ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΓΕΝΙΚΟ ΤΜΗΜΑ

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΜΕΘΟΔΟΙ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΜΕΓΑΛΩΝ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΕ ΠΑΡΑΛΛΗΛΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ

Διδακτορική Διατριβή

Εμμανουήλ Ν. Μαθιουδάκης

Επιβλέπων καθηγητής : Καθηγητής Ι. Σαριδάκης

Χανιά - Ιούνιος 2001

Μέρος της παρούσας εργασίας υποστηρίχθηκε από το ερευνητικό πρόγραμμα ΠΕΝΕΔ 99ΕΔ566.

Ευχαριστίες

Πρώτα θα ήθελα να ευχαριστήσω το σύμβουλο καθηγητή μου, Καθηγητή του Γενικού Τμήματος Γιάννη Γ. Σαριδάκη, ο οποίος μου παρείχε την άρτια επιστημονική καθοδήγηση και την ηθική υποστήριξη στην ολοκλήρωση της παρούσας Διατριβής.

Την Αναπληρώτρια Καθηγήτρια Ελενα Παπαδοπούλου ευχαριστώ για την πολύτιμη συνεισφορά της στην επιστημονική καθοδήγηση.

Για ένα σημαντικό μέρος της Διατριβής ήταν απαραίτητη η χρήση παράλληλου υπολογιστικού συστήματος. Ευχαριστώ τον Καθηγητή Θεόδωρο Παπαθεοδώρου για την παραχώρηση πρόσβασης στο παράλληλο σύστημα του Πανεπιστημίου Πατρών.

Τέλος ευχαριστώ τα μέλη της οικογενείας μου για την ηθική υποστήριξη που μου παρείχαν σε όλη τη διάρκεια της προσπάθειας μου για την ολοκλήρωση των σπουδών μου.

Περίληψη

Η εξέλιξη της τεχνολογίας των υπολογιστικών συστημάτων και ιδιαίτερα στην κατηγορία των παράλληλων αρχιτεκτονικών, αναδείχθηκε σ' έναν από τους σημαντικότερους παράγοντες για την ανάπτυξη αποδοτικών εφαρμογών. Έτσι και ο υπολογισμός μεγάλης - κλίμακας αριθμητικών προβλημάτων σε παράλληλα περιβάλλοντα έχει γίνει ένα από τα κεντρικά θέματα της Αριθμητικής Ανάλυσης και των Επιστημονικών Υπολογισμών (Scientific Computing) γενικότερα. Μια ερευνητική περιοχή, στην οποία υπάρχει άμεση επίδραση, είναι αυτή των αριθμητικών μεθόδων επίλυσης Διαφορικών Εξισώσεων με Μερικές Παραγώγους. Έτσι μέθοδοι, όπως αυτή της Collocation βασισμένη στα Hermite bi-cubic πεπερασμένα στοιχεία, άρχισαν να έχουν πρακτικές εφαρμογές στις δυο και τρεις διαστάσεις, για Ελλειπτικά κυρίως Προβλήματα Συνοριακών Τιμών (ΠΣΤ). Επειδή, όμως οι κλασσικοί αλγόριθμοι επαναληπτικής επίλυσης των παραγόμενων αραιών και γενικών γραμμικών συστημάτων, αδυνατούν να εκμεταλλευτούν τις δυνατότητες των παράλληλων μηχανών, έχει δοθεί κύρια έμφαση στην παραλληλοποίηση των σειριακών αλγορίθμων, ώστε η απεικόνισή τους σε παράλληλες αρχιτεκτονικές να είναι βέλτιστα αποδοτική.

Στο πνεύμα όλων των παραπάνω ο κύριος στόχος της παρούσας διατριβής είναι, αρχικά σε σειριακό επίπεδο, η διερεύνηση της συμπεριφοράς επαναληπτικών μεθόδων για την επίλυση του γραμμικού συστήματος μεγάλης τάξης, όπως αυτό προκύπτει από τη διακριτοποίηση ελλειπτικών προβλημάτων συνοριακών τιμών κάνοντας χρήση της μεθόδου collocation, και στη συνέχεια η κατασκευή αποδοτικών αλγορίθμων για την εφαρμογή τους σε παράλληλες αρχιτεκτονικές.

Για την επίλυση των collocation γραμμικών συστημάτων έχουν χρησιμοποιηθεί στο παρελθόν κυρίως stationary επαναληπτικές μέθοδοι, για τις οποίες έχουμε παρουσιάσει και υλοποιήσει στο πρόσφατο παρελθόν παράλληλους αλγορίθμους για αρχιτεκτονικές βασισμένες στην τεχνολογία των transputers και για εικονικές παράλληλες μηχανές λογικού τύπου (PVM). Τα γραμμικά αυτά συστήματα έχουν προκύψει σύμφωνα με την block τριδιαγώνια αρίθμηση αγνώστων και εξισώσεων, όπως αυτή έχει προταθεί από τον Θ. Παπαθεοδώρου . Σε μια πιο πρόσφατη εργασία τους οι Lai, Χατζηδήμος, Χούστης και Rice παρουσίασαν μια νέα διαμέριση του block τριδιαγώνιου πίνακα, όπου η βέλτιστη SOR stationary επαναληπτική μέθοδος εμφάνιζε ταχύτερη σύγκλιση από τις πιο γνωστές stationary επαναληπτικές μεθόδους, αλλά και από την non-stationary Jacobi preconditioned GMRES(50). Η ίδια διαμέριση θα χρησιμοποιηθεί για όλες τις επαναληπτικές μεθόδους που θα μελετηθούν στην παρούσα διατριβή.

Στο πρώτο κεφάλαιο παρουσιάζονται συνοπτικά οι βασικότερες stationary και non-stationary επαναληπτικές μέθοδοι, οι οποίες είναι κατάλληλες για την επίλυση των collocation γραμμικών συστημάτων. Επίσης παρουσιάζονται οι κυριότερες τεχνικές preconditioning καθώς και τα πιο συνηθισμένα κριτήρια ελέγχου σύγκλισης των επαναληπτικών μεθόδων.

Στο δεύτερο κεφάλαιο παρουσιάζεται η αριθμητική μέθοδος επίλυσης ΠΣΤ Collocation βασισμένη στα Hermite bi-cubic πεπερασμένα στοιχεία για το Poisson πρόβλημα, το οποίο θα χρησιμοποιηθεί ως πρόβλημα μοντέλο. Ακόμα εμφανίζεται η μορφή του collocation γραμμικού συστήματος, σύμφωνα με την block τριδιαγώνια αρίθμηση αγνώστων και εξισώσεων.

Στο τρίτο κεφάλαιο γίνεται μελέτη της συμπεριφοράς των τριών non-stationary επαναληπτικών μεθόδων ORTHOMIN, GMRES(m) και BiCGSTAB με ταυτόχρονη διερεύνηση της αποδοτικότερης τεχνικής preconditioning ανάμεσα στις Jacobi, SSOR και SGS, οι οποίες δεν απαιτούν επιπλέον κόστος κατασκευής. Ειδικότερα για την GMRES(m) έγιναν πειραματικές μετρήσεις για διάφορες τιμές της παραμέτρου

επανεκκίνησης m. Όσο αφορά την μέθοδο αποθήκευσης του collocation πίνακα χρησιμοποιήθηκε η κλασσική μέθοδος αραιάς αποθήκευσης κατά στήλες και αυτή της block μορφής. Η διαφορά ανάμεσα στις δυο μορφές αποθήκευσης, όσο αφορά τις αριθμητικές πράξεις, είναι ότι στην πρώτη περίπτωση χρησιμοποιείται η ατελής παραγοντοποίηση για την επίλυση των preconditioned γραμμικών συστημάτων, γεγονός που έχει άμεση επίδραση στη συμπεριφορά σύγκλισης κάθε επαναληπτικής μεθόδου, όπως προέκυψε από τις πειραματικές μετρήσεις και επαληθεύτηκε από την ανάλυση των τιμών Ritz. Το μέρος αυτής της ερευνητικής διαδικασίας παρουσιάστηκε με την εργασία με τίτλο *"Non-stationary Iterative Schemes for the solution of Elliptic Collocation Systems"* στο διεθνές συνέδριο HERCMA'98 στην Αθήνα. Στο ίδιο κεφάλαιο επίσης παρουσιάζονται πειραματικές δοκιμές ανάμεσα στην αποδοτικότερη των non-stationary επαναληπτικών μεθόδων SSOR preconditioned BiCGSTAB και στην SOR, η συμπεριφορά των οποίων εμφανίζεται στην εργασία *"Stationary and Non-stationary preconditioned Iterative Schemes for the solution of Elliptic Collocation Systems"*, η οποία παρουσιάστηκε στο διεθνές συνέδριο 5th IMACS on Iterative methods in Scientific Computing στο Ηράκλειο το 2001. Θα πρέπει να σημειωθεί ότι η συμπεριφορά των non-stationary επαναληπτικών μεθόδων, όπως αυτή προκύπτει από τις πειραματικές δοκιμές, επαληθεύτηκε και από την ανάλυση των τιμών Ritz, η οποία περιλαμβάνεται στο παραπάνω τμήμα της έρευνας.

Το τέταρτο κεφάλαιο περιλαμβάνει την κατασκευή και υλοποίηση παράλληλων αλγορίθμων για τις αποδοτικότερες επαναληπτικές μεθόδους κάθε κατηγορίας, δηλαδή της SOR και της SSOR preconditioned BiCGSTAB, για αρχιτεκτονικές κοινής και διανεμημένης μνήμης. Επειδή η δομή του collocation πίνακα σε block τριδιαγώνια μορφή οδηγεί σε αυστηρά σειριακούς αλγορίθμους και για τις δυο προαναφερθήσες μεθόδους, έγινε εφαρμογή ενός μετασχηματισμού ομοιότητας στο γραμμικό σύστημα με αποτέλεσμα ο collocation πίνακας να μετασχηματιστεί στην 2-cyclic κανονική μορφή του. Ο παραπάνω μετασχηματισμός ομοιότητας ισοδυναμεί με την red-black επαναρίθμηση

αγνώστων και εξισώσεων, βασισμένη πάντα στην ίδια διαμέρηση του collocation πίνακα. Στη συνέχεια εφαρμόστηκε ένας επιπλέον μετασχηματισμός στο red-black collocation γραμμικό σύστημα μέσω του οποίου διπλασιάστηκε ο παραλληλισμός του. Η νέα ανάλυση των τιμών Ritz έδειξε ότι η SSOR preconditioned BiCGSTAB εξακολουθεί να θεωρείται ως την αποδοτικότερη των non-stationary επαναληπτικών μεθόδων και για το red-black collocation γραμμικό σύστημα. Ακολουθεί η κατασκευή του παράλληλου αλγορίθμου της SOR για αρχιτεκτονικές κοινής μνήμης και η υλοποίησή του στο υπολογιστικό σύστημα SGI Origin 2000. Ο παράλληλος αλγόριθμος της SOR για αρχιτεκτονικές διανεμημένης μνήμης καθώς και η υλοποίησή του στο υπολογιστικό σύστημα Parsytec CC2, που παρουσιάζονται στη συνέχεια, είναι το αντικείμενο της εργασίας μας *"Iterative solution of Elliptic Collocation Systems on a Cognitive Parallel Computer"*, η οποία έχει γίνει δεκτή προς δημοσίευση στο διεθνές περιοδικό Advances in Partial Differential Equations. Ανάλογοι παράλληλοι αλγόριθμοι με τις υλοποιήσεις τους ακολουθούν μέχρι το τέλος του τετάρτου κεφαλαίου και για την SSOR preconditioned BiCGSTAB. Έτσι τόσο η κατασκευή του παράλληλου αλγορίθμου της για αρχιτεκτονικές κοινής μνήμης, όσο και η υλοποίηση του στο υπολογιστικό σύστημα SGI Origin 2000 μπορεί να βρεθεί στην εργασία μας με τίτλο *" BiCGSTAB for Collocation systems on shared memory parallel architectures"*, η οποία θα παρουσιαστεί στο διεθνές συνέδριο Numerical Algorithms 2001 στο Marrakesh-Μαρόκο και θα δημοσιευτεί σε ειδική έκδοση του ομόνυμου διεθνούς επιστημονικού περιοδικού. Επίσης η κατασκευή και η υλοποίηση του αντίστοιχου παράλληλου αλγορίθμου για αρχιτεκτονικές διανεμημένης μνήμης θα παρουσιαστεί στο συνέδριο ENUMATH 2001 (European Conference on Numerical Mathematics and Advanced Applications) με την εργασία μας που έχει τίτλο *" BiCGSTAB for Collocation systems on distributed memory parallel architectures"*.

Συμπερασματικά αξίζει να σημειωθεί ότι, σε σειριακό επίπεδο η μελέτη των non-stationary επαναληπτικών μεθόδων για την επίλυση των collocation γραμμικών

συστημάτων έδειξε ότι θα πρέπει να προτιμάται η SSOR ή SGS preconditioned BiCGSTAB. Για την παράλληλη επίλυση των collocation γραμμικών συστημάτων προτείνονται αποδοτικοί παράλληλοι αλγόριθμοι για αρχιτεκτονικές κοινής και διανεμημένης μνήμης για τις δυο επαναληπτικές μεθόδους SOR και SSOR preconditioned BiCGSTAB, όπως προκύπτει από τις πειραματικές μετρήσεις απόδοσης.

ΠΕΡΙΕΧΟΜΕΝΑ

Σελ.

Σελ.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Σελ.

Σελ.

Σελ.

Σελ.

Σελ.

1 ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΓΕΝΙΚΩΝ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

1.1 Εισαγωγή

Για την επίλυση ενός γραμμικού συστήματος εξισώσεων $A\boldsymbol{x} = \boldsymbol{b}$, όπου $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$ και με αντιστρέψιμο πίνακα $A \in \mathbb{R}^{n,n}$, η γνωστότερη αριθμητική μέθοδος είναι η απαλοιφή Gauss. Γενικά η μέθοδος αυτή χρειάζεται $O(n^2)$ θέσεις αποθήκευσης και $O(n^3)$ αριθμητικές πράξεις. Όμως οι πίνακες των συντελεστών των αγνώστων που εμφανίζονται στην πράξη, όπως για παράδειγμα κατά την αριθμητική επίλυση των Μερικών Διαφορικών Εξισώσεων με κάποια από τις μεθόδους των Πεπερασμένων Διαφορών ή των Στοιχείων, έχουν κάποιες συγκεκριμένες ιδιότητες τις οποίες η απαλοιφή Gauss μπορεί να τις εκμεταλλευτεί μόνο εν μέρη. Έτσι για αραιούς πίνακες, στους οποίους τα περισσότερα στοιχεία είναι μηδενικά, μπορεί να βρεθεί μια περιοχή-ζώνη κάποιου εύρους $2m$ και να περιλάβει όλα τα μη μηδενικά στοιχεία. Αυτό έχει ως αποτέλεσμα ο χώρος αποθήκευσης του πίνακα να μειωθεί σε $O(mn)$ και οι πράξεις σε $O(m^2)$. Όμως τα μηδενικά στοιχεία εντός της ζώνης δεν είναι δυνατό να αγνοηθούν, γιατί κατά την παραγοντοποίηση υπάρχουν γεμίσματα, δηλαδή κάποια από αυτά γίνονται μη μηδενικά.

Εκτός από το γεγονός να είναι αραιός ο πίνακας των συντελεστών συχνά έχει και άλλες ιδιότητες, που μπορούμε να τις εκμεταλλευτούμε με τον πολλαπλασιασμό του πίνακα με ένα διάνυσμα. Σ' αυτή τη περίπτωση το σύνολο των πράξεων μπορεί να μειωθεί σε $O(n)$, αντί για $O(n^2)$ που απαιτούνται στην περίπτωση πυκνού πίνακα,

ενώ χρειάζεται να αποθηκευτούν μόνο τα μη μηδενικά στοιχεία του. Άρα, για την επίλυση του γραμμικού συστήματος, είναι προτιμότερο να χρησιμοποιηθεί μια άλλη τεχνική, η οποία κάνοντας χρήση κυρίως του πολλαπλασιασμού του πίνακα των συντελεστών των αγνώστων με κάποια διανύσματα, θα μπορεί να παράξει μια ικανοποιητική προσέγγιση της ακριβούς λύσης.

Οι *Επαναληπτικές Μέθοδοι* είναι τέτοιες τεχνικές, οι οποίες δεν αλλάζουν τον πίνακα των συντελεστών των αγνώστων, όπως η απαλοιφή Gauss, αλλά ξεκινώντας από μια αρχική προσέγγιση $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$ (συνήθως τυχαία), κατασκευάζουν μια ακολουθία διαδοχικών προσεγγίσεων $\boldsymbol{x}^{(m)}$, $m = 1, 2, 3, ...$, η οποία υπό ορισμένες προϋποθέσεις συγκλίνει στη λύση $\boldsymbol{x}$, όταν $m \to +\infty$. Στη διατριβή αυτή θα αναφερθούν δυο βασικές κατηγορίες των επαναληπτικών μεθόδων, οι $Stationary$ και οι $Non\text{-}Stationary$ μέθοδοι, οι οποίες είναι και οι πιο συνηθισμένες στην αριθμητική επίλυση των αραιών γραμμικών συστημάτων που προκύπτουν από τις σύγχρονες μεθόδους αριθμητικής επίλυσης διαφορικών εξισώσεων.

Οι Stationary επαναληπτικές μέθοδοι είναι χρονικά παλαιότερες, περισσότερο απλές στην κατανόηση, αλλά και στην εφαρμογή από αυτές της άλλης ομάδας. Έτσι, οι Non-Stationary μέθοδοι έχουν αναπτυχθεί στο πρόσφατο παρελθόν και η ανάλυσή τους είναι πιο δύσκολη. Αυτό οφείλεται στο γεγονός, ότι η βασική ιδέα της προσέγγισης της λύσης στηρίζεται στις ακολουθίες ορθογώνιων πολυωνύμων ή και ορθογώνιων διανυσμάτων. Ο ρυθμός σύγκλισης μιας non-stationary μεθόδου κατά κύριο λόγο εξαρτάται από το φάσμα του πίνακα των συντελεστών των αγνώστων κι έτσι μια επαναληπτική μέθοδος συνήθως εμπλέκει κι έναν δεύτερο πίνακα, ο οποίος μετασχηματίζει τον πίνακα των συντελεστών σε κάποιον άλλο με καλλίτερο φάσμα. Ο πίνακας που επιτυγχάνει τον μετασχηματισμό αυτό λέγεται πίνακας $Preconditioner$. Η χρήση ενός κατάλληλου τέτοιου πίνακα μπορεί να βελτιώσει τη σύγκλιση της μεθόδου σε τέτοιο βαθμό, ώστε το επιπλέον υπολογιστικό κόστος της δημιουργίας και εφαρμογής του μετασχηματισμού στο γραμμικό σύστημα να

θεωρηθεί αμελητέο σε σχέση με το συνολικό υπολογιστικό όφελος. Επίσης, αρκετές επαναληπτικές μέθοδοι, είναι πιθανόν να μην καταφέρουν να επιτύχουν σύγκλιση χωρίς την εφαρμογή ενός τέτοιου κατάλληλου μετασχηματισμού.

Στη συνέχεια γίνεται αναφορά στις κυριότερες γενικές ιδιότητες των επαναληπτικών μεθόδων. Ακολουθεί σύντομη παρουσίαση των βασικότερων Stationary και Non-Stationary μεθόδων, ενώ υπάρχει μια ξεχωριστή ενότητα που αναφέρεται στις τεχνικές Preconditioning. Τέλος παρουσιάζονται τα κυριότερα κριτήρια τερματισμού και ελέγχου σύγκλισης μιας επαναληπτικής μεθόδου.

1.2 Βασικές Ιδιότητες των Επαναληπτικών Μεθόδων

Στην ενότητα αυτή παρουσιάζονται οι βασικότερες ιδιότητες των επαναληπτικών μεθόδων, οι οποίες αφορούν γενικά γραμμικά συστήματα με πραγματικούς συντελεστές.

Ιδιότητα 1.1

Για το γραμμικό σύστημα

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

όπου ο πίνακας $A \in \mathbb{R}^{n,n}$ και $\boldsymbol{b} \in \mathbb{R}^n$, το διάνυσμα της λύσης $\boldsymbol{x} \in \mathbb{R}^n$ υπάρχει και είναι μοναδικό **αν και μόνο αν** ο πίνακας A είναι αντιστρέψιμος.

Ορισμός 1.1

Για την αριθμητική επίλυση του παραπάνω γραμμικού συστήματος θεωρούμε τη γενική επαναληπτική μέθοδο της μορφής

$$\boldsymbol{x}^{(m+1)} = \mathcal{L}\,\boldsymbol{x}^{(m)} + \boldsymbol{k} \ , \ m = 0, 1, \ldots \tag{2}$$

όπου $\mathcal{L} \in \mathbb{R}^{n,n}$ και ονομάζεται **επαναληπτικός πίνακας** της μεθόδου και $\boldsymbol{k} \in \mathbb{R}^n$ είναι το διάνυσμα των γνωστών τιμών. Για την περίπτωση κατά την οποία ο πίνακας $\mathcal{L}$, αλλά και το διάνυσμα $\boldsymbol{k}$ είναι ανεξάρτητα του βήματος m, τότε η επαναληπτική μέθοδος είναι **Stationary**.

Ορισμός 1.2

Μια επαναληπτική μέθοδος είναι *πλήρως συμβατή - completely consistent* αν η λύση που παράγει είναι η μοναδική λύση του συστήματος

$$(I - \mathcal{L})\, \boldsymbol{x} \;=\; \boldsymbol{k} \tag{3}$$

και κοινή με αυτή του αρχικού γραμμικού συστήματος (1).

Ιδιότητα 1.2

Για τη λύση ενός γενικού γραμμικού συστήματος με μια πλήρους συμβατής επαναληπτικής μεθόδου τα παρακάτω είναι ισοδύναμα :

1. Η επαναληπτική μέθοδος $\boldsymbol{x}^{(m+1)} = \mathcal{L}\,\boldsymbol{x}^{(m)} + \boldsymbol{k}$, συγκλίνει, δηλαδή $\forall\ \boldsymbol{x}^{(0)} \in \mathbb{R}^n$ έχουμε $\boldsymbol{x}^{(m)} \to \boldsymbol{x}$, με $m \to +\infty$.

2. $\rho(\mathcal{L}) \;<\; 1$, όπου $\rho(\mathcal{L})$ είναι η μέγιστη κατά απόλυτη τιμή των ιδιοτιμών του πίνακα $\mathcal{L}$ και ονομάζεται φασματική ακτίνα του.

3. Υπάρχει φυσική νόρμα πινάκων $\|.\|$ τέτοια ώστε $\|\mathcal{L}\| \;<\; 1$.

4. $\mathcal{L}^m \to 0$, όταν $m \to +\infty$.

Η απόδειξη της παραπάνω ιδιότητας μπορεί να βρεθεί σε όλα τα βασικά βιβλία επαναληπτικών μεθόδων, πχ [1].

Ορισμός 1.3

Μπορούμε να θεωρήσουμε την παρακάτω *διάσπαση* του πίνακα των συντελεστών των αγνώστων

$$A \;=\; M \;-\; N \;\;, \tag{4}$$

οπότε

$$\mathcal{L} = I - M^{-1}A = M^{-1}N \quad , \quad \boldsymbol{k} = \mathrm{M}^{-1}\boldsymbol{b} \tag{5}$$

για κάποιο αντιστρέψιμο πίνακα $M \in \mathbb{R}^{n,n}$, ο οποίος καλείται **πίνακας διάσπασης**.

Ιδιότητα 1.3

Το γενικό βήμα μιας επαναληπτικής μεθόδου έχει επίσης την εξής μορφή

$$\boldsymbol{x}^{(m+1)} = M^{-1} N \, \boldsymbol{x}^{(m)} + M^{-1} \boldsymbol{b} \ , \tag{6}$$

δηλαδή

$$M \, \boldsymbol{x}^{(m+1)} = N \, \boldsymbol{x}^{(m)} + \boldsymbol{b} \ , \tag{7}$$

και επειδή το γραμμικό σύστημα μπορεί να γραφεί στη μορφή $M\boldsymbol{x} = N\boldsymbol{x} + \boldsymbol{b}$, εξαιτίας της διάσπασης $A = M - N$, θα ισχύει

$$M \, (\boldsymbol{x}^{(m+1)} - \boldsymbol{x}) = N \, \boldsymbol{x}^{(m)} + \boldsymbol{b} - (N\boldsymbol{x} + \boldsymbol{b}) \ , \tag{8}$$

άρα,

$$\boldsymbol{x}^{(m+1)} - \boldsymbol{x} = \mathcal{L} \, (\boldsymbol{x}^{(m)} - \boldsymbol{x}) = \mathcal{L}^2 (\boldsymbol{x}^{(m-1)} - \boldsymbol{x}) = \ldots = \mathcal{L}^m (\boldsymbol{x}^{(0)} - \boldsymbol{x}) \ . \tag{9}$$

Έτσι γράφοντας την παραπάνω σχέση για $m = 1, 2, ...$ θα ισχύει $\forall \, m \geq 0$

$$\boldsymbol{x}^{(m)} - \boldsymbol{x} = \mathcal{L}^m \, (\boldsymbol{x}^{(0)} - \boldsymbol{x}) \ . \tag{10}$$

και συνεπώς, για κάθε διανυσματική νόρμα και την αντίστοιχη φυσική νόρμα πινάκων, θα ισχύει η εκτίμηση του σφάλματος $\| \, \boldsymbol{x}^{(m)} - \boldsymbol{x} \, \|$ στο m βήμα της επαναληπτικής διαδικασίας

$$\| \, \boldsymbol{x}^{(m)} - \boldsymbol{x} \, \| \leq \| \, \mathcal{L}^m \, \| \, \| \, \boldsymbol{x}^{(0)} - \boldsymbol{x} \, \| \leq \| \, \mathcal{L} \, \|^m \, \| \, \boldsymbol{x}^{(0)} - \boldsymbol{x} \, \| \ . \tag{11}$$

Ορισμός 1.4

Έστω $\mathcal{L}$ ο επαναληπτικός πίνακας μιας επαναληπτικής μεθόδου, τότε ορίζεται ως μέση τάξη σύγκλισης της μεθόδου η τιμή

$$R_m(\mathcal{L}) \equiv -\frac{ln \, \| \, \mathcal{L}^m \, \|}{m} \ ,$$

και αν $\rho(\mathcal{L}) < 1$, τότε

$$\lim_{m \to +\infty} (\| \, \mathcal{L}^m \, \|)^{\frac{1}{m}} = \rho(\mathcal{L})$$

και ορίζεται ως ασυμπτωτική τάξη σύγκλησης ή απλά τάξη σύγκλισης της μεθόδου η τιμή

$$R_\infty(\mathcal{L}) = \lim_{m \to +\infty} R_m(\mathcal{L}) \equiv -ln\rho(\mathcal{L}) \quad .$$

Ορισμός 1.5

Θεωρούμε τον πίνακα $A \in \mathbb{R}^{n,n}$ και το σύνολο των φυσικών αριθμών $\{n_i\}$, με $i = 1, \ldots, k$ για τους οποίους ισχύει ότι $n_1 + n_2 + \ldots + n_k = n$. Μπορούμε να ορίσουμε την $k \times k$ block διαμέριση του πίνακα A ως τη μορφή

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} \end{bmatrix} ,$$

με A_{ij} να είναι κάθε $n_i \times n_j$ υποπίνακας. Ειδικότερα αν $k = n$, δηλαδή $n_i = 1$ για όλα τα $i = 1, \ldots, k$ τότε η διαμέριση του πίνακα A ονομάζεται σημειακή (point) διαμέριση.

1.3 Stationary Επαναληπτικές Μέθοδοι

Για την αριθμητική επίλυση του γραμμικού συστήματος (1) με κάποια Stationary επαναληπτική μέθοδο της μορφής (2), αρχικά θεωρούμε την παρακάτω διάσπαση του πίνακα συντελεστών των αγνώστων του συστήματος

$$\mathrm{A} = D_A - L_A - U_A \tag{12}$$

όπου ο πίνακας D_A περιέχει τα διαγώνια στοιχεία του πίνακα A αντίστοιχα, ενώ οι πίνακες L_A και U_A είναι αυστηρά κάτω και άνω τριγωνικοί αντίστοιχα.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
for $k = 1, 2, 3, ...$
  for $i = 1, 2, ..., n$

    $\widetilde{\boldsymbol{x}_i} = 0$

    for $j = 1, 2, ..., i-1, i+1, ..., n$

      $\widetilde{\boldsymbol{x}_i} = \widetilde{\boldsymbol{x}_i} + a_{i,j}\, \boldsymbol{x}_j^{(k-1)}$
    end

    $\widetilde{\boldsymbol{x}_i} = (\boldsymbol{b}_i - \widetilde{\boldsymbol{x}_i})/a_{i,i}$
  end

  $\boldsymbol{x}^{(k)} = \widetilde{\boldsymbol{x}}$
  Έλεγχος σύγκλισης, αλλαγή βήματος αν απαιτείται
end

Εικόνα 1 : Αλγόριθμος της μεθόδου Jacobi.

Μέθοδος Jacobi

Η πιο απλή stationary επαναληπτική μέθοδος ονομάζεται Jacodi. Αν θεωρήσουμε τη διάσπαση του πίνακα A ως εξής

$$\mathrm{A} = D_A - (L_A + U_A) \quad ,$$

δηλαδή $\mathrm{M} = D_A$ και $N = L_A + U_A$, τότε ο γενικός τύπος της ορίζεται ως εξής

$$\begin{cases} \boldsymbol{x}^{(m+1)} &= \mathcal{L}_J\, \boldsymbol{x}^{(m)} + \boldsymbol{k}_J \;\; , \; m = 0, 1, \ldots \\ \mathcal{L}_J &= D_A^{-1}(L_A + U_A) \\ \boldsymbol{k}_J &= D_A^{-1}\boldsymbol{b} \end{cases} \quad , \tag{13}$$

Στην Εικόνα 1 εμφανίζεται ο αλγόριθμος που υλοποιεί την μέθοδο Jacobi. Για τον υπολογισμό της νέας προσεγγιστικής τιμής της λύσης ${\boldsymbol{x}_i}^{(m+1)}$ της i-στης συνιστώσας, η μέθοδος Jacobi χρησιμοποιεί αποκλειστικά την προσέγγιση του προηγουμένου βήματος ${\boldsymbol{x}_i}^{(m)}$ και δεν προϋποθέτει γνώση καμίας άλλης νέας τιμής ${\boldsymbol{x}_j}^{(m+1)}$ με $i \neq j$.

Έτσι η μέθοδος αυτή έχει την ιδιότητα της *παραλληλίας*, δηλαδή έχει την δυνατότητα του υπολογισμού της λύσης σε παράλληλα περιβάλλοντα.

Μέθοδος Gauss-Seidel

Για την περίπτωση κατά την οποία θεωρήσουμε την διάσπαση του πίνακα A ως

$$\mathrm{A} = (D_A - L_A) - U_A$$

δηλαδή $\mathrm{M} = D_A - L_A$ και $\mathrm{N} = U_A$ και στον υπολογισμό της λύσης $\boldsymbol{x}_i{}^{(m+1)}$ χρησιμοποιηθούν όποιες συνιστώσες έχουν ήδη υπολογιστεί από τη νέα αυτή προσέγγιση, τότε η παραγόμενη μέθοδος ονομάζεται Gauss-Seidel και έχει τη γενική μορφή

$$\left\{ \begin{array}{rcl} \boldsymbol{x}^{(m+1)} & = & \mathcal{L}_{GS}\, \boldsymbol{x}^{(m)} + \boldsymbol{k}_{GS} \ , \ m = 0, 1, \ldots \\ \mathcal{L}_{GS} & = & (D_A - L_A)^{-1} U_A \\ \boldsymbol{k}_{GS} & = & (D_A - L_A)^{-1} \boldsymbol{b} \end{array} \right. . \qquad (14)$$

Η μέθοδος αυτή είναι περισσότερο σειριακή από την Jacobi, όπως άλλωστε φαίνεται και από τον αλγόριθμο της που εμφανίζεται στην Εικόνα 2.

Μέθοδος SOR

Όταν η φασματική ακτίνα $\rho(\mathcal{L}_{\mathcal{GS}})$ του επαναληπτικού πίνακα της μεθόδου Gauss Seidel είναι κοντά στη μονάδα, η σύγκλιση είναι πάρα πολύ αργή. Για τέτοιες περιπτώσεις έχουν επινοηθεί τεχνικές **επιτάχυνσης** της σύγκλισης της επαναληπτικής μεθόδου.

Μια τέτοια μέθοδος, η λεγόμενη Successive Overrelaxation (SOR), προέκυψε από την εξής παρατήρηση :

- Έστω $\boldsymbol{x}^{(m)}$ μια γνωστή προσέγγιση της ακριβής λύσης $\boldsymbol{x}$ και έστω $\tilde{\boldsymbol{x}}^{(m+1)}$ η προσέγγιση της μεθόδου Gauss Seidel που κατασκευάζεται από την $\boldsymbol{x}^{(m)}$. Τότε η $\boldsymbol{x}^{(m+1)}$ που ορίζεται ως γραμμικός συνδυασμός $\boldsymbol{x}^{(m+1)} = \omega\, \tilde{\boldsymbol{x}}^{(m+1)} +$

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
for $k = 1, 2, 3, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k)}$
    end
    for $j = i+1, ..., n$
      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k-1)}$
    end
    $\boldsymbol{x}_i^{(k)} = (\boldsymbol{b}_i - \sigma)/a_{i,i}$
  end
  Έλεγχος σύγκλισης, αλλαγή βήματος αν απαιτείται
end

Εικόνα 2 : Αλγόριθμος της μεθόδου Gauss-Seidel.

$(1-\omega)\ \boldsymbol{x}^{(m)}$, έχει μικρότερο σφάλμα από την $\tilde{\boldsymbol{x}}^{(m+1)}$ κάτω από ορισμένες προϋποθέσεις και για ορισμένες τιμές της παραμέτρου ω.

Η γενική της μορφή της Successive Overrelaxation μεθόδου, η οποία βασίζεται στη διάσπαση του πίνακα A ως εξής με $\omega \neq 0$

$$\mathrm{A} = \frac{1}{\omega}(D_A - \omega L_A) - \frac{1}{\omega}[(1-\omega)D_A + \omega U_A]$$

είναι

$$\begin{cases} \boldsymbol{x}^{(m+1)} &= \mathcal{L}_s\, \boldsymbol{x}^{(m)} + \boldsymbol{k}_s\ ,\ m = 0, 1, \ldots \\ \mathcal{L}_s &= (D_A - \omega L_A)^{-1}[\omega U_A + (1-\omega)D_A] \\ \boldsymbol{k}_s &= \omega(D_A - \omega L_A)^{-1}\boldsymbol{b} \end{cases} . \qquad (15)$$

Στην Εικόνα 3 εμφανίζεται ο αλγόριθμος της μεθόδου SOR.

Θεώρημα Kahan

Η επαναληπτική μέθοδος SOR αποκλίνει $\forall\, \omega \in (-\infty, 0] \cup [2, +\infty)$.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
for $k = 1, 2, 3, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k)}$
    end
    for $j = i+1, ..., n$
      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k-1)}$
    end
    $\sigma = (\boldsymbol{b}_i - \sigma)/a_{i,i}$
    $\boldsymbol{x}_i^{(k)} = \boldsymbol{x}_i^{(k-1)} + \omega(\sigma - \boldsymbol{x}_i^{(k-1)})$
  end
  Έλεγχος σύγκλισης, αλλαγή βήματος αν απαιτείται
end

Εικόνα 3 : Αλγόριθμος της μεθόδου SOR.

Η απόδειξη μπορεί να βρεθεί στο [67]. Γενικά δεν είναι εφικτός ο υπολογισμός της βέλτιστης τιμής του ω, αλλά και όταν κάτι τέτοιο είναι δυνατό το υπολογιστικό κόστος είναι μεγάλο. Γι' αυτό πολύ συχνά επιλέγεται η τιμή $\omega = 2 - O(h)$, όπου h είναι το βήμα διακριτοποίησης μιας περιοχής Ω με κάποια αριθμητική μέθοδο.

Στη μαθηματική βιβλιοθήκη ITPACK [70] υπάρχουν υλοποιημένες προσαρμοστικές μέθοδοι προσέγγισης της βέλτιστης τιμής ω_{opt}, σύμφωνα με το ρυθμό σύγκλισης της μεθόδου SOR.

Μέθοδος AOR

Βασιζόμενοι στη παρακάτω διάσπαση του πίνακα A

$$\mathrm{A} = \frac{1}{\omega}(D_A - rL_A) - \frac{1}{\omega}[(1-\omega)D_A + (\omega - r)L_A + \omega U_A] \tag{16}$$

όπου $\omega \neq 0$, η επαναληπτική μέθοδος Accelarated OverRelaxation (AOR) ορίζεται ως εξής

$$\begin{cases} \boldsymbol{x}^{(m+1)} &= \mathcal{L}_{r,\omega}\boldsymbol{x}^{(m)} + \boldsymbol{c}_{r,\omega} \ , \ m = 0, 1, \ldots \\ \mathcal{L}_{r,\omega} &= (D_A - rL_A)^{-1}[(1-\omega)D_A + (\omega - r)L_A + \omega U_A] \quad , \\ \boldsymbol{c}_{r,\omega} &= \omega(D_A - rL_A)^{-1}\boldsymbol{b} \end{cases} \tag{17}$$

όπου $\boldsymbol{x}^{(0)}$ είναι κάποια αρχική προσέγγιση της λύσης, $\mathcal{L}_{r,\omega}$ είναι ο AOR επαναληπτικός πίνακας και οι παράμετροι r, ω αναφέρονται ως $acceleration$ και $overrelaxation$ παράγοντες αντίστοιχα.

Η διάσπαση του πίνακα A σύμφωνα με την (16) εγγυάται ότι η μέθοδος AOR είναι πλήρως συμβατή. Η συνθήκη σύγκλισης $\rho(\mathcal{L}_{r,\omega}) < 1$ καθώς και η ταχύτητα σύγκλισης της μεθόδου AOR δεν εξαρτάται μόνο από την επιλογή των παραμέτρων r και ω, αλλά και από την επιλογή των πινάκων διάσπασης D_A, L_A και U_A, δηλαδή από τη **διαμέριση** του πίνακα A.

Επιλέγοντας διάφορες τιμές για τις παραμέτρους ω και r, παράγονται αντίστοιχα κάποιες πιο ειδικές και πιο εύχρηστες stationary επαναληπτικές μέθοδοι, αφού ο προσδιορισμός των βέλτιστων τιμών δυο παραμέτρων της AOR δημιουργεί προβλήματα στην εφαρμογή της. Η μέθοδος SOR [2, 59, 112, 125] είναι μια ειδική περίπτωση της AOR για $\omega = r$, ενώ για $\omega = 1$ προκύπτει η Gauss Seidel.

Μέθοδοι SSOR και SGS

Συνδυάζοντας δυο βήματα, έτσι ώστε στο πρώτο να είναι αυτό της SOR, ενώ το δεύτερο να είναι το ανάστροφό του, προκύπτει η μέθοδος Symmetric Successive Overrelaxation (SSOR) με γενικό βήμα

$$\begin{cases} \boldsymbol{x}^{(m+1)} &= \mathcal{L}_{ss}\ \boldsymbol{x}^{(m)} + \boldsymbol{k}_{ss} \ , \ m = 0, 1, \ldots \\ \mathcal{L}_{ss} &= B_1\ B_2 \\ \boldsymbol{k}_{ss} &= \omega(2-\omega)(D_A - \omega U_A)^{-1} D_A (D_A - \omega L_A)^{-1}\boldsymbol{b} \quad . \\ B_1 &= (D_A - \omega U_A)^{-1}[\omega L_A + (1-\omega)D_A] \\ B_2 &= (D_A - \omega L_A)^{-1}[\omega U_A + (1-\omega)D_A] \end{cases} \tag{18}$$

Εύκολα μπορεί κανείς να παρατηρήσει, ότι ο πίνακας B_2 είναι ο επαναληπτικός πίνακας της SOR, ενώ ο B_1 είναι ο ίδιος με αναστροφή θέσεων μεταξύ L_A και U_A. Ο αλγόριθμος υλοποίησης της μεθόδου εμφανίζεται στην Εικόνα 4.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
for $k = 1, 2, 3, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$

      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k-\frac{1}{2})}$
    end
    for $j = i+1, ..., n$

      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k-1)}$
    end
    $\sigma = (\boldsymbol{b}_i - \sigma)/a_{i,i}$

    $\boldsymbol{x}_i^{(k-\frac{1}{2})} = \boldsymbol{x}_i^{(k-1)} + \omega(\sigma - \boldsymbol{x}_i^{(k-1)})$
  end
  for $i = n, n-1, ..., 1$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$

      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k-\frac{1}{2})}$
    end
    for $j = i+1, ..., n$

      $\sigma = \sigma + a_{i,j}\, \boldsymbol{x}_j^{(k)}$
    end
    $\boldsymbol{x}_i^{(k)} = \boldsymbol{x}_i^{(k-\frac{1}{2})} + \omega(\sigma - \boldsymbol{x}_i^{(k-\frac{1}{2})})$

  Έλεγχος σύγκλισης, αλλαγή βήματος αν απαιτείται
end

Εικόνα 4 : Αλγόριθμος της μεθόδου SSOR.

Η μέθοδος αυτή είναι από τις πιο διαδεδομένες μεθόδους preconditioning, όπως άλλωστε και η παραγόμενη μέθοδος για $\omega = 1$, η οποία ονομάζεται Symmetric Gauss Seidel. Πλήρης ανάλυση των μεθόδων αυτών μπορεί να βρεθεί στα [112, 125, 59].

Σε όλους τους παραπάνω αλγορίθμους των επαναληπτικών μεθόδων έχει θεωρηθεί μια κατάλληλη διάσπαση του πίνακα A, η οποία έχει βασιστεί σε σημειακή (point) διαμέρισή του. Έτσι, αν θεωρήσουμε μια block διαμέριση του πίνακα των συντελεστών των αγνώστων του γραμμικού συστήματος, όλοι οι αλγόριθμοι θα πρέπει να προσαρμοστούν κατάλληλα στην αντίστοιχη διάσπαση που θα προκύψει από αυτή τη block διαμέριση. Στα επόμενα κεφάλαια όλες οι επαναληπτικές μέθοδοι που θα χρησιμοποιηθούν θα βασίζονται πάντοτε σε κάποια block διαμέριση του πίνακα, η οποία θα παρουσιάζεται κάθε φορά μαζί με την ανάλογη διάσπασή του.

1.4 Non-Stationary Επαναληπτικές Μέθοδοι

Στις Stationary επαναληπτικές μέθοδοι οι υπολογισμοί εμπλέκουν πληροφορίες οι οποίες είναι σταθερές σε κάθε επαναληπτικό βήμα. Αντίθετα στις Non-Stationary αυτές μεταβάλλονται και επιπλέον οι διάφορες σταθερές που εμφανίζονται υπολογίζονται κυρίως με εσωτερικά γινόμενα υπολοίπων σφαλμάτων ή γενικότερα διανυσμάτων που προκύπτουν σε κάθε επαναληπτικό βήμα [31, 40, 47].

Μέθοδος Richardson

Η βασική ιδέα, όπως και στις stationary επαναληπτικές μέθοδες, είναι η διάσπαση του πίνακα των συντελεστών των αγνώστων A, ενός γενικού γραμμικού συστήματος (1). Έτσι θεωρώντας τη διάσπαση $A = I - (I - A)$ μπορούμε να παράξουμε εύκολα την πιο απλή επαναληπτική μέθοδος, η οποία ονομάζεται Richardson [48], ως εξής:

$$\boldsymbol{x}^{(m+1)} = (I - A)\,\boldsymbol{x}^{(m)} + \boldsymbol{b} = \boldsymbol{x}^{(m)} + \boldsymbol{r}^{(m)}\ ,\ m = 0, 1, \ldots\ , \tag{19}$$

όπου

$$\boldsymbol{r}^{(m)} \equiv \boldsymbol{b} - A\ \boldsymbol{x}^{(m)}\ ,\ m = 0, 1, \ldots\ . \tag{20}$$

Πολλαπλασιάζοντας με $-A$ και προσθέτοντας το $\boldsymbol{b}$ προκύπτει

$$\boldsymbol{b} \; - \; A \, \boldsymbol{x}^{(m+1)} \;\; = \;\; \boldsymbol{b} \; - \; A \, \boldsymbol{x}^{(m)} \; - \; A \, \boldsymbol{r}^{(m)} \tag{21}$$

ή

$$\boldsymbol{r}^{(m+1)} \;\; = \;\; (I \; - \; A) \, \boldsymbol{r}^{(m)} \;\; = \;\; (I \; - \; A)^m \, \boldsymbol{r}^{(0)} \;\; = \;\; P_m(A) \, \boldsymbol{r}^{(0)} \quad . \tag{22}$$

Από την τελευταία ισότητα προκύπτει ότι θα έχουμε γρηγορότερη σύγκλιση όσο μικρότερη της μονάδας είναι η νόρμα $\| \; I \; - \; A \; \|_2$. Καταφέραμε λοιπόν να εκφράσουμε το υπόλοιπο σφάλματος $\boldsymbol{r}^{(m+1)}$ στο τυχαίο επαναληπτικό βήμα m στη μορφή ενός πολυωνύμου του A επί το αρχικό υπόλοιπο $\boldsymbol{r}^{(0)}$. Αυτή η δυνατότητα είναι χαρακτηριστική των non-stationary μεθόδων και το πολυώνυμο $P_i(A)$ είναι αυτό που προσδίδει τις χαρακτηριστικές ιδιότητες της μεθόδου, όπως η ταχύτητα σύγκλισης.

Μια πιο γενική μορφή της μεθόδου Richardson είναι η

$$\boldsymbol{x}^{(m)} \;\; = \;\; \boldsymbol{x}^{(m-1)} \; + \; \alpha_m \boldsymbol{r}^{(m-1)} \;\; , \; m = 1, 2, \ldots \quad , \tag{23}$$

όπου τα α_m μπορούν να επιλεγούν, για παράδειγμα ώστε, να ελαχιστοποιούν τη $\| \; \boldsymbol{r}^{(m)} \; \|$. Έτσι θα ισχύει

$$\begin{array}{rcl} \boldsymbol{r}^{(m)} & = & \boldsymbol{b} \; - \; A \;\; \boldsymbol{x}^{(m)} \; = \; \boldsymbol{b} \; - \; A \;\; \boldsymbol{x}^{(m-1)} \; - \alpha_m \; A \, \boldsymbol{r}^{(m-1)} \; = \\ & = & (I - \alpha_m \; A) \, \boldsymbol{r}^{(m-1)} \end{array} \quad , \tag{24}$$

οπότε το υπόλοιπο έχει την πολυωνυμική έκφραση

$$\boldsymbol{r}^{(m)} \;\; = \;\; P_m(A) \, \boldsymbol{r}^{(0)} \qquad , \qquad P_m(A) \; = \; \prod_{i=1}^{m} (I - \alpha_i \; A) \quad . \tag{25}$$

Στην επαναληπτική μέθοδο Richardson η συγκεκριμένη επιλογή των παραμέτρων $\{\alpha_i\}$ οδήγησε στο πολυώνυμο της μορφής $P_m(t) \; = \; \prod_{i=1}^{m} (1 - \alpha_i \; t)$. Οι περισσότερες μέθοδοι επιλέγουν με κάποιο κριτήριο αυτές τις παραμέτρους, δηλαδή από μια συγκεκριμένη οικογένεια πολυωνύμων. Τα πολυώνυμα αυτά σχεδόν ποτέ δεν κατασκευάζονται στην πράξη, αλλά χρησιμοποιούνται μέσα από έμμεσους τύπους έκφρασης της ειδικής σχέσης του $\boldsymbol{r}^{(m)}$ και του $\boldsymbol{r}^{(0)}$, ενώ οι ιδιότητες της κάθε οικογένειας των πολυωνύμων

χρησιμοποιούνται στην ανάλυση σύγκλισης της μεθόδου. Οι πιο σημαντικές μέθοδοι, όπως η οικογένεια των μεθόδων GMRES, επιλέγουν τις παραμέτρους αυτές έμμεσα από κριτήρια ελαχιστοποίησης ή προβολής.

Χωρίς περιορισμό της γενικότητας μπορούμε να θεωρήσουμε ότι $\alpha_i = 1$. Στη μέθοδο Richardson θα ισχύει

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{r}^{(0)} + \boldsymbol{r}^{(1)} + \boldsymbol{r}^{(2)} + \ldots + \boldsymbol{r}^{(m)} = \sum_{i=0}^{m} (I - A)^i \, \boldsymbol{r}^{(0)} \quad , \tag{26}$$

$$\boldsymbol{x}^{(m+1)} \in \{\boldsymbol{r}^{(0)}, \, A \, \boldsymbol{r}^{(0)}, \ldots, A^m \, \boldsymbol{r}^{(0)}\} \equiv \mathcal{K}^{m+1}(A; \boldsymbol{r}^{(0)}) \quad . \tag{27}$$

Ο υπόχωρος $\mathcal{K}^m(A; \boldsymbol{r}^{(0)})$ ονομάζεται υπόχωρος Krylov διάστασης m, ο οποίος παράγεται από τον πίνακα A και το αρχικό διάνυσμα υπολοίπου $\boldsymbol{r}^{(0)}$. Μια non-stationary μέθοδος, όπως η Richardson παράγει προσεγγίσεις της λύσης του γραμμικού συστήματος στον υπόχωρο αυτό με αυξανόμενη διάσταση. Οι επαναληπτικοί παράμετροι μπορούν να επιλεγούν έτσι ώστε να παράγεται μια άλλη προσέγγιση της λύσης κάθε φορά στον Krylov υπόχωρο, δηλαδή διαφορετικές πολυωνυμικές εκφράσεις του σφάλματος και του υπολοίπου. Είναι εύλογο το ερώτημα της βέλτιστης προσέγγισης της λύσης του γραμμικού συστήματος στον Krylov υπόχωρο. Μαθηματικά ο όρος **βέλτιστη** μπορεί να ερμηνευτεί ως την ελαχιστοποίηση της νόρμας του υπολοίπου ή το διάνυσμα του υπολοίπου να είναι ορθογώνιο στον Krylov υπόχωρο.

Προς την κατεύθυνση της εύρεσης καλλίτερης προσέγγισης της λύσης του γραμμικού συστήματος μέσα σε ένα Krylov υπόχωρο, χρειάζεται μια κατάλληλη βάση του υποχώρου. Η βάση $\boldsymbol{r}^{(0)}, \, A \, \boldsymbol{r}^{(0)}, \ldots, A^{m-1} \, \boldsymbol{r}^{(0)}$ δεν είναι πρακτική, γιατί τα διανύσματα $A^i \, \boldsymbol{r}^{(0)}$ τείνουν στην κατεύθυνση του κυρίαρχου ιδιοδιανύσματος, όσο αυξάνει η διάσταση του Krylov υποχώρου (σύμφωνα με τη μέθοδο των δυνάμεων) [102], άρα τα διανύσματα βάσης εξαρτούνται από την αριθμητική της πεπερασμένης ακρίβειας και επομένως το υπολογιστικό σύστημα υλοποίησης της μεθόδου. Αυτό μπορεί να διορθωθεί επιλέγοντας μια ορθοκανονική βάση του υποχώρου.

Ο Arnoldi [4] το 1951 πρότεινε τον υπολογισμό μιας τέτοιας βάσης. Θεωρώντας ότι $\{u_1, \ldots, u_j\}$ είναι ήδη μια ορθοκανονική βάση του υπόχωρου $K^j(A; r^{(0)})$, τότε η βάση αυτή μπορεί να επεκταθεί υπολογίζοντας $t = Au_j$ και κανονικοποιώντας το διάνυσμα t ως προς τα διανύσματα $u_1, \ldots, u_j$. Η διαδικασία θα ξεκινήσει από το $u_1 \equiv \frac{\boldsymbol{r}^{(0)}}{\|\boldsymbol{r}^{(0)}\|_2}$ και υπάρχουν αρκετές διαδικασίες που μπορούν να το επιτύχουν με πιο γνωστή αυτή της modified Gram-Schmidt [49]. Παρακάτω εμφανίζεται ο αλγόριθμος της μεθόδου με την οποία ένας πίνακας A μπορεί να εμφραστεί στην άνω Hessenberg μορφή του, με την βοήθεια του πίνακα με στήλες τα διανύσματα της ορθοκανονικής βάσης του Krylov υποχώρου K^m.

Αλγόριθμος Arnoldi με Modified Gram-Schmidt

Επιλογή του διανύσματος u_1 με νόρμα 1.

for $j = 1, 2, 3, ..., m$

Υπολογισμός του $w_1 = Au_1$

for $i = 1, 2, ..., j$

$h_{i,j} = (w_j, u_i)$

$w_j = w_j - h_{i,j}u_i$

end

$h_{j+1,j} = \| w_j \|$

if $h_{j+1,j} = 0$ **stop**

$u_{j+1} = \frac{w_j}{h_{j+1,j}}$

end

Σύμφωνα με την κατασκευή τους τα $u_j,\ j = 1, \ldots, m$ είναι ορθοκανονικά μεταξύ τους. Ακόμα παράγουν τον υπόχωρο K^m, γιατί κάθε διάνυσμα u_j είναι της μορφής $Q_{j-1}(A)u_1$, όπου Q_{j-1} είναι ένα πολυώνυμο $j-1$ βαθμού. Αυτό μπορεί να αποδειχτεί επαγωγικά κι έτσι για $j = 1$ ισχύει ότι $u_1 = Q_0(A)u_1$, με $Q_0(t) = 1$. Υποθέτοντας ότι ισχύει για την τιμή j θα έχουμε για την τιμή $j + 1$ ότι

$$h_{j+1,j}u_{j+1} = Au_j - \sum_{i=1} j h_{i,j}u_i = AQ_{j-1}(A)u_1 - \sum_{i=1} j h_{i,j}Q_{j-1}(A)u_1.$$

Οπότε το u_{j+1} είναι της μορφής $Q_j(A)u_1$ με Q_j πολυώνυμο βαθμού j. Έστω τώρα ότι V_j είναι ο πίνακας με στήλες τα $\{u_1, \ldots, u_j\}$ τότε

$$A\, V_{m-1} \;=\; V_m\; H_{m,m-1} \quad . \tag{28}$$

Ο $m \times (m-1)$ πίνακας $H_{m,m-1}$ είναι άνω Hessenberg και τα στοιχεία του $h_{i,j}$ παράγονται από τη διαδικασία Arnoldi. Όπως είναι φανερό η ορθογωνιοποίηση αυτή αυξάνει το υπολογιστικό κόστος με την αυξηση της διάστασης του υποχώρου, αφού για τον υπολογισμό του κάθε $h_{i,j}$ απαιτείται ένα εσωτερικό γινόμενο διανυσμάτων και μια ανανέωση της τιμής ενός διανύσματος.

Η προσεγγιστική λύση $\boldsymbol{x}^{(m)} \in \mathcal{K}^m(A; \boldsymbol{r}^{(0)})$ μπορεί να γραφεί στη μορφή

$$\boldsymbol{x}^{(m)} \;=\; V_m\; \boldsymbol{y} \quad , \tag{29}$$

όπου $\boldsymbol{y} \in \mathbb{R}^m$. Για την κατασκευή μιας τέτοιας λύσης με κάποια από τις non-stationary μεθόδους έχουν παρατηρηθεί τέσσερις διαφορετικές ομάδες προσεγγίσεων ανάλογα με το είδος προβολής στον Krylov υπόχωρο ως εξής :

1. Η προσέγγιση Ritz-Galerkin, όπου $\boldsymbol{b} - A\,\boldsymbol{x}^{(m)} \;\perp\; \mathcal{K}^m(A; \boldsymbol{r}^{(0)})$.

2. Η προσέγγιση ελαχιστοποίησης υπολοίπου (minimum residual), όπου θα πρέπει το $\| \boldsymbol{b} - A\,\boldsymbol{x}^{(m)} \|_2$ να ελαχιστοποιείται στον $\mathcal{K}^m(A; \boldsymbol{r}^{(0)})$.

3. Η προσέγγιση Petrov-Galerkin, όπου το $\boldsymbol{b} - A\,\boldsymbol{x}^{(m)}$ είναι ορθογώνιο σ' ένα άλλο κατάλληλο υπόχωρο διάστασης m.

4. Η προσέγγιση ελαχιστοποίησης σφάλματος (minimum error), όπου η νόρμα $\| \boldsymbol{x} - \boldsymbol{x}^{(m)} \|_2$ θα πρέπει να είναι ελάχιστη στον $\mathrm{A}^T\mathcal{K}^m(A^T; \boldsymbol{r}^{(0)})$.

