αφορά στον αναλυτή-εκτελεστή *(Code_Blasting.py)* όπου και δομείται ο αλγόριθμος. Είναι της μορφής .py, δηλαδή αρχείο σε μορφή Python. Σε αρχικό στάδιο καταχωρούνται τα πακέτα υποστήριξης τα οποία απαιτώνται για το πρόγραμμα ή αλλιώς οι βιβλιοθήκες της Python όπως για παράδειγμα matplotlib, numpy κ.α. Στο παραπάνω *Σχήμα 3.1.* παρουσιάζεται το διάγραμμα ροής του αλγορίθμου του αρχείου Code_Blasting.py.

Ακολούθως, δημιουργείται το αντικείμενο/κλάση **Blaster()**. Έπειτα, καταχωρούνται –εντός της παραμέτρου self.- οι τιμές εκείνες οι οποίες υπάρχουν στο λεξικό *(config.json)* και αφορούν τις σταθερές καθ'όλη την πορεία του προγράμματος. Για παράδειγμα, το πλάτος του υπογείου μετώπου δηλώνεται σαν self.width . Μέσα στην κλάση *Blaster()* έχουν δημιουργηθεί κάποιες συναρτήσεις (*functions*). Οι υπολογισμοί γίνονται ομαδοποιημένα σε κάθε τομέα του σχεδιασμού της ανατίναξης και για κάθε τομέα χρησιμοποιείται μια διαφορετική συναρτηση (*def()*). Για παράδειγμα, για τον σχεδιασμό και τους υπολογισμούς των διατρημάτων δαπέδου (*lifters*), η αντίστοιχη συνάρτηση που τους υλοποιεί είναι η **def lifters(self)**. Το self καταχωρείται σε κάθε συνάρτηση για να απομνημονεύει τις αρχικές τιμές που δηλώθηκαν με αυτό τον τρόπο και να τις χρησιμοποιεί αυτές καθ' αυτές όταν καλούνται.

Οι τιμές που προκύπτουν από κάθε τμήμα του σχεδιασμού και είναι απαραίτητες για τον τελικό πίνακα δεδομένων και για την υλοποίηση του τελικού διαγράμματος σχεδιασμού, αποθηκεύονται σε λίστες. Σκοπός της λίστας είναι η αποθήκευση στοιχείων κατά τη σειρά που προσθέτονται σε αυτήν και οποιαδήποτε στιγμή εντός προγράμματος μπορεί να γίνει κλήση ολόκληρης της λίστας ή ενός στοιχείου της. Για το λόγο αυτό ήταν αναγκαία η δημιουργία πολλών λιστών, ώστε να αποθηκεύονται τα απαραίτητα στοιχεία και να προσθέτονται επιπλέον στοιχεία από την κάθε συνάρτηση του προγράμματος. Για παράδειγμα, η λίστα με όνομα self.practicalb[ ] αφορά όλα τα πρακτικά φορτία *(practical burden)* που καταχωρήθηκαν από κάθε συνάρτηση του προγράμματος, ενώ το self.practicalb[0] αφορά στο πρώτο πρακτικό φορτίο που καταχωρήθηκε.

Στο τελικό στάδιο του προγράμματος, δημιουργούνται δύο τελικοί πίνακες δεδομένων και ένα διάγραμμα που απεικονίζει τη γεωμετρία του υπογείου μετώπου και τα διατρήματα που απαιτώνται για το βέλτιστο σχεδιασμό της ανατίναξης όπως υπολογίστηκε από το πρόγραμμα. Τέλος, αφού η μέθοδος σχεδιασμού αναλύθηκε πλήρως σε κώδικα ζητείται από το πρόγραμμα να «διαβάσει» και να καταχωρήσει τις αρχικές τιμές *(self.)* από το αρχείο config.json και να «τρέξει» την κλάση και τις ακόλουθες συναρτήσεις της ως αποτέλεσμα του κώδικα.

Σε αυτό το σημείο να επισημανθεί ότι ο κώδικας γράφτηκε σε κειμενογράφο VSCode.

Αρχείο .json

Το πρόγραμμα χωρίζεται σε δύο αρχεία όπως προαναφέρθηκε. Το πρώτο αρχείο που δημιουργείται είναι της μορφής .json και ουσιαστικά παριστάνει ένα λεξικό. Μέσα σε αυτό υπάρχουν όλες οι τιμές που σχετίζονται με τα χαρακτηριστικά των φυσιγγίων της εκρηκτικής ύλης καθώς επίσης και τα γεωμετρικά στοιχεία της στοάς. Πριν τη χρήση του κώδικα, είναι

απαραίτητο ο χειριστής να επεξεργαστεί το παραπάνω λεξικό και να καθορίσει τις τιμές ανάλογα με τις προδιαγραφές του. Το αρχείο αυτό είναι συμβατό με τη γλώσσα Python, επομένως οι υπολογισμοί και τα αποτελέσματα του κώδικα πηγάζουν από τα στοιχεία που ορίζει ο κάθε χειριστής στο αρχείο config.json .

Αρχικά, δίνεται το όνομα της εκρηκτικής ύλης που συμπληρώνεται στο κενό μετά την έκφραση “name”: . Έπειτα, σειρά έχουν τα χαρακτηριστικά των φυσιγγίων (“cartridges”:) που είναι διαθέσιμα και τα οποία θα επιλεχθούν για το βέλτιστο σχεδιασμό της διάτρησης. Με τις εκφράσεις “name”: , “diameter”: , “length”: , ο χειριστής καλείται να γράψει το όνομα, τη διάμετρο και το μήκος του κάθε τύπου φυσιγγίου.

Στο τέλος απαιτείται η εκχώρηση των γεωμετρικών παραμέτρων της στοάς καθώς και κάποια γενικά χαρακτηριστικά της εκρηκτικής ύλης. Στο αρχείο παρουσιάζονται ως:

"hdiam": Διάμετρος διατρημάτων *(m)*

"emptyf": Διάμετρος αρχικού κενού διατρήματος *(m)*

"width": Πλάτος μετώπου *(m)*

"abutmentH": Ύψος μετώπου *(m)*

"archHeight": Ύψος κυκλικού τμήματος οροφής *(m)*

"g": Γωνιακή απόκλιση για τα περιμετρικά διατρήματα *(rad)*

"a": Σφάλμα κλίσης *(m)*

"b": Σφάλμα τοποθέτησης *(m)*

"Q": Εκλυόμενη ενέργεια *(MJ/kg)* αερίων εκρηκτικής ύλης κατά την ανατίναξη

"V": Όγκος εκλυόμενων *(m^3/kg)* αερίων εκρηκτικής ύλης κατά την ανατίναξη

"Density": Πυκνότητα εκρηκτικής ύλης *(kg/m^3)*

"c": Σταθερά πετρώματος

Κατά την περιγραφή του κώδικα, χρησιμοποιούνται εξισώσεις και τύποι από τη μέθοδο του Holmberg (1975). Οι εξισώσεις αυτές αναφέρονται στο ΚΕΦΑΛΑΙΟ 2 (2.1.) αναλυτικά και πλέον θα γίνεται επισήμανσή τους.

Αρχή κώδικα

Ο κώδικας ξεκινάει με την καταχώρηση της κλάσης μέσα η οποία περιλαμβάνει τις συναρτήσεις για τους διάφορους υπολογισμούς της μεθόδου του Holmberg. Στο αρχείο .py λοιπόν, καταχωρείται η κλάση του προγράμματος με τίτλο **Blaster()**. Έπειτα, σειρά έχει η

πρώτη συνάρτηση του κώδικα. Η πρώτη συνάρτηση είναι η **def __init__(self)**. Σε αυτήν, γίνεται ορισμός των τιμών του αρχείου config.json καθώς και των λιστών που πρέπει να αποθηκευτούν για να χρησιμοποιηθούν στη συνέχεια. Όλες οι τιμές της γεωμετρίας της στοάς και των χαρακτηριστικών της εκρηκτικής ύλης που καθορίζει ο χειριστής, δηλώνονται στην αρχική αυτή συνάρτηση και παίρνουν την τιμή 0. Αυτό γίνεται για να αποθηκευτούν στη μνήμη του υπολογιστή και αργότερα να αποκτήσουν την τιμή που όρισε ο χρήστης του προγράμματος. Επίσης, όλες οι λίστες κατά τη διάρκεια του προγράμματος που απαιτώνται να αποθηκευτούν για να μπορούν να χρησιμοποιηθούν ακέραια σε κάθε κομμάτι του κώδικα, έχουν δηλωθεί στην αρχική αυτή συνάρτηση σαν κενές λίστες. Για παράδειγμα, self.hdiam = 0 αναφέρεται στη διάμετρο των διατρημάτων που παίρνει τον όρο self. για να αποθηκευτεί στη μνήμη του υπολογιστή και να λάβει την τιμή της διαμέτρου που όρισε ο χειριστής στο λεξικό αρχείο config.json. Έπειτα, κάθε φορά που θα απαιτείται η διάμετρος των διατρημάτων θα γίνεται αναφορά στον όρο self.hdiam. Επίσης, η λίστα self.practicalb = [ ] αφορά στη λίστα όλων των πρακτικών φορτίων των διατρημάτων που υπολογίζονται κατά την εργασία και αποθηκεύονται σε αυτήν για να είναι άμεσα προσβάσιμα όταν η λίστα κληθεί από τον κώδικα.

Παρακάτω αναγράφονται αναλυτικά οι σταθερές τιμές και οι λίστες που χρησιμοποιούνται στον κώδικα:

self.Sanfo=0 : γραμμική πυκνότητα εκρηκτικής ύλης ως προς ANFO

self.hdepth=0 : Βάθος διατρήματος

self.advance=0 : Προχώρηση μετώπου

self.massph=[ ] : Λίστα συνολικής γόμωσης *(kg)* ανά διάτρημα

self.total=[ ] : Συνολική μάζα εκρηκτικών *(kg)*

self.charge=[ ] : Λίστα γραμμικής πυκνότητας γόμωσης φυσιγγίων

self.cartridges = [ ] : Λίστα φυσιγγίων εκρηκτικής ύλης

self.B=[ ] : Λίστα πλευρών τετραγώνων του Cut

self.practicalb=[ ] : Λίστα πρακτικών φορτίων

self.diam=[ ] : Λίστα διαμέτρων φυσιγγίων που χρησιμοποιείται σαν τίτλος διατρημάτων στον τελικό πίνακα

self.holes=[ ] : Λίστα αριθμού διατρημάτων για κάθε τμήμα

self.number_first=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα του πρώτου τετραγώνου προεκσκαφής

self.number_second=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα του δεύτερου τετραγώνου προεκσκαφής. Χρησιμοποιείται για τον τελικό συγκεντρωτικό πίνακα.

self.number_third=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα του τρίτου τετραγώνου προεκσκαφής

self.number_fourth=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα του τέταρτου τετραγώνου προεκσκαφής

self.number_lifters=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα δαπέδου

self.number_roof=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα οροφής

self.number_wall=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα του τοίχου

self.number_stoping=[ ] : Λίστα ποσότητας φυσιγγίων στα διατρήματα διεύρυνσης

self.spacing=[ ] : Λίστα με τις αποστάσεις των εκρηκτικών

self.No_columns=[ ] : Ο αριθμός των στηλών οριζοντίων διατρημάτων διεύρυνσης

self.No_rows=[ ] : Ο αριθμός των σειρών οριζοντίων διατρημάτων διεύρυνσης

self.spacing_columns=[ ] : Η απόσταση μεταξύ των διατρημάτων των στηλών (διεύρυνσης)

self.spacing_rows=[ ] : Η απόσταση μεταξύ των διατρημάτων σε κάθε σειρά (διεύρυνσης)

self.corner_holes=[ ] : Η απόσταση των γωνιακών διατρημάτων από τα επόμενα, στα διατρήματα δαπέδου

self.Spacing=[ ] : Η απόσταση στηλών καθέτων διατρημάτων

self.cols_downwards=[ ] : Ο αριθμός στηλών καθέτων διατρημάτων

Υπολογισμοί

Μετά τον ορισμό των σταθερών τιμών και λιστών, δημιουργείται η συνάρτηση **def addcartridge(self, name, diameter, length)**. Μέσα σε αυτή τη συνάρτηση γίνεται καταχώρηση των χαρακτηριστικών των φυσιγγίων που υπάρχουν στο αρχείο .json στη λίστα self.cartridges. Με τον όρο self.cartridges.append προστίθενται στοιχεία στη λίστα, η οποία είχε αρχικώς δηλωθεί κενή. Για παράδειγμα, το πρώτο στοιχείο στη λίστα self.cartridges των φυσιγγίων είναι το *«cart1»* με διάμετρο *0.028 m* και μήκος *0.255 m*. Επομένως με τη χρήση του self.cartridges[0]["length"] μας δίνεται το μήκος του πρώτου φυσιγγίου, στη συγκεκριμένη περίπτωση ο αριθμός 0.255 m.

Επόμενη συνάρτηση που χρησιμοποιείται και αφορά σε αρχικούς υπολογισμούς είναι η συνάρτηση **def charge_concentration(self)**. Σκοπός της συνάρτησης αυτής είναι να υπολογιστεί η γραμμική πυκνότητα εκρηκτικής γόμωσης του κάθε τύπου φυσιγγίου που χρησιμοποιείται και να καταχωρηθεί στην αντίστοιχη λίστα self.charge. Ο υπολογισμός γίνεται με την εξίσωση:

$$I = \left(\pi * \frac{d^2}{4}\right) * Density \qquad \text{(kg/m)} \quad [\text{Εξ. 31}]$$

Με το βρόγχο *for x in self.cartridges*, ο υπολογιστής ανατρέχει στη λίστα των εκρηκτικών που έχουν καταχωρηθεί προηγουμένως και παίρνει δεδομένα από το κάθε στοιχείο της λίστας. Έπειτα υπολογίζει τη γραμμική πυκνότητα γόμωσης του κάθε φυσιγγίου που καταχωρήθηκε ανάλογα με τη διάμετρό του *(d)* και την πυκνότητα της εκρηκτικής ύλη με τον παραπάνω τύπο *[Εξ. 31]*. Μετά από κάθε επανάληψη, γίνεται προσθήκη της πυκνότητας γόμωσης που υπολογίστηκε, με τη χρήση του self.charge.append . Έτσι λοιπόν, για κάθε στοιχείο στη λίστα self.cartridges υπολογίστηκε και μία πυκνότητα γόμωσης που τοποθετήθηκε στη λίστα self.charge στην αντίστοιχη θέση. Για παράδειγμα, σε πρώτη φάση το πρόγραμμα ανέτρεξε στο φυσίγγιο «cart1» και υπολόγισε την γραμμική πυκνότητα γόμωσής του ανάλογα με τη διάμετρό του *(0.028 m*). Επομένως καλώντας το self.charge[0], εμφανίζεται η πυκνότητα γόμωσης που υπολογίστηκε για το πρώτο φυσίγγιο. Ο όρος *round(.. , 2)* χρησιμοποιείται σχεδόν σε όλους τους υπολογισμούς για να γίνεται αυτόματη στρογγυλοποίηση στο δεύτερο δεκαδικό του κάθε αριθμού.

Ένα βήμα πριν τον υπολογισμό των διατρημάτων της προεκσκαφής *(cut)*, σειρά έχει η συνάρτηση **def calculation(self)**. Πρόκειται για τον τελευταίους υπολογισμούς πριν το κύριο σώμα της μεθόδου του Holmberg. Εδώ, υπολογίζεται η σχετική ισχύς του εκρηκτικού που χρησιμοποιείται ως προς τη δυναμίτιδα LFB και έπειτα ο αριθμός αυτός διαιρείται με το 0.84 για να υπολογιστεί η σχετική ισχύς του εκρηκτικού ως προς τον πετραμμωνίτη-ANFO *[Εξ. 1]*. Επίσης υπολογίζονται κατά σειρά το βάθος των διατρημάτων που είναι σταθερό σε όλη την εργασία (*self.hdepth*) καθώς και η προχώρηση της διάτρησης (*self.advance*). *[Εξ. 2] , [Εξ. 3]*

Εκτός αυτών των υπολογισμών, είναι σημαντικό να αναφερθεί η σημασία των λιστών που υπάρχουν στη συνάρτηση αυτή. Για κάθε τομέα της μεθόδου του Holmberg, θα παρουσιαστεί τελικός συγκεντρωτικός πίνακας που θα περιλαμβάνει τον τύπο και την ποσότητα του φυσιγγίου εκρηκτικής ύλης που χρησιμοποιείται για κάθε διάτρημα. Η λίστα *self.number_first* δηλώνει την ποσότητα του κάθε φυσιγγίου που θα χρησιμοποιηθεί στο πρώτο τετράγωνο της προεκσκαφής. Για παράδειγμα, αν στο πρώτο τετράγωνο χρειαστούν 2 τεμάχια του 1^{ου} τύπου φυσιγγίων, 0 του 2^{ου} τύπου φυσιγγίων και 3 τεμάχια του 3^{ου} τύπου φυσιγγίων, τότε το στοιχείο self.number_first[0] θα βγάλει αποτέλεσμα [2, 0, 3]. Επομένως, στη συνάρτηση αυτή ο κώδικας ανατρέχει στην ποσότητα των τύπων φυσιγγίων που έχουν δηλωθεί και δημιουργεί λίστες με την ίδια ποσότητα θέσεων που αρχικά λαμβάνουν την τιμή 0, για να επεξεργαστούν στη συνέχεια.

Πρώτο Τετράγωνο Προεκσκαφής

Η συνάρτηση **def first_quadrangle(self)** περιλαμβάνει όλους τους απαραίτητους υπολογισμούς για τον υπολογισμό των διατρημάτων του πρώτου τετραγώνου προεκσκαφής από τον αλγόριθμο. Υπολογίζεται αρχικά το μέγιστο φορτίο *V1max* ως συνάρτηση του αρχικού κενού διατρήματος *(self.emptyf)*. Έπειτα, ανάλογα αν η απόκλιση των διατρημάτων είναι μικρότερη του 1% , το πρακτικό φορτίο *V1* θα πάρει τιμή 1.5 φορές τη διάμετρο του αρχικού κενού διατρήματος. Αν όμως υπερβαίνει το 1% , τότε το πρακτικό φορτίο *V1* θα μειωθεί βάσει του μεγίστου φορτίου *V1max* και της απόκλισης διάτρησης *[Εξ. 6]*. Ακολουθεί υπολογισμός της γραμμικής πυκνότητας που απαιτείται για τη γόμωση των διατρημάτων *[Εξ. 9]*, με τον όρο charge. Σε αυτό το σημείο ο κώδικας καλείται να ανατρέξει στη λίστα self.charge *(for i in range(len(self.charge))* όπου βρίσκονται οι γραμμικές πυκνότητες γόμωσης των φυσιγγίων που διατίθενται και να τις συγκρίνει με την τιμή που υπολογίστηκε. Θέτοντας ένα ελάχιστο σημείο (*minimum=1000*), εξετάζεται η διαφορά των πυκνοτήτων. Αν η διαφορά αυτή είναι μικρότερη του *minimum*, και αν η πυκνότητα γόμωσης της λίστας είναι μικρότερη απ'αυτή που απαιτείται τότε ο κώδικας δίνει στον όρο *minimum* τη νέα διαφορά και εξετάζει πάλι τα ενδεχόμενα. Με αυτή τη λογική επιλέγεται η θέση *(th=i)* του φυσιγγίου που έχει την ελάχιστα μικρότερη γραμμική πυκνότητα γόμωσης με αυτή που υπολογίστηκε. Το h αναφέρεται στην επιγόμωση που υπολογίζεται 10 φορές τη διάμετρο των διατρημάτων self.hdiam *[Εξ. 19]*.

Έπειτα υπολογίζεται η πλευρά του πρώτου τετραγώνου της προεκσκαφής *B1* ως τετραγωνική ρίζα του πρακτικού φορτίου *V1* και ο αριθμός των φυσιγγίων που απαιτώνται (*No_first*). Ο αριθμός των φυσιγγίων υπολογίζεται με τη σχέση :

$$No\_first = \frac{H - h}{self.cartridges[th][\text{length}]} \qquad [\text{Εξ. } 32]$$

Όπου *H* βάθος διατρήματος και *self.cartridges[th][length]* μήκος φυσιγγίου, αναγράφονται σαν στοιχεία της αντίστοιχης λίστας καθώς υπολογίστηκαν σε διαφορετικές συναρτήσεις. Στον τελικό συγκεντρωτικό πίνακα του κώδικα υπολογίζεται και η συνολική γόμωση ανά διάτρημα σαν συνάρτηση της πυκνότητας γόμωσης του φυσιγγίου που χρησιμοποιείται επί τον αριθμό των φυσιγγίων και του μήκους τους. Αυτός ο αριθμός συμβολίζεται με το *mass1* και ο υπολογισμός είναι ο εξής:

$$mass1 = self.cartridges[th][\text{length}] * No\_first * I \quad \text{(kg)} \quad [\text{Εξ. } 33]$$

Όπου *self.cartridges[th]["length"]* το μήκος φυσιγγίου που επιλέγεται, *No_first* ο αριθμός των φυσιγγίων και *I* η γραμμική πυκνότητα γόμωσής του.

Η απόσταση των διατρημάτων *B1* (πλευρά πρώτου τετραγώνου) υπολογίζεται ως εξής:

$$B1 = \sqrt{2} * V1 \qquad \text{(m)} \quad [\text{Εξ. 34}]$$

Τοποθετούνται στις αντίστοιχες λίστες το πρακτικό φορτίο *V1*, η απόσταση των διατρημάτων *B1* και η συνολική γόμωση ανά διάτρημα (*mass1*). Οι λίστες αυτές είναι κατά σειρά self.practicalb, self.B και self.massph. Ο αριθμός των διατρημάτων είναι σταθερός καθώς μιλάμε για τετράγωνο προεκσκαφής οπότε και ισούται με 4 (*Holes_first*).

Στο τελευταίο σκέλος του υπολογισμού της συνάρτησης, δημιουργείται ένας βρόγχος κατά τον οποίο ο κώδικας λαμβάνει τη λίστα self.number_first και τοποθετεί στην αντίστοιχη θέση *(th)* που επιλέχθηκε παραπάνω, τον αριθμό του τύπου φυσιγγίων που υπολογίστηκε (*No_first*). Έτσι λοιπόν η λίστα θα έχει στη συγκεκριμένη θέση του φυσιγγίου τον αριθμό που χρειάζεται και οι υπόλοιπες θα παραμείνουν 0. Αν λοιπόν χρειαστούν 5 τεμάχια του 2^{ου} τύπου φυσιγγίου η λίστα θα γραφτεί ως εξης: [0, 5, 0].

Δεύτερο Τετράγωνο Προεκσκαφής

Αρχικά υπολογίζεται η θεωρητική πλευρά του δευτέρου τετραγώνου *B2temp* βάσει της σχέσης *[Εξ.12]* :

$$B2temp = \sqrt{2} * (V1 - F) \qquad \text{(m)} \quad [\text{Εξ. 35}]$$

Όπου *V1* πρακτικό φορτίο πρώτου τετραγώνου και *F* απόκλιση διάτρησης.

