ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Διπλωματική Εργασία Ειδίκευσης στα Εφαρμοσμένα Μαθηματικά

Θέμα

ΠΡΟΡΡΥΘΜΙΣΗ ΜΕ ΓΡΑΦΗΜΑΤΑ ΚΑΙ ΠΡΟΣΕΓΓΙΣΤΙΚΗ ΑΝΤΙΣΤΡΟΦΗ

ΔΕΤΣΗΣ ΣΤΥΛ. ΦΩΤΙΟΣ

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πολύ τον επιβλέποντα καθηγητή μου κύριο Δρακόπουλο Μιχάλη για την καθοδήγηση και επίβλεψη της διπλωματικής εργασίας μου.

Επίσης ευχαριστώ τα μέλη ΔΕΠ του τμήματος Μαθηματικών του Ε.Κ.Π.Α. κυρίους Δουγαλή Βασίλειο και Θηλυκό Δημήτριο για την τιμή που μου έκαναν να βρίσκονται στην τριμελή επιτροπή.

Και τέλος ένα ευχαριστώ σε όσους με στήριξαν για να την τελειώσω.

Περιεχόμενα

Περίληψη. Στην παρούσα εργασία θα μελετήσουμε δύο είδη προρρυθμιστών για την επίλυση γραμμικών εξισώσεων με επαναληπτικές μεθόδους. Ο πρώτος βασίζεται σε προσεγγιστική αντιστροφή πίνακα και ο δεύτερος κατασκευάζεται με χρήση γραφημάτων. Αφού δούμε αυτές τις δύο τεχνικές θα γενικεύσουμε τη δεύτερη ώστε να λειτουργεί για μεγαλύτερο πλήθος πινάκων. Στο τέλος θα δοκιμάσουμε τη βελτιωμένη μέθοδο σε ένα εκτενές σύνολο προβλημάτων από επιστημονικές και τεχνολογικές εφαρμογές για να εξετάσουμε την αποτελεσματικότητας αυτής της γενίκευσης.

Λέξεις κλειδιά. αραιά γραμμικά συστήματα, αραιοί πίνακες, γραφήματα, επαναληπτικές μέθοδοι, προρρύθμιση, προσεγγιστική αντιστροφή.

1 Εισαγωγή.

Θεωρούμε το γραμμικό σύστημα εξισώσεων

$$A\mathbf{x} = \mathbf{b},\ A \in \mathbb{R}^{n\times n}, \mathbf{x}, \mathbf{b} \in \mathbb{R}^n. \tag{1.1}$$

Ο πίνακας A είτε είναι πυκνός, με n^2 το πλήθος στοιχεία, είτε είναι αραιός, όπου το μεγαλύτερο μέρος των στοιχείων του είναι μηδέν. Εδώ ο A είναι ένας μεγάλος σε μέγεθος, αραιός, και συμμετρικός πίνακας.

Τέτοια συστήματα συναντούμε πολύ συχνά, όπως για παράδειγμα στις μερικές διαφορικές εξισώσεις. Η επίλυση τέτοιων συστημάτων γίνεται είτε με άμεσες μεθόδους είτε με επαναληπτικές μεθόδους. Για μεγάλα προβλήματα οι άμεσες μέθοδοι γίνονται απαγορευτικές λόγω του μεγέθους εργασίας και του χώρου αποθήκευσης που απαιτούν. Οι πιο αποτελεσματικές και ευρέως διαδεδομένες επαναληπτικές μέθοδοι είναι οι εξής :

- μέθοδος **συζυγών κλίσεων** (conjugate gradient CG).
- μέθοδος **δισυζυγών κλίσεων** (biconjugate gradient BiCG),
- μέθοδος **σταθεροποιημένων δισυζυγών κλίσεων** (biconjugate gradient stabilized BiCGSTAB),
- και μέθοδος **γενικευμένων ελαχίστων υπολοίπων** (generalized minimal residual GMRES),

Στις επαναληπτικές μεθόδους δεδομένης της αρχικής τιμής $\mathbf{x_0}$, υπολογίζονται επαναληπτικά νέες προσεγγίσεις $\mathbf{x_k}$ της ακριβούς λύσης

$$\mathbf{x} = A^{-1}\mathbf{b}.$$

Η προσέγγιση $\mathbf{x_k}$ είναι αποδεκτή σαν λύση εάν το υπόλοιπο

$$\mathbf{r_k} = \mathbf{b} - A\mathbf{x_k}$$

ικανοποιεί την

$$\|\mathbf{r_k}\| / \|\mathbf{b}\| \leq tol,$$

όπου *tol* η τάξη ακρίβειας που θέλουμε να έχουμε.

Γενικά, η σύγκλιση δεν είναι εγγυημένη ή μπορεί να είναι εξαιρετικά αργή. Για αυτό το αρχικό σύστημα (1.1) μετασχηματίζεται σε ένα ισοδύναμο, με καλύτερες ιδιότητες σύγκλισης μέσω της προρρύθμισης. Για να γίνει αυτό, θεωρούμε έναν προρρυθμιστή M και εφαρμόζουμε τον επαναληπτικό αλγόριθμο επίλυσης είτε στο κατά δεξιά είτε στο κατά αριστερά προρρυθμισμένο σύστημα

$$AM\mathbf{y} = \mathbf{b},\ \mathbf{x} = M\mathbf{y}, \quad \text{ή} \quad MA\mathbf{x} = M\mathbf{b}. \tag{1.2}$$

Επομένως, ο M θα πρέπει να επιλεγεί έτσι ώστε το γινόμενο AM (ή MA) να είναι μια καλή προσέγγιση του μοναδιαίου. «Καλός» προρρυθμιστής είναι αυτός που θα βελτιώσει το χρόνο σύγκλισης της επαναληπτικής μεθόδου. Ο χρόνος υλοποίησης του προρρυθμιστή μαζί με τον χρόνο της επαναληπτικής διαδικασίας με προρρύθμιση θα πρέπει να είναι πολύ μικρότερος από το χρόνο της επαναληπτικής διαδικασίας χωρίς προρρύθμιση. Η εύρεση ενός καλού προρρυθμιστή, ο οποίος να είναι και εύκολο να κατασκευαστεί είναι ένα δύσκολο πρόβλημα, και συχνά δεν είναι εφικτό να ικανοποιηθούν και οι δύο ιδιότητες. Πολλοί προρρυθμιστές είναι κατάλληλοι μόνο για συγκεκριμένα είδη προβλημάτων.

Στην παρούσα εργασία θα χρησιμοποιήσουμε δυο μεθόδους προρρύθμισης: τη μέθοδο SPAI [10] στην οποία ο προρρυθμιστής προσεγγίζει τον *αντίστροφο* του A και τη μέθοδο προρρύθμισης του Vaidya που βασίζεται στην προσέγγιση της αραιής δομής του A από ένα αραιότερο γράφημα.

2 Η μέθοδος των συζυγών κλίσεων.

Όπως αναφέραμε παραπάνω υπάρχουν αρκετές επαναληπτικές μέθοδοι για επίλυση γραμμικών συστημάτων. Μια βασική μέθοδος είναι αυτή των συζυγών κλίσεων (conjugate gradient) [9]. Την χρησιμοποιούμε για προβλήματα των οποίων ο πίνακας είναι αραιός, συμμετρικός και θετικά ορισμένος. Τέτοια προβλήματα προκύπτουν σε πολλές εφαρμογές όπως προβλήματα μερικών διαφορικών εξισώσεων και προβλήματα βελτιστοποίησης.

Conjugate Gradient

$\mathbf{r_0} = \mathbf{b} - A\mathbf{x_0}$

$\mathbf{p_0} = \mathbf{r_0}$
$k = 0$
επανάλαβε
$\quad \mathbf{a_k} = \dfrac{\mathbf{r_k}^T \mathbf{r_k}}{\mathbf{p_k}^T A \mathbf{p_k}}$
$\quad \mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{a_k p_k}$
$\quad \mathbf{r_{k+1}} = \mathbf{r_k} - \mathbf{a_k} A \mathbf{p_k}$
$\quad$ εάν $\|\mathbf{r_{k+1}}\| / \|\mathbf{b}\| \leq$ tol
$\quad\quad$ έξοδος από επαναληπτική διαδικασία
$\quad \beta_\mathbf{k} = \dfrac{\mathbf{r_{k+1}}^T \mathbf{r_{k+1}}}{\mathbf{r_k}^T \mathbf{r_k}}$
$\quad \mathbf{p_{k+1}} = \mathbf{r_{k+1}} + \beta_\mathbf{k} \mathbf{p_k}$
$\quad k = k + 1$
επέστρεψε το $\mathbf{x_{k+1}}$

Η μέθοδος των συζυγών κλίσεων έχει αποκτήσει το όνομα της διότι δημιουργεί μια ακολουθία συζυγών διανυσμάτων. Αυτά τα διανύσματα είναι υπόλοιπα που προκύπτουν κατά την διάρκεια των επαναλήψεων. Καθώς επίσης και η κλίση του δευτεροβάθμιου συναρτησοειδούς,

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - b^T\mathbf{x} + c \tag{2.1}$$

η ελαχιστοποίηση του οποίου, εφόσον ο πίνακας A είναι συμμετρικός, ισούται με τη λύση του συστήματος. Η ελαχιστοποίηση της (2.1) προκύπτει αν μηδενίσουμε την

$$f'(\mathbf{x}) = \frac{1}{2}A^T\mathbf{x} + \frac{1}{2}A\mathbf{x} - b \tag{2.2}$$

Ορισμός 2.1. *Ορίζουμε ως σφάλμα στο i βήμα της μεθόδου το διάνυσμα :*

$$\mathbf{e_i} = \mathbf{x_i} - \mathbf{x}. \tag{2.3}$$

όπου $\mathbf{x_i}$ η προσεγγιστική λύση στο i βήμα και $\mathbf{x}$ η ακριβής λύση του συστήματος.

Ορισμός 2.2. *Ορίζουμε την νόρμα ενέργειας σύμφωνα με τον τύπο*

$$\|\mathbf{e}\|_A = (\mathbf{e}^T A\mathbf{e})^{\frac{1}{2}}. \tag{2.4}$$

Θεωρητικά η μέθοδος φτάνει στην ακριβή λύση σε ακριβώς n επαναλήψεις. Όμως λόγω των υπολογιστικών σφαλμάτων στρογγύλευσης αυτό δεν είναι πάντα δυνατό. Όταν όμως η σύγκλιση της είναι ικανοποιητική μπορούμε να χρησιμοποιήσουμε τη μέθοδο συζύγων κλίσεων σαν επαναληπτική μέθοδο και να σταματήσουμε τις επαναλήψεις όταν επιτύχουμε την επιθυμητή ακρίβεια για τα δεδομένα του προβλήματος που επιλύουμε.

Η σύγκλιση της μεθόδου εξαρτάται από τον δείκτη κατάστασης $k(A)$ του πίνακα A. Όσο μεγαλύτερο δείκτη κατάστασης έχει ο πίνακας μας τόσο πιο αργή θα είναι η σύγκλιση της μεθόδου. Ένα άνω φράγμα του σφάλματος στο βήμα i της μεθόδου, είναι το

$$\|\mathbf{e_i}\|_A \leq \left(\frac{k(A)-1}{k(A)+1}\right)^i \|\mathbf{e_0}\|_A . \tag{2.5}$$

Επομένως εάν έχουμε ένα πίνακα A με μεγάλο δείκτη κατάστασης θα χρειαστεί να προρρυθμίσουμε το σύστημα μας με ένα πίνακα M ώστε ο νέος πίνακας MA να έχει μικρότερο δείκτη κατάστασης και επομένως μία γρηγορότερη σύγκλιση της μεθόδου. Για αυτό και στα πειράματα μας θα χρησιμοποιήσουμε τη μέθοδο συζυγών κλίσεων σε προρρυθμισμένο γραμμικό σύστημα γνωστή και ως Preconditioned Conjugate Gradient.

Preconditioned Conjugate Gradient
$\mathbf{r_0} = \mathbf{b} - A\mathbf{x_0}$
$\mathbf{z_0} = M^{-1}\mathbf{r_0}$
$\mathbf{p_0} = \mathbf{z_0}$
$k = 0$
επανάλαβε
  $\mathbf{a_k} = \dfrac{\mathbf{r_k}^T\mathbf{z_k}}{\mathbf{p_k}^T A\mathbf{p_k}}$
  $\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{a_k}\mathbf{p_k}$
  $\mathbf{r_{k+1}} = \mathbf{r_k} - \mathbf{a_k}A\mathbf{p_k}$
  εάν $\|\mathbf{r_{k+1}}\| / \|\mathbf{b}\| \leq$ tol
    έξοδος από επαναληπτική διαδικασία
  $\mathbf{z_{k+1}} = M^{-1}\mathbf{r_{k+1}}$
  $\beta_\mathbf{k} = \dfrac{\mathbf{z_{k+1}}^T\mathbf{r_{k+1}}}{\mathbf{z_k}^T\mathbf{r_k}}$
  $\mathbf{p_{k+1}} = \mathbf{z_{k+1}} + \beta_\mathbf{k}\mathbf{p_k}$
  $k = k + 1$
επέστρεψε το $\mathbf{x_{k+1}}$

Αυτή είναι η μέθοδος που θα χρησιμοποιήσουμε στα αριθμητικά μας πειράματα.

Στο κεφάλαιο 3 θα δούμε κάποιους προρρυθμιστές. Θα παρουσιάσουμε τον αραιό προσεγγιστικό αντίστροφο από τον αλγόριθμο SPAI και ένα εύκολα αντιστρέψιμο πίνακα όμοιο με τον πίνακα A από τον αλγόριθμο του Vaidya. Έπειτα θα κάνουμε κάποιες αλλαγές στον αλγόριθμο του Vaidya ώστε να είναι

λειτουργικός για μεγαλύτερη γκάμα πινάκων. Και τέλος στο κεφάλαιο 4 θα παρουσιάσουμε αριθμητικά αποτελέσματα από προβλήματα που επιλύθηκαν χρησιμοποιώντας τους προρρυθμιστές αυτούς.