Σύμφωνα με την προσέγγιση Ritz-Galerkin παράγονται οι βασικές μέθοδοι της οικογένειας Conjugate Gradients και Lanczos. Αν θεωρήσουμε χωρίς περιορισμό της γενικότητας ότι $\boldsymbol{x}^{(0)} = 0$, θα ισχύει ότι $\boldsymbol{r}^{(0)} = \boldsymbol{b}$. Η συνθήκη Ritz-Galerkin $\boldsymbol{r}^{(m)} \;\perp\; \mathcal{K}^m(A; \boldsymbol{r}^{(0)})$ μπορεί να εκφρασθεί ισοδύναμα

$$V_m^T\, (\boldsymbol{b} - A\boldsymbol{x}^{(m)}) \;=\; 0 \quad . \tag{30}$$

Επειδή $\boldsymbol{b} = \boldsymbol{r}^{(0)} = \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{u}_1$, θα ισχύει ότι $V_m^T \boldsymbol{b} = \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}_1$, με $\boldsymbol{e}_1 \in \mathbb{R}^m$ το μοναδιαίο διάνυσμα. Από τη σχέση $\boldsymbol{x}^{(m)} = V_m \, \boldsymbol{y}$ θα έχουμε

$$V_m^T \, A \, V_m \, \boldsymbol{y} = \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}_1 \quad . \tag{31}$$

Το παραπάνω σύστημα αποτελεί τη προβολή του αρχικού γραμμικού συστήματος, του οποίου ζητείται μια προσέγγιση της πραγματικής του λύσης, στον υπόχωρο Krylov $\mathcal{K}^m(A; \boldsymbol{r}^{(0)})$. Έτσι χρειάζεται η κατασκευή του $m \times m$ πίνακα $V_m^T \, A \, V_m$, ο οποίος όμως έχει ήδη κατασκευαστεί από τη διαδικασία ορθογωνιοποίησης

$$V_m^T \, A \, V_m = H_{m,m} \quad , \tag{32}$$

ώστε η $\boldsymbol{x}^{(m)}$ για την οποία ισχύει $\boldsymbol{r}^{(m)} \perp \mathcal{K}^m(A; \boldsymbol{r}^{(0)})$ μπορεί εύκολα να υπολογιστεί λύνοντας το $H_{m,m} \boldsymbol{y} = \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}^1$ και στη συνέχεια εκφράζοντας το ως $\boldsymbol{x}^{(m)} = V_m \boldsymbol{y}$.

Μέθοδος Conjugate Gradient for Normal Equations - CGNE

Για την περίπτωση συμμετρικού πίνακα A, ο πίνακας $H_{m,m}$ γίνεται τριδιαγώνιος και παράγεται η κλασσική μέθοδος Lanczos [71]. Αν επιπλέον ο πίνακας είναι και θετικά διευθετιμένος, τότε η παραγόμενη μέθοδος ονομάζεται Conjugate Gradient [60, 5, 87, 109] . Για γενικούς πίνακες υπάρχει η δυνατότητα εφαρμογής της θεωρίας [27, 55, 57, 66, 69, 99, 101, 123, 124, 125, 115], αφού εφαρμοστεί ο παρακάτω μετασχηματισμός στο γραμμικό σύστημα

$$\mathrm{A}^T \mathrm{A} \boldsymbol{x} = \mathrm{A}^T \boldsymbol{b} \tag{33}$$

για να εξασφαλιστεί η νόρμα με την οποία θα ελαχιστοποιηθεί το σφάλμα, αλλά και η δημιουργία της βάσης του υπόχωρου Krylov. Η αντίστοιχη μέθοδος για γενικούς πίνακες ονομάζεται CGNE (Conjugate Gradient for Normal Equations) και έχει μικρή πρακτική αξία, γιατί έχει μεγάλο υπολογιστικό κόστος και αργή σύγκλιση σε σχέση με άλλες Krylov μεθόδους. Αυτό συμβαίνει, γιατί χρειάζονται δυο πολλαπλασιασμοί πίνακα με διάνυσμα σε κάθε βήμα, με τον A και τον A^T, ενώ επιπλέον επειδή

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$s^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

$r^{(0)} = A^T s^{(0)}$

$p^{(0)} = r^{(0)}$

$\rho_0 = (r^{(0)})^T r^{(0)}$

$p^{(-1)} = 0$

$\beta_{-1} = 0$

for $i = 0, 1, 2, ...$

$p^{(i)} = r^{(i)} + \beta_{i-1}\, p^{(i-1)}$

$w^{(i)} = A\, p^{(i)}$

$\alpha_i = \frac{\rho_i}{(w^{(i)})^T w^{(i)}}$

$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \alpha_i\, p^{(i)}$

$s^{(i+1)} = s^{(i)} - \alpha_i\, w^{(i)}$

$r^{(i+1)} = A^T\, s^{(i+1)}$

Έλεγχος σύγκλισης και

έξοδος για ικανοποιητική τιμή του $\boldsymbol{x}^{(i+1)}$

$\rho_{i+1} = (r^{(i+1)})^T\, r^{(i+1)}$

$\beta_i = \frac{\rho_{i+1}}{\rho_i}$

end

Εικόνα 5a : Αλγόριθμος της μεθόδου CGNE.

$cond(A^TA) = cond^2(A)$ αναμένεται μεγάλος αριθμός βημάτων. Ο αλγόριθμος της CGNE εμφανίζεται στην Εικόνα 5a.

Μέθοδος ORTHOMIN

Το βασικό επαναληπτικό βήμα μιας non-stationary μεθόδου, όπως είναι γνωστό περιγράφεται από την σχέση

$$\boldsymbol{x}^{(m)} \;=\; \boldsymbol{x}^{(m-1)} + \alpha_m \boldsymbol{r}^{(m-1)} \;,\; m = 1, 2, \ldots \quad ,$$

όπου τα α_m μπορούν να επιλεγούν, για παράδειγμα ώστε, να ελαχιστοποιούν τη ευκλείδεια νόρμα του $\boldsymbol{r}^{(m)}$. Έτσι για

$$\alpha_m = \frac{(\boldsymbol{r}^{(m)}, A\boldsymbol{r}^{(m)})}{(A\boldsymbol{r}^{(m)}, A\boldsymbol{r}^{(m)})}$$

έχουμε ότι το $\boldsymbol{r}^{(m)}$ θα είναι ίσο με το $\boldsymbol{r}^{(m-1)}$ πλην τη προβολή του πάνω στο $A\boldsymbol{r}^{(m-1)}$. Έτσι η αναδρομική σχέση των υπολοίπων θα έχει τη μορφή

$$\boldsymbol{r}^{(m+1)} \;=\; \boldsymbol{r}^{(m)} - \alpha_m A\boldsymbol{r}^{(m)} \;,\; m = 1, 2, \ldots \quad .$$

Θα πρέπει να αναφερθεί ότι η μέθοδος ORTHOMIN είναι ισοδύναμη με την επόμενη μέθοδο GMRES, η οποία όμως έχει εξελιχθεί με την προσθήκη μιας παραμέτρου επανεκκίνησης κι έτσι έχει μειωθεί το κόστος αποθήκευσης ανά επαναληπτικό βήμα. Στην Εικόνα 5b εμφανίζεται ο αλγόριθμος της μεθόδου ORTHOMIN.

Μέθοδος Generalized Minimum Residual - GMRES

Αν προσεγγίσουμε τη λύση του γραμμικού συστήματος σύμφωνα με τη προσέγγιση της ελαχιστοποίησης του υπολοίπου σε ένα υπόχωρο Krylov, παράγεται η πιο δημοφιλής μέθοδος επίλυσης γενικών γραμμικών συστημάτων GMRES [104]. Όπως είναι γνωστό η δημιουργία ορθογώνιας βάσης του υπόχωρου Krylov καταλήγει στη σχέση

$$A\, V_m \;=\; V_{m+1}\, H_{m+1,m} \quad . \tag{34}$$

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

$p^{(0)} = r^{(0)}$

for $i = 0, 1, 2, ...$

$\alpha_i = \frac{(r^{(i)}, Ap^{(i)})}{(Ap^{(i)}, Ap^{(i)})}$

$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \alpha_i\, p^{(i)}$

$r^{(i+1)} = r^{(i)} - \alpha_i A\, p^{(i)}$

$\beta_i = -\frac{(Ar^{(i+1)}, Ap^{(i)})}{(Ap^{(i)}, Ap^{(i)})}$

$p^{(i+1)} = r^{(i+1)} + \beta_i\, p^{(i)}$

Έλεγχος σύγκλισης και

έξοδος για ικανοποιητική τιμή του $\boldsymbol{x}^{(i+1)}$

end

Εικόνα 5b : Αλγόριθμος της μεθόδου ORTHOMIN.

Αν θεωρήσουμε, χωρίς περιορισμό της γενικότητας ξανά ότι $\boldsymbol{x}^{(0)} = 0$, θα ισχύει ότι $\boldsymbol{r}^{(0)} = \boldsymbol{b}$. Αναζητούμε λύση $\boldsymbol{x}^{(m)} \in \mathcal{K}^m(A; \boldsymbol{r}^{(0)})$, δηλαδή $\boldsymbol{x}^{(m)} = V_m \, \boldsymbol{y}$, για την οποία ισχύει ότι $\| \boldsymbol{b} - \mathrm{A} \, \boldsymbol{x}^{(m)} \|_2$ γίνεται ελάχιστη.

Αυτό έχει ως αποτέλεσμα την παρακάτω έκφραση της ευκλείδιας νόρμας του υπολοίπου

$$\begin{aligned} \| \boldsymbol{b} - \mathrm{A} \, \boldsymbol{x}^{(m)} \|_2 &= \| \boldsymbol{b} - \mathrm{A} \, V_m \, \boldsymbol{y} \|_2 \\ &= \| \boldsymbol{b} - V_{m+1} \, H_{m+1,m} \, \boldsymbol{y} \|_2 \qquad (35) \\ &= \| V_{m+1} (\| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}_1 - H_{m+1,m} \, \boldsymbol{y}) \|_2 \quad . \end{aligned}$$

Συνδυάζοντας την παραπάνω σχέση με το γεγονός ότι V_{m+1} είναι ένας ορθοκανονικός μετασχηματισμός στον υπόχωρο Krylov $\mathcal{K}^{m+1}(A; \boldsymbol{r}^{(0)})$, προκύπτει

$$\| \boldsymbol{b} - \mathrm{A} \, \boldsymbol{x}^{(m)} \|_2 = \| \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}_1 - H_{m+1,m} \, \boldsymbol{y} \|_2 \quad , \qquad (36)$$

η οποία ελαχιστοποείται υπολογίζοντας την ελάχιστη ευκλείδια νόρμα της λύσης με τη μέθοδο των ελαχίστων τετραγώνων του συστήματος

$$H_{m+1,m} \, \boldsymbol{y} = \| \boldsymbol{r}^{(0)} \|_2 \, \boldsymbol{e}_1 \quad . \qquad (37)$$

Η μέθοδος, η οποία υλοποιεί τα παραπάνω ονομάζεται GMRES και το επιτυγχάνει εφαρμόζοντας Givens rotations, οι οποίοι απαλοίφουν τα υποδιαγώνια στοιχεία του πάνω Hessenberg πίνακα $H_{m+1,m}$.

Από τα κυριότερα μειονεκτήματα της μεθόδου GMRES [20, 51, 52, 86, 118, 119, 120, 121] είναι η απαίτηση αποθήκευσης όλων των διανυσμάτων υπολοίπων κάθε βήματος, καθώς επίσης και ότι η κατασκευή του προβαλόμενου συστήματος στον υπόχωρο είναι αρκετά πολύπλοκη. Η λύση σ' αυτό το πρόβλημα είναι η αποθήκευση των διανυσμάτων υπολοίπων μέχρι ένα προκαθορισμένο επαναληπτικό βήμα m και η επανεκίνησή της στη συνέχεια. Η παράμετρος αυτή αναφέρεται ως παράμετρος επανεκίνησης της μεθόδου GMRES(m) και η επιλογή της κατάλληλης τιμής δεν είναι εύκολη υπόθεση. Στο [64] υπάρχουν διάφορες διαπιστώσεις για την επιλογή της τιμής m από πειραματικές μετρήσεις. Προκύπτει άμεση εξάρτηση της τιμής της από

το αρχικό πρόβλημα. Συνήθως επιλέγοντας μικρές τιμές της παραμέτρου η σύγκλιση είναι αργή ή και μερικές φορές αδύνατη. Στην Εικόνα 6 εμφανίζεται ο αλγόριθμος της GMRES(m).

Μέθοδος BiConjugate Gradient - BiCG

Στη θεωρία προσέγγισης Petrov-Galerkin στηρίζεται μια άλλη μεγάλη ομάδα non-stationary επαναληπτικών μεθόδων, οι οποίες χρησιμοποιούνται για την επίλυση γενικών γραμμικών συστημάτων. Ανάμεσα στις μεθόδους είναι οι δημοφιλής QMR (Quasi-Minimal Residual), Bi-CG (BiConjugate Gradient) και οι υβριδικού τύπου CGS (Conjugate Gradient Squared) και Bi-CGSTAB (BiConjugate Gradient Stabilized). Σε αυτή τη κατηγορία μεθόδων κατασκευάζεται μια κατάλληλη ορθογώνια βάση του Krylov υποχώρου η οποία όμως όμως είναι ορθογώνια με κάποια άλλη βάση. Ας ανακαλέσουμε την βασική σχέση $A\,V_m \;=\; V_{m+1}\;H_{m+1,m}$, η οποία γράφεται ως

$$h_{m+1,m}\;u^{m+1} \;=\; A\;u^m \;-\; \sum_{j=1}^{m} h_{j,m}u_j \quad . \tag{38}$$

Επειδή ο πίνακας A δεν είναι συμμετρικός, δεν είναι δυνατός ο μετασχηματισμός του σε τριδιαγώνιο σε ένα υπόχωρο μικρότερης διάστασης μέσω ορθογώνιων προβολών, γιατί δεν είναι εφικτή η κατασκευή ορθογώνιας βάσης του Krylov υπόχωρου μέσω αναδρομικής σχέσης τριών μόνο όρων [36]. Έτσι δεν μπορούμε να χρησιμοποιήσουμε τον πίνακα V_m στη προβολή. Υποθέτουμε ότι υπάρχει ένας άλλος πίνακας W_m, ώστε $W_m^T V_m \;=\; D_m$, για D_m διαγώνιο πίνακα και να ισχύει ότι $W_m^T u^{m+1} \;=\; 0$. Τα στοιχεία $h_{m+1,m}$ μπορούν να επιλεγούν και είναι προτιμότερο να επιλεγούν ώστε $\|\; u^{m+1}\;\|_2 = \;1$. Τότε ισχύει

$$W_m^T A V_m \;=\; D_m H_{m,m} \tag{39}$$

κι έτσι θα πρέπει να βρεθεί ο κατάλληλος πίνακας W_m για τον οποίο ο $H_{m,m}$ είναι τριδιαγώνιος. Αυτό σημαίνει ότι θα πρέπει να είναι τριδιαγώνιος και ο $V_m^T A^T W_m$, δηλαδή να κατασκευαστούν τα w^m κάνοντας χρήση του A^T.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r \;=\; \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

for $j \;=\; 1, 2, ...$

$\beta \;=\; \| \; r \; \|_2 \;, \quad u^1 \;=\; \frac{r}{\beta} \;, \quad \hat{b} \;=\; \beta \; e^1$

for $i \;=\; 1, 2, ..., m$

$w \;=\; A \; u^i$

for $k \;=\; 1, 2, ..., i$

$h_{k,i} \;=\; (u^k)^* \; w \;, \quad w \;=\; w \;-\; h_{k,i} u^k$

$h_{i+1,i} \;=\; \| \; w \; \|_2 \;, \quad u^{i+1} \;=\; \frac{w}{h_{i+1,i}}$

for $k \;=\; 2, ..., i$

$r_{k-1,i} \;=\; c_{k-1} \; h_{k-1,i} \;+\; s_{k-1} \; h_{k,i}$

$r_{k,i} \;=\; -s_{k-1} \; h_{k-1,i} \;+\; c_{k-1} \; h_{k,i}$

$\delta \;=\; \sqrt{h_{i,i}^2 \;+\; h_{i+1,1}^2}$

$c_i \;=\; \frac{h_{i,i}}{\delta} \;, \quad s_i \;=\; \frac{h_{i+1,i}}{\delta}$

$r_{i,i} \;=\; c_i \; h_{i,i} \;+\; s_i \; h_{i+1,i}$

$\hat{b}_{i+1} \;=\; -s_i \; \hat{b}_i \;, \quad \hat{b}_i \;=\; c_i \;*\; \hat{b}_i$

$\rho \;=\; |\; \hat{b}_{i+1} \;| \quad (=\| \; \boldsymbol{b} - A\boldsymbol{x}^{((j-1)m+i)} \; \|_2)$

if ρ αρκετά μικρό **then**

$n_r \;=\; i \quad$ **goto** SOL

$n_r \;=\; m \;, \quad y_{n_r} \;=\; \frac{\hat{b}_{n_r}}{r_{n_r,n_r}}$

SOL : **for** $k \;=\; n_r - 1, ..., 1$

$y_k \;=\; \frac{\hat{b}_k \;-\; \sum_{i=k+1}^{n_r} r_{k,i} y_i}{r_{k,k}}$

$\boldsymbol{x} \;=\; \sum_{i=1}^{n_r} y_i u^i$

if ρ αρκετά μικρό **quit**

$r \;=\; \boldsymbol{b} - A\boldsymbol{x}$

Εικόνα 6 : Αλγόριθμος της μεθόδου GMRES(m).

Η διαδικασία αρχίζει με κάποιο $w_1 \neq 0$, ώστε $(w_1)^T u_1 \neq 0$. Στη συνέχεια κατασκευάζουμε με τη βοήθεια της (39) το w_2, ώστε να είναι ορθογώνιο με το w_1, δηλαδή $h_{1,1} = (w_1)^T A u_1 / ((w_1)^T u_1)$. Αφού $(w_1)^T A u_1 = (u_1)^T A^T w_1$ το w_2 μπορεί να κατασκευαστεί από το τύπο

$$h_{2,1} w_2 = A^T w_1 - h_{1,1} w_1 \tag{40}$$

και θα είναι ορθογώνιο με το u_1.

Συνεχίζοντας όμοια είναι δυνατή η κατασκευή μιας bi-orthogonal βάσης $\{u_j\}$ και $\{w_j\}$, κατασκευάζοντας τα καινούρια u_m ορθογώνια στα w_1 έως το w_{m-1} και στη συνέχεια κατασκευάζοντας τα w_m με τον ίδιο τρόπο, αλλά χρησιμοποιώντας τον A^T αντί του A. Έτσι θα ισχύει ότι $W_m^T A V_m = D_m H_{m,m}$ και επίσης $V_m A^T W_m = D_m H_{m,m}$, δηλαδή ο πίνακας $D_m H_{m,m}$ θα είναι συμμετρικός κι έτσι ο $H_{m,m}$ είναι τριδιαγώνιος, οπότε προκύπτει η ζητούμενη αναδρομική σχέση τριών όρων για τα u^j και τα w^j κι έτσι θα ισχύει

$$A\, V_m = V_{m+1}\, T_{m+1,m} \tag{41}$$

με T τριδιαγώνιο πίνακα. Κάνοντας χρήση του πίνακα $W_m = [w_1, w_2, \ldots, w_m]$ για την προβολή του συστήματος έχουμε

$$W_m^T \, (\boldsymbol{b} - A\, \boldsymbol{x}^{(m)}) = 0 \tag{42}$$

ή

$$W_m^T \, \mathrm{A} \, V_m \, \boldsymbol{y} - W_m^T \, \boldsymbol{b} = 0 \quad . \tag{43}$$

Οπότε το $\boldsymbol{y}$ θα ικανοποιεί τη σχέση

$$T_{m,m} \, \boldsymbol{y} = \| \, \boldsymbol{r}^{(0)} \, \|_2 \, \boldsymbol{e}_1 \quad , \tag{44}$$

και $\boldsymbol{x}^{(m)} = V_m \, \boldsymbol{y}$.

Η παραγόμενη μέθοδος ονομάζεται Bi-Lanczos [71, 97] και αν χρησιμοποιηθεί η τεχνική των σύντομων αναδρομικών σχέσεων αναφέρεται ως Bi-CG [12], η οποία

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}^{(0)}$ (συνήθως $\hat{r}^{(0)} = r^{(0)}$)

for $i = 1, 2, ...$

$\rho_{i-1} = (r^{(i-1)})^* \hat{r}^{(i-1)}$

if $i = 1$

$p^1 = r^{(0)}$

$\hat{p}^1 = \hat{r}^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}}$

$p^i = r^{(i-1)} + \beta_{i-1} p^{i-1}$

$\hat{p}^i = \hat{r}^{(i-1)} + \beta_{i-1} \hat{p}^{i-1}$

endif

$q^i = A\, p^i$

$\hat{q}^i = A^*\, \hat{p}^i$

$\alpha_i = \frac{\rho_{i-1}}{(\hat{p}^i)^* q^i}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i\, p^i$

$r^{(i)} = r^{(i-1)} - \alpha_i\, q^i$

$\hat{r}^{(i)} = \hat{r}^{(i-1)} - \alpha_i\, \hat{q}^i$

Έλεγχος σύγκλισης

end

Εικόνα 7 : Αλγόριθμος της μεθόδου Bi-CG.

μπορεί να εννοηθεί σαν την τροποποίηση της CG για μη συμμετρικά γραμμικά συστήματα. Στην Εικόνα 7 εμφανίζεται ο αλγόριθμος της Bi-CG.

Κατά τη δημιουργία της bi-orthogonal βάσης έχουμε υποθέσει ότι $(w_1)^T u_1 \neq 0$, διαφορετικά η κατασκευή της είναι αδύνατη. Έτσι η μέθοδος αποτυγχάνει αν για κάποια τιμή του i συμβεί $(w_i)^T u_i = 0$. Το φαινόμενο αυτό αναφέρεται στη βιβλιογραφία ως serious breakdown, ενώ όταν $(w_i)^T u_i \approx 0$ έχουμε near breakdown. Λύση σ' αυτό το πρόβλημα μπορεί να δωθεί με τις λεγόμενες look-ahead στρατηγικές, δηλαδή με τεχνικές ανίχνευσης και πρόβλεψης των προβληματικών καταστάσεων. Σύμφωνα με αυτές δημιουργείται μια ομάδα αποδεκτών διανυσμάτων βάσης, η οποία γίνεται στη συνέχεια ολόκληρη bi-orthogonal. Η μέθοδος αυτή αναλύεται λεπτομερώς στα [41, 42, 43, 94].

Μια άλλη πιο απλή μέθοδος είναι η επανεκίνηση της μεθόδου μόλις αποτύχει η κατασκευαστική διαδικασία. Έτσι όμως χάνεται όλο το τμήμα της βάσης που είχε μέχρι τότε δημιουργηθεί, με αποτέλεσμα την αργότερη σύγκλιση της μεθόδου.

Μια άλλη προβληματική κατάσταση εμφανίζεται κατά την επίλυση του τριδιαγώνιου συστήματος. Η επίλυση γίνεται με LU διάσπαση χωρίς οδήγηση και αν η διαδικασία αυτή δεν είναι επιτυχής τότε έχουμε breakdown δευτέρου βαθμού. Αυτό αντιμετωπίζεται εφαρμόζοντας LU διάσπαση στα 2×2 block διαγώνια στοιχεία [11].

Μέθοδος Quasi-Minimal Residual - QMR

Η μέθοδος Bi-CG δεν είναι σε θέση να υπολογίσει τη λύση του ελάχιστου υπολοίπου με αναδρομικές σχέσεις λίγων όρων. Η μέθοδος που είναι πιο κοντά σε αυτό λέγεται QMR [43]. Ας ξεκινήσουμε από τις αναδρομικές σχέσεις

$$A\, V_m \;=\; V_{m+1}\, T_{m+1,m} \quad . \tag{45}$$

Ζητάμε μια προσεγγιστική λύση $\boldsymbol{x}^{(m)}$, με $\boldsymbol{x}^{(m)} \in \mathcal{K}^m(A; \boldsymbol{r}^{(m)})$, δηλαδή $\boldsymbol{x}^{(m)} = V_m\, \boldsymbol{y}$, για την οποία η

$$\| \boldsymbol{b} - A\, \boldsymbol{x}^{(m)} \|_2 = \| \boldsymbol{b} - A\, V_m\, \boldsymbol{y} \|_2 = \| \boldsymbol{b} - V_{m+1}\, T_{m+1,m}\, \boldsymbol{y} \|_2 \tag{46}$$

γίνεται ελάχιστη. Το πρόβλημα είναι με τον πίνακα V_{m+1}, ο οποίος δεν είναι ορθογώνιος. Αν όμως ήταν, τότε θα είχαμε ότι

$$\begin{aligned} \| \boldsymbol{b} - \mathrm{A}\, \boldsymbol{x}^{(m)} \|_2 &= \| V_{m+1} \, (\| \boldsymbol{r}^{(0)} \|_2 \ \boldsymbol{e}_1 \ - \ T_{m+1,m} \, \boldsymbol{y}) \|_2 \\ &= \| \, (\| \boldsymbol{r}^{(0)} \|_2 \ \boldsymbol{e}_1 \ - \ T_{m+1,m} \, \boldsymbol{y}) \, \|_2 \equiv \ q_Q^k \quad , \end{aligned} \tag{47}$$

και στο [43] προτείνεται η λύση του προβλήματος των ελαχίστων τετραγώνων της ελάχιστης νόρμας του δεξιού μέλους, που έχει προβληθεί στον Krylov υπόχωρο. Αφού, γενικά, οι στήλες του πίνακα V_{m+1} δεν είναι ορθογώνιες η έκφραση $\boldsymbol{x}^{(m)} = V_m \ \boldsymbol{y}$ δεν επιλύει το πρόβλημα του ελαχίστου υπολοίπου, γι' αυτό η μέθοδος αυτή αναφέρεται [43] ως quasi-minimum. Επίσης, όταν ο πίνακας V_{m+1} δεν είναι ορθογώνιος η quasi νόρμα υπολοίπου q_Q^k δεν θα είναι ίση με την $\| \boldsymbol{b} - \mathrm{A}\, \boldsymbol{x}^{(m)} \|_2$, αλλά θα ισχύει ότι $\| \boldsymbol{b} - \mathrm{A}\, \boldsymbol{x}^{(k)} \|_2 \leq \sqrt{k}\ q_Q^k$.

Η μέθοδος που υλοποιεί τα παραπάνω ονομάζεται QMR και εμφανίζεται σε διάφορες μορφές[13, 37, 38, 39, 84]. Στην Εικόνα 8 εμφανίζεται ο αλγόριθμος της QMR χωρίς τη στρατηγική πρόβλεψης (look-ahead).

Έχει παρατηρηθεί μετά από πειραματικές μετρήσεις ότι η σύγκλιση της QMR είναι πιο ομαλή από της Bi-CG, με αποτέλεσμα να προτιμάται από την Bi-CG. Όμως η QMR δεν έχει σημαντικά γρηγορότερη σύγκλιση, γιατί με χρήση ακριβούς αριθμητικής προκύπτει [29, 50] η παρακάτω σχέση υπολοίπων των δυο μεθόδων

$$\| \boldsymbol{r}_B^{(m)} \|_2 = \frac{q_Q^m}{\sqrt{1 - (\frac{q_Q^m}{q_Q^{m-1}})^2}} \quad , \tag{48}$$

όπου $\boldsymbol{r}_B^{(m)}$ είναι το υπόλοιπο της μεθόδου Bi-CG. Εύλογο είναι το ερώτημα αν είναι η QMR καλύτερη από την GMRES. Η απάντηση δεν είναι εύκολη, αφού η σύγκριση ανάμεσά τους έχει δυσκολίες. Έτσι η GMRES ελαχιστοποιεί την ευκλίδεια νόρμα του υπολοίπου, με το κόστος της φύλαξης όλων των ορθογώνιων υπολοίπων. Η QMR δεν ελαχιστοποιεί αυτή την νόρμα, αλλά συνήθως συγκλίνει με τον ίδιο ρυθμό, όμως με διπλάσιο υπολογιστικό κόστος σε πολλαπλασιασμούς πίνακα με διάνυσμα ανά επαναληπτικό βήμα. Η κατασκευή της βάσης του υποχώρου στην QMR είναι

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$; $y = \tilde{u}^{(1)}$; $\rho_1 = \| y \|_2$
Επιλογή $\tilde{w}^{(1)}$, για παράδειγμα $\tilde{w}^{(1)} = r^{(0)}$
$z = \tilde{w}^{(1)}$; $\xi_1 = \| z \|_2$
$\gamma_0 = 1$; $\eta_0 = -1$
for $i = 1, 2, ...$
if $\rho_i = 0$ **or** $\xi_i = 0$ η μέθοδος αποτυγχάνει
$u^{(i)} = \frac{\tilde{u}^{(i)}}{\rho_i}$; $y = \frac{y}{\rho_i}$
$w^{(i)} = \frac{\tilde{w}^{(i)}}{\xi_i}$; $z = \frac{z}{\xi_i}$; $\delta_i = z^T y$
if $\delta_i = 0$ η μέθοδος αποτυγχάνει
$\tilde{y} = y$; $\tilde{z} = z$
if $i = 1$
$p^{(1)} = \tilde{y}$; $q^{(1)} = \tilde{z}$
else
$p^{(i)} = \tilde{y} - \frac{\xi_i \delta_i}{\epsilon_{i-1}} p^{(i-1)}$; $q^{(i)} = \tilde{z} - \frac{\rho_i \delta_i}{\epsilon_{i-1}} q^{(i-1)}$
endif
$\tilde{p} = A p^{(i)}$; $\epsilon_i = q^{(i)T}$
if $\epsilon_i = 0$ η μέθοδος αποτυγχάνει
$\beta_i = \frac{\epsilon_i}{\delta_i}$
if $\beta_i = 0$ η μέθοδος αποτυγχάνει
$\tilde{u}^{(i+1)} = \tilde{p} - \beta_i u^{(i)}$; $y = \tilde{u}^{(i+1)}$
$\rho_{i+1} = \| y \|_2$; $\tilde{w}^{(i+1)} = A^T q^{(i)} - \beta_i w^{(i)}$
$z = \tilde{w}^{(i+1)}$; $\xi_{i+1} = \| z \|_2$
$\theta_i = \frac{\rho_{i+1}}{\gamma_{i-1}|\beta_i|}$; $\gamma_i = \frac{1}{\sqrt{1+\theta_i^2}}$
if $\gamma_i = 0$ η μέθοδος αποτυγχάνει
$\eta_i = \frac{-\eta_{i-1}\rho_i\gamma_i^2}{\beta_i\gamma_{i-1}^2}$
if $i = 1$
$d^{(1)} = \eta_1 p^{(1)}$; $s^{(1)} = \eta^1 \tilde{p}$
else
$d^{(i)} = \eta_i p^{(i)} + (\theta_{i-1}\gamma_i)^2 d^{(i-1)}$
$s^{(i)} = \eta_i \tilde{p} + (\theta_{i-1}\gamma_i)^2 s^{(i-1)}$
endif
$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + d^{(i)}$, $r^{(i)} = r^{(i-1)} - s^{(i)}$
Έλεγχος σύγκλισης
end

Εικόνα 8 : Αλγόριθμος της μεθόδου QMR.

πάλι σχετικά πιο οικονομική και με λιγότερες απαιτήσεις στη μνήμη. Πρέπει να πούμε ότι προταθεί αρκετές τροποποιήσεις της QMR, αλλά και της Bi-CG, οι οποίες αυξάνουν την αποδοτικότητα των μεθόδων. Οι Zhou και Walker [127] έδειξαν ότι και άλλες μέθοδες μπορούν να ακολουθήσουν την προσέγγιση της QMR. Η κύρια ιδέα είναι ότι σε αυτές τις μεθόδους η προσεγγιστική λύση ανανεώνεται σύμφωνα με τη σχέση

$$\boldsymbol{x}^{(m+1)} \;=\; \boldsymbol{x}^{(m)} \;+\; \alpha_m \; p^m \quad , \tag{49}$$

οπότε για το υπόλοιπο θα ισχύει

$$\boldsymbol{r}^{(m+1)} \;=\; \boldsymbol{r}^{(m)} \;-\; \alpha_m \; A \; p^m \quad . \tag{50}$$

Αυτό σημαίνει ότι $A\ P_m \;=\; W_m\ R_{m+1}$ με W_m κάτω διδιαγώνιο πίνακα. Τα $\boldsymbol{x}^{(m)}$ είναι συνδυασμοί των p^m, κι έτσι θα πρέπει να βρεθεί μια σχέση των $P_m\ \boldsymbol{y}_m$ για τον οποίο η νόρμα $\| \ \boldsymbol{b} \;-\; A\ P_m\ \boldsymbol{y}_m \ \|_2$ γίνεται ελάχιστη. Αν εισάγουμε την έκφραση A P_m και αγνοήσουμε το γεγονός ότι τα $\boldsymbol{r}^{(m)}$ δεν είναι ορθογώνια, τότε μπορούμε να ελαχιστοποιήσουμε την νόρμα του υπολοίπου με την έννοια quasi-minimum των ελαχίστων τετραγώνων, όμοια με την QMR.

Μέθοδος Conjugate Gradients Squared - CGS

Επειδή για αρκετές από τις μεθόδους της κατηγορίας της BiCG έχουμε $\boldsymbol{r}^{(m)} \;\in\; \mathcal{K}^{m+1}(A; \boldsymbol{r}^{(0)})$ και επομένως μπορούμε να γράψουμε $\boldsymbol{r}^{(m)} \;=\; P_m(A)\ \boldsymbol{r}^{(0)}$, όπου P_m είναι πολυώνυμο βαθμού m. Άρα και για τα διανύσματα βάσης $\widehat{\boldsymbol{r}}^{(m)}$ θα ισχύει η σχέση $\widehat{\boldsymbol{r}}^{(m)} \;=\; P_m(A^T)\ \widehat{\boldsymbol{r}}^{(0)}$ και θα έχουμε για τους συντελεστές από τις bi-orthogonality σχέσεις

$$\begin{aligned} \alpha_m \;&=\; (\widehat{\boldsymbol{r}}^{(m)}\,, A\ \boldsymbol{r}^{(m)}) \\ &=\; (P_m\ (A^T)\widehat{\boldsymbol{r}}^{(0)}\ ,\ A\ P_m(A)\boldsymbol{r}^{(0)}) \\ &=\; (\widehat{\boldsymbol{r}}^{(0)}\ ,\ A\ P_m^2(A)\boldsymbol{r}^{(0)}) \quad . \end{aligned} \tag{51}$$

Ο Sonneveld [110] παρατήρησε ότι μπορούμε επίσης να κατασκευάσουμε τα $\widetilde{\boldsymbol{r}}^{(m)} \;=\; P_m^2(A)\ \boldsymbol{r}^{(0)}$ από επαναληπτικές σχέσεις χωρίς τον υπολογισμό των $\boldsymbol{r}^{(m)}$ και των $\widehat{\boldsymbol{r}}^{(m)}$.

Έτσι έχουμε τα εξής πλεονεκτήματα

1. Επειδή δεν υπολογίζονται τα $\widehat{\boldsymbol{r}}^{(m)}$ δεν γίνονται υπολογισμοί με τον A^T. Έτσι μπορεί να εφαρμοστεί η μέθοδος για προβλήματα που ο A^T δεν είναι διαθέσιμος.

2. Ο τελεστής $P_m(A)$ εκφράζει την μείωση του $\boldsymbol{r}^{(0)}$ σε $\boldsymbol{r}^{(m)}$, οπότε ο $P_m(A)^2$ θα περίμενε κανείς ότι θα έφερνε το διπλάσιο αποτέλεσμα. Αυτό όμως δεν ισχύει πάντα γιατί με το τελεστή $P_m(A)$ παράγεται από το $\boldsymbol{r}^{(0)}$ το $\boldsymbol{r}^{(m)}$, το οποίο είναι ορθογώνιο στον υπόχωρο. Έτσι δεν μπορεί να εξασφαλιστεί ότι το $\widetilde{\boldsymbol{r}}^{(m)} = P_m^2(A)\,\boldsymbol{r}^{(0)}$ θα είναι ορθογώνιο στον υπόχωρο $\mathcal{K}^{2m}(A^T;\widetilde{\boldsymbol{r}}^{(0)})$.

Η παραγόμενη μέθοδος ονομάζεται Conjugate Gradients-Squared (CGS) [23] και ο αλγόριθμος υλοποίησης εμφανίζεται στην Εικόνα 9. Η CGS έχει ευρεία χρήση κυρίως λόγω της γρήγορης σύγκλισης. Συγκριτικές δοκιμές ανάμεσα της με τις GMRES και Bi-CG μπορεί να βρεθούν στα [96, 22, 95, 85]. Η CGS όμως εμφανίζει πολύ ασταθή συμπεριφορά κι αυτό μπορεί να δικαιολογηθεί με το γεγονός ότι, όπως και η σύγκλιση, έτσι και οι παράγοντες αστάθειας της Bi-CG [17, 18, 19] έχουν υψωθεί στο τετράγωνο. Αυτό μπορεί να οδηγήσει σε απώλεια ακρίβειας της λύσης [117]. Για συμμετρικούς και θετικά διευθετιμένους πίνακες η CGS δεν αποτυγχάνει, γιατί παράγονται τα ίδια $\boldsymbol{r}^{(m)}$ και $\boldsymbol{x}^{(m)}$ όπως στην CG. Αντίθετα για μη συμμετρικά συστήματα δεν έχει δοθεί ακόμα κάποια ανάλυση σύγκλισης.

Αντίστοιχα με την Bi-CG και την CGS ο Freund [44] έχει παράξει την Squared QMR (TFQMR) και σε πειραματικές μετρήσεις εμφανίζει πιο ομαλή σύγκλιση από την CGS, αλλά όχι απαραίτητα γρηγορότερη.

Μέθοδος BiCGSTAB

Η μέθοδος Bi-CGSTAB βασίζεται στην παρατήρηση, ότι στην Bi-CG τα $\boldsymbol{r}^{(m)}$ είναι ορθογώνια σ' όλο τον Krylov υπόχωρο $\mathcal{K}^m(A^T, w_1)$ κι έτσι μπορούμε να τα κατασκευάσουμε απαιτώντας να είναι ορθογώνια σε άλλα διανύσματα βάσης

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}$ (συνήθως $\hat{r} = r^{(0)}$)

for $i = 1, 2, ...$

$\rho_{i-1} = \hat{r}^* \, r^{(i-1)}$

if $\rho_{i-1} = 0$ η μέθοδος αποτυγχάνει

if $i = 1$

$u^{(1)} = r^{(0)}$

$p^{(1)} = u^{(1)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}}$

$u^{(i)} = r^{(i-1)} + \beta_{i-1} q^{(i)}$

$p^{(i)} = u^{(i)} + \beta_{i-1} \, (q^{(i-1)} + \beta_{i-1} \, p^{(i-1)})$

endif

$\hat{p} = p^{(i)}$

$\hat{v} = A \, \hat{p}$

$\alpha_i = \frac{\rho_{i-1}}{\hat{r}^* \hat{v}}$

$q^{(i)} = u^{(i)} - \alpha_i \, \hat{v}$

$\hat{u} = u^{(i)} + q^{(i)}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{u}$

$r^{(i)} = r^{(i-1)} - \alpha_i \, A \, \hat{u}$

Έλεγχος σύγκλισης

end

Εικόνα 9 : Αλγόριθμος της μεθόδου CGS.

του $\mathcal{K}^m(A^T, w_1)$. Αντί λοιπόν να χρησιμοποιηθεί το τετράγωνο του πολυωνύμου της Bi-CG μπορούν τα $\boldsymbol{x}^{(m)}$ να κατασκευαστούν από τα $\boldsymbol{r}^{(m)} = Q_m(A)\, P_m(A)\, \boldsymbol{r}^{(0)}$, για κάποια πολυώνυμα Q_m.

Μια προφανής μορφή αυτών των πολυωνύμων είναι η

$$Q_m(t) = (1-\omega_1 t)\,(1-\omega_2 t)\ldots(1-\omega_m t) \quad , \tag{52}$$

για κάποιες σταθερές $\omega_j \quad j = 1, \ldots, m$. Οι σταθερές αυτές επιλέγονται έτσι ώστε στο βήμα i το $\boldsymbol{r}^{(i)}$ να γίνεται ελάχιστο. Στην Εικόνα 10 εμφανίζεται ο αλγόριθμος που υλοποιεί την μέθοδο Bi-CGSTAB και όπως φαίνεται σε κάθε επαναληπτικό βήμα χρειάζονται δυο επιπλέον εσωτερικά γινόμενα διανυσμάτων σε σχέση με την CGS. Η Bi-CGSTAB [24] ανήκει στην κατηγορία των υβριδικών μεθόδων, δηλαδή είναι αποτέλεσμα συνδυασμού κάποιων άλλων μεθόδων και πιο συγκεκριμένα της Bi-CG και της GMRES(1). Έτσι συγκλίνει όσο γρήγορα και η CGS, αλλά λόγω της τοπικής ελαχιστοποίησης της νόρμας του υπολοίπου διανύσματος έχει πιο ομαλή σύγκλιση. Έχουν προταθεί και γενικεύσεις της Bi-CGSTAB, δηλαδή συνδυασμός της Bi-CG και της GMRES(m) και εμφανίζονται ως Bi-CGSTAB(m) [56, 107, 108].

1.5 Preconditioners

Αρκετές φορές και ανάλογα με το είδος του γραμμικού συστήματος οι επαναληπτικές μέθοδοι αποκλίνουν ή όταν καταφέρουν να συγκλίνουν αυτό γίνεται πολύ αργά. Έτσι σχεδόν πάντα είναι απαραίτητη η εφαρμογή κάποιου μετασχηματισμού, μέσω κάποιου πίνακα M στο γραμμικό σύστημα, ώστε τουλάχιστον να βελτιωθεί ο ρυθμός σύγκλισης της μεθόδου. Αυτό επιτυγχάνεται στηριζόμενοι την ιδέα ότι ο πίνακας $M^{-1}A$ ίσως έχει καλλίτερες ιδιότητες, έτσι ώστε η επαναληπτική μέθοδος να συγκλίνει γρηγορότερα. Δηλαδή, αντί της επίλυσης του $A\boldsymbol{x} = \boldsymbol{b}$ τελικά επιλύεται το $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$. Η επιλογή του πίνακα M, ο οποίος ονομάζεται preconditioner ή precondition matrix [21, 53], βασίζεται στα παρακάτω κριτήρια :

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}$ (συνήθως $\hat{r} = r^{(0)}$)

for $i = 1, 2, ...$

$\rho_{i-1} = \hat{r}^T \, r^{(i-1)}$

if $\rho_{i-1} = 0$ η μέθοδος αποτυγχάνει

if $i = 1$

$p^{(1)} = r^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \, \frac{\alpha_{i-1}}{\omega_{i-1}}$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} \, (p^{(i-1)} - \omega_{i-1} \, v^{(i-1)})$

endif

$\hat{p} = p^{(i)}$

$v^{(i)} = A \, \hat{p}$

$\alpha_i = \frac{\rho_{i-1}}{\hat{r}^T v^{(i)}}$

$s = r^{(i-1)} - \alpha_i \, v^{(i)}$

if $\| \, s \, \|$ αρκετά μικρό **then**

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{p}$ έξοδος

$z = s \quad ; \quad t = A \, z$

$\omega_i = \frac{s^T t}{t^T t}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{p} + \omega_i \, z$

Έλεγχος σύγκλισης

if $\omega_i = 0$ έξοδος

$r^{(i)} = s - \omega_i \, t$

end

Εικόνα 10 : Αλγόριθμος της μεθόδου Bi-CGSTAB.

1. Ο πίνακας M να αποτελεί όσο γίνεται καλύτερη προσέγγιση του πίνακα A.

2. Το κόστος κατασκευής του M είναι σε σχετικά μικρό.

3. Η επίλυση του συστήματος $M\boldsymbol{x} = \boldsymbol{b}$ είναι πιο εύκολη από του αρχικού.

Αξίζει να σημειωθεί ότι κατά την πράξη $M^{-1}A\boldsymbol{u}$ το γινόμενο $M^{-1}A$ δεν απαιτεί τη δημιουργία του M^{-1}, αλλά η πράξη υπολογίζεται έμμεσα μέσω του υπολογισμού $\boldsymbol{w} = A\,\boldsymbol{u}$ και στη συνέχεια της επίλυσης του $M\,\boldsymbol{v} = \boldsymbol{w}$.

Υπάρχουν διαφορετικοί τρόποι εφαρμογής preconditioning. Όλες όμως οδηγούν στις ιδιοτιμές του preconditioned πίνακα, οι οποίες επηρεάζουν την σύγκλιση της μεθόδου. Επειδή όμως, οδηγούν ταυτόχρονα σε διαφορετικά ιδιοδιανύσματα, τα οποία επηρεάζουν κι αυτά με τη σειρά τους την σύγκλιση της μεθόδου, συμπαιρένεται ότι ο τρόπος εφαρμογής του preconditioning καθορίζει και τη σύγκλιση της επαναληπτικής μεθόδου. Υπάρχουν τρεις διαφορετικοί τρόποι εφαρμογής του μετασχηματισμού στο αρχικό σύστημα.

1. *Μέθοδος αριστερού preconditioning* : Εφαρμόζεται στο σύστημα με τη μορφή $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$. Για τις περιπτώσεις συμμετρικών πινάκων M και A θα πρέπει να προσεχθεί το γεγονός ότι η συμμετρία δεν διατηρείται απαραίτητα και στον πίνακα $M^{-1}A$. Επίσης κατά τη χρήση των μεθόδων της οικογενείας GMRES με την εφαρμογή του preconditioning ελαχιστοποιείται το υπόλοιπο $M^{-1}\,(\boldsymbol{b} - A\boldsymbol{x}^{(m)})$ αντί του $\boldsymbol{b} - A\boldsymbol{x}^{(m)}$. Το γεγονός αυτό μπορεί να έχει συνέπειες σε κριτήρια τερματισμού που χρησιμοποιούν τη νόρμα του υπολοίπου.

2. *Μέθοδος δεξιού preconditioning* : Εφαρμόζεται στο σύστημα με τη μορφή $A\,M^{-1}\boldsymbol{y} = \boldsymbol{b}$, όπου $\boldsymbol{x} = M^{-1}\boldsymbol{y}$. Ισχύουν όπως και στην προηγούμενη περίπτωση οι παρατηρήσεις περί συμμετρικότητας και υπολοίπου. Πιο συγκεκριμένα το σφάλμα $\| \boldsymbol{y} - \boldsymbol{y}^{(m)} \|_2$ μπορεί να είναι πολύ μικρότερο του

$\| \boldsymbol{x} - \boldsymbol{x}^{(m)} \|_2$ το οποίο είναι ισοδύναμο με $\| M^{-1} (\boldsymbol{y} - \boldsymbol{y}^{(m)}) \|_2$ που μας ενδιαφέρει. Ακόμα είναι αξιοσημείωτο ότι η μέθοδος του δεξιού preconditioning δεν εμπλέκει καθόλου το δεξίο μέλος του γραμμικού συστήματος κι αυτό μπορεί να θεωρηθεί ως σημαντικό πλεονέκτημα κατά την κατασκευή λογισμικού.

3. *Μέθοδος διπλού preconditioning* : Εφαρμόζεται στο σύστημα με τη μορφή $M_1^{-1} A M_2^{-1} \boldsymbol{z} = M_1^{-1} \boldsymbol{b}$, όπου $\boldsymbol{x} = M_2^{-1} \boldsymbol{z}$ και για $M = M_1 M_2$. Η μέθοδος αυτή προτιμάται για παραγοντοποιημένους preconditioners πίνακες M και είναι συνδυασμός των προηγουμένων δύο ειδών.

Η εφαρμογή ενός preconditioning σε ένα γραμμικό σύστημα έχει ώς αποτέλεσμα την αύξηση του υπολογιστικού κόστους και της πολυπλοκότητας της επαναληπτικής μεθόδου, γι' αυτό θα πρέπει η εφαρμογή αυτή να συνοδεύεται με σημαντική αύξηση της ταχύτητας σύγκλισης, δηλαδή μείωση των επαναληπτικών βημάτων της μεθόδου. Αν υποθέσουμε, ότι σε μια επαναληπτική μέθοδο με σταθερό υπολογιστικό κόστος ανά βήμα, ο υπολογιστικός χρόνος των πολλαλασιασμών πινάκων με διανύσματα είναι t_A και ο αντίστοιχος των εσωτερικών γινομένων και λοιπών πράξεων είναι t_o, μετά από k βήματα ο συνολικός χρόνος θα είναι

$$T_U = k(t_A + t_o) \quad . \tag{53}$$

Κατά τη διαδικασία του preconditioning κάνουμε τις εξής παραδοχές :

- Η δράση του πίνακα preconditioner, για παράδειγμα ο υπολογισμός του $M^{-1} \boldsymbol{z}$ χρειάζεται χρόνο at_A
- Η κατασκευή του πίνακα preconditioner χρειάζεται χρόνο t_c
- Με το μετασχηματισμό του συστήματος τα βήματα της μεθόδου έχουν μειωθεί με κατά ένα παράγοντα $f \quad (f > 1)$.

Ο χρόνος λοιπόν για την κατασκευή της ίδιας προσεγγιστικής λύσης με την παραπάνω μέθοδο, αλλά χρησιμοποιώντας την τεχνική του preconditioning, θα είναι

$$T_P \;=\; \frac{k}{f}\,(\,(\;a\;+\;1\;)\,t_A\;+\;t_o\;)\;+\;t_c \quad . \tag{54}$$

Για να είναι συμφέρουσα η τεχνική του preconditioning θα πρέπει να ισχύει ότι $T_P \;<\; T_U$, το οποίο έχει ως αποτέλεσμα

$$f \;>\; \frac{(\;a\;+\;1\;)\,t_A\;+\;t_o}{t_A\;+\;t_o\;-\;\frac{t_c}{k}} \quad . \tag{55}$$

Άρα η κατασκευή preconditioners με μεγάλο υπολογιστικό κόστος δεν είναι συμφέρουσα για την περίπτωση κατά την οποία η μέθοδος χωρίς preconditioning συγκλίνει σε μικρό αριθμό βημάτων k. Δηλαδή preconditioning πρέπει να εφαμόζεται για περιπτώσεις, όπου τα βήματα k είναι τόσα πολλά, ώστε ο χρόνος κατασκευής του preconditioner να είναι μικρός σχετικά, να ισχύει ότι $t_A \;+\; t_o \gg \frac{t_c}{k}$. Επίσης συχνά ο πολλαπλασιασμός του πίνακα A με ένα διάνυσμα είναι το πιο ακριβό μέρος της μεθόδου σε υπολογιστικό κόστος , δηλ $t_A \geq t_o$ κι έτσι η τεχνική preconditioning είναι αποδοτική όταν ο λόγος μείωσης των επαναληπτικών βημάτων f είναι σημαντικά μεγαλύτερος του $a \;+\; 1$. Η παρατήρηση αυτή σε συνδυασμό με το γεγονός ότι η τεχνική preconditioning δεν έχει συνήθως *παράλληλες* ιδιότητες έχει σημαντική αξία.

Ειδικότερα κατά την εφαρμογή preconditioning στην οικογένεια των μεθόδων GMRES υπάρχουν κάποιες διαφοροποιήσεις, επειδή η υπολογιστική επιβάρυνση αυξάνει τετραγωνικά σε σχέση με τα επαναληπτικά βήματα. Ας υποθέσουμε ότι για την GMRES(m), ο υπολογιστικός χρόνος χωρίζεται στα παρακάτω τμήματα

- ο υπολογιστικός χρόνος πολλαπλασιασμού του A με διανύσματα είναι t_A.
- ο υπολογιστικός χρόνος των εσωτερικών γινομένων διανυσμάτων μαζί με ένα πολλαπλασιασμό διανύσματος με σταθερά και την πρόσθεση του σε ένα διανύσμα είναι t_o.

Έτσι ο συνολικός υπολογιστικός χρόνος της GMRES(m) χωρίς preconditioning για k βήματα είναι

$$T_U \;=\; k(m\,t_A \;+\; \frac{1}{2}\,m^2\,t_o) \quad . \tag{56}$$

Υποθέτουμε πάλι ότι km είναι αρκετά μεγάλο σε σχέση με το κόστος κατασκευής του preconditioner και η τεχνική αυτή μειώνει f φορές τα επαναληπτικά βήματα της GMRES(m). Αν η δράση ξανά του preconditioner είναι $a\,t_A$, τότε ο χρόνος της μεθόδου με preconditioning θα είναι

$$T_P \;=\; \frac{k}{f}\,(\;m\,(\;a\;+\;1\;)\,t_A \;+\; \frac{1}{2}\,m^2\,t_o) \tag{57}$$

και ξανά η τεχνική preconditioning είναι αποδοτική αν

$$f \;>\; \frac{(\;a\;+\;1\;)\,t_A \;+\; \frac{1}{2}\,m\,t_o}{t_A \;+\; \frac{1}{2}\,m\,t_o} \quad . \tag{58}$$

Είναι φανερό, ότι για μικρές τιμές του m και μεγάλο χρόνο t_A, το preconditioning συμφέρει αν η μείωση f είναι αρκετά μεγαλύτερη του $a+1$. Ακόμα για μεγάλες τιμές του m, όπου ο χρόνος $\frac{1}{2}mt_o$ να είναι σημαντικά μεγαλύτερος του χρόνου t_A, η τεχνική preconditioning συμφέρει και για μικρές τιμές του f.

Οι πιο συνηθισμένοι preconditioners [100, 99, 103, 113, 116, 122] είναι αυτοί που παράγονται άμεσα από τον πίνακα των συντελεστών των αγνώστων του γραμμικού συστήματος σύμφωνα με κάποια διαμέριση του [26]. Αυτό ισχύει γιατί το κόστος κατασκευής του preconditioner είναι πολύ μικρό. Έτσι μπορεί να χρησιμοποιηθεί ο πίνακας πίνακας διάσπασης της Jacobi μεθόδου σε block μορφή, ο οποίος είναι από τους πιο δημοφιλής εξαιτείας της πολύ εύκολης υλοποίησης του και σε παράλληλα περιβάλλοντα. Οι πίνακες διάσπασης των μεθόδων Gauss-Seidel και SOR δεν έχουν πρακτική εφαρμογή ως preconditioners στις non-stationary μεθόδους. Σε πρόσφατες μελέτες από τους Eiermann και Varga [34] απεδείχθει ότι η πολυωνυμική επιτάχυνση μιας non-stationary μεθόδου με SOR και με ω κοντά στο ω_{opt} δεν βελτιώνει τη σύγκλιση έναντι της stationary SOR με ω_{opt}. Αντίθετα, όπως ο επαναληπτικός πίνακας της

Jacobi, έτσι και αυτός της SSOR [6] μπορεί να κατασκευαστεί χωρίς επιπλέον κόστος, με αποτέλεσμα να χρησιμοποιείται ευρέως. Επίσης ο πίνακας της SSOR αποτελεί μια ικανοποιητική προσέγγιση του πίνακα των συντελεστών των αγνώστων του γραμμικού συστήματος κι έτσι μπορεί να θεωρηθεί καλός preconditioner για ορισμένα πάντα προβλήματα, αν φυσικά η μείωση των επαναληπτικών βημάτων είναι αρκετή ώστε να αντισταθμίσει την σημαντική επιβάρυνση κάθε βήματος από το preconditioning της SSOR.

Μια άλλη κατηγορία preconditioners είναι αυτή των incomplete factorization [7, 8, 54, 75]. Δηλαδή ο πίνακας $M = LU$ δίνεται σε κάποια μορφή παραγοντοποίησης με τους πίνακες L και U κάτω και άνω τριγωνικούς αντίστοιχα. Ο πίνακας M επιλέγεται ώστε M^{-1} να αποτελεί ικανοποιητική προσέγγιση του A^{-1}. Ο όρος incomplete factorization αποδίδεται στο γεγονός ότι κατά την παραγοντοποίηση του πίνακα M έχουν αγνοηθεί τα στοιχεία που πριν την παραγοντοποίηση ήταν μηδενικά και κατά τη διάρκειά της έγιναν μη μηδενικά. Σημειώνουμε επίσης ότι η δημιουργία τους δεν είναι εξασφαλισμένη, αφού η διαδικασία της ατελούς παραγοντοποίησης μπορεί να αποτύχει. Το κόστος της παραγοντοποίησης, που μπορεί να φτάσει το κόστος μερικών επαναληπτικών βημάτων ανάλογα με το πρόβλημα, αποτελεί σημαντικό παράγοντα, ο οποίος θα πρέπει να λαμβάνεται σοβαρά υπόψη, σε συνδυασμό με το γεγονός ότι η διαδικασία αυτή δεν έχει ιδιότητες παραλληλίας ώστε να είναι εκμεταλλευσιμες σε παράλληλα περιβάλλοντα.