Έπειτα, μέσω της χρήσης βρόγχου υπολογίζεται το μέγιστο φορτίο *V2max* κάθε φυσιγγίου με τη θεωρητική πλευρά τετραγώνου *B2temp [εξ. 11]* και το πρακτικό φορτίο *V2 [εξ. 14]*. Τα πρακτικά φορτία που υπολογίζονται καταχωρούνται σε μια προσωρινή λίστα practicalb=[] η οποία δεν χρειάζεται να αποθηκευτεί γι'αυτό δεν παίρνει τον όρο self. Μέσα από αυτή τη λίστα η συγκεκριμένη συνάρτηση θα επιλέξει τη θέση του φυσιγγίου (*th)* του οποίου το πρακτικό φορτίο που υπολογίστηκε πληροί τις προϋποθέσεις. Μέσα στο βρόγχο «τρέχει» μια συνθήκη η οποία επιλέγει το φυσίγγιο (*th)* από τη λίστα practicalb που θα χρησιμοποιηθεί δεδομένου ότι το πρακτικό φορτίο του θα είναι το μέγιστο δυνατό, μεγαλύτερο του μισού της θεωρητικής πλευράς δευτέρου τετραγώνου *B2temp* και δεν θα ξεπερνά τη διπλάσια τιμή της *[εξ. 15, εξ. 18]*. Στη συνέχεια, αφού πλέον είναι γνωστός ο τύπος φυσιγγίου που θα χρησιμοποιηθεί *(th)* , ο κώδικας υπολογίζει το μήκος επιγόμωσης *h* , τον αριθμό των φυσιγγίων *No_second* και τη συνολική μάζα γόμωσης *mass2* όπως ακριβώς και για το πρώτο τετράγωνο και τα τοποθετεί στις αντίστοιχες λίστες τους. Η απόσταση των διατρημάτων στο δεύτερο τετράγωνο υπολογίζεται ως εξής:

$$B2 = \sqrt{2} * (V2 + \frac{B1}{2}) \quad \text{(m)} \quad \text{[Εξ. 36]}$$

Όπου *V2* το πρακτικό φορτίο δευτέρου τετραγώνου και *B1* απόσταση διατρημάτων πρώτου τετραγώνου. Όπως έχει αναφερθεί, οι υπολογισμοί προηγούμενης συνάρτησης μπορούν να χρησιμοποιηθούν μόνο μέσω λίστας επομένως η απόσταση διατρημάτων *B1* αναγράφεται σαν το στοιχείο στην αντίστοιχη λίστα *self.B[0]*. Τέλος, τοποθετείται και αυτή η απόσταση στη λίστα *(self.B.append).*

Τρίτο & Τέταρτο Τετράγωνο Προεκσκαφής

Η διαδικασία υπολογισμού είναι αποκλειστικά παρόμοια με αυτή που πραγματοποιήθηκε για τον υπολογισμό του δευτέρου τετραγώνου προεκσκαφής. Παρ'όλα αυτά, υπάρχουν διαφορές στον υπολογισμό της θεωρητικής πλευράς τρίτου τετραγώνου *B3temp* και της απόστασης διατρημάτων *B3*. Θεωρητική πλευρά τετραγώνου:

$$B3temp = \sqrt{2} * (V2 + \frac{B1}{2} - F) \quad \text{(m)} \quad \text{[Εξ. 37]}$$

Ενώ η απόσταση των διατρημάτων του τρίτου τετραγώνου:

$$B3 = \sqrt{2} * (V3 + \frac{B2}{2}) \quad \text{(m)} \quad \text{[Εξ. 38]}$$

Όπου *V3* το πρακτικό φορτίο των διατρημάτων του τρίτου τετραγώνου. Κατ'αντιστοιχία γίνονται και οι υπολογισμοί του τέταρτου και τελευταίου τετραγώνου της προεκσκαφής.

Διατρήματα Δαπέδου

Στα διατρήματα δαπέδου **def_lifters(self)** η λογική επιλογής φυσιγγίου γίνεται με την προϋπόθεση το μέγιστο φορτίο να είναι μικρότερο από το 60% του βάθους διατρήματος. Για αυτόν τον περιορισμό, ο όρος *x* είναι το 60% του βάθους διατρήματος. Στο βρόγχο που αναπτύσσεται , υπολογίζεται το μέγιστο φορτίο *vlmax* των φυσιγγίων με συντελεστή περιορισμού f=1.45 και σχέση απόστασης διατρημάτων-φορτίου E/V=1 *[Εξ. 20]*. Έπειτα επιλέγεται το φυσίγγιο από τη λίστα φυσιγγίων *(th)*, του οποίου το μέγιστο φορτίο *VLmax* είναι μικρότερο του 60% του βάθους του διατρήματος. Στη συνέχεια, σε άλλον έναν περιορισμό της μεθόδου του Holmberg, αν το φορτίο αυτό είναι μικρότερο από 1.4 m, η διορθωμένη σταθερά πετρώματος $\overline{c}$ στον τύπο υπολογισμού του φορτίου αλλάζει κατά c+ 0.07/VLmax *[Εξ. 21]*.

Ακολουθεί ο υπολογισμός του συνολικού αριθμού των διατρημάτων δαπέδου *N [Εξ. 22]*, η απόσταση των ενδιαμέσων διατρημάτων δαπέδου *El [Εξ. 23]*, καθώς και η απόσταση των 2 γωνιακών διατρημάτων δαπέδου από τα υπόλοιπα *corner_El [εξ. 24]*. Υπολογίζεται το πρακτικό φορτίο των διατρημάτων δαπέδου *VL [Εξ. 25]* και προστίθεται στη λίστα πρακτικών φορτίων self.practicalb. Η απόσταση των διατρημάτων δαπέδου *El* καταχωρείται στη λίστα self.spacing, ενώ η απόσταση των γωνιακών διατρημάτων *corner_El* τοποθετείται σε άλλη λίστα self.corner_holes. Επειδή στα διατρήματα δαπέδου υπάρχει επιγόμωση, η πυκνότητα γόμωσης στη στήλη και στον πυθμένα του διατρήματος αλλάζει. Επομένως ενδεχομένως να αλλάξει και ο τύπος και η ποσότητα των φυσιγγίων που απαιτώνται. Έτσι λοιπόν, αρχικά υπολογίζεται το μήκος γόμωσης του πυθμένα h_b *[Εξ. 26]*. Το μήκος γόμωσης της στήλης *hc* υπολογίζεται βάσει του μήκους γόμωσης πυθμένα *[Εξ. 27]*. Αφού ολοκληρωθούν οι παραπάνω υπολογισμοί, ο κώδικας θα πρέπει να υπολογίζει την πυκνότητα γόμωσης στήλης *charge_column*:

$$charge\_column = 70\% * self.charge[th] \quad \text{(kg/m)} \quad [\text{Εξ. 39}]$$

Όπου *self.charge[th]* είναι η πυκνότητα γόμωσης του φυσιγγίου που επιλέχθηκε για τη γόμωση του πυθμένα.

Με τη δημιουργία ενός ακόμα βρόγχου, και με νέα θέση φυσιγγίου (*th1)*, υπολογίζεται το φυσίγγιο που θα επιλεχθεί για τη γόμωση στήλης αυτή τη φορά όμως με βάση την πυκνότητα γόμωσης στήλης *charge_column* (+0.05 για να λαμβάνεται τυχόν φυσίγγιο που θα έχει μεγαλύτερη πυκνότητα γόμωσης απ'αυτή που απαιτείται αλλά με ελάχιστη διαφορά). Κατά τον υπολογισμό, ο κώδικας θα επιλέξει το φυσίγγιο εκείνο με την πλησιέστερη πυκνότητα γόμωσης απ'αυτή που υπολογίστηκε. Με αυτό τον τρόπο στο τέλος της συνάρτησης θα υπάρχουν δύο τύποι φυσιγγίων, αυτά στη θέση *th* για τη γόμωση πυθμένα και αυτά στη θέση *th1* για τη γόμωση στήλης. Ο αριθμός των φυσιγγίων για τον πυθμένα *No_cartridges_bottom* και ο αριθμός φυσιγγίων για τη στήλη *No_cartridges_column* υπολογίζονται βάσει της εξίσωσης:

$$No\_cartridges\_bottom = \frac{hb}{self.cartridges[th][\text{length}]} \quad [\text{Εξ. 40}]$$

Όπου *hb* το μήκος επιγόμωσης πυθμένα και *self.cartridges[th]["length"]* το μήκος του φυσιγγίου που επιλέχθηκε για τη γόμωση πυθμένα.

Αντίστοιχα για τον αριθμό φυσιγγίων για τη στήλη χρησιμοποιείται ο ίδιος τύπος με το hc και το self.cartridges[th1]["length"] *[Εξ. 40]*. Στο τελευταίο κομμάτι της συνάρτησης υπολογίζεται η συνολική μάζα γόμωσης *massL* αυτή τη φορά με τη χρήση και των δύο ειδών φυσιγγίων που χρησιμοποιήθηκαν και τοποθετείται στην αντίστοιχη λίστα self.massph. Επίσης στη λίστα self.holes που αναφέρεται στον αριθμό των διατρημάτων, τοποθετείται ο

αριθμός των διατρημάτων δαπέδου *N*. Ο βρόγχος που έπεται, αφορά στην καταχώρηση του αριθμού του κάθε φυσιγγίου που απαιτήθηκε για τα διατρήματα δαπέδου. Για λόγους κάλυψης των ενδεχομένων, σε περίπτωση που το *th* είναι ίδιο με το *th1*, δηλαδή στην περίπτωση που χρησιμοποιείται το ίδιο φυσίγγιο για τη γόμωση πυθμένα και στήλης, ο κώδικας προσθέτει τον αριθμό των φυσιγγίων εκατέρωθεν και δεν επιλέγει μόνο το ένα. Η λίστα που απεικονίζει τον αριθμό των φυσιγγίων για τα διατρήματα δαπέδου είναι η self.number_lifters.

Διατρήματα Οροφής

Η συνάρτηση ονομάζεται **def_roof(self)**. Αρχικά δηλώνεται σταθερά k που λαμβάνει τυχαία τιμή στο διάστημα [15,16]. Αυτή η τιμή χρησιμεύει στον υπολογισμό της απόστασης των διατρημάτων οροφής *E* και στο συγκεκριμένο παράδειγμα ορίστηκε k=15 *[Εξ. 29]*. Έπειτα υπολογίζεται το μέγιστο φορτίο διατρημάτων οροφής *VRmax (=E/0.84)* και το πρακτικό φορτίο *VR* και καταχωρείται στην αντίστοιχη λίστα *[Εξ. 25]*. Στα διατρήματα οροφής χρησιμοποιείται, βάσει της μεθόδου του Holmberg, τεχνική ελεγχόμενης ανατίναξης ή αλλιώς *smooth blasting*. Αυτό σημαίνει ότι η επιλογή του τύπου φυσιγγίου (*th)* θα γίνει με βάση την ελάχιστη πυκνότητα γόμωσης που απαιτείται *min_charge [Εξ. 30]*. Μέσα από βρόγχο επιλέγεται το κατάλληλο φυσίγγιο *th* από τη λίστα των φυσιγγίων, αυτό δηλαδή με την πλησιέστερη πυκνότητα γόμωσης σε σχέση με την ελάχιστη που απαιτείται. Λόγω έλλειψης επιγόμωσης, κάθε διάτρημα οροφής θα γομωθεί εξ'ολοκλήρου επομένως ο αριθμός φυσιγγίων θα υπολογιστεί ως εξής:

$$No\_cartridges\_roof = \frac{H}{self.cartridges[th][\text{length}]} \qquad [\text{Εξ. } 41]$$

Όπου H το βάθος διατρήματος και ο παρονομαστής αναφέρεται στο μήκος του φυσιγγίου που επιλέχθηκε. Ο συνολικός αριθμός διατρημάτων οροφής υπολογίζεται με τη χρήση της παρακάτω σχέσης:

$$No\_holes\_roof = int(\frac{Height + (Height * sing) + Arch\ Height}{E} + 1) \qquad [\text{Εξ. } 42]$$

Όπου *Height* ύψος μετώπου, g γωνιακή απόκλιση για τα περιμετρικά διατρήματα, *Arch Height* ύψος αψίδας και Ε απόσταση διατρημάτων οροφής. Λαμβάνεται ο ακέραιος της τιμής αυτής. Ο αριθμός των διατρημάτων εντάσσεται επίσης στη λίστα self.holes. Τέλος,

υπολογίζεται επίσης η συνολική μάζα γόμωσης διατρημάτων οροφής *massR*, τοποθετείται στη λίστα και με έναν ακόμη βρόγχο όπως και στις προηγούμενες συναρτήσεις τοποθετείται στη λίστα self.number_roof ο αριθμός των φυσιγγίων.

Διατρήματα Τοίχου

Η συνάρτηση που χρησιμοποιείται εδώ είναι η **def_wall(self)**. Η λογική των υπολογισμών είναι η ίδια με αυτή των διατρημάτων δαπέδου. Με τον όρο x που συμβολίζεται το 60% του βάθους διατρήματος, μέσα από βρόγχο επιλέγεται το φυσίγγιο *th* του οποίου το μέγιστο φορτίο είναι ελάχιστα μικρότερο από το x. Αυτή τη φορά, για τον υπολογισμό του μεγίστου φορτίου των φυσιγγίων, χρησιμοποιείται συντελεστής περιορισμού f=1.2 και η σχέση απόστασης-φορτίου παίρνει την τιμή E/V=1.25 *[Εξ. 20]*. Στην αρχή της συνάρτησης υπολογίζεται ο χώρος που μένει για τα διατρήματα του τοίχου *distance*. Αυτός ο χώρος αφορά στο συνολικό ύψος του μετώπου αφαιρώντας το πρακτικό φορτίο των διατρημάτων δαπέδου και αυτό των διατρημάτων οροφής. Ο αριθμός των διατρημάτων οροφής επιλέγεται από τη σχέση:

$$No\_holes\_wall = 2 * int\left(\frac{distance}{V * \frac{E}{V}} + 2\right) \qquad \text{[Εξ. 43]}$$

Όπου *distance* η απόσταση που απομένει για τα διατρήματα τοίχου, και ο παρονομαστής αφορά στη θεωρητική απόσταση των διατρημάτων αυτών.

Λαμβάνεται ο ακέραιος αριθμός και πολλαπλασιάζεται επί 2 γιατί το μέτωπο έχει δύο τοίχους εκατέρωθεν επομένως οι υπολογισμοί θα είναι ίδιοι και θα αφορούν και τα απέναντι διατρήματα. Η απόσταση των διατρημάτων τοίχου υπολογίζεται ως εξής:

$$Spacing = \frac{distance}{\frac{No\_holes\_wall}{2}} + 1 \qquad \text{[Εξ. 44]}$$

Όπου ο αριθμός διατρημάτων διαιρείται δια 2, επειδή υπολογίζεται η απόσταση *Spacing* για κάθε τοίχο και τα διατρήματα υπολογίστηκαν για τους δύο τοίχους συνολικά.

Όπως και στα διατρήματα δαπέδου, υπάρχει και εδώ επιγόμωση επομένως και διαφορετικός τύπος φυσιγγίου για τη γόμωση της στήλης (*th1)*. Η διαδικασία παραμένει η ίδια με τον υπολογισμό των μηκών γόμωσης πυθμένα *hb* και στήλης *hc*, τον υπολογισμό πυκνότητας γόμωσης στήλης *charge_column* και τον βρόγχο που θα επιλέξει το νέο φυσίγγιο με την

κατάλληλη πυκνότητα για τη γόμωση της στήλης. Έπειτα υπολογίζεται ο αριθμός των φυσιγγίων για τον πυθμένα και τη στήλη *No_cartridges_bottom* και *No_cartridges_column*, η συνολική μάζα γόμωσης ανά διάτρημα και ο τελικός βρόγχος που προσθέτει τον αριθμό του κάθε τύπου φυσιγγίων στη λίστα *self.number_wall*. Όλες οι τιμές όπως η απόσταση *spacing*, ο αριθμός των διατρημάτων *No_holes_wall* η συνολική μάζα γόμωσης ανά διάτρημα *massW*, τοποθετούνται στις αντίστοιχες λίστες.

Διατρήματα Διεύρυνσης

Η συνάρτηση για τον υπολογισμό των διατρημάτων διεύρυνσης είναι η **def_stoping(self)**. Αρχικά, τα διατρήματα διεύρυνσης χωρίζονται σε οριζόντια διατρήματα και κάθετα. Τα οριζόντια διατρήματα τοποθετούνται δίπλα στο τετράγωνο προεκσκαφής ενώ τα κάθετα διατρήματα αφορούν στο χώρο που βρίσκεται πάνω από το τετράγωνο προεκσκαφής. Χ ορίζεται το 60% του βάθους διατρήματος. Με τον βρόγχο που χρησιμοποιήθηκε σε κάθε προηγούμενη συνάρτηση, υπολογίζεται το φυσίγγιο εκείνο *(th)* του οποίου το μέγιστο φορτίο είναι μικρότερο από την τιμή x. Στα οριζόντια διατρήματα διεύρυνσης, για τον τύπο του μεγίστου φορτίου χρησιμοποιείται συντελεστής περιορισμού f=1.45 και σχέση απόστασης-φορτίου E/V=1.25 *[Εξ. 20]*. Υπολογίζεται η διαθέσιμη απόσταση *distance_h* κατά την οποία θα τοποθετηθούν τα διατρήματα αυτά. Στη συγκεκριμένη περίπτωση, το τετράγωνο προεκσκαφής τοποθετείται στο αριστερό άκρο του μετώπου αμέσως μετά τα διατρήματα του αριστερού τοίχου. Επομένως ο διαθέσιμος χώρος για οριζόντια διατρήματα είναι ανάμεσα στο τετράγωνο προεκσκαφής και τα διατρήματα του τοίχου δεξιά. Αφού επιλέγεται το κατάλληλο φυσίγγιο για τα οριζόντια διατρήματα διεύρυνσης και αφού έχει υπολογιστεί το μέγιστο φορτίο ,σειρά έχει το πρακτικό φορτίο *VH [Εξ. 14]*.

Στη συνέχεια, δημιουργείται μία συνθήκη που θα καλύπτει όλες τις περιπτώσεις υπολογισμού των οριζοντίων διατρημάτων διεύρυνσης. Για τον υπολογισμό των διατρημάτων αυτών απαιτείται η γνώση των στηλών και των σειρών διατρημάτων που μπορούν να «χωρέσουν» στο χώρο ανάμεσα στο τετράγωνο προεκσκαφής και τα διατρήματα τοίχου. Οι τιμές cols_h=0 και rows_h=0 αναφέρονται στις στήλες και στις σειρές οριζοντίων διατρημάτων διεύρυνσης και αρχικά θέτονται ως μηδέν. Έπειτα δημιουργείται μια μεγάλη συνθήκη που καλύπτει όλες τις περιπτώσεις:

- Σε περίπτωση που το πρακτικό φορτίο *VH* είναι μεγαλύτερο ή ίσο από το διαθέσιμο χώρο *distance_h*, τότε δημιουργείται νέο πρακτικό φορτίο με την τιμή του διαθέσιμου χώρου. Σε αυτή την περίπτωση όπως είναι αντιληπτό, οι στήλες διατρημάτων *cols_h* θα είναι 1 και η απόσταση *Spacing_columns_h* θα είναι όλος ο διαθέσιμος χώρος.

- Σε περίπτωση που το πρακτικό φορτίο είναι μικρότερο από τον διαθέσιμο χώρο, τότε οι στήλες υπολογίζονται ως η ακέραια τιμή της διαίρεσης του χώρου δια του φορτίου και η απόστασή τους *Spacing_columns_h* θα είναι το αποτέλεσμα της διαίρεσης του διαθέσιμου χώρου *distance_h* δια τον αριθμό στηλών *cols_h*.

- Αν το πρακτικό φορτίο *VH* είναι μικρότερο της πλευράς του τέταρτου τετραγώνου προεκσαφής *self.B[3]*, τότε ο αριθμός των σειρών *rows_h* οριζοντίων διατρημάτων υπολογίζεται από το ακέραιο μέρος της διαίρεσής τους και η απόσταση μεταξύ των σειρών *Spacing_rows_h* από τη διαίρεση του χώρου *self.B[3]* δια του πρακτικού φορτίου *VH*.

- Στην περίπτωση που το πρακτικό φορτίο είναι μεγαλύτερο ή ίσο από το μήκος της πλευράς του τέταρτου τετραγώνου και δεν «χωράνε» άλλες σειρές διατρημάτων, οι σειρές *rows_h* γίνονται 2 και η απόστασή τους ορίζεται σαν το μήκος της πλευράς.

Με το πέρας της συνθήκης ορίζεται ο συνολικός αριθμός οριζοντίων διατρημάτων (σειρές επί στήλες) με το όνομα *No_holes_h* ο οποίος θα προστεθεί με τον αντίστοιχο αριθμό των καθέτων διατρημάτων και θα αποτελούν τα συνολικά διατρήματα διεύρυνσης. Επίσης, ο αριθμός των σειρών και στηλών καταχωρείται σε ειδικές λίστες *self.No_rows* και *self.No_columns* . Σε αντίστοιχες λίστες καταχωρούνται και οι αποστάσεις των στηλών και των σειρών *self.spacing_rows, self.spacing_columns*. (Αυτές οι λίστες αφορούν μόνο στη σχεδίαση των διατρημάτων στο τελικό διάγραμμα.)

Ο υπολογισμός για τα κάθετα διατρήματα διεύρυνσης είναι λίγο διαφορετικός. Υπολογίζεται το μέγιστο φορτίο *VDmax* με τον ίδιο τύπο φυσιγγίου αλλά με συντελεστή περιορισμού f=1.2 και σχέση απόστασης φορτίου E/V=1.25 *[Εξ. 20]*. Έπειτα υπολογίζεται το πρακτικό φορτίο *VD [Εξ. 14]*. Τα κάθετα διατρήματα διεύρυνσης θα τοποθετηθούν στο χώρο μεταξύ διατρημάτων οροφής και διατρημάτων προεκσκαφής. Επειδή όμως το πρακτικό φορτίο των διατρημάτων οροφής είναι σε μορφή καμπύλης (λόγω οροφής), η διαδικασία είναι διαφορετική. Με το πρακτικό τους φορτίο *VD* και τη διαθέσιμη απόσταση μεταξύ των τοιχών υπολογίζονται οι στήλες που χρειάζονται για να καλύψουν τον χώρο κατά μήκος του μετώπου. Έπειτα, μέσα από βρόγχο υπολογίζεται η κατακόρυφη απόσταση μεταξύ του κάθε σημείου της καμπύλης και του αντίστοιχου σημείου πάνω από την προεκσκαφή. Η απόσταση αυτή διαιρείται με το πρακτικό φορτίο των καθέτων διατρημάτων διεύρυνσης *VD* και το αποτέλεσμα απεικονίζει πόσα διατρήματα χωράνε στην κάθετη αυτή απόσταση με δεδομένο το πρακτικό φορτίο τους. Αν για παράδειγμα ο αριθμός αυτός είναι 2, τότε χωράνε 2 διατρήματα. Αν όμως ο αριθμός αυτός είναι 0 (μεγαλύτερο φορτίο από απόσταση), τότε θα τοποθετηθεί ένα διάτρημα πάνω στην καμπύλη με πρακτικό φορτίο όσο η απόσταση καμπύλης-προεκσκαφής. Για κάθε στήλη (υπολογίστηκαν πριν) γίνεται η ίδια επανάληψη και τα διατρήματα διεύρυνσης τοποθετούνται αναλόγως βάσει συντεταγμένων. Κάθε διάτρημα που τοποθετείται, προστίθεται στον όρο carts που αναφέρεται στο συνολικό αριθμό καθέτων διατρημάτων. Τέλος, ο αριθμός αυτός προστίθεται στον συνολικό αριθμό οριζοντίων διατρημάτων και αφορούν τα συνολικά διατρήματα διεύρυνσης που καταχωρούνται στη λίστα self.holes.

Μετά από τα παραπάνω, σειρά έχει ο υπολογισμός των φυσιγγίων για τα διατρήματα διεύρυνσης. Αρχικά υπολογίζεται το μήκος επιγόμωσης πυθμένα *hb* και στήλης *hc*. Αυτή τη φορά, η πυκνότητα γόμωσης στήλης είναι κατά 50% μικρότερη απ'αυτή του πυθμένα. Επομένως, αφού υπολογιστεί η πυκνότητα γόμωσης στήλης *charge_column*, μέσα από τον συνηθισμένο βρόγχο, επιλέγεται το φυσίγγιο *(th1)* από τη λίστα φυσιγγίων του οποίου η

πυκνότητα γόμωσης είναι πλησιέστερη σε αυτή της στήλης. Η διαδικασία παρακάτω είναι γνωστή. Ακολουθούν ο αριθμός των φυσιγγίων πυθμένα και στήλης *No_cartridges_bottom*, *No_cartridges_column*, έπειτα η συνολική μάζα γόμωσης ανά διάτρημα για τα διατρήματα διεύρυνσης *massS* και καταχωρούνται στις αντίστοιχες λίστες.