3 Προρρυθμιστές

3.1 Jacobi.

Ο προρρυθμιστής Jacobi δημιουργείται από την αντίστοιχη επαναληπτική μέθοδο Jacobi η οποία προσπαθεί να λύσει το σύστημα (1.1) ως προς κάθε εξίσωση, δηλαδή

$$A\mathbf{x} = \mathbf{b}$$

$$\sum_{j=1}^{n} a_{i,j}x_j = b_i \quad , \quad i = 1, \ldots, n$$

$$a_{i,i}x_i + \sum_{j=1, j\neq i}^{n} a_{i,j}x_j = b_i \quad , \quad i = 1, \ldots, n$$

$$a_{i,i}x_i = -\sum_{j=1, j\neq i}^{n} a_{i,j}x_j + b_i \quad , \quad i = 1, \ldots, n$$

$$x_i = \left(-\sum_{j=1, j\neq i}^{n} a_{i,j}x_j + b_i \right) / a_{i,i} \quad , \quad i = 1, \ldots, n$$

Από την τελευταία σχέση παίρνουμε την επαναληπτική μέθοδο **Jacobi** που ορίζεται ως,

$$x_i^{(k)} = \left(-\sum_{j=1, j\neq i}^{n} a_{i,j}x_j^{(k-1)} + b_i \right) / a_{i,i} \tag{3.1.1}$$

Εάν εκφράσουμε τον πίνακα A ως $A = D - L - U$, όπου

- $D = \begin{cases} a_{i,j} & i = j \\ 0 & i \neq j \end{cases}$
- $L = \begin{cases} -a_{i,j} & i > j \\ 0 & i \leq j \end{cases}$
- $U = \begin{cases} -a_{i,j} & i < j \\ 0 & i \geq j \end{cases}$

Τώρα με την βοήθεια της παραπάνω έκφρασης του A μπορούμε να γράψουμε τη σχέση $(3.1.1)$ σαν εξίσωση πινάκων

$$Dx_i^{(k)} = \left((U+L)x_j^{(k-1)} + b_i\right)$$

$$x_i^{(k)} = D^{-1}\left((U+L)x_j^{(k-1)} + b_i\right)$$

$$x_i^{(k)} = D^{-1}(U+L)x_j^{(k-1)} + D^{-1}b_i$$

$$x_i^{(k)} = D^{-1}(D-A)x_j^{(k-1)} + D^{-1}b_i$$

$$x_i^{(k)} = (I - D^{-1}A)x_j^{(k-1)} + D^{-1}b_i \tag{3.1.2}$$

Από την $(3.1.2)$ παίρνουμε τον προρρυθμιστή Jacobi, ο οποίος είναι αποτελεσματικός σε περιπτώσεις που ο πίνακας A είναι διαγώνια κυρίαρχος. Αλλά όπως θα δούμε και στα παραδείγματα στο κεφάλαιο 4 δεν ισχύει πάντα αυτό.

3.2 Incomplete Cholesky

Μια άλλη κατηγορία προρρυθμιστών στηρίζεται στην παραγοντοποίηση

$$M = HH^T \tag{3.2.1}$$

Ορισμός 3.1. *Παραγοντοποίηση Cholesky ενός ερμιτιανού και θετικά ορισμένου πίνακα A λέγεται η παραγοντοποίηση της μορφής*

$$A = LL^T.$$

Όπου ο L είναι ένας κάτω τριγωνικός πίνακας με πραγματικά και θετικά διαγώνια στοιχεία. Κάθε ερμιτιανός, θετικά ορισμένος πίνακας έχει μοναδική παραγοντοποίηση Cholesky.

Ο αλγόριθμος Cholesky [1] κατά γραμμές είναι ο εξής :

Για $i = 1, \ldots, n$

Για $j = 1, \ldots, i-1$

$$L_{i,j} = \frac{1}{L_{j,j}}\left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k}L_{j,k}\right)$$

$$L_{i,i} = \left(A_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2 \right)^{\frac{1}{2}}$$

Ανάλογα έχουμε και τον αλγόριθμο Cholesky κατά στήλες :

Για $j = 1, \ldots, n$

$$L_{j,j} = \left(A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2 \right)^{\frac{1}{2}}$$

Για $i = j + 1, \ldots, n$

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right)$$

Η επιλογή του αλγόριθμου γίνετε βάση το πως είναι αποθηκευμένος ο A ή από την δομή των A και L στην περίπτωση που είναι αραιοί κ.λ.π.

Ένας τρόπος για να επιλέξουμε τον H στην $(3.2.1)$ είναι να τροποποιήσουμε το L με μια ατελή παραγοντοποίηση Cholesky. Δηλαδή $LL^T = HH^T + C$, όπου C καλείται πίνακας σφάλματος και είναι εκείνος που προσπαθεί να περιορίσει το γέμισμα (fill-in) στοιχείων σε μηδενικές θέσεις του πίνακα A. Έτσι ο πίνακας H προκύπτει από την ακριβή παραγοντοποίηση Cholesky του πίνακα A και απορρίπτοντας κάποια στοιχεία. Τα κριτήρια για την απόρριψη στοιχείων είναι τα εξής:

- απόρριψη με κριτήρια μεγέθους,
- απόρριψη με κριτήρια θέσης.

Οπότε ο αλγόριθμος της παραγοντοποίηση Cholesky τροποποιείται ως εξής ώστε να έχουμε την ατελή παραγοντοποίηση Cholesky

Για $i = 1, \ldots, n$

Για $j = 1, \ldots, i - 1$

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j}^{\star} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right)$$

$$L_{i,i} = \left(A_{i,i}^{\star} - \sum_{k=1}^{i-1} L_{i,k}^2 \right)^{\frac{1}{2}}$$

Στη θέση των $A_{i,i}^{\star}$ και $A_{i,j}^{\star}$ μπορούμε να έχουμε τα στοιχεία $A_{i,i}$ και $A_{i,j}$ αντίστοιχα ή τροποποιημένα σύμφωνα με την μέθοδο προρρύθμισης μας [2].

Για το κριτήριο της απόρριψης λόγω μεγέθους κάνουμε τον εξής έλεγχο

$$\left(A_{i,j} - \sum_{k=1}^{i-1} L_{k,i} L_{k,j}\right)^2 < \psi A_{i,i} A_{j,j}$$

και απορρίπτουμε εκείνα τα στοιχεία που δεν ικανοποιούν την παραπάνω σχέση.

Για το κριτήριο της απόρριψης λόγω θέσης απλά απορρίπτουμε εκείνα τα στοιχεία $L_{j,i}$ για τα οποία τα οποία ισχύει $A_{j,i} = 0$.

3.3 Αραιός προσεγγιστικός αντίστροφος.

Θα εξετάσουμε στη συνέχεια τον προρρυθμιστή αραιό προσεγγιστικό αντίστροφο που παράγεται με τον αλγόριθμο SPAI (SParse Approximate Inverse) [10]. Όπως φανερώνει και το όνομα ο προρρυθμιστής έχει την δομή του A^{-1} και την ιδιότητα $AM \approx I$. Επειδή όμως εξ' αρχής δεν γνωρίζουμε την δομή του A^{-1} ξεκινάμε συνήθως με την δομή του διαγώνιου και επαυξάνουμε σταδιακά την δομή του προρρυθμιστή ώστε να αποκτήσει περίπου την δομή του A^{-1}.

Για μια δοσμένη αραιή δομή ο πίνακας M είναι η λύση του προβλήματος ελαχιστοποίησης,

$$\min_{m_k} \|Am_k - e_k\|_2, \ k = 1, \ldots, n \tag{3.3.1}$$

όπου $e_k = (0, \ldots, 0, 1, 0, \ldots, 0)^T$,
η οποία ανάγεται σε n ανεξάρτητα προβλήματα ελαχίστων τετραγώνων

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|(AM - I)e_k\|_2^2, \tag{3.3.2}$$

Εδώ χρησιμοποιούμε την Frobenius νόρμα για διευκόλυνση στους υπολογισμούς μας.

Καθώς οι στήλες του M είναι ανεξάρτητες η μία από την άλλη, αρκεί να παρουσιάσουμε τον αλγόριθμο για μία από αυτές, την οποία θα την συμβολίζουμε με m_k. Τώρα ας ονομάσουμε $\mathcal{J}$ το σύνολο των δεικτών j τέτοιοι ώστε $m_k(j) \neq 0$. Συμβολίζουμε το διάνυσμα που περιέχει μόνο τα μη-μηδενικά στοιχεία των αγνώστων $m_k(\mathcal{J})$ με $\hat{m}_k$. Έπειτα, ας ονομάσουμε $\mathcal{I}$ το σύνολο των δεικτών i τέτοιοι ώστε το $A(i, \mathcal{J})$ δεν είναι ταυτοτικά μηδέν. Αυτό μας επιτρέπει να αποβάλλουμε όλες τις μηδενικές γραμμές στον υποπίνακα $A(., \mathcal{J})$. Συμβολίζουμε τον υποπίνακα $A(\mathcal{I}, \mathcal{J})$ με $\hat{A}$. Ομοίως, ορίζουμε $\hat{e}_k = e_k(I)$. Εάν τώρα θέσουμε $n_1 = |\mathcal{I}|$ και $n_2 = |\mathcal{J}|$, βλέπουμε ότι η επίλυση του $(3.3.1)$

ως προς m_k είναι ισοδύναμη με την επίλυση του

$$\min_{m_k} \left\| \hat{A}\hat{m}_k - \hat{e}_k \right\|_2 \tag{3.3.3}$$

ως προς $\hat{m}_k$. Το $n_1 \times n_2$ πρόβλημα ελαχίστων τετραγώνων (2.3) είναι εξαιρετικά μικρό διότι οι A και M είναι πολύ αραιοί πίνακες. Εάν ο A είναι μη ιδιάζων, ο υποπίνακας $\hat{A}$ πρέπει να είναι πλήρους βαθμού (full rank). Συνεπώς, η QR παραγοντοποίηση του $\hat{A}$ είναι

$$\hat{A} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{3.3.4}$$

όπου ο R είναι ένας μη ιδιάζων άνω τριγωνικός $n_2 \times n_2$ πίνακας. Εάν θέσουμε $\hat{c}$=Q^T $\hat{e}_k$, η λύση του $(3.3.3)$ είναι

$$\hat{m}_k = R^{-1}\hat{c}(1 : n_2). \tag{3.3.5}$$

Επιλύουμε την $(3.3.5)$ για κάθε $k = 1, \ldots, n$ και θέτουμε $m_k(\mathcal{J}) = \hat{m}_k$. Αυτό δίνει έναν προσεγγιστικό αντίστροφο M, ο οποίος ελαχιστοποιεί την $\|AM - I\|_F$ για τη δοσμένη αραιή δομή.

Τώρα σκοπός μας είναι να βελτιώσουμε τον M αυξάνοντας την αραιή δομή του ώστε να πάρουμε έναν πιο αποτελεσματικό προρρυθμιστή. Για να το επιτύχουμε αυτό, θα ελαττώσουμε το τρέχον σφάλμα $\|AM - I\|_F$, δηλαδή το $\|Am_k - e_k\|_2$ για κάθε $k = 1, \ldots, n$. Υπενθυμίζουμε ότι $\hat{m}_k$ είναι η βέλτιστη λύση του προβλήματος ελαχίστων τετραγώνων $(3.3.1)$, και συμβολίζουμε το υπόλοιπό του με

$$r = A(., \mathcal{J})\hat{m}_k - e_k. \tag{3.3.6}$$

Εάν $r = 0$, το m_k είναι ακριβώς η k-στήλη του A^{-1} και δε βελτιώνεται περαιτέρω. Τώρα υποθέτουμε ότι $r \neq 0$ και αποδεικνύουμε πώς αυξάνουμε το σύνολο των δεικτών $\mathcal{J}$ για να ελαττώσουμε το $\|r\|_2$. Εφόσον οι A και m_k είναι αραιοί, τα περισσότερα στοιχεία του r είναι μηδενικά και συμβολίζουμε με $\mathcal{L}$ το εναπομείναν σύνολο δεικτών l για τους οποίους $r(l) \neq 0$. Ουσιαστικά το $\mathcal{L}$ ισούται με το $\mathcal{I}$, καθώς το $\hat{r}$ δεν έχει ακριβώς μηδενικές εισόδους σε πεπερασμένη ακρίβεια. Όμως, εάν το $\mathcal{I}$ δεν περιέχει το k, θα πρέπει να συμπεριληφθεί στο $\mathcal{L}$ εφόσον το $r(k)$ είναι τότε ίσο με 1. Για κάθε $l \in \mathcal{L}$ αντιστοιχεί ένα σύνολο δεικτών $\mathcal{N}_l$, το οποίο αποτελείται από τους δείκτες των μη μηδενικών στοιχείων του $A(l, .)$ τα οποία δεν ανήκουν ακόμη στο $\mathcal{J}$. Οι πιθανοί νέοι υποψήφιοι που ίσως προστεθούν στο $\mathcal{J}$ περιέχονται στο

$$\tilde{\mathcal{J}} = \bigcup_{l \in \mathcal{L}} \mathcal{N}_l. \tag{3.3.7}$$

Τώρα πρέπει να επιλέξουμε νέους δείκτες j οι οποίοι θα οδηγήσουν στην πιο συμφέρουσα μείωση του $\|r\|_2$. Για να το κάνουμε αυτό με οικονομικό

αλλά και αποτελεσματικό τρόπο, θεωρούμε για κάθε $j \in \tilde{\mathcal{J}}$ το μονοδιάστατο πρόβλημα ελαχιστοποίησης

$$\min_{\mu_j \in \mathbb{R}} \|r + \mu_j A e_j\|_2 . \tag{3.3.8}$$

Η λύση του $(3.3.8)$ είναι

$$\mu_j = -\frac{r^T A e_j}{\|A e_j\|_2^2}. \tag{3.3.9}$$

Για κάθε j υπολογίζουμε τη 2-νόρμα ρ_j του νέου υπολοίπου r+$\mu_j A e_j$ όπου το μ_j δίνεται από την $(3.3.9)$:

$$\rho_j^2 = - \|r\|_2^2 - \frac{(r^T A e_j)^2}{\|A e_j\|_2^2}. \tag{3.3.10}$$

Υπάρχει τουλάχιστον ένας δείκτης $j \in \tilde{\mathcal{J}}$ τέτοιος ώστε $r^T A e_j \neq 0$, ο οποίος θα οδηγήσει σε ένα μικρότερο υπόλοιπο στη (2.10). Διαφορετικά

$$0 = r(\mathcal{L})^T A(\mathcal{L}, \tilde{\mathcal{J}}) = r(\mathcal{L})^T A(\mathcal{L}, \mathcal{J} \cup \tilde{\mathcal{J}}), \tag{3.3.11}$$

το οποίο συνεπάγεται ότι το $r(\mathcal{L})$ είναι μηδέν, καθώς ο $A(\mathcal{L}, .)$ είναι πλήρους βαθμού (full rank). Σημειώνουμε ότι το $\mathcal{J} \bigcup \tilde{\mathcal{J}}$ περιέχει τους δείκτες στηλών όλων των μη μηδενικών στοιχείων του $A(\mathcal{L}, .)$ και ότι $\mathcal{J} \bigcap \tilde{\mathcal{J}} = \varnothing$. Ελαττώνουμε το $\tilde{\mathcal{J}}$ στο σύνολο των πιο κερδοφόρων δεικτών j με ελάχιστο ρ_j και το προσθέτουμε στο $\mathcal{J}$. Χρησιμοποιώντας το επεκταμένο σύνολο δεικτών $\mathcal{J}$, επιλύουμε ξανά το αραιό πρόβλημα ελαχίστων τετραγώνων $(3.3.1)$. Αυτό δίνει μία καλύτερη προσέγγιση m_k της k-στήλης του A^{-1}. Επαναλαμβάνουμε αυτή τη διαδικασία για κάθε $k = 1, ..., n$ έως ότου το υπόλοιπο ικανοποιεί μία ορισμένη ανοχή ή μέχρι μία μέγιστη ποσότητα γεμίσματος fill-in επιτευχθεί στο m_k.

