ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΙΙ: ΑΝΑΛΥΣΗΣ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΑΝΑΠΤΥΞΗΣ ΔΙΕΡΓΑΣΙΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΑΔΙΚΤΥΑΚΗ ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ ΣΕ ΣΥΣΤΟΙΧΙΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΦΑΡΜΟΓΗ ΣΤΗΝ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΠΕΠΕΡΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ

Διπλωματική εργασία του Γιώργου – Αλέξανδρου Πετρίδη

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΑΝΔΡΕΑΣ Γ. ΜΠΟΥΝΤΟΥΒΗΣ

ΑΘΗΝΑ 2014

Περίληψη

Σκοπός της εργασίας αυτής είναι η παρουσίαση και η ανάλυση τόσο του σχεδιασμού όσο και υλοποίησης της κατασκευής μιας διαδικτυακής διεπαφής για την ευκολία μελέτης και παρουσίασης μιας βιβλιοθήκης επίλυσης προβλημάτων πεπερασμένων στοιχείων με παράλληλη επεξεργασία σε συστοιχία υπολογιστών.

Στην εργασία παρουσιάζονται οι μέθοδοι επικοινωνίας μεταξύ μιας διαδικτυακής διεπαφής και μιας εφαρμογής γραμμής εντολών σε linux, με ειδικότερη αναφορά στις δυσκολίες που παρουσιάζονται από την επιπλέον πολυπλοκότητα μιας συστοιχίας υπολογιστών καθώς και η εφαρμογή αυτών σε συγκεκριμένη υλοποίηση.

Αναλύεται ο κώδικας της διεπαφής για την εισαγωγή παραμέτρων από τον χρήστη και ειδικότερα για το περίπλοκο πρόβλημα της κατανομής ενός μεγάλων διαστάσεων πλέγματος στοιχείων σε ομάδες.

Τέλος, παρουσιάζεται ο κώδικας της διεπαφής για την οπτική παρουσίαση των αποτελεσμάτων και ειδικότερα για τον σχεδιασμό διαγραμμάτων.

Abstract

The purpose of this thesis is the presentation and analysis of the design, implementation and construction of a web interface to assist the studying and presentation of the results of a library containing finite element solvers for parallel processing in computational clusters.

Methods of communication between the web interface and a command line linux application are presented, focusing on the specific difficulties that arise due to the added complexity of a computational cluster. Furthermore the implementation of them in a specific web interface is analyzed.

The code of the web interface for the user input of the execution variables, focusing on the complex problem of a graphical input of the division of a large grid into groups, is analyzed.

Finally, the code of the web interface used to visually present the results, focusing on the construction of graphical diagrams is presented.

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα Καθηγητή Ανδρέα Γ. Μπουντουβή για την ευκαιρία που μου έδωσε να πραγματοποιήσω τη διπλωματική μου εργασία σε συνεργασία με τον ίδιο και την ερευνητική του ομάδα, καθώς και για το χρόνο που διέθεσε αλλά και για τις πολύτιμες συμβουλές και διορθώσεις κατά τη συγγραφή της εργασίας.

Ευχαριστώ τον Δρα Αντώνη Σπυρόπουλο για την συνεχή βοήθεια που μου παρείχε στην κατανόηση της λειτουργίας της συστοιχίας υπολογιστών, της βιβλιοθήκης FeBUI και των αλγορίθμων αυτής. Η συνεργασία μαζί του στον σχεδιασμό και την εκπόνηση της εργασίας αυτής καθώς και οι συμβουλές του και ο χρόνος που μου προσέφερε ήταν πολύ σημαντικές για την ολοκλήρωση της εργασίας αυτής.

Τέλος ευχαριστώ τους γονείς μου, τον αδερφό μου αλλά και την αρραβωνιαστικιά μου για την υπομονή και την υποστήριξη που μου έδειξαν.

Η παρούσα εργασία ήταν εφικτή χάρη στην διαθεσιμότητα της συστοιχίας «Πήγασος» του υπολογιστικού κέντρου της Σχολής Χημικών Μηχανικών του Ε.Μ.Π.

Πίνακας Περιεχομένων

1. Εισαγωγή

1.1. Συστοιχίες Υπολογιστών

Η ραγδαία εξέλιξη των υπολογιστών τις τελευταίες δεκαετίες έχει οδηγήσει σε συστήματα υψηλών αποδόσεων με μικρό κόστος. Ταυτόχρονα η εξέλιξη των δικτύων επικοινωνίας υπολογιστών τόσο σε χωρητικότητα (bandwidth) όσο και στην απόκριση (latency) οδήγησαν στην δημιουργία των συστοιχιών υπολογιστών. Μια συστοιχία υπολογιστών είναι ένας εικονικός (virtual) υπολογιστής πολλαπλών επεξεργαστών που αποτελείται από ένα σύνολο υπολογιστών διασυνδεδεμένων σε ένα αποκλειστικό (private) δίκτυο υψηλής ταχύτητας [**23**].

Ο εικονικός αυτός υπολογιστής χρησιμοποιεί συνήθως μια τροποποιημένη έκδοση του λειτουργικού συστήματος linux [**20**] και λειτουργεί σαν ένας υπολογιστής πολλαπλών επεξεργαστών κατανεμημένης μνήμης, δηλαδή ο κάθε επεξεργαστής χρησιμοποιεί την δικιά του μνήμη και δεν έχει πρόσβαση στην μνήμη των υπολοίπων.

Ο σχεδιασμός της διεπαφής έγινε στην συστοιχία Πήγασος (Pegasus) της Σχολής Χημικών Μηχανικών του Εθνικού Μετσόβιου Πολυτεχνείου [**14**]. Ο Πήγασος αποτελείται από 16 υπολογιστές 2 επεξεργαστών ο καθένας (συνολικά 32 επεξεργαστές) συνδεδεμένους με 2 δίκτια: ένα myrinet [**12**] για την επικοινωνία μεταξύ των επεξεργαστών κατά την διάρκεια εκτέλεσης μιας εφαρμογής και ένα gigabit Ethernet για την επικοινωνία των κόμβων του πλέγματος με τον υπολογιστή διεπικοινωνίας. Το σύστημα τρέχει το λειτουργικό σύστημα Rocks 1.4 [**16**].

1.2. Η Βιβλιοθήκη FeBUI

Η βιβλιοθήκη FeBUI είναι μια βιβλιοθήκη γραμμένη σε Fortran 90 για την εκτέλεση αλγόριθμων επίλυσης με την χρήση ενός υπόχωρου Krylov [**17**]. Η βιβλιοθήκη FeBUI χρησιμοποιεί τις βιβλιοθήκες BLAS [**2**], LAPACK [**9**] και MPICH [**11**] ή LAM/MPI [**8**] και περιέχει μια παράλληλη έκδοση του αλγόριθμου επίλυσης GMRES(m). Πέραν του αλγόριθμου επίλυσης, η βιβλιοθήκη περιέχει και έναν αλγόριθμο προσταθεροποίησης καθώς και διαδικασίες καταμερισμού των στοιχείων σε επεξεργαστές [**17**].

1.2.1. Η μέθοδος GMRES

Η μέθοδος GMRES είναι μία επαναληπτική μέθοδος για την επίλυση μεγάλων αλγεβρικών συστημάτων με την χρήση υπόχωρων Krylov. Η GMRES επιχειρεί να προσεγγίσει την λύση του συστήματος:

$$Ax = b \tag{1.1}$$

όπου $A \in \mathbb{R}^{N\times N}$ και $x, b \in \mathbb{R}^N$, χρησιμοποιώντας μια αρχική εκτίμηση της λύσης x_0. Χρησιμοποιώντας τον αλγόριθμο Arnoldi, συνδυασμένο με την μέθοδο Gram – Schmidt, κατασκευάζεται μια ορθοκανονική βάση $V_m \in \mathbb{R}^{N\times m}$ του υπόχωρου Krylov:

$$K_m(A, u) = span\{u, Au, A^2u, \cdots, A^{m-1}u\} \tag{1.2}$$

όπου $v = \frac{r_o}{\|r_o\|_2}$ και $r_o = b - Ax_0$. Ταυτόχρονα υπολογίζεται το διάνυσμα y_m που ελαχιστοποιεί την:

$$\|\beta e_1 - \bar{H}_m y_m\|_2 \tag{1.3}$$

όπου $e_1 = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}$, $\beta = \|r_o\|_2$ και $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ είναι ένας πίνακας Hessenberg εκ του οποίου προκύπτει ο $H_m = V_m^T A V_m, H_m \in \mathbb{R}^{m\times m}$ διαγράφοντας την τελευταία του γραμμή. Έτσι υπολογίζεται η νέα προσέγγιση της λύσης :

$$x_m = x_0 + V_m y_m \tag{1.4}$$

Η GMRES(m) είναι μια παραλλαγή της GMRES που μετά από m επαναλήψεις επανεκκινεί χρησιμοποιώντας ως x_0 την τελευταία προσέγγιση της λύσης. Αυτή η μέθοδος χρησιμοποιείται συχνά για να αποφευχθεί η μεγάλη απαίτηση της μεθόδου Arnoldi σε μνήμη και επεξεργαστική ισχύ μετά από πολλαπλές επαναλήψεις. Κατασκευάζεται έτσι ένας αλγόριθμος με 2 είδη επαναλήψεων: τις εσωτερικές επαναλήψεις, δηλαδή τις επαναλήψεις που λαμβάνουν τόπο εντός μια εκτέλεσης της GMRES, και τις εξωτερικές επαναλήψεις που αντιπροσωπεύουν το πλήθος των επανεκκινήσεων της GMRES.

1.2.2. Προσταθεροποίηση στην βιβλιοθήκη FeBUI

Για την επιτάχυνση της σύγκλισης της μεθόδου επίλυσης η βιβλιοθήκη FeBUI χρησιμοποιεί έναν αλγόριθμο προσταθεροποίησης μετατρέποντας το αρχικό σύστημα (1.1) σε ένα ισοδύναμο με ευνοϊκότερες προς την σύγκλιση φασματικές ιδιότητες. Το FeBUI χρησιμοποιεί τον προσταθεροποιητή $M^{-1} \in \mathbb{R}^{N\times N}$ που υπολογίζεται με την απομάκρυνση ιδιοτιμών από τον πίνακα A με την χρήση της ορθοκανονικής βάσης $U \in \mathbb{R}^{N\times r}$ του υπόχωρου P_r που αντιστοιχεί στις r μικρότερες τιμές των ιδιοτιμών του A σύμφωνα με την:

$$M^{-1} = I_N + U(|\mu|T^{-1} - I_r)U^T \tag{1.5}$$

όπου $T = U^T AU$, $\mu \in \mathbb{R}$ και είναι η μεγαλύτερη ιδιοτιμή του πίνακα A, $I_N \in \mathbb{R}^{N \times N}$ και $I_r \in \mathbb{R}^{r \times r}$ ιδιοδιανύσματα. Επειδή η διαδικασία της προσταθεροποίησης έχει μεγάλες απαιτήσεις επεξεργαστικής ισχύος και μνήμης, ορίζεται ένα ανώτατο όριο στο πλήθος τον ιδιοτιμών r για τον έλεγχο της διακοπής της διαδικασίας.

1.2.3. Κλήση βιβλιοθήκης FeBUI

Η κλήση της βιβλιοθήκης FEBUI γίνεται με την χρήση αρκετών ορισμάτων. Για την συγκεκριμένη εφαρμογή χρησιμοποιήθηκαν τα ακόλουθα:

- **Nprocs.** Το πλήθος των προς χρήση επεξεργαστών. (integer)
- **NN_global.** Το πλήθος των κόμβων του πλέγματος (nodes) του προς επίλυση συστήματος. (integer)
- **NE_global**. Το πλήθος των στοιχείων του προς επίλυση συστήματος. (integer)
- **MaxNNperElement.** Το μέγιστό πλήθος κόμβων ανά στοιχείο. (integer)
- **NOP_global**. Ένας πίνακας (**NE_global**, **MaxNNperElement**) που περιέχει την αντιστοίχιση των κόμβων του πλέγματος ανά στοιχείο με τους αγνώστους του συστήματος.
- **NNperElem**. Ένας πίνακας (**NE_global**) που περιέχει το πλήθος των κόμβων του πλέγματος ανά στοιχείο.
- **user_sub**. Μία υπορουτίνα (subroutine) που υπολογίζει τις συνεισφορές των στοιχείων και το δεξί μέρος του προς επίλυση γραμμικού συστήματος.
- **xglobal.** Η αρχική εκτίμηση της λύσης του συστήματος.
- **GMRESKrylov**. Το μέγιστο πλήθος των εσωτερικών επαναλήψεων της GMRES(m) . (integer)
- **GMRESIter**. Το μέγιστο πλήθος των εξωτερικών επαναλήψεων της GMRES(m). (integer)
- r_**Deflation**. Το πλήθος των επαναλήψεων προσταθεροποίησης εκφρασμένο σαν ακέραιος.
- **tolgmres** Η ακρίβεια της προσέγγισης της λύση από την μέθοδο GMRES(m). (real)
- **mypart.** Η κατανομή των αγνώστων του συστήματος ανά επεξεργαστήl.
- **use_metis.** Δηλώνει εάν θα γίνει χρήση της βιβλιοθήκης METIS [**10**] για την αυτόματη βέλτιστη κατανομή των αγνώστων ανά επεξεργαστή. (logical)
- **memecheck_u.** Δηλώνει εάν θα γίνει αναφορά της μνήμης που χρησιμοποιήθηκε. (logical)
- **printconv_u.** Δηλώνει εάν θα γίνει εκτύπωση των σφαλμάτων των επαναλήψεων κατά την διάρκεια της επίλυσης. (logical)

1.3. Το Πρόβλημα.

Το πρόβλημα που επιλύει το FeBUI στην εφαρμογή αυτή είναι η αγωγή θερμότητας σε δισδιάστατο πτερύγιο.

Σε μόνιμη κατάσταση η κατανομή θερμοκρασίας, T, σε ένα ορθογώνιο πτερύγιο ικανοποιεί την εξίσωση Laplace:

$$\nabla^2 T(x,y) = 0\,, 0 < x < L, 0 < x < w$$

Συνοριακές συνθήκες επιλέγονται οι εξής:

$x = 0,$	$\frac{\partial T}{\partial x} = 0$	(μόνωση στην αριστερή πλευρά του πτερυγίου)
$x = L,$	$\frac{\partial T}{\partial x} = -h(T - T_o)$	(μεταφορά θερμότητας από την δεξιά πλευρά προς το περιβάλλον)
$y = 0,$	$T = T_1$	(σταθερή θερμοκρασία στην κάτω πλευρά του πτερυγίου)
$y = w,$	$\frac{\partial T}{\partial y} = -h(T - T_o)$	(μεταφορά θερμότητας από την πάνω πλευρά προς το περιβάλλον)

Όπου T_o είναι η θερμοκρασία περιβάλλοντος, T_1 είναι σταθερή θερμοκρασία και h ο συντελεστής μεταφοράς θερμότητας.

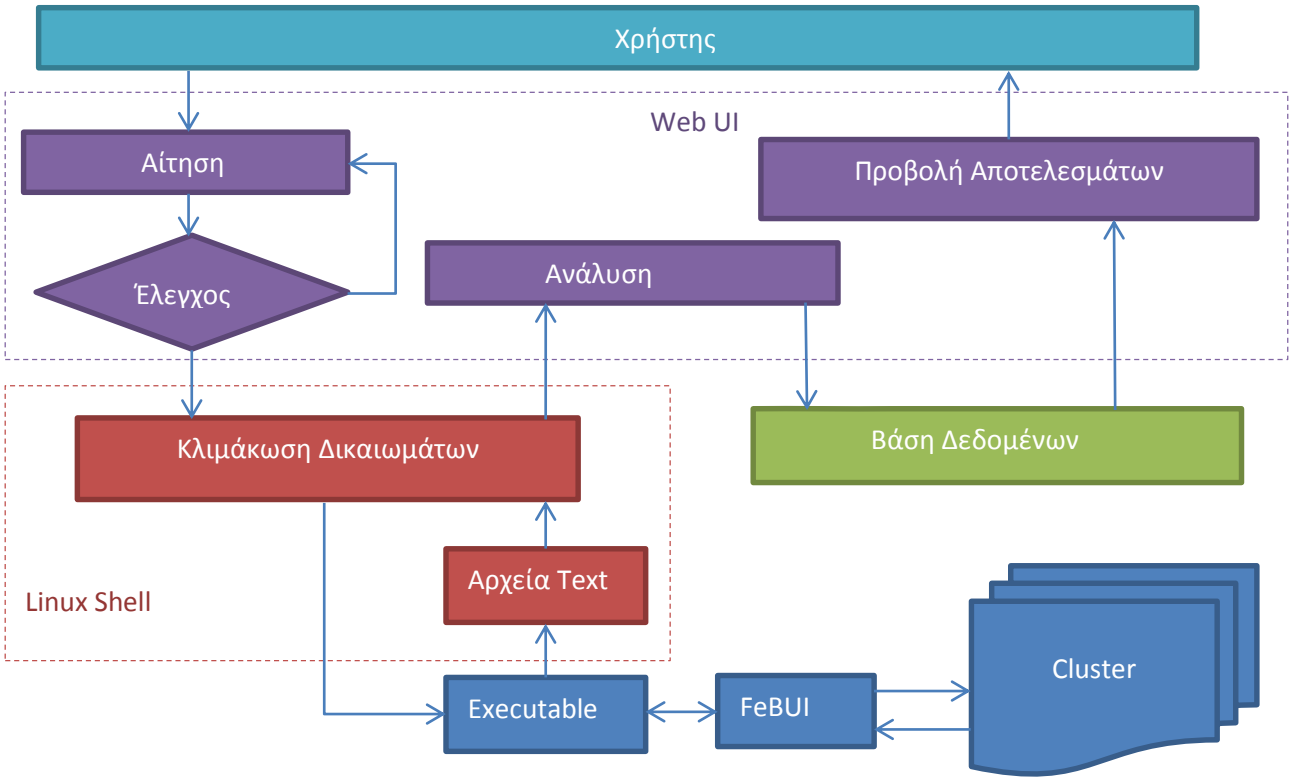
Η επίλυση γίνεται με την χρήση 9 κόμβων ανά στοιχείο του πλέγματος.

2. Δομή Προγράμματος

Το πρόγραμμα αποτελείται από 4 κύρια τμήματα:

- Η διαδικτυακή διεπαφή χρήστη (Web User Interface, Web UI) με την οποία έρχεται σε επαφή ο χρήστης. Το Interface είναι γραμμένο σε php και javascript και αναλαμβάνει την δημιουργία της αίτησης εκτέλεσης, την ανάλυση και αποθήκευση των αποτελεσμάτων στην βάση και την παρουσίασή τους στον χρήστη.
- Ένα πρόγραμμα που καλείται από τη διαδικτυακή διεπαφή χρήστη, το οποίο επιλύει το πρόβλημα πεπερασμένων στοιχείων. Το πρόγραμμα είναι γραμμένο σε fortran και χρησιμοποιεί την βιβλιοθήκη FeBUI για την επίλυση του προβλήματος.
- Μια εφαρμογή-υπηρεσία (service) και μια σειρά από εντολές σε BASH linux shell που χρησιμεύουν για την κλιμάκωση των δικαιωμάτων εκτέλεσης της υπηρεσίας και την επικοινωνία μεταξύ του προγράμματος fortan και της διαδικτυακής διεπαφής χρήστη.
- Και τέλος μία βάση δεδομένων MySQL για την αποθήκευση των δεδομένων των εκτελέσεων.

Αναλυτικά η δομή φαίνεται στο Σχήμα 2-1.



Σχήμα 2-1. Δομή Προγράμματος

2.1. Βάση Δεδομένων

Για την αποθήκευση των δεδομένων επιλέχτηκε η χρήση μιας βάσης δεδομένων MySQL [**13**]. Η χρήση μιας βάσης δεδομένων έχει αρκετά πλεονεκτήματα και αυξάνει τις δυνατότητες της σελίδας. Η ευκολία πρόσβασης των πληροφοριών σε μία βάση επιτρέπει την αποθήκευση και εύκολη εύρεση παλαιότερων εκτελέσεων. Ταυτόχρονα διευκολύνεται ο σχεδιασμός διαγραμμάτων και η ικανότητα ανάλυσης των αποτελεσμάτων και γίνεται εφικτή η σύγκριση 2 ή περισσοτέρων εκτελέσεων.

Ο σχεδιασμός της δομής της βάσης είναι από τα πρωταρχικά και σημαντικότερα βήματα στον σχεδιασμό μιας εφαρμογής που διαχειρίζεται πληροφορίες. Η επιλογή των πεδίων της βάσης καθορίζει τις μεταβλητές του προβλήματος, στην συγκεκριμένη περίπτωση τις παραμέτρους εκτέλεσης της εφαρμογής, καθώς και τα αποτελέσματα που θα παρουσιαστούν, και συνεπώς η σωστή δόμηση τους από την αρχή διευκολύνει τον σχεδιασμό της εφαρμογής και την δημιουργία μιας καθαρότερα δομημένης εφαρμογής.

Η συγκεκριμένη εφαρμογή καλείται να εκτελέσει μία επίλυση με βάση κάποιες παραμέτρους καθώς και να παρουσιάσει τα αποτελέσματα τα της επίλυσης αυτής. Συνεπώς το κύριο αντικείμενο πληροφορίας είναι μια εντολή εκτέλεσης της εφαρμογής. Για κάθε μια εντολή εκτέλεσης υπάρχουν 3 κατηγορίες πληροφοριών:

- Πληροφορίες της εκτέλεσης, όπως αύξων αριθμός, ημερομηνία και ώρα εκτέλεσης κλπ.
- Παράμετροι της επίλυσης.
- Αποτελέσματα της εκτέλεσης.

Ο Πίνακας 2-1 περιέχει αναλυτικά τις μεταβλητές ανά κατηγορία.

Καθότι οι μεταβλητές είναι μονοδιάστατες, με την εξαίρεση των σφαλμάτων της κάθε εκτέλεσης και την αντιστοίχιση των κόμβων του πλέγματος ανά επεξεργαστή, χρησιμοποιήθηκε ένας πίνακας που περιέχει όλες αυτές τις μεταβλητές που ονομάστηκε GMRES_Requests με κλειδί το πεδίο του αύξοντα αριθμού.

Για τα σφάλματα επαναλήψεων χρησιμοποιήθηκε ένας πίνακας με 4 πεδία:

- την αντιστοίχιση με τον αύξοντα αριθμό της εκτέλεσης,
- τον αριθμό της εξωτερικής επανάληψης,
- τον αριθμό της εσωτερικής επανάληψης,
- το σφάλμα της επανάληψης.

Ο πίνακας ονομάστηκε GMRES_Request_Iterations και το πρωτεύον κλειδί αποτελείται από όλα τα πεδία πλην του σφάλματος. Επίσης ορίζεται μία σχέση αναφοράς μεταξύ της αντιστοίχησης με τον αύξοντα αριθμό της εκτέλεσης και το ξένο κλειδί του αύξοντα αριθμού από τον GMRES_Requests.

Μεταβλητή	Είδος	Κατηγορία
Αύξων αριθμός	Ακέραιος	Πληροφορία εκτέλεσης
Ημερομηνία και ώρα εκτέλεσης	Ημερομηνία	Πληροφορία εκτέλεσης
Δείκτης ολοκλήρωσης	Λογικός	Πληροφορία εκτέλεσης
IP από το οποίο εκτελέστηκε η εντολή	Αλφαριθμητικό	Πληροφορία εκτέλεσης
Αριθμός πυρήνων	Ακέραιος	Παράμετρος επίλυσης
Διάσταση Χ	Ακέραιος	Παράμετρος επίλυσης
Διάσταση Υ	Ακέραιος	Παράμετρος επίλυσης
Εξωτερικές επαναλήψεις	Ακέραιος	Παράμετρος επίλυσης
Εσωτερικές επαναλήψεις	Ακέραιος	Παράμετρος επίλυσης
Επαναλήψεις προσταθεροποίησης	Ακέραιος	Παράμετρος επίλυσης
Ανοχή επαναλήψεων	Δεκαδικός	Παράμετρος επίλυσης
Μεθοδολογία αντιστοίχησης κόμβων/επεξεργαστών	Ακέραιος	Παράμετρος επίλυσης
Χρόνος εκτέλεσης	Δεκαδικός	Αποτέλεσμα εκτέλεσης
Πυρήνες εκτέλεσης	Ακέραιος	Αποτέλεσμα εκτέλεσης
Μνήμη που χρησιμοποιήθηκε	Δεκαδικός	Αποτέλεσμα εκτέλεσης
Αριθμός κόμβων	Ακέραιος	Αποτέλεσμα εκτέλεσης
Σφάλματα εκτέλεσης	Αλφαριθμητικό	Αποτέλεσμα εκτέλεσης
Σφάλματα επαναλήψεων	Πίνακας ακεραίων	Αποτέλεσμα εκτέλεσης
Αντιστοίχιση των κόμβων του πλέγματος σε επεξεργαστές	Πίνακας ακεραίων	Αποτέλεσμα εκτέλεσης

Πίνακας 2-1 Μεταβλητές εκτέλεσης προς αποθήκευση στην βάση.

Για την αντιστοίχιση των κόμβων του πλέγματος σε επεξεργαστές χρησιμοποιήθηκε ένας ακόμα πίνακας με 4 πεδία:

- την αντιστοίχιση με τον αύξοντα αριθμό της εκτέλεσης,
- την συντεταγμένη Χ του κόμβου του πλέγματος,
- την συντεταγμένη Υ του κόμβου του πλέγματος,
- ον αριθμό του επεξεργαστή στον οποίο αντιστοιχεί.

Ο πίνακας ονομάστηκε GMRES_Requests_Nodes και το πρωτεύον κλειδί αποτελείται από όλα τα πεδία πλην του αριθμό του επεξεργαστή. Επίσης ορίζεται μία σχέση αναφοράς μεταξύ της αντιστοίχησης με τον αύξοντα αριθμό της εκτέλεσης και το ξένο κλειδί του αύξοντα αριθμού από τον GMRES_Requests.

Αναλυτικά η κατασκευή της βάσης με SQL φαίνεται στο Παράρτημα 1.

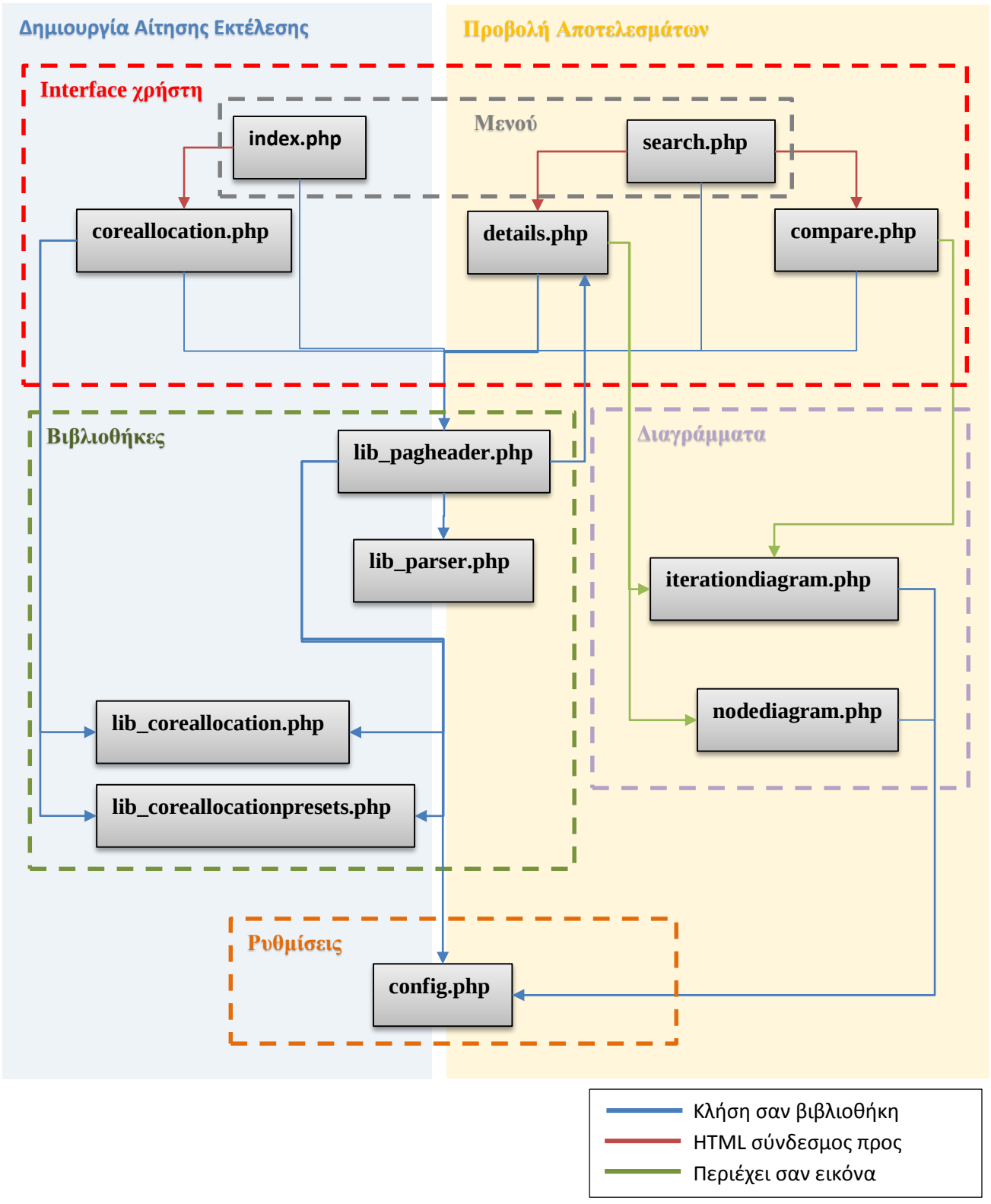
2.2. Διαδικτυακή Διεπαφή Χρήστη

2.2.1. Δομή

Η διαδικτυακή διεπαφή χρήστη αποτελείται από 12 σελίδες php (php pages) [**15**]:

- Μία σελίδα που περιέχει τις ρυθμίσεις της εφαρμογής. Η σελίδα ονομάζεται **“config.php”** και βρίσκεται στον υποφάκελο **“config”**.
- Τέσσερεις σελίδες-βιβλιοθήκες που περιέχουν συναρτήσεις (functions) και τμήματα κώδικα που χρειάζεται να υπάρχει πρόσβαση από πολλαπλές σελίδες. Οι σελίδες αυτές ονομάζονται:
    - **“lib_pageheader.php”** που περιέχει την κεφαλίδα των σελίδων της διεπαφής,
    - **“lib_parser.php”** που περιέχει συναρτήσεις ανάγνωσης των δεδομένων που εξάγει η εφαρμογή,
    - **“lib_coreallocation.php”** που περιέχει συναρτήσεις για την μοντελοποίηση του συστήματος κατανομής κόμβων σε επεξεργαστές,
    - **“lib_coreallocationpresets.php”** που περιέχει προκαθορισμένες μεθόδους κατανομής κόμβων σε επεξεργαστές.
- Δύο σελίδες σχεδιασμού διαγραμμάτων. Οι σελίδες αυτές βρίσκονται στον υποφάκελο **“img”** και ονομάζονται **“nodediagram.php”** και **“iterationdiagram.php”.**
- Και τέλος, πέντε σελίδες της διεπαφής που βλέπει ο χρήστης:
    - **“index.php”** : είναι η αρχική σελίδα της διεπαφής και περιέχει την φόρμα δημιουργίας νέας αίτησης. Είναι προσβάσιμη από το κεντρικό μενού.
    - **“coreallocation.php”** : είναι μια βοηθητική σελίδα που περιέχει την φόρμα σχεδιασμού κατανομής των κόμβων σε επεξεργαστές.
    - **“search.php”** : είναι η σελίδα της διεπαφής που περιέχει την φόρμα εύρεσης παλαιότερων σελίδων. Είναι προσβάσιμη από το κεντρικό μενού.
    - **“details.php”** : είναι η σελίδα προβολής των αποτελεσμάτων και των διαγραμμάτων μιας εκτέλεσης. Καλείται από την **“search.php”** ή κατά την ολοκλήρωση μιας εκτέλεσης.
    - **“compare.php”** : είναι η σελίδα της διεπαφής και περιέχει την φόρμα δημιουργίας νέας αίτησης. Καλείται από την **“search.php”.**

Το διάγραμμα επικοινωνίας μεταξύ αυτών των 12 σελίδων φαίνεται στο Σχήμα 2-2.



Σχήμα 2-2 Διάγραμμα επικοινωνίας μεταξύ των σελίδων της διαδικτυακής διεπαφής χρήστη και κατηγοριοποίησή τους.

2.2.2. Αρχείο ρυθμίσεων

Το αρχείο ρυθμίσεων περιέχει τις πληροφορίες που μοιράζονται οι σελίδες. Το αρχείο αυτό καλείται από όλες τις υπόλοιπες σελίδες κατά την εκκίνησή τους και συνεπώς οι μεταβλητές που ορίζονται σε αυτό είναι προσβάσιμες παντού.

Στο αρχείο ορίζεται μια array η $Settings που περιέχει τις εξής τιμές :

- **SQL_Host –** Η διεύθυνση της βάσης που θα χρησιμοποιηθεί.
- **SQL_db –** Η ονομασία της βάσης δεδομένων που περιέχει τα δεδομένα της σελίδας.
- **SQL_User –** Το όνομα χρήστη της βάσης.
- **SQL_Pass –** Ο κωδικός ασφαλείας του χρήστη στην βάση.
- **Output_path –** Ο φάκελος όπου ο apache έχει δικαίωμα εγγραφής και συνεπώς εκεί θα τοποθετούνται τα αρχεία για την μεταφορά πληροφοριών.
- **Exec –** Το όνομα του εξωτερικού εκτελέσιμου όπως αυτό θα καλεστεί στην εντολή εκτέλεσής του.
- **timeout –** Ο χρόνος σε δευτερόλεπτα μετά από τον οποίο το εξωτερικό εκτελέσιμο πρέπει να τερματιστεί εάν δεν έχει ολοκληρωθεί.
- **timeoutcom –** Το όνομα μια εντολής ή ενός script το οποίο θα χρησιμεύσει για τον τερματισμό του εξωτερικού εκτελέσιμου. Συνήθως τα λειτουργικά linux περιέχουν είτε την εντολή timeout, που κάνει ακριβώς αυτή την λειτουργία, είτε ένα προκατασκευασμένο script. Στην συγκεκριμένη περίπτωση στο Rocks [**16**], υπάρχει ένα προκατασκευασμένο script bash που καλύπτει αυτή την λειτουργία στο φάκελο “/usr/share/doc/”, και έγινε ένα αντίγραφο αυτού στον φάκελο από τον οποίο γίνεται η εκτέλεση. Το script καλείται με την παράμετρο “-INT”, που σημαίνει ότι πρώτα επιχειρεί να στείλει το σήμα SIGINT [**20**], δηλαδή Interrupt και είναι το σήμα που λαμβάνει μια εντολή σε γραμμή εντολών όταν πατηθεί “ctrl+c”, πριν την τερματίσει βίαια.

Επίσης στην σελίδα αυτή γίνεται εκκίνηση της σύνδεσης προς την βάση δεδομένων. Ο κώδικας τις σελίδας φαίνεται στο Σχήμα 2-3.

```php
<?php

$Settings =array
(
        'SQL_Host' => "localhost",
        'SQL_db' => "gmres",
        'SQL_User' => "gpetr",
        'SQL_Pass' => "**********",
        'Output_path' => $_SERVER{'DOCUMENT_ROOT'} . '/WebFEBUI/binout/',
        'Exec' => './a.out',
        'timeout' => 120,
        'timeoutcom' => './timeout -INT '
);
$MYSQLconnection = mysql_connect($Settings['SQL_Host'],$Settings['SQL_User'] ,$Settings['SQL_Pass'] );
if (!$MYSQLconnection) {
   die('Could not connect: ' . mysql_error());
}

mysql_select_db($Settings['SQL_db'], $MYSQLconnection);
?>
```

Σχήμα 2-3 Κώδικας σελίδας ρυθμίσεων

2.2.3. Κεφαλίδα σελίδας

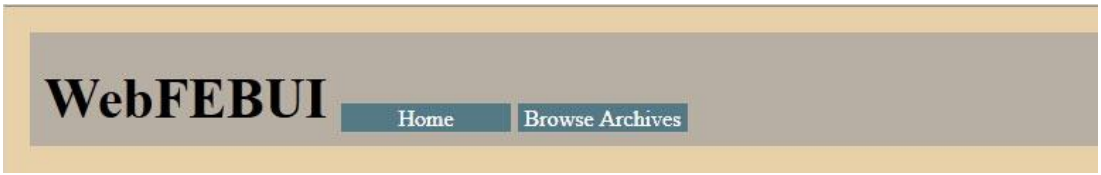
Η σελίδα κεφαλίδας σελίδων είναι μία σελίδα php που επισυνάπτεται στην αρχή κάθε σελίδας που είναι μέρος της διεπαφής με τον χρήστη. Η χρήση αυτή της σελίδας βοηθάει σε 3 σημεία:

- Η σελίδα περιέχει την κλήση βασικών βιβλιοθηκών που είναι απαραίτητο να επισυναφθούν σε όλες τις σελίδες, όπως η σελίδα ρυθμίσεων και η βιβλιοθήκη του javascript jquery.
- Η σελίδα περιέχει το κεντρικό μενού και καλεί το αρχείο μορφοποίησης css ώστε να δοθεί μια αισθητική ομοιογένεια στις σελίδες.
- Τέλος περιέχει τον κώδικα ελέγχου για την ολοκλήρωση μιας τρέχουσας αίτησης ώστε να δοθεί δυνατότητα στον χρήστη να κινείται ελεύθερα στις σελίδες και να μελετάει τα προηγούμενα αποτελέσματα ενώ υπάρχει αναμονή ολοκλήρωσης μιας νέας αίτησης.

Η συγκέντρωση αυτών των πληροφοριών σε μία σελίδα βοηθάει την εύκολη διόρθωση, αλλαγή ή αναβάθμισή τους έτσι ώστε να επηρεάζονται όλες οι σελίδες που βλέπει ο χρήστης.

Η σελίδα μπορεί να χωριστεί σε δύο τμήματα.

Το πρώτο είναι το κομμάτι της σε html το οποίο αναλαμβάνει να σχεδιάσει την κεφαλίδα κάθε σελίδας συμπεριλαμβανόμενου και του μενού, καθώς και να εισάγει το αρχείο μορφοποίησης css και τη βιβλιοθήκη jquery [**7**]. Το οπτικό αποτέλεσμα του κώδικα φαίνεται στο Σχήμα 2-4 ενώ ο HTML κώδικας στο Σχήμα 2-5.



Σχήμα 2-4 Κεφαλίδα σελίδας

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>WebFEBUI</title>
    <link rel="stylesheet" type="text/css" href="style/style.css">
    <script language="javascript" type="text/javascript" src="js/jquery.js"></script>
  </head>
  <body> <center>
<div class="header"><div><table cellpadding="0" cellspacing="0" >
            <tr><td class="headertitle">WebFEBUI</td>
                <td><a href="index.php" >Home</a> </td>
                <td><a href="search.php" >Browse Archives</a></td>
                <td width="100%"></td></tr>
</table> </div> </div>
```

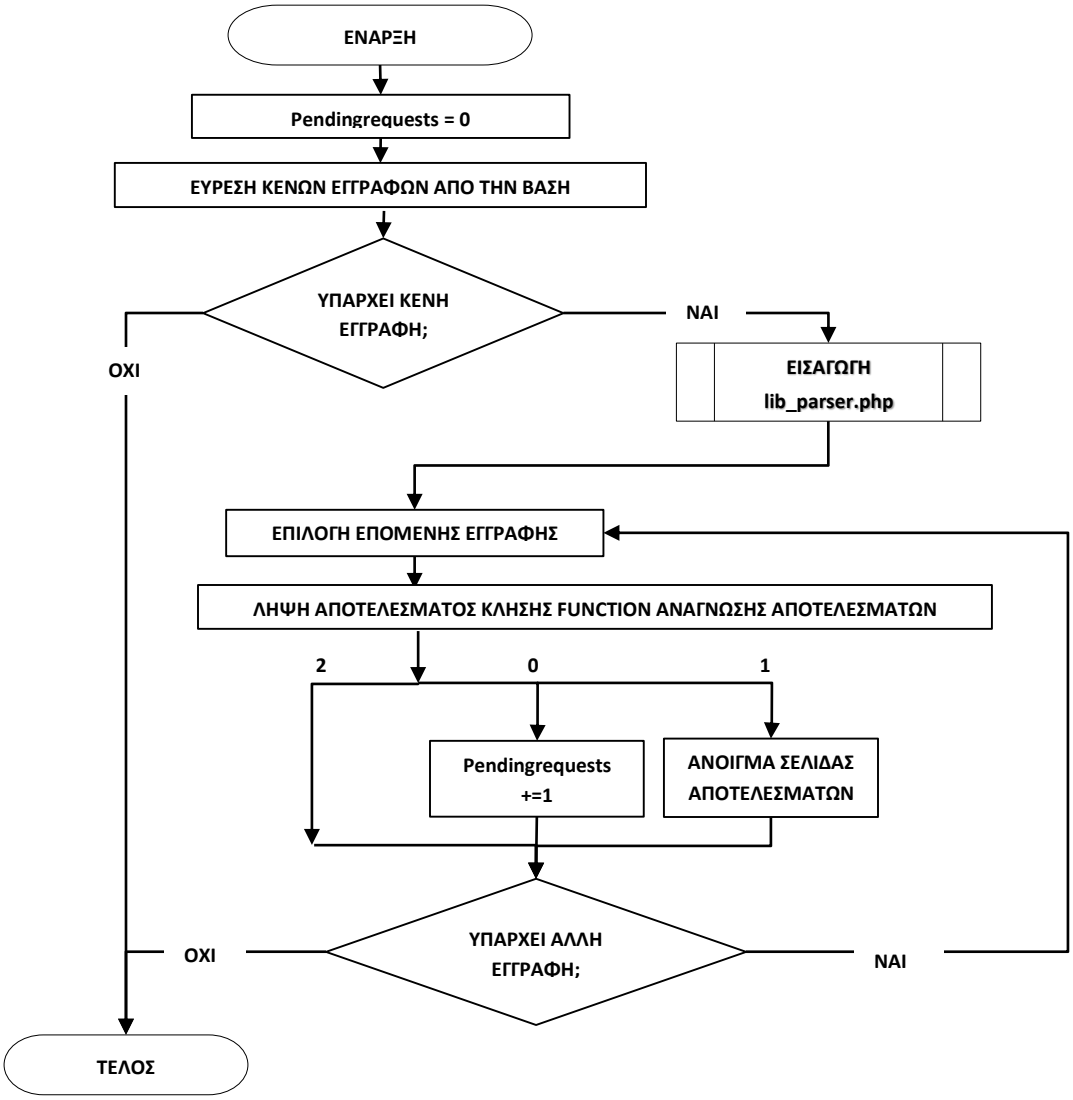
Σχήμα 2-5 Κώδικας HTML κεφαλίδας σελίδας

Το δεύτερο τμήμα της σελίδας είναι η διαδικασία ελέγχου ολοκλήρωσης εγγραφής. Αυτό το κομμάτι κώδικα ελέγχει την βάση για μη ολοκληρωμένες εκτελέσεις. Εφόσον βρει μία, επισυνάπτει στην εφαρμογή την σελίδα-βιβλιοθήκη lib_parser.php που περιέχει τις συναρτήσεις ανάγνωσης των αρχείων αποτελεσμάτων (βλ. Κεφάλαιο 3.3). Από την lib_parser.php καλεί την συνάρτηση parseresult με όρισμα το κλειδί της μη ολοκληρωμένης εγγραφής και με βάση το αποτέλεσμα ενημερώνει τον χρήστη για την κατάσταση της εγγραφής. Περαιτέρω εάν η διαδικασία είχε ολοκληρωθεί ανοίγει τα αποτελέσματα αυτής σε μία άλλη σελίδα, ενώ εάν είναι ακόμα σε αναμονή αποθηκεύει το πλήθος των εγγραφών σε αναμονή σε μία μεταβλητή για να υπάρχει πρόσβαση σε αυτήν από την συνέχεια της κάθε σελίδας.

Ο κώδικας αυτής της διαδικασίας φαίνεται στο Σχήμα 2-6 ενώ το διάγραμμα ροής φαίνεται στο Σχήμα 2-7.