Σχεδιασμός Πινάκων

Οι επόμενες συναρτήσεις αφορούν στη δημιουργία δύο τελικών συγκεντρωτικών πινάκων σαν αποτέλεσμα των υπολογισμών του κώδικα. Η συνάρτηση **def_sum(self)** χρησιμοποιείται για τον υπολογισμό της συνολικής μάζας εκρηκτικών στο κάθε μέρος της διάτρησης *self.total*. Με έναν βρόγχο, για κάθε διαφορετικό τύπο διατρημάτων πολλαπλασιάζεται η συνολική μάζα ανά διάτρημα *self.massph* επί τον αριθμό των διατρημάτων *self.holes*.

Στη συνέχεια, με τη συνάρτηση **def_table1(self)** δημιουργείται ο πρώτος πίνακας σε μορφή DataFrame καθώς οι τιμές θα πηγάζουν από στοιχεία λιστών. Ο πίνακας θα αναγράφει για κάθε τύπο διατρημάτων (π.χ. "Lifters") , τον αριθμό των διατρημάτων που απαιτώνται *(self.holes)*, την ποσότητα του κάθε τύπου φυσιγγίου που θα χρησιμοποιηθεί *(self.number_lifters)*, καθώς επίσης τη μάζα γόμωσης ανά διάτρημα *massph* και τη συνολική μάζα εκρηκτικού *(self.total)*.

Έπειτα, στο τέλος του προγράμματος παρουσιάζεται άλλος ένας πίνακας **def_table2(self)** που αφορά σε καίρια χαρακτηριστικά της διάτρησης όπως είναι ο συνολικός αριθμός διατρημάτων, η συνολική μάζα εκρηκτικών που θα χρησιμοποιηθούν για όλα τα τμήματα, η ειδική διάτρηση, η ειδική πυκνότητα γόμωσης και το εμβαδό του υπογείου μετώπου. Η συνολική μάζα εκρηκτικών *Total_weight* υπολογίζεται σαν το άθροισμα των στοιχείων της λίστας self.total όπου υπάρχουν οι μάζες των εκρηκτικών για κάθε τμήμα της διάτρησης. Ο συνολικός αριθμός διατρημάτων *Total_holes* πηγάζει από το άθροισμα των στοιχείων της λίστας self.holes όπου αναγράφονται τα διατρήματα για κάθε τμήμα της διάτρησης. Το εμβαδόν του μετώπου *Cross_sectional_area* υπολογίζεται ως [εμβαδόν παραλληλογράμμου + εμβαδόν κυκλικού τμήματος οροφής]. Με τη χρήση διαβήτη και χάρακα υπολογίστηκε ότι η ακτίνα του κύκλου που δημιουργεί η οροφή είναι ίση με r = (ύψος μετώπου + ύψος οροφής). Επομένως, υπολογίζοντας το εμβαδόν του κυκλικού δίσκου που δημιουργείται από την οροφή *(Cycle_area)* και αφαιρώντας απ'αυτόν το εμβαδόν του τριγώνου *Triangle_area* ($\frac{\pi\lambda\acute{\alpha}\tau o\varsigma * \acute{\upsilon}\psi o\varsigma}{2}$), το τελικό αποτέλεσμα είναι το εμβαδό της οροφής στο οποίο και προστίθεται το εμβαδό του υπόλοιπου μετώπου (παραλληλόγραμμο) και υπολογίζεται το συνολικό εμβαδόν *Cross_sectional_area*. Η τιμή της ειδικής κατανάλωσης εκρηκτικών *Specific_charge* δίνεται από τον τύπο:

$$Specific_{charge} = \frac{Total\_weight}{Cross\_sectional\_area * self.advance} \qquad [\text{Εξ. } 45]$$

Όπου *self.advance* η προχώρηση του μετώπου και η ειδική κατανάλωση υπολογίζεται σε μονάδες (kg/m^3). Η ειδική διάτρηση *Specific_drilling* δίνεται από τον τύπο:

$$Specific\_drilling = \frac{Total\_holes * self.hdepth}{Cross\_sectional\_area * self.advance} \qquad [Εξ. 46]$$

Όπου *self.hdepth* το βάθος διατρήματος και η ειδική διάτρηση υπολογίζεται σε μονάδες (m/m^3). Όλες οι παραπάνω τιμές εμφανίζονται στον πίνακα με τη μορφή DataFrame.

3.2.1. Σχεδιασμός Τελικού Σχεδίου διάτρησης του Μετώπου

Γεωμετρική Διατομή Στοάς

Με τη χρήση της τελευταίας συνάρτησης του προγράμματος **def_plot(self)** γίνεται ο σχεδιασμός του τελικού σχεδίου διάτρησης του μετώπου. Κάθε τμήμα της διάτρησης υπολογίζεται ξεχωριστά. Ο σχεδιασμός γίνεται μέσω συντεταγμένων και των λιστών που έχουν υπολογιστεί από τον κώδικα στις προηγούμενες συναρτήσεις. Αρχικά, ξεκινώντας από το σημείο (0,0) δημιουργείται ένα παραλληλόγραμμο που απεικονίζει το κύριο σώμα του μετώπου. Με τους όρους self.abutmentH (*ύψος*) και self.width (*πλάτος*) να προσθέτονται στις συντεταγμένες, ο κώδικας δημιουργεί μια γραφική παράσταση του ύψους και του πλάτους του μετώπου.

Σειρά έχει η οροφή η οποία αποτελεί τμήμα κύκλου. Για το σχεδιασμό της γίνεται χρήση της παρακάτω γνωστής εξίσωσης υπολογισμού κύκλου:

$$(\frac{\chi - \chi o}{\alpha})^2 + (\frac{\psi - \psi o}{\beta})^2 = 1 \qquad [Εξ. 47]$$

Όπου, $\alpha = \frac{πλάτος}{2}$, β το ύψος κυκλικού τμήματος οροφής *(self.abutmentH)*, $\chi_o = \frac{πλάτος}{2}$, ψ_o το ύψος κυκλικού τμήματος οροφής *(self.abutmentH)* και χ η καμπύλη στο διάστημα (0,πλάτος). Λύνοντας την εξίσωση ως προς ψ, και γράφοντας το ax.plot(x,y,color='black') ο κώδικας απεικονίζει το σκελετό του μετώπου με το σωστό τμήμα κύκλου για την οροφή.

Διατρήματα Δαπέδου

Στη συνέχεια γίνεται σχεδιασμός των ορίων και των διατρημάτων στα διατρήματα δαπέδου. Οι συντεταγμένες xL,yL αφορούν στο σχεδιασμό των ορίων των διατρημάτων δαπέδου ανάλογα με το πρακτικό τους φορτίο *self.practicalb[4]*. Με τον όρο plt.plot(xL,yL,color="blue") γίνεται η απεικόνιση των παραπάνω συντεταγμένων στο διάγραμμα του μετώπου. Έπειτα δημιουργείται ένας βρόγχος με πολλές συνθήκες για να υπολογίζει ο κώδικας την απόσταση μεταξύ των διατρημάτων δαπέδου και την τοποθέτησή τους αναλόγως. Θέτωντας αρχικά τον αριθμό των διατρημάτων δαπέδου l=self.holes[4] υπάρχουν οι παρακάτω περιπτώσεις:

- Το πρώτο διάτρημα *for i in range(0,l+1): if i==0:* θα τοποθετείται στο σημείο (0,0) δηλαδή στην άκρη του μετώπου

- Το δεύτερο διάτρημα θα τοποθετείται με την γωνιακή απόσταση δαιτρημάτων δαπέδου *(self.corner_holes[0])* όπως υπολογίστηκε στη συγκεκριμένη συνάρτηση, από το δεύτερο διάτρημα

- Το τελευταίο διάτρημα θα τοποθετείται στο τέλος του χώρου διατρημάτων δαπέδου. Επίσης το προηγούμενο απ'αυτό θα τοποθετείται με τη γωνιακή απόσταση *(self.corner_holes[0]).*

- Τέλος, τα ενδιάμεσα διατρήματα δαπέδου θα τοποθετούνται με την κανονική μεταξύ τους απόσταση *self.spacing[0].*

Έπειτα, με τον όρο plt.scatter(x1,y1,color='navy') γίνονται οι απεικονίσεις σαν σημεία στο διάγραμμα, όπου x1 και y1 οι συντεταγμένες των διατρημάτων δαπέδου.

Διατρήματα Οροφής

Ακολουθεί η απεικόνιση του ορίου και των διατρημάτων της οροφής. Με τη χρήση της εξίσωσης κύκλου που χρησιμοποιήθηκε παραπάνω, και με τη χρήση του πρακτικού φορτίου οροφής *(self.practicalb[5])* αναπαρίσταται το όριο των διατρημάτων οροφής, δηλαδή η ίδια καμπύλη μειωμένη κατά self.practicalb[5] .

Αμέσως μετά, με τη χρήση βρόγχου, κάθε διάτρημα στη λίστα των διατρημάτων οροφής *self.holes[5]* θα τοποθετείται στην εξίσωση της καμπύλη της οροφής με συντεταγμένες xr,yr και θα απεικονίζεται με τον όρο plt.scatter(xr,yr,color='brown'). Επομένως η καμπύλη της οροφής θα αποκτήσει τα απαραίτητα διατρήματα περιμετρικά στις αποστάσεις που υπολογίστηκαν.

Διατρήματα Τοίχων

Με την ίδια λογική του σχεδιασμού των διατρημάτων δαπέδου, γίνεται και ο σχεδιασμός για τα διατρήματα του τοίχου. Ορίζονται αρχικές συντεταγμένες xW1,yW2 που θα απεικονίζουν μία γραμμή μεταξύ οροφής και διατρημάτων δαπέδου, με πλάτος όσο το πρακτικό φορτίο των διατρημάτων του τοίχου *self.practicalb[4]*. Ουσιαστικά πρόκειται για τα όρια του τοίχου. Επίσης, μιας και μιλάμε για δύο τοίχους εκατέρωθεν του μετώπου, η ίδια διαδικασία γίνεται και για την απέναντι πλευρά αυτή τη φορά με συντεταγμένες xW2,yW2 . Αφού μιλάμε για γραφική παράσταση, απεικονίζονται αμφότερα με τους όρους plt.plot(xW1,yW1, color='black') και plt.plot(xW2,yW2,color='black') αντίστοιχα. Έπειτα, μέσα από βρόγχο, κάθε διάτρημα του τοίχου θα τοποθετείται στην κατάλληλη θέση συντεταγμένων x1,y1 και x2,y2 με απόσταση μεταξύ τους *self.spacing[2]* όπως αυτή υπολογίστηκε από τη συνάρτηση για τα διατρήματα τοίχου. Τέλος, χρησιμοποιείται ο όρος plt.scatter(x1,y1,color='green') και plt.scatter(x2,y2, color='green') αντίστοιχα για να απεικονιστούν στο διάγραμμα.

Διατρήματα Προεκσκαφής

Σειρά στον κώδικα έχει ο σχεδιασμός των τετραγώνων προεκσκαφής. Αυτή τη φορά, αρχικό σημείο ορίζεται το αρχικό κενό διάτρημα με συντεταγμένες xQ,yQ. Όπως έχει προαναφερθεί, η προεκσκαφή γίνεται στο αριστερό σημείο του μετώπου ακριβώς δίπλα από το όριο του τοίχου και πάνω από το όριο των διατρημάτων δαπέδου. Για τον ακριβή εντοπισμό του βέβαια, χρησιμοποιείται το μισό της πλευρά τους 4^{ου} τετραγώνου προεκσκαφής μιας και θα βρίσκεται στο επίκεντρο του τετραγώνου. Με δεδομένες τις συντεταγμένες του κενού διατρήματος στη μέση, ξεκινά ο υπολογισμός των συντεταγμένων των υπολοίπων τετραγώνων. Η διαδικασία αυτή περιλαμβάνει τη χρήση ημιτόνων και συνημιτόνων των πλευρών τους, μιας και θα πρέπει να τοποθετηθούν υπό γωνία για το σχηματισμό των τετραγώνων. Κατ'αντιστοιχία χρησιμοποιούνται οι συντεταγμένες x1,y1 για το πρώτο τετράγωνο, x2,y2 για το δεύτερο τετράγωνο , x3y3 για το τρίτο τετράγωνο και x4,y4 για το τελευταίο. Οι συντεταγμένες υπολογίζονται με βάση τις πλευρές του κάθε τετραγώνου και τα πρακτικά φορτία τα οποία βρίσκονται στις λίστες self.B() και self.practicalb() αντίστοιχα. Με τους όρους plt.scatter(x..,y..) απεικονίζονται στο τελικό διάγραμμα.

Διατρήματα Διεύρυνσης

Το τελευταίο κομμάτι του σχεδιασμού περιλαμβάνει την τοποθέτηση διατρημάτων διεύρυνσης στο χώρο που απομένει στο μέτωπο. Για τα οριζόντια διατρήματα διεύρυνσης ορίζονται αρχικές συντεταγμένες xS,yS στο πάνω δεξί άκρο του τέταρτου τετραγώνου και αφορούν στα διατρήματα που θα τοποθετηθούν στο διαθέσιμο χώρο μεταξύ προεκσκαφής και τοίχου. Με δύο βρόγχους, και με τη χρήση του αριθμού στηλών και σειρών διατρημάτων και της απόστασής τους ο κώδικας ελέγχει και τοποθετεί το κάθε διάτρημα που απαιτείται στην κατάλληλη απόσταση. Έτσι οι στήλες οριζοντίων διατρημάτων έχουν συντεταγμένες xh,yh και οι σειρές των οριζοντίων διατρημάτων έχουν συντεταγμένες xh1,yh1.

Για τα κάθετα διατρήματα διεύρυνσης ακολουθείται διαφορετική διαδικασία, όπως και η διαδικασία υπολογισμού τους. Για κάθε στήλη καθέτων διατρημάτων, ο κώδικας υπολογίζει τον αριθμό διατρημάτων που «χωράνε» στον κατακόρυφο άξονα και ανάλογα τον συγκεκριμένο αριθμό, οι συντεταγμένες XS1,YS1, προσθέτουν διατρήματα στο διάγραμμα μέσα από βρόγχο. Σε κάθε κομμάτι του σχεδιασμού, υπάρχει ο όρος ax.annotate ο οποίος χρησιμοποιείται για την απεικόνιση των αποστάσεων μεταξύ των διατρημάτων.

3.2.2. Τελικό τμήμα Κώδικα

Το τελευταίο κομμάτι του προγράμματος αποτελεί το κομμάτι εκκίνησης του κώδικα. Αναγράφοντας τα πακέτα υποστήριξης της Python που απαιτώνται για τις προηγούμενες συναρτήσεις, χρησιμοποιείται ο όρος explosives για να «διαβάσει» ο κώδικας την κλάση *Blaster()* που δημιουργήθηκε. Παράδειγμα με την εισαγωγή του όρου explosives.lifters() , ο κώδικας ανατρέχει στην κλάση *Blaster* και πραγματοποιεί μόνο τη συνάρτηση **def_lifters**. Είναι σημαντικό λοιπόν στο τέλος του προγράμματος να τοποθετηθούν όλες οι συναρτήσεις με τον όρο explosives κατά σειρά για να γίνουν με την αντίστοιχη σειρά και οι υπολογισμοί του κώδικα.

Έπειτα, ο κώδικας καλείται να «διαβάσει» το αρχείο config.json και να αντιστοιχήσει τις σταθερές που αναγράφονται με τις αντίστοιχες τιμές του αρχείου. Για παράδειγμα, με το explosives.hdiam = json_value["geometry"]["hdiam"] ο κώδικας διαβάζει από το αρχείο config.json το βάθος διατρημάτων *(hdiam*) που όρισε ο χειριστής στην κατηγορία *geometry* και το αντιστοιχεί με την τιμή explosives.hdiam δηλαδή self.hdiam όπως αναφέρθηκε στην αρχή του προγράμματος. Πλέον το πρόγραμμα είναι έτοιμο για χρήση!

ΚΕΦΑΛΑΙΟ 4: ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα της παρούσας Διπλωματικής Εργασίας. Παρουσιάζονται επακριβώς οι Πίνακες και το Διάγραμμα του κώδικα όπως απεικονίζονται στον κειμενογράφο VSCode

4.1. Επιμέρους Αποτελέσματα Κώδικα

Με την ολοκλήρωση του κώδικα για το σχεδιασμό ανατινάξεων υπογείων μετώπων με τη μορφή στοάς, σειρά έχει ο ορισμός των παραμέτρων της συγκεκριμένης διπλωματικής εργασίας στο αρχείο config.json. Οι τιμές που ορίζονται στο αρχείο, αναγράφονται στο ΚΕΦΑΛΑΙΟ 2 (2.2. Παράμετροι Προγράμματος). Με την αποθήκευση του αρχείου έπειτα την επεξεργασία και την καταχώρηση των τιμών, ο κώδικας ξεκινάει τη διαδικασία υπολογισμού (Run).

Η στοά στην οποία καλείται να γίνει ο βέλτιστος σχεδιασμός είναι στοά διευθυντική με διατομή 4.5 m x 4.5 m. Το ύψος των κατακόρυφων τοιχωμάτων είναι 4 μέτρα και η στέψη είναι μορφής αψίδας που αποτελεί τμήμα κυκλικού δίσκου με ύψος 0.5 m *(height of arch).*

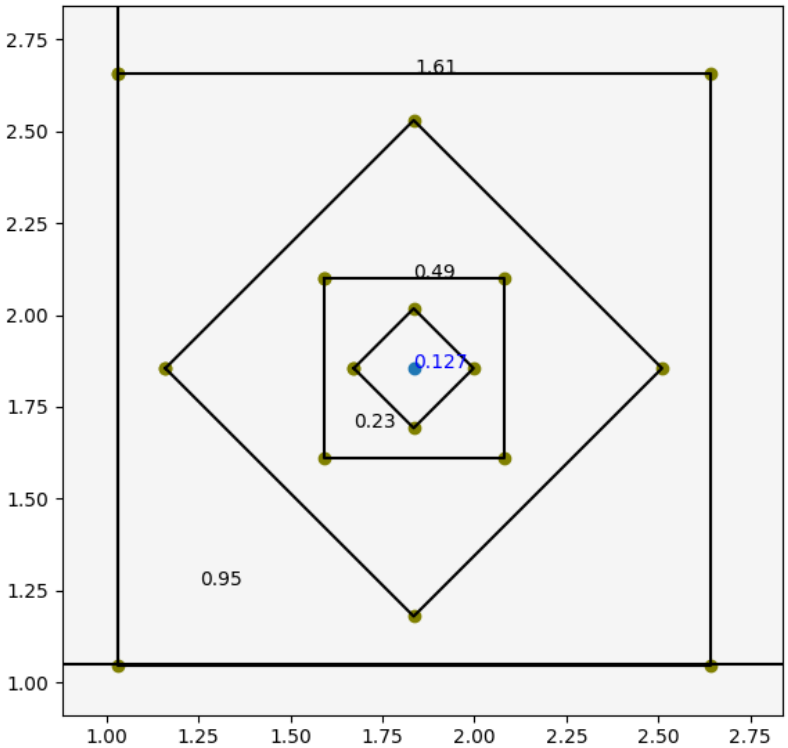
Σαν αρχικό κενό διάτρημα τοποθετείται διάτρημα διαμέτρου 0.127 m *(δύο διατρήματα των 0.089 m)* και διάμετρος κανονικού διατρήματος 0.045 m για τα υπόλοιπα διατρήματα *(hole diameter).*

Το σφάλμα κλίσης (παρέκκλιση) ισούται με 10 mm/m *(angular deviation, a),* ενώ το σφάλμα τοποθέτησης ή κολάρου λαμβάνει την τιμή 20 mm *(collar deviation, b).* Τέλος, η γωνιακή απόκλιση για τα περιμετρικά διατρήματα ανέρχεται στις 3 μοίρες ή αλλίως 0.05 rad *(lookout for contour holes, g).* Η προχώρηση εκτιμάται στο 95% του βάθους διατρήματος.

4.1.1. Τετράγωνα Προεκσκαφής

Σύμφωνα με τη μέθοδο του Holmberg και με τις παραμέτρους που τέθηκαν στο αρχείο config.json, τα πρακτικά φορτία που πηγάζουν για τα τετράγωνα προεκσκαφής είναι αντίστοιχα: V_1=0.16 m , V_2=0.23m ,V_3=0.43 m και V_4=0.66 m.

Στο πρώτο τετράγωνο, ο βέλτιστος σχεδιασμός απαιτεί τη χρήση 13.5 τεμαχίων του φυσιγγίου με διαστάσεις 28x255 mm . Η απόσταση διατρημάτων και συνεπώς η πλευρά του τετραγώνου είναι B_1=0.23 m. Στο δεύτερο τετράγωνο προεκσκαφής επιλέγονται 9 τεμάχια του φυσιγγίου με διαστάσεις 38x355 mm, με απόσταση διατρημάτων B_2=0.49 m. Στο τρίτο τετράγωνο προτείνεται η χρήση 9 τεμαχίων επίσης του φυσιγγίου με διαστάσεις 38x255 mm, με απόσταση διατρημάτων B_3=0.95 m. Το τελευταίο τετράγωνο προεκσκαφής έχει πλευρά μήκους B_4=1.61 m.Τα φυσίγγια που τοποθετούνται στα διατρήματα είναι του τύπου 38x255 mm και ο αριθμός τους είναι 9.

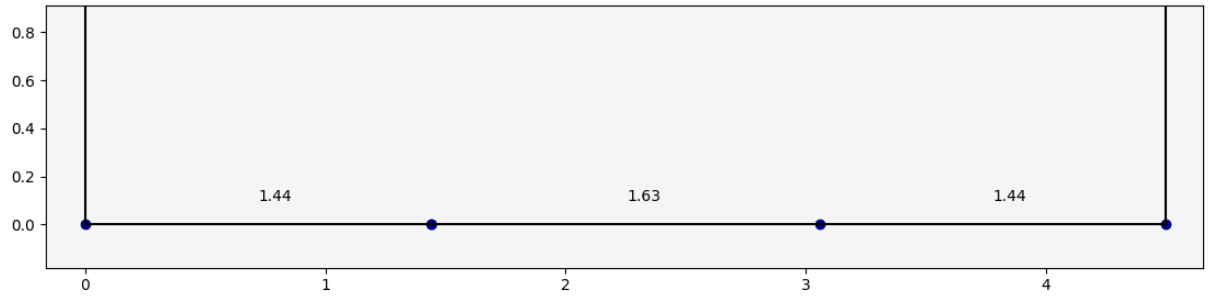


Σχήμα 4.1. Τετράγωνα προεκσκαφής

Στο παραπάνω *Σχήμα 4.1.* απεικονίζεται το αποτέλεσμα του κώδικα στη δημιουργία των τετραγώνων προεκσκαφής. Οι γραμμές σχεδιάστηκαν για να είναι πιο εμφανή τα τετράγωνα διατρημάτων που σχηματίζονται. Όπως είναι αντιληπτό τα διατρήματα είναι οι πράσινες κουκίδες στις γωνίες των τετραγώνων. Επίσης, απεικονίζονται οι αποστάσεις των διατρημάτων/πλευρών του κάθε τετραγώνου. Τέλος, στο εσωτερικό των τετραγώνων καταγράφεται η διάμετρος του αρχικού κενού διατρήματος.

