ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΙΕΡΑΡΧΙΚΟΙ ΠΙΝΑΚΕΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΟΥΣ ΣΕ ΔΥΚΤΙΑ ΤΡΟΦΟΔΟΣΙΑΣ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΝΟΝΑ ΜΑΡΙΑΣ

Βόλος, Οκτώβριος 2012

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΙΕΡΑΡΧΙΚΟΙ ΠΙΝΑΚΕΣ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΟΥΣ ΣΕ ΔΥΚΤΙΑ ΤΡΟΦΟΔΟΣΙΑΣ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΝΟΝΑ ΜΑΡΙΑΣ

Επιβλέπων : Νέστορας Ευμορφόπουλος
Λέκτορας Καθηγητής Τ.Μ.Η.Υ.Τ.Δ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Φεβρουάριο 2012.

(Υπογραφή)	*(Υπογραφή)*
.....	.....
Νέστορας Ευμορφόπουλος	Γεώργιος Σταμούλης
Λέκτορας Τ.Μ.Η.Υ.Τ.Δ.	Καθηγητής Τ.Μ.Η.Υ.Τ.Δ.

Βόλος, Οκτώβριος 2012

(Υπογραφή)

.....

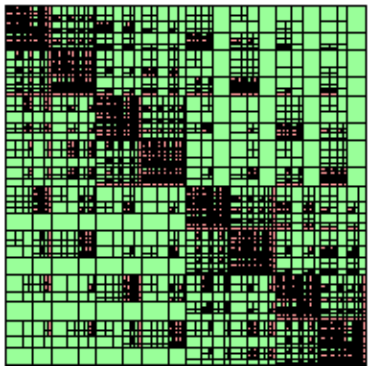
NONA MARIA

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Περίληψη

Σε αυτή τη διπλωματική εργασία παρουσιάζεται η κατασκευή και η μελέτη των ιεραρχικών πινάκων ή Hmatrices. Αρχικά πραγματοποιείται η διαδικασία της ιεραρχικής αναπαράστασης ενός πίνακα, ενώ στη συνέχεια υλοποιούνται αλγόριθμοι για την αριθμητική των ιεραρχικών πινάκων. Τέλος παρουσιάζονται τα αποτελέσματα από την εφαρμογή των υλοποιήσεων σε ένα δίκτυο τροφοδοσίας.

Η κεντρική ιδέα που υπάρχει πίσω από την τεχνική των Hmatrices είναι η διάσπαση ενός δοσμένου πίνακα σε μια ιεραρχία από μπλοκ και η προσπάθεια για την προσέγγιση όσων περισσότερων γίνεται από αυτά με πίνακες χαμηλής τάξης (low rank approximations). Οι πίνακες που επιδέχονται low rank approximation ονομάζονται Rkmatrices, και καθώς αναπαρίστανται σε παραγοντοποιημένη μορφή είναι αυτοί ουσιαστικά που κάνουν αποδοτική την όλη διαδικασία. Στην παρακάτω εικόνα φαίνεται η τελική μορφή που αποκτά ένας πίνακας στην ιεραρχική αναπαράσταση. Τα σκούρα χρωματισμένα μπλοκ αντιστοιχούν σε πίνακες που δεν επιδέχονται low rank αναπαράσταση, ενώ τα υπόλοιπα αντιστοιχούν σε Rkmatrices.



Εικόνα 1.1: Γενική μορφή ενός Hmatrix

Λέξεις Κλειδιά: << Ιεραρχική συσταδοποίηση, Hmatrices, admissibility condition, low rank approximation, rSVD ανάλυση >>

Πίνακας περιεχομένων

Πίνακας Εικόνων

1

Δίκτυα τροφοδοσίας

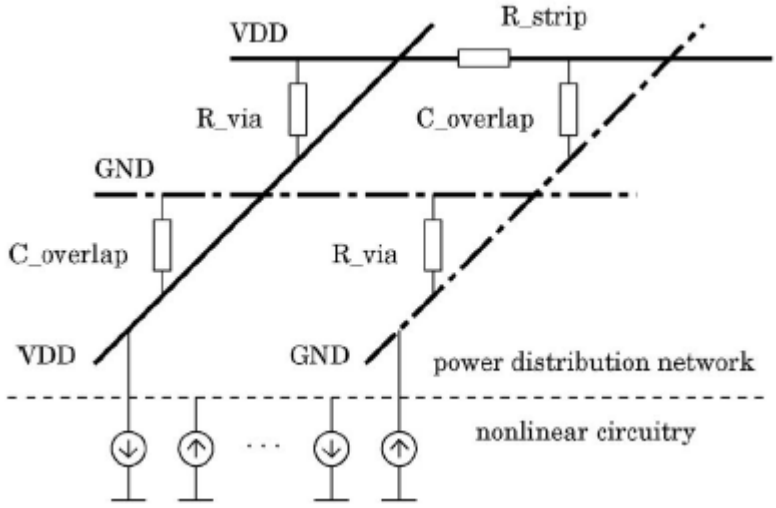
1.1 Προσομοίωση

Ένα αξιόπιστο δίκτυο τροφοδοσίας αποτελεί ίσως το πιο σημαντικό κομμάτι ενός VLSI σχεδίου υψηλής απόδοσης. Ο προσεκτικός σχεδιασμός και ακριβής ανάλυση του δικτύου κατανομής ενέργειας σε ένα τσιπ είναι αυτά που εξασφαλίζουν τη σωστή λειτουργικότητα στην επιθυμητή απόδοση. Η αλγοριθμική προσέγγιση για την εύρεση της βέλτιστης λύσης στο σχεδιασμό ενός δικτύου τροφοδοσίας, βασίζεται στην επίλυση ενός συστήματος της μορφής A v = b και συνεπώς στον υπολογισμό του αντίστροφου του πίνακα A. Ωστόσο, ο ολοένα αυξανόμενος αριθμός των τρανζίστορ στα τσιπ αυξάνει σημαντικά το μέγεθος του δικτύου τροφοδοσίας, καθιστώντας την παραπάνω διαδικασία ιδιαίτερα δύσκολη. Οι ήδη υπάρχουσες τεχνικές για την επίλυση συστημάτων μεγάλων αραιών πινάκων όπως η παραγοντοποίηση Cholesky(direct method) ή η μέθοδος των συζυγών κλίσεων (conjugate gradient) με χρήση preconditioners (iterative method) λειτουργούν αποδοτικά μόνο όταν εφαρμόζονται σε επίπεδο (μη ιεραρχικό) μοντέλο δικτύου ενέργειας. Έτσι τίθεται σημαντικός περιορισμός αναφορικά με το μέγεθος του προβλήματος που μπορούν να λύσουν, ο οποίος εξαρτάται αποκλειστικά από την άμεσα διαθέσιμη για τον υπολογισμό μνήμη. Πολλά από τα δίκτυα τροφοδοσίας των σημερινών μικροεπεξεργαστών είναι τόσο μεγάλα, που δύσκολα λύνονται με το τρέχον τεχνολογικό επίπεδο, καθώς οι διαθέσιμοι υπολογιστικοί πόροι είναι ανεπαρκείς για να τα προσομοιώσουν. Ας πάρουμε για παράδειγμα έναν τυπικό μικροεπεξεργαστή υψηλής απόδοσης σε 0,18 micron design και με

έξι στρώματα μετάλλου. Το δίκτυο τροφοδοσίας του είναι της τάξης των δέκα έως εκατό εκατομμυρίων κόμβων και η προσομοίωση του απαιτεί την λύση ενός γραμμικού συστήματος που ακόμα και η χρήση των πιο ευφυών μεθόδων οδηγεί σε απαγορευτική πολυπλοκότητα ($O(n^2)$ ή $O(n^3)$).

1.2 Ανάλυση ενός τυπικού μοντέλου

Στην παρακάτω εικόνα παρουσιάζεται ένα τρισδιάστατο μοντέλο δικτύου τροφοδοσίας, όπου ο αριθμός των στρωμάτων μετάλλου που αφορά το δίκτυο κατανομής ενέργειας μπορεί να θεωρηθεί αυθαίρετα. Αν και αρκετά απλοποιημένο, αυτό το δίκτυο τροφοδοσίας αποτελεί αντιπροσωπευτικό δείγμα των μοντέλων που χρησιμοποιούνται στα εμπορικά προϊόντα.



Εικόνα 1.1: Τυπικό μοντέλο power grid σε 3D αναπαράσταση

Υποθέτοντας ότι έχουμε n κόμβους, οι εξισώσεις του κυκλώματος μπορούν να γραφτούν στην παρακάτω γενική μορφή: $C\, dv(t)/dt + G\, v(t) = i(t)$, όπου $C, G \in R^{n\times n}$ είναι οι πίνακες που μοντελοποιούν αντίστοιχα τα δυναμικά και στατικά συστατικά στοιχεία του δικτύου, ενώ $v \in R^n$ είναι το διάνυσμα των τάσεων στους κόμβους του πλέγματος και $i \in R^n$ το διάνυσμα των ρευμάτων αντίστοιχα. Οι πίνακες G, C είναι υπερβολικά αραιοί, πράγμα που σημαίνει ότι στην πλειοψηφία των στοιχείων τους απαρτίζονται από μηδενικά. Τώρα προκειμένου να γίνει ανάλυση του δικτύου στο πεδίο του χρόνου, μπορεί να χρησιμοποιηθεί σαν παράδειγμα η μέθοδος του Euler και να γίνει η διακριτοποίηση των κρίσιμων χρονικών διαστημάτων σε βήματα σταθερού μεγέθους h. Ακολούθως προκύπτει: $(C/h+ G)v(t) = i(t) + C/hv(t - h))$, όπου $C/h+ G = A$, ο πίνακας του συστήματος και $b(t) = i(t) + C/hv(t - h))$, είναι το δεξιόστροφο

διάνυσμα σε κάθε χρονικό βήμα. Η ανάλυση του δικτύου τροφοδοσίας, επιβάλλει επομένως τη λύση του συστήματος $Av(t) = b(t)$ σε κάθε διάστημα. Η τετριμμένη λύση του συστήματος είναι η $v = A^{-1}b$. Όμως, παρόλο που ο Α είναι αραιός, δεν συμβαίνει το ίδιο και με τον αντίστροφο του. Ο A^{-1} μπορεί να προκύπτει πυκνός και τότε ο υπολογισμός του δεν είναι εφικτός, τόσο από άποψη κόστους, όσο και από άποψη χρόνου.

1.3 Το Πρόβλημα της αντιστροφής

1.3.1 Απαιτήσεις μνήμης

Οι πίνακες που προκύπτουν από προβλήματα της τάξης μεγέθους που περιγράφεται πιο πάνω, όταν αποθηκεύονται σε πλήρη μορφή, είναι πολύ μεγάλοι για να χωρέσουν στην κεντρική μνήμη του υπολογιστή. Αν ο πίνακας είναι αρκετά μεγάλος ώστε να μην χωράει στη RAM, τότε το λειτουργικό σύστημα του υπολογιστή αναγκάζεται να διατηρεί κάποια τμήματα του πίνακα σε κομμάτια της εικονικής μνήμης, που βρίσκονται στο δίσκο. Αυτό έχει σαν αποτέλεσμα η ανάκτηση των τμημάτων που βρίσκονται στο δίσκο να είναι πολλές φορές πιο αργή, σε σχέση με χρόνο που απαιτείται για την πρόσβαση σε τμήματα που βρίσκονται στην κύρια μνήμη.

Για τον υπολογισμό του αντιστρόφου ενός πίνακα απαιτείται επανειλημμένη πρόσβαση σε διάφορα τμήματα του πίνακα, τα οποία μπορεί να έχουν αποθηκευτεί σε απομακρυσμένα μεταξύ τους κομμάτια της εικονικής μνήμης στο δίσκο. Αυτό έχει σαν αποτέλεσμα ο υπολογισμός του αντιστρόφου ενός μεγάλου πίνακα, που δεν χωράει στη διαθέσιμη φυσική μνήμη, να καθυστερεί.