```
$pendingrequests = 0
$result = mysql_query("SELECT * FROM GMRES_Requests WHERE complete=0 ");
$rows = mysql_num_rows($result);
if ($rows > 0) {
        include "lib_parser.php";
        while ($row = mysql_fetch_array($result)) {
           $res = parseresult($row['id'],$Settings);
           if ($res == 1) {
                echo ' Request Complete: ' . $row['id'] . ' <br />';
                ?>
                <script  type="text/javascript"> window.open('details.php?myID=<?php echo $row['id'];
                ?>','_blank');
                </script>
                <?php
           } else {
                if ($res == 0) {
                     echo ' Pending request: ' . $row['id'] . ' <br /> ';
                $pendingrequests += 1;
                 }
           }
        }
}
```

Σχήμα 2-6 Κώδικας κλήσης ελέγχου ολοκλήρωσης εξωτερικής εκτέλεσης.



Σχήμα 2-7 Διάγραμμα ροής κλήσης ελέγχου ολοκλήρωσης εξωτερικής εκτέλεσης.

3. Διεπικοινωνία

Επειδή η βιβλιοθήκη FeBUI είναι υλοποιημένη σε κώδικα Fortran [**4**] δεν είναι δυνατή η απευθείας χρήση της από την σελίδα. Είναι συνεπώς αναγκαία η χρήση ενός εκτελέσιμου αρχείου fortran και ο σχεδιασμός ενός συστήματος επικοινωνίας μεταξύ αυτού και της σελίδας. Η αξιοπιστία είναι από τους σημαντικότερους παράγοντες αυτού του σχεδιασμού, καθώς η σταθερότητά του στην ακριβή μεταφορά δεδομένων από τα εκτελέσιμο στην σελίδα είναι αναγκαία για την μείωση των σφαλμάτων που πιθανόν να παρουσιαστούν κατά την χρήση της σελίδας.

Η κατασκευή αυτού του συστήματος διεπικοινωνίας σε ένα σύστημα συστοιχίας υπολογιστών όπως αυτό που χρησιμοποιήθηκε παρουσιάζει εκτεταμένες δυσκολίες διότι η εκτέλεση εφαρμογών στον cluster απαιτεί την χρήση πρωτόκολλων επικοινωνίας όπως το SSH [**11**]. Ο http server apache, παρότι εκτελείται από τον root, χρησιμοποιεί ένα χρήστη χαμηλών δικαιωμάτων για την διαχείριση των αιτημάτων του καθότι είναι ένας σχετικά ευάλωτος χρήστης και τα αρχεία, στα οποία αυτός έχει δικαίωμα ανάγνωσης, είναι εύκολα προσβάσιμα διαδικτυακά [**1**]. Η χρήση πρωτοκόλλων επικοινωνίας, όπως το SSH, από ευάλωτους χρήστες, είναι σημαντικό πρόβλημα ασφάλειας και απαιτεί προσοχή.

Ταυτόχρονα, το λειτουργικό σύστημα που διαχειρίζεται τη συστοιχία υπολογιστών που χρησιμοποιήθηκε, χρησιμοποιεί διαδικτυακή διεπαφή χρήστη για την προβολή διαγνωστικών πληροφοριών και συνεπώς εκτεταμένες αλλαγές στα δικαιώματα ή τον τρόπο εκτέλεσης του apache μπορεί να προκαλέσουν προβλήματα λειτουργίας στον cluster.

Συνεπώς ο σχεδιασμός του συστήματος διεπικοινωνίας είναι μια αρκετά περίπλοκη διαδικασία καθώς υπάρχει πληθώρα προδιαγραφών σχεδιασμού που πρέπει να ληφθούν υπόψη.

3.1. Επικοινωνία και παραμετροποίηση εκτελέσιμου fortran.

Στην fortran υπάρχουν αρκετές μέθοδοι για την μεταφορά παραμέτρων και μεταβλητών μεταξύ του εκτελέσιμου και του χρήστη ή άλλων εφαρμογών η καθεμία από τις οποίες έχει πλεονεκτήματα και μειονεκτήματα.

3.1.1. Χρήση παραμέτρων γραμμής εντολών

Ένας από τους πλέον πιο διαδεδομένους τρόπους εισαγωγής τιμών μεταβλητών ή διαφόρων επιλογών σε ένα πρόγραμμα είναι διαμέσου παραμέτρων που αναγράφονται μετά την εφαρμογή κατά την κλήση εκτέλεσης της. Αυτή η μεθοδολογία παρουσιάζει αρκετά πλεονεκτήματα. Είναι ευανάγνωστη και φιλική προς τον χρήστη, δεν απαιτεί την χρήση εξωτερικών αρχείων ή εφαρμογών και γίνεται αποσφαλμάτωση (debug) εύκολα, διότι είναι απλό να καταλάβει κάποιος τι εκτελέστηκε γρήγορα με απλή ανάγνωση της εντολής. Το κύριο μειονέκτημα είναι η αδυναμία της να μεταφέρει μεγάλες ποσότητες πληροφορίων, όπως παραδείγματος χάριν ένα μεγάλο πίνακα τιμών.

Στην fortran η ανάκληση των παραμέτρων αυτών γίνεται με την χρήση μίας συνάρτησης, της IARGC() και μίας υπορουτίνας (subroutine), της GETARG() [**18**]. Η IARGC() επιστρέφει σαν ακέραιο τον αριθμό των παραμέτρων που έχουν δοθεί, χωρισμένων με κενό χαρακτήρα. Η GETARG() λαμβάνει δύο ορίσματα. Το πρώτο είναι ακέραιος και αντιστοιχεί στην θέση της παραμέτρου, με την τιμή 0 να αντιστοιχεί στο όνομα του ίδιου του προγράμματος. Το δεύτερο είναι character και αντικαθίσταται η τιμή του με την παράμετρό που έχει επιλεχτεί από το πρώτο όρισμα.

Η εισαγωγή των τιμών στο πρόγραμμα γίνεται με την χρήση μίας επανάληψης από 1 στο IARGC() και την λήψη της τιμής σε κάθε θέση.

3.1.2. Χρήση αρχείων κειμένου

Μία άλλη μεθοδολογία είναι η χρήση αρχείων κειμένου και η γραμμική εισαγωγή των τιμών από αυτό στην εφαρμογή. Αυτή η μέθοδος δύναται να περιέχει μεγάλες ποσότητας πληροφοριών σε αντίθεση με τις παραμέτρους γραμμής εντολών αλλά η ύπαρξη του αρχείου καθώς και η ύπαρξη των τιμών εντός του αρχείου αυξάνει την περιπλοκότητα και καθιστά την διαδικασία λιγότερο φιλική προς τον χρήστη, και συνεπώς η αποσφαλμάτωση γίνεται δυσκολότερη.

Στην fortran η ανάγνωση του αρχείου γίνεται με την χρήση λογικών μονάδων. Η κάθε λογική μονάδα αντιπροσωπεύεται με έναν αριθμό στην fortran. Για τον ορισμό μίας λογικής μονάδας στην fortran χρησιμοποιείται η εντολή open [**3**]. Η εντολή open λαμβάνει πληθώρα ορισμάτων, από τα οποία τα δύο βασικότερα είναι το UNIT και το FILE. Το UNIT υποδηλώνει το αριθμό που θα αντιστοιχεί στην συγκεκριμένη λογική μονάδα και το FILE την ονομασία του αρχείου που η λογική μονάδα αντιπροσωπεύει. (Για περαιτέρω ανάλυση της διαδικασίας βλ. Παράρτημα 2.α.)

3.1.3. Χρήση προκαθορισμένων λογικών μονάδων

Πέραν της χρήσης αρχείων είναι δυνατόν να χρησιμοποιηθούν οι προκαθορισμένες λογικές μονάδες μέσω της χρήσης συμβόλων στην γραμμή εντολών. Οι προκαθορισμένες λογικές μονάδες στα LINUX/UNIX είναι κυρίως η standard input (stdin), που υποδηλώνει την προκαθορισμένη μονάδα εισόδου, η standard output (stdout), που υποδηλώνει την προκαθορισμένη μονάδα εξόδου και η standard error (stderr) που υποδηλώνει την προκαθορισμένη μονάδα εξόδου σφαλμάτων [**21**]. Αυτές οι λογικές μονάδες μπορούν είτε να μεταφερθούν από ένα πρόγραμμα στο άλλο είτε να οριστούν σαν αρχεία με την χρήση ειδικών συμβόλων στην γραμμή εντολών. Για περαιτέρω ανάλυση των συμβόλων και των λειτουργιών τους βλ. Παράρτημα 2.β.)

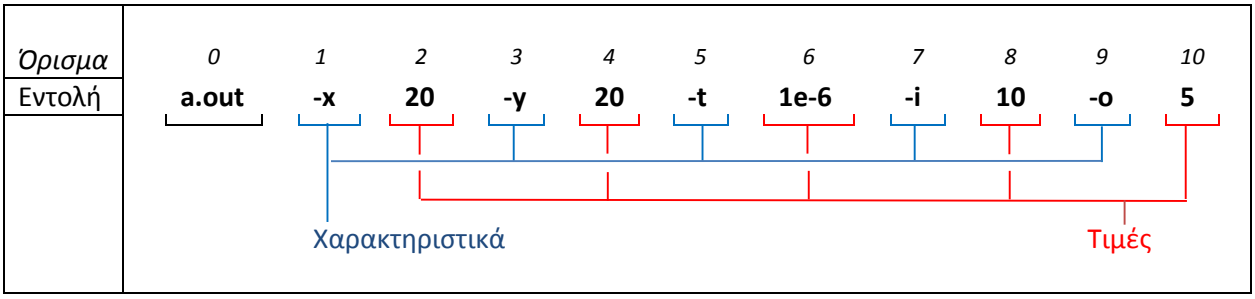
3.1.4. Εφαρμογή

Στην συγκεκριμένη εφαρμογή χρησιμοποιούνται αρκετές μεταβλητές εισόδου και εξόδου για το εκτελέσιμο αρχείο fortran. Για κάθε μία από αυτές χρησιμοποιήθηκε η μεθοδολογία που ταιριάζει περισσότερο. Για τις μεταβλητές που είναι πίνακες τιμών χρησιμοποιήθηκαν αρχεία κειμένου καθότι δεν είναι εύχρηστη η μεταφορά μεγάλων ποσοτήτων πληροφοριών με τις άλλες μεθοδολογίες. Στις μεταβλητές εισόδου που είναι μόνο μια τιμή, χρησιμοποιήθηκαν παράμετροι γραμμής εντολών διότι διευκολύνουν την αποσφαλμάτωση και την αναγνωσιμότητα του προγράμματος. Στις μεταβλητές εξόδου που είναι μόνο μια τιμή χρησιμοποιήθηκε το standard output για τους ιδίους λόγους. Αναλυτικά οι μεταβλητές που χρησιμοποιήθηκαν και οι μέθοδοι φαίνονται στον Πίνακας 3-1.

Μεταβλητή	Είδος	Τύπος	Μεθοδολογία
Διάσταση Χ	Εισόδου	Ακέραιος Αριθμός	Παράμετρος γραμμής εντολών
Διάσταση Υ	Εισόδου	Ακέραιος Αριθμός	Παράμετρος γραμμής εντολών
Εσωτερικές Επαναλήψεις	Εισόδου	Ακέραιος Αριθμός	Παράμετρος γραμμής εντολών
Εξωτερικές Επαναλήψεις	Εισόδου	Ακέραιος Αριθμός	Παράμετρος γραμμής εντολών
Επαναλήψεις Προσταθεροποίησης	Εισόδου	Ακέραιος Αριθμός	Παράμετρος γραμμής εντολών
Ανοχή Επαναλήψεων	Εισόδου	Αριθμός	Παράμετρος γραμμής εντολών
Κατανομή κόμβων/επεξεργαστών	Εισόδου/Εξόδου	Πίνακας Αριθμών	Αρχείο κειμένου
Χρησιμοποιημένοι επεξεργαστές	Εξόδου	Ακέραιος Αριθμός	Standard output
Χρόνος εκτέλεσης	Εξόδου	Αριθμός	Standard output
Αριθμός κόμβων	Εξόδου	Ακέραιος Αριθμός	Standard output
Χρήση μνήμης	Εξόδου	Αριθμός	Standard output

Πίνακας 3-1. Μεταβλητές εισόδου και εξόδου στο εκτελέσιμο fortran και μέθοδοι επικοινωνίας.

Για την εισαγωγή των μεταβλητών μέσω των παραμέτρων γραμμής εντολών χρησιμοποιήθηκε η εξής δομή: "-<χαρακτηριστικό γράμμα> <τιμή>" (π.χ. για διάσταση x ίση με 20 εισάγεται "-x 20"). Πολλές διαφορετικές επιλογές χωρίζονται μεταξύ τους με κενό (π.χ. "-x 20 –y 20 –t 1e-6" αντιστοιχεί σε μία εκτέλεση σε πίνακα 20x20 με ανοχή 10^{-6}). Όπως αναφέρθηκε προηγουμένως, οι παράμετροι αυτές λαμβάνονται εντός του προγράμματος μέσω μιας υπορουτίνας χρησιμοποιώντας την θέση της καθεμίας σαν όρισμα, διαχωρισμένες με τον κενό χαρακτήρα. Η συγκεκριμένη δομή οδηγεί στην δημιουργία ενός ιδεατού πίνακα με διαστάσεις από 0 ως το αποτέλεσμα της IARGC(), όπου οι θέσεις με μονό δείκτη χαρακτηρίζουν την μεταβλητή ενώ οι ακριβώς επόμενες μας δίνουν την τιμή τους, όπως φαίνεται στο Σχήμα 3-1.



Σχήμα 3-1. Παράδειγμα ανάλυσης παραμέτρων γραμμής εντολών.

Η λήψη των τιμών αυτών από την εφαρμογή γίνεται με τον εξής αλγόριθμο: Βρίσκεται το πλήθος των παραμέτρων από την IARGC(). Εάν είναι μονός αριθμός διακόπτεται η εφαρμογή με αποτέλεσμα λάθους, ειδάλλως γίνεται μια επανάληψη από 1 μέχρι το πλήθος λαμβάνοντας μόνο τα στοιχεία με μονό δείκτη. Εάν ο κάθε δείκτης αναγνωριστεί αποθηκεύεται η τιμή στην σωστή μεταβλητή στο πρόβλημα, ειδάλλως διακόπτεται η εφαρμογή με αποτέλεσμα λάθους. Ο κώδικας του μηνύματος σφάλματος φαίνεται στο Σχήμα 3-2 και φαίνεται αναλυτικά ο κώδικας εισαγωγής στο Παράρτημα 2.γ.

Η αντιστοίχιση των κόμβων του πλέγματος σε επεξεργαστές γίνεται στο FeBUI μέσω ενός μονοδιάστατου πίνακα. Τα δεδομένα εισάγονται σαν αρχείο με την μορφή *"i␣n↵"* όπου *i* η αρίθμηση όπως αυτή απαιτείται από το όρισμα στο FeBUI και *n* ο αριθμός του επεξεργαστή (η Ανάλυση της μετατροπής φαίνεται στο Παράρτημα 3). Για την εισαγωγή αυτή χρησιμοποιείται μια ακόμα παράμετρος γραμμής εντολών που αντιστοιχεί στο αρχείο που περιέχει την αντιστοίχιση. Εάν η παράμετρος αυτή δεν δοθεί χρησιμοποιείται το METIS, ειδάλλως γίνεται ανάγνωση του αρχείου ανά γραμμή και κάθε τιμή αντιστοιχίζεται στην σωστή θέση του πίνακα-όρισμα.

Η εξαγωγή των ιδίων στοιχείων γίνεται σε ένα αρχείο ονομαζόμενο "*used_nodes*" με την μορφή *"x|y|n↵"* όπου *x,y* οι συντεταγμένες του κόμβου του πλέγματος και *n* ο επεξεργαστής που χρησιμοποιήθηκε.

Οι υπόλοιπες μεταβλητές εξόδου απλώς εξάγονται στο standard output με την χρήση της εντολής "*write(*,*)*".

```
      subroutine badinput()
c bad argument output
          write ( 0, *) 'Error: Invalid Input Arguments'
          write ( 0, *) ' Viable argument options: '
          write ( 0, *) '             -x <NUMBER> : Dimension X '
          write ( 0, *) '             -y <NUMBER> : Dimension Y '
          write ( 0, *) '             -i <NUMBER> : KRYLOV iterations '
          write ( 0, *) '             -o <NUMBER> : outer iterations '
          write ( 0, *) '             -r <NUMBER> : preconditioning '
          write ( 0, *) '             -t <NUMBER> : tolerance '
          write ( 0, *) '             -n <FILE> : nodes file '
      end
```

Σχήμα 3-2 Κώδικας μηνύματος σφάλματος παραμέτρων γραμμής εντολών.

3.2. Κλήση της εξωτερικής εφαρμογής fortran από την σελίδα.

3.2.1. Απευθείας κλήση της εξωτερικής εφαρμογής από php.

Είναι δυνατή η απευθείας κλήση μιας εξωτερικής εφαρμογής από php με την χρήση της subroutine exec ή της συνάρτησης shell_exec [**15**]. Η shell_exec λαμβάνει ως όρισμα την εντολή προς εκτέλεση, την εκτελεί χρησιμοποιώντας τον χρήστη για την διαχείριση των αιτημάτων του apache και επιστρέφει ως τιμή το standard output αυτής. Αντίστοιχά, η subroutine exec λαμβάνει 3 ορίσματα. Το πρώτο είναι η εντολή προς εκτέλεση, την οποία εκτελεί χρησιμοποιώντας τον χρήστη για την διαχείριση των αιτημάτων του apache όπως και πριν. Το δεύτερο είναι προαιρετικό και είναι μια array στην οποία επισυνάπτεται ανά γραμμή το standard output και το τρίτο είναι επίσης προαιρετικό και είναι ένας ακέραιος και επιστρέφει την αριθμητική τιμή αποτελέσματος την εφαρμογής. Παράδειγμα χρήσης των δύο εντολών εμφανίζεται στο Σχήμα 3-3.

```
$standard_output= shell_exec("my_command_line_executable")

$exec("my_command_line_executable", $stadard_ouput_array, $command_exit_code)
```

Σχήμα 3-3 Παράδειγμα χρήσης των exec και shell_exec

Οι δύο αυτές διαδικασίες λαμβάνουν και οι δύο μία εντολή που εκτελείται στο ίδιο μέρος από τον ίδιο χρήστη και επιστρέφουν το standard output με διαφορετική δομή η καθεμία. Αυτό το γεγονός παρουσιάζει τρείς αξιοσημείωτες δυσκολίες:

- Καμία από τις δύο διαδικασίες δεν διαχειρίζεται η standard error άρα, σε περίπτωση που η εντολή εξέλθει με σφάλμα που εμφανίζεται εκεί, δεν είναι δυνατή η εξακρίβωση αυτού.
- Οι διαδικασίες παρουσιάζουν και επιστρέφουν τη standard output. Αυτό υποδηλώνει ότι και οι δυο αυτές διαδικασίες αναμένουν την ολοκλήρωση της εφαρμογής προτού ολοκληρωθούν και αυτές, ώστε να είναι δυνατό να αναγνώσουν την αυτή λογική μονάδα. Καθότι η php εκτελείται κατά την λήψη του αιτήματος μέχρι και την προβολή της σελίδας, αυτό συνεπάγεται ότι η προς εκτέλεση εντολή πρέπει να εκτελεστεί κατά την διάρκεια φόρτωσης της σελίδας, και ο χρόνος εκτέλεσης αυτής προστίθεται στον χρόνο φόρτωσης της σελίδας. Αυτό οδηγεί σε σελίδες με πολύ υψηλό χρόνο φόρτωσης, καθώς η εξωτερική εφαρμογή μπορεί να παρουσιάσει πολύ μεγάλους χρόνους εκτέλεσης.
- Και οι δύο διαδικασίες χρησιμοποιούν τον χρήστη για την διαχείριση των ψ του apache. Ο χρήστης αυτός είναι εύκολα προσβάσιμος και ευάλωτος και για αυτόν το λόγο έχει περιορισμένα δικαιώματα. Ταυτόχρονα όμως, οι clusters σαν αυτόν που χρησιμοποιήθηκε απαιτούν την χρήση πρωτόκολλων επικοινωνίας, όπως το SSH, για την εκτέλεση εφαρμογών, γεγονός που απαιτεί αρκετά υψηλότερα δικαιώματα από τα συνήθη για ένα χρήστη σαν αυτόν που χρησιμοποιείται στην συγκεκριμένη περίπτωση. Η επιλογή της κατάλληλης διαδικασίας κλιμάκωσης των δικαιωμάτων αυτών είναι

σημαντικός παράγοντας για την ασφάλεια του συστήματος καθώς και του γύρω δικτύου του αφού αφορά πρωτόκολλα επικοινωνίας.

3.2.2. Διαχείριση των προκαθορισμένων λογικών μονάδων

Η αδυναμία ανάγνωσης των σφαλμάτων από τις εντολές της php μπορεί να εύκολα προσπεραστεί με την διαμόρφωση της εντολής και την διαχείριση των προκαθορισμένων λογικών μονάδων. Συγκεκριμένα μπορεί να χρησιμοποιηθεί είτε η παράμετρος *"2>&1"* μετά την εντολή, οπότε τα σφάλματα θα επισυναφθούν στη standard output ώστε ο κώδικας μετά να μπορεί να τα διαχειριστεί, είτε η παράμετρος *"2> όνομα αρχείου"* με την οποία μεταφέρεται η standard error σε ένα αρχείο και κατόπιν μπορεί να γίνει ανάγνωση αυτού από την php. Στο Σχήμα 3-4 φαίνεται τροποποιημένο το παράδειγμα όπως φαίνεται στο Σχήμα 3-3. Σε αυτό το παράδειγμα, για την μεν shell_exec η standard error και η standard output είναι ενωμένες ενώ στην exec η standard error έχει καταγραφεί σε αρχείο κειμένου.

```
$standard_output= shell_exec("my_command_line_executable 2&>1")

$exec("my_command_line_executable 2> error.log", $stadard_ouput_array,
                                $command_exit_code)
```

Σχήμα 3-4 Παράδειγμα χρήσης των exec και shell_exec με διαχειρίσιμη τη standard error

Το γεγονός ότι οι εντολές της php αναμένουν την ολοκλήρωση της εφαρμογής για να λάβουν το περιεχόμενο της standard output στην περίπτωση όπου η εφαρμογή αργεί να εκτελεστεί, πράγμα αρκετά πιθανό καθότι η εφαρμογή καλείται να επιλύσει περίπλοκα μαθηματικά προβλήματα, θα οδηγήσει σε αργή φόρτωση της σελίδας και πιθανά, λόγω ανυπομονησίας του χρήστη, σε πλήρη απώλεια των αποτελεσμάτων. Ένα τέτοιο σενάριο μπορεί να αποφευχθεί με 2 απλές μετατροπές της προς εκτέλεση εντολής. Κατ' αρχήν η μεταφορά της standard output σε μία άλλη λογική μονάδα, όπως ένα αρχείο, ώστε να είναι προσβάσιμη με άλλο τρόπο, και η χρήση του χαρακτήρα "&" στο τέλος της εντολής. Η χρήση του χαρακτήρα "&" στο τέλος μίας εντολής σε linux υποδηλώνει ότι η εντολή πρέπει να εκτελεστεί σε ένα νέο thread στο παρασκήνιο, συνεπώς ο συνδυασμός των δύο κατασκευάζει μια εντολή μια εντολή που εκτελείται στο παρασκήνιο που αποθηκεύει το αποτέλεσμα της σε ένα αρχείο, με αποτέλεσμα η εντολές της php να ολοκληρώνονται στιγμιαία. Ο έλεγχος ολοκλήρωσης της εφαρμογής και η λήψη των αποτελεσμάτων πλέων θα πρέπει να γίνουν με την χρήση περιοδικού ελέγχου ύπαρξης και της ανάγνωσης του αρχείο όπου περιέχει το standard output. Στο Σχήμα 3-5 φαίνεται τροποποιημένο το παράδειγμα όπως φαίνεται στο Σχήμα 3-4 με την μεταφορά του standard output σε ένα αρχείο ονομασμένο "command.log" και εκτέλεση στο παρασκήνιο.

```
$standard_output= shell_exec("my_command_line_executable 2&>1 > command.log &")

      $exec("my_command_line_executable 2> error.log 1 > command.log & ",
                  $stadard_ouput_array, $command_exit_code)
```

Σχήμα 3-5 Παράδειγμα χρήσης των exec και shell_exec στο παρασκήνιο

3.2.3. Κλιμάκωση δικαιωμάτων για την εκτέλεση της εφαρμογής στον cluster.

Η κυριότερη δυσκολία που παρουσιάζεται με την εκτέλεση σε συστοιχία υπολογιστών από σελίδα είναι η έλλειψη δικαιωμάτων από τον χρήστη που εκτελεί τα αιτήματα του apache. Υπάρχουν αρκετοί τρόποι να λυθεί αυτό το πρόβλημα.

Η πιο απλή λύση είναι να δοθούν τα δικαιώματα που χρειάζεται στον προκαθορισμένο χρήστη του apache. Η κίνηση αυτή παρ' ότι διευκολύνει αρκετά στην κατασκευή της σελίδας παρουσιάζει σημαντικά προβλήματα ασφαλείας. Ο χρήστης αυτός είναι αρκετά ευάλωτος και δίνοντας του δικαιώματα πρόσβασης σε πρωτόκολλα επικοινωνίας δικτύου υποβαθμίζεται η ασφάλεια όχι μόνο του cluster αλλά και όλου του τοπικού δικτύου.

Μια άλλη λύση είναι η χρήση της εντολής sudo που επιτρέπει σε ένα χρήστη να εκτελέσει τη εντολή σαν ένας άλλος χρήστης. Δυστυχώς η χρήση αυτής της εντολής παρουσιάζει και πάλι θέματα ασφαλείας επειδή, δίνοντας στον χρήστη του apache δικαίωμα κλιμάκωσης του εαυτού του για να εκτελέσει αυτή την εντολή, μπορεί να χρησιμοποιηθούν για την πρόσβαση σε οποιαδήποτε εντολή. Ταυτόχρονα, ο κωδικός του χρήστη που χρησιμοποιείται πρέπει να υπάρχει γραμμένος σαν κείμενο εντός του κώδικα της σελίδας και, καθώς ο κώδικας τις σελίδας είναι αναγνώσιμος από όλους τους χρήστες του cluster, μπορεί εύκολα να χρησιμοποιηθεί κακόβουλα.

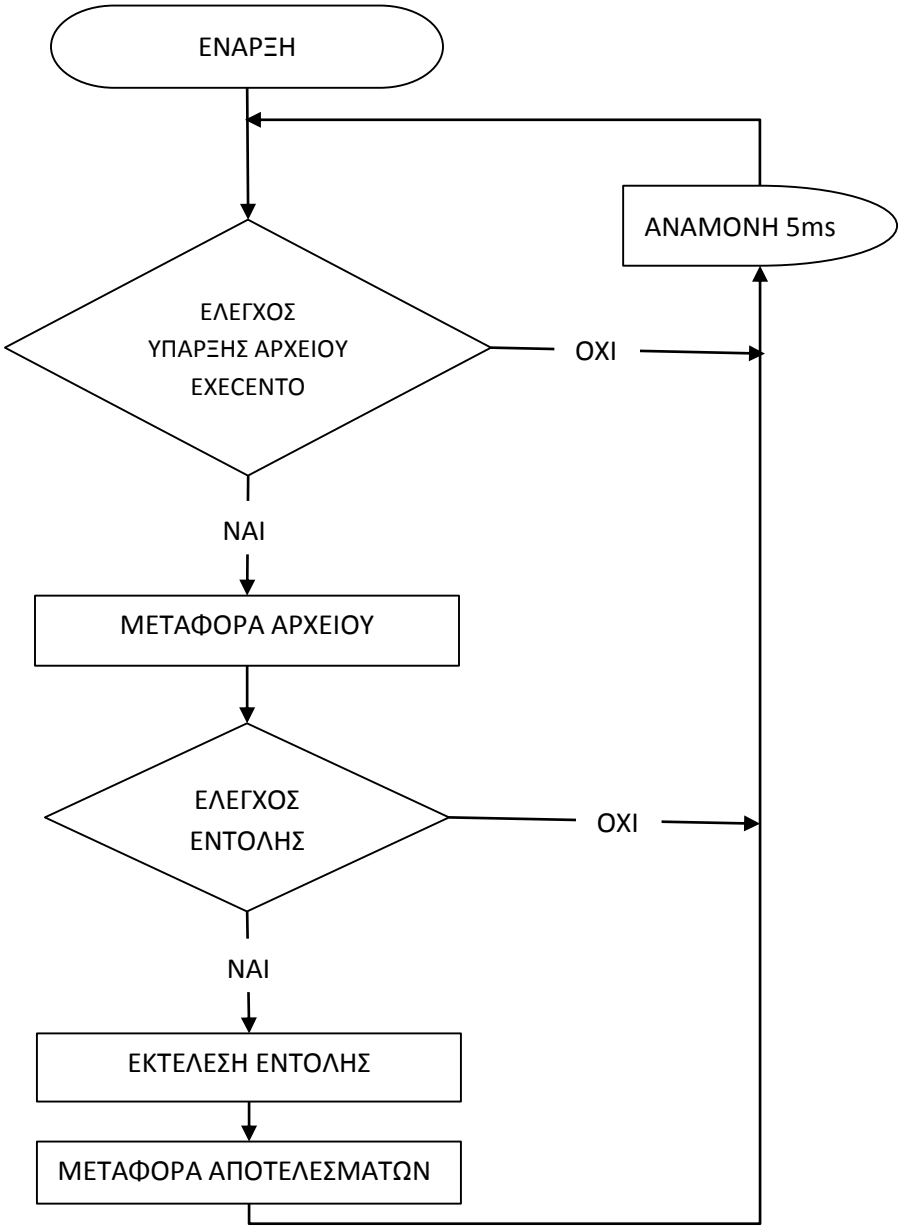
Η τρίτη λύση είναι η χρήση μίας εξωτερικής υπηρεσίας (service) εκτελούμενο από έναν άλλο χρήστη με τα απαραίτητα δικαιώματα το οποίο αναμένει την ύπαρξη μίας αίτησης και κατόπιν την εκτελεί και μεταφέρει τα αποτελέσματα σε προσβάσιμο μέρος για τον χρήστη του apache. Αυτή η λύση είναι αρκετά ασφαλέστερη καθότι δεν υποβαθμίζει την ασφάλεια του cluster, όμως πρόκειται και για μια αρκετά περιπλοκότερη λύση και δυσκολεύει την εκτέλεση της σελίδας καθότι πλέον απαιτεί δύο υπηρεσίες να τρέχουν (το νέο και τον apache) αντί του ενός. Παρά τις δυσκολίες η υψηλότερη ασφάλεια που παρουσιάζεται με την χρήση της υπηρεσίας την καθιστά την προτιμώμενη λύση.

3.2.3.α Σχεδιασμός υπηρεσίας κλιμάκωσης δικαιωμάτων.

Για την υπηρεσία χρησιμοποιήθηκε BASH script σαν γλώσσα προγραμματισμού [**19**]. Ο λόγος που προτιμήθηκε είναι η απλότητα στην γραφή καθώς και η ευκολία πρόσβασης/μεταφοράς αρχείων και εκτέλεσης εφαρμογών μέσω αυτής.

Η υπηρεσία αποτελείται από ένα μικρό αλγόριθμο που επαναλαμβάνεται κάθε 5 δευτερόλεπτα εως ότου η υπηρεσία τερματιστεί. Η σελίδα καταγράφει την εντολή προς εκτέλεση σε ένα αρχείο

κειμένου, ονομαζόμενο execento, σε έναν ειδικό υποφάκελο, όπου ο χρήστης του apache έχει δικαιώματα εγγραφής. Ο αλγόριθμος της υπηρεσίας ελέγχει για την ύπαρξη του αρχείου αυτού και εάν υπάρχει το μεταφέρει στον φάκελο με το εκτελέσιμο. Κατόπιν ελέγχει ότι η προς εκτέλεση εντολή είναι η σωστή χρησιμοποιώντας μια επαναληπτική διαδικασία. Αφού λαμβάνονται μια μία οι λέξεις της προς εκτέλεση εντολής, διαχωρισμένες με τον κενό χαρακτήρα, για κάθε μία ελέγχεται με μια απλή regular expression [**5**] ότι είναι αριθμός. Εάν δεν είναι, ελέγχεται ότι περιέχεται σε μία λίστα αναμενόμενων τιμών, και εάν καμία τιμή δεν είναι και μη αριθμητική και εκτός των αναμενόμενων τιμών, η εντολή εκτελείται. Μετά την ολοκλήρωση της εφαρμογής μεταφέρει τα αρχεία εξόδου στον υποφάκελο όπου μπορεί να γράψει ο apache. Το διάγραμμα ροής φαίνεται στο Σχήμα 3-6 και ο κώδικας ολοκληρωμένος στο Σχήμα 3-7.



Σχήμα 3-6 Διάγραμμα ροής υπηρεσίας.

```
#!/bin/bash
validwords='./timeout -INT mpirun -np ./a.out -x -y -o -i -r -t >'
while true
do
if [ -f "/var/www/html/WebFEBUI/binout/execento" ]
then
echo "filefound"
mv /var/www/html/WebFEBUI/binout/execento /home/gpetr/WEB/
mv /var/www/html/WebFEBUI/binout/*.corepartition /home/gpetr/WEB/
fcon=`cat /home/gpetr/WEB/execento`
#echo $fcon
isgood=1
for word in $fcon
do
if [[ $word =~ '^[0-9]' ]];then continue; fi
isgood=0
for test in $validwords
do
if [[ $word == "$test" ]] ; then isgood=1; continue; fi
done
done
if [[ $isgood == 1 ]]
then
echo 'command fine'
chmod 775 /home/gpetr/WEB/execento
rm /home/gpetr/WEB/pgmres_conv
rm /home/gpetr/WEB/used_nodes
/home/gpetr/WEB/execento
mv /home/gpetr/WEB/pgmres_conv /var/www/html/WebFEBUI/binout/pgmres_conv
mv /home/gpetr/WEB/used_nodes /var/www/html/WebFEBUI/binout/used_nodes
mv /home/gpetr/WEB/*.output /var/www/html/WebFEBUI/binout/
mv /home/gpetr/WEB/*.errors /var/www/html/WebFEBUI/binout/
rm /home/gpetr/WEB/execento
rm /home/gpetr/WEB/*.corepartition
else
echo "bad command"
fi
fi
sleep 5
done
```

Σχήμα 3-7 Κώδικας υπηρεσίας.

Επειδή πρόκειται για υπηρεσία είναι ένας αλγόριθμος που δεν τερματίζει ποτέ και πρέπει να εκτελείται πάντα στο παρασκήνιο. Για να διευκολυνθεί η λειτουργία της χρειάζεται ένα ακόμα μικρό script που διαχειρίζεται την έναρξη και τον τερματισμό της. Τα script αυτής της μορφής είναι τυποποιημένα σε όλα τα linux συστήματα και δέχονται 4 εντολές *start, stop, restart* και *status*. Η τυποποίηση αυτή βοηθάει την ομαλή εκτέλεση των υπηρεσιών από τις προϋπάρχουσες υπηρεσίες και διαδικασίες αυτόματης εκκίνησης τους από το λειτουργικό.

Το script καλείται πάντα μαζί με μια εντολή. Εάν η εντολή είναι *start* ελέγχει την ύπαρξη ενεργούς εκτέλεσης της υπηρεσίας και εάν δεν υπάρχει, την ξεκινά. Εάν είναι *stop* ελέγχει την ύπαρξη ενεργούς εκτέλεσης της υπηρεσίας και εάν υπάρχει την τερματίζει. Εάν είναι *restart* εκτελεί πρώτα την *stop* και μετά την *start.* Και τέλος, εάν είναι *status* απλά αναφέρει την ύπαρξη ή όχι της ενεργούς εκτέλεσης και το process id αυτής. Ο κώδικας του script αυτού φαίνεται στο Παράρτημα 4.

3.2.3.β Κλήση της υπηρεσίας μέσω της σελίδας.

Η υπηρεσία αναμένει την ύπαρξη ενός αρχείου σε μία συγκεκριμένη τοποθεσία, οπότε η εκτέλεση της εντολής από την php ανάγεται σε δημιουργία ενός αρχείου κειμένου που περιέχει την προς εκτέλεση εντολή. Για την εγγραφή ενός αρχείου κειμένου από την php χρησιμοποιούνται 3 συναρτήσεις, οι *fopen*, *fwrite* και *fclose* [**15**].

Η *fopen* λαμβάνει 2 ορίσματα: το πρώτο είναι η τοποθεσία του αρχείου και το δεύτερο ένα string το οποίο χαρακτηρίζει την μέθοδο με την οποία η σελίδα θα μπορεί να διαχειριστεί το αρχείο και επιστέφει ένα αντικείμενο τύπου *file pointer resource* που είναι το αντικείμενο που η php χρησιμοποιεί για την διαχείριση εγγραφής και ανάγνωσης σε αρχεία. Οι βασικότερες τιμές που μπορεί να λάβει το δεύτερο όρισμα είναι *"r","w","a"* που αντίστοιχα υποδηλώνουν άνοιγμα για ανάγνωση μόνο, άνοιγμα για εγγραφή από την αρχή του αρχείου και άνοιγμα για εγγραφή στο τέλος του αρχείου. Εφόσον η *fopen* καλείται με επιλογή που επιτρέπει την εγγραφή, εάν δεν υπάρχει το αρχείο θα δοκιμάσει να το δημιουργήσει.

Η *fwrite* λαμβάνει 2 ορίσματα, ένα αντικείμενο τύπου *file pointer resource,* όπως έχει δημιουργηθεί από την *fopen,* και μια αλφαριθμητική μεταβλητή που περιέχει ένα κείμενο προς εγγραφή στο αρχείο και επιστρέφει ένα ακέραιο που περιέχει τον αριθμό των bytes που γράφτηκαν για επιβεβαίωση. Καθότι η εγγραφή γίνεται από αλφαριθμητική μεταβλητή έχει σημασία η διαδικασία αλλαγής γραμμής, καθότι η fortran λαμβάνει τα δεδομένα ανά γραμμή. Για την αλλαγή της γραμμής εντός μίας αλφαριθμητικής μεταβλητής χρησιμοποιούνται οι ειδικοί χαρακτήρες "\r" και "\n". Ο ειδικός χαρακτήρας "\r" υποδηλώνει την πληκτρολόγηση του πλήκτρου return (ή enter) και συνεπώς τον τερματισμό της τρέχουσας γραμμής, ενώ ο χαρακτήρας "\n" υποδηλώνει την δημιουργία μίας νέας γραμμής. Διαφορετικά λειτουργικά και γλώσσες αντιλαμβάνονται το τέλος της γραμμής με διαφορετικό τρόπο. Χαρακτηριστικά οι περισσότερες linux εφαρμογές προτιμούν την χρήση του "\n" μόνο αγνοώντας πλήρως τον "\r" ενώ σχεδόν όλες οι εφαρμογές windows απαιτούν τον συνδυασμό και των δύο ("\r\n") για την υποδήλωση μιας νέας γραμμής. Εφόσον το αρχείο αυτό θα οδηγηθεί προς ανάγνωση από μια άλλη εφαρμογή είναι σημαντικό να γίνει επιβεβαίωση ότι ο τρόπος αλλαγής γραμμής που επιλέχτηκε είναι συμβατός με την μεθοδολογία ανάγνωσης της άλλης εφαρμογής.

Τέλος η *fclose* λαμβάνει ως όρισμα ένα αντικείμενο τύπου *file pointer resource* όπως έχει δημιουργηθεί από την *fopen* και επιχειρεί να το αποθηκεύσει και να τερματίσει την σύνδεση προς αυτό. Η συνάρτηση επιστρέφει μια λογική μεταβλητή που υποδηλώνει ότι η διαδικασία ήταν επιτυχής.

Η διαδικασία δημιουργίας μίας αίτησης εκτέλεσης της εφαρμογής γίνεται συνεπώς στα ακόλουθα στάδια:

- Υπολογίζεται ο επόμενος αύξων αριθμός που θα αποτελέσει το κλειδί της αίτησης.
- Κατασκευάζεται η προς εκτέλεση εντολή σε μία αλφαριθμητική μεταβλητή.
- Εάν χρειάζεται, κατασκευάζεται μία αλφαριθμητική μεταβλητή που περιέχει την αντιστοίχιση των κόμβων του πλέγματος ανά επεξεργαστή και αποθηκεύεται σε αρχείο.
- Κατασκευάζεται μία *file pointer resource* για το προς εγγραφή αρχείο με την *fopen*.
- Γίνεται εγγραφή της αλφαριθμητικής που περιέχει την εντολή στο αρχείο με την *fwrite*.

- Αποθηκεύονται οι αλλαγές και τερματίζεται η σύνδεση στο αρχείο με την *fclose*.
- Δημιουργείται μια εγγραφή στην βάση δεδομένων που περιέχει πληροφορίες για την δημιουργία της αίτησης καθώς και τις παραμέτρους με τις οποίες καλέστηκε ώστε η σελίδα να έχει γνώση ότι έγινε η κλήση της εφαρμογής.

Το Σχήμα 3-8 περιέχει τον κώδικα κλήσης της υπηρεσίας από την σελίδα. Στον κώδικα χρησιμοποιείται ένα array ονομαζόμενο $Settings που περιέχει διάφορες μεταβλητές ρυθμίσεων, όπως έχουν οριστεί αλλού στην σελίδα, ώστε να είναι εύκολη η παραμετροποίηση τους. Ειδικότερα η $Settings['Exec'] περιέχει την ονομασία του εκτελέσιμου αρχείου, η $Settings['timeoutcom'] περιέχει την εντολή που χρησιμοποιείται για τον αυτόματο τερματισμό του εκτελέσιμου, η $Settings[‘timeout’] περιέχει τον χρόνο σε δευτερόλεπτα μετά από τον οποίο πρέπει να τερματιστεί αυτόματα η εντολή και η $Settings['Output_path'] περιέχει τη απόλυτη τοποθεσία του φάκελου όπου θα δημιουργηθούν τα αρχεία. Επίσης έχει αφαιρεθεί το τμήμα δημιουργίας του αρχείου-μεταβλητή που περιέχει την αντιστοίχιση των κόμβων του πλέγματος ανά επεξεργαστή, το οποίο θα αναλυθεί σε μεταγενέστερο κεφάλαιο.