4.1.2. Διατρήματα Δαπέδου

Με πρακτικό φορτίο διατρημάτων V_L=1.05 m και χρήση 5 φυσιγγίων με διαστάσεις 32x420 mm για τη γόμωση στήλης και 3.5 φυσιγγίων με διαστάσεις 38x385 mm για τη γόμωση πυθμένα, υπολογίστηκε να ορυχθούν 4 διατρήματα δαπέδου.

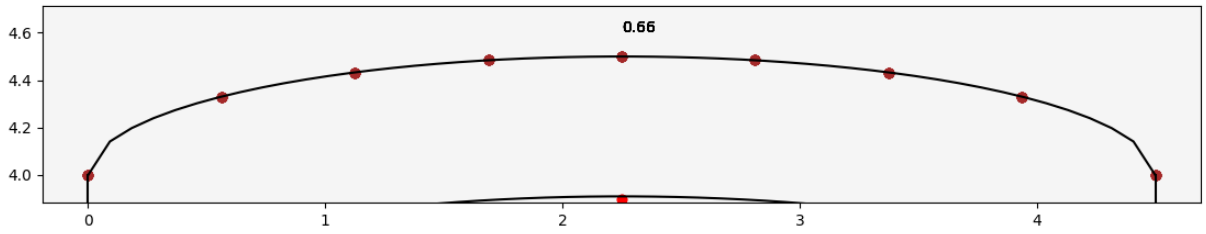


Σχήμα 4.2. Διάταξη Διατρημάτων Δαπέδου

Στο *Σχήμα 4.2.* απεικονίζεται ο σχεδιασμός των δαιτρημάτων δαπέδου κατά μήκος του πλάτους του μετώπου. Η απόσταση των γωνιακών διατρημάτων είναι της τάξης των 1.44 m, ενώ τα υπόλοιπα ενδιάμεσα διατρήματα απέχουν 1.63 m μεταξύ τους.

4.1.3. Διατρήματα Οροφής

Σειρά έχει ο υπολογισμός των διατρημάτων οροφής. Βάσει της μεθόδου του Holmberg και σε συνδυασμό με την τεχνική ελεγχόμενων ανατινάξεων με τις παραμέτρους που τέθηκαν, για τη βέλτιστη ανατίναξη της οροφής του μετώπου απαιτώνται 9 διατρήματα. Το πρακτικό φορτίο τους υπολογίστηκε σε VR=0.59 m και χρησιμοποιούνται 15 φυσίγγια με διαστάσεις 28x255 mm.

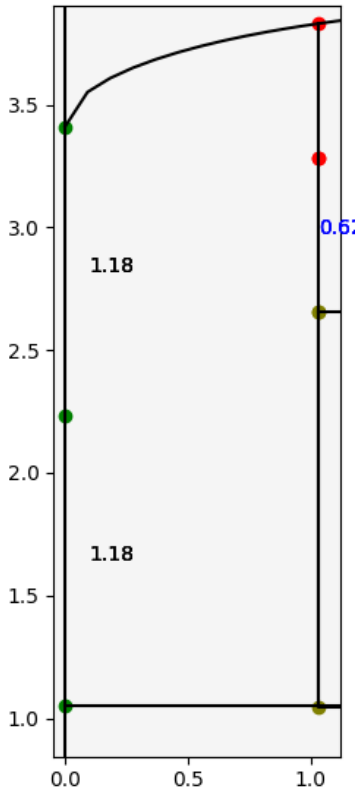


Σχήμα 4.3. Διάταξη Διατρημάτων Οροφής

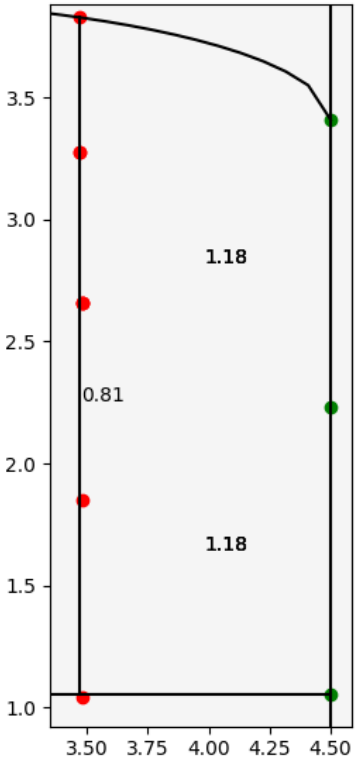
Στο παραπάνω *Σχήμα 4.3.* απεικονίζονται τα διατρήματα οροφής (σκούρο κόκκινο). Η απόσταση μεταξύ τους είναι 0.66 m, ενώ παρακάτω φαίνεται το όριο του πρακτικού φορτίου των διατρημάτων οροφής και ξεκινάνε τα διατρήματα διεύρυνσης του μετώπου.

4.1.4. Διατρήματα Τοίχου

Για τα διατρήματα του τοίχου, υπολογίστηκε πρακτικό φορτίο της τάξης του V_w=1.03 m. Προτείνεται βάσει της μεθόδου, η χρήση 5 φυσιγγίων του τύπου 32x420 mm για τη γόμωση στήλης και 3.5 φυσιγγίων 38x385 mm για τη γόμωση πυθμένα. Συνολικά ορύσσονται 3 διατρήματα στον κάθε τοίχο. Παρακάτω, *Σχήμα 4.4.* απεικονίζεται ο αριστερός τοίχος του μετώπου.



Σχήμα 4.4. Διάταξη διατρημάτων αριστερού τοίχου μετώπου.

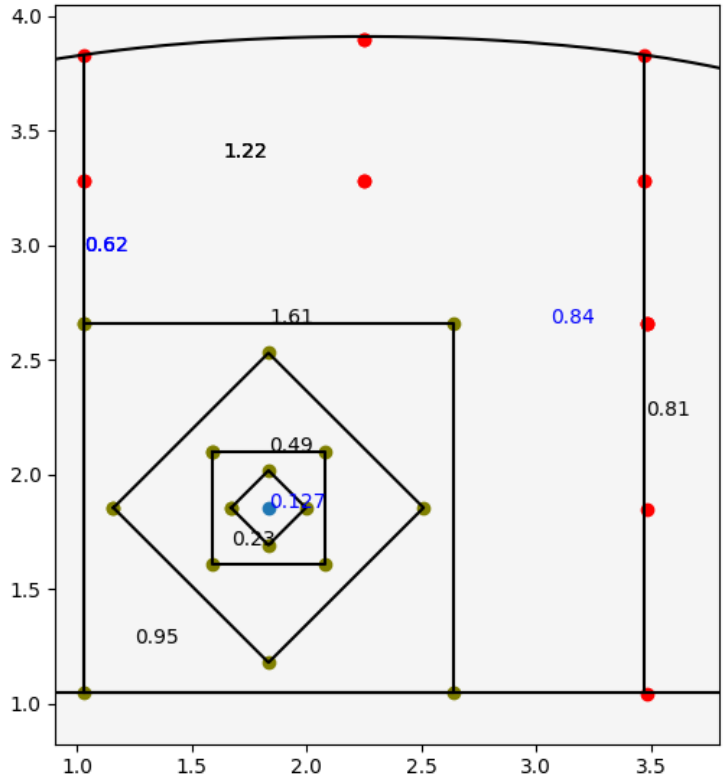


Σχήμα 4.5. Διάταξη διατρημάτων δεξιού τοίχου μετώπου

Έπειτα ο απέναντι τοίχος του μετώπου με τις ίδιες ακριβώς προδιαγραφές *(Σχήμα 4.5)*. Η απόσταση μεταξύ των διατρημάτων είναι 1.18 m και το όριο των διατρημάτων με τα διατρήματα διεύρυνσης είναι όσο το πρακτικό φορτίο, δηλαδή V_w=1.03 m.

4.1.5. Διατρήματα Διεύρυνσης

Βάσει της μεθόδου υπολογίστηκε η χρήση 9.5 φυσιγγίων διαστάσεων 28x255 mm για τη γόμωση στήλης των διατρημάτων και 2.5 φυσίγγια διαστάσεων 38x385 για τη γόμωση πυθμένα των διατρημάτων.



Σχήμα 4.6. Διατρήματα διεύρυνσης

Όπως φαίνεται στο παραπάνω σχήμα *(Σχήμα 4.6.)*, τα οριζόντια διατρήματα διεύρυνσης τοποθετούνται σε μία στήλη των 3 διατρημάτων κατά 0.84 m δεξιά από το τελευταίο τετράγωνο προεκσκαφής. Τα διατρήματα απέχουν 0.81 m μεταξύ τους. Άνωθεν της προεκσκαφής τοποθετούνται δύο σειρές των 3 διατρημάτων σε απόσταση 0.62 m (πρακτικό φορτίο) από την πλευρά του τετραγώνου και σε μεταξύ τους απόσταση 1.22 m για τον σχηματισμό ενός μεγάλου παραληλλογράμου με τα οριζόντια διατρήματα, για την ανατίναξη όλο του μετώπου.

4.2. Τελικά Αποτελέσματα Διπλωματικής Εργασίας

Με την εκτέλεση του κώδικα, ο χειριστής λαμβάνει τρία στοιχεία που αφορούν στον βέλτιστο σχεδιασμό ανατίναξης υπογείου μετώπου με τη μορφή στοάς. Αυτά είναι δύο συγκεντρωτικοί πίνακες διάτρησης και ένα διάγραμμα του μετώπου που απεικονίζει ακριβώς τη γεωμετρία της στοάς και τις θέσεις των διατρημάτων που υπολογίστηκαν. Ο πρώτος πίνακας αφορά στην επιλογή των κατάλληλων φυσιγγίων που θα χρησιμοποιηθούν σε κάθε τμήμα της διάτρησης και την ποσότητά τους (π.χ. Διατρήματα Δαπέδου, ή Διατρήματα Οροφής, κ.λ.π), τον αριθμό των διατρημάτων που πρέπει να ορυχθούν καθώς και τη συνολική μάζα εκρηκτικών που θα χρησιμοποιηθούν στο κάθε τμήμα.

Πίνακας 4.1. Συγκεντρωτικός πίνακας φυσιγγίων

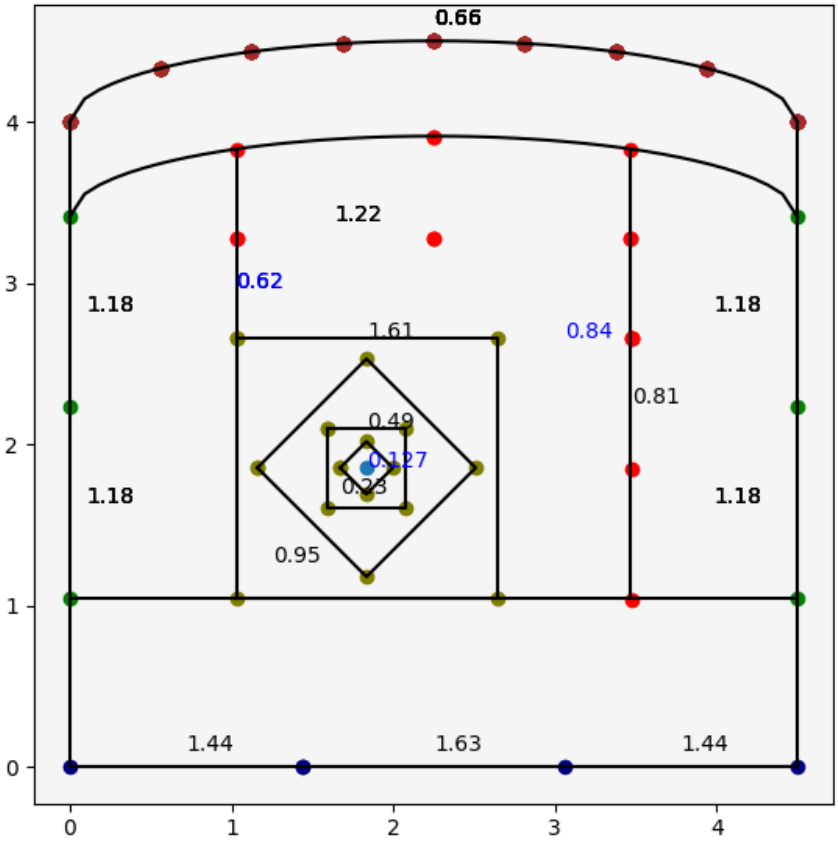
	('No. Holes',)	('Charge per Hole , kg',)	('Total , kg',)	((28.0, 'mm'),)	((32.0, 'mm'),)	((38.0, 'mm'),)
First Quadrangle	4	2.55	10.2	13.5	0	0
Second Quandrangle	4	4.71	18.84	0	0	9
Third Quandrangle	4	4.71	18.84	0	0	9
Fourth Quandrangle	4	4.71	18.84	0	0	9
Lifters	4	3.87	15.48	0	5	3.5
Roof	9	2.83	25.47	15	0	0
Wall	6	3.87	23.22	0	5	3.5
Stoping	9	3.1	27.9	9.5	0	2.5

Όπως είναι εμφανές από τον *Πίνακα 4.1.* , για τη διάτρηση των διατρημάτων του πρώτου τετραγώνου προεκσαφής απαιτώνται 4 διατρήματα, 13.5 φυσίγγια διαστάσεων 28x255 mm και η συνολική μάζα εκρηκτικής ύλης που θα διατεθεί είναι 10.2 kg. Στη συνέχεια εμφανίζεται ο *Πίνακας 4.2.* , με περαιτέρω στοιχεία του σχεδιασμού ανατίναξης του μετώπου. Η διάμετρος του διατρήματος του συγκεκριμένου κεντρικού παραδείγματος είναι 0.045 m.

Πίνακας 4.2. Συγκεντρωτικός πίνακας στοιχείων διάτρησης

Total charge weight (kg)	158.79
Total No. of holes	44
Advance (m)	3.66
Hole depth (m)	3.85
Cross sectional area (m2)	19.44
Specific charge (kg/m3)	2.23
Specific drilling (m/m3)	2.38

Στον παραπάνω *Πίνακα 4.2.* παρουσιάζονται τα υπόλοιπα στοιχεία του σχεδιασμού, όπως είναι η συνολική μάζα εκρηκτικών υλών, ο συνολικός αριθμός διατρημάτων, το εμβαδόν του μετώπου καθώς και οι τιμές της ειδικής κατανάλωσης εκρηκτικής ύλης και της ειδικής διάτρησης.



Σχήμα 4.7. Τελικό διάγραμμα διατρημάτων

Στο *Σχήμα 4.7.* παρουσιάζεται το τελικό διάγραμμα που απεικονίζει τη γεωμετρία της στοάς με όλα τα διατρήματα που υπολογίστηκαν και με τις κατάλληλες αποστάσεις μεταξύ τους.

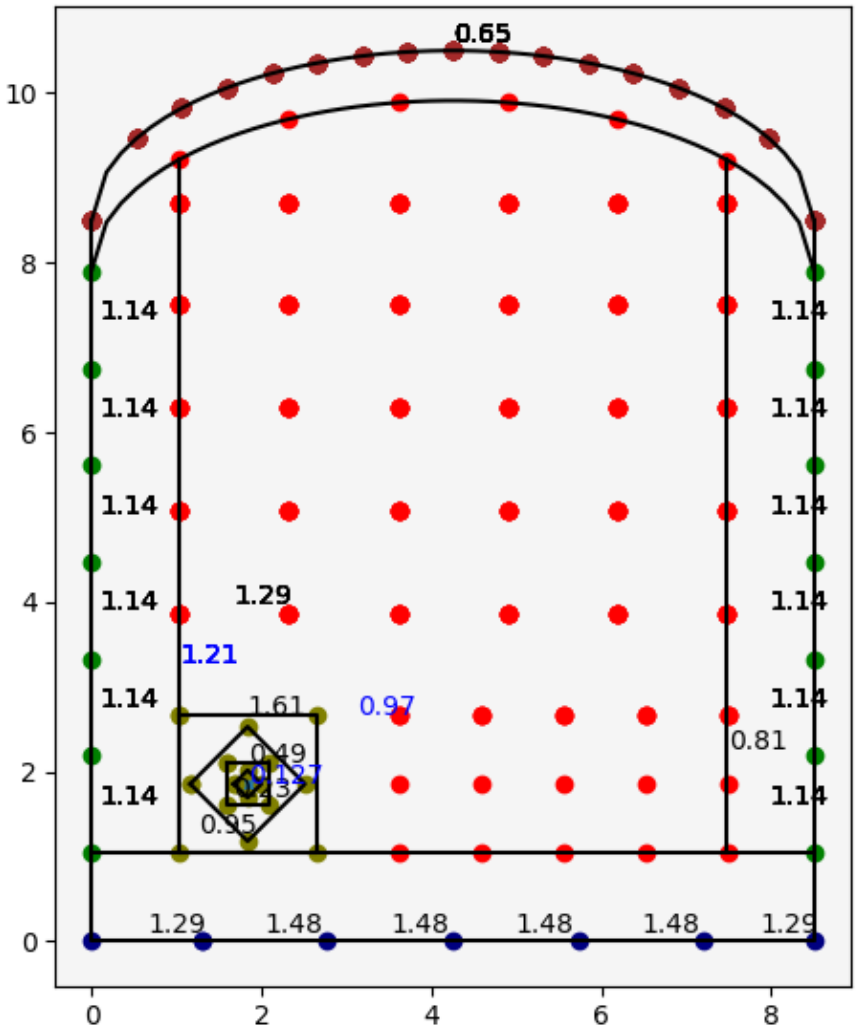
4.2.1. Σχεδιασμός διάτρησης στοάς προσπέλασης (έλεγχος ορθότητας του αλγορίθμου)

Ο έλεγχος της ορθότητας του αλγορίθμου πραγματοποιήθηκε με συνεχόμενες δοκιμές του σε στοές διαφορετικών γεωμετρικών χαρακτηριστικών.

4.2.1.1. 1η Δοκιμή – Στοά Προσπέλασης

Θα εφαρμοστεί βέλτιστος σχεδιασμός διάτρησης-ανατίναξης σε στοά προσπέλασης διατομής 8.5 m x 8.5 m όπου 8.5 m το ύψος της στοάς και 8.5 μέτρα το πλάτος της. Αρχικό κενό διάτρημα σαν ελέθευρη επιφάνεια τοποθετείται διάτρημα διαμέτρου 0.127 m (δύο διατρήματα των 0,089 m) και διάμετρος κανονικού διατρήματος 0.045 m για τα υπόλοιπα. Η

οροφή αποτελεί τμήμα κυκλικού δίσκου με ύψος αψίδας 2 m *(height of arch)*. Σφάλμα κλίσης (παρέκκλιση) ισούται με 10 mm/m *(angular deviation, a)*, ενώ το σφάλμα τοποθέτησης ή κολάρου λαμβάνει την τιμή 20 mm *(collar deviation, b)*. Τέλος, η γωνιακή απόκλιση για τα περιμετρικά διατρήματα ανέρχεται στις 3 μοίρες ή αλλίως 0.05 rad *(lookout for contour holes, g)*. Η εκρηκτική ύλη παραμένει ίδια καθώς και οι διαθέσιμοι τύποι φυσιγγίων.



Σχήμα 4.8. Τελικό διάγραμμα διατρημάτων 1ης δοκιμής

Στο παραπάνω *Σχήμα 4.8.*, παρουσιάζεται το τελικό διάγραμμα διατρημάτων της στοάς προσπέλασης με διατομή 8.5 m x 8.5 m. Οι αριθμοί με το μαύρο χρώμα αποτελούν τις αποστάσεις μεταξύ των διατρημάτων, ενώ στα διατρήματα διεύρυνσης απεικονίζεται και το πρακτικό φορτίο τους (μπλε απόχρωση).

Οι *Πίνακας 4.3* και *Πίνακας 4.4.* παρουσιάζουν τα υπόλοιπα χαρακτηριστικά του σχεδιασμού διάτρησης-ανατίναξης της δοκιμαστικής στοάς.

Πίνακας 4.3. Συγκεντρωτικός πίνακας φυσιγγίων 1ης δοκιμής

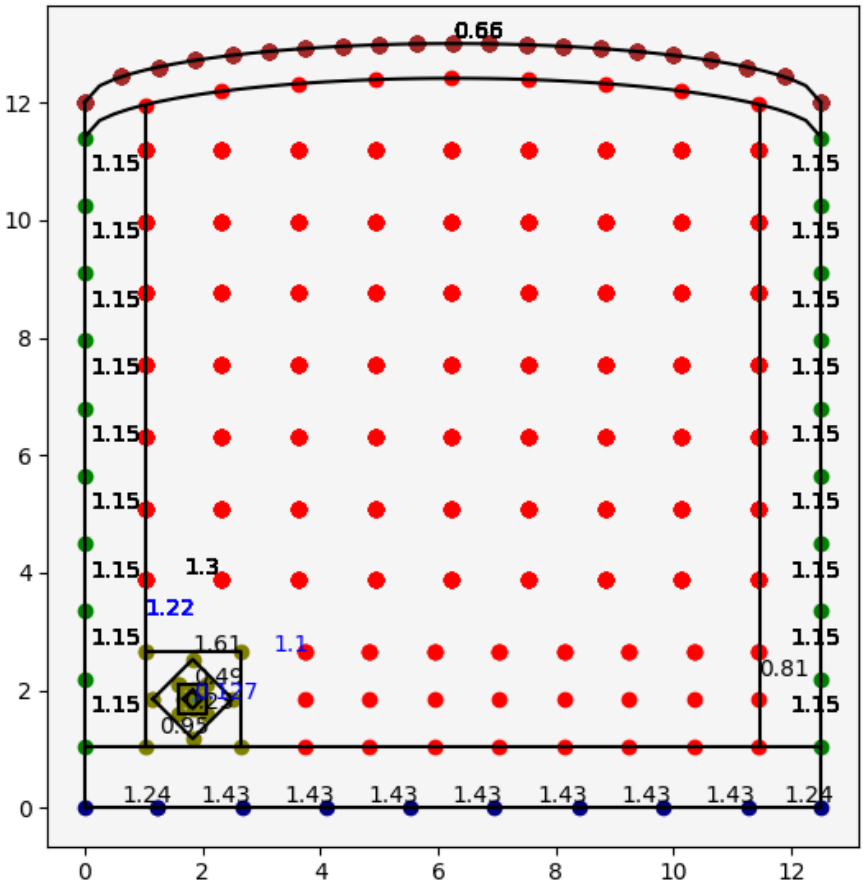
	('No. Holes',)	('Charge per Hole , kg',)	('Total , kg',)	((28.0, 'mm'),)	((32.0, 'mm'),)	((38.0, 'mm'),)
First Quadrangle	4	2.55	10.2	13.5	0	0
Second Quandrangle	4	4.71	18.84	0	0	9
Third Quandrangle	4	4.71	18.84	0	0	9
Fourth Quandrangle	4	4.71	18.84	0	0	9
Lifters	7	3.87	27.09	0	5	3.5
Roof	17	2.83	48.11	15	0	0
Wall	14	3.87	54.18	0	5	3.5
Stoping	51	3.1	158.1	9.5	0	2.5

Πίνακας 4.4. Συγκεντρωτικός πίνακας στοιχείων διάτρησης 1ης δοκιμής

Total charge weight (kg)	354.2
Total No. of holes	105
Advance (m)	3.66
Hole depth (m)	3.85
Cross sectional area (m2)	84.1
Specific charge (kg/m3)	1.15
Specific drilling (m/m3)	1.31

4.2.1.2. 2η Δοκιμή – Στοά Προσπέλασης

Στη δεύτερη δοκιμή, εφαρμόζεται βέλτιστος σχεδιασμός διάτρησης-ανατίναξης με τη μέθοδο Holmberg σε άλλη μία στοά προσπέλασης διατομής 12.5 m (πλάτος) x 12 m (ύψος), με ύψος αψίδας 1 m και διάμετρο διατρημάτων 0.045 m. Αυτές οι δοκιμές γίνονται για να ελεγχθεί η λειτουργία του αλογρίθμου σε στοές ή σήραγγες μεγαλύτερης διατομής.