Τα σύγχρονα λειτουργικά συστήματα για να βελτιώσουν τις επιδόσεις τους κατά την πρόσβαση δεδομένων της εικονικής μνήμης που βρίσκεται στο δίσκο, βασίζονται στην αρχή της τοπικότητας των αναφορών. Σύμφωνα με αυτή την αρχή, όταν ένα πρόγραμμα ζητά πρόσβαση σε μια περιοχή μνήμης, είναι σχεδόν βέβαιο ότι σύντομα θα χρειαστεί δεδομένα που βρίσκονται σε γειτονική περιοχή μνήμης. Έτσι, τα λειτουργικά συστήματα μεταφέρουν μεγάλα μπλοκ δεδομένων από τον δίσκο στην κύρια μνήμη, προκειμένου τα γειτονικά δεδομένα να είναι άμεσα διαθέσιμα σε μια πιθανή αίτηση πρόσβασης σε αυτά. Το μέγεθος του μπλοκ δεδομένων προς μεταφορά, είναι διαφορετικό σε κάθε λειτουργικό σύστημα και ενώ συνήθως επιταχύνει τη λειτουργία των προγραμμάτων, κάποιες φορές μεταφέρει και δεδομένα που δεν θα χρησιμοποιηθούν. Παράλληλα ακολουθείται μια πολιτική εκδίωξης των λιγότερο χρησιμοποιούμενων δεδομένων της κύριας μνήμης, ώστε να δημιουργηθεί χώρος για τα δεδομένα που έρχονται από το δίσκο. Για να είναι όμως η πολιτική εκδίωξης αποτελεσματική, πρέπει τα δεδομένα που απομακρύνει από την φυσική μνήμη να μην ζητηθούν ξανά σύντομα. Η αποτελεσματικότητα των δυο αυτών μηχανισμών είναι κρίσιμη για την ταχύτητα εκτέλεσης προγραμμάτων με απαιτήσεις μνήμης μεγαλύτερες από τη διαθέσιμη φυσική μνήμη. Σε αυτή την κατηγορία προγραμμάτων εμπίπτει και ο υπολογισμός του αντιστρόφου ενός πίνακα, που όμως λόγο της φύσης του αλγορίθμου της αντιστροφής,

που απαιτεί επανειλημμένη πρόσβαση σε διάφορα τμήματα του πίνακα, η αρχή της τοπικότητας των δεδομένων δεν έχει τόσο καλά αποτελέσματα.

1.3.2 Αραιοί πίνακες

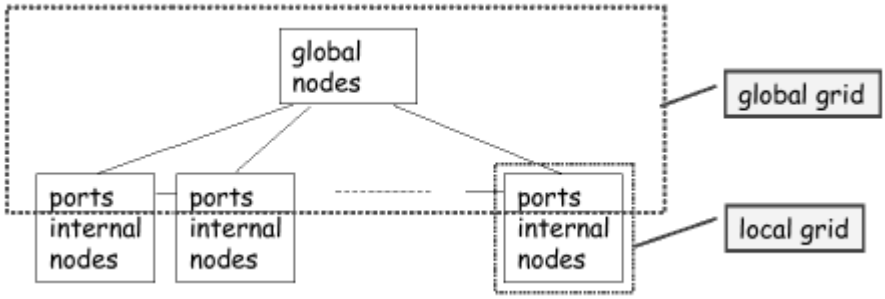
Μιας και οι πίνακες που προκύπτουν από την ανάλυση δικτύων τροφοδοσίας, έχουν μεγάλο ποσοστό μηδενικών, γίνεται εύκολα αντιληπτό ότι εμπίπτουν στην κατηγορία των αραιών πινάκων. Γι' αυτό η χρήση κάποιας από τις δομές αποθήκευσης αραιών πινάκων, θα έκανε αποδοτικότερη την αποθήκευση και προσπέλαση των στοιχείων τους. Αυτές οι δομές δεδομένων, εκμεταλλεύονται την ιδιαίτερη μορφή των αραιών πινάκων και αποθηκεύουν μόνο τα μη μηδενικά στοιχεία και τον αντίστοιχο δείκτη τους, έτσι ώστε να μην γίνεται κατασπατάληση της μνήμης. Έτσι, οι απαιτήσεις σε μνήμη για το πρόβλημα του δικτύου τροφοδοσίας μειώνονται σε μεγάλο βαθμό. Παρόλα αυτά προστίθεται ένα επιπλέον επεξεργαστικό κόστος για τη δημιουργία της δομής και κατά την ανάκτηση δεδομένων από αυτή. Το κόστος αυτό είναι βέβαια πολύ πιο μικρό από αυτό που απαιτείται για την πρόσβαση δεδομένων στο δίσκο, οπότε συνολικά επιταχύνεται η λειτουργία των προγραμμάτων που χρησιμοποιούν τέτοιους πίνακες.

Η ανάκτηση συνεχόμενων περιοχών ενός αραιού πίνακα έχει καλό χρόνο εκτέλεσης. Όταν όμως απαιτούνται κομμάτια διάσπαρτα, όπως συμβαίνει κατά την αντιστροφή ενός πίνακα, η ταχύτητα ανάκτησης από τη δομή δεδομένων είναι χαμηλή. Ενώ η απευθείας κατασκευή ενός αραιού πίνακα από ένα πυκνό είναι γρήγορη, η δυναμική κατασκευή κατά τον υπολογισμό του αντιστρόφου έχει εξαιρετικά μεγάλο κόστος λόγω της ανάγκης μετακίνησης δεδομένων της δομής. Επιπλέον, ο αντίστροφος πίνακας μπορεί να προκύψει πυκνός. Σε αυτή την περίπτωση επανεμφανίζονται τα προβλήματα μνήμης που αναφέρθηκαν πιο πάνω, ενώ η χρήση δομών αραιών πινάκων δεν αποτελεί πλέον συμφέρουσα λύση. Έτσι γίνεται αντιληπτό, ότι για την αντιστροφή το πρόβλημα με τη χρήση πυκνών πινάκων παραμένει.

1.3.3 Ιεραρχικοί πίνακες

Μια δελεαστική λύση θα ήταν η χρήση μιας κοινής μορφής αποθήκευσης τόσο για τον αρχικό πίνακα, όσο και τον αντίστροφο, η ποία δεν θα έχει απαγορευτικό κόστος κατά τη δυναμική κατασκευή του αντιστρόφου. Επιπλέον θα ήταν αποδοτικό αν μπορούσε να συνδυαστεί η γρήγορη ταχύτητα ανάκτησης πυκνών περιοχών του πίνακα και η εξοικονόμηση μνήμης με την συμπιεσμένη αποθήκευση των αραιών περιοχών. Η τεχνική των ιεραρχικών πινάκων, που παρουσιάζεται σε αυτή την εργασία, βασίζεται ακριβώς στην παραπάνω ιδέα: Πίνακες που είναι πυκνοί μπορούν να αναπαρασταθούν με ευφυή τρόπο

στην αραιή μορφή. Αυτό επιτυγχάνεται μέσω της ιεραρχικής ανάλυσης με την αναδρομική πολυεπίπεδη διάσπαση των μπλοκ του πίνακα μέχρι να βρεθούν μπλοκ που μπορούν να αναπαρασταθούν σε παραγοντοποιημένη μορφή (Rkmatrices) ή να αποφασιστεί ότι πρέπει να χρησιμοποιηθούν Fullmatrices για την αναπαράσταση των μπλοκ που δεν επιδέχονται περαιτέρω διάσπαση. Ο σκοπός είναι η επίτευξη του μεγαλύτερου δυνατού αριθμού από Rkmatrices, έτσι ώστε να επιτευχτεί και το μεγαλύτερο κέρδος σε μνήμη.



Εικόνα 1.2: Ιεραρχική ανάλυση Δικτύων Τροφοδοσίας

2

Ιεραρχικοί Πίνακες

2.1 Ιστορικά

Οι ιεραρχικοί πίνακες προτάθηκαν αρχικά από τον Wolfgang Hackbush το 1999 στη δημοσίευση του με τίτλο ‘A sparse matrix arithmetic based on Hmatrices. Part I: Introduction to Hmatrices’. Η εργασία του πραγματοποιήθηκε στο Max- Planck- Institut Mathematik in den Naturwissenschaften, στο Leipzig.

Οι ιεραρχικοί πίνακες αναπτύχθηκαν με σκοπό να μπορούν να εκτελούνται οι διάφορες λειτουργίες μεταξύ πινάκων με πολυπλοκότητα όσο το δυνατόν πιο κοντά στη γραμμική πολυπλοκότητα, προσεγγίζοντας την τάξη του $O(n\log^a(n))$, για μικρή σταθερά α, ξεπερνώντας έτσι το πρόβλημα που παρουσιάζεται με άλλες direct ή iterative μεθόδους (πολυπλοκότητες τάξης $O(n^2)$, $O(n^3)$). Έκτοτε ακολούθησε έρευνα τόσο σε θεωρητικό, όσο και σε πρακτικό κομμάτι, έτσι ώστε σήμερα η τεχνική των Hmatrices να χρησιμοποιείται ευρέως σε μεγάλο πλήθος εφαρμογών. Η χρήση τους μπορεί να είναι είτε άμεση στις εφαρμογές, είτε να χρησιμοποιούνται ως preconditioners για άλλες μεθόδους. Τα κυρία πεδία εφαρμογών τους είναι: Direct Domain Decomposition, Elliptic Partial Differential Equations, FEM/BEM applications, Integral Operators, Integrated Circuits Power Grids applications.

2.2 Ιεραρχική Αναπαράσταση: Η Διαδικασία

Ορισμός Ιεραρχικού πίνακα: Έστω T_{IxI} το block cluster tree του index set I. Το σύνολο των Hmatrices ορίζεται ως:

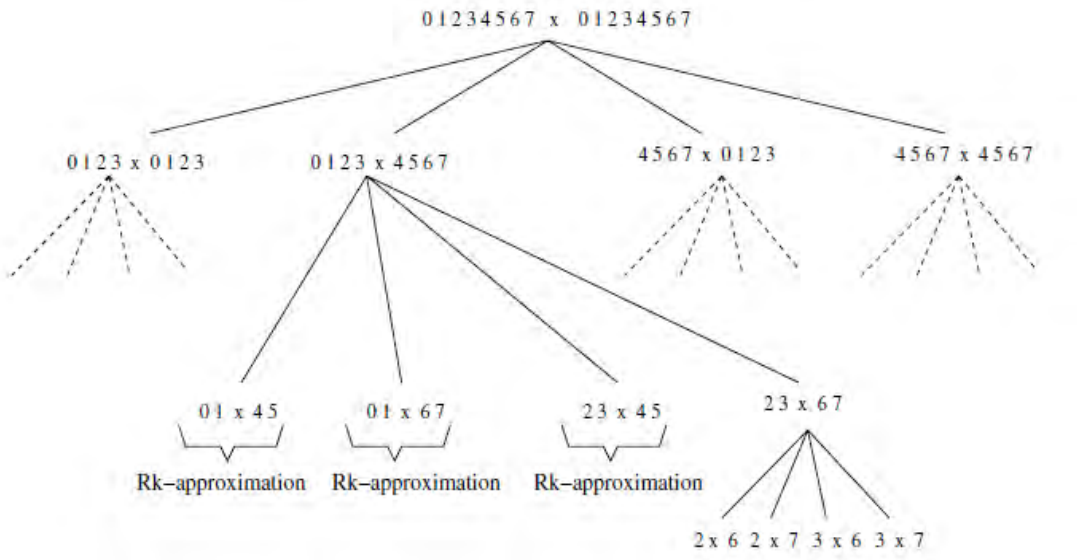
$H(T_{IxI}, k) := \{ M \in R^{I \times I} \mid rank(M|_{t \times s}) \leq k$ *για όλα admissible φύλλα* $t \times s$ *of* $T_{I \times I} \}$.

Ορισμός Ιεραρχικής Αναπαράστασης: *Έστω T_{IxI} το block cluster tree του index set I. Ένας πίνακας* $M \in H(T_{I \times I}, k)$ *λέμε ότι αποθηκεύεται στην ιεραρχική αναπαράσταση αν οι υποπίνακες που αντιστοιχούν σε inadmissible φύλλα αποθηκεύονται σε αναπαράσταση Fullmatrix και αυτοί που αντιστοιχούν σε admissible φύλλα αποθηκεύονται σε αναπαράσταση Rkmatrix.*

2.2.1 Ιεραρχική Συσταδοποίηση

Γενικά ένας ιεραρχικός πίνακας χρησιμοποιεί μια αραιή αναπαράσταση (data sparse representation) για να προσεγγίσει έναν πίνακα που δεν είναι αραιός (dense/fully populated matrix), στον οποίο τα μπλοκ που επιδέχονται low rank approximation αναπαρίστανται στη μορφή Rkmatrix. Η διαδικασία αυτή επιτυγχάνεται μέσω της ιεραρχικής συσταδοποίησης.