Μια άλλη μέθοδος προσέγγισης του αντιστρόφου του πίνακα των συντελεστών των αγνώστων του γραμμικού συστήματος αποτελεί το πολυωνυμικό preconditioning [10, 28, 35, 65]. Η μέθοδος αυτή στηρίζεται στο γεγονός της διάσπασης του πίνακα των συντελεστών $A = I - B$, όταν η φασματική ακτίνα του πίνακα B είναι μικρότερη της μονάδος. Άρα σύμφωνα με την σειρά Neumann ο αντίστροφος μπορεί να εκφραστεί στη μορφή

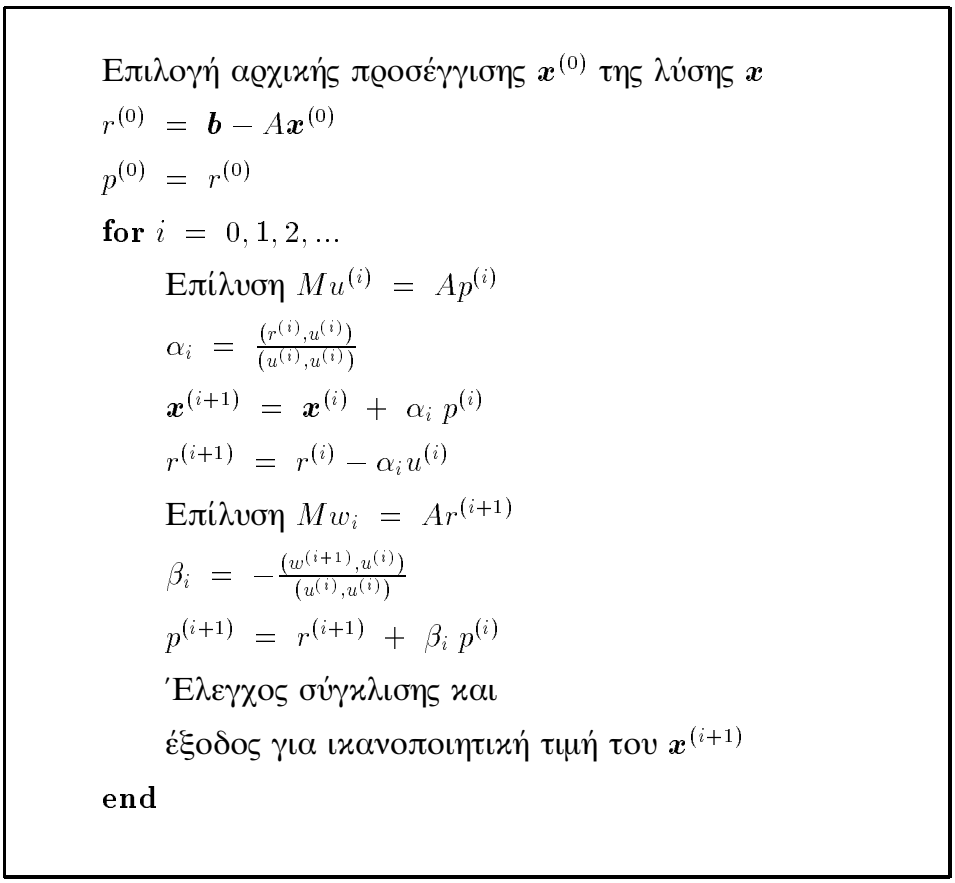
$$A^{-1} = \sum_{k=0}^{\infty} B^k \tag{59}$$

Οι Dubois, Greenbaum και Rodrigue [33] ασχολήθηκαν με τη σχέση της βασικής επαναληπτικής μεθόδου με τη διάσπαση του πίνακα $A = M - \mathrm{N}$ και στη πολυωνυμική μέθοδο preconditioning με

$$M_p^{-1} = (\sum_{i=0}^{p-1} (I - M^{-1} A)^i) M^{-1} \quad . \tag{60}$$

Το βασικό αποτέλεσμα είναι ότι για τις συνήθης non-stationary επαναληπτικές μεθόδους k επαναληπτικά βήματα με πολυωνυμικό preconditioning είναι ισοδύναμα με kp βήματα της μεθόδου χωρίς preconditioning.

Στη συνέχεια παρουσιάζονται οι αλγόριθμοι των βασικότερων non-stationary μεθόδων, όπως αυτοί τροποποιούνται μετά την εφαρμογή της τεχνικής του preconditioning.



Εικόνα 11a : Αλγόριθμος της μεθόδου ORTHOMIN με preconditioning.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

Επίλυση του M $r \;=\; \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

for $j \;=\; 1, 2, ...$

$\beta \;=\; \| \; r \; \|_2 \;, \quad u^1 \;=\; \frac{r}{\beta} \;, \quad \hat{b} \;=\; \beta \; e^1$

for $i \;=\; 1, 2, ..., m$

Επίλυση του $M \; w \;=\; A \; u^i$

for $k \;=\; 1, 2, ..., i$

$h_{k,i} \;=\; (u^k)^* \; w \;, \quad w \;=\; w \;-\; h_{k,i} u^k$

$h_{i+1,i} \;=\; \| \; w \; \|_2 \;, \quad u^{i+1} \;=\; \frac{w}{h_{i+1,i}}$

for $k \;=\; 2, ..., i$

$r_{k-1,i} \;=\; c_{k-1} \; h_{k-1,i} \;+\; s_{k-1} \; h_{k,i}$

$r_{k,i} \;=\; -s_{k-1} \; h_{k-1,i} \;+\; c_{k-1} \; h_{k,i}$

$\delta \;=\; \sqrt{h_{i,i}^2 \;+\; h_{i+1,1}^2}$

$c_i \;=\; \frac{h_{i,i}}{\delta} \;, \quad s_i \;=\; \frac{h_{i+1,i}}{\delta}$

$r_{i,i} \;=\; c_i \; h_{i,i} \;+\; s_i \; h_{i+1,i}$

$\hat{b}_{i+1} \;=\; -s_i \; \hat{b}_i \;, \quad \hat{b}_i \;=\; c_i \; * \; \hat{b}_i$

$\rho \;=\; | \; \hat{b}_{i+1} \; | \quad (=\| \; \boldsymbol{b} - A\boldsymbol{x}^{((j-1)m+i)} \; \|_2)$

if ρ αρκετά μικρό **then**

$n_r \;=\; i \quad$ **goto** SOL

$n_r \;=\; m \;, \quad y_{n_r} \;=\; \frac{\hat{b}_{n_r}}{r_{n_r,n_r}}$

SOL : **for** $k \;=\; n_r - 1, ..., 1$

$y_k \;=\; \frac{\hat{b}_k \; - \sum_{i=k+1}^{n_r} r_{k,i} y_i}{r_{k,k}}$

$\boldsymbol{x} \;=\; \sum_{i=1}^{n_r} y_i u^i$

if ρ αρκετά μικρό **quit**

$r \;=\; \boldsymbol{b} - A\boldsymbol{x}$

Εικόνα 11b : Αλγόριθμος της μεθόδου GMRES(m) με preconditioning.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}^{(0)}$ (συνήθως $\hat{r}^{(0)} = r^{(0)}$)

for $i = 1, 2, ...$

Επίλυση του $M\, z^{(i-1)} = r^{(i-1)}$

Επίλυση του $M^T\, \hat{z}^{(i-1)} = \hat{r}^{(i-1)}$

$\rho_{i-1} = (z^{(i-1)})^T\, \hat{r}^{(i-1)}$

if $i = 1$

$p^1 = z^{(0)}$

$\hat{p}^1 = \hat{z}^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}}$

$p^i = z^{(i-1)} + \beta_{i-1} p^{i-1}$

$\hat{p}^i = \hat{z}^{(i-1)} + \beta_{i-1} \hat{p}^{i-1}$

endif

$q^i = A\, p^i$

$\hat{q}^i = A^T\, \hat{p}^i$

$\alpha_i = \frac{\rho_{i-1}}{(\hat{p}^i)^T q^i}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i\, p^i$

$r^{(i)} = r^{(i-1)} - \alpha_i\, q^i$

$\hat{r}^{(i)} = \hat{r}^{(i-1)} - \alpha_i\, \hat{q}^i$

Έλεγχος σύγκλισης

end

Εικόνα 12 : Αλγόριθμος της μεθόδου Bi-CG με preconditioning.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$
$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$; Επίλυση του $\mathrm{M}_1\ y = \tilde{u}^{(1)}$; $\rho_1 = \| y \|_2$
Επιλογή $\tilde{w}^{(1)}$, για παράδειγμα $\tilde{w}^{(1)} = r^{(0)}$
Επίλυση του $M2^T\ z = \tilde{w}^{(1)}$; $\xi_1 = \| z \|_2$
$\gamma_0 = 1$; $\eta_0 = -1$
for $i = 1, 2, ...$
if $\rho_i = 0$ **or** $\xi_i = 0$ η μέθοδος αποτυγχάνει
$u^{(i)} = \frac{\tilde{u}^{(i)}}{\rho_i}$; $y = \frac{y}{\rho_i}$
$w^{(i)} = \frac{\tilde{w}^{(i)}}{\xi_i}$; $z = \frac{z}{\xi_i}$; $\delta_i = z^T\ y$
if $\delta_i = 0$ η μέθοδος αποτυγχάνει
Επίλυση του $\mathrm{M}_2\ \tilde{y} = y$; Επίλυση του $\mathrm{M}_1^T\ \tilde{z} = z$
if $i = 1$
$p^{(1)} = \tilde{y}$; $q^{(1)} = \tilde{z}$
else
$p^{(i)} = \tilde{y} - \frac{\xi_i\ \delta_i}{\epsilon_{i-1}}\ p^{(i-1)}$; $q^{(i)} = \tilde{z} - \frac{\rho_i\ \delta_i}{\epsilon_{i-1}}\ q^{(i-1)}$
endif
$\tilde{p} = A\ p^{(i)}$; $\epsilon_i = q^{(i)^T}$
if $\epsilon_i = 0$ η μέθοδος αποτυγχάνει
$\beta_i = \frac{\epsilon_i}{\delta_i}$
if $\beta_i = 0$ η μέθοδος αποτυγχάνει
$\tilde{u}^{(i+1)} = \tilde{p} - \beta_i\ u^{(i)}$; Επίλυση του $M_1\ y = \tilde{u}^{(i+1)}$
$\rho_{i+1} = \| y \|_2$; $\tilde{w}^{(i+1)} = A^T\ q^{(i)} - \beta_i\ w^{(i)}$
Επίλυση του $M_2^T\ z = \tilde{w}^{(i+1)}$; $\xi_{i+1} = \| z \|_2$
$\theta_i = \frac{\rho_{i+1}}{\gamma_{i-1}\,|\beta_i|}$; $\gamma_i = \frac{1}{\sqrt{1+\theta_i^2}}$
if $\gamma_i = 0$ η μέθοδος αποτυγχάνει
$\eta_i = \frac{-\eta_{i-1}\,\rho_i\,\gamma_i^2}{\beta_i\,\gamma_{i-1}^2}$
if $i = 1$
$d^{(1)} = \eta_1 p^{(1)}$; $s^{(1)} = \eta^1 \tilde{p}$
else
$d^{(i)} = \eta_i p^{(i)} + (\theta_{i-1}\gamma_i)^2\ d^{(i-1)}$
$s^{(i)} = \eta_i \tilde{p} + (\theta_{i-1}\gamma_i)^2\ s^{(i-1)}$
endif
$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + d^{(i)}$, $r^{(i)} = r^{(i-1)} - s^{(i)}$
Έλεγχος σύγκλισης
end

Εικόνα 13 : Αλγόριθμος της μεθόδου QMR με preconditioning $M = M_1\ M_2$.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}$ (συνήθως $\hat{r} = r^{(0)}$)

for $i = 1, 2, ...$

$\rho_{i-1} = \hat{r}^T \, r^{(i-1)}$

if $\rho_{i-1} = 0$ η μέθοδος αποτυγχάνει

if $i = 1$

$u^{(1)} = r^{(0)}$

$p^{(1)} = u^{(1)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}}$

$u^{(i)} = r^{(i-1)} + \beta_{i-1} q^{(i)}$

$p^{(i)} = u^{(i)} + \beta_{i-1} \, (q^{(i-1)} + \beta_{i-1} \, p^{(i-1)})$

endif

Επίλυση του $\mathrm{M} \, \hat{p} = p^{(i)}$

$\hat{v} = A \, \hat{p}$

$\alpha_i = \frac{\rho_{i-1}}{\hat{r}^T \hat{v}}$

$q^{(i)} = u^{(i)} - \alpha_i \, \hat{v}$

Επίλυση του $M \, \hat{u} = u^{(i)} + q^{(i)}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{u}$

$r^{(i)} = r^{(i-1)} - \alpha_i \, A \, \hat{u}$

Έλεγχος σύγκλισης

end

Εικόνα 14 : Αλγόριθμος της μεθόδου CGS με preconditioning.

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ της λύσης $\boldsymbol{x}$

$r^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}$ (συνήθως $\hat{r} = r^{(0)}$)

for $i = 1, 2, ...$

$\rho_{i-1} = \hat{r}^T \, r^{(i-1)}$

if $\rho_{i-1} = 0$ η μέθοδος αποτυγχάνει

if $i = 1$

$p^{(1)} = r^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \, \frac{\alpha_{i-1}}{\omega_{i-1}}$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} \, (p^{(i-1)} - \omega_{i-1} \, v^{(i-1)})$

endif

Επίλυση του $\mathrm{M}\,\hat{p} = p^{(i)}$

$v^{(i)} = A\,\hat{p}$

$\alpha_i = \frac{\rho_{i-1}}{\hat{r}^T v^{(i)}}$

$s = r^{(i-1)} - \alpha_i \, v^{(i)}$

if $\| \, s \, \|$ αρκετά μικρό **then**

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{p}$ έξοδος

Επίλυση του $\mathrm{M}\,z = s \quad ; \quad t = A\,z$

$\omega_i = \frac{s^T t}{t^T t}$

$\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i \, \hat{p} + \omega_i \, z$

Έλεγχος σύγκλισης

if $\omega_i = 0$ έξοδος

$r^{(i)} = s - \omega_i \, t$

end

Εικόνα 15 : Αλγόριθμος της μεθόδου Bi-CGSTAB με preconditioning.

1.6 Κριτήρια Τερματισμού των Non-Stationary Επαναληπτικών Μεθόδων

Οι επαναληπτικές μέθοδοι κατά την επίλυση του γραμμικού συστήματος (1) παράγουν μια ακολουθία διανυσμάτων $\{\boldsymbol{x}^{(m)}\}$ προσεγγίσεων της πραγματικής λύσης $\boldsymbol{x}$ του συστήματος. Η μέθοδος θα πρέπει να σταματήσει την παραγωγή των προσεγγίσεων αυτών, όταν αποφασιστεί ότι έχει παραχθεί μια ικανοποιητική προσέγγιση. Τα κριτήρια τερματισμού είναι αυτά που ελέγχουν και αποφασίζουν σε κάθε επαναληπτικό βήμα αν χρειάζεται να συνεχιστεί η διαδικασία. Έτσι τα κριτήρια τερματισμού εξετάζουν τα παρακάτω τρια ενδεχόμενα σε κάθε βήμα

1. Έλεγχος αν $\| \boldsymbol{e}^{(m)} \| = \| \boldsymbol{x} - \boldsymbol{x}^{(m)} \|$ είναι αρκετά μικρό για τερματισμό της διαδικασίας

2. Έλεγχος ρυθμού σύγκλισης και τερματισμός αν η σύγκλιση είναι αργή ή αν η μέθοδος αποκλίνει

3. Έλεγχος αριθμού επαναληπτικών βημάτων και τερματισμός αν έχει υπερβεί η προκαθορισμένη μέγιστη τιμή τους.

Η ιδανική τιμή ελέγχου είναι το σφάλμα $\boldsymbol{e}^{(m)} = \boldsymbol{x} - \boldsymbol{x}^{(m)}$. Όμως ο ακριβής υπολογισμός του είναι αδύνατος κι έτσι χρησιμοποιείται το υπόλοιπο $\boldsymbol{r}^{(m)} = \boldsymbol{b} - A\,\boldsymbol{x}^{(m)}$ το οποίο συνήθως έχει ήδη υπολογιστεί. Στη συνέχεια παρουσιάζονται οι πέντε πιο διαδεδομένες εκφράσεις ελέγχου τερματισμού μιας non-stationary επαναληπτικής μεθόδου. Σε καθένα από τα παρακάτω κριτήρια καθορίζουμε μια σταθερά tol, μέσω της οποίας ουσιαστικά απαιτούμε να σταματήσει η επαναληπτική διαδικασία, όταν ικανοποιηθεί το ανάλογο κριτήριο ελέγχου.

- *Κριτήριο Τερματισμού 1* : Μπορούμε να βρούμε μια σχέση ανάμεσα στο σφάλμα $\boldsymbol{e}^{(m)}$ και στο υπόλοιπο $\boldsymbol{r}^{(m)}$ ως εξής

$$\boldsymbol{e}^{(m)} = \boldsymbol{x} - \boldsymbol{x}^{(m)} = A^{-1}(-A\,\boldsymbol{x}^{(m)} + \boldsymbol{b}) = A^{-1}\,\boldsymbol{r}^{(m)} \quad . \tag{61}$$

Έτσι μπορεί να δοθεί ένα φράγμα του υπολοίπου, το οποίο είναι ικανό να φράσει και το σφάλμα. Το πρώτο κριτήριο τερματισμού έχει τη μορφή

$$\| \boldsymbol{r}^{(m)} \| \leq \ S_1 \ \equiv \ tol \ (\| \ A \ \| \cdot \| \ \boldsymbol{x}^{(m)} \ \| \ + \ \| \ \boldsymbol{b} \ \|)$$

το οποίο είναι ισοδύναμο με

$$\| \boldsymbol{e}^{(m)} \| \leq \| \ A^{-1} \ \| \cdot \| \ \boldsymbol{r}^{(m)} \ \| \leq \ \ tol \ (\| \ A \ \| \cdot \| \ \boldsymbol{x}^{(m)} \ \| \ + \ \| \ \boldsymbol{b} \ \|) \quad .$$

- <u>*Κριτήριο Τερματισμού 2*</u> : Αυτό το κριτήριο τερματισμού δεν περιέχει την $\| \ A \ \|$ και έχει την μορφή

$$\| \boldsymbol{r}^{(m)} \| \leq \ S_2 \ \equiv \ tol \ \cdot \| \ \boldsymbol{b} \ \| \ .$$

Κάθε επαναληπτική μέθοδος θα έχει πρόβλημα ικανοποίησης του κριτήριου αυτού για την περίπτωση που $\| \ A \ \| \ \| \ \boldsymbol{x} \ \| \gg \| \ \boldsymbol{b} \ \|$, δηλαδή όταν ο πίνακας A είναι ill-conditioned και η λύση $\boldsymbol{x}$ βρίσκεται κοντά στο μηδενοχώρο του πίνακα A. Έτσι το σφάλμα θα εκφράζεται ως

$$\| \boldsymbol{e}^{(m)} \| \leq \| \ A^{-1} \ \| \cdot \| \ \boldsymbol{r}^{(m)} \ \| \leq \ \ tol \ \| \ A^{-1} \ \| \ \cdot \ \| \ \boldsymbol{b} \ \| \ .$$

- <u>*Κριτήριο Τερματισμού 3*</u> : Το κριτήριο αυτό χρησιμοποιείται όταν δεν είναι διαθέσιμη η $\| \ A^{-1} \ \|$ και έχει τη μορφή

$$\| \boldsymbol{r}^{(m)} \| \leq \ S_3 \ \equiv \ tol \ \frac{\| \ \boldsymbol{x}^{(m)} \ \|}{\| \ A^{-1} \ \|}.$$

Έτσι μπορεί να εκφραστεί το σφάλμα ως

$$\frac{\| \ \boldsymbol{e}^{(m)} \ \|}{\| \ \boldsymbol{x}^{(m)} \ \|} \ \leq \ \frac{\| \ A^{-1} \ \| \ \cdot \ \| \ \boldsymbol{r}^{(m)} \ \|}{\| \ \boldsymbol{x}^{(m)} \ \|} \ \leq \ tol$$

οπότε ο χρήστης καθορίζοντας το tol καθορίζει την σχετική ακρίβεια της προσεγγιστικής λύσης σύμφωνα με αυτό το κριτήριο.

- <u>*Κριτήριο Τερματισμού 4*</u> : Σύμφωνα με το κριτήριο αυτό ο χρήστης εκτός τη σταθερά tol θα πρέπει να ορίσει και ένα πίνακα E με μη αρνητικά στοιχεία

καθώς και ένα διάνυσμα f επίσης με μη αρνητικά στοιχεία. Έτσι το κριτήριο αυτό έχει τη μορφή $S_4 \equiv max_j \, (\frac{|\boldsymbol{r}^{(m)}|_j}{(E \cdot |\boldsymbol{x}^{(m)}| + f)_j}) \leq tol$ και η έκφραση του σφάλματος είναι

$$\| \boldsymbol{e}^{(m)} \|_\infty \leq \| \, | A^{-1} | \cdot | \boldsymbol{r}^{(m)} | \, \| \leq S_4 \cdot \| \, | A^{-1} | \, (E \, | \boldsymbol{x}^{(m)} | + f \,) \, \|_\infty$$

όπου $| A^{-1} |$ είναι ο A^{-1} με στοιχεία τις απολυτες τιμές τους.

- <u>*Κριτήριο Τερματισμού 5*</u> : Το κριτήριο αυτό είναι το πιο δημοφιλές, αλλά και το πιο απλό. Έχει τη μορφή $\| \boldsymbol{r}^{(m)} \| \leq S_5 \equiv tol \cdot \| \boldsymbol{r}^{(0)} \|$. Το μειονέκτημά του είναι η ισχυρή εξάρτησή του από την αρχική προσέγγιση της λύσης $\boldsymbol{x}^{(0)}$ και αν έχει επιλεγεί το δεύτερο μέλος του γραμμικού συστήματος, τότε το κριτήριο είναι ισοδύναμο με το δεύτερο κριτήριο που αναφέρθηκε παραπάνω. Η αντίστοιχη έκφραση του σφάλματος είναι

$$\| \boldsymbol{e}^{(m)} \| \leq S_5 \cdot \| A^{-1} \| \, .$$

Στο σημείο αυτό θα πρέπει να σημειωθεί, ότι τα παραπάνω κριτήρια τερματισμού αφορούν την επίλυση γενικών γραμμικών συστημάτων $A\boldsymbol{x} = \boldsymbol{b}$ χωρίς preconditioning. Για τις περιπτώσεις preconditioning και ανάλογα με το είδος του, θα πρέπει να λαμβάνεται υπόψη η δράση του preconditioner πίνακα M στις εκφράσεις των σφαλμάτων και υπολοίπων.

2 ΜΕΘΟΔΟΙ ΑΡΙΘΜΗΤΙΚΗΣ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΣΥΝΟΡΙΑΚΩΝ ΤΙΜΩΝ ΕΛΛΕΙΠΤΙΚΟΥ ΤΥΠΟΥ

2.1 Εισαγωγή

Είναι γνωστό ότι η γενική μορφή Προβλημάτων Συνοριακών Τιμών (ΠΣΤ) περιγράφεται από τις σχέσεις

$$(\Pi\Sigma\mathrm{T}) \quad \begin{cases} \mathcal{L}u(x,y) \;=\; f(x,y) \;\;,\;\; (x,y) \in \Omega \\ \\ \mathcal{B}u(x,y) \;=\; g(x,y) \;\;,\;\; (x,y) \in \partial\Omega \end{cases} \tag{62}$$

όπου οι διαφορετικοί τελεστές $\mathcal{L}$ και $\mathcal{B}$ ορίζονται αντίστοιχα σαν

$$\begin{cases} \mathcal{L} \;\equiv\; a(x,y)\frac{\partial^2}{\partial x^2} + 2b(x,y)\frac{\partial^2}{\partial x \partial y} + c(x,y)\frac{\partial^2}{\partial y^2} + d(x,y)\frac{\partial}{\partial x} + e(x,y)\frac{\partial}{\partial y} + h(x,y) \\ \\ \mathcal{B} \;\equiv\; \alpha(x,y) + \beta(x,y)\frac{\partial}{\partial \vec{n}} \end{cases} \tag{63}$$

Η συνθήκη $a(x,y)c(x,y) > b^2(x,y)$ χαρακτηρίζει τον **ελλειπτικό** τύπο του προβλήματος και συνεπάγεται ότι οι συναρτήσεις $a(x,y)$ και $c(x,y)$ είναι ομόσημες και μη μηδενικές.

Από τα κλασσικότερα παραδείγματα ελλειπτικών ΠΣΤ είναι και το Poisson-Dirichlet πρόβλημα που ορίζεται σαν

$$\begin{cases} u_{xx}(x,y) + u_{yy}(x,y) \;=\; f(x,y) \;\;,\;\; (x,y) \in \Omega \equiv [0,1] \times [0,1] \\ \\ u(x,y) \;=\; g(x,y) \;\;,\;\; (x,y) \in \partial\Omega \end{cases} \tag{64}$$

Το Poisson-Dirichlet πρόβλημα αναφέρεται στην βιβλιογραφία ως το πρόβλημα μοντέλο ελλειπτικών ΠΣΤ για την ανάπτυξη και μελέτη αριθμητικών μεθόδων και ως τέτοιο θα χρησιμοποιηθεί στην παρούσα μελέτη.

Ανάμεσα στις περισσότερο διαδεδομένες μεθόδους αριθμητικής επίλυσης ΠΣΤ είναι και αυτές των Πεπερασμένων Στοιχείων, οι οποίες περιγράφονται σχηματικά με τα παρακάτω βήματα :

- **Βήμα 0:** Γεωμετρική Διαμέριση του πεδίου Ω σε πεπερασμένο πλήθος στοιχείων.

- **Βήμα 1:** Επιλογή n γραμμικά ανεξάρτητων κατά τμήματα συνεχών πολυωνυμικών συναρτήσεων $\Phi_1, \ldots, \Phi_n$, οι οποίες ονομάζονται συναρτήσεις **βάσης** και χρησιμοποιούνται στη προσέγγιση της πραγματικής λύσης $u(x, y)$ του ΠΣΤ ως εξής

$$u(x,y) \simeq u_n(x,y) = a_1\Phi_1(x,y) + \ldots + a_n\Phi_n(x,y) = \sum_{k=1}^{n} a_k\Phi_k(x,y). \quad (65)$$

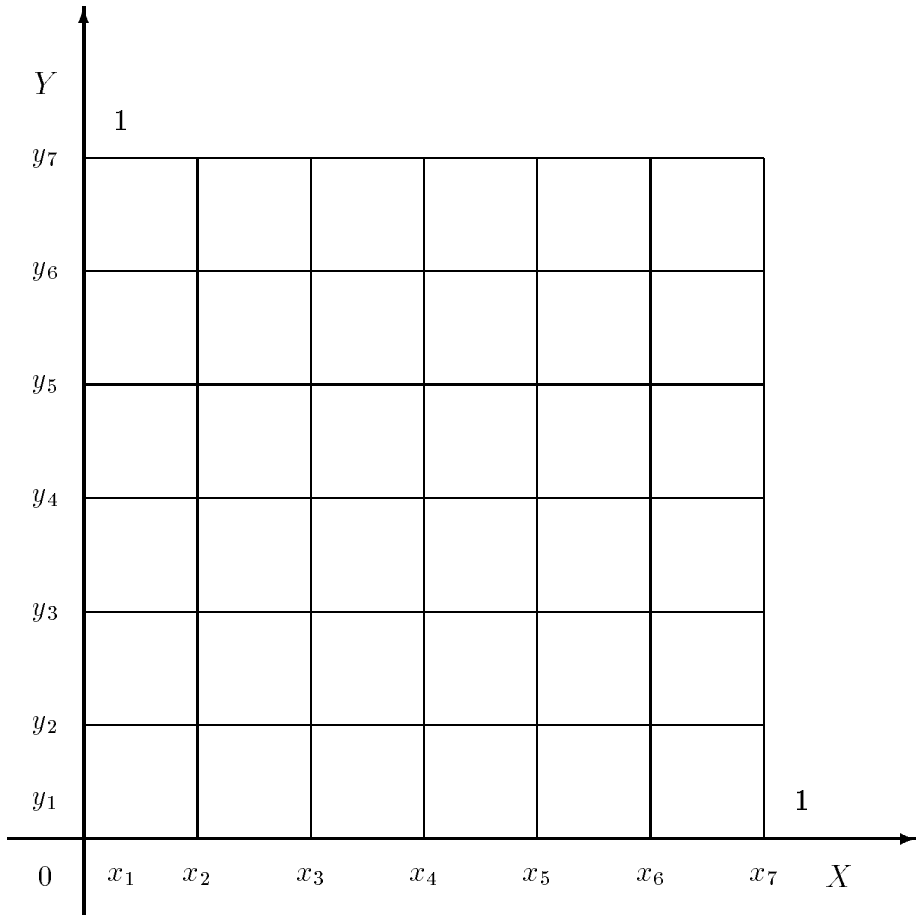
- **Βήμα 2:** Επιλογή μεθόδου διακριτοποίησης (Galerkin, Rayleigh-Ritz, Least Squares, Collocation) [32, 9] για μετάβαση από το **συνεχή** χώρο στο **διακριτό** δηλαδή μετατροπή του ΠΣΤ σε ένα γραμμικό σύστημα

$$C\vec{\mathbf{a}} = \vec{\mathbf{b}} \quad (66)$$

όπου $C \in \mathbb{R}^{n,n}$, $\vec{\mathbf{a}} = [a_1 \;\; a_2 \;\; \cdots \;\; a_n]^T$ και $\vec{\mathbf{b}} = [b_1 \;\; b_2 \;\; \cdots \;\; b_n]^T$.

- **Βήμα 3:** Επιλογή μεθόδου για την επίλυση του γραμμικού συστήματος και προσδιορισμός των αγνώστων $a_1, \ldots, a_n$. Η επιλογή αυτή βασίζεται στο μέγεθος και στις ιδιότητες του παραγόμενου γραμμικού συστήματος.

Διαφορετικές επιλογές σε κάθε ένα από τα παραπάνω βήματα συνεπάγονται και διαφορετικές μεθόδους για την επίλυση ΠΣΤ. Στην παρούσα εργασία το προς μελέτη πρόβλημα χαρακτηρίζεται από τις ακόλουθες συγκεκριμένες επιλογές :



Εικόνα 16 : Γεωμετρική απεικόνιση της διαμέρισης του πεδίου Ω

- **Βήμα 0:** Θεωρούμε ομοιόμορφο διαμερισμό των διαστημάτων $I^x = I^y = [0,1]$ σε n_s υποδιαστήματα $I^x_m = I^y_m$, $m = 1$, ... , n_s τα οποία παράγουν ένα ομοιόμορφο πλέγμα με βήμα διακριτοποίησης $h = \frac{1}{n_s}$ και συντεταγμένες κόμβων (x_i, y_j), όπου $x_i = (i-1)h$ και $y_j = (j-1)h$, με $i, j = 1$, ... , $(n_s + 1)$. Η Εικόνα 16 εμφανίζει την διαμέριση του Ω για $n_s = 6$.

- **Βήμα 1:** Ως συναρτήσεις βάσεις επιλέγονται τα Hermite Bicubic πολυώνυμα [91], με γενική μορφή $\Phi_k(x,y) = \Phi_i(x)\Phi_j(y)$, και η συνάρτηση $u(x,y)$ προσεγγίζεται από την

$$u(x,y) \simeq u_n(x,y) = \sum_{i=1}^{\tilde{n}} \sum_{j=1}^{\tilde{n}} a_{i,j} \Phi_i(x) \Phi_j(y) \tag{67}$$

όπου $\tilde{n} = 2(n_s + 1)$.

- **Βήμα 2:** Ως μέθοδος διακριτοποίησης επιλέγεται η μέθοδος της **Collocation** [16, 63, 61, 62], η οποία κατασκευάζει το γραμμικό σύστημα $C\vec{\mathbf{a}} = \vec{\mathbf{b}}$ απαιτώντας οι συνθήκες $\mathcal{L}u_n - f = 0$ και $\mathcal{B}u_n - g = 0$ να ισχύουν για n καθορισμένα εσωτερικά και συνοριακά collocation σημεία και ως τέτοια επιλέγονται τα σημεία Gauss.

- **Βήμα 3:** Για την επίλυση του γραμμικού συστήματος $C\vec{\mathbf{a}} = \vec{\mathbf{b}}$ επιλέγεται κάποια επαναληπτική μέθοδος.

Στις παραγράφους που ακολουθούν παρουσιάζεται περισσότερο αναλυτικά η τεχνική περιγραφή των παραπάνω επιλογών.

2.2 Πολυώνυμα Hermite

Τα τμηματικά κυβικά πολυώνυμα Hermite ορίζονται ως εξής :

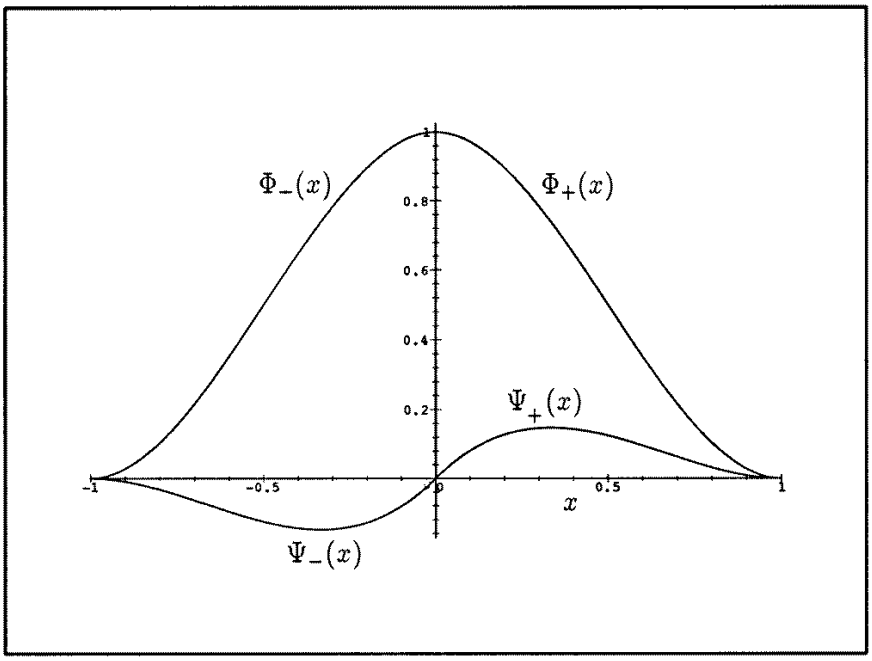
$$\Phi(x) \doteq \begin{cases} \Phi_+(x) & , \quad x \in [0,1] \\ \Phi_-(x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,1] \end{cases} , \tag{68}$$

$$\Psi(x) \doteq \begin{cases} \Psi_+(x) & , \quad x \in [0,1] \\ \Psi_-(x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,1] \end{cases} , \tag{69}$$

όπου

$$\Phi_+(x) \doteq \begin{cases} (1-x)^2(1+2x) & , \quad x \in [0,1] \\ 0 & , \quad x \notin [0,1] \end{cases} , \tag{70}$$

$$\Phi_-(x) = \Phi_+(-x) \doteq \begin{cases} (1+x)^2(1-2x) & , \quad x \in [-1,0] \\ 0 & , \quad x \notin [-1,0] \end{cases} , \tag{71}$$



Εικόνα 17 : Κυβικά πολυώνυμα Hermite

$$\Psi_+(x) \doteq \begin{cases} x(1-x)^2 & , \quad x \in [0,1] \\ \\ 0 & , \quad x \notin [0,1] \end{cases} , \tag{72}$$

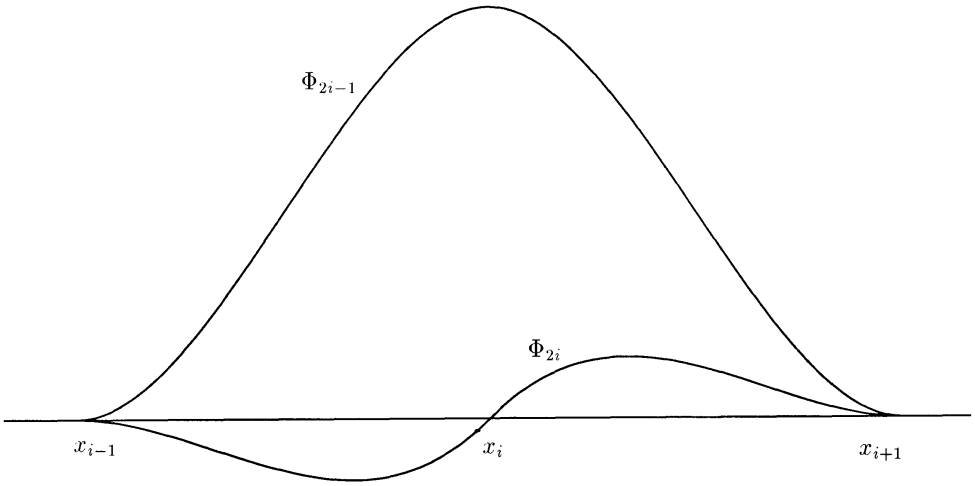
$$\Psi_-(x) = -\Psi_+(-x) \doteq \begin{cases} x(1+x)^2 & , \quad x \in [-1,0] \\ \\ 0 & , \quad x \notin [-1,0] \end{cases} . \tag{73}$$

Η Εικόνα 17 εμφανίζει τα κυβικά πολυώνυμα Hermite, όπως αυτά ορίζονται στο $[-1,1]$.

Σε κάθε κόμβο x_m αντιστοιχούν δύο συναρτήσεις και ορίζονται ως εξής:

$$\Phi_{2m-1}(x) \doteq \begin{cases} \Phi\left(\frac{x-x_m}{h}\right) & , \quad x \in I_{m-1}^x \cup I_m^x \\ \\ 0 & , \quad \text{διαφορετικά} \end{cases} , \tag{74}$$

$$\Phi_{2m}(x) \doteq \begin{cases} \Psi\left(\frac{x-x_m}{h}\right) & , \quad x \in I_m^x \cup I_{m-1}^x \\ \\ 0 & , \quad \text{διαφορετικά} \end{cases} , \tag{75}$$



Εικόνα 18 : Πολυώνυμα Hermite ορισμένα στον κόμβο x_i.

όπου $m = 1, \ldots, (n_s + 1)$, $I_i^x \equiv [x_i, x_{i+1}]$, $i = 1, \ldots, n_s$.

Για να ισχύουν οι ορισμοί (13) και (14) για $m = 1$ και $m = n_s + 1$, θεωρούμε δύο υποθετικούς κόμβους $x_0 = -h$ και $x_{n_s+2} := 1 + h$.

Στην Εικόνα 18 εμφανίζεται ένας τυχαίος κόμβος x_i και παρουσιάζονται οι αντίστοιχες συναρτήσεις όπως αυτές ορίζονται στο κόμβο αυτόν.

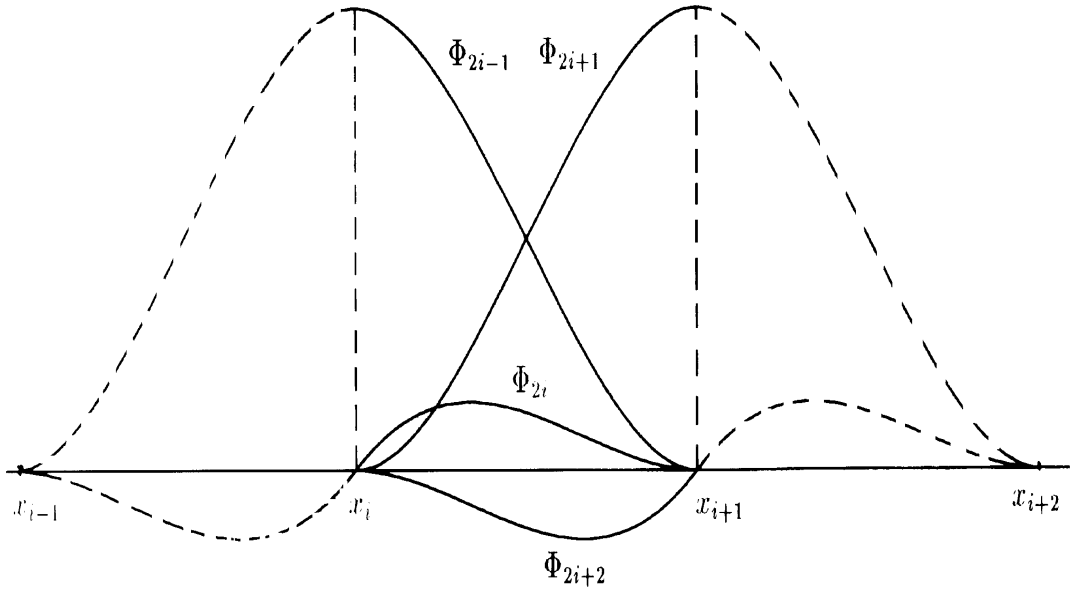
Παρατηρούμε ότι ισχύουν οι εξής ιδιότητες

$$\begin{cases} \Phi_{2m-1}(x_i) = h\frac{d}{dt}\Phi_{2m}(x_i) = \delta_m^i \\ \Phi_{2m}(x_i) = \frac{d}{dt}\Phi_{2m-1}(x_i) = 0 \end{cases} \tag{76}$$

για όλα τα $m = 1, \ldots, (n_s + 1)$ και όπου δ_m^i : Δέλτα του Kronecker.

Επιπλέον σημειώνουμε ότι σε κάθε υποδιάστημα I_i^x διέρχονται τέσσερα μόνο μη μηδενικά πολυώνυμα Hermite με δείκτες : Φ_{2i-1} , Φ_{2i} , Φ_{2i+1} και Φ_{2i+2}.

Το γεγονός αυτό επιδεικνύεται σχηματικά και στην Εικόνα 19.



Εικόνα 19 : Μη μηδενικά Πολυώνυμα Hermite στο υποδιάστημα $[x_i, x_{i+1}]$.

Βασισμένοι στις παραπάνω ιδιότητες των πολυωνύμων Hermite εύκολα προκύπτουν και οι ιδιότητες των διδιάστατων bicubic πολυωνύμων Hermite. Έτσι, παρατηρούμε ότι σε κάθε διδιάστατο κόμβο (x_i, y_j) ορίζονται τα παρακάτω τέσσερα Hermite Bicubic πολυώνυμα :

$$\begin{cases} \Phi_{2i-1,2j-1}(x,y) &= \Phi_{2i-1}(x)\Phi_{2j-1}(y) \\ \Phi_{2i-1,2j}(x,y) &= \Phi_{2i-1}(x)\Phi_{2j}(y) \\ \Phi_{2i,2j-1}(x,y) &= \Phi_{2i}(x)\Phi_{2j-1}(y) \\ \Phi_{2i,2j}(x,y) &= \Phi_{2i}(x)\Phi_{2j}(y) \end{cases} \tag{77}$$

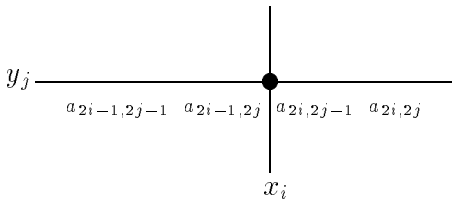
με τις εξής ιδιότητες :

$$\begin{cases} \Phi_{2i-1,2j-1}(x_i,y_j) = 1 \quad , \quad h\frac{\partial}{\partial y}\Phi_{2i-1,2j}(x_i,y_j) = 1 \\ h\frac{\partial}{\partial x}\Phi_{2i,2j-1}(x_i,y_j) = 1 \quad , \quad h^2\frac{\partial^2}{\partial x \partial y}\Phi_{2i,2j}(x_i,y_j) = 1 \end{cases} \tag{78}$$

Σαν άμεση συνέπεια των προηγουμένων σχέσεων προκύπτει ότι

$$\begin{cases} u_n(x_i,y_j) & = & a_{2i-1,2j-1} & , & h\frac{\partial}{\partial y}u_n(x_i,y_j) & = & a_{2i-1,2j} \\ h\frac{\partial}{\partial x}u_n(x_i,y_j) & = & a_{2i,2j-1} & , & h^2\frac{\partial^2}{\partial x\partial y}u_n(x_i,y_j) & = & a_{2i,2j} \end{cases} \tag{79}$$

όπου $i\ ,\ j\ =\ 1,\ldots,(n_s+1)$ και στο παρακάτω σχήμα εμφανίζονται οι τέσσερις άγνωστοι όπως αυτοί αντιστοιχούν στον κόμβο (x_i,y_i).



Επιπλέον παρατηρούμε ότι σε κάθε στοιχείο I_{ij}^{xy} αντιστοιχούν 16 μη μηδενικές συναρτήσεις βάσης και επομένως για $(x,y)\in I_{ij}^{xy}$ ισχύει ότι :

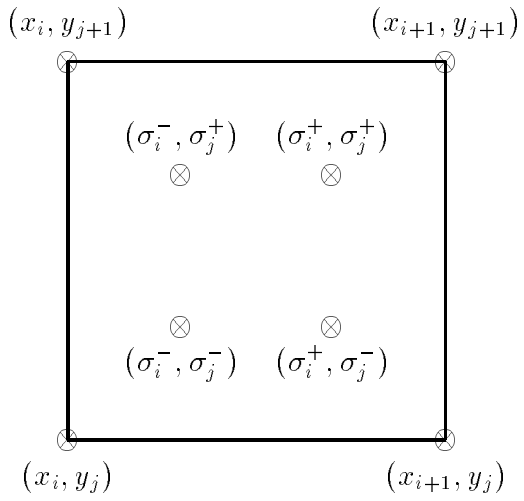
$$u_n(x,y)\ =\ \sum_{k=2i-1}^{2i+2}\sum_{l=2j-1}^{2j+2}\alpha_{k,l}\Phi_k(x)\Phi_l(y)\ . \tag{80}$$

Γι' αυτό το λόγο το στοιχείο I_{ij}^{xy} είναι στοιχείο με 16 βαθμούς ελευθερίας.

2.3 Η μέθοδος Collocation

Οι Collocation εξισώσεις κατασκευάζονται απαιτώντας το υπόλοιπο $\mathcal{L}u_n - f$ να μηδενίζεται σε $n_I = 4n_s^2$ **εσωτερικά** collocation σημεία και το υπόλοιπο $\mathcal{B}u_n - g$ να μηδενίζεται σε $n_b = 4(2n_s+1)$ **συνοριακά** collocation σημεία. Παρατηρούμε ότι το πλήθος $n_I + n_b$ των collocation εξισώσεων ισούται με τον αριθμό των αγνώστων, ο οποίος προκύπτει από τη χρήση των πολυωνύμων Hermite Bicubic [15].

Πρέπει να τονίσουμε εδώ, ότι στην περίπτωση των Dirichlet συνοριακών συνθηκών κάποιοι από τους αγνώστους, οι οποίοι βρίσκονται πάνω στο σύνορο του Ω προσδιορίζονται άμεσα από αυτές, με αποτέλεσμα την απαλοιφή τους από το γραμμικό σύστημα. Το πλήθος των αγνώστων αυτών είναι $4(2n_s+1)$ με αποτέλεσμα



Εικόνα 20 : Τα τέσσερα σημεία Gauss στο πεπερασμένο στοιχείο I_{ij}^{xy}

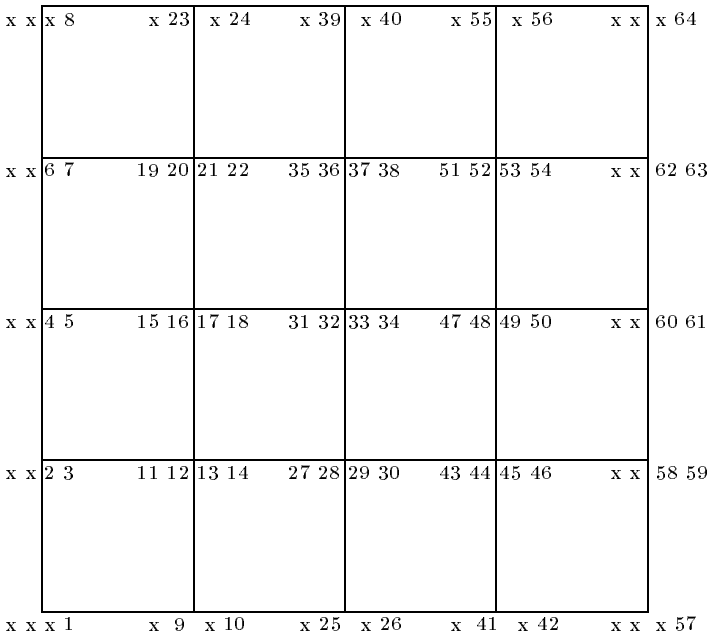
να μην χρειάζεται η επιλογή συνοριακών collocation σημείων.

Οπότε για την Dirichlet περίπτωση το πλήθος των αγνώστων και συνεπώς και των εσωτερικών collocation σημείων ισούται με $n = 4n_s^2$.

Στην περίπτωση ελλειπτικών ΠΣΤ η standard επιλογή εσωτερικών collocation σημείων είναι αυτή των σημείων Gauss [16] με συντεταγμένες $(\sigma_i^{\pm}, \sigma_j^{\pm})$, όπου για κάθε $i = 1, \ldots, n_s$ ισχύει ότι $\sigma_i^{\pm} = 2i - 1 \pm \frac{\sqrt{3}}{3}$. Στην Εικόνα 20 εμφανίζονται τα τέσσερα σημεία Gauss του στοιχείου I_{ij}^{xy}.

Η αρίθμηση αγνώστων και εξισώσεων καθορίζει την δομή του collocation πίνακα, και κατά συνέπεια την επιλογή της μεθόδου επίλυσης του παραγόμενου γραμμικού συστήματος. Η Εικόνα 21 παρουσιάζει την αρίθμηση αγνώστων και η Εικόνα 22 την αρίθμηση των εξισώσεων σύμφωνα με την standard collocation μέθοδο. Οι άγνωστοι, οι οποίοι έχουν απαλοιφθεί εξαιτίας των συνοριακών συνθηκών σημειώνονται με "x" .

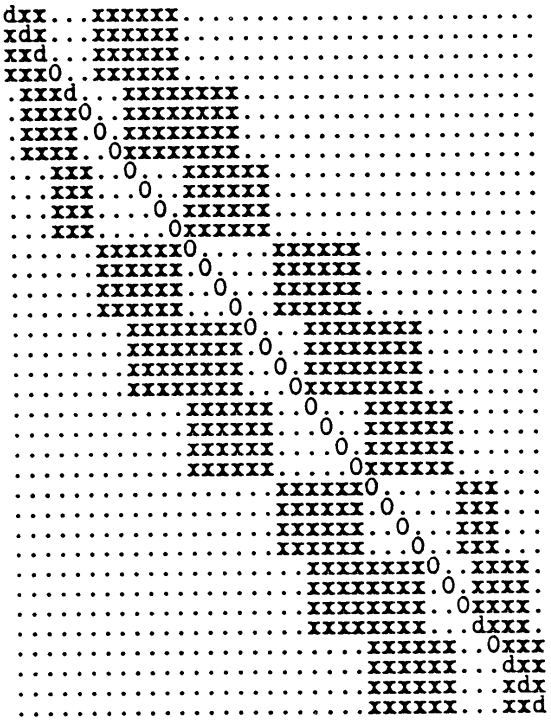
Η δομή του standard collocation πίνακα εμφανίζεται στην Εικόνα 23. Εύκολα συνάγεται ότι η δομή αυτή δεν είναι κατάλληλη για την χρήση επαναληπτικών μεθόδων. Στην εργασία [92] ο Θ. Σ. Παπαθεοδώρου πρότεινε την αρίθμηση των αγνώστων, η οποία παρουσιάζεται στην Εικόνα 25, και των εξισώσεων, η οποία



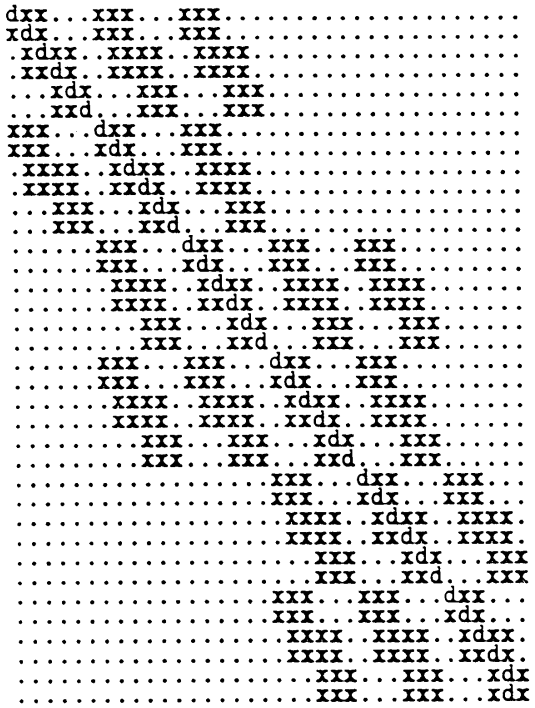
Εικόνα 21 : Standard Αρίθμηση αγνώστων για $n_s = 4$

14	16	30	32	46	48	62	64
13	15	29	31	45	47	61	63
10	12	26	28	42	44	58	60
9	11	25	27	41	43	57	59
6	8	22	24	38	40	54	56
5	7	21	23	37	39	53	55
2	4	18	20	34	36	50	52
1	3	17	19	33	35	49	51

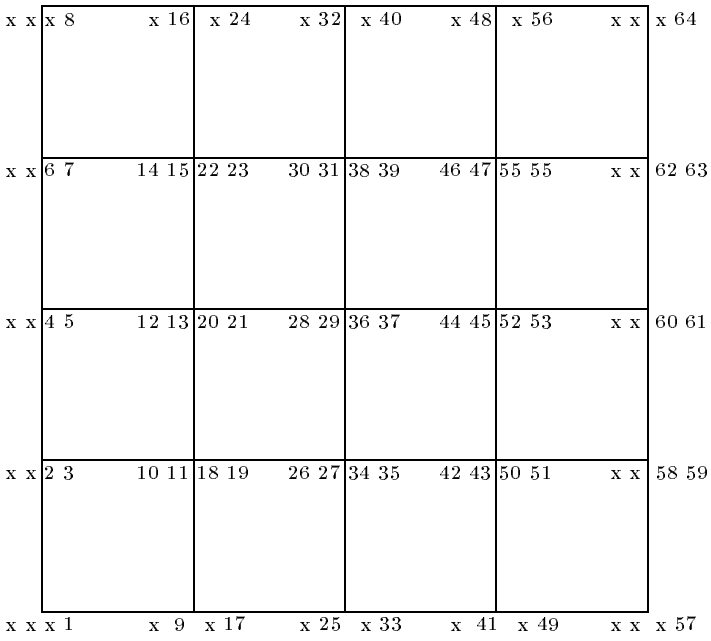
Εικόνα 22 : Standard Αρίθμηση εξισώσεων για $n_s = 4$



Εικόνα 23 : Δομή του Standard Collocation Πίνακα για $n_s = 3$



Εικόνα 24 : Δομή του Block Τριδιαγώνιου Collocation Πίνακα για $n_s = 3$



Εικόνα 25 : Block Τριδιαγώνια Αρίθμηση αγνώστων για $n_s = 4$

8	16	24	32	40	48	56	64
7	15	23	31	39	47	55	63
6	14	22	30	38	46	54	62
5	13	21	29	37	45	53	61
4	12	20	28	36	44	52	60
3	11	19	27	35	43	51	59
2	10	18	26	34	42	50	58
1	9	17	25	33	41	49	57

Εικόνα 26 : Block Τριδιαγώνια Αρίθμηση εξισώσεων για $n_s = 4$

παρουσιάζεται στην Εικόνα 26, ώστε να προκύψει ο collocation block τριδιαγώνιος πίνακας του οποίου η δομή εμφανίζεται στην Εικόνα 24.

Κάνοντας χρήση της μεθόδου αρίθμησης του Παπαθεοδώρου στο Dirichlet - Poisson πρόβλημα παράγεται το ακόλουθο γραμμικό σύστημα

$$A\boldsymbol{x} \;=\; \boldsymbol{b} \;\;, \tag{81}$$

όπου $A \in \mathbb{R}^{n,n} \;\; (n = 4{n_s}^2)$ είναι ο Collocation πίνακας συντελεστών και

$\boldsymbol{x} \;=\; [x_1 \;\; x_2 \;\; \cdots \;\; x_n]^T \;=\; [\alpha_{1,1} \;\; \cdots \;\; \alpha_{2n_s,2n_s}]^T$ είναι το διάνυσμα των αγνώστων.

Πιο συγκεκριμένα, θεωρώντας ότι έχει βγει κοινός παράγοντας το $1/9h^2$, αρχικά ορίζομε τους $2n_s \times 2n_s \;\; seed$-βασικούς πίνακες

$$A_i \;=\; \begin{bmatrix} a_2 & a_3 & -a_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ a_4 & a_1 & -a_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & -a_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & a_4 & a_1 & -a_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_1 & a_2 & a_3 & -a_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_3 & a_4 & a_1 & -a_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_1 & a_2 & -a_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_3 & a_4 & -a_2 \end{bmatrix} \;\;, \;\; i = 1,2,3,4 \tag{82}$$

όπου οι τιμές a_i's δίνονται ως[90, 92]

Table 1	a_1	a_2	a_3	a_4
A_1	$-r^+$	$-s^+$	q	t^+
A_2	$-s^+$	$-u^+$	t^-	0
A_3	q	t^-	$-r^-$	$-s^-$
A_4	t^+	0	$-s^-$	$-u^-$

(83)

με $q = 24$, $r^{\pm} = 24 \pm 18\sqrt{3}$, $s^{\pm} = 12 \pm 8\sqrt{3}$, $t^{\pm} = 3 \pm \sqrt{3}$, $u^{\pm} = 3 \pm 2\sqrt{3}$.

Κάνοντας χρήση των παραπάνω ορισμών ο block τριδιαγώνιος collocation πίνακας ορίζεται ως εξής

$$A = \begin{bmatrix} A_2 & A_3 & -A_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ A_4 & A_1 & -A_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & A_1 & A_2 & A_3 & -A_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & A_3 & A_4 & A_1 & -A_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & A_1 & A_2 & A_3 & -A_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & A_3 & A_4 & A_1 & -A_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_1 & A_2 & -A_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_3 & A_4 & -A_2 \end{bmatrix} . \quad (84)$$

Είναι γνωστό ότι για block τριδιαγώνιους πίνακες μπορεί κανείς να βρει πλούσια βιβλιογραφία [58, 72, 76, 77, 92, 106, 112, 125, 79] για την εφαρμογή και ανάλυση επαναληπτικών μεθόδων.