Σχήμα 4.9. Τελικό διάγραμμα διατρημάτων 2ης δοκιμής

Πίνακας 4.5. Συγκεντρωτικός πίνακας φυσιγγίων 2ης δοκιμής

	('No. Holes',)	('Charge per Hole , kg',)	('Total , kg',)	((28.0, 'mm'),)	((32.0, 'mm'),)	((38.0, 'mm'),)
First Quadrangle	4	2.55	10.2	13.5	0	0
Second Quandrangle	4	4.71	18.84	0	0	9
Third Quandrangle	4	4.71	18.84	0	0	9
Fourth Quandrangle	4	4.71	18.84	0	0	9
Lifters	10	3.87	38.7	0	5	3.5
Roof	21	2.83	59.43	15	0	0
Wall	20	3.87	77.4	0	5	3.5
Stoping	96	3.1	297.6	9.5	0	2.5

Πίνακας 4.6. Συγκεντρωτικός πίνακας στοιχείων διάτρησης 2ης δοκιμής

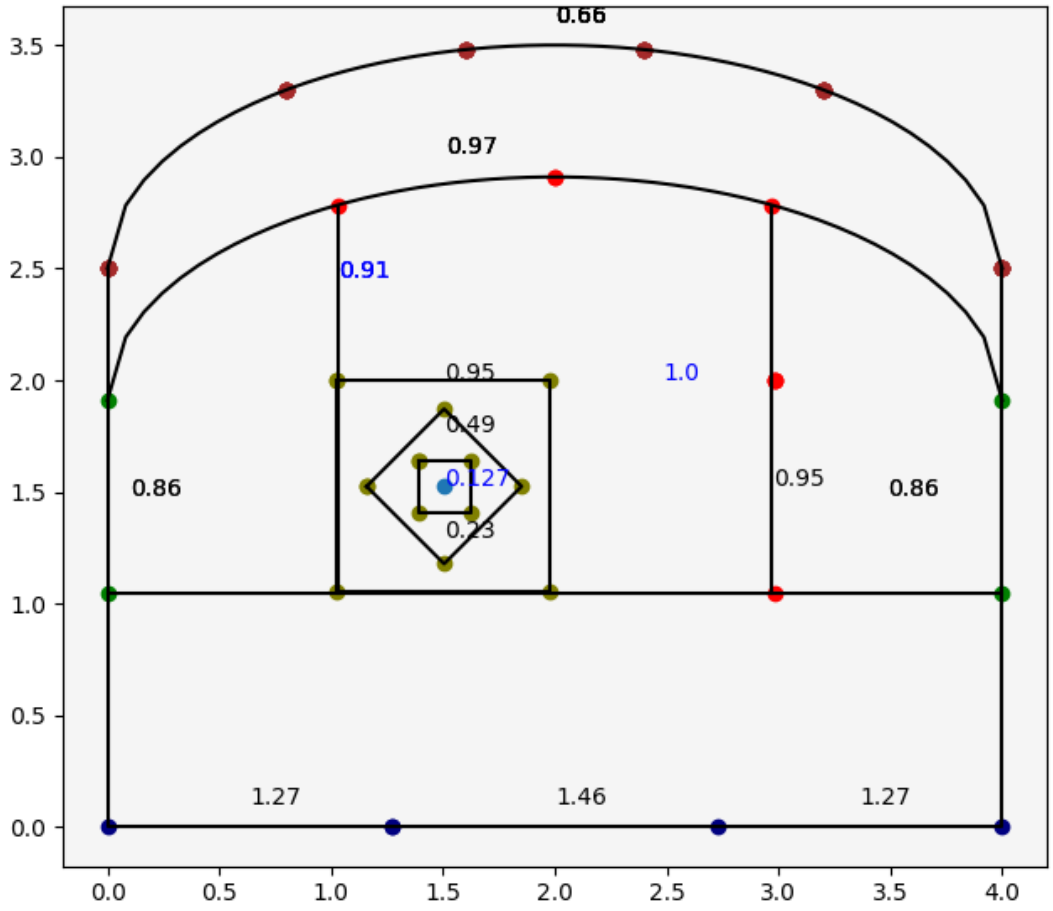
Total charge weight (kg)	539.85
Total No. of holes	163
Advance (m)	3.66
Hole depth (m)	3.85
Cross sectional area (m2)	157.44
Specific charge (kg/m3)	0.94
Specific drilling (m/m3)	1.09

Με βάση τα παραπάνω αποτελέσματα, επαληθεύεται εν μέρει η σωστή λειτουργία του αλγορίθμου. Οι τιμές των στοιχείων διάτρησης της πρώτης και δεύτερης δοκιμής είναι λογικά υπολογισμένες και τα διατρήματα σωστά κατανεμημένα στο μέτωπο της στοάς.

4.2.2. Σχεδιασμός διάτρησης διευθυντικής στοάς

Ένα ακόμη παράδειγμα που επιβεβαιώνει την ορθότητα και τη σωστή λειτουργία του αλγορίθμου είναι ο σχεδιασμός διάτρησης-ανατίναξης μιας διευθυντικής στοάς με τη μέθοδο του Holmberg.

Στην περίπτωση αυτή, η στοά που θα ορυχθεί θα έχει διατομή 4 m x 2.5 m με ύψος αψίδας 1 m. Τα υπόλοιπα γεωμετρικά χαρακτηριστικά του μετώπου παραμένουν ίδια όπως και οι διαθέσιμοι τύποι φυσιγγίων εκρηκτικού γαλακτώματος. Υπό αυτές τις συνθήκες, ο κώδικας παρουσιάζει τα παρακάτω αποτελέσματα:



Σχήμα 4.10. Τελικό διάγραμμα διατρημάτων διευθυντικής στοάς

Στο παραπάνω *Σχήμα 4.10.* απεικονίζονται πλήρως τα διατρήματα που θα ορυχθούν για τη διάτρηση μιας διευθυντική στοάς. Είναι τοποθετημένα έτσι ώστε να ικανοποιούν τη σχέση E/V (απόσταση διατρημάτων – πρακτικό φορτίου) σύμφωνα με τη μέθοδο του Holmberg και παράλληλα να μην αφήνουν περιθώριο για ατελή θρυμματισμό.

Πίνακας 4.7. Συγκεντρωτικός πίνακας φυσιγγίων διευθυντικής στοάς

	('No. Holes',)	('Charge per Hole , kg',)	('Total , kg',)	((28.0, 'mm'),)	((32.0, 'mm'),)	((38.0, 'mm'),)
First Quadrangle	4	2.55	10.2	13.5	0	0
Second Quandrangle	4	4.71	18.84	0	0	9
Third Quandrangle	4	4.71	18.84	0	0	9
Fourth Quandrangle	0	0	0	0	0	0
Lifters	4	3.87	15.48	0	5	3.5
Roof	6	2.83	16.98	15	0	0
Wall	4	3.87	15.48	0	5	3.5
Stoping	5	3.17	15.85	8.5	0	3

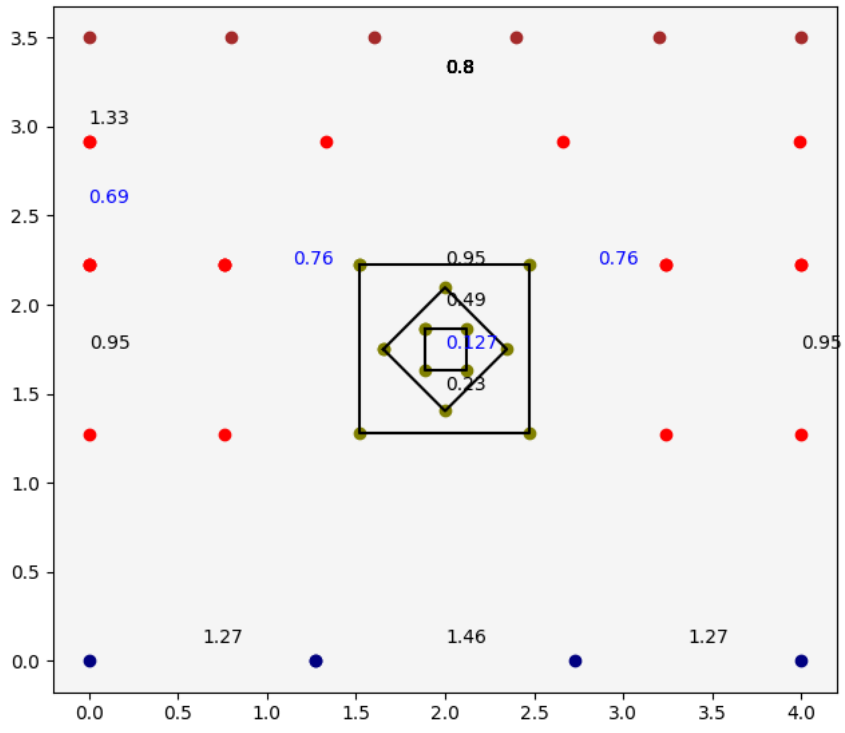
Πίνακας 4.8. Συγκεντρωτικός πίνακας στοιχείων διάτρησης διευθυντικής στοάς

Total charge weight (kg)	111.67
Total No. of holes	31
Advance (m)	3.66
Hole depth (m)	3.85
Cross sectional area (m2)	12.78
Specific charge (kg/m3)	2.39
Specific drilling (m/m3)	2.55

4.2.3. Σχεδιασμός διάτρησης τετραγωνικής διατομής στοάς παραγωγής

Στο τελευταίο παράδειγμα ορθότητας αλγορίθμου παρατίθεται ο σχεδιασμός διάτρησης σε στοά τετραγωνικής διατομής 4 m x 3.5 m με του ίδιους τύπους φυσιγγίων εκρηκτικού γαλακτώματος. Η διατομή του αρχικού κενού διατρήματος είναι παρόμοια με τις προηγούμενες *(φ=0.127 m)*.

Αρχικά σχεδιάζονται τα διατρήματα προεκσκαφής στο κέντρο του μετώπου και έπειτα τα υπόλοιπα διατρήματα διεύρυνσης και οροφής με τη μέθοδο των ελεγχόμενων ανατινάξεων (smooth blasting) καθώς και τα διατρήματα δαπέδου.



Σχήμα 4.11. Διάταξη διατρημάτων σε τετραγωνική στοά παραγωγής με παράλληλη διάταξη

Πίνακας 4.9. Συγκεντρωτικός πίνακας φυσιγγίων τετραγωνικής στοάς παραγωγής

	('No. Holes',)	('Charge per Hole , kg',)	('Total , kg',)	((28.0, 'mm'),)	((32.0, 'mm'),)	((38.0, 'mm'),)
First Quadrangle	4	2.55	10.2	13.5	0	0
Second Quandrangle	4	4.71	18.84	0	0	9
Third Quandrangle	4	4.71	18.84	0	0	9
Fourth Quandrangle	0	0	0	0	0	0
Lifters	4	3.87	15.48	0	5	3.5
Roof	6	2.83	16.98	15	0	0
Stoping	12	3.51	42.12	7.5	0	4

Πίνακας 4.10. Συγκεντρωτικός πίνακας στοιχείων διάτρησης τετραγωνικής στοάς παραγωγής

Total charge weight (kg)	122.46
Total No. of holes	34
Advance (m)	3.66
Hole depth (m)	3.85
Cross sectional area (m2)	14
Specific charge (kg/m3)	2.39
Specific drilling (m/m3)	2.55

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο κεφάλαιο αυτό γίνεται σχολιασμός των αποτελεσμάτων της Διπλωματικής Εργασίας και παραθέτονται τρόποι βελτίωσης της.

5.1. Σχολιασμός Αποτελεσμάτων

Με μία πρώτη ανάγνωση των αποτελεσμάτων του κώδικα, βάσει των υπολογισμών της μεθόδου του Holmberg, για το βέλτιστο σχεδιασμό διάτρησης-ανατίναξης μετώπου των συγκεκριμένων γεωμετρικών χαρακτηριστικών του κεντρικού παραδείγματος ήτοι στοάς 4.5 m x 4.5 m απαιτείται η όρυξη 44 διατρημάτων. Η προχώρηση του μετώπου κατά την ανατίναξη θα είναι της τάξης των 3.66 m και το μήκος των διατρημάτων στα 3.85 m.

Τα δεδομένα του πρώτου συγκεντρωτικού πίνακα *(Πίνακας 4.1.)* αναγράφουν αναλυτικά τον τύπο των φυσιγγίων που απαιτείται για τον κάθε τομέα της διάτρησης, τον αριθμό τους, τον αριθμό των διατρημάτων και τη συνολική μάζα εκρηκτικής ύλης που θα τοποθετηθεί. Για τα διατρήματα της οροφής, λόγω της τεχνικής ελεγχόμενης ανατινάξης, επιλέγεται η χρήση φυσιγγίων με τις μικρότερες διαστάσεις 28x255 mm. Αυτό είναι απολύτως λογικό για τη διαφύλαξη των ορίων της στοάς μιας και αποτελούν περιμετρικά διατρήματα. Έτσι λοιπόν η γόμωση ανά διάτρημα στην οροφή (2.83 kg/hole) είναι μικρότερη από τη γόμωση ανά διάτρημα στους υπόλοιπους τομείς της διάτρησης. Στο δεύτερο συγκεντρωτικό πίνακα *(Πίνακας 4.2.)* αναγράφονται τα γενικά στοιχεία της διαδικασίας διάτρησης του μετώπου. Η συνολική μάζα εκρηκτικού γαλακτώματος που θα χρησιμοποιηθεί είναι 158.79 kg. Το εμβαδόν της συγκεκριμένης στοάς ανέρχεται στα 19.44 m^2. Τέλος η ειδική κατανάλωση εκρηκτικής ύλης θα είναι 2.23 kg/m^3 και η ειδική διάτρηση 2.38 m/m^3.

Το τελικό διάγραμμα *(Σχήμα 4.7.)* αποτελεί μια σύνοψη των γεωμετρικών χαρακτηριστικών της στοάς σε συνδυασμό με τον αριθμό, τις θέσεις και τις αποστάσεις μεταξύ των διατρημάτων που υπολογίστηκαν. Τα αποτελέσματα του κώδικα αντικατοπτρίζουν πλήρως τη μεθοδολογία και τις παραμέτρους της μεθόδου του Holmberg ως προς το σχεδιασμό της διάτρησης και ανατίναξης του μετώπου. Όλες οι τιμές είναι λογικές και η απεικόνισή τους στο τελικό διάγραμμα διατρημάτων μαρτυρά τη σωστή λειτουργία του κώδικα και την ακρίβεια στους υπολογισμούς. Τα διατρήματα είναι τοποθετημένα στις κατάλληλες αποστάσεις μεταξύ τους, τόσο για να ακολουθείται η σχέση απόστασης-φορτίου όσο και για να μην «αφήνουν» περιθώρια μη θρυμματισμού ορισμένων περιοχών.

5.2. Προτάσεις Βελτίωσης

Ο συγκεκριμένος κώδικας προσφέρει στον μηχανικό τη δυνατότητα της άμεσης πρόσβασης στα ακριβή αποτελέσματα της μεθόδου R. Holmberg (1975) για το σχεδιασμό διάτρησης-ανατίναξης υπογείου μετώπου με τη μορφή στοάς. Παρ'όλα αυτά, θα μπορούσαν σαφώς να γίνουν βελτιώσεις στο συγκεκριμένο πρόγραμμα ώστε να αυξηθούν οι δυνατότητές του.

Με τους κατάλληλους υπολογισμούς των χρόνων επιβράδυνσης που απαιτούνται για την ανατίναξη των διατρημάτων, ο παραπάνω κώδικας θα μπορούσε να προσφέρει μια ακριβή

προσομοίωση της ανατίναξης του μετώπου ανάλογα με τις ελεύθερες επιφάνειες που σχηματίζονται.

Μία άλλη χρησιμότητα του προγράμματος θα ήταν η απεικόνιση των οικονομικών δεδομένων της διάτρησης-ανατίναξης του μετώπου με τη μέθοδο R. Holmberg (1975). Με την καταγραφή του κόστους των εκρηκτικών υλών καθώς και το κόστος χρήσης του μηχανοποιημένου εξοπλισμού για τη διάτρηση των διατρημάτων, το πρόγραμμα θα προσέφερε στον χειριστή τη δυνατότητα ελέγχου του συνολικού κόστους διάτρησης-ανατίναξης του υπογείου μετώπου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Διεθνής Βιβλιογραφία

- Holmberg, R., 1975, "Computer Calculations of Drilling Patterns for Surface and Underground Blastings", *Design Methods in Rock Mechanics*, C. Fairhurst and S. Crouch, eds., 16th Symposium on Rock Mechanics, University of Minnesota, Minneapolis.

- Holmberg, R., and Persson, P.-A., 1979, "Design of Tunnel Perimeter Blasthole Patterns to Prevent Rock Damage," Proceedings, Tunelling '79, M.J. Jones, ed., Institution of Mining and Metallurgy, London.

- Gustafsson, R., 1973, Swedish Blasting Technique, Gothenburg, Sweden.

- Holmberg, R., and Hustrulid, W., 1981, "Swedish Cautious Blast Excavations at the CSM/ONWI Test Site in Colorado," 7th Conference of Explosives and Blasting Technique, Phoenix.

- (Langefors, U., and Kihlstrom, B., 1963, *The Modern Technique of Rock Blasting*, Almqvist and Wiskell, Stockholm)

- Persson, P.A., 1973, "The Influence of Blasting on the Remaining Rock," Report DS 1973:15, Swedish Detonic Research Foundation.

Ελληνική Βιβλιογραφία

- Στοιχεία Διάτρησης και Ανατίναξης, Αγιουτάντης Ζαχαρίας, Πολυτεχνείο Κρήτης

- Σημειώσεις Μαθήματος «Διάτρηση, Ανατίναξη και Εισαγωγή στα Υπόγεια Έργα», Εξαδάκτυλος Γεώργιος, Λιόλιος Παντελής, Πολυτεχνείο Κρήτης

Ηλεκτρονική Βιβλιογραφία

- https://www.extraco.gr/index.php
- https://www.python.org
- https://www.codecademy.com/
- https://www.pcsteps.gr
- http://users.tem.uoc.gr/~komineas/python-course/Lectures/01_lecture.html
- https://cyberpython.github.io/byte-of-python/introduction.html

Υπόμνημα Κώδικα

Λεξικό config.json

```
{
            "name": "em-ex",
            "cartridges":[
                        {"name": "cart1",
                              "diameter" : 0.028,
                              "length": 0.255
                              },
                              { "name": "cart2",
                              "diameter": 0.032,
                              "length": 0.420
                              },
                              { "name": "cart3",
                              "diameter": 0.038,
                              "length": 0.385
                              }
                              ],
                  "geometry":{
                                    "hdiam": 0.045,
                                    "emptyf":0.127,
                                    "width":4.5,
                                    "abutmentH":4,
                                    "archHeight":0.5,
                                    "g":0.05,
                                    "a":0.01,
                                    "b":0.02,
                                    "Q":4.03,
                                    "V":0.92,
                                    "Density":1200,
                                    "c":0.4
                              }


}
```