Η data sparse αναπαράσταση των ιεραρχικών πινάκων χρησιμοποιεί μια δενδρική δομή, που ονομάζεται block cluster tree $T_{IxI.}$ Το T_{IxI} περιγράφει μια ιεραρχική μπλοκ αναπαράσταση ενός πίνακα $M_{IxI,}$ μέσω του καρτεσιανού γινομένου ενός index set I και αποθηκεύει τα δεδομένα. Η ρίζα του δέντρου είναι IxI και αναπαριστά ολόκληρο τον πίνακα. Ένας εσωτερικός κόμβος sxt ∈ IxI αναπαριστά ένα υπό-μπλοκ του πίνακα, το οποίο θα διασπαστεί περαιτέρω στο επόμενο επίπεδο. Τα φύλλα του T_{IxI} αναπαριστούν τα μικρότερα μπλοκ του πίνακα, αυτά που δεν επιδέχονται άλλη διάσπαση. Τα φύλλα- μπλοκ αποθηκεύονται είτε ως low rank πίνακες (Rkmatrices), είτε ως Fullmatrices. Στην τελική μορφή του block cluster tree όλα τα δεδομένα βρίσκονται αποθηκευμένα στα φύλλα του δέντρου, ενώ οι εσωτερικοί κόμβοι διατηρούν τη δομή του δέντρου και κρατούν στοιχεία για το κάθε υπο- δέντρο, όπως ο βαθμός, οι διαστάσεις και ο τύπος.



Εικόνα 2.1: Παράδειγμα ενός block cluster tree με τρία επίπεδα διάσπασης

2.2.2 Admissibility condition

Τα admissibility conditions αποτελούν κριτήρια παραδοχής, τα οποία αποφασίζουν κατά πόσο ένα υπό-μπλοκ του πίνακα, s x t θα αναπαρασταθεί στην Rkmatrix μορφή κατά τη διάρκεια κατασκευής ενός ιεραρχικού πίνακα. Αν δηλαδή ένα υπό-μπλοκ πληρεί το κριτήριο, τότε χαρακτηρίζεται ως admissible και αναπαρίσταται σε Rkmatrix μορφή. Διαφορετικά χαρακτηρίζεται ως inadmissible και παραμένει στη Fullmatrix μορφή με πιθανή διάσπαση στο επόμενο βήμα.

Τα admissibility conditions ποικίλουν, ανάλογα με τις διαφορετικές τεχνικές κατασκευής των Hmatrices και ανάλογα με τη δομή του πίνακα. Στις κλασικές προσεγγίσεις κατασκευής, τα admissibility conditions καθορίζονται από τις γεωμετρικές πληροφορίες του προβλήματος, όπως π.χ. το domain ή η υποστήριξη (support) του index cluster. Για παράδειγμα, με δοσμένα τα T_I, T_J και s x t ∈ T_{IxJ}, η γενική μορφή ενός admissibility condition μπορεί να οριστεί ως:

s x t admissible ⇔ min {diam(Ω_s), diam(Ω_t)} ≤ μ dist(Ω_s, Ω_t), όπου μ ∈ R είναι μια παράμετρος για τον έλεγχο του αριθμού των μπλοκ από Rkmatrices. Με Ω_s και Ω_t συμβολίζεται η υποστήριξη για τις συστάδες s, t αντίστοιχα. Η διάμετρος μιας συστάδας, diam(s) και η απόσταση μεταξύ δύο συστάδων dist(s, t) ορίζονται χρησιμοποιώντας την Ευκλείδεια νόρμα όπως ακολουθεί:

diam(s) = max { $||x_i-x_j||$: x_i, $x_j \in \Omega_s$ }

dist(s, t) = min { $||x_i-x_j||$: $x_i \in \Omega_s$, $x_j \in \Omega_t$ }

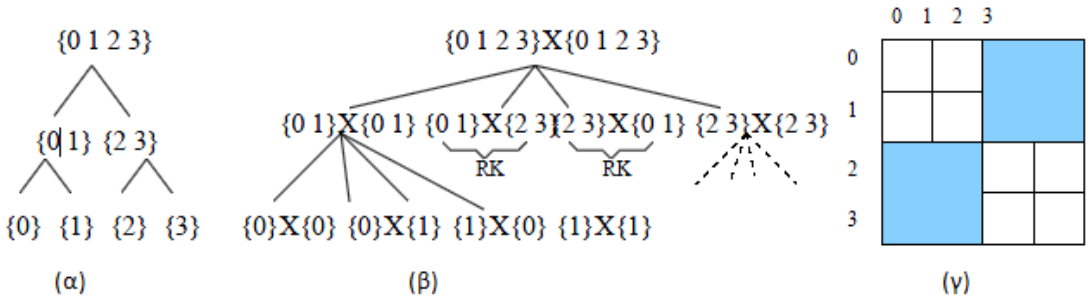
Αντίθετα, το admissibility condition που χρησιμοποιείται για αλγεβρικές προσεγγίσεις κατασκευής Hmatrices, δεν χρησιμοποιεί γεωμετρικές πληροφορίες, άλλα πληροφορίες που

προέρχονται από το γράφημα του πίνακα και μπορεί να διατυπωθεί στην παρακάτω γενική μορφή:

s x t inadmissible ⇔ s, t δεν συνδέονται μεταξύ τους στο γράφημα του πίνακα.

Εδώ, για τις ανάγκες του προβλήματος έχει χρησιμοποιηθεί το γνωστό και ως strong admissibility condition:

s x t admissible ⇔ max {diam(Ω_s), diam(Ω_t)} ≤ dist(Ω_s, Ω_t), όπου το min έχει αντικατασταθεί με max και μ=1.



Εικόνα 2.2: (α) index cluster tree TI, (β) block cluster tree T_{IxI}, (γ) ο αντίστοιχος Hmatrix.

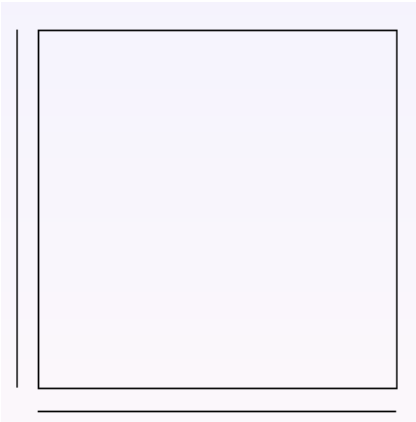
Στο (β) το RK υποδηλώνει τα μπλοκ που ικανοποιούν το admissibility condition, ενώ στο (γ) με μπλε χρώμα σημειώνονται οι Rkmatrices και με άσπρο οι Fullmatrices.

2.2.3 Στάδια κατασκευής

Παρακάτω φαίνεται βήμα προς βήμα η διαδικασία κατασκευής ενός ιεραρχικού πίνακα, ξεκινώντας από ένα δοσμένο πίνακα.

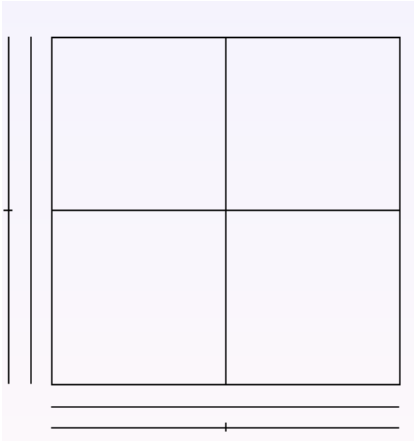
Η διαδικασία ξεκινάει με τον αρχικό πίνακα, τον οποίο αποθηκεύουμε σε μορφή supermatrix που διατηρεί όλα τα δεδομένα του σε ένα Fullmatrix κόμβο . Το domain είναι Ω x Ω και θα χωριστεί σε subdomains σύμφωνα με το admissibility condition max{diam(Ω_s), diam(Ω_t)} ≤ dist(Ω_s, Ω_t).

Εδώ ισχύει t = s και τίποτε δεν είναι admissible



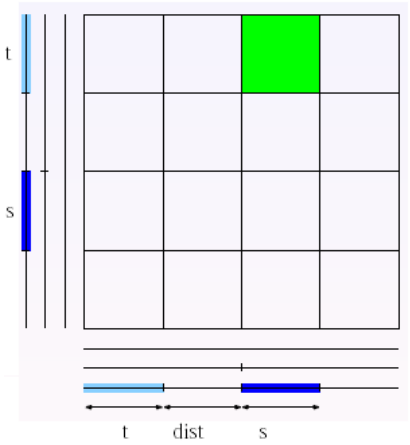
Εικόνα 2.3: Αρχικό domain

Αφού τίποτε δεν είναι admissible, το αρχικό domain θα διασπαστεί σε τέσσερα subdomains. Στο cluster tree που σχηματίζεται, ο αρχικός Supermatrix θα δείχνει σε τέσσερις απογόνους Supermatrices, για κάθε έναν από τους οποίους θα ελεγχθεί και πάλι το admissibility condition. Και πάλι κανένα από τα μπλοκ δεν πληρεί το κριτήριο, οπότε και προχωράμε στη διάσπαση του καθενός σε τέσσερα Supermatrices.



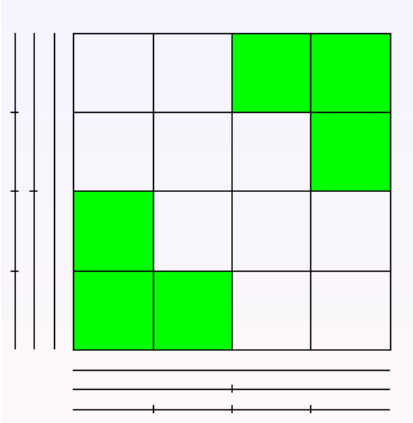
Εικόνα 2.4: Domain μετά από μία διάσπαση

Ελέγχουμε και πάλι το admissibility condition και συναντάμε το πρώτο μπλοκ που είναι admissible. Το μπλοκ αυτό (σημειώνεται με πράσινο χρώμα) δεν μπορεί να διασπαστεί περαιτέρω και αποθηκεύεται σε Rkmatrix μορφή. Το domain αυτού του μπλοκ είναι (0.5, 0,75),(0, 0,25), αν οι διαστάσεις του πίνακα είναι δυνάμεις του 2.



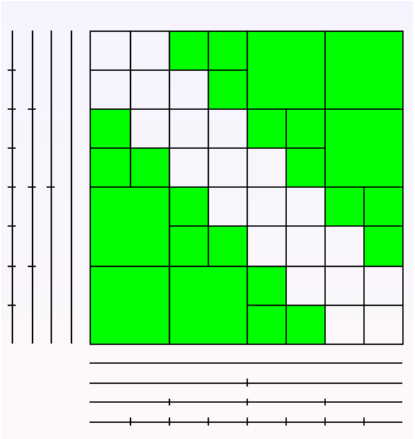
Εικόνα 2.5: admissibility condition ενός μπλοκ

Συνεχίζοντας βρίσκουμε συνολικά σε αυτό το επίπεδο έξι μπλοκ, τα οποία είναι admissible. Όπως αναφέρθηκε και πριν, αυτά τα μπλοκ αποθηκεύονται στην ειδική παραγοντοποιημένη μορφή του Rkmatrix και χαρακτηρίζονται ως φύλλα στο block cluster tree.



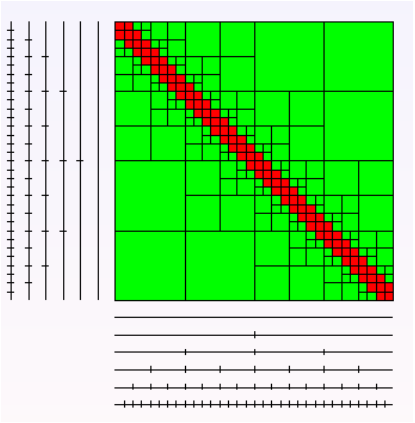
Εικόνα 2.6: Admissible μπλοκ μετά από δυο διασπάσεις

Η διαδικασία συνεχίζεται για τα block που δεν είναι admissible, με κάθε διάσπαση να προσθέτει ένα ακόμα επίπεδο στο block cluster tree. (Τα admissible blocks σημειώνονται με πράσινο χρώμα).



Εικόνα 2.7: Admissible μπλοκ μετά από τρεις διασπάσεις

Η διαδικασία επαναλαμβάνεται μέχρις ότου φτάσουμε στο επιθυμητό μέγεθος subdomain ή το μέγιστο επιθυμητό βάθος του δέντρου.



Εικόνα 2.8: Τελική διασπασμένη μορφή

2.2.4 Αποτελέσματα διάσπασης

Το αρχικό domain Ω x Ω διασπάται σε blocks/συστάδες t x s, οι οποίες οργανώνονται σε cluster tree. Το μέγιστο μέγεθος των μπλοκ που προκύπτουν είναι προκαθορισμένο, μιας και το μέγιστο αποδεκτό μέγεθος domain είναι μία από τις παραμέτρους τερματισμού της διαδικασίας. Έτσι καταφέρνουμε να διασπάσουμε τον πίνακα σε ένα σύνολο από μπλοκ, το μέγεθος των οποίων δεν υπερβαίνει το μέγιστο μέγεθος μνήμης που μπορούμε να διαχειρισθούμε.

