ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΟΜΕΑΣ ΘΕΩΡΗΤΙΚΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ένας τυχαιοποιημένος αλγόριθμος για την μέθοδο απαλοιφής του Gauss

Ιωάννης Ευαγγέλου Καρακωνσταντής

Επιβλέποντες: **Νικόλαος Μισυρλής,** Καθηγητής
Φίλιππος Τζαφέρης, Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ένας τυχαιοποιημένος αλγόριθμος για την μέθοδο απαλοιφής του Gauss

Ιωάννης Ε. Καρακωνσταντής
Α.Μ.: M1269

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Νικόλαος Μισυρλής,** Καθηγητής
Φίλιππος Τζαφέρης, Επίκουρος Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Φίλιππος Τζαφέρης,** Επίκουρος Καθηγητής
Νικόλαος Μισυρλής, Καθηγητής

Φεβρουάριος 2015

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την τροποποίηση της μεθόδου απαλοιφής του Gauss χρησιμοποιώντας την τυχαιότητα, έτσι ώστε να έχει μια μορφή κατάλληλη για να εφαρμοστεί σε υψηλής επίδοσης, παράλληλους υπολογιστές.

Η μέθοδος απαλοιφής του Gauss είναι μια από τις πιο γνωστές μεθόδους για την επίλυση γραμμικών συστημάτων, η οποία όμως δεν μπορεί να εφαρμοστεί ανεξάρτητα για την επίλυση ενός προβλήματος. Για την παραγωγή ακριβών αποτελεσμάτων, απαιτείται επιπρόσθετα η χρήση της τεχνικής της οδήγησης (πχ. ολική, μερική). Σε παράλληλες αρχιτεκτονικές η οδήγηση δεν εισάγει μόνο επιπρόσθετο υπολογιστικό φόρτο, αλλά και σημαντική επιβάρυνση λόγω του κόστους επικοινωνίας που απαιτείται μεταξύ των επεξεργαστών.

Η μέθοδος που περιγράφουμε στην εργασία αυτή ονομάζεται Τυχαίος Μετασχηματισμός Πεταλούδας (Random Butterfly Transformation (RBT)) είναι μια μέθοδος που με πιθανότητα σχεδόν 1 μπορεί να μετασχηματίσει ένα γραμμικό σύστημα σε μορφή στην οποία δεν είναι απαραίτητη η χρήση οδήγησης. Αυτό επιτυγχάνεται πολλαπλασιάζοντας κατάλληλα τους πίνακες του αρχικού γραμμικού συστήματος με «κατάλληλα τυχαίους» αναδρομικούς πίνακες πεταλούδας.

Επιπρόσθετα, στην εργασία παρουσιάζεται η επίδραση της μεθόδου στον αριθμό συνθήκης (condition number) των πινάκων και ασχολούμαστε με ορισμένα θέματα επιλογής παραμέτρων (tuning) για την μέθοδο, όπως το εύρος των τυχαίων αριθμών και τα επίπεδα αναδρομής. Στη συνέχεια, παρουσιάζονται ορισμένα θέματα για την πιο αποτελεσματική υλοποίηση, όπως η αποδοτική αποθήκευση των αναδρομικών πινάκων πεταλούδας. Επίσης, υποδεικνύονται ορισμένα βασικά στοιχεία για την παράλληλη υλοποίηση της μεθόδου RBT σε κάρτες γραφικών (GPUs).

Τέλος, για την πειραματική επαλήθευση της απόδοσης της μεθόδου, έχουν δημιουργηθεί ορισμένες δοκιμαστικές κλάσεις πινάκων στις οποίες εφαρμόζεται η μέθοδος RBT και προκύπτουν τα απαραίτητα αριθμητικά αποτελέσματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Αριθμητική Άλγεβρα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: γραμμικό σύστημα, αριθμητική άλγεβρα, μέθοδος απαλοιφής του Gauss, Random Butterfly Transformation, παράλληλοι αλγόριθμοι, οδήγηση, ανάλυση σφάλματος

ABSTRACT

This Master thesis presents a modification over Gaussian elimination method using randomness in order to be used in high performance parallel computers.

Gauss elimination method is one of the most known and documented method for solving linear systems, but it cannot be applied directly for solving a problem. In order to produce accurate results, additional pivoting (eg. complete, partial) is required. In a parallel architecture, pivoting not only introduces additional computational cost, but also high communication overhead generated by data movement among the processors.

The method proposed in this thesis is called Random Butterfly Transformation – RBT and with a probability near to 1 is able to transform a linear system to a new form, in which pivoting is not required. That can be managed with a procedure that involves the generation of “sufficient random” recursive matrices called “recursive butterfly matrices” and the multiplication of the linear system matrices with those recursive butterfly matrices.

Additionally, we present the impact of the transformation to the condition number of the matrices and we discuss some issues about the fine tuning of the algorithm like the range of the random numbers or the recursion depth. Furthermore, additional implementation issues are examined like how to store in an efficient way the recursive butterfly matrices and implementing the method on GPU.

In order to be able to confirm the performance of the method, we conducted some tests based on testing classes of matrices proposed in the literature. Comments on the arithmetic accuracy and overall performance of the method are provided too.

SUBJECT AREA: Computational Arithmetic Algebra

KEYWORDS: linear system, arithmetic algebra, Gauss elimination method, Random Butterfly Transformation, parallel algorithms, pivoting, error analysis

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Στην εποχή μας, η υπολογιστική επιστήμη έχει διεισδύσει όσο περισσότερο στις καθημερινές μας ζωές. Το περιβάλλον από τον οποίο περιτριγυριζόμαστε είναι μια προβολή των σύγχρονων τεχνολογικών επιτευγμάτων τα οποία δεν θα μπορούσαν να επιτευχθούν χωρίς την παράλληλη πρόοδο των επιστημών. Οι σύγχρονες τεχνολογικές ανακαλύψεις, πολύ συχνά είναι το αποτέλεσμα χιλιάδων εργαστηριακών προσομοιώσεων. Οι μηχανικοί, εφοδιασμένοι με τις τεχνολογικές και επιστημονικές γνώσεις του σήμερα είναι σε θέση να μελετήσουν δυσνόητα φυσικά φαινόμενα μέσα από πολύπλοκες αριθμητικές προσομοιώσεις. Η προσομοίωση, είναι το εργαλείο εκείνο που επιτρέπει στον επιστήμονα την παρατήρηση και πρόβλεψη ενός φαινομένου, χωρίς το φαινόμενο αυτό να έχει εκδηλώσει την φυσική του παρουσία.

Για την παρατήρηση και πρόβλεψη τέτοιων φαινομένων (πχ μετεωρολογικά φαινόμενα), επιστρατεύονται υψηλής απόδοσης ηλεκτρονικοί υπολογιστές. Οι υπολογιστές αυτοί συνήθως αποτελούνται από κατανεμημένες παράλληλες αρχιτεκτονικές ή υβριδικές αρχιτεκτονικές που εμπλέκουν κεντρικές μονάδες επεξεργασίας (CPU) και μονάδες επεξεργασίας γραφικών (GPU), ενώ συναντάμε και διανυσματικούς (vector) υπολογιστές.

Πολυάριθμες είναι οι αριθμητικές μέθοδοι που εμπλέκουν στον πυρήνα τους την επίλυση γραμμικών συστημάτων. Ανατρέχοντας στην βιβλιογραφία μπορούμε να βρούμε δεκάδες τεχνικές επίλυσης γραμμικών συστημάτων, προσαρμοσμένες σε διαφορετικές απαιτήσεις και υλοποιήσεις. Στην παρούσα διπλωματική εργασία, θα ασχοληθούμε με την γνωστότερη από αυτές τις μεθόδους, την μέθοδο απαλοιφής του Gauss.

Η μέθοδος απαλοιφής του Gauss είναι από τις παλαιότερες και πιο καλά τεκμηριωμένες μεθόδους για την επίλυση γραμμικών συστημάτων και έχει επεκταθεί ώστε να προσαρμόζεται και σε παράλληλες αρχιτεκτονικές. Η μέθοδος απαλοιφής του Gauss όταν εφαρμόζεται απευθείας σε ένα γραμμικό σύστημα είναι με μεγάλη πιθανότητα ασταθής. Κατά την διάρκεια της εκτέλεσης της, είναι πολύ πιθανό να προκληθεί διαίρεση με το μηδέν, ή με κάποια πολύ μικρή ποσότητα. Σε συνδυασμό με την πεπερασμένη αριθμητική ακρίβεια των ηλεκτρονικών υπολογιστών, τα αριθμητικά αποτελέσματα που προκύπτουν πολύ συχνά στερούνται ακρίβειας.

Για την αντιμετώπιση αυτού του φαινομένου, σε συνδυασμό με την εκτέλεση της μεθόδου απαλοιφής του Gauss εφαρμόζεται και η διαδικασία της οδήγησης (pivoting). Πολλές τεχνικές οδήγησης έχουν προταθεί, με γνωστότερες την ολική (complete) και την μερική (partial) οδήγηση. Η ολική οδήγηση (με πολυπλοκότητα $O(n^3)$) είναι μια σχετικά απαισιόδοξη επιλογή, αφού στην πράξη η χρήση μερικής οδήγησης προσφέρει ικανοποιητική αριθμητική ακρίβεια και έχει πολυπλοκότητα $O(n^2)$. Επίσης, εφαρμόζονται και άλλες τεχνικές οδήγησης όπως η rook οδήγηση της οποίας η πολυπλοκότητα κυμαίνεται μεταξύ του $O(n^2)$ και του $O(n^3)$. Σε πολυπύρηνους υπολογιστές με περισσότερα νήματα εφαρμόζονται τεχνικές όπως η οδήγηση σε ζεύγη (pairwise pivoting) και η τεχνική αυξανόμενης οδήγησης (incremental pivoting).

Ωστόσο, η εισαγωγή της οδήγησης στις παράλληλες υψηλής απόδοσης αρχιτεκτονικές, επιφέρει και αύξηση όχι μόνο του υπολογιστικού χρόνου αλλά και της καθυστέρησης εξαιτίας των επικοινωνιών μεταξύ των επεξεργαστών και της μεταφοράς δεδομένων μεταξύ τους. Σε αυτή τη κατεύθυνση αναπτύσσονται αλγόριθμοι οδήγησης βέλτιστης επικοινωνίας (communication optimal).

Σε αντιδιαστολή με αυτή τη προσέγγιση, ο μετασχηματισμός τυχαίας πεταλούδας (RBT) είναι μια προκαταρτική διαδικασία κατά την οποία το αρχικό γραμμικό σύστημα μετασχηματίζεται σε μια μορφή κατά την οποία αν εφαρμοστεί σε αυτό η μέθοδος

απαλοιφής του Gauss, να μην είναι αναγκαία η χρήση οδήγησης. Είναι μια απλή διαδικασία, εύκολα υλοποιήσιμη σε παράλληλα συστήματα, και παρουσιάζει υπολογιστική πολυπλοκότητα της τάξης του $O(n^2)$.

Ο μετασχηματισμός RBT δημιουργεί δυο κατάλληλα διαμορφωμένους «τυχαίους» πίνακες με τους οποίους πολλαπλασιάζεται ο αρχικός πίνακας A του γραμμικού συστήματος. Στην συνέχεια ένα ενδιάμεσο γραμμικό σύστημα επιλύεται χωρίς την χρήση οδήγησης, ενώ η λύση στο αρχικό πρόβλημα δίνεται ως το γινόμενο της ενδιάμεσης λύσης επί έναν τυχαίο πίνακα.

Η διάρθρωση της εργασίας είναι η ακόλουθη:

Στο πρώτο κεφάλαιο της εργασίας γίνεται μια παρουσίαση της μεθόδου απαλοιφής του Gauss. Παρουσιάζεται η μέθοδος καθώς και οι δυο πιο χαρακτηριστικές τεχνικές οδήγησης: η μερική και η ολική. Παρουσιάζεται ο ψευδοκώδικας της τεχνικής και η ανάλυση σφάλματος.

Στον πυρήνα του θέματος της εργασίας, βρίσκεται το δεύτερο κεφάλαιο, το οποίο περιλαμβάνει το απαραίτητο θεωρητικό υπόβαθρο για την περιγραφή της μεθόδου. Η ίδια η μέθοδος του τυχαίου μετασχηματισμού πεταλούδας παρουσιάζεται αναλυτικά καθώς και η επίδραση της στους αρχικούς πίνακες

Το τρίτο κεφάλαιο της εργασίας πραγματεύεται ορισμένα θέματα υλοποίησης της τεχνικής και είναι σαφώς προσανατολισμένο σε πιο τεχνικά ζητήματα. Σε αυτό το κεφάλαιο αναλύεται και παρουσιάζεται ο ψευδοκώδικας της μεθόδου και η υπολογιστική της πολυπλοκότητα. Επιπρόσθετα παρουσιάζεται ένας αποδοτικός τρόπος αποθήκευσης των πινάκων πεταλούδας και οι κατευθυντήριες γραμμές για την υλοποίηση της μεθόδου σε κάρτες γραφικών.

Στο τέταρτο και τελευταίο κεφάλαιο, περιγράφεται το περιβάλλον δοκιμής στο οποίο επιβεβαιώνουμε την αριθμητική ακρίβεια της μεθόδου. Η μέθοδος δοκιμάστηκε σε ένα σύνολο από δοκιμαστικούς πίνακες που προτείνονται στην βιβλιογραφία και τα αριθμητικά αποτελέσματα των δοκιμών συγκρίνονται με την απόδοση του linear system solver του MatLab.

Κλείνοντας, στα παραρτήματα της εργασίας μπορούμε να βρούμε μια σύντομη επεξήγηση και παρουσίαση για το λογισμικό που αναπτύχθηκε στο πλαίσιο της εργασίας. Πρόκειται για μια υλοποίηση στο περιβάλλον του MatLab της μεθόδου RBT, στην οποία η μέθοδος δοκιμάζεται για ένα σύνολο από δοκιμαστικούς πίνακες και η αριθμητική της ακρίβεια συγκρίνεται με τον linear system solver του MatLab.

1. Επίλυση γραμμικών συστημάτων με την μέθοδο απαλοιφής του Gauss

Σε αυτή την ενότητα της εργασίας παρουσιάζεται η μέθοδος απαλοιφής του Gauss με ή χωρίς οδήγηση για την επίλυση γραμμικών συστημάτων. Η μέθοδος απαλοιφής του Gauss είναι μια από τις πιο γνωστές άμεσες μεθόδους επίλυσης γραμμικών συστημάτων. Επιπρόσθετα, μπορεί να χρησιμοποιηθεί για τον υπολογισμό παραγοντοποιήσεων σε πίνακες ή τον υπολογισμό οριζουσών ή την αντιστροφή πίνακα. Σκοπός μας, είναι να παρουσιάσουμε την μέθοδο, τον ψευδοκώδικα της καθώς και τις διάφορες τεχνικές οδήγησης που χρησιμοποιούνται για να εξασφαλίσουν την αριθμητική ευστάθεια της [1] [2].

Για λόγους απλότητας, θα θεωρήσουμε τον πίνακα A έναν $n \times n$ τετραγωνικό και αντιστρέψιμο πίνακα, καθώς και τον πίνακα στήλη b (πίνακας σταθερών όρων) διάστασης $n \times 1$. Η μέθοδος απαλοιφής του Gauss αποτελείται από δύο κύριες φάσεις: την φάση της τριγωνοποίησης και την φάση της προς τα πίσω αντικατάστασης.

Κατά την **φάση της τριγωνοποίησης,** η μέθοδος απαλείφει (μηδενίζει) σε κάθε διαδοχικό βήμα τα στοιχεία του πίνακα A που βρίσκονται κάτω από την κύρια διαγώνιο. Μετά από $\mathrm{n}-1$ βήματα προκύπτουν δυο πίνακες U (άνω τριγωνικός) και L (κάτω τριγωνικός). Σε αυτή τη φάση, το αρχικό σύστημα (A, b) μετασχηματίζεται στο ισοδύναμο άνω τριγωνικό σύστημα (U, L^{-1}b).

Κατά την **φάση της προς τα πίσω αντικατάστασης,** η μέθοδος επιλύει το άνω τριγωνικό σύστημα που προκύπτει από την φάση της τριγωνοποίησης. Έτσι, το (U, L^{-1}b) μετασχηματίζεται στο ισοδύναμο σύστημα (I, $U^{-1}L^{-1}$b).

Για την πιο αναλυτική περιγραφή της μεθόδου θεωρούμε το ακόλουθο γραμμικό σύστημα n εξισώσεων με n αγνώστους:

$$\begin{gathered}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\
a_{21}^{(1)}x_1 + a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\
a_{31}^{(1)}x_1 + a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \cdots + a_{3n}^{(1)}x_n = b_3^{(1)} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \cdots + a_{nn}^{(1)}x_n = b_n^{(1)}
\end{gathered} \tag{1}$$

Όπου το στοιχείο $a_{ij}^{(k)}$ συμβολίζει το a_{ij} στοιχείο του πίνακα $A^{(k)}$, δηλαδή το στοιχείο που βρίσκεται στην i-στη γραμμή, j-οστή στήλη του πίνακα A, στο k-οστό βήμα της μεθόδου. Το σύστημα σε μορφή πινάκων γράφεται ισοδύναμα ως εξής:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix} \tag{2}$$

ή πιο απλά ως: $A^{(1)}x = b^{(1)}$ με $\det A^{(1)} \neq 0$ και $b^{(1)} \neq 0$ έτσι ώστε να υπάρχει μια και μοναδική λύση.

1.1.1 Φάση τριγωνοποίησης

Σε αυτό το σημείο, περιγράφεται η φάση της τριγωνοποίησης. Ο αλγόριθμος της μεθόδου βρίσκεται στην επανάληψη 1 (k=1) και ο στόχος μας είναι η απαλοιφή του αγνώστου x_1 από τις γραμμές i=2,3,…,n. Αφού απαλειφθεί ο x_1, το σύστημα παίρνει την ακόλουθη (πιο απλή) ισοδύναμη μορφή:

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\
a_{32}^{(2)}x_2 + a_{33}^{(2)}x_3 + \cdots + a_{3n}^{(2)}x_n &= b_3^{(2)} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\
a_{n2}^{(2)}x_2 + a_{n3}^{(2)}x_3 + \cdots + a_{nn}^{(2)}x_n &= b_n^{(2)}
\end{aligned} \tag{3}
$$

Σε μορφή πινάκων θα έχει την μορφή:

$$
\begin{bmatrix}
a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1n}^{(1)} \\
0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\
0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix} \tag{4}
$$

ή πιο απλά ως: $A^{(2)}x = b^{(2)}$.

Στο επόμενο βήμα, ο αλγόριθμος θα επαναλάβει την ίδια διαδικασία προκειμένου να απαλείψει τον άγνωστο x_2 κ.ο.κ.

Μετά από n-1 επαναληπτικά βήματα, το γραμμικό σύστημα θα πάρει την ακόλουθη μορφή:

$$
\begin{aligned}
a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 + \cdots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\
a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 + \cdots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\
a_{33}^{(3)}x_3 + \cdots + a_{3n}^{(3)}x_n &= b_3^{(3)} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\
a_{nn}^{(n)}x_n &= b_n^{(n)}
\end{aligned} \tag{5}
$$

ή πιο απλά ως: $A^{(n)}x = b^{(n)}$. Τελικά ο πίνακας $A^{(n)} = U$ που προκύπτει θα είναι άνω τριγωνικός.

Προκειμένου να απαλειφθεί ο άγνωστος x_1 από την i-στη εξίσωση προσθέτουμε m_{i1} (πολλαπλασιαστής) φορές την πρώτη εξίσωση (οδηγός εξίσωση), στην i-οστή εξίσωση. Ο υπολογίζεται ως εξής:

$$m_{i1} = -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i = 2, \dots, n \qquad a_{11}^{(1)} \neq 0 \tag{6}$$

Το στοιχείο $a_{11}^{(1)}$ ονομάζεται οδηγό στοιχείο. Αν θεωρήσουμε τον πίνακα $M^{(1)}$:

$$M^{(1)} = \left[\begin{array}{c|c} 1 & 0 \\ \hline m_{21} & \\ m_{31} & I_{n-1} \\ \vdots & \\ m_{n1} & \end{array}\right] \tag{7}$$

τότε, παρατηρούμε ότι η πρώτη επανάληψη της μεθόδου μπορεί να γίνει πολλαπλασιάζοντας από αριστερά το αρχικό σύστημα με τον πίνακα $M^{(1)}$:

$$M^{(1)}A^{(1)}x = M^{(1)}b^{(1)} \quad \text{ή} \tag{8}$$

$$A^{(2)}x = b^{(2)} \tag{9}$$

Το σύστημα $A^{(1)}x = b^{(1)}$ είναι ισοδύναμο με το σύστημα $A^{(2)}x = b^{(2)}$ που προκύπτει.

Η μέθοδος απαλείφει και τους υπόλοιπους αγνώστους διαδοχικά με όμοιο τρόπο. Για την απαλοιφή του αγνώστου x_r από τις υπόλοιπες n - r εξισώσεις, προσθέτουμε m_{ir} φορές την οδηγό εξίσωση στην i-οστή εξίσωση, για i = r + 1,….,n . Οι πολλαπλασιαστές υπολογίζονται ως εξής:

$$m_{ir} = -\frac{a_{ir}^{(r)}}{a_{rr}^{(r)}}, \quad i = r+1, \dots, n \qquad a_{rr}^{(r)} \neq 0 \tag{10}$$

Θεωρώντας τον πίνακα $M^{(r)}$:

$$M^{(r)} = \left[\begin{array}{c|ccccc} I_{r-1} & 0 & & \cdots & & 0 \\ \hline 0 & 1 & & & & \\ & m_{r+1,r} & 1 & & 0 & \\ \vdots & m_{r+2,r} & 0 & 1 & & \\ & \vdots & & & \ddots & \\ 0 & m_{n,r} & 0 & \cdots & & 1 \end{array}\right] \tag{11}$$

τότε στην r επανάληψη της μεθόδου το σύστημα θα έχει την μορφή:

$$M^{(r)}A^{(r)}x = M^{(r)}b^{(r)} \quad \text{ή} \tag{12}$$

$$A^{(r+1)}x = b^{(r+1)} \tag{13}$$

Η φάση της τριγωνοποίησης μπορεί να προκύψει μετά τον πολλαπλασιασμό του αρχικού συστήματος με τον πίνακα $M = M^{(n-1)} \dots M^{(2)}M^{(1)}$:

$$MA^{(1)}x = Mb^{(1)} \tag{14}$$

που είναι ισοδύναμο με

$$A^{(n)}x = b^{(n)} \tag{15}$$

1.1.2 Φάση προς τα πίσω αντικατάστασης

Ο αλγόριθμος της μεθόδου απαλοιφής του Gauss, στην φάση της προς τα πίσω αντικατάστασης δέχεται σαν είσοδο έναν άνω τριγωνικό πίνακα $A^{(n)}$ και τον πίνακα των σταθερών όρων $b^{(n)}$ όπως αυτοί έχουν προκύψει από την προηγούμενη φάση της τριγωνοποίησης. Έτσι, ο τελευταίος άγνωστος x_n προκύπτει άμεσα $x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}$ από την τελευταία εξίσωση, ενώ οι υπόλοιποι άγνωστοι υπολογίζονται διαδοχικά προς τα πίσω ως εξής:

$$x_i = \frac{b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j}{a_{ii}^{(i)}}, \qquad i = 1, \dots, n-1 \tag{16}$$

1.2 Μέθοδος απαλοιφής του Gauss χωρίς οδήγηση

Σε αυτή την ενότητα της εργασίας, θα ασχοληθούμε με την τεχνική περιγραφή του αλγορίθμου της μεθόδου απαλοιφής του Gauss χωρίς την χρήση της τεχνικής της οδήγησης. Παρακάτω δίνεται ο ψευδοκώδικας της μεθόδου:

```
for k = 1 to n - 1 do
    for i = k + 1 to n do
        mik = aik / akk; //υπολογισμός πολλαπλασιαστή m_ik = a_ik^(k) / a_kk^(k)
        for j = k + 1 to n
            aij = aij - mik * akj; //a_ij^(k+1) = a_ij^(k) - m_ik a_kj^(k)
        end for
    end for
end for
```

Αποδεικνύεται ότι η μέθοδος έχει πολυπλοκότητα $O\left(\frac{n^3}{3}\right)$ υπολογιστικές πράξεις.

Όπως έχουμε αναφέρει ο πίνακας A κατά την διάρκεια εκτέλεσης του αλγορίθμου μετασχηματίζεται σε έναν άνω τριγωνικό και σε έναν κάτω τριγωνικό. Ο ψευδοκώδικας που παρουσιάζουμε δεν ενημερώνει τα μηδενικά στοιχεία κάτω από την κύρια διαγώνιο (γιατί δεν εξυπηρετεί κάποιο σκοπό) με αποτέλεσμα ο κάτω τριγωνικός πίνακας να περιέχει μη αξιοποιήσιμη πληροφορία. Αν όμως, στην θέση των μηδενικών στοιχείων

αποθηκευθούν οι αντίστοιχοι πολλαπλασιαστές, τότε στο τέλος της μεθόδου απαλοιφής του Gauss, ο πίνακας A θα είναι της μορφής $A = [L\backslash U]$, όπου:

$$L = \begin{pmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{31} & m_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \dots & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} a_{11}^{(n)} & a_{12}^{(n)} & a_{13}^{(n)} & \dots & a_{1n}^{(n)} \\ & a_{22}^{(n)} & a_{23}^{(n)} & \dots & a_{2n}^{(n)} \\ & & a_{33}^{(n)} & \dots & a_{3n}^{(n)} \\ & & & \ddots & \vdots \\ & & & & a_{nn}^{(n)} \end{pmatrix} \qquad (17)$$

Η συγκεκριμένη υλοποίηση της μεθόδου απαλοιφής του Gauss (χωρίς την χρήση οδήγησης) μπορεί να δημιουργήσει τα παρακάτω προβλήματα:

1. Ένα ή περισσότερα από τα διαγώνια στοιχεία $a_{kk}^{(k)}$ μπορεί να είναι μηδέν. Όταν ο πίνακας A έχει τάξη (rank) μικρότερη από n, τότε αυτό συμβαίνει πάντα.

2. Οι τιμές των στοιχείων στους πίνακες $\hat{L}$ και $\hat{U}$ που υπολογίζονται από την μέθοδο (ο χαρακτήρας $\widehat{\ }$ συμβολίζει την προσέγγιση) μπορεί να μην είναι ακριβής εξαιτίας των σφαλμάτων στρογγύλευσης που οφείλονται στην πεπερασμένη αναπαράσταση των αριθμών κινητής υποδιαστολής σε έναν υπολογιστή. Ο πίνακας A θα ισούται με το γινόμενο των πινάκων $\hat{L}$ και $\hat{U}$ συν το σφάλμα στρογγύλευσης. Δηλαδή, $A = \hat{L}\hat{U} + E$, όπου E ο πίνακας των σφαλμάτων στρογγύλευσης. Πιο αναλυτικά, $A = \hat{L}\hat{U} + \sum_{k=1}^{n-1} E^{(k)}$ όπου $E^{(k)} = \left(\epsilon_{ij}^{(k)}\right)$ είναι το σφάλμα στο k-οστό βήμα του αλγορίθμου για το στοιχείο στην θέση (i,j). Το σφάλμα $\epsilon_{ij}^{(k)}$ ορίζεται ως εξής:

$$\epsilon_{ij}^{(k)} = \begin{cases} \hat{a}_{ij}^{(k)}\delta_{ij} & i \geq k+1, \quad j = k \\ -\hat{m}_{ik}\hat{a}_{kj}^{(k)} - \hat{a}_{ij}^{(k+1)}\delta_{ij} & i \geq k+1, \quad j \geq k+1 \\ 0 & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases} \qquad (18)$$

Όπου δ_{ij} είναι η μονάδα σφάλματος στρογγύλευσης (unit roundoff error) που εισάγεται από τις πράξεις μεταξύ αριθμών κινητής υποδιαστολής. Οι τιμές $\hat{a}_{ij}^{(k)}, \hat{m}_{ik}$ που υπολογίζονται από τον αλγόριθμο εμπεριέχουν το σφάλμα αυτό, και σε περίπτωση που ο πίνακας A έχει μεγάλο αριθμό συνθήκης (ill-conditioned), το σφάλμα αυξάνει δραματικά.

Για τον λόγο αυτό, η μέθοδος πρέπει να εφαρμόζεται με κάποια τεχνική οδήγησης κάτι που εξετάζεται στην ακόλουθη παράγραφο.

1.3 Μέθοδος απαλοιφής του Gauss με μερική οδήγηση

Όταν η μέθοδος απαλοιφής του Gauss χρησιμοποιείται με ολική οδήγηση, τότε εκτελείται σάρωση σε όλο το ενεργό τμήμα του πίνακα για την εύρεση του απόλυτα μεγαλύτερου στοιχείου μια διαδικασία που κοστίζει $O(n^3)$. Το ενεργό τμήμα του πίνακα στην μέθοδο απαλοιφής του Gauss στο k-οστό βήμα είναι ο υποπίνακας που βρίσκεται κάτω από την k-οστή σειρά και δεξιά από την k-οστή στήλη. Στην πράξη, αυτή η αντιμετώπιση είναι υπολογιστικά αρκετά αποθαρυντική αφού συχνά η μερική οδήγηση

επιτυγχάνει ικανοποιητική ακρίβεια. Σε αντίθεση με την ολική, η μερική οδήγηση πραγματοποιεί αναζήτηση για το μεγαλύτερο στοιχείο μόνο στην τρέχουσα στήλη και όχι σε όλο το ενεργό τμήμα του πίνακα με αποτέλεσμα να εμφανίζει πολυπλοκότητα $O(n^2)$. Παρακάτω ακολουθεί ένα ενδεικτικό τμήμα ψευδοκώδικα της μεθόδου απαλοιφής του Gauss με μερική οδήγηση:

```
for k = 1 to n - 1 do
    maxValue = 0;
    r = 0;
    for i = k to n do // εύρεση της μεγαλύτερης τιμής στην τρέχουσα στήλη
        if |aik| > maxValue then
            maxValue = |aik|;
            r = i;
        end if
    end for
    if maxValue = 0 then
        HALT; // τερμάτισε την μέθοδο αν δεν υπάρχει μη μηδενικό στοιχείο
    end if
    //αντιμετάθεση των γραμμών k και r
    for j = k to n do
        tmp = akj;
        akj = arj;
        arj = tmp;
    end for
    //εφάρμοσε την μέθοδο απαλοιφής του Gauss στην k στήλη
    for i = k + 1 to n do
        mik = aik / akk;
        for j = k + 1 to n do
            aij = aij - mik * akj;
        end for
    end for
end for
```

Στην πράξη, μπορεί η μερική οδήγηση να παρέχει ικανοποιητική αριθμητική ακρίβεια σε συνδυασμό με μικρότερο υπολογιστικό κόστος, ωστόσο σε αυτή τη περίπτωση, το φράγμα σφάλματος (βλ. επόμενη ενότητα) για το στοιχείο στην θέση (i,j) του k-οστού βήματος εξασθενεί. Σύμφωνα με τον Wilkinson [3] [4] το φράγμα $\left|\hat{a}_{ij}^{(k)}\right| \leq 2^{k-1} max_{i,j}|a_{ij}|$ εξακολουθεί να ισχύει. Αυτό το φράγμα παρέχει μια μικρή εγγύηση για την ακρίβεια της μεθόδου για μέτριες ή μεγάλες τιμές του n. Στην βιβλιογραφία [5] [6] έχουν παρουσιαστεί ορισμένα προβλήματα στα οποία η μερική οδήγηση δίνει μη ακριβή αριθμητικά αποτελέσματα.

1.4 Μέθοδος απαλοιφής του Gauss με ολική οδήγηση

Η ολική οδήγηση (complete pivoting) είναι μια τεχνική που χρησιμοποιείται στην μέθοδο απαλοιφής του Gauss και έχει σαν αποτέλεσμα την αντιμετάθεση των γραμμών και των στηλών του πίνακα A, έτσι ώστε να μην εμφανίζονται σε αυτόν μηδενικά στοιχεία στην κύρια διαγώνιο. Ο αλγόριθμος της ολικής οδήγησης εντοπίζει το απόλυτα μεγαλύτερο στοιχείο στο ενεργό τμήμα του πίνακα και αντιμεταθέτει γραμμές και στήλες έτσι ώστε να βρεθεί στην κύρια διαγώνιο του πίνακα.