3 ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΠΙΛΥΣΗ ΤΩΝ COLLOCATION ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

3.1 Εισαγωγή

Η μέθοδος collocation βασισμένη στα πολυώνυμα βάσης Hermite bi-cubic έχει αποδειχτεί ως μια από τις ισχυρότερες και αποδοτικότερες αριθμητικές μεθόδους επίλυσης διαφορικών εξισώσεων σε δυο διαστάσεις με μερικές παραγώγους. Το παραγόμενο γραμμικό σύστημα, όπως φάνηκε στο προηγούμενο καφάλαιο, έχει block τριδιαγώνια μορφή, δεν είναι συμμετρικό αλλά και ούτε διαγωνίως ενισχυμένο ή θετικά ορισμένο κάνοντας χρήση της ανάλογης αρίθμησης εξίσωσεων και αγνώστων, όπως αυτή προτάθηκε από τον Θ. Παπαθεοδώρου [92]. Λόγω του μεγέθους του, αλλά της αραιότητας του οι γνωστές άμεσες μέθοδοι, έστω και σε μορφή ζώνης, δεν είναι σε θέση να εφαρμοστούν για ικανοποιητικές διαμερίσεις [72]. Έτσι είμαστε υποχρεωμένοι να χρησιμοποιήσουμε κάποια stationary ή non-stationary επαναληπτική μέθοδο και μάλιστα block μορφής, αφού οι point επαναληπτικές μέθοδοι της μαθηματικής βιβλιοθήκης ITPACK [70] συνήθως δεν συγκλίνουν ακόμα και για

τα προβλήματα των μοντέλων. Η ανάπτυξη αποδοτικών επαναληπτικών μεθόδων, οι οποίες θα εκμεταλλεύονται τις ιδιότητες των collocation γραμμικών συστημάτων είναι απαραίτητη για τη μετάβαση στην επίλυση των τριδιάστατων διαφορικών εξισώσεων, αλλά και για την εφαρμογή τους σε παράλληλα περιβάλλοντα. Αντίθετα οι άμεσες μέθοδοι δεν είναι σε θέση να επιλύουν το γραμμικό σύστημα για μικρό βήμα διακριτοποίησης και η παραλληλοποίησή τους εμφανίζει σημαντικές δυσκολίες.

Αρχικά στην εργασία [92] ο Θ. Παπαθεοδώρου πρότεινε την block AOR επαναληπτική μέθοδο και αργότερα μελετήθηκαν ως προς τη σύγκλιση μέθοδοι τύπου SOR στις [58, 105]. Στην [72] χρησιμοποιήθηκαν διάφορες stationary επαναληπτικές μέθοδοι βασισμένες σε τρεις διαμερίσεις του collocation πίνακα και η βέλτιστη SOR εμφάνισε ταχύτερη σύγκλιση για μια από αυτές τις διαμερίσεις.

Στη συνέχεια παρουσιάζεται η επίλυση του collocation γραμμικού συστήματος κάνοντας χρήση των πιο γνωστών non-stationary μεθόδων, ενώ ταυτόχρονα γίνεται αναζήτηση του καταλληλότερου precoditioner. Τα αποτελέσματα και οι πειραματικές μετρήσεις της συμπεριφοράς της απόδοσης κάθε μεθόδου μπορούν να δικαιολογηθούν και ταυτίζονται απόλυτα με την ανάλυση των τιμών Ritz του collocation πίνακα. Τέλος γίνονται συγκριτικές δοκιμές απόδοσης ανάμεσα στις αποδοτικότερες stationary και non-stationary επαναληπτικές μεθόδους.

3.2 Διαμέριση του πίνακα των συντελεστών των αγνώστων

Το βασικότερο βήμα κατά την εφαρμογή μιας επαναληπτικής μεθόδου της μορφής (2) είναι η διαμέριση του πίνακα των συντελεστών των αγνώστων. Αυτό συμβαίνει, γιατί από τον τρόπο διαμέρισης του πίνακα, καθορίζονται οι βασικές ιδιότητες της επαναληπτικής μεθόδου, όπως αν είναι completely consistent και ο ρυθμός σύγκλισής της. Έτσι για τον τριδιαγώνιο collocation πίνακα A προτάθηκε από τον Θ. Σ. Παπαθεοδώρου [92] η παρακάτω block διαμέριση

$$
A \;=\; \left[\begin{array}{cc|cc|ccc|cc|cc}
A_2 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\
A_4 & A_1 & -A_2 & O & O & \cdots & O & O & O & O & O \\
\hline
O & A_1 & A_2 & A_3 & -A_4 & \cdots & O & O & O & O & O \\
O & A_3 & A_4 & A_1 & -A_2 & \cdots & O & O & O & O & O \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
O & O & O & O & O & \cdots & A_1 & A_2 & A_3 & -A_4 & O \\
O & O & O & O & O & \cdots & A_3 & A_4 & A_1 & -A_2 & O \\
\hline
O & O & O & O & O & \cdots & O & O & A_1 & A_2 & -A_4 \\
O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_2
\end{array}\right]
$$

με $A = D_A - L_A - U_A$, όπου

$$
D_A \;=\; diag[D \quad D \quad \ldots \quad D \quad \hat{D}] \quad , \tag{85}
$$

$$
L_A \;=\; -\left[\begin{array}{ccccccc}
O & O & O & \ldots & O & O & O \\
L & O & O & \ldots & O & O & O \\
O & L & O & \ldots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \ldots & L & O & O \\
O & O & O & \ldots & O & L & O
\end{array}\right] \quad , \tag{86}
$$

$$
U_A \;=\; -\left[\begin{array}{ccccccc}
O & U & O & \ldots & O & O & O \\
O & O & U & \ldots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \ldots & O & U & O \\
O & O & O & \ldots & O & O & U \\
O & O & O & \ldots & O & O & O
\end{array}\right] \tag{87}
$$

με

$$
D \;=\; \left[\begin{array}{cc} A_2 & A_3 \\ A_4 & A_1 \end{array}\right] \quad , \quad \hat{D} \;=\; \left[\begin{array}{cc} A_2 & -A_4 \\ A_4 & -A_2 \end{array}\right] \quad ,
$$

$$L = \begin{bmatrix} O & A_1 \\ O & A_3 \end{bmatrix} \quad , \quad U = \begin{bmatrix} -A_4 & O \\ -A_2 & O \end{bmatrix} \quad .$$

Στην εργασία [92] γίνεται ανάλυση της σύγκλισης της επαληπτικής μεθόδου AOR και ο καθορισμός των βέλτιστων τιμών των παραμέτρων της. Όμως στην πρόσφατη εργασία των Lai, Hadjidimos, Houstis και Rice [72] διερευνήθηκε η αποτελεσματικότητα διαφορετικών διαμερίσεων του πίνακα A και απεδείχθει, ότι με τη διαμέριση

$$\left[\begin{array}{c|ccc|cc|cc|c} A_2 & A_3 & -A_4 & O & O & \cdots & O & O & O & O \\ \hline A_4 & A_1 & -A_2 & O & O & \cdots & O & O & O & O \\ O & A_1 & A_2 & A_3 & -A_4 & \cdots & O & O & O & O \\ \hline O & A_3 & A_4 & A_1 & -A_2 & \cdots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & A_1 & A_2 & A_3 & -A_4 \\ \hline O & O & O & O & O & \cdots & A_3 & A_4 & A_1 & -A_2 \\ O & O & O & O & O & \cdots & O & O & A_1 & A_2 \\ \hline O & O & O & O & O & \cdots & O & O & A_3 & A_4 \end{array}\right] \quad \left[\begin{array}{c} -A_2 \\ \hline O \\ O \\ \hline O \\ \vdots \\ O \\ \hline O \\ -A_4 \\ \hline -A_2 \end{array}\right]$$

επιτυγχάνεται πολύ μεγαλύτερη ταχύτητα σύγκλισης αν χρησιμοποιηθεί η SOR με βέλτιστο ω. Επίσης στην ίδια εργασία εμφανίζονται πειραματικά αποτελέσματα της GMRES(50) με block Jacobi precoditioning και προκύπτει, ότι η βέλτιστη SOR μπορεί να την ανταγωνιστεί ικανοποιητικά τουλάχιστον για διακριτοποιήσεις με μεγάλο μέγεθος βήματος.

Με βάση την παραπάνω διαμέριση οι πίνακες διάσπασης D_A, L_A και U_A ορίζονται από τις σχέσεις

$$D_A = diag[A_2 \quad Q \quad \ldots \quad Q \quad -A_2] \quad , \tag{88}$$

όπου $Q = \begin{bmatrix} A_1 & -A_2 \\ A_1 & A_2 \end{bmatrix}$ και

$$L_A = -\begin{bmatrix} O & O & O & \dots & O & O & O & O & O \\ A_4 & O & O & \dots & O & O & O & O & O \\ O & O & O & \dots & O & O & O & O & O \\ O & A_3 & A_4 & \dots & O & O & O & O & O \\ O & O & O & \dots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & \dots & A_3 & A_4 & O & O & O \\ O & O & O & \dots & O & O & O & O & O \\ O & O & O & \dots & O & O & A_3 & A_4 & O \end{bmatrix}, \tag{89}$$

$$U_A = -\begin{bmatrix} O & A_3 & -A_4 & O & O & \dots & O & O & O \\ O & O & O & O & O & \dots & O & O & O \\ O & O & O & A_3 & -A_4 & \dots & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \dots & O & O & O \\ O & O & O & O & O & \dots & A_3 & -A_4 & O \\ O & O & O & O & O & \dots & O & O & O \\ O & O & O & O & O & \dots & O & O & -A_4 \\ O & O & O & O & O & \dots & O & O & O \end{bmatrix} \tag{90}$$

Στη συνέχεια κάνοντας χρήση της παραπάνω διαμέρισης θα επιλυθεί το collocation γραμμικό σύστημα με τις πιο γνωστές preconditioned non-stationary επαναληπτικές μεθόδους.

3.3 Non-Stationary επαναληπτική επίλυση του Collocation γραμμικού συστήματος

Όπως έχει ήδη αναφερθεί, οι non-stationary επαναληπτικές μέθοδοι σχεδόν πάντα για να καταφέρουν να επιτύχουν σύγκλιση ή για να την βελτιώσουν, χρησιμοποιούν και μια τεχνική preconditioning. Αυτό σημαίνει ότι, εκτός από τη βασική πράξη που είναι ο πολλαπλασιασμός του πίνακα των συντελεστών με ένα διάνυσμα

σε κάθε επαναληπτικό βήμα εμπλέκεται και μια πιο δαπανηρή σε υπολογιστικό κόστος πράξη αυτή της δράσης του preconditioner στο γραμμικό σύστημα. Έτσι αν για την τεχνική του preconditioning χρησιμοποιηθεί κάποιος από τους πίνακες διάσπασης μιας stationary επαναληπτικής μεθόδου, οι οποίοι είναι διαθέσιμοι χωρίς επιπλέον κόστος κατασκευής, τότε στην διαδικασία εφαρμογής της non-stationary επαναληπτικής μεθόδου πρωταρχικό ρόλο έχει η διαμέριση του πίνακα των συντελεστών και η αντίστοιχη διάσπασή του. Για την εφαρμογή των preconditioned non-stationary επαναληπτικών μεθόδων στο collocation γραμμικό σύστημα θα γίνει χρήση της διάσπασης του πίνακα, όπως περιγράφηκε στην προηγούμενη ενότητα, ενώ θα χρησιμοποιηθούν δυο είδη αποθήκευσης των πινάκων, τα οποία περιγράφονται στη συνέχεια.

3.3.1 Μέθοδοι αποθήκευσης του collocation πίνακα

Η δομή του collocation πίνακα επιτρέπει την διαχείρησή του μέσω των βασικών πινάκων A_1, A_2, A_3 και A_4, οι οποίοι, όπως έχει αναφερθεί σε προηγούμενη ενότητα, είναι πίνακες ζώνης διαστάσεων $5 \times 2n_s$. Έτσι όλες οι πράξεις οι οποίες εμπλέκουν τον collocation πίνακα ή πίνακες, οι οποίοι παράγονται από αυτόν, μπορούν να γίνουν σε block μορφή.

Ένας άλλος τρόπος διαχείρησης είναι αυτός της κλασσικής μορφής ενός αραιού πίνακα K. Σύμφωνα με αυτή τη δομή αποθηκεύονται σε μορφή διανύσματος μόνο τα μη μηδενικά στοιχεία του. Έτσι, είναι απαραίτητη η χρήση κάποιων βοηθητικών διανυσμάτων δεικτών, τα οποία χρησιμεύουν για τον καθορισμό της θέσης κάθε μη μηδενικού στοιχείου στον αρχικό αραιό πίνακα. Υπάρχουν πολλά είδη τέτοιου είδους *αραιής* αποθήκευσης με βασικότερη την *αποθήκευση κατά στήλες*. Σύμφωνα με αυτή σε ένα διάνυσμα SK αποθηκεύονται τα μη μηδενικά στοιχεία του πίνακα K κατά στήλες με την προϋπόθεση ότι το στοιχείο της διαγωνίου που αντιστοιχεί

σε κάθε στήλη θα αποθηκευτεί πρώτο. Υπάρχει ένα διάνυσμα ακεραίων IK, το οποίο περιέχει πληροφορίες για την θέση των στοιχείων σε κάθε γραμμή και ένα άλλο διάνυσμα JK δεικτών των στηλών του πίνακα K στα διανύσματα SK και IK. Δηλαδή $IK(JK(k))$, $SK(JK(k))$ είναι οι θέσεις όπου περιέχουν τις πληροφορίες για το πρώτο στοιχείο της k-στης στήλης του πίνακα, ενώ για το τελευταίο θα έχουμε $IK(JK(k+1)-1)$ και $SK(JK(k+1)-1)$. Η τελευταία θέση $JK(n+1) = NELT+1$, όπου ο αραιός πίνακας έχει n στήλες και $NELT$ μη μηδενικά στοιχεία.

Έτσι για τον πίνακα

$$K = \begin{bmatrix} k_{11} & k_{12} & 0 & 0 & k_{15} \\ k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & k_{33} & 0 & k_{35} \\ 0 & 0 & 0 & k_{44} & 0 \\ k_{51} & 0 & k_{53} & 0 & k_{55} \end{bmatrix} ,$$

η αραιή αποθήκευσή του κατά στήλες θα είναι

$$SK = [\ k_{11}\ \ k_{21}\ \ k_{51}\ \ k_{22}\ \ k_{12}\ \ k_{33}\ \ k_{53}\ \ k_{44}\ \ k_{55}\ \ k_{15}\ \ k_{35}\ ] ,$$

$$IK = [\ 1\ \ 2\ \ 5\ \ 2\ \ 1\ \ 3\ \ 5\ \ 4\ \ 5\ \ 1\ \ 3\ ] ,\quad JK = [\ 1\ \ 4\ \ 6\ \ 8\ \ 9\ \ 12\ ] .$$

Αντίστοιχα ορίζεται και η *αποθήκευση κατά γραμμές*.

3.3.2 Μέθοδοι Preconditioning

Κατά την εφαρμογή των μεθόδων μπορούν να χρησιμοποιηθούν διάφορα είδη preconditioning τα οποία βασίζονται στην παραπάνω διαμέριση του collocation πίνακα κι έτσι ο preconditioner πίνακας M να μην χρειάζεται επιπλέον κόστος κατασκευής. Από αυτό το είδος των πινάκων αν επιλεγεί ο διαγώνιος πίνακας D_A, τότε προκύπτει ο Block Jacobi preconditioner, ο οποίος χρησιμοποιήθηκε και στις υλοποιήσεις στην [72].

Λαμβάνοντας υπόψη τα αποτελέσματα της έρευνας στην εργασία [34] μπορεί να χρησιμοποιηθεί ως πίνακας preconditioner ο πίνακας διάσπασης της μεθόδου

SSOR βασισμένος πάντοτε στην παραπάνω διαμέριση του collocation πίνακα. Έτσι ο preconditioner πίνακας M θα έχει τη μορφή

$$M \;=\; \frac{1}{\omega(2-\omega)}(D_A - \omega L_A)D_A^{-1}(D_A - \omega U_A)\;,\; \omega \in (0,2).$$

Αν επιλέξουμε την τιμή $\omega \;=\; 1$ στην παραπάνω έκφραση του πίνακα M προκύπτει ο πίνακας διάσπασης της μεθόδου SGS.

Jacobi Preconditioning

Η εφαρμογή του Jacobi Preconditioning σε μια non-stationary επαναληπτική μέθοδο είναι ισοδύναμη με την επίλυση ενός γραμμικού της μορφής $D_A \;\boldsymbol{y} \;=\; \boldsymbol{u}$ σε κάθε επαναληπτικό βήμα. Έτσι με την εφαρμογή του πίνακα μετασχηματισμού

$$T \;=\; \text{diag}[I\; G\; \cdots\; G\; I] \tag{91}$$

όπου

$$G \;=\; \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix} \;\;\text{και}\;\; I \in \mathbb{R}^{2n_s,2n_s} \;\;\text{είναι ο μοναδιαίος πίνακας}\;\;, \tag{92}$$

θα ισχύει ότι

$$T\,D_A = \text{diag}[A_2\; A_1\; A_2\; \cdots\; A_1\; A_2\;\; -A_2] \tag{93}$$

και

$$T\,\boldsymbol{u} \;=\; \frac{1}{2}\begin{bmatrix} 2\boldsymbol{u}_1 \\ \boldsymbol{u}_2 + \boldsymbol{u}_3 \\ \boldsymbol{u}_3 - \boldsymbol{u}_2 \\ \\ \vdots \\ \\ \boldsymbol{u}_{4p-2} + \boldsymbol{u}_{4p-1} \\ \boldsymbol{u}_{4p-1} - \boldsymbol{u}_{4p-2} \\ 2\boldsymbol{u}_{4p} \end{bmatrix}\;. \tag{94}$$

Η επίλυση του preconditioning συστήματος με τον τρόπο αυτό είναι εφικτή και σε παράλληλα περιβάλλοντα, αφού οι ομάδες των αγνώστων πλήθους $2n_s$ σύμφωνα

με την διαμέριση που έχει γίνει σε αυτούς, είναι ανεξάρτητες μεταξύ τους. Οπότε χρειάζεται η επίλυση $2n_s$ *μικρών* γραμμικών συστημάτων με πίνακες συντελεστών τους βασικούς πίνακες A_1 και A_2.

SSOR Preconditioning

Η εφαρμογή του SSOR Preconditioning σε μια επαναληπτική μέθοδο αντιστοιχεί με την επίλυση δυο τριγωνικών γραμμικών συστημάτων και ένα πολλαπλασιασμό ενός διανύσματος με τον πίνακα D_A. Πιό αναλυτικά αν το σύστημα preconditioning είναι το $M\boldsymbol{y} = \boldsymbol{u}$ θα έχουμε αρχικά την επίλυση του block κάτω τριγωνικού συστήματος

$$M_1\,\boldsymbol{v} \;=\; \omega(2-\omega)\boldsymbol{u} \;\;,\;\; \text{όπου} \quad M_1 \;=\; D_A - \omega L_A \quad , \tag{95}$$

στη συνέχεια ο πολλαπλασιασμός

$$D_A\,\boldsymbol{v} \;=\; \boldsymbol{z} \tag{96}$$

και τέλος η επίλυση του block άνω τριγωνικού συστήματος

$$M_2\,\boldsymbol{y} \;=\; \boldsymbol{z} \;\;,\;\; \text{όπου} \quad M_2 \;=\; D_A - \omega U_A \quad . \tag{97}$$

Για την επίλυση του πρώτου γραμμικού συστήματος μπορεί να εφαρμοστεί ο πίνακας μετασχηματισμού

$$T_1 \;=\; \text{diag}[I\; G_1\; \cdots\; G_1\; I] \tag{98}$$

όπου

$$G_1 \;=\; \begin{bmatrix} I & I \\ O & I \end{bmatrix} \;\; \text{και} \;\; I \in \mathbb{R}^{2n_s,2n_s} \;\; \text{είναι ο μοναδιαίος πίνακας} \;\; , \tag{99}$$

θα ισχύει ότι

$$T_1\, M_1 \;=\; \begin{bmatrix} A_2 & O & O & \dots & O & O & O & O & O \\ \omega A_4 & 2A_1 & O & \dots & O & O & O & O & O \\ O & A_1 & A_2 & \dots & O & O & O & O & O \\ O & \omega A_3 & \omega A_4 & \dots & O & O & O & O & O \\ O & O & O & \dots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & \dots & \omega A_3 & \omega A_4 & 2A_1 & O & O \\ O & O & O & \dots & O & O & A_1 & A_2 & O \\ O & O & O & \dots & O & O & \omega A_3 & \omega A_4 & -A_2 \end{bmatrix}, \tag{100}$$

και

$$T_1\, \boldsymbol{u} \;=\; \omega(2-\omega) \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 + \boldsymbol{u}_3 \\ \boldsymbol{u}_3 \\ \\ \vdots \\ \\ \boldsymbol{u}_{4p-2} + \boldsymbol{u}_{4p-1} \\ \boldsymbol{u}_{4p-1} \\ \boldsymbol{u}_{4p} \end{bmatrix}. \tag{101}$$

Με την εφαρμογή του πίνακα T_1 στο παραπάνω γραμμικό σύστημα τα blocks έχουν υποδιπλασιαστεί και το μέγεθός τους είναι $2n_s$ κι έτσι είναι πιο εύκολη η επίλυσή του με χρήση μόνο των βασικών πινάκων A_1, A_2, A_3 και A_4. Όμοια και για το δεύτερο σύστημα μπορεί να εφαρμοστεί ο πίνακας μετασχηματισμού

$$T_2 \;=\; \text{diag}[I\; G_2\; \cdots\; G_2\; I] \tag{102}$$

όπου

$$G_2 \;=\; \begin{bmatrix} I & O \\ -I & I \end{bmatrix} \quad \text{και} \quad I \in \mathbb{R}^{2n_s, 2n_s} \quad \text{είναι ο μοναδιαίος πίνακας} \quad , \tag{103}$$

θα ισχύει ότι

$$T_2 M_2 \;=\; \begin{bmatrix} A_2 & \omega A_3 & -\omega A_4 & O & O & \dots & O & O & O \\ O & A_1 & -A_2 & O & O & \dots & O & O & O \\ O & O & 2A_2 & \omega A_3 & -\omega A_4 & \dots & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \dots & \omega A_3 & -\omega A_4 & O \\ O & O & O & O & O & \dots & A_1 & -A_2 & O \\ O & O & O & O & O & \dots & O & 2A_2 & -\omega A_4 \\ O & O & O & O & O & \dots & O & O & -A_2 \end{bmatrix}, \tag{104}$$

και

$$T_2\, \boldsymbol{z} \;=\; \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \boldsymbol{z}_3 - \boldsymbol{z}_2 \\ \\ \vdots \\ \\ \boldsymbol{z}_{4p-2} \\ \boldsymbol{z}_{4p-1} - \boldsymbol{z}_{4p-2} \\ \boldsymbol{z}_{4p} \end{bmatrix} . \tag{105}$$

3.4 Υλοποίηση Non-Stationary επαναληπτικών μεθόδων για το Collocation γραμμικό σύστημα

Για την μελέτη της συμπεριφοράς των Non-Stationary επαναληπτικών μεθόδων στην επίλυση του Collocation γραμμικού συστήματος, έγινε υλοποίηση των πιο δημοφιλών από αυτές. Έτσι, σε πρώτη φάση έγιναν πειραματικές δοκιμές για την ταξινόμησή τους με τις τρεις διαφορετικές τεχνικές preconditioning σύμφωνα με την διαχείριση των δεδομένων σε block μορφή. Σε δεύτερη φάση όλοι οι πίνακες A, M_1, M_2 και D_A θεωρήθηκαν ότι είχαν την κλασσική δομή της αραιάς αποθήκευσης κατά στήλες. Τα δεδομένα που χρειάζονται να αποθηκευτούν σε αυτή την περίπτωση είναι σημαντικά μεγαλύτερα σε μέγεθος από την block μορφή κι έτσι οι υπολογισμοί θα έχουν ένα επιπλέον κόστος διαχείρισης λόγω μεγέθους, το οποίο θα πρέπει να λαμβάνεται σοβαρά υπόψη ιδιαίτερα για πίνακες μεγάλης διάστασης.

Θα πρέπει να σημειωθεί, ότι η μελέτη της συμπεριφοράς των επαναληπτικών μεθόδων για block αραιή αποθήκευση των βασικών πινάκων A_1, A_2, A_3 και A_4, δεν έχει ιδιαίτερο ερευνητικό ενδιαφέρον. Αυτή η μέθοδος αποθήκευσης δεν προσφέρει μεγάλη εξοικονόμηση μνήμης, αφού χρειάζεται να παραγοντοποιηθούν μόνο οι πίνακες A_1 και A_2 κι έτσι εξοικονομούνται μόνο $2 \times 4n_s$ θέσεις μνήμης. Έτσι, η προσθήκη σφαλμάτων κατά την ατελή παραγοντοποίηση και η αναμενόμενη αύξηση του κόστους των υπολογισμών εξαιτείας αυτών των σφαλμάτων, δεν

μπορεί να διακαιολογηθεί με την ανάλογη μείωση του χώρου αποθήκευσης. Γι' αυτό προτιμήθηκε να μελετηθεί η συμπεριφορά επαναληπτικών μεθόδων με χρήση της αραιής αποθήκευσης των μεγάλων πινάκων, παρόλο που δεν ενδείκνυται για το συγκεκριμένο πρόβλημα, αφού είναι εφικτοί οι υπολογισμοί σε block μορφή. Ωστόσο τα αποτελέσματα είναι ενδεικτικά της επίδρασης των σφαλμάτων της ατελούς παραγοντοποίησης στη συμπεριφορά των non-stationary preconditioned επαναληπτικών μεθόδων στο collocation γαρμμικό σύστημα.

Ως πρόβλημα μοντέλο χρησιμοποιήθηκε το ίδιο πρόβλημα Dirichet, όπως και στην εργασία [72] για την καλλίτερη σύγκριση των αποτελεσμάτων, με ακριβή λύση την

$$u(x,y) = 10\phi(x)\phi(y) \quad , \quad \phi(x) = \exp^{-100(x-0.1)^2}(x^2 - x) \quad .$$

Το υπολογιστικό σύστημα στο οποίο έγινε η υλοποίηση ήταν Hewlett Packard model 730 Risc αρχιτεκτονικής με 96MB μνήμη , ενώ ο κώδικας της εφαρμογής γράφτηκε σε γλώσσα Fortran με χρήση διπλής ακρίβειας στους υπολογισμούς. Επίσης χρησιμοποιήθηκαν οι μαθηματικές βιβλιοθήκες BLAS, LAPACK, TEMPLATES, SLAP και SLATEC για την βέλτιστη υλοποίηση της εφαρμογής.

Ως αρχική προσέγγιση της λύσης $\boldsymbol{x}^{(0)}$ δοκιμάστηκαν διάφορα διανύσματα, για τα οποία κάποιες μέθοδοι δεν κατάφεραν πάντα να συγκλίνουν. Έτσι παρακάτω παρουσιάζονται μόνο τα αποτελέσματα των μεθόδων, οι οποίες κατάφεραν πάντα να συγκλίνουν και η αρχική προσέγγιση ήταν σε όλες τις περιπτώσεις το δεξιό μέλος $\boldsymbol{b}$ του collocation γραμμικού συστήματος.

Ως κριτήριο τερματισμού χρησιμοποιήθηκε το

$$\frac{\| M^{-1}\boldsymbol{r}^{(k)} \|_2}{\| M^{-1}\boldsymbol{b} \|_2} \leq tol(j) \quad , \quad \boldsymbol{r}^{(k)} = \boldsymbol{b} - A\boldsymbol{x}^{(k)}$$

όπου η τιμή της σταθεράς $tol(j)$ (η τιμή j αναφέρεται σε συγκεκριμένη μέθοδο) έχει επιλεγεί κάθε φορά, ώστε όλες οι μέθοδοι μετά από N_j επαναληπτικά βήματα, να έχουν παράξει περίπου το ίδιο υπόλοιπο $\| \boldsymbol{r}^{(N_j)} \|_\infty$.

Αρχικά έγιναν δοκιμές με όλες τις γνωστές non-stationary επαναληπτικές μεθόδους χωρίς preconditioning. Όμως, δεν υπήρχε πάντα σύγκλιση ή όταν κάποια μέθοδος κατάφερε να συγκλίνει η σύγκλιση ήταν υπερβολικά αργή σε σχέση πάντα με την περίπτωση του preconditioning. Κάποιες μέθοδοι επίσης παρουσίασαν πολύ ασταθή συμπεριφορά σε σχέση με την αρχική προσέγγιση $\boldsymbol{x}^{(0)}$ της λύσης. Τέτοιες μέθοδοι δεν μελετήθηκαν λεπτομερέστερα και δεν παρουσιάζονται οι πειραματικές τους μετρήσεις, αφού η ασταθής συμπεριφορά τους δεν επέτρεψε την σύγκρισή τους με τις υπόλοιπες μεθόδους για όλες τις περιπτώσεις.

Στις δυο παρακάτω ενότητες παρουσιάζονται τα αποτελέσματα των πειραματικών μετρήσεων για τις τρεις πιο ευσταθείς σε συμπεριφορά γνωστές non-stationary επαναληπτικές μεθόδους. Αυτές είναι η μέθοδος ORTHOMIN [114], η οποία είναι από τις πρώτες non-stationary επαναληπτικές μεθόδους, η γνωστή GMRES(m) η οποία ουσιαστικά αποτελεί μια πιο εξελιγμένη μορφή της ORTHOMIN και δοκιμάστηκε για διάφορες τιμές της παραμέτρου επανεκκίνησης m και τέλος η υβριδική BiCGSTAB, η οποία αποτελεί συνδυασμό BiCG και της GMRES(1). Τα αποτελέσματα παρουσιάζονται σε δυο ενότητες ανάλογα με τον τρόπο διαχείρησης των δεδομένων, δηλαδή στη μορφή της block ζώνης των πινάκων και της αραιάς αποθήκευσης.

Α. Υπολογισμοί με μορφή Block ζώνης

Στους υπολογισμούς μορφής Block ζώνης χρειάζεται να αποθηκευτούν μόνο οι βασικοί πίνακες A_1, A_2, A_3 και A_4 σε μορφή ζώνης εύρους 5. Η επίλυση των συστημάτων preconditioning γίνεται με την άμεση μέθοδο της LDU διάσπασης, οπότε οι πίνακες αυτοί παραγοντοποιούνται μια φορά στην αρχή της επαναληπτικής διαδικασίας. Ο χρόνος της παραγοντοποίησης αυτής δεν μετριέται στο συνολικό χρόνο της επαναληπτικής μεθόδου στις παρακάτω μετρήσεις.

Α1. Jacobi Preconditioning

Οι παρακάνω πίνακες Τ1 και Τ2 εμφανίζουν τις πειραματικές μετρήσεις για την

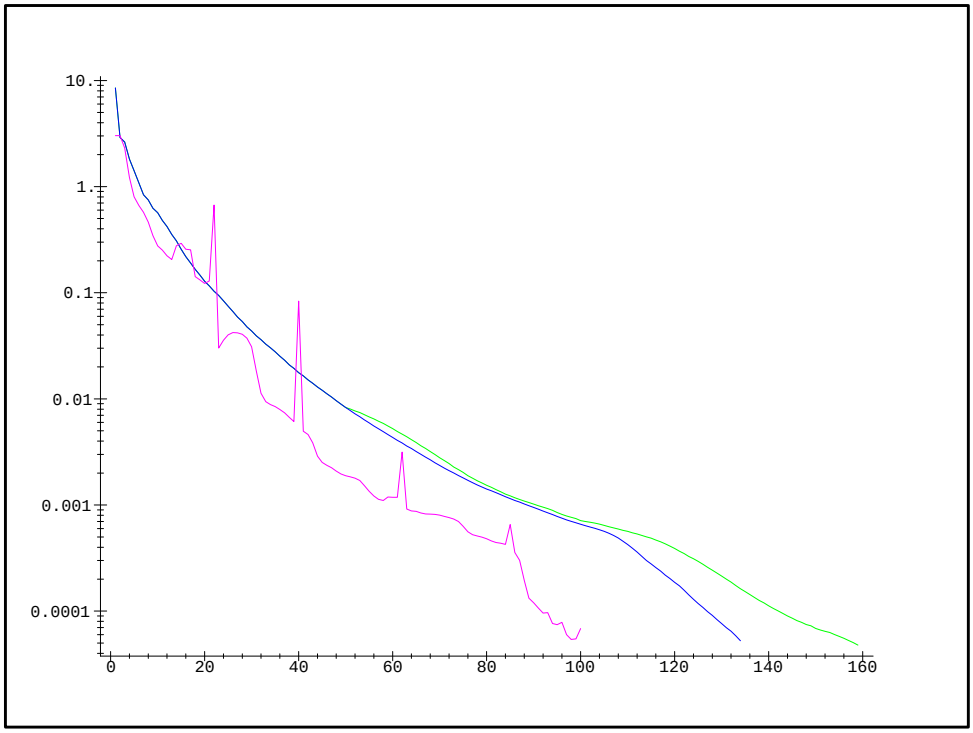
περίπτωση της τεχνικής Jacobi Preconditioning για τις τρεις non-stationary επαναληπτικές μεθόδους ORTHOMIN, GMRES(50) και BiCGSTAB. Οι μετρήσεις του χρόνου είναι σε δευτερόλεπτα.

T1	GMRES (50)				ORTHOMIN			
n_s	Steps m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	Steps m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	11	0.02	6.96e-1	2.65e-4	11	0.01	6.96e-1	2.65e-4
8	22	0.13	2.03e-2	3.72e-5	22	0.15	2.03e-2	3.72e-5
16	38	0.93	6.08e-4	2.25e-5	38	1.18	6.08e-4	2.24e-5
32	78	9.6	3.02e-5	5.49e-6	71	14.55	3.33e-5	5.73e-6
64	159	119	1.49e-5	1.70e-6	135	232	5.51e-6	1.68e-6
128	300	1327	7.18e-5	1.22e-6	-	-	-	-

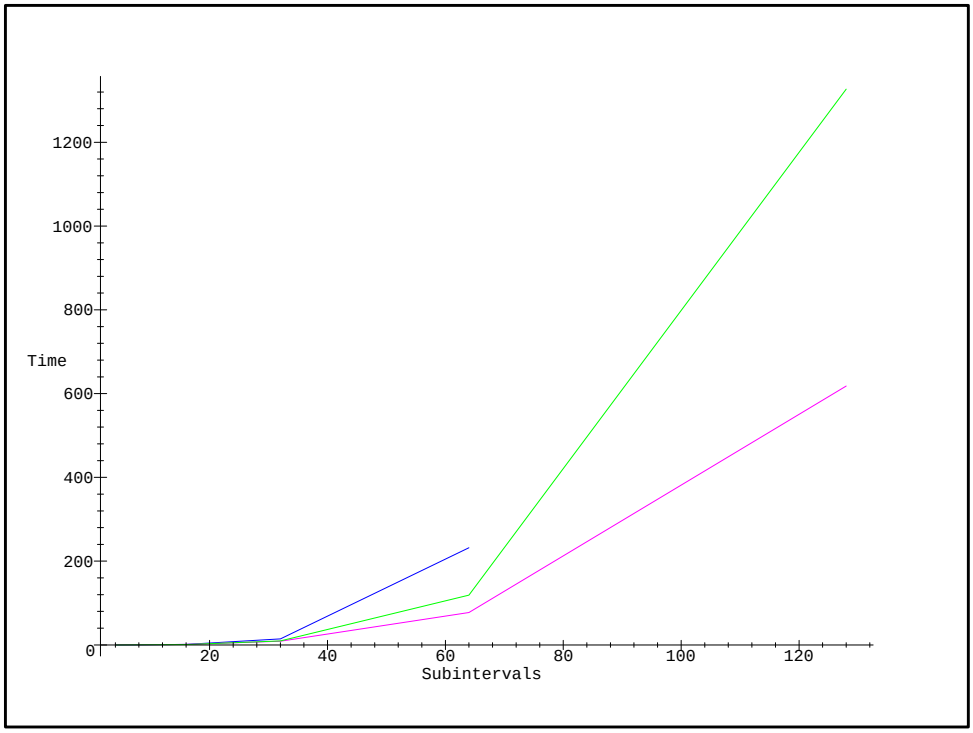
T2	BiCGSTAB			
n_s	Steps m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	7	0.08	6.95e-1	1.38e-4
8	15	0.23	2.03e-2	4.42e-5
16	27	1.26	6.06e-4	1.48e-5
32	50	8.7	3.16e-5	5.16e-6
64	100	77.5	1.92e-6	1.69e-7
128	182	618	2.75e-5	4.94e-7

Οι εμφανιζόμενες νόρμες είναι οι άπειρες νόρμες των σφαλμάτων και υπολοίπων για κάθε περίπτωση. Θα πρέπει να σημειωθεί ότι η μέθοδος ORTHOMIN δεν κατάφερε να εκτελεστεί για $n_s = 128$, επειδή οι απαιτήσεις της μεθόδου σε μνήμη ήταν μεγαλύτερες από τις δυνατότητες του συγκεκριμένου υπολογιστικού συστήματος που έγινε η υλοποίηση. Η μέθοδος BiCGSTAB, όπως άλλωστε αναμενόταν για μεγάλες διαμερίσεις συγκλίνει στα μισά βήματα από ότι οι υπόλοιπες δυο. Στην Εικόνα 27 εμφανίζεται η μείωση του σφάλματος που αναφέρεται στο κριτήριο τερματισμού των μεθόδων σε σχέση με τα επαναληπτικά βήματα. Από αυτό το γράφημα προκύπτει ότι η μέθοδος BiCGSTAB εμφανίζει σημαντικές αυξομειώσεις στο σφάλμα, παρόλαυτα καταφέρνει να το μειώσει αρκετά γρηγορότερα από τις άλλες δύο μεθόδους, οι οποίες παρουσιάζουν αρκετά πιο ομαλή μείωσή του. Επίσης η μέθοδος BiCGSTAB εμφανίζει γρηγορότερη σύγκλιση για μεγάλες διαμερίσεις ($n_s \geq 32$), ενώ η GMRES(50) για μικρές, όπως άλλωστε φαίνεται καλλίτερα και από

το γράφημα της Εικόνας 28, η οποία εμφανίζει το συνολικό χρόνο εκτέλεσης των μεθόδων για όλες τις διαμερίσεις που δοκιμάστηκαν. Έτσι η πιο αργή μέθοδος είναι η ORTHOMIN, ενώ γρηγορότερη της προκύπτει η GMRES(50) με ακόμα ταχύτερη την BiCGSTAB με μισό περίπου χρόνο εκτέλεσης από την GMRES(50).



Εικόνα 27 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους BiCGSTAB-ρόζ, GMRES(50)-πράσινο και ORTHOMIN-μπλέ με Jacobi preconditioning και $n_s = 64$.



Εικόνα 28 : Χρόνος εκτέλεσης των μεθόδων BiCGSTAB-ρόζ, ORTHOMIN-μπλέ, και GMRES(50)-πράσινο για Jacobi preconditioning

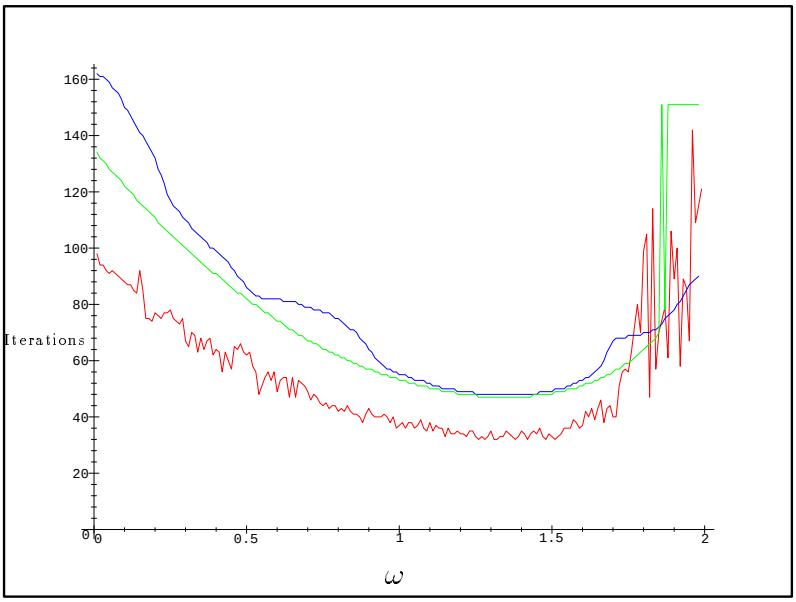
A2. SSOR Preconditioning

Οι παρακάνω πίνακες T3, T4 και T5 εμφανίζουν τις πειραματικές μετρήσεις για την περίπτωση της τεχνικής SSOR Preconditioning. Ειδικά για την μέθοδο GMRES(m) έγιναν δοκιμές με τρεις διαφορετικού μεγέθους παραμέτρους επανεκκίνησης ίσες με 5, 10 και 50. Η βέλτιστη τιμή της παραμέτρου ω_{opt} έχει υπολογιστεί πειραματικά και ειδικά για την μέθοδο GMRES η τιμή της δεν επηρεάζεται από την παράμετρο επανεκκίνησης m.

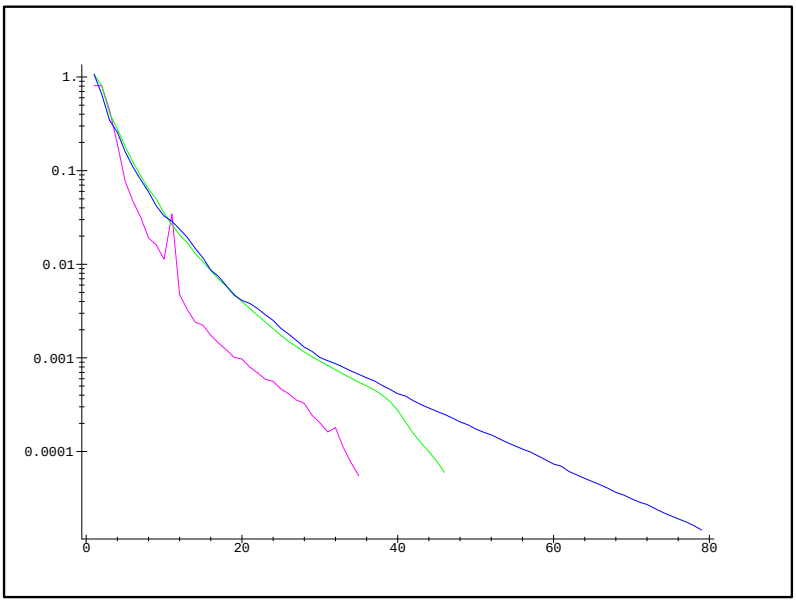
T3	BiCGSTAB				
n_s	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	1.3	4	0.03	6.95e-1	1.23e-4
8	1.3	7	0.18	2.03e-2	2.66e-5
16	1.3	10	0.96	6.07e-4	2.62e-5
32	1.4	16	5.95	2.93e-5	2.27e-5
64	1.3	34	53.5	3.02e-5	4.56e-6
128	1.3	56	377	5.31e-5	2.76e-6

T4	GMRES(5)				ORTHOMIN				
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	5	0.02	6.95e-1	3.30e-4	1.1	5	0.02	6.96e-1	3.30e-4
8	10	0.11	2.03e-2	1.02e-4	1.2	10	0.1	2.03e-2	5.66e-4
16	18	0.7	6.08e-4	2.33e-5	1.3	15	0.63	6.08e-4	4.46e-5
32	36	5.4	3.03e-5	2.16e-5	1.3	25	4.9	3.35e-5	2.15e-5
64	105	65.4	3.58e-6	5.12e-6	1.3	47	66	6.73e-6	5.55e-6
128	151	458	5.71e-5	3.06e-6	-	-	-	-	-
T5	GMRES(10)				GMRES(50)				
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	5	0.02	6.95e-1	3.30e-4	1.1	5	0.02	6.95e-1	3.30e-4
8	9	0.11	2.03e-2	2.68e-4	1.2	10	0.11	2.03e-2	1.02e-4
16	17	0.7	6.07e-4	4.18e-5	1.4	16	0.66	6.08e-4	2.54e-5
32	33	5.05	3.02e-5	1.97e-5	1.4	26	4.72	3.35e-5	1.64e-5
64	83	56	1.07e-5	2.13e-6	1.4	51	53	2.24e-6	2.42e-6
128	151	455	5.71e-5	3.07e-6	1.4	100	553	8.43e-6	3.79e-6

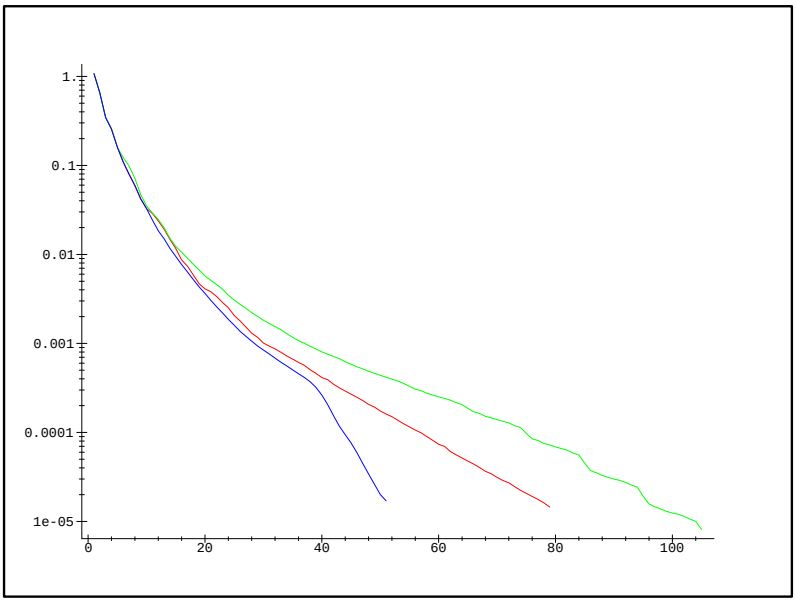
Στην Εικόνα 29 εμφανίζεται το γράφημα της τιμής της παραμέτρου ω σε σχέση με τον αριθμό των επαναληπτικών βημάτων κάθε μεθόδου για $n_s = 64$. Παρατηρούμε ότι υπάρχει ένα αρκετά ευρύ πεδίο τιμών της παραμέτρου, λίγο πιο πάνω από την μονάδα και μέχρι το 1.5, όπου εμφανίζεται ο μικρότερος αριθμός βημάτων. Αξίζει να σημειωθεί, ότι για τιμές κοντά στο 2, οι μέθοδοι BiCGSTAB και ORTHOMIN εμφανίζουν μεγάλη διαφορά τον αριθμό των επαναληπτικών βημάτων για μικρή αυξομείωση της τιμής του ω. Η Εικόνα 30 παρουσιάζει την μείωση του σφάλματος του κριτηρίου σύγκλισης για τις μεθόδους BiCGSTAB, ORTHOMIN και GMRES(10). Παρατηρούμε ότι η BiCGSTAB εξακολουθεί να παρουσιάζει κάποιες αυξομειώσεις, αλλά είναι πιο ήπιες από την περίπτωση του Jacobi preconditioning, και ξανά πετυχαίνει μεγαλύτερη μείωση σφάλματος από τις άλλες δυο μεθόδους, οι οποίες εμφανίζουν πιο αργή και περισσότερο ομαλή μείωσή του. Στην Εικόνα 31 παρουσιάζεται η ίδια μείωση για τις τρεις περιπτώσεις της GMRES. Η μέθοδος GMRES(50) πετυχαίνει την γρηγορότερη μείωση σε σχέση με τις άλλες δυο μεθόδους. Όπως και για Jacobi preconditioning η μέθοδος BiCGSTAB εμφανίζει ταχύτερη σύγκλιση για μεγάλες διαμερίσεις ($n_s \geq 64$), ενώ η GMRES(5) για μικρές, όπως είναι πιο φανερό και από το γράφημα της Εικόνας 32.



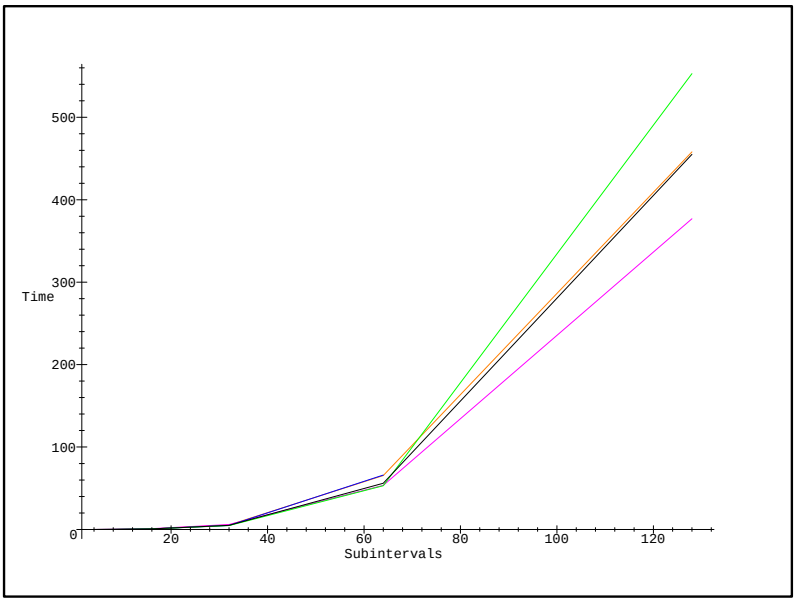
Εικόνα 29 : Αριθμός επαναλήψεων για τις διάφορες τιμές του ω με SSOR preconditioning για τις μεθόδους BiCGSTAB-κόκκινο, GMRES(50)-μπλέ και ORTHOMIN-πράσινο με $n_s = 64$.



Εικόνα 30 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους BiCGSTAB-ρόζ, GMRES(10)-μπλέ και ORTHOMIN-πράσινο με SSOR preconditioning για $n_s = 64$.



Εικόνα 31 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους GMRES(5)-πράσινο, GMRES(10)-κόκκινο και GMRES(50)-μπλέ με SSOR preconditioning για $n_s = 64$.



Εικόνα 32 : Χρόνος εκτέλεσης των μεθόδων BiCGSTAB-ρόζ, ORTHOMIN-μπλέ, GMRES(5)-κίτρινο, GMRES(10)-μαύρο και GMRES(50)-πράσινο με SSOR preconditioning.

A3. SGS Preconditioning

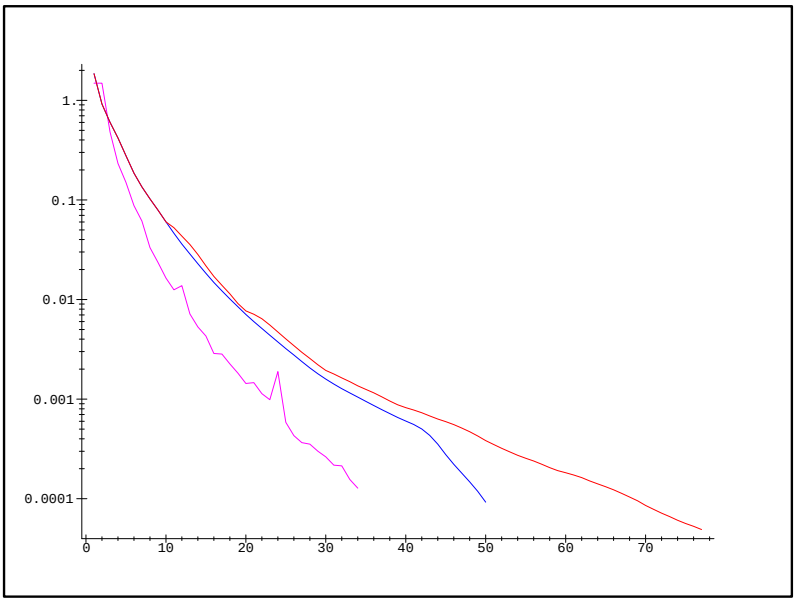
Οι παρακάνω πίνακες T6, T7 και T8 εμφανίζουν τις πειραματικές μετρήσεις για την περίπτωση του SGS Preconditioning.

T6	GMRES(5)				GMRES(10)			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	5	0.02	6.96e-1	3.30e-4	5	0.02	6.95e-1	2.75e-4
8	9	0.11	2.03e-2	2.70e-4	9	0.11	2.03e-2	6.16e-5
16	17	0.72	6.07e-4	3.18e-5	17	0.72	6.07e-4	3.39e-5
32	39	5.75	3.19e-5	1.03e-5	33	5.1	3.04e-5	1.22e-5
64	102	62.5	2.17e-5	3.15e-6	77	52	2.14e-5	3.07e-6
128	189	514	3.07e-5	2.23e-6	163	493	9.60e-5	2.41e-6

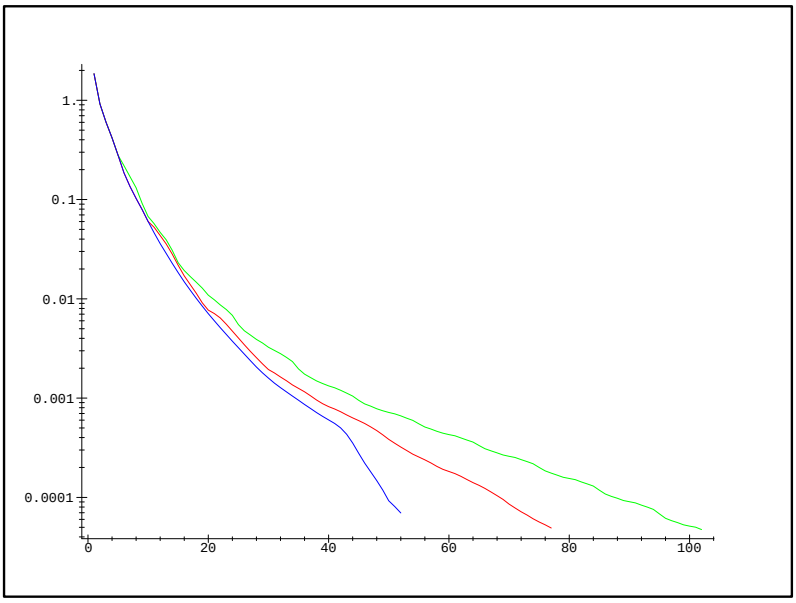
T7	GMRES(50)				BiCGSTAB			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	5	0.02	6.95e-1	2.76e-4	4	0.02	6.95e-1	6.13e-7
8	8	0.1	2.03e-2	2.89e-4	7	0.21	2.03e-2	4.79e-4
16	15	0.66	6.08e-4	5.38e-5	11	1.1	6.07e-4	3.64e-5
32	26	4.95	3.31e-5	2.47e-5	17	6.44	6.39e-5	2.30e-5
64	52	56.6	1.50e-5	4.11e-6	34	53	2.82e-5	3.34e-6
128	106	569	2.68e-5	2.44e-6	57	384	9.46e-5	2.45e-6

T8	ORTHOMIN			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	5	0.02	6.95e-1	2.75e-4
8	9	0.1	2.03e-2	2.75e-4
16	15	0.65	6.08e-4	5.38e-5
32	27	5.4	3.32e-5	2.47e-5
64	50	71.9	1.57e-5	6.99e-6

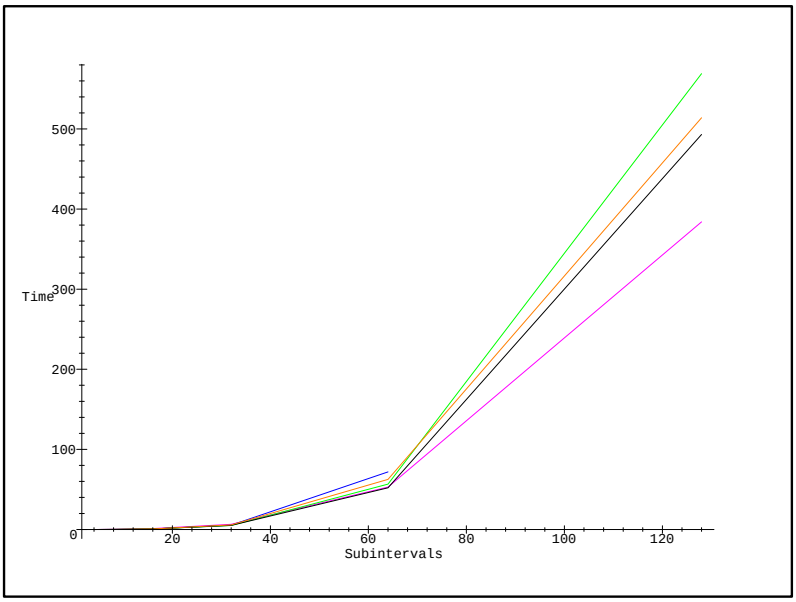
Τα αποτελέσματα, όπως άλλωστε αναμενόταν, είναι παρόμοια με αυτά της περίπτωσης του SSOR preconditioning. Η επολογή του $\omega = 1$ σύμφωνα με το γράφημα της Εικόνας 29 δεν επιβαρύνει σημαντικά τις μεθόδους με επαναληπτικά βήματα κι έτσι δεν παρατηρούμε κάποια σημαντική αλλαγή στη συμπεριφορά όλων των μεθόδων. Τα γραφήματα των επόμενων τεσσάρων εικόνων άλλωστε είναι σχεδόν τα ίδια με τα αντίστοιχα της προηγούμενης περίπτωσης του SSOR preconditioning.



Εικόνα 33 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους BiCGSTAB-ρόζ, GMRES(10)-κόκκινο και ORTHOMIN-μπλε με SGS preconditioning για $n_s = 64$.



Εικόνα 34 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους GMRES(5)-πράσινο, GMRES(10)-κόκκινο και GMRES(50)-μπλε με SGS preconditioning για $n_s = 64$.



Εικόνα 35 : Χρόνος εκτέλεσης των μεθόδων BiCGSTAB-ρόζ, ORTHOMIN-μπλέ, GMRES(5)-κίτρινο, GMRES(10)-μαύρο και GMRES(50)-πράσσινο με SGS preconditioning.