Στο κεφάλαιο 3 παρουσιάζεται και η εφαρμογή του low rank approximation σε κάθε ένα από τα admissible block, ώστε να έχουμε το μέγιστο κέρδος σε απαιτήσεις μνήμης.

2.3 Δομικά στοιχεία

2.3.1 Αναπαράσταση Rkmatrix

Τα βασικά δομικά στοιχεία των Hmatrices είναι οι πίνακες χαμηλής τάξης (low rank), οι οποίοι ονομάζονται Rkmatrices. Ένας m x n πίνακας M ονομάζεται Rkmatrix αν για την τάξη του ισχύει: rank(M) $\leq$ k και αναπαρίσταται σε μορφή γινομένου πινάκων ως εξής: $M = A_{mxk}B_{nxk}^{T}$, όπου ο M είναι m x n, ο A είναι m x κ και ο B είναι n x k[1]. Ισχύει δηλαδή ότι κάθε πίνακας τάξης το πολύ k μπορεί να αναπαρασταθεί σαν Rkmatrix και κάθε Rkmatrix έχει τάξη το πολύ k. Μπορούμε να ορίσουμε ένα μέγιστο επιθυμητό k για έναν Rkmatrix, το οποίο θα εισάγει ένα σχετικό σφάλμα στην αναπαράσταση του πίνακα, αν ο βαθμός του πίνακα είναι μεγαλύτερος από το k. Αν όμως το k είναι πολύ μικρότερο από το m και το n, τότε η αναπαράσταση του M στην μορφή Rkmatrix μπορεί να εξοικονομήσει πολύ αποθηκευτικό χώρο και να μειώσει την υπολογιστική πολυπλοκότητα σε πράξεις, όπως πολ/μος Rkmatrix με διάνυσμα και πολ/μος Rkmatrix με Rkmatrix. Η μείωση της πολυπλοκότητας σε πράξεις κατά τον πολ/μο προέρχεται από την μείωση των στοιχείων που λαμβάνουν μέρος στην πράξη και την εφαρμογή τεχνικών δυναμικού προγραμματισμού για τη σειρά του πολ/μου.

Οι Rkmatrices παρουσιάζουν κάποιες ενδιαφέρουσες ιδιότητες, όπως για παράδειγμα ότι η αποθήκευση ενός Rkmatrix απαιτεί μόνο k (m + n) θέσεις μνήμης, σε σχέση με την αποθήκευση σε πλήρη μορφή που απαιτεί m x n.

Η αναπαράσταση ενός πίνακα στην Rkmatrix μορφή αποτελείται από τα παρακάτω δομικά στοιχεία:

- **`k:`** αποτελεί την τάξη του πίνακα και προκύπτει ουσιαστικά από το πλήθος των μη μηδενικών ιδιοτιμών του πίνακα.
- **`kt:`** υποδηλώνει την προσωρινή (τρέχουσα) τάξη του πίνακα. Είναι ίσο με τον αριθμό των ιδιοτιμών που κρατάμε για την αναπαράσταση (όπου **kt** <= **k**) και λειτουργεί σαν άνω όριο για την απαίτηση μνήμης της αναπαράστασης.
- **`rows:`** αναπαριστά τον αριθμό των γραμμών του πίνακα.
- **`cols:`** αναπαριστά τον αριθμό των στηλών του πίνακα.
- **`a:`** διατηρεί τα δεδομένα του πρώτου παράγοντα της παραγοντοποιημένης μορφής του πίνακα.
- **`b:`** διατηρεί τα δεδομένα του δεύτερου παράγοντα της παραγοντοποιημένης μορφής του πίνακα.

[1] Ο πίνακας B αποθηκεύεται κατά στήλες.

2.3.2 Αναπαράσταση Fullmatrix

Όπως αναφέρθηκε και πιο πάνω, τα υπό μπλοκ του πίνακα που δεν πληρούν το admissibility condition αποθηκεύονται σε μορφή Fullmatrix. Αυτό σημαίνει ότι καταλαμβάνουν πολύ περισσότερο χώρο από τους Rkmatrices, καθώς αποθηκεύονται στην πλήρη, ασυμπίεστη μορφή του συστήματος.

Η αναπαράσταση ενός πίνακα στην Fullmatrix μορφή αποτελείται από τα παρακάτω δομικά στοιχεία:

- **`rows:`** αναπαριστά τον αριθμό των γραμμών του πίνακα.
- **`cols:`** αναπαριστά τον αριθμό των στηλών του πίνακα.
- **`e:`** διατηρεί τα δεδομένα του προς αναπαράσταση πίνακα.

2.3.3 Αναπαράσταση Supermatrix

Ένας πίνακας supermatrix αποτελείται από block_rows × block_cols υποπίνακες και κάθε ένας από αυτούς ανήκει σε μια από τις τρεις κατηγορίες, ανάλογα με την πληρότητα του admissibility condition.

Η αναπαράσταση ενός πίνακα στην Supermatrix μορφή αποτελείται από τα παρακάτω δομικά στοιχεία:

- **`rows:`** αναπαριστά τον αριθμό των γραμμών του πίνακα.
- **`cols:`** αναπαριστά τον αριθμό των στηλών του πίνακα.
- **`block_rows:`** αναπαριστά το πλήθος των (Supermatrix) παιδιών του τρέχοντος κόμβου ανά γραμμές.
- **`block_cols:`** αναπαριστά το πλήθος των (Supermatrix) παιδιών του τρέχοντος κόμβου ανά στήλες.
- **`r:`** διατηρεί την αναφορά στο Rkmatrix παιδί του τρέχοντος κόμβου.
- **`f:`** διατηρεί την αναφορά στο Fullmatrix παιδί του τρέχοντος κόμβου.
- **`s:`** διατηρεί τις αναφορές στα Supermatrix παιδιά του τρέχοντος κόμβου.

Όπως αναφέρθηκε αμέσως πιο πάνω κάθε κόμβος/παιδί μπορεί να είναι:

α) Rkmatrix, οπότε και ισχύει ότι r != 0 x 0 και f = 0 x 0, s = 0 x 0. Οπότε στο πεδίο r αποθηκεύεται η Rkmatrix αναπαράσταση του πίνακα.

β) Fullmatrix, οπότε ισχύει f != 0 x 0 και r = 0 x 0, s = 0 x 0. Οπότε στο πεδίο f αποθηκεύεται η Fullmatrix αναπαράσταση του πίνακα.

γ) Supermatrix, οπότε s != 0 x 0 και r = 0 x 0, f = 0 x 0. Οπότε το s περιλαμβάνει τους δείκτες στους μπλοκ υποπίνακες, καθένας από τους οποίους μπορεί να είναι Rkmatrix, Fullmatrix, ή Supermatrix.

3

Αριθμητική Ιεραρχικών Πινάκων

Η αριθμητική των ιεραρχικών πινάκων που παρουσιάζεται σε αυτό το κεφάλαιο περιλαμβάνει πρόσθεση, πολλαπλασιασμό και αντιστροφή. Αυτές οι πράξεις, καθώς επίσης και η SVD ανάλυση είναι απαραίτητες για την επίλυση του προβλήματος δικτύων τροφοδοσίας που εξετάζεται.

3.1 rSVD Ανάλυση

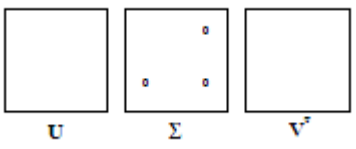
Η SVD ανάλυση αποτελεί τη βάση για την αποδοτική υλοποίηση όλων των υπολοίπων αριθμητικών πράξεων. Καθώς δεν είναι αποδοτική η μετατροπή ενός Rkmatrix σε πλήρη μορφή, ώστε να γίνει SVD ανάλυση, είναι αναγκαία μια μέθοδος που να μπορεί να εφαρμοσθεί σε παραγοντοποιημένους πίνακες. Έτσι για την ανάλυση ενός Rkmatrix χρησιμοποιείται η rSVD ανάλυση.

Αρχικά δίνεται ο ορισμός της SVD ανάλυσης και στη συνέχεια ο ορισμός της rSVD προκειμένου να γίνει πιο κατανοητή η διαφοροποίηση τους.

SVD: Έστω πίνακας $M \in R(k, n, m)$.

Μια SVD ανάλυση (singular value decomposition) του πίνακα M, είναι μια παραγοντοποίηση της μορφής:

$M = U\Sigma V^T$, με τριγωνικούς πίνακες $U \in R^{n\times n}$, $V \in R^{m\times m}$ και έναν διαγώνιο πίνακα $\Sigma \in R^{n\times m}$, όπου τα διαγώνια



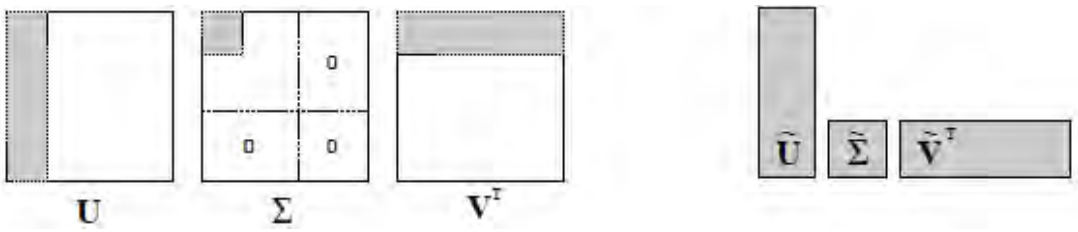
Εικόνα 3.1: SVD ανάλυση

στοιχεία του πίνακα είναι:

$\Sigma_{11} \geq \Sigma_{22} \geq \ldots \geq \Sigma_{kk} \geq \Sigma_{k+1,k+1} = \ldots = \Sigma_{\min\{n,m\},\min\{n,m\}} = 0$ και αποτελούν τις ιδιοτιμές του πίνακα M.

Ο Σ επιπλέον, λόγω της ιδιαίτερης μορφής του, μπορεί να αποθηκευτεί αποδοτικά, κρατώντας μόνο ένα διάνυσμα με τις μη μηδενικές ιδιοτιμές του. Αν ο βαθμός του M είναι k, αυτό σημαίνει ότι απαιτείται χώρος για k <= min(m, n) < m x n στοιχεία.

rSVD: Μια rSVD ανάλυση (reduced singular value decomposition) του πίνακα M, είναι μια παραγοντοποίηση της μορφής: $M = U\Sigma V^T$ με ορθοκανονικούς πίνακες $U \in R^{n\times\kappa}$, $V \in R^{m\times\kappa}$ και έναν διαγώνιο πίνακα $\Sigma \in R^{k\times k}$, όπου τα διαγώνια στοιχεία του είναι : $\Sigma_{11} \geq \Sigma_{22} \geq \ldots \geq \Sigma_{kk} > 0$.



Εικόνα 3.2: rSVD ανάλυση

Για να γίνει αποδοτικά η rSVD ανάλυση σε έναν πίνακα Rkmatrix χρησιμοποιήθηκε ο Αλγόριθμος 1. Με αυτό τον τρόπο μπορεί να γίνει SVD ανάλυση σε έναν Rkmatrix, χωρίς να χρειάζεται να υπολογιστεί πρώτα ο αρχικός πίνακας της RK αναπαράστασης (εκτελώντας τον πολλαπλασιασμό $A*B^T$). Για να κάνουμε προσέγγιση χαμηλής τάξης k, αρκεί από τους πίνακες v_a και v_b να κρατήσουμε μόνο τις k πρώτες στήλες.

```
function [u s v] = rsvd(RK)
      [u_a, v_a] = qr(RK.a);
      [u_b, v_b] = qr(RK.b);
      usv = v_a * v_b';

      [u_s, s, v_s] =  svd(usv);
      u = u_a * u_s;
      v = u_b * v_s;
end
```

Αλγόριθμος 1: rSVD ανάλυση σε RKmatrix

Μιας και οι ιδιοτιμές της SVD ανάλυσης φθίνουν πολύ γρήγορα, η συμβολή των μικρότερων ιδιοτιμών στο σχηματισμό του αποτελέσματος του πολλαπλασιασμού $U\Sigma V^T$ είναι αμελητέα. Όταν υπάρχουν ιδιοτιμές που είναι κατά πολύ μικρότερες από τη πρώτη ιδιοτιμή (π.χ. χίλιες

φορές μικρότερες), τότε η αντικατάσταση τους με μηδενικά προκαλεί μικρό σχετικό σφάλμα στην ακρίβεια του πίνακα. Για κάθε ιδιοτιμή που μπορεί να θεωρηθεί μηδενική, δεν χρειάζεται πλέον η αποθήκευση των αντίστοιχων γραμμών και στηλών, από τους πίνακες U και V, που πολλαπλασιάζονται με αυτή. Οπότε, για κάθε αυθαίρετα μικρή, ώστε να θεωρείται μηδενική, ιδιοτιμή μπορούμε να αποθηκεύσουμε (m + n + 1) λιγότερα στοιχεία από την SVD ανάλυση του M.