Στην συνέχεια, παρατίθεται ένα ενδεικτικό τμήμα ψευδοκώδικα της μεθόδου με ολική οδήγηση:

```
for k = 1 to n - 1 do
    maxValue = 0; // εύρεση της μεγαλύτερης τιμής στον ενεργό πίνακα
    r = 0;
    c = 0;
    for i = k to n do
        for j = k to n do
            if |aij| > maxValue then
                maxValue = |aij|;
                r = i;  // εύρεση της θέσης του στοιχείου
                c = j;
            end if
        end for
    end for
    if maxValue = 0 then
        HALT; // τερμάτισε την μέθοδο αν δεν υπάρχει μη μηδενικό στοιχείο
    end if
    //αντιμετάθεση των γραμμών k και r
    for j = k to n do
        tmp = akj;
        akj = arj;
        arj = tmp;
    end for
    //αντιμετάθεση των στηλών k και c
    for i = 1 to n do
        tmp = aik;
        aik = aic;
        aic = tmp;
    end for
    //εφάρμοσε την μέθοδο απαλοιφής του Gauss στην k στήλη
    for i = k + 1 to n do
        mik = aik / akk;
        for j = k + 1 to n do
            aij = aij - mik * akj;
        end for
    end for
end for
```

Η μέθοδος με ολική οδήγηση στο κάθε k-βήμα τοποθετεί στην διαγώνιο του πίνακα (στην θέση (k,k)) το κατά απόλυτη τιμή μεγαλύτερο στοιχείο, συνεπώς παρατηρούμε ότι πάντα θα ισχύει $|m_{ik}| \leq 1$. Αυτή η παρατήρηση οδηγεί στο θεώρημα του Wilkinson [3] [4]:

Θεώρημα: Έστω A ένας μη ιδιάζων πίνακας διάστασης $n \times n$ και έστω ότι χρησιμοποιείται αριθμητική κινητής υποδιαστολής με t σημαντικά ψηφία και βάση β, όπου $\beta^{1-t} \leq \frac{1}{n}$. Τότε οι πίνακες $\hat{L}$ και $\hat{U}$ που υπολογίζονται από την μέθοδο απαλοιφής του Gauss με ολική οδήγηση με την χρήση αριθμών κινητής υποδιαστολής με μονάδα σφάλματος στρογγύλευσης (unit roundoff error) u ικανοποιούν την σχέση $\hat{L}\hat{U} = A + E$ όπου E ο πίνακας των σφαλμάτων στρογγύλευσης τέτοια ώστε:

$$\|E\|_\infty \leq n^2 max_{i,j,k}\left|\hat{a}_{ij}^{(k)}\right| u \qquad (19)$$

Επιλύοντας το γραμμικό σύστημα $Ax = b$ με την μέθοδο, η λύση $\hat{x}$ που προκύπτει ικανοποιεί την συνθήκη $(A + \delta A)\hat{x} = b$ όπου δA είναι ο πίνακας διάστασης $n \times n$ των σφαλμάτων στον A τέτοιος ώστε:

$$\|\delta A\|_\infty \leq p(n) max_{i,j,k} \left|\hat{a}_{ij}^{(k)}\right| u \tag{20}$$

όπου $p(n) = O(n^3)$ είναι ένα πολυώνυμο. Όταν το σύστημα επιλύεται με ολική οδήγηση ισχύει:

$$\left|\hat{a}_{ij}^{(\kappa)}\right| \leq k^{\frac{1}{2}} \left(2^1 3^{\frac{1}{2}} 4^{\frac{1}{3}} \dots k^{\frac{1}{k-1}}\right)^{\frac{1}{2}} max_{i,j}|a_{ij}| = O\left(k^{\frac{1}{2}} k^{\frac{1}{4} ln\, k}\right) max_{i,j}|a_{ij}| \tag{21}$$

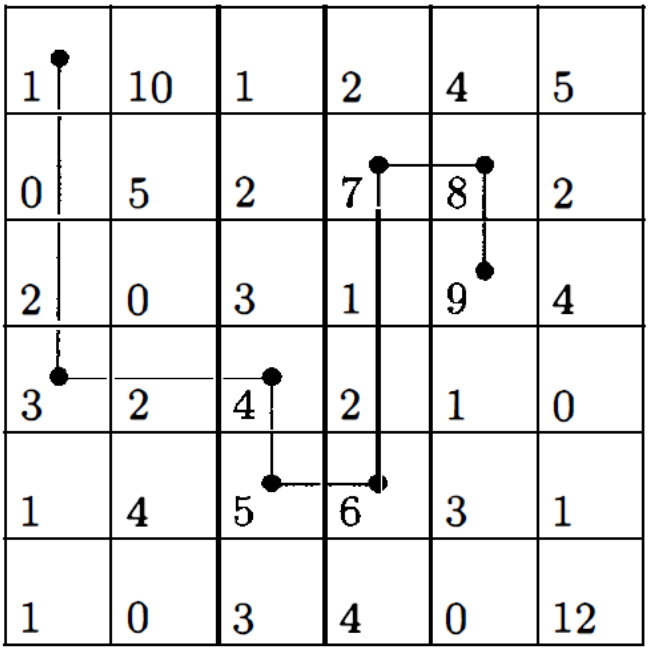
1.5 Μέθοδος απαλοιφής του Gauss με Rook οδήγηση

Η rook οδήγηση είναι μια λιγότερο γνωστή στρατηγική οδήγησης και έχει πάρει το όνομα της από το σκάκι, επειδή παρομοιάζει με τις κινήσεις που επιτρέπεται να κάνει το πιόνι του πύργου σε μια παρτίδα σκάκι [7]. Η rook οδήγηση σε κάθε βήμα επιλέγει ένα οδηγό στοιχείο το μέγεθος του οποίου βρίσκεται μεταξύ του μεγέθους του οδηγού στοιχείο που θα επέλεγε η μερική και η πλήρης οδήγηση.

Ο αλγόριθμος της rook οδήγησης, στο k-στο βήμα αντιμεταθέτει τις σειρές k και r καθώς και τις στήλες k και s όπου:

$$\left|a_{rs}^{(k)}\right| = max_{k \leq i \leq n} \left|a_{is}^{(k)}\right| = max_{k \leq j \leq n} \left|a_{rj}^{(k)}\right| \tag{22}$$

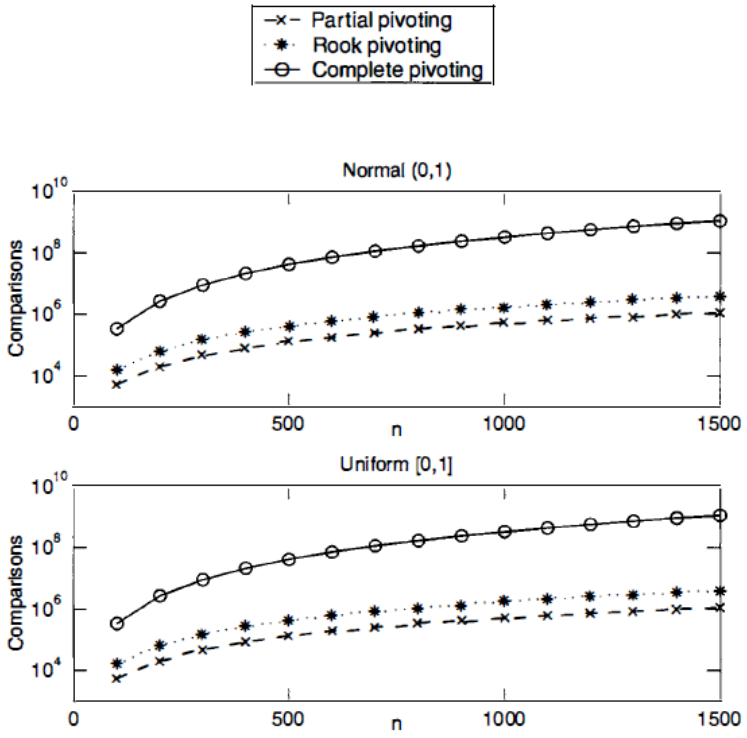
Δηλαδή, το οδηγό στοιχείο επιλέγεται έτσι ώστε να είναι το κατά απόλυτη τιμή μεγαλύτερο στην σειρά και στην στήλη του. Επιπρόσθετα, κατά την διάρκεια της αναζήτησης για το οδηγό στοιχείο, τα στοιχεία που έχουν συγκριθεί σε προηγούμενα βήματα μπορούν να παραληφθούν.



Σχήμα 1: Διαδικασία αναζήτησης του πρώτου οδηγού στοιχείου για έναν 6X6 πίνακα. Οι τελείες υποδηλώνουν ένα υποψήφιο οδηγό στοιχείο [7].

Στην βέλτιστη περίπτωση, οι συγκρίσεις που πραγματοποιεί η rook οδήγηση είναι οι διπλάσιες από αυτές στην μερική οδήγηση, ενώ στην χειρότερη περίπτωση θα πραγματοποιήσει όσες συγκρίσεις και η ολική οδήγηση.

Αποδεικνύεται από τον Foster [8] ότι αν τα στοιχεία $\left\{\alpha_{ij}^{(k)}\right\}_{i,j=k}^{n}$ ακολουθούν μια οποιαδήποτε συνεχή τυχαία κατανομή, τότε ο αναμενόμενος αριθμός συγκρίσεων στο βήμα k της μεθόδου, θα είναι το πολύ $(n-k)e$. Εάν η υπόθεση ικανοποιείται, τότε ο συνολικός αριθμός συγκρίσεων φράσσεται από το $\frac{(n-1)ne}{2}$ οποίο είναι της ίδια τάξης με την μερική οδήγηση. Αριθμητικά πειράματα τα οποία έχουν διεξαχθεί [7] [9], υποδεικνύουν ότι στην πράξη η rook οδήγηση έχει υπολογιστικό κόστος ένα μικρό πολλαπλάσιο του κόστους της μερικής οδήγησης αλλά μικρότερο από αυτό της ολικής. Στο σχήμα 2 που ακολουθεί, παρουσιάζονται δυο γραφήματα με τον αριθμό συγκρίσεων που απαιτούνται για την μερική, την ολική και την rook οδήγηση. Πρόκειται για τυχαίους πίνακες διάστασης μέχρι 1500 με στοιχεία επιλεγμένα από μια τυχαία ομοιόμορφη και μια τυχαία κανονική κατανομή.



Σχήμα 2: Πλήθος συγκρίσεων σε τυχαίους πίνακες με μερική, ολική και rook οδήγηση [7].

Ο ψευδοκώδικας για την επιλογή οδηγού στοιχείου στην rook οδήγηση παρουσιάζεται παρακάτω [7] [10]:

```
row = k;
column = k;
(colMax, rowIndex) = max(|A(k: n, k)|);
rowMax = 0;
while rowMax < colMax
```

```
    row = rowIndex + k - 1;
    (rowMax, colIndex) = max(|A(row, k: n)|);
    if colMax < rowMax
        column = colIndex + k - 1;
        (colMax, rowIndex) = max(|A(k: n, column)|);
    end if
end while
```

1.6 Η μέθοδος απαλοιφής του Jordan

Με την χρήση της μεθόδου απαλοιφής του Jordan είναι δυνατόν να μετασχηματιστεί ο πίνακας των συντελεστών των αγνώστων σε ένα διαγώνιο πίνακα. Η μορφή του πίνακα που προκύπτει είναι απλούστερη από εκείνη της μεθόδου απαλοιφής του Gauss και οι άγνωστοι x_i προκύπτουν με μια διαίρεση χωρίς να απαιτείται η διαδικασία της προς τα πίσω αντικατάστασης [2].

Μετά από n βήματα της μεθόδου απαλοιφής του Jordan προκύπτει το ακόλουθο διαγώνιο σύστημα (βασιζόμαστε στο σύστημα που παρουσιάστηκε στην μέθοδο απαλοιφής του Gauss):

$$\begin{aligned} a_{11}^{(1)}x_1 &= b_1^{(1)} \\ a_{22}^{(2)}x_2 &= b_2^{(2)} \\ a_{33}^{(3)}x_3 &= b_3^{(3)} \\ \ddots &\ \vdots \\ a_{nn}^{(n)}x_n &= b_n^{(n)} \end{aligned} \tag{23}$$

το οποίο ισοδύναμα γράφεται σαν:

$$A^{(n)}x = b^{(n+1)} \tag{24}$$

όπου ο $A^{(n)}$ είναι ένας διαγώνιος πίνακας. Η λύση του γραμμικού συστήματος προκύπτει εύκολα:

$$x_i = \frac{1}{a_{ii}^{(i)}} b_i^{(n+1)} \tag{25}$$

Ο αλγόριθμος της μεθόδου απαλοιφής του Jordan μετασχηματίζει τον πίνακα A σε ένα διαγώνιο πίνακα ίδιας τάξης σε n βήματα που αντιστοιχούν σε n πράξεις των γραμμών του πίνακα. Υπό μορφή πινάκων η μέθοδος του Jordan αναζητεί έναν μη ιδιάζων $n \times n$ πίνακα M για τον οποίο ισχύει:

$$MAx = Mb \tag{26}$$

με

$$MA = I \tag{27}$$

Συνεπώς

$$M = A^{-1} \tag{28}$$

και

$$X = Mb \tag{29}$$

Αν υποθέσουμε ότι το αρχικό σύστημα είναι το:

$$A^{(1)}x = b^{(1)} \tag{30}$$

Τότε αν ο $M^{(1)}$ είναι ο πίνακας:

$$M^{(1)} = \left[\begin{array}{c|c} \mu_{11} & 0 \\ \hline \mu_{21} & \\ \mu_{31} & I_{n-1} \\ \vdots & \\ \mu_{n1} & \end{array}\right] \tag{31}$$

όπου για $\alpha_{11}^{(1)} \neq 0$

$$\mu_{i1} = \begin{cases} \dfrac{1}{a_{11}^{(1)}}, & i = 1 \\ -\dfrac{a_{i1}^{(1)}}{a_{11}^{(1)}}, & i \neq 1 \end{cases} \tag{32}$$

Το πρώτο βήμα της μεθόδου απαλοιφής του Jordan γίνεται πολλαπλασιάζοντας τους πίνακες του γραμμικού συστήματος με τον πίνακα $M^{(1)}$:

$$M^{(1)}A^{(1)}x = M^{(1)}b \tag{33}$$

ή

$$A^{(2)}x = b^{(2)} \tag{34}$$

Το r-οστό βήμα της μεθόδου θα περιγράφεται από την εξίσωση:

$$M^{(r)}A^{(r)}x = M^{(r)}b$$

όπου

$$M^{(r)} = \left[\begin{array}{ccc|c|ccc} & & & \mu_{1r} & & & \\ & I_{r-1} & & \vdots & & \mathbf{0} & \\ & & & \mu_{r-1,r} & & & \\ \hline 0 & \cdots & 0 & \mu_{rr} & 0 & \cdots & 0 \\ \hline & & & \mu_{r+1,r} & & & \\ & \mathbf{0} & & \vdots & & I_{n-r} & \\ & & & \mu_{nr} & & & \end{array}\right] \tag{35}$$

και τα μ_{ir} δίνονται από τον τύπο:

$$\mu_{ir} = \begin{cases} \dfrac{1}{a_{rr}^{(r)}}, & i = r \\ -\dfrac{a_{ir}^{(r)}}{a_{rr}^{(r)}}, & i \neq r \end{cases}, \qquad a_{rr}^{(r)} \neq 0, \qquad r = 1, \dots, n \tag{36}$$

Στο r βήμα το σύστημα θα έχει την μορφή:

$$A^{(r+1)}x = b^{(r+1)} \tag{37}$$

με

$$A^{(r+1)} = \begin{bmatrix} I_r & * \\ 0 & * \end{bmatrix} \tag{38}$$

όπου * συμβολίζει την ύπαρξη στοιχείων. Η όλη διαδικασία μπορεί να περιγραφεί από τον πολλαπλασιασμό του αρχικού συστήματος με τον πίνακα:

$$M = M^{(n)}M^{(n-1)} \dots M^{(2)}M^{(1)} \tag{39}$$

Με τον πίνακα $M^{(n)}$ να δίνεται ως εξής:

$$M^{(n)} = \left[\begin{array}{ccc|c} & & & \mu_{1,n} \\ & I_{n-1} & & \vdots \\ & & & \mu_{n-1,n} \\ \hline 0 & \cdots & 0 & \mu_{nn} \end{array}\right] \tag{40}$$

Συνεπώς η

$$MA^{(1)}x = Mb^{(1)} \tag{41}$$

δίνει την

$$x = Mb^{(1)} \tag{42}$$

Παρακάτω παρουσιάζεται ο ψευδοκώδικας της μεθόδου απαλοιφής του Jordan:

```
for k = 1 to n - 1 do
    for i = 1 to n do

     if i ≠ k then
        mik = aik / akk;
     end if
     if i = k then
        mik = 1 / akk;
     end if

        for j = k + 1 to n + 1
            aij = aij - mik * akj;
        end for
    end for
end for
```

Η μέθοδος του Jordan μπορεί να εφαρμοστεί σε συνδυασμό με κάποια τεχνική οδήγησης προκειμένου να επιτευχθεί αριθμητική ευστάθεια. Στην περίπτωση της μεθόδου του Jordan παρατηρούμε ότι το ενεργό τμήμα του πίνακα στο k-οστό βήμα της μεθόδου θα είναι ο υποπίνακας που βρίσκεται δεξιά από την k-οστή στήλη του πίνακα A. Στην συνέχεια παρουσιάζεται ο ψευδοκώδικας της μεθόδου του Jordan με χρήση μερικής οδήγησης:

```
for k = 1 to n - 1 do
     for i = 1 to n do
     h(i) = i; //αρχικοποίηση του πίνακα με τις θέσεις των οδηγών στοιχείων
     end for
     Έστω p ο μικρότερος ακέραιος r ≤ p ≤ n και
      |α(h(p),r)| = max_{r≤j≤n}|a(h(j),r)|
     if α(h(p),r) = 0 then
          HALT; //δεν υπάρχει μοναδική λύση
     end if
     if h(r) ≠ h(p) then //εναλλαγή γραμμών
          tmp = h(r);
          h(r) = h(p);
          h(p) = tmp;
     end if

    for i = 1 to n do


     if i ≠ r then
        m(h(i)k) = a(h(i)k)  / a(h(k)k);
     end if
     if i = r then
```

```
        m(h(i)k) = 1 / a(h(k)k);
     end if

        for j = k + 1 to n + 1
            a(h(i)j) = a(h(i)j) - m(h(i)k) * a(h(k)j);
        end for
    end for
end for
```

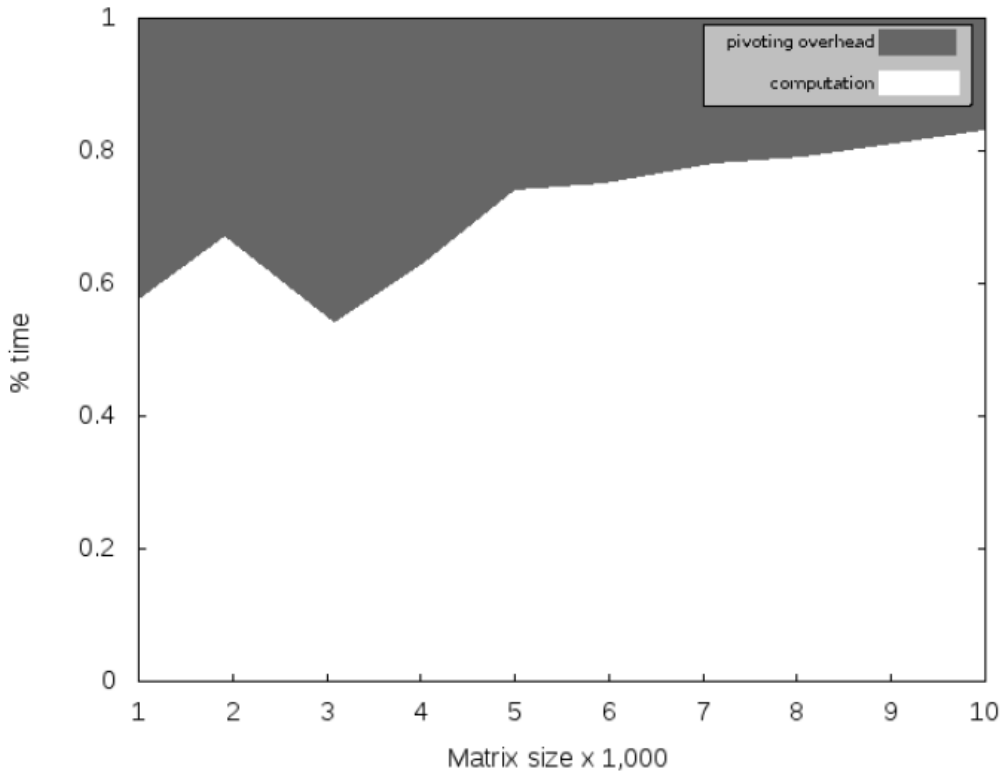
2. Τυχαίος Μετασχηματισμός Πεταλούδας (RBT)

2.1 Εισαγωγή

Σκοπός αυτού του κεφαλαίου είναι η παρουσίαση του τρόπου με τον οποίο μπορεί να επιταχυνθεί η επίλυση ενός γραμμικού συστήματος μέσω τυχαιοποιημένης τεχνικής. Παρουσιάζεται ο τυχαίος μετασχηματισμός πεταλούδας (Random Butterfly transformation (RBT)) κατά τον οποίο το αρχικό γραμμικό σύστημα μετασχηματίζεται σε ένα ενδιάμεσο ισοδύναμο σύστημα, το οποίο μπορεί να επιλυθεί χωρίς την χρήση οδήγησης.

Η διαδικασία αυτή, αν και εισάγει έναν πρόσθετο υπολογιστικό κόστος συμβάλει στον περιορισμό της μετακίνησης δεδομένων που οφείλονται στην διαδικασία της οδήγησης με αποτέλεσμα την αύξηση της ταχύτητας, όταν η μέθοδος εκτελείται σε παράλληλους υψηλής επίδοσης υπολογιστές.

Η υπολογιστική επιβάρυνση που εισάγει η διαδικασία της ολικής οδήγησης κατά την επίλυση ενός γραμμικού συστήματος παρουσιάζεται στο παρακάτω σχήμα [11]:



Σχήμα 3: Ποσοστό συμμετοχής στον υπολογιστικό χρόνο της οδήγησης συναρτήσει της διάστασης του πίνακα. Πηγή: [11].

Στο σχήμα διακρίνεται η συμμετοχή που έχει στον υπολογιστικό χρόνο η διαδικασία της οδήγησης στην περίπτωση της μεθόδου LU σε έναν υβριδικό υπολογιστή με CPU 1 X Quad-Core Intel Core2 Processor Q9300 @ 2.50 GHz και GPU C2050 | 14 Multiprocessors (X 32 CUDA cores) @ 1.15GHz. Παρατηρούμε ότι σε πίνακες τάξης από 1000 έως 3000 η υπολογιστική επιβάρυνση της οδήγησης μπορεί να ξεπεράσει το 40% του συνολικού υπολογιστικού χρόνου, ενώ σε μεγαλύτερες διαστάσεις της τάξης των 10000 η επιβάρυνση περιορίζεται κοντά στο 20% του συνολικού υπολογιστικού χρόνου.

2.2 Εναλλακτικές προσεγγίσεις

Στο παρελθόν, έχουν προταθεί ορισμένες εναλλακτικές προσεγγίσεις για να εξασφαλιστεί το γεγονότος οι πίνακες να βρίσκονται σε μη εκφυλισμένη μορφή (non degenerate), δηλαδή να είναι μη ιδιάζοντες. Οι Bunch και Hopcroft [12] παρατήρησαν ότι οι ιδιάζοντες δοθέντες πίνακες πάντα μπορούν να αντιμετατεθούν έτσι ώστε η απαίτηση αντιστρεψιμότητας που απαιτείται στην μέθοδο απαλοιφής του Gauss να ικανοποιείται. Επίσης, οι Bunch και Hopcroft επισήμαναν (στηριζόμενοι σε ιδέα του Schönhage [13]) ότι κάθε μη ιδιάζων πίνακας A μπορεί να γίνει αναδρομικά μπλοκ μη ιδιάζων (block nonsingular) πολλαπλασιάζοντας με τον συζυγή ανάστροφο (conjugate transpose) A*. Ο τελικός πίνακας A*A θα είναι Ερμιτιανός (Hermitian) και θετικά ορισμένος (positive definite). Με αυτή την προσέγγιση, ο αντίστροφος A^{-1} μπορεί να υπολογίσει ως εξής:

$$A^{+} = (A^{*}A)^{-1}A^{*} \tag{43}$$

Με την προϋπόθεση ότι ο αρχικός πίνακας A είναι μη ιδιάζον, θα ισχύει $A^{-1} = A^{+}$. Η συγκεκριμένη προσέγγιση όχι μόνο είναι σχετικά υπολογιστικά ακριβή, αλλά επιπλέον οδηγεί σε μεγαλύτερη αριθμητική αστάθεια συγκρινόμενη με την μέθοδο απαλοιφής του Gauss με την χρήση οδήγησης. Αυτό οφείλεται στο γεγονός ότι ο αριθμός συνθήκης (condition number) του πίνακα $A^{*}A$ θα είναι πάντα μεγαλύτερος από τον αριθμό συνθήκης του πίνακα A: $\|A^{*}A\|_2 = \|A\|_2^2$, όπου $\|\cdot\|_2$ είναι η φασματική νόρμα πίνακα [14].

Πρόσφατα, οι Baboulin Dogarra και άλλοι [15] [16], επηρεασμένοι από τις δημοσιεύσεις του Parker και άλλοι [17] [18] [19], έχουν παρουσιάσει μια σειρά εργασιών για την εφαρμογή τόσο της μεθόδου RBT σε παράλληλες και υβριδικές (CPU / GPU) αρχιτεκτονικές [11] [20] , όσο και για μεθόδους περιορισμού της χρήσης οδήγησης στην επίλυση συμμετρικών και πυκνών γραμμικών συστημάτων.

2.3 Βασικές έννοιες

Σε αυτήν την ενότητα, θα εξετάσουμε ορισμένες έννοιες που είναι χρήσιμες για την κατανόηση της εργασίας.

2.3.1 Υποπίνακες

Θα λέμε ότι ο $A[i_1, \dots, i_p \,|j_1, \dots, j_q]$ είναι ένας υποπίνακας (submatrix) του πίνακα A που αποτελείται από τις γραμμές $i_1, \dots, i_p$ και από τις στήλες $j_1, \dots, j_q$ του πίνακα A, και

σχηματίζεται ως εξής (θεωρούμε ότι η αρίθμηση των στοιχείων του πίνακα A ξεκινάει από το 1):

$$A[i_1, \dots, i_p \,|j_1, \dots, j_q] = \begin{pmatrix} \alpha_{i_1,j_1} & \alpha_{i_1,j_2} & \cdots & \alpha_{i_1,j_q} \\ \alpha_{i_2,j_1} & \alpha_{i_2,j_2} & \cdots & \alpha_{i_2,j_q} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{i_p,j_1} & \alpha_{i_p,j_2} & \dots & \alpha_{i_p,j_q} \end{pmatrix} \tag{44}$$

Για παράδειγμα ο υποπίνακας του πίνακα A με γραμμές τις 3,2 και στήλες τις 1,3,1,4 είναι ο ακόλουθος:

$$A[3,2\,|1,3,1,4] = \begin{pmatrix} \alpha_{31} & \alpha_{33} & \alpha_{31} & \alpha_{34} \\ \alpha_{21} & \alpha_{23} & \alpha_{21} & \alpha_{24} \end{pmatrix} \tag{45}$$

Θεωρούμε ότι για p = 0 και q = 0, ο υποπίνακας θα είναι ίσος με 1, δηλαδή A[|] = 1.

2.3.2 Μοναδιακοί μετασχηματισμοί

Ένας μιγαδικός τετραγωνικός πίνακας U λέγεται μοναδιακός (unitary) όταν ο Hermitian adjoint (conjugate transpose) U* είναι και ο αντίστροφος του, δηλαδή ισχύει: UU* = U*U = I.

Ένας μοναδιακός μετασχηματισμός πάνω σε έναν τετραγωνικό πίνακα A είναι ένα οποιοδήποτε γινόμενο U*AV, όπου οι πίνακες U και V είναι μοναδιακοί.

Οι μοναδιακοί μετασχηματισμοί έχουν ορισμένες χρήσιμες ιδιότητες:

Οι μοναδιακοί μετασχηματισμοί μπορούν εύκολα να αντιστραφούν.
Οι ακόλουθες πράξεις όταν εφαρμόζονται πάνω σε μοναδιακούς πίνακες είναι κλειστές:
Γινόμενο: Αν U,V δυο μοναδιακοί πίνακες, τότε το γινόμενο τους UV είναι κι αυτό μοναδιακός πίνακας.
Ευθύ άθροισμα (direct sum): Αν U,V δυο μοναδιακοί πίνακες διάστασης $n \times n$, τότε το ευθύ άθροισμα τους:

$$U \oplus V = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \tag{46}$$

Θα είναι ένας μοναδιακός πίνακας διάστασης $2n \times 2n$.

Γινόμενο του Kronecker: Αν U,V δυο μοναδιακοί πίνακες διάστασης $m \times m$ και $n \times n$, τότε το γινόμενο του Kronecker [21]:

$$U \otimes V = \begin{pmatrix} u_{11}V & u_{12}V & \dots & u_{1m}V \\ u_{21}V & u_{22}V & \dots & u_{2m}V \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}V & u_{m2}V & \dots & u_{mm}V \end{pmatrix} \tag{47}$$

Είναι ένας μοναδιακός πίνακας διάστασης $mn \times mn$:

$$(U \otimes V)^* = U^* \otimes V^* = U^{-1} \otimes V^{-1} = (U \otimes V)^{-1} \tag{48}$$

Οι μοναδιακοί μετασχηματισμοί διατηρούν την νόρμα και τον αριθμό συνθήκης (condition number) του πίνακα. Το αντίστοιχο της νόρμας ενός μοναδιακού πίνακα $\|\cdot\|$ είναι μια νόρμα που ικανοποιεί την συνθήκη: $\|UAV\| = \|A\|$ για κάθε μοναδιακό πίνακα U, V και για κάθε τετραγωνικό πίνακα A [22]. Υπάρχουν πολλά είδη νόρμας πίνακα αλλά θα επισημάνουμε την φασματική νόρμα (spectral norm):

$$\|A\|_2 = \sigma_1(A) = max\{\sqrt{\lambda} \mid \lambda \, \iota\delta\iota o\tau\iota\mu\acute{\eta} \, \tau o\upsilon \, A^*A\} \tag{49}$$

Όπου $\sigma_i(A)$ είναι η i-οστή ιδιάζουσα τιμή (singular value).

Χρησιμοποιώντας την φασματική νόρμα, ο αριθμός συνθήκης ενός τετραγωνικού μη ιδιάζοντα πίνακα A είναι:

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_1(A)}{\sigma_n(A)} \ \mu\varepsilon\ \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \qquad (50)$$

Οι μοναδιακοί μετασχηματισμοί παρουσιάζουν αριθμητική ευστάθεια. Όχι μόνο δεν μεταβάλουν τον αριθμό συνθήκης των πινάκων, αλλά και προσφέρουν εύκολη ανάλυση σφάλματος. Έστω ότι ο πίνακας A έχει ένα σφάλμα E, τότε ο μοναδιακός μετασχηματισμός με τους πίνακες U και V θα ικανοποιεί την συνθήκη:

$$U(A+E)V = UAV + UEV = UAV + F \qquad (51)$$

Και το συνολικό σφάλμα F θα έχει νόρμα $\|F\| = \|E\|$ [23].

2.3.3 Πίνακες Hadamard – Walsh

Ένας πίνακας με στοιχεία 1 ή -1, οι γραμμές του οποίου είναι αμοιβαία ορθογώνιες ονομάζεται πίνακας Hadamard προς τιμήν του Γάλλου μαθηματικού Jacques Hadamard [24], [25].

Ένας Hadamard πίνακας H τάξης n ικανοποιεί την σχέση $HH^T = nI_n$ (όπου H^T ο ανάστροφος και I_n ο μοναδιαίος πίνακας), και παρατηρούμε ότι παρομοιάζει με την ιδιότητα του αντίστροφου. Αυτή η κατηγορία πινάκων έχει εφαρμογές στην θεωρία κωδίκων για διόρθωση σφαλμάτων και στην στατιστική.