```
$id = 1;
$query = "SELECT MAX(id) as id FROM GMRES_Requests";
$result = mysql_query($query);
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
            $id += $row['id'];
}
if ($custom_nodes == 1) {
     $corefile = "-n $id.corepartition";
} else {
    $corefile = "";
}

$execstr = $Settings['timeoutcom']. $Settings['timeout'] . " mpirun -np $c_cores ".$Settings['Exec']." -x
$c_x -y $c_y -o $c_outer_itr -i $c_inner_itr -r $c_rdefl -t $c_tolerance $corefile > $id.output 2> $id.errors
"; //

// Δημιουργία αρχείου που περιέχει την κατανομή των node
$file = $Settings['Output_path']."execento";

$fp = fopen($file, "w") or die("Couldn't open $file for writing!");
fwrite($fp, $execstr) or die("Couldn't write values to file!");
fclose($fp);

$sqlstring = " INSERT INTO GMRES_Requests (`id`, `cores`, `dimx`, `dimy`, `outerit`, `innerit`,
`rdeflation`, `tolerance`, `complete`, `create_date`, `IP`, `custom_nodes`) ";

$sqlstring .= " VALUES ($id, '$c_cores', '$c_x', '$c_y', '$c_outer_itr', '$c_inner_itr', '$c_rdefl',
'$c_tolerance', '0', NOW(), '" . mysql_real_escape_string($_SERVER['REMOTE_ADDR']) .
"','$custom_nodes'); ";

mysql_query($sqlstring) or die(mysql_error());
```

Σχήμα 3-8 Κώδικας κλήσης της υπηρεσίας από την php.

3.3. Ανάγνωση αποτελεσμάτων εφαρμογής από την σελίδα.

3.3.1. Ανάγνωση αρχείων κειμένου από php.

Η ανάγνωση αρχείων κειμένου από την php είναι μια σχετικά απλή διαδικασία. Η αρχική ανάγνωση γίνεται με την χρήση της συνάρτησης *file_get_contents* [**15**]. Η *file_get_contents* λαμβάνει ως όρισμα την απόλυτη τοποθεσία του αρχείου προς ανάγνωση και επιστρέφει τα περιεχόμενα σαν μεταβλητή κειμένου. Η εύρεση της απόλυτης τοποθεσίας ενός αρχείου που βρίσκεται σε συγκεκριμένη τοποθεσία σχετικά με την σελίδα γίνεται εύκολα με την χρήση της μεταβλητής συστήματος *$_SERVER{'DOCUMENT_ROOT'},* η οποία υπάρχει εξορισμού και περιέχει την απόλυτη τοποθεσία του καταλόγου από όπου γίνεται ανάγνωση των αρχείων του http server, όπως ορίζεται στις ρυθμίσεις αυτού.

Υπάρχουν άλλες δυο διαδικασίες της php που βοηθάνε στην διαχείριση των προς ανάγνωση αρχείων, η συνάρτηση *file_exists* και η υπορουτίνα *unlink* [**15**]. Η συνάρτηση *file_exists* λαμβάνει ως όρισμα την απόλυτη τοποθεσία του αρχείου προς ανάγνωση και επιστρέφει μια λογική μεταβλητή που είναι true εάν υπάρχει το αρχείο, και χρησιμοποιείται για τον έλεγχο της ολοκλήρωσης της εξωτερικής εφαρμογής. Η υπορουτίνα unlink λαμβάνει ως όρισμα την απόλυτη τοποθεσία του αρχείου και κατόπιν, εφόσον αυτό υπάρχει, το διαγράφει.

Καθότι η php χρησιμοποιεί την ύπαρξη του αρχείου σαν ένδειξη ότι η εξωτερική εφαρμογή έχει εκτελεστεί, κατά την ανάγνωση των αρχείων θα χρησιμοποιείται ο εξής αλγόριθμος: Η σελίδα ελέγχει για την ύπαρξη του αρχείου, λαμβάνει τα δεδομένα που περιέχει, τα επεξεργάζεται, τα αποθηκεύει στην βάση δεδομένων και κατόπιν διαγράφει το αρχείο, ώστε να μπορέσει να αντιληφθεί την ολοκλήρωση της επόμενης εκτέλεσης από τη εκ νέου ύπαρξη του αρχείου. Ένα παράδειγμα χρήσης του παραπάνω αλγόριθμου φαίνεται στο Σχήμα 3-9.

```
$filename = "path\to\file.output";
if (file_exists($filename)){
        $file_contents = file_get_contents($filename);
        unlink($filename);
};
```

Σχήμα 3-9. Παράδειγμα κώδικά μεταφοράς περιεχομένων αρχείου σε μεταβλητή στην php.

3.3.2. Εφαρμογή

Για την εφαρμογή του συστήματος ανάγνωσης κατασκευάστηκε μια σελίδα–βιβλιοθήκη που περιέχει 4 συναρτήσεις, η lib_parser.php. Η σελίδα περιέχει τις εξής συναρτήσεις:

- "**parseouptputfile**" για την ανάγνωση του αρχείου στο οποίο μεταφέρεται το standard output της εξωτερικής εφαρμογής,
- "**parseconvfile**" για την ανάγνωση του αρχείου το οποίο περιέχει τα σφάλματα επαναλήψεων,
- "**parsenodefile**" για την ανάγνωση του αρχείου το οποίο περιέχει τα την κατανομή των κόμβων του πλέγματος σε επεξεργαστές,
- "**parseresult**" για την συνολική κλήση της διαδικασίας ανάγνωσης αρχείων.

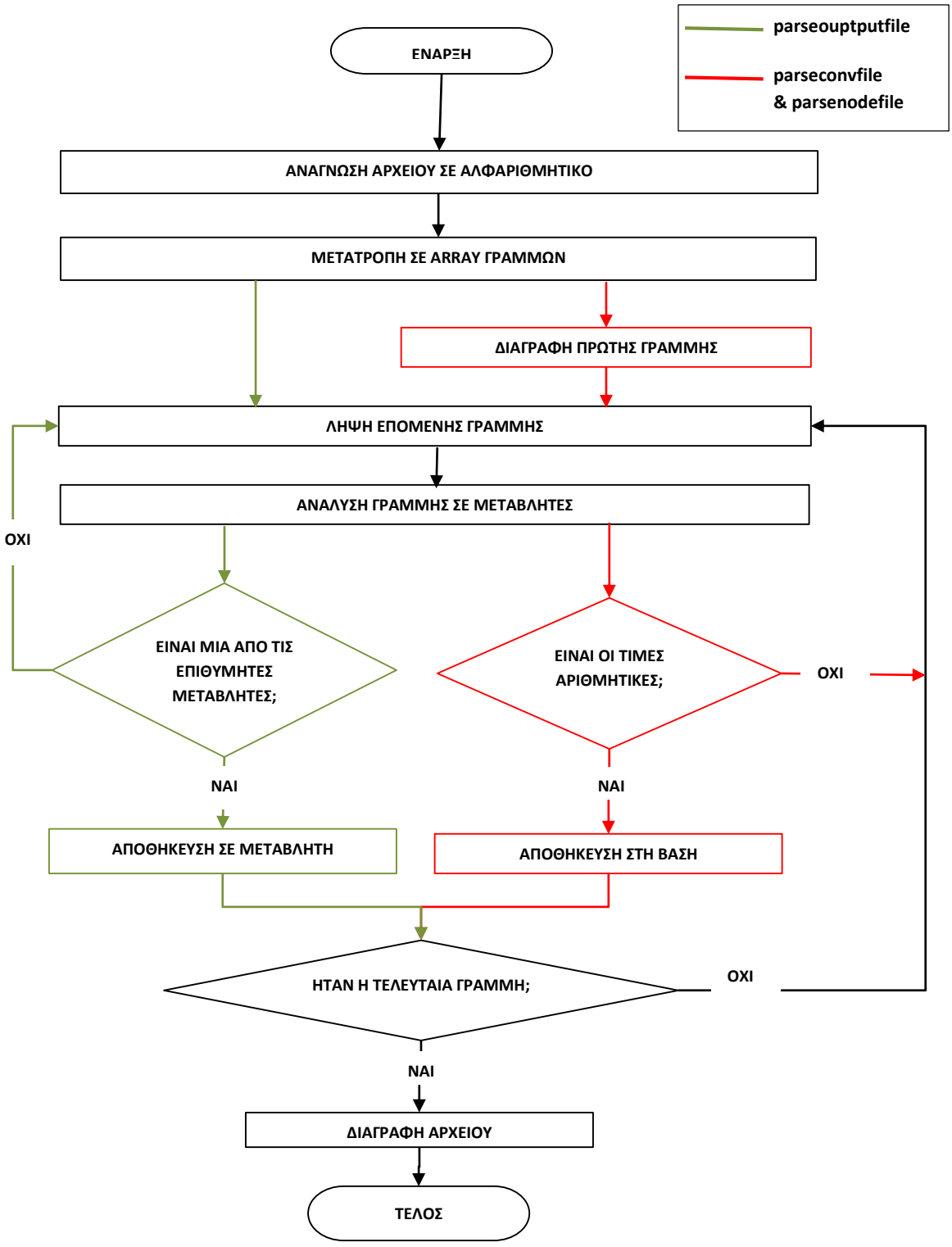
Οι"**parseconvfile**" και "**parsenodefile**" λαμβάνουν 2 ορίσματα, ένα αλφαριθμητικό, που υποδηλώνει την τοποθεσία του αρχείου και έναν ακέραιο, που υποδηλώνει το κλειδί της εγγραφής στην βάση, ενώ δεν επιστρέφουν τίποτα. Ενώ η "**parseouptputfile**" λαμβάνει τα ίδια ορίσματα και επιστρέφει μια array με τις αναγνωσμένες τιμές. Οι αλγόριθμοι λειτουργίας τους φαίνονται στο Σχήμα 3-10.

Η "**parseresult**" λαμβάνει δύο ορίσματα: ένα ακέραιο, πού υποδηλώνει το κλειδί της εγγραφής στην βάση, και την array $Settings, που περιέχει τις γενικές ρυθμίσεις της σελίδας και επιστρέφει ένα ακέραιο που υποδηλώνει το αποτέλεσμά της. Ο ακέραιος που επιστρέφεται παίρνει την τιμή 0 στην περίπτωση που δεν εντοπίστηκε τίποτα, 1 στην περίπτωση που τα αρχεία βρέθηκαν και η εισαγωγή ήταν επιτυχής και 2 στην περιπτώσεις που είτε δεν βρέθηκαν τα αρχεία αποτελεσμάτων αλλά βρέθηκε αρχείο με σφάλματα είτε έχει περάσει ο μέγιστος επιτρεπτός χρόνος εκτέλεσης εντολής (timeout) οπότε και η εντολή εκτέλεσης θεωρείται αποτυχημένη και το γεγονός αυτό αποθηκεύεται στην βάση.

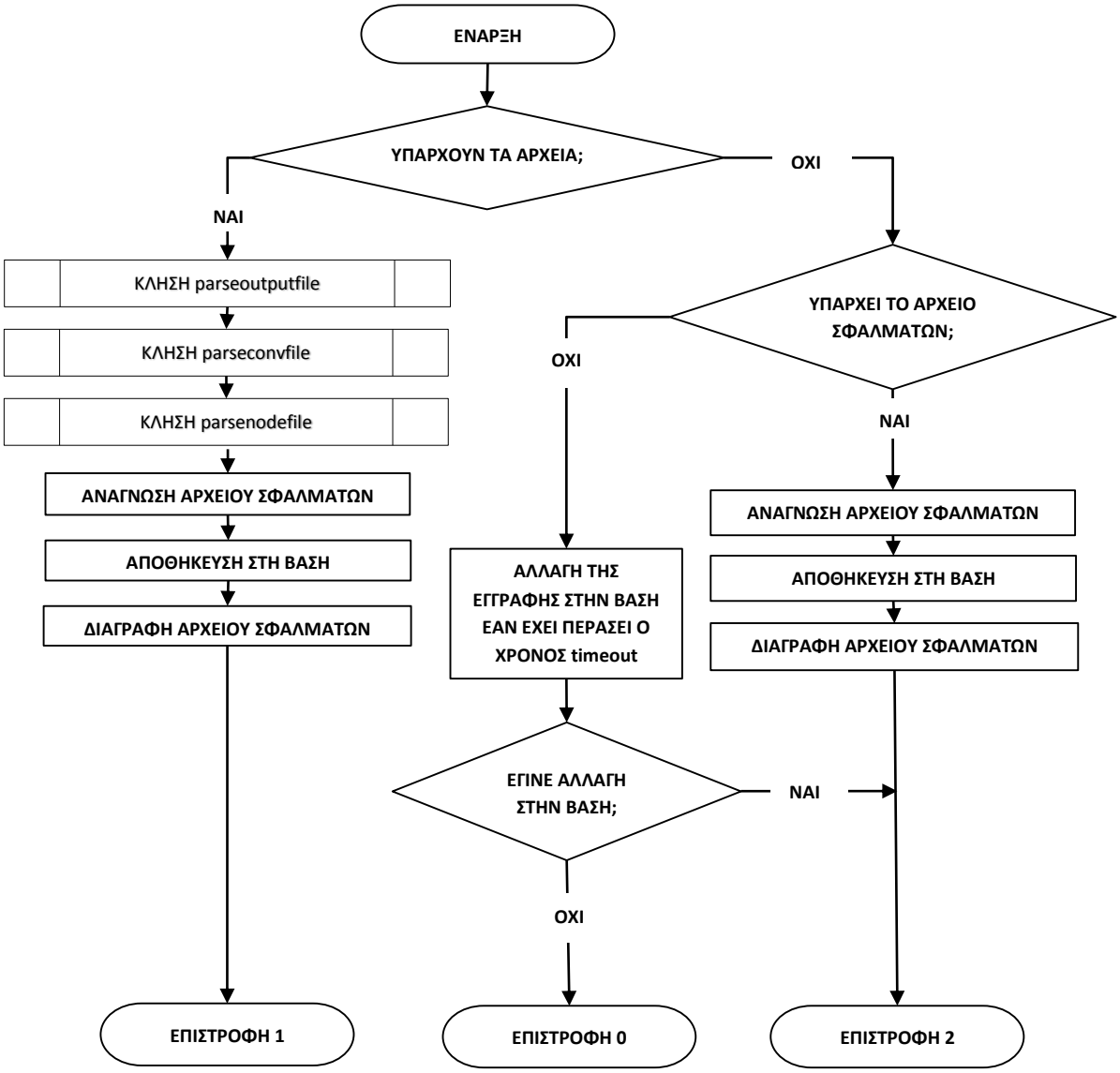
Ο υπολογισμός, εάν έχει περάσει ο μέγιστος επιτρεπτός χρόνος εκτέλεσης εντολής, γίνεται στη βάση, καθότι ο αρχικός χρόνος δημιουργίας αποθηκεύτηκε από την βάση και η χρήση του ακριβώς ίδιου ρολογιού μειώνει το πιθανό σφάλμα λόγω μη συγχρονισμού ώρας μεταξύ εφαρμογών ή και υπολογιστών. Εφόσον βρεθεί εγγραφή στην βάση που αντιστοιχεί στο κλειδί της επιλεγμένης εγγραφής όπου έχει περάσει ο μέγιστος επιτρεπτός χρόνος εκτέλεσης, η εγγραφή αποθηκεύεται αμέσως σαν εσφαλμένη.

Ο αλγόριθμος λειτουργίας της "**parseresult**" φαίνεται στο Σχήμα 3-11.

Αναλυτικά ο κώδικας των συναρτήσεων αυτών παρουσιάζεται στο Παράρτημα 5.



Σχήμα 3-10 Διάγραμμα ροής των συναρτήσεων ανάγνωσης αρχείων.



Σχήμα 3-11 Διάγραμμα ροής parseresult.

3.4. Συνοπτική παρουσίαση της διαδικασίας

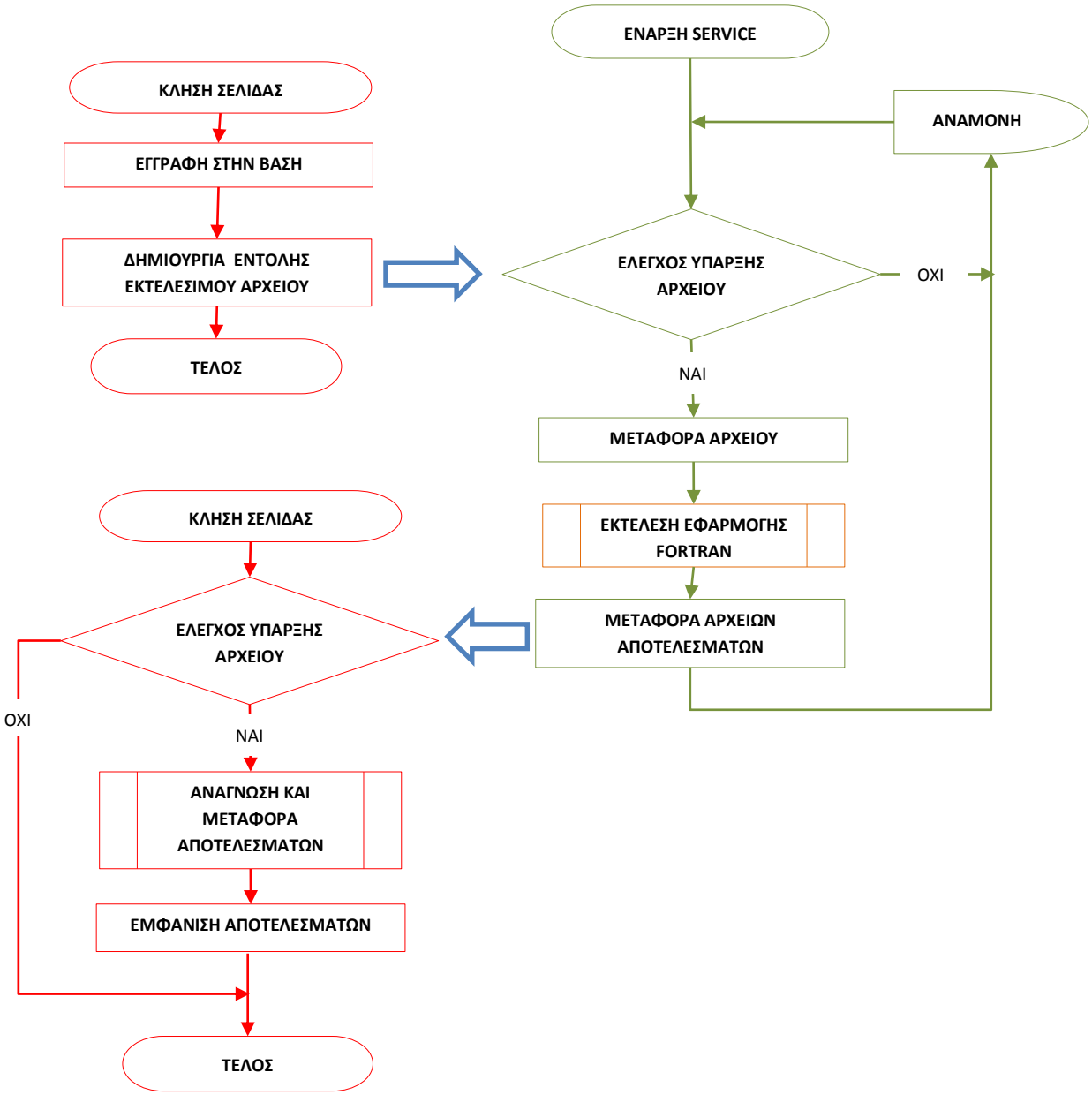
Όπως παρουσιάστηκε παραπάνω, η διαδικασία είναι μια αρκετά περίπλοκη διαδικασία που λαμβάνει χώρα μεταξύ αρκετών εφαρμογών.

Αρχικά γίνεται κλήση της σελίδας δημιουργίας νέας αίτησης. Η νέα αίτηση αυτή αποθηκεύεται στην βάση δεδομένων και κατασκευάζεται ένα αρχείο κειμένου που περιέχει την εντολή εκτέλεσης του αρχείου.

Παράλληλα βρίσκεται υπό εκτέλεση η υπηρεσία που έχει εκτελεστεί από χρήστη με δικαιώματα εκτέλεσης της εντολής και ανά τακτικά χρονικά διαστήματα ψάχνει για το αρχείο με την εντολή εκτέλεσης που μόλις έχει δημιουργηθεί από την σελίδα. Μόλις εντοπιστεί από την υπηρεσία το αρχείο, η υπηρεσία το μεταφέρει στον σωστό φάκελο προς εκτέλεση, ελέγχει ότι η εντολή που περιέχει είναι εντολή εκτέλεσης του σωστού εκτελέσιμου, και εφόσον αυτό ισχύει, την εκτελεί. Αφού ολοκληρωθεί η εκτέλεση της εντολής η υπηρεσία μεταφέρει τα αποτελέσματα αυτής στον φάκελο που αρχικά περιείχε το αρχείο με την εντολή, και συνεχίζει περιοδικά να ψάχνει για νέο αρχείο με εντολή.

Παράλληλα, εφόσον υπάρχει μη ολοκληρωμένη εγγραφή στην βάση, κάθε φορά που καλούνται οι σελίδες της διεπαφής προσπαθούν να εντοπίσουν αρχεία με αποτελέσματα στον φάκελο που είχε δημιουργηθεί η εντολή εκτέλεσης. Αφού μεταφερθούν εκεί τα αποτελέσματα από την υπηρεσία, η σελίδα τα εντοπίζει και αποθηκεύει τα περιεχόμενα τους στην βάση και τα εμφανίζει στον χρήστη, ολοκληρώνοντας την διαδικασία.

Συνολικά η διαδικασία φαίνεται στο Σχήμα 3-12.



Σχήμα 3-12 Διάγραμμα ροής διαδικασίας επικοινωνίας.

4. Δημιουργία αίτησης εκτέλεσης εξωτερικής εφαρμογής

Η δημιουργία μιας αίτησης εκτέλεσης της εφαρμογής λαμβάνει χώρα σε δύο σελίδες στη διεπαφή του χρήστη καθώς και σε 2 σελίδες-βιβλιοθήκες. Οι δύο σελίδες-βιβλιοθήκες περιέχουν μαθηματικές συναρτήσεις και μετατροπές που χρησιμεύουν στον σχεδιασμό της κατανομής των κόμβων του πλέγματος σε επεξεργαστές. Η ίδια η δημιουργία της αίτησης λαμβάνει χώρα στην μία από τις δύο σελίδες, ενώ στην δεύτερη λαμβάνει χώρα ο σχεδιασμός της κατανομής των κόμβων του πλέγματος σε επεξεργαστές από τον χρήστη.

4.1. Μοντελοποίηση της διαδικασίας κατανομής των κόμβων του πλέγματος σε επεξεργαστές.

Μία από τις κυριότερες δυσκολίες στον σχεδιασμό της σελίδας είναι η εύρεση ενός μοντέλου σχεδιασμού της κατανομής των κόμβων του πλέγματος ανά επεξεργαστή. Η ίδια η κατανομή είναι ένας πίνακας μεγάλων διατάσεων και συνεπώς παρουσιάζεται πλήθος δυσκολιών:

- Είναι δύσχρηστο για τον χρήστη να επιλέγει τιμές μία-μία σε ένα πίνακα μεγάλων διαστάσεων.
- Υπάρχει μεγάλος όγκος πληροφοριών που καλείται να μεταφερθεί μεταξύ σελίδων όπου η μεταφορά του μέσω headers σελίδων είναι αδύνατη και η συνεχής αποθήκευση του στην βάση οδηγεί σε μία αργή διαδικασία σχεδιασμού.

Συνεπώς, είναι απαραίτητη η μετατροπή του πίνακα αυτού σε μια πιο συμπαγή μορφή που να είναι πιο εύκολη στην συμπλήρωση από τον χρήστη και να μπορεί να μετατραπεί σε ένα σχετικά μικρό αλφαριθμητικό, ώστε να μεταφέρεται εύκολα μεταξύ σελίδων.

4.1.1. Σχεδιασμός μοντέλου σχεδίασης κατανομής των κόμβων του πλέγματος σε επεξεργαστές.

Για αυτή την μετατροπή επιλέχτηκε η χρήση παραλληλόγραμμων σε διαφορετικά επίπεδα.

Ορίζεται ένα αντικείμενο box το οποίο αντιπροσωπεύει ένα παραλληλόγραμμο παράλληλο στους άξονες X και Y, σε ένα συγκεκριμένο ύψος στον άξονα Z. Το κάθε box χαρακτηρίζεται από 6 μεγέθη:

- Τον επεξεργαστή στον οποίο αντιστοιχεί (οριζόμενο σαν core).
- Την θέση του στον άξονα Z (οριζόμενη σαν z).
- Την θέση της κοντινότερης στον άξονα X πλευράς του (οριζόμενη σαν left).
- Την θέση της μακρινότερης στον άξονα X πλευράς του (οριζόμενη σαν right).
- Την θέση της κοντινότερης στον άξονα Y πλευράς του (οριζόμενη σαν bottom).
- Την θέση της μακρινότερης στον άξονα Y πλευράς του (οριζόμενη σαν top).

Αρχικά ορίζεται ένα box με τιμές στο core=1, z=0, left=0, right = το μέγεθος της διάστασης X του πίνακα των κόμβων του πλέγματος, bottom = 0 και top = το μέγεθος της διάστασης Y του πίνακα των κόμβων του πλέγματος.

Κατόπιν ορίζονται box με τους ακόλουθους περιορισμούς:

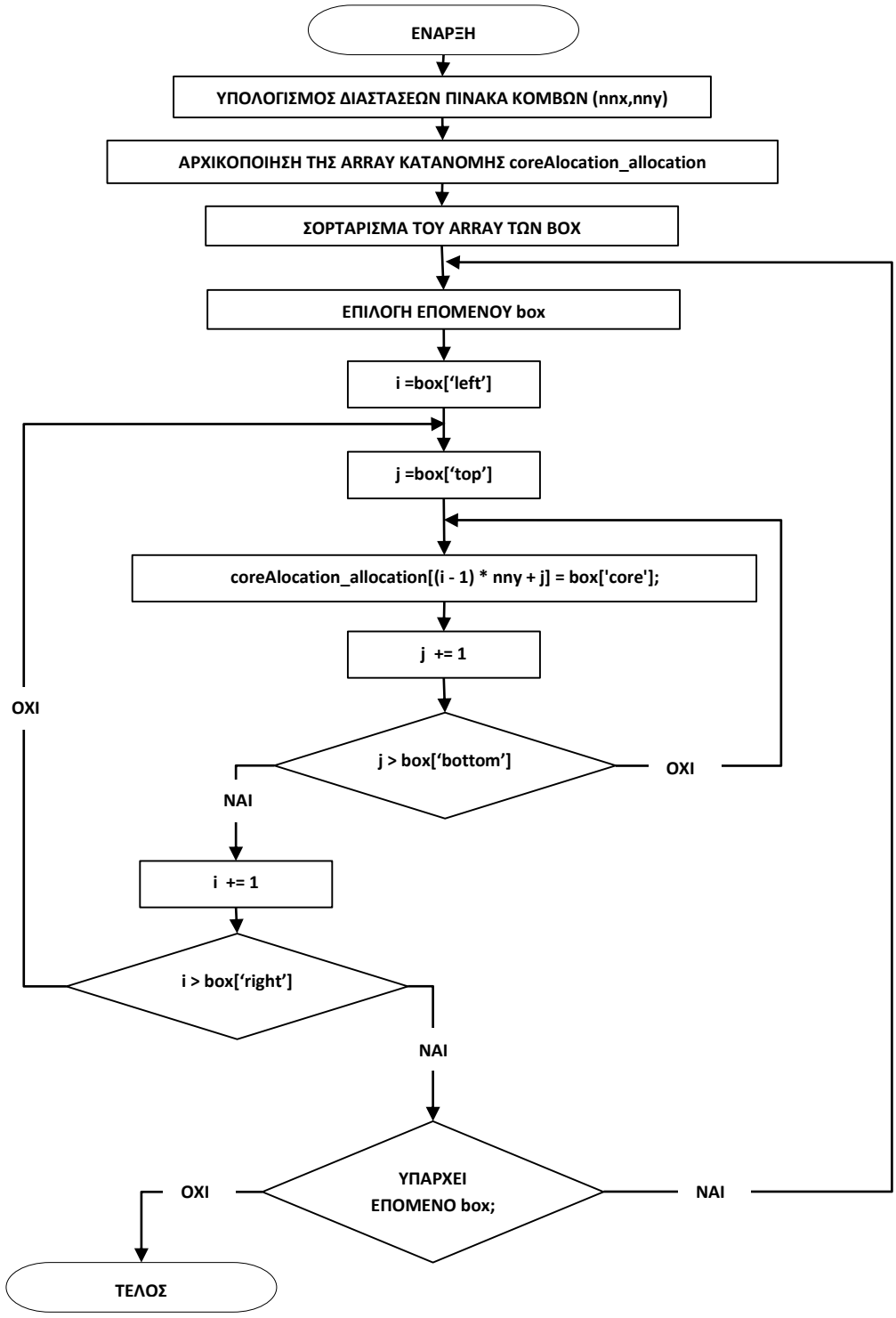
- Το core είναι φυσικό αριθμός μικρότερος ή ίσος του πλήθους των προς χρήση πυρήνων.
- Το Z είναι φυσικός αριθμός και δεν αντιστοιχεί σε κανένα άλλο box.
- Τα left και right είναι φυσικοί αριθμοί ανάμεσα στο 0 και το μέγεθος της διάστασης X του πίνακα των κόμβων του πλέγματος.
- Τα top και bottom είναι φυσικοί αριθμοί ανάμεσα στο 0 και το μέγεθος της διάστασης Y του πίνακα των κόμβων του πλέγματος.

Η δημιουργία αντικειμένων box με αυτούς τους κανόνες οδηγεί στην δημιουργία ενός συνόλου παραλληλόγραμμών στον χώρο, το καθένα από τα οποία καταλαμβάνει ένα τμήμα του πίνακα των κόμβων του πλέγματος αντιστοιχεί σε έναν επεξεργαστή και βρίσκεται σε διαφορετικό ύψος στον άξονα Z. Η κάτοψη αυτού του συνόλου αντιστοιχεί στην κατανομή των κόμβων του πλέγματος ανά επεξεργαστή. Το παραπάνω μοντέλο τηρεί τις σχεδιαστικές προδιαγραφές καθώς είναι και πιο εύχρηστο στον σχεδιασμό αλλά και αρκετά πιο συμπαγές από τον αρχικό πίνακα.

Η διαδικασία μετατροπής από το παραπάνω μοντέλο στον πίνακα τιμών λαμβάνει χώρα στην συνάρτηση fillallocation στην σελίδα-βιβλιοθήκη lib_coreallocation.php. Η συνάρτηση λαμβάνει τρία ορίσματα: ένα array από αντικείμενα box και τις διαστάσεις X και Y του πίνακα των στοιχείων, και επιστρέφει ένα μονοδιάστατο array όπως χρειάζεται για χρήση στο FeBUI (βλ. Παράρτημα 3). Η συνάρτηση αρχικοποιεί την array με την τιμή 1, ταξινομεί (sorts) την array των box με βάση το ύψος και κατόπιν ξεκινώντας από το χαμηλότερο αποδίδει στις περιοχές που καταλαμβάνουν στον πίνακα την τιμή του επεξεργαστή που αντιστοιχεί στην καθεμία και επιστρέφει το αποτέλεσμα. Ο κώδικας φαίνεται στο Σχήμα 4-1 και το διάγραμμα ροής στο Σχήμα 4-2.

```
function fillallocation($coreAlocation_boxes, $coreAlocation_dimx, $coreAlocation_dimy) {
   $nnx = 2 * $coreAlocation_dimx + 1;
   $nny = 2 * $coreAlocation_dimy + 1;
   $coreAlocation_allocation = array();
   for ($i = 1; $i <= $nnx; $i++) {
      for ($j = 1; $j <= $nny; $j++) {
         $coreAlocation_allocation[($i - 1) * $nny + $j] = 1;
      }
   }
   ksort($coreAlocation_boxes);
   foreach ($coreAlocation_boxes as $box) {
     for ($i = $box['left']; $i <= $box['right']; $i++) {
         for ($j = $box['top']; $j <= $box['bottom']; $j++) {
            $coreAlocation_allocation[($i - 1) * $nny + $j] = $box['core'];
         }
      }
   }
   return $coreAlocation_allocation;
}
```

Σχήμα 4-1 Κώδικας της συνάρτησης fillallocation.



Σχήμα 4-2 Διάγραμμα ροής της συνάρτησης fillallocation.

Επίσης, ορίζεται η συνάρτηση που λαμβάνει σαν όρισμα το αποτέλεσμα της fillallocation και το μετατρέπει σε ένα αλφαριθμητικό για εξαγωγή σε αρχείο για να χρησιμοποιηθεί από την εξωτερική εφαρμογή. Ο κώδικας της συνάρτησης φαίνεται στο Σχήμα 4-3.

```
function exportfile($coreAlocation_allocation) {
   $partfile = "";
   foreach ($coreAlocation_allocation as $i => $j) {
    $partfile .= "$i $j\r\n";
   }
   return $partfile;
}
```

Σχήμα 4-3 Κώδικας της συνάρτησης exportfile.

Μετά τον σχεδιασμό του μοντέλου οι μεταβλητές είναι ακόμα ακόμα σε μία μη εύκολα διαχειρίσιμη μορφή για την μεταφορά τους από σελίδα σε σελίδα. Συνεπώς κατασκευάζεται μία σειρά από συναρτήσεις με σκοπό την μετατροπή των σχεδιασμένων αντικειμένων σε ένα μικρό και εύκολα μεταφερόμενο αλφαριθμητικό. Με την χρήση αυτών επιτυγχάνεται η μετατροπή και συμπίεση ενός array από box σε ένα αλφαριθμητικό μήκους πολλαπλάσιου του 10, το οποίο είναι εύκολα μεταφερόμενο. Αναλυτικά η μεθοδολογία συμπίεσης και ο κώδικας των συναρτήσεων παρουσιάζεται στο Παράρτημα 6.

4.1.2. Ορισμός προκαθορισμένων κατανομών και αρχικών τιμών.

Αφού έχει οριστεί και δομηθεί το μοντέλο, με το οποίο θα γίνει ο σχεδιασμός της κατανομής των κόμβων του πλέγματος, είναι δυνατή η κατασκευή μιας συνάρτησης που παράγει αρχικοποιημένες τιμές για διάφορες προκαθορισμένες μεθόδους κατανομής.

Η συνάρτηση αυτή ονομάζεται loadcoreAlocationPreset και βρίσκεται στο αρχείο lib_coreallocationpresets.php. Η συνάρτηση αυτή επιλέχτηκε να είναι σε διαφορετικό αρχείο, αφενός γιατί σχεδιάστηκε να είναι εύκολο να επεκταθεί και να τροποποιηθεί με την προσθήκη νέων μεθόδων αρχικοποίησης του σχεδιασμού και αφετέρου γιατί η ύπαρξη της σε διαφορετικό αρχείο από το ίδιο το μοντέλο δίνει περισσότερη ελευθερία πειραματισμού με αυτή.

Η loadcoreAlocationPreset λαμβάνει 4 ορίσματα και επιστρέφει ένα array από box. Τα ορίσματα που λαμβάνει είναι:

- ένα αλφαριθμητικό που υποδηλώνει την προς επιλογή μέθοδο,
- οι διαστάσεις X και Y του πίνακα των στοιχείων,
- ο αριθμός των προς χρήση πυρήνων.

Η συνάρτηση με βάση το πρώτο όρισμα σχεδιάζει την κατασκευή της κατανομής που αντιστοιχεί σε αυτό. Έχουν οριστεί 2 μέθοδοι: η κατανομή σε ισομήκεις κάθετες και οριζόντιες λωρίδες. Αναλυτικά, ο κώδικας παρουσιάζεται στο Παράρτημα 7.

4.2. Σελίδα δημιουργίας αίτησης

Η διαδικασία δημιουργίας αίτησης εκτέλεσης γίνεται από την αρχική σελίδα της διεπαφής ονομαζόμενη index.php.

Η διαδικασία αποτελείται από τα εξής στάδια:

- Έλεγχος ύπαρξης μη ολοκληρωμένης εκτέλεσης
- Μία φόρμα εισαγωγής τιμών
- Ο έλεγχος ότι οι τιμές είναι εντός των επιτρεπτών συνόλων.
- Ο σχεδιασμός μιας κατανομής κόμβων, εάν χρειάζεται
- Η κλήση και κατασκευή της ίδιας της εντολής εκτέλεσης

Για την διαδικασία εισαγωγής και ελέγχου των αρχικών τιμών ορίζεται μία σειρά μεταβλητών. Για κάθε παράμετρο εισόδου ορίζονται 2 μεταβλητές, η "$c_<όνομα παραμέτρου>" και η "$c_<όνομα παραμέτρου>_err". Η πρώτη για την αποθήκευση της τιμής της μεταβλητής και η δεύτερη για την αποθήκευση ενός μηνύματος σφάλματος, στην περίπτωση που η εισαχθείσα τιμή είναι μη αποδεκτή. Επίσης ορίζεται μία λογική μεταβλητή, ονομαζόμενη $errorsfound, η οποία υποδηλώνει την ύπαρξη ή όχι μεταβλητών με μη αποδεκτές τιμές. Ο Πίνακας 4-1 περιέχει τις παράμερους εισόδου, τις αντίστοιχες μεταβλητές καθώς και τους περιορισμούς τιμών της καθεμίας.

Παράμετρος	Μεταβλητή αποθήκευσης	Περιορισμοί τιμής	Μεταβλητή σφαλμάτων	Περιγραφικό κείμενο φόρμας
Αριθμός Πυρήνων	$c_cores	$\in \mathbb{N}, < 14$	$c_cores_err	Cores
Ανοχή επαναλήψεων	$c_tolerance	$\in \mathbb{R}$	$c_tolerance_err	Tolerance
Διάσταση X	$c_x	$\in \mathbb{N}, < 1000$	$c_x_err	Dimension X
Διάσταση Y	$c_y	$\in \mathbb{N}, < 1000$	$c_y_err	Dimension Y
Εξωτερικές επαναλήψεις	$c_outer_itr	$\in \mathbb{N}$	$c_outer_itr_err	Outer Iterations
Εσωτερικές επαναλήψεις	$c_inner_itr	$\in \mathbb{N}$	$c_inner_itr_err	Krylov Iterations
Επαναλήψεις προσταθεροποίησης	$c_rdelf	$\in \mathbb{N}, < \$c\_outer\_itr$	$c_rdelf_err	r-Deflation
Μεθοδολογία αντιστοίχησης κόμβων σε επεξεργαστές	$c_core_part	$\in$ {"METIS","verticalstripes","horisontalstripes","custom"}	$c_core_part_err	Custom Partitioning

Πίνακας 4-1 Παράμετροι δημιουργίας αίτησης εκτέλεσης

4.2.1. Έλεγχος ύπαρξης μη ολοκληρωμένης κλήσης.

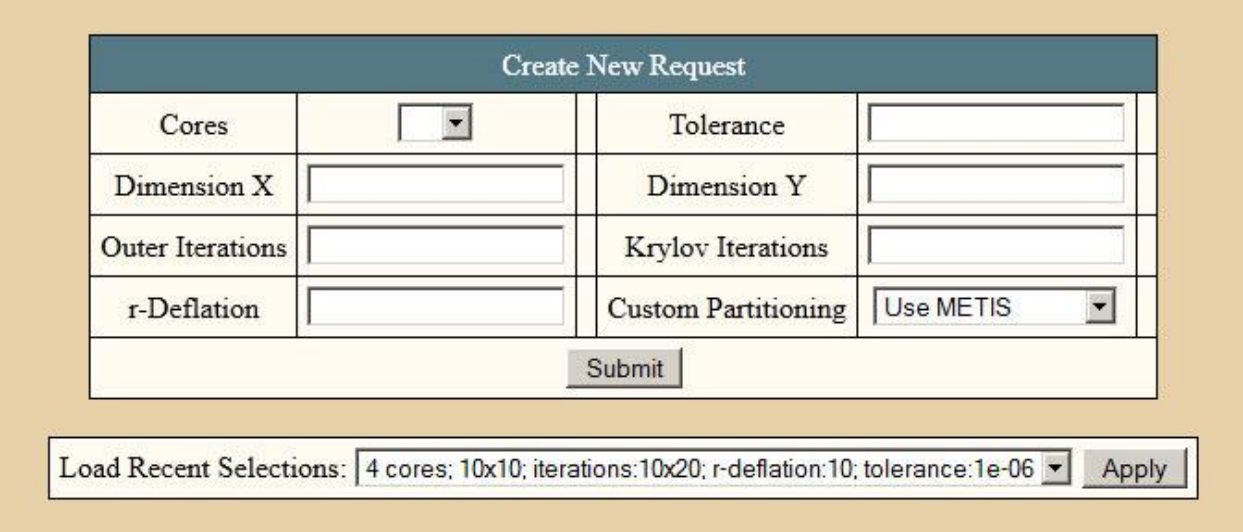
Η σελίδα είναι σχεδιασμένη ώστε να μην επιτρέπει την κλήση δύο διαφορετικών αιτήσεων ταυτόχρονα. Συνεπώς προτού εμφανίσει οτιδήποτε η αρχική σελίδα ελέγχει εάν υπάρχει προυπάρχουσα μη ολοκληρωμένη εκτέλεση. Αυτό είναι αρκετά απλό αφού έχει επισυναφθεί στην σελίδα ήδη η κεφαλίδα και συνεπώς έχει ήδη γίνει ο έλεγχος και έχει οριστεί η μεταβλητή $pendingrequests (βλ. Κεφ. 2.2.3). Αρκεί λοιπόν να γίνει έλεγχος ότι η $pendingrequests έχει τιμή μεγαλύτερη του μηδενός. Στην περίπτωση που βρεθεί υπάρχουσα μη ολοκληρωμένη εκτέλεση, η σελίδα εμφανίζει μονάχα ένα μήνυμα και ανανεώνει τον εαυτό της κάθε 5 δευτερόλεπτα μέχρι να ολοκληρωθεί η υπάρχουσα εκτέλεση. Ο κώδικας εμφανίζεται στο Σχήμα 4-4.