Η παραπάνω παρατήρηση έχει εφαρμογή σε μεθόδους συμπίεσης εικόνας και προοδευτικής αποστολής των δεδομένων τους.

3.2 Πρόσθεση Ιεραρχικών πινάκων

Η πρόσθεση ιεραρχικών πινάκων πραγματοποιείται προσθέτοντας τα αντίστοιχα υπό-μπλοκ δύο ιεραρχικών πινάκων, που έχουν την ίδια block cluster tree δομή. Ο πίνακας του αποτελέσματος είναι επίσης ιεραρχικός με την ίδια δενδρική δομή. Αν τα υπό-μπλοκ που προστίθενται είναι τύπου Fullmatrix, τότε χρησιμοποιείται η ακριβής πρόσθεση πινάκων, όπως τη γνωρίζουμε, και το αποτέλεσμα είναι ένα μπλοκ τύπου Fullmatrix. Για την περίπτωση που τα μπλοκ είναι τύπου Rkmatrix ακολουθείται λίγο διαφορετική διαδικασία.

Το άθροισμα δυο Rkmatrices θα δώσει έναν πίνακα Rkmatrix, του οποίου η (μέγιστη) τάξη θα ισούται με το άθροισμα των τάξεων των πινάκων που προστίθενται. Έστω ότι προστίθενται οι $M_1 = AB^T$, $M_2 = CD^T$,τάξης k. Το άθροισμα θα είναι της μορφής $M = M_1 + M_2 = AB^T + CD^T = [A, C]\,[B, D]^T$. Από τον προηγούμενο τύπο γίνεται κατανοητό, ότι το άθροισμα μπορεί να υπολογισθεί απλά σαν τη δημιουργία δύο μπλοκ-πινάκων A’ = [A, C] και B’ = [B, D] και χωρίς καμία αριθμητική πράξη. Με αυτό τον τρόπο, παρότι το αποτέλεσμα θα έχει βαθμό το πολύ 2k, η απαιτούμενη μνήμη θα είναι πάντα επαρκής για την αποθήκευση πινάκων βαθμού 2k. Προκειμένου να μειώσουμε τις απαιτήσεις σε μνήμη για την αποθήκευση των πινάκων A και B και να κάνουμε την παραγοντοποιημένη αναπαράσταση πιο αποδοτική χωρίς να μειωθεί η ακρίβεια της, χρησιμοποιείται η rSVD ανάλυση.

Στην υλοποίηση που δίνεται υπάρχει δυνατότητα είτε να πραγματοποιηθεί η πρόσθεση για τάξη προσαρμοζόμενη ανάλογα με το αποτέλεσμα του αθροίσματος, είτε για συγκεκριμένη εκ των προτέρων τάξη πίνακα (fixed rank).

- Στην πρώτη περίπτωση, αφού υπολογιστεί το άθροισμα των δυο Rkmatrices, τότε δεσμεύεται κατάλληλος χώρος για την ακριβή αποθήκευση του αποτελέσματος.
- Στη δεύτερη περίπτωση, ορίζεται από πριν η επιθυμητή τάξη του αποτελέσματος και αυτό μας δίνει τη δυνατότητα να γνωρίζουμε από πριν το μέγεθος μνήμης που

απαιτείται. Έτσι το αποτέλεσμα μπορεί να αποθηκευτεί ή σε ένα από τα δυο ορίσματα της πρόσθεσης, πράγμα που δεν απαιτεί επαναδεσμεύσεις μνήμης, ή σε μια τρίτη προδεσμευμένη μεταβλητή.

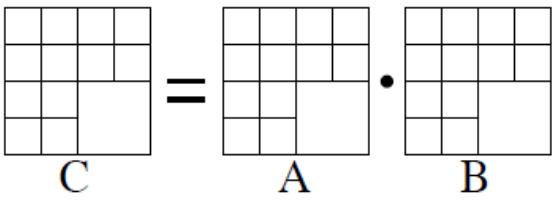
Η δεύτερη περίπτωση αποτελεί καλύτερη επιλογή για την επίλυση της κατηγορίας προβλημάτων που εξετάζεται, αφού μιας και δεν απαιτεί επαναδεσμεύσεις μνήμης είναι γρήγορη και παράλληλα διατηρεί το από πριν ορισμένο επίπεδο ακρίβειας.

Ο αλγόριθμος ο οποίος χρησιμοποιείται στην πρόσθεση πινάκων Rkmatrix της μορφής $M_1 = A_1 B_1^T$, $M_2 = A_2 B_2^T$ με επιθυμητό βαθμό αποτελέσματος k, συνοψίζεται στα παρακάτω βήματα:

1. Δημιουργία RK πίνακα με A το σύνθετο πίνακα $[A_1, A_2]$ και B το σύνθετο πίνακα $[B_1, B_2]$.
2. Υπολογισμός της rSVD ($= U\Sigma V^T$) ανάλυσης του παραπάνω πίνακα.
3. Επιλογή των πρώτων k στηλών του πίνακα U ως τον πίνακα A του αποτελέσματος
4. Επιλογή των πρώτων k γραμμών και στηλών του πίνακα Σ και k στηλών του πίνακα V και αποθήκευση του γινομένου τους ως τον πίνακα B του αποτελέσματος.

3.3 Πολλαπλασιασμός Hmatrix με Hmatrix

Ο πολλαπλασιασμός δυο ιεραρχικών πινάκων θα δώσει σαν αποτέλεσμα έναν ιεραρχικό πίνακα. Ιδιαίτερο ενδιαφέρον, από άποψη απόδοσης, παρουσιάζει η περίπτωση που και οι δυο πίνακες έχουν την ίδια δενδρική δομή.



Εικόνα 3.3: Πολλαπλασιασμός Hmatrix με Hmatrix

Σε αύτη την περίπτωση πραγματοποιείται πολλαπλασιασμός των αντίστοιχων μπλοκ σε γραμμές και στήλες, με τρόπο όμοιο με τον πολλαπλασιασμό αντίστοιχων στοιχείων στον πολλαπλασιασμό πινάκων. Λόγω συμμετρίας, που προκύπτει από το admissibility condition που χρησιμοποιείται, το αποτέλεσμα του πολλαπλασιασμού θα έχει την ίδια δομή με τους τελεστές. Η παραπάνω παρατήρηση γίνεται ευκολότερα αντιληπτή παρατηρώντας το σχήμα της Εικόνα 3.3: Πολλαπλασιασμός Hmatrix.

Ο αλγόριθμος μπορεί να περιγραφεί ως εξής:

Για κάθε γραμμή του A, για κάθε στήλη του B, υπολόγισε το εσωτερικό γινόμενο των αντίστοιχων μπλοκ και αποθήκευσε το αποτέλεσμα στο μπλοκ γραμμή, στήλη του C.

Διακρίνουμε ουσιαστικά τέσσερεις περιπτώσεις για το είδος των πολλαπλασιαζόμενων μπλοκ:

- Αν τα μπλοκ είναι τύπου Supermatrix, τότε ο αλγόριθμος εκτελείται αναδρομικά.
- Αν τα μπλοκ είναι τύπου Fullmatrix, τότε εκτελείται ο απλός αλγόριθμος πολλαπλασιασμού πινάκων.
- Αν τα μπλοκ είναι τύπου Rkmatrix, τότε το αποτέλεσμα βρίσκεται από το γινόμενο $A.A * A.B^T * B.A * B.B^T$. Το γινόμενο αυτό μπορεί να υπολογισθεί με το γρηγορότερο τρόπο, αξιοποιώντας κάποιο αλγόριθμο δυναμικού προγραμματισμού, ώστε να επιλεγεί η ταχύτερη σειρά εκτέλεσης των γινομένων.
- Αν τα μπλοκ είναι διαφορετικού τύπου, τότε δεν μπορούμε να τα πολλαπλασιάσουμε άμεσα, χωρίς να έχει προηγηθεί η από κοινού μετατροπή τους είτε σε Fullmatrix είτε σε Rkmatrix.

Τέλος γίνεται αντιληπτό, ότι κατά τον πολλαπλασιασμό χρησιμοποιούνται κατά κόρον τόσο η πρόσθεση, όσο και η λειτουργία Truncation, καθώς στο γινόμενο πινάκων εμφανίζονται πολλοί όροι αθροισμάτων.

3.4 Αντιστροφή Hmatrix

Κατά την αντιστροφή ενός ιεραρχικού πίνακα, το κύριο πρόβλημα που πρέπει να αντιμετωπιστεί είναι η αντιστροφή των Supermatrices, καθώς τα δεδομένα είναι αποθηκευμένα σε τέσσερα τμήματα.

Το αποτέλεσμα μπορεί να υπολογιστεί αλγεβρικά όπως φαίνεται στην Εικόνα 3.4, όπου $S = M_{22} - M_{21}M_{11}^{-1}M_{12}$, αλλά οι κύριοι παράγοντες για να είναι αποδοτική η αντιστροφή, είναι η επιπλέον ενδιάμεση μνήμη που απαιτείται και το πλήθος των προσβάσεων μνήμης.

$$M^{-1} = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1}M_{12}S^{-1}M_{21}M_{11}^{-1} & -M_{11}^{-1}M_{12}S^{-1} \\ -S^{-1}M_{21}M_{11}^{-1} & S^{-1} \end{bmatrix}, \qquad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

Εικόνα 3.4: Αντιστροφή Supermatrix

Ο Αλγόριθμος 2 που χρησιμοποιήθηκε για την αντιστροφή Hmatrix, αποτελεί μια βελτιωμένη εκδοχή του αλγορίθμου που παρουσιάζεται στο paper Hierarchical Matrices[1]. Η βελτιωμένη εκδοχή δεν απαιτεί την χρήση καμίας ενδιάμεσης/προσωρινής μεταβλητής για τον

υπολογισμό του αντιστρόφου πίνακα, πράγμα που τον κάνει γρηγορότερο, μιας και απαιτεί λιγότερη μνήμη και εκτελεί λιγότερες προσβάσεις σε διαφορετικές θέσεις μνήμης.

```
Function HInvert(var M)
    If m has Supermatrix children
        M11 = HInvert(M11)
        M12 = - M11 * M12
        M22 = M22 + M21 * M12
        M21 = - M21 * M11
        M22 = HInvert(M22)
        M12 = M12 * M22
        M11 = M11 + M12 * M21
        M21 = M22 * M21
    Else
        Invert(M)
    End
End Function
```

Αλγόριθμος 2: Αντιστροφή Hmatrix

Για την αντιστροφή των Rkmatrices και των Fullmatrices, χρησιμοποιήθηκε η κλασσική μέθοδος αντιστροφής πινάκων.

4

Περιβάλλον Ανάπτυξης και Αρχιτεκτονική Υλοποίησης

4.1 Περιβάλλον ανάπτυξης

Το περιβάλλον εργασίας που χρησιμοποιήθηκε για όλες τις δοκιμές είναι το MATLAB (έκδοση 32-bit 7.10.0(P2010a).)

4.1.1 MATLAB

Το όνομα MATLAB προέρχεται από τα αρχικά των λέξεων matrix laboratory και γράφτηκε με αρχικό σκοπό την εύκολη πρόσβαση σε λογισμικό πινάκων. Μέσα σε διάσημα λίγων ετών το MATLAB εξελίχθηκε, χάρη στη συμβολή πολλών χρηστών και σήμερα χρησιμοποιείται ευρέως σε επιστημονικές πανεπιστημιακές εφαρμογές, αλλά και ως βιομηχανικό εργαλείο στα πεδία της έρευνας, της εφαρμογής και της ανάλυσης.

Σήμερα αποτελεί ουσιαστικά ένα διαδραστικό σύστημα, ιδανικό για υπολογισμούς που περιέχουν δεδομένα βασισμένα σε μορφή πίνακα, καθώς αυτό είναι το βασικό του στοιχείο δεδομένων. Πολλά τεχνικά υπολογιστικά προβλήματα, ιδίως αυτά που εμπεριέχουν εφαρμογές πινάκων και διανυσμάτων, μπορούν να λυθούν σε πολύ λίγο χρόνο συγκριτικά με

αυτόν που θα χρειαζόταν για ένα πρόγραμμα σε κάποια μη διαδραστική γλώσσα όπως η C ή η Fortran.