Η ορίζουσα ενός πίνακα Hadamard H υπολογίζεται με τον τύπο: $\det(\mathrm{H}) = \pm n^{\frac{n}{2}}$

Η πρώτη κατασκευή έγινε από τον Sylvester το 1987 [26] για πίνακες τάξης 2^κ, όπου κ θετικός ακέραιος. Έστω H ένας Hadamard πίνακας τάξης n, τότε ένας Hadamard πίνακας τάξης 2^n σχηματίζεται ως εξής:

$$\begin{pmatrix} H & H \\ H & -H \end{pmatrix} \qquad (52)$$

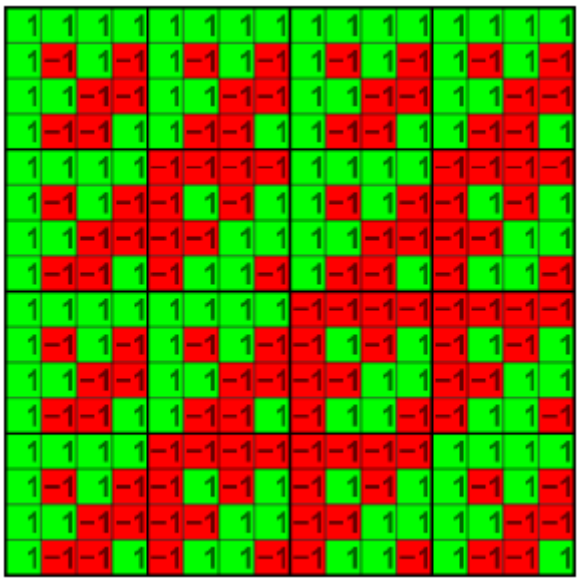
Για n = 1,2 έχουμε:

$$H_1 = (1)\,, \qquad H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad (53)$$

Γενικά:

$$H_{2^\kappa} = \begin{pmatrix} H_{2^{\kappa-1}} & H_{2^{\kappa-1}} \\ H_{2^{\kappa-1}} & -H_{2^{\kappa-1}} \end{pmatrix} = H_2 \otimes H_{2^{\kappa-1}} \qquad (54)$$

όπου $2 \leq \kappa \in N$, και $\otimes$ το γινόμενο του Kronecker. Αυτή η ακολουθία πινάκων ονομάζεται επίσης και πίνακες Walsh [27].



Εικόνα 1: πίνακας Walsh ή πίνακας Hadamard τάξης 16 [28]

Οι πίνακες του Sylvester έχουν ορισμένες χαρακτηριστικές ιδιότητες, όπως το ότι είναι συμμετρικοί, έχουν ίχνος ίσο με το μηδέν και τα στοιχεία στην πρώτη σειρά και στην πρώτη στήλη είναι πάντα θετικά.

2.4 Βασική ιδέα

Η βασική ιδέα για τον τυχαίο μετασχηματισμό πεταλούδας (RBT) είναι η αποφυγή της οδήγησης κατά την επίλυση ενός γραμμικού συστήματος με την χρήση της μεθόδου απαλοιφής του Gauss και η «αντικατάσταση» της από μια τυχαιοποιημένη τεχνική [17] [18].

Θεωρούμε τον μη ιδιάζοντα (nonsingular) πραγματικό πίνακα A διάστασης $n \times n$ και τους τυχαίους μη ιδιάζοντες πίνακες U και V διάστασης $n \times n$. Για να εξασφαλίσουμε ότι πίνακες U και V θα είναι μη ιδιάζοντες, θα τους επιλέξουμε από μια συγκεκριμένη κατηγορία πινάκων, ενώ τα στοιχεία τους μπορούν να επιλεγούν από μια κατάλληλη τυχαία κατανομή. Τότε μπορούμε να εφαρμόσουμε την μέθοδο απαλοιφής του Gauss πάνω στο γινόμενο UAV χωρίς να απαιτείται η χρήση οδήγησης, αφού ο πίνακας UAV θα είναι «κατάλληλα τυχαίος».

Για να επιλύσουμε το ζητούμενο γραμμικό σύστημα Ax = b, λύνουμε το UAVy = Ub και στην συνέχεια υπολογίζουμε τον ζητούμενο πίνακα των αγνώστων, ως x = Vy.

Υποθέτοντας ότι οι πίνακες είναι μη ιδιάζοντες, έπεται ότι μπορεί να υπολογιστεί ο $(UAV)^{-1}$ και συνεπώς, ο αντίστροφος του A (A^{-1}) θα είναι ο $V(UAV)^{-1}U$.

Η ορίζουσα του πίνακα A μπορεί να υπολογιστεί υπολογίζοντας την ορίζουσα: det(UAV)/(detU detV).

2.4.1 Αριθμητικό παράδειγμα

Για την απλούστερη κατανόηση του τυχαίου μετασχηματισμού RBT θεωρούμε τον 2×2 πίνακα A και τον 2×1 πίνακα b:

$$A = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Σκοπός μας είναι η επίλυση του γραμμικού συστήματος Ax = b και προφανώς σε αυτή την περίπτωση ο πίνακας x των αγνώστων θα είναι ο ακόλουθος:

$$x = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Τώρα, επιλέγουμε τυχαία τους πίνακες U και V:

$$U = \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix}, \qquad V = \begin{pmatrix} +.7279 & +.6857 \\ +.6857 & -.7279 \end{pmatrix}$$

και υπολογίζουμε τον τυχαίο μετασχηματισμό:

$$UAV = \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix}\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} +.7279 & +.6857 \\ +.6857 & -.7279 \end{pmatrix} = \begin{pmatrix} 1.444 & -.4220 \\ .5721 & -1.554 \end{pmatrix}$$

και το γινόμενο Ub:

$$Ub = \begin{pmatrix} +.6483 & +.7614 \\ +.7614 & -.6483 \end{pmatrix}\begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 3.581 \\ -.4220 \end{pmatrix}$$

Στη συνέχεια μπορούμε να επιλύσουμε το γραμμικό σύστημα UAVy = Ub χωρίς να εφαρμόσουμε οδήγηση. Ο πίνακας των αγνώστων y θα είναι ο ακόλουθος:

$$y = \begin{pmatrix} 2.868 \\ 1.327 \end{pmatrix}$$

Τέλος, υπολογίζουμε τον πίνακα των αγνώστων x του αρχικού γραμμικού συστήματος:

$$x = Vy = \begin{pmatrix} 2.998 \\ 1.001 \end{pmatrix}$$

Παρατηρούμε, ότι ο πίνακας x είναι ακριβής στα πρώτα 3 σημαντικά ψηφία.

2.5 Ιδιότητες του μετασχηματισμού RBT

Για την εφαρμογή του μετασχηματισμού RBT θα πρέπει να εξασφαλίστεί ότι ο μετασχηματισμός θα πληροί ορισμένες προϋποθέσεις ποιοτικές (αριθμητική ακρίβεια) και ποσοτικές (υπολογιστικό κόστος). Πιο αναλυτικά, το αποτέλεσμα του RBT μετασχηματισμού θα πρέπει είναι τέτοιο ώστε να μπορεί να εφαρμοστεί σε αυτό η μέθοδος απαλοιφής του Gauss. Αν A πίνακας διάστασης $n \times n$ τότε, για να εφαρμοστεί η μέθοδος απαλοιφής του Gauss θα πρέπει να ισχύει:

$$\det A[1, \dots, k|1, \dots k] \neq 0, \;\; 1 \leq k \leq n \qquad (55)$$

Επίσης η υπολογιστική επιβάρυνση που εισάγει ο μετασχηματισμός θα πρέπει να έχει ένα αποδεκτό υπολογιστικό κόστος. Εφαρμόζοντας τον μετασχηματισμό στον πίνακα A, εισάγουμε μερικούς διαδοχικούς πολλαπλασιασμούς πινάκων (UAV). Αντίστοιχα, κατά την αντιστροφή του μετασχηματισμού (Vy) εμφανίζεται πάλι ένας πολλαπλασιασμός πινάκων. Γίνεται αντιληπτό, ότι στην περίπτωση συστημάτων μεγάλης διάστασης, ο μετασχηματισμός εισάγει ένα σημαντικό υπολογιστικό κόστος. Συνεπώς, πρέπει να

επιλέξουμε τους τυχαίους πίνακες από μια συγκεκριμένη κατηγορία πινάκων, έτσι ώστε σε ορισμένες υλοποιήσεις να δικαιολογούν την πρόσθετη αυτή επιβάρυνση.

Τέλος, ο μετασχηματισμός θα πρέπει να εισάγει το μικρότερο δυνατό σφάλμα κατά την στρογγυλοποίηση. Η μέθοδος απαλοιφής του Gauss χωρίς οδήγηση σε πίνακες με μεγάλο αριθμό συνθήκης εισάγει μεγάλο σφάλμα στρογγύλευσης (roundoff error) [29] , κάτι το οποίο σημαίνει αυξημένη αριθμητική αστάθεια.

2.6 Αναδρομικοί Πίνακες Πεταλούδας

Για να υλοποιήσουμε τον RBT μετασχηματισμό υπολογιστικά αποδοτικά, θα ορίσουμε την κλάση «πινάκων πεταλούδας» (butterfly matrices).

Ο πίνακας $B^{<n>}$ διαστάσεων $n \times n$ θα ονομάζεται «πίνακας πεταλούδα» (butterfly matrix) αν έχει την μορφή:

$$B^{<n>} = \frac{1}{\sqrt{2}} \begin{pmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{pmatrix} \tag{56}$$

Όπου n ≥ 2, R_0 και R_1 διαγώνιοι και μη ιδιάζοντες πίνακες διάστασης $(\frac{n}{2})$ X $(\frac{n}{2})$. Όταν οι R_0 και R_1 είναι μοναδιακοί (έχουν την μορφή $\exp(i\Theta), \Theta \in \mathbb{R}$) τότε και ο $B^{<n>}$ θα είναι μοναδιακός. Στην ειδική περίπτωση για $\Theta = 0$, ο $B^{<n>}$ γίνεται:

$$O_n = \frac{1}{\sqrt{2}} \begin{pmatrix} I_{n/2} & I_{n/2} \\ I_{n/2} & -I_{n/2} \end{pmatrix} \tag{57}$$

Ισοδύναμα $B^{<n>} = O_n \mathrm{R}$, όπου $R = (R_0 \oplus R_1)$ είναι διαγώνιος πίνακας.

Ένας αναδρομικός πίνακας πεταλούδα (recursive butterfly matrix) είναι το γινόμενο επιμέρους «πινάκων πεταλούδας». Ένας «αναδρομικός πίνακας πεταλούδα» διάστασης 1×1 γράφεται σαν $U^{<1>}$. Ένας αναδρομικός πίνακας πεταλούδα $U^{<n>}$ διάστασης $n \times n$ όπου n = 2^v είναι το γινόμενο του ευθέως αθροίσματος δυο μικρότερων αναδρομικών πινάκων πεταλούδας $U_0^{<n/2>}$ και $U_1^{<n/2>}$ με έναν επίσης $n \times n$ πίνακα πεταλούδα $B^{<n>}$:

$$U^{<n>} = \left(U_0^{<\frac{n}{2}>} \oplus U_1^{<\frac{n}{2}>} \right) B^{<n>} = \begin{pmatrix} U_0^{<\frac{n}{2}>} & 0 \\ 0 & U_1^{<\frac{n}{2}>} \end{pmatrix} B^{<n>} \tag{58}$$

Για παράδειγμα, αν n = 4, τότε ο αναδρομικός πίνακας πεταλούδα $B^{<4>}$ διαμορφώνεται ως εξής:

$$\begin{aligned} U^{<4>} &= (U_0^{<2>} \oplus U_1^{<2>}) B^{<4>} \\ &= \left(\left((U_{00}^{<1>} \oplus U_{01}^{<1>}) B_0^{<2>} \right) \oplus \left((U_{10}^{<1>} \oplus U_{11}^{<1>}) B_1^{<2>} \right) \right) B^{<4>} \\ &= (U_{00}^{<1>} \oplus U_{01}^{<1>} \oplus U_{10}^{<1>} \oplus U_{11}^{<1>}) \, (B_0^{<2>} \oplus B_1^{<2>}) \, B^{<4>} \end{aligned} \tag{59}$$

Εφαρμόζοντας το ευθύ άθροισμα και αναλύοντας τους επιμέρους πίνακες πεταλούδας, ο $U^{<4>}$ διαμορφώνεται ως εξής:

$$U^{<4>} = (U_{00}^{<1>} \oplus U_{01}^{<1>} \oplus U_{10}^{<1>} \oplus U_{11}^{<1>}) \, (B_0^{<2>} \oplus B_1^{<2>}) \, B^{<4>} =$$

$$= \frac{1}{2}\begin{pmatrix} r_1^{<1>} & & & \\ & r_2^{<1>} & & \\ & & r_3^{<1>} & \\ & & & r_4^{<1>} \end{pmatrix}\begin{pmatrix} r_1^{<2>} & r_2^{<2>} & & \\ r_1^{<2>} & -r_2^{<2>} & & \\ & & r_3^{<2>} & r_4^{<2>} \\ & & r_3^{<2>} & -r_4^{<2>} \end{pmatrix}\begin{pmatrix} r_1^{<4>} & & r_3^{<4>} & \\ & r_2^{<4>} & & r_4^{<4>} \\ r_1^{<4>} & & -r_3^{<4>} & \\ & r_2^{<4>} & & -r_4^{<4>} \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} r_1^{<1>}r_1^{<2>}r_1^{<4>} & r_1^{<1>}r_2^{<2>}r_2^{<4>} & r_1^{<1>}r_1^{<2>}r_3^{<4>} & r_1^{<1>}r_2^{<2>}r_4^{<4>} \\ r_2^{<1>}r_1^{<2>}r_1^{<4>} & -r_2^{<1>}r_2^{<2>}r_2^{<4>} & r_2^{<1>}r_1^{<2>}r_3^{<4>} & -r_2^{<1>}r_2^{<2>}r_4^{<4>} \\ r_3^{<1>}r_3^{<2>}r_1^{<4>} & r_3^{<1>}r_4^{<2>}r_2^{<4>} & -r_3^{<1>}r_3^{<2>}r_3^{<4>} & -r_3^{<1>}r_4^{<2>}r_4^{<4>} \\ r_4^{<1>}r_3^{<2>}r_1^{<4>} & -r_4^{<1>}r_4^{<2>}r_2^{<4>} & -r_4^{<1>}r_3^{<2>}r_3^{<4>} & r_4^{<1>}r_4^{<2>}r_4^{<4>} \end{pmatrix} \quad (60)$$

Τα στοιχεία του $U^{<n>}$ πίνακα δίνονται από την παρακάτω σχέση:

Έστω $U^{<n>}$ ένας αναδρομικός πίνακας πεταλούδα, όπου $n = 2^v$, τότε τα στοιχεία του $U^{<n>} = (u_{ij})$ δίνονται ως εξής:

$$u_{ij} = \pm \frac{1}{\sqrt{n}} \prod_{l=0}^{v} r_{f(i,j,l)}^{<2^l>} \quad (61)$$

Όπου $f(i,j,l)$ είναι μια συνάρτηση του l και της δυαδικής αναπαράστασης του $(i-1)$ και του $(j-1)$.

Απόδειξη [17]**:** Παρατηρούμε ότι ο $U^{<n>}$ είναι το γινόμενο ενός διαγώνιου πίνακα και v ευθέως αθροισμάτων (direct sums) πινάκων πεταλούδας, οπότε μπορεί να γραφεί ως εξής:

$$u_{ij} = r_{ii}^1 \sum k_1 \sum k_2 \ \dots \sum k_{v-1}\, b_{i,k_1}^1 b_{k_1,k_2}^2 \ \dots \ b_{k_{v-1},j}^v \quad (62)$$

Όπου b_{pq}^l είναι τα στοιχεία του $l-οστού$ ευθύ αθροίσματος πινάκων πεταλούδας. Για να ισχύει η σχέση, πρέπει το $b_{pq}^l \neq 0$ το οποίο ισχύει αν και μόνο αν $(p-1)$ και $(q-1)$ είναι ίσα ή διαφέρουν στο τελευταίο ψηφίο της δυαδικής αναπαράστασης. Σε αυτή την περίπτωση έχουμε ότι:

$$b_{pq}^l = \pm \frac{1}{\sqrt{2}} r_q^{<2^l>} \ \ \gamma\iota\alpha \ \ l \geq 1 \quad (63)$$

Αν $n = 2^v$ και οι πεταλούδες $B^{<n>}$ είναι πάντα της μορφής O_n τότε παίρνουμε έναν αναδρομικό πίνακα Hadamard ως εξής:

$$H_n = \left(\frac{1}{\sqrt{2}}\right)^v \overset{v}{\otimes} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{n}} \overset{log_2(n)}{\otimes} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (64)$$

Ο πίνακας Hadamard δεν χρειάζεται να πολλαπλασιαστεί με τον παράγοντα $\frac{1}{\sqrt{n}}$ όπως γίνεται εδώ. Ο πολλαπλασιασμός γίνεται για να εξασφαλιστεί ότι ο πίνακας θα είναι μοναδιακός [17].

2.7 Ο Τυχαίος Μετασχηματισμός Πεταλούδας (RBT)

Ορισμός: θα λέμε ότι ο πίνακας A είναι ένα τυχαίος πίνακας (random matrix) εάν οι τιμές των στοιχείων του είναι συναρτήσεις τυχαίων μεταβλητών που ακολουθούν μια κοινή τυχαία κατανομή.

Ορισμός: θα ονομάζουμε Τυχαίο Μετασχηματισμό Πεταλούδας (RBT) ενός τετραγωνικού πίνακα A, το γινόμενο:

$$\tilde{A} = U^* A V \tag{65}$$

όπου U και V είναι τυχαίοι αναδρομικοί πίνακες πεταλούδας, δηλαδή αναδρομικοί πίνακες πεταλούδας οι οποίοι παρήχθησαν από τους τυχαίους αρχικούς διαγώνιους πίνακες R_0 και R_1 [19].

Στη συνέχεια θα δείξουμε ότι οι πίνακες που παράγονται από τον μετασχηματισμό RBT είναι σε μη εκφυλισμένη (μη ιδιάζοντες) μορφή.

Ορισμός: S_k^m είναι το σύνολο των γνησίως αυξουσών ακολουθιών μεγέθους k από το $\{1, \dots, m\}$. Για παράδειγμα, $1,2,5 \in S_3^5$ αλλά $2,2,5 \notin S_3^5$, και $1,2,5 \in S_3^6$.

Ορισμός: CS_k^m είναι το σύνολο των διαδοχικών γνησίως αυξουσών ακολουθιών στο S_k^m. Για παράδειγμα, $1,2,5 \in CS_3^5$ αλλά $1,3,5 \notin S_3^5$, και $1,2,5 \notin S_3^6$.

Ορισμός: Ένας πίνακας A διάστασης $n \times n$ θα λέμε ότι είναι μπλοκ μη εκφυλισμένος (block nondegenerate) εάν για κάθε $1 \le \mathrm{k} \le \mathrm{n}$ και για κάθε ακολουθία διαδοχικών ακεραίων $\alpha \in \mathrm{CS}_\mathrm{k}^\mathrm{n}$, ο κύριος μπλοκ υποπίνακας (block principal submatrix) $\mathrm{A}[\alpha|\alpha]$ είναι μη ιδιάζων.

Θεώρημα: Έστω A μη ιδιάζον πίνακας διάστασης $n \times n$, όπου ο n είναι δύναμη του 2. Εάν U και V είναι τυχαίοι αναδρομικοί πίνακες πεταλούδας, τότε με πιθανότητα 1, ο μετασχηματισμός RBT $\widetilde{\mathrm{A}} = \mathrm{U}^*\mathrm{AV}$ είναι μπλοκ μη εκφυλισμένος.

Απόδειξη: Έστω ότι ο $\widetilde{\mathrm{A}}$ είναι μπλοκ εκφυλισμένος. Συνεπώς για κάποιο k και για κάποια ακολουθία $\alpha \in \mathrm{CS}_\mathrm{k}^\mathrm{n}$ θα ισχύει ότι $\det\big(\widetilde{\mathrm{A}}[\alpha|\alpha]\big) = 0$. Όμως, χρησιμοποιώντας το θεώρημα Binet-Cauchy [30], για την ορίζουσα του $\widetilde{\mathrm{A}} = \mathrm{U}^*\mathrm{AV}$ θα ισχύει:

$$\begin{aligned}\det\big(\widetilde{\mathrm{A}}[\alpha|\alpha]\big) &= \sum_{\mathrm{k} \in \mathrm{S}_\mathrm{k}^\mathrm{n}} \sum_{\lambda \in \mathrm{S}_\mathrm{k}^\mathrm{n}} \det(\mathrm{U}^*[\alpha|\mathrm{k}])\det(\mathrm{A}[\kappa|\lambda])\det(\mathrm{V}[\lambda|\alpha]) \\ &= \sum_{\mathrm{k} \in \mathrm{S}_\mathrm{k}^\mathrm{n}} \sum_{\lambda \in \mathrm{S}_\mathrm{k}^\mathrm{n}} \overline{\det(\mathrm{U}[\mathrm{k}|\alpha])}\det(\mathrm{A}[\kappa|\lambda])\det(\mathrm{V}[\lambda|\alpha])\end{aligned} \tag{66}$$

όπου ο συμβολισμός πάνω μπάρα, συμβολίζει τον μιγαδικό συζυγή (complex conjugate) και το $\mathrm{S}_\mathrm{k}^\mathrm{n}$ συμβολίζει μια γνησίως αύξουσα ακολουθία μήκους κ με στοιχεία {1,…,n}. Επειδή η α είναι μια διαδοχική ακολουθία, από το θεώρημα 3 στην εργασία του Parker [17] προκύπτει ότι η ορίζουσα $\det(\mathrm{U}[\mathrm{k}|\alpha])$ είναι ένα μη μηδενικό πολυώνυμο βαθμού το πολύ ένα. Αντίστοιχα, η ορίζουσα $\det(\mathrm{V}[\lambda|\alpha])$ είναι μη μηδενική. Συνεπώς, η ορίζουσα $\det\big(\widetilde{\mathrm{A}}[\alpha|\alpha]\big)$ είναι πολυώνυμο βαθμού ένα.

Από την υπόθεση, είναι γνωστό ότι $\det\big(\widetilde{\mathrm{A}}[\alpha|\alpha]\big) = 0$. Από το θεώρημα 1 στην εργασία του Parker [17], συμπεραίνουμε ότι με πιθανότητα ένα είναι $\det(\mathrm{A}[\kappa|\lambda]) = 0$ και θα ισχύει για κάθε επιλογή των k και λ στην ακολουθία $\mathrm{S}_\mathrm{k}^\mathrm{n}$. Ωστόσο, για κάθε ακολουθία

$k \in S_k^n$ μπορούμε να εφαρμόσουμε το γενικό θεώρημα αναπτύγματος ορίζουσας του Laplace [30]:

$$\det A = (-1)^{sum(k)} \sum_{\lambda \in S_k^n} (-1)^{sum(\lambda)} \det(A[\kappa|\lambda]) \det(A[k'|\lambda']) \qquad (67)$$

Όπου k' είναι η γνησίως αύξουσα συμπληρωματική της k μεταξύ των δεικτών του A και $sum(k) = \sum_i k_i$. Άρα σύμφωνα με την υπόθεση το δεξί μέρος της εξίσωσης είναι ίσο με μηδέν, με πιθανότητα 1. Αυτό είναι άτοπο, αφού ο πίνακας A είναι μη ιδιάζον. Συνεπώς, με πιθανότητα 1 η υπόθεση ότι ο $\tilde{A}$ είναι μπλοκ εκφυλισμένος δεν ισχύει. Επομένως, με πιθανότητα ένα, ο πίνακας $\tilde{A}$ είναι μπλοκ μη εκφυλισμένος.

2.8 Εφαρμογή του μετασχηματισμού RBT στην μέθοδο απαλοιφής του Gauss

Σε αυτό το σημείο, θα αναπτύξουμε το κυρίως θέμα της εργασίας: με ποιον τρόπο θα χρησιμοποιήσουμε την μέθοδο του τυχαίου μετασχηματισμού πεταλούδας (RBT) με σκοπό να επιταχύνουμε την επίλυση γραμμικών συστημάτων. Όπως έχουμε ήδη αναφέρει από την εισαγωγή της εργασίας, αυτό επιτυγχάνεται αν εφαρμόσουμε τον μετασχηματισμό RBT κατά την προετοιμασία των δεδομένων και στην συνέχεια επιλύοντας ένα ενδιάμεσο γραμμικό σύστημα με την μέθοδο απαλοιφής του Gauss (δεν περιορίζεται αποκλειστικά στην συγκεκριμένη μέθοδο) χωρίς την χρήση της οδήγησης, η οποία εισάγει μεγάλη υπολογιστική επιβάρυνση λόγω της αυξημένης μεταφοράς δεδομένων που απαιτεί.

Εύλογα, γεννιέται το ερώτημα αν ο ενδιάμεσος πίνακας που προκύπτει από τον μετασχηματισμό RBT είναι σε κατάλληλη μορφή έτσι ώστε να εφαρμοστεί σε αυτόν η μέθοδος απαλοιφής του Gauss χωρίς την χρήση οδήγησης. Για να συμβεί αυτό, θα πρέπει τα στοιχεία στην διαγώνιο του πίνακα να είναι όλα μη μηδενικά. Αυτό ισοδυναμεί με τον εξής ορισμό [29] [17]:

Ορισμός: Ένας πίνακας A διάστασης $n \times n$ είναι Gauss απαλείψιμος (Gauss eliminable) αν ισχύει: $\det A[1, \dots, k|1, \dots, k] \neq 0$ για $1 \leq k \leq n$.

Θεώρημα: Ο μετασχηματισμός RBT παράγει πίνακες Gauss απαλείψιμους, δηλαδή με πιθανότητα 1 η μέθοδος απαλοιφής του Gauss μπορεί να εφαρμοστεί στο αποτέλεσμα $\tilde{A} = U^* A V$ του μετασχηματισμού RBT χωρίς την χρήση οδήγησης.

Απόδειξη: Στην προηγούμενη ενότητα δείξαμε ότι ο πίνακας $\tilde{A}$ που προκύπτει από τον RBT είναι μπλοκ μη εκφυλισμένος. Όμως, όταν ένας πίνακας είναι μπλοκ μη εκφυλισμένος συνεπάγεται ότι είναι και Gauss απαλείψιμος [17], [18], [31]. Συνεπώς ο $\tilde{A}$ είναι Gauss απαλείψιμος.

2.8.1 Ανάλυση σφάλματος

Από την κλασσική ανάλυση της μεθόδου απαλοιφής του Gauss προκύπτει ότι η μέθοδος δημιουργεί τους τριγωνικούς πίνακες $\hat{L}$ και $\hat{U}$, τέτοιους ώστε $A = \hat{L}\hat{U} + E$ όπου $E = \sum_{k=1}^{n-1} E^{(k)}$ είναι ο πίνακας των αριθμητικών σφαλμάτων τέτοια ώστε $E^{(k)} = (\epsilon_{ij}^{(k)})$ και:

$$\left|\epsilon_{ij}^{(k)}\right| = \begin{cases} \left|\hat{\alpha}_{ij}^{(k)}\right| u & i \geq k+1, \quad j = k \\ \left|\hat{\alpha}_{ij}^{(k+1)}\right| 2u + \left|\hat{\alpha}_{ij}^{(k)}\right| u & i \geq k+1, \quad j \geq k+1 \\ 0 & διαφορετικά \end{cases} \tag{68}$$

Όπου u είναι η μονάδα σφάλματος στρογγύλευσης (unit roundoff error) που προκύπτει από την πεπερασμένη αριθμητική ακρίβεια των υπολογιστών. Η τιμή την οποία αποθηκεύει και επεξεργάζεται ο υπολογιστής διαφέρει από την πραγματική εξαιτίας της αναπαράστασης του σε μορφή αριθμού κινητής υποδιαστολής (float-point number). Αξίζει να σημειώσουμε ότι το σφάλμα στρογγύλευσης δεν είναι το μοναδικό σφάλμα το οποίο εμφανίζεται σε μια αριθμητική μέθοδο που επιλύεται σε έναν υπολογιστή. Τα σφάλματα στρογγύλευσης εμφανίζονται κατά την διάρκεια στην οποία ο υπολογιστής εκτελεί τις λοιπές αριθμητικές πράξεις με τους αριθμούς κινητής υποδιαστολής. [32], [7], [33].

Ο αλγόριθμος της απαλοιφής του Gauss μετά από μια επαναληπτική διαδικασία n-1 βημάτων μετασχηματίζει τον αρχικό πίνακα $A = A^{(1)}$ στον τελικό άνω τριγωνικό πίνακα $U = A^{(n)}$. Ένα φράγμα του ολικού σφάλματος στρογγύλευσης $|E|$ δίνεται σαν φράγμα του $\left|\hat{\alpha}_{ij}^{(k+1)}\right|$. Αυτό το φράγμα μπορούμε να το εκφράσουμε με την έννοια του συντελεστή αύξησης (growth factor) $g(A)$ ως εξής [29] [17] [18]:

$$g(A) = \frac{max_{ijk}\left|\hat{\alpha}_{ij}^{(k)}\right|}{\|A\|} \tag{69}$$

Για τον συντελεστή αύξησης είναι γνωστά ασθενή φράγματα τόσο στην περίπτωση της ολικής όσο και στην περίπτωση της μερικής οδήγησης. Στην περίπτωση της μερικής οδήγησης αυξάνει εκθετικά ως προς την τάξη (rank) του πίνακα Α. Γενικά, αν ένα γραμμικό σύστημα επιλυθεί με την μέθοδο απαλοιφής του Gauss χωρίς την χρήση οδήγησης, με μεγάλη πιθανότητα θα είναι ασταθές. Στην πράξη η μέθοδος απαλοιφής του Gauss με την χρήση ολικής ή και η μερικής οδήγηση θα είναι ευσταθής.

Η τιμή στην θέση (i,j) $\hat{\alpha}_{ij}^{(k)}$ του πίνακα στον οποίο εφαρμόζεται η μέθοδος απαλοιφής του Gauss στο k-οστό βήμα, μπορεί να προσεγγιστεί με τον $a_{ij}^{(k)}$ ως εξής:

Θεώρημα: Εάν ο Α είναι ένας πίνακας διάστασης $n \times n$ Gauss απαλείψιμος, τότε για $1 \leq k < i \leq n$, $1 \leq k < j \leq n$ ισχύει [29]:

$$a_{ij}^{(k)} = \frac{\det A[1, \dots, k, i \,|1, \dots, k, j]}{\det A[1, \dots, k \mid 1, \dots, k]} \tag{70}$$

Προκύπτει το ακόλουθο σφικτό φράγμα (tight bound):

Θεώρημα: Για ισχύει $1 \leq k < i \leq n$, $1 \leq k < j \leq n$ [29]:

$$\alpha_{ij}^{(k+1)} = \alpha_{ij} - A[i \,|1, \dots, k](A[1, \dots, k \mid 1, \dots, k])^{-1} A[1, \dots, k \mid j] \tag{71}$$

Επομένως,

$$\left|\alpha_{ij}^{(k+1)}\right| \leq \left|\alpha_{ij}\right| + k_2(A[1,\dots,k \mid 1,\dots,k]) \frac{\|A[i \mid 1,\dots,k]\|_2 \, \|A[1,\dots,k \mid j]\|_2}{\|A[1,\dots,k \mid 1,\dots,k]\|_2} \quad (72)$$

όπου $k_2(A)$ ο αριθμός συνθήκης του πίνακα A ως προς την $\| \ \|_2$. Η παραπάνω σχέση εκφράζει το σφάλμα στην μέθοδο απαλοιφής του Gauss σε όρους $k_2(A[1,\dots,k \mid 1,\dots,k])$. Η προσέγγιση των αριθμών συνθήκης για τυχαίους πίνακες δίνεται από τον Edelman [29] [34].