```
<?php

include 'lib_pageheader.php';

if ($pendingrequests > 0) {
    echo 'Request Pending... cannot create new! <br />';
    ?>
    <script language="JavaScript" type="text/javascript">
        var count =6
        function countDown(){
            if (count <=0){
                location.reload(true);
            }else{
                count--;
                document.getElementById("timer").innerHTML = "This page will refresh in "+count+" seconds."
                setTimeout("countDown()", 1000)
            }
        }
    </script>
    <span id="timer">
        <script>
            countDown();
        </script>
    </span>

    <?php
} else {

// Διαδικασία ελέγχου τιμών μεταβλητών.

}
```

Σχήμα 4-4 Κώδικας ανανέωσης σελίδας στην περίπτωση ύπαρξης ενεργής εντολής

4.2.2. Φόρμα εισαγωγής τιμών

Η φόρμα εισαγωγής τιμών κατασκευάστηκε με συγκεκριμένη δομή. Αποτελείται από εάν πίνακα html με 6 στήλες. Κάθε γραμμή περιέχει χώρο εισαγωγής 2 μεταβλητών, ο καθένας από τους οποίους καταλαμβάνει 3 στήλες: μία που περιέχει περιγραφικό κείμενο της μεταβλητής, μία που περιέχει το αντικείμενο φόρμας html που χρησιμεύει για την εισαγωγή και μία που θα εμφανιστεί το τυχόν σφάλμα ελέγχου τιμής της μεταβλητής. Ο πίνακας αυτός ακολουθείται από ένα κουμπί υποβολής και μία μικρή φόρμα που περιέχει ένα πεδίο επιλογής dropdown html, το οποίο περιέχει τιμές από προηγούμενες εκτελέσεις και είναι κουμπί για την επιλογή και ανάκτηση αυτών των τιμών προς χρήση. Το αποτέλεσμα σχεδιασμού φαίνεται στο Σχήμα 4-5.



Σχήμα 4-5 Φόρμα εισαγωγής τιμών για την δημιουργία αίτησης.

Σε κάθε τριάδα στηλών που αντιστοιχεί στην εισαγωγή μίας παραμέτρου εισάγεται στο μεν πεδίο εισαγωγής η τιμή της αντίστοιχης μεταβλητής αποθήκευσης ενώ στην τρίτη στήλη εμφανίζονται τα περιεχόμενα της αντίστοιχής μεταβλητής σφάλματων. Η τοποθέτηση των περιεχομένων αυτών των μεταβλητών με αυτό τον τρόπο οδηγεί στην προβολή των τιμών που είχαν εισαχθεί και των σφαλμάτων στην καθεμία (εάν γίνει εισαγωγή τιμών εκτός ορίων), και συνεπώς στην ευκολότερη διόρθωση της αίτησης. Το όνομα καθώς και το id του πεδίου εισαγωγής ορίζονται ίδια με την ονομασία της αντίστοιχης μεταβλητής αποθήκευσής για λόγους διευκόλυνσης της χρήσης του και της τιμής που αποστέλλει μετά την συμπλήρωση. Ένα παράδειγμα κώδικα φαίνεται στο Σχήμα 4-6.

```
<td> Dimension Y </td>
<td> <input type="text" name="c_y" id="c_y"  value="<?php echo $c_y; ?>" > </td>
<td><?php echo $c_y_err; ?></td>
```

Σχήμα 4-6 Παράδειγμα κώδικα HTML πεδίου εισαγωγής τιμών.

Μετά το πάτημα του πλήκτρου υποβολής (submit), γίνεται επανάκληση της ίδια σελίδας, επισυνάπτοντας στην διεύθυνση της μια σειρά από μεταβλητές και τιμές. Οι μεταβλητές που εισάγονται λαμβάνουν το όνομα τους από το όνομα του πεδίου και την τιμή τους από την τιμή του πεδίου. Π.χ. η κλήση του πλήκτρου υποβολής, εάν στο input του παραδείγματος στο Σχήμα 4-6 έχει εισαχθεί η τιμή 30, θα επισύναπτε στην διεύθυνση το ”c_y=30”. Επίσης στο πλήκτρο υποβολής έχει και αυτό λάβει όνομα και τιμή “Submit”, οπότε αυτόματα επισυνάπτει στην κλήση “Submit=Submit”, και έτσι περνάει η πληροφορία στην επόμενη κλήση της σελίδας ότι το πλήκτρο έχει πατηθεί.

Για την κατασκευή του συστήματος επιλογής ρυθμίσεων γίνεται μία κλήση στην βάση για τις τελευταίες 10 μοναδικές ομάδες παραμέτρων από τις προϋπάρχουσες εγγραφές. Με βάση αυτές κατασκευάζεται ένα αντικείμενο τύπου select στην HTML, με τιμές κάθε επιλογής ένα αλφαριθμητικό που περιέχει τις τιμές των παραμέτρων σε σειρά διαχωρισμένες από τον χαρακτήρα “:”. Στο κουμπί

επιλογής ορίζεται στην παράμετρο του “onclick” μία συνάρτησης javascript ώστε να γίνει κλήση αυτής όταν αυτό πατηθεί. Τέλος ορίζεται και στο select και στο κουμπί επιλογής στην παράμετρο “autopostback” η τιμή false ώστε να μην γίνει επανάκληση της σελίδας με την κλήση αυτής της διαδικασίας και να γίνει μόνο κλήση της συνάρτησης στο onclick. Ο κώδικας του τμήματος αυτού φαίνεται στο Σχήμα 4-7.

```
Load Recent Selections:
<select name="preset_select" id="preset_select" autopostback="false">
    <?php
    $result = mysql_query("SELECT DISTINCT dimx, dimy, cores, outerit, innerit, tolerance, rdeflation FROM
GMRES_Requests Order By id DESC LIMIT 10 ");
    while ($row = mysql_fetch_array($result)) {
      echo ' <option value="' . $row['cores'] . ':' . $row['dimx'] . ':' . $row['dimy'] . ':' . $row['outerit'] . ':' .
$row['innerit'] . ':' . $row['tolerance'] . ':' . $row['rdeflation'] . '">' . $row['cores'] . ' cores; ' . $row['dimx'] . 'x' .
$row['dimy'] . '; iterations:' . $row['outerit'] . 'x' . $row['innerit'] . '; r-deflation:' . $row['rdeflation'] . '; tolerance:' .
$row['tolerance'] . '</option>';
    }
    ?>
</select>
<button id="applyPreset"' onclick="applyPreset()" autopostback="false">Apply</button>
```

Σχήμα 4-7 Κώδικας κατασκευής επιλογής παραμέτρων τελευταίων εκτελέσεων.

Η συνάρτηση που καλείται στο javascript χρησιμοποιεί την βιβλιοθήκη jquery για τον εντοπισμό των πεδίων προς αλλαγή και την ανάθεση τιμών σε αυτά. Η βιβλιοθήκη jquery επιτρέπει τον εντοπισμό και την χρήση των πεδίων στην σελίδα με την κλήση “$(“#id”)”. Ο χαρακτήρας “#” πριν το πεδίο id του αντικειμένου που επιθυμείται υποδηλώνει ότι το κείμενο που ακολουθεί είναι id. Εφόσον το αντικείμενο HTML που εντοπίζεται μέσω της συνάρτησης “$(“#id”)” είναι πεδίο φόρμας, είναι δυνατόν να ληφθεί και να αλλάξει η τιμή των περιεχομένων του με την χρήση το μέλους του αντικειμένου val. Εφόσον το val κληθεί χωρίς όρισμα, επιστρέφει την τρέχουσα τιμή, ενώ εάν κληθεί με όρισμα, αλλάζει την τιμή των περιεχομένων του πεδίου στο δοθέν όρισμα [**7**].

Στην συνάρτηση αυτή αρχικά λαμβάνεται η τιμή του dropdown που περιέχει την ομάδα παραμέτρων και μετατρέπεται σε array με την χρήση της εντολής split, που είναι η αντίστοιχη στη javascript της συνάρτησης explode της php (πρβλ. Κεφ. Παράρτημα 5), με διαχωριστικό χαρακτήρα τον “:”. Κατόπιν η κάθε τιμή παραμέτρου της ομάδας ορίζεται σαν τιμή στο αντίστοιχο πεδίο εισαγωγής.

Η χρήση javascript [**6**] για αυτήν την λειτουργία επιτρέπει την ανανέωση των τιμών των πεδίων χωρίς να γίνει επανάκληση της σελίδας και συνεπώς, η όλη διαδικασία είναι γρήγορη και αισθητικά πιο φιλική για τον χρήστη. Ο κώδικας της συνάρτησης javascript φαίνεται στο Σχήμα 4-8.

```
<script language="javascript" type="text/javascript" >
   function applyPreset(){
      presetstr=$("#preset_select").val();
      if (!(presetstr == '')){
         var presetArray = presetstr.split(":");
         $("#c_cores").val(presetArray[0]);
         $("#c_x").val(presetArray[1]);
         $("#c_y").val(presetArray[2]);
         $("#c_outer_itr").val(presetArray[3]);
         $("#c_inner_itr").val(presetArray[4]);
         $("#c_tolerance").val(presetArray[5]);
         $("#c_rdefl").val(presetArray[6]);
      }
   }
</script>
```

Σχήμα 4-8 Κώδικας εισαγωγής τιμών από το σύστημά επιλογής τελευταίων εκτελέσεων.

4.2.3. Διαδικασία ελέγχου τιμών εισόδου.

Εφόσον γίνει επανάκληση της σελίδας, κατόπιν της κλήσης της εφαρμογής, ξεκινά η διαδικασία ελέγχου των τιμών των παραμέτρων.

Οι παράμετροι, όπως έχουν επισυναφθεί στην διεύθυνση της σελίδας, λαμβάνονται από την php από την προκαθορισμένη array $_GET [**15**]. Η $_GET περιέχει ονομασμένα στοιχεία ορισμένα από τις παραμέτρους της διεύθυνσης της σελίδας. Π.χ. εάν το παράδειγμα στο Σχήμα 4-6 έχει εισαχθεί η τιμή 30 στην php, θα είχαμε το ”$_GET[‘c_y’]” να έχει την τιμή 30. Όπως έχει αναφερθεί προηγουμένως, το πάτημα του πλήκτρου υποβολής από μόνο του επισυνάπτει την παράμετρο “Submit=Submit” στην διεύθυνση της σελίδας. Επομένως η ύπαρξη ορισμένης τιμής Submit στην $_GET υποδηλώνει ότι η συγκεκριμένη φόρτωση της σελίδας έγινε μετά από εισαγωγή στοιχείων και συνεπώς πρέπει να γίνει έλεγχος τιμών και εάν οι τιμές είναι αποδεκτές κλήση της εφαρμογής. Ο έλεγχος ότι μία μεταβλητή έχει τιμή στην php γίνεται με την συνάρτηση isset η οποία λαμβάνει όρισμα την μεταβλητή και επιστρέφει μια λογική μεταβλητή που είναι true εάν η προς έλεγχο μεταβλητή έχει ορισμένη τιμή.

Η διαδικασία ελέγχου ξεκινάει εφόσον βρεθεί ορισμένη η $_GET[‘Submit’]. Κατόπιν αρχικοποιείται η μεταβλητή $errorsfound με τιμή false και γίνεται έλεγχος σε κάθε παράμετρο ότι έχει οριστεί και είναι εντός των επιθυμητών τιμών. Αν μία από αυτές βρεθεί εκτός αποδεκτών τιμών ή μη ορισμένη, τότε στην αντίστοιχη μεταβλητή σφάλματος καταγράφεται ένα μήνυμα σφάλματος και η $errorsfound λαμβάνει την τιμή true. Ένα παράδειγμα ελέγχου μίας μεταβλητής εμφανίζεται στο Σχήμα 4-9. Το μόνο πεδίο που ακολουθεί διαφορετική λογική είναι το πεδίο επιλογής τρόπου κατανομής κόμβων σε επεξεργαστές. Το συγκεκριμένο, εάν βρεθεί μη ορισμένο λαμβάνει αυτόματα την τιμή “METIS” που υποδηλώνει την μη εισαγωγή κατανομής και την χρήση του συστήματος METIS [**10**]. Επίσης λαμβάνεται από το $_GET και άλλη μια μεταβλητή ονομαζόμενη c_core_part_str, για την οποία δεν υπάρχει πεδίο σε αυτή την φόρμα. Αυτή με μεταβλητή περιέχει ένα αλφαριθμητικό που περιέχει την κατανομή των κόμβων του πλέγματος όπως έχει μορφοποιηθεί σύμφωνα με την μορφοποίηση το κεφ.

4.1. Αυτή η μεταβλητή λαμβάνει τιμές από την σελίδα κατανομής των κόμβων του πλέγματος εάν είχε κληθεί.

```
$c_tolerance_err = '';
$c_tolerance = $_GET["c_tolerance"];
if (!(isset($c_tolerance))) {
        $c_tolerance_err = "Field Required!";
        $errorsfound = true;
} else {
        if (!is_numeric($c_tolerance)) {
                $c_tolerance_err = "Invalid Value, must be a number!";
                 $errorsfound = true;
          }
}
```

Σχήμα 4-9. Παράδειγμα κώδικα ελέγχου παραμέτρου.

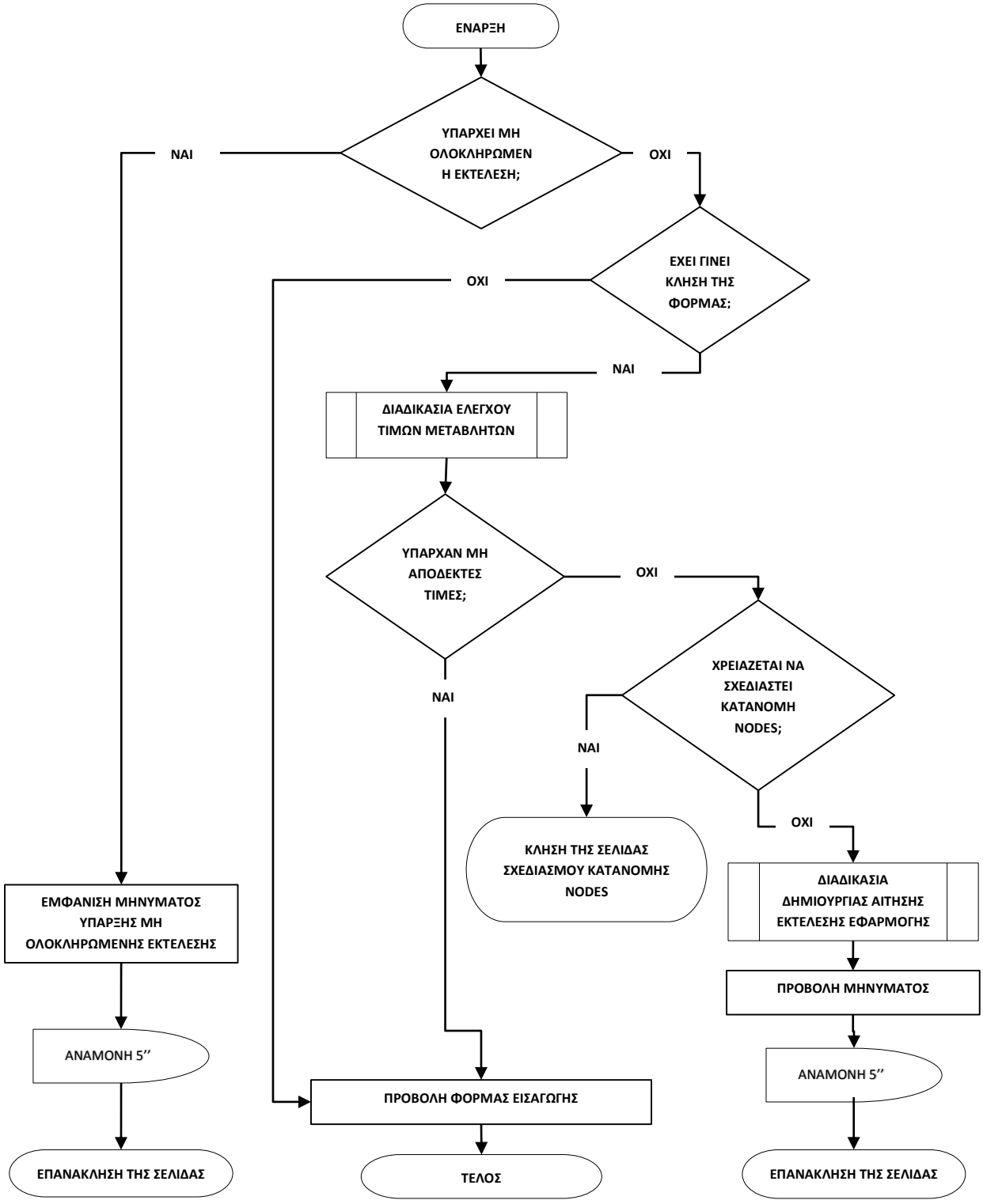
Εφόσον δεν βρεθεί κανένα σφάλμα, διερευνάται η επιλογή της μεθόδου κατανομής των κόμβων του πλέγματος, όπως έχει οριστεί στην μεταβλητή $c_core_part:

- Εάν έχει επιλεγεί η τιμή “custom”, που αντιστοιχεί στην επιλογή σχεδιασμού της κατανομής από τον χρήστη, τότε ελέγχεται η ύπαρξη η όχι τιμής στην μεταβλητή $c_core_part_str. Η $c_core_part_str λαμβάνει τιμές μόνο εάν έχει γίνει κλήση αυτής της σελίδας από την σελίδα σχεδιασμού κατανομής και περιέχει την κατανομή. Εφόσον η μεταβλητή αυτή είναι κενή γίνεται κλήση της σελίδας σχεδιασμού κατανομής με ακριβώς τις ίδιες παράμερους και σταματάει η εκτέλεση αυτής της σελίδας. Εάν δεν είναι κενή προστίθεται η βιβλιοθήκη με τις συναρτήσεις διαχείρισης της κατανομής στον κώδικα και η σελίδα συνεχίζει κανονικά.
- Εάν έχει επιλεγεί η τιμή “METIS”, που υποδηλώνει την χρήση της προκαθορισμένης μεθοδολογίας, απλά αποθηκεύεται αυτή η επιλογή και η διεργασία συνεχίσει κανονικά.
- Τέλος, εάν έχει λάβει οποιαδήποτε άλλη τιμή, αυτή η τιμή αντιστοιχεί σε μια προκαθορισμένη μεθοδολογία κατανομής, όπως υπάρχουν στην συνάρτηση loadcoreAlocationPreset (πρβλ Κεφ.4.1.2). Κατόπιν προστίθενται η βιβλιοθήκη με τις συναρτήσεις διαχείρισης της κατανομής και η βιβλιοθήκη με την συνάρτηση loadcoreAlocationPreset και κατασκευάζεται το αλφαριθμητικό που αντιστοιχεί στην επιλεγμένη μεθοδολογία και αποθηκεύεται στη $c_core_part_str.

Αφού ολοκληρωθεί η διερεύνηση της κατανομής, ξεκινάει η διαδικασία κλήσης της εντολής εκτέλεσης, όπως αναλύθηκε στο Κεφ. 3.2.3.β. Το κύριο κομμάτι του κώδικα που εκτελείται φαίνεται στο Σχήμα 4-10 και το διάγραμμα ροής της αρχικής σελίδας στο Σχήμα 4-11.

```
$errorsfound = true;
 $c_submit = $_GET["Submit"];
if (isset($c_submit)) {

// Διαδικασίες ελεγχου τιμών μεταβλητων.

        $c_core_part_err = '';
        $c_core_part = $_GET["c_core_part"];
        if (!(isset($c_core_part))) {
           $c_core_part = "METIS";
         }
        $c_core_part_str = $_GET["c_core_part_str"];
        if (!(isset($c_core_part_str))) {
           $c_core_part_str = "";
        }
 }


 if (!$errorsfound) {
             $custom_nodes = 0;
              if ($c_core_part == "custom") {
                        $custom_nodes = 1;
                        if ($c_core_part_str == "") {
                            $urlredirect = "coreallocation.php?c_cores=" . $c_cores . "&c_x=" . $c_x.
                            "&c_y=" . $c_y . "&c_outer_itr=" . $c_outer_itr . "&c_inner_itr=" . $c_inner_itr
                            . "&c_rdefl=" . $c_rdefl ."&c_tolerance=" . $c_tolerance;
                               ?>
                                <script language="JavaScript" type="text/javascript">
                                window.location.href = '<?php echo $urlredirect; ?>';
                                </script>
                                <?php
                                 die();
                        }
                       include 'lib_coreallocation.php';


                } elseif ($c_core_part == "METIS") {
                      $custom_nodes = 0;
                } else {
                      include 'lib_coreallocation.php';
                      include 'lib_coreallocationpresets.php';

                      $custom_nodes = 1;
                     $c_core_part_str = exportstr(loadcoreAlocationPreset($c_core_part, $c_x, $c_y,
                    $c_cores));
                  }

//Κλήση Service για εκτέλεση εφαρμογής
} else {
// Προβολή Φόρμας
}
```

Σχήμα 4-10 Κώδικας ελέγχου ροής αρχικής σελίδας.

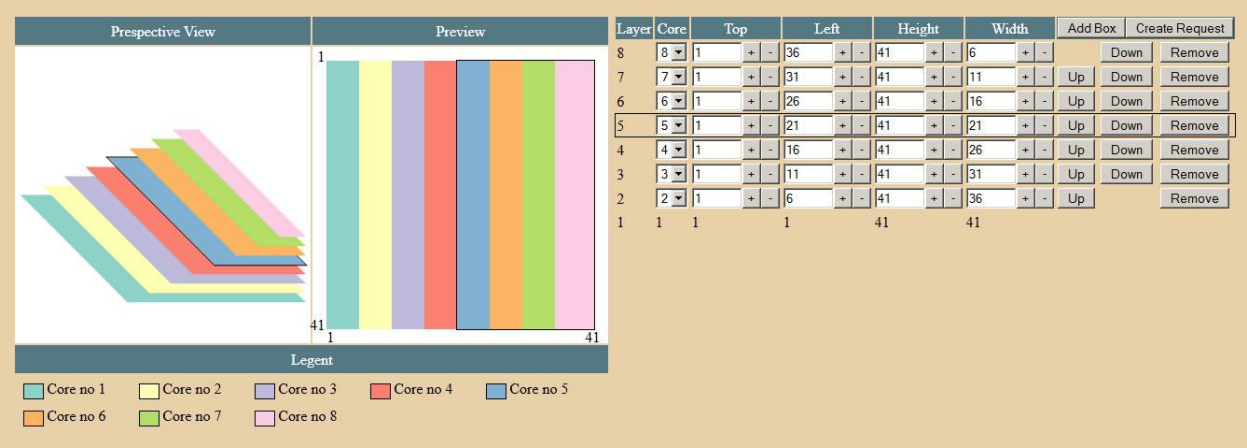


Σχήμα 4-11 Διάγραμμα ροής αρχικής σελίδας

4.3. Σελίδα κατανομής των κόμβων του πλέγματος

Η σελίδα κατανομής των κόμβων του πλέγματος αποτελείται από 4 τμήματα όπως φαίνεται στο Σχήμα 4-12:

- Μια εικόνα προεπισκόπησης της κατανομής των κόμβων του πλέγματος
- Μια εικόνα προβολής του μοντέλου υπό γωνία, ώστε να γίνεται κατανοητό εύκολα από τον χρήστη
- Ένα υπόμνημα με τα χρώματα που χρησιμοποιούνται στις εικόνες
- Και τέλος, έναν πίνακα εισαγωγής των τιμών.



Σχήμα 4-12 Φόρμα σχεδιασμού κατανομής των κόμβων του πλέγματος.

Η σελίδα λαμβάνει τις παραμέτρους που χρησιμοποιεί και η σελίδα δημιουργίας αίτησης εκτέλεσης (πρβλ. Πίνακας 4-1), χωρίς τις μεταβλητές σφάλματος, τις οποίες λαμβάνει κατά την κλήση της από την σελίδα δημιουργίας αίτησης εκτέλεσης. Μετά την ολοκλήρωση της διαδικασίας κατανομής τις χρησιμοποιεί για να ξανακαλέσει τη σελίδα δημιουργίας αίτησης εκτέλεσης. Οι παράμετροι αυτές λαμβάνονται με την χρήση της $_GET και, εφόσον δεν βρεθεί τιμή, ορίζεται μια για την καθεμία για την λειτουργία της σελίδας. Για την δημιουργία αρχικής τιμής για την $c_core_part_str χρησιμοποιούνται οι συναρτήσεις από τις βιβλιοθήκες μοντελοποίησης της κατανομής των κόμβων του πλέγματος για την κατασκευή της προκαθορισμένης κατανομής σε κάθετες στήλες για ένα πρόβλημα των δοθείσων διαστάσεων. Ένα παράδειγμα τέτοιας λήψης φαίνεται στο Σχήμα 4-13.

```
$c_cores = $_GET["c_cores"];
if (!(isset($c_cores))) {
   $c_cores = "5";
}
$c_core_part_str = $_GET["c_core_part_str"];
if (!(isset($c_core_part_str))) {
   $c_core_part_str = exportstr(loadcoreAlocationPreset("verticalstripes", $c_x, $c_y, $c_cores));
}
```

Σχήμα 4-13. Παραδείγματα λήψης παραμέτρων.

Επειδή δεν είναι χρήσιμες όλες οι παράμετροι ξεχωριστά, μετά την λήψη τους κατασκευάζεται μια σειρά από μεταβλητές:

- η $requeststr, που περιέχει όλες τις παραμέτρους με τις τιμές τους έτοιμες για επισύναψη στην διεύθυνση μιας σελίδας,
- οι $c_nny και $c_nny, που περιέχουν τις διαστάσεις του πλέγματος κόμβων υπολογισμένες από τις διαστάσεις του προβλήματος,
- η $drawstep, που περιέχει την κλίμακα σχεδιασμού για την εικόνα προεπισκόπησης,
- οι $drawx και $drawy, που είναι οι διαστάσεις της εικόνας προεπισκόπησης,
- οι $movestep_x και $movestep_y, που είναι το βήμα κατά το οποίο κα πλήκτρα της φόρμας εισαγωγής θα μετακινούν της αντίστοιχες μεταβλητές τους,
- η $coreAllocation_boxes, που είναι το array με τα box που αντιπροσωπεύει το αλφαριθμητικό της $c_core_part_str,
- η $boxcount, που περιέχει το πλήθος των στοιχείων της $coreAllocation_boxes.

Ο κώδικας κατασκευής των παραπάνω μεταβλητών φαίνεται στο Σχήμα 4-14.

```
$requeststr = "?c_cores=" . $c_cores . "&c_x=" . $c_x . "&c_y=" . $c_y . "&c_outer_itr=" . $c_outer_itr
. "&c_inner_itr=" . $c_inner_itr . "&c_rdefl=" . $c_rdefl ."&c_tolerance=" . $c_tolerance .
"&c_core_part=" . $c_core_part . "&c_core_part_str=";

$c_nnx = $c_x * 2 + 1;
$c_nny = $c_y * 2 + 1;
$drawstep_x = floor(300 / $c_nnx);
$drawstep_y = floor(300 / $c_nny);
$drawstep = $drawstep_x;
if ($drawstep_y < $drawstep) {
   $drawstep = $drawstep_y;
}
$drawx = $drawstep * $c_nnx;
$drawy = $drawstep * $c_nny;

if ($c_nnx > 100) {    $movestep_x = 5;
} else {    $movestep_x = 1; }
if ($c_nny > 100) {    $movestep_y = 5;
} else {    $movestep_y = 1; }

$coreAlocation_boxes = parsestr($c_core_part_str);
$boxcount = 0;
foreach ($coreAlocation_boxes as $box) {
   if ($box['z'] > $boxcount) {
      $boxcount = $box['z'];
   }
}
```

Σχήμα 4-14 Κώδικας ορισμού αρχικών τιμών μεταβλητών σελίδας κατανομής των κόμβων του πλέγματος.

Εκτός από τις παραπάνω παραμέτρους χρησιμοποιείται και η παράμετρος $c_draw_action. Η παράμετρος αυτή χρησιμεύει στην αποστολή στην σελίδα των εντολών διαγραφής ή δημιουργίας νέων στοιχείων στην array των box. Η παράμετρος αυτή λαμβάνει είτε την τιμή “addbox”, που υποδηλώνει την εντολή προσθήκης νέου στοιχείου στην array των box, είτε την τιμή “removebox_#” που υποδηλώνει την εντολή διαγραφής του στοιχείου στην θέση #. Η διαγραφή γίνεται με την κατασκευή ενός νέου array και *την προσθήκη* σε αυτό όλων των στοιχείων πλην του προς διαγραφή, φροντίζοντας ταυτόχρονα να ελαττωθεί η τιμή z σε κάθε box με z μεγαλύτερο του διαγραμμένου κατά ένα, ώστε τα z να παραμείνουν συνεχείς αριθμοί. Ο κώδικας αυτών των εντολών φαίνεται στο Σχήμα 4-15.

```
$c_draw_action = "" . $_GET["draw_action"];
if ($c_draw_action == "addbox") {
   $boxcount+=1;
   $new_coreAlocation_boxes[$boxcount] = array(
      'z' => $boxcount,
      'core' => 1,
      'left' => 1,
      'top' => 1,
      'right' => 1,
      'bottom' => 1);
   for ($i = $boxcount - 1; $i > 1; $i--) {
      $new_coreAlocation_boxes[$i] = $coreAlocation_boxes[$i];
   }
   $coreAlocation_boxes = $new_coreAlocation_boxes;
} elseif (substr($c_draw_action, 0, 10) == "removebox_") {
   $rowid = substr($c_draw_action, 10);
   $boxcount-=1;
   for ($i = $boxcount; $i > 1; $i--) {
      if ($i >= $rowid) {
         $new_coreAlocation_boxes[$i] = $coreAlocation_boxes[$i + 1];
         $new_coreAlocation_boxes[$i]['z'] = $i;
      } else {
         $new_coreAlocation_boxes[$i] = $coreAlocation_boxes[$i];
      }
   }
   $coreAlocation_boxes = $new_coreAlocation_boxes;
   echo $rowid;
}
```

Σχήμα 4-15 Κώδικας διαγραφής και δημιουργίας νέου στοιχείου στην array των box.

Η σελίδα χρησιμοποιεί μια σειρά από javascript συναρτήσεις για τον έλεγχο των τιμών, την προβολή της προεπισκόπησης και την εκτέλεση των εντολών. Η χρήση javascript επιτρέπει στον σχεδιασμό μιας δυναμικής προεπισκόπησης των αλλαγών του χρήστη χωρίς ανανέωση της σελίδας και ελαττώνει τον φόρτο του server, επιταχύνοντας ταυτόχρονα την διαδικασία για τον χρήστη.

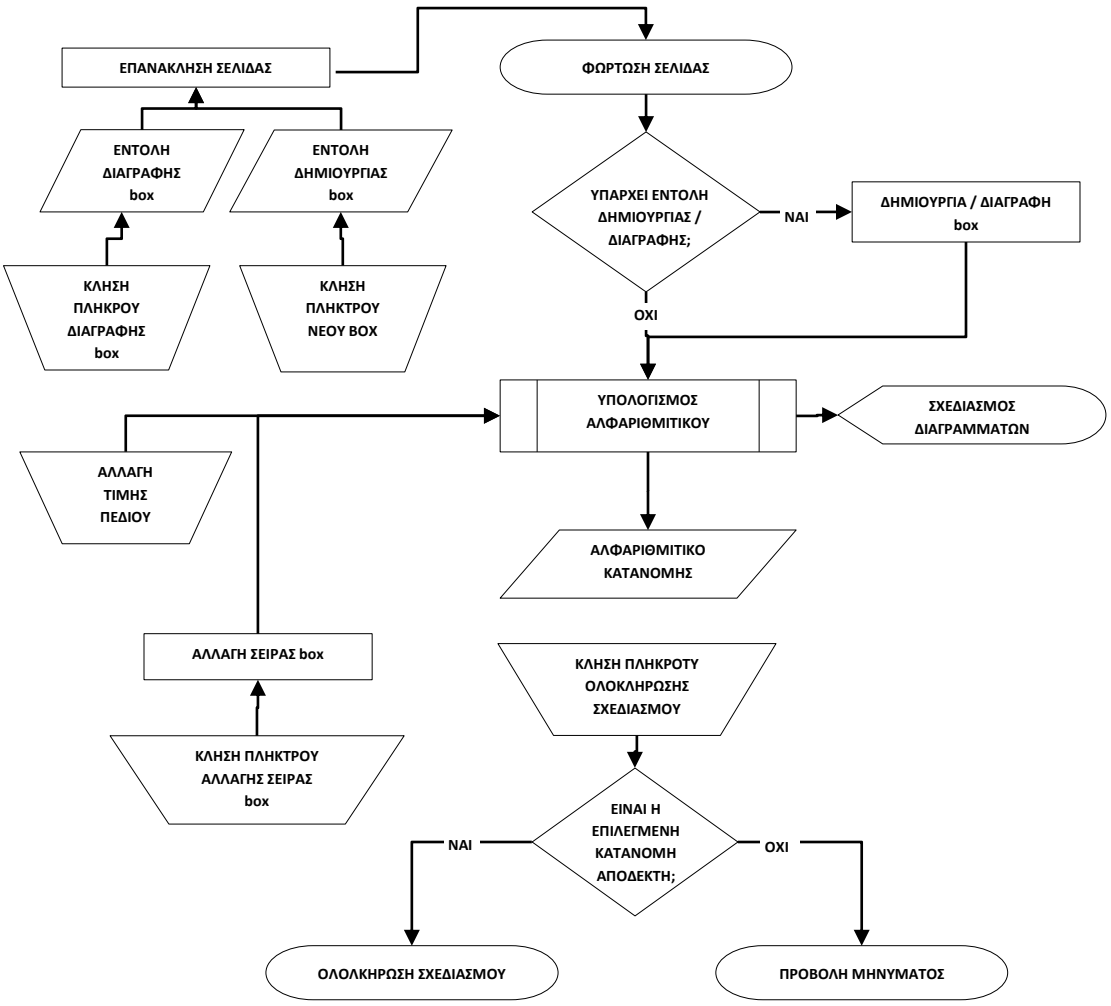
Για να μπορέσει να γίνει αυτή η δυναμική προεπισκόπηση, οι εικόνες δεν είναι αρχεία εικόνων αλλά αποτελούνται από μια σειρά στοιχείων HTML τύπου div τοποθετημένα έτσι ώστε οπτικά το αποτέλεσμα να είναι ισοδύναμο. Επειδή το μοντέλο σχεδιασμού που έχει επιλεχτεί χρησιμοποιεί παραλληλόγραμμα για τον σχεδιασμό της κατανομής, είναι εύκολο να χρησιμοποιηθούν απλά στοιχεία HTML που είναι εξ' ορισμού παραλληλόγραμμα για τον σχεδιασμό της προβολής. Τα δύο αυτά διαγράμματα, της προεπισκόπησης και υπο γωνία προβολής, κατασκευάζονται σχεδιάζοντας ένα στοιχείο τύπου div για κάθε box της array. Κάνοντας click σε οποιοδήποτε από αυτά εμφανίζεται ένα μαύρο περίγραμμα και στα δύο div που αντιστοιχούν στο box αλλά και στην γραμμή του πίνακα που του αντιστοιχεί, ώστε να υπάρχει μια οπτική σύνδεση μεταξύ του πίνακα τιμών και των διαγραμμάτων.

Ο πίνακας εισαγωγής τιμών περιέχει μια γραμμή για κάθε box και 7 στήλες:

- Η πρώτη δεν είναι δυνατόν να επεξεργαστεί και είναι ένας αριθμός που υποδηλώνει το ύψος (z) στο οποίο σχεδιάζεται το θεωρητικό παραλληλόγραμμο που αντιστοιχεί στο box.
- Η δεύτερη είναι τύπου dropdown και περιέχει σαν επιλογές τους διαθέσιμους επεξεργαστές και χρησιμεύει στην αντιστοίχιση επεξεργαστή με το box. Όταν αλλάξει τιμή, καλεί τον επανυπολογισμό του αλφαριθμητικού που αντιστοιχεί στην κατανομή μέσω javascript.
- Οι επόμενες τέσσερεις στήλες περιέχουν η κάθε μία ένα πεδίο εισαγωγής τιμών και δύο πλήκτρα με το κείμενο + και – αντίστοιχα. Το κάθε ζεύγος πλήκτρων αυξάνει ή μειώνει την τιμή του αντίστοιχου πεδίου κατά ένα προπροσδιορισμένο βήμα, ανάλογα με τις διαστάσεις του πλέγματος. Οι στήλες περιέχουν την αριθμητική τιμή των συντεταγμένων της άνω αριστερά γωνίας, το ύψος και το μήκος του θεωρητικού παραλληλόγραμμου που αντιστοιχεί στο box και καλούν τον επανυπολογισμό του αλφαριθμητικού που αντιστοιχεί στην κατανομή μέσω javascript όποτε αλλάξουν τιμή.
- Η τελευταία στήλη περιέχει μία σειρά πλήκτρων για την εκτέλεση διαφόρων λειτουργιών. Τα πλήκτρα αυτά εμφανίζονται εφόσον είναι δυνατό να εκτελεστούν οι εντολές που αντιστοιχεί στο καθένα. Το πλήκτρο up ανταλλάσσει το ύψος του επιλεγμένου box με το ακριβώς από πάνω του ενώ το πλήκτρο down με το ακριβώς από κάτω του. Τα πλήκτρα αυτά εμφανίζονται εφόσον υπάρχει παραπάνω ή παρακάτω box αντίστοιχα από το επιλεγμένο. Το πλήκτρο remove επανακαλεί την σελίδα διαγράφοντας το επιλεγμένο box.

Στην κεφαλίδα περιέχονται 2 πλήκτρα: ένα που επανακαλεί την σελίδα για την προσθήκη νέου box και μία που ελέγχει και ολοκληρώνει την διαδικασία σχεδιασμού και κατασκευάζει την εντολή εκτέλεσης. Τέλος ορίζεται και ένα κρυφό πεδίο για την αποθήκευση του αλφαριθμητικού που αντιστοιχεί στην κατανομή.

Το διάγραμμα ροής της διαδικασίας φαίνεται στο Σχήμα 4-16 και στο Παράρτημα 8 παρουσιάζεται αναλυτικά η διαδικασία σχεδιασμού και ο κώδικας της σελίδας.



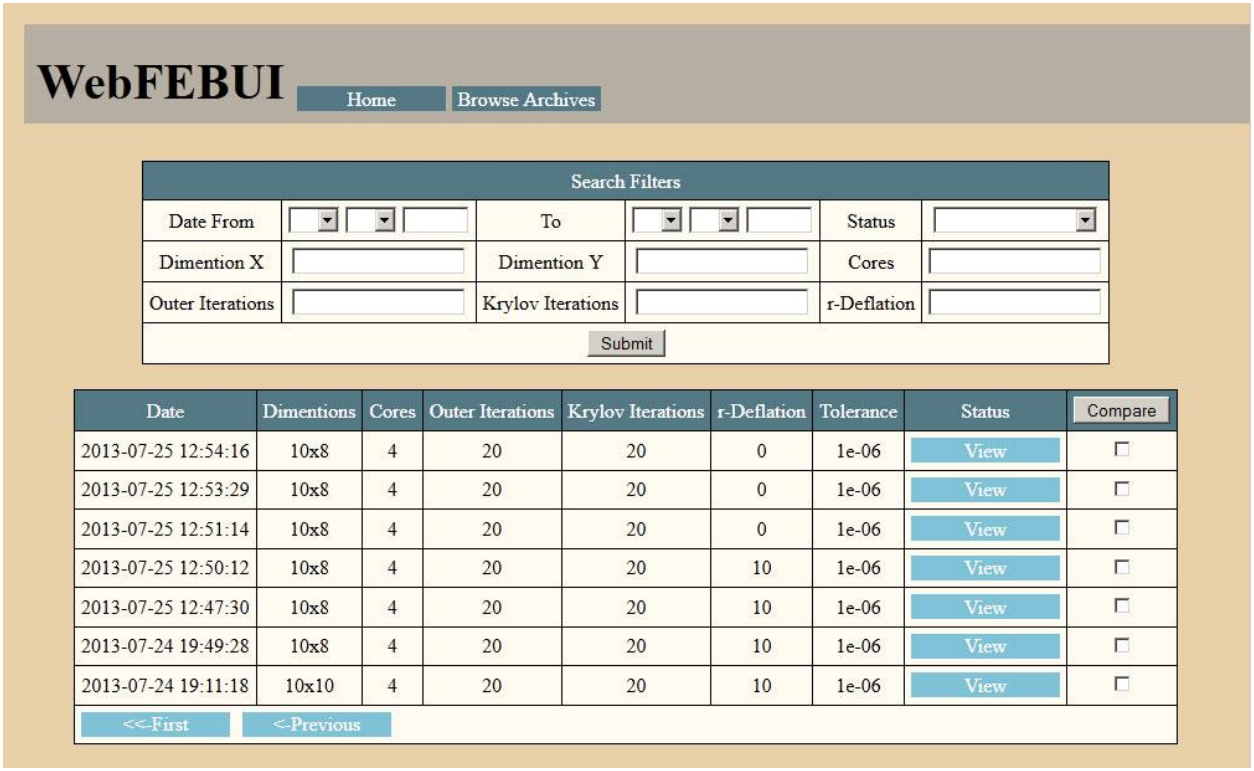
Σχήμα 4-16 Διάγραμμα ροής σχεδιασμού κατανομής κόμβων του πλέγματος.

5. Προβολή αποτελεσμάτων

Η προβολή και η ανάλυση των αποτελεσμάτων γίνεται σε τρεις σελίδες με την βοήθεια δυο διαγραμμάτων. Η διαδικασία προβολής αποτελεσμάτων ξεκινάει είτε από την σελίδα εύρεσης παλαιών εκτελέσεων είτε αυτόματα μετά την ολοκλήρωση μιας εκτέλεσης. Η σελίδα εύρεσης παλαιών εκτελέσεων δίνει πρόσβαση στις σελίδες προβολής λεπτομερειών εκτέλεσης και συγκριτικής προβολής εκτελέσεων, όπου φαίνονται αναλυτικά τα αποτελέσματα μίας ή περισσοτέρων εκτελέσεων αντίστοιχα.

5.1. Εύρεση παλαιών εκτελέσεων

Η σελίδα εύρεσης παλαιών εκτελέσεων αποτελείται από δύο τμήματα: μια φόρμα φίλτρων εύρεσης και έναν πίνακα προβολής αποτελεσμάτων, όπως φαίνεται στο Σχήμα 5-1.



Σχήμα 5-1 Σελίδα εύρεσης παλαιών εκτελέσεων.

Η φόρμα φίλτρων χρησιμοποιεί ένα σύνολο μεταβλητών για την μεταφορά των πληροφοριών όπως και η φόρμα δημιουργίας (πρβλ Κεφ. 4.2). Οι μεταβλητές αυτές ξεκινάνε όλες με τους χαρακτήρες “$c_” και εισάγονται από το array $_GET [**15**] από παραμέτρους τους με το ίδιο όνομα. Η φόρμα περιέχει τα εξής φίλτρα και χρησιμοποιεί τις εξής μεταβλητές:

- δυο φίλτρα που υποδηλώνουν ένα διάστημα ημερομηνιών για τον περιορισμό της ημερομηνίας εκτέλεσης. Η κάθε ημερομηνία αποθηκεύεται σε 3 μεταβλητές: μια για την ημέρα, μια για τον μήνα και μία για το έτος, ο χαρακτηρισμός των οποίων γίνεται με την προσθήκη των χαρακτήρων “d_day_”,”d_month_” ή ”d_year_” στο όνομα της μεταβλητής αντίστοιχα. Τέλος στο όνομα της μεταβλητής προστίθενται οι χαρακτήρες “f” ή “t” για να χαρακτηρίσουν το κάτω και το πάνω όριο της ζώνης εύρεσης αντίστοιχα. Π.χ. η μεταβλητή “$c_d_month_f” αντιστοιχεί στον μήνα την ημερομηνίας από την οποία ζητείται η ημερομηνία εκτέλεσης να είμαι μεγαλύτερη.
- ένα φίλτρο χαρακτηρισμού της εκτέλεσης σαν ολοκληρωμένης που αποθηκεύεται στη μεταβλητή “$c_completed”.
- την διάσταση X του προς επίλυση συνόλου, που αποθηκεύεται στη μεταβλητή “$c_dim_x”.
- την διάσταση Y του προς επίλυση συνόλου, που αποθηκεύεται στη μεταβλητή “$c_dim_y”.
- το πλήθος των πυρήνων που χρησιμοποιήθηκαν, που αποθηκεύεται στη μεταβλητή “$c_cores”.
- τον αριθμό των εξωτερικών επαναλήψεων της εκτέλεσης, που αποθηκεύεται στη μεταβλητή “$s_outer_itr ”.
- τον αριθμό των εσωτερικών επαναλήψεων της εκτέλεσης, που αποθηκεύεται στη μεταβλητή “$c_inner_itr ”.
- τον αριθμό των επαναλήψεων της προσταθεροποίησης της εκτέλεσης, που αποθηκεύεται στη μεταβλητή “$c_rdef ”.

Για την διαδικασία της εύρεσης χρησιμοποιούνται επίσης μερικές βοηθητικές μεταβλητές:

- η $page, που υποδηλώνει την σελίδα του πίνακα ευρέσεως που εμφανίζεται αυτή την στιγμή. Εισάγεται σαν παράμετρος στην σελίδα από το array $_GET.
- η $whereclause, ένα αλφαριθμητικό που περιέχει τα φίλτρα όπως αυτά θα χρησιμοποιηθούν στο SQL query και κατασκευάζεται με βάση τις επιλογές στην φόρμα για την τελευταία κλήση της.
- η $extrahtml, ένα αλφαριθμητικό που κατασκευάζει τις παραμέτρους όπως αυτές θα είχαν κατασκευαστεί εάν ξαναγινόταν ακριβώς η ίδια κλήση της φόρμας εύρεσης με την τελευταία. Χρησιμεύει στην σελιδοποίηση του πίνακα αποτελεσμάτων.

Η σελίδα αφού καλεστεί, δίνει αρχικές τιμές στις βοηθητικές μεταβλητές και στη συνέχεια, της δίνεται σαν τιμή, για κάθε μεταβλητή της φόρμας που δεν έχει τιμή, ένα κενό αλφαριθμητικό, ενώ εφόσον έχει τιμή προστίθενται στην μεν μεταβλητή $whereclause το φίλτρο για την sql, αρχίζοντας από “AND” με ένα κενό πριν πάντα, και στην μεταβλητή $extrahtml η τιμή για εισαγωγή στην διεύθυνση της σελίδας. Ένα παραδείγμα αυτού του τμήματος κώδικα φαίνεται στο Σχήμα 5-2. Αφού ολοκληρωθεί η διαδικασία κατασκευής της, εάν η $whereclause έχει τουλάχιστον ένα χαρακτήρα θα πρέπει να ξεκινάει με τους χαρακτήρες “ AND “, και συνεπώς η αντικατάσταση των 5 πρώτων αυτών χαρακτήρων με τους χαρακτήρες “ WHERE “ μετατρέπει το περιεχόμενο του αλφαριθμητικού στην σωστή δομή για χρήση σε

sql query [**13**]. Μετά την αρχικοποίηση των μεταβλητών των παραμέτρων ακολουθούν ο σχεδιασμός της φόρμας εύρεσης και η προβολή του πίνακα των αποτελεσμάτων, όπως φαίνεται στο Σχήμα 5-3.