4.1.2 Κίνητρα επιλογής

Η χρήση του MATLAB φαίνεται ιδανική για τις ανάγκες του προβλήματος που εξετάζεται. Η γλώσσα που χρησιμοποιεί, αποτελεί μια υψηλού επιπέδου matrix/array γλώσσα, πιο κατανοητή και φιλική προς το χρήστη σε σχέση με γλώσσες χαμηλότερου επιπέδου (π.χ. C). Το MATLAB παρέχει μεγάλο πλήθος λειτουργιών, όπως συναρτήσεις, απλές αλλά και πιο σύνθετες (π.χ. matrix inverse, matrix eigenvalues), δηλώσεις ελέγχου ροής, βιβλιοθήκες, δομές δεδομένων, γραφικό περιβάλλον, αλλά και χαρακτηριστικά αντικειμενοστραφούς προγραμματισμού. Όλες οι παραπάνω αυτοματοποιημένες λειτουργιές επιτρέπουν στο χρήστη την επαναχρησιμοποίηση τετριμμένων διαδικασιών, όπως η πρόσθεση και ο πολλαπλασιασμός πινάκων, όμως παρέχουν και την δυνατότητα για υλοποίηση αλγορίθμων και δημιουργία σύνθετων μεγάλων προγραμμάτων. Στην εργασία αυτή, η διαχείριση μεγάλων αραιών πινάκων και οι αλγόριθμοι που υλοποιούν τις μεταξύ τους πράξεις εκμεταλλεύονται όλα τα παραπάνω πλεονεκτήματα.

4.2 Αρχιτεκτονική υλοποίησης

Ο κώδικας της εργασίας δομήθηκε με τη μορφή επαναχρησιμοποιήσιμης βιβλιοθήκης, ακολουθώντας το πνεύμα και τις συμβάσεις των ήδη ενσωματωμένων στο Matlab βιβλιοθηκών.

Το πρώτο πράγμα που έπρεπε να αξιολογηθεί για την δημιουργία της βιβλιοθήκης, ήταν η επιλογή των δομών δεδομένων που θα χρησιμοποιούνταν. Το paper Hierarchical Matrices[1], το οποίο ήταν και η βασική πηγή αλγορίθμων, σε αρκετά παραδείγματα κώδικα χρησιμοποιεί τη δομή των structs, στη γλώσσα προγραμματισμού C. Έτσι η χρήση μιας αντίστοιχης δομής δεδομένων στο Matlab θα διευκόλυνε τη διαδικασία ανάπτυξης, λόγω της πιο άμεσης αντιστοίχησης. Το γεγονός όμως ότι στο Matlab κάθε συνάρτηση είθισται να υλοποιείται σε διαφορετικό αρχείο, θα είχε σαν αποτέλεσμα ο κώδικας της βιβλιοθήκης να είναι χαώδης.

Κύριες υποψήφιες επιλογές αποτελούσαν η δομή struct και η δομή των κλάσεων (class).

- Η δομή struct στο Matlab ουσιαστικά αποτελεί μια δυναμική ανώνυμη συλλογή από μεταβλητές. Έτσι δεν χρειάζεται η δήλωση ενός τύπου struct με συγκεκριμένα πεδία και έπειτα ο ορισμός μεταβλητών αυτού του τύπου. Αρκεί να ανατεθεί σε μια μεταβλητή μια συλλογή από ζεύγη κλειδιών-τιμών για να αρχικοποιηθεί ένα struct. Ταυτόχρονα δίνεται η δυνατότητα να προστεθούν δυναμικά καινούργια πεδία στη συλλογή, ακόμα και μετά την αρχικοποίηση της. Το παραπάνω χαρακτηριστικό θα μπορούσε να μειώσει την κατανάλωση μνήμης για την αποθήκευση κόμβων Supermatrix μιας και θα χρειαζόταν να δεσμευθεί χώρος στο struct μόνο για τον τύπο απογόνου που είναι απαραίτητος σε κάθε περίπτωση (Fullmatrix, Rkmatrix ή πίνακα από Supermatrix).
- Η δομή των κλάσεων από την άλλη, είναι πολύ κοντά στον τρόπο αντικειμενοστραφούς προγραμματισμού όπως συναντάται στις σύγχρονες γλώσσες προγραμματισμού. Δηλώνοντας μια κλάση, ουσιαστικά δημιουργείται ένας επώνυμος τύπος δεδομένων που μπορεί να περιλαμβάνει ιδιότητες, κατασκευαστές και όλες τις μεθόδους που σχετίζονται με χρήση του.

Αρχικά λοιπόν έγιναν δοκιμές για την επιλογή της κατάλληλης δομής για την υλοποίηση. Μιας και η χρησιμότητα των ιεραρχικών πινάκων, είναι για την επίλυση προβλημάτων με μεγάλες απαιτήσεις μνήμης, κύριο κριτήριο για την επιλογή ήταν το ποια από τις δυο δομές έχει το χαμηλότερο overhead σε μνήμη. Η ανάπτυξη με τη χρήση της δυναμικής και χαμηλότερου επιπέδου δομής των struct φαινόταν αρχικά πιο ταιριαστή, ακόμα και αν δυσχέρανε την ανάπτυξη. Έτσι αναπτύχθηκαν οι απαραίτητες δομές Fullmatrix, Rkmatrix και

Supermatrix τόσο με χρήση struct όσο και με χρήση κλάσεων. Τα αποτελέσματα σε απαιτήσεις μνήμης δεν ήταν τα αναμενόμενα και όπως φαίνεται στην Εικόνα 4.1 η χρήση κλάσεων απαιτεί λιγότερη μνήμη[2], οπότε και επιλέχθηκε σαν καταλληλότερη για την υλοποίηση. Πιθανόν λόγω της δυναμικής φύσης των struct στο Matlab να υπάρχει μια τεχνική προδέσμευσης μνήμης, ώστε να είναι άμεσα διαθέσιμη όταν ένα επιπλέον κλειδί πρέπει να προστεθεί σε ένα struct.

Μια επιπλέον παρατήρηση που έγινε, αφορούσε σε μια πλεονάζουσα κατανάλωση μνήμης και από τις δομές των κλάσεων. Συγκεκριμένα κάθε αντικείμενο κλάσης φαίνεται να διατηρεί 48 επιπλέον bytes (6 πεδία τύπου double) πιθανόν για κάποιες εσωτερικές λειτουργίες του Matlab.

Name	Size	Bytes	Class	Attributes
A	64x64	3716	double	sparse
Afull	64x64	32768	double	
full_matrix	1x1	3788	fullmatrix	
fullstruct	1x1	4104	struct	
rk_matrix	1x1	65624	rkmatrix	
rkstruct	1x1	66312	struct	
su_matrix	1x1	3876	supermatrix	
sustruct	1x1	4368	struct	

Εικόνα 4.1: Σύγκριση struct και κλάσεων

Στην Εικόνα 4.1: Σύγκριση struct και κλάσεων επιπλέον φαίνεται η ενσωματωμένη λειτουργία του Matlab για την αποθήκευση αραιών πινάκων. Ο πίνακας A που χρησιμοποιήθηκε για τις παραπάνω δοκιμές, κρίθηκε αυτόματα από το Matlab σαν αραιός και αποθηκεύθηκε στην ενσωματωμένη δομή αποθήκευσης αραιών πινάκων. Έτσι συγκρίνοντας τον πίνακα A, με την πλήρη αναπαράστασή του (που περιέχεται στη μεταβλητή Afull), παρατηρείται η μείωση των απαιτήσεων σε μνήμη περισσότερο από οκτώ φορές.

Η παραπάνω συμπληρωματική λειτουργία, αναμένεται να προσθέσει ένα μικρό υπολογιστικό κόστος στην ανάκτηση και αποθήκευση στοιχείων των πινάκων που αποθηκεύονται σε αυτή τη μορφή. Ωστόσο η επίδρασή της αναμένεται να είναι θετική, ιδίως σε προβλήματα όπου η διαθέσιμη φυσική μνήμη δεν είναι επαρκής. Η υλοποίηση που έγινε δεν στηρίχτηκε στην παραπάνω λειτουργία και είναι ανεξάρτητη από αυτή. Στις περιπτώσεις μετρήσεων όπου τα αποτελέσματα μπορεί να επηρεάζονταν (κυρίως μετρήσεις ταχύτητας), η λειτουργία απενεργοποιείται πάντα ρητά.

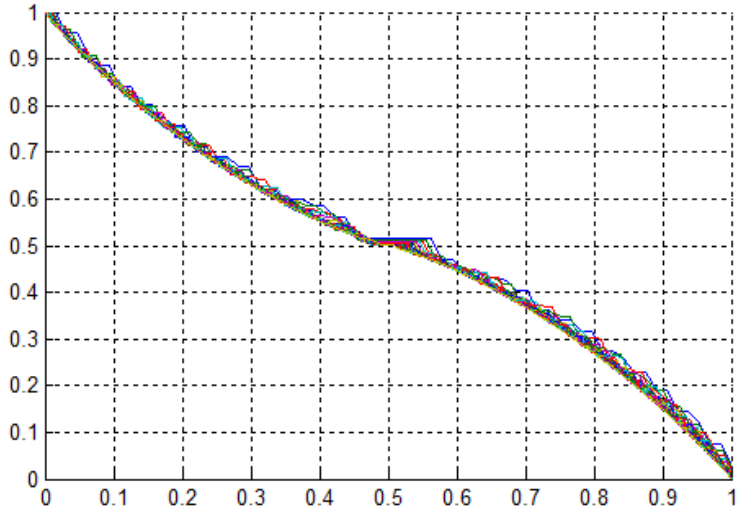
[2] Για την Rkmatrix αναπαράσταση χρησιμοποιήθηκε η svd ανάλυση. Πιο συγκεκριμένα, στον πίνακα a εκχωρείται ο πίνακας u της svd, ενώ στον πίνακα b εκχωρείται το γινόμενο v * s'.

5

Αποτελέσματα

5.1 Αποτελέσματα πινάκων delsq

Στην Εικόνα 5.1 φαίνεται ο ρυθμός που φθίνουν οι ιδιοτιμές των πινάκων που παράγονται από την εντολή delsq(numgrid('S', i)) στο Matlab, η οποία παράγει πίνακες αντίστοιχους με αυτούς των δικτύων τροφοδοσίας. Οι πίνακες αυτοί είναι πλήρους τάξης, οπότε η απευθείας αποθήκευσή τους στη μορφή RKmatrix δεν κρίνεται ιδανική.



Εικόνα 5.1: Φθίνουσες ιδιοτιμές δικτύων τροφοδοσίας.

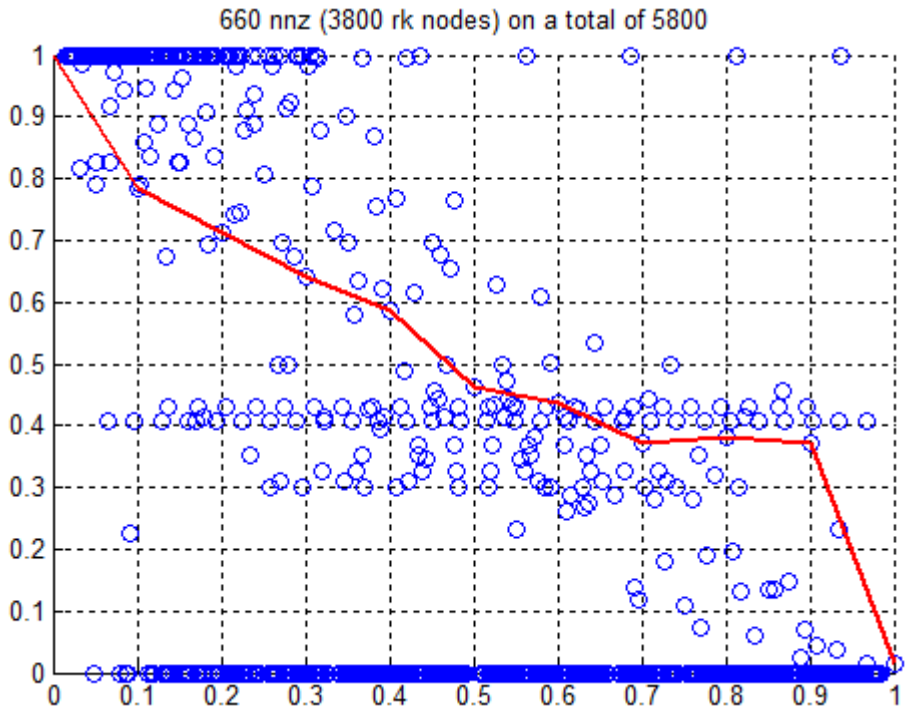
Αυτό συμβαίνει για δύο λόγους:

- Αν κάνουμε ακριβή αποθήκευση ενός πίνακα πλήρους τάξης σε μορφή Rkmatrix, τότε θα χρειαζόταν διπλάσιος αποθηκευτικός χώρος, από ότι για τον αρχικό πίνακα.
- Αν από την άλλη χρησιμοποιηθεί προσέγγιση χαμηλής τάξης, μειώνοντας τον αριθμό από τις αποθηκευόμενες ιδιοτιμές, τότε λόγο του χαμηλού ρυθμού πτώσης των ιδιοτιμών (Εικόνα 5.1), το αποτέλεσμα θα εμπεριείχε μεγάλο σχετικό σφάλμα.