Β. Υπολογισμοί με μορφή Αραιής αποθήκευσης

Στα πειραματικά αποτελέσματα που ακολουθούν, τόσο ο collocation πίνακας αποθηκεύτηκε με την δομή της αραιής αποθήκευσης κατά στήλες, όσο και οι preconditioner πίνακες . Έτσι για την λύση των preconditioning γραμμικών συστημάτων χρησιμοποιήθηκε η κλασσική μέθοδος της ατελούς LDU παραγοντοποίησης. Όπως και στην προηγούμενη περίπτωση θεωρείται ότι οι κατάλληλοι πίνακες δίνονται ήδη παραγοντοποιημένοι στην επαναληπτική μέθοδο κι έτσι ο χρόνος παραγοντοποίησης τους δεν υπολογίζεται στο χρόνο της επαναληπτικής διαδικασίας. Εξαιτείας της σημαντικής αύξησης των απαιτήσεων για χώρο μνήμης λόγω της αποθήκευσης όλων των μη μηδενικών στοιχείων των εμπλεκομένων πινάκων δεν ήταν δυνατή η πραγματοποίηση δοκιμών στο συγκεκριμένο υπολογιστικό σύστημα για διαμερίσεις με $n_s > 64$ για καμία επαναληπτική μέθοδο. Επίσης η τεχνική του Jacobi preconditioning δεν στάθηκε ικανή να κάνει τις μεθόδους να συγκλίνουν ιδιαίτερα για μεγάλες διαμερίσεις γι' αυτό παρουσιάζονται πειραματικά αποτελέσματα μόνο για

τις περιπτώσεις με SSOR και SGS preconditioning.

B1. SSOR Preconditioning

Οι παρακάτω πίνακες T9, T10 και T11 παρουσιάζουν τις πειραματικές μετρήσεις για την περίπτωση της αραιής αποθήκευσης με την εφαρμογή της τεχνικής του SSOR preconditioning για τις τρεις επαναληπτικές μεθόδους.

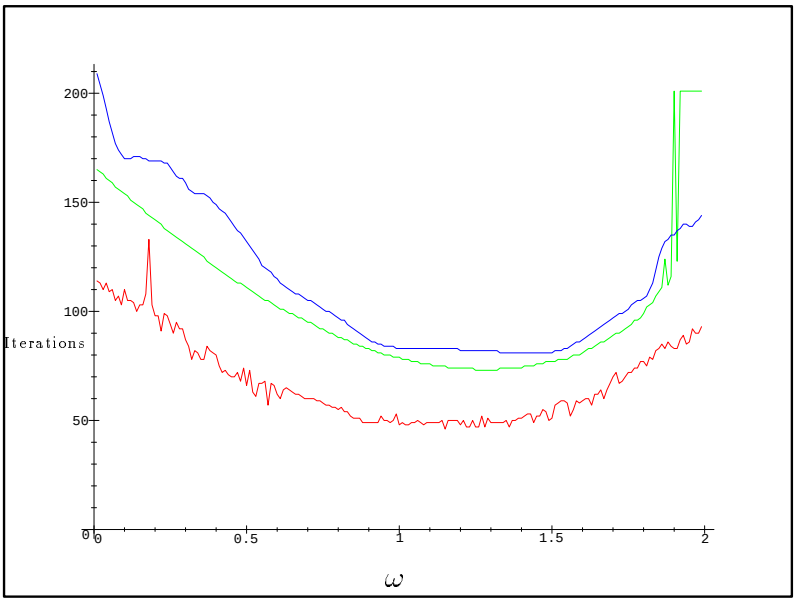
T9	GMRES(50)					BiCGSTAB				
n_s	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	1.1	8	0.03	6.96e-1	2.65e-4	1.0	4	0.06	6.96e-1	1.75e-5
8	1.1	14	0.09	2.03e-2	5.56e-5	1.2	8	0.07	2.03e-2	4.13e-5
16	1.3	23	0.57	6.08e-4	7.52e-5	1.2	14	0.68	6.08e-4	1.30e-5
32	1.3	39	5.23	3.36e-5	2.23e-5	1.2	25	5.38	3.25e-5	7.86e-5
64	1.4	78	58.3	1.08e-5	9.37e-6	1.3	48	46.7	3.33e-5	8.45e-6

T10	GMRES(5)				GMRES(10)			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	8	0.01	6.95e-1	2.28e-4	8	0.01	6.95e-1	2.65e-4
8	14	0.08	2.03e-2	6.89e-4	14	0.08	2.03e-2	7.12e-5
16	28	0.61	6.07e-4	4.88e-5	25	0.56	6.07e-4	5.84e-5
32	55	4.93	2.93e-5	2.85e-5	46	4.46	2.97e-5	2.34e-5
64	160	65.63	3.13e-5	9.39e-6	112	51.7	2.65e-5	8.11e-6

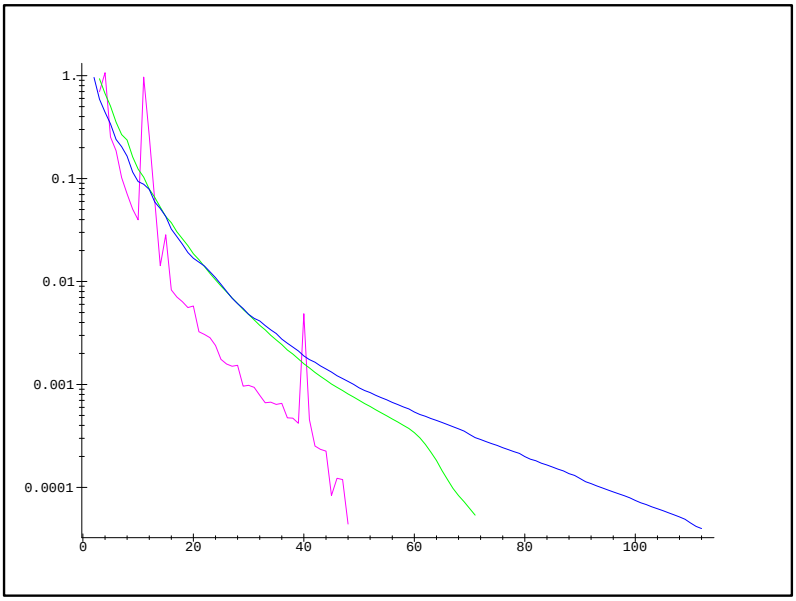
T11	ORTHOMIN				
n_s	ω_{opt}	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	1.1	8	0.01	6.96e-1	2.65e-4
8	1.1	14	0.08	2.03e-2	5.56e-5
16	1.3	23	0.65	6.08e-4	7.52e-5
32	1.3	38	6.12	3.35e-5	3.24e-5
64	1.3	71	105	8.93e-6	9.71e-6

Όπως και στην περίπτωση της χρήσης της block μορφής των πινάκων στους υπολογισμούς η τιμή της παραμέτρου ω_{opt} βρέθηκε πειραματικά και η Εικόνα 36 εμφανίζει το γράφημα της τιμής της παραμέτρου αυτής ως προς τα επαναληπτικά βήματα για διαμέριση $n_s = 64$. Αν το συγκρίνουμε με το αντίστοιχο γράφημα της Εικόνας 29 δεν παρατηρούμε κάποια σημαντική διαφοροποίηση ως προς την βέλτιστη τιμή της. Η μόνη διαφορά που υπάρχει αφορά την συμπεριφορά των

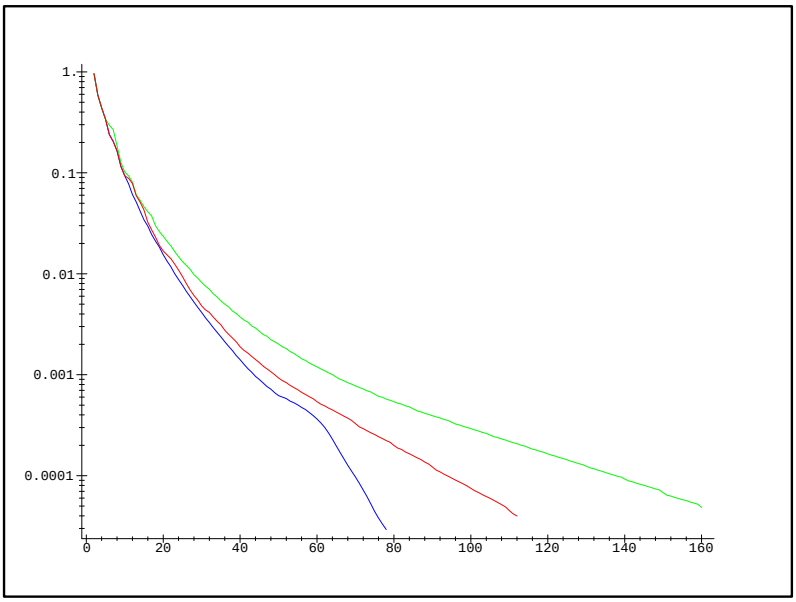
μεθόδων BiCGSTAB και ORTHOMIN για τις τιμές του ω πάνω από το 1.5, όπου δεν εμφανίζουν τόσο μεγάλη διαφορά στα επαναληπτικά βήματα για μικρές διαφορές στην τιμή του ω σε σχέση με την block μορφή. Παρατηρούμε ξανά ότι η τιμή του $\omega = 1$ είναι αρκετά κοντά στην βέλτιστη, οπότε η χρήση της τεχνικής του SGS preconditioning αποτελεί μια ικανοποιητική επιλογή. Η παράμετρος ω για την μέθοδο GMRES(50) δεν επηρεάστηκε καθόλου από το σφάλμα της ατελούς παραγοντοποίησης κατά τη διάρκεια της τεχνικής του preconditioning σύμφωνα με το γράφημα. Οι Εικόνες 37 και 38 εμφανίζουν την μείωση του σφάλματος του κριτηρίου τερματισμού των μεθόδων ως προς τα επαναληπτικά βήματα για τις μεθόδους BiCGSTAB, ORTHOMIN, GMRES(10) και της ομάδας των GMRES αντίστοιχα. Συγκρίνοντας τα με τα αντίστοιχα των Εικόνων 30 και 31, εκτός από την σημαντική επιβάρυνση των επαναληπτικών βημάτων σε όλες τις μεθόδους λόγω του σφάλματος της ατελούς παραγοντοποίησης και της διαχείρησης μεγάλου μεγέθους δεδομένων, δεν παρατηρούμε κάποια διαφοροποίηση στην συμπεριφορά τους. Ίσως η μόνη διαφορά να είναι η αύξηση των αυξομειώσεων του σφάλματος για την μέθοδο BiCGSTAB, η οποία εξακολουθεί να παρουσιάζει την μεγαλύτερη και γρηγορότερη μείωση του έναντι των υπολοίπων μεθόδων. Η Εικόνα 39 εμφανίζει το γράφημα για τον συνολικό χρόνο εκτέλεσης όλων των μεθόδων για τις διάφορες τιμές της διαμέρισης. Αν το συγκρίνουμε με αυτό της Εικόνας 32 και έχοντας υπόψη, ότι δεν υπάρχουν αποτελέσματα για την περίπτωση της αραιάς αποθήκευσης για διαμέριση μεγαλύτερη του 64, εύκολα παρατηρούμε ότι όσο αφορά την γρηγορότερη μέθοδο δεν υπάρχει κάποια διαφοροποίηση. Έτσι η BiCGSTAB συγκλίνει γρηγορότερα για μεγάλες διαμερίσεις και η GMRES(10) για μικρές, ενώ η ORTHOMIN εμφάνισε ξανά την αργότερη σύγκλιση.



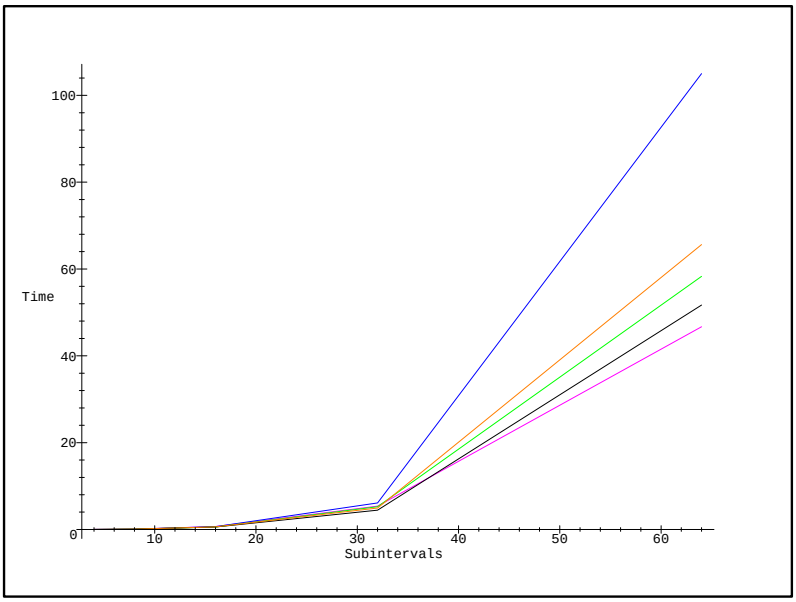
Εικόνα 36 : Αριθμός επαναλήψεων για τις διάφορες τιμές του ω με SSOR preconditioning για τις μεθόδους BiCGSTAB-κόκκινο, GMRES(50)-μπλέ και ORTHOMIN-πράσινο με $n_s = 64$.



Εικόνα 37 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους BiCGSTAB-ροζ, GMRES(10)-μπλε και ORTHOMIN-πράσινο με SGS preconditioning για $n_s = 64$.



Εικόνα 38 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους GMRES(5)-πράσινο, GMRES(10)-κόκκινο και GMRES(50)-μπλε με SSOR preconditioning για $n_s = 64$.



Εικόνα 39 : Χρόνος εκτέλεσης των μεθόδων BiCGSTAB-ρόζ, ORTHOMIN-μπλέ, GMRES(5)-κίτρινο, GMRES(10)-μαύρο και GMRES(50)-πράσινο για SSOR preconditioning

B2. SGS Preconditioning

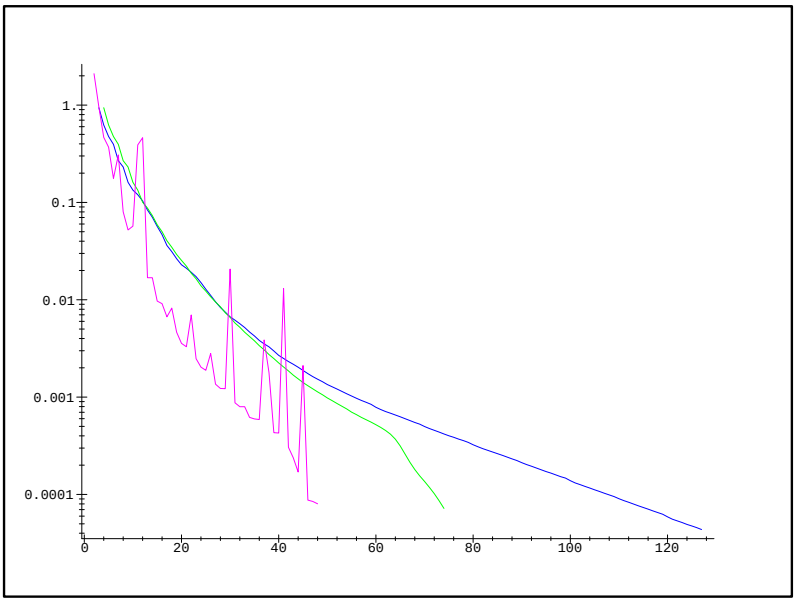
Οι πίνακες T12, T13 και T14 περιέχουν τις πειραματικές μετρήσεις για την τεχνική του SGS preconditioning με χρήση αραιής αποθήκευσης των πινάκων στους υπολογισμούς των επαναληπτικών μεθόδων.

T12	GMRES(50)				BiCGSTAB			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	8	0.03	6.95e-1	1.91e-4	4	0.06	6.96e-1	1.75e-5
8	14	0.09	2.02e-2	1.24e-4	9	0.15	2.03e-2	1.04e-5
16	23	0.58	6.08e-4	5.18e-4	15	0.78	6.08e-4	1.65e-5
32	41	5.6	3.34e-5	1.56e-5	26	5.6	3.25e-5	7.19e-6
64	81	61.6	6.01e-6	3.61e-6	49	47.8	2.98e-5	4.54e-6

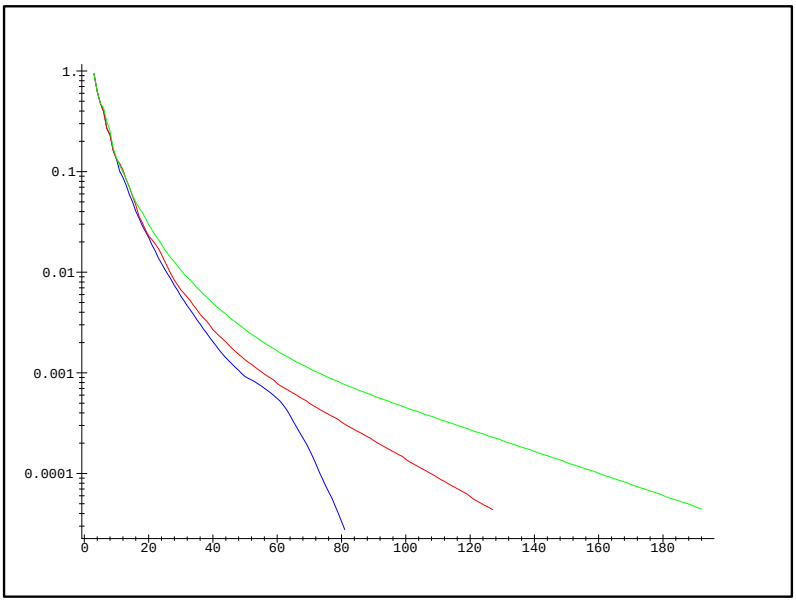
T13	GMRES(5)				GMRES(10)			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	8	0.01	6.96e-1	2.28e-4	8	0.02	6.95e-1	1.91e-4
8	14	0.07	2.03e-2	1.24e-4	14	0.07	2.03e-2	1.25e-4
16	23	0.5	6.01e-4	1.48e-4	26	0.59	6.07e-4	3.18e-5
32	57	5.12	4.23e-5	2.94e-5	49	4.74	3.07e-5	1.15e-5
64	192	78.66	2.33e-5	3.19e-6	127	58.4	2.24e-5	3.31e-6

T14	ORTHOMIN			
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$
4	8	0.01	6.96e-1	1.91e-4
8	15	0.1	2.03e-2	2.08e-5
16	23	0.68	6.08e-4	5.18e-5
32	40	6.7	3.33e-5	2.34e-5
64	74	112	9.59e-6	6.85e-6

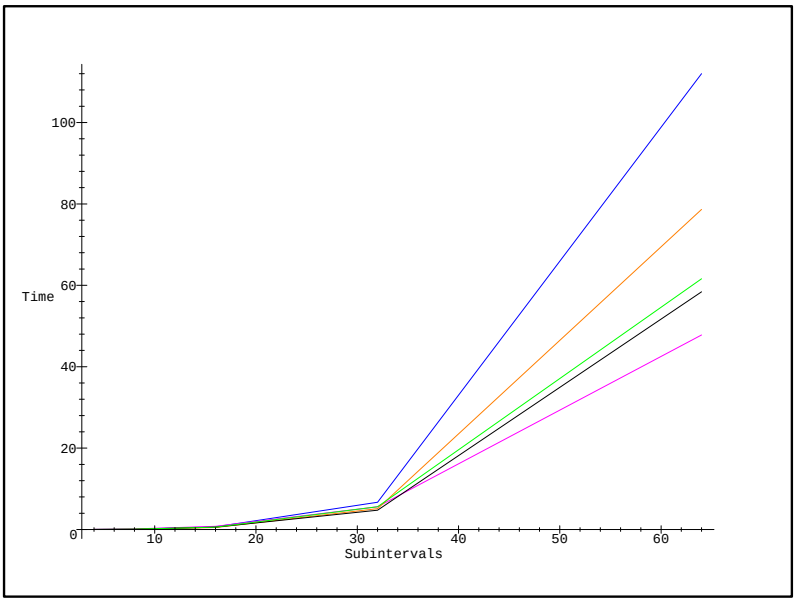
Συγκρίνοντας τα γραφήματα των Εικόνων 40 και 41, τα οποία εμφανίζουν την μείωση του σφάλματος των κριτηρίων για κάθε μέθοδο σε σχέση με τα επαναληπτικά βήματα,με τα αντίστοιχα των Εικόνων 33 και 34 παρατηρούμε, ότι η μόνη μέθοδος που επηρεάστηκε από το σφάλμα της ατελούς παραγοντοποίησης είναι η BiCGSTAB, η οποία αύξησε τις αυξομειώσεις του σφάλματος, αλλά εξακολουθεί να παρουσιάζει την ταχύτερη μείωση του. Έτσι αν και έχει αυξηθεί ο αριθμός των επαναληπτικών βημάτων και αντίστοιχα ο χρόνος εκτέλεσης των μεθόδων, όπως προκύπτει από την σύγκριση των γραφημάτων των Εικόνων 35 και 42, δεν παρατηρούμε κάποια αλλαγή στην συμπεριφορά τους.



Εικόνα 40 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους BiCGSTAB-ροζ, GMRES(10)-μπλε και ORTHOMIN-πράσινο με SGS preconditioning για $n_s = 64$.



Εικόνα 41 : Μείωση του σφάλματος στο κριτήριο τερματισμού ως προς τις επαναλήψεις για τις μεθόδους GMRES(5)-πράσινο, GMRES(10)-κόκκινο και GMRES(50)-μπλε με SSOR preconditioning για $n_s = 64$.

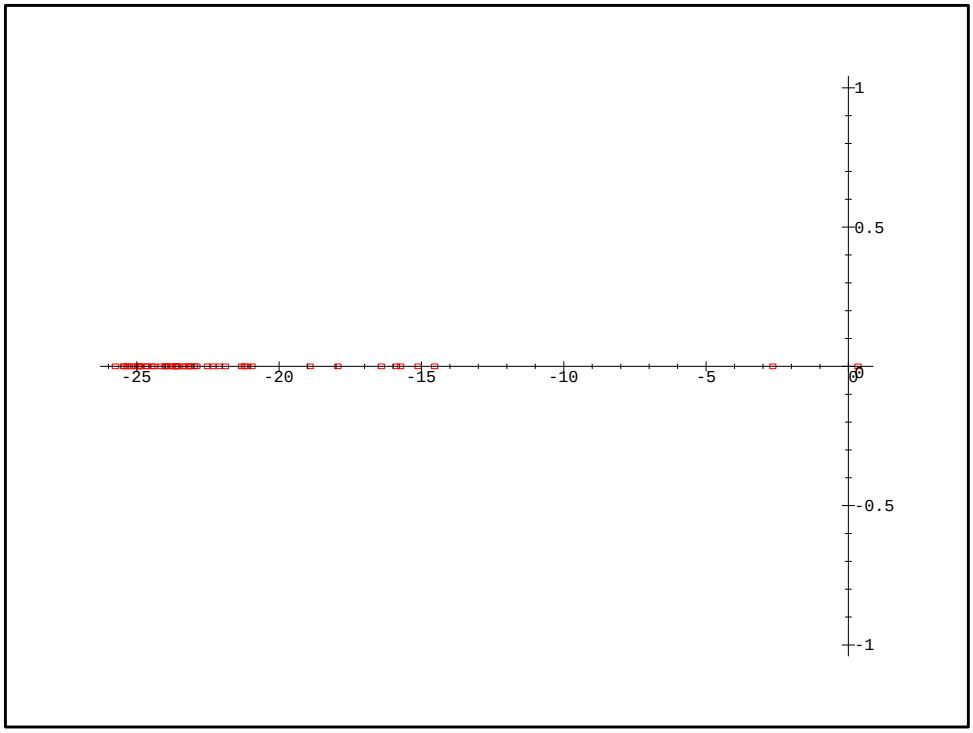


Εικόνα 42 : Χρόνος εκτέλεσης των μεθόδων BiCGSTAB-ρόζ, ORTHOMIN-μπλέ, GMRES(5)-κίτρινο, GMRES(10)-μαύρο και GMRES(50)-πράσινο με SGS preconditioning.

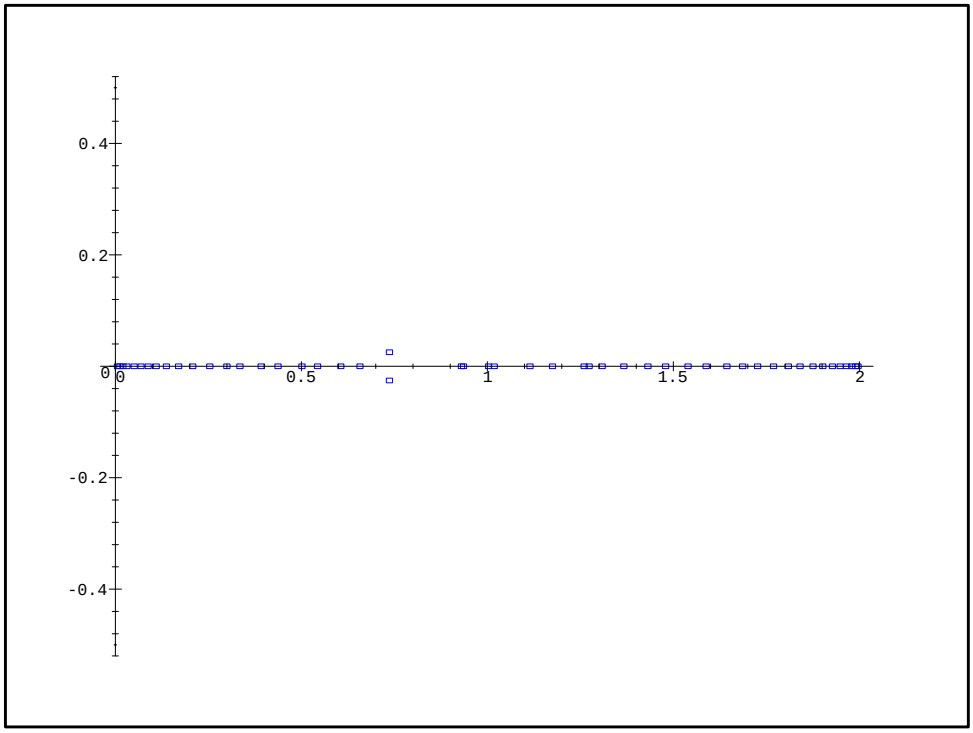
Ανάλυση Τιμών Ritz

Μετά από την παρουσίαση των παραπάνω πειραματικών μετρήσεων της υλοποίησης των non-stationary επαναληπτικών μεθόδων στο collocation γραμμικό σύστημα είναι φανερό, ότι η τεχνική του preconditioning επηρέασε θετικά την επίδοση και την συμπεριφορά κάθε μεθόδου. Αυτό μπορεί να στοιχειοθετηθεί και με την μελέτη των τιμών Ritz. Οι τιμές αυτές είναι οι ιδιοτιμές του reduced πίνακα H, ο οποίος σε άλλες Krylov subspace μεθόδους κατασκευάζεται και χρησιμοποιείται άμεσα και σε άλλες μεθόδους χρησιμοποιείται με έμεσο τρόπο. Έτσι στη μέθοδο GMRES(m) ο πίνακας αυτός είναι το κυρίαρχο τμήμα $H_{m,m}$ του άνω Heissenberg πίνακα $H_{m+1,m}$ πριν αυτός παραγοντοποιηθεί. Έχει αποδειχτεί [109, 118] ότι για μεγάλες τιμές της παραμέτρου m οι τιμές Ritz συγκλίνουν στις ιδιοτιμές του preconitioned πίνακα $M^{-1}A$ του γραμμικού συστήματος. Έτσι αν οι τιμές Ritz χωρίς preconditioning είναι διασκορπισμένες στο μιγαδικό επίπεδο και κυρίως αν η τιμή μηδέν βρίσκεται εντός του χωρίου που τις περικλείει, τότε η εφαρμογή μιας

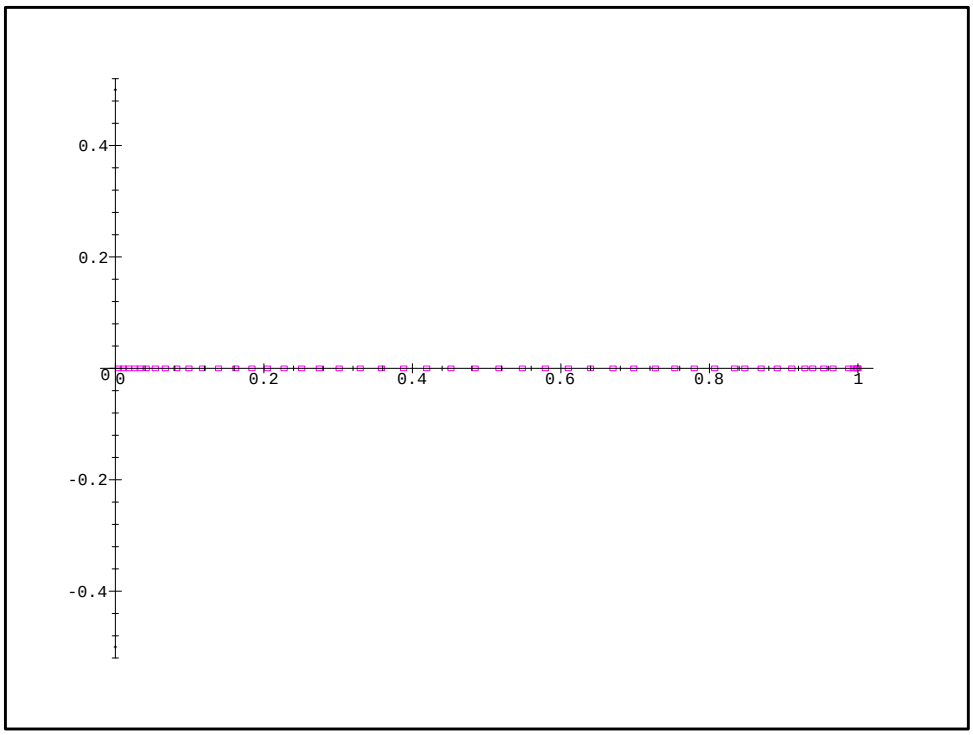
αποδοτικής τεχνικής preconditioning θα έχει ως αποτέλεσμα τον περιορισμό τους σε μια μικρότερη περιοχή του μιγαδικού επιπέδου και αν είναι δυνατόν στο ένα ημιεπίπεδο. Η Εικόνα 43 εμφανίζει τις τιμές Ritz χωρίς preconditioning. Παρατηρούμε ότι όλες οι ιδιοτιμές είναι παραγματικές και μάλιστα ότι η πλειοψηφία τους ανήκει στον αρνητικό ημιάξονα, ενώ υπάρχει και μια θετική. Η Εικόνα 44 παρουσιάζει τις τιμές Ritz μετά την εφαρμογή του Jacobi preconditioning με εκτέλεση υπολογισμών σε block μορφή. Είναι φανερό ότι οι ιδοτιμές έχουν περιοριστεί το πρώτο μιγαδικό ημιεπίπεδο και σε μια περιοχή μεταξύ της αρχής των αξόνων και του παραγματικού άξονα μέχρι την τιμή 2, ενώ υπάρχουν και δυο ιδιοτιμές με φανταστικό μέρος. Οι Εικόνες 45 και 46 παρουσιάζουν τις τιμές Ritz για SGS και SSOR preconditioning και παρατηρούμε ότι οι ιδιοτιμές περιορίστηκαν στον θετικό πραγματικό ημιάξονα και μάλιστα σε μια περιοχή μεταξύ της αρχής του και της μονάδας. Στις Εικόνες 47 και 48 εμφανίζονται οι τιμές Ritz για την περίπτωση που υπήρχε αραιή αποθήκευση στους πίνακες με SGS και SSOR preconditioning. Προκύπτει ότι αυτές δεν κατάφεραν να περιοριστούν σε μια πραγματική περιοχή μεταξύ της αρχής των αξόνων και της μονάδας, όπως στην προηγούμενη περίπτωση, αλλά σε μια λίγο ευρύτερη περιοχή και επιπλέον είναι αξιοσημείωτη η εμφάνιση ιδιοτιμών με φανταστικό μέρος. Αυτό είναι αποτέλεσμα της ατελούς παραγοντοποίησης των preconditioner πινάκων. Όμως πάντοτε η πλειοψηφία των τιμών Ritz βρίσκονταν πιο κοντά στον πραγματικό άξονα και σύμφωνα με όσα έχουν παρατηρηθεί για αυτές τις περιπτώσεις η πιο αποδοτική μέθοδος θεωρείται ότι θα είναι η μέθοδος BiCGSTAB. Αυτό επιβεβαιώθηκε και με τα πιο πάνω πειραματικά αποτελέσματα. Επίσης η ύπαρξη ιδιοτιμών με φανταστικό μέρος μπορεί να δικαιολογήσει της αυξομειώσεις στο ρυθμό μείωσης του σφάλματος του κριτηρίου τερματισμού της μεθόδου BiCGSTAB, όπως παρατηρήθηκε στις Εικόνες 27, 37 και 40, αφού κάτι τέτοιο έχει αρνητικές επιπτώσεις στη συμπεριφορά σύγκλισης της BiCGSTAB [31]. Ακολούθως παρουσιάζονται τα συγκριτικά αποτελέσματα



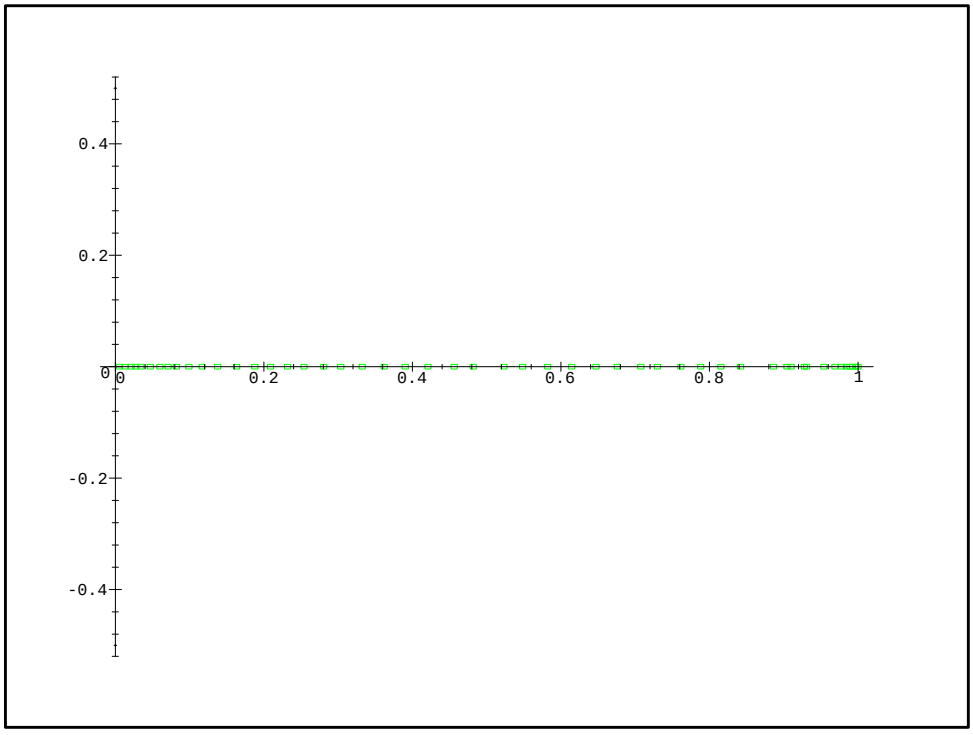
Εικόνα 43 : Οι τιμές Ritz χωρίς preconditioning για $n_s = 64$.



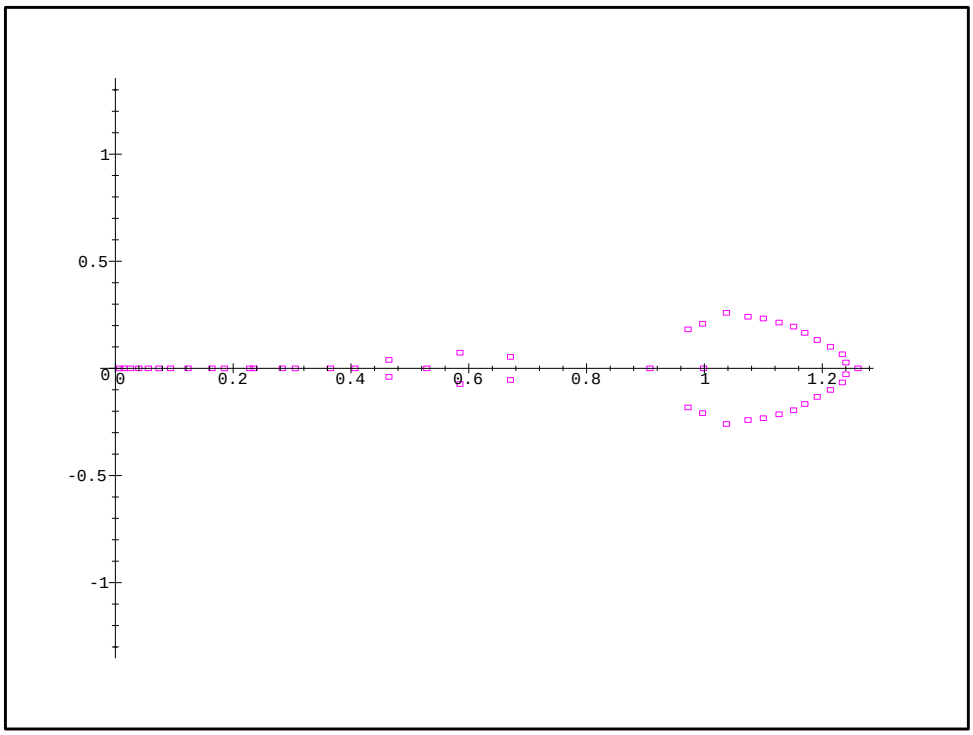
Εικόνα 44 : Οι τιμές Ritz με Jacobi preconditioning για $n_s = 64$ και block αποθήκευση πινάκων.



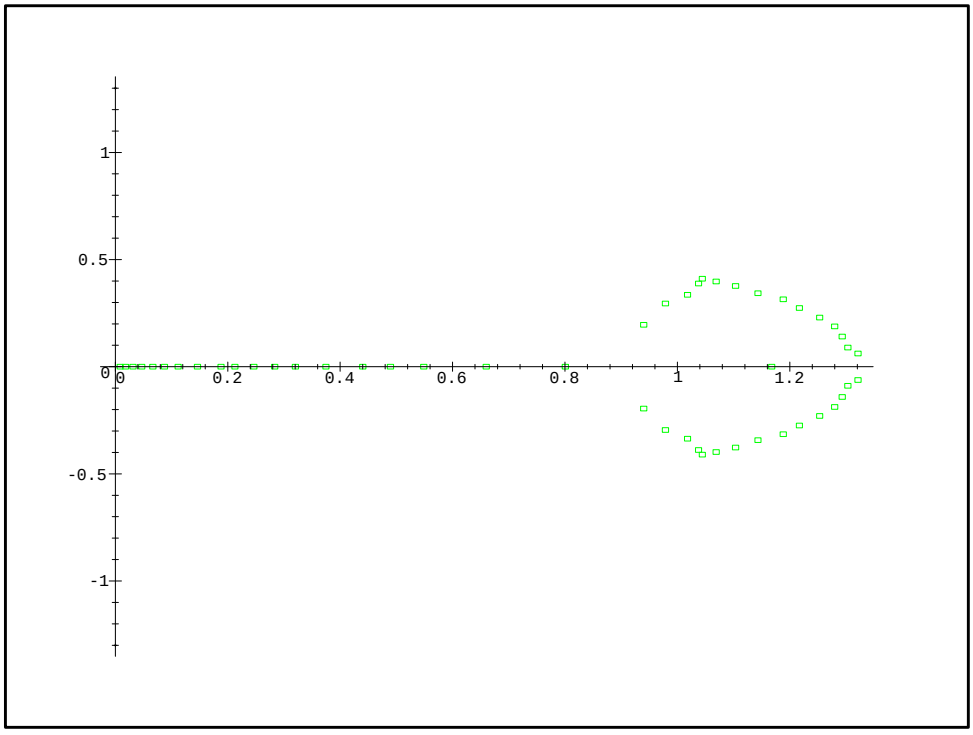
Εικόνα 45 : Οι τιμές Ritz με SGS preconditioning για $n_s = 64$ και block αποθήκευση πινάκων.



Εικόνα 46 : Οι τιμές Ritz με SSOR preconditioning για $n_s = 64$ και block αποθήκευση πινάκων.



Εικόνα 47 : Οι τιμές Ritz με SGS preconditioning για $n_s = 64$ και αραιή αποθήκευση πινάκων.



Εικόνα 48 : Οι τιμές Ritz με SSOR preconditioning για $n_s = 64$ και αραιή αποθήκευση πινάκων.

των μετρήσεων της απόδοσης ανάμεσα στην αποδοτικότερη non-stationary μέθοδο BiCGSTAB με SSOR preconditioning και στην stationary SOR, η οποία στην εργασία [72] εμφάνισε την καλύτερη απόδοση για τις stationary μεθόδους για την ίδια διάσπαση του collocation πίνακα. Σαν μέθοδος αποθήκευσης των πινάκων χρησιμοποιήθηκε η block μορφή ζώνης. Οι μετρήσεις έγιναν σε ένα υπολογιστικό σύστημα Risc αρχιτεκτονικής τύπου Hewlett Packard C3600 με ένα επεξεργαστή χρονισμένο στα 550 MHz και 1.5 MB μνήμης cache on chip. Η συνολική του μνήμη ήταν 512 MB και το λειτουργικό του σύστημα HP-Unix 11.00. Η υλοποίηση έγινε σε Fortan με χρήση διπλής ακρίβειας στους υπολογισμούς.

Ο πίνακας T15 παρουσιάζει τα αποτελέσματα των μετρήσεων της SOR που προέκυψαν για την επίλυση του ίδιου γραμμικού συστήματος με την ίδια αρχική προσέγγιση της λύσης, όπως στις προηγούμενες δοκιμές, ενώ η τιμή της παραμέτρου ω αποτελεί μια αριθμητική προσέγγιση της βέλτιστης τιμής της. Αξίζει να σημειωθεί ότι ως κριτήριο τερματισμού χρησιμοποιήθηκε το

$$\frac{\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\|_\infty}{\|\boldsymbol{x}^{(m+1)}\|_\infty} \leq tol$$

όπου η επιλογή της σταθεράς tol κάθε φορά ήταν τέτοια ώστε κάθε επαναληπτική μέθοδος μετά από ένα ορισμένο αριθμό βημάτων να παράξει μια προσέγγιση της λύσης $\boldsymbol{x}^{(m)}$ με ίδιο περίπου σφάλμα από την πραγματική λύση της διαφορικής εξίσωσης $\| \boldsymbol{u} - \boldsymbol{x}^{(m)} \|_\infty$.

T15	SOR					
n_s	ω	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	1.0600	6	0.0001	6.95e-1	1.97e-4	3.83e-5
8	1.3900	19	0.004	2.03e-2	2.15e-5	2.52e-5
16	1.6150	30	0.019	6.09e-4	2.40e-5	2.41e-5
32	1.7550	55	0.137	3.47e-5	8.63e-6	4.87e-6
64	1.8310	76	0.71	1.55e-5	8.71e-6	5.66e-6
128	1.9040	359	13.98	1.99e-6	4.84e-7	2.79e-7
256	1.9362	1178	202	1.09e-6	9.72e-8	5.66e-8
512	1.9482	4200	2887	9.99e-7	2.66e-8	1.55e-8

Αντίστοιχα στον πίνακα T16 παρουσιάζονται τα ανάλογα αποτελέσματα της BiCGSTAB με SSOR preconditioning. Η επιλογή της παραμέτρου ω δεν είναι η βέλτιστη, αλλά κάποιες τυχαίες τιμές, οι οποίες όμως δεν επιβαρύνουν σημαντικά την μέθοδο με επαναληπτικά βήματα σύμφωνα και με το γράφημα της Εικόνας 29. Ως κριτήριο τερματισμού χρησιμοποιήθηκε το

$$\frac{\|M^{-1}(\boldsymbol{b} - A\boldsymbol{x}^{(m)})\|_2}{\|M^{-1}\boldsymbol{b}\|_2} \leq tol$$

όπου η επιλογή της σταθεράς tol ήταν ανάλογη με αυτήν της SOR.

Παρατηρούμε ότι ο συνολικός χρόνος εκτέλεσης της SOR είναι ελαφρά μειωμένος σε σχέση με τον αντίστοιχο της BiCGSTAB για μεγάλο βήμα διαμέρισης μέχρι περίπου το $\frac{1}{128}$. Όμως στο χρόνο αυτό δεν έχει υπολογιστεί ο χρόνος υπολογισμού της τιμής της παραμέτρου ω, η οποία θα πρέπει να σημειωθεί ότι για ελαφρές αποκλίσεις από την βέλτιστη τιμή της επιβαρύνει σημαντικά την μέθοδο με παραπάνω επαναληπτικά βήματα, το οποίο δεν ισχύει για την παράμετρο ω της μεθόδου BiCGTAB με SSOR preconditioning. Αυτό προκύπτει και από τον αντίστοιχο πίνακα T17, οποίος είναι ο ανάλογος με τους δυο προηγούμενους για SGS preconditioning της BiCGSTAB.

T16	BiCGSTAB - SSOR Preconditioning					
n_s	ω	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	1.3	4	0.004	6.95e-1	2.34e-5	2.49e-6
8	1.3	7	0.008	2.03e-2	2.43e-5	1.48e-5
16	1.3	10	0.04	6.07e-4	2.62e-5	1.08e-5
32	1.4	15	0.22	3.02e-5	8.42e-5	1.57e-4
64	1.3	35	2	1.20e-5	1.87e-6	2.21e-5
128	1.2	78	18.6	1.94e-6	1.21e-7	3.94e-6
256	1.3	158	158	1.11e-6	2.44e-8	3.44e-6
512	1.3	320	1280	9.86e-7	8.25e-9	4.90e-6

T17	BiCGSTAB - SGS Preconditioning				
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	3	0.004	6.95e-1	1.02e-4	1.23e-5
8	6	0.007	2.03e-2	9.09e-4	5.49e-4
16	10	0.04	6.07e-4	3.43e-5	4.85e-5
32	19	0.27	3.02e-5	1.16e-5	6.51e-5
64	39	2.2	1.21e-5	1.25e-6	2.47e-5
128	84	20	1.80e-6	4.42e-8	3.89e-6
256	182	181.6	1.08e-6	8.34e-9	2.29e-6
512	357	1436	8.01e-7	3.90e-8	1.83e-6

Παρατηρούμε ότι η σύγκλιση της BiCGSTAB με SGS preconditioning είναι λίγο πιο αργή από αυτή με SSOR preconditioning, όπως άλλωστε αναμενόταν.

Έτσι κάτω από την σημαντική προϋπόθεση της γνώσης της βέλτιστης παραμέτρου της SOR προκύπτει, ότι είναι προτιμότερο να χρησιμοποιείται έναντι κάθε non-stationary μεθόδου, τουλάχιστον από αυτές που δοκιμάστηκαν στις μετρήσεις του παρόντος κεφαλαίου για διακριτοποιήσεις μέχρι 128 περίπου. Για μεγαλύτερες διακριτοποιήσεις η καλύτερη επιλογή για την επίλυση του collocation γραμμικού συστήματος αποτελεί η επαναληπτική μέθοδος BiCGATAB με SSOR ή και SGS preconditioning με περίπου το μισό χρόνο επίλυσης έναντι της SOR.

4 ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΠΙΛΥΣΗ ΤΩΝ COLLOCATION ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΕ ΠΑΡΑΛΛΗΛΑ ΠΕΡΙΒΑΛΛΟΝΤΑ

4.1 Εισαγωγή

Τις τελευταίες δεκαετίες υπήρξε αλματώδης τεχνολογική εξέλιξη στον τομέα των παράλληλων αρχιτεκτονικών των υπολογιστικών συστημάτων. Η εξέλιξη αυτή επέδρασε άμεσα στους επιστημονικούς υπολογισμούς (scientific computing) [14] και ειδικότερα στις μεθόδους της αριθμητικής επίλυσης των Μερικών Διαφορικών Εξισώσεων [88]. Έτσι μέθοδοι δικριτοποίησης, όπως η μέθοδος Collocation βασισμένη στα Hermite bi-cubic πεπερασμένα στοιχεία [15, 16, 32], άρχισαν να έχουν πρακτική εφαρμογή και ταυτόχρονα να επικεντρώνουν το ερευνητικό ενδιαφέρον των επιστημόνων.

Η επίλυση του παραγόμενου αραιού και πολύ μεγάλης τάξης γραμμικού συστήματος με κάποια άμεση μέθοδο, δεν μπορεί να εφαρμοστεί ικανοποιητικά σε παράλληλα περιβάλλοντα, εξαιτείας του σειριακού χαρακτήρα των αλγορίθμων των άμεσων αυτών μεθόδων. Αντίθετα η επαναληπτική επίλυσή του μπορεί να αποδώσει ικανοποιητικά, επειδή είναι εφικτή η κατάλληλη εκμετάλλευση της σειράς των αριθμητικών πράξεων κάθε επαναληπτικού βήματος.

Όπως έχει ήδη αναφερθεί στα προηγούμενα κεφάλαια η αρίθμηση αγνώστων και εξισώσεων που προτάθηκε στην εργασία [92] οδηγεί στην κατασκευή ενός block τριδιαγωνίου Collocation πίνακα με αντιστρέψιμα block διαγώνια στοιχεία [91]. Η εφαρμογή επαναληπτικών μεθόδων σε παράλληλα περιβάλλοντα στη συγκεκριμένη μορφή του Collocation πίνακα δεν παράγει αποδοτικό αλγόριθμο κι έτσι ο Collocation πίνακας μετασχηματίζεται μέσω ενός μετασχηματισμού ομοιότητας στην 2-cyclic κανονική του μορφή. Αυτό ουσιαστικά είναι ισοδύναμο με μια επαναρίθμηση αγνώστων και εξισώσεων σύμφωνα με την κλασσική μέθοδο red-black, όπως εμφανίστηκε και στην εργασία [90].

Στις εργασίες [93, 90] εμφανίστηκαν υλοποιήσεις βασισμένες στον μετασχηματισμένο Collocation πίνακα διάφορων stationary επαναληπτικών μεθόδων σε συστολικά και VLSI περιβάλλοντα και πιο πρόσφατα [74, 76, 77, 78, 79, 80, 82, 83] σε λογικού τύπου (PVM) [30, 45, 46] παράλληλες μηχανές.

Στις παρακάτω ενότητες μετά την λεπτομερή παρουσίαση του μετασχηματισμού του Collocation πίνακα στην κανονική 2-cyclic μορφή του, εφαρμόζεται ένας επιπλέον μετασχηματισμός, ο οποίος διπλασιάζει τις παράλληλες ιδιότητες του πίνακα. Στη συνέχεια στο παραγόμενο γραμμικό σύστημα εφαρμόζεται η καλλίτερη stationary και non-stationary μέθοδος στα δυο πιο διαδεδομένα παράλληλα περιβάλλοντα. Δηλαδή εφαρμόζονται η SOR και η SSOR preconditioned Bi-CGSTAB μέθοδοι σε αρχιτεκτονικές Κοινής (Shared) και Διανεμημένης (Distributed) μνήμης.

4.2 Red - Black διαμέριση του Collocation πίνακα

Η διαδικασία παραλληλοποίησης της επίλυσης του collocation γραμμικού συστήματος χωρίζεται σε δυο κύριες φάσεις. Στην πρώτη φάση χρησιμοποιούμε με επιτυχία την ιδέα **επαναρίθμησης των αγνώστων και εξισώσεων** για να παραλληλοποιήσομε σ' ένα πρώτο βαθμό το υπολογιστικό πρόβλημα. Στη συνέχεια εφαρμόζουμε έναν

ματασχηματισμό για να βελτιώσουμε το βαθμό παραλληλοποίησης της εφαρμογής.

Για την επαναρίθμηση των αγνώστων και των εξισώσεων χρησιμοποιούμε [59, 125] την γνωστή ιδέα της red - black διάταξης. Σύμφωνα με αυτήν οι άγνωστοι / εξισώσεις χωρίζονται σε red και black υποομάδες αγνώστων / εξισώσεων με την παρακάτω αρχή :

Τα μέλη red υποομάδων "συνορεύουν" μόνο με μέλη black υποομάδων

Η ιδέα της επαναρίθμησης είναι ισοδύναμη με την εφαρμογή ενός μετασχηματισμού ομοιότητας στο γραμμικό σύστημα μέσω κάποιου μεταθετικού πίνακα $P \in \mathbb{R}^{n \times n}$, ο οποίος υπάρχει επειδή ο collocation πίνακας είναι 2-cyclic μορφής [112], ώστε ο πίνακας $P\ A\ P^T$ να έχει την παρακάτω κανονική 2-cyclic μορφή

$$PAP^T = \begin{bmatrix} D_R & -H_B \\ -H_R & D_B \end{bmatrix} , \tag{106}$$

όπου

$$D_R = \text{diag}[A_2 \underbrace{\tilde{A} \cdots \tilde{A}}_{p-1} -A_2] \tag{107}$$

$$H_R = -\text{diag}[A_4 \underbrace{\hat{A} \cdots \hat{A}}_{p-1} -A_4] \tag{108}$$

$$D_B = \text{diag}[\underbrace{\tilde{A} \cdots \tilde{A}}_{p}] \tag{109}$$

$$H_B = -\text{diag}[\underbrace{\hat{A} \cdots \hat{A}}_{p}] \tag{110}$$

$$\tilde{A} = \begin{bmatrix} A_1 & -A_2 \\ A_1 & A_2 \end{bmatrix} \quad \text{και} \quad \hat{A} = \begin{bmatrix} A_3 & -A_4 \\ A_3 & A_4 \end{bmatrix} . \tag{111}$$

Ο μεταθετικός πίνακας P θα έχει την ακόλουθη μορφή

$$P = \begin{bmatrix} I_1 & O & O & O & O & O & \dots & O & O & O & O \\ O & O & I_3 & O & O & O & \dots & O & O & O & O \\ O & O & O & O & I_5 & O & \dots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & O & \dots & O & O & O & O \\ O & O & O & O & O & O & \dots & O & I_{n_s-1} & O & O \\ O & O & O & O & O & O & \dots & O & O & O & I_{n_s+1} \\ O & I_2 & O & O & O & O & \dots & O & O & O & O \\ O & O & O & I_4 & O & O & \dots & O & O & O & O \\ O & O & O & O & O & I_6 & \dots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & O & \dots & I_{n_s-2} & O & O & O \\ O & O & O & O & O & O & \dots & O & O & I_{n_s} & O \end{bmatrix} , \tag{112}$$

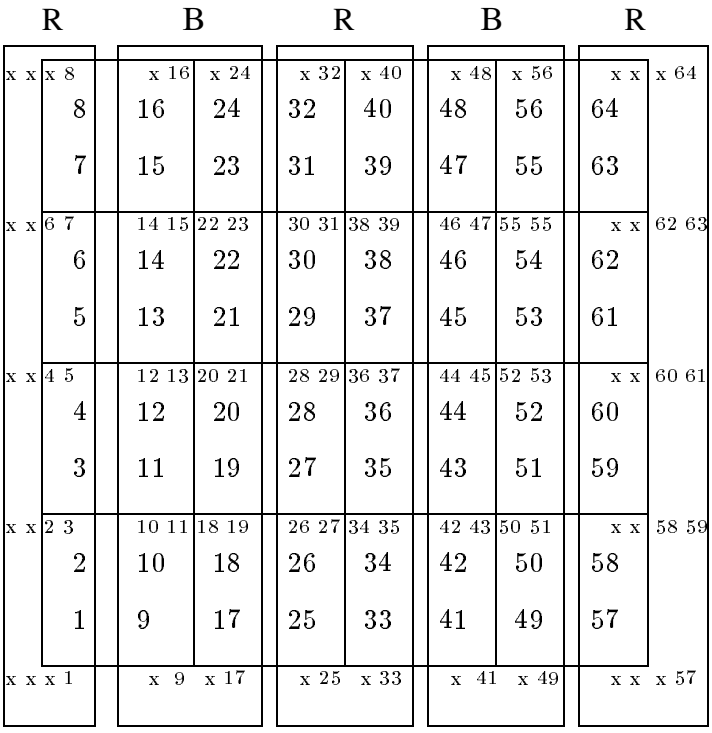
όπου $\mathrm{I}_j \in \mathbb{R}^{\nu,\nu}$ με $\nu = \begin{cases} 2n_s & \text{όταν} \quad j = 1 \text{ και } j = n_s + 1 \\ 4n_s & \text{όταν} \quad j = 2, \dots, n_s \end{cases}$.

Η Εικόνα 49 εμφανίζει την red-black ομαδοποίηση αυτή για την περίπτωση $n_s = 4$. Στην συνέχεια επαναριθμούμε τους αγνώστους και τις εξισώσεις, έτσι ώστε τα μέλη των red υποομάδων να καταλάβουν διαδοχικές θέσεις στο πλέγμα (grid), ακολουθούμενα με την ίδια ιδέα από τα μέλη των black υποομάδων. Στην Εικόνα 50 παρουσιάζεται η νέα αυτή αρίθμηση αγνώστων και εξισώσεων για $n_s = 4$. Η δομή των πινάκων που αντιστοιχούν στην αρίθμηση των Εικόνων 49 και 50 παρουσιάζεται αντίστοιχα στις Εικόνες 51 και 52.