```
if (!(isset($s_rdef))) {
   $s_rdef = "";
} else {
   $s_rdef = mysql_real_escape_string($s_rdef);
   if ($s_rdef != "") {
      $whereclause .= " AND rdeflation='$s_rdef' ";
      $extrahtml .= "&s_rdef=" . $s_rdef;
   }
}
```

Σχήμα 5-2 Παράδειγμα κώδικα συμπλήρωσης $whereclause για αλφαριθμητικό.

```
<?php
include "lib/pageheader.php";

$page = $_GET["page"];
$extrahtml = "";
$whereclause = "";

// Συμπλήρωση $whereclause

if ($whereclause != "") {
   $whereclause = " WHERE " . substr($whereclause, 5);
}
if (!(isset($page))) {
   $page = 1;
} else {
   $page = mysql_real_escape_string($page);
}
?>
<Center>
   <form>
// Σχεδιασμός φόρμας εισόδου.
   </form>
// Σχεδιασμός πίνακα αποτελεσμάτων..
</Center>
```

Σχήμα 5-3 Κώδικας κεντρικής δομής σελίδας εύρεσης παλαιών εκτελέσεων.

Η φόρμα εύρεσης αποτελείται από ένα πίνακα HTML με 6 στήλες. Κάθε γραμμή του πίνακα περιέχει 3 πεδία εύρεσης, το καθένα από τα οποία καταλαμβάνει δυο κελιά: το πρώτο περιέχει την περιγραφή του πεδίου και το δεύτερο ένα πεδίο εισαγωγής της τιμής προς εύρεση. Στο πεδίο εισαγωγής εισάγεται η τιμή της αντίστοιχης μεταβλητής, ώστε να φαίνονται οι τιμές που είχαν εισαχθεί στην τελευταία κλήση. Όπως και στην σελίδα δημιουργίας αίτησης, το όνομα καθώς και το id του πεδίου εισαγωγής ορίζονται ίδια με την ονομασία της αντίστοιχης μεταβλητής αποθήκευσης. Ένα παράδειγμα

κώδικα φαίνεται στο Σχήμα 5-4. Μετά τον σχεδιασμό των πεδίων σχεδιάζεται και ένα πλήκτρο υποβολής της φόρμας.

```
<tr>
 <td> Dimention X </td><td>  <input type="text" name="s_dim_x"  value="<?php echo $s_dim_x; ?>" > </td>
 <td> Dimention Y </td><td>  <input type="text" name="s_dim_y"  value="<?php echo $s_dim_y; ?>" >  </td>
<td> Cores </td><td>  <input type="text" name="s_cores"  value="<?php echo $s_cores; ?>" >  </td>
</tr>
```

Σχήμα 5-4 Παράδειγμα κώδικα σχεδιασμού φόρμας αναζήτησης παλαιών εκτελέσεων.

Ο πίνακας προβολής αποτελεσμάτων περιέχει ένα σύστημα σελιδοποίησης. Το σύστημα σελιδοποίησης αποτελείται από δύο τμήματα: έναν αρχικό κώδικα που φιλτράρει τα προς προβολή αποτελέσματα, ο οποίος βρίσκεται πριν το σχεδιασμό του πίνακα, και ένα σύνολο από πλήκτρα για την κίνηση μεταξύ σελίδων, το οποίο βρίσκεται μετά. Ο αριθμός της σελίδας που προβάλλεται αυτή την στιγμή είναι αποθηκεμένος στην μεταβλητή $page, που έχει ήδη εισαχθεί σαν παράμετρος της σελίδας, ενώ το πλήθος των εγγραφών ανά σελίδα περιέχεται στην παράμετρο $page_rows που έχει στατική τιμή.

Προτού αρχίσει η διαδικασία της σελιδοποίησης είναι απαραίτητος ο υπολογισμός του πλήθους των προς προβολή εγγραφών για να υπολογιστεί το πλήθος των σελίδων. Γίνεται συνεπώς μία κλήση της sql με τα φίλτρα, όπως έχουν συμπληρωθεί στην μεταβλητή $whereclause για την λήψη του πλήθους των εγγραφών, και η τιμή αποθηκεύεται στην μεταβλητή $rows. Με βάση αυτήν και το πλήθος των εγγραφών ανά σελίδα υπολογίζεται το πλήθος των σελίδων για τα δεδομένα που υπάρχουν και αποθηκεύεται στην μεταβλητή $last. Κατόπιν ελέγχεται ότι η σελίδα που έχει δηλωθεί στην $page έχει λογική τιμή, δηλαδή έχει τιμή ανάμεσα στο ένα και το πλήθος των σελίδων που υπάρχουν, και εάν όχι της δίνεται η κοντινότερη λογική τιμή. Εφόσον η $page έχει πλέον λογική τιμή κατασκευάζεται μια αλφαριθμητική μεταβλητή $max που περιέχει τον κώδικα φιλτραρίσματος των εγγραφών για sql με την χρήση της εντολής limit. Με την βοήθεια των $whereclause και αυτής κατασκευάζεται και καλείται το sql query που περιέχει τις απαιτούμενες πληροφορίες από τις προς προβολή εγγραφές για αυτή την σελίδα και ξεκινάει ο σχεδιασμός του ιδίου του πίνακα.

Ο πίνακας περιέχει εννιά στήλες. Οι επτά πρώτες περιέχουν πληροφορίες για την συγκεκριμένη εγγραφή, η όγδοη περιέχει έναν σύνδεσμο (link) προς την σελίδα προβολής λεπτομερειών εκτέλεσης ενώ η ένατη χρησιμοποιείται για την συγκριτική προβολή των αποτελεσμάτων. Ο σύνδεσμος της όγδοης στήλης είναι ένα απλό HTML link στην σελίδα “details.php” με πρόσθετη παράμετρο ανά γραμμή την μεταβλητή myID, που λαμβάνει σαν τιμή το κλειδί της εγγραφής που αντιστοιχεί στην συγκεκριμένη γραμμή. Εφόσον η εντολή έχει ολοκληρωθεί επιτυχώς, ο σύνδεσμος έχει το κείμενο “view”, ενώ εάν απέτυχε, το κείμενο “failed” . Για την συγκριτική προβολή των αποτελεσμάτων χρησιμοποιείται μια πιο περίπλοκη διαδικασία. Ολόκληρος ο πίνακας προβολής αποτελεσμάτων βρίσκεται μέσα σε μια φόρμα HTML που καλεί την σελίδα συγκριτικής προβολής, ενώ η κεφαλίδα της ένατης στήλης είναι ένα πλήκτρο submit που καλεί αυτή την φόρμα. Σε κάθε γραμμή μέσα στο κελί της ένατης στήλης βρίσκεται ένα checkbox με την παράμετρο name ίση με “myID[]” και value το κλειδί της εγγραφής που αντιστοιχεί στην συγκεκριμένη γραμμή. Η χρήση των χαρακτήρων “[]” στην παράμετρο name υποδηλώνει ότι αυτά τα checkbox αποτελούν τιμές ενός κοινά ονομασμένου array με το όνομα myID. Κατά την υποβολή, αυτό

θα οδηγήσει στην δημιουργία ενός array ”myID” με τιμές τα value όλων των τσεκαρισμένων checkboxes. Αυτό το array λαμβάνεται μέσω της $_GET όπως και κάθε άλλη παράμετρος από την php.

Αφού ληφθούν οι γραμμές και εκτυπωθούν τα δεδομένα τους στον πίνακα, σχεδιάζονται οι σύνδεσμοι αλλαγής σελίδων. Για την δημιουργία των συνδέσμων αυτών χρησιμοποιείται η μεταβλητή $extrahtml. Η $extrahtml περιέχει όλες τις παραμέτρους για την κλήση της σελίδας που έχει προβληθεί αυτήν την στιγμή, πλην του αριθμού της σελίδας των αποτελεσμάτων που προβάλλεται. Συνεπώς η επισύναψή της σε έναν σύνδεσμο, που οδηγεί σε αυτή την σελίδα php με διαφορετικό τον αριθμό της σελίδας των αποτελεσμάτων, θα οδηγήσει στην προβολή άλλης σελίδας αποτελεσμάτων με τα ίδια ακριβώς φίλτρα. Έτσι εφόσον αυτή η σελίδα αποτελεσμάτων αυτή δεν είναι η πρώτη, εκτυπώνονται δύο σύνδεσμοι που οδηγούν στην πρώτη σελίδα ή στην προηγούμενη αυτής και, εφόσον δεν είναι η τελευταία, εκτυπώνονται δύο σύνδεσμοι που οδηγούν στην τελευταία σελίδα ή στην επόμενη αυτής και ολοκληρώνεται το σύστημα της σελιδοποίησης των αποτελεσμάτων.

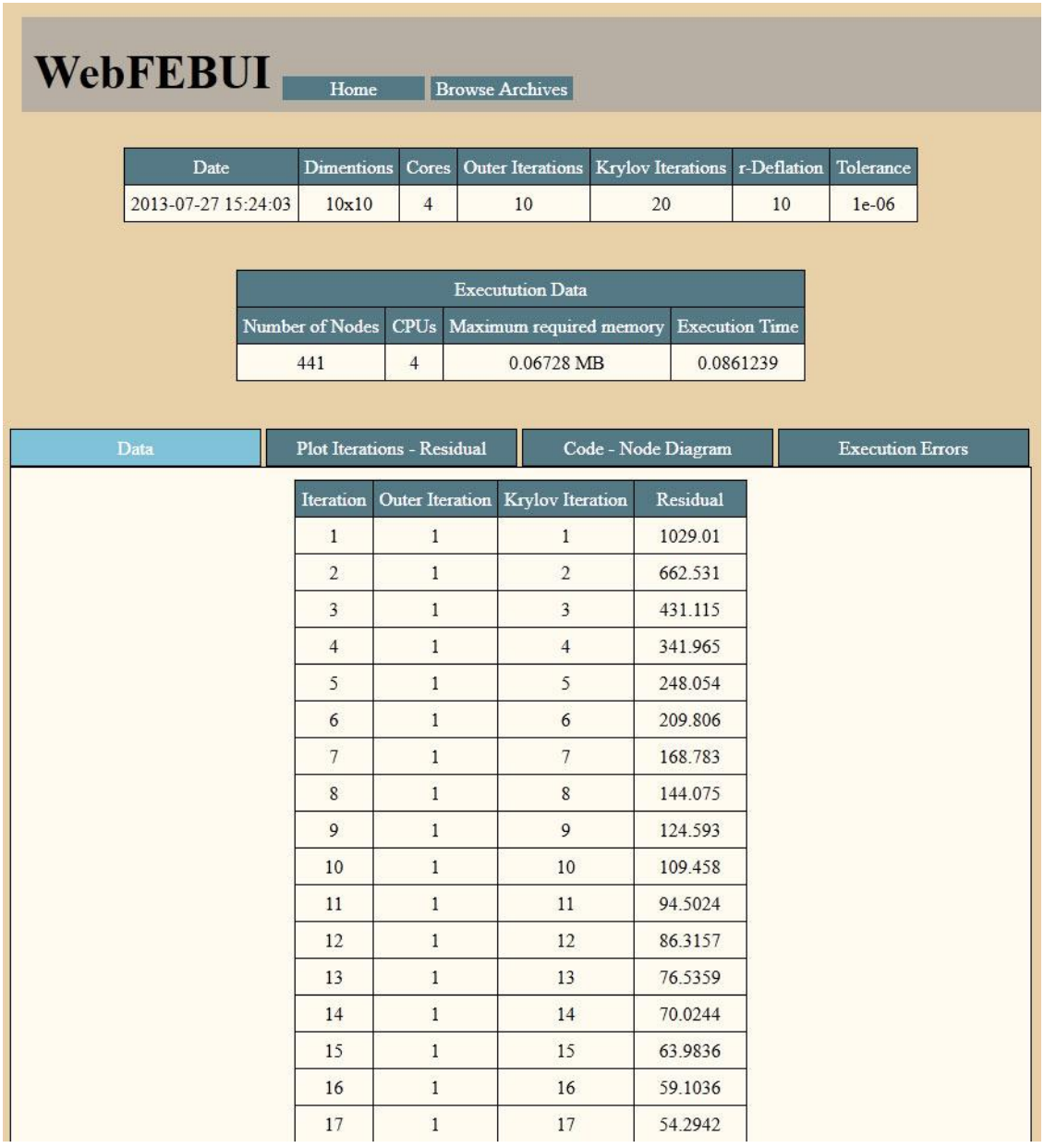
Ο κώδικας σχεδιασμού του πίνακα αυτού φαίνεται στο Σχήμα 5-5.

```
    $data = mysql_query("SELECT count(id) c FROM GMRES_Requests $whereclause ") or die(mysql_error());
    $countrow = mysql_fetch_array($data);
    $rows = $countrow['c'];
    $page_rows = 20;
    $last = ceil($rows / $page_rows);
    if ($page < 1) {
        $page = 1;
    } elseif ($page > $last) {
        $page = $last;
    }
    $max = 'limit ' . ($page - 1) * $page_rows . ',' . $page_rows;
    $result = mysql_query("SELECT * FROM GMRES_Requests  $whereclause ORDER BY `create_date` DESC $max
");
    echo '<form action="compare.php" method="get"> <table class="listtable" cellpadding="0" cellspacing="0" ><tr>
<td class="tableheader" > Date </td> <td class="tableheader" > Dimentions </td> ';
    echo '<td class="tableheader" > Cores </td><td class="tableheader" > Outer Iterations </td>';
    echo '<td class="tableheader" > Krylov Iterations </td><td class="tableheader" > r-Deflation </td>';
    echo '<td class="tableheader" > Tolerance </td><td class="tableheader" > Status </td><td class="tableheader" >
<input type="submit" value="Compare"> </td></tr>';
    while ($row = mysql_fetch_array($result)) {
        echo "<tr><td>" . $row['create_date'] . "</td>";
        echo "<td>" . $row['dimx'] . "x" . $row['dimy'] . "</td>";
        echo "<td>" . $row['cores'] . "</td>";
        echo "<td>" . $row['outerit'] . "</td>";
        echo "<td>" . $row['innerit'] . "</td><td>" . $row['rdeflation'] . "</td>";
        echo "<td>" . $row['tolerance'] . "</td><td>";
        if ($row['complete']==1) {
            echo '<a href="details.php?myID=' . $row['id'] . '" > View </a>';
        } elseif ($row['complete']==2) {
            echo '<a href="details.php?myID=' . $row['id'] . '" > Failed </a>';
        } else  {echo " Running... ";}
        echo '</td><td><input type="checkbox" name="myID[]" value="' . $row['id'] . '"></td></tr>';
    }
    echo  '<tr ><td colspan=9 > <table cellpadding="0" cellspacing="0" border="0" > <tr><td width="120px"
style="padding:0px 0px 0px 0px; border:none; "  >';
    if ($page != 1) {
        echo " <a href='{$_SERVER['PHP_SELF']}?page=1$extrahtml'> <<-First</a> "; }
    echo '</td><td width="100%" style="padding:0px 0px 0px 10px; border:none; ">';
    if ($page != 1) {
        $previous = $page - 1;
        echo " <a href='{$_SERVER['PHP_SELF']}?page=$previous$extrahtml'> <-Previous</a> ";}
    echo '</td> <td  width="120px" style="padding:0px 10px 0px 0px; border:none; ">';
    if ($page != $last) {
        $next = $page + 1;
        echo " <a href='{$_SERVER['PHP_SELF']}?page=$next$extrahtml'>Next -></a> ";}
    echo '</td> <td  width="120px" style="padding:0px 0px 0px 0px; border:none; ">';
    if ($page != $last) {
        echo " <a href='{$_SERVER['PHP_SELF']}?page=$last$extrahtml'>Last ->></a> ";}
    echo '</td></tr></table></td>  </tr></table> </form>';
```

Σχήμα 5-5 Κώδικας σχεδιασμού πίνακα αποτελεσμάτων σελίδας εύρεσης παλαιών εκτελέσεων.

5.2. Προβολή λεπτομερειών

Η σελίδα προβολής λεπτομερειών αποτελείται από τρία τμήματα όπως φαίνεται στο Σχήμα 5-6. Τα δύο πρώτα είναι δύο πίνακες, που περιέχουν τις πληροφορίες της αίτησης εκτέλεσης και τα αποτελέσματα της εκτέλεσης αντίστοιχα. Το τρίτο τμήμα αποτελείται από 4 καρτέλες (tabs): μία που περιέχει τη σύγκλιση των επαναλήψεων της GMRES σαν πίνακα, μία που περιέχει το διάγραμμα σύγκλισης (Σχήμα 5-7), μία που περιέχει το διάγραμμα κατανομής των κόμβων του πλέγματος (Σχήμα 5-8) και μία που περιέχει τα σφάλματα εκτέλεσης.



WebFEBUI Home Browse Archives

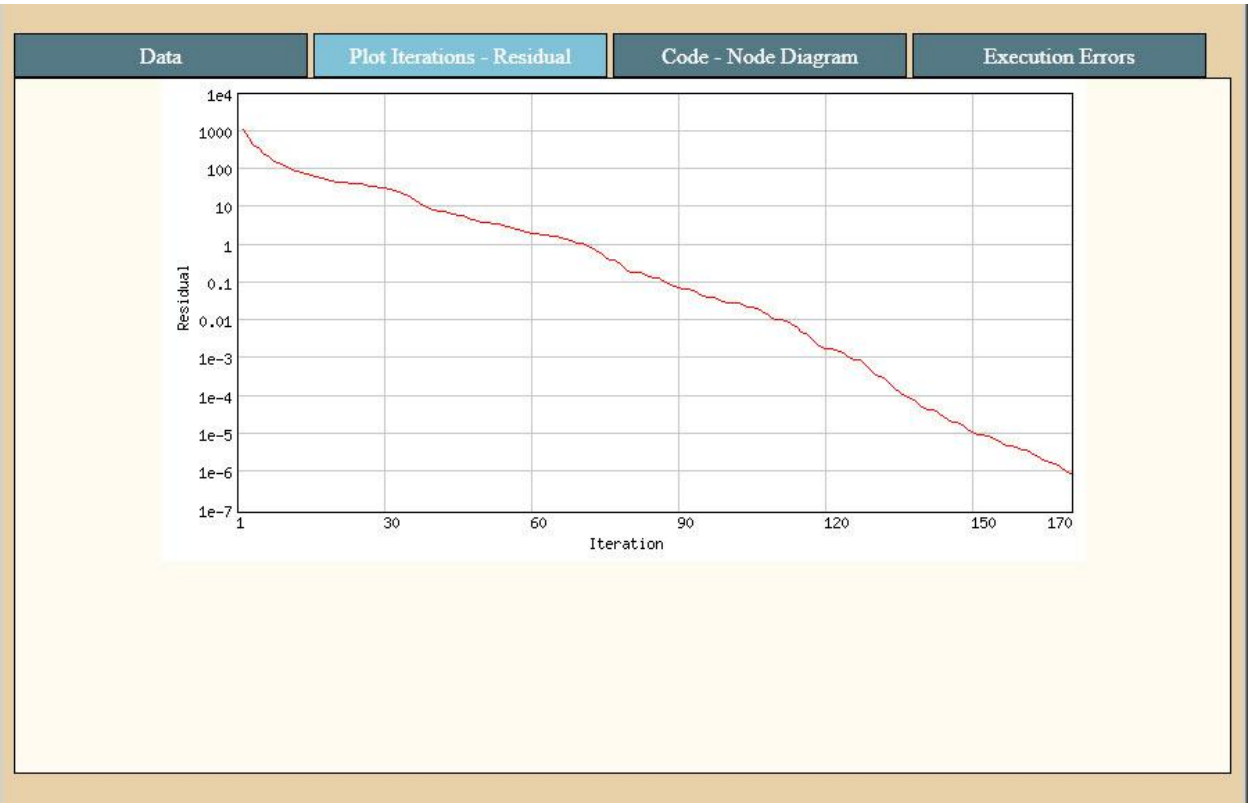
Date	Dimentions	Cores	Outer Iterations	Krylov Iterations	r-Deflation	Tolerance
2013-07-27 15:24:03	10x10	4	10	20	10	1e-06

Executution Data			
Number of Nodes	CPUs	Maximum required memory	Execution Time
441	4	0.06728 MB	0.0861239

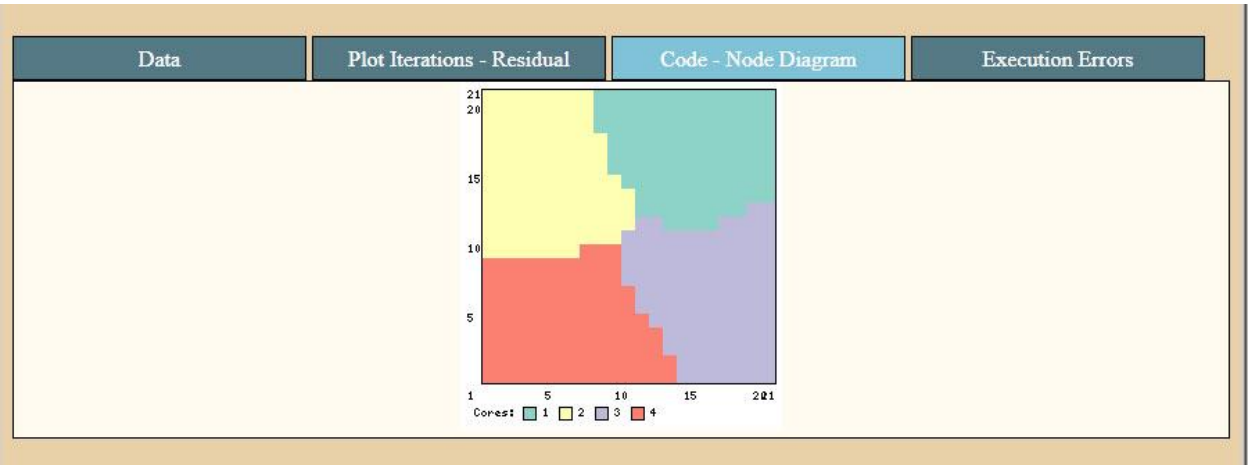
Data | Plot Iterations - Residual | Code - Node Diagram | Execution Errors

Iteration	Outer Iteration	Krylov Iteration	Residual
1	1	1	1029.01
2	1	2	662.531
3	1	3	431.115
4	1	4	341.965
5	1	5	248.054
6	1	6	209.806
7	1	7	168.783
8	1	8	144.075
9	1	9	124.593
10	1	10	109.458
11	1	11	94.5024
12	1	12	86.3157
13	1	13	76.5359
14	1	14	70.0244
15	1	15	63.9836
16	1	16	59.1036
17	1	17	54.2942

Σχήμα 5-6 Σελίδα προβολής λεπτομερειών.



Σχήμα 5-7 Προβολή διαγράμματος σύγκλισης στην σελίδα λεπτομερειών.



Σχήμα 5-8 Προβολή διαγράμματος κατανομής των κόμβων του πλέγματος στην σελίδα λεπτομερειών.

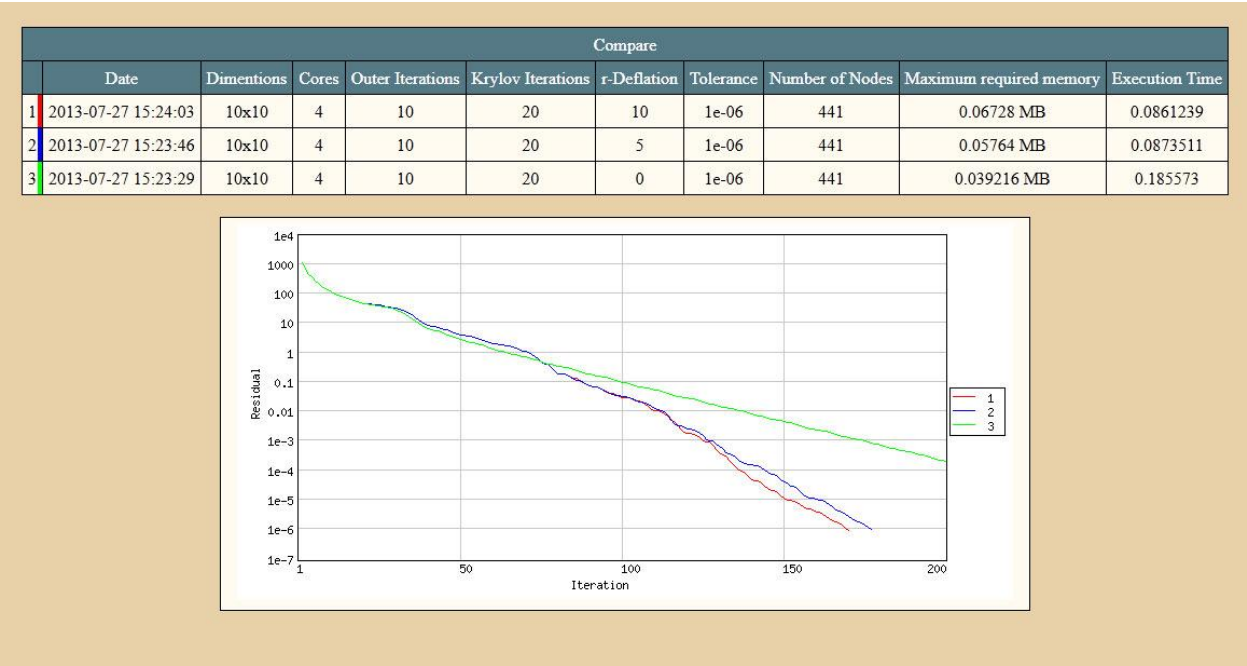
Ο κώδικας της σελίδας είναι αρκετά απλός. Αφού λάβει η σελίδα την παράμετρο myID, που περιέχει το κλειδί της εκτέλεσης της οποίας τα αποτελέσματα πρόκειται να προβληθούν, γίνεται με την χρήση αυτής η κλήση στην βάση, λαμβάνεται το array με τις μεταβλητές της εκτέλεσης και εκτυπώνονται οι δύο πίνακες με τις πληροφορίες. Κατόπιν σχεδιάζονται οι καρτέλες με την χρήση στοιχείων HTML div εντός των οποίων εκτυπώνονται οι πληροφορίες που τους αντιστοιχούν. Η προβολή των διαγραμμάτων γίνεται με την κλήση των αντίστοιχων σελίδων php για το σχεδιασμό του κάθε διαγράμματος, σαν να ήταν εικόνα, επισυνάπτοντας σε αυτή το κλειδί της εγγραφής σαν παράμετρο. Η προβολή των στοιχείων div σαν καρτέλες επιτυγχάνεται με την χρήση javascript και css (βλ. Παράρτημα 9). Ο κώδικας φαίνεται στο Σχήμα 5-9.

```
<center>
<?php
include 'lib_pageheader.php';
$id = $_GET["myID"];
if (isset($id)) {

   $id = mysql_real_escape_string($id);
   $result = mysql_query("SELECT * FROM GMRES_Requests WHERE id='$id' ");
   $row = mysql_fetch_array($result);

   if (isset($row)) {

      echo ' <table class="listtable" cellpadding="0" cellspacing="0" ><tr> <td class="tableheader" > Date </td>
<td class="tableheader" > Dimentions </td> ';
      echo '<td class="tableheader" > Cores </td><td class="tableheader" > Outer Iterations </td>';
      echo '<td class="tableheader" > Krylov Iterations </td><td class="tableheader" > r-Deflation </td>';
      echo '<td class="tableheader" > Tolerance </td></tr>';
      echo "<tr><td>" . $row['create_date'] . "</td>";
      echo "<td>" . $row['dimx'] . "x" . $row['dimy'] . "</td>";
      echo "<td>" . $row['cores'] . "</td>";
      echo "<td>" . $row['outerit'] . "</td>";
      echo "<td>" . $row['innerit'] . "</td><td>" . $row['rdeflation'] . "</td>";
      echo "<td>" . $row['tolerance'] . "</td>";
      echo "</tr></table><br /> <br />";

      echo ' <table class="listtable" cellpadding="0" cellspacing="0" ><tr> <td class="tableheader" colspan="4" >
Executution Data </td></tr><tr> <td class="tableheader" > Number of Nodes </td> ';
      echo '<td class="tableheader" > CPUs </td><td class="tableheader" > Maximum required memory </td>';
      echo '<td class="tableheader" > Execution Time </td></tr>';
      echo "<tr><td>" . $row['exec_np'] . "</td>";
      echo "<td>" . $row['exec_cores'] . "</td>";
      echo "<td>" . $row['exec_memory'] . " MB </td>";
      echo "<td>" . $row['exec_time'] . "</td>";
      echo "</tr></table>";
      ?>
         <br />
         <br />
// Σχεδιασμός  TABS.
   }
}
```

Σχήμα 5-9 Κώδικας σελίδας προβολής λεπτομερειών.

5.3. Συγκριτική προβολή

Η σελίδα συγκριτικής προβολής αποτελείται από δυο τμήματα: ένα πίνακα συγκριτικής προβολής των τιμών των εκτελέσεων και ένα διάγραμμα συγκριτικής προβολής των σύγκλισεων, όπως φαίνεται στο Σχήμα 5-10.



Compare

	Date	Dimentions	Cores	Outer Iterations	Krylov Iterations	r-Deflation	Tolerance	Number of Nodes	Maximum required memory	Execution Time
1	2013-07-27 15:24:03	10x10	4	10	20	10	1e-06	441	0.06728 MB	0.0861239
2	2013-07-27 15:23:46	10x10	4	10	20	5	1e-06	441	0.05764 MB	0.0873511
3	2013-07-27 15:23:29	10x10	4	10	20	0	1e-06	441	0.039216 MB	0.185573

Σχήμα 5-10 Σελίδα συγκριτικής προβολής εκτελέσεων.

Η σελίδα αρχικά λαμβάνει μια array τιμών κλειδιών εγγραφών σαν την παράμετρο ”myID”, που λαμβάνεται μέσω της $_GET και έχει συμπληρωθεί από τις τσεκαρισμένες εγγραφές στην σελίδα εύρεσης παλαιών εκτελέσεων (βλ. κεφ. 5.1). Με βάση αυτά τα κλειδιά λαμβάνονται από την βάση οι πληροφορίες της κάθε εκτέλεσης.

Οι πληροφορίες εκτυπώνονται ανά γραμμή σε ένα συγκριτικό πίνακα στην σελίδα λαμβάνοντας ένα αύξοντα αριθμό για την εύκολη αντιστοίχιση τους με τις καμπύλες των διαγραμμάτων. Επίσης σε κάθε γραμμή εμφανίζεται η εικόνα του διαγράμματος σύγκλισης με μια παράμετρο color με τιμή τον νέο αύξοντα αριθμό. Αυτό οδηγεί στην εμφάνιση, αντί του διάγραμματος, μιας μικρής εικόνας με το χρώμα της καμπύλης που αντιστοιχεί στο διάγραμμα σε αυτόν τον αύξοντα αριθμό, για να διευκολύνει περαιτέρω στην αντιστοίχιση μεταξύ πίνακα και διαγράμματος.

Αφού εκτυπωθεί ο πίνακας εμφανίζεται και το διάγραμμα σύγκλισης με παράμετρο myID με τιμή ένα αλφαριθμητικό που περιέχει τα κλειδιά των εγγραφών τα οποία έχουν εμφανιστεί στον πίνακα διαχωρισμένα με τον χαρακτήρα “:”. Το διάγραμμα εμφανίζεται τότε σαν συγκριτικό διάγραμμα και χαράζει όλες τις καμπύλες όλων αυτών των εγγραφών.

Ο κώδικας της σελίδας παρουσιάζεται στο Σχήμα 5-11.