2.8.2 Ανάλυση σφάλματος στον μετασχηματισμό RBT

Χρησιμοποιώντας τις παραπάνω σχέσεις μπορούμε να κάνουμε μια εκτίμηση της επίδρασης που έχει ο μετασχηματισμός RBT στους αριθμούς συνθήκης $k_2(\tilde{A}[1,\dots,k \mid 1,\dots,k])$ των υποπινάκων. Εφαρμόζοντας την μέθοδο απαλοιφής του Gauss στον $\tilde{A}$ θα έχουμε:

$$\tilde{\alpha}_{ij}^{(k+1)} = \frac{\det \tilde{A}[1,\dots,k,i \mid 1,\dots,k,j]}{\det \tilde{A}[1,\dots,k \mid 1,\dots,k]} \quad (73)$$

Από το θεώρημα Binet-Cauchy [30] έχουμε ότι για τους τετραγωνικούς πίνακες διάστασης $n \times n$ X, Y και για κάθε ακέραιο k τέτοιο ώστε $1 \leq k \leq n$ και για κάθε ακολουθία $\kappa, \lambda \in \mathrm{S}_{\mathrm{k}}^{\mathrm{n}}$ θα ισχύει:

$$\det((XY)[\kappa \mid \lambda]) = \sum_{\mu \in \mathrm{S}_{\mathrm{k}}^{\mathrm{n}}} \det X[\kappa \mid \lambda] \det Y[\mu \mid \lambda] \quad (74)$$

Όπου $\mathrm{S}_{\mathrm{k}}^{\mathrm{n}}$ είναι το σύνολο των φθινουσών ακολουθιών k αριθμών στο διάστημα {1,…,n}. Ισχύει $\tilde{A} = U^* A V$ συνεπώς:

$$\tilde{\alpha}_{ij}^{(k+1)} = \frac{\sum_{\kappa,\lambda \in \mathrm{S}_{\mathrm{k+1}}^{\mathrm{n}}} \det U^*[1,\dots,k,i \mid k] \det A[\kappa \mid \lambda] \det V[\lambda \mid 1,\dots,k,j]}{\sum_{\mu,\nu \in \mathrm{S}_{\mathrm{k}}^{\mathrm{n}}} \det U^*[1,\dots,k \mid \mu] \det A[\mu \mid \nu] \det V[\nu \mid 1,\dots,k]} \quad (75)$$

το οποίο δίνει ένα μέσο όρο του ρυθμού αύξησης. Κάθε άθροισμα εμπλέκει τις ορίζουσες των υποπινάκων του A επί το γινόμενο των στοιχείων στους τυχαίους πίνακες U* και V. Υποθέτοντας ότι (α) τα στοιχεία στους τυχαίους πίνακες U* και V είναι κατανεμημένα έτσι ώστε τα γινόμενα να είναι μη μηδενικά και (β) οι ορίζουσες των υποπινάκων ακολουθούν μια Γκαουσιανή (Gaussian) κατανομή, τότε οι τιμές των $\tilde{\alpha}_{ij}^{(k+1)}$ μπορούν να προσεγγιστούν. Από την ανάλυση προκύπτει ότι ο μετασχηματισμός RBT επηρεάζει τους υποπίνακες του αρχικού πίνακα και επιπρόσθετα, μπορεί να επηρεάσει τους αντίστοιχους συντελεστές αύξησης g των υποπινάκων [29] [17].

2.9 Επίδραση της μεθόδου RBT στον αριθμό συνθήκης

Για να επιτύχουμε την επιθυμητή αριθμητική ακρίβεια κατά την επίλυση ενός γραμμικού συστήματος θα θέλαμε ο αριθμός συνθήκης του πίνακα UAV που προκύπτει από την εφαρμογή του RBT να είναι όσο το δυνατόν μικρότερος, συνεπώς όσο το δυνατόν πιο κοντά σε αυτόν του πίνακα A. Έχουμε ότι [11]:

$$\kappa_2(UAV) \leq \kappa_2(U)\kappa_2(A)\kappa_2(V) \quad (76)$$

Όπου $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$ είναι ο αριθμός συνθήκης ενός πίνακα Α. Στην ιδανική περίπτωση, θα θέλαμε ο αριθμός συνθήκης του αναδρομικού πίνακα πεταλούδας να είναι κοντά στο 1, ώστε ο αριθμός συνθήκης του πίνακα UAV να είναι κοντά σε αυτόν του Α. Σύμφωνα με τον Edelman [34], οι τυχαίοι πίνακες τείνουν να παρουσιάζουν χαμηλό αριθμό συνθήκης. Στην περίπτωση των αναδρομικών πινάκων πεταλούδας έχουμε ότι:

$$B^TB = \frac{1}{\sqrt{2}}\begin{pmatrix} R_0 & R_0 \\ R_1 & -R_1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{pmatrix} = \begin{pmatrix} R_0^2 & 0 \\ 0 & R_0^2 \end{pmatrix} \tag{77}$$

$$= diag(r_1, \dots, r_n)^2 \tag{78}$$

Όπου r_i είναι τυχαία στοιχεία, και από [35] έχουμε:

$$\kappa_2(B) = \sqrt{\kappa_2(B^TB)} = \frac{max|r_i|}{min|r_i|} \tag{79}$$

Από την τελευταία εξίσωση προκύπτει ότι τα στοιχεία r_i επιδρούν στον αριθμό συνθήκης του πίνακα Β. Συνεπώς, επιλέγοντας πολύ μικρές τιμές στα στοιχεία r_i μπορεί να οδηγήσει σε μεγάλο αριθμό συνθήκης για τον πίνακα Α.

Γενικότερα, ένας αναδρομικός πίνακας πεταλούδας με βάθος αναδρομής p είναι το γινόμενο μπλοκ διαγώνιων πινάκων της μορφής: $\mathcal{B} = diag(B_1, \dots, B_p)$ όπου $1 \le p \le 2^{d-1}$ και B_i είναι πίνακες πεταλούδας διάστασης $\frac{n}{p}$. Συνεπώς θα έχουμε ότι:

$$\mathcal{B}^T\mathcal{B} = \begin{pmatrix} B_1^TB_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B_p^TB_p \end{pmatrix} \tag{80}$$

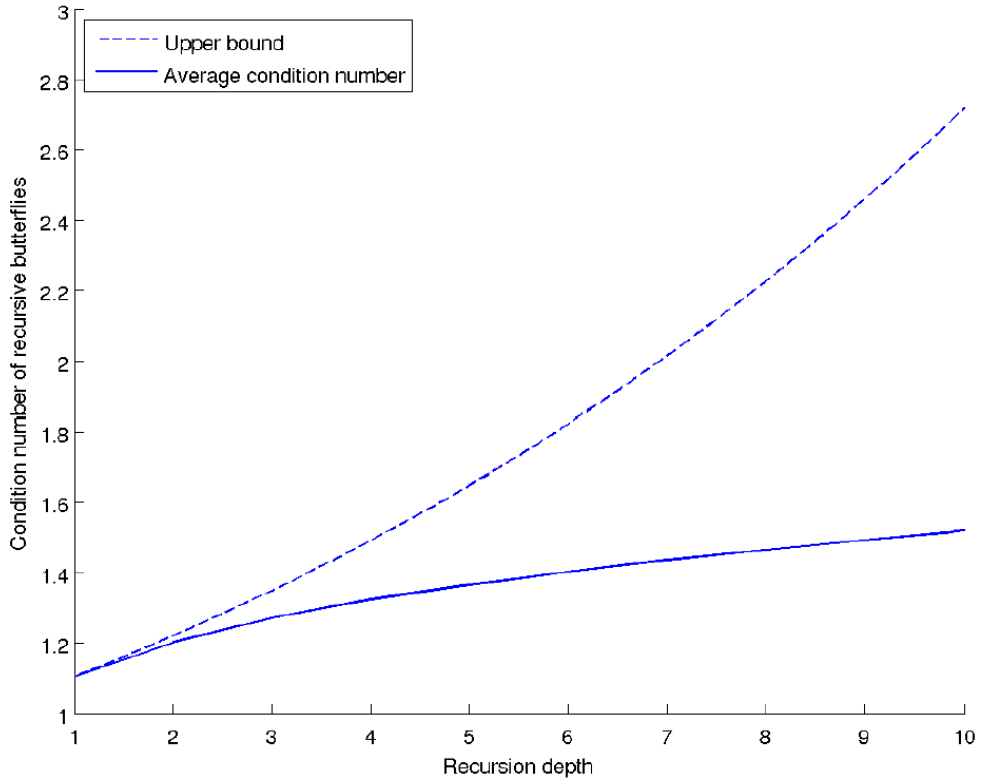
Και ο $\mathcal{B}^T\mathcal{B}$ είναι διαγώνιος πίνακας. Ο αριθμός συνθήκης $\kappa_2(\mathcal{B})$ μπορεί να εκφραστεί σαν $\frac{max|r_i|}{min|r_i|}$, όπου r_i είναι τυχαία στοιχεία που σχηματίζουν την διαγώνιο του πίνακα $\mathcal{B}^T\mathcal{B}$.

Αν τα στοιχεία r_i βρίσκονται στο διάστημα $[\alpha, \beta]$: $r_i \in [\alpha, \beta]$, $\alpha > 0$ τότε συνεπάγεται ότι $\kappa_2(\mathcal{B}) \le \frac{\beta}{\alpha}$. Γενικεύοντας, έστω U ένας αναδρομικός πίνακας πεταλούδας βάθους αναδρομής d, τότε ο αριθμός συνθήκης του U θα είναι:

$$\kappa_2(U) \le \left(\frac{\beta}{\alpha}\right)^d \tag{81}$$

Από αυτό το αποτέλεσμα, μπορούμε να αντιληφθούμε την σημασία επιλογής των τυχαίων στοιχείων για τον σχηματισμό των αναδρομικών πινάκων πεταλούδας. Όπως γίνεται αντιληπτό, ο αριθμός συνθήκης αυξάνεται εκθετικά ως προς το βάθος αναδρομής, οπότε ο λόγος $\frac{\beta}{\alpha}$ θα πρέπει να κρατείται κοντά στο 1 με κατάλληλη επιλογή του διαστήματος. Στην αρχική υλοποίηση της μεθόδου από τον Parker [18] [17], τα τυχαία στοιχεία είναι της μορφής $\exp(\frac{r}{10})$, όπου ο r επιλέγεται τυχαία από το διάστημα $[-\frac{1}{2}, \frac{1}{2}]$. Η επιλογή αυτή είναι κατάλληλη αφού ο λόγος $\frac{\beta}{\alpha} = e^{0.1} \approx 1.105$ διαμορφώνεται κοντά στο 1. Περισσότερα για την συμπεριφορά της μεθόδου σε σχέση με το

συγκεκριμένο διάστημα των τυχαίων διαγώνιων στοιχείων ακολουθούν στο επόμενο κεφάλαιο.



Σχήμα 4: μέσος αριθμός συνθήκης συναρτήσει του βαθμού αναδρομής [11].

Στο παραπάνω σχήμα φαίνεται ο μέγιστος και μέσος αριθμός συνθήκης πινάκων διάστασης 1024 από ένα δείγμα 500 πινάκων [11]. Το σχήμα δικαιολογεί τις παρατηρήσεις των Baboulin, Dogarra και άλλων [11] [16], δηλαδή ότι στην πράξη δυο επίπεδα αναδρομής είναι ικανά για παραγωγή ευσταθών αποτελεσμάτων, αφού όπως φαίνεται στο σχήμα μετά το δεύτερο επίπεδο αναδρομής ο αριθμός συνθήκης αυξάνει αλλά με πολύ μικρό ρυθμό ώστε να διατηρείται η ευστάθεια της μεθόδου.

2.10 Παραγωγή αναδρομικών πινάκων τυχαίων πεταλούδων

Σε αυτό το σημείο της εργασίας, παρουσιάζεται ένα αριθμητικό παράδειγμα για την παραγωγή των τυχαίων πινάκων πεταλούδας και την επίλυση του γραμμικού συστήματος, πριν περάσουμε στην παρουσίαση του αλγορίθμου. Για λόγους απλότητας δεν θα χρησιμοποιήσουμε γενικούς τυχαίους πίνακες πεταλούδας αλλά μη ιδιάζοντες πίνακες με πραγματικούς αριθμούς και με ορίζουσα 1.

Ας θεωρήσουμε το ακόλουθο γραμμικό σύστημα $Ax = b$:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 2 \end{pmatrix}$$

Στην συνέχεια παράγουμε τους ακόλουθους τυχαίους πίνακες πεταλούδας:

$$U^{<1>} = \begin{pmatrix} .8872 & 0 & 0 & 0 \\ 0 & .7445 & 0 & 0 \\ 0 & 0 & .7727 & 0 \\ 0 & 0 & 0 & .9585 \end{pmatrix}$$

$$V^{<1>} = \begin{pmatrix} 1.632 & 0 & 0 & 0 \\ 0 & .4677 & 0 & 0 \\ 0 & 0 & .5073 & 0 \\ 0 & 0 & 0 & 2.137 \end{pmatrix}$$

$$U^{<2>} = \begin{pmatrix} 1.101 & 1.502 & 0 & 0 \\ 1.101 & -1.502 & 0 & 0 \\ 0 & 0 & .4623 & 1.444 \\ 0 & 0 & .4623 & -1.444 \end{pmatrix}$$

$$V^{<2>} = \begin{pmatrix} 1.675 & 0.9265 & 0 & 0 \\ 1.675 & -0.9265 & 0 & 0 \\ 0 & 0 & 1.268 & 2.001 \\ 0 & 0 & 1.268 & -2.001 \end{pmatrix}$$

$$U^{<4>} = \begin{pmatrix} 1.188 & 0 & 0.6729 & 0 \\ 0 & 1.499 & 0 & 0.7312 \\ 1.188 & 0 & -0.6729 & 0 \\ 0 & 1.499 & 0 & -0.7312 \end{pmatrix}$$

$$V^{<4>} = \begin{pmatrix} 2.104 & 0 & 0.5667 & 0 \\ 0 & 0.5583 & 0 & 0.8901 \\ 2.104 & 0 & -0.5667 & 0 \\ 0 & 0.5583 & 0 & -0.8901 \end{pmatrix}$$

Συνεπώς, μπορούμε να ορίσουμε τους δυο αναδρομικούς πίνακες πεταλούδα $U = U^{<1>}U^{<2>}U^{<4>}$ και $V = V^{<1>}V^{<2>}V^{<4>}$. Εφαρμόζοντας τον μετασχηματισμό RBT προκύπτει το ενδιάμεσο γραμμικό σύστημα $U^*AVy = U^*b$, το οποίο είναι Gauss απαλείψιμο:

$$\begin{pmatrix} 20.68 & .085 & 1.869 & 3.887 \\ .77 & -13.83 & -7.333 & 10.19 \\ 4.632 & 2.122 & 2.541 & -0.735 \\ -18.19 & 2.588 & -.027 & -8.649 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 5.434 \\ -.485 \\ .4426 \\ -1.084 \end{pmatrix}$$

Το οποίο και επιλύουμε με την μέθοδο απαλοιφής του Gauss χωρίς την χρήση οδήγησης, και βρίσκουμε την ενδιάμεση λύση y:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} .5653 \\ -.2578 \\ -.9704 \\ -1.138 \end{pmatrix}$$

Και τέλος, για να βρούμε την λύση στο αρχικό γραμμικό σύστημα:

$$x = Vy = \begin{pmatrix} -.001 \\ 1.002 \\ 2.001 \\ 0.998 \end{pmatrix}$$

Η οποία είναι σωστή στα πρώτα 3 σημαντικά ψηφία.

Παρατηρούμε ότι ο πίνακας A του γραμμικού συστήματος έχει μικρό αριθμό συνθήκης $\kappa_2(A) \approx 2.61$ και σαν συνέπεια, τα αποτελέσματα της μεθόδου είναι αρκετά ακριβή. Στην περίπτωση που ο πίνακας A έχει μεγάλο αριθμό συνθήκης τότε το αποτέλεσμα που θα πάρουμε θα είναι λιγότερο ακριβές. Για παράδειγμα, αν θεωρήσουμε τον ακόλουθο πίνακα A' [17]:

$$A' = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 100 & 0 & 1 & 0 \\ 0 & 1 & 0 & 100 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

ο οποίος έχει αριθμό συνθήκης $\kappa_2(A') = 100$. Χρησιμοποιώντας τους ίδιους τυχαίους πίνακες U και V με το προηγούμενο παράδειγμα θα πάρουμε την ενδιάμεση λύση:

$$\begin{pmatrix} y_1{}' \\ y_2{}' \\ y_3{}' \\ y_4{}' \end{pmatrix} = \begin{pmatrix} .5132 \\ -.1404 \\ -.8063 \\ -1.177 \end{pmatrix}$$

και η λύση του γραμμικού συστήματος θα είναι:

$$x' = Vy' = \begin{pmatrix} 0.001 \\ .9758 \\ 1.972 \\ 0.019 \end{pmatrix}$$

ενώ, η ακριβής λύση είναι:

$$x_{exact}{}' = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 0.02 \end{pmatrix}$$

Παρατηρούμε ότι σε αυτή τη περίπτωση η λύση που παίρνουμε είναι ακριβής μόνο στο πρώτο ψηφίο.

3. Υλοποίηση της μεθόδου RBT

3.1 Υλοποίηση της μεθόδου RBT

Για να είμαστε σε θέση να εξακριβώσουμε την απόδοση και την ποιότητα της μεθόδου θα παρουσιάσουμε μια σειριακή έκδοση της μεθόδου όπως αυτή περιγράφεται μέσα από τις εργασίας του Parker [17] [18] [19]. Επιπρόσθετα, θα παρουσιάσουμε έναν αποδοτικό τρόπο αποθήκευσης των πινάκων πεταλούδας. Μια παράλληλη υλοποίηση της μεθόδου για υβριδικές αρχιτεκτονικές (CPU / GPU) θα μπορούσε να εκμεταλλευτεί το συγκριτικό πλεονέκτημα της, το οποίο είναι η ελαχιστοποίηση της μεταφοράς δεδομένων που είναι αναγκαία σε μια παράλληλη αρχιτεκτονική.

3.2 Αποδοτική αποθήκευση αναδρομικών πινάκων πεταλούδας

Στην εργασία των Baboulin, Dogarra και άλλοι [11] παρουσιάζεται ένας τρόπος αποδοτικής αποθήκευσης των αναδρομικών πινάκων πεταλούδας, χρησιμοποιώντας μια «συμπαγή» (compact) αναπαράσταση που αποτελείται από έναν πίνακα διανυσμάτων. Όπως ξέρουμε, ένας πίνακας πεταλούδα έχει την μορφή:

$$B^{<n>} = \frac{1}{\sqrt{2}} \begin{pmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{pmatrix} \tag{82}$$

Όπου R_0 και R_1 είναι τυχαίοι διαγώνιοι πίνακες. Τότε ο $B^{<n>}$ μπορεί να αποθηκευθεί σε «συμπαγή» αναπαράσταση από ένα διάνυσμα w διάστασης n. Οι πρώτες $\frac{n}{2}$ θέσεις του διανύσματος περιέχουν τους συντελεστές του διαγώνιου πίνακα R_0 και οι υπόλοιπες θέσεις τους συντελεστές του R_1.

Ένας αναδρομικός πίνακας πεταλούδα είναι το γινόμενο πινάκων πεταλούδας. Όπως ξέρουμε, ένας αναδρομικός πίνακας πεταλούδα διάστασης n με βάθος αναδρομής d θα είναι το γινόμενο πινάκων πεταλούδας:

$$W^{<n,d>} = \begin{pmatrix} B_1^{<\frac{n}{2^{d-1}}>} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B_{2^{d-1}}^{<\frac{n}{2^{d-1}}>} \end{pmatrix} \times \cdots \times \begin{pmatrix} B_1^{<\frac{n}{2}>} & 0 \\ 0 & B_2^{<\frac{n}{2}>} \end{pmatrix} \times B^{<n>} \tag{83}$$

Είναι εύκολο να παρατηρήσουμε ότι κάθε όρος του γινομένου (δηλ οι επιμέρους πίνακες πεταλούδας) μπορεί να αποθηκευτεί σαν ένα διάνυσμα διάστασης n, ενώ οι ότοι του γινομένου είναι d. Συνεπώς ο αναδρομικός πίνακας πεταλούδα $W^{<n,d>}$ μπορεί να αποθηκευθεί σε «συμπαγή» αναπαράσταση με την βοήθεια του πίνακα W_p διάστασης $n \times d$, όπου η k-οστή στήλη αναπαριστά τον πίνακα

$$\begin{pmatrix} B_1^{<\frac{n}{2^{k-1}}>} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B_{2^{k-1}}^{<\frac{n}{2^{k-1}}>} \end{pmatrix} \tag{84}$$

το οποίο σημαίνει ότι κάθε διάνυσμα $W_p\left((i-1)*\frac{n}{2^{k-1}}+1:i*\frac{n}{2^{k-1}},k\right)$ αποθηκεύει τον πίνακα πεταλούδα $B_i^{<\frac{n}{2^{k-1}}>}$. Αυτό σημαίνει ότι ο αναδρομικός πίνακας πεταλούδα $W^{<n,d>}$ μπορεί να παραχθεί και να αποθηκευθεί σε έναν πίνακα W_p διάστασης $n \times d$.

3.3 Υπολογιστική πολυπλοκότητα της μεθόδου RBT

Σε αυτή την ενότητα θα ασχοληθούμε με την ανάλυση της μεθόδου και τον προσδιορισμό της υπολογιστικής πολυπλοκότητας.

Για την εφαρμογή της μεθόδου διακρίνουμε τα ακόλουθα βήματα:

Υπολογισμός γινομένου UAV και Ub (διαδικασία η οποία εμπεριέχει την δημιουργία των αναδρομικών πινάκων πεταλούδας)
Επίλυση του γραμμικού συστήματος $\mathrm{UAVy} = \mathrm{Ub}$ με την χρήση της μεθόδου απαλοιφής του Gauss χωρίς οδήγηση
Εύρεση της λύσης του αρχικού συστήματος ως το γινόμενο Vy

Εδώ θα ακολουθήσουμε την ισοδύναμη διατύπωση των Baboulin, Dongarra και άλλοι [11], κατά την οποία το ενδιάμεσο γραμμικό σύστημα διαμορφώνεται ως εξής: $U^T\mathrm{AVy} = U^T\mathrm{b}$.

Αρχικά, θα ασχοληθούμε με το βήμα 1 το οποίο είναι το γινόμενο ($U^T\mathrm{AV}$) ενός πυκνού πίνακα A από τα αριστερά και από τα δεξιά με έναν πίνακα πεταλούδα. Έστω B και B' δυο πίνακες πεταλούδας αποθηκευμένοι σε «συμπαγή» αναπαράσταση χρησιμοποιώντας τα διανύσματα w και w':

$$B = \frac{1}{\sqrt{2}}\begin{pmatrix} R_0 & R_1 \\ R_0 & -R_1 \end{pmatrix},\ B' = \frac{1}{\sqrt{2}}\begin{pmatrix} R_0{}' & R_1{}' \\ R_0{}' & -R_1{}' \end{pmatrix} \tag{85}$$

Παρατηρούμε ότι ο πολλαπλασιασμός και από τις δυο πλευρές του πίνακα A με τους B και B' μπορεί να γραφεί ως εξής:

$$B^T A B' = \frac{1}{2}\begin{pmatrix} R_0 & R_0 \\ R_1 & -R_1 \end{pmatrix} A \begin{pmatrix} R_0{}' & R_1{}' \\ R_0{}' & -R_1{}' \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} R_0 & R_0 \\ R_1 & -R_1 \end{pmatrix}\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} R_0{}' & R_1{}' \\ R_0{}' & -R_1{}' \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} R_0 & 0 \\ 0 & R_1 \end{pmatrix} C \begin{pmatrix} R_0{}' & 0 \\ 0 & R_1{}' \end{pmatrix}$$

$$= \frac{1}{2} diag(W) C diag(W') \tag{86}$$

όπου

$$C = \begin{pmatrix} A_{11}+A_{12}+A_{21}+A_{22} & A_{11}-A_{12}+A_{21}-A_{22} \\ A_{11}+A_{12}+A_{21}-A_{22} & A_{11}-A_{12}-A_{21}+A_{22} \end{pmatrix} \tag{87}$$

Τότε $(B^T AB')_{i,j} = W_i C_{i,j} W'_j$ και ο υπολογισμός του $B^T AB'$ απαιτεί $4n^2$ πράξεις [11]. Αυτός ο πυρήνας αντιστοιχεί στην εφαρμογή του μετασχηματισμού RBT με βάθος αναδρομής 1. Έτσι, για βάθος αναδρομής $d = 2$ έχουμε:

$$W^{<n,2>} = B^T \begin{pmatrix} B_1^T & 0 \\ 0 & B_2^T \end{pmatrix} A \begin{pmatrix} B_1{}' & 0 \\ 0 & B_2{}' \end{pmatrix} B'$$

$$= B^T \begin{pmatrix} B_1^T A_{11} B_1{}' & B_1^T A_{12} B_2{}' \\ B_2^T A_{21} B_1{}' & B_2^T A_{22} B_2{}' \end{pmatrix} B' \quad (88)$$

Συνεπώς, στον υπολογισμό του γινομένου εμπλέκονται 4 πίνακες πεταλούδας διάστασης $\frac{n}{2}$ και έναν πίνακα πεταλούδας διάστασης n, με αποτέλεσμα το γινόμενο $B^T AB'$ να απαιτεί $8n^2$ πράξεις.

Στην γενική περίπτωση, έστω ο A ένας τετραγωνικός πίνακας διάστασης n, και M(n) το κόστος υπολογισμού του γινομένου $B^T AB'$, όπου B και B' πίνακες πεταλούδας διάστασης n. Τότε το γινόμενο $B^T AB'$ που υπολογίζεται με τον μετασχηματισμό RBT για βάθος αναδρομής d είναι:

$$c(n,d) = \sum_{k=1}^{d} \left((2^{k-1})^2 \times M\left(\frac{n}{2^{k-1}}\right) \right)$$

$$= \sum_{k=1}^{d} \left((2^{k-1})^2 \times 4\left(\frac{n}{2^{k-1}}\right)^2 \right) = \sum_{k=1}^{d} (4n^2) = 4dn^2 \quad (89)$$

Χρησιμοποιώντας τους υπολογιστικούς πυρήνες που χρησιμοποιήσαμε πριν, προκύπτει ότι το κόστος του μετασχηματισμού RBT στην περίπτωση που χρησιμοποιηθεί το μέγιστο βάθος αναδρομής είναι:

$$c(n, log_2 n) = 4n^2 log_2 n \quad (90)$$

Από την τελευταία σχέση, προκύπτει ότι το βάθος αναδρομής $d < log_2 n \ll n$ πρέπει να επιλέγεται κατάλληλα έτσι ώστε το γινόμενο να είναι υπολογιστικά φθηνό. Όπως έχουμε σχολιάσει και προηγουμένως, συνήθως 2 επίπεδα αναδρομής προσφέρουν ικανοποιητική ακρίβεια, συνεπώς στην περίπτωση αυτή, το υπολογιστικό κόστος του βήματος 1 θα είναι $8n^2$ πράξεις συν το γινόμενο του πίνακα πεταλούδα U με το διάνυσμα b των σταθερών όρων.

Αντίστοιχα, ο υπολογισμός του γινομένου πίνακα πεταλούδας με το διάνυσμα των σταθερών όρων $U^T b$ στο βήμα 1, καθώς και το γινόμενο Vy στο βήμα 3 απαιτούν $O(dn^2)$ πράξεις [11].

Τέλος, το βήμα 2 της μεθόδου απαιτεί την επίλυση ενός γραμμικού συστήματος διάστασης n με την μέθοδο απαλοιφής του Gauss χωρίς την χρήση οδήγησης. Η διαδικασία αυτή έχει πολυπλοκότητα $O(\frac{n(n^2-1)}{3})$ [18] αν και είναι μια διαδικασία που δεν σχετίζεται με τον μετασχηματισμό άμεσα.

Συνοψίζοντας, αν θέλαμε να παρουσιάσουμε την πολυπλοκότητα του μετασχηματισμού θα πρέπει να αθροίσουμε το υπολογιστικό κόστος των γινομένων UAV, Ub και Vy, το οποίο είναι: $O(4dn^2 + 2 * dn^2) \approx O(n^2)$ αφού το βάθος αναδρομής είναι $d < log_2 n \ll n$ είναι φραγμένο κατά πολύ από την διάσταση του πίνακα A.

Παρατηρούμε, ότι η πολυπλοκότητα του μετασχηματισμού είναι παρόμοια με αυτήν της μερικής οδήγησης (επίσης $O(n^2)$). Παρόλο που αυτές οι δυο τεχνικές παρουσιάζουν παρόμοια υπολογιστική πολυπλοκότητα, η ειδοποιός διαφορά είναι ότι η μέθοδος RBT δεν προσπαθεί να ελαχιστοποιήσει την υπολογιστική πολυπλοκότητα, αλλά την επικοινωνία μεταξύ των επεξεργαστών σε παράλληλες αρχιτεκτονικές.

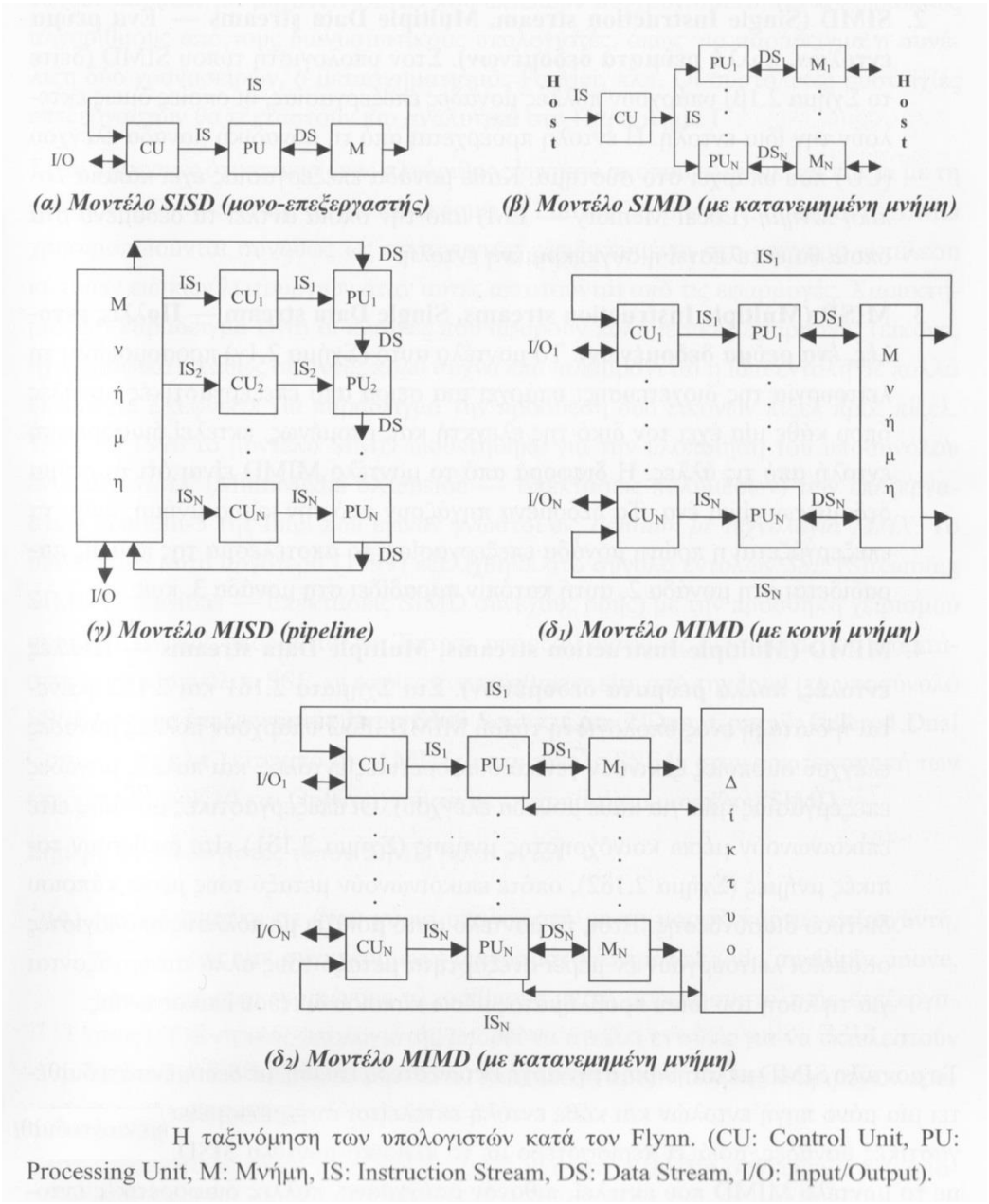
3.4 Η διαφοροποίηση της μεθόδου RBT σε σχέση με άλλες τεχνικές οδήγησης στις παράλληλες αρχιτεκτονικές

Πολλά πραγματικά επιστημονικά προβλήματα, ενσωματώνουν στον πυρήνα τους την επίλυση γραμμικών συστημάτων. Συνήθως, τα πραγματικά προβλήματα (πχ. πρόγνωση καιρού) εμπλέκουν συστήματα ιδιαίτερα μεγάλης διάστασης τα οποία για να λυθούν αποδοτικά απαιτούν την χρήση παράλληλων υπολογιστών υψηλών επιδόσεων. Σε αυτές τις αρχιτεκτονικές, η επιλογή του κατάλληλου αλγόριθμου επίλυσης γραμμικών συστημάτων είναι καταλυτική για την επίτευξη της μέγιστης απόδοσης. Μάλιστα, σε κατανεμημένες αρχιτεκτονικές οι διασυνδεδεμένοι υπολογιστές βρίσκονται σε διαφορετικές φυσικές μηχανές με αποτέλεσμα το κόστος και η καθυστέρηση επικοινωνίας μεταξύ των επεξεργαστών να είναι σημαντική. Για τον λόγο αυτό, επιλέγονται αλγόριθμοι με περιορισμένους βρόχους επανάληψης που εμπλέκουν απλές πράξεις που εκτελούνται όσο το δυνατόν ανεξάρτητα και εκτελούνται στην κρυφή μνήμη (cache memory) έτσι ώστε οι επικοινωνίες και η καθυστέρηση συγχρονισμού που εισάγονται να ελαχιστοποιούνται.