Κάθε φορά που αυξάνουμε το σύνολο των μη μηδενικών στοιχείων στο m_k, επιλύουμε ακριβώς το πρόβλημα ελαχίστων τετραγώνων. Τώρα θα αποδείξουμε πώς μπορεί κανείς εύκολα να ενημερώσει την QR παραγοντοποίηση και να μειώσει εξαιρετικά το φόρτο εργασίας της. Υπενθυμίζουμε ότι το $\mathcal{J}$ είναι το τρέχον σύνολο και ότι το $\tilde{\mathcal{J}}$ είναι το σύνολο των νέων δεικτών που θα προστεθούν στο m_k. Συμβολίζουμε με $\tilde{\mathcal{I}}$ τις νέες αντίστοιχες γραμμές, οι οποίες περιέχουν τις μη μηδενικές γραμμές του $A(., \mathcal{J} \cup \tilde{\mathcal{J}})$, και με $\tilde{n}_1$ και $\tilde{n}_2$ το πλήθος των δεικτών στο $\tilde{\mathcal{I}}$ και $\tilde{\mathcal{J}}$. Κατά συνέπεια, χρειάζεται να αντικαταστήσουμε στο $(3.3.3)$ τον υποπίνακα $\hat{A}$ με τον μεγαλύτερο υποπίνακα $A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}})$. Για να επιλύσουμε το $(3.3.3)$, χρησιμοποιούμε τη γνωστή QR παραγοντοποίηση του $\hat{A}$ για να ανανεώσουμε την QR παραγοντοποίηση

του επαυξημένου πίνακα $A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}})$:

$$A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}}) = \begin{pmatrix} \hat{A} & A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & A(\tilde{\mathcal{I}}, \tilde{\mathcal{J}}) \end{pmatrix} \tag{3.3.12}$$

$$= \begin{pmatrix} Q & \\ & I_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} R & Q_1^T A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & Q_2^T A(\mathcal{I}, \tilde{\mathcal{J}}) \\ 0 & A(\tilde{\mathcal{I}}, \tilde{\mathcal{J}}) \end{pmatrix} = \begin{pmatrix} Q & \\ & \mathcal{I}_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & B_2 \end{pmatrix}. \tag{3.3.13}$$

Αυτό απαιτεί μόνο τον υπολογισμό της παραγοντοποίησης QR του B_2. Σημειώνουμε ότι $A(\tilde{\mathcal{I}}, \mathcal{J}) = 0$, διότι το $\mathcal{I}$ περιέχει ήδη τους δείκτες των μη μηδενικών εισόδων που εμφανίζονται στις στήλες $\mathcal{J}$. Έστω

$$B_2 = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \tag{3.3.14}$$

όπου B_2 ένας πίνακας $\tilde{n}_1 + n_1 - n_2 \times \tilde{n}_2$. Χρησιμοποιώντας την (2.14), ξαναγράφουμε την (2.13) ως:

$$A(\mathcal{I} \cup \tilde{\mathcal{I}}, \mathcal{J} \cup \tilde{\mathcal{J}}) = \begin{pmatrix} Q & \\ & I_{\tilde{n}_1} \end{pmatrix} \begin{pmatrix} I_{n_2} & \\ & \tilde{Q} \end{pmatrix} \begin{pmatrix} R & B_1 \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}. \tag{3.3.15}$$

Αυτή η διαδικασία μας επιτρέπει να προσθέσουμε νέους δείκτες στο $\mathcal{J}$ και να λύσουμε το πρόβλημα ελαχίστων τετραγώνων για τη βέλτιστη λύση χωρίς να υπολογίσουμε ξανά ολόκληρη την παραγοντοποίηση QR σε κάθε βήμα. Εάν δε σταματήσουμε τη διαδικασία, ο αλγόριθμος θα υπολογίσει την k-στήλη του A^{-1}. Στην πράξη, παρόλα αυτά, σταματάμε τη διαδικασία τη στιγμή που φτάσουμε την καθορισμένη ανοχή ή φτάσουμε μία μέγιστη ποσότητα γεμίσματος (fill-in). Τώρα θα παρουσιάσουμε ολόκληρο τον αλγόριθμο SPAI, όπου το SPAI σημαίνει Αραιός (SParse) Προσεγγιστικός (Approximate) Αντίστροφος (Inverse).

Ο Αλγόριθμος SPAI.
Για κάθε στήλη $\mathbf{m_k}$ του M:

(a) Επέλεξε μία αρχική αραιή δομή $\mathcal{J}$.

(b) Υπολόγισε τους δείκτες γραμμών $\mathcal{I}$ των αντίστοιχων μη μηδενικών γραμμών και την QR παραγοντοποίηση $(3.3.4)$ του $A(\mathcal{I}, \mathcal{J})$. Έπειτα, υπολόγισε τη λύση $\mathbf{m_k}$ του προβλήματος ελαχίστων τετραγώνων (2.1) και το υπόλοιπό του $\mathbf{r}$ όπως δίνεται από την $(3.3.6)$.

Όσο το $\|\mathbf{r}\|_2 > \varepsilon$:

(c) Θέσε το $\mathcal{L}$ ίσο με το σύνολο των δεικτών l για τα οποία $\mathbf{r}(l) \neq 0$.

(d) Θέσε το $\tilde{\mathcal{J}}$ ίσο με το σύνολο όλων των νέων δεικτών στηλών του A που εμφανίζονται σε όλες τις $\mathcal{L}$ γραμμές αλλά όχι στο $\mathcal{J}$.

(e) Για κάθε $j \in \tilde{\mathcal{J}}$ λύσε το πρόβλημα ελαχιστοποίησης $(3.3.8)$.

(f) Για κάθε $j \in \tilde{\mathcal{J}}$ υπολόγισε το ρ_j από την $(3.3.10)$ και διέγραψε από το $\tilde{\mathcal{J}}$ όλους τους δείκτες εκτός των πιο κερδοφόρων.

(g) Υπολόγισε του αντίστοιχους μη-μηδενικούς δείκτες γραμμών $\tilde{\mathcal{I}}$ και ενημέρωσε την QR παραγοντοποίηση χρησιμοποιώντας την $(3.3.15)$. Ύστερα λύσε τα νέα προβλήματα ελαχίστων τετραγώνων, υπολόγισε το νέο υπόλοιπο $\mathbf{r} = A\mathbf{m_k} - \mathbf{e_k}$, και θέσε $\mathcal{I} = \mathcal{I} \bigcup \tilde{\mathcal{I}}$ και $\mathcal{J} = \mathcal{J} \bigcup \tilde{\mathcal{J}}$.

Παρατηρήσεις.

1. Η αρχική αραιή δομή του M είναι τυχαία και ίσως επιλεγεί κενή ή διαγώνια, εάν καμία a priori πληροφορία για την αραιότητα του A^{-1} δεν είναι διαθέσιμη. Ακόμη για την επίλυση μιας ακολουθίας προβλημάτων με παρόμοια αραιή δομή αλλά ποικίλες εισόδους στο A, μία έξυπνη αρχική σκέψη για τον αρχική αραιή δομή είναι να επιλεγεί η ίδια από τον προηγούμενο υπολογισμένο προσεγγιστικό αντίστροφο. Κάτι τέτοιο μειώνει εξαιρετικά το υπολογιστικό κόστος του M, εφόσον η αρχική αραιή δομή θα είναι σχεδόν βέλτιστη. Σε όλα τα αριθμητικά παραδείγματά μας η αρχική αραιή δομή του M επελέγη να είναι διαγώνια.

2. Εκτός από το κριτήριο τερματισμού στο $\|\mathbf{r}\|_2$, περιορίζουμε το βρόγχο σε ένα μέγιστο αριθμό επαναλήψεων ώστε να περιορίσουμε το μέγιστο γέμισμα (fill-in) ανά στήλη του M. Αυτό το κατώτατο όριο σχεδόν ποτέ δεν επετεύχθη και ο συνολικός αριθμός των μη-μηδενικών εισόδων του M είναι συνήθως συγκρίσιμος με το αντίστοιχο αριθμό στο A.

3. Στο (f) πρώτα ελαττώνουμε το $\tilde{\mathcal{J}}$ στο σύνολο των δεικτών j για τους οποίους το ρ_j είναι μικρότερο ή ίσο από τη μέση τιμή όλων των ρ_j. Από τους εναπομείναντες δείκτες κρατάμε το πολύ s δείκτες με ελάχιστο ρ_j. Εδώ το s θα πρέπει να είναι ένας μικρός ακέραιος προς αποφυγή υπερβολικού γεμίσματος (fill-in), και εμείς θέσαμε το s ίσο με 5 στους περισσότερους αριθμητικούς υπολογισμούς. Αυτό το κριτήριο είναι πολύ φθηνό στον υπολογισμό και αφαιρεί αποτελεσματικά άχρηστους δείκτες. Είναι ανεξάρτητο από παραμέτρους, καθώς χρησιμοποιεί ένα δυναμικό κριτήριο μέσης τιμής και δεν απαιτεί ένα κατώτατο όριο ως είσοδο από το χρήστη. Μια πιο περίπλοκη στάθμιση θα μπορούσε να εφαρμοστεί στην κατανομή του ρ_j για να ελέγχει το ποσοστό στο οποίο γεμίζει ο M.

4. Όταν ενημερώνουμε την QR παραγοντοποίηση (2.15), αποθηκεύουμε τους πίνακες Householder που προκύπτουν από τις παραγοντοποιήσεις των πινάκων B_2 ξεχωριστά και ποτέ δεν υπολογίζουμε τον Q άμεσα.

5. Η διαδικασία εκλογής στο (c) ίσως περιοριστεί στις πιο εύκολα προσβάσιμες γραμμές, για να ελαττώσουμε τις επικοινωνίες και τη ροή δεδομένων. Μπορεί επίσης να περιοριστεί μόνο στα μέγιστα στοιχεία του r.

6. Η μονοδιάστατη ελαχιστοποίηση μπορεί να αντικατασταθεί από άλλη μέθοδο ελαχιστοποίησης όπως τη μέθοδο μεγίστης καθόδου (steepest descent)ή το πρόβλημα ακριβούς ελαχιστοποίησης (exact minimization problem) που σχετίζεται με το $\mathcal{J} \bigcup \{j\}$. Από την εμπειρία μας, το πρώτο δεν είναι αρκετά ακριβές και το τελευταίο είναι πολύ ακριβό.

7. Ο προσεγγιστικός αντίστροφος M που υπολογίζει ο SPAI αλγόριθμος είναι αναλλοίωτος μεταθέσεων. Εάν ο A αντικατασταθεί από τον $P_1 A P_2$, όπου P_1 και P_2 μεταθετικοί πίνακες, λαμβάνουμε τον $P_2^T M P_1^T$ αντί του M.

Ο αλγόριθμος SPAI μπορεί επίσης να εφαρμοστεί για να υπολογίσει έναν αραιό προσεγγιστικό αριστερό αντίστροφο του A, ο οποίος μπορεί να χρησιμοποιηθεί ως αριστερός προρρυθμιστής στο (1.2). Αυτό ίσως δώσει καλύτερο αποτέλεσμα εάν όλες οι γραμμές του A^{-1} είναι αραιές αλλά λίγες στήλες του είναι γεμάτες.

Εάν η επαναληπτική μέθοδος προρρυθμισμένη με τον συγκεκριμένο M δε συγκλίνει, είναι εύκολο να βελτιώσουμε τον M χρησιμοποιώντας την αραιή δομή του M ως αρχική αραιή δομή για τον αλγόριθμο SPAI. Η επανάληψη τότε θα προχωρήσει με το νέο προρρυθμιστή M. Επιπλέον, εφόσον υπολογίζουμε το υπόλοιπο για κάθε μία στήλη $\mathbf{m_k}$ ξεχωριστά του M, είναι εύκολο να ξεχωρίσουμε τις πιο δύσκολες στήλες και να συγκεντρωθούμε σε αυτές για να βελτιώσουμε τη σύγκλιση της επαναληπτικής διαδικασίας. Αυτό ίσως αποδειχθεί χρήσιμο σε σχέση με την ευέλικτη προρρύθμιση (flexible preconditioning), όπου ο προρρυθμιστής έχει αναπροσαρμοστεί κατά τη διάρκεια της επαναληπτικής διαδικασίας (βλέπε [7], [11]).

3.4 Ο προρρυθμιστής του Vaidya

Ο Pravin Vaidya το 1991 είχε προτείνει δύο είδη προρρυθμιστών για συμμετρικούς και διαγώνια υπερτερούντες πίνακες σε ένα επιστημονικό συνέδριο αλλά ποτέ δεν δημοσίευσε τίποτα πάνω σε αυτά που παρουσίασε.

Ορισμός 3.2. ***Διαγώνια υπερτερών πίνακας*** *είναι ο πίνακας του οποίου το άθροισμα των απολύτων τιμών των μη διαγώνιων στοιχείων μίας γραμμής είναι μικρότερο ή ίσο από το αντίστοιχο κατ' απόλυτη τιμή διαγώνιο στοιχείο. Δηλαδή ισχύει,*

$$|a_{i,i}| \geq \sum_{j=1, j\neq i}^{n} |a_{i,j}|, \quad \forall i$$

Ωστόσο κάποια από τη θεωρία που πιθανώς χρησιμοποίησε είχε δημοσιευτεί στην διδακτορική εργασία του, υποψήφιου τότε διδάκτορα και φοιτητή του Vaidya, Keith D. Gremban [8]. Έπειτα από καιρό οι Donor Chen και Sivan Toledo [6] προσπάθησαν να υλοποιήσουν σε κώδικα τον προρρυθμιστή που είχε προτείνει ο Vaidya, ενώ υπήρχαν και αρκετές δημοσιεύσεις θεωρημάτων [4], [5] για τη θεωρία πάνω στην οποία πιθανώς στηρίχτηκε ο Vaidya, η οποία ονομάστηκε support graph theory.

3.4.1 Θεωρία και κατασκευή

Κάθε τετραγωνικός $n \times n$ πίνακας A μπορεί να θεωρηθεί ως ένα βεβαρημένο γράφημα.

Ορισμός 3.3. *Το παραγόμενο γράφημα $G_A = (V_A, E_A)$ ενός $n \times n$ συμμετρικού πίνακα A είναι ένα βεβαρημένο γράφημα του οποίου το σύνολο κορυφών είναι το $V_A = \{1, 2, \ldots, n\}$ και το σύνολο ακμών του είναι το $E_A = \{(i, j) : i \neq j$ και $A_{i,j} \neq 0\}$. Το βάρος της ακμής (i, j) είναι το $A_{i,j}$. Το βάρος μίας κορυφής i είναι το άθροισμα των στοιχείων στην i γραμμή του A.*

Αν ο πίνακας A είναι μη συμμετρικός τότε το γράφημα G_A είναι κατευθυνόμενο, ενώ αν ο πίνακας A είναι συμμετρικός τότε το γράφημα G_A είναι μη κατευθυνόμενο. Εμείς θα ασχοληθούμε με συμμετρικούς πίνακες.

Ορισμός 3.4. *Μέγιστος δεντροπαράγοντας (maximum spanning tree) ενός συνεκτικού γραφήματος είναι το παραγόμενο δέντρο του γραφήματος όπου το άθροισμα των βαρών των ακμών του είναι το μεγαλύτερο δυνατό.*

Ο Vaidya πρότεινε δύο είδη προρρυθμιστών M για αραιούς διαγώνια υπερτερούντες πίνακες. Ο πρώτος είναι ο μέγιστος δεντροπαράγοντας του παραγόμενου γραφήματος του πίνακα A, ενώ ο δεύτερος είναι επέκταση του πρώτου. Δηλαδή αφού βρούμε τον μέγιστο δεντροπαράγοντα προσθέτοντας επιπλέον ακμές του γραφήματος κατασκευάζουμε ένα προρρυθμιστή M του οποίου το παραγόμενο γράφημα G_M είναι υπογράφημα του G_A. Το γράφημα G_M έχει το ίδιο σύνολο κορυφών με το G_A, αλλά έχει ένα υποσύνολο του συνόλου ακμών.