```php
<?php
include 'lib_pageheader.php';

$id_array = $_GET["myID"];

if (isset($id_array)) {
   arsort($id_array);
   $idwhere = "";
   foreach ($id_array as $id) {
      $idwhere .= "'" . mysql_real_escape_string($id) . "',";
   }
   $idwhere = substr($idwhere, 0, -1);

   $result = mysql_query("SELECT*    FROM GMRES_Requests WHERE id in (" . $idwhere . ") order by id desc
;");
   echo '<center> <table class="listtable" cellpadding="0" cellspacing="0" ><tr><td class="tableheader"
colspan="12"> Compare </td> </tr> <tr> <td class="tableheader" > </td><td class="tableheader" > Date </td> <td
class="tableheader" > Dimentions </td> ';
   echo '<td class="tableheader" > Cores </td><td class="tableheader" > Outer Iterations </td>';
   echo '<td class="tableheader" > Krylov Iterations </td><td class="tableheader" > r-Deflation </td>';
   echo '<td class="tableheader" > Tolerance </td><td class="tableheader" > Number of Nodes </td> ';
   echo '<td class="tableheader" > Maximum required memory </td>';
   echo '<td class="tableheader" > Execution Time </td></tr>';
   $i = 0;
   while ($row = mysql_fetch_array($result)) {
      $i++;
      echo '<tr><td style="background-image:url(\'img/iterationdiagram.php?color=' . $i . '\');background-
repeat:repeat-y;background-position:right top; ">' . $i . '</td>';
      echo "<td>" . $row['create_date'] . "</td>";
      echo "<td>" . $row['dimx'] . "x" . $row['dimy'] . "</td>";
      echo "<td>" . $row['cores'] . "</td>";
      echo "<td>" . $row['outerit'] . "</td>";
      echo "<td>" . $row['innerit'] . "</td><td>" . $row['rdeflation'] . "</td>";
      echo "<td>" . $row['tolerance'] . "</td>";
      echo "<td>" . $row['exec_np'] . "</td>";
      echo "<td>" . $row['exec_memory'] . " MB </td>";
      echo "<td>" . $row['exec_time'] . "</td>";
      echo "</tr>";
   }
   echo "</table><br />";
   $idstr = '';
   foreach ($id_array as $id) {
      $idstr.=$id . ":";  }
   $idstr = substr($idstr, 0, -1);
   echo '<div class="listtable" style="padding:5px;width:740px;">';
   echo '<img src="img/iterationdiagram.php?myID=' . $idstr . '"  /></div> </center> ';
}
?>
```

Σχήμα 5-11 Κώδικας σελίδας συγκριτικής προβολής εκτελέσεων.

5.4. Σχεδιασμός διαγραμμάτων

Ο σχεδιασμός των διαγραμμάτων γίνεται με την βοήθεια της βιβλιοθήκης GD της php [**15**]. Η βιβλιοθήκη GD περιέχει συναρτήσεις δημιουργίας, σχεδιασμού και διαχείρισης εικόνων.

Το διάγραμμα κατασκευάζεται σαν μια σελίδα php. Στην αρχή της σελίδας τοποθετείται η συνάρτηση header με το όρισμα "Content-Type: image/png". Η header εισάγει στην κεφαλίδα HTTP το αλφαριθμητικό που δίνεται σαν όρισμα. Στην συγκεκριμένη περίπτωση η πληροφορία που εισάγεται ορίζει ότι το περιεχόμενο της σελίδας που ακολουθεί είναι μία εικόνα τύπου png. Είναι σημαντικό να μην έχει εμφανιστεί τίποτα στο περιεχόμενο της σελίδας πριν γίνει κλήση της header, αλλιώς η συνάρτηση δεν λειτουργεί καθόλου. Η ύπαρξη ακόμα και ενός κενού χαρακτήρα στο περιεχόμενο της σελίδας θα οδηγήσει σε σφάλματα, συνεπώς είναι σημαντικό η σελίδα να ξεκινάει ακριβώς με το tag κώδικα php και η κλήση της header να είναι η πρώτη εντολή που καλείται στην php και να ακολουθήσει η εισαγωγή βιβλιοθηκών.

Η εικόνα αντιπροσωπεύεται στην βιβλιοθήκη GD από ένα αντικείμενο resource. Το αντικείμενο δημιουργείται με την κλήση της συνάρτησης **imagecreate** [**15**]. Η **imagecreate** λαμβάνει σαν ορίσματα τις διαστάσεις της επιθυμητής εικόνας και επιστρέφει ένα resource εικόνας, μέσα στο οποίο θα σχεδιαστεί το διάγραμμα.

Για τον σχεδιασμό της εικόνας χρειάζονται να οριστούν, εκτός από την εικόνα, τα χρώματα που θα χρησιμοποιηθούν. Ένα χρώμα αντιπροσωπεύεται στην βιβλιοθήκη GD και αυτό από ένα αντικείμενο resource που ορίζεται από την συνάρτηση **imagecolorallocate** [**15**]. Η **imagecolorallocate** λαμβάνει σαν ορίσματα ένα resource εικόνας και 3 ακεραίους και επιστρέφει σαν τιμή ένα resource χρώματος χρησιμοποιώντας τους 3 ακεραίους σαν παραμέτρους RGB. Η πρώτη κλήση της ορίζει και το χρώμα του φόντου της εικόνας.

Αφού οριστούν η εικόνα και τα χρώματα γίνεται ο σχεδιασμός του διαγράμματος με την χρήση των παρακάτω συναρτήσεων:

- **imagefilledrectangle** [**15**]. Λαμβάνει σαν ορίσματα ένα resource εικόνας, 4 ακεραίους και ένα resource χρώματος. Σχεδιάζει ένα παραλληλόγραμμο στην εικόνα που της δόθηκε με το χρώμα που της δόθηκε. Ο σχεδιασμός ξεκινάει από το σημείο με συντεταγμένες τους δύο πρώτους ακεραίους που δόθηκαν σαν ορίσματα και ολοκληρώνεται στο σημείο με συντεταγμένες τους δύο επόμενους ακεραίους.
- **imagestring** [**15**]. Λαμβάνει σαν ορίσματα ένα resource εικόνας, 3 ακεραίους, ένα αλφαριθμητικό και ένα resource χρώματος. Σχεδιάζει το κείμενο που δόθηκε στο αλφαριθμητικό όρισμα, στην εικόνα που του δόθηκε με το χρώμα που του δόθηκε χρησιμοποιώντας μία από τις προκαθορισμένες γραμματοσειρές που ορίζεται από το πρώτο ακέραιο. Ο σχεδιασμός ξεκινάει από το σημείο με συντεταγμένες τους δύο επόμενους ακεραίους που δόθηκαν σαν ορίσματα.
- **imagestringup** [**15**]. Λαμβάνει σαν ορίσματα ένα resource εικόνας, 4 ακεραίους, ένα αλφαριθμητικό και ένα resource χρώματος. Σχεδιάζει το κείμενο που δόθηκε στο

αλφαριθμητικό όρισμα, στην εικόνα που του δόθηκε με το χρώμα που του δόθηκε χρησιμοποιώντας μία από τις προκαθορισμένες γραμματοσειρές που ορίζεται από το πρώτο ακέραιο. Ο σχεδιασμός ξεκινάει από το σημείο με συντεταγμένες τους δύο επόμενους ακεραίους που δόθηκαν σαν ορίσματα και γίνεται με κλίση που ορίζεται από τον τέταρτο ακέραιο.

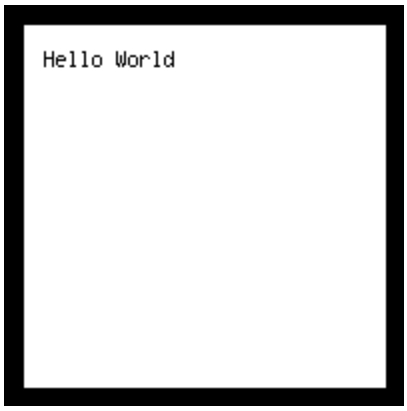
- Και **imageline** [**15**]. Λαμβάνει σαν ορίσματα ένα resource εικόνας, 4 ακεραίους και ένα resource χρώματος. Σχεδιάζει ένα ευθύγραμμο τμήμα στην εικόνα που του δόθηκε με το χρώμα που του δόθηκε. Ο σχεδιασμός ξεκινάει από το σημείο με συντεταγμένες τους δύο πρώτους ακεραίους που δόθηκαν σαν ορίσματα και ολοκληρώνεται στο σημείο με συντεταγμένες τους δύο επόμενους ακεραίους.

Μετά το τέλος τους σχεδιασμού της εικόνας καλούνται δύο ακόμα συναρτήσεις, η **imagepng** και η **imagedestroy,** με όρισμα μόνο το resource εικόνας [**15**]. Η **imagepng** εάν καλεστεί με όρισμα μόνο το resource εικόνας, εκτυπώνει το περιεχόμενο της εικόνας με την μορφή png στην σελίδα (γίνεται να καλεστεί και με διαφορετικά ορίσματα εάν χρειάζεται να αποθηκευτεί η εικόνα σε αρχείο). Η **imagedestroy** καταστρέφει το αντικείμενο resource εικόνας, ελευθερώνοντας την μνήμη που καταλαμβάνει στον server.

Στο Σχήμα 5-12 παρουσιάζεται ένα παράδειγμα σχεδιασμού εικόνας από php. Το παράδειγμα σχεδιάζει μία μαύρη εικόνα διαστάσεων 200 pixel επί 200 pixel. Εντός της εικόνας σχεδιάζει ένα λευκό παραλληλόγραμμο σε απόσταση 10 pixel από τα όρια της εικόνας. Κατόπιν εκτυπώνει το κείμενο "Hello World" με μαύρα γράμματα μέσα στο παραλληλόγραμμο και εμφανίζει την εικόνα. Το αποτέλεσμα φαίνεται στο Σχήμα 5-13.

```
<?php
header("Content-Type: image/png");

$im = @imagecreate(200, 200);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);

imagefilledrectangle($im, 10, 10, 190, 190, $white);
imagestring($im, 2, 20, 20,  "Hello World", $black);

imagepng($im);
imagedestroy($im);
?>
```

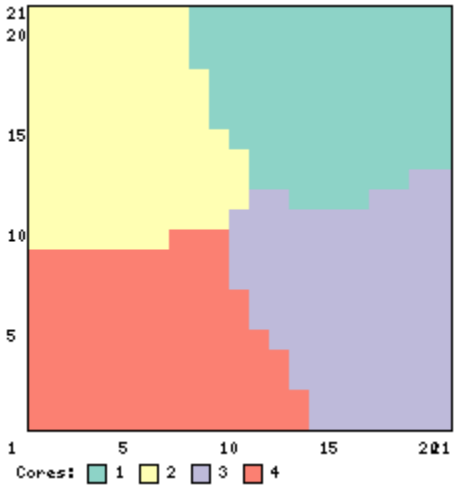
Σχήμα 5-12 Παράδειγμα κώδικα σχεδιασμού εικόνας.



Σχήμα 5-13 Αποτέλεσμα παραδείγματος κώδικα σχεδιασμού εικόνας.

5.4.1. Διάγραμμα κατανομής των κόμβων του πλέγματος

Το διάγραμμα κατανομής των κόμβων του πλέγματος παρουσιάζει γραφικά την κατανομή των κόμβων του πλέγματος ανά επεξεργαστή και βρίσκεται στο αρχείο “nodediagram.php” στον υποφάκελο “img’. Αποτελείται από ένα παραλληλόγραμμο με δύο άξονες. Το κάθε τμήμα του παραλληλόγραμμου χρωματίζεται με βάση τον επεξεργαστή στον οποίο ανατέθηκε ο εκάστοτε κόμβος που αντιστοιχεί στις συντεταγμένες του. Επίσης υπάρχει ένα υπόμνημα που υποδηλώνει ποιος επεξεργαστής αντιστοιχεί σε κάθε χρώμα. Ένα παράδειγμα του διαγράμματος εμφανίζεται στο Σχήμα 5-14.



Σχήμα 5-14 Παράδειγμα διαγράμματος κατανομής των κόμβων του πλέγματος.

Το διάγραμμα λαμβάνει σαν παράμετρο το κλειδί της εγγραφής στην βάση. Η παράμετρος λαμβάνεται από την array $_GET όπως και στις κανονικές σελίδες php. Εφόσον έχει δοθεί τιμή στην παράμετρο, το διάγραμμα διαβάζει από την βάση τις πληροφορίες της εκτέλεσης και ξεκινάει η διαδικασία σχεδιασμού. Στην περίπτωση που δεν έχει δοθεί η παράμετρος εμφανίζεται απλά μια εικόνα που περιέχει ένα μήνυμα σφάλματος.

```php
<?php

header("Content-Type: image/png");
include "../config/config.php";

$id = $_GET["myID"];

if (isset($id)) {
   $id = mysql_real_escape_string($id);
   $result = mysql_query("SELECT * FROM GMRES_Requests WHERE id='$id' ");
   $row = mysql_fetch_array($result);
   $dimx = 2 * $row['dimx'] + 1;
   $dimy = 2 * $row['dimy'] + 1;
   $cores = $row['cores'];

// Σχεδιασμός διαγράμματος.

} else {
   $im = @imagecreate(110, 20)
         or die("Cannot Initialize new GD image stream");
   $background_color = imagecolorallocate($im, 0, 0, 0);
   $text_color = imagecolorallocate($im, 233, 14, 91);
   imagestring($im, 1, 5, 5, "Invalid ID", $text_color);
   imagepng($im);
   imagedestroy($im);
}
?>
```

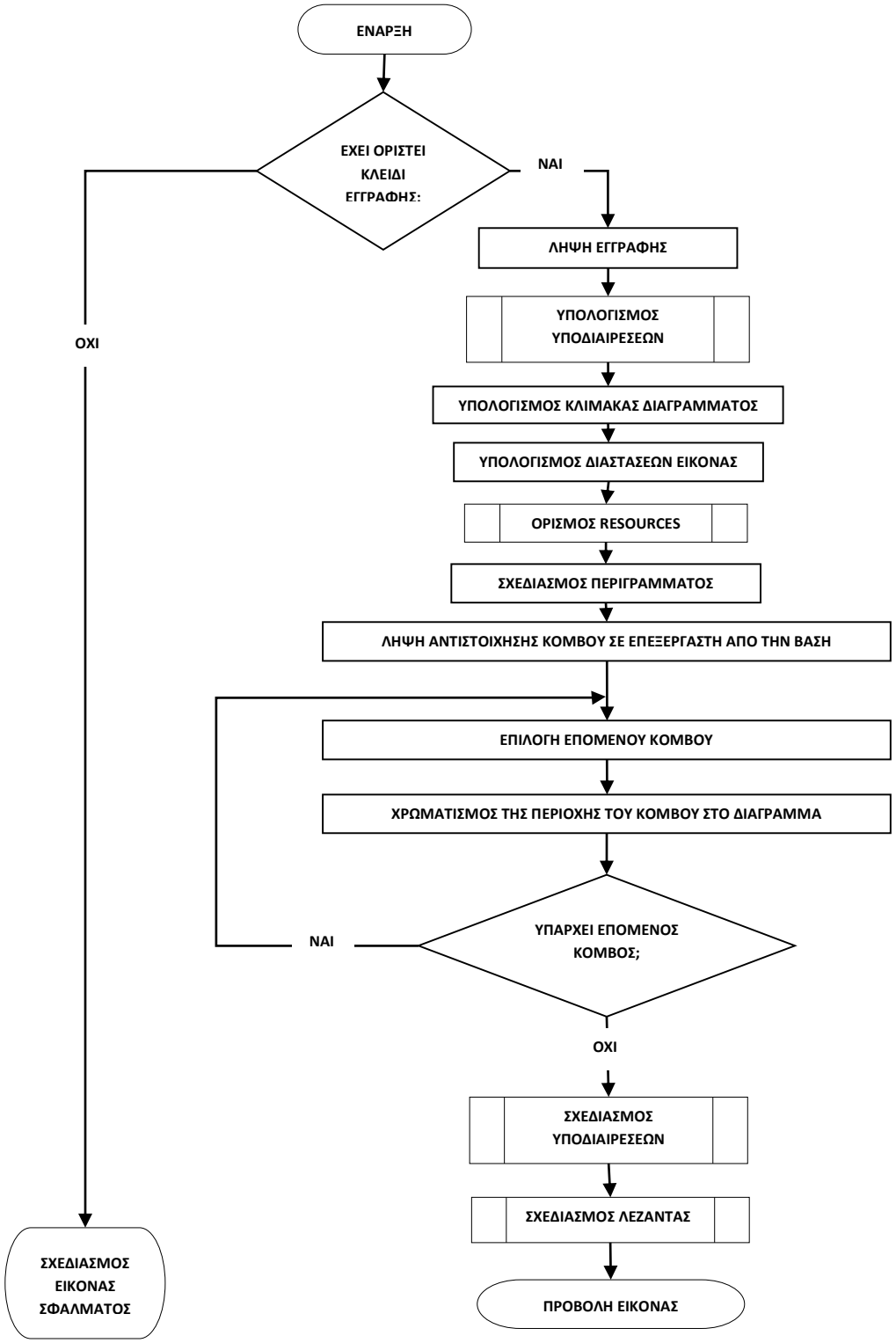
Σχήμα 5-15 Κώδικας εκκίνησης και έλεγχος παραμέτρων σχεδιασμού διαγράμματος κατανομής των κόμβων του πλέγματος.

Αφού ληφθεί η εγγραφή από την βάση με βάση τις διαστάσεις του πλέγματος υπολογίζονται οι προτεινόμενες τιμές για τις υποδιαιρέσεις των αξόνων και την κλίμακα του διαγράμματος ώστε το διάγραμμα να είναι ευδιάκριτο. Κατόπιν, με βάση τις διαστάσεις του πλέγματος και την επιλεγμένη κλίμακα υπολογίζονται οι διαστάσεις της εικόνας, επί τις οποίας θα χαραχτεί το διάγραμμα, και αρχικοποιούνται η εικόνα και τα χρώματα που θα χρησιμοποιηθούν στον σχεδιασμό.

Ο σχεδιασμός του διαγράμματος ξεκινά με την χάραξη ενός μαύρου παραλληλόγραμμου που θα χρησιμοποιηθεί σαν περίγραμμα. Κατόπιν λαμβάνεται ο πίνακας αντιστοιχήσεων κόμβων με επεξεργαστές από την βάση. Για κάθε ζεύγος κόμβου με επεξεργαστή όπου είχε ανατεθεί ο υπολογισμός του υπολογίζεται με βάση τις συντεταγμένες του κόμβου στο πλέγμα η αντίστοιχη τοποθεσία αυτού εντός του περιγράμματος και χρωματίζεται αυτή με το χρώμα που αντιστοιχεί στον επεξεργαστή.

Αφού ολοκληρωθεί η σχεδίαση των αντιστοιχήσεων, χαράσσονται στο διάγραμμα οι υποδιαιρέσεις των αξόνων και ακριβώς κάτω από αυτές σχεδιάζεται μια λεζάντα που υποδεικνύει ποιο χρώμα αντιστοιχεί σε ποιόν επεξεργαστή.

Στο Σχήμα 5-16 εμφανίζεται το διάγραμμα ροής όλης της διαδικασίας σχεδιασμού του διαγράμματος και στο Παράρτημα 10 παρουσιάζεται αναλυτικά η διαδικασία σχεδιασμού και ο κώδικας της σελίδας.



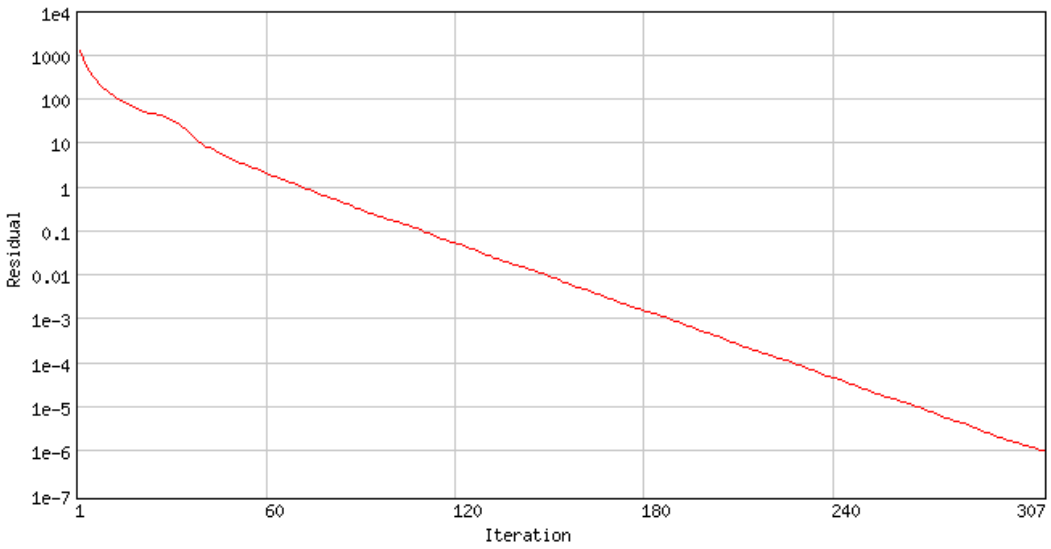
Σχήμα 5-16 Διάγραμμα ροής διαδικασίας σχεδιασμού διαγράμματος κατανομής των κόμβων του πλέγματος.

5.4.2. Διάγραμμα σύγκλισης

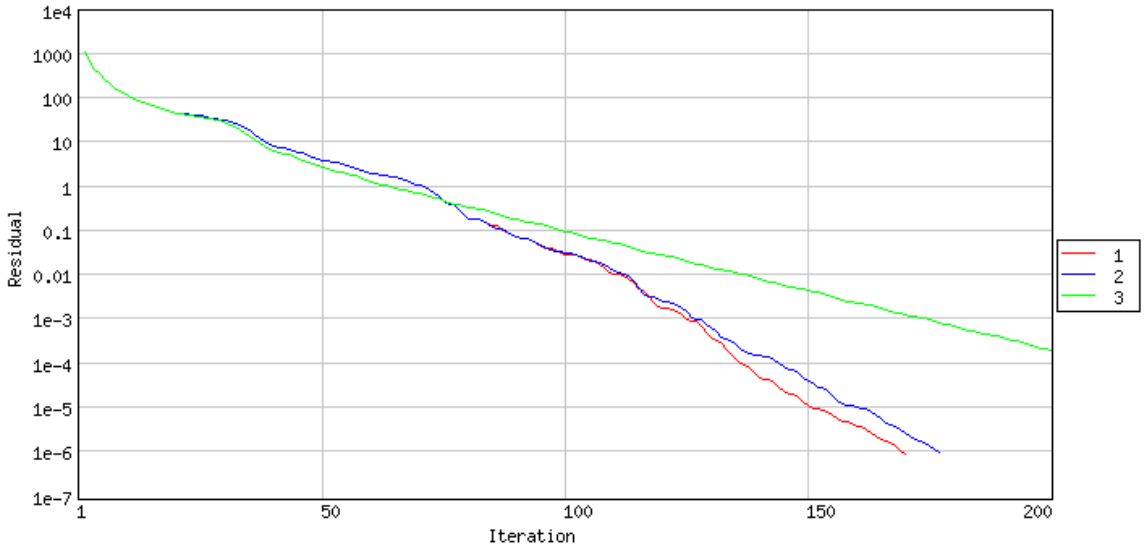
Το διάγραμμα σύγκλισης παρουσιάζει γραφικά την μείωση του σφάλματος ανά επανάληψη της GMRES και αποτελείται από το αρχείο “iterationdiagram.php” στον υποφάκελο “img”. Το διάγραμμα εμφανίζεται με τρείς διαφορετικές μορφές:

- Σαν ένα απλό διάγραμμα για μία εκτέλεση (Σχήμα 5-17).
- Σαν συγκριτικό διάγραμμα που περιέχει πολλαπλές εκτελέσεις και υπόμνημα (Σχήμα 5-18).
- Και τέλος μπορεί να κληθεί κα να επιστρέψει μια απλή μονόχρωμη εικόνα που χρησιμεύει για την αντιστοίχιση των χρωμάτων με τις εξωτερικές πληροφορίες στην περίπτωση που καλεστεί συγκριτικά.

Η μορφή που θα λάβει επιλέγεται από δυο παραμέτρους που λαμβάνει η σελίδα. Η πρώτη ονομάζεται myID και η δεύτερη color. Εφόσον η myID έχει τιμή, εξετάζεται εάν η τιμή είναι ακέραιος ή array ακεραίων και εμφανίζεται το διάγραμμα μίας εκτέλεσης ή το συγκριτικό πολλών εκτελέσεων αντίστοιχα. Ειδάλλως, εφόσον η color έχει τιμή επιστρέφει την απλή μονόχρωμη εικόνα με το χρώμα που αντιστοιχεί στην καμπύλη με αριθμό την τιμή της παραμέτρου. Η διαδικασία λήψης των παραμέτρων καθώς και η επιλογή του τύπου διαγράμματος φαίνεται στο Σχήμα 5-19.



Σχήμα 5-17 Διάγραμμα σύγκλισης



Σχήμα 5-18 Διάγραμμα σύγκλισης με πολλαπλές εκτελέσεις.

```php
<?php
header("Content-Type: image/png");
include "../config/config.php";
$idstr = $_GET["myID"];
$colorid = intval($_GET["color"]);
// Κωδικας functions.
if (isset($idstr)) {

// Σχεδιασμός διαγράμματος.

} elseif (isset($colorid)) {
   $im = @imagecreate(4 , 4) or die("Cannot Initialize new GD image stream");
   $white = imagecolorallocate($im, 255, 255, 255);
   $colorarray = loadcolors($im);
   imagefilledrectangle($im, 0, 0, 4, 4, $colorarray[$colorid]);
   imagepng($im);
   imagedestroy($im);
} else {
   $im = @imagecreate(110, 20)   or die("Cannot Initialize new GD image stream");
   $background_color = imagecolorallocate($im, 0, 0, 0);
   $text_color = imagecolorallocate($im, 233, 14, 91);
   imagestring($im, 1, 5, 5, "Invalid ID", $text_color);
   imagepng($im);
   imagedestroy($im);
}
?>
```

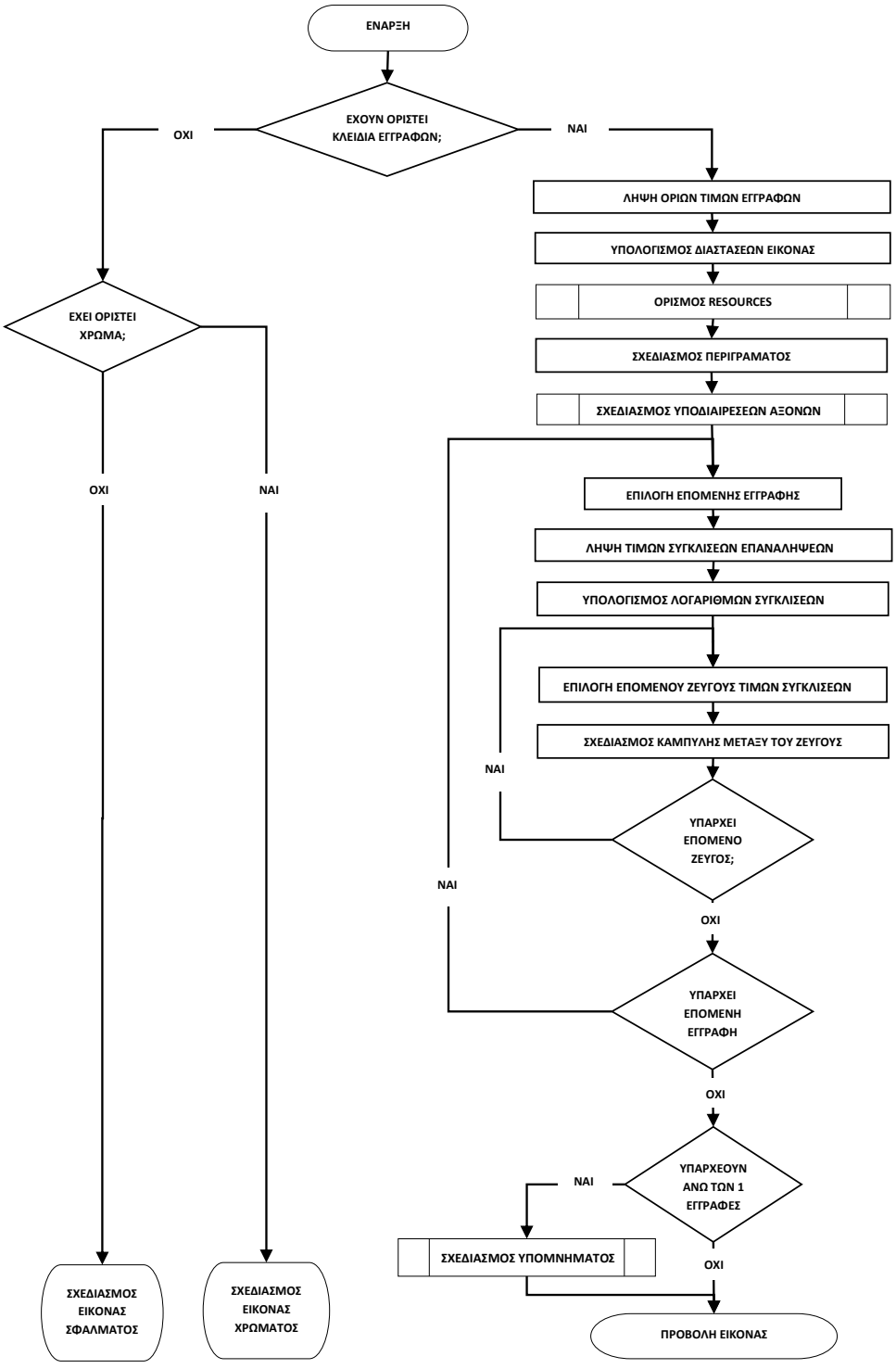
Σχήμα 5-19 Κώδικας εκκίνησης και έλεγχος παραμέτρων σχεδιασμού διαγράμματος σύγκλισης.

Ο σχεδιασμός του διαγράμματος ξεκινάει λαμβάνοντας από την βάση τις οριακές τιμές του από το σύνολο των επαναλήψεων όλων των εκτελέσεων που πρόκειται να χαραχθούν. Οι οριακές τιμές του άξονα των επαναλήψεων είναι ο αριθμός 1 και ο μέγιστος αριθμός επανάληψης που υπάρχει στο σύνολο αυτό, ενώ του άξονα της συγκλίσης είναι η μεγίστη και η ελάχιστη τιμή των συγκλίσεων από αυτό το σύνολο. Με βάση τις οριακές τιμές αυτές υπολογίζονται οι διαστάσεις του διαγράμματος και επιλέγονται οι υποδιαιρέσεις των αξόνων καθώς και οι κλίμακες των αξόνων, ώστε να είναι ευδιάκριτο το διάγραμμα. Αφού έχουν γίνει οι υπολογισμοί των διαστάσεων του διαγράμματος, υπολογίζονται οι διαστάσεις της εικόνας επί της οποίας θα χαραχτεί το διάγραμμα και αρχικοποιούνται η εικόνα και τα χρώματα που θα χρησιμοποιηθούν στον σχεδιασμό.

Ο σχεδιασμός ξεκινάει με την χάραξη των δύο αξόνων και του περιγράμματος. Κατόπιν, εντός αυτού σχεδιάζεται το πλέγμα των υποδιαιρέσεων των αξόνων, όπως αυτές έχουν επιλεχτεί. Λαμβάνονται από την βάση για κάθε εγγραφή που πρόκειται να σχεδιαστεί το σύνολο των ζευγών επανάληψης – σύγκλισης και υπολογίζονται οι λογάριθμοι των συγκλίσεων. Ξεκινώντας από την πρώτη επανάληψη λαμβάνονται τα γειτονικά ζεύγη τιμών σύγκλισης και για καθένα από αυτά χαράσσεται μια ευθεία που ενώνει τις συντεταγμένες που αντιστοιχούν στο καθένα, χρησιμοποιώντας το χρώμα που αντιστοιχεί στην επιλεγμένη εγγραφή.

Τέλος, εφόσον γίνεται προβολή άνω των μία εγγραφών, σχεδιάζεται και ένα υπόμνημα αντιστοίχησης των χρωμάτων με τις εγγραφές.

Συνολικά το διάγραμμα ροής της σελίδας του διαγράμματος φαίνεται στο Σχήμα 5-20 και στο Παράρτημα 11 παρουσιάζεται αναλυτικά η διαδικασία σχεδιασμού και ο κώδικας της σελίδας.



Σχήμα 5-20 Διάγραμμα ροής διαδικασίας σχεδιασμού διαγράμματος σύγκλισης.

Βιβλιογραφικές αναφορές

[1] Apache HTTP Server Documentation. [Online]. Apache HTTP Server Documentation

[2] BLAS (Basic Linear Algebra Subprograms). [Online]. http://www.netlib.org/blas

[3] T.M. R. Ellis, Ivor R. Phillips, and M. Thomas Lahey, *Fortran 90 Programming*.: Addison Wesley, 1994.

[4] FeBUI User Manual. [Online]. http://febui.chemeng.ntua.gr/febui.pdf

[5] The Open Group. (1997) The Single UNIX Specification, Version 2. [Online]. http://pubs.opengroup.org/onlinepubs/007908799/xbd/re.html

[6] Javascript Documentation. [Online]. https://developer.mozilla.org/en-US/docs/Web/JavaScript

[7] jQuery API Documentation. [Online]. http://api.jquery.com/

[8] LAM/MPI Parallel Computing. [Online]. http://www.lam-mpi.org

[9] LAPACK—Linear Algebra PACKage. [Online]. http://www.netlib.org/lapack

[10] Yandong Mao, Frans Kaashoek, and Robert Morris. METIS. [Online]. http://pdos.csail.mit.edu/metis/

[11] MPICH | High-Performance Portable MPI. [Online]. www.mpich.org

[12] Myricom. [Online]. http://www.myricom.com/

[13] MySQL 5.7 Reference Manual. [Online]. http://dev.mysql.com/doc/refman/5.7/en/index.html

[14] Pegasus Cluster. [Online]. http://febui.chemeng.ntua.gr/pegasus.htm

[15] PHP Manual. [Online]. http://php.net/manual/en/index.php

[16] Rocks. [Online]. http://www.rocksclusters.org/wordpress/

[17] Antony N. Spyropoulos, Athanasios G. Papathanasiou, John A. Palyvos, and Andreas G. Boudouvis, "FE-BUI - Finite Element Beowulf User Interface: A Homemade Package For Automated Parallel Finite Element computations," in *5th GRACM International Congress on computational Mechanics*, Limassol, 2005.

[18] The GNU Fortran Compiler Documentation. [Online]. http://gcc.gnu.org/onlinedocs/gfortran/

[19] The GNU Project. [Online]. http://www.gnu.org/software/bash/manual/bash.htm

[20] The Linux Documentation Project. [Online]. http://www.tldp.org/

[21] The Linux Information Project. [Online]. http://www.linfo.org

[22] The Message Passing Interface (MPI) standard. [Online]. http://www.mcs.anl.gov/research/projects/mpi/

[23] Αντώνιος Ν. Σπυρόπουλος, *υπολογισμοί μεγάλης κλίμακας με μεθόδους παράλληλης επεξεργασίας σε γραμμικά προβλήματα διεπιφανειακής μαγνητο-ρευστομηχανικής*. Αθήνα: Διδακτορική Διατριβή. Εθνικό Μετσόβιο Πολυτεχνείο., 2003.

Παράρτημα 1. Κώδικας MySQL κατασκευής βάσης

```
CREATE TABLE GMRES_Requests
(
  id int(11) NOT NULL default '0',
  complete tinyint(1) NOT NULL default '0',
  create_date datetime default NULL,
  IP varchar(20) NOT NULL default '',
  cores int(11) NOT NULL default '0',
  dimx int(11) NOT NULL default '0',
  dimy int(11) NOT NULL default '0',
  outerit int(11) NOT NULL default '0',
  innerit int(11) NOT NULL default '0',
  rdeflation int(11) NOT NULL default '0',
  tolerance float NOT NULL default '0',
  custom_nodes int(10) unsigned NOT NULL default '0',
  exec_time float NOT NULL default '0',
  exec_cores int(11) NOT NULL default '0',
  exec_memory float NOT NULL default '0',
  exec_np float NOT NULL default '0',
  errors blob NOT NULL,
PRIMARY KEY (id)
);


CREATE TABLE GMRES_Request_Iterations
(
  request_id int(11) NOT NULL default '0',
  outer_iter int(11) NOT NULL default '0',
  inner_iter int(11) NOT NULL default '0',
  diff float NOT NULL default '0',
  PRIMARY KEY  (request_id,outer_iter,inner_iter)
FOREIGN KEY (request_id) REFERENCES GMRES_Requests(id) ON UPDATE
CASCADE ON DELETE CASCADE
);

CREATE TABLE GMRES_Requests_Nodes (
  request_id int(11) NOT NULL default '0',
  dimx int(11) NOT NULL default '0',
  dimy int(11) NOT NULL default '0',
  core int(11) NOT NULL default '0',
  PRIMARY KEY  (request_id,dimx,dimy),
  FOREIGN KEY (request_id) REFERENCES GMRES_Requests(id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

Παράρτημα 2. Ανάλυση επικοινωνίας εκτελέσιμου fortran.

α. Ανάγνωση και εγγραφή αρχείων κειμένου από fortran.

Η ανάγνωση και η εγγραφή σε ένα αρχείο κειμένου γίνεται με τις εντολές read και write αντίστοιχα [**3**]. Η εντολές αυτές λαμβάνουν δυο υποχρεωτικά ορίσματα. Το πρώτο είναι ο αριθμός που υποδηλώνει την λογική μονάδα που θα χρησιμοποιηθεί και το δεύτερο είναι μια εντολή μορφοποίησης που υποδηλώνει τον τρόπο εγγραφής ή ανάγνωσης. Σημαντικές τιμές λογικών μονάδων αποτελούν οι τιμές:

- *** ή 5** στην read, που αντιστοιχούν στην standard input (stdin), που υποδηλώνει την προκαθορισμένη μονάδα εισόδου
- *** ή 6** στην write, που αντιστοιχούν στην standard output (stdout), που υποδηλώνει την προκαθορισμένη μονάδα εξόδου (στην fortran)
- και **0** στην write, που αντιστοιχεί στην standard error (stderr) που υποδηλώνει την προκαθορισμένη μονάδα εξόδου σφαλμάτων.

Στην θέση της εντολής μορφοποίησης μπορεί να χρησιμοποιηθεί ο χαρακτήρας * που υποδηλώνει την χρήση της προκαθορισμένης μορφοποίησης ανάλογα με το όρισμα στο οποίο βρίσκεται. Πέραν των υποχρεωτικών ορισμάτων υπάρχουν αρκετά προαιρετικά, το κυριότερο από τα οποία είναι το IOSTAT που επιστρέφει την τιμή του σφάλματος εάν ένα παρουσιαστεί και μπορεί κατά την ανάγνωση αρχείου να χρησιμοποιηθεί για την αναγνώριση από την εφαρμογή του τέλους του αρχείου. Η κλήση της εντολής ακολουθείται από μια σειρά μεταβλητών που περιέχουν τις προς εγγραφή τιμές ή θα λάβουν τις προς ανάγνωση τιμές ανάλογα με την εντολή που καλέστηκε.

Τέλος, η εντολή close καλείται για να τερματίσει την επικοινωνία με μία λογική μονάδα και συνήθως καλείται με μόνο όρισμα τον αριθμό που αντιστοιχεί στην λογική μονάδα προς τερματισμό. [**3**]

β. Παράμετροι γραμμής εντολών για την διαχείριση των προκαθορισμένων λογικών μονάδων

Η χρήση μιας σειράς συμβόλων στην γραμμή εντολών επιτρέπει την λήψη και επεξεργασία των σφαλμάτων από την μία εφαρμογή σε μία άλλη και δίνει την δυνατότητα επεξεργασίας και φιλτραρίσματος διαφόρων πληροφοριών που θα προβάλλονται στον χρήστη, εάν η εφαρμογή εκτελεστεί από γραμμή εντολών. Ειδικότερα:

- Η χρήση του **<** αλλάζει τον ορισμό της προκαθορισμένης μονάδας εισόδου σε ένα αρχείο, με αποτέλεσμα την χρήση του αρχείου σαν να ήταν το πληκτρολόγιο στην εφαρμογή.
- Η χρήση των συμβόλων **>** και ***2>*** αλλάζει τον ορισμό των προκαθορισμένων μονάδων εξόδου και εξόδου σφαλμάτων σε αρχεία.
- Η χρήση του χαρακτήρα ***&*** μετά από τους χαρακτήρες < ή > υποδηλώνει ότι το αποτέλεσμα πρέπει να επισυναφθεί στο στόχο αντί να τον αντικαταστήσει, π.χ. η επισύναψη του "*2>& error.log*" μετά από μια εντολή θα δημιουργήσει ένα αρχείο

"error.log" στο οποίο θα επισυνάπτονται τα σφάλματα εκτέλεσης του προγράμματος μετά από την κάθε εκτέλεση [**20**].

Αξιοσημείωτο είναι επίσης ότι είναι δυνατή η χρήση ενός αριθμού λογικής μονάδας αντί για αρχείο στόχο, π.χ. η επισύναψη του *"2>&1"* μετά από μια εντολή θα επισυνάψει την standard error στην standard output. Η χρήση αυτής της παραμέτρου είναι ιδιαίτερα σημαντική καθότι πολλές εντολές/εφαρμογές έχουν πρόσβαση μόνο στη standard output και σε αυτές τις περιπτώσεις η μη χρήση της παραμέτρου οδηγεί σε απώλεια χρήσιμων πληροφοριών.

γ. Κώδικας εισαγωγής τιμών παραμέτρων γραμμής εντολών.

```
ARGUMENT_COUNT=IARGC()

IF (MOD(ARGUMENT_COUNT,2)==1) THEN
 call badinput()
 STOP
END IF
DO ARGUMENT_LOOP_I = 1 , ARGUMENT_COUNT , 2
    CALL GETARG(ARGUMENT_LOOP_I,BUFFER)
    SELECT CASE (BUFFER)
        CASE ('-x')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) nex
        CASE ('-y')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) ney
        CASE ('-i')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) ARG_GMRESKrylov
        CASE ('-o')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) ARG_GMRESIter
        CASE ('-r')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) ARG_r_Deflation
        CASE ('-t')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ(BUFFER2,*) ARG_tolgmres
        CASE ('-n')
           CALL GETARG(ARGUMENT_LOOP_I+1,BUFFER2)
           READ (BUFFER2,*) ARG_NodesFile
        CASE DEFAULT
          call badinput()
          STOP
    END SELECT
END DO
```

δ. Κώδικας εισαγωγής τιμών αντιστοίχησης κόμβων σε επεξεργαστή.

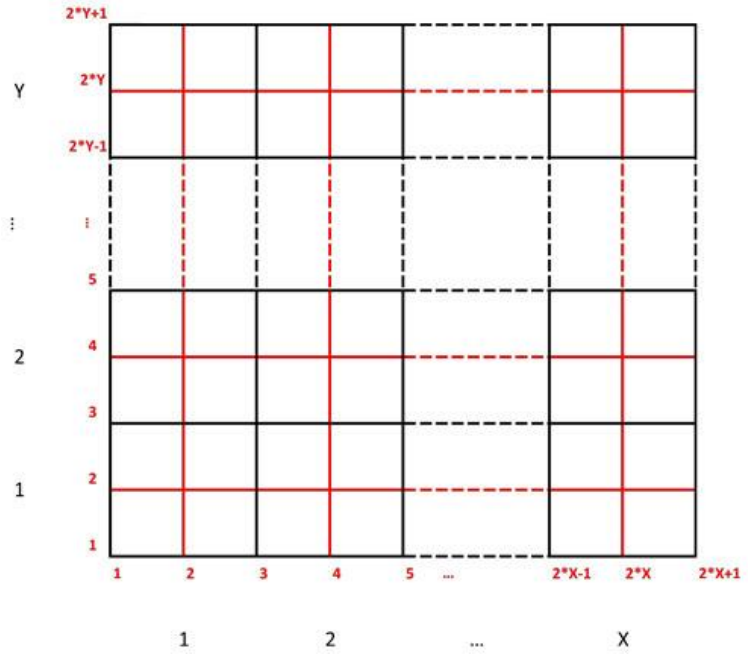
```
      c load partitioning

        IF (ARG_NodesFile=='') THEN
            use_metis=.TRUE.
            mypart=0
        ELSE
            use_metis=.FALSE.
            OPEN(unit=2,FILE=ARG_NodesFile)
              DO
                  READ(2,*,IOSTAT=err) i,l
                  if (err==-1) EXIT
                  if (i>np) EXIT
                  mypart(i)=l
              END DO
            CLOSE(2)
        END IF
```

Παράρτημα 3. Ανάλυση μετατροπής των κόμβων του πλέγματος σε πίνακα

Επειδή το FEBUI καλείται με σταθερά την χρήση 9 κόμβων ανά στοιχείο (το κέντρο, οι 4 γωνίες και το μέσο καθεμίας από τις 4 πλευρές), δημιουργείται ένας πίνακας κόμβων με διαστάσεις:

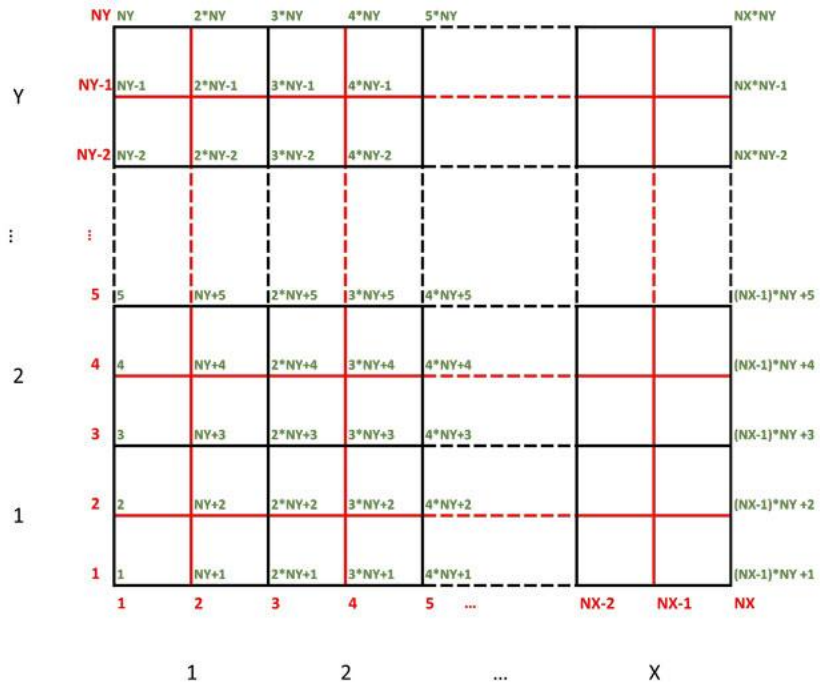
$$NX = 2 * X + 1$$
$$NY = 2 * Y + 1$$

Όπου X,Y οι διαστάσεις του προς επίλυση πλέγματος και NX,NY οι διαστάσεις του νέου πίνακα, όπως φαίνεται στο Σχήμα Π 3-1.