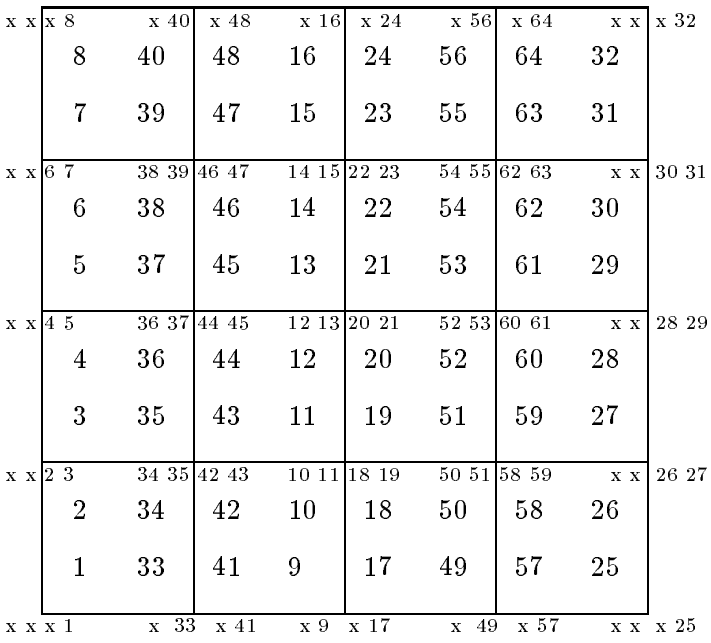
Όπως παρατηρούμε από τη δομή του collocation πίνακα μετά την επαναρίθμηση υπάρχουν διαγώνια blocks κι έτσι έχουμε τη δυνατότητα της απεξάρτησης των αγνώστων μεταξύ τους. Αυτό έχει ως άμεση συνέπεια την αύξηση των *παράλληλων* χαρακτηριστικών του γραμμικού συτήματος, αφού υπάρχει πλέον η δυνατότητα του υπολογισμού των αγνώστων κατά ομάδες, οι οποίες είναι ανεξάρτητες μεταξύ τους.

Έτσι το collocation γραμμικό σύστημα θα έχει την ισοδύναμη μορφή

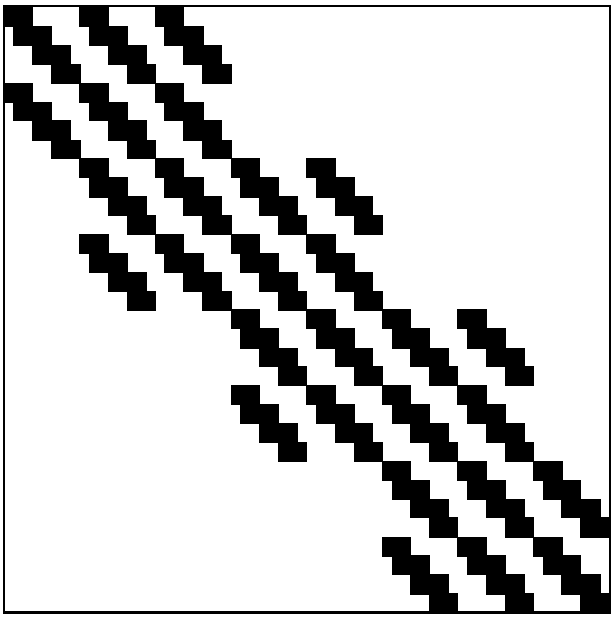
$$PAP^T(P\boldsymbol{x}) = P\boldsymbol{b} \ , \tag{113}$$



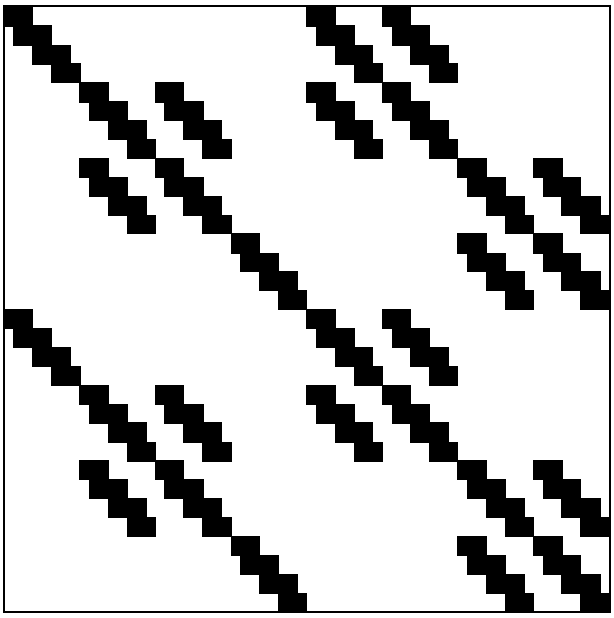
Εικόνα 49 : Red - Black ομαδοποίηση αγνώστων και εξισώσεων



Εικόνα 50 : Red - Black αρίθμηση αγνώστων και εξισώσεων



Εικόνα 51 : Δομή του Block Τριδιαγώνιου Collocation Πίνακα



Εικόνα 52 : Δομή του Red - Black Collocation Πίνακα

όπου

$$PAP^T = \begin{bmatrix} A_2 & O & O & \dots & O & O & O & A_3 & -A_4 & \dots & O & O \\ O & A_1 & -A_2 & \dots & O & O & O & A_3 & A_4 & \dots & O & O \\ O & A_1 & A_2 & \dots & O & O & O & O & O & \dots & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & O & \dots & A_1 & -A_2 & O & O & O & \dots & O & O \\ O & O & O & \dots & A_1 & A_2 & O & O & O & \dots & A_3 & -A_4 \\ O & O & O & \dots & O & O & -A_2 & O & O & \dots & A_3 & A_4 \\ A_4 & O & O & \dots & O & O & O & A_1 & -A_2 & \dots & O & O \\ O & A_3 & -A_4 & \dots & O & O & O & A_1 & A_2 & \dots & O & O \\ O & A_3 & A_4 & \dots & O & O & O & O & O & \dots & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & O & \dots & A_3 & -A_4 & O & O & O & \dots & O & O \\ O & O & O & \dots & A_3 & A_4 & O & O & O & \dots & A_1 & -A_2 \\ O & O & O & \dots & O & O & -A_4 & O & O & \dots & A_1 & A_2 \end{bmatrix} \quad .(114)$$

Αν θεωρήσουμε την παρακάτω διάσπαση του

$$PAP^T \;=\; D_A \;-\; L_A \;-\; U_A \;\;, \tag{115}$$

όπου

$$D_A \;=\; \begin{bmatrix} D_R & O \\ O & D_B \end{bmatrix} \;\;,\;\; L_A \;=\; \begin{bmatrix} O & O \\ H_R & O \end{bmatrix} \;\; \text{και} \;\; U_A \;=\; \begin{bmatrix} O & H_B \\ O & O \end{bmatrix} \;\;, \tag{116}$$

και αν θεωρήσουμε αντίστοιχα τις ανάλογες διαμερίσεις των διανυσμάτων των αγνώστων και του δεξιού μέλους, ώστε

$$P\boldsymbol{x} \;=\; \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} \;\; \text{and} \;\; P\boldsymbol{b} \;=\; \begin{bmatrix} \boldsymbol{b}_R \\ \boldsymbol{b}_B \end{bmatrix} \;\;, \tag{117}$$

τότε κάθε επαναληπτικό βήμα οποιασδήποτε επαναληπτικής μεθόδου χωρίζεται σε δυο φάσεις. Αρχικά υπάρχει η red φάση, όπου υπολογίζεται το red $\boldsymbol{x}_B^{(m+1)}$ τμήμα της προσεγγιστικής λύσης και στη συνέχεια η black για το υπόλοιπο black $\boldsymbol{x}_B^{(m+1)}$ τμήμα.

Εξετάζοντας λεπτομερέστερα την δομή των αντιστρέψιμων [91] block διαγώνιων πινάκων D_R και D_B, είναι φανερό ότι για τον υπολογισμό του $\boldsymbol{x}_R^{(m+1)}$ τμήματος της λύσης του γραμμικού συστήματος θα πρέπει να επιλυθούν δύο υποσυστήματα μεγέθους $2n_s$ και $p-1$ υποσυστήματα μεγέθους $4n_s$, όπου $n_s = 2p$. Για τον

υπολογισμό του υπολοίπου τμήματος της λύσης $\boldsymbol{x}_B^{(m+1)}$ χρειάζεται η επίλυση p υποσυστημάτων μεγέθους $4n_s$ και επειδή τα υποσυστήματα αυτά είναι ανεξάρτητα μεταξύ τους η επίλυσή τους μπορεί να επιτευχθεί παράλληλα. Αυτό έγινε δυνατό με το μετασχηματισμό ομοιότητας που εφαρμόσαμε στο γραμμικό σύστημα μέσω του μεταθετικού πίνακα P.

Όμως είναι δυνατή η παραπέρα απεξάρτηση των αγνώστων με άμεση συνέπεια τον διπλασιασμό του βαθμού παραλληλοποίησης με την εφαρμογή στο γραμμικό σύστημα του πίνακα

$$T \;=\; \text{diag}\,[\,T_R\;T_B\,] \tag{118}$$

με

$$T_R \;=\; -\text{diag}[I\;G\;\cdots\;G\;I] \quad \text{και} \quad T_B \;=\; -\text{diag}[G\;\cdots\;G]\;\;, \tag{119}$$

όπου

$$G \;=\; \frac{1}{2}\begin{bmatrix} I & I \\ -I & I \end{bmatrix} \quad \text{και} \quad I \in \mathbb{R}^{2n_s,2n_s} \;\; \text{είναι ο μοναδιαίος πίνακας} \;\;. \tag{120}$$

Έτσι το γραμμικό σύστημα θα πάρει τη μορφή

$$\begin{bmatrix} T_R & O \\ O & T_B \end{bmatrix}\begin{bmatrix} D_R & -H_B \\ -H_R & D_B \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} \;=\; \begin{bmatrix} T_R & O \\ O & T_B \end{bmatrix}\begin{bmatrix} \boldsymbol{b}_R \\ \boldsymbol{b}_B \end{bmatrix} \;\;. \tag{121}$$

Για τα παραγόμενα γινόμενα πινάκων θα ισχύει ότι

$$T_R D_R = \text{diag}[A_2\;A_1\;A_2\;\cdots\;A_1\;A_2\;\;-A_2] \tag{122}$$

$$T_B D_B = \text{diag}[A_1\;A_2\;\cdots\;A_1\;A_2] \tag{123}$$

$$T_R H_B = -\frac{1}{2}\begin{bmatrix} 2A_3 & -2A_4 & O & O & \cdots & O & O & O & O \\ A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 \\ O & O & O & O & \cdots & O & O & 2A_3 & 2A_4 \end{bmatrix} \tag{124}$$

$$T_B H_R = -\frac{1}{2}\begin{bmatrix} A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ -A_4 & A_3 & -A_4 & O & O & \cdots & O & O & O & O & O \\ O & A_3 & A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ O & -A_3 & -A_4 & A_3 & -A_4 & \cdots & O & O & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & O & \cdots & A_3 & A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & -A_3 & -A_4 & A_3 & -A_4 & O \\ O & O & O & O & O & \cdots & O & O & A_3 & A_4 & -A_4 \\ O & O & O & O & O & \cdots & O & O & -A_3 & -A_4 & -A_4 \end{bmatrix}, \quad (125)$$

και

$$T_R \boldsymbol{b}_R = \frac{1}{2}\begin{bmatrix} 2\boldsymbol{b}_1 \\ \boldsymbol{b}_2 + \boldsymbol{b}_3 \\ \boldsymbol{b}_3 - \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{2p-2} + \boldsymbol{b}_{2p-1} \\ \boldsymbol{b}_{2p-1} - \boldsymbol{b}_{2p-2} \\ 2\boldsymbol{b}_{2p} \end{bmatrix}, \quad T_B \boldsymbol{b}_B = \frac{1}{2}\begin{bmatrix} \boldsymbol{b}_{2p+1} + \boldsymbol{b}_{2p+2} \\ \boldsymbol{b}_{2p+2} - \boldsymbol{b}_{2p+1} \\ \vdots \\ \boldsymbol{b}_{4p-1} + \boldsymbol{b}_{4p} \\ \boldsymbol{b}_{4p} - \boldsymbol{b}_{4p-1} \end{bmatrix}. \quad (126)$$

Επειδή, οι πίνακες T_R και T_B είναι αντιστρέψιμοι και μελετώντας τη μορφή των block διαγώνιων πινάκων $T_R D_R$ και $T_B D_B$ είναι φανερό, ότι για τον υπολογισμό του πρώτου τμήματος της λύσης $\boldsymbol{x}_R^{(m+1)}$ του γραμμικού συστήματος, απαιτείται η λύση $2p$ υποσυστημάτων μεγέθους $4n_s$ το καθένα. Επίσης για τον υπολογισμό του υπόλοιπου τμήματος $\boldsymbol{x}_B^{(m+1)}$ χρειάζεται η επίλυση επιπλέον $2p$ υποσυστημάτων μεγέθους $4n_s$ το καθένα. Άρα με την εφαρμογή του μετασχηματισμού μέσω του πίνακα T **διπλασιάστηκε** ο βαθμός παραλληλοποίησης του γραμμικού συστήματος.

Στο σημείο αυτό θα πρέπει να αναφερθεί ότι η εφαρμογή της επαναληπτικής μεθόδου SOR στο red-black collocation γραμμικό σύστημα θα παρουσιάσει την ίδια ακριβώς συμπεριφορά με την εφαρμογή της στο block τριδιαγώνιο σύστημα, αφού και οι δυο περιπτώσεις βασίστηκαν στην ίδια διαμέριση. Έτσι η μέθοδος αυτή θα συγκλίνει με τον ίδιο ρυθμό και για τις δυο μορφές του collocation γραμμικού

συστήματος, καθώς επίσης θα ισχύουν οι ίδιες τιμές της βέλτιστης τιμής της παραμέτρου ω_{opt}. Αυτό συμβαίνει γιατί οι δυο μετασχηματισμοί που εφαρμόστηκαν μέσω των πινάκων P και T δεν επηρέασαν τις ιδιοτιμές του επαναληπτικού πίνακα του Jacobi J.

Πραγματικά ας θεωρήσουμε τον collocation πίνακα A σε block τριδιαγώνια μορφή και κάνοντας χρήση της γνωστής διαμέρισής του θα έχουμε την block διάσπασή του

$$A = D - L - U \quad ,$$

όπου ο πίνακας D είναι ο block τριδιαγώνιος που προκύπτει από την διαμέριση του A και $-L$, $-U$ οι αυστηρά κάτω και άνω block πίνακες αντίστοιχα. Ο επαναληπτικός πίνακας της Jacobi μεθόδου για την ίδια διαμέριση του A θα έχει την μορφή

$$J = D^{-1}(L + U) \quad .$$

Η εφαρμογή του μετασχηματισμού ομοιότητας στο γραμμικό σύστημα μέσω του μεταθετικού πίνακα P θα επιδράσει αρχικά στον collocation πίνακα ως εξής PAP^T και στη συνέχεια η εφαρμογή του επόμενου μετασχηματισμού με τον πίνακα T θα επιδράσει τελικά στη διάσπασή του ως

$$T(PAP^T) = T[P(D - L - U)P^T] = TPDP^T - TPLP^T - TPUP^T$$

κι έτσι ο αντίστοιχος επαναληπτικός πίνακας J_{rb} της Jacobi μεθόδου θα έχει τη μορφή

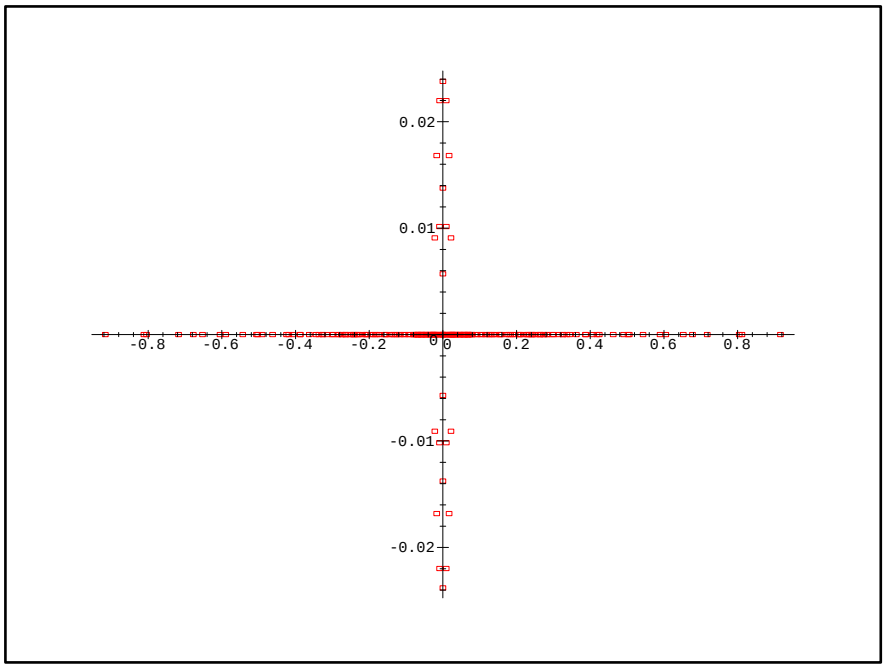
$$J_{rb} = (TPDP^T)^{-1}(TPLP^T + TPUP^T) = (PDP^T)^{-1}T^{-1}(TPL + TPU)P^T =$$

$$= (DP^T)^{-1}P^{-1}(T^{-1}TPL + T^{-1}TPU)P^T = (P^T)^{-1}D^{-1}(P^{-1}PL + P^{-1}PU)P^T =$$

$$= P[D^{-1}(L + U)]P^T = PJP^T \quad .$$

Επειδή ο πίνακας J_{rb} αποτελεί το αποτέλεσμα της εφαρμογής ενός μετασχηματισμού ομοιότητας στον πίνακα J θα ισχύει ότι οι δυο αυτοί επαναληπτικοί πίνακες θα

έχουν τις ίδιες ιδιοτιμές. Πράγματι το επόμενο γράφημα εμφανίζει τις ιδιοτιμές αυτές για την περίπτωση διακριτοποίησης με $n_s = 8$ υποδιαστήματα, οι οποίες δεν μεταβλήθηκαν κατά την εφαρμογή των μετασχηματισμών στο collocation γραμμικό σύστημα.



Η ίδια ακριβώς σχέση συνδέει και τους επαναληπτικούς πίνακες της SOR. Πράγματι αν $\mathcal{L}$ είναι επαναληπτικός πίνακας της SOR για την block τριδιαγώνια μορφή του collocation γραμμικού συστήματος θα έχει τη μορφή

$$\mathcal{L} = (D - \omega L)^{-1}[\omega U + (1 - \omega)D]$$

τότε ο επαναληπτικός πίνακας της SOR για την red black μορφή θα είναι

$$\mathcal{L}_{rb} = (TPDP^T - \omega TPLP^T)^{-1}[\omega TPUP^T + (1 - \omega)TPDP^T] =$$

$$[T(PD - \omega PL)P^T]^{-1}[\omega TPU + (1 - \omega)TPD]P^T =$$

$$[(PD - \omega PL)P^T]^{-1}T^{-1}[\omega TPU + (1 - \omega)TPD]P^T =$$

$$P[P(D-\omega L)]^{-1}[\omega PU+(1-\omega)PD]P^T=$$

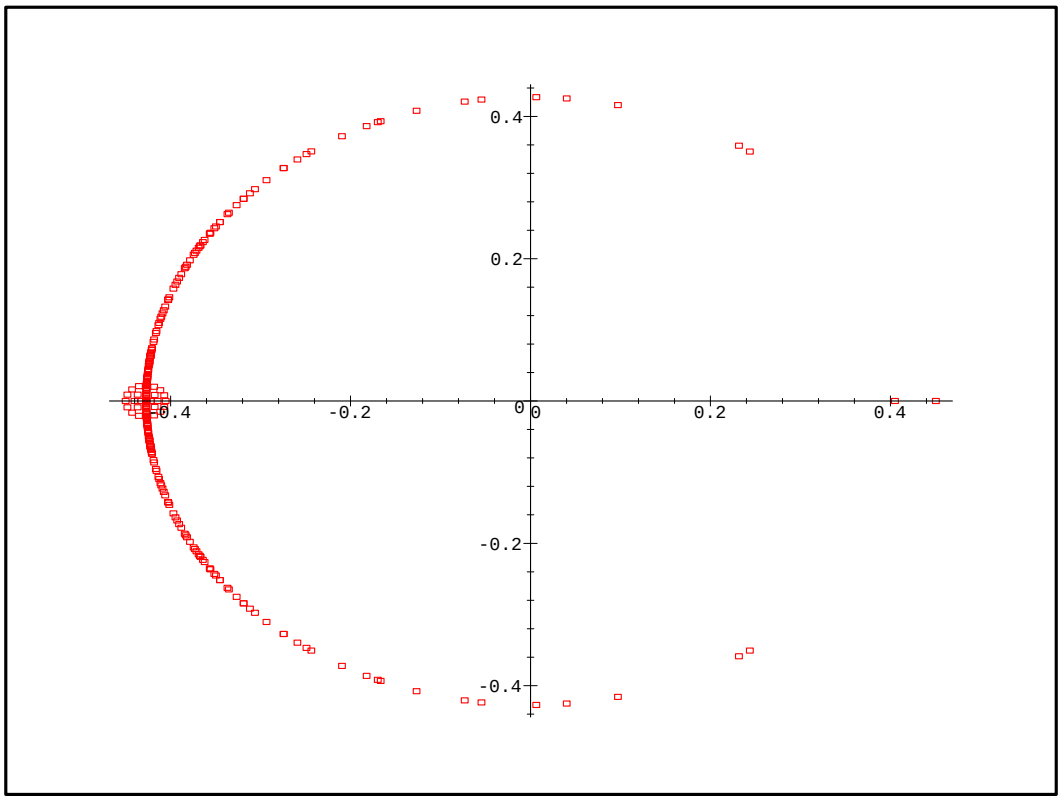
$$P(D-\omega L)^{-1}P^{-1}[\omega PU+(1-\omega)PD]P^T=$$

$$P[(D-\omega L)^{-1}[\omega U+(1-\omega)D]]P^T=P\mathcal{L}P^T.$$

Στο επόμενο γράφημα έχουν υπολογιστεί οι ιδιοτιμές του επαναληπτικού πίνακα της SOR για διακριτοποίηση με $n_s=8$ υποδιαστήματα, αφού η τιμή της βέλτιστης παραμέτρου ω_{opt} υπολογίστηκε από την γνωστή σχέση [72]

$$\omega_{opt}=\frac{2}{1+(1+b^2-a^2)^{\frac{1}{2}}} \quad ,$$

όπου b και a είναι το μήκος των αξόνων της έλλειψης που περικλείει όλες τις ιδιοτιμές του Jacobi επαναληπτικού πίνακα και για $n_s=8$ είναι $a=0.91621$ και $b=0.02379$, οπότε $\omega_{opt}=1.4271$.



4.3 Εφαρμογή της επαναληπτικής μεθόδου SOR σε παράλληλες αρχιτεκτονικές

Στην προηγούμενη ενότητα έγινε μετασχηματισμός του collocation πίνακα στην κανονική red-black 2-cyclic μορφή του και στη συνέχεια έγινε εφαρμογή ενός επιπλέον μετασχηματισμού T στο παραγόμενο γραμμικό σύστημα με σκοπό την επιπλέον αύξηση των παράλληλων χαρακτηριστικών του. Αυτό είχε άμεση συνέπεια την δυνατότητα κατασκευής ενός αποδοτικού παράλληλου αλγορίθμου κατά την εφαρμογή της επαναληπτικής μεθόδου SOR στο collocation γραμμικό σύστημα (122). Έτσι το επαναληπτικό βήμα της μεθόδου SOR θα έχει την ακόλουθη μορφή

$$\begin{cases} T_R D_R \boldsymbol{x}_R^{(m+1)} &= (1-\omega) T_R D_R \boldsymbol{x}_R^{(m)} + \omega T_R H_B \boldsymbol{x}_B^{(m)} + \omega T_R \boldsymbol{b}_R \\ T_B D_B \boldsymbol{x}_B^{(m+1)} &= (1-\omega) T_B D_B \boldsymbol{x}_B^{(m)} + \omega T_B H_R \boldsymbol{x}_R^{(m+1)} + \omega T_B \boldsymbol{b}_B \end{cases} \tag{127}$$

ή ισοδύναμα

$$\begin{cases} T_R D_R \widehat{\boldsymbol{x}}_R &= T_R H_B \boldsymbol{x}_B^{(m)} + T_R \boldsymbol{b}_R \\ T_B D_B \widehat{\boldsymbol{x}}_B &= T_B H_R \boldsymbol{x}_R^{(m+1)} + T_B \boldsymbol{b}_B \end{cases} \tag{128}$$

όπου

$$\widehat{\boldsymbol{x}} = \frac{1}{\omega}[\boldsymbol{x}^{(m+1)} - (1-\omega)\boldsymbol{x}^{(m)}] \ .$$

Αν θεωρήσουμε την ομοιόμορφη block διαμέριση μήκους $n_s = 2p$ των παρακάτω διανυσμάτων

$$\boldsymbol{x}_R = [\boldsymbol{x}_1^T \ \boldsymbol{x}_2^T \ \cdots \ \boldsymbol{x}_{2p}^T]^T \quad \text{και} \quad \boldsymbol{x}_B = [\boldsymbol{x}_{2p+1}^T \ \boldsymbol{x}_{2p+2}^T \ \cdots \ \boldsymbol{x}_{4p}^T]^T \tag{129}$$

$$\boldsymbol{b}_R = [\boldsymbol{b}_1^T \ \boldsymbol{b}_2^T \ \cdots \ \boldsymbol{b}_{2p}^T]^T \quad \text{και} \quad \boldsymbol{b}_B = [\boldsymbol{b}_{2p+1}^T \ \boldsymbol{b}_{2p+2}^T \ \cdots \ \boldsymbol{b}_{4p}^T]^T \quad , \tag{130}$$

μπορούμε να καταλήξουμε στον παρακάτω αλγόριθμο, ο οποίος περιγράφει την επαναληπτική μέθοδο SOR, όπως αυτή εφαρμόζεται στο collocation γραμμικό σύστημα

Αλγόριθμος SOR

For $m = 1$ *to* $maxit$ or *until* <convergence> *do*

S1: *Solve* $A_2\widehat{\boldsymbol{x}}_1 = -A_3\boldsymbol{x}_{2p+1}^{(m)} + A_4\boldsymbol{x}_{2p+2}^{(m)} + \boldsymbol{b}_1$

For $j = 1$ *to* $p-1$ *do*

S2: *Solve*
$$2A_1\widehat{\boldsymbol{x}}_{2j} = -A_3[\boldsymbol{x}_{2(j+p)-1}^{(m)} + \boldsymbol{x}_{2(j+p)+1}^{(m)}] + \\ +A_4[\boldsymbol{x}_{2(j+p)+2}^{(m)} - \boldsymbol{x}_{2(j+p)}^{(m)}] + (\boldsymbol{b}_{2j} + \boldsymbol{b}_{2j+1})$$

S3: *Solve*
$$2A_2\widehat{\boldsymbol{x}}_{2j+1} = A_3[\boldsymbol{x}_{2(j+p)-1}^{(m)} - \boldsymbol{x}_{2(j+p)+1}^{(m)}] + \\ +A_4[\boldsymbol{x}_{2(j+p)+2}^{(m)} + \boldsymbol{x}_{2(j+p)}^{(m)}] + (\boldsymbol{b}_{2j+1} - \boldsymbol{b}_{2j})$$

Enddo

S4: *Solve* $A_2\widehat{\boldsymbol{x}}_{2p} = A_3\boldsymbol{x}_{4p-1}^{(m)} + A_4\boldsymbol{x}_{4p}^{(m)} - \boldsymbol{b}_{2p}$

For $j = 1$ *to* $2p$ *do*

S5: $\boldsymbol{x}_j^{(m+1)} = \omega\widehat{\boldsymbol{x}}_j + (1-\omega)\boldsymbol{x}_j^{(m)}$

Enddo

S6: *Solve*
$$2A_1\widehat{\boldsymbol{x}}_{2p+1} = -A_3\boldsymbol{x}_2^{(m+1)} + A_4[\boldsymbol{x}_3^{(m+1)} - \boldsymbol{x}_1^{(m+1)}] \\ +(\boldsymbol{b}_{2p+1} + \boldsymbol{b}_{2p+2})$$

S7: *Solve*
$$2A_2\widehat{\boldsymbol{x}}_{2p+2} = -A_3\boldsymbol{x}_2^{(m+1)} + A_4[\boldsymbol{x}_3^{(m+1)} + \boldsymbol{x}_1^{(m+1)}] \\ +(\boldsymbol{b}_{2p+2} - \boldsymbol{b}_{2p+1})$$

For $j = 2$ *to* $p-1$ *do*

S8: *Solve*
$$2A_1\widehat{\boldsymbol{x}}_{2(j+p)-1} = -A_3[\boldsymbol{x}_{2j-2}^{(m+1)} + \boldsymbol{x}_{2j}^{(m+1)}] + A_4[\boldsymbol{x}_{2j+1}^{(m+1)} - \boldsymbol{x}_{2j-1}^{(m+1)}] \\ +[\boldsymbol{b}_{2(j+p)-1} + \boldsymbol{b}_{2(j+p)}]$$

S9: $\underline{Solve}\ \ 2A_2\widehat{\boldsymbol{x}}_{2(j+p)} = A_3[\boldsymbol{x}_{2j-2}^{(m+1)} - \boldsymbol{x}_{2j}^{(m+1)}] + A_4[\boldsymbol{x}_{2j+1}^{(m+1)} + \boldsymbol{x}_{2j-1}^{(m+1)}]$
$+[\boldsymbol{b}_{2(j+p)} - \boldsymbol{b}_{2(j+p)-1}]$

$\underline{Enddo}$

S10: $\underline{Solve}\ \ 2A_1\widehat{\boldsymbol{x}}_{4p-1} = -A_3\boldsymbol{x}_{2p-2}^{(m+1)} + A_4[\boldsymbol{x}_{2p}^{(m+1)} - \boldsymbol{x}_{2p-1}^{(m+1)}]$
$+(\boldsymbol{b}_{4p-1} + \boldsymbol{b}_{4p})$

S11: $\underline{Solve}\ \ 2A_2\widehat{\boldsymbol{x}}_{4p} = A_3\boldsymbol{x}_{2p-2}^{(m+1)} + A_4[\boldsymbol{x}_{2p}^{(m+1)} + \boldsymbol{x}_{2p-1}^{(m+1)}]$
$+(\boldsymbol{b}_{4p} - \boldsymbol{b}_{4p-1})$

$\underline{For}\quad j = 2p+1\quad \underline{to}\quad 4p\quad \underline{do}$

S12: $\boldsymbol{x}_j^{(m+1)} = \omega\widehat{\boldsymbol{x}}_j + (1-\omega)\boldsymbol{x}_j^{(m)}$

$\underline{Enddo}$

S13: $\underline{compute}$ the $error\ norm$ and set <convergence>

enddo

Είναι φανερό ότι στη σειριακή μορφή του παραπάνω αλγορίθμου κάθε επαναληπτικό βήμα περιλαμβάνει τον πολλαπλασιασμό κάποιου διανύσματος με έναν από τους πίνακες A_3 ή A_4 και την LU επίλυση κάποιου γραμμικού συστήματος με πίνακα συντελεστών των αγνώστων τους πίνακες A_1 ή A_2. Για την μείωση των πράξεων στο ελάχιστο θα πρέπει να ληφθεί υπόψη η δομή των πινάκων αυτών, δηλαδή ότι είναι πίνακες ζώνης με εύρος 5. Επίσης οι πίνακες ζώνης A_1 και A_2 θα παραγοντοποιηθούν μια φορά στην αρχή της διαδικασίας κι έτσι σε κάθε επαναληπτικό βήμα θα εκτελείται μόνο η μπρος πίσω αντικατάσταση. Έτσι οι συνολικές πράξεις σε κάθε επαναληπτικό βήμα είναι $O(n_s)$.

Εφαρμογή - Υλοποίηση του Red-Black αλγορίθμου SOR σε σειριακή μορή

Ο παραπάνω αλγόριθμος της SOR εφαρμόστηκε και μελετήθηκε η συμπεριφορά του σε σειριακή μορφή σε σχέση με τον αντίστοιχο αλγόριθμο που προέκυψε από την διάσπαση του collocation πίνακα σε block τριδιαγώνια μορφή. Η υλοποιήση έγινε στο ίδιο υπολογιστικό σύστημα HP C3600 που χρησιμοποιήθηκε για την μελέτη των σειριακών αλγορίθμων των non-stationary επαναληπτικών μεθόδων και της SOR, τα αποτελέσματα των οποίων παρουσιάστηκαν στο τέλος του προηγούμενου κεφαλαίου. Φυσικά χρησιμοποιήθηκε το ίδιο ακριβώς πρόβλημα για τις ίδιες ακριβώς περιπτώσεις. Τα αποτελέσματα των μετρήσεων της εφαρμογής της SOR στο Red-Black collocation γραμμικό σύστημα παρουσιάζονται στον παρακάτω πίνακα T18

T18	Block Red-Black SOR					
n_s	ω	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	1.1700	10	0.0001	6.95e-1	1.42e-4	1.48e-5
8	1.4140	21	0.004	2.03e-2	4.24e-5	3.02e-5
16	1.6250	35	0.023	6.09e-4	2.85e-5	1.57e-5
32	1.7700	51	0.12	3.21e-5	7.89e-5	3.55e-5
64	1.8750	98	0.87	1.52e-5	4.42e-6	1.30e-6
128	1.9342	281	10.8	1.98e-6	1.58e-6	4.70e-7
256	1.9496	1032	187	1.07e-6	2.46e-7	7.25e-8
512	1.9530	4054	3020	9.99e-7	5.87e-8	1.72e-8

Θα πρέπει να σημειωθεί ότι η τιμή της παραμέτρου ω, η οποία εμφανίζεται στον παραπάνω πίνακα αποτελεί μια αριθμητική προσέγγιση της βέλτιστης τιμής. Έτσι τα παραπάνω αποτελέσματα είναι ενδεικτικά της συμπεριφοράς της SOR σε σειριακό επίπεδο στο Red-Black collocation γραμμικό σύστημα. Στη συνέχεια παρουσιάζεται η εφαρμογή της σε παράλληλες αρχιτεκτονικές.

4.3.1 Εφαρμογή της επαναληπτικής μεθόδου SOR σε παράλληλες αρχιτεκτονικές κοινής μνήμης

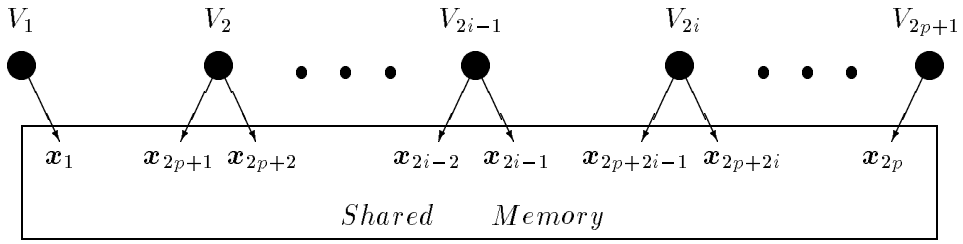
Ο βασικότερος παράγοντας κατά την κατασκευή ενός παράλληλου αλγορίθμου είναι η αρχιτεκτονική του υπολογιστικού συστήματος στο οποίο θα γίνει η υλοποίηση. Έτσι το είδος και ο αριθμός των επεξεργαστών καθώς ο τρόπος σύνδεσης τους καθορίζει τη φιλοσοφία σχεδιασμού του αλγορίθμου. Στη συγκεκριμένη περίπτωση θεωρούμε ότι το παράλληλο υπολογιστικό σύστημα είναι κοινής μνήμης - *shared memory* - κι έτσι θα έχει τα παρακάτω χαρακτηριστικά :

- Ο τύπος των επεξεργαστών είναι ίδιος γι όλους και καθένας τους έχει διαθέσιμη όλη τη μνήμη του υπολογιστικού συστήματος οποιαδήποτε στιγμή.
- Κάθε επεξεργαστής έχει σημαντική υπολογιστική ισχύ.
- Κάθε επεξεργαστής έχει τον ίδιο χρόνο πρόσβασης στην κοινή μνήμη με τους υπόλοιπους. Έτσι το επικοινωνιακό κόστος εξαρτάται αποκλειστικά από το υπολογιστικό σύστημα και ο χρήστης δεν έχει την δυνατότητα να το διαχειριστεί.
- Κάθε επεξεργαστής μπορεί να χρησιμοποιεί για ανάγνωση τα κοινά δεδομένα την ίδια χρονική στιγμή με οποιονδήποτε άλλο, αρκεί να μην τροποποιεί κοινή θέση μνήμης ταυτόχρονα με άλλο επεξεργαστή.
- Δεν θα πρέπει να υπάρχουν αδρανείς επεξεργαστές, γιατί έτσι θα μειωθεί η απόδοση του παράλληλου αλγορίθμου. Αυτό προϋποθέτει σωστή κατανομή υπολογιστικού κόστους.

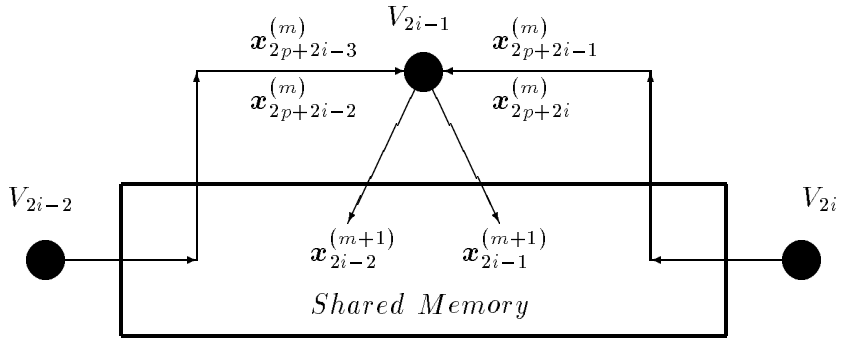
Έχοντας υπόψη τις παραπάνω επισημάνσεις θα γίνει μια αρχική σχεδίαση ενός παράλληλου αλγορίθμου σε ένα εικονικό παράλληλο σύστημα κοινής μνήμης με απεριόριστο αριθμό επεξεργαστών και στη συνέχεια θα γίνει τροποποίηση του αλγορίθμου για παράλληλα συστήματα με σταθερό αριθμό επεξεργαστών.

Απεικόνιση του αλγορίθμου σε ένα εικονικό παράλληλο σύστημα κοινής μνήμης

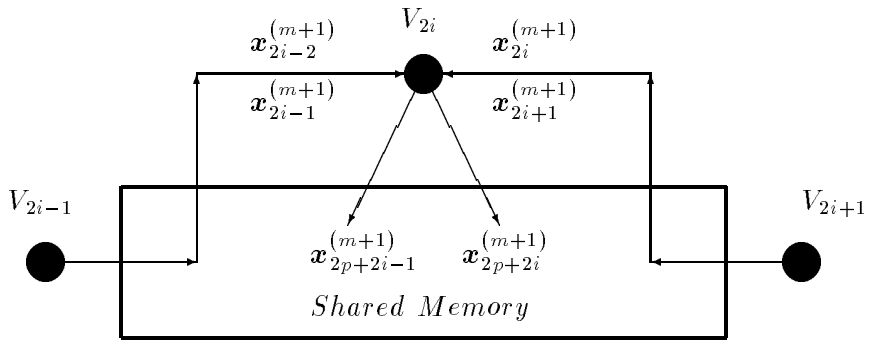
Η απεικόνιση του αλγορίθμου σε ένα εικονικό παράλληλο σύστημα διαφέρει από αυτή πραγματικού μόνο ως προς το πλήθος των επεξεργαστών του γι' αυτό θεωρούμε ότι έχουμε διαθέσιμους όσους επεξεργαστές απαιτείται. Αν θεωρήσουμε, ότι έχει γίνει διαμέριση του χωρίου Ω σε $n_s = 2p$ άρτιο το πλήθος υποδιαστήματα ως προς την x και την y κατεύθυνση, δηλαδή υπάρχουν $2p+1$ κάθετες και άλλες τόσες οριζόντιες γραμμές πλέγματος, μπορούμε να απεικονίσουμε τους αγνώστους που αντιστοιχούν σε κάθε κάθετη γραμμή πλέγματος σε καθένα επεξεργαστή. Σύνδεση των επεξεργαστών μεταξύ τους δεν απαιτείται, εκτός από την άμεση σύνδεσή τους με την κοινή μνήμη. Το παρακάτω σχήμα παρουσιάζει τους V_j , $j = 1, \ldots, 2p+1$ επεξεργαστές και την απεικόνιση σε αυτούς των δυο τμημάτων της λύσης του γραμμικού συστήματος που αντιστοιχεί από την δεξιά και αριστερή πλευρά κάθε κάθες γραμμής πλέγματος.



Παρατηρούμε ότι στους *περιττούς* επεξεργαστές έχουν απεικονισθεί οι *red* στήλες πλέγματος και στους *άρτιους* οι *black*. Επίσης, σύμφωνα με την block διαμέριση μεγέθους $2p$ όλων των διανυσμάτων της προηγούμενης ενότητας, σε κάθε περιττό επεξεργαστή V_{2i-1} , $i = 1, \ldots, p+1$, έχει απεικονισθεί ο υπολογισμός των $\boldsymbol{x}_{2i-2}$ και $\boldsymbol{x}_{2i-1}$ τμημάτων της λύσης, ενώ σε κάθε άρτιο επεξεργαστή V_{2i} , $i = 1, \ldots, p$, έχουν απεικονισθεί τα τμήματα $\boldsymbol{x}_{2p+2i-1}$ και $\boldsymbol{x}_{2p+2i}$. Η εμφανιζόμενη ανομοιομορφία στον πρώτο V_1 και τελευταίο V_{2p+1} επεξεργαστή είναι αποτέλεσμα των συνοριακών συνθηκών της διαφορικής εξίσωσης.



Παρατηρούμε επίσης ότι για να υπολογίσει κάθε περιττός (red) επεξεργαστής V_{2i-1} την νέα προσέγγιση $\boldsymbol{x}^{(m+1)}_{2i-2}$ και $\boldsymbol{x}^{(m+1)}_{2i-1}$ της λύσης, σύμφωνα με τα βήματα S2 και S3 του αλγορίθμου SOR της προηγούμενης ενότητας, θα χρειαστεί να έχουν ήδη υπολογίσει οι γειτονικοί $black$ V_{2i-2} και V_{2i} τα διανύσματα $\boldsymbol{x}^{(m)}_{2p+2i-3}$, $\boldsymbol{x}^{(m)}_{2p+2i-2}$ και $\boldsymbol{x}^{(m)}_{2p+2i-1}$, $\boldsymbol{x}^{(m)}_{2p+2i}$ αντίστοιχα για την κατασκευή του δεύτερου μέλους των υποσυστημάτων.



Ομοίως, οι άρτιοι $black$ επεξεργαστές V_{2i} για να υπολογίσουν τα νέα τμήματα $\boldsymbol{x}^{(m+1)}_{2p+2i-1}$ και $\boldsymbol{x}^{(m+1)}_{2p+2i}$ της προσεγγιστικής λύσης, σύμφωνα με τα βήματα S8 και S9 του αλγορίθμου της SOR, θα πρέπει να έχουν ήδη υπολογίσει οι γειτονικοί red V_{2i-1} και V_{2i+1} επεξεργαστές τα ζεύγη διανυσμάτων $\boldsymbol{x}^{(m+1)}_{2i}$, $\boldsymbol{x}^{(m+1)}_{2i-1}$ και $\boldsymbol{x}^{(m+1)}_{2i}$, $\boldsymbol{x}^{(m+1)}_{2i+1}$ αντίστοιχα.

Συνδυάζοντας τα παραπάνω με την αρχιτεκτονική της εικονικής παράλληλης μηχανής κοινής μνήμης το επαναληπτικό βήμα της SOR σε παράλληλη μορφή θα περιγράφεται με τον παρακάτω αλγόριθμο :

for $m = 0, \ldots, maxit$ or *until* <convergence> *do*

execute Red cycle

execute Black cycle

compute the $error\ norm$ and set <convergence>

enddo

όπου Red και Black Cycles ορίζονται να είναι οι παρακάτω υπολογιστικές διαδικασίες:

Red Cycle

Computation Phase

for $i = 1$ *to* $p + 1$ *do in parallel*

V_{2i-1} *computes* $\boldsymbol{x}_{2i-2}^{(m+1)}$ and $\boldsymbol{x}_{2i-1}^{(m+1)}$

enddo

Black Cycle

Computation Phase

for $i = 1$ *to* p *do in parallel*

V_{2i} *computes* $\boldsymbol{x}_{2p+2i-1}^{(m+1)}$ and $\boldsymbol{x}_{2p+2i}^{(m+1)}$

enddo

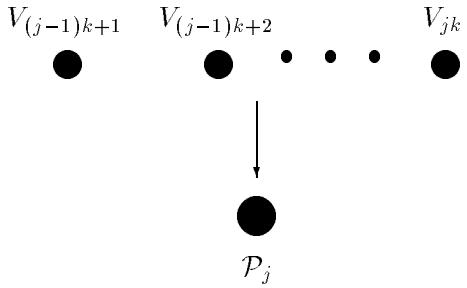
Από τον παραπάνω παράλληλο αλγόριθμο προκύπτει, ότι το υπολογιστικό κόστος $t_{comp}^{(m)}$ στο επαναληπτικό βήμα m της μεθόδου SOR είναι $O(n_s)$. Αυτό ισχύει, γιατί οι βασικές πράξεις γραμμικής άλγεβρας για τον υπολογισμό κάθε τμήματος της λύσης, περιλαμβάνουν κυρίως πολλαπλασιασμούς των πινάκων ζώνης A_3 και A_4 διαστάσεων $5 \times 2n_s$ καθώς επίσης και μπρος πίσω αντικαταστάσεις με τους παραγοντοποιημένους ήδη πίνακες ζώνης A_1 και A_2.

Απεικόνιση του αλγορίθμου σε ένα παράλληλο σύστημα κοινής μνήμης σταθερής διάστασης

Ο αριθμός των επεξεργαστών P κάθε παράλληλου υπολογιστικού συστήματος είναι πεπερασμένος, οπότε σε κάθε υλοποίηση συνήθως θα ισχύει $n_s >> P$. Έτσι το υπολογιστικό φορτίο θα πρέπει να ανακατανεμηθεί σε $\mathcal{P}_j$, $j = 1 \ldots P$, επεξεργαστές, δηλαδή στον παραπάνω παράλληλο αλγόριθμο θα πρέπει να γίνει μια ομαδοποίηση κάποιου αριθμού εικονικών επεξεργαστών σε καθένα πραγματικό $\mathcal{P}_j$. Άρα θα υπάρχουν δυο δυνατές περιπτώσεις :

Περίπτωση όπου $n_s = k\,P$

Αυτή θεωρείται η ιδανική περίπτωση, γιατί επιτυγχάνεται ισοκατανομή του υπολογιστικού φορτίου, επειδή δεν θα υπάρξουν αδρανείς επεξεργαστές κάποιες χρονικές στιγμές. Έτσι η επιλογή της διαμέρισης αν είναι δυνατόν θα πρέπει να είναι ακέραια πολλαπλάσια του πλήθους των επεξεργαστών του υπολογιστικού συστήματος. Γίνεται ομαδοποίηση k το πλήθος διαδοχικών εικονικών επεξεργαστών οι οποίοι αντιστοιχίζονται σε καθένα πραγματικό επεξεργαστή $\mathcal{P}_j \quad , \quad (j = 1, \ldots, P)$, όπως εμφανίζεται στο παρακάτω σχήμα.



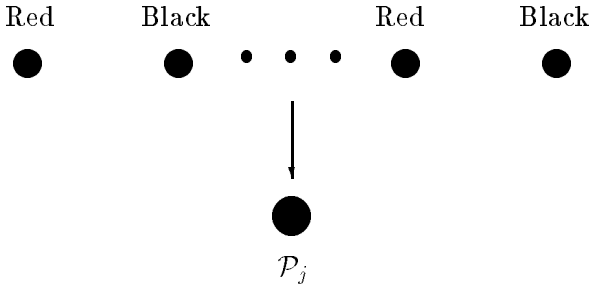
Σύμφωνα με την αντιστοίχιση του παραπάνω σχήματος σε κάθε πραγματικό επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί οι k εικονικοί επεξεργαστές $V_{(j-1)k+1}, \ldots, V_{jk}$, εκτός από τον τελευταίο $\mathcal{P}_P$, στον οποίο θα αντιστοιχιστούν $k+1$ εικονικοί επεξεργαστές, αφού το εικονικό παράλληλο σύστημα απαρτίζεται από $n_s + 1$ επεξεργαστές.

Όσο αφορά την κατανομή των δεδομένων στους επεξεργαστές υπάρχουν οι εξής παρατηρήσεις :

- Αν ο αριθμός k είναι άρτιος τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ είναι *περιττός* ενώ ο jk είναι *άρτιος*.

  Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι αντίστοιχα *red (περιττοί)* και *black (άρτιοι)* επεξεργαστές σύμφωνα με το παρακάτω σχήμα



  Άρα στον τυχαίο επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί k *red* διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k, \ldots, jk-1$$

  και k *black* διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk \, .$$

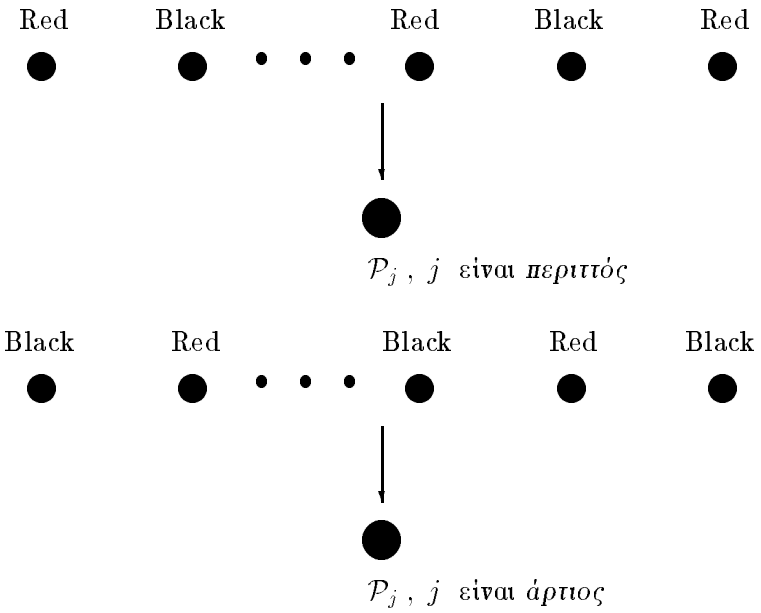
- Αν ο αριθμός k είναι περιττός τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ και jk είναι *περιττά*, όταν το j είναι *περιττό*

  ενώ τα

  $(j-1)k+1$ και jk είναι *άρτια*, όταν το j είναι *άρτιο*.

  Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι και οι δύο *red (περιττοί)* όταν το j είναι *περιττό*, ενώ θα είναι και οι δύο *black (άρτιοι)* όταν το j είναι *άρτιος*. Αυτό απεικονίζεται σχηματικά παρακάτω.



Άρα, όταν το j είναι *περιττό*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k+1$ red διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k, \ldots, jk$$

και τα $k-1$ $black$ διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk-1 \ ,$$

ενώ, όταν το j είναι *άρτιο*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k-1$ red διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k+1, \ldots, jk-1$$

και τα $k+1$ $black$ διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k, \ldots, jk \ .$$

Θα πρέπει να τονίσουμε ότι το πρώτο red διάνυσμα $\boldsymbol{x}_0$ θα έχει υπολογιστεί από τον πρώτο επεξεργαστή $\mathcal{P}_1$, ενώ το τελευταίο $\boldsymbol{x}_{4p}$ αντίστοιχα από τον $\mathcal{P}_P$. Λαμβάνοντας υπόψη την παραπάνω ανάλυση, καταλήγουμε στις δυο παρακάτω παρατηρήσεις.

- όταν ο δείκτης k είναι *άρτιος* η απεικόνιση των k εικονικών επεξεργαστών θα γίνεται με την διάταξη $RB\,RB\cdots RB$, όπου R συμβολίζει "Red" και αντίστοιχα το B "Black" εικονικό επεξεργαστή.

- όταν ο δείκτης k είναι *περιττός* η απεικόνιση θα ακολουθεί τη διάταξη $RB\,RB\cdots RB\;R$, όταν ο δείκτης j είναι *περιττός* και $BR\,BR\cdots BR\,B$, όταν είναι *άρτιος*.

Σύμφωνα με την παραπάνω ανάλυση ο παράλληλος αλγόριθμος της SOR για παράλληλα συστήματα κοινής μνήμης και σταθερής διάστασης δεν αλλάζει τη γενική του μορφή, αλλά υπάρχει κάποια διαφοροποίηση στις επιμέρους διαδικασίες σύμφωνα με την κατανομή του υπολογιστικού φορτίου οι οποίες περιγράφονται στον παρακάτω αλγόριθμο :

Red Cycle για άρτιο k

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k$ *to* $jk-1$ *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
  enddo
enddo

Black Cycle για άρτιο k

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k+1$ *to* jk *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
  enddo
enddo

Red Cycle για περιττό k

for $j = 1$ *to* P *do in parallel*
  if j *odd* *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  else
    for $l = (j-1)k+1$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  endif
enddo

Black Cycle για περιττό k

for $j = 1$ *to* P *do in parallel*
  if j *odd* *then*
    for $l = (j-1)k+1$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  else
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  endif
enddo

Στον παραπάνω αλγόριθμο είναι προφανές ότι το υπολογιστικό κόστος από $O(n_s)$ διαφοροποιήθηκε σε $O(kn_s) = O(\frac{n_s^2}{P})$ κι αυτό μπορεί να δικαιολογηθεί από τις σειριακές ανακυκλώσεις της τάξεως k, οι οποίες εμφανίστηκαν κατά την μετάβαση στο σύστημα της σταθερής διάστασης.

Περίπτωση όπου $n_s = kP + v \ , \ 1 \leq v \leq P-1$

Στην περίπτωση αυτή γίνεται απεικόνιση των k διαδοχικών εικονικών επεξεργαστών σε καθένα από τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ και $k+1$ εικονικοί αντιστοιχίζονται στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$. Αυτό έχει ως αποτέλεσμα να χαθεί η ισοκατανομή του υπολογιστικού φορτίου στους επεξεργαστές. Παρότι ο παράλληλος αλγόριθμος γίνεται περισσότερο πολύπλοκος, είναι δυνατόν να ελαχιστοποιηθεί ο χρόνος κατά τον οποίο κάποιοι επεξεργαστές θα μείνουν υποχρεωτικά αδρανείς.

Πιο αναλυτικά για τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ ισχύουν τα ίδια με την περίπτωση όπου $n_s = kP$. Για στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ απεικονίζουμε τους $V_{(j-1)k+j-P+v+1}, \ldots, V_{jk+j-P+v+1}$ εικονικούς επεξεργαστές. Έτσι έχουμε να παρατηρήσουμε τα παρακάτω

- Όταν ο δείκτης k είναι *άρτιος*, τότε και οι δυο δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα είναι

  περιττοί όταν $j-P+v$ είναι *άρτιος*

  ενώ είναι

  άρτιοι όταν $j-P+v$ είναι *περιττός* .

  Έτσι, όταν το $j-P+v$ είναι *άρτιος* , τότε και οι δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *red (περιττοί)* κι έτσι στον επεξεργαστή $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ θα αντιστοιχούν $k+2$ *red* διανύσματα

  $$\boldsymbol{x}_l \ , \ l = (j-1)k+j-P+v, \ldots, jk+j-P+v+1$$

  και k *black* διανύσματα

  $$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+j-P+v+1, \ldots, jk+j-P+v.$$

Ομοίως, όταν το $j-P+v$ είναι *περιττός*, τότε και δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *black (άρτιοι)* γι'αυτό στον επεξεργαστή $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ θα αντιστοιχούν k *red* διανύσματα

$$\boldsymbol{x}_l\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v$$

και $k+2$ *black* διανύσματα

$$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v+1.$$

- Όταν το k είναι *περιττός* για τους δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα ισχύει

  $(j-1)k+j-P+v+1$ θα είναι *άρτιος* ενώ $jk+j-P+v+1$ είναι *περιττός*.

  Άρα οι εικονικοί επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι αντίστοιχα *black (άρτιος)* και *red (περιττός)*. Έτσι σε καθένα επεξεργαστή $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ αντιστοιχούμε $k+1$ *red* διανύσματα

  $$\boldsymbol{x}_l\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v+1$$

  και $k+1$ *black* διανύσματα

  $$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v.$$

Από την παραπάνω ανάλυση προκύπτει ότι για k *άρτιο* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j$ $(j=1,\ldots,P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\cdots RB$, ενώ οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\cdots RB\ R$, όταν το j είναι *περιττό* και την $BR\,BR\cdots BR\,B$,όταν το j είναι *άρτιο*. Επίσης αφού το n_s και το k είναι *άρτιοι* θα είναι και το v *άρτιος*, δηλαδή το $P-v$ *περιττός (αντ. άρτιος)*, όταν το P είναι *περιττός (αντ. άρτιος)*.

Στη συνέχεια εμφανίζεται ο αλγόριθμος της SOR για *άρτιο* k

Red Cycle

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  else
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  endif
enddo

Black Cycle

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k + 1$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  else
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  endif
enddo

Για την περίπτωση όπου το k είναι *περιττό* οι πρώτοι $P - v - 1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P - v - 1)$ ακολουθούν την αντιστοίχιση $RB\,RB \cdots RB\,R$ για j περιττό, ενώ για j άρτιο την $B\,RB \cdots RB$. Οπότε οι τελευταίοι $v + 1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P - v, \ldots, P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB \cdots RB \ \ RB$. Επίσης, αφού το n_s είναι *άρτιος* και το k είναι *περιττό*, θα είναι το v *περιττός* για P *περιττός* και το v *άρτιος* για P *άρτιο*. Οπότε το $P - v - 1$ είναι *περιττός* για κάθε περίπτωση.