Η είσοδος του αλγόριθμου είναι ένας $n \times n$ συμμετρικός, θετικά ορισμένος, διαγώνια υπερτερών πίνακας (SPDDD) A με $2m+n$ μη-μηδενικά στοιχεία και μία παράμετρος t. Ο αλγόριθμος λειτουργεί ως εξής :

1. Βρίσκουμε πρώτα τον κατά μέγιστη τιμή δεντροπαράγοντα T του γραφήματος G_A.

2. Έπειτα υποδιαιρούμε το δέντρο T σε ένα σύνολο από k συνεκτικά υπογραφήματα $V_1, V_2, \ldots, V_k$ έτσι ώστε κάθε V_i να έχει από n/t εώς $(dn/t)+1$ κορυφές, όπου το d είναι είναι ο μέγιστος αριθμός παιδιών που έχουν οι κορυφές στο δέντρο μας.

3. Δημιουργούμε το γράφημα G_M προσθέτοντας στο δέντρο T την βαρύτερη ακμή μεταξύ των V_i και V_j $\forall i, j$. Δεν προσθέτουμε ακμή στις περιπτώσεις που, δεν υπάρχει ακμή μεταξύ των V_i και V_j, ή η βαρύτερη ακμή ήδη ανήκει στο σύνολο ακμών του δέντρου μας.

Ο προρρυθμιστής μας είναι ο πίνακας που προκύπτει από το γράφημα G_M. Τα βάρη των ακμών (i, j) για $i \neq j$ του προρρυθμιστή είναι ίδια με τα αντίστοιχα βάρη του αρχικού γραφήματος. Κάνουμε όμως μία διόρθωση στα διαγώνια στοιχεία του προρρυθμιστή M, έτσι ώστε το άθροισμα κάθε γραμμής του πίνακα A να είναι ίδιο με το άθροισμα της αντίστοιχης γραμμής του πίνακα M.

Η είσοδος t είναι αυτή που αλλάζει τον προρρυθμιστή μας, όσο μεγαλώνουμε το t τόσο αυξάνονται τα μη-μηδενικά στοιχεία του προρρυθμιστή, με αποτέλεσμα να χάνουμε περισσότερο χρόνο στην κατασκευή του προρρυθμιστή και να αυξάνεται ο χρόνος του κάθε βήματος της επαναληπτικής μεθόδου. Για $t = n$ ο προρρυθμιστής γίνετε ίσος με τον αρχικό πίνακα A. Ενώ για $t = 1$ έχουμε ως προρρυθμιστή τον μέγιστο δεντροπαράγοντα.

3.4.2 Λεπτομέρειες υλοποίησης του αλγορίθμου.

Όπως αναφέρθηκε προηγουμένως το πρώτο βήμα για την δημιουργία του προρρυθμιστή είναι να βρεθεί ο μέγιστος δεντροπαράγοντας. Για τον σκοπό αυτό θα χρησιμοποιήσουμε τον αλγόριθμο του $Prim$ [3].

Ο Αλγόριθμος Prim.

(a) Θεώρησε το γράφημα T με μία κορυφή που παίρνεις στην τύχη.

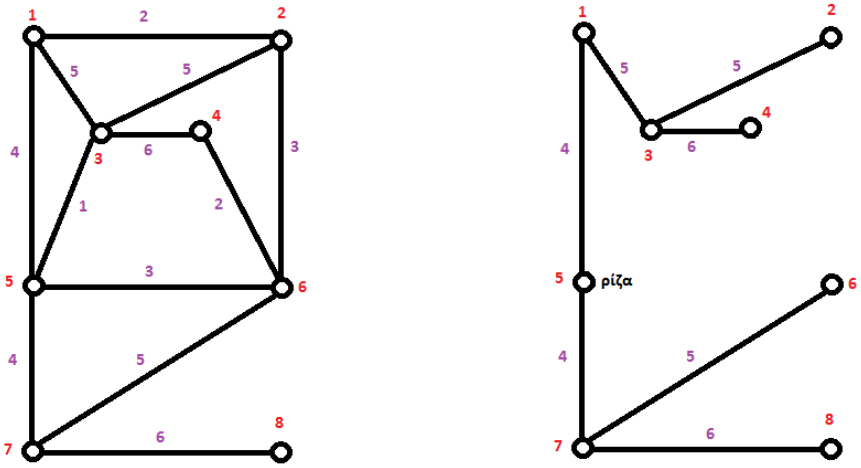
(b) Διάλεξε μια ακμή (i, j) με μέγιστο βάρος από το σύνολο των ακμών που έχουν την κορυφή i στο T και την κορυφή j στο συμπλήρωμα του T. Πρόσθεσε αυτή την ακμή στο T και την κορυφή j στις κορυφές του T.

(c) Αν το T είναι δεντροπαράγοντας του για το γράφημα σταμάτησε, διαφορετικά πήγαινε στο **(b)**.

Ο συγκεκριμένος αλγόριθμος είναι γρήγορος και μπορεί να μας επιστρέψει ένα δέντρο με ρίζα. Ο αλγόριθμος του $Prim$ έχει ως είσοδο το γράφημα μας, με την μορφή πίνακα, και μία κορυφή του γραφήματος ως ρίζα του δέντρου. Ο αλγόριθμος θα μας επιστρέψει το δέντρο με ρίζα ως ένα διάνυσμα π ακεραίων, διάστασης $n \times 1$. Στη θέση $\pi(i)$ έχουμε τον πατέρα της κορυφής i.

Παραθέτουμε ένα παράδειγμα ενός SPDDD 8×8 πίνακα A.

$$A = \begin{bmatrix} 12 & -2 & -5 & 0 & -4 & 0 & 0 & 0 \\ -2 & 13 & -5 & 0 & 0 & -3 & 0 & 0 \\ -5 & -5 & 17 & -6 & -1 & 0 & 0 & 0 \\ 0 & 0 & -6 & 8 & 0 & -2 & 0 & 0 \\ -4 & 0 & -1 & 0 & 12 & -3 & -4 & 0 \\ 0 & -3 & 0 & -2 & -3 & 16 & -5 & 0 \\ 0 & 0 & 0 & 0 & -4 & -5 & 15 & -6 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & 6 \end{bmatrix}$$

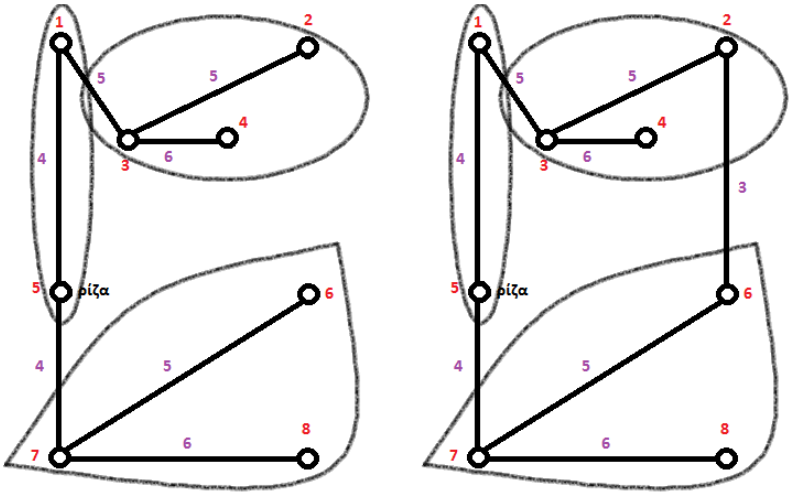


Σχήμα 1: Αριστερά : Το παραγόμενο γράφημα G_A από τον παραπάνω πίνακα.
Δεξιά : Ο μέγιστος δεντροπαράγοντας του γραφήματος.
Με κόκκινο χρώμα η αρίθμηση των κορυφών και με μωβ χρώμα τα βάρη των ακμών.

Το δεύτερο σημαντικό βήμα είναι η υποδιαίρεση του δέντρου T σε k συνεκτικά γραφήματα που έχουν περίπου τον ίδιο αριθμό κορυφών. Θα χρησιμοποιήσουμε τον αλγόριθμο από το [6], όπου βρίσκεται και η απόδειξη

(Θεώρημα 2.1) ότι η συνάρτηση TreePartition χωρίζει το δέντρο μας σε υποδέντρα με πλήθος ακμών ανάμεσα σε n/t και $dn/t + 1$.

TreePartition(vertex i)
#comment : s_i = πλήθος κορυφών στο υποδέντρο με ρίζα την κορυφή i
$s_i = 1$
για κάθε παιδί j του i
εάν $(s_j > n/t + 1)$
TreePartition(j)
εάν $(s_j \geq n/t)$
δημιούργησε ένα καινούριο υποδέντρο με ρίζα το j
αποδέσμευσε το j από παιδί του i
αλλιώς
$s_i = s_i + s_j$



Σχήμα 2: Αριστερά : Η υποδιαίρεση του δέντρου σε υποδέντρα.
Δεξιά : Η επιπλέον ακμή που προσθέτουμε στο δέντρο

Στο παράδειγμά μας ο αλγόριθμος TreePartition χωρίζει το δέντρο μας σε 3 υποδέντρα με σύνολα κορυφών $V_1 = \{1, 5\}$, $V_2 = \{2, 3, 4\}$ και $V_3 = \{6, 7, 8\}$. Έπειτα παίρνουμε κάθε ζευγάρι υποδέντρων και παρατηρούμε ότι μεταξύ των G_1, G_2 και G_1, G_3 η πιθανή ακμή που θα έπρεπε να προστεθεί ανήκει ήδη στο δέντρο μας. Δεν συμβαίνει το ίδιο μεταξύ των G_2, G_3 και επομένως προσθέτουμε την ακμή $(2, 6)$.

3.4.3 Βελτίωση - γενίκευση του αλγορίθμου

Ο αλγόριθμος του Vaidya λειτουργεί για πίνακες Stieltjes.

Ορισμός 3.5. *Πίνακας Stieltjes είναι ο συμμετρικός και θετικά ορισμένος πίνακας που τα μη διαγώνια στοιχεία του είναι αρνητικά.*

Εμείς όμως θέλουμε να γενικεύσουμε αυτό τον αλγόριθμο ώστε να λειτουργεί για περισσότερες κατηγορίες πινάκων. Για αυτό πραγματοποιήσαμε τις παρακάτω αλλαγές:

- Η πρώτη έχει να κάνει με τον αλγόριθμο του Prim. Επειδή τα μη μηδενικά στοιχεία στους πίνακες που εξετάσαμε είναι και θετικά και αρνητικά, βρίσκουμε το κατ' απόλυτη τιμή μέγιστο δεντροπαράγοντα του γραφήματος.

- Η δεύτερη αλλαγή είναι συναρτήσει της παραπάνω επιλογή μας. Και συγκεκριμένα λαμβάνει χώρα στον τρόπο διόρθωσης των διαγώνιων στοιχείων του προρρυθμιστή μας. Υπενθυμίζουμε ότι αφού έχουν επιλεγεί όλες οι ακμές του υπογραφήματος κάνουμε μια διόρθωση στα διαγώνια στοιχεία του προρρυθμιστή μας. Η διόρθωση αυτή γίνετε για να έχει το ίδιο άθροισμα γραμμής ο προρρυθμιστής M με τον πίνακα A του συστήματος (1.1) . Επιλέξαμε το διαγώνιο στοιχείο του προρρυθμιστή μας να έχει τιμή ίση με τη διαφορά του αθροίσματος των απολύτων τιμών της γραμμής του πίνακα A μείον το άθροισμα των απολύτων τιμών των μη διαγώνιων στοιχείων της γραμμής του πίνακα M, δηλαδή

$$M(i,i) = \sum_{j=1}^{n} |A(i,j)| - \sum_{j=1, j\neq i}^{n} |M(i,j)| \quad i = 1, \ldots, n$$

- Η τελευταία αλλαγή που πραγματοποιήσαμε έχει να κάνει με την επιλογή των ακμών, δηλαδή των μη διαγώνιων στοιχείων του πίνακα A, στην περίπτωση που έχουμε να επιλέξουμε μεταξύ ακμών που έχουν το ίδιο βάρος. Στο άρθρο για την υλοποίηση του αλγορίθμου του Vaidya δεν αναφέρεται καθόλου πως αντιμετωπίζεται αυτή η περίπτωση. Στον αλγόριθμο μας όταν έχουμε να πάρουμε απόφαση για το παραπάνω πρόβλημα επιλέγουμε την ακμή με την μικρότερη απόσταση από την κύρια διαγώνιο. Φτάσαμε σε αυτή την απόφαση ώστε όταν έχουμε να εφαρμόσουμε παραγοντοποίηση Cholesky στον προρρυθμιστή μας να μειώσουμε όσο το δυνατόν το fill-in που μπορεί να προκύψει. Όπως θα δούμε και παρακάτω στο παράδειγμα 6 του κεφαλαίου 4 έχουμε

καλύτερα αποτελέσματα από τον υλοποιημένο αλγόριθμο του Vaidya που βρήκαμε στο πακέτο taucs[1].

4 Πειραματικά αποτελέσματα

Τα πειράματα μας έγιναν σε περιβάλλον Linux (διανομή Kubuntu) και με τη χρήση του προγράμματος GNU-Octave.

Έγινε χρήση πινάκων που βρήκαμε από τις ιστοσελίδες συλλογής πινάκων

- του National Institute of Standards and Technology, (http://math.nist.gov/MatrixMarket/)
- και του πανεπιστημίου της Φλόριντα, (http://www.cise.ufl.edu/research/sparse/matrices/).

Για το σύστημα (1.1) αν το διάνυσμα $\mathbf{b}$ δεν δίνεται θα δημιουργούμε ένα τυχαίο διάνυσμα $\mathbf{x}$ με στοιχεία μεταξύ του μηδέν και ένα και θα το πολλαπλασιάζουμε με τον εκάστοτε πίνακα.

Ως επαναληπτική μέθοδο θα χρησιμοποιήσουμε αυτή των συζυγών κλίσεων (Preconditioned Conjugate Gradient). Όπου δεν αναφέρεται η ακρίβεια της μεθόδου την έχουμε ορίσει ίση με 10^{-8}.

Στα επόμενα παραδείγματα οι πίνακες που έχουμε επιλέξει είναι όλοι συμμετρικοί. Όμως ο αλγόριθμος για τον προρρυθμιστή SPAI δεν δημιουργεί συμμετρικούς πίνακες, για αυτό το λόγο αφού δημιουργήσουμε τον προρρυθμιστή μας τον μετατρέπουμε σε συμμετρικό ως εξής :

$$M := (M + M^T)/2$$

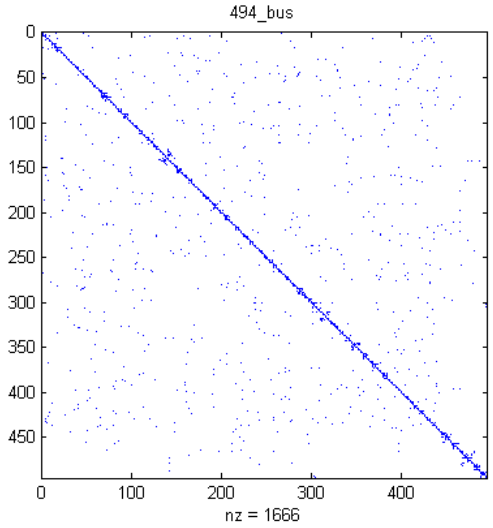
όπου M^T ο ανάστροφος του πίνακα M.