3.4.1 Παράλληλες υπολογιστικές αρχιτεκτονικές

Σε αυτή την ενότητα θα παρουσιάσουμε τις κύριες αρχιτεκτονικές παράλληλων υπολογιστών. Διακρίνονται δυο κύριες κατηγοριοποιήσεις: ως προς τις ακολουθίες (ροές) εντολών και δεδομένων και ως προς την κατανομή της μνήμης [36].

Το 1972 ο Flynn [37] πρότεινε την ταξινόμηση των υπολογιστικών συστημάτων με βάση το πλήθος των διαφορετικών ρευμάτων εντολών (instruction streams) που μπορούν να εκτελεστούν ταυτόχρονα σε πόσα ρεύματα δεδομένων (data streams). Η ταξινόμηση αυτή δημιουργεί τέσσερεις κλάσεις οι οποίες απεικονίζονται στο σχήμα 5 και είναι οι ακόλουθες:

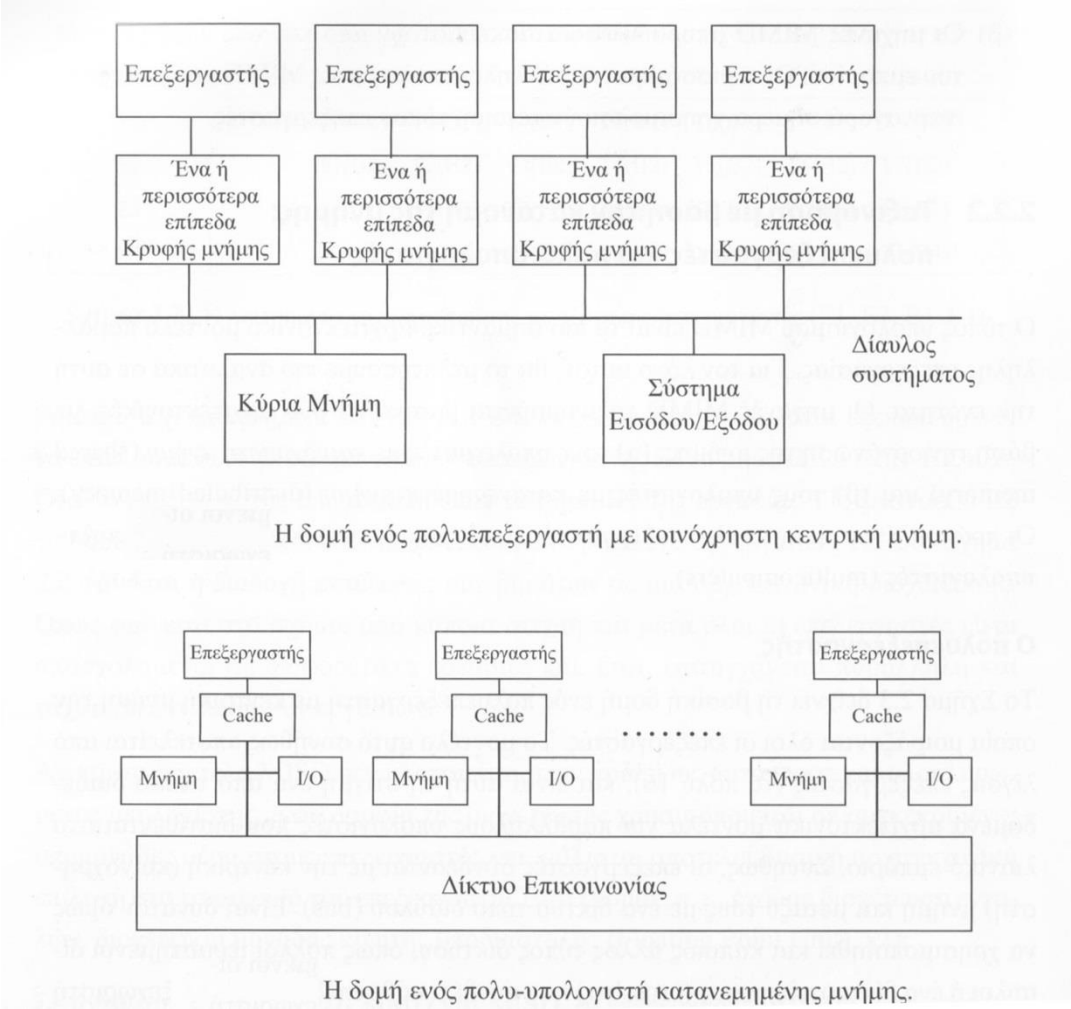


Σχήμα 5: Η ταξινόμηση των υπολογιστών κατά τον Flynn [36].

SISD (Single Instruction stream, Single Data stream – Ένα ρεύμα εντολών, ένα ρεύμα δεδομένων): ο υπολογιστής εκτελεί μια μόνο εντολή κάθε χρονική στιγμή σε ένα δεδομένο.
SIMD (Single Instruction stream, Multiple Data streams – Ένα ρεύμα εντολών, πολλά ρεύματα δεδομένων): πολλές μονάδες επεξεργασίας εκτελούν την ίδια εντολή.
MISD (Multiple Instruction streams, Single Data stream – Πολλές εντολές, ένα ρεύμα δεδομένων): πολλές μονάδες επεξεργασίας εκτελούν διαφορετικές εντολές στο ίδιο ρεύμα δεδομένων.
MIMD (Multiple Instruction streams, Multiple Data streams - Πολλές εντολές, πολλά ρεύματα δεδομένων): πολλές μονάδες επεξεργασίας εκτελούν διαφορετικές εντολές, επικοινωνούν μεταξύ τους μέσω κοινόχρηστης μνήμης ή μέσω δικτύου.

Η αρχιτεκτονική MIMD είναι η κυριότερη αρχιτεκτονική στα παράλληλα συστήματα, και οι υπολογιστές αυτής της κατηγορίας μπορούν να διακριθούν ως προς την κατανομή και οργάνωση της μνήμης [36] (όπως παρουσιάζονται στο Σχήμα 6):

Υπολογιστές με κοινόχρηστη μνήμη (shared memory) ονομάζονται και «πολυεπεξεργαστές» : συνήθως η αρχιτεκτονική αυτή περιλαμβάνει λίγους επεξεργαστές οι οποίοι συνδέονται μεταξύ τους και με την κοινόχρηστη μνήμη με ένα δίκτυο διαύλου (bus) ή με δίκτυο πλέγματος (crossbar). Όλοι οι επεξεργαστές έχουν ομοιόμορφη πρόσβαση σε όλες τις διευθύνσεις μνήμης, πραγματοποιώντας αιτήματα για ανάγνωση ή εγγραφή στο δίαυλο. Ο χρόνος απόκρισης είναι ο ίδιος ανεξάρτητα με την διεύθυνση των δεδομένων. Επίσης ονομάζεται και μοντέλο ομοιόμορφης προσπέλασης μνήμης (uniform memory access)
Υπολογιστές με κατανεμημένη μνήμη (distributed memory) ονομάζονται και «πολύ-υπολογιστές» : κάθε επεξεργαστής έχει την δική του τοπική μνήμη, καθώς και δική του είσοδο και έξοδο. Οι επεξεργαστές επικοινωνούν μέσα από ένα δίκτυο διασύνδεσης (interconnection network). Αυτή η αρχιτεκτονική έχει σαν πλεονέκτημα την δυνατότητα προσπέλασης μεγαλύτερης ποσότητας μνήμης σε μικρότερο χρόνο με την αποφυγή της διαιτησίας από τον δίαυλο. Σαν αποτέλεσμα, οι επεξεργαστές επιβαρύνονται με το κόστος επικοινωνίας για να ανταλλάξουν δεδομένα.



Σχήμα 6: Υπολογιστές με κοινόχρηστη και κατανεμημένη μνήμη [36].

3.4.2 Η επιβάρυνση της οδήγησης στις παράλληλες αρχιτεκτονικές και η συνεισφορά της μεθόδου RBT

Όπως έχουμε αναφέρει, για την επίλυση ενός γραμμικού συστήματος με την μέθοδο απαλοιφής του Gauss, πρέπει να χρησιμοποιηθεί κάποιο είδος οδήγησης έτσι ώστε η μέθοδος να παράγει αριθμητικά ακριβείς λύσεις.

Η οδήγηση είναι μια διαδικασία κατά την οποία οι γραμμές και οι στήλες του πίνακα αντιμετατίθενται με σκοπό τα μεγάλα στοιχεία να βρεθούν στην κύρια διαγώνιο του πίνακα. Η διαδικασία της οδήγησης δεν περιλαμβάνει πράξεις με αριθμούς κινητής υποδιαστολής αλλά προκαλεί σημαντική μεταφορά δεδομένων, αφού για παράδειγμα στην περίπτωση της ολικής οδήγησης απαιτούνται $O(n^3)$ συγκρίσεις. Μια διαφοροποίηση που στην πράξη έχει ικανοποιητική απόδοση είναι η χρήση της μερικής οδήγησης, η οποία απαιτεί $O(n^2)$ συγκρίσεις αντιμεταθέτοντας μόνο γραμμές, ωστόσο και αυτή η μέθοδος «διακόπτει» την φυσική ροή της μεθόδου απαλοιφής του Gauss, διατρέχοντας την τρέχουσα στήλη του πίνακα για την αναζήτηση των οδηγών στοιχείων οδήγησης. Υπάρχει και μια ενδιάμεση στρατηγική οδήγησης που ονομάζεται «rook pivoting» η οποία απαιτεί μεταξύ του $O(n^2)$ και του $O(n^3)$ συγκρίσεις [7].

Συνήθως, η επιλογή της μεθόδου οδήγησης εξαρτάται από το υπολογιστικό κόστος που είναι διατεθειμένος να θυσιάσει κάποιος για να επιτύχει μια μεγαλύτερη αριθμητική ακρίβεια.

Υπάρχουν δυο χαρακτηριστικά τα οποία πρέπει να διαθέτει μια αριθμητική μέθοδος για να είναι αποδοτική: να ελαχιστοποιεί τον συντελεστή αύξησης (που δείχνει πόσο μεγάλα τα στοιχεία του πίνακα μπορεί να γίνουν κατά την απαλοιφή) ώστε να εξασφαλίζει την αριθμητική ευστάθεια και να περιορίζει την οδήγηση ώστε να ελαχιστοποιείται η υπολογιστική επιβάρυνση [7] [11].

Το πρόβλημα του περιορισμού του υπολογιστικού κόστους χωρίς σημαντική απώλεια στην ποιότητα των λύσεων έχει οδηγήσει στην ανάπτυξη υλοποιήσεων για παράλληλους και διανυσματικούς υπολογιστές υψηλών επιδόσεων [38] [39].

Η επιβάρυνση επικοινωνίας που εισάγει η οδήγηση μπορεί είναι τέτοια, ώστε σε μια παράλληλη μηχανή ο χρόνος της οδήγησης ($O(n^2)$) να ξεπερνάει τον χρόνο εκτέλεσης του αλγορίθμου της απαλοιφής του Gauss ($O(n^3)$) [18]. Η επιβάρυνση αυτή είναι σημαντική και στις αρχιτεκτονικές MIMD και SIMD γι αυτό και στην βιβλιογραφία έχουν προταθεί αλγόριθμοι για τον χρονοπρογραμματισμό της μεθόδου στην αρχιτεκτονική MIMD [39]. Οι Bampis και άλλοι [40], έδειξαν ότι στην αρχιτεκτονική MIMD η επιβάρυνση των επικοινωνιών και του χρόνου αδράνειας (idle time) είναι της τάξης του $O(n^3)$ χωρίς να λάβουμε υπόψη την οδήγηση. Επίσης, η εργασία του Veldhorst [41] ασχολείται με την επίδραση της μερικής οδήγησης στην αρχιτεκτονική MIMD. Τέλος, υλοποιήσεις του αλγορίθμου της απαλοιφής του Gauss με ολική οδήγηση στην αρχιτεκτονική SIMD σε τοπολογία υπερκύβου, έχουν δείξει ότι υπάρχει αύξηση της επιβάρυνσης των επικοινωνιών όσο το πλήθος των επεξεργαστών αυξάνει [42].

Σήμερα, με την εξέλιξη των πολυπύρηνων επεξεργαστών και των μονάδων επεξεργασίας γραφικών (GPU), η διαφορά μεταξύ του κόστους των πραγματικών υπολογισμών και του κόστους επικοινωνίας εξαιτίας της οδήγησης γίνεται πιο κρίσιμο. Έτσι, για πολυπύρηνους υπολογιστές εφαρμόζεται η τεχνική της οδήγησης κατά ζεύγη (pairwise pivoting) η οποία προκαλεί αντιμεταθέσεις γραμμών σε ζεύγη υποπινάκων (blocks), παρόλο που η επιβάρυνση που εισάγεται είναι επίσης σημαντική [43]. Επίσης, σε αρχιτεκτονικές με πολλά νήματα (multithread) έχει προταθεί η τεχνική της αυξανόμενης οδήγησης (incremental pivoting) [44]. Μια κατηγορία αλγορίθμων που

αποκαλούνται αλγόριθμοι βέλτιστης επικοινωνίας (communication optimal) δημιουργήθηκε με την τεχνική των Grigori και άλλοι [45]. Η τεχνική αυτή ελαχιστοποιεί τον αριθμό των μηνυμάτων που ανταλλάσσονται κατά την διάρκεια της παραγοντοποίησης. Στο χώρο των υλοποιήσεων με κάρτες γραφικών, η επιβάρυνση της οδήγησης έχει περιοριστεί με την χρήση καινοτόμων δομών δεδομένων [46].

Η διαφορά της μεθόδου RBT σε σχέση με τις περισσότερες υλοποιήσεις που συναντώνται στην βιβλιογραφία, είναι ότι μπορεί να αποφεύγει τελείως την διαδικασία της οδήγησης, που όπως είδαμε έχει επιβάρυνση κάποιες φορές αντίστοιχή με την πολυπλοκότητα του ίδιου του αλγορίθμου της απαλοιφής του Gauss. Για να εφαρμοστεί ο μετασχηματισμός RBT απαιτούνται δυο επιμέρους διαδικασίες: η δημιουργία των «τυχαίων» αναδρομικών πινάκων πεταλούδας και ο πολλαπλασιασμός τους με τον πίνακα A. Χρησιμοποιώντας την «συμπαγή» αναπαράσταση οι πίνακες πεταλούδας μπορούν να δημιουργηθούν και να αποθηκευθούν αποδοτικά, ενώ επιπρόσθετα, μπορούν να υπολογίζονται offline και να χρησιμοποιούνται οι ίδιοι πίνακες για την επίλυση διαφορετικών συστημάτων. Τέλος, η διαδικασία του πολλαπλασιασμού πινάκων είναι ιδιαίτερα αποδοτική σε παράλληλες αρχιτεκτονικές, αφού μπορεί να κατανεμηθεί και να εκτελεστεί ανεξάρτητα από τους επεξεργαστές, ενώ μπορεί να επιταχυνθεί σημαντικά σε διανυσματικούς υπολογιστές.

3.5 Βασικά στοιχεία υλοποίησης της μεθόδου RBT σε κάρτες γραφικών (GPUs)

Παρακάτω θα σκιαγραφήσουμε την υλοποίηση της μεθόδου RBT σε κάρτες γραφικών όπως προτείνεται από τους Baboulin και άλλοι [11].

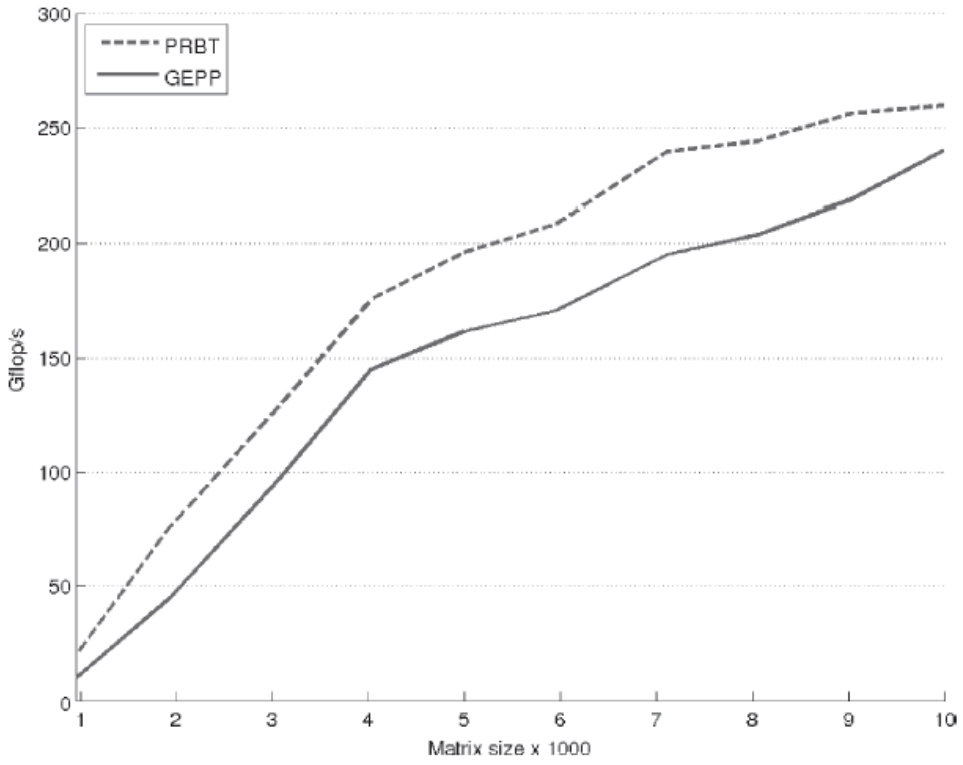
Αρχικά, ένα αντίγραφο του πίνακα A δημιουργείται στην μνήμη της κάρτας γραφικών και ενημερώνεται κατά την διάρκεια δημιουργίας των τυχαίων πινάκων. Ο υπολογισμός του γινομένου $B^T AB'$ γίνεται χρησιμοποιώντας $\frac{n^2}{4}$ ανεξάρτητα νήματα (threads) που αντιστοιχούν στις συντεταγμένες (i,j) με $i,j \leq \frac{n}{2}$. Κάθε νήμα διαβάζει και ενημερώνει 4 τιμές του πίνακα A: $A(i,j), A\left(i+\frac{n}{2},j\right), A\left(i,j+\frac{n}{2}\right)$ και $A\left(i+\frac{n}{2},j+\frac{n}{2}\right)$. Αυτό έχει σαν αποτέλεσμα η ανάγνωση και η εγγραφή του πίνακα να γίνεται μόνο μια φορά. Το κόστος υπολογισμών είναι σχετικά μικρό, ενώ η πραγματική επιβάρυνση προκύπτει από το κόστος των εγγραφών / αναγνώσεων από την μνήμη. Για να υπολογίσουμε έναν αναδρομικό πίνακα πεταλούδας με βάθος αναδρομής 2 απαιτούνται 5 επιμέρους γινόμενα για τον σχηματισμό του $B^T AB'$ (μεταξύ ενός πίνακα με βάθος αναδρομής 1 και τεσσάρων βάθους 2). Αυτά τα 5 γινόμενα μπορούν να υπολογιστούν με μια μόνο ανάγνωση και εγγραφή στην μνήμη της κάρτας. Ωστόσο, εξαιτίας της περιορισμένης κρυφής μνήμης (cache memory) της κάρτας γραφικών μια υλοποίηση με 2 διαδοχικές αναγνώσεις και εγγραφές ενδέχεται να είναι πιο αποδοτική.

Η μέθοδος RBT υλοποιημένη σε μια υβριδική αρχιτεκτονική με CPU και GPU πραγματοποιεί τις ακόλουθες διαδικασίες:

Αρχικά δημιουργούνται οι πίνακες U και V (οι B και B') σε συμπαγή μορφή στην CPU.
Ο πίνακας A και οι πίνακες U και V σε συμπαγή μορφή στέλνονται από την κύρια μνήμη στην μνήμη της κάρτας γραφικών.
Ο πίνακας A ενημερώνεται στην κάρτα γραφικών και υπολογίζεται το γινόμενο $B^T AB'$ σε αυτήν.

Η φάση της παραγοντοποίησης γίνεται παράλληλα στην κάρτα γραφικών και στην CPU με την χρήση της μεθόδου απαλοιφής του Gauss χωρίς οδήγηση.
Στην κάρτα γραφικών υπολογίζεται το γινόμενο $U^T b$ και επιλύεται το ενδιάμεσο γραμμικό σύστημα. Τέλος, η κάρτα γραφικών υπολογίζει την λύση $x = Vy$.
Η λύση του αρχικού συστήματος μεταφέρεται στην κύρια μνήμη του υπολογιστή.

Στο σχήμα 7 που ακολουθεί παρουσιάζεται η απόδοση σε Gflops/sec (10^9 πράξεις κινητής υποδιαστολής / δευτερόλεπτο) της μεθόδου RBT και της μεθόδου απαλοιφής του Gauss με μερική οδήγηση, σε υβριδική αρχιτεκτονική υλοποιημένη από τους Baboulin και άλλοι [11].



Σχήμα 7: Απόδοση των μεθόδων RBT και απαλοιφής του Gauss με μερική οδήγηση σε υβριδική αρχιτεκτονική. Πηγή: [11].

Στην συγκεκριμένη υλοποίηση, χρησιμοποιήθηκε ένας υβριδικός υπολογιστής με αριθμητική διπλής ακρίβειας αποτελούμενος από επεξεργαστή AMD Opteron 6172 (2.1 GHz) με 48 συνολικά πυρήνες (4 sockets x 12 cores) και κάρτα γραφικών Fermi Tesla S2050 (1.15 GHz, 2687.4 MB memory).

Παρατηρούμε, ότι για μικρά προβλήματα η απόδοση του RBT είναι σχεδόν διπλασία από την μέθοδο απαλοιφής του Gauss με μερική οδήγηση. Σε πίνακες με διάσταση μέχρι 3.000 η απόδοση της RBT είναι 33% μεγαλύτερη από την μέθοδο απαλοιφής του Gauss με μερική οδήγηση, ενώ σε πίνακες μεγάλης διάστασης πλησιάζει το 10%.

3.6 Αλγοριθμική περιγραφή της μεθόδου RBT

Σε αυτό το σημείο, θα παρουσιάσουμε σε μορφή ψευδοκώδικα τον αλγόριθμο για την εφαρμογή του μετασχηματισμού RBT σε ένα γραμμικό σύστημα. Σαν είσοδος απαιτείται ο πίνακας A και η στήλη των σταθερών όρων b του συστήματος Ax = b. Το βάθος αναδρομής μπορεί να οριστεί από τον χρήστη από την παράμετρο depth.

```
procedure Random_Butterfly_Transformation(matrix A, vector b)

//προσδιορισμός του βάθους αναδρομής από 1 έως log2N. Συνήθως ίσο με 2
int depth = 2;

//δημιουργία των αναδρομικών πινάκων πεταλούδας
matrix U = generateRecursiveButterfly(size(A), depth);
matrix V = generateRecursiveButterfly(size(A), depth);

//επίλυση του ενδιάμεσου συστήματος με την μέθοδο απαλοιφής του Gauss χωρίς
//οδήγηση
vector y = nakedGE(U*A*V,U*b); //ο αλγόριθμος δίνεται στο κεφάλαιο 1

//η λύση του αρχικού συστήματος δίνεται από το γινόμενο V*y
vector x = V*y;

return x;

end of procedure
```

Ενδιαφέρον παρουσιάζει η αναδρομική συνάρτηση generateRecursiveButterfly() με ορίσματα την διάσταση του πίνακα και το βάθος αναδρομής. Εύλογα γεννάται το ερώτημα πως αντιμετωπίζεται η περίπτωση που η διάσταση του A δεν είναι δύναμη του 2, έτσι ώστε να παραχθούν οι αντίστοιχοι πίνακες πεταλούδας. Σε αυτή την περίπτωση, πρέπει να επαυξήσουμε τους πίνακες A και b προσθέτοντας «εικονικούς» αγνώστους έτσι ώστε η διάσταση του συστήματος να είναι δύναμη του 2 (για λόγους απλότητας δεν θα παρουσιάσουμε εδώ αυτή τη βελτίωση). Παρακάτω ακολουθεί ο ψευδοκώδικας της αναδρομικής συνάρτησης generateRecursiveButterfly():

```
function generateRecursiveButterfly(int size, int depth)
//ακολουθεί η περιγραφή σύμφωνα με την μεθοδολογία του Parker [19]

//προσδιορισμός του εύρους [α,β] των τυχαίων στοιχείων
int min = -1/2;
int max = 1/2;

if depth > 0 then //για κάθε επίπεδο αναδρομής
matrix rbm = (generateRecursiveButterfly(size/2 , depth -1) ⊕
generateRecursiveButterfly(size/2 , depth -1))
*generateButterfly(size,min,max);
return rbm;
else //τελευταίο επίπεδο: δημιούργησε μια πεταλούδα διάστασης 1
   double r = exp(getRandom(min,max)/10);
   return r;
end if

end of function
```

Τέλος, πρέπει να δημιουργήσουμε και την συνάρτηση generateButterfly() η οποία δέχεται σαν ορίσματα την διάσταση του πίνακα πεταλούδα που δημιουργεί και το εύρος της κατανομής των τυχαίων στοιχείων από τα οποία αποτελείται:

```
function generateButterfly (int size, int min, int max)
//ακολουθεί η περιγραφή σύμφωνα με την μεθοδολογία του Parker [19]

//δημιουργία των διαγώνιων υποπινάκων R0 και R1 διάστασης size/2 X size/2
matrix R0 = generateDiagonal(size/2,exp(getRandom(min,max)/10));
matrix R1 = generateDiagonal(size/2,exp(getRandom(min,max)/10));

//υπολογισμός του πίνακα πεταλούδας διάστασης size X size
matrix B = 1/√2 (R0   R1
                 R0  −R1);

return B;

end of function
```

4. Πειραματικά αποτελέσματα υλοποίησης της μεθόδου RBT

4.1 Περιβάλλον δοκιμής της μεθόδου RBT

Στο πίνακα που ακολουθεί, παρουσιάζονται τα πειραματικά δεδομένα που προέκυψαν από την υλοποίηση και εκτέλεση της μεθόδου RBT στο περιβάλλον MatLab 7.12.0 (R2011a) [47]. Για τις δοκιμές χρησιμοποιήθηκε ένας desktop ηλεκτρονικός υπολογιστής με λειτουργικό Microsoft Windows 7 Professional 64bit με επεξεργαστή Intel Core i7 920 στα 2,67Ghz και 6GB DDR3 κύριας μνήμης.

Για λόγους συνέπειας, θα δημιουργήσουμε τις ακόλουθες 11 κλάσεις δοκιμαστικών πινάκων (test matrices), όπως παρουσιάζονται στις αντίστοιχες δοκιμές που πραγματοποίησε ο Parker [18] [17]. Οι πρώτες πέντε κλάσεις αφορούν τυχαίους πίνακες [48], ενώ οι υπόλοιπες κλάσεις πινάκων [49] αφορούν και ορισμένες περιπτώσεις πινάκων που ξέρουμε ότι εμφανίζουν πολύ μεγάλο αριθμό συνθήκης, όποτε μπορούμε να εκτιμήσουμε την συμπεριφορά της μεθόδου σε περιπτώσεις που της δοθεί σαν είσοδος ένας τέτοιος πίνακας.

Πίνακας 1: Κλάσεις δοκιμαστικών πινάκων

Όνομα κλάσης	Στοιχεία πινάκων A,b	Ακριβής λύση X
normal	a_{ij}, b_i από κανονική τυχαία κατανομή με μ=0, σ=1	-
[-1, 1]	a_{ij}, b_i από ομοιόμορφη τυχαία κατανομή στο διάστημα [-1, 1]	-
[0, 1]	a_{ij}, b_i από ομοιόμορφη τυχαία κατανομή στο διάστημα [0, 1]	-
{-1, 1}	a_{ij}, b_i από διακριτή ομοιόμορφη τυχαία κατανομή με $\mathrm{pr}(-1) = \mathrm{pr}(1) = \frac{1}{2}$	-
{0, 1}	a_{ij}, b_i από διακριτή ομοιόμορφη τυχαία κατανομή με $\mathrm{pr}(0) = \mathrm{pr}(1) = \frac{1}{2}$	-
Όνομα κλάσης	Στοιχεία πινάκων A,b	Ακριβής λύση X
|i-j|	$a_{ij} = \lvert i-j \rvert$ και $b_i = 1$	$x = \left(\frac{1}{n-1}, 0, \dots, 0, \frac{1}{n-1}\right)$
max(i,j)	$a_{ij} = \max(i,j)$ και $b_i = i$	$x = (1,0,\dots,0)$
C(i+j,j)	$a_{ij} = \binom{i+j-2}{j-1}$ και $b_i = 1$	$x = (1,0,\dots,0)$
Hadamard	$A = \bigotimes^{log_2(n)} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ και $b_i = 1$	$x = (1,0,\dots,0)$

Permute	Α τυχαία αντιμετάθεση του I_n, $(A^{-1}b)_i = i$	$x = (1,2,3,\dots,n)$
Turing	$a_{ij} = \begin{cases} 0 & i < j \\ 1 & i = j \\ -1 & i > j \end{cases}$ και $b_i = 1$[1]	$x = (1,2,4,\dots,2^{n-1})$

Οι δοκιμές που πραγματοποιήθηκαν αφορούν τις 11 κλάσεις πινάκων που αναφέρονται στον πίνακα 1. Για να εκτιμήσουμε την ποιότητα των αποτελεσμάτων που προέκυψαν ύστερα από την εφαρμογή της μεθόδου RBT, για κάθε κλάση δοκιμαστικών πινάκων θα δημιουργήσουμε τυχαίους πίνακες διάστασης 32×32, 64×64, 128×128, 256×256 και 512×512. Για κάθε κλάση και για κάθε διάσταση των πινάκων επαναλαμβάνεται η επίλυση του γραμμικού συστήματος για 256, 128, 64, 32 και 20 φορές αντίστοιχα, έτσι ώστε να εξάγουμε έναν μέσο χρόνο εκτέλεσης και μια εκτίμηση του μέσου, ελάχιστου και μέγιστου σφάλματος της λύσης του γραμμικού συστήματος σε σχέση με την λύση που θα παράγει η ενσωματωμένη ρουτίνα επίλυσης γραμμικών συστημάτων του MatLab. Για τις 6 κλάσεις δοκιμαστικών πινάκων για τις οποίες γνωρίζουμε την ακριβή τους λύση, θα συγκρίνουμε την λύση της μεθόδου RBT με την λύση της ρουτίνας του MatLab αλλά και με την γνωστή ακριβή λύση.

Επιπρόσθετα, για κάθε δοκιμή με τους δοκιμαστικούς πίνακες, καταγράφεται ο μέσος αριθμός συνθήκης των πινάκων που παρήχθησαν.