Στη συνέχεια εμφανίζεται ο αλγόριθμος της SOR για *περιττό* k

Red Cycle

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  elseif j odd *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  else
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  endif
enddo

Black Cycle

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  elseif j odd *then*
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  else
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  endif
enddo

Με μια πιο προσεκτική ματιά στους παραπάνω αλγορίθμους προκύπτει ότι κάθε ομάδα αγνώστων η οποία έχει απεικονισθεί στον ίδιο επεξεργαστή εμπλέκεται σε κοινούς υπολογισμούς. Για παράδειγμα, σύμφωνα με τον αλγόριθμο της SOR και στα βήματα S2 και S3, γίνεται η πράξη $A_4 \boldsymbol{x}_{2(j+p)+2}^{(m)}$ για τον υπολογισμό των αγνώστων $\boldsymbol{x}_{2j+1}^{(m+1)}$ και $\boldsymbol{x}_{2j}^{(m+1)}$. Επειδή οι συγκεκριμένοι άγνωστοι υπολογίζονται στον ίδιο επεξεργαστή, σύμφωνα με την απεικόνηση τους, η πράξη αυτή θα γίνει μόνο μια φορά. Επιτυγχάνεται με τον τρόπο αυτό ελαχιστοποίηση των υπολογισμών.

Ακόμα παρατηρούμε ότι το υπολογιστικό παρέμεινε σταθερό με την περίπτωση $n_s = kP$, αν και ο αλγόριθμος στην περίπτωση $n_s = kP + v \ \ , \ \ 1 \leq v \leq P - 1$ είναι σημαντικά πιο πολύπλοκος και η υλοποίησή του σημαντικά δυσκολότερη. Έτσι είναι προτιμότερο, αν φυσικά είναι εφικτή από το χρήστη η επιλογή της διαμέρισης n_s να επιλέγεται ως ακέραιο πολλαπλάσιο του αριθμού των επεξεργαστών του παράλληλου συστήματος.

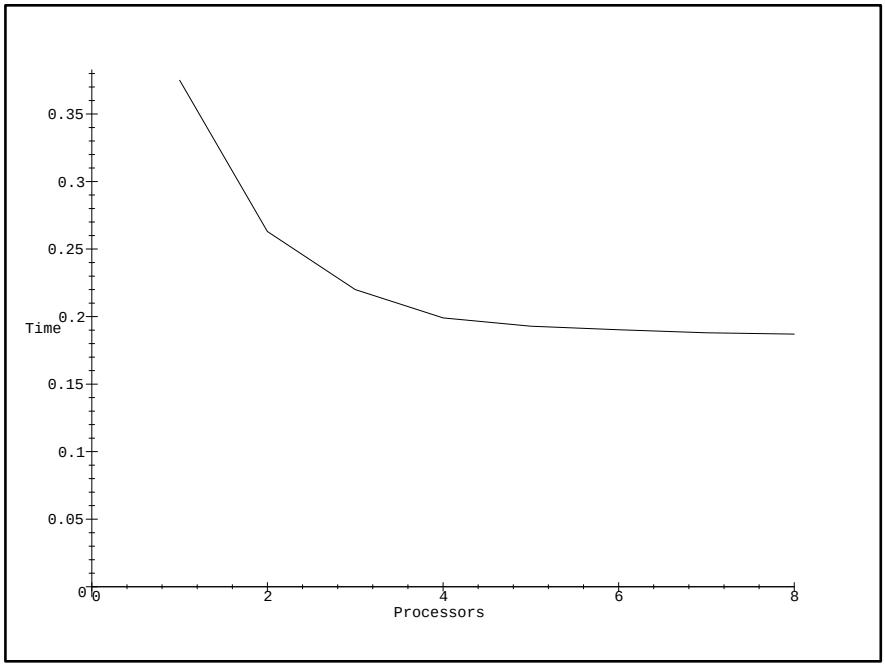
4.3.2 Υλοποίηση της επαναληπτικής μεθόδου SOR στο παράλληλο υπολογιστικό σύστημα SGI Origin 2000

Ο παραπάνω αλγόριθμος της επαναληπτικής μεθόδου SOR υλοποιήθηκε σε ένα παράλληλο υπολογιστικό σύστημα κοινής μνήμης του Πανεπιστημίου Πατρών τύπου SGI Origin 2000, αρχιτεκτονικής ccNUMA με οκτώ επεξεργαστές τύπου R10000, χρονισμένους στα 190 MHz, με 1 MB μνήμη L2 cache έκαστος. Οι επεξεργστές είναι τοποθετημένοι σε τέσσερις κόμβους, οι οποίοι συνδέονται ανά δυο με ένα router. Οι δυο routers συνδέονται μεταξύ τους με διασύνδεση craylink, η οποία προσφέρει μέγιστο εύρος ζώνης της τάξης των 1.56 GB/sec. Η συνολική μνήμη του συτήματος είναι 768 MB και το λειτουργικό του σύστημα είναι το Unix της SGI στην έκδοση IRIX 6.5. Από τους MipsPro compilers του συστήματος χρησιμοποιήθηκε αυτός της Fortran, επειδή η υλοποίηση έγινε σε Fortran με χρήση διπλής ακρίβειας στους υπολογισμούς. Χρησιμοποιήθηκαν οι μαθηματικές βιβλιοθήκες BLAS[73] και LAPACK[3], ενώ η τεχνική προγραμματισμού βασίστηκε σε OpenMP[25].

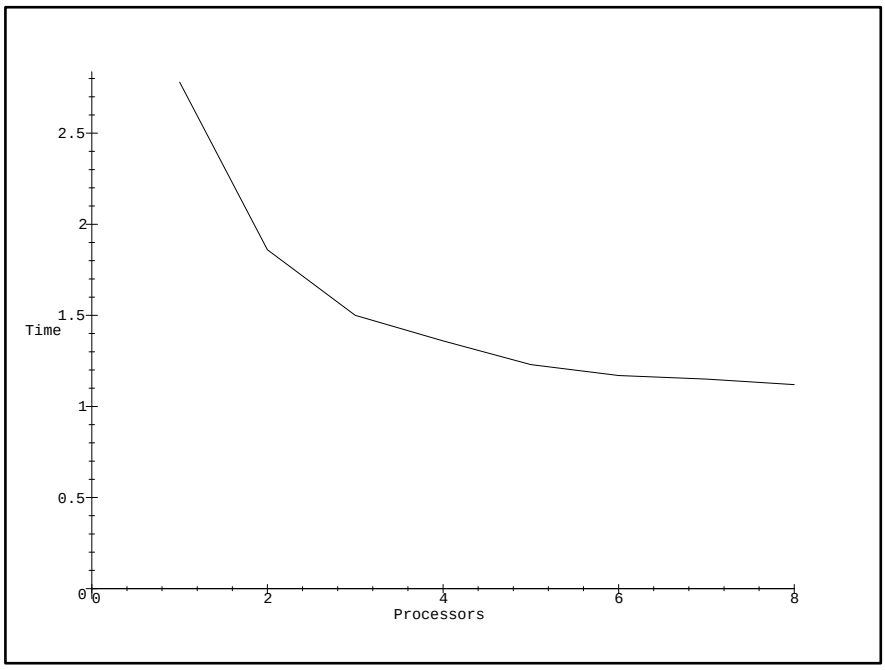
Σαν πρόβλημα μοντέλο χρησιμοποιήθηκε το ίδιο με στις προηγούμενες μετρήσεις και για τις ίδιες ακριβώς περιπτώσεις για λόγους καλλίτερης σύγκρισης των αποτελεσμάτων. Ως κριτήριο ελέγχου σύγκλισης θεωρήθηκε ξανά αυτό του σχετικού σφάλματος. Τα πειραματικά αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα T19, όπου εμφανίζεται ο συνολικός χρόνος εκτέλεσης της μεθόδου SOR μετρημένος σε δευτερόλεπτα για κάθε δυνατή επιλογή αριθμού επεξεργαστών.

T19	Red-Black SOR							
n_s	1 proc.	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	7.26e-2	6.20e-2	5.58e-2	5.46e-2	5.53e-2	5.51e-2	5.54e-2	5.53e-2
32	3.75e-1	2.63e-1	2.20e-1	1.99e-1	1.93e-1	1.90e-1	1.88e-1	1.87e-1
64	2.78e-0	1.86e-0	1.50e-0	1.36e-0	1.23e-0	1.17e-0	1.15e-0	1.12e-0
128	3.39e1	2.20e1	1.78e1	1.55e1	1.41e1	1.32e1	1.25e1	1.23e1
256	5.83e2	3.74e2	3.02e2	2.66e2	2.43e2	2.28e2	2.17e2	2.07e2

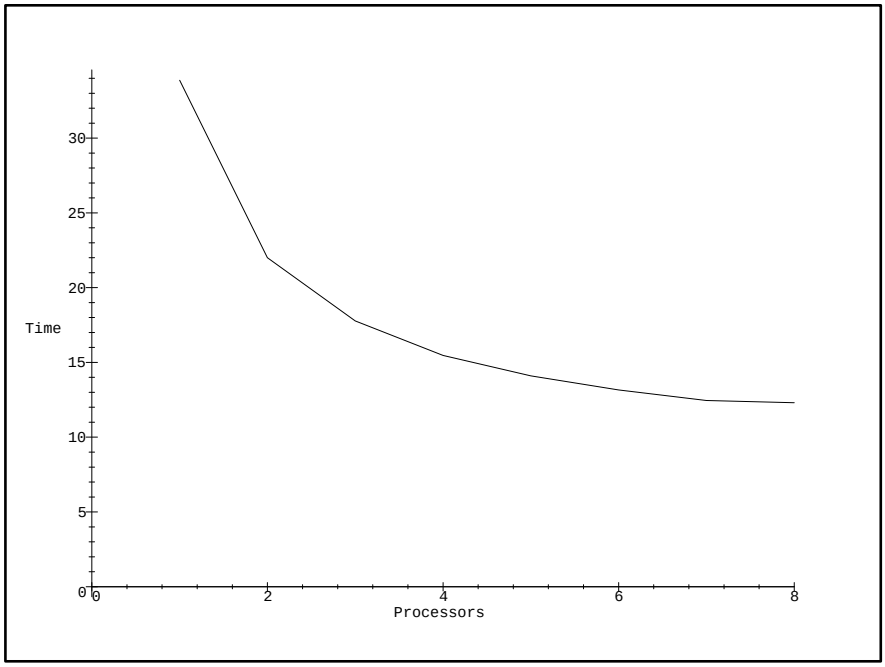
Οι επόμενες τέσσερις εικόνες εμφανίζουν τις γραφικές παραστάσεις, οι οποίες παρουσιάζουν την μείωση του χρόνου ως προς την αύξηση του αριθμού των επεξεργαστών για διακριτοποίηση 32,64,128 και 256.



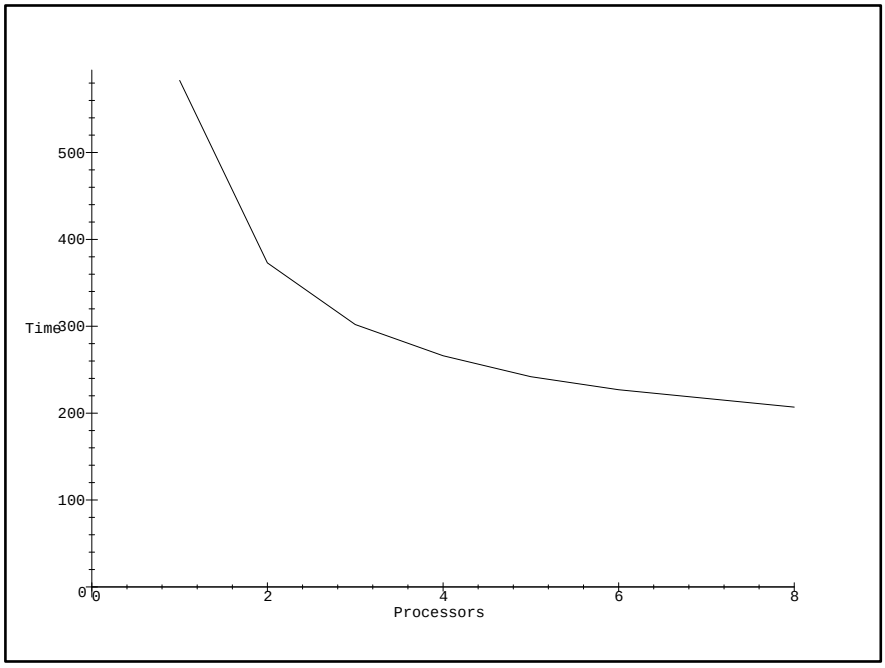
Εικόνα 53 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 32$ στην block red-black SOR .



Εικόνα 54 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 64$ στην block red-black SOR.



Εικόνα 55 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 128$ στην block red-black SOR.



Εικόνα 56 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 256$ στην block red-black SOR .

Για την μελέτη της συμπεριφοράς του παράλληλου αλγορίθμου της SOR θα πρέπει

να οριστεί η τιμη του speedup, η οποία αποτελεί μέτρο απόδοσης. Δηλαδή θεωρούμε

$$speedup_{rb} = \frac{t_{rb}}{t_n} \ ,$$

όπου t_{rb} είναι ο χρόνος εκτέλεσης σε ένα επεξεργαστή της SOR βασισμένη στην διάσπαση του collocation πίνακα σε red-black μορφή. Με t_n θεωρούμε το χρόνο εκτέλεσης του παράλληλου αλγορίθμου σε n επεξεργαστές. Ο παρακάτω πίνακας T20 εμφανίζει τις τιμές του speedup, όπως αυτές προκύπτουν για κάθε περίπτωση.

T20	$Speedup_{rb}$						
n_s	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	1.171	1.301	1.330	1.313	1.318	1.312	1.313
32	1.426	1.705	1.884	1.943	1.974	1.995	2.005
64	1.495	1.853	2.044	2.260	2.376	2.417	2.482
128	1.541	1.904	2.187	2.404	2.568	2.712	2.756
256	1.559	1.931	2.192	2.399	2.557	2.687	2.816

Από τις παραπάνω πειραματικές μετρήσεις, αλλά και από την μελέτη των γραφικών παραστάσεων προκύπτει ότι από την προσθήκη του δεύτερου επεξεργαστή αρχίζει να εμφανίζεται σημαντική μείωση του χρόνου εκτέλεσης της SOR. Με την επιπλέον προσθήκη επεξεργαστών παρατηρούμε ότι ο χρόνος συνεχίζει να μειώνεται όχι όμως με τον ίδιο ρυθμό. Αυτό οφείλεται καθαρά στην αρχιτεκτονική σύνδεσης του συγκεκριμένου υπολογιστικού συστήματος, όπου οι επεξεργαστές με την μνήμη είναι τοποθετημένοι ανά ζεύγη σε κάθε κόμβο και μεταξύ τους οι κόμβοι επικοινωνούν μέσω router. Παρολαυτά υπάρχει μια τάση για συνεχή μείωση, εκτός από την περίπτωση της διακριτοποίησης σε 16 υποδιαστήματα, όπου οι συνολικές πράξεις είναι σχετικά λίγες και γι'αυτό με την προσθήκη παραπάνω των τεσσάρων επεξεργαστών εμφανίζεται αύξηση του χρόνου εκτέλεσης.

4.3.3 Εφαρμογή της επαναληπτικής μεθόδου SOR σε παράλληλες αρχιτεκτονικές διανεμημένης μνήμης

Κατά τον σχεδιασμό ενός παράλληλου αλγορίθμου θα πρέπει να είναι διαθέσιμα η αρχιτεκτονική της διανεμημένης μνήμης - *distributed memory*[14] - παράλληλης

μηχανής, δηλαδή την τοπολογία σύνδεσης των επεξαργαστών, καθώς και ο αριθμός τους. Τα δυο αυτά στοιχεία έχουν καθοριστικό ρόλο στην διάταξη και κατάλληλη κατανομή των υπολογισμών. Εξετάζοντας τον αλγόριθμο SOR της προηγούμενης ενότητας, αλλά και λαμβάνοντας υπόψη ότι η υλοποίηση του θα πραγματοποιηθεί σε ένα παράλληλο σύστημα διανεμημένης μνήμης προκύπτουν τα παρακάτω χαρακτηριστικά ως προς τον τύπο του παράλληλου συστήματος :

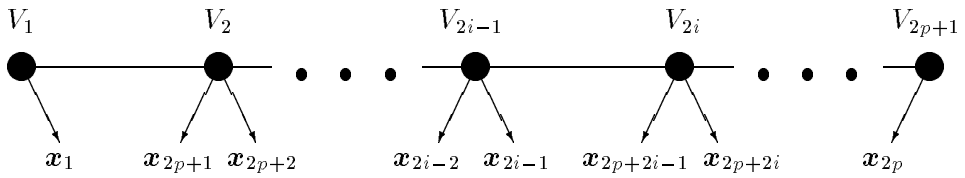
- Ο τύπος των επεξεργαστών είναι ίδιος γι όλους και καθένας τους έχει την δική του μνήμη.
- Κάθε επεξεργαστής έχει σημαντική υπολογιστική ισχύ και μνήμη.
- Όλες οι δυνατές αρχιτεκτονικές σύνδεσης ανάμεσα στους επεξεργαστές είναι εφικτές.
- Δεν θα πρέπει να υπάρχουν αδρανείς επεξεργαστές, γιατί έτσι θα μειωθεί η απόδοση του παράλληλου αλγορίθμου. Αυτό προϋποθέτει σωστή κατανομή υπολογιστικού και επικοινωνιακού κόστους.

Λαμβάνοντας υπόψη όλα τα παραπάνω θα γίνει αρχικά ο σχεδιασμός του παράλληλου αλγορίθμου σε μια εικονική παράλληλη μηχανή, η οποία θα συγκετρώνει όλα τα παραπάνω χαρακτηριστικά. Στη συνέχεια θα γίνει απεικόνιση του παραγόμενου αλγορίθμου σε μια αρχιτεκτονική με προκαθορισμένο αριθμό επεξεργαστών, όπως άλλωστε είναι στην πραγματικότητα τα παράλληλα υπολογιστικά συστήματα.

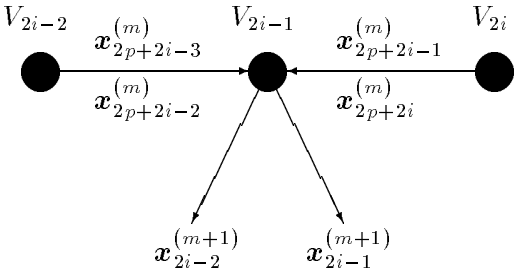
Απεικόνιση του αλγορίθμου σε ένα εικονικό παράλληλο σύστημα

Η απεικόνιση του αλγορίθμου σε ένα εικονικό παράλληλο σύστημα διαφέρει από αυτή πραγματικού μόνο ως προς το πλήθος των επεξεργαστών του. Έτσι θεωρούμε ότι έχουμε διαθέσιμους όσους επεξεργαστές απαιτείται. Αν θεωρήσουμε, ότι έχει γίνει διαμέριση του χωρίου Ω σε $n_s = 2p$ άρτιο το πλήθος υποδιαστήματα

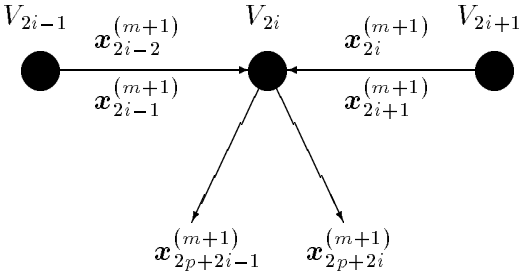
ως προς την x και την y κατεύθυνση, δηλαδή υπάρχουν $2p+1$ κάθετες και άλλες τόσες οριζόντιες γραμμές πλέγματος, μπορούμε να απεικονίσουμε τους αγνώστους που αντιστοιχούν σε κάθε κάθετη γραμμή πλέγματος σε καθένα επεξεργαστή. Θα χρειαστεί επίσης η σύνδεση των γειτονικών επεξεργαστών, αφού κάθε ομάδα εξισώσεων που αντιστοιχούν σε μια κάθετη γραμμή του πλέγματος σχετίζεται με τις ομάδες των εξισώσεων στις γειτονικές κάθετες γραμμές. Έτσι η σύνδεση των επεξεργαστών θα γίνει με την αρχιτεκτονική σε σειρά - $pipeline$. Το παρακάτω σχήμα παρουσιάζει τους $V_j\ ,\ j\ =\ 1,\ldots,2p+1$ επεξεργαστές σε σειρά και την απεικόνιση σε αυτούς των δυο τμημάτων της λύσης του γραμμικού συστήματος που αντιστοιχεί από την δεξιά και αριστερή πλευρά κάθε κάθες γραμμής πλέγματος.



Παρατηρούμε ότι στους *περιττούς* επεξεργαστές έχουν απεικονισθεί οι red γραμμές πλέγματος και στους *άρτιους* οι $black$. Επίσης, σύμφωνα με την block διαμέριση μεγέθους $2p$ όλων των διανυσμάτων της προηγούμενης ενότητας, σε κάθε περιττό επεξεργαστή $V_{2i-1}\ ,\ i=1,\ldots,p+1$, έχει απεικονισθεί ο υπολογισμός των $\boldsymbol{x}_{2i-2}$ και $\boldsymbol{x}_{2i-1}$ τμημάτων της λύσης, ενώ σε κάθε άρτιο επεξεργαστή $V_{2i}\ ,\ i=1,\ldots,p$, έχουν απεικονισθεί τα τμήματα $\boldsymbol{x}_{2p+2i-1}$ και $\boldsymbol{x}_{2p+2i}$. Η εμφανιζόμενη ανομοιομορφία στον πρώτο V_1 και τελευταίο V_{2p+1} επεξεργαστή είναι αποτέλεσμα των συνοριακών συνθηκών της διαφορικής εξίσωσης. Αυτό μπορεί να διορθωθεί αν ταυτιστούν οι δυο επεξεργαστές με συνέπεια τη δημιουργία δακτυλίου - ring - στον τρόπο σύνδεσης των επεξεργαστών.



Παρατηρούμε επίσης ότι για να υπολογίσει κάθε περιττός (red) επεξεργαστής V_{2i-1} την νέα προσέγγιση $\boldsymbol{x}^{(m+1)}_{2i-2}$ και $\boldsymbol{x}^{(m+1)}_{2i-1}$ της λύσης, σύμφωνα με τα βήματα S2 και S3 του αλγορίθμου SOR της προηγούμενης ενότητας, θα χρειαστεί να επικοινωνήσει με τους γειτονικούς $black$ V_{2i-2} και V_{2i} για να λάβει από αυτούς τα διανύσματα $\boldsymbol{x}^{(m)}_{2p+2i-3}$, $\boldsymbol{x}^{(m)}_{2p+2i-2}$ και $\boldsymbol{x}^{(m)}_{2p+2i-1}$, $\boldsymbol{x}^{(m)}_{2p+2i}$ αντίστοιχα για την κατασκευή του δεύτερου μέλους των υποσυστημάτων.



Ομοίως, οι άρτιοι $black$ επεξεργαστές V_{2i} για να υπολογίσουν τα νέα τμήματα $\boldsymbol{x}^{(m+1)}_{2p+2i-1}$ και $\boldsymbol{x}^{(m+1)}_{2p+2i}$ της προσεγγιστικής λύσης, σύμφωνα με τα βήματα S8 και S9 του αλγορίθμου της SOR, θα πρέπει να λάβουν από τους γειτονικούς red V_{2i-1} και V_{2i+1} επεξεργαστές τα ζεύγη διανυσμάτων $\boldsymbol{x}^{(m+1)}_{2i}$, $\boldsymbol{x}^{(m+1)}_{2i-1}$ και $\boldsymbol{x}^{(m+1)}_{2i}$, $\boldsymbol{x}^{(m+1)}_{2i+1}$ αντίστοιχα.

Συνδυάζοντας τα παραπάνω με την αρχιτεκτονική της εικονικής παράλληλης μηχανής το επαναληπτικό βήμα της SOR σε παράλληλη μορφή θα περιγράφεται με τον παρακάτω αλγόριθμο :

for $m = 0, \ldots, maxit$ or *until* <convergence> *do*

execute Red cycle

execute Black cycle

compute the $error\ norm$ and set <convergence>

enddo

όπου Red και Black Cycles ορίζονται να είναι οι παρακάτω υπολογιστικές διαδικασίες:

Red Cycle

Communication Phase

for $i = 2$ *to* $p+1$ *do in parallel*

V_{2i-2} *sends to* V_{2i-1} the vectors $\boldsymbol{x}^{(m)}_{2p+2i-3}$, $\boldsymbol{x}^{(m)}_{2p+2i-2}$

enddo

for $i = 1$ *to* p *do in parallel*

V_{2i} *sends to* V_{2i-1} the vectors $\boldsymbol{x}^{(m)}_{2p+2i-1}$, $\boldsymbol{x}^{(m)}_{2p+2i}$

enddo

Computation Phase

for $i = 1$ *to* $p+1$ *do in parallel*

V_{2i-1} *computes* $\boldsymbol{x}^{(m+1)}_{2i-2}$ and $\boldsymbol{x}^{(m+1)}_{2i-1}$

enddo

Black Cycle

Communication Phase

for $i = 1$ *to* p *do in parallel*

V_{2i-1} *sends to* V_{2i} the vectors $\boldsymbol{x}_{2i-2}^{(m+1)}$, $\boldsymbol{x}_{2i-1}^{(m+1)}$

enddo

for $i = 1$ *to* p *do in parallel*

V_{2i+1} *sends to* V_{2i} the vectors $\boldsymbol{x}_{2i}^{(m+1)}$, $\boldsymbol{x}_{2i+1}^{(m+1)}$

enddo

Computation Phase

for $i = 1$ *to* p *do in parallel*

V_{2i} *computes* $\boldsymbol{x}_{2p+2i-1}^{(m+1)}$ and $\boldsymbol{x}_{2p+2i}^{(m+1)}$

enddo

Από τον παραπάνω παράλληλο αλγόριθμο προκύπτει, ότι το υπολογιστικό κόστος $t_{comp}^{(m)}$ στο επαναληπτικό βήμα m της μεθόδου SOR είναι $O(n_s)$. Αυτό ισχύει, γιατί οι βασικές πράξεις γραμμικής άλγεβρας για τον υπολογισμό κάθε τμήματος της λύσης, περιλαμβάνουν κυρίως πολλαπλασιασμούς των πινάκων ζώνης A_3 και A_4 διαστάσεων $5 \times 2n_s$ καθώς επίσης και μπρος πίσω αντικαταστάσεις με τους παραγοντοποιημένους ήδη πίνακες ζώνης A_1 και A_2.

Το κόστος επικοινωνίας $t_{comm}^{(m)}$ κάθε επαναληπτικού βήματος υπολογίζεται στο χρόνο μεταφοράς από έναν επεξεργαστή στον γειτονικό του 8 διανυσμάτων (4 κατά την διάρκεια της Red Cycle διαδικασίας και 4 κατά την διάρκεια της Black Cycle) μεγέθους $4p = 2n_s$ το καθένα, ή πιο σωστά ο χρόνος μεταφοράς 4 διανυσμάτων (2 κατά την διάρκεια της Red Cycle διαδικασίας και 2 κατά την Black Cycle) μεγέθους $8p = 4n_s$ το καθένα.

Επίσης σε κάθε επαναληπτικό βήμα χρειάζεται ένας χρόνος $t_{norm}^{(m)}$ για τον υπολογισμό της νόρμας της παραγόμενης προσεγγιστικής λύσης και τον έλεγχο

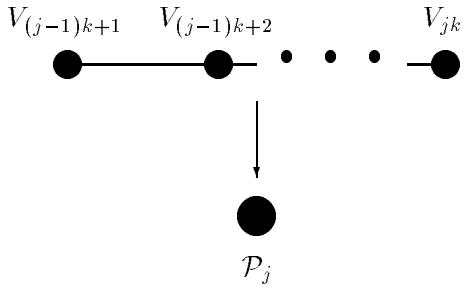
σύγκλισης. Ο χρόνος αυτός εξαρτάται από το κριτήριο τερματισμού και τον τύπο της νόρμας. Σε επόμενη ενότητα, στην οποία περιγράφεται η υλοποίηση του παράλληλου αλγορίθμου έχει χρησιμοποιηθεί ως κριτηρίο τερματισμού αυτό του σχετικού σφάλματος και θα αναλυθεί περισσότερο το τμήμα που αφορά τον έλεγχο σύγκλισης της μεθόδου.

Απεικόνιση του αλγορίθμου σε ένα παράλληλο σύστημα σταθερής διάστασης

Στην ενότητα αυτή θα γίνει απεικόνιση του παράλληλου SOR αλγορίθμου σε ένα υπολογιστικό σύστημα όμοιο με το εικονικό της προηγούμενης ενότητας, αλλά θα διαθέτει ένα προκαθορισμένο σταθερό αριθμό επεξεργαστών. Αυτό άλλωστε ισχύει και στην πραγματικότητα, δηλαδή το παράλληλο υπολογιστικό σύστημα απαρτίζεται από P επεξεργαστές $\mathcal{P}_j \ , \ (j = 1, \ldots, P)$, οι οποίοι είναι συνδεμένοι σε σειρά. Έτσι το υπολογιστικό φορτίο θα πρέπει να ανακατενεμηθεί σε αυτούς, με άλλα λόγια θα γίνει κάποια ομαδοποίηση των εικονικών επεξεργαστών και θα αντιστοιχιστεί κάθε ομάδα τους σε ένα πραγματικό επεξεργαστή. Οι δυνατές περιπτώσεις είναι δύο και αναλύονται παρακάτω.

Περίπτωση όπου $n_s = k\,P$

Η περίπτωση αυτή θεωρείται ιδανική, γιατί επιτυγχάνεται ισοκατανομή του υπολογιστικού φορτίου. Έτσι η επιλογή της διαμέρισης αν είναι δυνατόν θα πρέπει να είναι ακέραια πολλαπλάσια του πλήθους των επεξεργαστών του υπολογιστικού συστήματος. Γίνεται ομαδοποίηση k το πλήθος διαδοχικών εικονικών επεξεργαστών οι οποίοι αντιστοιχίζονται σε καθένα πραγματικό επεξεργαστή $\mathcal{P}_j \ , \ (j = 1, \ldots, P)$, όπως εμφανίζεται στο παρακάτω σχήμα.



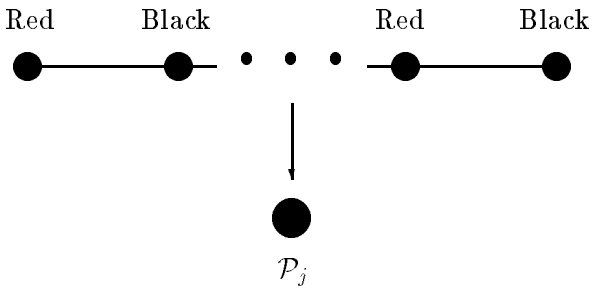
Η απεικόνιση αυτή είναι **βέλτιστη**, γιατί για κάθε άλλη δυνατή αντιστοίχιση αυξάνεται το κόστος επικοινωνίας. Θα μπορούσε δηλαδή να γίνει απευθείας απεικόνιση των αγνώστων στους επεξεργαστές, με πιθανό αποτέλεσμα τον υπολογισμό των δυο τμημάτων της προσεγγιστικής λύσης, τα οποία ανήκουν στην ίδια κάθετη γραμμή πλεγματος και έχουν αντιστοιχιστεί σε ένα εικονικό επεξεργαστή, από διαφορετικούς πραγματικούς. Το κόστος επικοινωνίας σε μια τέτοια περίπτωση θα είναι πολλαπλάσιο, γιατί απαιτείται η μεταφορά περισσότερων διανυσμάτων από κάθε επεξεργαστή. Άρα η αντιστοίχιση του παραπάνω σχήματος είναι η βέλτιστη και σε κάθε πραγματικό επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί οι k εικονικοί επεξεργαστές $V_{(j-1)k+1}, \ldots, V_{jk}$, εκτός από τον τελευταίο $\mathcal{P}_P$, στον οποίο θα αντιστοιχιστούν $k+1$ εικονικοί επεξεργαστές, αφού το εικονικό παράλληλο σύστημα απαρτίζεται από $n_s + 1$ επεξεργαστές.

Όσο αφορά την κατανομή των δεδομένων στους επεξεργαστές υπάρχουν οι εξής παρατηρήσεις :

- Αν ο αριθμός k είναι άρτιος τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ είναι *περιττός* ενώ ο jk είναι *άρτιος*.

  Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι αντίστοιχα *red (περιττοί)* και *black (άρτιοι)* επεξεργαστές σύμφωνα με το παρακάτω σχήμα



Άρα στον τυχαίο επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί k *red* διανύσματα

$$\boldsymbol{x}_l\ ,\ l=(j-1)k,\ldots,jk-1$$

και k *black* διανύσματα

$$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+1,\ldots,jk\,.$$

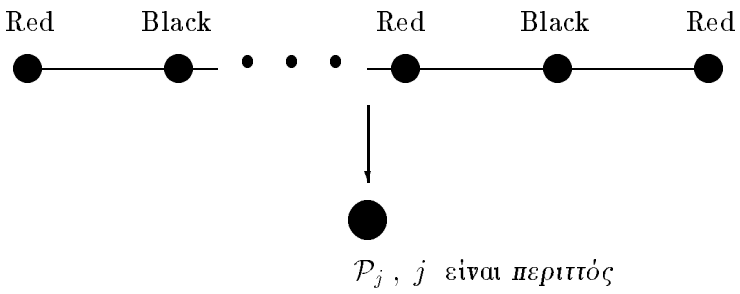
- Αν ο αριθμός k είναι περιττός τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

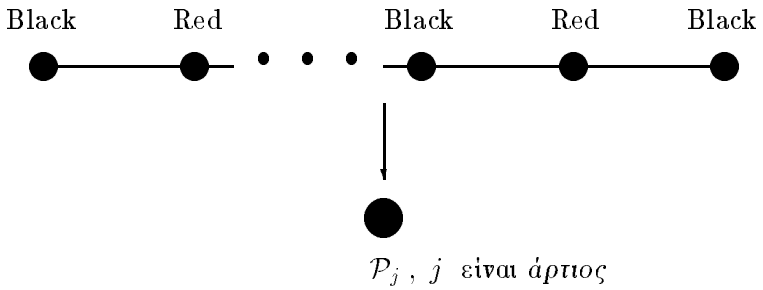
$$(j-1)k+1\ \text{ και }\ jk\ \text{ είναι }\textit{περιττά},\ \text{ όταν το }\ j\ \text{ είναι }\textit{περιττό}$$

ενώ τα

$$(j-1)k+1\ \text{ και }\ jk\ \text{ είναι }\textit{άρτια},\ \text{ όταν το }\ j\ \text{ είναι }\textit{άρτιο}.$$

Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι και οι δύο *red (περιττοί)* όταν το j είναι *περιττό*, ενώ θα είναι και οι δύο *black (άρτιοι)* όταν το j είναι *άρτιος*. Αυτό απεικονίζεται σχηματικά παρακάτω.





Άρα, όταν το j είναι *περιττό*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k+1$ red διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k, \ldots, jk$$

και τα $k-1$ $black$ διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk-1 \ ,$$

ενώ, όταν το j είναι *άρτιο*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k-1$ red διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k+1, \ldots, jk-1$$

και τα $k+1$ $black$ διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k, \ldots, jk \ .$$

Σε αυτό το σημείο θα πρέπει να τονίσουμε ότι το πρώτο red διάνυσμα $\boldsymbol{x}_0$ θα έχει υπολογιστεί από τον πρώτο επεξεργαστή $\mathcal{P}_1$, ενώ το τελευταίο $\boldsymbol{x}_{4p}$ αντίστοιχα από τον $\mathcal{P}_P$. Λαμβάνοντας υπόψη την παραπάνω ανάλυση, καταλήγουμε στις δυο παρακάτω παρατηρήσεις.

- όταν ο δείκτης k είναι *άρτιος* η απεικόνιση των k εικονικών επεξεργαστών θα γίνεται με την διάταξη $RB\,RB\cdots RB$, όπου R συμβολίζει "Red" και αντίστοιχα το B "Black" εικονικό επεξεργαστή. Έτσι κατά την διάρκεια της Red Cycle διαδικασίας ο επεξεργαστής $\mathcal{P}_{\hat{j}}$ $(\hat{j} = 1, \ldots, P-1)$, θα πρέπει να

$στείλει$ στον επεξεργαστή $\mathcal{P}_{\hat{j}+1}$ τα διανύσματα που σχετίζονται με τον τελευταίο του black εικονικό επεξεργαστή V_{jk}, ενώ κατά την διάρκεια της Black Cycle διαδικασίας ο επεξεργαστής $\mathcal{P}_{\hat{j}+1}$ θα πρέπει να $στείλει$ στον $\mathcal{P}_{\hat{j}}$ τα διανύσματα που σχετίζονται με τον πρώτο του red εικονικό επεξεργαστή V_{jk+1}

- όταν ο δείκτης k είναι $περιττός$ η απεικόνιση θα ακολουθεί τη διάταξη $RB\,RB\cdots RB\;\;R$, όταν ο δείκτης j είναι $περιττός$ και $BR\,BR\cdots BR\,B$, όταν είναι $άρτιος$. Έτσι κατά τη διάρκεια της Red (αντίστοιχα Black) Cycle διαδικασίας ο επεξεργαστής $\mathcal{P}_j$ $(j = \text{άρτιος})$ $(\text{αντ.}j = \text{περιττός})$ θα πρέπει να $στείλει$ στους επεξεργαστές $\mathcal{P}_{j-1}$ και $\mathcal{P}_{j+1}$ αντίστοιχα τα διανύσματα που σχετίζονται με το πρώτο και τελευταίο black (αντ. red) εικονικούς επεξεργαστές $V_{(j-1)k+1}$ και V_{jk}

Σύμφωνα με την παραπάνω ανάλυση ο παράλληλος αλγόριθμος της SOR για παράλληλα συστήματα σταθερής διάστασης δεν αλλάζει τη γενική του μορφή, αλλά υπάρχει κάποια διαφοροποίηση στις επιμέρους διαδικασίες όπως αυτές περιγράφονται στον παρακάτω αλγόριθμο :

Red Cycle για άρτιο k

Communication Phase

for $q = 1$ *to* s *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
$\boldsymbol{x}^{(m)}_{2p+(2q-1)k-1}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$

enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors
$\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

for $l = (j-1)k$ *to* $jk-1$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$

enddo

enddo

Black Cycle για άρτιο k

Communication Phase

for $q = 1$ *to* $\hat{s}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
$\boldsymbol{x}^{(m+1)}_{2qk}$, $\boldsymbol{x}^{(m+1)}_{2qk+1}$

enddo

for $q = 1$ *to* s *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors
$\boldsymbol{x}^{(m+1)}_{(2q-1)k}$, $\boldsymbol{x}^{(m+1)}_{(2q-1)k+1}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

for $l = (j-1)k+1$ *to* jk *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_{2p+l}$

enddo

enddo

Red Cycle για περιττό k

Communication Phase

for $q = 1$ *to* s *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors
    $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k+1}$
enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors
    $\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk}$
enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if j *odd* *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$
    enddo
  else
    for $l = (j-1)k+1$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$
    enddo
  endif
enddo

Black Cycle για περιττό k

Communication Phase

for $q = 1$ *to* s *do in parallel*
  $\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
    $\boldsymbol{x}^{(m+1)}_{(2q-1)k-1}$, $\boldsymbol{x}^{(m+1)}_{(2q-1)k}$
enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*
  $\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
    $\boldsymbol{x}^{(m+1)}_{2qk}$, $\boldsymbol{x}^{(m+1)}_{2qk+1}$
enddo

```
Computation Phase
for   j = 1  to  P  do in parallel
  if   j  odd   then
    for   l = (j − 1)k + 1  to  jk − 1  do
      P_j computes x_{2p+l}^{(m+1)}
    enddo
  else
    for   l = (j − 1)k  to  jk  do
      P_j computes x_{2p+l}^{(m+1)}
    enddo
  endif
enddo
```

όπου οι παραπάνω ακέραιοι s και $\hat{s}$ ορίζονται ως εξής

$$s = \begin{cases} \frac{P-1}{2} & , \text{ αν } P \text{ περιττός} \\ \frac{P}{2} & , \text{ αν } P \text{ άρτιος} \end{cases} \quad , \quad \hat{s} = \begin{cases} \frac{P-1}{2} & , \text{ αν } P \text{ περιττός} \\ \frac{P-2}{2} & , \text{ αν } P \text{ άρτιος} \end{cases} \quad . \qquad (131)$$

Παρατηρώντας τον παραπάνω αλγόριθμο είναι προφανές ότι το κόστος επικοινωνίας δεν μεταβλήθηκε κατά την μετάβαση στο παράλληλο σύστημα σταθερής διάστασης από το εικονικό. Αντίθετα το υπολογιστικό κόστος από $O(n_s)$ διαφοροποιήθηκε σε $O(kn_s) = O(\frac{n_s^2}{P})$ κι αυτό μπορεί να δικαιολογηθεί από τις σειριακές ανακυκλώσεις της τάξεως k, οι οποίες εμφανίστηκαν κατά την μετάβαση στο σύστημα της σταθερής διάστασης.

Περίπτωση όπου $n_s = kP + v \ , \ 1 \leq v \leq P-1$

Κατά την περίπτωση αυτή γίνεται απεικόνιση των k διαδοχικών εικονικών επεξεργαστών σε καθένα από τους πρώτους $P - v - 1$ επεξεργαστές $\mathcal{P}_j$ $(j = 1, \ldots, P - v - 1)$ και $k + 1$ εικονικοί αντιστοιχίζονται στους υπόλοιπους $v + 1$ επεξεργαστές $\mathcal{P}_j$ $(j = P - v, \ldots, P)$. Αυτό έχει ως αποτέλεσμα να χαθεί η ισοκατανομή του υπολογιστικού φορτίου στους επεξεργαστές κι έτσι αν και ο παράλληλος αλγόριθμος γίνεται περισσότερο πολύπλοκος είναι δυνατή η αποφυγή της πρόσθετης επιβάρυνσης σε

υπολογιστικό και επικοινωνιακό κόστος.

Πιο αναλυτικά για τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j=1,\ldots,P-v-1)$ ισχύουν τα ίδια με την περίπτωση όπου $n_s = kP$. Για στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j=P-v,\ldots,P)$ απεικονίζουμε τους $V_{(j-1)k+j-P+v+1},\ldots,V_{jk+j-P+v+1}$ εικονικούς επεξεργαστές. Έτσι έχουμε να παρατηρήσουμε τα παρακάτω

- Όταν ο δείκτης k είναι *άρτιος*, τότε και οι δυο δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα είναι

  περιττοί όταν $j-P+v$ είναι *άρτιος*

  ενώ είναι

  άρτιοι όταν $j-P+v$ είναι *περιττός* .

  Έτσι, όταν το $j-P+v$ είναι *άρτιος* , τότε και οι δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *red (περιττοί)* κι έτσι στον επεξεργαστή $\mathcal{P}_j \ \ (j=P-v,\ldots,P)$ θα αντιστοιχούν $k+2$ *red* διανύσματα

  $$\boldsymbol{x}_l \ , \ l=(j-1)k+j-P+v,\ldots,jk+j-P+v+1$$

  και k *black* διανύσματα

  $$\boldsymbol{x}_{2p+l} \ , \ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v.$$

  Ομοίως, όταν το $j-P+v$ είναι *περιττός*, τότε και δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *black (άρτιοι)* γι'αυτό στον επεξεργαστή $\mathcal{P}_j \ \ (j=P-v,\ldots,P)$ θα αντιστοιχούν k *red* διανύσματα

  $$\boldsymbol{x}_l \ , \ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v$$

  και $k+2$ *black* διανύσματα

  $$\boldsymbol{x}_{2p+l} \ , \ l=(j-1)k+j-P+v,\ldots,jk+j-P+v+1.$$

- Όταν το k είναι *περιττός* για τους δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα ισχύει

  $$(j-1)k+j-P+v+1 \text{ θα είναι } \textit{άρτιος} \quad \text{ενώ} \quad jk+j-P+v+1 \text{ είναι } \textit{περιττός}.$$

  Άρα οι εικονικοί επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι αντίστοιχα *black (άρτιος)* και *red (περιττός)*. Έτσι σε καθένα επεξεργαστή $\mathcal{P}_j \ (j = P-v, \ldots, P)$ αντιστοιχούμε $k+1$ *red* διανύσματα

  $$\boldsymbol{x}_l \ , \ l = (j-1)k+j-P+v+1, \ldots, jk+j-P+v+1$$

  και $k+1$ *black* διανύσματα

  $$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+j-P+v, \ldots, jk+j-P+v.$$

Από την παραπάνω ανάλυση προκύπτει ότι για k *άρτιο* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j \ (j = 1, \ldots, P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\cdots RB$, ενώ οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j \ (j = P-v, \ldots, P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\cdots RB\ R$, όταν το j είναι *περιττό* και την $BR\,BR\cdots BR\,B$,όταν το j είναι *άρτιο*. Αυτό σημαίνει ότι κατά τη διάρκεια των διαδικασιών Red και Black Cycle ο τρόπος επικοινωνίας μεταξύ των πρώτων $P-v-1$ επεξεργαστών $\mathcal{P}_j \ (j = 1, \ldots, P-v-1)$ θα είναι ίδιος με αυτόν της περίπτωσης $k = \text{άρτιος}$ για $n_s = kP$, ενώ για τους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ (j = P-v, \ldots, P)$ θα είναι ίδιος με αυτόν της περίπτωσης $k = \text{περιττός}$. Επίσης αφού το n_s και το k είναι *άρτιοι* θα είναι και το v *άρτιος*, δηλαδή το $P-v$ *περιττός (αντ. άρτιος)*, όταν το P είναι *περιττός (αντ. άρτιος)*.
Στη συνέχεια εμφανίζεται ο αλγόριθμος της SOR για *άρτιο* k

Red Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2q-1)k-1}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2\hat{q}(k+1)-k-P+v}$, $\boldsymbol{x}^{(m)}_{2p+2\hat{q}(k+1)-k-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v}$, $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+1}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2q-1)k-1}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2\hat{q}-1)(k+1)-P+v}$, $\boldsymbol{x}^{(m)}_{2p+(2\hat{q}-1)(k+1)-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+2}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P-v-1$ *then*
    for $l = (j-1)k$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  elseif $j-P+v$ even *then*
    for $l = (j-1)k+j-P+v$ *to* $jk+j-P+v+1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  else
    for $l = (j-1)k+j-P+v+1$ *to* $jk+j-P+v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_l^{(m+1)}$
    enddo
  endif
enddo

Black Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{x}_{(2q-1)k}^{(m+1)}$, $\boldsymbol{x}_{(2q-1)k+1}^{(m+1)}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}_{(2\hat{q}-1)(k+1)-P+v}^{(m+1)}$, $\boldsymbol{x}_{(2\hat{q}-1)(k+1)-P+v+1}^{(m+1)}$

enddo

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}_{2qk}^{(m+1)}$, $\boldsymbol{x}_{2qk+1}^{(m+1)}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}_{2(k+1)\hat{q}-P+v+1}^{(m+1)}$, $\boldsymbol{x}_{2(k+1)\hat{q}-P+v+2}^{(m+1)}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2q-1)k}$, $\boldsymbol{x}^{(m+1)}_{(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)k+2\hat{q}-P+v}$, $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)k+2\hat{q}-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m+1)}_{2qk}$, $\boldsymbol{x}^{(m+1)}_{2qk+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v}$, $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v+1}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

  if $1 \leq j \leq P-v-1$ *then*

    for $l = (j-1)k+1$ *to* jk *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_{2p+l}$

    enddo

  elseif $j-P+v$ even *then*

    for $l = (j-1)k+j-P+v+1$ *to* $jk+j-P+v$ *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_{2p+l}$

    enddo

  else

    for $l = (j-1)k+j-P+v$ *to* $jk+j-P+v+1$ *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_{2p+l}$

    enddo

  endif

enddo

Για την περίπτωση όπου το k είναι *περιττό* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j$ $(j = 1,\ldots,P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\,\cdots\,RB\,R$ για j περιττό, ενώ για j άρτιο την $B\,RB\,\cdots\,RB$. Οπότε οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j = P-v,\ldots,P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\,\cdots\,RB\;RB$. Αυτό σημαίνει ότι κατά τη διάρκεια των διαδικασιών Red και Black Cycle ο τρόπος επικοινωνίας μεταξύ των πρώτων $P-v-1$ επεξεργαστών $\mathcal{P}_j$ $(j = 1,\ldots,P-v-1)$ θα είναι ίδιος με αυτόν της περίπτωσης $k =$ περιττός για $n_s = kP$, ενώ για τους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j = P-v,\ldots,P)$ θα είναι ίδιος με αυτόν της περίπτωσης $k =$ άρτιος. Επίσης, αφού το n_s είναι *άρτιος* και το k είναι *περιττό*, θα είναι το v *περιττός* για P *περιττός* και το v *άρτιος* για P *άρτιο*. Οπότε το $P-v-1$ είναι *περιττός* για κάθε περίπτωση.

Στη συνέχεια εμφανίζεται ο αλγόριθμος της SOR για *περιττό* k

Red Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2\hat{q}(k+1)-k-P+v}$, $\boldsymbol{x}^{(m)}_{2p+2\hat{q}(k+1)-k-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+2}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and* *for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2q-1)k}$, $\boldsymbol{x}^{(m)}_{2p+(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+(2\hat{q}-1)(k+1)-P+v+1}$, $\boldsymbol{x}^{(m)}_{2p+(2\hat{q}-1)(k+1)-P+v+2}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and* *for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}^{(m)}_{2p+2qk-2}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+2}$, $\boldsymbol{x}^{(m)}_{2p+2(k+1)\hat{q}-P+v+3}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

if $P - v \leq j \leq P$ *then*

for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$

enddo

elseif j odd *then*

for $l = (j-1)k$ *to* jk *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$

enddo

else

for $l = (j-1)k + 1$ *to* $jk - 1$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{x}^{(m+1)}_l$

enddo

endif

enddo

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2q-1)k}$, $\boldsymbol{x}^{(m+1)}_{(2q-1)k-1}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)(k+1)-P+v}$, $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)(k+1)-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m+1)}_{2qk+2}$, $\boldsymbol{x}^{(m+1)}_{2qk+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2q-1)k}$, $\boldsymbol{x}^{(m+1)}_{(2q-1)k-1}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)k+2\hat{q}-P+v-1}$, $\boldsymbol{x}^{(m+1)}_{(2\hat{q}-1)k+2\hat{q}-P+v-2}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{x}^{(m+1)}_{2qk}$, $\boldsymbol{x}^{(m+1)}_{2qk+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v-1}$, $\boldsymbol{x}^{(m+1)}_{2(k+1)\hat{q}-P+v}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  elseif j odd *then*
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  else
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{x}_{2p+l}^{(m+1)}$
    enddo
  endif
enddo

Από την μορφή των παραπάνω αλγορίθμων προκύπτει ότι το υπολογιστικό, αλλά και το επικοινωνιακό κόστος, παρέμεινε σταθερό με την περίπτωση $n_s = kP$, αν και ο αλγόριθμος στην περίπτωση $n_s = kP + v \quad , \quad 1 \leq v \leq P - 1$ είναι σημαντικά πιο πολύπλοκος και η υλοποίησή του σημαντικά δυσκολότερη. Έτσι είναι προτιμότερο η επιλογή της διαμέρισης n_s να επιλέγεται ως ακέραιο πολλαπλάσιο του αριθμού των επεξεργαστών του παράλληλου συστήματος.

Επίσης, όπως και στην περίπτωση της αρχιτεκτονικής κοινής μνήμης, ο τρόπος απεικόνισης των αγνώστων στους επεξεργαστές συντελεί στην ελαχιστοποίηση των υπολογισμών.

4.3.4 Υλοποίηση της επαναληπτικής μεθόδου SOR στο παράλληλο υπολογιστικό σύστημα Parsytec CC2

Έγινε υλοποίηση του παραλλήλου αλγορίθμου της επαναληπτικής μεθόδου SOR σ'ένα παράλληλο υπολογιστικό σύστημα διανεμημένης μνήμης τύπου Parsytec Cognitive Computer. Το σύστημα αυτό ανήκει στην κατηγορία των MIMD παράλληλων συστημάτων και διαθέτει γενικά P επεξεργαστές του τύπου PowerPC-604 συγχρονισμένους στα 133 MHz, οι οποίοι έχουν συνδεθεί μεταξύ τους με ένα P way cross-bar HS-Link Router, με αποτέλεσμα να είναι δυνατή η υλοποίηση οποιασδήποτε αρχιτεκτονικής. Το συγκεκριμένο μηχάνημα διέθετε μόνο δυο επεξεργαστές με μνήμη 64 MB και 512 KB L2 cache ο καθένας τους. Το λειτουργικό του σύστημα ήταν Unix AIX για κάθε επεξεργαστή και πάνω σε αυτό υπήρχε εκτέλεση του παράλληλου λειτουργικού PARIX. Η υλοποίηση έγινε με μια ειδική έκδοση της Fortran για το συγκεκριμένο παράλληλο σύστημα, ενώ χρησιμοποιήθηκαν οι μαθηματικές βιβλιοθήκες BLAS[73] και LAPACK[3].

Σαν πρόβλημα μοντέλο χρησιμοποιήθηκε το ίδιο, όπως και στις προηγούμενες μετρήσεις και για τις ίδιες ακριβώς περιπτώσεις με κριτήριο τερματισμού πάλι αυτό του σχετικού σφάλματος.

Θα πρέπει να σημειωθεί, ότι η παράλληλη υλοποίηση του αλγορίθμου έγινε μέσω του προγραμματιστικού μοντέλου SPMD (Single Program Multiple Data), δηλαδή υπήρχε ένας κώδικας προγράμματος, του οποίου το κατάλληλο τμήμα που αφορά τα συγκεκριμένα δεδομένα εκτελέστηκε στον ανάλογο επεξεργαστή. Οι περιπτώσεις που υλοποιήθηκαν είναι για $n_s = kP\ ,\ k = \text{άρτιος}$, αφού η διαμέριση n_s επιλέχθηκε κάθε φορά ως δύναμη του 2.

Η σειριακή μορφή του red-black παράλληλου αλγορίθμου υλοποιήθηκε στον ένα από τους δυο επεξεργαστές. Τα πειραματικά αποτελέσματα εμφανίζονται στον παρακάτω πίνακα T21. Παρουσιάζεται ο συνολικός χρόνος εκτέλεσης κάθε επαναληπτικού βήματος της μεθόδου SOR μετρημένος σε δευτερόλεπτα. Ξεχωριστά

παρουσιάζονται οι χρόνοι της Red και Black διαδικασίας καθώς και του κριτηρίου σύγκλισης πάντα για τις ανάλογες διαμερίσεις.

T21	Red-Black SOR			
n_s	Red	Black	Norm	Total
4	3.75e-3	3.78e-3	9.00e-5	7.62e-3
8	2.86e-2	2.87e-2	2.00e-4	5.75e-2
16	1.88e-1	1.89e-1	3.90e-2	4.16e-1
32	1.07e0	1.07e0	1.30e-1	2.27e0
64	8.29e0	8.29e0	1.32e0	1.79e1

Η παρατηρούμενη μικρή διαφορά στο χρόνο εκτέλεσης ανάμεσα στις Red και Black διαδικασίες οφείλεται στο γεγονός υπολογισμού του πρώτου και του τελευταίου red διανύσματος, τα οποία έχουν μειωμένο υπολογιστικό κόστος. Στον παρακάτω πίνακα T22 παρουσιάζονται τα αποτελέσματα των μετρήσεων του χρόνου εκτέλεσης του παράλληλου αλγορίθμου της SOR στους δυο επεξεργαστές. Ο χρόνος μετρήθηκε σε δευτερόλεπτα και εμφανίζονται ξεχωριστά οι χρόνοι των υπολογισμών των red και black φάσεων, καθώς και της διαδικασίας υπολογισμού της νόρμας του κριτηρίου σύγκλισης. Επίσης παρουσιάζονται και οι αντίστοιχοι χρόνοι που αφορούν την επικοινωνία μεταξύ των επεξεργαστών σε κάθε επιμέρους διαδικασία.