4.1 Οικογένεια πινάκων PSADMIT.

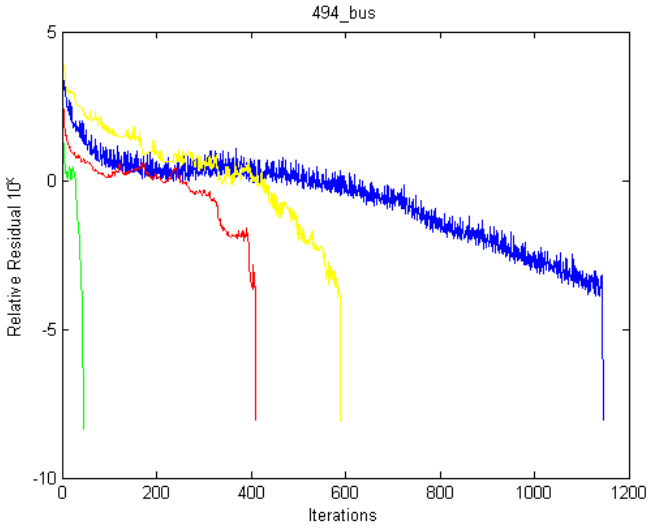
Το πρώτο παράδειγμα που θα δούμε είναι η οικογένεια πινάκων **PSADMIT** (Power systems admittance matrices) που αποτελείται από 4 πίνακες. Προκύπτουν από προβλήματα δικτύων συστημάτων παραγωγής ενέργειας (power system network). Είναι συμμετρικοί με πραγματικά στοιχεία και θετικά ορισμένοι.

[1] http://www.tau.ac.il/ stoledo/taucs/

- **494_bus**, μεγέθους 494×494 με 1666 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 5.8×10^4 και δείκτη κατάστασης 3.9×10^6.

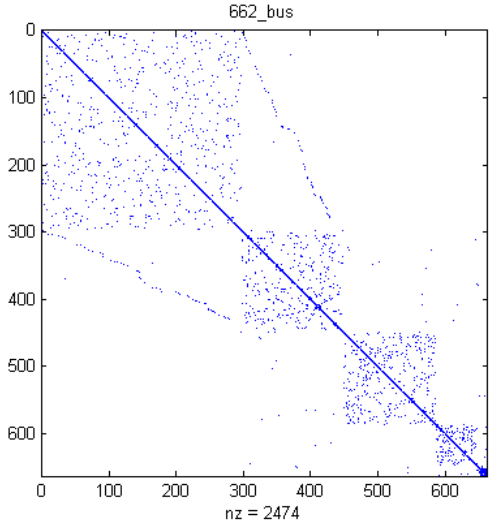


Σχήμα 3: 494_bus

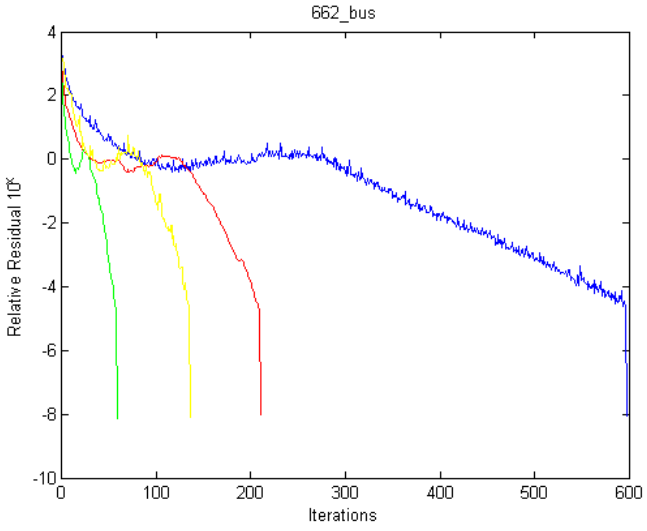


Σχήμα 4: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

- **662_bus**, μεγέθους 662×662 με 2474 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 7.4×10^3 και δείκτη κατάστασης 8.3×10^5.

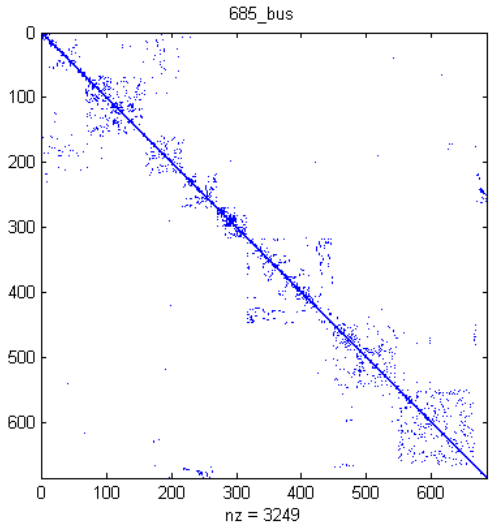


Σχήμα 5: 662_bus

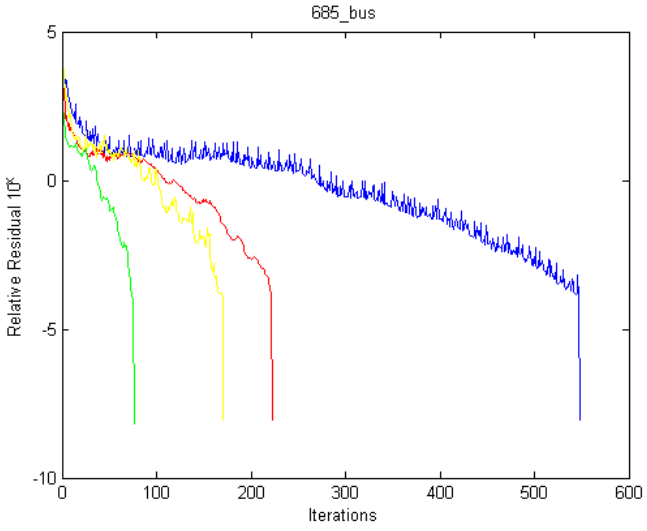


Σχήμα 6: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

- **685_bus**, μεγέθους 685×685 με 3249 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 4×10^4 και δείκτη κατάστασης 5.3×10^5.

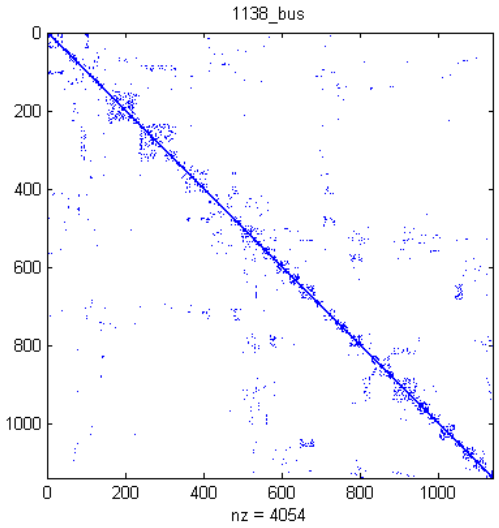


Σχήμα 7: 685_bus

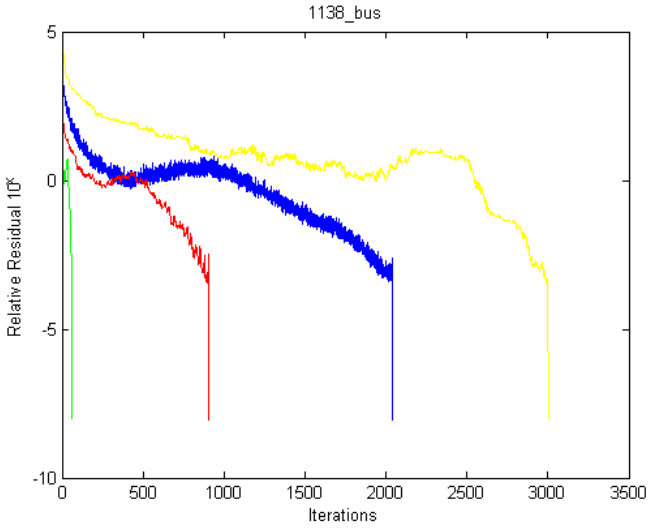


Σχήμα 8: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

- **1138_bus**, μεγέθους 1138×1138 με 4054 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 1.3×10^5 και δείκτη κατάστασης 1×10^2.



Σχήμα 9: 1138_bus



Σχήμα 10: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

Στον παρακάτω πίνακα βλέπουμε τον αριθμό των επαναλήψεων που χρειάστηκε για να συγκλίνει η PCG με ακρίβεια 10^{-8} χρησιμοποιώντας καθόλου προρρύθμιση και με τους προρρυθμιστές Jacobi, Vaidya και SPAI.

Πίνακας 1

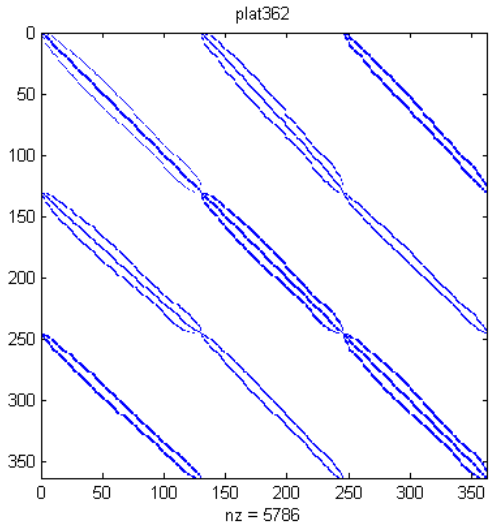
	494_bus	**662_bus**	**685_bus**	**1138_bus**
No Preconditioner	1144	606	547	2041
Jacobi	409	208	222	908
Maximum Spanning Tree	44	56	75	62
SPAI	588	131	170	3001

Παρόλο που δεν είναι διαγώνια υπερτερών πίνακες πετυχαίνουμε πάρα πολύ καλή σύγκλιση χρησιμοποιώντας ως προρρυθμιστή το μέγιστο δεντροπαράγοντα.

4.2 Οικογένεια πινάκων PLATZ.

Η οικογένεια πινάκων **PLATZ** (Platzman's oceanographic models) αποτελείται από 2 πίνακες, οι οποίοι δημιουργούνται από μοντέλα πεπερασμένων διαφορών για ρηχά κύματα στον Ατλαντικό και στον Ινδικό ωκεανό. Οι πίνακες είναι οι εξής :

- ο plat362 ένας συμμετρικός με πραγματικά στοιχεία και θετικά ορισμένος πίνακας με μέγεθος 362×362 και 3074 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 5 και δείκτη κατάστασης 7.1×10^{11}.



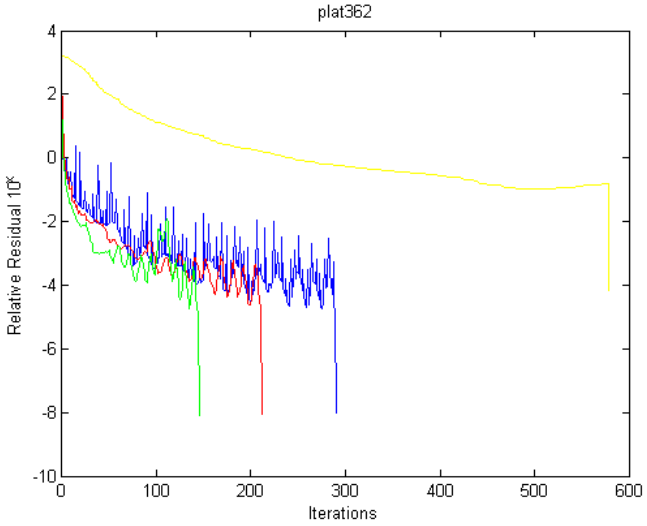
Σχήμα 11: plat362

Στον πίνακα 2 βλέπουμε τον αριθμό των επαναλήψεων που χρειάστηκε για να συγκλίνει η CG με ακρίβεια 10^{-8} και 10^{-12}, χρησιμοποιώντας καθόλου προρρύθμιση και με τους προρρυθμιστές Jacobi, Vaidya και SPAI.

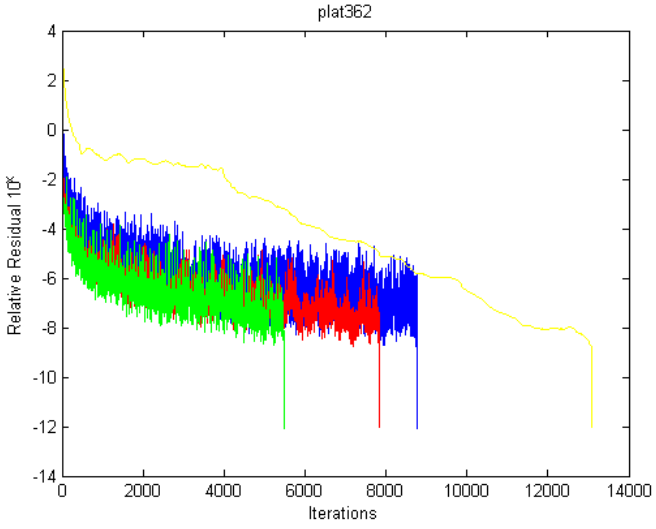
Πίνακας 2

	Iterations $(rr = 10^{-8})$	Iterations $(rr = 10^{-12})$
No Preconditioner	289	8784
Jacobi	211	7836
Maximum Spanning Tree	144	5476
SPAI	7730	13086

Στα επόμενα δύο γραφήματα (σχήμα 12 και 13) έχουμε τη σύγκλιση της μεθόδου μας για τάξη ακρίβειας 10^{-8} και 10^{-12}. Όπως παρατηρείται, από τα γραφήματα και τον προηγούμενο πίνακα, ο προσεγγιστικός αντίστροφος δεν συγκλίνει σε επιθυμητό μέγιστο πλήθος επαναλήψεων.



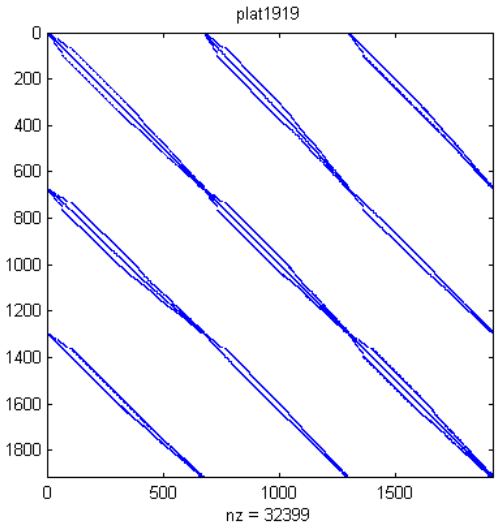
Σχήμα 12: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.



Σχήμα 13: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

- και ο plat1919 ένας συμμετρικός με πραγματικά στοιχεία και θετικά

ορισμένος πίνακας με μέγεθος 1919×1919 και 17159 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 22 και δείκτη κατάστασης 1×10^2.