Για ορισμένες κλάσεις (βλ. πίνακα 1) πινάκων γνωρίζουμε εκ των προτέρων την ακριβή λύση τους, συνεπώς μπορούμε να εξακριβώσουμε το μέσο και μέγιστο σφάλμα της λύσης που προέκυψε από τον RBT σε σχέση με την ακριβής λύση.

Στηριζόμενοι στις παρατηρήσεις των Baboulin και άλλοι [11] [20], θα εκτελέσουμε επιπρόσθετα και μια σειρά δοκιμών οι οποίες θα αφορούν το βάθος αναδρομής κατά τον σχηματισμό των αναδρομικών πινάκων πεταλούδας. Ο Baboulin παρατήρησε ότι στην πράξη το βάθος της αναδρομής μπορεί να περιοριστεί σε 1 ή 2 επίπεδα αναδρομής και όχι απαραίτητα σε $\log_2 n$ όπως γίνεται στην αρχική διατύπωση της μεθόδου από τον Parker [17] [18] [19]. Για τον λόγο αυτό, κάθε δοκιμή από τις αντίστοιχες κλάσεις δοκιμαστικών πινάκων, θα πραγματοποιηθεί με 1, 2 ή $\log_2 n$ επίπεδα αναδρομής.

Ο Parker στα πειράματα που διεξήγαγε με τις ίδιες κλάσεις δοκιμαστικών πινάκων [17] [18] χρησιμοποίησε τυχαίους διαγώνιους πίνακες με στοιχεία της μορφής exp(r/10), όπου r τυχαίος αριθμός ομοιόμορφα κατανεμημένος στο διάστημα [-1/2, 1/2]. Το διάστημα από το οποίο μπορεί να πάρει τιμές ο r μπορεί να θεωρηθεί σαν παράμετρος για τον αλγόριθμο. Ωστόσο, σύμφωνα με τον Parker [17] μια (περιορισμένη) μεταβολή του διαστήματος δεν πρόκειται να επιφέρει αξιοσημείωτη μεταβολή στα σφάλματα στογγύλευσης που θα προκύψουν. Για λόγους συνέπειας, στην παρούσα υλοποίηση θα ακολουθήσουμε την ίδια μεθοδολογία με τον Parker.

4.2 Πειραματικά δεδομένα της μεθόδου RBT με την μέθοδο απαλοιφής του Gauss

[1] Wilkinson's test

Σε αυτή την ενότητα παρουσιάζονται τα αριθμητικά δεδομένα που προέκυψαν από την μέθοδο RBT σε 11 δοκιμαστικές κλάσεις πινάκων.

Ο πίνακας 2 που ακολουθεί, περιέχει 5 κλάσεις πινάκων για τις οποίες δεν γνωρίζουμε την ακριβή τους λύση, ενώ στον πίνακα 3 παρουσιάζονται οι υπόλοιπες 6 κλάσεις πινάκων για τις οποίες γνωρίζουμε την ακριβή τους λύση:

Πίνακας 2: Πειραματικά δεδομένα για πίνακες με μη γνωστή λύση με την μέθοδο απαλοιφής του Gauss

Κλάση δοκιμαστικού πίνακα	Διάσταση πίνακα (N)	Πλήθος επαναλήψεων	Βάθος αναδρομής	Μέσος χρόνος RBT (sec)	Μέσος χρόνος MatLab Solver (sec)	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο condition number
normal	32	256	1	0.0011432	5.5511e-005	9.1708e-011	1.1755e-012	3.2237e-010	615.51
normal	64	128	1	0.0030065	0.00013449	9.0551e-012	2.7183e-011	1.5533e-013	414.3421
normal	128	64	1	0.016396	0.0004921	3.4882e-010	3.6005e-012	1.2508e-009	3040.4807
normal	256	32	1	0.11469	0.002403	6.7146e-010	3.8509e-012	2.7082e-009	4745.9239
normal	512	20	1	2.6529	0.0093107	1.2984e-010	3.1185e-013	5.2678e-010	2833.5956
normal	32	256	2	0.0017385	5.5225e-005	4.1455e-009	4.5553e-011	1.3964e-008	90096.3417
normal	64	128	2	0.0039468	0.0001445	4.2195e-009	1.9914e-010	1.4718e-008	106623.3356
normal	128	64	2	0.016372	0.00045531	1.8742e-009	9.6802e-012	7.549e-009	119106.0205
normal	256	32	2	0.11686	0.0023475	5.0707e-010	2.8332e-012	1.9453e-009	45153.6024
normal	512	20	2	2.6694	0.010248	2.7744e-008	9.4957e-011	1.0299e-007	125006.0438
normal	32	256	$\log_2 N$	0.0082896	5.5755e-005	1.3045e-011	9.7547e-013	3.6772e-011	370.5733
normal	64	128	$\log_2 N$	0.018679	0.00013592	6.4735e-008	2.3401e-009	2.1319e-007	7850.9934
normal	128	64	$\log_2 N$	0.049139	0.00045627	1.5622e-010	6.1031e-012	6.3592e-010	971.6853
normal	256	32	$\log_2 N$	0.18224	0.00247	1.9568e-010	5.5655e-013	8.2591e-010	3498.1218
normal	512	20	$\log_2 N$	2.7989	0.010478	6.786e-008	3.3378e-011	3.2406e-007	19192.3494
[-1, 1]	32	256	1	0.0011362	5.5131e-005	1.0489e-010	1.1547e-011	2.6852e-010	502.3242
[-1, 1]	64	128	1	0.0031337	0.00014016	4.8209e-011	2.6643e-012	1.4491e-010	635.4962
[-1, 1]	128	64	1	0.016076	0.00046631	2.5885e-011	4.9386e-013	9.0035e-011	1004.1611
[-1, 1]	256	32	1	0.11315	0.0023598	5.4271e-011	6.1362e-013	1.7839e-010	1657.3861
[-1, 1]	512	20	1	2.6558	0.010078	1.2991e-010	3.1082e-013	4.8865e-010	2996.2544
[-1, 1]	32	256	2	0.0016589	5.4674e-005	1.7584e-009	9.1943e-011	4.3367e-009	68897.9457
[-1, 1]	64	128	2	0.0041157	0.00013536	3.3702e-008	2.1799e-009	8.7664e-008	192540.2483

[-1, 1]	128	64	2	0.016604	0.00044873	1.459e-008	1.2893e-010	5.1997e-008	131832.1266
[-1, 1]	256	32	2	0.11753	0.0023663	1.4684e-008	4.1784e-011	6.5226e-008	202259.521
[-1, 1]	512	20	2	2.6701	0.010401	1.5355e-008	3.7319e-011	6.3355e-008	157708.2615
[-1, 1]	32	256	$\log_2 N$	0.0085664	5.5515e-005	2.7045e-011	2.6888e-012	6.2449e-011	400.889
[-1, 1]	64	128	$\log_2 N$	0.01902	0.00013728	1.3956e-010	4.4704e-012	4.0752e-010	1621.9251
[-1, 1]	128	64	$\log_2 N$	0.04936	0.00045326	2.7223e-010	3.6651e-012	1.0125e-009	1517.9054
[-1, 1]	256	32	$\log_2 N$	0.18328	0.0024418	1.0884e-010	2.8145e-013	4.0477e-010	2356.1013
[-1, 1]	512	20	$\log_2 N$	2.807	0.011107	3.5458e-009	2.6753e-012	1.4316e-008	10705.7591
[0, 1]	32	256	1	0.001137	5.5244e-005	2.6452e-012	2.0397e-013	7.3323e-012	1306.6284
[0, 1]	64	128	1	0.0032658	0.00014687	7.1914e-009	2.8094e-010	2.5381e-008	32091.2501
[0, 1]	128	64	1	0.016025	0.00047163	2.1525e-010	4.8343e-012	7.3572e-010	24040.4658
[0, 1]	256	32	1	0.11288	0.0023482	1.2971e-010	4.2889e-012	4.5492e-010	39413.9851
[0, 1]	512	20	1	2.6579	0.010123	1.4325e-009	1.1908e-012	6.1351e-009	73184.5399
[0, 1]	32	256	2	0.0017295	5.6147e-005	8.6982e-009	3.2786e-01	2.4698e-008	1317355.9785
[0, 1]	64	128	2	0.0037606	0.00013934	6.2968e-009	1.1386e-010	2.0865e-008	616108.3122
[0, 1]	128	64	2	0.01683	0.00046121	2.1174e-008	1.7901e-010	6.5939e-008	1099843.7307
[0, 1]	256	32	2	0.11744	0.0023838	2.9509e-009	9.7341e-012	1.1145e-008	939598.8541
[0, 1]	512	20	2	2.6721	0.0098359	1.5597e-008	8.8857e-012	6.0176e-008	1753740.3342
[0, 1]	32	256	$\log_2 N$	0.0084688	5.5024e-005	1.4862e-010	7.0954e-013	3.7255e-010	5693.4383
[0, 1]	64	128	$\log_2 N$	0.01954	0.00013834	4.5565e-011	4.2986e-013	1.5417e-010	3830.9339
[0, 1]	128	64	$\log_2 N$	0.051761	0.00048879	7.785e-009	2.5664e-010	2.2521e-008	52578.4421
[0, 1]	256	32	$\log_2 N$	0.18565	0.0024362	2.0195e-008	2.2623e-011	8.0873e-008	135870.4295
[0, 1]	512	20	$\log_2 N$	2.8003	0.01014	1.0343e-010	2.08e-013	4.273e-010	44631.0084
{-1, 1}	32	256	1	0.0011282	5.4756e-005	1.2405e-010	5.3314e-012	3.2932e-010	394.4554
{-1, 1}	64	128	1	0.0030297	0.0001358	7.6005e-011	9.9078e-013	2.1614e-010	575.0601
{-1, 1}	128	64	1	0.014853	0.00043525	4.9118e-010	1.9055e-011	1.4349e-009	1794.5096
{-1, 1}	256	32	1	0.11966	0.0060775	7.3001e-011	3.7778e-013	2.6643e-010	2512.982
{-1, 1}	512	20	1	2.6673	0.010233	7.6833e-010	6.5734e-013	3.0426e-009	6856.82
{-1, 1}	32	256	2	0.0016674	5.443e-005	5.2539e-008	6.5202e-010	1.281e-007	109507.7209
{-1, 1}	64	128	2	0.003654	0.00013507	1.3254e-009	2.1088e-011	4.4211e-009	86265.8514
{-1, 1}	128	64	2	0.015795	0.00043221	1.8959e-008	1.4833e-010	9.6905e-008	128088.6594

{-1, 1}	256	32	2	0.11262	0.0021706	6.5882e-009	6.5531e-011	2.4426e-008	97359.0053
{-1, 1}	512	20	2	2.6722	0.0092963	3.1587e-009	5.2347e-012	1.2476e-008	64491.4284
{-1, 1}	32	256	$\log_2 N$	0.0084227	5.5428e-005	2.0491e-011	6.7548e-013	6.3469e-011	320.1263
{-1, 1}	64	128	$\log_2 N$	0.019773	0.00014742	3.1735e-011	5.3468e-013	9.5246e-011	907.0328
{-1, 1}	128	64	$\log_2 N$	0.048189	0.00044777	1.4586e-011	1.711e-013	4.9737e-011	748.2975
{-1, 1}	256	32	$\log_2 N$	0.17655	0.0023204	8.0293e-009	2.4699e-011	3.149e-008	11741.4786
{-1, 1}	512	20	$\log_2 N$	2.8066	0.0091062	1.9506e-008	5.7553e-011	1.0356e-007	8789.892
{0, 1}	32	256	1	0.0011224	5.4517e-005	2.63e-011	1.6326e-012	7.434e-011	1583.1498
{0, 1}	64	128	1	0.0030409	0.000132	8.4105e-012	2.2642e-013	2.8629e-011	2266.8261
{0, 1}	128	64	1	0.01504	0.00044403	1.8261e-010	5.8301e-012	6.046e-010	13695.1294
{0, 1}	256	32	1	0.11265	0.0021364	2.4248e-010	1.8274e-012	8.5716e-010	21757.8324
{0, 1}	512	20	1	2.6656	0.009111	1.4267e-009	1.1054e-012	6.027e-009	30978.1539
{0, 1}	32	256	2	0.0016692	5.4598e-005	5.8537e-010	1.4713e-011	1.5925e-009	212510.8594
{0, 1}	64	128	2	0.0035848	0.00013161	6.3096e-009	2.8929e-010	2.3065e-008	396644.8669
{0, 1}	128	64	2	0.015974	0.00043226	2.9883e-008	6.8744e-011	9.7483e-008	502234.6679
{0, 1}	256	32	2	0.12347	0.0027333	4.9393e-009	8.6073e-011	2.0737e-008	392547.1802
{0, 1}	512	20	2	2.687	0.0089787	3.6179e-009	1.486e-011	1.4989e-008	646390.6515
{0, 1}	32	256	$\log_2 N$	0.0083638	5.4909e-005	5.2521e-012	2.4519e-013	1.3659e-011	1274.7069
{0, 1}	64	128	$\log_2 N$	0.018695	0.00013419	5.8127e-009	2.052e-010	1.7353e-008	25988.1097
{0, 1}	128	64	$\log_2 N$	0.048217	0.00045241	1.6466e-011	2.29e-013	5.9806e-011	6395.2943
{0, 1}	256	32	$\log_2 N$	0.17492	0.0021263	4.318e-010	9.494e-011	1.5063e-009	23822.1378
{0, 1}	512	20	$\log_2 N$	2.8072	0.0093451	6.9181e-009	3.5092e-011	2.9736e-008	90094.7048

Πίνακας 3: Πειραματικά δεδομένα για πίνακες με γνωστή λύση με την μέθοδο απαλοιφής του Gauss

Κλάση δοκιμα-στικού πίνακα	Διά-σταση πίνακα (N)	Πλήθος επαναλή-ψεων	Βάθος αναδρ ομής	Μέσος χρόνος RBT	Μέσος χρόνος MatLab Solver	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο σφάλμα |RBT - Exact|	Μέσο ελάχιστο σφάλμα |RBT - Exact|	Μέσο μέγιστο σφάλμα |RBT - Exact|	Μέσο condition number
|i-j|	32	256	1	0.0011485	5.5475e-005	2.2597e-016	9.9137e-018	6.5991e-016	1.423e-016	7.5986e-018	3.8926e-016	708.9846
|i-j|	64	128	1	0.0029895	0.00014206	6.1932e-016	8.7343e-018	1.8814e-015	1.5213e-016	3.4092e-018	4.7606e-016	2843.4584
|i-j|	128	64	1	0.01579	0.00052858	3.7677e-016	3.8726e-018	1.5142e-015	1.7505e-016	2.3224e-018	5.9963e-016	11381.3634
|i-j|	256	32	1	0.11324	0.0020052	3.6755e-016	1.7509e-018	1.5599e-015	2.1145e-016	1.1315e-018	7.9579e-016	45532.9858
|i-j|	512	20	1	2.7118	0.0094049	2.7018e-015	6.0008e-018	1.5058e-014	2.5399e-015	6.6114e-018	1.1329e-014	182139.4756
|i-j|	32	256	2	0.0018617	5.8212e-005	2.4102e-016	1.0383e-017	6.9891e-016	1.6386e-016	9.4831e-018	4.255e-016	708.9846
|i-j|	64	128	2	0.004052	0.00014896	6.4417e-016	1.084e-017	1.9061e-015	1.8172e-016	5.1661e-018	5.265e-016	2843.4584
|i-j|	128	64	2	0.017733	0.00054431	3.945e-016	4.052e-018	1.5697e-015	1.9792e-016	2.3886e-018	6.427e-016	11381.3634
|i-j|	256	32	2	0.12211	0.0020774	3.7306e-016	2.3287e-018	1.6003e-015	2.2493e-016	1.3303e-018	8.255e-016	45532.9858
|i-j|	512	20	2	2.7094	0.010231	2.0395e-015	1.0785e-017	1.032e-014	1.8693e-015	5.8288e-018	6.1258e-015	182139.4756
|i-j|	32	256	$\log_2 N$	0.0084086	5.7348e-005	1.8684e-016	6.4369e-018	5.8213e-016	7.4474e-017	2.9971e-018	2.3128e-016	708.9846
|i-j|	64	128	$\log_2 N$	0.018544	0.00014275	5.939e-016	4.4757e-018	1.8138e-015	8.4916e-017	1.7457e-018	3.0763e-016	2843.4584

|i-j|	128	64	$\log_2 N$	0.047615	0.00050253	3.4153e-016	2.6225e-018	1.4301e-015	9.9951e-017	1.1765e-018	4.4099e-016	11381.3634
|i-j|	256	32	$\log_2 N$	0.18279	0.0020084	3.1951e-016	1.5347e-018	1.479e-015	1.2617e-016	6.1209e-019	6.0711e-016	45532.9858
|i-j|	512	20	$\log_2 N$	2.7926	0.0087146	1.0607e-015	1.6846e-018	9.5362e-015	8.5285e-016	2.192e-018	5.2164e-015	182139.4756
max(i,j)	32	256	1	0.001138	5.5045e-005	1.8772e-016	1.1779e-017	4.8061e-016	0.032227	1.3579e-017	1	2900.0064
max(i,j)	64	128	1	0.0030697	0.00013727	2.2092e-016	5.4789e-018	6.4133e-016	0.015869	6.0956e-018	1	11499.7772
max(i,j)	128	64	1	0.016344	0.00057462	2.301e-016	2.6707e-018	7.747e-016	0.0078735	2.6707e-018	1	45775.0249
max(i,j)	256	32	1	0.11804	0.0020787	2.7136e-016	1.5853e-018	1.0008e-015	0.0039215	1.5853e-018	1	182628.7113
max(i,j)	512	20	1	2.7357	0.0097746	4.6607e-014	1.8983e-016	1.9036e-013	0.0019569	1.8983e-016	1	729549.0367
max(i,j)	32	256	2	0.0016916	5.4571e-005	1.8326e-016	1.8159e-017	4.0592e-016	0.032227	2.5464e-017	1	2900.0064
max(i,j)	64	128	2	0.0036477	0.00013121	1.9171e-016	7.8867e-018	5.1424e-016	0.015869	8.7222e-018	1	11499.7772
max(i,j)	128	64	2	0.017954	0.00058844	2.0524e-016	1.8443e-018	6.1277e-016	0.0078735	2.1517e-018	1	45775.0249
max(i,j)	256	32	2	0.12041	0.0020691	2.4373e-016	1.2627e-018	8.697e-016	0.0039215	1.2627e-018	1	182628.7113
max(i,j)	512	20	2	2.7153	0.0092273	2.7021e-014	2.2231e-017	1.0142e-013	0.0019569	2.2231e-017	1	729549.0367
max(i,j)	32	256	$\log_2 N$	0.0085429	5.5334e-005	3.3925e-017	1.8182e-018	1.1226e-016	0.032227	2.1451e-018	1	2900.0064
max(i,j)	64	128	$\log_2 N$	0.018919	0.00013687	4.1111e-017	1.4349e-018	1.6259e-016	0.015869	1.7546e-018	1	11499.7772
max(i,j)	128	64	$\log_2 N$	0.049282	0.00054758	5.7672e-017	1.0731e-018	2.2671e-016	0.0078735	1.0953e-018	1	45775.0249

max(i,j)	256	32	$\log_2 N$	0.18108	0.0019482	3.3732e-016	4.0859e-019	8.0405e-017	0.0039215	4.0859e-019	1	182628.7113
max(i,j)	512	20	$\log_2 N$	2.7948	0.00895	2.2458e-016	5.8469e-019	1.2896e-015	0.0019569	5.8469e-019	1	729549.0367
C(i+j,j)	32	256	1	0.0011449	7.6539e-005	0.33896	4.0682e-005	1.9076	0.33896	4.0682e-005	1.9076	1.986698e+025
C(i+j,j)	64	128	1	0.003243	0.00018911	0.015625	3.395e-018	1	0.015625	3.395e-018	1	1.937767e+035
C(i+j,j)	128	64	1	0.01629	0.00058183	0.0078125	1.2443e-051	1	0.0078125	1.2443e-051	1	2.438210e+045
C(i+j,j)	256	32	1	0.1147	0.0023965	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	2.060447e+060
C(i+j,j)	512	20	1	2.7188	0.013078	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
C(i+j,j)	32	256	2	0.0017418	7.7866e-005	0.7113	0.00026894	2.6935	0.7113	0.00026894	2.6935	1.986698e+025
C(i+j,j)	64	128	2	0.0037513	0.00018291	0.015625	7.584e-021	1	0.015625	7.584e-021	1	1.937767e+035
C(i+j,j)	128	64	2	0.017121	0.00062858	0.0078125	2.4305e-056	1	0.0078125	2.4305e-056	1	2.438210e+045
C(i+j,j)	256	32	2	0.11871	0.0024509	0.0039063	2.6628e-128	1	0.0039063	2.6628e-128	1	2.060447e+060
C(i+j,j)	512	20	2	2.7013	0.0099829	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
C(i+j,j)	32	256	$\log_2 N$	0.008622	7.6798e-005	0.70665	2.3022e-004	2.5697	0.70665	2.3022e-004	2.5697	1.986698e+025
C(i+j,j)	64	128	$\log_2 N$	0.019005	0.00018353	0.015625	2.7515e-022	1	0.015625	2.7515e-022	1	1.937767e+035
C(i+j,j)	128	64	$\log_2 N$	0.047117	0.00054442	0.0078125	1.8608e-060	1	0.0078125	1.8608e-060	1	2.438210e+045
C(i+j,j)	256	32	$\log_2 N$	0.17693	0.0022961	0.0039063	1.2409e-136	1	0.0039063	1.2409e-136	1	2.060447e+060

C(i+j,j)	512	20	$\log_2 N$	2.8224	0.011491	1.9531e-004	9.4733e-290	1	1.9531e-004	9.4733e-290	1	∞
Hadamard	32	256	1	0.001184 6	4.8083e-005	6.135335e-017	6.5849492e-019	2.44591e-016	6.135335e-017	6.5849492e-019	2.44591e-016	1
Hadamard	64	128	1	0.003162 9	0.00011523	5.301622e-017	2.133056e-019	2.41342e-01	5.301622e-017	2.133056e-019	2.41342e-01	1
Hadamard	128	64	1	0.016203	0.00040089	4.918947e-017	7.960061e-020	2.587961 4e-016	4.918947e-017	7.960061e-020	2.5879614e-016	1
Hadamard	256	32	1	0.11692	0.0018211	4.51183e-017	4.034758e-020	2.751024 e-016	4.51183e-017	4.034758e-020	2.751024e-016	1
Hadamard	512	20	1	2.6989	0.0090414	4.4487e-017	4.06099e-020	3.207321 e-016	4.4487e-017	4.06099e-020	3.207321e-016	1
Hadamard	32	256	2	0.001783 1	4.9946e-005	9.5694e-017	1.349e-018	3.6002e-016	9.5694e-017	1.349e-018	3.6002e-016	1
Hadamard	64	128	2	0.004235 8	0.00012347	7.8682e-017	5.2464e-019	3.5621e-016	7.8682e-017	5.2464e-019	3.5621e-016	1
Hadamard	128	64	2	0.018596	0.0004608	6.9016e-017	1.6987e-019	3.7224e-016	6.9016e-017	1.6987e-019	3.7224e-016	1
Hadamard	256	32	2	0.12549	0.0018627	6.4975e-017	7.9973e-020	3.6348e-016	6.4975e-017	7.9973e-020	3.6348e-016	1
Hadamard	512	20	2	2.845	0.010406	6.3626e-017	6.4189e-020	4.7578e-016	6.3626e-017	6.4189e-020	4.7578e-016	1
Hadamard	32	256	$\log_2 N$	0.008575 7	4.993e-005	2.3885e-016	9.0124e-018	7.6637e-016	2.3885e-016	9.0124e-018	7.6637e-016	1
Hadamard	64	128	$\log_2 N$	0.020116	0.0001312	2.8809e-016	4.5265e-018	1.1527e-015	2.8809e-016	4.5265e-018	1.1527e-015	1
Hadamard	128	64	$\log_2 N$	0.048495	0.00039898	3.6005e-016	3.1171e-018	1.6586e-015	3.6005e-016	3.1171e-018	1.6586e-015	1
Hadamard	256	32	$\log_2 N$	0.18177	0.0016727	5.0653e-016	2.2633e-018	2.8053e-015	5.0653e-016	2.2633e-018	2.8053e-015	1
Hadamard	512	20	$\log_2 N$	2.8514	0.0086295	5.8743e-016	1.8865e-018	3.8319e-015	5.8743e-016	1.8865e-018	3.8319e-015	1

Permute	32	256	1	0.0011865	4.7853e-005	0.001084542	4.2925e-014	0.006892031	0.001084542	4.2925e-014	0.006892031	1
Permute	64	128	1	0.0032032	0.00012126	2.1897e-010	9.8827e-012	8.9857e-010	2.1897e-010	9.8827e-012	8.9857e-010	1
Permute	128	64	1	0.015794	0.00041449	1.1908e-010	1.2927e-012	7.5427e-010	1.1908e-010	1.2927e-012	7.5427e-010	1
Permute	256	32	1	0.11442	0.0016485	5.2997e-011	9.005e-013	9.5188e-009	5.2997e-011	9.005e-013	9.5188e-009	1
Permute	512	20	1	2.6867	0.0082932	6.2521e-009	7.0095e-011	9.0812e-009	7.2521e-009	7.0095e-011	9.0812e-009	1
Permute	32	256	2	0.001777	4.8635e-005	0.00011516	1.2566e-013	0.00048511	0.00011516	1.2566e-013	0.00048511	1
Permute	64	128	2	0.0043875	0.00014363	0.0001798	6.0203e-014	0.0014391	0.0001798	6.0203e-014	0.0014391	1
Permute	128	64	2	0.018184	0.00046569	8.2384e-005	2.302e-015	0.00083227	8.2384e-005	2.302e-015	0.00083227	1
Permute	256	32	2	0.12765	0.0018683	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Permute	512	20	2	2.7363	0.0085117	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Permute	32	256	$\log_2 N$	0.008496	4.8073e-005	4.4075e-013	2.2429e-014	1.096e-012	4.4075e-013	2.2429e-014	1.096e-012	1
Permute	64	128	$\log_2 N$	0.019583	0.00012879	1.9283e-012	3.8265e-014	5.9637e-012	1.9283e-012	3.8265e-014	5.9637e-012	1
Permute	128	64	$\log_2 N$	0.048468	0.00042817	1.1997e-011	1.1027e-013	4.5447e-011	1.1997e-011	1.1027e-013	4.5447e-011	1
Permute	256	32	$\log_2 N$	0.18106	0.0016603	4.2432e-011	2.085e-013	1.5143e-010	4.2432e-011	2.085e-013	1.5143e-010	1
Permute	512	20	$\log_2 N$	2.8349	0.009028	3.2071e-010	7.8195e-013	1.297e-009	3.2071e-010	7.8195e-013	1.297e-009	1
Turing	32	256	1	0.0011805	4.7101e-005	21.648863	2.1087544e-007	346.38182	21.648863	2.1087544e-007	346.38182	2.78774e+010

Turing	64	128	1	0.0030161	0.00011609	2.867547e+017	3.2098734	9176151e+018	2.867547e+017	3.2098734	9176151e+018	1.11781e+019
Turing	128	64	1	0.015876	0.00038778	2.6584559e+037	2.53031892	1.70141183e+038	2.6584559e+037	2.53031892	1.70141183e+038	9.14668e+017
Turing	256	32	1	0.11527	0.001745 2	4.52312848e+050	2.3165759	5.7896044e+070	4.52312848e+050	2.3165759	5.7896044e+070	9.86347e+017
Turing	512	20	1	2.6913	0.007909 4	2.6187e+151	3.2062935	6.7039e+153	2.6187e+151	3.2062935	6.7039e+153	1.23420e+019
Turing	32	256	2	0.001708	4.6159e-005	21.0263	1.8999e-007	336.4214	21.0263	1.8999e-007	336.4214	2.78774e+010
Turing	64	128	2	0.0037993	0.00012248	2.87728e+017	1.9998	9.2073e+018	2.87728e+017	1.9998	9.2073e+018	1.11781e+019
Turing	128	64	2	0.016332	0.00037709	2.658455e+035	1.8075	1.7014e+037	2.658455e+035	1.8075	1.7014e+037	9.14668e+017
Turing	256	32	2	0.11389	0.0016351	4.52312e+039	1.0853	5.7896e+076	4.52312e+039	1.0853	5.7896e+076	9.86347e+017
Turing	512	20	2	2.6612	0.0071192	2.618712e+080	1.2251	6.703903e+090	2.618712e+080	1.2251	6.703903e+090	1.23420e+019
Turing	32	256	$\log_2 N$	0.0083865	4.683e-005	156.1744	1.2907e-006	2498.7899	156.1744	1.2907e-006	2498.7899	2.78774e+010
Turing	64	128	$\log_2 N$	0.019249	0.00012855	2.88255e+017	1.108	9.22418e+018	2.88255e+017	1.108	9.22418e+018	1.11781e+019
Turing	128	64	$\log_2 N$	0.049826	0.00041986	2.65845e+036	0.99281	1701411e+038	2.65845e+036	0.99281	1701411e+038	9.14668e+017
Turing	256	32	$\log_2 N$	0.18772	0.0019282	4.523128e+070	1.289	5.789604e+076	4.523128e+070	1.289	5.789604e+076	9.86347e+017
Turing	512	20	$\log_2 N$	2.9346	0.0094688	2.618712e+080	0.97634	6.703903e+090	2.618712e+080	0.97634	6.703903e+090	1.23420e+019

4.3 Πειραματικά δεδομένα της μεθόδου RBT με την μέθοδο απαλοιφής του Jordan

Στην συνέχεια, παρουσιάζονται τα αριθμητικά δεδομένα που προέκυψαν από την εφαρμογή της μεθόδου RBT στις ίδιες κλάσεις δοκιμαστικών πινάκων με την προηγούμενη ενότητα, μόνο που αυτή την φορά η επίλυση του ισοδύναμου ενδιάμεσου γραμμικού συστήματος έγινε με την μέθοδο απαλοιφής του Jordan όπως αυτή περιγράφηκε στο κεφάλαιο 1.6 . Η υλοποίηση της μεθόδου απαλοιφής του Jordan έγινε στο περιβάλλον του MatLab με σειριακή υλοποίηση όπως και στην περίπτωση των αριθμητικών πειραμάτων με την μέθοδο απαλοιφής του Gauss.

Στα αριθμητικά πειράματα με την μέθοδο απαλοιφής του Jordan χρησιμοποιήθηκαν αναδρομικοί πίνακες πεταλούδας με βάθος αναδρομής $d = 1$. Η επιλογή αυτή δικαιολογείται τόσο από την βιβλιογραφία και τις παρατηρήσεις των Baboulin, Dogarra και άλλοι [11] όσο και από τα αποτελέσματα που προέκυψαν σε αυτή την εργασία και παρουσιάζονται στα συμπεράσματα της εργασίας.