Από την άλλη, αν οι παραπάνω πίνακες δικτύων τροφοδοσίας αναπαρασταθούν σε Hmatrix μορφή, τότε όπως φαίνεται και από την Εικόνα 5.2 οι υποπίνακες που αναπαριστώνται σε Rkmatrix μορφή έχουν μικρό ποσοστό μη μηδενικών ιδιοτιμών. Έτσι οι μηδενικοί Rkmatrix κόμβοι θα καταλαμβάνουν, λόγω της rSVD ανάλυσης ελάχιστο χώρο.

Εικόνα 5.2: Ποσοστό Rkmatrix κόμβων και ποσοστό μη μηδενικών ιδιοτιμών.

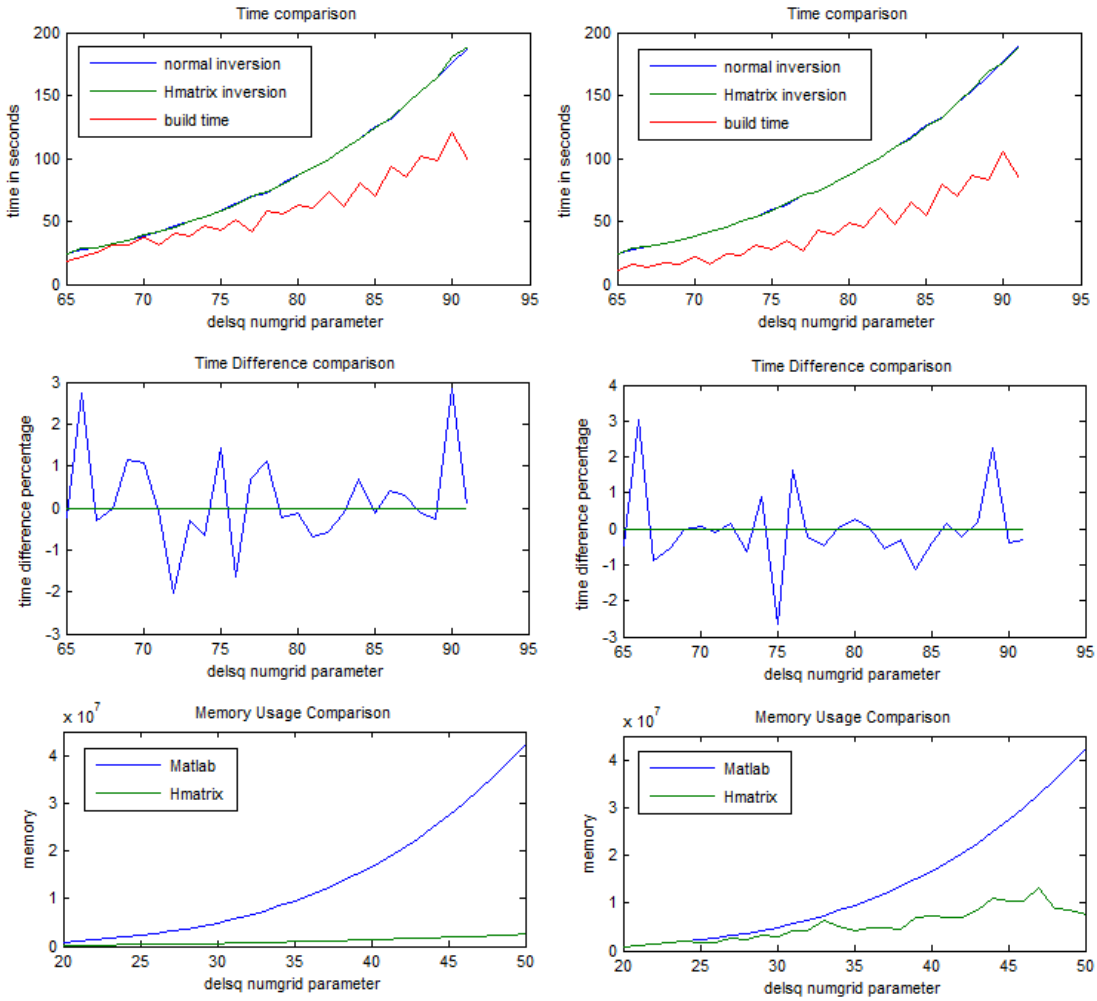
Όπως φαίνεται και στην Εικόνα 5.3 οι ιδιοτιμές των μη μηδενικών Rkmatrix κόμβων (αριστερά στην εικόνα) φθίνουν πολύ πιο γρήγορα από τις ιδιοτιμές του αρχικού πίνακα. Έτσι η αποθήκευση μέσω της rSVD του 90% των ιδιοτιμών, θα επέφερε όφελος όσον αφορά τις απαιτήσεις μνήμης για την αποθήκευση του Rkmatrix κόμβου, με μικρή επίπτωση στην ακρίβεια της αναπαράστασης. Στον τίτλο της ίδιας εικόνας φαίνεται και το ποσοστό των μηδενικών Rkmatrix κόμβων, το οποίο φτάνει το 82%. Για αυτούς τους πίνακες η SVD ανάλυση θα παράγει μηδενικούς πίνακες, οι οποίοι δεν θα επιβαρύνουν την κατανάλωση μνήμης της Hmatrix αναπαράστασης.



Εικόνα 5.3: Φθίνουσες ιδιοτιμές μη μηδενικών Rkmatrix κόμβων.

Οι παραπάνω παρατηρήσεις ενισχύονται από το γεγονός ότι, το ποσοστό των φύλων του Hmatrix δέντρου που αναπαρίσταται σε Rkmatrix μορφή φτάνει το 65%.

Όπως έχει γίνει λόγος σε πολλά σημεία παραπάνω, κατά την διαδικασία της κατασκευής ενός Hmatrix, πραγματοποιούνται αναδρομικές διασπάσεις του κάθε Fullmatrix μπλοκ σε μικρότερα υπό-μπλοκ. Σαν κριτήριο ολοκλήρωσης του αλγορίθμου αναδρομικής διάσπασης, ορίστηκε το ελάχιστο αποδεκτό μέγεθος ενός Fullmatrix block. Έτσι οι κόμβοι Fullmatrix που έχουν μικρότερο μέγεθος από το ορισμένο, δεν διασπόνται περεταίρω καθώς θεωρείται ότι το μέγεθός τους είναι αρκετά μικρό ώστε να είναι εύκολα διαχειρήσιμοι από το τον υπολογιστή.

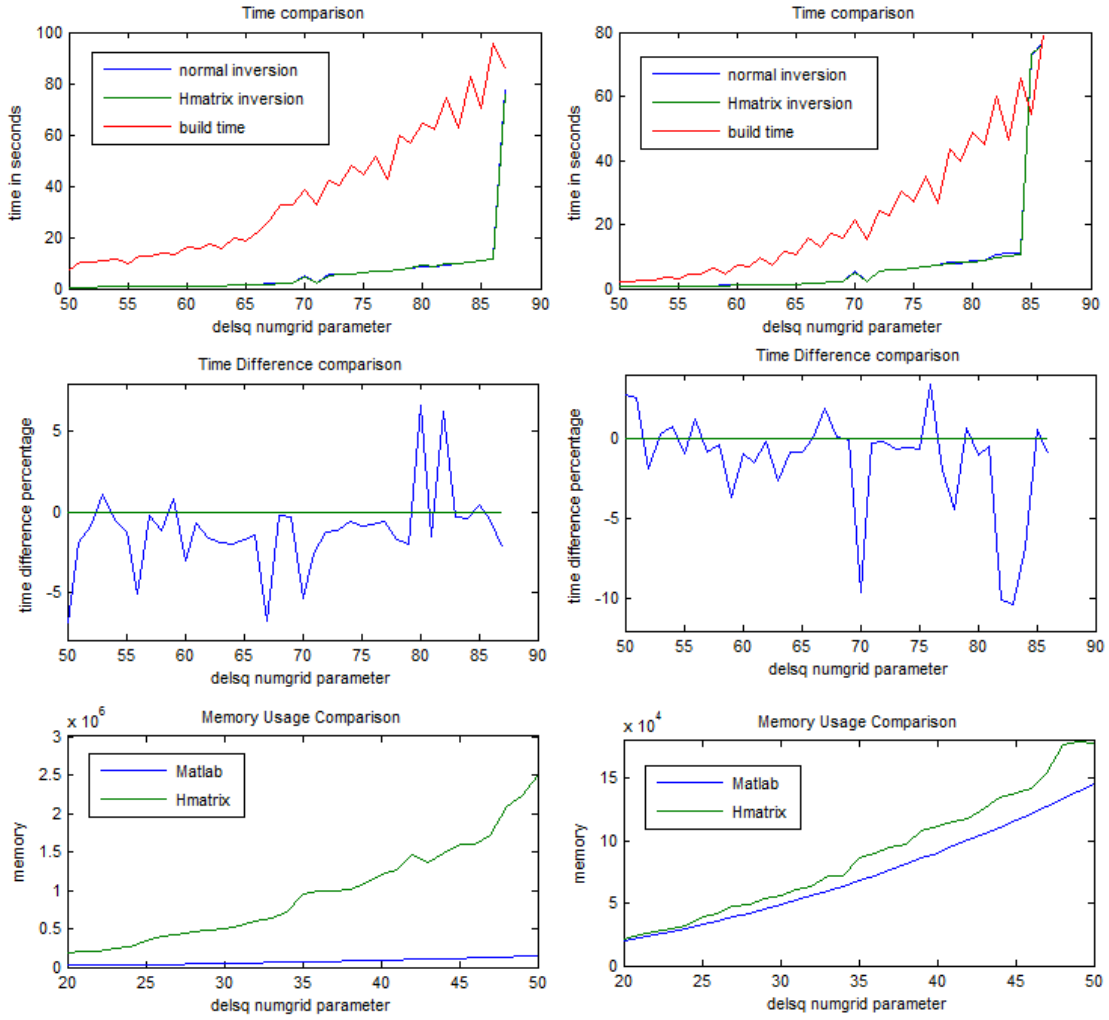


Εικόνα 5.4: Αποτελέσματα αντιστροφής delsq numgrid για πλήρη μορφή.

Κατά την εκτέλεση των δοκιμών, ως ελάχιστο μέγεθος μπλοκ (min block size) επιλέχτηκαν τα μεγέθη 8^2 και 256^2. Ο λόγος της μεγάλης διακύμανσης των δυο επιλεγμένων τιμών, είναι για να εντοπιστεί ο βαθμός που τα αποτελέσματα θα επηρεάζονταν από ελάχιστο μέγεθος μπλοκ.

Καθώς το Matlab εξ ορισμού αποθηκεύει αραιούς πίνακες με την χρήση δομών αποθήκευσης αραιών πινάκων, κάτι που πιθανώς θα επηρέαζε την ταχύτητα αντιστροφής, πραγματοποιήθηκαν δύο σετ δοκιμών έχοντας το παραπάνω χαρακτηριστικό απενεργοποιημένο ή ενεργοποιημένο.

Συνολικά παρουσιάζονται τα αποτελέσματα από τέσσερα σενάρια δοκιμών, για ελάχιστο μέγεθος μπλοκ ίσο με 8^2 και 256^2 και έχοντας την λειτουργία αραιής αναπαράστασης ενεργοποιημένη και απενεργοποιημένη. Οι μετρήσεις έγιναν για τη διαδικασία μετατροπής σε μορφή ιεραρχικού πίνακα και την αντιστροφή, των πινάκων που παράγει η εντολή του Matlab delsq(numgrid('S', i)) για διάφορες τιμές της παραμέτρου i.



Εικόνα 5.5: Αποτελέσματα αντιστροφής delsq numgrid για αραιή μορφή.