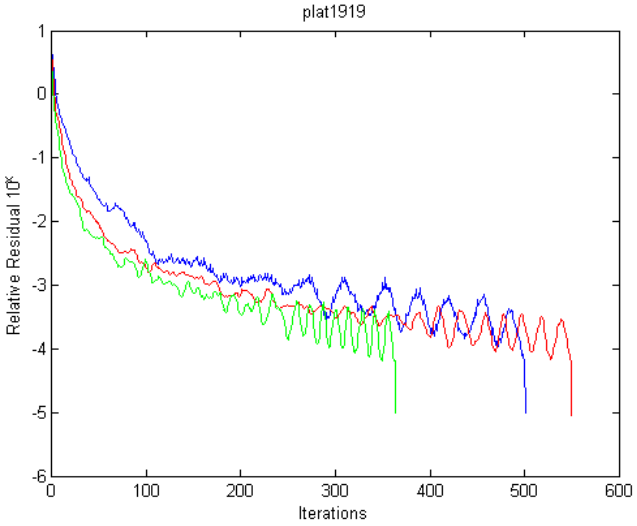
Σχήμα 14: plat1919

Στον πίνακα 3 βλέπουμε τον αριθμό των επαναλήψεων που χρειάστηκε για να συγκλίνει η CG με ακρίβεια 10^{-8} και 10^{-12}, χρησιμοποιώντας καθόλου προρρύθμιση και με τους προρρυθμιστές Jacobi, Vaidya. Λόγω της μεγάλη υπολογιστικής ανάγκης δεν ήταν εφικτό να υπολογιστεί ο προσεγγιστικός αντίστροφος.

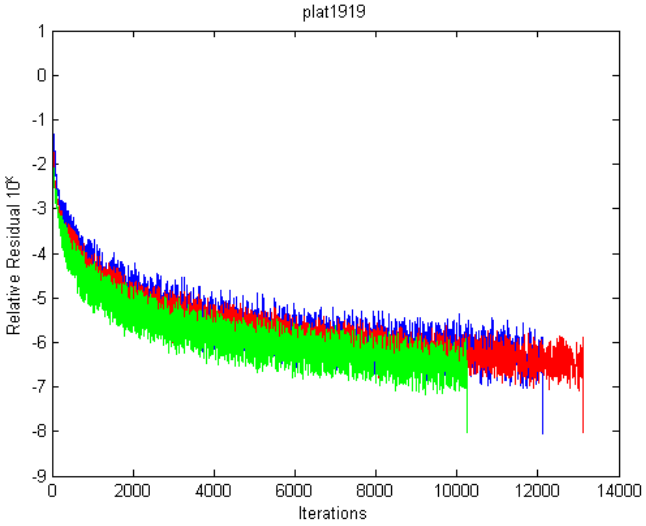
Πίνακας 3

	Iterations $(rr = 10^{-8})$	Iterations $(rr = 10^{-12})$
No Preconditioner	500	12367
Jacobi	549	13125
Maximum Spanning Tree	363	9424

Στα επόμενα δύο γραφήματα (σχήμα 15 και 16) έχουμε τη σύγκλιση της μεθόδου μας για τάξη ακρίβειας 10^{-8} και 10^{-12}.



Σχήμα 15: Με μπλε χρώμα έχουμε την σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε την σύγκλιση με προρρυθμιστή Jacobi.
Και με πράσινο χρώμα την σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.

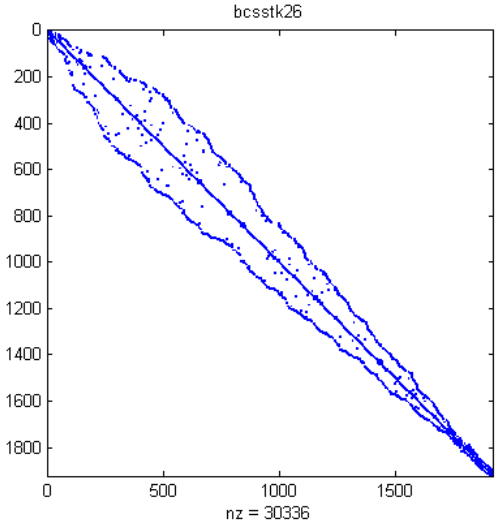


Σχήμα 16: Με μπλε χρώμα έχουμε την σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε την σύγκλιση με προρρυθμιστή Jacobi.
Και με πράσινο χρώμα την σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα).

4.3 Οικογένεια πινάκων BCSSTRUC4

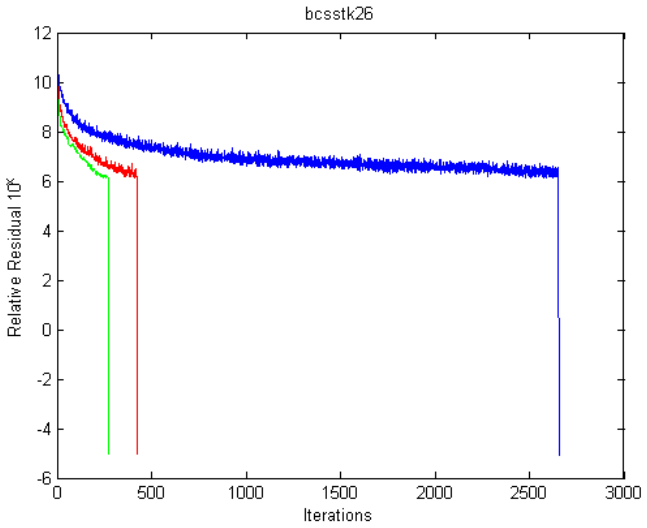
Από την οικογένεια πινάκων BCSSTRUC4, (BCS Structural Engineering Matrices) θα δούμε τον πίνακα BCSSTK26. Προκύπτει από σεισμική ανάλυση για πυρηνικούς αντιδραστήρες.

Είναι συμμετρικός και θετικά ορισμένος πίνακας με μέγεθος 1922×1922 με 30336 μη μηδενικά στοιχεία. Έχει Frobenius νόρμα 4×10^{11} και δείκτη κατάστασης 2.8×10^{8}.

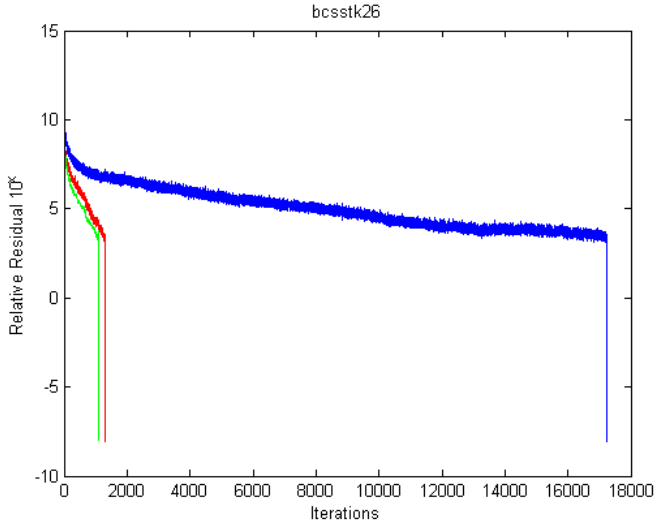


Σχήμα 17: bcsstk26

Στο σχήμα 17 έχουμε την δομή του πίνακα ενώ στα σχήματα 18 και 19 έχουμε την σύγκλιση της μεθόδου για τάξη ακρίβειας 10^{-5} και 10^{-8} αντίστοιχα. Λόγω της μεγάλη υπολογιστικής ανάγκης δεν ήταν εφικτό να υπολογιστεί ο προσεγγιστικός αντίστροφος.



Σχήμα 18: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.



Σχήμα 19: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.

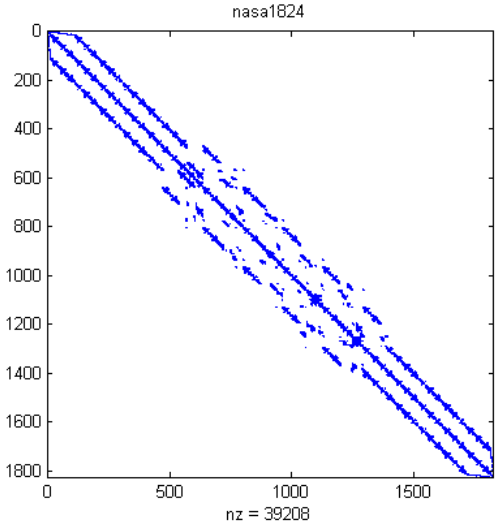
Στον πίνακα 4 έχουμε τις επαναλήψεις της PCG για τάξη ακρίβειας σφάλματος 10^{-5} και 10^{-8}.

Πίνακας 4

	Iterations ($rr = 10^{-5}$)	Iterations ($rr = 10^{-8}$)
No Preconditioner	2655	17226
Jacobi	424	1308
Maximum Spanning Tree	270	1102

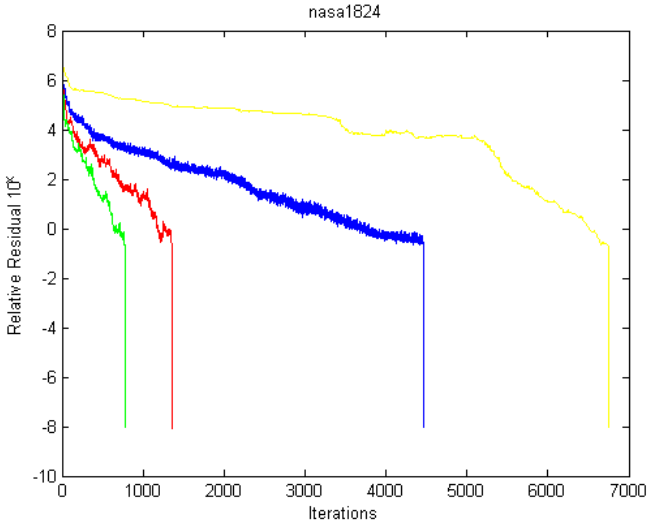
4.4 Ο πίνακας Boeing/nasa1824.

Ο πίνακας Boeing/nasa1824 έχει μεγέθος 1824×1824 με 39208 μη μηδενικά στοιχεία. Έχει νόρμα 2.1×10^7 και δείκτη κατάστασης 5.9×10^6. Είναι συμμετρικός, θετικά ορισμένος πίνακας με πραγματικά στοιχεία δεν είναι όμως διαγώνια υπερτερών.

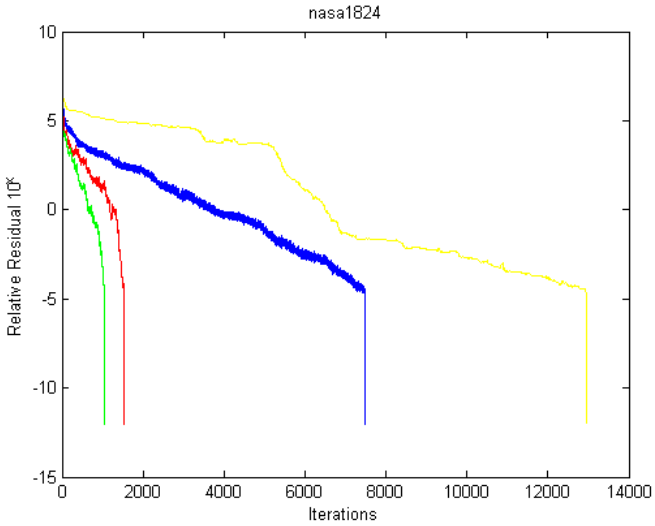


Σχήμα 20: nasa1824

Στα σχήματα 21 και 22 παρατηρούμε τη σύγκλιση της επαναληπτικής μεθόδου μας σε τάξη ακρίβειας 10^{-8} και 10^{-12} αντίστοιχα.



Σχήμα 21: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.



Σχήμα 22: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

Όπως παρατηρούμε από τον πίνακα 4 χρησιμοποιώντας ως προρρυθμιστή

το μέγιστο δεντροπαράγοντα πετυχαίνουμε καλή σύγκλιση. Δεν συμβαίνει το ίδιο όμως με τον προσεγγιστικό αντίστροφο.

Πίνακας 5

	Iterations ($rr = 10^{-8}$)	Iterations ($rr = 10^{-12}$)
No Preconditioner	4461	7473
Jacobi	1354	1525
Maximum Spanning Tree	744	1045
SPAI	6751	12956

4.5 Πακτωμένος πρόβολος

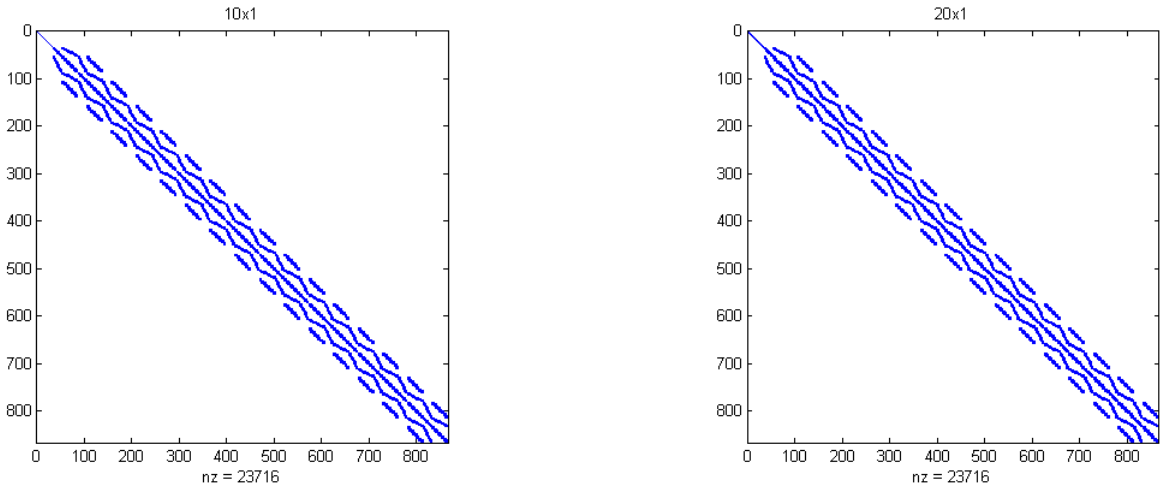
Ο πακτωμένος πρόβολος είναι μια δοκός στερεωμένη μόνο στο ένα άκρο της. Ασκώντας μια κάθετη δύναμη ομοιογενώς σε όλο το μήκος της δοκού μεταφέρει όλη αυτή τη δύναμη που δέχεται στην στήριξη της.

Στο παραδείγματα μας θα θεωρήσουμε διαφορετικά μήκη της δοκού αλλά ίδια πλάτη ίσα με 1. Η διαμέριση που θα έχει θα είναι παντού η ίδια και ίση με 16×8. Μεγαλώνοντας το μήκος της δοκού και κρατώντας την σταθερή την διαμέριση αυξάνεται ο δείκτης κατάστασης του πίνακα, ενώ έχουμε σταθερό το μέγεθος ίσο με 866×866.