Κώδικας Code_Blasting.py

```
class Blaster():
    import json

                                #Οι τιμές self είναι αυτές που θα πάρει απο το αρχείο .json
    def __init__(self):
        self.name = " "         # Όνομα εκρηκτικού
        self.Density = 0         # Πυκνότητα εκρηκτικής ύλης
        self.Q = 0              # Ενέργεια εκρητικού
        self.V = 0               # Όγκος αερίων κατα την ανατίναξη του εκρηκτικού
        self.hdiam=0            # Διάμετρος διατρήματος
        self.emptyf =0           # Διάμετρος κενού διατρήματος
        self.width =0           # Πλάτος μετώπου
        self.abutmentH =0        # Ύψος μετώπου
        self.archHeight =0      # Ύψος κυκλικού τμήματος οροφής
        self.a =0                # Γωνιακή απόκλιση
        self.b =0               # Απόκλιση κολάρου
        self.c =0                # Σταθερά πετρώματος
        self.g=0                # Γωνιακή απόκλιση για περιμετρικά διατρήματα
        self.Sanfo=0             # Γραμμική πυκνότητα εκρηκτικού ως προς ANFO
        self.hdepth=0           # Βάθος διατρήματος
        self.advance=0           # Προχώρηση μετώπου
        self.massph=[]          # Λίστα μάζας γόμωσης εκρηκτικού ανά διάτρημα
        self.total=[]            # Συνολική μάζας εκρηκτικών
        self.charge=[]          # Λίστα γραμμικής πυκνότητας γόμωσης φυσιγγίων
        self.cartridges = []     # Λίστα φυσιγγίων εκρηκτικής ύλης
        self.B=[]               # Λίστα πλευρών τετραγώνων της προεκσκαφής
        self.practicalb=[]       # Λίστα πρακτικών φορτίων
        self.diam=[]            # Λίστα διαμέτρων φυσιγγίων που χρησιμοποιείται σαν τίτλος
διατρημάτων στον τελικό πίνακα
        self.holes=[]            # Λίστα αριθμού διατρημάτων για κάθε τμήμα
        self.number_first=[]
        self.number_second=[]           # Λίστες φυσιγγίων στα διατρήματα του κάθε
τμήματος. Χρησιμοποιείται για τον τελικό συγκεντρωτικό πίνακα.
        self.number_third=[]            # π.χ. [4,4.5,0] ποσότητα κάθε φυσιγγίου στα
διατρήματα
        self.number_fourth=[]
        self.number_lifters=[]
        self.number_roof=[]
        self.number_wall=[]
        self.number_stoping=[]
        self.spacing=[]                # Λίστα με τις αποστάσεις των εκρηκτικών δαπέδου και
τοίχου
        self.No_columns=[]            # Ο αριθμός των στηλών διατρημάτων που πρέπει να
δημιουργηθούν στα stoping holes
        self.No_rows=[]                # Ο αριθμός των σειρών διατρημάτων που πρέπει να
δημιουργηθούν στα stoping holes
        self.spacing_columns=[]       # Η απόσταση μεταξύ των στηλών οριζοντίων διατρημάτων
        self.spacing_rows=[]           # Η απόσταση μεταξύ των οριζοντίων διατρημάτων σε
κάθε σειρά
        self.corner_holes=[]          # Η απόσταση των γωνιακών διατρημάτων από τα επόμενα,
στα Lifters
        self.Spacing=[]                # Απόσταση στηλών καθέτων διατρημάτων
        self.cols_downwards=[]        # Αριθμός στηλών καθέτων διατρημάτων
        self.number_stoping1=[]
        self.number_roof1=[]
        self.number_lifters1=[]
        self.test_no=[]
        self.test_total=[]            # Λίστες για τετραγωνική στοά 3.5 m x 4 m !!! ΟΧΙ
ΥΠΟΧΡΕΩΤΙΚΟ
        self.test_mass=[]
```

```
    def addcartridge(self, name, diameter, length):
        self.cartridges.append({"name":name, "diameter":diameter,"length":length })

        # Για κάθε φυσίγγιο εκρηκτικού που δηλώνω στο αρχείο .json, καταχωρούνται οι τιμές
του (όνομα, διάμετρος, μήκος) στη λίστα self.cartridges.

    def charge_concentration(self):
        for x in self.cartridges:
            d=x["diameter"]
            l=round(((math.pi*(d**2))/4)*self.Density,2)
            self.charge.append(l)

        # Υπολογισμός γραμμικής πυκνότητας γόμωσης I για κάθε φυσίγγιο που έχει καταχωρηθεί
και τοποθέτηση τους στη λίστα self.charge . [((π*d^2)*ρ)/4].

    def calculation(self):
        Semex=round(((5*self.Q)/(6*5))+((1*self.V)/(6*0.85)),2)
        self.Sanfo=round(Semex/0.84,2)
        self.hdepth=round(0.15+(34.1*self.emptyf)-(39.4*(self.emptyf**2)),2)
        self.advance=round(0.95*self.hdepth,2)
        for i in range(len(self.cartridges)):
            a=self.cartridges[i]['diameter']*1000,"mm"
            x=0
            self.diam.append(a)
            self.number_first.append(x)
            self.number_second.append(x)
            self.number_third.append(x)
            self.number_fourth.append(x)
            self.number_lifters.append(x)
            self.number_roof.append(x)
            self.number_wall.append(x)
            self.number_stoping.append(x)
            self.number_stoping1.append(x)
            self.number_roof1.append(x)
            self.number_lifters1.append(x)
        # Υπολογισμός σχετικής ισχύος εκρηκτικής ύλης ως προς LBF και μετά (/0.84) ως προς
ANFO.
        # Ο υπολογισμός αυτός γίνεται με βάση το Q και το V του εκρηκτικού που χρησιμοποιώ.

        # Οι παρακάτω λίστες θα χρησιμοποιηθούν για την καταγραφή των φυσιγγίων που θα
χρησιμοποιηθούν στα διατρήμματα.
        # Για παράδειγμα , αν στα stoping holes έχω 4 φυσίγια "α", 0 φυσίγγια "β" και 3
φυσίγγια "γ", η λίστα θα γράφει [4,0,3].
        # Δηλώνω αρχική τιμή της λίστας το 0 (x), για να ξεκινάνε με [0,0,0] και να
μεταβάλλονται παρακάτω ανάλογα με τα διατρήματα.

    def first_quadrangle(self):
        V1max=round(1.7*self.emptyf,2)
# Μέγιστο φορτίο
        #if (((self.a*self.hdepth)+self.b)*10)<1:                                    #
Έλεχγος για το άν η απόκλιση είναι κάτω απο 1% για τον υπολογισμό του πρακτικού φορτίου .
           #V1=round(1.5*self.emptyf,2)
        #else:
        V1=round((V1max-((self.a*self.hdepth)+self.b)),2) #a,b (m)
        charge=round((((55*self.hdiam)*((round(V1max,2)/self.emptyf)**1.5)*(round(V1max,2)-
(self.emptyf/2))*(self.c/0.4))/self.Sanfo),2) # γραμμική πυκνότητα γόμωσης ανάλογα του
φορτίου
        minimum=1000
        th=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge)<minimum and (self.charge[i]<=charge):         #
Επιλογή του φυσιγγίου του οποίου η γραμμ. πυκνότητα γόμωσης είναι η πλησιέστερη (προς τα
κάτω) με αυτή που υπολογίστηκε
                minimum=abs(self.charge[i]-charge)
                th=i
```

```
        h=10*self.hdiam                                                                 #
Επιγόμωση (10d)
        B1=round(math.sqrt(2)*V1,2)                                                     #
Πλευρά τετραγώνου
        No_first=round((self.hdepth-h)/self.cartridges[th]["length"]*2)/2
#Αριθμός φυσιγγίων
        mass1=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_first)),2)
#Μάζα γόμωσης ανά διάτρημα
        self.massph.append(mass1)
#Λίστα μάζας γόμωσης/διάτρημα
        self.B.append(B1)                                                               #
Λίστα Πλευράς τετραγώνου
        self.practicalb.append(V1)                                                      #
Εισαγωγή πρακτικού φορτίου στην λίστα
        Holes_first=4                                                                   #
Αριθμός διατρημάτων
        self.holes.append(Holes_first)
        self.test_no.append(Holes_first)
        self.test_mass.append(mass1)
        # Λίστα αριθμών διατρημάτων
        for i in range(len(self.number_first)):
            self.number_first[th]=No_first                                              #
Για κάθε τύπο φυσιγγίου , τοποθετείται στη λίστα ο αριθμός που θα χρησιμοποιηθεί πχ.[4,0,4]

        # Υπολογισμός αρχικά του πρακτικού φορτίου ανάλογα με την απόκλιση της διάτρησης
και ανάλογα της διαμέτρου του κενού αρχικού διατρήματος

        # Έπειτα υπολογίζω την γραμμική πυκνότητα γόμωσης του μέγιστου φορτίου και επιλέγει
το φυσίγγιο με την πλησιέστερη πυκνότητα γόμωσης

        # Υπολογίζω επιγόμωση, πλευρά τετραγώνου, αριθμό φυσιγγίων και διατρημάτων, μάζα
ανά διάτρημα και τα βάζω στις αντίστοιχες λίστες για να τα χρησιμοποιήσω αργότερα


    def second_quadrangle(self):
        B2temp=round((math.sqrt(2))*(self.practicalb[0]-((self.a*self.hdepth)+self.b)),2)
# Θεωρητική πλευρά τετραγώνου
        practicalb=[]
        minimum=1000
        th=0
        for i in range(len(self.charge)):

V2max=round((8.8*0.01)*math.sqrt((B2temp*self.charge[i]*self.Sanfo)/(self.hdiam*self.c)),2)
                Vpractical=round(V2max-((self.a*self.hdepth)+self.b),2)
# Με βάση την θεωρητική πλευρά τετραγώνου υπολογίζω το μέγιστο φορτίο και το πρακτικό
φορτίο για κάθε τύπο φυσιγγίου
                practicalb.append(Vpractical)
# Άν το πρακτικό φορτίο είναι <=2B και >0.5B ,επιλέγει το αντίστοιχο φυσίγγιο
                if Vpractical<(2*B2temp) and  abs(Vpractical-2*B2temp)<minimum and
Vpractical>0.5*B2temp:
                        minimum=abs(Vpractical-2*B2temp)
                        th=i
        V2=practicalb[th]
# Καταχώρηση του πρακτικού φορτίου ανάλογα στη θέση (th) του φυσιγγίου που επιλέχθηκε
        self.practicalb.append(V2)
# Λίστα πρακτικών φορτίων
        h=10*self.hdiam
# Επιγόμωση
        B2=round(math.sqrt(2)*(round(V2,2)+(self.B[0]/2)),2)
# Πλευρά τετραγώνου
        self.B.append(B2)
# Λίστα πλευράς τετραγώνου
        No_second=round((self.hdepth-h)/(self.cartridges[th]["length"])*2)/2
# Αριθμός φυσιγγίων
        mass2=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_second)),2)
# Μάζα γόμωσης ανά διάτρημα
```

```
        self.massph.append(mass2)
# Λίστα μάζας γόμωσης/διάτρημα
        Holes_second=4
# Αριθμός διατρημάτων
        self.holes.append(Holes_second)
# Λίστα αριθμού διατρημάτων (4 επειδή ειναι τετράγωνα)
        self.test_no.append(Holes_second)
        self.test_mass.append(mass2)
        for i in range(len(self.number_second)):
            self.number_second[th]=No_second
# Για κάθε τύπο φυσιγγίου , τοποθετείται στη λίστα ο αριθμός που θα χρησιμοποιηθεί
πχ.[4,0,4]

        # Η διαδικασία είναι η ίδια με του πρώτου τετραγώνου .

        # Η μόνη διαφορά είναι ότι η την επιλογή του αντίστοιχου φυσιγγίου που θα
χρησιμοποιήσω θα γίνει με βάση τη θεωρητική πλευρά τετραγώνου

    def third_quadrangle(self):
        B3temp=round(math.sqrt(2)*(round(self.practicalb[1],2)+(round(self.B[0]/2,2))-
(self.a*self.hdepth+self.b)),2)
        practicalb=[]
        minimum=1000
        th=0
        for i in range(len(self.charge)):

V3max=round((8.8*0.01)*math.sqrt((B3temp*self.charge[i]*self.Sanfo)/(self.hdiam*self.c)),2)
                Vpractical=round(V3max-((self.a*self.hdepth)+self.b),2)
                practicalb.append(Vpractical)
                if Vpractical<=(2*B3temp) and  abs(Vpractical-2*B3temp)<minimum and
Vpractical>0.5*B3temp:
                        minimum=abs(Vpractical-2*B3temp)
                        th=i
        V3=practicalb[th]
        self.practicalb.append(V3)
        h=10*self.hdiam
        B3=round(math.sqrt(2)*(V3+(self.B[1]/2)),2)
        self.B.append(B3)
        No_third=round((self.hdepth-h)/(self.cartridges[th]["length"])*2)/2
        mass3=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_third)),2)
        self.massph.append(mass3)
        Holes_third=4
        self.holes.append(Holes_third)
        self.test_no.append(Holes_third)
        self.test_mass.append(mass3)
        for i in range(len(self.number_third)):
            self.number_third[th]=No_third

     # Ομοίως με τα παραπάνω τετράγωνα


    def fourth_quadrangle(self):
        B4temp=round(math.sqrt(2)*(round(self.practicalb[2],2)+(round(self.B[1],2)/2)-
(self.a*self.hdepth+self.b)),2)
        practicalb=[]
        minimum=1000
        th=0
        for i in range(len(self.charge)):

V4max=round((8.8*0.01)*math.sqrt((round(B4temp,2)*self.charge[i]*self.Sanfo)/(self.hdiam*se
lf.c)),2)
                Vpractical=round(V4max-((self.a*self.hdepth)+self.b),2)
                practicalb.append(Vpractical)
                if Vpractical<=(2*B4temp) and  abs(Vpractical-2*B4temp)<minimum and
Vpractical>0.5*B4temp:
                        minimum=abs(Vpractical-2*B4temp)
```

```
                    th=i
        V4=practicalb[th]
        B4=round(math.sqrt(2)*(V4+(self.B[2]/2)),2)
        self.practicalb.append(V4)
        if self.abutmentH>=4.5 or self.abutmentH+self.archHeight>=4.5:
            h=10*self.hdiam
            self.B.append(B4)
            No_fourth=round((self.hdepth-h)/(self.cartridges[th]["length"])*2)/2
            mass4=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_fourth)),2)
            self.massph.append(mass4)
            Holes_fourth=4
            self.holes.append(Holes_fourth)
            self.test_no.append(Holes_fourth)
            self.test_mass.append(mass4)
            for i in range(len(self.number_fourth)):
                self.number_fourth[th]=No_fourth
        else:
            self.practicalb.pop(3)
            self.B.append(self.B[2])
            self.practicalb.append(self.practicalb[2])
            h=10*self.hdiam
            B4==self.B[2]
            No_fourth=0
            mass4=0
            self.massph.append(mass4)
            Holes_fourth=0
            self.holes.append(Holes_fourth)
            self.test_no.append(Holes_fourth)
            self.test_mass.append(mass4)
            for i in range(len(self.number_fourth)):
                self.number_fourth[th]=No_fourth
        # Ομοίως με τα παραπάνω τετράγωνα


    def lifters(self):
        x=round(0.6*self.hdepth,2)
# Το πρακτικό φορτίο των Lifters πρέπει να είναι μικρότερο ή ίσο με 0.6 φορές το Βάθος του
διατρήματος
        minimum=1000
        th=0
        for i in range(len(self.charge)):

v1max=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
            if (v1max<=x) and (x-v1max)<minimum:

vl=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)    #
Παράλληλα , χρησιμοποιούμε το φυσίγγιο με τη μέγιστη δυνατή γραμμική πυοκνότητας γόμωσης
                minimum=(x-v1max)
                th=i
        vl=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+0.05))),2)
        if vl>=1.4:
            VLmax=vl
# Αν το πρακτικό φορτίο είναι μεγαλύτερο από 1.4 , η διορθωμένη σταθερά c αλλάζει
        else:

VLmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+(0.07/vl)))),2)
        N=int(((self.width+2*self.hdepth*math.sin(self.g))/VLmax)+1)
# Αριθμός διατρημάτων δαπέδου
        El=round(((self.width+2*self.hdepth*math.sin(self.g))/(N-1)),2)
# Απόσταση διατρημάτων δαπέδου
        corner_El=round(El-(self.hdepth*math.sin(self.g)),2)
# Απόσταση γωνιακών διατρημάτων δαπέδου
        VL=round(VLmax-(self.hdepth*math.sin(self.g))-(self.a*self.hdepth+self.b),2)
# Πρακτικό φορτίο (Μέγιστο - F)
        self.practicalb.append(VL)
```

```
        hb=round(1.25*VL,2)
# Μήκος επιγόμωσης πυθμένα
        hc=self.hdepth-hb-(10*self.hdiam)
# Μήκος επιγόμωσης στήλης
        charge_column=round(0.7*self.charge[th],2)
# Γραμμική πυκνότητα γόμωσης στήλης
        minimum=1000
        th1=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge_column)<minimum and (self.charge[i]-
0.05<=charge_column):  # Επιλογή φυσιγγίου για την γόμωση στήλης ανάλογα με την γραμμική
πυκνότητα γόμωσης στήλης
                minimum=abs(self.charge[i]-charge_column)
                th1=i
        No_cartridges_bottom= round((hb/self.cartridges[th]["length"])*2)/2
# Αριθμός φυσιγγίων πυθμένα
        No_cartridges_column= round((hc/self.cartridges[th1]["length"])*2)/2
# Αριθμός φυσιγγίων στήλης

massL=round((((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_bottom))+((s
elf.charge[th1])*(self.cartridges[th1]["length"])*(No_cartridges_column))),2)
        self.massph.append(massL)
# Λίστα μάζας γόμωσης/διάτρημα
        Holes_lifters=N
# Αριθμός διατρημάτων δαπέδου
        self.holes.append(Holes_lifters)
# Λίστα αριθμού διατρημάτων
        for i in range(len(self.number_lifters)):
             if th==th1:
# Αν τα φυσίγγια στήλης είναι τα ίδια με τα φυσίγγια πυθμένα (th1,th) πρέπει να προστεθούν
οι αριθμοί και όχι να καταχωρυθεί μόνο το ένα
                self.number_lifters[th]=No_cartridges_bottom+No_cartridges_column
                self.number_lifters[th1]=No_cartridges_bottom+No_cartridges_column
             else:
                self.number_lifters[th]=No_cartridges_bottom
# Για κάθε τύπο φυσιγγίου , τοποθετείται στη λίστα ο αριθμός που θα χρησιμοποιηθεί
πχ.[4,0,4]
                self.number_lifters[th1]=No_cartridges_column
        self.spacing.append(El)
#Λίστα απόστασης διατρημάτων
        self.corner_holes.append(corner_El)
#Λίστα απόστασης γωνιακών διατρημάτων

        # Αφού υπολογίζεται το πρακτικό φορτίο , τον αριθμό των διατρημάτων και τις
αποστάσεις, σειρά έχει το μήκος επιγόμωσης πυθμένα και στήλης.
        # Αυτό γίνεται γιατί χρησιμοποιούνται διαφορετικά φυσίγγια στο καθένα , ανάλογα με
τη γραμμική πυκνότητα γομώσής τους.


    def roof(self):
        #k=secure_random.uniform(15,16)
        k=15
#Smooth blasting
        E=round(k*self.hdiam,2)
# Spacing
        VRmax=round((E/0.8),2)
# Μέγιστο φορτίο
        VR=round(VRmax-(self.hdepth*math.sin(self.g))-(self.a*self.advance+self.b),2)
# Πρακτικό φορτίο
        self.practicalb.append(VR)
        R=round(((self.archHeight**2+((self.width/2)**2))/self.archHeight)/2,2)
        angle_rad=round(math.acos((R-self.archHeight)/R)*2,2)
        angle_deg=round((angle_rad*180)/math.pi,2)
        Lr=round(angle_deg*(math.pi/180)*R,2)
        #No_holes_roof=int(((Lr+(self.abutmentH*math.sin(self.g)))/E)+1)
        spacing_roof=round(Lr/(No_holes_roof-1),2)
```

```
        min_charge=round(90*(self.hdiam**2),2)
# Ελάχιστη γραμμική πυκνότητα γόμωσης για smooth blasting
        minimum=1000
        th=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-min_charge)<minimum and (self.charge[i]>min_charge):
# Επιλογή φυσιγγίου ανάλογα με την ελάχιστη γραμμική πυκνότητα γόμωσης
                minimum=abs(self.charge[i]-min_charge)
                th=i
        No_cartridges_roof=int(self.hdepth/self.cartridges[th]["length"])
# Αριθμός φυσιγγίων. Δεν έχει επιγόμωση

massR=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_roof)),2)
# Μάζα γόμωσης ανά διάτρημα
        self.massph.append(massR)

No_holes_roof=int(((self.abutmentH+(self.abutmentH*math.sin(self.g))+self.archHeight)/E)+2)
#Συν 2 ,γιατί αλλιώς βγαίνει 7,91 .επομένως επιλέγει το 7 αντί του 8!!!!
        Holes_roof=No_holes_roof
# Αριθμός διατρημάτων οροφής
        self.holes.append(Holes_roof)
        for i in range(len(self.number_roof)):
            self.number_roof[th]=No_cartridges_roof
# Λίστα ποσότητας κάθε τύπου φυσιγγίων
        self.spacing.append(spacing_roof)


    def wall(self):
        distance=round(self.abutmentH-round(self.practicalb[4],2)-
round(self.practicalb[5],2),2)    # Χώρος που απομένει για τα διατρήματα τοίχου
        x=round(0.6*self.hdepth,2)
        minimum=1000
        th=0
        for i in range(len(self.charge)):

VWmax=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.2*(self.c+0.05)*1.25)),2)      #
Ίδιο μοτίβο με διατρήματα δαπέδου
            if (VWmax<=x) and (x-VWmax)<minimum:

vw=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.2*(self.c+0.05)*1.25)),2)
                minimum=(x-vw)
                th=i
        V=vw
# Μέγιστο φορτίο με άλλη ονομασία για να μην συγχωνευτεί με το VRmax του βρόγχου
        VW=round(V-(self.hdepth*math.sin(self.g))-((self.a*self.advance)+self.b),2)
        self.practicalb.append(VW)
        No_holes_wall=2*(int((distance/(VW*1.25)+2)))
        Spacing=round(distance/((No_holes_wall/2)-1),2)
# Απόσταση διατρημάτων τοίχου
        hb=round(1.25*VW,2)
# Μήκος επιγόμωσης πυθμένα
        hc=self.hdepth-hb-(10*self.hdiam)
# Μήκος επιγόμωσης στήλης
        charge_column=round(0.7*self.charge[th],2)
# charge_column=0.7*(charge_bottom) γραμμική πυκνότητα στήλης
        minimum=1000
        th1=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge_column)<minimum and (self.charge[i]-
0.05<=charge_column):  # Επιλογή φυσιγγίου γόμωσης στήλης
                minimum=abs(self.charge[i]-charge_column)
                th1=i
        No_cartridges_bottom=round((hb/self.cartridges[th]["length"])*2)/2
        No_cartridges_column=round((hc/self.cartridges[th1]["length"])*2)/2
```

```
massW=round((((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_bottom))+((s
elf.charge[th1])*(self.cartridges[th1]["length"])*(No_cartridges_column))),2)
        self.massph.append(massW)
        Holes_wall=No_holes_wall
        self.holes.append(Holes_wall)
        for i in range(len(self.number_wall)):
            if th==th1:
                self.number_wall[th]=No_cartridges_bottom+No_cartridges_column
            else:
                self.number_wall[th]=No_cartridges_bottom
                self.number_wall[th1]=No_cartridges_column
        self.spacing.append(Spacing)

    def stoping(self):
        x=round(0.6*self.hdepth,2)
        #Horizontal cartridges

        minimum=1000
        th=0
        for i in range(len(self.charge)):

VHmax=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
            if (VHmax<=x) and (x-VHmax)<minimum:

vh=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
                minimum=(x-VHmax)
                th=i
        distance_h=round(self.width-round(self.B[3],2)-(2*round(self.practicalb[6],2)),2)
# Χώρος που απομένει για τα οριζόντια διατρήματα διερεύνησης

Vhmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+0.05)*1.25)),2)
# Μέγιστο φορτίο φυσιγγίου που επιλέχθηκε
        VH=round(Vhmax-(self.a*self.advance+self.b),2)
# Πρακτικό φορτίο
        cols_h=0
        rows_h=0
        if VH>=distance_h:
# VH πρακτικό φορτίο, Vh νέο πρακτικό φορτίο αν ισχύει η συνθήκη
            VH=round(distance_h+0.01,2)
            cols_h=1
# Αριθμός στηλών
            Spacing_columns_h=VH
# Απόσταση διατρημάτων στήλης
        else:
            cols_h=int(distance_h/VH)
            Spacing_columns_h=VH
            if (cols_h)*Spacing_columns_h<distance_h:
                cols_h=int(distance_h/VH)+1
                VH=round(distance_h/cols_h,2)
                Spacing_columns_h=VH
            else:
                cols_h=int(distance_h/VH)
        if (1.25*round(Vhmax-(self.a*self.advance+self.b),2))>=self.B[3]:
            rows_h=2
            Spacing_rows_h=self.B[3]
        else:
            VH=(round(Vhmax-(self.a*self.advance+self.b),2))
            rows_h=int((self.B[3]/(0.9*VH))+1)
            Spacing_rows_h=round(0.9*VH,2)
            if (rows_h-1)*Spacing_rows_h<self.B[3]:
                rows_h=rows_h+1
                VH=round(self.B[3]/(rows_h-1),2)
                Spacing_rows_h=VH
            else:
                rows_h=int((self.B[3]/(0.9*VH)))
```

```
        No_holes_h=cols_h*rows_h