Σχήμα Π 3-1 Ορισμός κόμβων σε σύστημα X*Y

Η βιβλιοθήκη FeBUI λαμβάνει το σύνολο των κόμβων του πλέγματος σαν ένα μονοδιάστατο πίνακα του οποίου η αρίθμηση αρχίζει από το κόμβο (1,1) και ξεκινά από κάτω προς το πάνω και κατόπιν αριστερά προς τα δεξιά όπως φαίνεται στο Σχήμα Π 3-2.



Σχήμα Π 3-2 Αρίθμηση nodes σαν όρισμα της FeBUI.

Για την μετατροπή των συντεταγμένων ενός κόμβου στην θέση που του αντιστοιχεί στον πίνακα-όρισμα χρησιμοποιούνται οι παρακάτω εξισώσεις.

$$i = (x-1) * N_Y + y$$

$$x = \lceil i \,/\, N_Y \rceil$$

$$y = i \; mod \; N_Y$$

Όπου :

- *i* η θέση στον πίνακα όρισμα,
- *x,y* οι συντεταγμένες και
- N_Y το ύψος του πίνακα των κόμβων του πλέγματος.

Εντός της εφαρμογής χρησιμοποιούνται οι παραπάνω συναρτήσεις για την μετατροπή αυτού του μονοδιάστατου πίνακα σε συντεταγμένες στο πλέγμα όπως φαίνεται στο Σχήμα Π 3-3.

```
c export used nodes
      OPEN(unit=3,FILE='')
             WRITE(3,*) " DIMX | DIMY | CORE "
           DO i=1,np
               WRITE(3,'(I6,"|",I6,"|",I6)') ceiling(1.*i/nny),
     $          i-(ceiling(1.*i/nny)-1)*nny,mypart(i)
           END DO
      CLOSE(3)
```

Σχήμα Π 3-3 Κώδικας εξαγωγής τιμών αντιστοίχησης κόμβων σε επεξεργαστή.

Παράρτημα 4. Script αυτοματοποίησης εκτέλεσης υπηρεσίας

```
#!/bin/bash
PID=""
function get_pid {
   PID=`ps ax |grep "/bin/sh" |grep WebFEBUIChecker.sh |cut -d " " -f 2`
}
function stop {
   get_pid
   if [ -z $PID ]; then
      echo "WebFEBUI Checker is not running."
      exit 1
   else
      echo -n "Stopping WebFEBUI Checker.."
      kill $PID
      sleep 1
      echo ".. Done."
   fi
}
function start {
   get_pid
   if [ -z $PID ]; then
      echo  "Starting WebFEBUI Checker.."
      nohup /home/gpetr/WEB/WebFEBUIChecker.sh 2>&1 &
      get_pid
      echo "Done. PID=$PID"
   else
      echo "WebFEBUI Checker is already running, PID=$PID"
   fi
}
function restart {
   echo  "Restarting WebFEBUI Checker.."
   get_pid
   if [ -z $PID ]; then
      start
   else
      stop
      start
   fi
}
function status {
   get_pid
   if [ -z  $PID ]; then
      echo "WebFEBUI Checker is not running."
      exit 1
   else
      echo "WebFEBUI Checker is running, PID=$PID"
   fi
}
case "$1" in
   start)
      start
      ;;
   stop)
      stop
      ;;
   restart)
      restart
      ;;
   status)
      status
      ;;
   *)
      echo "Usage: $0 {start|stop|restart|status}"
esac
```

Παράρτημα 5. Ανάγνωση και διαχείριση των δεδομένων από την php.

α. Βοηθητικές συναρτήσεις στην php.

Τα δεδομένα λαμβάνονται από τα αρχεία σαν μια αλφαριθμητική μεταβλητή και είναι απαραίτητη η ανάγνωση από αυτά των απαιτητών πληροφοριών και η αποθήκευσή τους στη βάση σε μία μορφή που είναι πιο εύκολο να διαχειριστεί.

Εφόσον τα αρχεία περιέχουν πληροφορίες ανά γραμμή και στην σελίδα έχουν εισαχθεί σαν μία συνεχής μεταβλητή, είναι απαραίτητο να διαχωριστούν οι γραμμές κάθε αρχείου. Αυτό επιτυγχάνεται με την χρήση της συνάρτησης *explode* [**15**]. Η συνάρτηση explode λαμβάνει 2 ορίσματα, που είναι και τα δύο αλφαριθμητικοί χαρακτήρες, διαχωρίζει τον δεύτερο σε κομμάτια χρησιμοποιώντας σαν διαχωριστικό τον πρώτο και επιστρέφει τα κομμάτια σαν array αλφαριθμητικών. Η συνάρτηση αυτή μπορεί συνεπώς εύκολα να τεμαχίσει έναν μεγάλο αλφαριθμητικό σε μικρότερους, και συνεπώς ευκολότερους να διαχειριστούν, αλφαριθμητικούς. Σαν πρώτο όρισμα, δηλαδή διαχωριστικός χαρακτήρας, μπορεί να χρησιμοποιηθεί είτε ένας συγκεκριμένος χαρακτήρας είτε ένας από τους ειδικούς χαρακτήρες που υποδηλώνουν την αλλαγή γραμμής. Στα αρχεία, όπως εξάγονται από την έκδοση της fortran που χρησιμοποιήθηκε, η αλλαγή γραμμής ορίζεται με τον χαρακτήρα “\r” μόνο, συνεπώς η χρήση αυτού σαν πρώτο όρισμα στην explode θα δημιουργήσει ένα array από γραμμές από το εισαχθέν αρχείο. Στην μετατροπή των περιεχομένων των αρχείων σε array βοηθάει και η εντολή *array_shift,* η οποία απλά αφαιρεί την πρώτη εγγραφή ενός array από το array και συνεπώς στην συγκεκριμένη περίπτωση αφαιρεί την πρώτη γραμμή κειμένου που περιέχει την κεφαλίδα σε περιπτώσεις πινάκων.

Επίσης χρησιμοποιούνται διάφορες συναρτήσεις διαχείρισης αλφαριθμητικών όπως:

- Η *trim,* που αφαιρεί όλα τα κενά από την αρχή και το τέλος ενός αλφαριθμητικού.
- Η *substr,* που λαμβάνει 3 ορίσματα ένα αλφαριθμητικό και δύο ακεραίους και επιστρέφει ένα αλφαριθμητικό που ξεκινάει από τον N-στο χαρακτήρα του δοθέντος αλφαριθμητικού και έχει μήκος M όπου N και M τα δύο αριθμητικά ορίσματα αντίστοιχα.
- Η *mysql_real_escape_string,* που μετατρέπει ένα αλφαριθμητικό σε μία ασφαλή μορφή για να χρησιμοποιηθεί σε SQL query.
- Και η *is_numeric,* που εξετάζει εάν τα περιεχόμενα ενός αλφαριθμητικού είναι ένας αριθμός.

β. Κώδικας συνάρτησης parseoutputfile

```
function parseoutputfile($filename_output,$id){
   $txt_file_output = file_get_contents($filename_output);
      $txtrows_output = explode("\n", $txt_file_output);
      $exec_time = 0.0;
      $exec_np = 0;
      $exec_cores = 0;
      $exec_mem = 0.0;
      foreach ($txtrows_output as $txtrow2 => $data2) {
         $values = explode("=", $data2);
         $leftside = trim($values[0]);
         switch ($leftside) {
            case "np" :
               $exec_np = mysql_real_escape_string(trim($values[1]));
               break;
            case "CPUs" :
               $exec_cores = mysql_real_escape_string(trim($values[1]));
               break;
            case "max required memory in MBs" :
               $exec_mem = mysql_real_escape_string(trim($values[1]));
               break;
            case "Time from main code" :
               $exec_time = mysql_real_escape_string(trim($values[1]));
               break;
         }
      }
      unlink($filename_output);
      return array(        "time" =>  $exec_time, "np" =>  $exec_np,
                           "cores" => $exec_cores,"mem" =>  $exec_mem ); }
```

γ. Κώδικας συνάρτησης parsenodefile

```
function parsenodefile($filename_nodes,$id){
      $txt_file_nodes = file_get_contents($filename_nodes);
      $txtrows_nodes = explode("\n", $txt_file_nodes);
      array_shift($txtrows_nodes);
      foreach ($txtrows_nodes as $txtrow => $data) {
         $values = explode("|", $data);
         if (is_numeric(trim($values[0])) && is_numeric(trim($values[1])) && is_numeric(trim($values[2]))) {
            $insertsqlstr = " INSERT INTO GMRES_Requests_Nodes (request_id,dimx,dimy,core) VALUES('$id', '" .
trim($values[0]) . "', '" . trim($values[1]) . "', '" . trim($values[2]) . "') ; ";
            mysql_query($insertsqlstr) or die(mysql_error());
         }
      }
      unlink($filename_nodes);
}
```

δ. Κώδικας συνάρτησης parseconvfile

```
function parseconvfile($filename_conv,$id){
      $txt_file_conv = file_get_contents($filename_conv);
      $txtrows_conv = explode("\n", $txt_file_conv);
      array_shift($txtrows_conv);
      foreach ($txtrows_conv as $txtrow => $data) {
         $outer_iter = trim(substr($data, 0, 12));
         $inner_iter = trim(substr($data, 12, 12));
         $residual = trim(substr($data, 24));
         if (is_numeric($outer_iter) && is_numeric($inner_iter) && is_numeric($residual)) {
            $insertsqlstr = " INSERT INTO GMRES_Request_Iterations (request_id,outer_iter,inner_iter,diff)
VALUES('$id', '$outer_iter', '$inner_iter', '$residual') ; ";
            mysql_query($insertsqlstr) or die(mysql_error());}
      }
      unlink($filename_conv);
}
```

στ. Κώδικας συνάρτησης parseresult

Ο εντοπισμός του εάν έχει περάσει ο μέγιστος επιτρεπτός χρόνος γίνεται απευθείας μέσω της βάσης δεδομένων με την χρήση των συναρτήσεων NOW, TIME_TO_SEC και TIMEDIFF της mysql [**13**].

Η NOW επιστρέφει την τωρινή χρονική στιγμή εκφρασμένη σαν πεδίο ημερομηνίας και ώρας (datetime). Η TIMEDIFF επιστρέφει την διαφορά 2 πεδίων ημερομηνίας και ώρας σαν πεδίο ημερομηνίας και ώρας, ενώ η TIME_TO_SEC μετατρέπει την διάρκεια ενός πεδίου ημερομηνίας και ώρας σε ακέραιο αριθμό δευτερολέπτων. Ο συνδυασμός των τριών αυτών συναρτήσεων επιστρέφει την διαφορά ενός πεδίου ημερομηνίας και ώρας από την τωρινή χρονική στιγμή σε δευτερόλεπτα.

Στην συγκεκριμένη περίπτωση ελέγχεται εάν υπάρχει εγγραφή με το δοθέν κλειδί όπου η διαφορά της ημερομηνίας και ώρας εκτέλεσης της εγγραφής με την τωρινή στιγμή είναι μεγαλύτερη του μεγίστου επιτρεπτού χρόνου συν δέκα δευτερόλεπτα και αυτομάτως αποθηκεύεται η εγγραφή στην βάση σαν αποτυχημένη. Τα 10 δευτερόλεπτα προστίθενται στον μέγιστο χρόνο όπως αυτός έχει δοθεί στην εντολή εκτέλεσης της εφαρμογής, έτσι ώστε να δοθεί χρόνος στην εντολή να ακυρώσει την εκτέλεση η ίδια, δίνοντας πίσω μερικώς αποτελέσματα ή και πιθανές πληροφορίες σφαλμάτων και αιτιολόγησης της καθυστέρησης.

Εφόσον όλη η διαδικασία γίνεται εντός της βάσης, ο εντοπισμός της από την php γίνεται μέσω της συνάρτησης mysql_affected_rows [**15**], η οποία επιστρέφει το πλήθος των εγγραφών που άλλαξαν με την εκτέλεση της τελευταίας εντολής sql που κλήθηκε.

```
function parseresult($id,$Settings) {
   $insertsqlstr = '';
   $filename_conv = $Settings['Output_path'].'pgmres_conv';
   $filename_nodes = $Settings['Output_path'].'used_nodes';
   $filename_output = $Settings['Output_path']."$id.output";
   $filename_errors = $Settings['Output_path']."$id.errors";
   if (file_exists($filename_conv) && file_exists($filename_output) && file_exists($filename_nodes)) {

      $exec=parseoutputfile($filename_output,$id,$Settings);

       parseconvfile($filename_conv,$id);

       parsenodefile($filename_nodes,$id);

            $txt_file_errors=mysql_real_escape_string(str_replace("\n","<BR />",
            file_get_contents($filename_errors)));
             $insertsqlstr = " UPDATE  GMRES_Requests SET complete=1 ,exec_time='".$exec["time"]."' ,
exec_cores='".$exec["cores"]."' , exec_memory= '".$exec["mem"]."', exec_np='".$exec["np"]."',
errors='$txt_file_errors'  WHERE id ='$id';";
       mysql_query($insertsqlstr) or die(mysql_error());

       unlink($filename_errors);
       return 1;
   } else {
       if (file_exists($filename_errors)) {
            echo "Request $id exited with errors: <BR />";
            $error =  mysql_real_escape_string(str_replace("\n", "<BR/>",file_get_contents($filename_errors)));
             if (strlen(trim($error))>0){echo  $error; }
             echo "<BR /> canceling request : $id. <BR />";
             $insertsqlstr = " UPDATE  GMRES_Requests SET complete=2, errors='$error'  WHERE id='$id';";
              mysql_query($insertsqlstr) or die(mysql_error());
              unlink($filename_errors);
              return 2;
            }

            mysql_query("UPDATE  GMRES_Requests SET complete=2 , errors='Request timed out.' WHERE
            complete=0      and      id='$id'      and      TIME_TO_SEC(TIMEDIFF(now(),create_date))      >
            '".($Settings['timeout']+10)."' ;");
             $rows = mysql_affected_rows();
                  if ($rows > 0) { echo "Request $id timed out. <BR />";
                                        return 2; }
                  return 0;
               }
            }
```

Παράρτημα 6. Συμπίεση σχεδιασμένου μοντέλου κατανομής κόμβων σε αλφαριθμητικό

Μετά τον σχεδιασμό του μοντέλου, οι μεταβλητές είναι ακόμα ακόμα σε μία μη εύκολα διαχειρίσιμη μορφή για την μεταφορά τους από σελίδα σε σελίδα. Συνεπώς κατασκευάζεται μία σειρά από συναρτήσεις με σκοπό την μετατροπή των σχεδιασμένων αντικειμένων σε ένα μικρό και εύκολα μεταφερόμενο αλφαριθμητικό.

Κατ' αρχήν ορίζονται δύο συναρτήσεις για την συμπίεση των αριθμών σε αλφαριθμητικά και αντιστρόφως. Ορίζεται ένα αλφαριθμητικό κλειδί με βάση το οποίο ο αριθμός μετατρέπεται σε αριθμό βάσης ίσης με το μήκος του κλειδιού και αντιστρόφως. Στα Σχήμα Π 6-1, Σχήμα Π 6-2 και Σχήμα Π 6-3 φαίνεται ο κώδικας υλοποίησης της μετατροπής σε php και javascript [**6**].

```
function numbertochar($inval, $digits = 1) {
   $chars = '0123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';
   $retstr = '.';
   $charslen = strlen($chars);
   if ($inval < pow($charslen, $digits)) {
      $retstr = '';
      for ($i = $digits; $i > 0; $i-=1) {
         $currentpow = pow($charslen, $i - 1);
         $retstr .= $chars[floor($inval / $currentpow)];
         $inval = $inval - floor($inval / $currentpow) * $currentpow;
      }
   }
   return $retstr;
}
```

Σχήμα Π 6-1 Κώδικάς συμπίεσης αριθμητικού σε αλφαριθμητικό για php.

```
function chartonum($instr) {
   $chars = '0123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';
   $retval = 0;
   $charslen = strlen($chars);
   $digits = strlen($instr);
   for ($i = $digits; $i > 0; $i-=1) {
      $currentpow = pow($charslen, $i - 1);
      $retval += strpos($chars, $instr[$digits - $i]) * $currentpow;
   }
   return $retval;
}
```

Σχήμα Π 6-2 Κώδικας μετατροπής αλφαριθμητικού σε αριθμό για php.

```
<script language="javascript" type="text/javascript" >
   function numbertochar(inval, digits) {
      chars =
'0123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';
      retstr1 = '.';
      charslen = chars.length;
      if (inval < Math.pow(charslen, digits)) {
         retstr1 = '';
         for (j = digits; j > 0; j-=1) {
            currentpow =  Math.pow(charslen, j - 1);
            retstr1 = retstr1.concat(chars.charAt( Math.floor(inval / currentpow)));
            inval = inval -  Math.floor(inval / currentpow) * currentpow;
         }
      }
      return retstr1;
   }
 </script>
```

Σχήμα Π 6-3 Κώδικάς συμπίεσης αριθμητικού σε αλφαριθμητικό για javascript

Αφού οριστεί η διαδικασία μετατροπής σε αλφαριθμητικά, τα αντικείμενα box μετατρέπονται σε ένα αλφαριθμητικό με την εξής διαδικασία: κάθε μέλος του box μετατρέπεται σε αλφαριθμητικό με συγκεκριμένο μήκος. Τα z και core έχουν μήκος 1 χαρακτήρα ενώ τα υπόλοιπα μήκος 2 χαρακτήρων. Τα μέλη επισυνάπτονται σε έναν αλφαριθμητικό με συγκεκριμένη σειρά και έτσι μετατρέπεται το κάθε box σε έναν αλφαριθμητικό μήκους 10 χαρακτήρων. Έτσι επιτυγχάνεται η μετατροπή ενός array από box σε ένα αλφαριθμητικό μήκους πολλαπλάσιου του 10. Ο κώδικας των μετατροπών φαίνεται στα Σχήμα Π 6-4 και Σχήμα Π 6-5.

```php
function exportstr($coreAlocation_boxes) {
    $result = "";
    foreach ($coreAlocation_boxes as $box) {
        $result .= numbertochar($box['z'], 1);
        $result .= numbertochar($box['core'], 1);
        $result .= numbertochar($box['left'], 2);
        $result .= numbertochar($box['top'], 2);
        $result .= numbertochar($box['right'], 2);
        $result .= numbertochar($box['bottom'], 2);
    }
    return $result;
}
```

Σχήμα Π 6-4 Κώδικας μετατροπής array από box σε αλφαριθμητικό.

```php
function parsestr($instr = '') {
    $coreAlocation_boxes = array();
    $len = floor(strlen($instr) / 10);
    for ($i = 0; $i < $len; $i++) {
        $b = substr($instr, $i * 10, 10);
        $z = chartonum(substr($b, 0, 1));
        $core = chartonum(substr($b, 1, 1));
        $left = chartonum(substr($b, 2, 2));
        $top = chartonum(substr($b, 4, 2));
        $right = chartonum(substr($b, 6, 2));
        $bottom = chartonum(substr($b, 8, 2));
        $coreAlocation_boxes[$z] = array(
            'z' => $z,
            'core' => $core,
            'left' => $left,
            'top' => $top,
            'right' => $right,
            'bottom' => $bottom);
    }
    return $coreAlocation_boxes;
}
```

Σχήμα Π 6-5 Κώδικας μετατροπής αλφαριθμητικού σε array από box.

Όλες οι παραπάνω συναρτήσεις μετατροπής ορίζονται στην lib_coreallocation.php ώστε να είναι προσβάσιμες από όποια σελίδα τις χρειάζεται.

Παράρτημα 7. Κώδικας συνάρτησης κατασκευής προκαθορισμένων κατανομών κόμβων

```
function loadcoreAlocationPreset($instr = "verticalstripes", $dimx, $dimy, $cores) {
   $nny = 2 * $dimy + 1;
   $nnx = 2 * $dimx + 1;
   $boxes = array();
   switch ($instr) {
      case "verticalstripes":
         $stripelen = floor($nnx / $cores);
         for ($i = $cores; $i > 1; $i-=1) {
            $boxes[$i] = array(
               'z' => $i,
               'core' => $i,
               'left' => ($i - 1) * $stripelen + ($nnx - $cores * $stripelen),
               'top' => 1,
               'right' => $i * $stripelen + ($nnx - $cores * $stripelen),
               'bottom' => $nny);
         }
         break;
      case "horisontalstripes":
         $stripelen = floor($nny / $cores);
         for ($i = $cores; $i > 1; $i-=1) {
            $boxes[$i] = array(
               'z' => $i,
               'core' => $i,
               'left' => 1,
               'top' => 1,
               'right' => $nnx,
               'bottom' => $nny - ($i - 1) * $stripelen);
         }
         break;
      default:
         $stripelen = floor($nnx / $cores);
         for ($i = $cores; $i > 1; $i-=1) {
            $boxes[$i] = array(
               'z' => $i,
               'core' => $i,
               'left' => ($i - 1) * $stripelen + ($nnx - $cores * $stripelen),
               'top' => 1,
               'right' => $i * $stripelen + ($nnx - $cores * $stripelen),
               'bottom' => $nny);
         }
   }
   return $boxes;
}
```

Παράρτημα 8. Ανάλυση σχεδιασμού σελίδας κατανομής των κόμβων του πλέγματος

Για την διαχείριση όλων των τμημάτων της σελίδας με javascript χρειάζεται να χρησιμοποιηθεί ένα καλά δομημένο σύστημα ονομασίας των στοιχείων. Για τον εντοπισμό των HTML στοιχείων με το javascript χρησιμοποιείται η παράμετρος id των στοιχείων. Για κάθε στοιχείο της array των box καθώς και για το αρχικό box 1 όπως έχει οριστεί κατά την μοντελοποίηση (πρβλ. Κεφ. 4.1) ορίζεται μια σειρά από στοιχεία HTML:

- μια γραμμή πίνακα με id “box_tr_#” στον πίνακα εισαγωγής των τιμών. Για τα box της array επιπλέων ορίζονται 5 πεδία εισαγωγής τιμών: μία λίστα επιλογής με id “box_core_#” και 4 πεδία εισαγωγής κειμένου με id “box_top_#”, “box_left_#”, “box_height_#” και “box_width_#” για την εισαγωγή του core του box και της θέσης και των διαστάσεων του box αντίστοιχα.
- δύο στοιχεία div με id “drawingbox_#” και “drawingbox3d_#” για τον σχεδιασμό της αναπαράστασης του box στην προεπισκόπηση και υπό γωνία προβολή αντίστοιχα.

Όπου “#” η τιμή του z στο επιλεγμένο box.

Για την διαχείριση αυτών των κατασκευάζεται μια σειρά από συναρτήσεις σε javascript.

Η συνάρτηση loadiddata λαμβάνει σαν όρισμα έναν ακέραιο i και επιστρέφει σαν τιμές τις μεταβλητές box_core, box_left, box_top, box_right, box_bottom, box_height και box_width που αντιστοιχούν στον επεξεργαστή, τις συντεταγμένες της άνω αριστερά γωνίας, τις συντεταγμένες τις κάτω δεξιά γωνίας, το πλάτος και το μήκος του box που βρίσκεται το z ίσο με i. Καθότι η ονομασία των στοιχείων που περιέχουν την πληροφορία αυτή έχει γίνει με βάση το z, η συνάρτηση λαμβάνει τις τιμές από τα στοιχεία κατασκευάζοντας το id που αντιστοιχεί στο καθένα. Η συνάρτηση χρησιμεύει για την λήψη των πληροφοριών ενός box και ο κώδικας της φαίνεται στο Σχήμα Π 8-1.

```
function loadiddata(i){
    coreid="#box_core_".concat(i);
    leftid="#box_left_".concat(i);
    topid="#box_top_".concat(i);
    widthid="#box_width_".concat(i);
    heightid="#box_height_".concat(i);
    box_core=parseInt($(coreid).val());
    box_left=parseInt($(leftid).val());
    box_top=parseInt($(topid).val());
    box_height=parseInt($(heightid).val());
    box_width=parseInt($(widthid).val());
    box_right=box_left+box_width-1;
    box_bottom=box_top+box_height-1;
}
```

Σχήμα Π 8-1 Κώδικας javascript της συνάρτησης loadiddata.

Η συνάρτηση drawboxes ανανεώνει την προεπισκόπηση. Η συνάρτηση λαμβάνει σαν σταθερές τιμές το πλήθος των στοιχείων της array των box καθώς και την κλίμακα και τις διαστάσεις του διαγράμματος προεπισκόπησης από την php κατά την φόρτωση της σελίδας. Κατόπιν για κάθε στοιχείο της array των box καλεί την loadiddata και με βάση τις τιμές που λαμβάνει από εκεί αλλάζει την θέση τις διαστάσεις και το χρώμα του div που αντιστοιχεί στο box αυτό, έτσι ώστε το αποτέλεσμα που βλέπει ο χρήστης να ανταποκρίνεται στο αποτέλεσμα του μοντέλου σχεδίασης της κατανομής των κόμβων του πλέγματος. Η συνάρτηση αυτή καλείται μετά από κάθε αλλαγή τιμής και ο κώδικας της φαίνεται στο Σχήμα Π 8-2.

```
function drawboxes(){
      numfields = <?php echo $boxcount; ?>;
      drawstep= <?php echo $drawstep; ?>;
      widthbox=<?php echo $drawx;?>;
      heightbox=<?php echo $drawy;?>;
      for (i = numfields; i > 1; i -= 1){
         loadiddata(i)
          drawboxid="#drawingbox_".concat(i);

         $(drawboxid).attr("class", "corecolor".concat(box_core));
         $(drawboxid).css('top',(box_top-1)*drawstep);
         $(drawboxid).css('left',(box_left-1)*drawstep);
         $(drawboxid).css('width',box_width*drawstep);
         $(drawboxid).css('height',box_height*drawstep);

         drawbox3did="#drawingbox3d_".concat(i);

         $(drawbox3did).attr("class", "corecolor".concat(box_core));
         $(drawbox3did).css('top',(box_top-1)*drawstep/2.5 + heightbox/2-10*(i-1));
         $(drawbox3did).css('left',(box_left-1)*drawstep/1.5 + widthbox/6);
         $(drawbox3did).css('width',box_width*drawstep/1.5);
         $(drawbox3did).css('height',box_height*drawstep/2.5 );
      }
   }
```

Σχήμα Π 8-2 Κώδικας javascript της συνάρτησης drawboxes.

Η συνάρτηση makecorestr ελέγχει τις τιμές των πεδίων του πίνακα, τις διορθώνει ώστε να είναι εντός των επιτρεπτών και κατασκευάζει ένα συμπιεσμένο αλφαριθμητικό που αντιστοιχεί στην τρέχουσα κατανομή και το αποθηκεύει σε ένα κρυφό πεδίο κειμένου της σελίδας με id c_core_part_str. Ο κώδικας της φαίνεται στο Σχήμα Π 8-3.

```
function makecorestr(){
    numfields = <?php echo $boxcount; ?>;
    maxx = <?php echo $c_nnx; ?>;
    maxy = <?php echo $c_nny; ?>;
    retstr = '';
    for (i = numfields; i > 1; i -= 1){
        retstr =retstr.concat(numbertochar(i, 1));
        loadiddata(i)
        retstr =retstr.concat(numbertochar(box_core,1));

        if(box_top<1){box_top=1;$(topid).val(box_top);}
        if(box_left<1){box_left=1;$(leftid).val(box_left);}
        if(box_width<1){box_width=1;$(widthid).val(box_width);box_right=box_width+box_left-1;}
        if(box_height<1){box_height=1;$(heightid).val(box_height);box_bottom=box_height+box_top-1;}

        if(box_top>maxy){box_top=maxy;$(topid).val(box_top);}
        if(box_height+box_top-1>maxy){box_height=maxy-
box_top+1;$(heightid).val(box_height);box_bottom=box_height+box_top-1;}

        if(box_left>maxx){box_left=maxx;$(leftid).val(box_left);}
        if(box_width+box_left-1>maxx){box_width=maxx-
box_left+1;$(widthtid).val(box_width);box_right=box_width+box_left-1;}

        retstr =retstr.concat(numbertochar(box_left, 2));
        retstr =retstr.concat(numbertochar(box_top, 2));
        retstr =retstr.concat(numbertochar(box_right, 2));
        retstr =retstr.concat(numbertochar(box_bottom, 2));
    }

    $("#c_core_part_str").val(retstr);
    drawboxes()
}
```

Σχήμα Π 8-3 Κώδικας javascript της συνάρτησης makecorestr.

Η συνάρτηση box_move αλλάζει την θέση μεταξύ 2 στοιχείων της array των box. Η συνάρτηση λαμβάνει 2 ακεραίους σαν ορίσματα. Ο πρώτος αντιστοιχεί στο z του ενός στοιχείου ενώ ο δεύτερος στην σχετική τιμή του z του άλλου στοιχείου ως προς το z του πρώτου. Η συνάρτηση αντί να αλλάξει θέση τα στοιχεία της array ανταλλάσσει μεταξύ των δύο στοιχείων τις τιμές των παραμέτρων τους, πλην της z που έχει το ίδιο αποτέλεσμα. Αρχικά προσθέτει τα δύο ορίσματα για να υπολογίσει το z του δεύτερου στοιχείου. Κατόπιν εντοπίζει τα πεδία του πίνακα που αντιστοιχούν στα 2 αυτά box, λαμβάνει τις τιμές τους και τις ανταλλάσσει μεταξύ τους με την χρήση βοηθητικών μεταβλητών. Τέλος, καλεί την makecorestr για να ανανεώσει το αλφαριθμητικό πεδίο και την προεπισκόπηση. Ο Κώδικας της φαίνεται στο Σχήμα Π 8-4.

Ορίζεται επίσης μια σειρά από 4 συναρτήσεις για την αυξομείωση των τιμών των πεδίων, δύο για την προσθαφαίρεση του βήματος για τον άξονα των X και δύο για την προσθαφαίρεση του βήματος για τον άξονα των Y. Οι συναρτήσεις λαμβάνουν σαν όρισμα το πεδίο, του οποίου η τιμή πρόκειται να αλλάξει, και αλλάζουν την τιμή του κατά το αντίστοιχο βήμα τις καθεμίας. Μετά την αλλαγή καλούν την

makecorestr για την ανανέωση του αλφαριθμητικού πεδίου και της προεπισκόπησης. Ο κώδικας τους φαίνεται στο Σχήμα Π 8-5.

```
function box_move(inid,step){
    newid=inid + step;
    in_coreid="#box_core_".concat(inid);
    in_leftid="#box_left_".concat(inid);
    in_topid="#box_top_".concat(inid);
    in_widthid="#box_width_".concat(inid);
    in_heightid="#box_height_".concat(inid);

    new_coreid="#box_core_".concat(newid);
    new_leftid="#box_left_".concat(newid);
    new_topid="#box_top_".concat(newid);
    new_widthid="#box_width_".concat(newid);
    new_heightid="#box_height_".concat(newid);

    in_core=$(in_coreid).val();
    in_box_left=$(in_leftid).val();
    in_box_top=$(in_topid).val();
    in_box_width=$(in_widthid).val();
    in_box_height=$(in_heightid).val();

    $(in_coreid).val($(new_coreid).val());
    $(in_leftid).val($(new_leftid).val());
    $(in_topid).val($(new_topid).val());
    $(in_widthid).val($(new_widthid).val());
    $(in_heightid).val($(new_heightid).val());

    $(new_coreid).val(in_core);
    $(new_leftid).val(in_box_left);
    $(new_topid).val(in_box_top);
    $(new_widthid).val(in_box_width);
    $(new_heightid).val(in_box_height);

    makecorestr();
}
```

Σχήμα Π 8-4 Κώδικας javascript της συνάρτησης box_move.

```
function step_increase_x(input){
    $(input).val(parseInt($(input).val()) + <?php echo $movestep_x; ?>);
    makecorestr();
}
function step_decrease_x(input){
    $(input).val(parseInt($(input).val()) - <?php echo $movestep_x; ?>);
    makecorestr();
}
function step_increase_y(input){
    $(input).val(parseInt($(input).val()) + <?php echo $movestep_y; ?>);
    makecorestr();
}
function step_decrease_y(input){
    $(input).val(parseInt($(input).val()) - <?php echo $movestep_y; ?>);
    makecorestr();
}
```

Σχήμα Π 8-5 Κώδικας javascript συναρτήσεις αυξομείωσης τιμών πίνακα.

Η συνάρτηση selectedbox λαμβάνει σαν όρισμα ένα ακέραιο και σχεδιάζει ένα μαύρο περίγραμμα γύρω από την γραμμή του πίνακα και τα δύο div που αντιστοιχούν στο στοιχείο του array των box με z ίσο με το όρισμα. Η συνάρτηση καθαρίζει το περίγραμμα για την γραμμή και τα δύο div που αντιστοιχούν σε κάθε στοιχείο και κατόπιν ορίζει ένα σε αυτά που αντιστοιχούν στο όρισμα. Η συνάρτηση χρησιμεύει στην διευκόλυνση του χρήστη να αναγνωρίζει οπτικά σε ποιο ακριβώς τμήμα της προεπισκόπησης αντιστοιχεί κάθε γραμμή του πίνακα. Ο κώδικας της φαίνεται στο Σχήμα Π 8-6.

```
function selectedbox(id){
        numfields = <?php echo $boxcount; ?>;
        for (i = numfields; i > 0; i -= 1){
            drawboxid="#drawingbox_".concat(i);
            boxtrid="#box_tr_".concat(i);
            drawbox3did="#drawingbox3d_".concat(i);
            $(drawboxid).css('outline','0px solid black');
            $(boxtrid).css('outline','0px solid black');
            $(drawbox3did).css('outline','0px solid black');
        }
        drawboxid="#drawingbox_".concat(id);
        boxtrid="#box_tr_".concat(id);
        drawbox3did="#drawingbox3d_".concat(id);
        $(drawboxid).css('outline','1px solid black')
        $(boxtrid).css('outline','1px solid black')
        $(drawbox3did).css('outline','1px solid black')
    }
```

Σχήμα Π 8-6 Κώδικας javascript της συνάρτησης selectedbox.

Για κάθε μία από τις εντολές κλήσης σελίδας κατασκευάζεται μία συνάρτηση ακόμα. Έτσι ορίζονται 3 συναρτήσεις, η addrow, η removerow και η createrequest.

Οι addrow και removerow αναλαμβάνουν την επανάκληση της σελίδας με την προσθήκη της παραμέτρου draw_action με τις αντίστοιχες τιμές για προσθήκη ή διαγραφή στοιχείου από το array των box. Για να επιτευχθεί αυτό, απλά κατασκευάζουν μια διεύθυνση url με την χρήση της μεταβλητής $requeststr που έχει ήδη οριστεί και περιέχει όλες τις παραμέτρους, πλην τις κατανομής, και επισυνάπτοντας σε αυτή την κατανομή όπως είναι αποθηκευμένη στο κρυφό πεδίο c_core_part_str και την παράμετρο draw_action με την επιθυμητή τιμή. Αφού κατασκευαστεί η διεύθυνση url, απλά την καλούν σαν νέα τοποθεσία του ενεργού παραθύρου του browser. Ο κώδικας τους παρουσιάζεται στο Σχήμα Π 8-7.

Αντίστοιχα η createrequest κατασκευάζει ένα παρόμοιο url, με στόχο όμως την αρχική σελίδα κατασκευής εντολής εκτέλεσης, για να ολοκληρωθεί η διαδικασία. Πριν αποστείλει την εντολή κλήσης της σελίδας όμως, κατασκευάζει το array της κατανομής των κόμβων του πλέγματος σύμφωνα με το μοντέλο και ελέγχει ότι κάθε επεξεργαστής έχει τουλάχιστον έναν κόμβο που του αντιστοιχεί για να είναι αποδεκτή η κατανομή. Εάν δεν συμβαίνει αυτό, η κλήση διακόπτεται και εμφανίζεται μήνυμα σφάλματος στον χρήστη. Ο κώδικας της αναλυτικά φαίνεται στο Σχήμα Π 8-8.

```
function addrow(){
    target_url='coreallocation.php<?php echo $requeststr;
?>'.concat($("#c_core_part_str").val()).concat('&draw_action=addbox');
    window.location = target_url;
}

function removerow(id){
    target_url='coreallocation.php<?php echo $requeststr;
?>'.concat($("#c_core_part_str").val()).concat('&draw_action=removebox_').concat(id);
    window.location = target_url;
}
```

Σχήμα Π 8-7 Κώδικας javascript συναρτήσεις κλήσης προσθήκης και διαγραφής box.

```
function createrequest(){
    createrequest:{
        var corealoc = new Array();
        nnx=<?php echo $c_nnx; ?>;
        nny=<?php echo $c_nny; ?>;
        for (k=1; k <= nnx;k +=1){
            for (j=1; j <= nny;j +=1){
                corealoc[(k - 1) * nny + j]=1;
            }
        }
        numfields = <?php echo $boxcount; ?>;
        numcores = <?php echo $c_cores; ?>;
        for (i = 2; i <= numfields; i += 1){
            loadiddata(i);
            for (k=box_left; k <= box_right ;k +=1){
                for (j=box_top; j <= box_bottom;j +=1){
                    corealoc[(k - 1) * nny + j]=box_core;
                }
            }
        }

        for (i=1; i < numcores+1 ; i += 1) {
            if( corealoc.indexOf(i) == -1){
                window.alert("Core number ".concat(i).concat(" is not used! Please make sure each core is assigned
to at least one node."));
                break createrequest;
            }
        }
        target_url='index.php<?php echo $requeststr;
?>'.concat($("#c_core_part_str").val()).concat('&Submit=Submit');
        window.location = target_url;
    }
}
```

Σχήμα Π 8-8 Κώδικας javascript της συνάρτησης createrequest.

Τέλος, κατασκευάζεται μια συνάρτηση bindmakestr, η οποία δεσμεύει σε όλα τα πεδία εισαγωγής τιμών την συνάρτηση makecorestr σε περίπτωση αλλαγής τιμών. Έτσι ανανεώνεται το κρυφό πεδίο αποθήκευσης του αλφαριθμητικού και η προεπισκόπηση κάθε φορά που αλλάζει τιμή στο καθένα από αυτά. Η συνάρτηση αυτή εντοπίζει τα πεδία που αντιστοιχούν σε κάθε στοιχείο της array των box με μία επανάληψη και σε καθένα από αυτά δεσμεύει την makecorestr με το κατάλληλο event. Ο κώδικας της παρουσιάζεται στο Σχήμα Π 8-9 και γίνεται κλήση της αμέσως μετά την ολοκλήρωση της φόρτωσης της σελίδας.

```
function bindmakestr(){
   numfields = <?php echo $boxcount; ?>;
   for (i = numfields; i > 1; i -= 1){
      id="#box_core_".concat(i);
      $(id).change(function() {makecorestr();});
      id="#box_left_".concat(i);
      $(id).bind('input', function() {makecorestr();});
      id="#box_top_".concat(i);
      $(id).bind('input', function() {makecorestr();});
      id="#box_width_".concat(i);
      $(id).bind('input', function() {makecorestr();});
      id="#box_height_".concat(i);
      $(id).bind('input', function() {makecorestr();});
   }
}
```

Σχήμα Π 8-9 Κώδικας javascript της συνάρτησηςδέσμευσης της makecorestr στην αλλαγή τιμών πεδίων.