Οι πίνακες του παραπάνω προβλήματος δεν είναι συμμετρικοί. Έτσι πριν δημιουργήσουμε τους προρρυθμιστές μας θα τους μετατρέψουμε σε συμμετρικούς με το εξής τρόπο :

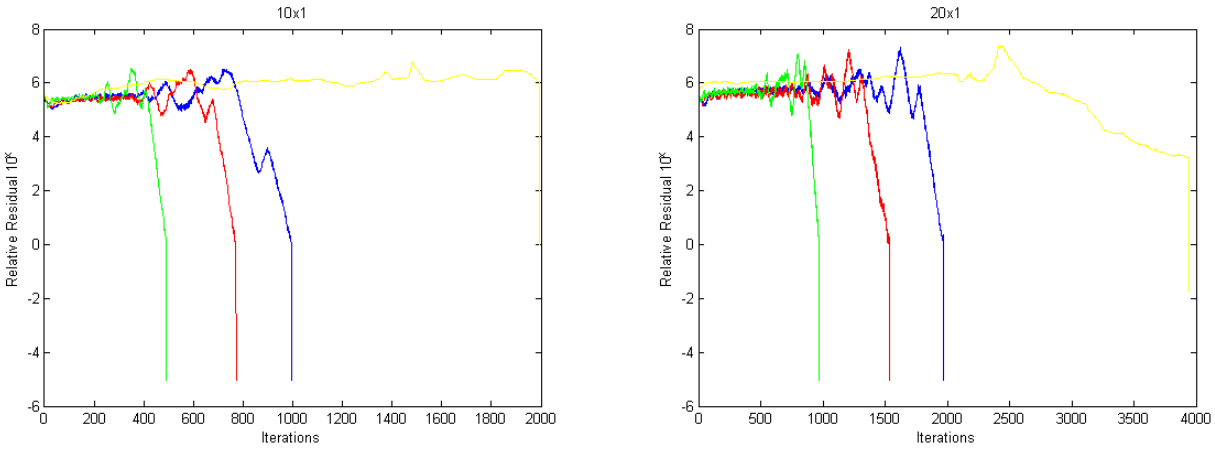
$$A := (A + A^T)/2$$

Στα σχήματα 23 και 25 έχουμε την δομή των πινάκων σε αντιστοιχία με τα διαφορετικά μήκη δοκών. Ενώ στα σχήματα 24 και 26 έχουμε τη σύγκλιση της μεθόδου με τις διάφορες μεθόδους προρρυθμισης.

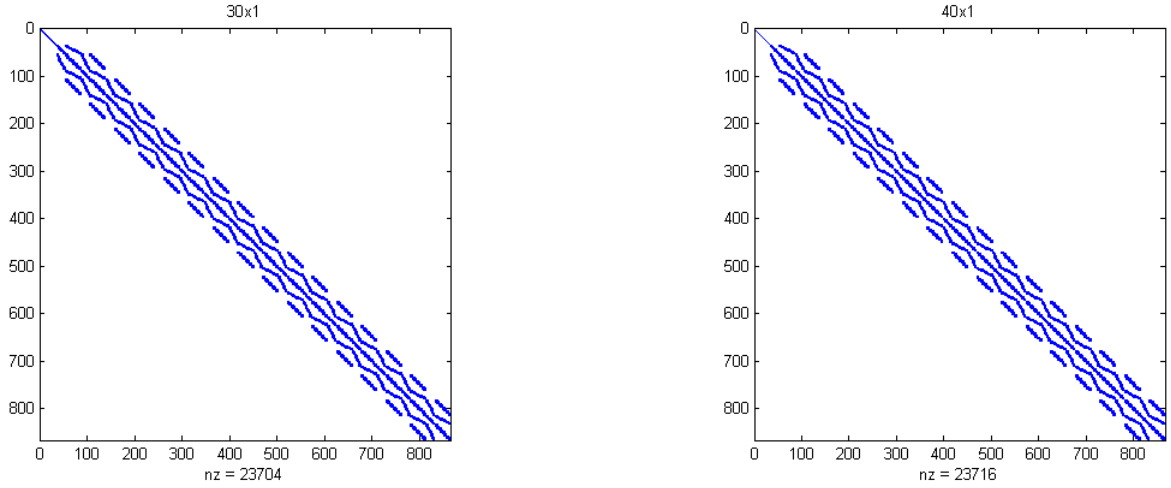


Σχήμα 23: Αριστερά ο πίνακας με μήκος ακτίνας 10 και δεξιά με μήκος ακτίνας 20.

Ο πίνακας της ακτίνας με μήκος 10 έχει δείκτη κατάστασης 1.36×10^{13} και ο πίνακας της ακτίνας με μήκος 20 έχει 2.69×10^{13}.

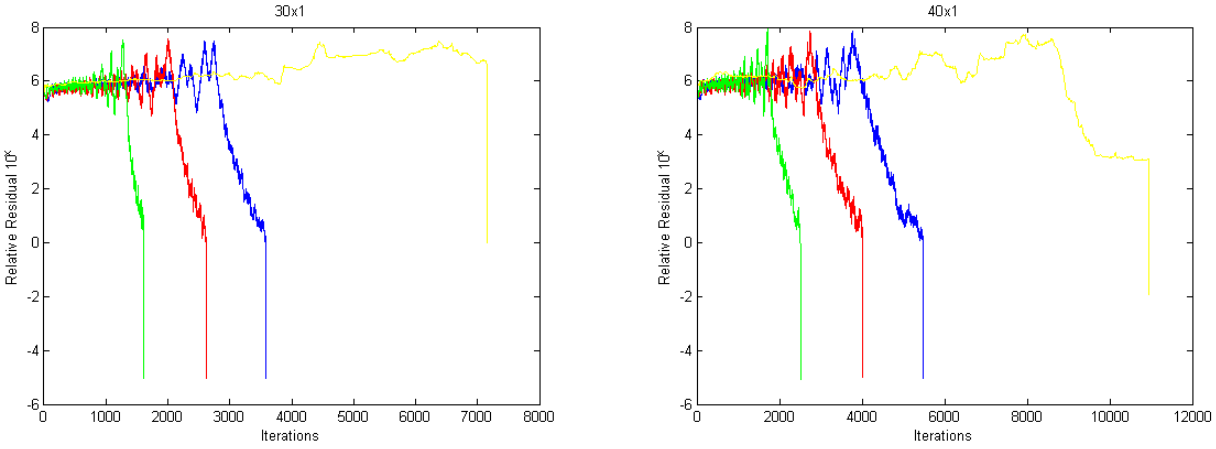


Σχήμα 24: Αριστερά ο πίνακας με μήκος ακτίνας 10 και δεξιά με μήκος ακτίνας 20.
Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.



Σχήμα 25: Αριστερά ο πίνακας με μήκος ακτίνας 30 και δεξιά με μήκος ακτίνας 40.

Οι πίνακες της ακτίνας με μήκος 30 και 40 έχουν δείκτη κατάστασης 4.03×10^{13} και 5.36×10^{13} αντίστοιχα.



Σχήμα 26: Αριστερά ο πίνακας με μήκος ακτίνας 30 και δεξιά με μήκος ακτίνας 40.
Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Και με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.

Στον πίνακα 6 έχουμε αναλυτικά τις επαναλήψεις της PCG με και χωρίς

προρρύθμιση για κάθε ένα από τους πίνακες. Η τάξη ακρίβειας που χρησιμοποιήσαμε και για τους 4 πίνακες είναι 10^{-5}.

Πίνακας 6

	10×1	20×1	30×1	40×1
No Preconditioner	996	1970	3579	5473
Jacobi	771	1538	2617	4010
Maximum Spanning Tree	491	970	1609	2507
SPAI	3115	5582	12163	15169

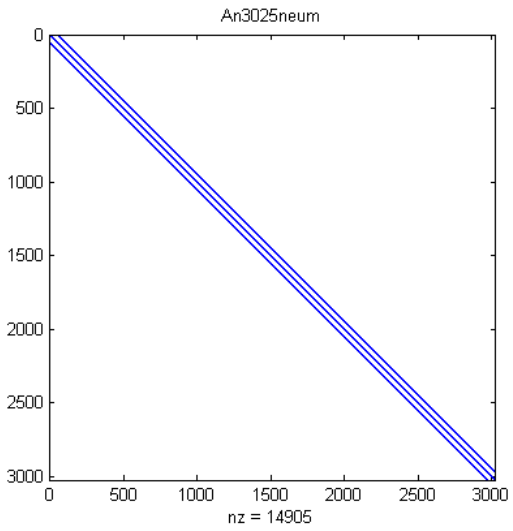
4.6 Προβλήματα μερικών διαφορικών εξισώσεων

Τα επόμενα δύο προβλήματα τα δημιουργήσαμε μέσω του πακέτου taucs.

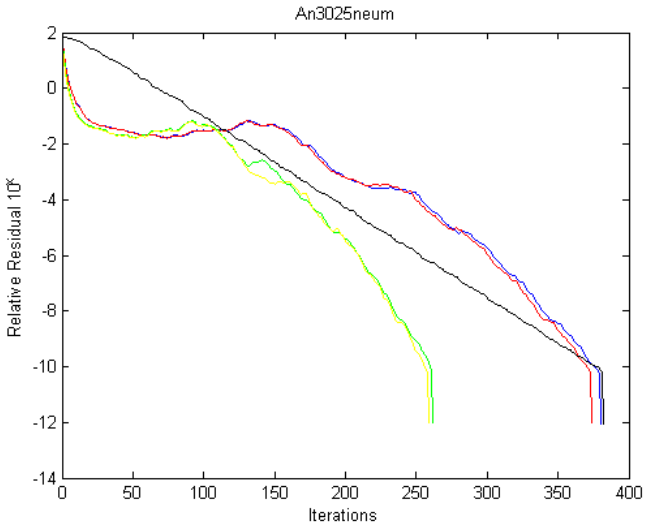
Και στις δύο περιπτώσεις έχουμε Stieltjes πίνακες μεγέθους 3025×3025. Δυστυχώς όμως τα προβλήματα που δημιουργούνται μέσω του πακέτου είναι εύκολα επιλύσιμα. Ο μόνος λόγος που τα παραθέτουμε είναι για να γίνει η σύγκριση μεταξύ του προρρυθμιστή του Vaidya και του αντίστοιχου βελτιωμένου δικού μας.

4.6.1 Δύο διαστάσεων με συνοριακές συνθήκες Neumann

Ο πίνακας μας έχει 14905 μη μηδενικά στοιχεία και δείκτη κατάστασης 8.69×10^4. Στο σχήμα 27 έχουμε την δομή του πίνακα μας και στο σχήμα 28 την σύγκλιση της PCG.



Σχήμα 27: An3025neum.



Σχήμα 28: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.
Και με μαύρο χρώμα την σύγκλιση με τον δεντροπαράγοντα από το taucs.

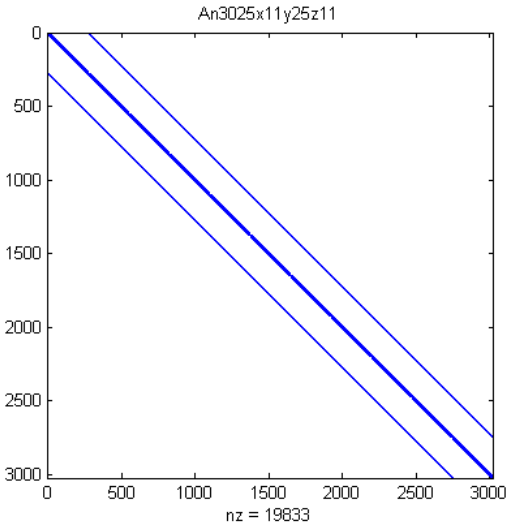
Στον επόμενο πίνακα βλέπουμε τον αριθμό των επαναλήψεων της επαναληπτικής μεθόδου μας με τους προρρυθμιστές μας.

Πίνακας 7

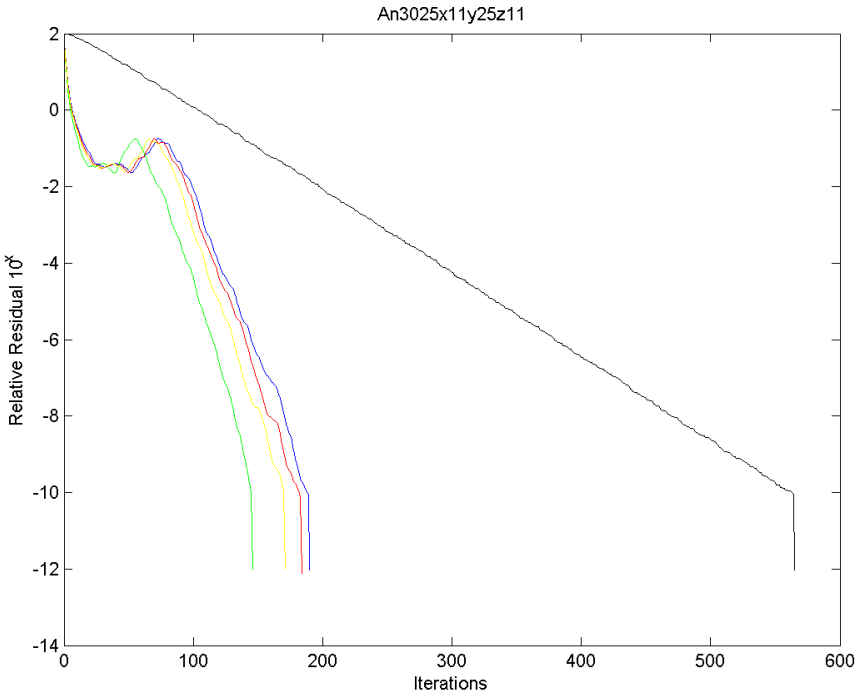
	Iterations $(rr = 10^{-12})$
No Preconditioner	379
Jacobi	373
Maximum Spanning Tree	261
SPAI	258
Maximum Spanning Tree (taucs)	381

4.6.2 Τριών διαστάσεων

Ο πίνακας μας έχει 19833 μη μηδενικά στοιχεία και δείκτη κατάστασης 6.21×10^6. Στο σχήμα 29 έχουμε την δομή του πίνακα και στο σχήμα 30 την σύγκλιση της PCG.



Σχήμα 29: Ax11y25z11.



Σχήμα 30: Με μπλε χρώμα έχουμε τη σύγκλιση χωρίς προρρυθμιστή.
Με κόκκινο χρώμα έχουμε τη σύγκλιση με τον προρρυθμιστή Jacobi.
Με πράσινο χρώμα τη σύγκλιση με προρρυθμιστή τον μέγιστο δεντροπαράγοντα.
Με κίτρινο χρώμα τη σύγκλιση με τον προρρυθμιστή SPAI.
Και με μαύρο χρώμα την σύγκλιση με τον δεντροπαράγοντα από το taucs.

Στον πίνακα 8 βλέπουμε τον αριθμό των επαναλήψεων της PCG με τους προρρυθμιστές μας.

Πίνακας 8

	Iterations $(rr = 10^{-12})$
No Preconditioner	189
Jacobi	183
Maximum Spanning Tree	145
SPAI	170
Spanning Tree (taucs)	564

5 Συμπεράσματα

Στην εργασία αυτή διερευνήσαμε κυρίως τον προρρυθμιστή του Vaidya και κατά δεύτερο λόγο το SPAI. Με τις τροποποιήσεις που δοκιμάσαμε είδαμε ότι η προρρύθμιση με γραφήματα μπορεί να γενικευτεί και σε πίνακες που δεν είναι κατ' ανάγκη Stieltjies και διαγώνια υπερτερούντες. Στα περισσότερα παραδείγματα, η τροποποιημένη προρρύθμιση Vaidya, συνέκλινε στο μικρότερο αριθμό επαναλήψεων από τις άλλες μεθόδους προρρύθμισης που δοκιμάσαμε.