# Αριθμός διατρημάτων οριζοντίων . στήλες x σειρές
        Spacing_horizontal_columns=Spacing_columns_h
# Απόσταση στηλών οριζοντίων διατρημάτων για το διάγραμμα
        Spacing_horizontal_rows=Spacing_rows_h
# Απόσταση σειρών οριζοντίων διατρημάτων για το διάγραμμα
        self.No_columns.append(cols_h)
# Λίστα αριθμού στηλών
        self.No_rows.append(rows_h)
# Λίστα αριθμού σειρών
        self.spacing_columns.append(Spacing_horizontal_columns)
# Λίστες αποστάσεων στηλών / σειρών
        self.spacing_rows.append(Spacing_horizontal_rows)
        self.practicalb.append(VH)


        #Downward Cartrtidges

        height_d=round((self.abutmentH+self.archHeight)-round(self.B[3],2)-
round(self.practicalb[4],2)-round(self.practicalb[5],2),2)
        VDmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.2*(self.c+0.05)*1.25)),2)
        VD=round(VDmax-(self.a*self.advance+self.b),2)
        xS1=0+self.practicalb[6]
# Αρχικές συντεταγμένες για τα κάθετα διατρήματα διεύρυνσης, το πάνω αριστερό άκρο του 4ου
τετραγώνου του cut
        yS1=0+self.practicalb[4]+self.B[3]
        carts=0
        if VD>height_d:
            vd=round(height_d,2)
# Αν το πρακτικό φορτίο είναι μεγαλύτερο από το διαθέσιμο χώρο, τότε χρησιμοποείται νέο
πρακτικό φορίτο όσο ο διαθέσιμος χώρος
            if (self.width-2*self.practicalb[6])<(2*VD):
                cols_d=3
            else:
                cols_d=int(((self.width-2*self.practicalb[6])/(1*VD))+1)
# Στήλες καθέτων διατρημάτων
            Spacing_columns_d=round((self.width-2*self.practicalb[6])/(cols_d-1),2)
# Απόσταση στηλών καθέτων διατρημάτων
            for i in range(0,cols_d):
                XS=xS1+(i*Spacing_columns_d)
                YS=0+self.practicalb[4]+self.B[3]
                a3= self.width/2
                b3 =(self.abutmentH+self.archHeight)-(self.abutmentH-self.practicalb[5])-
self.practicalb[5]
                x03 =self.width/2
                y03 =self.abutmentH-self.practicalb[5]
                x3 = XS
                y3 = b3 * np.sqrt(1 - ((x3-x03) / a3) ** 2) + y03
# Το πρακτικό φορτίο οροφής είναι μια καμπύλη επομένως πρέπει ο διαθέσιμος χώρος να
υπολογιστεί βάσει της καμπύλης
                d=int(((y3+0.1-YS)/vd))
                if d<1:
                    carts=carts+1
                else:
                    if (d)*vd<(y3-YS-0.1):
                        carts=carts+d+1
                    else:
                        carts=carts+d
        else:
            vd1=VD
            #cols_d=int(((self.width-2*self.practicalb[6])/(1*VD))+1)
            if (self.width-2*self.practicalb[6])<(2*VD):
                cols_d=3
            else:
                cols_d=int(((self.width-2*self.practicalb[6])/(1*VD))+1)
# Στήλες καθέτων διατρημάτων
```

```
            Spacing_columns_d=round((self.width-2*self.practicalb[6])/(cols_d-1),2)
            d=int((height_d/vd1)+1)
            if d*vd1>height_d:
                vd=round((height_d/d),2)
            else:
                vd=vd1
            for i in range(0,cols_d):
                XS=xS1+(i*Spacing_columns_d)
                YS=0+self.practicalb[4]+self.B[3]
                a3= self.width/2
                b3 =(self.abutmentH+self.archHeight)-(self.abutmentH-self.practicalb[5])-
self.practicalb[5]
                x03 =self.width/2
                y03 =self.abutmentH-self.practicalb[5]
                x3 = XS
                y3 = b3 * np.sqrt(1 - ((x3-x03) / a3) ** 2) + y03
                d=int((y3+0.1-YS)/vd)
                if d<1:
                    carts=carts+1
                else:
                    if (d)*vd<(y3-YS-0.1):
                        carts=carts+d+1
                    else:
                        carts=carts+d
        No_holes_d= carts
        No_holes_stoping=No_holes_d+No_holes_h
#Συνολικός αριθμός διατρημάτων διεύρυνσης
        self.holes.append(No_holes_stoping)
        self.spacing_columns.append(Spacing_columns_d)
        self.cols_downwards.append(cols_d)
        self.practicalb.append(vd)


        #Cartridges Calculation

        hb=round(1.25*VH,2)
# Μήκος επιγόμωσης πυθμένα
        hc=round(self.hdepth-round(hb,2)-(10*self.hdiam),2)
# Μήκος επιγόμωσης στήλης
        charge_column=round(0.5*self.charge[th],2)
# Γραμμική πυκνότητα γόμωσης στήλης
        minimum=1000
        th1=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge_column)<minimum and (self.charge[i]-
0.05<=charge_column):  # Επιλογή φυσιγγίου για επιγόμωση στήλης
                minimum=abs(self.charge[i]-charge_column)
                th1=i
        No_cartridges_bottom= round((hb/self.cartridges[th]["length"])*2)/2
        No_cartridges_column= round((hc/self.cartridges[th1]["length"])*2)/2

massS=round((((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_bottom))+((s
elf.charge[th1])*(self.cartridges[th1]["length"])*(No_cartridges_column))),2)
        self.massph.append(massS)
        for i in range(len(self.number_stoping)):
            if th==th1:
                self.number_stoping[th]=No_cartridges_bottom+No_cartridges_column
            else:
                self.number_stoping[th]=No_cartridges_bottom
                self.number_stoping[th1]=No_cartridges_column


    def sum(self):
        for i in range(len(self.massph)):
            x=round((self.massph[i]*self.holes[i]),2)
            self.total.append(x)
```

```
        # Υπολογισμός συνολικής μάζας εκρηκτικών ( μάζα εκρηκτικών ανά διάτρημα x
διατρήματα)



    def table1(self):

li=[self.number_first,self.number_second,self.number_third,self.number_fourth,self.number_l
ifters,self.number_roof,self.number_wall,self.number_stoping]
        data=pd.DataFrame(li)
# Δημιουργία DataFrame σαν συγκεντρωτικός πίνακας
        data.columns=[self.diam]
        data.index=["First Quadrangle","Second Quandrangle","Third Quandrangle","Fourth
Quandrangle","Lifters","Roof","Wall","Stoping"]  # Σειρές πίνακα, μέρη της διάτρησης
        data.insert(0,"No. Holes",self.holes,True)
# Εισαγωγή στηλών πίνακα ( αριθμός διατρημάτων, γραμμική πυκνότητα ανά διάτρημα, συνολική
μάζα )
        data.insert(1,"Charge per Hole , kg",self.massph,True)
        data.insert(2,"Total , kg",self.total,True)
        print(tabulate(data,headers='keys',tablefmt ='fancy_grid'))
# Μορφή κελιών πίνακα (pretty ή fancy_grid)



        # Τελικός πίνακας
    def table2(self):
        Total_weight = round(sum(self.total),2)
# Συνολική μάζα εκρηκτικών
        Total_holes = sum(self.holes)
# Συνολικά διατρήματα (πρόσθεση των διατρημάτων στη λίστα διατρημάτων)
        R=round(((self.archHeight**2+((self.width/2)**2))/self.archHeight)/2,2)
# Ακτίνα κύκλου
        angle_rad=round(math.acos((R-self.archHeight)/R)*2,2)
# Γωνία ακτίνας σε rad
        angle_deg=round((angle_rad*180)/math.pi,2)
        Cross_sectional_area=round((self.width*self.abutmentH)+((angle_rad/2)*(R**2))-((R-
self.archHeight)*(0.5*self.width)),2)  # Συνολικό εμβαδό μετώπου
        Specific_charge= round((Total_weight/(Cross_sectional_area*self.advance)),2)
# Ειδική κατανάλωση
        Specific_drilling=
round(((Total_holes*self.hdepth)/(Cross_sectional_area*self.advance)),2) # Ειδική διάτρηση

li=[Total_weight,Total_holes,self.advance,self.hdepth,Cross_sectional_area,Specific_charge,
Specific_drilling] # Στήλες Πίνακα , τιμές στοιχείων
        data=pd.DataFrame(li)
        data.index=['Total charge weight (kg)','Total No. of holes ','Advance (m)','Hole
depth (m)','Cross sectional area (m2)','Specific charge (kg/m3)','Specific drilling
(m/m3)']
        print(tabulate(data, tablefmt ='fancy_grid'))



        # ΠΑΡΑΔΕΙΓΜΑ ΤΕΤΡΑΓΩΝΙΚΗΣ ΣΤΟΑΣ 3.5 m X 4 m
    def case_study2(self):

        fig, ax = plt.subplots(1, 1)
        ax.set_aspect('equal')
        x_asix = np.array([self.width])
        y_asix = np.array([self.abutmentH])
        x_coordenates = np.concatenate([ x_asix])
        y_coordenates = np.concatenate([y_asix])
        ax.set_facecolor('whitesmoke')
        ax.plot(x_coordenates, y_coordenates, color= 'black')
        x=round(0.6*self.hdepth,2)
        if self.abutmentH>=4.5 or self.abutmentH+self.archHeight>=4.5:
```

```
            xQ=(self.width/2)                                                    #
Συντεταγμένες αρχής cut (τέρμα  αριστερά δίπλα απο το πρακτικό φορτίο του τοίχου)
            yQ=(self.abutmentH/2)                                                #
Ορίζεται αρχικό σημείο το κενό διάτρημα που βρίσκεται στο μέσο του cut.

# Συντεταγμένη ψ υπολογίζεται από το τέλος του πρακτικού φορτίου των lifters και το μισό
από την πλευρά του 4ου τετραγώνου
            plt.scatter(xQ,yQ)
# Συντεταγμένη χ υπολογίζεται από το τέλος του πρακτικού φορτίου του τοίχου και το μισό από
την πλευρά του 4ου τετραγώνου

            x1=[xQ-
(round(self.B[0],2)*math.cos(0.25*math.pi)),xQ,xQ+(round(self.B[0],2)*math.cos(0.25*math.pi
)),xQ,xQ-(round(self.B[0],2)*math.cos(0.25*math.pi))]
            y1=[yQ,yQ+(round(self.B[0],2)*math.sin(0.25*math.pi)),yQ,yQ-
(round(self.B[0],2)*math.sin(0.25*math.pi)),yQ]

        # Με αρχικές συντεταγμένες τα xQ, yQ, υπολογίζονται οι συντεταγμένες του κάθε
σημείου του τετραγώνου
        # Ο υπολογισμός γίνεται με τα συνημίτονα και τα ημίτονα των γωνιών και τις
αντίστοιχες πλευρές του κάθε τετραγώνου self.B

            x2=[(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[1],2),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[1],2),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))]

y2=[(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[1],2),(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[1],2),(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*
math.pi)))]

            x3=[xQ-
(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),xQ,xQ+(round(self.practicalb[2],2)+(ro
und(self.B[1],2)/2)),xQ,xQ-(round(self.practicalb[2],2)+(round(self.B[1],2)/2))]
            y3=[yQ,yQ+(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),yQ,yQ-
(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),yQ]


            x4=[(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[3],2),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[3],2),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))]

y4=[(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[3],2),(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[3],2),(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*
math.pi)))]

            plt.scatter(x1,y1,color='olive')
            plt.scatter(x2,y2,color='olive')
            plt.scatter(x3,y3,color='olive')
```

```
            plt.scatter(x4,y4,color='olive')
            plt.plot(x1,y1,color='black')
# Δημιουργία ευθειών ανάμεσα στα διατρήματα για την απεικόνιση τετραγώνων!!!
            plt.plot(x2,y2,color='black')
            plt.plot(x3,y3,color='black')
            plt.plot(x4,y4,color='black')
            ax.annotate(self.B[2],(xQ-(x3[3]-x3[2]/2),y3[4]-y3[3]/2))
            ax.annotate(self.B[3],(x4[0]+self.B[3]/2,y4[0]))
# ΄Οπου annotate, η απεικόνιση της πλευράς του κάθε τετραγώνου!!!
            ax.annotate(self.B[1],(xQ,yQ+self.B[1]/2))
            ax.annotate(self.B[0],(xQ-(x1[2]-x1[1]),yQ-(y1[1]-y1[2])))
            ax.annotate(self.emptyf,(xQ,yQ),color='blue')

        else:
            xQ=(self.width/2)                                                    #
Συντεταγμένες αρχής cut (τέρμα  αριστερά δίπλα απο το πρακτικό φορτίο του τοίχου)
            yQ=(self.abutmentH/2)                                                #
Ορίζεται αρχικό σημείο το κενό διάτρημα που βρίσκεται στο μέσο του cut.
            x1=[(xQ-((round(self.practicalb[0],2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))+round(self.B[0],2),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))+round(self.B[0],2),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))]

y1=[(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi))),(yQ+((round(self.practicalb
[0],2))*math.sin(0.25*math.pi))),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi))
)-round(self.B[0],2),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi)))-
round(self.B[0],2),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi)))]

            x2=[xQ-
(round(self.B[1],2)*math.cos(0.25*math.pi)),xQ,xQ+(round(self.B[1],2)*math.cos(0.25*math.pi
)),xQ,xQ-(round(self.B[1],2)*math.cos(0.25*math.pi))]
            y2=[yQ,yQ+(round(self.B[1],2)*math.sin(0.25*math.pi)),yQ,yQ-
(round(self.B[1],2)*math.sin(0.25*math.pi)),yQ]

            x3=[(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[2],2),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[2],2),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))]

y3=[(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[2],2),(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[2],2),(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*
math.pi)))]

            plt.scatter(x1,y1,color='olive')
            plt.scatter(x2,y2,color='olive')
            plt.scatter(x3,y3,color='olive')
            plt.plot(x1,y1,color='black')
# Δημιουργία ευθειών ανάμεσα στα διατρήματα για την απεικόνιση τετραγώνων!!!
            plt.plot(x2,y2,color='black')
            plt.plot(x3,y3,color='black')
            ax.annotate(self.B[2],(xQ,yQ+self.B[2]/2))
# ΄Οπου annotate, η απεικόνιση της πλευράς του κάθε τετραγώνου!!!
            ax.annotate(self.B[1],(xQ,yQ+self.B[1]/2))
            ax.annotate(self.B[0],(xQ-(x1[2]-x1[1]),yQ-(y1[1]-y1[2])))
            ax.annotate(self.emptyf,(xQ,yQ),color='blue')
```

```
        # Lifters
        x=round(0.6*self.hdepth,2)
# Το πρακτικό φορτίο των Lifters πρέπει να είναι μικρότερο ή ίσο με 0.6 φορές το Βάθος του
διατρήματος
        minimum=1000
        th=0
        for i in range(len(self.charge)):

v1max=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
            if (v1max<=x) and (x-v1max)<minimum:

vl=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)    #
Παράλληλα , χρησιμοποιούμε το φυσίγγιο με τη μέγιστη δυνατή γραμμική πυοκνότητας γόμωσης
                minimum=(x-v1max)
                th=i
        vl=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+0.05))),2)
        if vl>=1.4:
            VLmax=vl
# Αν το πρακτικό φορτίο είναι μεγαλύτερο από 1.4 , η διορθωμένη σταθερά c αλλάζει
        else:

VLmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+(0.07/vl)))),2)
        N=int(((self.width+2*self.hdepth*math.sin(self.g))/VLmax)+1)
# Αριθμός διατρημάτων δαπέδου
        El=round(((self.width+2*self.hdepth*math.sin(self.g))/(N-1)),2)
# Απόσταση διατρημάτων δαπέδου
        corner_El=round(El-(self.hdepth*math.sin(self.g)),2)
# Απόσταση γωνιακών διατρημάτων δαπέδου
        VL=round(VLmax-(self.hdepth*math.sin(self.g))-(self.a*self.hdepth+self.b),2)
# Πρακτικό φορτίο (Μέγιστο - F)
        self.practicalb.append(VL)
        hb=round(1.25*VL,2)
# Μήκος επιγόμωσης πυθμένα
        hc=self.hdepth-hb-(10*self.hdiam)
# Μήκος επιγόμωσης στήλης
        charge_column=round(0.7*self.charge[th],2)
# Γραμμική πυκνότητα γόμωσης στήλης
        minimum=1000
        th1=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge_column)<minimum and (self.charge[i]-
0.05<=charge_column):  # Επιλογή φυσιγγίου για την γόμωση στήλης ανάλογα με την γραμμική
πυκνότητα γόμωσης στήλης
                minimum=abs(self.charge[i]-charge_column)
                th1=i
        No_cartridges_bottom= round((hb/self.cartridges[th]["length"])*2)/2
# Αριθμός φυσιγγίων πυθμένα
        No_cartridges_column= round((hc/self.cartridges[th1]["length"])*2)/2
# Αριθμός φυσιγγίων στήλης

massL=round((((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_bottom))+((s
elf.charge[th1])*(self.cartridges[th1]["length"])*(No_cartridges_column))),2)
        self.test_mass.append(massL)
# Λίστα μάζας γόμωσης/διάτρημα
        Holes_lifters=N
# Αριθμός διατρημάτων δαπέδου
        self.test_no.append(Holes_lifters)
# Λίστα αριθμού διατρημάτων
        for i in range(len(self.number_lifters)):
             if th==th1:
# Αν τα φυσίγγια στήλης είναι τα ίδια με τα φυσίγγια πυθμένα (th1,th) πρέπει να προστεθούν
οι αριθμοί και όχι να καταχωρυθεί μόνο το ένα
                self.number_lifters1[th]=No_cartridges_bottom+No_cartridges_column
                self.number_lifters1[th1]=No_cartridges_bottom+No_cartridges_column
             else:
```

```
                self.number_lifters1[th]=No_cartridges_bottom
# Για κάθε τύπο φυσιγγίου , τοποθετείται στη λίστα ο αριθμός που θα χρησιμοποιηθεί
πχ.[4,0,4]
                self.number_lifters1[th1]=No_cartridges_column

        # Lifters Design
        l=Holes_lifters
# Αριθμός διατρημάτων δαπέδου
        for i in range(0,l+1):
            if i==0:
# Τοποθέτηση του πρώτου αρχικού διατρήματος στην άκρη
                x1=0
                y1=0
            else:
                if i==1:
# Τοποθέτηση του αμέσως επόμενου διατρήματος με απόσταση γωνιακών διατρημάτων Lifters
                    x1=0+corner_El
                    y1=0
                    ax.annotate(corner_El,(x1/2,y1+0.1))
# Τοποθέτηση της απόστασης των γωνιακών διατρημάτων σαν διάσταση ανάμεσα στα διατρήματα
                else:
                    if i==l:
# Τοποθέτηση του τελευταίου διατρήματος στο τέλος των Lifters
                        x1=self.width
                        y1=0
                    else:
                        if i==l-1:
# Τοποθέτηση του προτελευταίου διατρήματος με απόσταση γωνιακών διατρημάτων δαπέδου από το
τελευταίο
                            x1=self.width-corner_El
                            y1=0
                            ax.annotate(corner_El,(self.width-corner_El/2,y1+0.1))
# Τοποθέτηση διάστασης ανάμεσα στα διατρήματα
                        else:
                            x1=corner_El+(i-2)*self.spacing[0]
# Τα ενδιάμεσα διατρήματα με κανονική απόσταση διατρημάτων Lifters
                            y1=0
                            ax.annotate(El,(x1+El/2,y1+0.1))
# Τοποθέτηση διάστασης ανάμεσα στα διατρήματα
            plt.scatter(x1,y1,color='navy')


        # Roof
        k=15
        E=round(k*self.hdiam,2)
        VRmax=round((E/0.8),2)
        VR=round(VRmax-(self.hdepth*math.sin(self.g))-(self.a*self.advance+self.b),2)

        min_charge=round(90*(self.hdiam**2),2)
# Ελάχιστη γραμμική πυκνότητα γόμωσης για smooth blasting
        minimum=1000
        th=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-min_charge)<minimum and (self.charge[i]>min_charge):
# Επιλογή φυσιγγίου ανάλογα με την ελάχιστη γραμμική πυκνότητα γόμωσης
                minimum=abs(self.charge[i]-min_charge)
                th=i
        No_cartridges_roof=int(self.hdepth/self.cartridges[th]["length"])

massR=round(((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_roof)),2)
        No_holes_roof=int(self.width/E)+1
        spacing=round(self.width/(No_holes_roof-1),2)
        Holes_roof=No_holes_roof
        self.test_no.append(Holes_roof)
        self.test_mass.append(massR)
        total=massR*No_holes_roof
```

```
        for i in range(len(self.number_roof)):
            self.number_roof1[th]=No_cartridges_roof
        for i in range(0,Holes_roof):
            xR=(0+(i)*spacing)
            yR=self.abutmentH
            plt.scatter(xR,yR,color='brown')
            xM=(self.width/2)
            yM=self.abutmentH-0.2
            ax.annotate(spacing,(xM,yM))

        # Horizontal stoping τετραγωνικής
        minimum=1000
        th=0
        for i in range(len(self.charge)):

VHmax=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
            if (VHmax<=x) and (x-VHmax)<minimum:

vh=round(0.9*math.sqrt((self.charge[i]*self.Sanfo)/(1.25*(self.c+0.05)*1.25)),2)
                minimum=(x-VHmax)
                th=i
        distance_h=round(((0+self.width/2)-(self.B[3]/2)),2)

Vhmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.45*(self.c+0.05)*1.25)),2)
        VH=round(Vhmax-(self.a*self.advance+self.b),2)
        cols_h=0
        rows_h=0
        if VH>=distance_h:
            VH=round(distance_h+0.01,2)
            cols_h=1
            Spacing_columns_h=VH
        else:
            cols_h=int((distance_h/VH)+1)
            Spacing_columns_h=VH
            if (cols_h-1)*Spacing_columns_h<distance_h:
                cols_h=int((distance_h/VH)+1)
                VH=round(distance_h/cols_h,2)
                Spacing_columns_h=VH
            else:
                cols_h=int(distance_h/VH)
        if (1.25*round(Vhmax-(self.a*self.advance+self.b),2))>=self.B[3]:
            rows_h=2
            Spacing_rows_h=self.B[3]
        else:
            VH=(round(Vhmax-(self.a*self.advance+self.b),2))
            rows_h=int((self.B[3]/(0.9*VH))+1)
            Spacing_rows_h=round(0.9*VH,2)
            if (rows_h-1)*Spacing_rows_h<self.B[3]:
                rows_h=rows_h+1
                VH=round(self.B[3]/(rows_h-1),2)
                Spacing_rows_h=VH
            else:
                rows_h=int((self.B[3]/(0.9*VH)))
        No_holes_h=2*(cols_h*rows_h)
        xS=(self.width/2)-(self.B[3]/2)
        yS=(self.abutmentH/2)+(self.B[3]/2)
        xZ=(self.width/2)+(self.B[3]/2)
        yZ=(self.abutmentH/2)+(self.B[3]/2)
        c1=cols_h
        r1=rows_h
        for i in range(0,c1):
            xh=xS-((i+1)*Spacing_columns_h)
            yh=yS
            xz=xZ+((i+1)*Spacing_columns_h)
            yz=yZ
            plt.scatter(xh,yh,color='red')
```

```
        plt.scatter(xz,yz,color='red')
        for z in range(0,r1):
            xh1=xh
            yh1=yS-((z)*Spacing_rows_h)
            xz1=xz
            yz1=yZ-((z)*Spacing_rows_h)
            plt.scatter(xh1,yh1,color='red')
            plt.scatter(xz1,yz1,color='red')
        plt.scatter(xh,yh,color='red')
    for i in range(0,c1):
        x1=xS-(i*VH)
        y1=yS
        xz1=xZ+(i*VH)
        yz1=yS
        if i==0:
            ax.annotate(VH,(x1-Spacing_columns_h/2,y1),color='blue')
            ax.annotate(VH,(xz1+Spacing_columns_h/2,y1),color='blue')
    for z in range(0,r1-1):
        x2=xS-c1*Spacing_columns_h
        y2=yS-z*Spacing_rows_h
        xz2=xZ+c1*Spacing_columns_h
        yz2=yZ-z*Spacing_rows_h
        if z==0:
            ax.annotate(Spacing_rows_h,(x2,y1-Spacing_rows_h/2),color='black')
            ax.annotate(Spacing_rows_h,(xz2,y1-Spacing_rows_h/2),color='black')


    # Downwards stoping τετραγωνικής
    height_d1=round(self.abutmentH-VR-((self.abutmentH/2)+(self.B[3]/2)),2)
    VDmax=round(0.9*math.sqrt((self.charge[th]*self.Sanfo)/(1.2*(self.c+0.05)*1.25)),2)
    VD=round(VDmax-(self.a*self.advance+self.b),2)

    cols_d1=0
    rows_d1=0
    if VD>height_d1:
        vd=round(height_d1,2)
        if (self.width<(2*VD)):
            cols_d1=3