Πίνακας 4: Πειραματικά δεδομένα για πίνακες με μη γνωστή λύση με την μέθοδο απαλοιφής του Jordan

Κλάση δοκιμαστικού πίνακα	Διάσταση πίνακα (N)	Πλήθος επαναλήψεων	Βάθος αναδρομής	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο condition number
normal	32	256	1	4.2509e-011	9.8131e-013	1.1199e-010	440.2801
normal	64	128	1	6.8591e-012	1.7208e-013	2.2086e-011	365.5706
normal	128	64	1	4.5486e-011	3.788e-013	1.5655e-010	1438.1163
normal	256	32	1	3.9539e-010	3.3225e-012	1.5266e-009	2479.2996
normal	512	20	1	2.6568e-009	7.0608e-012	1.0365e-008	3749.5667
[-1, 1]	32	256	1	1.6832e-008	3.3414e-010	4.0768e-008	4039.6582
[-1, 1]	64	128	1	1.2742e-009	2.4957e-011	3.6707e-009	2180.3103
[-1, 1]	128	64	1	3.9763e-010	3.0639e-012	1.3245e-009	2488.6762
[-1, 1]	256	32	1	5.0979e-011	2.4812e-013	1.9361e-010	1823.9802
[-1, 1]	512	20	1	5.3566e-008	3.8025e-010	2.4765e-007	44997.574
[0, 1]	32	256	1	5.8399e-012	1.7121e-011	1.7121e-011	1428.1736
[0, 1]	64	128	1	6.2545e-012	2.2346e-013	2.1804e-011	4495.2175
[0, 1]	128	64	1	8.4001e-012	1.9826e-013	3.0651e-011	7528.8178
[0, 1]	256	32	1	2.125e-009	6.7647e-012	7.5106e-009	50156.4167
[0, 1]	512	20	1	4.4336e-010	1.7352e-012	1.8339e-009	90311.0968
{-1, 1}	32	256	1	1.8788e-010	2.4629e-011	4.1487e-010	886.6155
{-1, 1}	64	128	1	5.7054e-008	1.9103e-01	1.9141e-007	9625.1122
{-1, 1}	128	64	1	4.9321e-011	2.3679e-013	1.6142e-010	728.0978
{-1, 1}	256	32	1	1.5697e-010	1.2565e-012	5.7824e-010	1589.0696
{-1, 1}	512	20	1	9.314e-010	3.0138e-012	3.5071e-009	2583.1032
{0, 1}	32	256	1	3.3655e-012	9.2803e-014	9.3084e-012	931.8963

{0, 1}	64	128	1	8.081e-010	1.0985e-011	2.016e-009	4592.9736
{0, 1}	128	64	1	7.4555e-012	9.5752e-014	2.6816e-011	5136.0302
{0, 1}	256	32	1	4.5056e-011	1.6242e-013	1.6265e-010	13210.471
{0, 1}	512	20	1	9.2809e-011	1.9455e-013	3.5224e-010	38634.6714

Πίνακας 5: Πειραματικά δεδομένα για πίνακες με γνωστή λύση με την μέθοδο απαλοιφής του Jordan

Κλάση δοκιμα-στικού πίνακα	Διάσταση πίνακα (N)	Πλήθος επαναλή-ψεων	Βάθος αναδρομής	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο σφάλμα |RBT - Exact|	Μέσο ελάχιστο σφάλμα |RBT - Exact|	Μέσο μέγιστο σφάλμα |RBT - Exact|	Μέσο condition number
|i-j|	32	256	1	2.258570097574961e-016	1.0395330315757129e-017	6.7460812512560793e-016	1.4272985396774435e-016	7.3042686869230903e-018	3.8785743096972164e-016	708.9846
|i-j|	64	128	1	6.2006209587647089e-016	8.2224128764728148e-018	1.8707328082095628e-015	1.5626646607632655e-016	4.2380716766383572e-018	4.8462003241497102e-016	2843.4584
|i-j|	128	64	1	3.7571812896083152e-016	3.5200286128449837e-018	1.5211521453551175e-015	1.7285648993352703e-016	1.6734217170309662e-018	5.9905347040043241e-016	11381.3634
|i-j|	256	32	1	3.6192731644931072e-016	1.6906004868930998e-018	1.5104369912818993e-015	2.0962660622385023e-016	1.1765014559436233e-018	7.9857526981216139e-016	45532.9858
|i-j|	512	20	1	2.3101000810008186e-015	5.2370321496442033e-018	1.1609336112468436e-014	2.1488137068290411e-015	6.2479653505630412e-018	7.8009316937189869e-015	182139.4756

max(i,j)	32	256	1	1.9767099236019152e-016	1.1793198178207026e-017	5.0391084338820224e-016	0.032226562500000083	1.3574497888565221e-017	1	2900.0064
max(i,j)	64	128	1	2.0987712447857977e-016	5.2866443918570136e-018	6.1640600804372191e-016	0.015869140625000208	5.4633454797581198e-018	1	11499.7772
max(i,j)	128	64	1	2.2771545076985953e-016	3.0570826547284585e-018	7.4405293404613158e-016	0.007873535156250203	3.2281043843852590e-018	1	45775.0249
max(i,j)	256	32	1	3.7306437376817520e-016	2.8732724802968079e-018	1.3900793959106926e-015	0.003921508789062869	2.8732724802968079e-018	1	182628.7113
max(i,j)	512	20	1	9.9311206072337313e-016	7.6160170886609364e-018	3.7706950954706246e-015	0.001956939697266609	7.6160170886609364e-018	0.9999999999999978	729549.0367
C(i+j,j)	32	256	1	0.27649798411973742	3.0064441406163542e-005	1.4297402422648222	0.27649798411973742	3.0064441406163542e-005	1.4297402422648222	1.986698e+025
C(i+j,j)	64	128	1	0.015625000000955409	3.5164597043613149e-018	0.999999999998189	0.015625000000955409	3.5164597043613149e-018	0.999999999998189	1.937767e+035
C(i+j,j)	128	64	1	0.0078125	1.0157905336861474e-051	1	0.0078125	1.0157905336861474e-051	1	2.438210e+045
C(i+j,j)	256	32	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	2.060447e+060
C(i+j,j)	512	20	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
Hadamard	32	256	1	6.8428569944742631e-017	1.0659914485809831e-018	3.1868036662364384e-016	6.8428569944742631e-017	1.0659914485809831e-018	3.1868036662364384e-016	1

Hadamard	64	128	1	6.1741763696711567e-017	5.1917998691841556e-019	4.3800494394380633e-016	6.1741763696711567e-017	5.1917998691841556e-019	4.3800494394380633e-016	1
Hadamard	128	64	1	5.4655028803026649e-017	2.5141474133331924e-019	5.5628756567446568e-016	5.4655028803026649e-017	2.5141474133331924e-019	5.5628756567446568e-016	1
Hadamard	256	32	1	5.2089268370470472e-017	9.5600184862691949e-020	7.7528706577597216e-016	5.2089268370470472e-017	9.5600184862691949e-020	7.7528706577597216e-016	1
Hadamard	512	20	1	5.0084563064678601e-017	6.0560624349311581e-020	1.3225531780847177e-015	5.0084563064678601e-017	6.0560624349311581e-020	1.3225531780847177e-015	1
Permute	32	256	1	0.054617538206400015	1.0404437728039895e-014	0.23582035914100463	0.054617538206400015	1.0404437728039895e-014	0.23582035914100463	1
Permute	64	128	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Permute	128	64	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Permute	256	32	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Permute	512	20	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	1
Turing	32	256	1	19.690458572559464	1.561966200824827e-007	315.04733697418123	19.690458572559464	1.561966200824827e-007	315.04733697418123	2.78774e+010
Turing	64	128	1	288477035408624960	1.867450237274169	9231265133075998700	288477035408624960	1.867450237274169	9231265133075998700	1.11781e+019
Turing	128	64	1	2.65845e+037	1.4891171495310118	1.70141183e+038	2.65845e+037	1.4891171495310118	1.70141183e+038	9.14668e+017
Turing	256	32	1	4.5231284e+056	1.9618510658665682	5.789604e+078	4.5231284e+056	1.9618510658665682	5.789604e+078	9.86347e+017
Turing	512	20	1	2.6187124e+111	1.4540780037357284	6.7039039e+140	2.6187124e+111	1.4540780037357284	6.7039039e+140	1.23420e+019

4.4 Πειραματικά δεδομένα σε πίνακες μεγάλης τάξης

Παρακάτω, παρουσιάζονται ορισμένες ακόμα κλάσεις πινάκων οι οποίες μπορούν να δημιουργηθούν ντετερμινιστικά, και για ορισμένες μπορούμε να γνωρίζουμε εκ των προτέρων τον αντίστροφο τους, και τον αριθμό συνθήκης. Οι δοκιμαστικοί πίνακες που αντλήθηκαν από την βιβλιογραφία και παρουσιάστηκαν στην προηγούμενη ενότητα στην πλειοψηφία τους περιέχουν σχετικά μικρά στοιχεία όπως -1, 0, 1 κλπ. Σκοπός αυτών των πειραμάτων είναι να εξακριβώσουμε την απόδοση και την ταχύτητα της μεθόδου σε πιο γενικούς πίνακες μεγαλύτερης διάστασης.

Θα θεωρήσουμε τις ακόλουθες κλάσεις δοκιμαστικών πινάκων:

Given's Matrix:

Ο πίνακας A έχει την μορφή: $A = (a_{ij}) = 2min(i,j) - 1$

Ο αντίστροφος του A έχει την μορφή: $A^{-1} = \frac{1}{2}\begin{pmatrix} 3 & -1 & 0 & & & \\ -1 & 2 & -1 & & & \\ 0 & -1 & 2 & -1 & & \\ & & & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & 0 & -1 & 1 \end{pmatrix}$

Οι πίνακες A και A^{-1} είναι συμμετρικοί και θετικά ορισμένοι. Ο A^{-1} είναι τρι-διαγώνιος. Ο αριθμός συνθήκης του A είναι: $\kappa_2(A) = \frac{16n^2}{\pi^2}$

Ειδική περίπτωση Pei Matrix:

Ο πίνακας A έχει την μορφή: $A = (a_{ij}) \quad a_{ij} = \begin{cases} n, & i = j \\ 1, & i \neq j \end{cases}$

Ο αντίστροφος του A έχει την μορφή: $A^{-1} = (a_{ij}) \quad a_{ij} = \begin{cases} \frac{2}{2n-1}, & i = j \\ \frac{-1}{(n-1)(2n-1)}, & i \neq j \end{cases}$

Όπου n η τάξη του πίνακα A. Ο πίνακας είναι συμμετρικός και θετικά ορισμένος για $n > 1$.

Συμμετρικός με μικρό αριθμό συνθήκης:

Ο πίνακας A έχει την μορφή: $A = (a_{ij}) \quad a_{ij} = n - |i - j|$

Ο αντίστροφος του A έχει την μορφή: $A^{-1} = (a_{ij}) \quad a_{ij} = \begin{cases} \frac{n+2}{2n+2}, & i = j = 1 \text{ ή } n \\ 1, & i = j \neq 1 \text{ ή } n \\ -\frac{1}{2}, & |i-j| = 1 \text{ και } n \neq 2 \\ -\frac{1}{3}, & |i-j| = 1 \text{ και } n = 2 \\ \frac{1}{2n+2}, & |i-j| = n-1 \neq 1 \\ 0, & 1 < |i-j| < n-1 \end{cases}$

Ο A είναι συμμετρικός, θετικά ορισμένος και παρουσιάζει μικρό αριθμό συνθήκης.

Hilbert Matrix:

Ο πίνακας Α έχει την μορφή: $A = (a_{ij}) \quad a_{ij} = \left(\frac{1}{1+j-1}\right) \; i,j = 1,\dots,n$

Ο αντίστροφος του Α έχει την μορφή: $A^{-1} = (a_{ij}) \quad a_{ij} = \frac{(-1)^{i+j}(n+i-1)!(n+j-1)!}{(i+j-1)[(i-1)!(j-1)!]^2(n-i)!(n-j)!}$

Ο αριθμός συνθήκης του πίνακα Α αυξάνει εκθετικά με την διάσταση του. Ο πίνακας Α είναι συμμετρικός.

Για αυτές τις κλάσεις δοκιμαστικών πινάκων θα διεξαχθούν αντίστοιχα αριθμητικά πειράματα με πίνακες διάστασης $1024 \times 1024, 2048 \times 2048$ και 4096×4096. Επειδή οι πίνακες θα είναι μεγάλης διάστασης θα εφαρμόσουμε τον μετασχηματισμό RBT με βάθος αναδρομής 1 (αφού σύμφωνα με τον [20] δεν αναμένεται σημαντική βελτίωση της ακρίβειας) προκειμένου να επιταχυνθεί η επίλυση των συστημάτων. Όπως και στις προηγούμενες δοκιμές, θα καταγράφεται ο μέσος αριθμός συνθήκης των πινάκων, καθώς και το μέσο, ελάχιστο και μέγιστο σφάλμα.

Στα αριθμητικά αποτελέσματα που παρουσιάζονται παρακάτω η λύση του συστήματος θα είναι γνωστή και θα είναι της μορφής $x = (1,\dots,n)$. Δοθέντων των πινάκων Α και x, ο πίνακας στήλη θα υπολογίζεται ως το γινόμενο $b = Ax$.

4.5 Πειραματικά αποτελέσματα για πίνακες μεγάλης διάστασης με την μέθοδο απαλοιφής του Gauss

Στον πίνακα που ακολουθεί παρουσιάζονται τα αριθμητικά αποτελέσματα της μεθόδου RBT στις δοκιμαστικές κλάσεις μεγάλης διάστασης που αναφέρθηκαν. Το ενδιάμεσο γραμμικό σύστημα επιλύεται με την μέθοδο απαλοιφής του Gauss.

Πίνακας 6: Αριθμητικά αποτελέσματα του RBT για πίνακες μεγάλης διάστασης με την μέθοδο απαλοιφής του Gauss

Κλάση δοκιμα-στικού πίνακα	Διά-σταση πίνακα (N)	Πλήθος επαναλή-ψεων	Βάθος αναδρ ομής	Μέσος χρόνος RBT	Μέσος χρόνος MatLab Solver	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο σφάλμα |RBT - Exact|	Μέσο ελάχιστο σφάλμα |RBT - Exact|	Μέσο μέγιστο σφάλμα |RBT - Exact|	Μέσο condition number
Given's Matrix	1024	8	1	26.4006	0.048281	1.5017484358118997e-006	1.374091063155447e-009	6.87318850900453 53e-006	1.5017484358118997e-006	1.374091063155447e-009	6.8731885090045353e-006	1699886.7167
Given's Matrix	2048	2	1	236.5061	0.3204	1.6968026939998535e-005	2.6756879378808662e-008	7.01576162924766 32e-005	1.6968026939998535e-005	2.6756879378808662e-008	7.0157616292476632e-005	6799548.8667
Given's Matrix	4096	1	1	2234.5003	2.0213	2.0608460839666208e-005	2.1957475837552 92e-009	9.88989381767169 11e-005	2.0608460839666208e-005	2.1957475837552 92e-009	9.8898938176716911e-005	27198197.4676
Pei Matrix	1024	8	1	26.321	0.044856	1.2351375889944893e-012	0	5.16209297529712 79e-012	5.576075625540744e-013	0	2.9274360713316128e-012	2.001
Pei Matrix	2048	2	1	236.2686	0.26216	2.4696008042376927e-012	0	1.35003119794419 04e-011	1.5052584829630677e-012	0	8.5265128291212022e-012	2.0005
Pei Matrix	4096	1	1	2233.0203	2.1581	7.499399755708791e-012	0	3.18323145620524 88e-011	4.3658168443275192e-012	0	2.3874235921539366e-011	2.0002
Well-Conditioned Symmetric	1024	8	1	26.3017	0.052753	2.4928310856253027e-006	2.2334631921694381e-009	1.09237464371858 55e-005	2.4448583414192276e-006	1.3259899844797474e-009	1.0461348009194182e-005	1416658.9521

Well-Conditioned Symmetric	2048	2	1	235.6985	0.31364	2.8297400930762918e-005	2.3321959474742471e-008	0.00014582537801288709	2.7986625314322159e-005	2.3868551579653285e-010	0.0001409042034765661 8	5666644.079
Well-Conditioned Symmetric	4096	1	1	2232.0167	2.0095	6.0885305445498806e-005	5.3323674364946783e-009	0.00030343497434870415	5.4156972050109758e-005	2.8065414880984463e-008	0.0003038091926583774 7	22666584.5866
Hilbert Matrix	1024	8	1	26.324	0.044443	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
Hilbert Matrix	2048	2	1	235.9251	0.24739	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
Hilbert Matrix	4096	1	1	2233.2896	1.8765	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞

4.6 Πειραματικά αποτελέσματα σε πίνακες μεγάλης τάξης με την μέθοδο απαλοιφής του Jordan

Το τελευταίο αριθμητικό πείραμα που παρουσιάζεται στην εργασία αφορά τους δοκιμαστικούς πίνακες μεγάλης διάστασης που παρουσιάστηκαν στο κεφάλαιο 4.4. Σε αυτό το πείραμα το ενδιάμεσο γραμμικό σύστημα επιλύεται με την μέθοδο απαλοιφής του Jordan

Πίνακας 7: Αριθμητικά αποτελέσματα του RBT για πίνακες μεγάλης διάστασης με την μέθοδο απαλοιφής του Jordan

Κλάση δοκιμα-στικού πίνακα	Διάσταση πίνακα (N)	Πλήθος επαναλή-ψεων	Βάθος αναδρομής	Μέσο σφάλμα |RBT - MatLab|	Μέσο ελάχιστο σφάλμα |RBT - MatLab|	Μέσο μέγιστο σφάλμα |RBT - MatLab|	Μέσο σφάλμα |RBT - Exact|	Μέσο ελάχιστο σφάλμα |RBT - Exact|	Μέσο μέγιστο σφάλμα |RBT - Exact|	Μέσο condition number
Given's Matrix	1024	8	1	3.02456678041893e-005	1.7008218833325373e-008	0.00012689725777503824	3.02456678041893e-005	1.7008218833325373e-008	0.00012689725777503824	1699886.7167
Given's Matrix	2048	2	1	6.0518752043289478e-005	1.1907474117833772e-008	0.00026026247979871187	6.0518752043289478e-005	1.1907474117833772e-008	0.00026026247979871187	6799548.8667
Given's Matrix	4096	1	1	0.00010506686296644376	1.2352757039479911e-008	0.00059705268495235941	0.00010506686296644376	1.2352757039479911e-008	0.00059705268495235941	27198197.4676
Pei Matrix	1024	8	1	1.3925120144432299e-012	0	5.9969806898152456e-012	8.5859145043231527e-013	0	4.1424641494813841e-012	2.001
Pei Matrix	2048	2	1	3.1338934268834717e-012	0	1.3528733688872308e-011	2.3931396144274975e-012	0	1.2732925824820995e-011	2.0005
Pei Matrix	4096	1	1	9.1033130469528323e-012	0	4.5019987737759948e-011	6.7069847939360055e-012	0	3.5242919693700969e-011	2.0002
Well-Conditioned Symmetric	1024	8	1	2.0509919701335125e-006	2.7830111548610148e-009	9.7881131893018392e-006	2.0024682252935377e-006	2.4020856415063463e-009	9.5111743192433096e-006	1416658.9521

Well-Conditioned Symmetric	2048	2	1	6.6668325071440665e-006	2.47647591322675e-009	3.4679176565077796e-005	6.1828818442873469e-006	1.8885089048126247e-009	2.8805271767851082e-005	5666644.079
Well-Conditioned Symmetric	4096	1	1	0.015337520633478267	8.9473442130838521e-006	0.071237000118745546 2	0.01533796978368062	5.9893955040024593e-006	0.071229719340408337	22666584.5866
Hilbert Matrix	1024	8	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
Hilbert Matrix	2048	2	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞
Hilbert Matrix	4096	1	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	∞

4.7 Αριθμητική ακρίβεια της μεθόδου RBT με την μέθοδο απαλοιφής του Gauss

Από τα πειράματα που διενεργήθηκαν με την μέθοδο RBT σε συνδυασμό με την μέθοδο απαλοιφής του Gauss για την επίλυση του ενδιάμεσου γραμμικού συστήματος, βλέπουμε ότι στις περισσότερες κλάσεις πινάκων η ακρίβεια που επετεύχθη είναι ικανοποιητική για τις περισσότερες πραγματικές εφαρμογές. Ενδεικτικά, για τις κλάσεις των τυχαίων πινάκων η μέση τιμή σφάλματος βρέθηκε πολύ κοντά στις τιμές που παρήγαγε η ρουτίνα του MatLab. Βέβαια, γνωρίζουμε ότι οι ομοιόμορφα κατανεμημένοι τυχαίοι πίνακες εμφανίζουν καλή αριθμητική ευστάθεια [34].

Για τις κλάσεις που επιλέχθηκαν να εμφανίζουν μεγάλο αριθμό συνθήκης C(i+j,j), Turing, ο αλγόριθμος ανταποκρίθηκε άσχημα ή απέτυχε όπως ήταν αναμενόμενο, και από την σχετική βιβλιογραφία. Αυτό επιβεβαιώνεται από τις παρατηρήσεις του Parker στα αντίστοιχα αριθμητικά πειράματα που διεξήγαγε [17] [18]. Σύμφωνα με τον Parker, το σφάλμα στρογγύλευσης αυξάνεται ως προς την διάσταση του προβλήματος. Προκύπτει, ότι για προβλήματα με μεγάλο αριθμό συνθήκης, τα σφάλματα στρογγύλευσης αυξάνονται περισσότερο από το matlab solver, χωρίς ωστόσο να γίνονται απαγορευτικά.

Επίσης, τα αποτελέσματα των Baboulin, Dorgarra και άλλοι [11] φαίνεται να επιβεβαιώνονται κι εδώ, αφού δεν προκύπτει άμεση συσχέτιση του αριθμού των αναδρομικών βημάτων που χρησιμοποιεί ο μετασχηματισμός RBT με την επιτυγχανόμενη αριθμητική ακρίβεια.

Τέλος, δοκιμάσαμε την μέθοδο με την βοήθεια 4 ακόμα κλάσεων δοκιμαστικών πινάκων σε μεγαλύτερες διαστάσεις. Για την κλάση Hilbert Matrix, ήταν γνωστό εκ των προτέρων ότι ο αριθμός συνθήκης των παραγόμενων πινάκων αυξάνει εκθετικά ως προς την διάσταση τους. Ήταν αναμενόμενο ότι τόσο ο MatLab solver, όσο και η μέθοδος RBT δεν κατάφεραν να παράγουν λύση. Στις υπόλοιπες κλάσεις πινάκων, η μέθοδος RBT παρήγαγε ακριβή αποτελέσματα συγκρινόμενα με την γνωστή ακριβής λύση. Σε σύγκριση με την λύση του MatLab solver, το σφάλμα στις περισσότερες περιπτώσεις ήταν της ίδιας τάξης ακρίβειας.

Συμπερασματικά, η μέθοδος φαίνεται να ανταποκρίνεται αρκετά καλά στην γενική περίπτωση και σε σχετικά μεσαίας διάστασης πίνακες και αποτελεί μια ενδιαφέρουσα εναλλακτική για την επιτάχυνση της μεθόδου απαλοιφής του Gauss σε υψηλών επιδόσεων παράλληλους, διανυσματικούς ή υβριδικούς (συνδυασμός CPU και GPU) υπολογιστές.

Ωστόσο, παρατηρήσαμε ότι σε πίνακες μεγαλύτερης διάστασης (τουλάχιστον 1024×1024) ο συνολικός υπολογιστικός χρόνος της μεθόδου RBT είναι ιδιαίτερα μεγαλύτερος από τον χρόνο του MatLab solver. Η διαφορά αυτή οφείλεται κατά κύριο λόγο στον βελτιστοποιημένο κώδικα του MatLab solver ο οποίος μπορεί να αξιοποιήσει έναν σύγχρονο πολυπύρηνο επεξεργαστή. Ο σειριακός κώδικας της μεθόδου RBT που υλοποιήσαμε δεν αξιοποιεί τις δυνατότητες του επεξεργαστή. Αυτό που αξίζει να σημειώσουμε, είναι ότι ο χρόνος για τον υπολογισμό των πινάκων πεταλούδας, ακόμα και στην μέγιστη διάσταση ήταν αμελητέος σε σχέση με τον χρόνο επίλυσης του γραμμικού συστήματος με την μέθοδο απαλοιφής του Gauss που υλοποιήσαμε. Η μέθοδος απαλοιφής του Gauss υλοποιήθηκε σύμφωνα με το υπόδειγμα ψευδοκώδικα που παρουσιάσαμε στο $1^{ο}$ κεφάλαιο της εργασίας. Μια τέτοια υλοποίηση δεν είναι

βελτιστοποιημένη και σε καμία περίπτωση δεν ενδείκνυται για πραγματικές υλοποιήσεις της μεθόδου.

4.8 Αριθμητική ακρίβεια της μεθόδου RBT με την μέθοδο απαλοιφής του Jordan

Παράλληλα με τα αριθμητικά πειράματα που διεξήχθησαν στην μέθοδο RBT με την μέθοδο απαλοιφής του Gauss, τα αντίστοιχα πειράματα επαναλήφθηκαν και για μια διαφοροποίηση της, κατά την οποία το ενδιάμεσο γραμμικό σύστημα επιλύθηκε με την μέθοδο απαλοιφής του Jordan.

Στις δοκιμαστικές κλάσεις πινάκων για τις οποίες δεν γνωρίζουμε την ακριβή τους λύση (normal, [0,1] κλπ) παρατηρήθηκε ότι κατά μέσο όρο η RBT με την μέθοδο απαλοιφής του Jordan έδωσε μια με δυο τάξεις χαμηλότερη αριθμητική ακρίβεια.

Στις κλάσεις |i-j| και max(i,j) η αριθμητική ακρίβεια μεταξύ των δυο υλοποιήσεων ήταν στο ίδιο επίπεδο. Για τις κλάσεις C(i+j,j) και Turing για τις οποίες γνωρίζουμε ότι παράγουν ill-conditioned πίνακες η μέθοδος RBT με την μέθοδο απαλοιφής του Jordan παρήγαγε χειρότερα αποτελέσματα (και στις δυο υλοποιήσεις δεν επετεύχθη επαρκής ακρίβεια), ενώ απέτυχε να παράγει λύση σε περισσότερες περιπτώσεις από την μέθοδο RBT με την μέθοδο απαλοιφής του Gauss. Οι κλάσεις Hadamard και Permute παράγουν πίνακες με μικρό αριθμό συνθήκης ωστόσο, η μέθοδος RBT με την μέθοδο απαλοιφής του Jordan κατάφερε να παράγει λύσεις ίδιας τάξης ακρίβειας μόνο για την κλάση Hadamard. Στην περίπτωση της κλάσης Permute για πίνακες διάστασης μεγαλύτερης από 32×32 δεν κατάφερε να παράγει λύση.

Τέλος, για τις δοκιμαστικές κλάσεις πινάκων μεγάλης διάστασης παρατηρήθηκε επίσης μια μικρή μείωση της αριθμητικής ακρίβειας. Πιο συγκεκριμένα, στις κλάσεις Given's Matrix και Well-Conditioned Symmetric παρατηρήθηκε μια πτώση της αριθμητικής ακρίβειας κατά μια με δυο τάξεις ακρίβειας. Στην περίπτωση της κλάσης Pei Matrix η αριθμητική ακρίβεια κυμάνθηκε στα ίδια επίπεδα με την μέθοδο απαλοιφής του Gauss. Για την κλάση Hilbert Matrix για την οποία γνωρίζουμε ότι παρουσιάζει εκθετικά μεγάλο αριθμό συνθήκης συναρτήσει της διάστασης, η μέθοδος RBT με την μέθοδο απαλοιφής του Jordan δεν κατάφερε να παράγει λύση, κάτι το οποίο συνέβη και στην περίπτωση της μεθόδου απαλοιφής του Gauss.

Από τα πειραματικά δεδομένα που προέκυψαν και από τις δυο υλοποιήσεις, συμπεραίνεται ότι η υλοποίηση της μεθόδου RBT με την μέθοδο απαλοιφής του Gauss υπερέχει σχεδόν σε όλες τις δοκιμές έναντι της υλοποίησης με την μέθοδο απαλοιφής του Jordan.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία, παρουσιάσαμε μια μέθοδο για την αποφυγή της οδήγησης στην μέθοδο απαλοιφής του Gauss μέσα από την αντικατάσταση της από έναν μετασχηματισμό. Ο μετασχηματισμός αυτός περιγράφηκε αναλυτικά τόσο σε θεωρητικό επίπεδο, όσο και μέσα από πρακτικά παραδείγματα. Δείξαμε με ποιον τρόπο ο τυχαίος μετασχηματισμός πεταλούδας επιδρά στον πίνακα A ενός γραμμικού συστήματος και τον μετασχηματίζει σε Gauss απαλείψιμη μορφή. Στο πλαίσιο της θεωρητικής περιγραφής της μεθόδου παρουσιάσαμε την συμπεριφορά της μεθόδου στον αριθμό συνθήκης των πινάκων και στο αναμενόμενο αριθμητικό σφάλμα.

Δόθηκε έμφαση σε θέματα που αφορούν την αποδοτική του υλοποίηση του, όπως η αποδοτική αποθήκευση των τυχαίων πινάκων πεταλούδας και οι βασικές αρχές για την υλοποίηση της μεθόδου σε υβριδικές αρχιτεκτονικές με χρήση καρτών γραφικών. Τέλος, διεξήγαμε μια σειρά από δοκιμές στηριζόμενοι σε δοκιμαστικές κλάσεις πινάκων που προτείνονται στην βιβλιογραφία.

Από τα αριθμητικά δεδομένα που προέκυψαν από τις δοκιμές αυτές, κάναμε μια πρώτη επιβεβαίωση της απόδοσης της μεθόδου, η οποία είναι σύμφωνη με παρόμοια πειραματικά δεδομένα που έχουν δημοσιευθεί στην βιβλιογραφία. Παρατηρήθηκε, ότι στην περίπτωση πινάκων που επιλέχθηκαν κατάλληλα έτσι ώστε να έχουν ιδιαίτερα μεγάλο αριθμό συνθήκης, η μέθοδος απέτυχε να παράγει ακριβή αποτελέσματα, κάτι όμως που συνέβη και στον MatLab solver. Η αριθμητική ακρίβεια που επετεύχθη στις περισσότερες δοκιμές, μας δείχνει ότι πρόκειται για μια υποσχόμενη εναλλακτική της οδήγησης, η οποία μπορεί να εφαρμοστεί και σε πραγματικά προβλήματα.

Από την ανάλυση του αλγορίθμου του τυχαίου μετασχηματισμού πεταλούδας αναδείξαμε την σημασία επιλογής παραμέτρων (fine tuning) της μεθόδου, κάτι το οποίο συμβάλει στο να επιταχυνθεί σημαντικά η μέθοδος.

Ο μετασχηματισμός παρουσιάζει μια απλότητα η οποία συμβάλει στην σχετικά εύκολη υλοποίηση του σε παράλληλες αρχιτεκτονικές, όπου η μείωση των μηνυμάτων επικοινωνίας και οι περιορισμένοι βρόχοι είναι επιβεβλημένοι. Η υπολογιστική πολυπλοκότητα του μετασχηματισμού είναι της τάξης του $O(n^2)$, η οποία μπορεί να είναι παρόμοια με την πολυπλοκότητα της μερικής οδήγησης, αλλά το γεγονός αυτό δεν μεταφράζεται αναγκαία και σε παρόμοιο χρόνο εκτέλεσης, αφού σε αντίθεση με την μερική οδήγηση ο τυχαίος μετασχηματισμός πεταλούδας ελαχιστοποιεί τα μηνύματα επικοινωνίας.

Οι διαφορές του προέκυψαν στους χρόνους εκτέλεσης (ιδιαίτερα στις μεγάλες διαστάσεις) μεταξύ του RBT και του MatLab solver κατά την διάρκεια των αριθμητικών πειραμάτων οφείλονται στο γεγονός ότι υλοποιήσαμε μια μη βελτιστοποιημένη σειριακή προσομοίωση της μεθόδου. Σε αντίθεση, ο MatLab solver διαθέτει βελτιστοποιημένες ρουτίνες και εκμεταλλεύεται αποδοτικά τον πολυπύρηνο επεξεργαστή του υπολογιστή δοκιμής. Αξίζει να σημειωθεί, ότι η συνεισφορά της διαδικασίας δημιουργίας των τυχαίων πινάκων πεταλούδας στον συνολικό υπολογιστικό χρόνο ήταν αμελητέα. Η κύρια χρονική επιβάρυνση επήλθε από την μη βελτιστοποιημένη επίλυση του γραμμικού συστήματος με την μέθοδο απαλοιφής του Gauss.