Στις Εικόνα 5.4 και Εικόνα 5.5 παρουσιάζονται τα αποτελέσματα των δοκιμών για την αναπαράσταση σε πλήρη και αραιή μορφή αντίστοιχα. Οι εικόνες της αριστερής στήλης αντιστοιχούν σε ελάχιστο μέγεθος μπλοκ 8^2, ενώ της δεξιάς στήλης σε 256^2.

- Στις εικόνες της πρώτης γραμμής και των δυο περιπτώσεων, το buildtime αναφέρεται στο χρόνο που χρειάζεται για να ολοκληρωθεί η εκτέλεση του αλγορίθμου αναδρομικής διάσπασης. Ο χρόνος αυτός επηρεάζεται κατά κύριο λόγο από το min blocksize, το οποίο ανάλογα με το μέγεθος του πίνακα καθορίζει το πλήθος των αναδρομικών κλήσεων της ιεραρχικής διάσπασης που πραγματοποιούνται. Σαν αποτέλεσμα στις εκτελέσεις με μεγαλύτερο min block size το buildtime είναι μικρότερο.
- Όσον αφορά τους χρόνους αντιστροφής παρατηρούμε ότι στην πρώτη περίπτωση ο χρόνος αντιστροφής με χρήση Hmatrix δεν αποκλίνει πολύ από τον χρόνο που απαιτεί η συμβατική αντιστροφή του Matlab. Στην περίπτωση όμως της αραιής

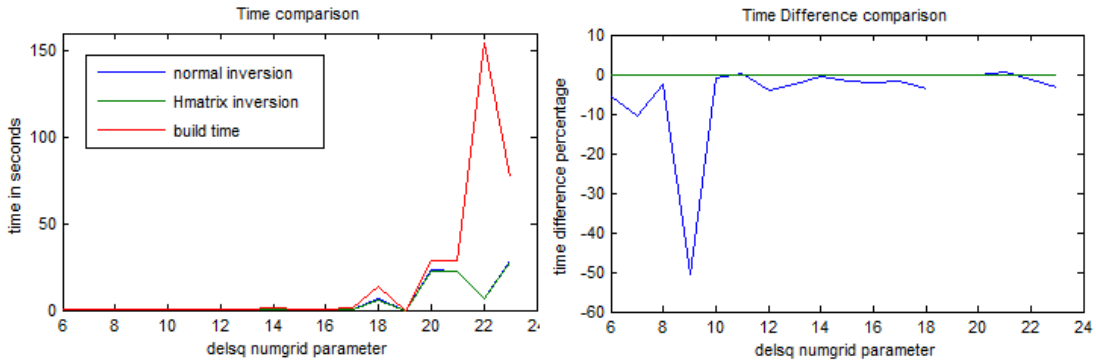
αναπαράστασης, η Hmatrix αντιστροφή φαίνεται να έχει καλύτερες επιδόσεις έως και 10% (για την περίπτωση όπου χρησιμοποιήθηκε ελάχιστο μέγεθος μπλοκ 256^2). Το γεγονός αυτό είναι περισσότερο εμφανές στις γραφικές παραστάσεις της δεύτερης γραμμής, όπου παρουσιάζεται η ποσοστιαία διαφορά των χρόνων αντιστροφής. Συγκεκριμένα, οι μετρήσεις που βρίσκονται πάνω από τον οριζόντιο άξονα υποδηλώνουν πιο γρήγορη αντιστροφή για το Matlab, ενώ οι μετρήσεις που βρίσκονται στο κάτω μέρος για τους Hmatrices. Στην Εικόνα 5.5 βλέπουμε πώς το μεγαλύτερο μέρος των διακυμάνσεων βρίσκεται κάτω από τον οριζόντιο άξονα.

- Στις εικόνες της τρίτης γραμμής γίνεται η σύγκριση της μνήμης που απαιτείται με χρήση Matlab και με χρήση ιεραρχικών πινάκων. Στην πλήρη αναπαράσταση, το Matlab φαίνεται να έχει πολύ περισσότερες απαιτήσεις μνήμης σε σχέση με την υλοποίηση με Hmatrices, καθώς αποθηκεύει και όλα τα μηδενικά στοιχεία του πίνακα. Από την άλλη, στην περίπτωση που χρησιμοποιείται η (εξ' ορισμού ενεργοποιημένη στο Matlab) λειτουργία αραιών δομών αποθήκευσης, τότε το Matlab επιτυγχάνει καλύτερη κατανάλωση μνήμης για την αποθήκευση των πινάκων. Παρατηρείται βέβαια, ότι ενώ για ελάχιστο μέγεθος μπλοκ ίσο με 8^2 η διαφορά είναι πολύ μεγάλη, για ελάχιστο μέγεθος μπλοκ ίσο με 256^2 η διαφορά είναι πολύ μικρότερη. Καθώς το πλήθος των κόμβων που χρειάζονται για την αναπαράσταση στη δενδρική δομή του Hmatrix είναι πολύ μεγαλύτερο στην πρώτη περίπτωση, για τις μεγαλύτερες απαιτήσεις μνήμης του Hmatrix θα μπορούσε να ευθύνεται εν μέρη η «πλεονάζουσα κατανάλωση μνήμης των κλάσεων στο Matlab» που αναφέρθηκε στην αρχή της παραγράφου 4.2.

5.2 Αποτελέσματα πινάκων UFget

Προκειμένου να σχηματιστεί μια πιο ολοκληρωμένη εικόνα, δοκιμάστηκαν και κάποιοι από τους πίνακες της βιβλιοθήκης UFget. Η UFget αποτελεί μια διεπαφή του Matlab και παρέχει μια μεγάλη συλλογή αραιών πινάκων (1877 πίνακες), από το ευρύτερο φάσμα των θετικών επιστημών. Για τις ανάγκες των δοκιμών, ο αλγόριθμος αντιστροφής ιεραρχικών πινάκων εφαρμόστηκε σε όλους τους πίνακες της UFget που αφορούν προσομοίωση δικτύων τροφοδοσίας.

Η σύγκριση έγινε για την αναπαράσταση σε πλήρη μορφή και για μέγιστο μέγεθος μπλοκ 256^2.



Εικόνα 5.6: Αντιστροφή πινάκων UFget για αραιή μορφή και μέγιστο μέγεθος μπλοκ 256^2

Σύμφωνα με τη δεύτερη γραφική παράσταση της Εικόνα 5.6, που αποτελεί τη σύγκριση της ποσοστιαίας διαφοράς χρόνων, φαίνεται ότι η αντιστροφή με ιεραρχικούς πίνακες, είναι σαφώς γρηγορότερη από ότι αυτή του Matlab. Το ποιο θετικό αποτέλεσμα σύμφωνα με τη γραφική παράσταση, αφορά στην αντιστροφή του ένατου πίνακα από το σετ δοκιμών μας, όπου ο χρόνος αντιστροφής με χρήση ιεραρχικών πινάκων ήταν 50% σε σύγκριση με την ενσωματωμένη μέθοδο αντιστροφής του Matlab. Ο πίνακας αυτός έχει αριθμό εγγραφής στη βιβλιοθήκη UFget 1627, ονομάζεται Bai/qh768 και οι διαστάσεις του είναι 768 x 768. Να σημειωθεί ότι για τις τιμές της γραφικής, παράστασης που απουσιάζουν δεν έγιναν υπολογισμοί καθώς δεν επαρκούσε η μνήμη κάνοντας χρήση της αντιστροφής του Matlab. Η παραπάνω παρατήρηση είναι εμφανής και από την πρώτη γραφική παράσταση, όπου μόνο η μπλε γραμμή διακόπτεται.

5.3 Γενικές παρατηρήσεις & συμπεράσματα

Οι ιεραρχικοί πίνακες, φαίνεται να αποτελούν μια πολλά υποσχόμενη τεχνική για την ανάλυση και προσομοίωση μεγάλων δικτύων τροφοδοσίας. Από τα πειραματικά αποτελέσματα φαίνεται πως επιτυγχάνουν ικανοποιητικούς χρόνους και με πολύ χαμηλότερες απαιτήσεις σε μνήμη, ειδικά στην πλήρη αναπαράσταση. Ενεργοποιώντας την αραιή αποθήκευση του Matlab, οι απαιτήσεις μνήμης των ιεραρχικών πινάκων είναι οριακά μεγαλύτερες, επιτυγχάνοντας όμως ακόμα καλύτερους χρόνους αντιστροφής.

Η δομημένη αποθήκευση των ιεραρχικών πινάκων βελτιώνει τις επιδόσεις, λόγω καλύτερης τοπικότητας των δεδομένων, οπότε και προκύπτουν λιγότερα σφάλματα σελίδας. Οι δοκιμές εκτελέστηκαν μέχρις ότου να μην επαρκεί η μνήμη για κάποια από τις δυο μεθόδους αντιστροφής. Από τα αποτελέσματα παρατηρήθηκε ότι πάντοτε η αντιστροφή του Matlab αποτύγχανε πρώτη, καθώς το μέγεθος του πίνακα σε αυτή την περίπτωση περιορίζεται από τη μέγιστη συνεχόμενη διαθέσιμη μνήμη. Έτσι μιας και η αντιστροφή των ιεραρχικών πινάκων επιτυγχάνει σε περισσότερες περιπτώσεις, ένα επιπλέον αποτέλεσμα της ιεραρχικής δόμησης φαίνεται να είναι η καλύτερη εκμετάλλευση της μνήμης.

Για την εξαγωγή συμπερασμάτων θα πρέπει να ληφθεί υπόψη το γεγονός ότι η παραπάνω υλοποίηση έγινε με τη διερμηνευόμενη (interpreted) γλώσσα του Matlab. Έτσι κρίνεται σημαντικό το γεγονός ότι προκύπτουν συγκρίσιμοι ή και καλύτεροι χρόνοι αντιστροφής από ότι με την ενσωματωμένη μέθοδο αντιστροφής του Matlab, η οποία είναι μεταγλωττισμένη (compiled) και σαφώς βελτιστοποιημένη για τις ανάγκες του προϊόντος. Αν χρησιμοποιηθεί κάποια γλώσσα χαμηλότερου επιπέδου για την υλοποίηση της αντιστροφής με ιεραρχικούς πίνακες, πιθανώς να βελτιωθούν οι χρόνοι αντιστροφής και οι διαφορές στην κατανάλωση μνήμης να εξαλειφθούν.

Επιπλέον πρέπει να σημειωθεί, ότι όλα τα αποτελέσματα των δοκιμών που έγιναν είναι ακριβή. Δοκιμάστηκε low rank approximation με όριο σχετικού σφάλματος 10^{-3}, αλλά δυστυχώς λόγω της φύσης των δοκιμαστικών πινάκων η προσέγγιση χαμηλής τάξης δεν ήταν τελικά εφικτή. Καθώς η προσέγγιση χαμηλής τάξης είναι από τα κύρια χαρακτηριστικά που εκμεταλλεύονται οι ιεραρχικοί πίνακες για να επιταχύνουν τη διαδικασία, η εικόνα που σχηματίστηκε δεν ήταν ολοκληρωμένη και αναμένεται ότι με πιο ρεαλιστικούς πίνακες τα αποτελέσματα θα είναι ακόμα καλύτερα σε σχέση με τις παραδοσιακές μεθόδους αντιστροφής.

Τα αρχεία κώδικα της υλοποίησης αλλά και τα αρχεία δοκιμών, βρίσκονται άμεσα διαθέσιμα στη σελίδα του project στο GitHub.

6

Αναφορές

1. Steffen Börm, Lars Grasedyck, Wolfgang Hackbusch: Hierarchical matrices
2. Hmatrices in Matlab GitHub page.
3. An Algebraic Approach for H-matrix Preconditioners, Suely Oliveira, Fang Yang
4. Hierarchical clustering
5. Hierarchical Matrices, Steffen Borm, Christian-Albrechts-Universitat, Kiel, Short Course on Boundary Element Methods
6. Communication- Optimal H-matrix multiplication, Jack Poulson, Lexing Ying
7. Introduction to Hierarchical matrices with applications
8. Fast construction of Hierarchical matrix representation from matrix-vector multiplication, Lin Lin, Jianfeng Lu, Lexing Ying.
9. Low- rank approximation of integral operators and kernel functions, Thomas Kastl, University of Zurich
10. Domain decomposotion based H-LU preconditioning, Lars Grasedyck, Ronald Kriemann, Sabine Le Borne
11. Construction and application of hierarchical matrix preconditioners, Fang Yang, University of Iowa
12. Efficient Representation and analysis of Power Grids
13. Hierarchical Analysis of Power Distribution Networks

14. Hierarchical Random- walk Algorithms for Power Grid Analysis