Δεν αναφέραμε καθόλου χρόνους ως προς τη σύγκλιση της μεθόδου και του χρόνου υλοποίησης των προρρυθμιστών διότι ακόμα οι αλγόριθμοι δέχονται βελτίωση. Σίγουρα θα είχαμε καλύτερους χρόνους υλοποίησης αν είχαμε προγραμματίσει με άλλη γλώσσα προγραμματισμού π.χ. C, C++, Java.

Δυστυχώς δεν βρέθηκαν άλλοι πίνακες εκτός του πακέτου taucs ώστε να γίνει καλύτερη σύγκριση μεταξύ του αλγορίθμου του Vaidya και του βελτιωμένου που παρουσιάσαμε. Παρόλα αυτά είδαμε σημαντική βελτίωση ως προς τη σύγκλιση της μεθόδου με το βελτιωμένο προρρυθμιστή μας.

Για να βρούμε τον μέγιστο δεντροπαράγοντα ενός γραφήματος θα πρέπει το γράφημα μας να είναι συνεκτικό. Ένα πρόβλημα που συναντήσαμε στην εύρεση πινάκων είναι ότι δεν είχαν συνεκτικά παραγόμενα γραφήματα. Μια επιπλέον βελτίωση του αλγορίθμου θα είναι όταν ένας πίνακας δεν έχει συνεκτικό παραγόμενο γράφημα να μπορεί να βρίσκει όλους τους δεντροπαράγοντες από τα συνεκτικά γραφήματα που υπάρχουν στον πίνακα και προσθέτει τις επιπλέον ακμές. Παρατηρήθηκε ακόμη ότι σε πίνακες που δεν συνέκλινε η επαναληπτική μέθοδος μας χωρίς προρρύθμιση αλλά είχαμε πολύ καλή σύγκλιση με τον προρρυθμιστή Jacobi, είχαμε καλύτερα αποτελέσματα αν δεν κάναμε διόρθωση στα διαγώνια στοιχεία του προρρυθμιστή που προκύπτει από το μέγιστο κατ' απόλυτη τιμή δεντροπαράγοντα.

Ο προρρυθμιστής SPAI θα είχε καλύτερη σύγκλιση αν βάζαμε πιο αυστηρά κριτήρια. Δεν το κάναμε όμως για να έχουμε λιγότερα μη-μηδενικά στοιχεία ώστε να έχουμε μια σύγκριση με το αντίστοιχο αριθμό από τον μέγιστο κατ' απόλυτη τιμή δεντροπαράγοντα. Επιπλέον όσο κάναμε πιο αυστηρά τα κριτήρια τόσο είχαμε μεγαλύτερο χρόνο υλοποίησης.

Η παρούσα εργασία αφήνει ανοιχτά κάποια ζητήματα που προσφέρονται για μελλοντική έρευνα. Χρειάζεται μια θεωρητική διερεύνηση για τη σύγκλιση σε σχέση με τις τροποποιήσεις και να εξεταστούν και άλλοι τρόποι προρρύθμισης με γραφήματα που να μην βασίζονται κατ' ανάγκη στο μέγιστο δεντροπαράγοντα. Είναι χρήσιμο επίσης να κατασκευαστεί ένα προγραμματιστικό πλαίσιο σε σε C/C++ πάνω στο οποίο να αξιολογούνται οι μέθοδοι και ως προς τον υπολογιστικό χρόνο.

Αναφορές

[1] Γ. Δ. Ακρίβης, Β. Α. Δουγαλής, *Εισαγωγή Στην Αριθμητική Ανάλυση*, Πανεπιστημιακές Εκδόσεις Κρήτης.

[2] Μιχάλης Δρακόπουλος, Μανώλης Παπαδρακάκης, *Improving the efficiency of Incomplete Cholesky Preconditionings*.

[3] Παναγιώτης Γ. Σπύρου, *Θεωρία Γραφημάτων*.

[4] Marshal Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, Sivan Toledo, *Support-Graph Preconditioners*.

[5] Eric G. Boman, Bruce Hendrickson, *Support Theory For Preconditioning*.

[6] Donor Chen, Sivan Toledo, *Vaidya's preconditioners: Implementation And Experimental Study*.

[7] E. Chow, Y. Saad, *Approximate Inverse Preconditioners for General Sparse Matrices*, Colorado Conference on Iterative Methods, April 5-9, 1994

[8] Keith D. Gremban, *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*.

[9] Magnus R. Hestene, Eduard Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*

[10] Thomas Huckle, Marcus Grote, *A New Approach to Parallel Preconditioning with Sparse Approximate Inverses*.

[11] Y. Saad, *A Flexible Inner-Outer Preconditioned GMRES Algorithm*, SIAM J. Sci. Stat. Comput. 14(2), p461-469, 1993.

A Κώδικες

A.1 Αλγόριθμοι για τον Vaidya

A.1.1 Prim

```
1  function [P,r]=PrimAbsMax(A,r)
2  %[P,r]=PrimAbsMax(A,r)
3  %Finds the maximum absolute spanning tree of a graph.
4  %Input:   i) A square positive definite matrix "A".
5  %         ii) The vertice "r" that gonna be the root of the
       tree. Default r = 1.
6  %Output: An array 'P' that indicates the tree.
7  %         The root 'r' of the tree.
8  %
9  %Last update 25/04/2014
10
11     L=size(A,1);
12     P=zeros(L,1);
13     if nargin==1
14         r=1;
15     elseif nargin==2
16         b=true;
17         while b
18             if nnz(A(:,r))>1
19                 b=false;
20             else
21                 r=input('Give a new vertice for root cause
                       the given got no edges :');
22             end
23         end
24     end
25
26     I=zeros(L,1);
27     I(1)=r;
28     sm = sum(A~=0)-1;
29     A = -abs(A);
30     A = A + diag(inf*ones(size(A,1),1));
31     A(r,find(A(:,r))) = inf;
32     for k=1:L-1
33         mn = min(min( A(:,I(1:k)) ));%Finding the heaviest
```

```
            edge.
34          if(mn==0)
35              %There is no edge that is connected with the
                    tree.
36              break
37          end
38          [ipnt,jpnt] = find( A(:,I(1:k)) == mn );%Finding the
                heaviest edges.
39          jpnt = I(jpnt);
40          mx = min( sm(ipnt) );
41          i2 = find( sm(ipnt) == mx );
42          mindist = inf;
43          for jj=1:length(i2)
44            if mindist > abs( ipnt(i2(jj)) - jpnt(i2(jj)) )
45              mindist = abs( ipnt(i2(jj)) - jpnt(i2(jj)) );
46              newv = ipnt(i2(jj));
47              J = jpnt(i2(jj));
48            end
49          end
50          I(k+1) = newv;
51          P(newv) = J;
52          A(newv,find(A(:,newv))) = inf;
53      end
54      P(P==0)=inf;
55      P=P';
56      if (sum(P == inf) > 1)
57        warning('Graph is not connected!');
58      end
59  end
```

A.1.2 NumVert

```
1   function NumVert(i)
2   %NumVert(i)
3   %NumVert : Enumerates the number of vertices in the subtree
        rooted at i.
4   %           Uses two global arrays,
5   %            i)  P=the output array of Prim's algorithm.
6   %           ii)  S=ones(1,length(P));
```

```
%Input:  The root i of the subtree.
%Output: Void.

    global P;
    global S;
    for k=find(P==i)
        if find(P==k)~=0
            NumVert(k);
        end
        S(i)=S(i)+S(k);
    end
end
```

A.1.3 TreePartition

```
function [] = TreePartition(r,t)
%[] = TreePartition(i,t)
%TreePartition decompose a tree into k subtrees. It uses two global arrays,
%"P" the output of Prim's algorithm and "S".
%   Input : i) The root "r" of the tree.
%           ii) A parameter "t" that will help to decompose the tree into k
%           subtrees.
%   Output: Void

    global S
    global P
    if nargin==1
        t=ceil(length(S)^0.25);
    end
    n=length(S);
    S(r)=1;
    for j=find(P==r)
        if (S(j)>=n/t+1)
            TreePartition(j,t)
        end
        if (S(j)>=n/t)
            P(j)=inf;
```

```
23          else
24              S(r)=S(r)+S(j);
25          end
26      end
27  end
```

A.1.4 Vaidya

```
1  function M = Vaidya(A,t)
2  %M = Vaidya(A,t)
3  %Vaidya's Preconditioner
4  %   Input : i) A square Stieltjes matrix "A".
5  %           ii) A parameter "t". Default t=length(A)^0.25
6  %   Output: The preconditioner "M" of A.
7
8    Meg=size(A,1);
9    if nargin==1
10     t=ceil(Meg^0.25);
11   end
12
13   global P;
14   global S;
15   b=true;
16   P=zeros(1,Meg);
17   S=ones(1,Meg);
18   M=sparse(Meg,Meg);
19   A=sparse(A);
20   while b
21     r=input('\nDo you want to give a root for the tree? (y
         or n) : ','s');
22       if (r=='y' || r=='Y')
23         fprintf('Give the a vertice for root= 1,...,%d :',
             Meg);
24         r=input('');
25         tic
26         [P,r]=PrimMin(A,r);
27         b=false;
28       elseif (r=='n' || r=='N')
29         tic
```

```
30          [P,r]=PrimAbsMax(A,1);
31          b=false;
32        end
33    end
34
35    %Filling the preconditioner with tree elements.
36    for k=[1:r-1,r+1:Meg]
37      M(k,P(k))=A(k,P(k));
38      M(P(k),k)=A(k,P(k));
39    end
40
41    if t ~= 1
42      NumVert(r);
43      TreePartition(r,t);
44      i=1;
45      for k=find(P==inf)
46          V{i}=SubTree(k);
47          i=i+1;
48       end
49
50
51      k=2;
52      l=length(V);
53      for i=1:l-1
54        for j=k:l
55          ii=1;
56          MX=zeros(length(V{i}),3);
57          for jj=V{i}
58            MX(ii,1)=min(A(V{j},jj));
59            MX(ii,2)=jj;
60            MX(ii,3)=V{j}(find(A(V{j},jj)==MX(ii,1),1));
61             ii=ii+1;
62          end
63          ii=find(MX(:,1)==min(MX(:,1)),1);
64          M(MX(ii,2),MX(ii,3))=A(MX(ii,2),MX(ii,3));
65          M(MX(ii,3),MX(ii,2))=A(MX(ii,2),MX(ii,3));
66        end
67        k=k+1;
68      end
69    end
70    M = M + diag(sum(abs(A)) - sum(abs(M)));
```

```
71    toc
72 end
```

A.2 Αλγόριθμοι για το SPAI

A.2.1 mins

```
1  function el=mins(x,s)
2  %el=mins(x,s)
3  %Returns the indices of the s minimum elements of array x.
4  %
5  %Detsis Fotios stoxasths@math.uoa.gr
6
7     el=zeros(s,1);
8     for i=1:s
9       tmp1=min(x);
10      tmp2=find(x==tmp1);
11      el(i)=min(tmp2);
12      x(el(i))=Inf;
13    end
14 end
```

A.2.2 vrisko

```
1  function vr=vrisko(A,B)
2  %vr=vrisko(A,B)
3  %Takes two arrays A,B with positive (>0) elements and
4  %returns an array with those belongs at A and not at B.
5  %
6  %Detsis Fotios stoxasths@math.uoa.gr
7
8     L=zeros(A(length(A)),1);
9     J=zeros(B(length(B)),1);
10    J=L;
11    L(A)=1;
```

```
12    J(B)=1;
13    K=L-J(1:length(L));
14    vr=find(L-J(1:length(L)));
15    K(vr)=K(vr)+1;
16    vr=find(K);
17  end
```

A.2.3 SPAI

```
1   function [M,b]=taSPAI(A,epsilon,maxit,s)
2   %M=taSPAI(A,epsilon,maxit,s)
3   %Computes the SPAI preconditioner
4   %      parameters:
5   %
6   %        A        (required): input matrix (square, real, and
        sparse)
7   %        epsilon  (optional): epsilon
8   %                             default is epsilon = 0.2
9   %        maxit    (optional): number of improvement steps
        allowed per column
10  %                             default is maxit = inf
11  %        s        (optional): maximum number of new nonzeros
        accepted per step
12  %                             default is s = 5
13  %
14  %Uses two non matlab functions vrisko,mins.
15  %
16  %Detsis Fotios stoxasths@math.uoa.gr
17
18
19    n=size(A,1);
20    if size(A,1)~=size(A,2)
21      waring('The matrix is not square!');
22      return
23    end
24    J=0;
25    if nargin==1
26      epsilon=2e-1;
27      maxit=inf;
```

```
28        s=5;
29      elseif nargin==2
30        maxit=inf;
31        s=5;
32      elseif nargin==3
33        s=5;
34      end
35      I=[];
36      JJ=[];
37      n1=0;
38      n2=0;
39      Mon=eye(n);
40      M=zeros(n);
41      for k=1:n
42        %----------------a
43        J=k;
44        %----------------b
45        [I,b]=find(A(:,J));
46        B=A(I,J);
47        n1=length(I);
48        n2=length(J);
49        [Q R]=qr(B);
50        [a,b]=find(R);
51        a=max(a);
52        R=R(1:a,:);
53        c=zeros(n1,1);
54        c=Q'*Mon(I,k);
55        M(J,k)=R\c(1:n2);
56        r=A(:,J)*M(J,k)-Mon(:,k);
57        mi=0;
58        while norm(r,'fro')>epsilon && mi<maxit
59            %----------------c
60            L=find(r);
61            %----------------d
62            JJ=vrisko(L,J);
63            %----------------e,f
64            mu=zeros(length(JJ),1);
65            rho=zeros(length(JJ),1);
66            for j=1:length(JJ)
67                b=A(:,JJ(j))'*r;
68                nrm=norm(A(:,JJ(j)),2)^2;
```

```
69                  mu(j)=-b/nrm;
70                  rho(j)=sqrt(norm(r,2)^2-b^2/nrm);
71              end
72              b=mean(rho);
73              lgcl=find(rho<=b);
74              x=zeros(length(lgcl),1);
75              x=rho(lgcl);
76              if  s<=length(lgcl)
77                  y=mins(x,s);
78                  JJ=JJ(lgcl(y));
79              else
80                  JJ=JJ(lgcl);
81              end
82              J=sort([J;JJ]);
83              %----------------g
84              [II,b]=find(A(:,J));
85              II=unique(II);
86              II=vrisko(II,I);
87              I=sort([I;II]);
88              n1=length(I);
89              n2=length(J);
90              B=A(I,J);
91              [Q R]=qr(B);
92              [a,b]=find(R);
93              a=max(a);
94              R=R(1:a,:);
95              c=zeros(n1,1);
96              c=Q'*Mon(I,k);
97              M(J,k)=R\c(1:n2);
98              r=A(:,J)*M(J,k)-Mon(:,k);
99              mi=mi+1;
100         end
101     end
102 end
```