Επιπρόσθετα, ο μετασχηματισμός RBT εισάγει μια νέα απαίτηση κατά την υλοποίηση του. Θα πρέπει να αποθηκεύουμε εκτός από τους πίνακες του αρχικού γραμμικού συστήματος και δυο ακόμα πίνακες ίδιας διάστασης με τον πίνακα A, τους U, V οι οποίοι είναι τυχαίοι αναδρομικοί πίνακες πεταλούδας. Μπορούμε να επιτύχουμε

σημαντική επιτάχυνση της μεθόδου αν αποθηκεύσουμε τους πίνακες αυτούς σε «συμπαγή» μορφή. Ωστόσο, αυτή η απαίτηση εισάγει και την αντίστοιχη απαίτηση σε κύρια μνήμη. Έτσι για την αποθήκευση τριών πινάκων (U,A,V) διάστασης $2^{14} \times 2^{14}$ απαιτούνται τουλάχιστον 6GB μνήμης, κάτι το οποίο δυσχεραίνει την εφαρμογή της μεθόδου σε κάρτες γραφικών. Τέλος, μια ακόμα βελτιστοποίηση που μπορούμε να εφαρμόσουμε είναι η δημιουργία των τυχαίων αναδρομικών πινάκων πεταλούδας ανεξάρτητα από την φάση εκτέλεσης του αλγορίθμου (offline). Στην περίπτωση που γνωρίζουμε ότι η κλάση των γραμμικών συστημάτων που πρόκειται να επιλύσουμε είναι της ίδιας ή παρόμοιας διάστασης, τότε η δημιουργία των τυχαίων αναδρομικών πινάκων πεταλούδας εκ των προτέρων μπορεί να επιταχύνει την διαδικασία επίλυσης. Αξίζει να αναφέρουμε, ότι οι τυχαίοι αναδρομικοί πίνακες πεταλούδας U και V μπορεί να είναι οι ίδιοι ακόμα και αν εφαρμόζονται σε διαφορετικά γραμμικά συστήματα. Αυτό οφείλεται στο γεγονός ότι η δημιουργία των πινάκων U και V είναι ανεξάρτητη από τους πίνακες του γραμμικού συστήματος, αλλά εξαρτάται μόνο από την διάσταση του.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
ill-conditioned	Ασταθές (αναφέρεται σε γραμμικό σύστημα λόγω μεγάλου αριθμού συνθήκης)
Condition number	Αριθμός συνθήκης
Unitary matrix	Μοναδιακός πίνακας
Norm	Νόρμα
Random Butterfly Transformation	Τυχαίος Μετασχηματισμός Πεταλούδας
Recursive Random Butterfly	Αναδρομικός Μετασχηματισμός Πεταλούδας
Butterfly matrices	Πίνακες πεταλούδας
Direct sum	Ευθύ άθροισμα
nonsingular	Μη ιδιάζον - αντιστρέψιμος
Kronecker product	Γινόμενο του Kronecker
(matrix) trace	Ίχνος (πίνακα)
conjugate transpose (matrix)	Συζυγής ανάστροφος (πίνακας)
Hermitian adjoint (matrix)	Συζυγής (πίνακας) του Hermite
Eigenvalue	Ιδιοτιμή
Spectral norm	Φασματική νόρμα
pivoting	Οδήγηση (αναφέρεται σε προπαρασκευαστική διαδικασία επίλυσης γραμμικού συστήματος)
Determinant (det)	Ορίζουσα
Roundoff error	Σφάλμα στρογγύλευσης
Submatrix	Υποπίνακας
Linear system	Γραμμικό σύστημα
Singular value	Ιδιάζουσα τιμή
Hadamard recursive matrix	Αναδρομικός πίνακας Hadamard
Random matrix	Τυχαίος πίνακας
Complex value	Μιγαδικός αριθμός
Nondegenerate	Μη εκφυλισμένος
Positive definite	Θετικά ορισμένος
Block nonsingular	Μπλοκ μη ιδιάζον
Block principal submatrix	Κύριος μπλοκ υποπίνακας
Gauss eliminable	Gauss απαλείψιμος
float-point number	Αριθμός κινητής υποδιαστολής
Growth factor	Συντελεστής αύξησης
(matrix) rank	Τάξη ή βαθμός (πίνακα)
Tight bound	Σφικτό φράγμα
Graphics Processor Unit	Μονάδα επεξεργασίας γραφικών
Compute Unified Device Architecture	Αρχιτεκτονική CUDA
Overhead (computational)	Επιβάρυνση (υπολογιστική)
Cache memory	Κρυφή μνήμη
Instruction stream	Ρεύμα εντολών
Data stream	Ρεύμα δεδομένων
Bus	Δίαυλος
Crossbar	Πλέγμα
Uniform memory access	Ομοιόμορφη προσπέλαση μνήμης
Interconnection network	Δίκτυο διασύνδεσης

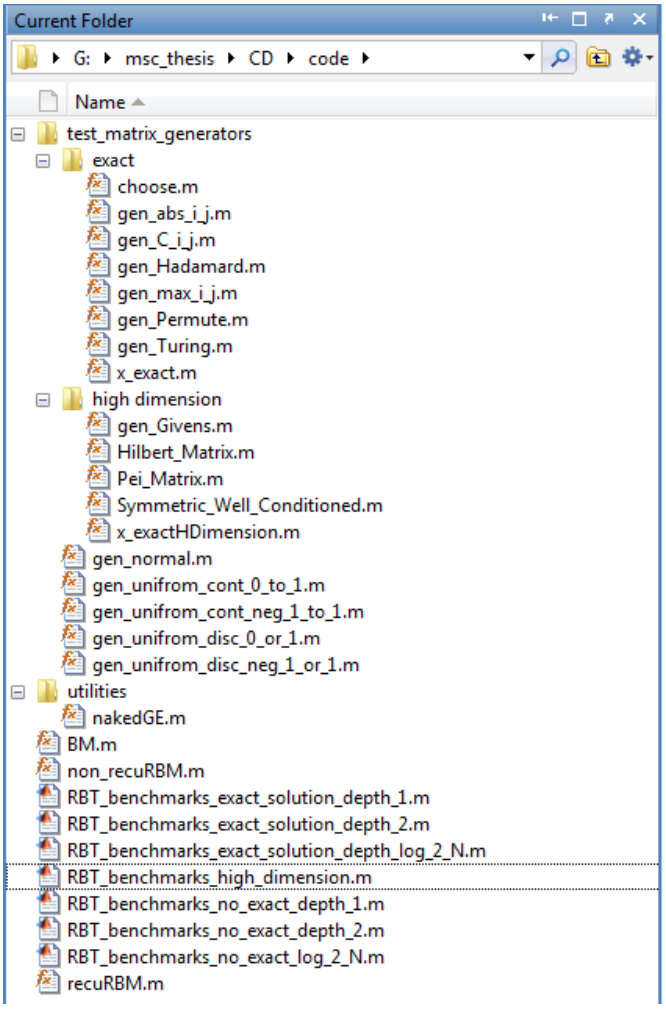
Pairwise pivoting	Οδήγηση κατά ζεύγη
Incremental pivoting	Αυξανόμενη οδήγηση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

RBT	Random Butterfly Transformation
GE	Gauss Elimination
CPU	Central Processing Unit
GPU	Graphics Processor Unit
CUDA	Compute Unified Device Architecture
SISD	Single Instruction stream, Single Data stream
SIMD	Single Instruction stream, Multiple Data streams
MISD	Multiple Instruction streams, Single Data stream
MIMD	Multiple Instruction streams, Multiple Data streams

ΠΑΡΑΡΤΗΜΑ Ι: οργάνωση CD εργασίας

Στο συνοδευτικό CD της εργασίας στον φάκελο code, περιλαμβάνεται ο πηγαίος κώδικας της μεθόδου RBT υλοποιημένος για το περιβάλλον του MatLab. Στην παρακάτω εικόνα διακρίνεται η διάρθρωση του φακέλου code:



Εικόνα 2: Ο πηγαίος κώδικας της μεθόδου για το περιβάλλον MatLab

Ο φάκελος test_matrix_generators περιέχει 15 κλάσεις δοκιμαστικών πινάκων οι οποίοι χρησιμοποιήθηκαν προκειμένου να προκύψουν τα αριθμητικά δεδομένα της μεθόδου. Οι 6 από αυτές τις κλάσεις περιέχονται στον υποφάκελο exact και αφορά κλάσεις δοκιμαστικών πινάκων για τις οποίες γνωρίζουμε εκ των προτέρων την ακριβή τους αριθμητική λύση.

Ο φάκελος utilities περιέχει την συνάρτηση nakedGE() η οποία επιλύει ένα γραμμικό σύστημα με την μέθοδο απαλοιφής του Gauss χωρίς την χρήση οδήγησης.

Τέλος, διακρίνουμε 7 αρχεία με το πρόθεμα RBT_benchmarks τα οποία περιέχουν δοκιμαστικά σενάρια με 15 κλάσεις δοκιμαστικών πινάκων.

ΠΑΡΑΡΤΗΜΑ ΙΙ: Τεκμηρίωση κώδικα της μεθόδου RBT

Στο πλαίσιο της εργασίας υλοποιήθηκε η μέθοδος RBT καθώς και μια σειρά από δοκιμαστικά σενάρια για την εξακρίβωση της αριθμητικής της ακρίβειας στο περιβάλλον του MatLab.

Ο φάκελος με τον κώδικα περιλαμβάνει το φάκελο test_matrix_generators στον οποίο βρίσκονται 15 συναρτήσεις παραγωγής δοκιμαστικών πινάκων ταξινομημένες σε 3 κατηγορίες: πίνακες με μη γνωστή ακριβή αριθμητική λύση, πίνακες με γνωστή ακριβή αριθμητική λύση και πίνακες που χρησιμοποιούνται σε δοκιμές υψηλής διάστασης. Οι συναρτήσεις παραγωγής των πινάκων αυτών είναι:

Πίνακες με μη γνωστή ακριβή αριθμητική λύση:

```
function [ A ] = gen_normal( n,m )
```
Είσοδος: οι διαστάσεις του πίνακα εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία που ακολουθούν την κατανομή $\mathrm{N}(0,1)$

```
function [ A ] = gen_unifrom_disc_neg_1_or_1( n,m )
```
Είσοδος: οι διαστάσεις του πίνακα εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία -1 ή 1 με ίση πιθανότητα $p(-1) = p(1) = \frac{1}{2}$

```
function [ A ] = gen_unifrom_disc_0_or_1( n,m )
```
Είσοδος: οι διαστάσεις του πίνακα εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία 0 ή 1 με ίση πιθανότητα $p(0) = p(1) = \frac{1}{2}$

```
function [ A ] = gen_unifrom_cont_neg_1_to_1( n,m )
```
Είσοδος: οι διαστάσεις του πίνακα εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία στο διάστημα $[-1,1]$ ακολουθώντας ομοιόμορφη κατανομή

```
function [ A ] = gen_unifrom_cont_0_to_1( n,m )
```
Είσοδος: οι διαστάσεις του πίνακα εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία στο διάστημα $[0,1]$ ακολουθώντας ομοιόμορφη κατανομή

Πίνακες με μη γνωστή ακριβή αριθμητική λύση: (για τον τρόπο δημιουργίας των πινάκων ανατρέξτε στο 4ο κεφάλαιο)

```
function [ A,b ] = gen_Turing( n,m )
```
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο Turing πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

```
function [ A,b ] = gen_Permute( n, m )
```
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας A ο οποίος προκύπτει σαν τυχαία αντιμετάθεση του I_n διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

```
function [ A,b ] = gen_max_i_j( n,m )
```
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία της μορφής $\max(i,j)$ και ο πίνακας στήλη b διάστασης $n \times 1$

```
function [ A,b ] = gen_Hadamard( n,m )
```
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο Hadamard πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

```
function [ A,b ] = gen_C_i_j( n,m )
```

Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία της μορφής $\binom{i+j-2}{j-1}$ και ο πίνακας στήλη b διάστασης $n \times 1$

`function [ A,b ] = gen_abs_i_j( n,m )`
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ με στοιχεία της μορφής $|i-j|$ και ο πίνακας στήλη b διάστασης $n \times 1$

Πίνακες για δοκιμές υψηλής τάξης:

`function [ A,b ] = gen_Givens( n,m )`
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας Given's A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

`function [ A,b ] = Hilbert_Matrix( n,m )`
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας του Hilbert A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

`function [ A,b ] = Pei_Matrix( n,m )`
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο πίνακας Pei A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

`function [ A,b ] = Symmetric_Well_Conditioned( n,m )`
Είσοδος: οι διαστάσεις των πινάκων εξόδου. **Έξοδος:** ο συμμετρικός και με μικρό αριθμό συνθήκης πίνακας A διάστασης $\mathrm{n} \times \mathrm{m}$ και ο πίνακας στήλη b διάστασης $n \times 1$

Η μέθοδος RBT απαιτεί την υλοποίηση κάποιας μεθόδου επίλυσης γραμμικών συστημάτων χωρίς οδήγηση για την επίλυση του ενδιάμεσου γραμμικού συστήματος. Στον φάκελο utilities του κώδικα, περιέχονται δυο συναρτήσεις για την επίλυση ενός γραμμικού συστήματος της μορφής $Ax = b$ με την μέθοδο απαλοιφής του Gauss και του Jordan χωρίς την χρήση οδήγησης.

`function [ x ] = nakedGE(A, b )`
Είσοδος: οι πίνακες A (συντελεστές αγνώστων) και b (σταθεροί όροι). **Έξοδος:** ο πίνακας x με την λύση του γραμμικού συστήματος $Ax = b$ με την μέθοδο απαλοιφής του Gauss χωρίς την χρήση οδήγησης

`function [ x ] = GaussJordan( A, b )`
Είσοδος: οι πίνακες A (συντελεστές αγνώστων) και b (σταθεροί όροι). **Έξοδος:** ο πίνακας x με την λύση του γραμμικού συστήματος $Ax = b$ με την μέθοδο απαλοιφής του Jordan χωρίς την χρήση οδήγησης

Τέλος, ακολουθούν οι συναρτήσεις για την παραγωγή των πινάκων πεταλούδας και των αναδρομικών πινάκων πεταλούδας:

`function [ B ] = BM( n, s )`
Είσοδος: Η διάσταση n του πίνακα πεταλούδας και ένας ακέραιος αριθμός s για τον έλεγχο του διαστήματος των τυχαίων αριθμών (για λόγους ευστάθειας ο αριθμός s πρέπει να επιλέγεται ίσος με 1). **Έξοδος:** ο πίνακας πεταλούδας διάστασης $n \times n$
`function [ U ] = recuRBM( n, s )`
Είσοδος: Η διάσταση n του αναδρομικού πίνακα πεταλούδας και ένας ακέραιος αριθμός s για τον έλεγχο του διαστήματος των τυχαίων αριθμών (για λόγους ευστάθειας ο αριθμός s πρέπει να επιλέγεται ίσος με 1). **Έξοδος:** ο αναδρομικός πίνακας πεταλούδας διάστασης $n \times n$ με βάθος αναδρομής $log_2 n$

```
function [ U ] = non_recuRBM( n,d,s )
```

Είσοδος: Η διάσταση n του αναδρομικού πίνακα πεταλούδας, το βάθος αναδρομής d (ακέραιος αριθμός 1 ή 2) και ένας ακέραιος αριθμός s για τον έλεγχο του διαστήματος των τυχαίων αριθμών (για λόγους ευστάθειας ο αριθμός s πρέπει να επιλέγεται ίσος με 1). **Έξοδος:** ο αναδρομικός πίνακας πεταλούδας διάστασης $n \times n$ με βάθος αναδρομής d

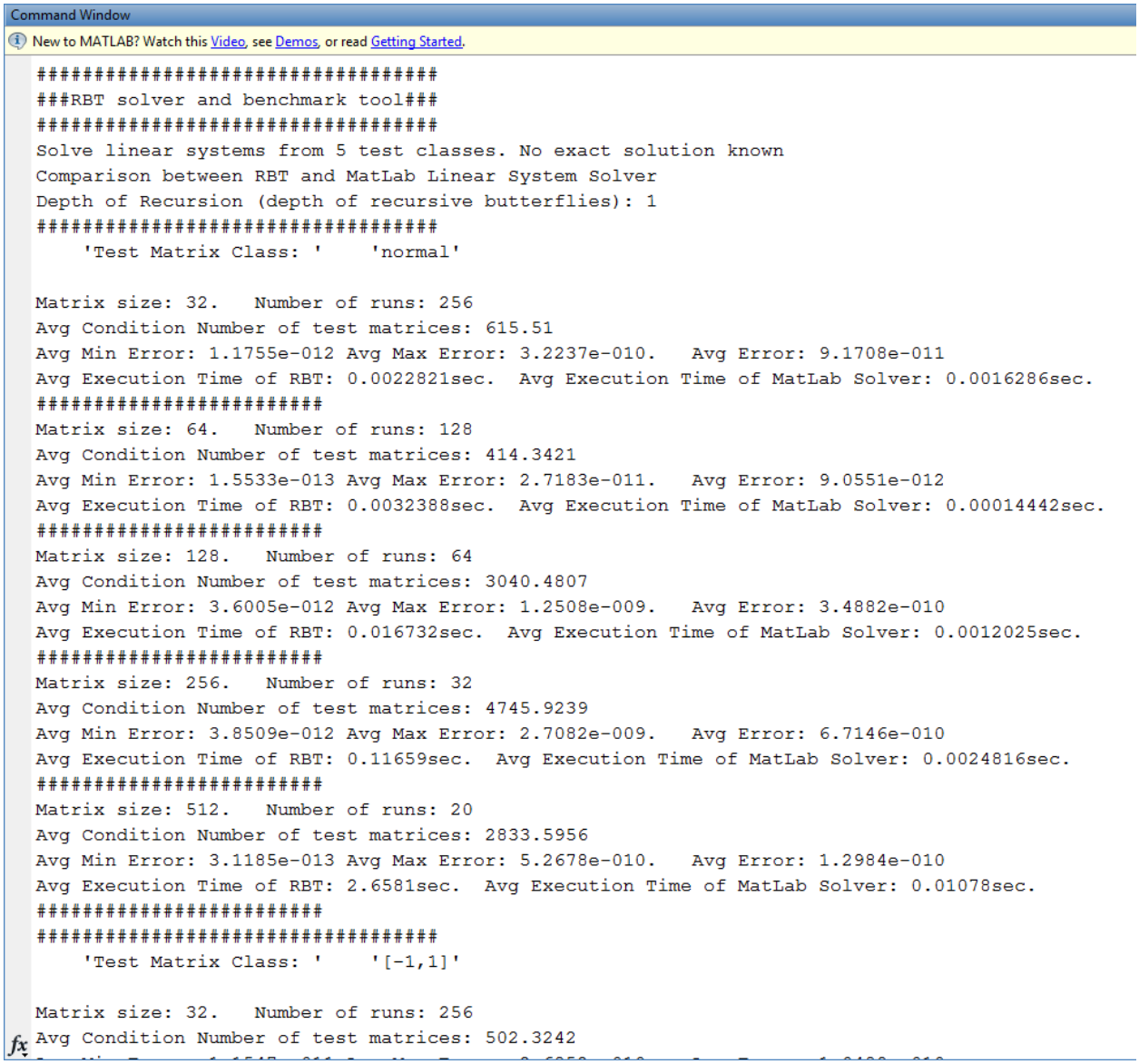
ΠΑΡΑΡΤΗΜΑ ΙΙΙ: Εκτέλεση δοκιμαστικών σεναρίων

Στον φάκελο code του CD της εργασίας υπάρχουν 10 αρχεία με το πρόθεμα RBT_benchmarks τα οποία περιέχουν σενάρια για το περιβάλλον του MatLab τα οποία είναι:

Πίνακας 8: Σενάρια εκτέλεσης της μεθόδου RBT

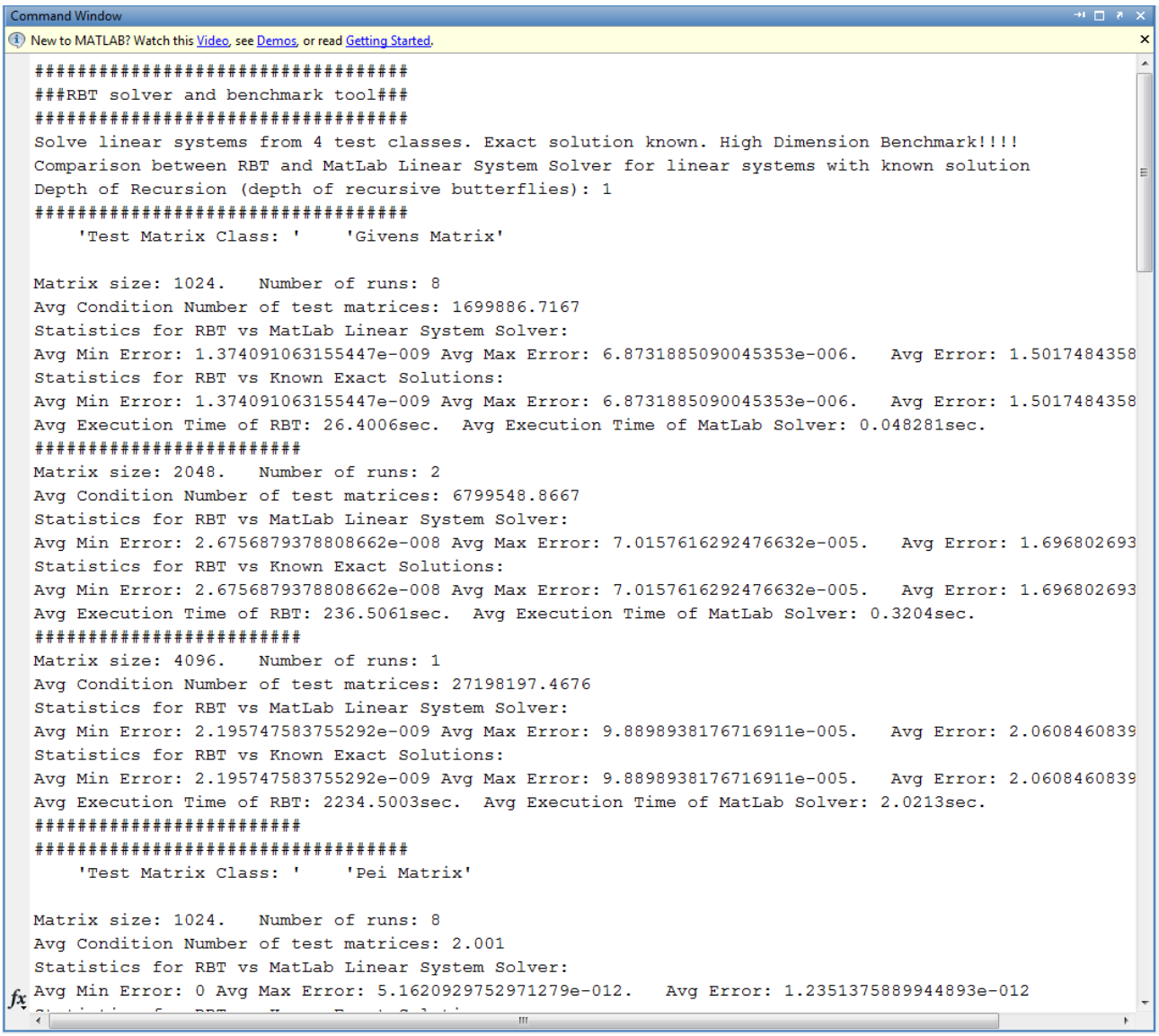
Όνομα σεναρίου	Βάθος αναδρο-μής	Ακριβής λύση γνωστή	Επίλυση ενδιάμεσου συστήματος
RBT_benchmarks_no_exact_depth_1.m	1	Όχι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_no_exact_depth_2.m	2	Όχι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_no_exact_depth_log_2_N.m	log_2N	Όχι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_exact_solution_depth_1.m	1	Ναι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_exact_solution_depth_2.m	2	Ναι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_exact_solution_depth_log_2_N.m	log_2N	Ναι	Μέθοδος απαλοιφής του Gauss
RBT_benchmarks_high_dimension.m	1	Ναι	Μέθοδος απαλοιφής του Gauss
Gauss_Jordan_RBT_benchmarks_no_exact_depth_1.m	1	Όχι	Μέθοδος απαλοιφής του Jordan
Gauss_Jordan_RBT_benchmarks_exact_solution_depth_1.m	1	Ναι	Μέθοδος απαλοιφής του Jordan
Gauss_Jordan_RBT_benchmarks_high_dimension.m	1	Ναι	Μέθοδος απαλοιφής του Jordan

Παρακάτω ακολουθεί ένα παράδειγμα εκτέλεσης για το σενάριο RBT_benchmarks_no_exact_depth_1.m:



Εικόνα 3: Εκτέλεση του σεναρίου RBT_benchmarks_no_exact_depth_1.m

Για να επιβεβαιώσουμε την απόδοση της μεθόδου RBT δημιουργήσαμε 4 ακόμα δοκιμαστικές κλάσεις πινάκων μεγάλης διάστασης. Στην Εικόνα 4 που ακολουθεί βλέπουμε ένα παράδειγμα εκτέλεσης του δοκιμαστικού σεναρίου.



Εικόνα 4: Εκτέλεση του σεναρίου RBT_benchmarks_high_dimension.m

ΑΝΑΦΟΡΕΣ

[1] Ν. Μισυρλής, Αριθμητική Ανάλυση: Μια Αλγοριθμική Προσέγγιση, Αθήνα: Αυτοέκδοση, 2009.

[2] Ν. Μισυρλής, Αριθμητική Γραμμική Άλγεβρα, Αθήνα: Σημειώσεις μαθήματος, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 2008.

[3] J. H. Wilkinson, "Error analysis of direct methods of matrix inversion," *J. ACM,* vol. 8, pp. 281-330, 1961.

[4] J. H. Wilkinson, Rounding Errors in Algebraic Processes, NJ: Prentice-Hall Inc., 1963.

[5] N. J. Higham and D. J. Higham, "Large Growth Factors in Gaussian Ellimination with Pivoting," *SIAM J. Matrix Anal. Appl.,* vol. 10, no. 3, pp. 155-164, 1989.

[6] S. J. Wright, "A Collection of Problems for which Gaussian Elimination with Partial Pivoting is Unstable," *SIAM J. Sci. Comput.,* vol. 14, no. 1, pp. 231-238, 1993.

[7] N. Higham, "Accuracy and Stability of Numerical Algorithms (2nd edition)," *SIAM,* 2002.

[8] L. V. Foster, "The growth factor and efficiency of Gaussian elimination with rook pivoting," *J. Comput. Appl. Math.,* vol. 86, pp. 177-194, 1997.

[9] G. Poole and L. Neal, "The Rook's pivoting strategy," *Journal of Computational and Applied Mathematics,* vol. 123, no. 1-2, p. 353–369, 2000.

[10] X. Chang, "Some Features of Gaussian Elimination with Rook Pivoting," *BIT Numerical Mathematics,* vol. 42, no. 1, pp. 66-83, 2002.

[11] M. Baboulin, J. Dongarra, J. Herrmann and S. Tomov, "Accelerating linear system solutions using randomization techniques," *ACM Trans. Math. Softw. ,* vol. 39, no. 2, Article 8, 2013.

[12] J. R. Bunch and J. Hopcroft, "Triangular Factorization and Inversion by Fast Matrix

Multiplication," *Mathematics of Computation,* vol. 125, no. 28, pp. 231-236, 1974.

[13] J. v. Neumann and H. H. Goldstine, "Numerical inverting of matrices of high order," *Bull. Amer. Math. Soc.,* vol. 53, pp. 1021-1099, 1947.

[14] A. W. Marshall and I. Olkin, Inequalities: Theory of Majorization and Its Applications, NY: Academic Press, 1979.

[15] M. Baboulin, S. Donfack, J. Dongarra, L. Grigori, A. Rémy and S. Tomov, "A class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines," in *Proceedings of the International Conference on Computational Science, ICCS 2012*, Procedia Computer Science, Elsevier, Vol. 9, pp. 17-26, 2012.

[16] D. Becker, M. Baboulin and J. Dongarra, "Reducing the amount of pivoting in symmetric indefinite systems," in *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics, PPAM 2011*, 2011.

[17] D. Stott Parker, «Random Butterfly Transformations with Applications in Computational Linear Algebra,» *Technical Report CSD-950023, UCLA Computer Science Department,* 1995.

[18] D. Stott Parker and D. Le, "How to Eliminate Pivoting from Gaussian Elimination - By Randomizing Instead," *Technical Report CSD-950022, UCLA Computer Science Department,* July 23 1995.

[19] D. Stott Parker, "A Randomizing Butterfly Transformation Useful in Block Matrix Computations," *Technical Report CSD-950024, UCLA Computer Science Department,* 1995.

[20] M. Baboulin, X. S. Li and F.-H. Rouet, "Using Random Butterfly Transformations to Avoid Pivoting in Sparse Direct Methods," *[Research Report] RR-8481,* 2014.

[21] S. Willi-Hans, Matrix Calculus and Kronecker Product with Applications and C++ Programs, World Scientific Publishing, 1997.

[22] R. A. Horn and C. R. Johnson, Matrix Analysis, NY: Cambridge University Press,

1985.

[23] G. W. Stewart, Introduction to Matrix Computations, Academic Press, May 1973, p. 278.

[24] K. J. Horadam, Hadamard Matrices and Their Applications, Princeton University Press, 2007.

[25] S. Georgiou, C. Koukouvinos and J. Seberry, "Hadamard matrices, orthogonal designs and construction algorithms," *Designs 2002: Further computational and constructive design theory,* p. 133–205, 2003.

[26] J. J. Sylvester, "Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers," *Philosophical Magazine,* vol. 34, p. 461–475, 1867.

[27] P. P. Kanjilal, Adaptive Prediction and Predictive Control, Stevenage: IET, 1995, p. 210.

[28] . E. W. Weisstein, "Hadamard Matrix," [Online]. Available: http://mathworld.wolfram.com/HadamardMatrix.html.

[29] D. S. Parker, "Explicit Formulas for the Results of Gaussian Elimination," *Technical Report CSD-950025, UCLA Computer Science Department,* 1995.

[30] M. Marcus and H. Minc, A Survey of Matrix Theory and Matrix Inequalities, NY: Dover, 1964.

[31] D. S. Parker and D. Le, "Quadtree Matrix Algorithms Revisited: Basic Issues and their Resolution," *Technical Report CSD-950028, UCLA Computer Science Department,* 1995.

[32] R. Butt, Introduction to Numerical Analysis Using MATLAB, Jones & Bartlett Learning, 2009.

[33] W. Kahan, Basic Issues in Floating Point Arithmetic and Error Analysis, Lecture

Notes, Berkeley University, 1999.

[34] A. Edelman, "Eigenvalues and Condition Numbers of Random Matrices," *SIAM J. Matrix anal. Appl.,* vol. 9, no. 4, pp. 543-560, 1988.

[35] Y. Saad, Iterative Methods for Sparse Linear Systems 2nd Ed, SIAM, 2000.

[36] Σ. Παπαδάκης και Κ. Διαμαντάρας, Προγραμματισμός και Αρχιτεκτονική Συστημάτων Παράλληλης επεξεργασίας, Εκδόσεις Κλειδάριθμος, 2012.

[37] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," *IEEE Trans. Comput.,* vol. 21, no. 9, p. 948–960, 1972.

[38] J. J. Dongarra, I. S. Duff, D. C. Sorensen and H. Van Der Vorst, Solving Linear Systems on Vector and Shared Memory Computers, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990.

[39] R. Yves, The impact of vector and parallel architectures on the Gaussian elimination algorithm, New York, NY, USA: Halsted Press, 1990.

[40] E. Bampis, J. C. Konig and D. Trystram, "Impact of Communications on the Complexity of the Parallel Gaussian Elimination," *Parallel Computing,* vol. 17, no. 1, pp. 55-61, 1991.

[41] M. Veldhorst, "Gaussian Elimination with Partial Pivoting on a MIMD Computer," *J. Parallel Distr. Computing,* vol. 6, pp. 62-68, 1989.

[42] F. F. Rivera, R. Doallo, J. D. Bruguera, E. L. Zapata and R. Peskin, "Gaussian Elimination with Pivoting on Hypercubes," *Parallel Computing,* vol. 14, no. 1, pp. 51-60, 1990.

[43] A. Buttari, J. Langou, J. Kurzak and J. Dongarra, "A class of parallel tiled linear algebra algorithms for multicore architectures," *Parallel Comput.,* vol. 35, pp. 38-53, 2009.

[44] G. Quintana-Orti, E. S. Quintana-Orti, R. A. van de Geijn, F. G. Zee and E. Chan, "Programming algorithms-by-blocks formatrix computations onmultithreaded

architectures," *ACMTrans.Math. Softw.,* vol. 36, no. 3, pp. 1-26, 2009.

[45] L. Grigori, J. W. Demmel and H. Xiang, "Communication avoiding Gaussian elimination," in *in Proceedings of the IEEE/ACM SuperComputing Conference*, 2008.

[46] V. Volkov and J. W. Demmel, "LU, QR and Cholesky factorizations using vector capabilities of GPUs," *Tech. rep. UCB/EECS-2008-49, University of California, Berkeley. LAPACK Working Note 202.,* 2008.

[47] MathWorks, Inc., MATLAB 7 Getting Started Guide, The MathWorks, Inc., 2008.

[48] L. N. Trefethen and R. S. Schreiber, "Average-Case Stability of Gaussian Elimination," *SIAM J. Matrix Anal. Appl.,* vol. 11, no. 3, pp. 335-360, 1990.

[49] R. T. Gregory and D. L. Karney, A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, 1969.