        else:
            cols_d1=int((self.width/(1*VD))+1)
        Spacing_columns_d1=round((self.width/(cols_d1-1)),2)
        rows_d1=1
        Spacing_rows_d1=vd
    else:
        vd=VD
        if (self.width<(2*VD)):
            cols_d1=3
        else:
            cols_d1=int((self.width/(1*VD))+1)
        Spacing_columns_d1=round((self.width/(cols_d1-1)),2)
        rows_d1=int(height_d1/vd)
        Spacing_rows_d1=vd
        if (rows_d1*Spacing_rows_d1)<height_d1:
            rows_d1=rows_d1+1
            Spacing_rows_d1=round((height_d1/rows_d1),2)

    x0=0
    y0=(self.abutmentH/2)+(self.B[3]/2)
    r1=rows_d1
    c1=cols_d1
    for i in range(0,r1):
        xd1=x0
        yd1=y0+((i+1)*(Spacing_rows_d1))
        plt.scatter(xd1,yd1,color='red')
        if i==0:
            xAnnot1=x0
```

```
                yAnnot1=y0+((i+1)*(Spacing_rows_d1)/2)
                ax.annotate(Spacing_rows_d1,(xAnnot1,yAnnot1),color='blue')
            for p in range(0,c1):
                xd2=x0+(p)*(Spacing_columns_d1)
                yd2=yd1
                plt.scatter(xd2,yd2,color='red')
                if p==0:
                    xAnnot2=xd2+((p)*(Spacing_columns_d1)/2)
                    yAnnot2=yd1+0.1
                    ax.annotate(Spacing_columns_d1,(xAnnot2,yAnnot2),color='black')
        height_d2=(self.abutmentH/2)-(self.B[3]/2)-VL
        cols_d2=0
        rows_d2=0
        if height_d2>0.5:
            cols_d2=cols_d1
            rows_d2=0
            if VD>height_d2:
                vd2=round(height_d2,2)
                cols_d2=cols_d1
                Spacing_columns_d2=Spacing_columns_d1
                rows_d2=1
                Spacing_rows_d2=vd2
            else:
                vd2=VD
                cols_d2=cols_d1
                Spacing_columns_d2=Spacing_columns_d1
                rows_d2=int(height_d2/vd2)
                Spacing_rows_d2=vd
                if (rows_d2*Spacing_rows_d2)<height_d2:
                    rows_d2=rows_d2+1
                    Spacing_rows_d2=round((height_d2/rows_d2),2)
            xO=0
            yO=(self.abutmentH/2)-(self.B[3]/2)
            r2=rows_d2
            c2=cols_d2
            for i in range(0,r2):
                xd3=xO
                yd3=yO-((i+1)*(Spacing_rows_d2))
                plt.scatter(xd3,yd3,color='green')
                if i==0:
                    xAnnot3=xO
                    yAnnot3=yO-((i+1)*(Spacing_rows_d1)/2)
                    ax.annotate(Spacing_rows_d2,(xAnnot3,yAnnot3),color='blue')
                for p in range(0,c2):
                    xd4=xO+(p)*(Spacing_columns_d2)
                    yd4=yd3
                    plt.scatter(xd4,yd4,color='red')
                    if p==0:
                        xAnnot4=xO+(i+1)*(Spacing_columns_d2/2)
                        yAnnot4=yd4
                        ax.annotate(Spacing_columns_d2,(xAnnot4,yAnnot4),color='blue')
        xL=[0,self.width]
        yL=[0+VL,0+VL]
        plt.plot(xL,yL,color='black')
        print(cols_d1)

        # Cartridges Calculation τετραγωνικής
        hb=round(1.25*VD,2)
        hc=round(self.hdepth-round(hb,2)-(10*self.hdiam),2)
        charge_column=round(0.5*self.charge[th],2)
        minimum=1000
        th1=0
        for i in range(len(self.charge)):
            if abs(self.charge[i]-charge_column)<minimum and (self.charge[i]-
0.05<=charge_column):
                minimum=abs(self.charge[i]-charge_column)
```

```
                th1=i
        No_cartridges_bottom= round((hb/self.cartridges[th]["length"])*2)/2
        No_cartridges_column= round((hc/self.cartridges[th1]["length"])*2)/2

massS=round((((self.charge[th])*(self.cartridges[th]["length"])*(No_cartridges_bottom))+((s
elf.charge[th1])*(self.cartridges[th1]["length"])*(No_cartridges_column))),2)
        No_holes=(2*(cols_h*rows_h))+(cols_d1*rows_d1)+(cols_d2*rows_d2)
        self.test_no.append(No_holes)
        self.test_mass.append(massS)
        total=massS*No_holes
        for i in range(len(self.number_stoping1)):
            if th==th1:
                self.number_stoping1[th]=No_cartridges_bottom+No_cartridges_column
            else:
                self.number_stoping1[th]=No_cartridges_bottom
                self.number_stoping1[th1]=No_cartridges_column

        for i in range(len(self.test_mass)):
            x=round((self.test_mass[i]*self.test_no[i]),2)
            self.test_total.append(x)

        # 1ος Πίνακας τετραγωνικής

li=[self.number_first,self.number_second,self.number_third,self.number_fourth,self.number_l
ifters1,self.number_roof1,self.number_stoping1]
        data=pd.DataFrame(li)
        data.columns=[self.diam]
        data.index=["First Quadrangle","Second Quandrangle","Third Quandrangle","Fourth
Quandrangle","Lifters","Roof","Stoping"]
        data.insert(0,"No. Holes",self.test_no,True)
        data.insert(1,"Charge per Hole , kg",self.test_mass,True)
        data.insert(2,"Total , kg",self.test_total,True)
        print(tabulate(data,headers='keys',tablefmt ='fancy_grid'))


        # 2ος Πίνακας τετραγωνικής
        Total_weight = round(sum(self.test_total),2)
        Total_holes = sum(self.test_no)
        Cross_sectional_area=round((self.abutmentH*self.width),2)
# Εμβαδό τετραγωνικής
        Specific_charge= round((Total_weight/(Cross_sectional_area*self.advance)),2)
        Specific_drilling=
round(((Total_holes*self.hdepth)/(Cross_sectional_area*self.advance)),2)

li=[Total_weight,Total_holes,self.advance,self.hdepth,Cross_sectional_area,Specific_charge,
Specific_drilling]
        data=pd.DataFrame(li)
        data.index=['Total charge weight (kg)','Total No. of holes ','Advance (m)','Hole
depth (m)','Cross sectional area (m2)','Specific charge (kg/m3)','Specific drilling
(m/m3)']
        print(tabulate(data, tablefmt ='fancy_grid'))

        plt.show()


    def plot(self):

     #ARCH DESIGN

        fig, ax = plt.subplots(1, 1)
        ax.set_aspect('equal')
        x_asix = np.array([0,0,0+self.width,0+self.width])
# Δημιουργία μετώπου με συντεταγμένες χ,ψ ανάλογα με το πλάτος και το μήκος
        y_asix = np.array([0+self.abutmentH,0,0,0+self.abutmentH])
        x_coordenates = np.concatenate([ x_asix])
        y_coordenates = np.concatenate([y_asix])
```

```
        ax.plot(x_coordenates, y_coordenates, color= 'black')

        a= self.width/2
        b = self.archHeight
        x0 = self.width/2
        y0 = self.abutmentH
        x = np.linspace(-a + x0, a + x0)
        y = b * np.sqrt(1 - ((x - x0) / a) ** 2) + y0
        ax.plot(x,y,color='black')
        ax.set_facecolor('whitesmoke')

    #LIFTERS

        xL=[0,0+self.width]
# Δημιουργία της γραμμής που απεικονίζει το όριο των Lifters
        yL=[0+self.practicalb[4],0+self.practicalb[4]]
        plt.plot(xL,yL,color="black")
        l=self.holes[4]
# Αριθμός διατρημάτων δαπέδου
        for i in range(0,l+1):
            if i==0:
# Τοποθέτηση του πρώτου αρχικού διατρήματος στην άκρη
                x1=0
                y1=0
            else:
                if i==1:
# Τοποθέτηση του αμέσως επόμενου διατρήματος με απόσταση γωνιακών διατρημάτων Lifters
                    x1=0+self.corner_holes[0]
                    y1=0
                    ax.annotate(self.corner_holes[0],(x1/2,y1+0.1))
# Τοποθέτηση της απόστασης των γωνιακών διατρημάτων σαν διάσταση ανάμεσα στα διατρήματα
                else:
                    if i==l:
# Τοποθέτηση του τελευταίου διατρήματος στο τέλος των Lifters
                        x1=self.width
                        y1=0
                    else:
                        if i==l-1:
# Τοποθέτηση του προτελευταίου διατρήματος με απόσταση γωνιακών διατρημάτων δαπέδου από το
τελευταίο
                            x1=self.width-self.corner_holes[0]
                            y1=0
                            ax.annotate(self.corner_holes[0],(self.width-
self.corner_holes[0]/2,y1+0.1)) # Τοποθέτηση διάστασης ανάμεσα στα διατρήματα
                        else:
                            x1=self.corner_holes[0]+(i-2)*self.spacing[0]
# Τα ενδιάμεσα διατρήματα με κανονική απόσταση διατρημάτων Lifters
                            y1=0
                            ax.annotate(self.spacing[0],(x1+self.spacing[0]/2,y1+0.1))
# Τοποθέτηση διάστασης ανάμεσα στα διατρήματα
            plt.scatter(x1,y1,color='navy')
# Απεικόνιση διατρημάτων σαν κουκίδες στο διάγραμμα

    #ROOF HOLES

        a1= self.width/2
        b1 =(self.abutmentH+self.archHeight)-(self.abutmentH-self.practicalb[5])-
self.practicalb[5] # Δημιουργία κυκλικού τμήματος παράλληλου της οροφής , που απεικονίζει
το πρακτικό φορτίο της οροφής
        x01 =self.width/2
        y01 =self.abutmentH-self.practicalb[5]
        x1 = np.linspace(-a1 + x01, a1 + x01)
        y1 = b1 * np.sqrt(1 - ((x1-x01) / a1) ** 2) + y01
        ax.plot(x1,y1,color='black')
        r=self.holes[5]
# Αριθμός διατρημάτων οροφής
```

```
        for i in range(0,r):
# Τοποθέτηση διατρημάτων οροφής κατά την απόσταση διατρημάτων οροφής στο κυκλικό τμήμα
            xr = np.linspace(-a1 + x01, a1 + x01,r)
            yr=(self.archHeight * np.sqrt(1 - ((xr-x0) / a) ** 2) + self.abutmentH)
            plt.scatter(xr,yr,color='brown')
# Απεικόνιση
            x2 = self.width/2
            y2=(self.abutmentH+self.archHeight)+0.1
            ax.annotate(self.spacing[1],(x2,y2))
# Τοποθέτηση απόστασης διατρημάτων πάνω από τα διατρήματα οροφής

    #WALL HOLES

        xW1=[self.practicalb[6],self.practicalb[6]]
# Συντεταγμένες πρακτικού φορτίου διατρημάτων τοίχου
        yW1=[0+self.practicalb[4],b1 * np.sqrt(1 - ((self.practicalb[6] - x01) / a1) ** 2)
+ y01]
        plt.plot(xW1,yW1, color='black')
        xW2=[self.width-self.practicalb[6],self.width-self.practicalb[6]]
# Οι ίδιες συντεταγμένες , για τον δεξιό τοίχο του μετώπου
        yW2=[0+self.practicalb[4],b1 * np.sqrt(1 - ((self.practicalb[6] - x01) / a1) ** 2)
+ y01]
        plt.plot(xW2,yW2,color='black')
        w=self.holes[6]//2
# Αριθμός διατρημάτων τοίχου δια 2. Για τον κάθε τοίχο δηλαδή
        for i in range(0,w):
            x1=0
            y1=self.practicalb[4]+(i*self.spacing[2])
# Τοποθέτηση διατρημάτων εκατέρωθεν , με την απόσταση διατρημάτων τοίχου
            x2=self.width
            y2=self.practicalb[4]+(i*self.spacing[2])
            plt.scatter(x1,y1,color='green')
            plt.scatter(x2,y2, color='green')
            for i in range(0,w-1):
                x3=0
                y3=self.practicalb[4]+(i*self.spacing[2])
                x4=self.width
                y4=self.practicalb[4]+(i*self.spacing[2])
                ax.annotate(self.spacing[2],(x3+0.1,y3+(self.spacing[2]/2)))
# Τοποθέτηση απόστασης ανάμεσα στα διατρήματα
                ax.annotate(self.spacing[2],(self.width-
self.practicalb[6]/2,y4+(self.spacing[2]/2)))# Τοποθέτηση απόστασης ανάμεσα στα διατρήματα

    #CUT HOLES

        if self.abutmentH>=4.5 or self.abutmentH+self.archHeight>=4.5:
            xQ=(0+self.practicalb[6]+self.B[3]/2)
# Συντεταγμένες αρχής cut (τέρμα  αριστερά δίπλα απο το πρακτικό φορτίο του τοίχου)
            yQ=(0+self.practicalb[4]+self.B[3]/2)
# Ορίζεται αρχικό σημείο το κενό διάτρημα που βρίσκεται στο μέσο του cut.

# Συντεταγμένη ψ υπολογίζεται από το τέλος του πρακτικού φορτίου των lifters και το μισό
από την πλευρά του 4ου τετραγώνου
            plt.scatter(xQ,yQ)
# Συντεταγμένη χ υπολογίζεται από το τέλος του πρακτικού φορτίου του τοίχου και το μισό από
την πλευρά του 4ου τετραγώνου

            x1=[xQ-
(round(self.B[0],2)*math.cos(0.25*math.pi)),xQ,xQ+(round(self.B[0],2)*math.cos(0.25*math.pi
)),xQ,xQ-(round(self.B[0],2)*math.cos(0.25*math.pi))]
            y1=[yQ,yQ+(round(self.B[0],2)*math.sin(0.25*math.pi)),yQ,yQ-
(round(self.B[0],2)*math.sin(0.25*math.pi)),yQ]

            # Με αρχικές συντεταγμένες τα xQ, yQ, υπολογίζονται οι συντεταγμένες του κάθε
σημείου του τετραγώνου
```

```
            # Ο υπολογισμός γίνεται με τα συνημίτονα και τα ημίτονα των γωνιών και τις
αντίστοιχες πλευρές του κάθε τετραγώνου self.B

            x2=[(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[1],2),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[1],2),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.cos(0.25*math.pi)))]

y2=[(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[1],2),(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[1],2),(yQ+((round(self.practicalb[1],2)+(round(self.B[0],2)/2))*math.sin(0.25*
math.pi)))]

            x3=[xQ-
(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),xQ,xQ+(round(self.practicalb[2],2)+(ro
und(self.B[1],2)/2)),xQ,xQ-(round(self.practicalb[2],2)+(round(self.B[1],2)/2))]
            y3=[yQ,yQ+(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),yQ,yQ-
(round(self.practicalb[2],2)+(round(self.B[1],2)/2)),yQ]


            x4=[(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[3],2),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[3],2),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.cos(0.25*math.pi)))]

y4=[(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[3],2),(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[3],2),(yQ+((round(self.practicalb[3],2)+(round(self.B[2],2)/2))*math.sin(0.25*
math.pi)))]

            plt.scatter(x1,y1,color='olive')
            plt.scatter(x2,y2,color='olive')
            plt.scatter(x3,y3,color='olive')
            plt.scatter(x4,y4,color='olive')
            plt.plot(x1,y1,color='black')
# Δημιουργία ευθειών ανάμεσα στα διατρήματα για την απεικόνιση τετραγώνων!!!
            plt.plot(x2,y2,color='black')
            plt.plot(x3,y3,color='black')
            plt.plot(x4,y4,color='black')
            ax.annotate(self.B[2],(xQ-(x3[3]-x3[2]/2),y3[4]-y3[3]/2))
            ax.annotate(self.B[3],(x4[0]+self.B[3]/2,y4[0]))
# ΄Οπου annotate, η απεικόνιση της πλευράς του κάθε τετραγώνου!!!
            ax.annotate(self.B[1],(xQ,yQ+self.B[1]/2))
            ax.annotate(self.B[0],(xQ-(x1[2]-x1[1]),yQ-(y1[1]-y1[2])))
            ax.annotate(self.emptyf,(xQ,yQ),color='blue')

        else:
            xQ=(0+self.practicalb[6]+self.B[2]/2)
# Συντεταγμένες αρχής cut (τέρμα  αριστερά δίπλα απο το πρακτικό φορτίο του τοίχου)
            yQ=(0+self.practicalb[4]+self.B[2]/2)
# Ορίζεται αρχικό σημείο το κενό διάτρημα που βρίσκεται στο μέσο του cut.
```

```
            x1=[(xQ-((round(self.practicalb[0],2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))+round(self.B[0],2),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))+round(self.B[0],2),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[0],2))*math.cos(0.25*math.pi)))]

y1=[(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi))),(yQ+((round(self.practicalb
[0],2))*math.sin(0.25*math.pi))),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi))
)-round(self.B[0],2),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi)))-
round(self.B[0],2),(yQ+((round(self.practicalb[0],2))*math.sin(0.25*math.pi)))]

            x2=[xQ-
(round(self.B[1],2)*math.cos(0.25*math.pi)),xQ,xQ+(round(self.B[1],2)*math.cos(0.25*math.pi
)),xQ,xQ-(round(self.B[1],2)*math.cos(0.25*math.pi))]
            y2=[yQ,yQ+(round(self.B[1],2)*math.sin(0.25*math.pi)),yQ,yQ-
(round(self.B[1],2)*math.sin(0.25*math.pi)),yQ]

            x3=[(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[2],2),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))+round(self.B
[2],2),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi))),(xQ-
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.cos(0.25*math.pi)))]

y3=[(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi))),(yQ+
((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi))),(yQ+((round(
self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*math.pi)))-
round(self.B[2],2),(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*
math.pi)))-
round(self.B[2],2),(yQ+((round(self.practicalb[2],2)+(round(self.B[1],2)/2))*math.sin(0.25*
math.pi)))]

            plt.scatter(x1,y1,color='olive')
            plt.scatter(x2,y2,color='olive')
            plt.scatter(x3,y3,color='olive')
            plt.scatter(xQ,yQ)
            plt.plot(x1,y1,color='black')
# Δημιουργία ευθειών ανάμεσα στα διατρήματα για την απεικόνιση τετραγώνων!!!
            plt.plot(x2,y2,color='black')
            plt.plot(x3,y3,color='black')
            ax.annotate(self.B[2],(xQ,yQ+self.B[2]/2))
# ΄Οπου annotate, η απεικόνιση της πλευράς του κάθε τετραγώνου!!!
            ax.annotate(self.B[1],(xQ,yQ+self.B[1]/2))
            ax.annotate(self.B[0],(xQ-(x1[2]-x1[1]),yQ-(y1[1]-y1[2])))
            ax.annotate(self.emptyf,(xQ,yQ),color='blue')
# Απεικόνιση διαμέτρου αρχικού κενού διατρήματος

    #STOPING HOLES

        xS=0+self.practicalb[6]+self.B[3]
# Αρχικές συντεταγμένες για τα διατρήματα διεύρυνσης , στο πάνω δεξιά άκρο του 4ου
τετραγώνου του cut
        yS=0+self.practicalb[4]+self.B[3]
        c1=self.No_columns[0]
# Αριθμός στηλών οριζοντίων διατρημάτων
        r1=self.No_rows[0]
# Αριθμός σειρών οριζοντίων διατρημάτων
        for i in range(0,c1):
            xh=xS+((i+1)*self.spacing_columns[0])
# Συντεταγμένες κάθε στήλης οριζοντίων διατρημάτων
            yh=yS
            plt.scatter(xh,yh,color='red')
            for z in range(0,r1):
# Συντεταγμένες κάθε σειράς οριζοντίων διατρημάτων
```

```
                xh1=xh
                yh1=yS-((z)*self.spacing_rows[0])
                plt.scatter(xh1,yh1,color='red')
            plt.scatter(xh,yh,color='red')
        for i in range(0,c1):
# Απεικόνιση αποστάσεων γραμμών/στηλών οριζοντίων διατρημάτων
            x1=xS+(i*self.practicalb[7])
            y1=yS
            if i==0:

ax.annotate(self.spacing_columns[0],(x1+self.spacing_columns[0]/2,y1),color='blue')
        for z in range(0,r1-1):
            x2=xS+c1*self.spacing_columns[0]
            y2=yS-z*(self.spacing_rows[0])
            if z==0:
                ax.annotate(self.spacing_rows[0],(x2,y1-
self.spacing_rows[0]/2),color='black')
        xS1=0+self.practicalb[6]
# Αρχικές συντεταγμένες για τα κάθετα διατρήματα διεύρυνσης, το πάνω αριστερό άκρο του 4ου
τετραγώνου του cut
        yS1=0+self.practicalb[4]+self.B[3]
        r2=self.spacing_columns[1]
        c2=self.cols_downwards[0]
        for i in range(0,c2):
            XS=xS1+(i*r2)
            YS=0+self.practicalb[4]+self.B[3]
            a3= self.width/2
            b3 =(self.abutmentH+self.archHeight)-(self.abutmentH-self.practicalb[5])-
self.practicalb[5]
            x03 =self.width/2
            y03 =self.abutmentH-self.practicalb[5]
            x3 = XS
            y3 = b3 * np.sqrt(1 - ((x3-x03) / a3) ** 2) + y03
            d=int((y3-YS)/(self.practicalb[8]))
            for i in range(0,c2-1):
                x4=0+self.practicalb[6]+(i*self.spacing_columns[1])
                y4=0+self.practicalb[4]+self.B[3]
                if i==0:

ax.annotate(self.spacing_columns[1],(x4+self.spacing_columns[1]/2,y4+self.practicalb[8]+0.1
),color='black')
                    x6=0+self.practicalb[6]
                    y6=0+self.practicalb[4]+self.B[3]

ax.annotate(self.practicalb[8],(x6,y6+(self.practicalb[8]/2)),color='blue')
                if d==1 and d*self.practicalb[8]>=y3-YS-0.1:
                    XS1=XS
                    YS1=y3
                    plt.scatter(XS1,YS1,color="red")
                else:
                    for p in range(0,d):
                        XS1=XS
                        YS1=YS+((p+1)*self.practicalb[8])
                        plt.scatter(XS1,YS1,color='red')
                        x6=0+self.practicalb[6]
                        y6=0+self.practicalb[4]+self.B[3]
            if d*self.practicalb[8]<y3-YS-0.1:
                XS2=XS
                YS2=y3
                plt.scatter(XS2,YS2,color='red')
        plt.show()


# Το πρακτικό φορτίο οροφής είναι της μορφής καμπύλης, επομένως ο κώδικας πρέπει να
διακρίνει την απόσταση μεταξύ του κάθε σημείου της καμπύλης με το αντίστοιχο του τετραγώνου
προεκσκαφης και να υπολογίζει με ακρίβεια τα διατρήματα και τις αποστάσεις τους.
```

```
# Για παράδειγμα, το τμήμα πάνω από το τετράγωνο προεκσκαφής ίσως απαιτεί λιγότερα
διατρήματα σε σχέση με το κεντρικό τμήμα της διάτρησης. Γιατί εκεί η καμπύλη πρακτικού
φορτίου της οροφής είναι πιο χαμηλά

#ΕΚΚΙΝΗΣΗ ΚΩΔΙΚΑ

import json
import math
# Ακολουθούν οι βιβλιοθήκες που απαιτώνται
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import numpy as np
import pandas as pd
from tabulate import tabulate
import random
secure_random = random.SystemRandom()
#για να μη βγάζει ψευδοτυχαίους
from scipy.optimize import curve_fit
#για την καμπύλη της οροφής

explosives = Blaster()
# Ο όρος explosives «διαβάζει» την κλάση Blaster()του προγράμματος


with open("config.json" , 'r') as f:
# Αρχείο .json, καταχώρηση σταθερών τιμών
            json_str = f.read()
            json_value = json.loads(json_str)
            explosives.name = json_value["name"]
# Αντιστοίχιση στοιχείων του προγράμματος με τις τιμές που δηλώνονται στο .json
            explosives.hdiam =json_value["geometry"]["hdiam"]
            explosives.emptyf =json_value["geometry"]["emptyf"]
            explosives.width =json_value["geometry"]["width"]
            explosives.abutmentH =json_value["geometry"]["abutmentH"]
            explosives.archHeight =json_value["geometry"]["archHeight"]
            explosives.g =json_value["geometry"]["g"]
            explosives.a =json_value["geometry"]["a"]
            explosives.b =json_value["geometry"]["b"]
            explosives.Q =json_value["geometry"]["Q"]
            explosives.V =json_value["geometry"]["V"]
            explosives.Density =json_value["geometry"]["Density"]
            explosives.c =json_value["geometry"]["c"]
            for x in json_value["cartridges"]:
                explosives.addcartridge(x["name"], x["diameter"], x["length"])
# Για κάθε φυσίγγιο καταγραφή ονόματος, διαμέτρου, μήκους
            explosives.charge_concentration()
            explosives.calculation()
            explosives.first_quadrangle()
            explosives.second_quadrangle()
# Για να «τρέξει» τις διάφορες συναρτήσεις def() του προγράμματος.
            explosives.third_quadrangle()
            explosives.fourth_quadrangle()
            explosives.lifters()
            explosives.roof()
            explosives.wall()
            explosives.stoping()
            explosives.sum()
            explosives.table1()
            explosives.table2()
            #explosives.case_study2()
# Παράδειγμα τετραγωνικής στοάς - Αποτελέσματα
            explosives.plot()
```

```
}
```