Αφού οριστούν οι συναρτήσεις διαχείρισης των στοιχείων html γίνεται ο σχεδιασμός των τμημάτων της σελίδας. Οι προεπισκόπηση και η υπο γωνία προβολή αποτελούνται από ένα div που περιέχει εντός του τα div που αντιστοιχούν στα box. Το αρχικό div έχει την css ιδιότητα “position:relative”, ενώ τα div που περιέχει έχουν την ιδιότητα “position:absolute”. Αυτό επιτρέπει τα div αυτά να σχεδιάζονται σε συντεταγμένες απόλυτες εντός του αρχικού div και διευκολύνει τον σχεδιασμό της προεπισκόπησης. Τα div των στοιχείων σχεδιάζονται με βάση τις αρχικές τιμές της κατανομής πού έλαβε η σελίδα, όπως αυτές περιέχονται στο array $coreAlocation_boxes, και σε καθένα από αυτά δεσμεύεται η javascript συνάρτηση selectedbox με όρισμα την τιμή z του box που τους αντιστοιχεί στο onclick, με αποτέλεσμα όταν γίνεται click του καθενός από αυτά να σχεδιάζεται ένα μαύρο περίγραμμα σε αυτό και την γραμμή που του αντιστοιχεί. Ο κώδικας σχεδιασμού των αρχικών στοιχείων των δύο διαγραμμάτων αυτών φαίνεται στο Σχήμα Π 8-10.

```
echo '<center><table cellspan="0px" colspan="0" spacing="0" ><tr><td valign="top">';

echo '<table > <tr>';
echo '<td class="tableheader"> Prespective View </td><td class="tableheader"> Preview </td></tr><tr>';
echo '<td style="background-color:#ffffff;z-index:-10;padding:15px;overflow:hidden;"><div class="pane3d"
style="background-color:#ffffff;height:' . $drawy . 'px;width:' . $drawx . 'px;" >';
echo '<div style="position:absolute;top:'.($drawy/2).'px;left:'.($drawx/6).'px;height:' . $drawy /2.5 . 'px;width:' .
$drawx / 1.5 . 'px;outline:1px solid black;" id="drawingbox3d_1" class="corecolor1" onclick="if (event.target
=== this) selectedbox(1);" ></div>';

foreach ($coreAlocation_boxes as $box) {
   echo '<div style="position:absolute;z-index:' . $box['z'] . ';outline:0px solid black;" class="corecolor' .
$box['core'] . '" id="drawingbox3d_' . $box['z'] . '" onclick="if (event.target === this) selectedbox(' . $box['z'] .
');"></div>';
}
echo '</div>';
echo '</td>';
echo '<td style="background-color:#ffffff;z-index:-10;padding:15px;" >';
echo '<div style="position:relative;height:' . $drawy . 'px;width:' . $drawx . 'px;">';
echo '<div style="position:absolute;top:0px;left:0px;height:' . $drawy . 'px;width:' . $drawx . 'px;outline:1px
solid black;" id="drawingbox_1" class="corecolor1" onclick="if (event.target === this) selectedbox(1);"
></div>';
foreach ($coreAlocation_boxes as $box) {
   echo '<div style="position:absolute;z-index:' . $box['z'] . ';outline:0px solid black;" class="corecolor' .
$box['core'] . '" id="drawingbox_' . $box['z'] . '" onclick="if (event.target === this) selectedbox(' . $box['z'] .
');"></div>';
}
echo '<div style="position:absolute;top:-15px;right:' . ($drawx + 2) . 'px;" >1</div>';
echo '<div style="position:absolute;top:' . ($drawy - 15) . 'px;right:' . ($drawx + 2) . 'px;" >' . $c_nny . '</div>';
echo '<div style="position:absolute;top:' . ($drawy - 2) . 'px;left:0px;" >1</div>';
echo '<div style="position:absolute;top:' . ($drawy - 2) . 'px;left:' . ($drawx - 10) . 'px;" >' . $c_nnx . '</div>';
echo '</div>';

echo '</td>';

echo '</tr><tr><td class="tableheader" colspan="2"> Legent </td></tr><tr><td  colspan="2" >';
echo '<table cellspan="0px" colspan="0" spacing="0" >';
for ($i = 1; $i <= $c_cores; $i++) {
   if ($i % 5 == 1) {echo '<tr>';}
   echo '<td width="120px" ><div style="position:relative;height:25px;" > <div
style="position:absolute;top:5px;left:5px;width:20px;height:15px;border:1px solid black;" class="corecolor' . $i
. '"></div><div style="position:absolute;left:30px;" > Core no ' . $i . '</div></div></td>';
   if ($i % 5 == 0) {echo '</tr>';}
}
echo '</table>';
echo '</td></tr></table>';
```

Σχήμα Π 8-10 Κώδικας σχεδιασμού προεπισκόπησης σελίδας κατανομής κόμβων σε επεξεργαστές.

Για τον σχεδιασμό του πίνακα εισαγωγής των τιμών ορίζονται 2 συναρτήσεις η createboxgrid, που αναλαμβάνει τον σχεδιασμό όλου του πίνακα, και η createboxfield, που σχεδιάζει την κάθε γραμμή αυτού. Η createboxgrid σχεδιάζει την κεφαλίδα του πίνακα καθώς και την γραμμή του αρχικού box 1, όπως έχει οριστεί κατά την μοντελοποίηση που περιέχει στατικές τιμές και δυο πλήκτρα. Τα δυο πλήκτρα

που ορίζονται καλούν όταν πατηθούν τις javascript της συνάρτησης addrow και createrequest για την προσθήκη νέου στοιχείου και την ολοκλήρωση της διαδικασίας σχεδιασμού μέσω του javascript. Εντός της createboxgrid καλείται με επανάληψη η createboxfield για κάθε στοιχείο του array των box.

Η createboxfield με βάση το z κατασκευάζει και ονομάζει μια γραμμή πίνακα στην οποία δεσμεύεται η javascript συνάρτηση selectedbox με όρισμα την τιμή z του box που της αντιστοιχεί στο onclick για τον σχεδιασμό του περιγράμματος. Εντός των κελιών κατασκευάζεται η σειρά από τα πεδία που της αντιστοιχούν και δίπλα από το καθένα κατασκευάζονται 2 μικρά πλήκτρα, που καλούν τις αντίστοιχες συναρτήσεις javascript για αύξηση και μείωση της τιμής του πεδίου με βάση το αντίστοιχο βήμα. Τέλος, ορίζονται 3 ακόμα πλήκτρα υπό προϋποθέσεις. Εάν δεν είναι η κορυφαία γραμμή, κατασκευάζεται ένα πλήκτρο που αλλάζει την θέση αυτής με την παραπάνω της, καλώντας την συνάρτηση box_move με ορίσματα το z που αντιστοιχεί σε αυτή και τον αριθμό 1. Αντίστοιχα κατασκευάζεται και ένα πλήκτρο για την αλλαγή της θέσης αυτής προς τα κάτω, εφόσον δεν είναι η τελευταία γραμμή. Και τέλος κατασκευάζεται ένα πλήκτρο διαγραφής αυτής της γραμμής, καλώντας τη συνάρτηση removebox με όρισμα το z, που αντιστοιχεί σε αυτή εφόσον δεν πρόκειται για την μοναδική γραμμή που έχει απομείνει.

Αφού ολοκληρωθεί ο σχεδιασμός του πίνακα, ορίζεται και το κρυφό πεδίο για την αποθήκευση του αλφαριθμητικού. Ο κώδικας των δύο αυτών των συναρτήσεων φαίνεται στο Σχήμα Π 8-11 και στο Σχήμα Π 8-12 αντίστοιχα.

```
function createboxgrid($c_cores, $c_nnx, $c_nny, $coreAlocation_boxes, $boxcount) {
   echo '<table cellspan="0px" colspan="0" spacing="0">
    <tr class="tableheader">
       <td>Layer</td>
       <td>Core</td>
       <td>Top</td>
       <td>Left</td>
       <td>Height</td>
       <td>Width</td>
       <td colspan="3">';
   echo '<button id="addbutton" onclick="addrow();">Add Box</button>';
   echo '<button  id="createbutton" onclick="createrequest();">Create Request</button>';
   echo '</td></tr>';
   foreach ($coreAlocation_boxes as $box) {
      createboxfield($box, $c_cores, $c_nnx, $c_nny, $boxcount);   }
   echo '<tr id="box_tr_1" style="outline:1px solid black;" onclick="selectedbox(1);">
       <td>1</td>
       <td>1</td>
       <td>1</td>
       <td>1</td>
       <td>' . $c_nny . '</td>
       <td>' . $c_nnx . '</td>
       <td></td>
        <td></td>
    </tr></table>  ';
   echo '<input id="c_core_part_str" name="c_core_part_str" type="hidden" style="width:300px" ></input>';}
```

Σχήμα Π 8-11 Κώδικας σχεδιασμού πίνακα εισαγωγής τιμών σελίδας κατανομής κόμβων σε επεξεργαστές.

```
function createboxfield($box, $c_cores, $c_nnx, $c_nny, $max_z) {
   echo '<tr id="box_tr_' . $box['z'] . '"  onclick="selectedbox(' . $box['z'] . ');">';
   echo '<td>' . $box['z'] . "</td>";
   echo '<td>';
   echo '<select id="box_core_' . $box['z'] . '">';
   for ($i = 1; $i <= $c_cores; $i++) {
      echo '<option  value="' . $i . '" ';
      if ($i == $box['core']) {
         echo 'selected';
      }
      echo '>' . $i . '</option>';
   }
   echo '</select></td>';
   echo '<td><input type="text" id="box_top_' . $box['z'] . '"  value="' . $box['top'] . '" class="gridinput"
><button class="adjustbutton" onclick="step_increase_y(\'#box_top_' . $box['z'] . '\');">+</button><button
class="adjustbutton" onclick="step_decrease_y(\'#box_top_' . $box['z'] . '\');">-</button></td>';
   echo '<td><input type="text" id="box_left_' . $box['z'] . '"  value="' . $box['left'] . '"
class="gridinput"><button class="adjustbutton" onclick="step_increase_x(\'#box_left_' . $box['z'] .
'\');">+</button><button class="adjustbutton" onclick="step_decrease_x(\'#box_left_' . $box['z'] . '\');">-
</button></td>';
   echo '<td><input type="text" id="box_height_' . $box['z'] . '"  value="' . ($box['bottom'] - $box['top'] + 1) . '"
class="gridinput" ><button class="adjustbutton" onclick="step_increase_y(\'#box_height_' . $box['z'] .
'\');">+</button><button class="adjustbutton" onclick="step_decrease_y(\'#box_height_' . $box['z'] . '\');">-
</button></td>';
   echo '<td><input type="text" id="box_width_' . $box['z'] . '"  value="' . ($box['right'] - $box['left'] + 1) . '"
class="gridinput" ><button class="adjustbutton" onclick="step_increase_x(\'#box_width_' . $box['z'] .
'\');">+</button><button class="adjustbutton" onclick="step_decrease_x(\'#box_width_' . $box['z'] . '\');">-
</button></td>';
   echo '<td>';
   if ($box['z'] <> $max_z) {
      echo '<button onclick="box_move(' . $box['z'] . ',1);">Up</button>';
   }
   echo '</td><td>';
   if ($box['z'] > 2) {
      echo '<button onclick="box_move(' . $box['z'] . ',-1);">Down</button>';
   }
   echo '</td><td>';
   if ($max_z > 2)
      echo '<button id="removebuton' . $box['z'] . '" onclick="removerow(' . $box['z'] . ')");>Remove</button>';
   echo '</td>';
   echo '</tr>';
}
```

Σχήμα Π 8-12 Κώδικας σχεδιασμού γραμμής πίνακα εισαγωγής τιμών σελίδας κατανομής κόμβων σε επεξεργαστές.

Παράρτημα 9. Σχεδιασμός καρτελών σελίδας προβολής των αποτελεσμάτων.

Για τον σχεδιασμό των καρτελών (tabs) πρώτα εκτυπώνεται στην σελίδα ένας πίνακας μία γραμμής, που περιέχει τις κεφαλίδες της κάθε καρτέλας σαν πεδία. Κάθε κεφαλίδα περιέχεται σε ένα link το οποίο έχει id “a_tab#”, όπου # ο αριθμός του tab που του αντιστοιχεί, και αντί για μετάβαση σελίδας στο onclick εκτελεί μία συνάρτηση javascript, ονομαζόμενη selecttab με όρισμα ”tab#”, όπου # ο αριθμός του tab που του αντιστοιχεί. Τέλος, χρησιμοποιούνται δύο css κλάσεις μία για το επιλεγμένο tab ονομαζόμενη “tabstyleactive”, και μια για τα μη επιλεγμένα, ονομαζόμενη “tabstyle” για τη μορφοποίηση τους. Αρχικά, εφόσον φαίνεται το tab 1 ορίζεται το class αυτού σαν “tabstyleactive” και των υπολοίπων σαν “tabstyle”.

Αφού σχεδιαστούν οι κεφαλίδες, σχεδιάζονται τα tabs. Κάθε καρτέλα περιέχεται σε ένα div HTML tag με όνομα ”tab#”, όπου # ο αριθμός της. Ανά πάσα στιγμή τα τρία από αυτά κρύβονται με την χρήση της css παραμέτρου “display:none” στο style του. Έτσι, μόνο ένα (το επιλεγμένο) παρουσιάζεται στον χρήστη. Η εμφάνιση και απόκρυψη των tabs γίνεται με την javascript της συνάρτησης selecttab που καλείται από τα link στις κεφαλίδες. Η selecttab λαμβάνει σαν όρισμα το id μιας καρτέλας. Αρχικά κρύβει όλες τις καρτέλες και μορφοποιεί όλες τις κεφαλίδες σαν μη επιλεγμένες με την css κλάση “tabstyle” και κατόπιν εμφανίζει την καρτέλα που έχει επιλεχθεί και μορφοποιεί την κεφαλίδα που του αντιστοιχεί σαν επιλεγμένο με την css κλάση “ tabstyleactive ”. Ο κώδικας του javascript αυτού φαίνεται στο Σχήμα Π 9-1.

Εντός του κάθε div που αντιστοιχεί στην κάθε καρτέλα εκτυπώνονται τα περιεχόμενά της:

- Στην πρώτη εκτυπώνεται ο πίνακας με τα σφάλματα επαναλήψεων.
- Στη δεύτερη παρουσιάζεται το διάγραμμα σύγκλισης. Για την εμφάνιση του διαγράμματος χρησιμοποιείται ένα img HTML tag, τοποθετώντας στο src την διεύθυνση για την σελίδα που κατασκευάζει το διάγραμμα με την παράμετρο myID επισυναπτόμενη σε αυτή. Στην παράμετρο αυτή δίνεται η τιμή του κλειδιού της εγγραφής που παρουσιάζεται αυτή την στιγμή.
- Στην τρίτη παρουσιάζεται το διάγραμμα κατανομής των κόμβων του πλέγματος. Για την εμφάνιση του ακολουθείται ακριβώς η ίδια διαδικασία με το διάγραμμα σύγκλισης.
- Τέλος, στην τέταρτη απλά εκτυπώνονται τα σφάλματα εκτέλεσης της εφαρμογής.

Ο κώδικας προβολής των καρτελών φαίνεται αναλυτικά στο Σχήμα Π 9-2.

```
<script type="text/javascript" language="javascript">
   function selecttab(tab) {
       document.getElementById('tab1').style.display = 'none';
       document.getElementById('tab2').style.display = 'none';
       document.getElementById('tab3').style.display = 'none';
       document.getElementById('tab4').style.display = 'none';
       document.getElementById('a_tab1').setAttribute('class', 'tabstyle');
       document.getElementById('a_tab2').setAttribute('class', 'tabstyle');
       document.getElementById('a_tab3').setAttribute('class', 'tabstyle');
       document.getElementById('a_tab4').setAttribute('class', 'tabstyle');
       document.getElementById(tab).style.display = 'block';
       document.getElementById('a_'+tab).setAttribute('class', 'tabstyleactive');
   };
</script>
```

Σχήμα Π 9-1 Κώδικας συνάρτησης επιλογής TABS σελίδας προβολής λεπτομερειών.

```
<table cellpadding="0" cellspacing="0"> <tr>
     <td  > <a id="a_tab1" href="#"  onclick="selecttab('tab1')" class="tabstyleactive"> Data </a> </td>
     <td>   </td>
     <td><a id="a_tab2" href="#" onclick="selecttab('tab2')" class="tabstyle"> Plot Iterations - Residual</a> </td>
     <td>   </td>
     <td> <a id="a_tab3" href="#" onclick="selecttab('tab3')" class="tabstyle"> Code - Node Diagram</a> </td>
      <td>   </td>
      <td><a id="a_tab4" href="#" onclick="selecttab('tab4')" class="tabstyle"> Execution Errors </a> </td>
      <td width="100%"></td><td>   </td>
   </tr></table>
<div id="tab1" class="listtable" style="padding:10px 10px 10px 10px;" >
      <?php
      echo ' <table class="listtable" cellpadding="0" cellspacing="0" ><tr> <td class="tableheader" > Iteration </td>
<td class="tableheader" > Outer Iteration </td> ';
      echo '<td class="tableheader" > Krylov Iteration </td><td class="tableheader" > Residual </td></tr>';
      $result2 = mysql_query("SELECT  outer_iter,inner_iter,diff,(outer_iter-1)*" . $row['innerit'] . "+inner_iter titer
FROM GMRES_Request_Iterations WHERE request_id='" . $id . "' ");
      while ($row2 = mysql_fetch_array($result2)) {
         echo "<tr><td>" . $row2['titer'] . "</td>";
         echo "<td>" . $row2['outer_iter'] . "</td>";
         echo "<td>" . $row2['inner_iter'] . "</td>";
         echo "<td>" . $row2['diff'] . "</td>";
         echo "</tr>";
      }
      echo "</table>"; ?>
</div>
<div id="tab2" class="listtable" style="display:none;height:500px;">
            <?php echo '<img src="img\iterationdiagram.php?myID=' . $row['id'] . '"  />'; ?>
</div>
<div id="tab3" class="listtable" style="display:none;">
            <?php echo '<img src="img\nodediagram.php?myID=' . $row['id'] . '"  />'; ?>
</div>
<div id="tab4" class="listtable" style="display:none;">
            <?php echo $row['errors']; ?>
</div>
```

Σχήμα Π 9-2 Κώδικας σχεδιασμού TABS σελίδας προβολής λεπτομερειών.

Παράρτημα 10. Κώδικας σχεδιασμού διαγράμματος κατανομής των κόμβων.

Μετά από την λήψη των πληροφοριών με βάση τις διαστάσεις ορίζεται η απόσταση μεταξύ των υποδιαιρέσεων στους άξονες. Οι υποδιαιρέσεις λαμβάνονται από ένα array προκαθορισμένων τιμών. Από το array λαμβάνεται για κάθε άξονα η μικρότερη τιμή που δεν χωράει 6 φορές στο μέγιστο του άξονα και αποθηκεύεται σε μια μεταβλητή. Ο κώδικας της διαδικασίας φαίνεται στο Σχήμα Π 10-1.

```
$tickmarkersizes = array(1, 2, 3, 5, 7, 10, 20, 30, 50, 60, 100, 150, 200, 250, 500, 750, 1000);

$notationwidthx = 10;
$notationwidthy = 10;

if ($dimx >= 100) {
   $notationwidthx = 20;
};
if ($dimy >= 100) {
   $notationwidthy = 20;
};

$notationintervalx = $tickmarkersizes[1];
$notationintervaly = $tickmarkersizes[1];

$i = 1;
while (floor($dimx / $notationintervalx) > 5) {
   $i++;
   $notationintervalx = $tickmarkersizes[$i];
}

$i = 1;
while (floor($dimy / $notationintervaly) > 5) {
   $i++;
   $notationintervaly = $tickmarkersizes[$i];
}
```

Σχήμα Π 10-1 Κώδικας υπολογισμού υποδιαιρέσεων αξόνων διαγράμματος κατανομής των κόμβων του πλέγματος.

Κατόπιν ορίζονται διάφορες προκαθορισμένες μεταβλητές σχεδιασμού, όπως διάκενα και πάχος περιγράμματος, και γίνεται ο υπολογισμός της κλίμακας του διαγράμματος, έτσι ώστε να είναι μεγαλύτερο από 200x200 pixel. Με βάση αυτές τις τιμές υπολογίζονται οι διαστάσεις της εικόνας και ορίζεται το αντικείμενο resource αυτής. Ο κώδικας των διαδικασιών αυτών φαίνεται στο Σχήμα Π 10-2.

Τέλος, προτού αρχίσει η διαδικασία σχεδιασμού του διαγράμματος, πρέπει να οριστούν τα χρώματα που θα χρησιμοποιηθούν. Ορίζεται πρώτα το λευκό καθώς το πρώτο ορισμένο χρώμα θα είναι και το φόντο της εικόνας. Μετά ορίζεται ένα array που περιέχει 13 διαφορετικά χρώματα, ένα για κάθε δυνατό επεξεργαστή, και το μαύρο χρώμα για τον σχεδιασμό των περιγραμμάτων και κειμένων. Ο κώδικας με τους ορισμούς φαίνεται στο Σχήμα Π 10-3.

```
    $padding = 5;
    $border = 1;
    $textheight = 7;
    $labellen = 30;

    $multiplier = 1;

    while ($multiplier * $dimx < 200 || $multiplier * $dimy < 200) {
        $multiplier++;
    }

    $imagex = ($multiplier * $dimx) + 2 * $padding + 2 * $border + $notationwidthx;
    $imagey = ($multiplier * $dimy) + 5 * $padding + 2 * $border + 2 * $textheight;
    if (3 * $padding + $labellen + 2 * $border + 2 * $textheight + (2 * $padding + 2 * $textheight
+ 2 * $border) * $cores > $imagex) {
        $imagex = 3 * $padding + $labellen + 2 * $border + 2 * $textheight + (2 * $padding + 2 *
$textheight + 2 * $border) * $cores;
    }
    $im = @imagecreate($imagex, $imagey)
            or die("Cannot Initialize new GD image stream");
```

Σχήμα Π 10-2 Κώδικας υπολογισμού κλίμακας αξόνων και δημιουργίας εικόνας διαγράμματος κατανομής των κόμβων του πλέγματος.

```
    $white = imagecolorallocate($im, 255, 255, 255);

    $colorarray = array(
        1 => imagecolorallocate($im, 141, 211, 199),
        2 => imagecolorallocate($im, 255, 255, 179),
        3 => imagecolorallocate($im, 190, 186, 218),
        4 => imagecolorallocate($im, 251, 128, 114),
        5 => imagecolorallocate($im, 128, 177, 211),
        6 => imagecolorallocate($im, 253, 180, 98),
        7 => imagecolorallocate($im, 179, 222, 105),
        8 => imagecolorallocate($im, 252, 205, 229),
        9 => imagecolorallocate($im, 217, 217, 217),
        10 => imagecolorallocate($im, 188, 128, 189),
        11 => imagecolorallocate($im, 204, 235, 197),
        12 => imagecolorallocate($im, 255, 237, 111),
        13 => imagecolorallocate($im, 217, 95, 2),
    );

    $black = imagecolorallocate($im, 0, 0, 0);
```

Σχήμα Π 10-3 Κώδικας ορισμού χρωμάτων διαγράμματος κατανομής των κόμβων του πλέγματος.

Εφόσον έχουν οριστεί τα απαραίτητα resources και έχουν υπολογιστεί οι απαραίτητες διαστάσεις, αρχίζει η διαδικασία σχεδιασμού του διαγράμματος. Πρώτα σχεδιάζεται ένα μαύρο παραλληλόγραμμο, ελαφρός μεγαλύτερο από το μέγεθος του διαγράμματος που θα χρησιμεύσει σαν περίγραμμα του διαγράμματος. Κατόπιν, λαμβάνεται από την βάση ο πίνακας με τις αντιστοιχίσεις κόμβων σε επεξεργαστές. Για κάθε node σχεδιάζεται ένα μικρό παραλληλόγραμμο με το χρώμα που

αντιστοιχεί στην θέση της ήδη ορισμένης array χρωμάτων που είναι ίση με τον αριθμό του επεξεργαστή που χρησιμοποιήθηκε. Το παραλληλόγραμμο σχεδιάζεται στις συντεταγμένες του node στο διάγραμμα και έχει μέγεθος σύμφωνα με την κλίμακα που ορίστηκε προηγουμένως. Αφού ολοκληρωθεί αυτός ο σχεδιασμός για κάθε κόμβο, σχεδιάζονται οι δείκτες υποδιαιρέσεων του κάθε άξονα με βάση τις ορισμένες υποδιαιρέσεις για κάθε άξονα και ολοκληρώνεται έτσι ο σχεδιασμός του διαγράμματος. Ο κώδικας σχεδιασμού φαίνεται στο Σχήμα Π 10-4.

```
    imagefilledrectangle($im, $notationwidthy + $padding, $padding, $notationwidthy + ($multiplier * $dimx) + $padding + 2 * $border, ($multiplier * $dimy) + $padding + 2 * $border, $black);

  $resultcores = mysql_query("SELECT * FROM GMRES_Requests_Nodes WHERE request_id='$id' ");
  $rowcores = mysql_fetch_array($resultcores);
  imagefilledrectangle($im, $notationwidthy + ($multiplier * ($rowcores['dimx'] - 1)) + $padding + $border, ($multiplier * ($dimy - $rowcores['dimy'])) + $padding + $border, $notationwidthy + ($multiplier * ($rowcores['dimx'])) + $padding + $border, ($multiplier * ($dimy - $rowcores['dimy'] + 1)) + $padding + $border, $colorarray[$rowcores['core']]);

  while ($rowcores = mysql_fetch_array($resultcores)) {
    imagefilledrectangle($im, $notationwidthy + ($multiplier * ($rowcores['dimx'] - 1)) + $padding + $border, ($multiplier * ($dimy - $rowcores['dimy'])) + $padding + $border, $notationwidthy + ($multiplier * ($rowcores['dimx'])) + $padding + $border, ($multiplier * ($dimy - $rowcores['dimy'] + 1)) + $padding + $border, $colorarray[$rowcores['core']]);
  }

  imagestring($im, 1, $padding, $padding, $dimy, $black);
  imagestring($im, 1, $padding, ($multiplier * $dimy) + 2 * $padding + 2 * $border, 1, $black);
  imagestring($im, 1, $notationwidthy + ($multiplier * $dimx) + $padding + 2 * $border - $notationwidthx, ($multiplier * $dimy) + 2 * $padding + 2 * $border, $dimx, $black);

  $interval = $notationintervaly;

  while ($interval < $dimy) {
    imagestring($im, 1, $padding, ($multiplier * ($dimy - $interval)) + $padding + $border, $interval, $black);
    $interval+=$notationintervaly;
  }


  $interval = $notationintervalx;
  while ($interval < $dimx) {
    imagestring($im, 1, $notationwidthy + ($multiplier * ($interval)) + $padding + $border - $notationwidthx / 2, ($multiplier * $dimy) + 2 * $padding + 2 * $border, $interval, $black);
    $interval+=$notationintervalx;
  }
```

Σχήμα Π 10-4 Κώδικας σχεδιασμού διαγράμματος κατανομής των κόμβων του πλέγματος.

Τέλος, πρέπει να σχεδιαστεί το υπόμνημα του πίνακα. Πρώτα σχεδιάζεται μια κεφαλίδα του υπομνήματος και μετά σχεδιάζονται για κάθε επεξεργαστή που χρησιμοποιήθηκε:

- ένα μικρό μαύρο παραλληλόγραμμο που χρησιμεύει σαν περίγραμμα,
- ένα ελαφρά μικρότερο παραλληλόγραμμο μέσα στο μαύρο το οποίο έχει το χρώμα που αντιστοιχεί στην θέση της array χρωμάτων που είναι ίση με τον αριθμό του επεξεργαστή,
- και τέλος, διπλά στα δύο παραλληλόγραμμα εκτυπώνεται ο αριθμός του επεξεργαστή.

Αφού σχεδιαστεί και το υπόμνημα, η εικόνα είναι έτοιμη προς προβολή και συνεπώς εκτυπώνεται το περιεχόμενο της εικόνας με την μορφή png στην σελίδα και ελευθερώνεται η μνήμη που είχε καταληφθεί κατά την διαδικασία σχεδιασμού.

Ο κώδικας σχεδιασμού του υπομνήματος και ολοκλήρωσης του διαγράμματος φαίνεται στο Σχήμα Π 10-5.

```
   imagestring($im, 1, 2 * $padding, ($multiplier * $dimy) + 3 * $padding + 2 * $border + $textheight, "Cores:",
$black);

  $i = 0;

   while ($i < $cores) {

      imagefilledrectangle($im, 3 * $padding + $labellen + (2 * $padding + 2 * $textheight + 2 * $border) * $i,
($multiplier * $dimy) + 3 * $padding + 2 * $border + $textheight, 3 * $padding + $labellen + 2 * $border +
$textheight + (2 * $padding + 2 * $textheight + 2 * $border) * $i, ($multiplier * $dimy) + 3 * $padding + 4 *
$border + 2 * $textheight, $black);
      imagefilledrectangle($im, 3 * $padding + $labellen + $border + (2 * $padding + 2 * $textheight + 2 * $border)
* $i, ($multiplier * $dimy) + 3 * $padding + 3 * $border + $textheight, 3 * $padding + $labellen + $border +
$textheight + (2 * $padding + 2 * $textheight + 2 * $border) * $i, ($multiplier * $dimy) + 3 * $padding + 3 *
$border + 2 * $textheight, $colorarray[$i + 1]);


      imagestring($im, 1, 4 * $padding + $labellen + 2 * $border + $textheight + (2 * $padding + 2 * $textheight
+ 2 * $border) * $i, ($multiplier * $dimy) + 3 * $padding + 2 * $border + $textheight, $i + 1, $black);

      $i++;
   }
   imagepng($im);
   imagedestroy($im);
```

Σχήμα Π 10-5 Κώδικας σχεδιασμού υπομνήματος διαγράμματος κατανομής των κόμβων του πλέγματος.

Παράρτημα 11. Κώδικας σχεδιασμού διαγράμματος σύγκλισης

Το διάγραμμα χρησιμοποιεί δύο βοηθητικές συναρτήσεις:

- η powstrformat, που λαμβάνει σαν όρισμα έναν ακέραιο και επιστρέφει μορφοποιημένο για λεζάντα ένα αλφαριθμητικό που αντιστοιχεί στο δέκα εις τον εισαχθέντα ακέραιο για λεζάντα σε λογαριθμική κλίμακα. Ο κώδικας του φαίνεται στο Σχήμα Π 11-1.
- η loadcolor, που λαμβάνει σαν όρισμα μια εικόνα και επιστρέφει ένα array χρωμάτων για τις καμπύλες του διαγράμματος. Ο κώδικας του φαίνεται στο Σχήμα Π 11-2.

```
function powstrformat($powin) {
   $resstring = "";
   if ($powin == 0) {
      $resstring = "1";
   } elseif ($powin == 1) {
      $resstring = "10";
   } elseif ($powin == 2) {
      $resstring = "100";
   } elseif ($powin == 3) {
      $resstring = "1000";
   } elseif ($powin == -1) {
      $resstring = "0.1";
   } elseif ($powin == -2) {
      $resstring = "0.01";
   } else {
      $resstring = "1e" . $powin;
   }
   return str_pad($resstring, 5, " ", STR_PAD_LEFT);
}
```

Σχήμα Π 11-1 Κώδικας συνάρτησης μορφοποίησης λεζάντας λογαριθμικού άξονα.

```
function loadcolors($im) {
   return array(
      1 => imagecolorallocate($im, 255, 0, 0),
      2 => imagecolorallocate($im, 0, 0, 255),
      3 => imagecolorallocate($im, 0, 255, 0),
      4 => imagecolorallocate($im, 200, 200, 0),
      5 => imagecolorallocate($im, 150, 0, 0),
      6 => imagecolorallocate($im, 0, 150, 0),
      7 => imagecolorallocate($im, 0, 0, 150),
      8 => imagecolorallocate($im, 100, 100, 0),
      9 => imagecolorallocate($im, 255, 100, 100),
      10 => imagecolorallocate($im, 100, 255, 100),
      11 => imagecolorallocate($im, 100, 100, 255),
      12 => imagecolorallocate($im, 200, 200, 100)
   );
}
```

Σχήμα Π 11-2 Κώδικας συνάρτησης ορισμού χρωμάτων καμπυλών.

Η διαδικασία σχεδιασμού του διαγράμματος ξεκινάει με τον διαχωρισμό των κλειδιώ,ν όπως αυτά έχουν εισαχθεί στην παράμετρο myID. Με βάση αυτά τα κλειδιά γίνεται λήψη από την βάση των οριακών τιμών της σύγκλισης και του μέγιστου αριθμού επαναλήψεων από τα αποτελέσματα όλων των δοθεισών εκτελέσεων. Με βάση τις οριακές τιμές των σφαλμάτων υπολογίζονται τα όρια και οι υποδιαιρέσεις του άξονα των τιμών σύγκλισης, δεδομένου ότι είναι λογαριθμικός άξονας με υποδιαιρέσεις τις ακέραιες δυνάμεις του 10. Κατόπιν ορίζονται διάφορες προκαθορισμένες μεταβλητές σχεδιασμού, όπως διάκενα και πάχος περιγράμματος. Στην περίπτωση που υπάρχουν άνω της μίας εγγραφές ορίζεται και περισσότερος χώρος για τον σχεδιασμό του υπομνήματος. Με βάση αυτές τις τιμές υπολογίζονται οι διαστάσεις της εικόνας και ορίζεται το αντικείμενο resource αυτής και τα χρώματα που θα χρησιμοποιηθούν. Ορίζεται πρώτα το λευκό καθότι το πρώτο ορισμένο χρώμα θα είναι και το φόντο της εικόνας. Μετά καλείται η loadcolor ορίζοντας το array που περιέχει τα χρώματα για την κάθε καμπύλη και τέλος, το μαύρο και το γκρι χρώμα για τον σχεδιασμό των περιγραμμάτων, του πλέγματος υποδιαιρέσεων και των κειμένων. Ο κώδικας με τους ορισμούς φαίνεται στο Σχήμα Π 11-3.

Αφού αρχικοποιηθεί η εικόνα, ξεκινάει η διαδικασία σχεδιασμού του ιδίου του διαγράμματος. Αρχικά σχεδιάζεται το περίγραμμα, σχεδιάζοντας ένα λευκό παραλληλόγραμμο μέσα σε ένα μεγαλύτερο μαύρο παραλληλόγραμμο και μετά σχεδιάζονται οι λεζάντες των αξόνων. Αφού έχει σχεδιαστεί το αρχικό περίγραμμα σχεδιάζονται οι υποδιαιρέσεις των αξόνων. Για τον λογαριθμικό άξονα αρχικά αναγράφονται μορφοποιημένες με την συνάρτηση powstrformat λεζάντες για την πρώτη μεγαλύτερη και την πρώτη μικρότερη ακέραια δύναμη του δέκα εκτός των μεγίστων ορίων των δεδομένων, όπως έχουν ήδη υπολογιστεί στην κορυφή και την αρχή του άξονα των Y αντίστοιχα. Μετά, για κάθε ακέραια δύναμη του δέκα εντός των ορίων των δεδομένων σχεδιάζεται μία λεζάντα μορφοποιημένη με την συνάρτηση powstrformat καθώς επίσης χαράσσεται μία γκρι ευθεία, στο σωστό ύψος κάθετη στον άξονα των Y.

Κατόπιν, υπολογίζονται οι υποδιαιρέσεις του άξονα των X με την ίδια ακριβώς μεθοδολογία που χρησιμοποιήθηκε στο διάγραμμα κατανομής των κόμβων του πλέγματος και χαράσσεται για κάθε μία από αυτές μία λεζάντα και μία γκρι ευθεία, όπως και για τον άξονα των Y. Ο κώδικας σχεδιασμού του αρχικού κενού αυτού διαγράμματος φαίνεται στο Σχήμα Π 11-4.

```
$id_array = explode(':', $idstr);
   $idwhere = "";
   foreach ($id_array as $id) {
      $idwhere .= "'" . mysql_real_escape_string($id) . "',";
   }
   $idwhere = substr($idwhere, 0, -1);
 $result = mysql_query("SELECT max(diff) max_diff, min(diff) min_diff, max((outer_iter-1)*innerit+inner_iter) maxiter     FROM GMRES_Requests INNER JOIN GMRES_Request_Iterations ON id=request_id WHERE request_id in (" . $idwhere . ") ;");
   $header_row = mysql_fetch_array($result);
   $maxiter = $header_row['maxiter'];
   $maxpow = 20;
   $minpow = -20;
   $horisontalmarkerscount = 1;
   for ($i = -20; $i <= 20; $i++) {
      if (pow(10, $i) <= $header_row['min_diff']) {
         $minpow = $i;
      } elseif (pow(10, $i) >= $header_row['max_diff']) {
         $maxpow = $i;
         break;
      } else {
         $horisontalmarkerscount+=1;
      }
   }
   $imagepadding = 10;
   $text_padding = 3;
   $notation_text_width = 30;
   $text_height = 10;
   $diagram_width = 600;
   $diagram_height = 300;
   $extralegendwidth = 0;
   if (count($id_array) > 1)  $extralegendwidth = $notation_text_width + $text_padding * 4 + $text_height;

   $im = @imagecreate($imagepadding * 2 + $text_height + $text_padding * 2 + $notation_text_width + $diagram_width + 2 + $extralegendwidth, $imagepadding * 2 + $text_height * 2 + $text_padding * 2 + $diagram_height + 2) or die("Cannot Initialize new GD image stream");

   $white = imagecolorallocate($im, 255, 255, 255);
   $colorarray = loadcolors($im);
   $black = imagecolorallocate ($im, 0, 0, 0);
   $grey = imagecolorallocate($im, 200, 200, 200);
```

Σχήμα Π 11-3 Κώδικας αρχικοποίησης και ορισμός εικόνας και χρωμάτων διαγράμματος σύγκλισης.

```
  imagefilledrectangle($im, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width, $imagepadding, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + $diagram_width + 2, $imagepadding + $diagram_height + 2, $black);
  imagefilledrectangle($im, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + 1, $imagepadding + 1, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + $diagram_width + 1, $imagepadding + $diagram_height + 1, $white);

  imagestringup($im, 2, $imagepadding, 180, "Residual", $black);
  imagestring($im, 2, 310, $imagepadding + $diagram_height + 2 + $text_height + $text_padding * 2, "Iteration", $black);

  imagestring($im, 2, $imagepadding + $text_height + $text_padding, $imagepadding - $text_height / 2, powstrformat($maxpow), $black);
  imagestring($im, 2, $imagepadding + $text_height + $text_padding, $imagepadding - $text_height / 2 + $diagram_height, powstrformat($minpow), $black);

  $markerstep = 1;
  while (ceil(1. * $horisontalmarkerscount / $markerstep) <= 10) {
    $markerstep+=1;
  }

  for ($i = $minpow + $markerstep; $i < $maxpow; $i+=$markerstep) {
    $markerplace = $imagepadding + $diagram_height * ($maxpow - $i) / ($horisontalmarkerscount);
    imagestring($im, 2, $imagepadding + $text_height + $text_padding, $markerplace - $text_height / 2, powstrformat($i), $black);
    imageline($im, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + 1, $markerplace, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + $diagram_width + 1, $markerplace, $grey);
  }

  imagestring($im, 2, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width, $imagepadding + $text_padding + $diagram_height, 1, $black);
  imagestring($im, 2, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + $diagram_width - $notation_text_width / 2, $imagepadding + $text_padding + $diagram_height, $maxiter, $black);

  $tickmarkersizes = array(1, 2, 3, 5, 7, 10, 20, 30, 50, 60, 100, 150, 200, 250, 500);
  $markerstep = $tickmarkersizes[1];
  $i = 1;
  while (floor($maxiter / $markerstep) > 5) {
    $i++;
    $markerstep = $tickmarkersizes[$i];
  }
  for ($i = $markerstep; $i < $maxiter - $markerstep / 2; $i+=$markerstep) {
    $markerplace = $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + 1 + $diagram_width * $i / $maxiter;
    imagestring($im, 2, $markerplace, $imagepadding + $text_padding + $diagram_height, $i, $black);
    imageline($im, $markerplace, $imagepadding + 1, $markerplace, $imagepadding + $diagram_height + 1, $grey);
  }
```

Σχήμα Π 11-4 Κώδικας σχεδιασμού διαγράμματος σύγκλισης.

Αφού έχει σχεδιαστεί το κενό διάγραμμα ξεκινάει η διαδικασία σχεδιασμού των καμπυλών του διαγράμματος. Για κάθε μία από τις εκτελέσεις λαμβάνεται το σύνολο των τιμών των σύγκλισης και ορίζεται ένα array με το λογάριθμο του κάθε σφάλματος. Κατόπιν χαράσσονται ευθύγραμμα τμήματα με το χρώμα που αντιστοιχεί την επιλεγμένη εγγραφή μεταξύ των σημείων που αντιστοιχούν στο κάθε ζεύγος γειτονικών τιμών της array αυτής, το σύνολο των οποίων είναι η επιθυμητή καμπύλη. Ο κώδικας αυτής της διαδικασίας φαίνεται στο Σχήμα Π 11-5.

Εφόσον υπάρχουν περισσότερες των δύο εγγραφές στο διάγραμμα σχεδιάζεται και ένα μικρό υπόμνημα με το χρώμα των καμπυλών και την αρίθμηση τους και τελικά εκτυπώνεται η εικόνα όπως φαίνεται στον κώδικα στο Σχήμα Π 11-6.

```
    $colorid = 0;
    foreach ($id_array as $id) {
        $colorid+=1;
        $detailsresult = mysql_query("SELECT  diff  FROM GMRES_Request_Iterations WHERE request_id='" .
mysql_real_escape_string($id) . "' ORDER BY outer_iter,inner_iter  ");
        $details_row = mysql_fetch_array($detailsresult);
        $diagram_values = array();
        $i = 1;
        $diagram_values[$i] = log10($details_row['diff']);
        while ($details_row = mysql_fetch_array($detailsresult)) {
            $i++;
            $diagram_values[$i] = log10($details_row['diff']);
        }
        for ($j = 1; $j < $i; $j++) {
            imageline($im, $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + 1 +
$diagram_width * $j / $maxiter, $imagepadding + $diagram_height * ($maxpow - $diagram_values[$j]) /
($horisontalmarkerscount), $imagepadding + $text_height + $text_padding * 2 + $notation_text_width + 1 +
$diagram_width * ($j + 1) / $maxiter, $imagepadding + $diagram_height * ($maxpow - $diagram_values[$j + 1])
/ ($horisontalmarkerscount), $colorarray[$colorid]);
        }
    }
```

Σχήμα Π 11-5 Κώδικας σχεδιασμού καμπυλών διαγράμματος σύγκλισης.

```
  if (count($id_array) > 1) {
    $legentboxtop = ($imagepadding * 2 + $text_height * 2 + $text_padding * 2 + $diagram_height + 2) / 2 -
(count($id_array) * ($text_height + $text_padding) + $text_padding) / 2 - 1;

    imagefilledrectangle($im, $imagepadding + $text_height + $text_padding * 3 + $notation_text_width +
$diagram_width + 2, $legentboxtop, $imagepadding + $text_height + $text_padding * 6 + $notation_text_width
* 2 + $text_height + $diagram_width + 4, $legentboxtop + count($id_array) * ($text_height + $text_padding) +
$text_padding + 2, $black);
    imagefilledrectangle($im, $imagepadding + $text_height + $text_padding * 3 + $notation_text_width +
$diagram_width + 3, $legentboxtop + 1, $imagepadding + $text_height + $text_padding * 6 +
$notation_text_width * 2 + $text_height + $diagram_width + 3, $legentboxtop + count($id_array) * ($text_height
+ $text_padding) + $text_padding + 1, $white);

    for ($i = 1; $i <= count($id_array); $i++) {
      imageline($im, $imagepadding + $text_height + $text_padding * 4 + $notation_text_width +
$diagram_width + 2, $legentboxtop + $i * $text_padding + ceil(($i - 0.5) * $text_height), imagepadding +
$text_height + $text_padding * 4 + $notation_text_width * 2 + $diagram_width + 2, $legentboxtop + $i *
$text_padding + ceil(($i - 0.5) * $text_height), $colorarray[$i]);
      imagestring($im, 2, $imagepadding + $text_height + $text_padding * 5 + $notation_text_width * 2 +
$diagram_width + 2, $legentboxtop + $i * $text_padding + ($i - 1) * $text_height, $i, $black);
    }
  }

  imagepng($im);
  imagedestroy($im);
```

Σχήμα Π 11-6 Κώδικας σχεδιασμού υπομνήματος σύγκλισης.