T22	Red	Cycle	Black	Cycle	ϵ_{rel}	norm	Total
n_s	Comp.	Comm.	Comp.	Comm.	Comp.	Comm.	
4	2.80e-3	4.13e-3	2.60e-3	3.92e-3	5.00e-5	3.27e-3	1.68e-2
8	2.06e-2	8.45e-3	2.01e-2	7.35e-3	7.00e-4	7.48e-3	6.47e-2
16	1.07e-1	1.51e-2	1.07e-1	1.20e-2	1.83e-2	1.31e-2	2.73e-1
32	5.55e-1	2.57e-2	5.52e-1	2.39e-2	7.21e-2	1.48e-2	1.24e0
64	4.22e0	6.62e-2	4.22e0	5.63e-2	6.69e-1	3.59e-2	9.27e0

Παρατηρούμε ότι για το κριτήριο σύγκλισης ο χρόνος υπολογισμού της νόρμας έχει μεγάλο επικοινωνιακό κόστος σε σχέση με το υπολογιστικό του και συγκρινόμενο με τις αντίστοιχες μετρήσεις των red και black διαδικασιών, αν και οι δυο επεξεργαστές αποτελούν την ιδανική περίπτωση από την πλευρά του επικοινωνιακού κόστους. Αυτό μπορεί να εξηγηθεί με το γεγονός ότι κάποιος επεξεργαστής θα χρειαστεί να συγκεντρώσει τις επιμέρους τιμές της νόρμας του σφάλματος, ώστε να υπολογιστεί η

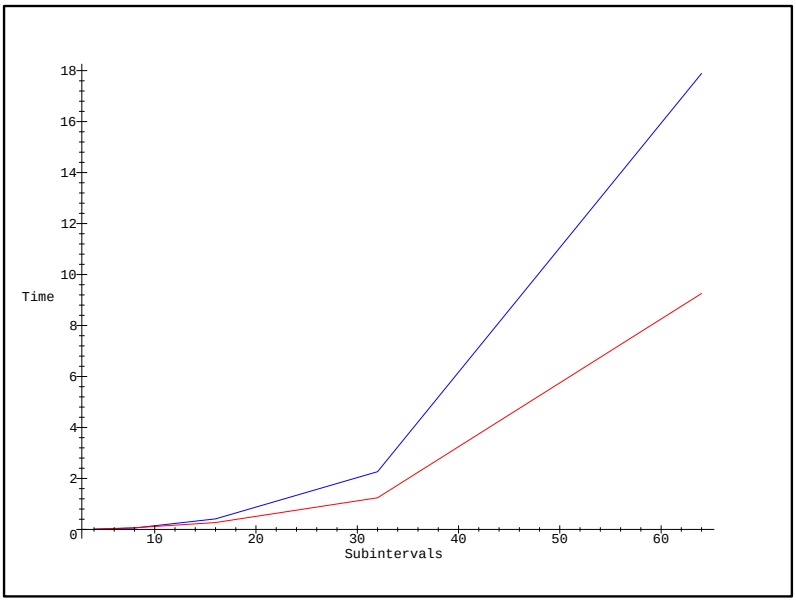
τιμή που αφορά συνολικά τη λύση $\boldsymbol{x}^{(m+1)}$. Ο χρόνος που απαιτείται για κάτι τέτοιο είναι $O(P)$ στη συνηθισμένη αρχιτεκτονική σύνδεσης των επεξεργαστών σε σειρά, ενώ για το συγκεκριμένο υπολογιστικό σύστημα CC, το οποίο διαθέτει cross-bar router, μπορεί να μειωθεί σε $O(log_2 P)$ για P πλήθος επεξεργαστών. Αυτό οφείλεται στο γεγονός, ότι μπορεί να επικοινωνήσει κάθε επεξεργαστής άμεσα με οποιοδήποτε άλλον, οπότε η μεταφορά των τιμών μπορεί να γίνει όμοια με αυτή δυαδικού δέντρου. Έτσι, η χρήση στη παράλληλη μορφή του αλγορίθμου του κριτηρίου του σχετικού σφάλματος εξελίσσεται σε σημαντικό παράγοντα καθυστέρησης, ιδιαίτερα για μεγάλο πλήθος επεξεργαστών. Το πρόβλημα αυτό μπορεί να αντιμετωπιστεί με την ιδέα της *τοπικής* σύγκλισης. Δηλαδή με την εφαρμογή του κριτηρίου σύγκλισης σε κάθε επεξεργαστή και στο τμήμα της προσεγγιστικής λύσης που το ίδιο παράγει.

Τα παραπάνω αποτελέσματα ωστόσο είναι ενδεικτικά της συμπεριφοράς του παράλληλου αλγορίθμου της προηγούμενης ενότητας, γιατί δεν υπήρχαν αρκετοί επεξεργαστές διαθέσιμοι, ώστε να φανεί η πραγματική διάσταση του επικοινωνιακού κόστους. Έτσι χρησιμοποιώντας ξανά ως ένδειξη απόδοσης του παράλληλου αλγορίθμου τις τιμές του speedup, όπως αυτές ορίστηκαν και στις μετρήσεις του αλγορίθμου στο υπολογιστικό σύστημα της κοινής μνήμης της προηγούμενης ενότητας, μπορούμε να περιμένουμε αποδόσεις πολύ κοντά στη θεωρητική βέλτιστη τιμή 2, δηλαδή τον αριθμό των επεξεργαστών. Ο παρακάτω πίνακας T23 παρουσιάζει τις τιμές του speedup για κάθε τιμή της διακριτοποίησης του προβλήματος.

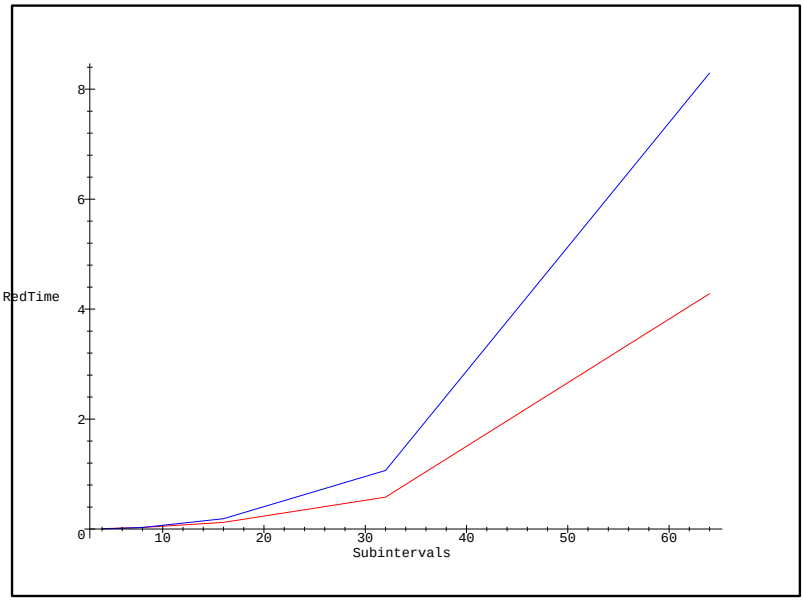
T23	Time in seconds		
n_s	Seq. Red-Black	Parallel	$speedup_{rb}$
4	7.62e-3	1.68e-2	-
8	5.75e-2	6.47e-2	-
16	4.16e-1	2.73e-1	1.524
32	2.27e0	1.24e0	1.831
64	1.79e1	9.27e0	1.931

Στην Εικόνα 57 εμφανίζεται η γραφική παράσταση του συνολικού χρόνου εκτέλεσης κάθε αλγορίθμου ως προς τη διακριτοποίηση. Είναι φανερό ότι ο χρόνος για διακριτοποιήσεις με $n_s \geq 16$ μειώνεται στο μισό κατά την παράλληλη υλοποίηση.

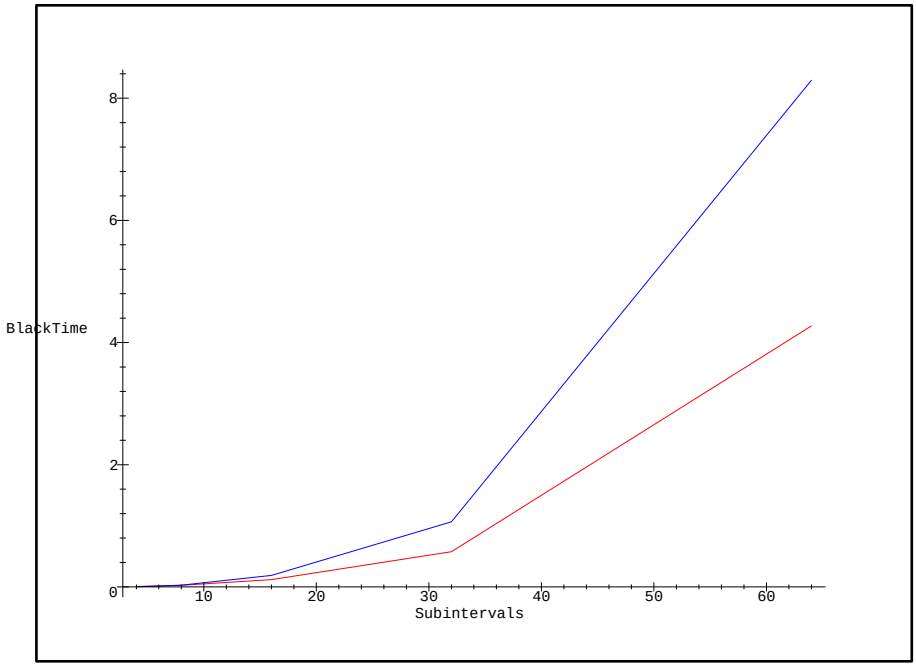
Στις τρεις επόμενες Εικόνες 58, 59 και 60 παρουσιάζονται τα γραφήματα των τριών διαδικασιών red, black και κριτηρίου σύγκλισης της red-black SOR για σειριακή και παράλληλη υλοποίηση ως προς τη διακριτοποίηση.



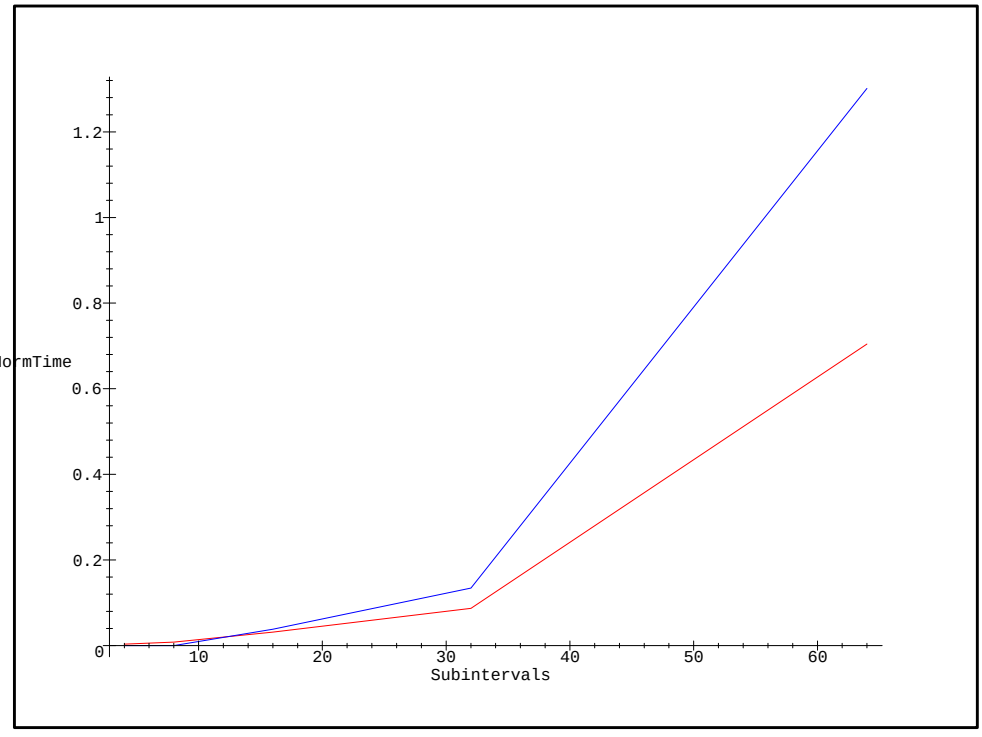
Εικόνα 57 : Ο συνολικός χρόνος εκτέλεσης ως προς τη διακριτοποίηση για τη σειριακή (μπλε) και την παράλληλη (κόκκινο) block red-black SOR.



Εικόνα 58 : Ο συνολικός χρόνος εκτέλεσης ως προς τη διακριτοποίηση της red διαδικασίας για την σειριακή (μπλε) και την παράλληλη (κόκκινο) block red-black SOR.



Εικόνα 59 : Ο συνολικός χρόνος εκτέλεσης ως προς τη διακριτοποίηση της black διαδικασίας για την σειριακή (μπλε) και την παράλληλη (κόκκινο) block red-black SOR.



Εικόνα 60 : Ο συνολικός χρόνος εκτέλεσης ως προς τη διακριτοποίηση της διαδικασίας υπολογισμού του κριτηρίου σύγκλισης για την σειριακή (μπλε) και την παράλληλη (κόκκινο) block red-black SOR.

Μετά από μελέτη των αποτελεσμάτων καταλήγουμε στα παρακάτω συμπεράσματα:

1. Ο υπολογιστικός χρόνος των διαδικασιών Red και Black Cycles του παράλληλου αλγορίθμου, όπως αναμενόταν είναι ο μισός σε σχέση με τις αντίστοιχες του σειριακού.

2. Στον χρόνο επικοινωνίας έχει συμπεριληφθεί και ο χρόνος συγχρονισμού των επεξεργαστών, αφού κατά της διαδικασίας Red Cycle ο πρώτος επεξεργαστής υπολογίζει $k-1$ red διανύσματα λύσης, ενώ ο τελευταίος $k+1$ διανύσματα.

3. Ο παράλληλος αλγόριθμος εμφανίζεται αποδοτικός για διαμερίσεις $n_s \geq 16$ λόγω του επικοινωνιακού κόστους. Έτσι η απόδοση

$$Speedup = \frac{\text{Σειριακός χρόνος}}{\text{Παράλληλος χρόνος}} \quad ,$$

αγγίζει τη βέλτιστη τιμή 2 για μεγάλες τιμές του n_s.

4.4 Εφαρμογή της επαναληπτικής μεθόδου BiCGSTAB σε παράλληλες αρχιτεκτονικές

Στο προηγούμενο κεφάλαιο παρουσιάστηκε η εφαρμογή των γνωστότερων non-stationary επαναληπτικών μεθόδων στο collocation γραμμικό σύστημα βασισμένο στη διάσπαση της block τριδιαγώνιας μορφής του. Όπως προέκυψε από την μελέτη των τιμών Ritz και επαληθεύτηκε από τις πειραματικές μετρήσεις, αποδοτικότερη μέθοδος είναι η SSOR preconditioned BiCGSTAB. Η εφαρμογή του αλγορίθμου αυτής της μεθόδου σε παράλληλες αρχιτεκτονικές δεν μπορεί να θεωρηθεί αποδοτική εξαιτείας της αυστηρά σειριακής μορφής της τεχνικής του preconditioning, η οποία συγκεντρώνει πάνω από το 75% του υπολογιστικού κόστους της μεθόδου. Δηλαδή κατά την τεχνική του preconditioning απαιτείται η επίλυση ενός άνω και ενός κάτω block τριγωνικού γραμμικού συστήματος, του οποίου η μορφή είναι τέτοια, ώστε δεν μπορεί να επιλυθεί με παράλληλες διαδικασίες. Έτσι θα πρέπει να χρησιμοποιηθεί η μορφή του collocation πίνακα σε block red-black, η οποία,

όπως προέκυψε στις προηγούμενες ενότητες του παρόντος κεφαλαίου, οδηγεί σε αποδοτικούς αλγορίθμους, τόσο σε παράλληλα περιβάλλοντα κοινής, όσο και σε διανεμημένης μνήμης.

Κατά την εφαρμογή της SSOR preconditioned επαναληπτικής μεθόδου BiCGSTAB στο red-black collocation γραμμικό σύστημα

$$\begin{bmatrix} T_R D_R & -T_R H_B \\ -T_B H_R & T_B D_B \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} T_R \boldsymbol{b}_R \\ T_B \boldsymbol{b}_B \end{bmatrix} \tag{132}$$

σύμφωνα με τον αλγόριθμο της στην Εικόνα 15, σε κάθε επαναληπτικό βήμα ο collocation πίνακας πολλαπλασιάζεται δυο φορές με κάποια διανύσματα και επίσης εφαρμόζεται άλλες δυο φορές ο preconditioner πίνακας. Η δράση του preconditioner πίνακα M, όπως είναι γνωστό επιλύει ένα γραμμικό σύστημα $M\boldsymbol{y} = \boldsymbol{u}$ και αποτελείται από τις τρεις παρακάτω διαδικασίες:

1. Επίλυση του block κάτω τριγωνικού συστήματος $M_1\, \boldsymbol{v} = \omega(2-\omega)\boldsymbol{u}$,
   όπου $M_1 = D_{RB} - \omega L_{RB}$

2. Πολλαπλασιασμός $D_{RB}\, \boldsymbol{v} = \boldsymbol{z}$

3. Επίλυση του block άνω τριγωνικού συστήματος $M_2\, \boldsymbol{y} = \boldsymbol{z}$,
   όπου $M_2 = D_{RB} - \omega U_{RB}$,

όπου οι παραπάνω πίνακες D_{RB}, L_{RB} και U_{RB} αναφέρονται στη διάσταση του collocation πίνακα και θα ισούται με

$$D_{RB} = \begin{bmatrix} T_R D_R & O \\ O & T_B D_B \end{bmatrix} , \tag{133}$$

$$L_{RB} = \begin{bmatrix} O & O \\ T_B H_B & O \end{bmatrix} \tag{134}$$

και

$$U_{RB} \;=\; \left[\begin{array}{cc} O & T_R H_B \\ O & O \end{array} \right] \quad . \tag{135}$$

Αν θεωρήσουμε την ομοιόμορφη block διαμέριση μήκους $2n_s = 4p$ των παρακάτω διανυσμάτων

$$\boldsymbol{z}_R = [\boldsymbol{z}_1^T \; \boldsymbol{z}_2^T \; \cdots \; \boldsymbol{z}_{2p}^T]^T \quad \text{και} \quad \boldsymbol{z}_B = [\boldsymbol{z}_{2p+1}^T \; \boldsymbol{z}_{2p+2}^T \; \cdots \; \boldsymbol{z}_{4p}^T]^T \tag{136}$$

$$\boldsymbol{t}_R = [\boldsymbol{t}_1^T \; \boldsymbol{t}_2^T \; \cdots \; \boldsymbol{t}_{2p}^T]^T \quad \text{και} \quad \boldsymbol{t}_B = [\boldsymbol{t}_{2p+1}^T \; \boldsymbol{t}_{2p+2}^T \; \cdots \; \boldsymbol{t}_{4p}^T]^T \quad , \tag{137}$$

μπορούμε να καταλήξουμε στον παρακάτω αλγόριθμο, ο οποίος περιγράφει τον πολλαπλασιασμό του red-black collocation πίνακα με ένα διάνυσμα $\boldsymbol{z}$, το οποίο παράγεται σε κάθε βήμα της επαναληπτικής μεθόδου BiCGSTAB και το αποτέλεσμα της πράξης είναι το διάνυσμα $\boldsymbol{t}$.

Αλγόριθμος πολλαπλασιασμού του red-black collocation πίνακα με ένα διάνυσμα $\boldsymbol{z}$

S1: $\boldsymbol{t}_1 = A_2\boldsymbol{z}_1 + A_3\boldsymbol{z}_{2p+1} - A_4\boldsymbol{z}_{2p+2}$

For $j = 2$ *to* $2p-1$ *do*

If j *even* *then*

$S2:$ $\boldsymbol{t}_j = 2A_1\boldsymbol{z}_j + A_3(\boldsymbol{z}_{2p+j-1} + \boldsymbol{z}_{2p+j+1}) + A_4(\boldsymbol{z}_{2p+j} - \boldsymbol{z}_{2p+j+2})$

else

$S3:$ $\boldsymbol{t}_j = 2A_2\boldsymbol{z}_j + A_3(\boldsymbol{z}_{2p+j} - \boldsymbol{z}_{2p+j-2}) - A_4(\boldsymbol{z}_{2p+j-1} + \boldsymbol{z}_{2p+j-1})$

endif

Enddo

S4: $\boldsymbol{t}_{2p} = -A_2\boldsymbol{z}_{2p} + A_3\boldsymbol{z}_{4p-1} + A_4\boldsymbol{z}_{4p}$

S5: $\boldsymbol{t}_{2p+1} = 2A_1\boldsymbol{z}_{2p+1} + A_3\boldsymbol{z}_2 + A_4(\boldsymbol{z}_1 - \boldsymbol{z}_3)$

S6: $\boldsymbol{t}_{2p+2} = 2A_2\boldsymbol{z}_{2p+2} + A_3\boldsymbol{z}_2 - A_4(\boldsymbol{z}_1 + \boldsymbol{z}_3)$

For $j = 3$ *to* $2p-2$ *do*

If j *odd* *then*

$S7:$ $\boldsymbol{t}_{2p+j} = 2A_1\boldsymbol{z}_{2p+j} + A_3(\boldsymbol{z}_{j-1} + \boldsymbol{z}_{j+1}) + A_4(\boldsymbol{z}_j - \boldsymbol{z}_{j+2})$

else

$S8:$ $\boldsymbol{t}_{2p+j} = 2A_2\boldsymbol{z}_{2p+j} + A_3(\boldsymbol{z}_j - \boldsymbol{z}_{j-2}) - A_4(\boldsymbol{z}_{j-1} + \boldsymbol{z}_{j-1})$

endif

Enddo

S9: $\boldsymbol{t}_{4p-1} = 2A_1\boldsymbol{z}_{4p-1} + A_3\boldsymbol{z}_{2p-2} + A_4(\boldsymbol{z}_{2p-1} - \boldsymbol{z}_{2p})$

S10: $\boldsymbol{t}_{4p} = 2A_2\boldsymbol{z}_{4p} - A_3\boldsymbol{z}_{2p-2} - A_4(\boldsymbol{z}_{2p-1} + \boldsymbol{z}_{2p})$

Στη συνέχεια, βασιζόμενοι στην ίδια διαμέριση των διανυσμάτων $\boldsymbol{t}$ και $\boldsymbol{z}$, ο αλγόριθμος επίλυσης του κάτω τριγωνικού συστήματος $M_1\boldsymbol{t} = \boldsymbol{z}$ της τεχνικής SSOR preconditioning θα έχει την παρακάτω μορφή :

Αλγόριθμος επίλυσης του block κάτω τριγωνικού συστήματος $M_1\boldsymbol{t} = \boldsymbol{z}$

S1: *Solve* $A_2\boldsymbol{t}_1 = \omega(2-\omega)\boldsymbol{z}_1$

For $j = 2$ *to* $2p-1$ *do*

If j *even* *then*

S2: *Solve* $A_1\boldsymbol{t}_j = \frac{\omega}{2}(2-\omega)\boldsymbol{z}_j$

else

S3: *Solve* $A_2\boldsymbol{t}_j = \frac{\omega}{2}(2-\omega)\boldsymbol{z}_j$

endif

Enddo

S4: *Solve* $A_2\boldsymbol{t}_{2p} = -\omega(2-\omega)\boldsymbol{z}_{2p}$

S5: *Solve* $A_1\boldsymbol{t}_{2p+1} = -\frac{\omega}{2}[A_4(\boldsymbol{t}_1 - \boldsymbol{t}_3) + A3\boldsymbol{t}_2] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{2p+1}$

S6: *Solve* $A_2\boldsymbol{t}_{2p+2} = \frac{\omega}{2}[A_4(\boldsymbol{t}_1 + \boldsymbol{t}_3) - A3\boldsymbol{t}_2] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{2p+2}$

For $j = 3$ *to* $2p-2$ *do*

If j *odd* *then*

S7: *Solve* $A_1\boldsymbol{t}_{2p+j} = -\frac{\omega}{2}[A_4(\boldsymbol{t}_j - \boldsymbol{t}_{j+2}) + A3(\boldsymbol{t}_{j-1} + \boldsymbol{t}_{j+1})] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{2p+j}$

else

S8: *Solve* $A_2\boldsymbol{t}_{2p+j} = \frac{\omega}{2}[A_4(\boldsymbol{t}_{j-1} + \boldsymbol{t}_{j+1}) + A3(\boldsymbol{t}_j - \boldsymbol{t}_{j-2})] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{2p+j}$

endif

Enddo

S9: *Solve* $A_1\boldsymbol{t}_{4p-1} = -\frac{\omega}{2}[A_4(\boldsymbol{t}_{2p-1} - \boldsymbol{t}_{2p}) + A3\boldsymbol{t}_{2p-2}] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{4p-1}$

S10: *Solve* $A_1\boldsymbol{t}_{4p} = \frac{\omega}{2}[A_4(\boldsymbol{t}_{2p-1} + \boldsymbol{t}_{2p}) + A3\boldsymbol{t}_{2p-2}] + \frac{\omega}{2}(2-\omega)\boldsymbol{z}_{4p}$

Αντίστοιχα ο αλγόριθμος του πολλαπλασιασμού του block διαγώνιου πίνακα D_{RB} με ένα διάνυσμα $\boldsymbol{z}$ στην τεχνική SSOR precondition θα έχει την πιο κάτω μορφή:

Αλγόριθμος πολλαπλασιασμού του block διαγώνιου πίνακα D_{RB} με ένα διάνυσμα $\boldsymbol{z}$

S1: $\boldsymbol{t}_1 = A_2\boldsymbol{z}_1$

For $j = 2$ *to* $2p-1$ *do*

If j *even* *then*

$S2:$ $\boldsymbol{t}_j = 2A_1\boldsymbol{z}_j$

else

$S3:$ $\boldsymbol{t}_j = 2A_2\boldsymbol{z}_j$

endif

Enddo

S4: $\boldsymbol{t}_{2p} = -A_2\boldsymbol{z}_{2p}$

S5: $\boldsymbol{t}_{2p+1} = 2A_1\boldsymbol{z}_{2p+1}$

S6: $\boldsymbol{t}_{2p+2} = 2A_2\boldsymbol{z}_{2p+2}$

For $j = 3$ *to* $2p-2$ *do*

If j *odd* *then*

$S7:$ $\boldsymbol{t}_{2p+j} = 2A_1\boldsymbol{z}_{2p+j}$

else

$S8:$ $\boldsymbol{t}_{2p+j} = 2A_2\boldsymbol{z}_{2p+j}$

endif

Enddo

S9: $\boldsymbol{t}_{4p-1} = 2A_1\boldsymbol{z}_{4p-1}$

S10: $\boldsymbol{t}_{4p} = 2A_2\boldsymbol{z}_{4p}$

Βασιζόμενοι στην ίδια διαμέριση των διανυσμάτων $\boldsymbol{t}$ και $\boldsymbol{z}$, ο αλγόριθμος επίλυσης του άνω τριγωνικού συστήματος $M_2\boldsymbol{t} = \boldsymbol{z}$ της τεχνικής SSOR precondition θα έχει την παρακάτω μορφή :

Αλγόριθμος επίλυσης του άνω block τριγωνικού συστήματος $M_2\boldsymbol{t} = \boldsymbol{z}$

S1: *Solve* $A_1\boldsymbol{t}_{2p+1} = \frac{1}{2}\boldsymbol{z}_{2p+1}$

S2: *Solve* $A_2\boldsymbol{t}_{2p+2} = \frac{1}{2}\boldsymbol{z}_{2p+2}$

For $j = 3$ *to* $2p-2$ *do*

If j *odd* *then*

S3: *Solve* $A_1\boldsymbol{t}_{2p+j} = \frac{1}{2}\boldsymbol{z}_{2p+j}$

else

S4: *Solve* $A_2\boldsymbol{t}_{2p+j} = \frac{1}{2}\boldsymbol{z}_{2p+j}$

endif

Enddo

S5: *Solve* $A_1\boldsymbol{t}_{4p-1} = \frac{1}{2}\boldsymbol{z}_{4p-1}$

S6: *Solve* $A_2\boldsymbol{t}_{4p} = \frac{1}{2}\boldsymbol{z}_{4p}$

S7: *Solve* $A_2\boldsymbol{t}_1 = \frac{\omega}{2}[A_4\boldsymbol{t}_{2p+2} - A3\boldsymbol{t}_{2p+1}] + \frac{1}{2}\boldsymbol{z}_1$

For $j = 2$ *to* $2p-1$ *do*

If j *even* *then*

S8: *Solve* $A_1\boldsymbol{t}_j = -\frac{\omega}{2}[A_4(\boldsymbol{t}_{2p+j} - \boldsymbol{t}_{2p+j+2}) + A3(\boldsymbol{t}_{2p+j-1} + \boldsymbol{t}_{2p+j+1})] + \frac{1}{2}\boldsymbol{z}_j$

else

S9: *Solve* $A_2\boldsymbol{t}_j = \frac{\omega}{2}[A_4(\boldsymbol{t}_{2p+j-1} + \boldsymbol{t}_{2p+j+1}) + A_3(\boldsymbol{t}_{2p+j-2} - \boldsymbol{t}_{2p+j})] + \frac{1}{2}\boldsymbol{z}_j$

endif

Enddo

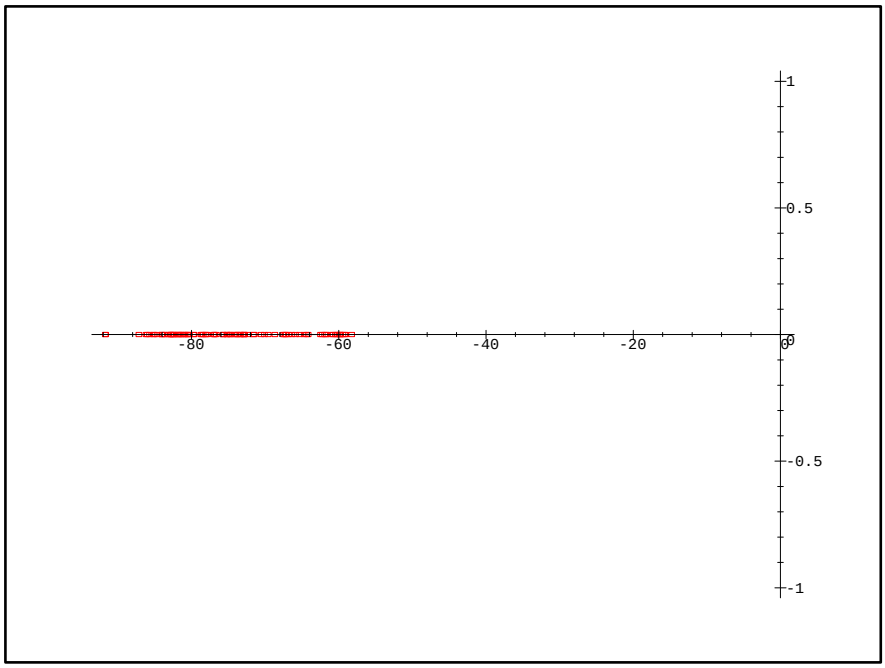
S10: *Solve* $A_2\boldsymbol{t}_{2p} = \omega(A_3\boldsymbol{t}_{4p-1} + A_4\boldsymbol{t}_{2p}) - \boldsymbol{z}_{2p}$

Είναι φανερό ότι οι βασικές πράξεις περιλαμβάνουν τον πολλαπλασιασμό κάποιου διανύσματος με έναν από τους πίνακες A_3 ή A_4 και την LU επίλυση κάποιου γραμμικού συστήματος με πίνακα συντελεστών των αγνώστων τους πίνακες A_1 ή A_2. Για την μείωση των πράξεων στο ελάχιστο θα πρέπει να ληφθεί υπόψη η δομή των πινάκων αυτών, δηλαδή ότι είναι πίνακες ζώνης με εύρος 5. Επίσης οι πίνακες ζώνης A_1 και A_2 θα παραγοντοποιηθούν μια φορά στην αρχή της διαδικασίας κι έτσι σε κάθε επαναληπτικό βήμα θα εκτελείται μόνο η προς τα εμπρός ή η προς τα πίσω αντικατάσταση.

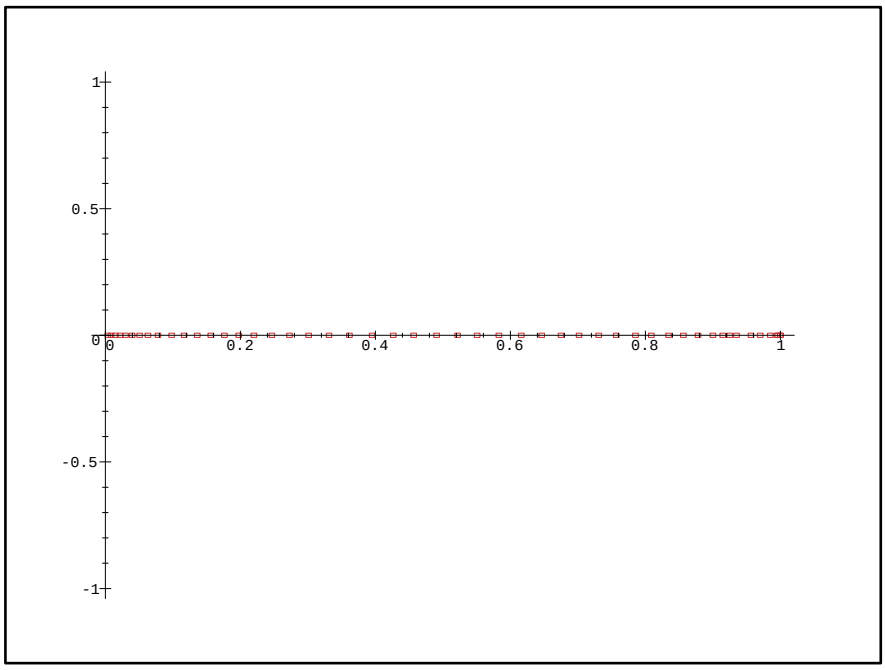
Εφαρμογή - Υλοποίηση του Red-Black αλγορίθμου της SSOR preconditioned BiCGSTAB σε σειριακή μορφή

Στην ενότητα αυτή θα μελετθεί η συμπεριφορά του αλγορίθμου της SSOR preconditioned BiCGSTAB σε σειριακή μορφή με εφαρμογή των αλγορίθμων που παρουσιάστηκαν στην προηγούμενη ενότητα. Ιδιαίτερα θα πρέπει να επαληθευτεί αν η συγκεκριμένη μέθοδος και με το συγκεκριμένο preconditioning θα εξακολουθεί να είναι η αποδοτικότερη από την ομάδα των non-stationary επαναληπτικών μεθόδων και για την block red-black μορφή του collocation πίνακα, όπως είχε συμβεί με την block τριδιαγώνια μορφή του. Αυτό σημαίνει, ότι πρέπει να υπολογιστούν οι νέες τιμές Ritz πριν και μετά την εφαρμογή της τεχνικής του precondition. Η υλοποιήση έγινε στο ίδιο υπολογιστικό σύστημα HP C3600 με αυτό που χρησιμοποιήθηκε για την μελέτη των σειριακών αλγορίθμων των non-stationary επαναληπτικών μεθόδων και της SOR, τα αποτελέσματα των οποίων έχουν ήδη παρουσιαστεί. Φυσικά χρησιμοποιήθηκε το ίδιο ακριβώς πρόβλημα για τις ίδιες ακριβώς περιπτώσεις. Η Εικόνα 61 παρουσιάζει τις τιμές Ritz χωρίς preconditioning και η Εικόνα 62 τις τιμές Ritz με SSOR preconditioning. Όπως παρατηρούμε από την πρώτη εικόνα οι τιμές Ritz δεν διαφέρουν σημαντικά από αυτές που εμφανίζονται στην Εικόνα 51 και αντιστοιχούν στην block τριδιαγώνια μορφή του collocation πίνακα. Άρα ξανά η

BiCGSTAB θεωρείται η αποδοτικότερη από τις non-staionary μεθόδους, σύμφωνα με όσο είχαν θεωρηθεί



Εικόνα 61 : Οι τιμές Ritz χωρίς preconditioning για $n_s = 64$ και τον block red-black collocation πίνακα.



Εικόνα 62 : Οι τιμές Ritz με SSOR preconditioning για $n_s = 64$ και τον red-black collocation πίνακα.

για την block τριδιαγώνια περίπτωση και επιπλέον, όπως φαίνεται από τις τιμές Ritz της Εικόνας 62 ο πίνακας της SSOR εξακολουθεί να είναι ο καλύτερος preconditioner. Τα αποτελέσματα των μετρήσεων της εφαρμογής της SSOR preconditioned στον Red-Black collocation πίνακα παρουσιάζονται στον παρακάτω πίνακα T24, ενώ στον T25 εμφανίζονται οι μετρήσεις για $\omega = 1$, δηλαδή για SGS precondition. Θα πρέπει να αναφέρουμε, ότι στις δυο υλοποιήσεις της SSOR η τιμή του ω δεν ήταν η βέλτιστη, αλλά κάποιες τυχαίες επιλογές μεγαλύτερες της μονάδας.

T24	Block Red-Black BiCGSTAB - SSOR Preconditioning					
n_s	ω	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	1.3	5	0.004	6.95e-1	4.26e-6	3.52e-7
8	1.3	9	0.01	2.03e-2	3.49e-5	1.70e-5
16	1.3	14	0.05	6.08e-4	1.01e-5	3.32e-6
32	1.4	27	0.35	3.06e-5	2.42e-5	3.15e-5
64	1.3	54	2.77	1.24e-5	4.63e-6	1.99e-5
128	1.2	113	24.5	3.37e-6	6.01e-7	7.92e-6
256	1.3	234	222	8.11e-7	4.98e-7	1.17e-5
512	1.3	467	1820	9.45e-7	3.74e-8	3.97e-6

T25	Block Red-Black BiCGSTAB - SGS Preconditioning				
n_s	m	Time	$\|u - \boldsymbol{x}^{(m)}\|$	$\|\boldsymbol{b} - A\boldsymbol{x}^{(m)}\|$	tol
4	3	0.003	6.95e-1	7.97e-3	5.12e-4
8	7	0.008	2.03e-2	3.59e-4	1.75e-4
16	14	0.047	6.08e-4	3.86e-6	2.54e-6
32	27	0.36	3.38e-5	2.79e-6	5.95e-6
64	58	2.97	1.26e-5	3.05e-6	2.43e-5
128	124	27	1.75e-6	8.58e-8	3.05e-6
256	238	225	1.01e-6	1.77e-8	2.00e-6
512	429	1663	8.65e-7	6.24e-9	3.11e-6

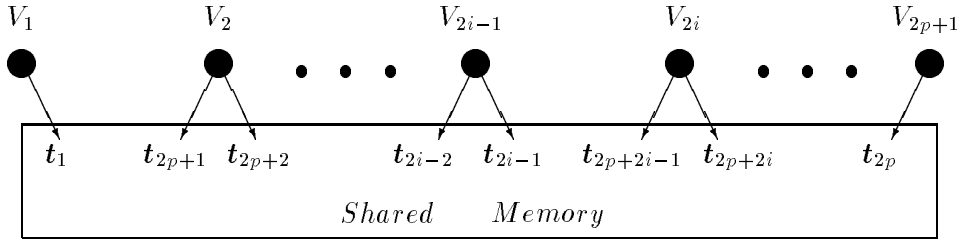
4.4.1 Εφαρμογή της επαναληπτικής μεθόδου BiCGSTAB σε παράλληλες αρχιτεκτονικές κοινής μνήμης

Σε αυτήν την ενότητα θα γίνει εφαρμογή του αλγορίθμου της SSOR preconditioned BiCGSTAB μεθόδου σε παράλληλες αρχιτεκτονικές κοινής μνήμης, δηλαδή ουσιαστικά θα εφαρμοστούν οι αλγόριθμοι του παρουσιάστηκαν στην προηγούμενη

ενότητα και αφορούν τις δυο βασικότερες διαδικασίες κάθε επαναληπτικού βήματος της BiCGSTAB. Έτσι αρχικά θα γίνει απεικόνιση τους σε ένα εικονικό σύστημα, το οποίο θα διαθέτει όσους επεξεργαστές χρειάζονται κάθε φορά και στη συνέχεια θα γίνει απεικόνιση σε ένα παράλληλο σύστημα κοινής μνήμης με σταθερή διάσταση. Θα πρέπει να τονίσουμε στο σημείο αυτό, ότι και για την παρούσα απεικόνιση των αλγορίθμων ισχύουν οι ανάλογες επισημάνσεις με την απεικόνιση του αλγορίθμου της SOR στην αντίστοιχη ενότητα και αφορούν την αρχιτεκτονική του υπολογιστικού συστήματος.

Απεικόνιση των αλγορίθμων σε ένα εικονικό παράλληλο σύστημα κοινής μνήμης

Ο πρώτος αλγόριθμος αναφέρεται στον πολλαπλασιασμό του block red-black collocation πίνακα με ένα διάνυσμα. Αν θεωρήσουμε, ότι έχουμε στη διάθεσή μας όσους επεξεργαστές χρειαζόμαστε και ότι έχει γίνει διαμέριση του χωρίου Ω σε $n_s = 2p$ το πλήθος υποδιαστήματα ως προς την x και την y κατεύθυνση, δηλαδή υπάρχουν $2p+1$ κάθετες και άλλες τόσες οριζόντιες γραμμές πλέγματος, μπορούμε να απεικονίσουμε σε καθένα επεξεργαστή τους αγνώστους του διανύσματος $\boldsymbol{t}$, σύμφωνα με τη αντιστοίχιση τους σε κάθε κάθετη γραμμή πλέγματος . Σύνδεση των επεξεργαστών μεταξύ τους δεν απαιτείται, εκτός από άμεση την σύνδεσή τους με την κοινή μνήμη. Το παρακάτω σχήμα παρουσιάζει τους V_j , $j = 1, \ldots, 2p+1$ επεξεργαστές και την απεικόνιση σε αυτούς των αγνώστων του διανύσματος $\boldsymbol{t}$.



Παρατηρούμε ότι οι περιττοί επεξεργαστές αντιστοιχούν σε red αγνώστους και οι άρτιοι σε black. Για να υπολογίσει κάθε επεξεργαστής V_i $i = 1, \ldots, 2p+1$, το

τμήμα του διανύσματος $\boldsymbol{t}_i$ σύμφωνα με τα βήματα S1 μέχρι και S10 του αλγορίθμου πολλαπλασιασμού διανύσματος με τον red-black collocation πίνακα δεν θα χρειαστεί να έχει υπολογισθεί κάποιο άλλο τμήμα του. Άρα η πράξη του πολλαπλασιασμού του block red-black collocation πίνακα είναι πλήρως παραλληλοποιήσιμη και η διαδικασίασια αυτή περιγράφεται αλγοριθμικά ως εξής:

Computation Phase of $A\boldsymbol{z} = \boldsymbol{t}$

for $i = 1$ *to* $2p + 1$ *do in parallel*

if i *odd* V_i *computes* $\boldsymbol{t}_{i-1}, \boldsymbol{t}_i$

else V_i *computes* $\boldsymbol{t}_{2p+i-1}, \boldsymbol{t}_{2p+i}$

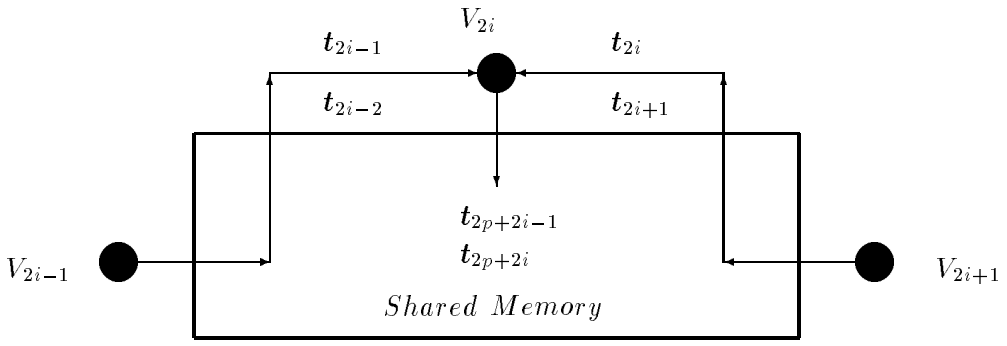
enddo

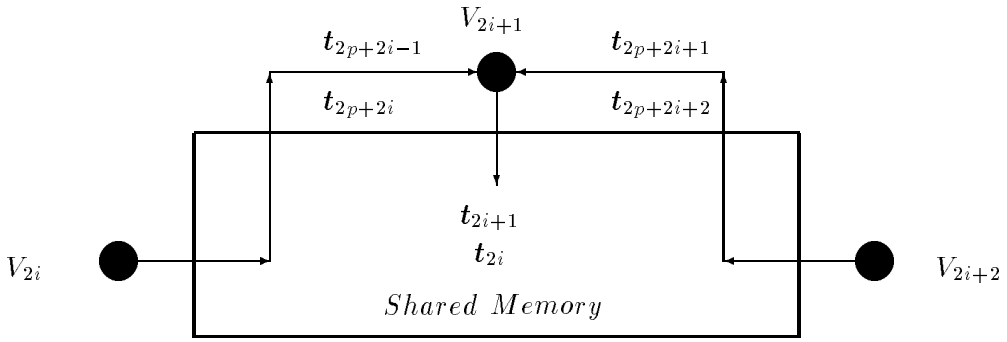
Ακριβώς την ίδια απεικόνιση αγνώστων στους επεξεργαστές ακολουθεί και η πράξη της τεχνικής του preconditioning που αναφέρεται στον πολλαπλασιασμό ενός διανύσματος με τα διαγώνια blocks του red-black collocation πίνακα.

Στη συνέχεια θα γίνει απεικόνιση των δυο αλγορίθμων που αναφέρονται στην block επίλυση των δυο τριγωνικών συστημάτων της τεχνικής του preconditioning. Χρησιμοποιώντας ξανά την ίδια block διάμεριση μεγέθους $4p$ όλων των διανυσμάτων, μπορεί να γίνει η ίδια παραπάνω αντιστοίχιση των αγνώστων στους $2p + 1$ επεξεργαστές. Έτσι ξανά οι περιττοί επεξεργαστές θα αναφέρονται στους red αγνώστους και οι άρτιοι στους black. Όμως κατά την επίλυση του κάτω (αντ. άνω) τριγωνικού συστήματος θα χρειαστεί να έχουν υπολογιστεί πρώτα κάποιοι red (αντ. black). Άρα ο τυχαίος V_{2i} , $i = 1, \ldots, p$ black επεξεργαστής στη λύση του κάτω τριγωνικού συστήματος χρειάζεται τα red διανύσματα $\boldsymbol{t}_{2i-2}$,$\boldsymbol{t}_{2i-1}$,$\boldsymbol{t}_{2i}$ και $\boldsymbol{t}_{2i+1}$, τα οποία έχουν υπολογιστεί από τους V_{2i-1} και V_{2i+1} γειτονικούς του red

επεξεργαστές. Για την περίπτωση της λύσης του άνω τριγωνικού συστήματος ο τυχαίος red επεξεργαστής V_{2i+1} , $i = 0, \ldots, p$ θα χρειαστεί τα black διανύσματα $\boldsymbol{t}_{2p+2i-1}$, $\boldsymbol{t}_{2p+2i}$,$\boldsymbol{t}_{2p+2i+1}$ και $\boldsymbol{t}_{2p+2i+2}$ τα οποία θα έχουν ήδη υπολογίσει οι γειτονικοί του V_{2i} και V_{2i+2} black επεξεργαστές για να υπολογίσει τα red διανύσματα του $\boldsymbol{t}_{2i}$ και $\boldsymbol{t}_{2i+1}$.

Τα δυο επόμενα σχήματα εμφανίζουν την περίπτωση black επεξεργαστή στη λύση του κάτω τριγωνικού συστήματος και την περίπτωση red επεξεργαστή στη λύση του άνω τριγωνικού συστήματος.





Αλγοριθμικά η διαδικασία επίλυσης του block κάτω τριγωνικού συστήματος $M_1\boldsymbol{t} = \boldsymbol{z}$ μπορεί να παρουσιαστεί ως εξής:

Red Cycle for the solution of $M_1\boldsymbol{t} = \boldsymbol{z}$

for $i = 0$ *to* p *do in parallel*

V_{2i+1} *computes* $\boldsymbol{t}_{2i}, \boldsymbol{t}_{2i+1}$

enddo

Black Cycle for the solution of $M_1\boldsymbol{t} = \boldsymbol{z}$

for $i = 1$ *to* p *do in parallel*

V_{2i} *computes* $\boldsymbol{t}_{2p+2i}, \boldsymbol{t}_{2p+2i-1}$

enddo

Ανάλογα η διαδικασία επίλυσης του άνω block τριγωνικού συστήματος $M_2\boldsymbol{t} = \boldsymbol{z}$ θα έχει τη δομή :

Black Cycle for the solution of $M_2\boldsymbol{t} = \boldsymbol{z}$

for $i = 1$ *to* p *do in parallel*

V_{2i} *computes* $\boldsymbol{t}_{2p+2i}, \boldsymbol{t}_{2p+2i-1}$

enddo

Red Cycle for the solution of $M_2\boldsymbol{t} = \boldsymbol{z}$

for $i = 0$ *to* p *do in parallel*

V_{2i+1} *computes* $\boldsymbol{t}_{2i}, \boldsymbol{t}_{2i+1}$

enddo

Από τον παραπάνω παράλληλο αλγόριθμο προκύπτει, ότι το υπολογιστικό κόστος $t_{comp}^{(m)}$ στο επαναληπτικό βήμα m της μεθόδου SSOR preconditioned BiCGSTAB είναι $O(n_s)$. Αυτό ισχύει, γιατί οι βασικές πράξεις γραμμικής άλγεβρας για τον υπολογισμό κάθε τμήματος της λύσης, περιλαμβάνουν κυρίως πολλαπλασιασμούς των πινάκων

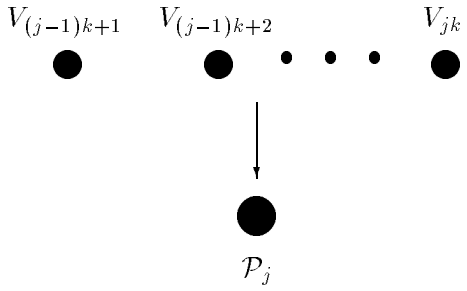
ζώνης A_1, A_2, A_3 και A_4 διαστάσεων $5 \times 2n_s$ καθώς επίσης και μπρος πίσω αντικαταστάσεις με τους παραγοντοποιημένους ήδη πίνακες ζώνης A_1 και A_2.

Απεικόνιση του αλγορίθμου σε ένα παράλληλο σύστημα κοινής μνήμης σταθερής διάστασης

Σε αυτή την ενότητα θα απεικονίσουμε τους αλγορίθμους καθεμιάς από τις προηγούμενες τέσσερις βασικές διαδικασίες της SSOR preconditioned BiCGSTAB σε ένα πεπερασμένο αριθμό επεξεργαστών P, αφού σε κάθε υλοποίηση συνήθως ισχύει $n_s >> P$. Έτσι το υπολογιστικό φορτίο θα πρέπει να ανακατανεμηθεί σε $\mathcal{P}_j$, $j = 1 \ldots P$, επεξεργαστές, δηλαδή στον παραπάνω παράλληλο αλγόριθμο θα πρέπει να γίνει μια ομαδοποίηση κάποιου αριθμού εικονικών επεξεργαστών σε καθένα πραγματικό $\mathcal{P}_j$. Θα υπάρχουν δυο δυνατές περιπτώσεις :

Περίπτωση όπου $n_s = k\,P$

Στην περίπτωση αυτή επιτυγχάνεται ισοκατανομή του υπολογιστικού φορτίου, οπότε δεν θα υπάρξουν αδρανείς επεξεργαστές κάποιες χρονικές στιγμές. Έτσι η επιλογή της διαμέρισης αν είναι δυνατόν θα πρέπει να είναι ακέραια πολλαπλάσια του πλήθους των επεξεργαστών του υπολογιστικού συστήματος. Σε αυτή τη περίπτωση γίνεται ομαδοποίηση k το πλήθος διαδοχικών εικονικών επεξεργαστών οι οποίοι αντιστοιχίζονται σε καθένα πραγματικό επεξεργαστή $\mathcal{P}_j$, $(j = 1, \ldots, P)$, όπως εμφανίζεται στο παρακάτω σχήμα.



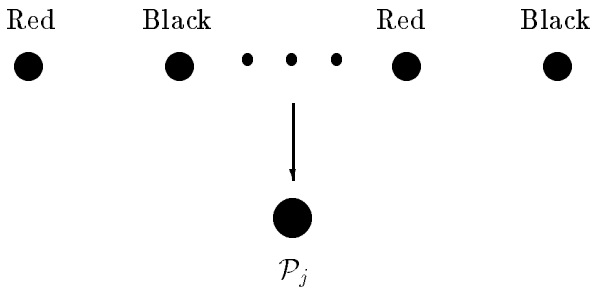
Σύμφωνα με την αντιστοίχιση του παραπάνω σχήματος σε κάθε πραγματικό

επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί οι k εικονικοί επεξεργαστές $V_{(j-1)k+1}, \ldots, V_{jk}$, εκτός του τελευταίου $\mathcal{P}_P$ που έχουν αντιστοιχηθεί $k+1$ εικονικοί επεξεργαστές.
Για την κατανομή των δεδομένων στους επεξεργαστές υπάρχουν οι εξής παρατηρήσεις:

- Αν ο αριθμός k είναι άρτιος τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ είναι *περιττός* ενώ ο jk είναι *άρτιος*.

  Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι αντίστοιχα *red (περιττοί)* και *black (άρτιοι)* επεξεργαστές σύμφωνα με το παρακάτω σχήμα



  Άρα στον τυχαίο επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί k *red* διανύσματα

  $$\boldsymbol{t}_l\ ,\ l=(j-1)k,\ldots,jk-1$$

  και k *black* διανύσματα

  $$\boldsymbol{t}_{2p+l}\ ,\ l=(j-1)k+1,\ldots,jk\,.$$

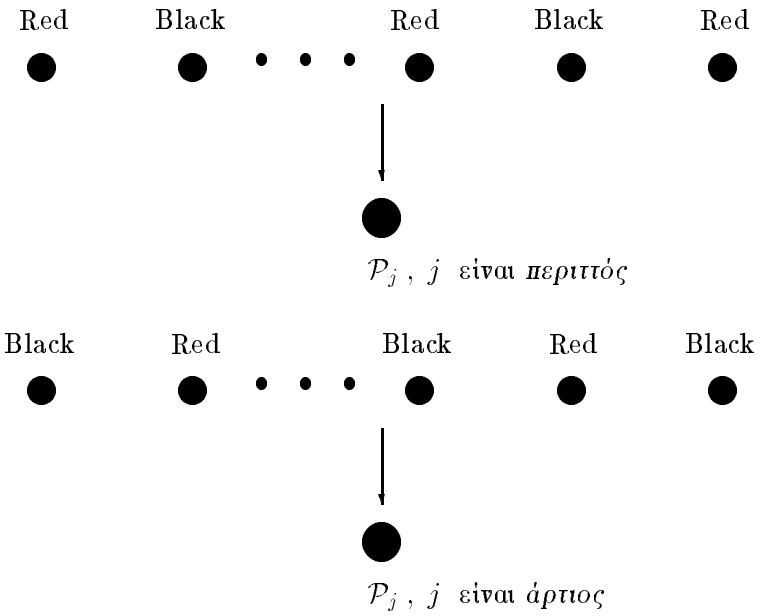
- Αν ο αριθμός k είναι περιττός τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ και jk είναι *περιττά*, όταν το j είναι *περιττό*

  ενώ τα

  $(j-1)k+1$ και jk είναι *άρτια*, όταν το j είναι *άρτιο*.

Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι και οι δύο *red (περιττοί)* όταν το j είναι *περιττό*, ενώ θα είναι και οι δύο *black (άρτιοι)* όταν το j είναι *άρτιος*. Αυτό απεικονίζεται σχηματικά παρακάτω.



Άρα, όταν το j είναι *περιττό*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k+1$ *red* διανύσματα

$$\boldsymbol{t}_l \ , \ l = (j-1)k, \ldots, jk$$

και τα $k-1$ *black* διανύσματα

$$\boldsymbol{t}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk-1 \ ,$$

ενώ, όταν το j είναι *άρτιο*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k-1$ *red* διανύσματα

$$\boldsymbol{t}_l \ , \ l = (j-1)k+1, \ldots, jk-1$$

και τα $k+1$ *black* διανύσματα

$$\boldsymbol{t}_{2p+l} \ , \ l = (j-1)k, \ldots, jk \ .$$

Θα πρέπει να τονίσουμε ότι το πρώτο red διάνυσμα $\boldsymbol{t}_0$ θα έχει υπολογιστεί από τον πρώτο επεξεργαστή $\mathcal{P}_1$, ενώ το τελευταίο $\boldsymbol{t}_{4p}$ αντίστοιχα από τον $\mathcal{P}_P$. Λαμβάνοντας υπόψη την παραπάνω ανάλυση, καταλήγουμε στις δυο παρακάτω παρατηρήσεις.

- όταν ο δείκτης k είναι *άρτιος* η απεικόνιση των k εικονικών επεξεργαστών θα γίνεται με την διάταξη $RB\,RB\cdots RB$, όπου R συμβολίζει "Red" και αντίστοιχα το B "Black" εικονικό επεξεργαστή.

- όταν ο δείκτης k είναι *περιττός* η απεικόνιση θα ακολουθεί τη διάταξη $RB\,RB\cdots RB\;R$, όταν ο δείκτης j είναι *περιττός* και $BR\,BR\cdots BR\,B$, όταν είναι *άρτιος*.

Σύμφωνα με την παραπάνω ανάλυση καθένας από τους παράλληλους αλγορίθμους για παράλληλα συστήματα κοινής μνήμης και σταθερής διάστασης δεν αλλάζει τη γενική του μορφή, αλλά υπάρχει κάποια διαφοροποίηση στις επιμέρους διαδικασίες σύμφωνα με την κατανομή του υπολογιστικού φορτίου οι οποίες περιγράφονται στον παρακάτω αλγόριθμο :

Red Cycle για άρτιο k

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k$ *to* $jk-1$ *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
  enddo
enddo

Black Cycle για άρτιο k

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k+1$ *to* jk *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
  enddo
enddo

Red Cycle για περιττό k

```
for  j = 1  to  P  do in parallel
  if  j  odd  then
    for  l = (j-1)k  to  jk  do
      P_j computes t_l
    enddo
  else
    for  l = (j-1)k+1  to  jk-1  do
      P_j computes t_l
    enddo
  endif
enddo
```

Black Cycle για περιττό k

```
for  j = 1  to  P  do in parallel
  if  j  odd  then
    for  l = (j-1)k+1  to  jk-1  do
      P_j computes t_{2p+l}
    enddo
  else
    for  l = (j-1)k  to  jk  do
      P_j computes t_{2p+l}
    enddo
  endif
enddo
```

Από τον παραπάνω αλγόριθμο είναι προφανές, ότι σε κάθε επαναληπτικό βήμα της SSOR preconditioned BiCGSTAB το υπολογιστικό κόστος από $O(n_s)$ διαφοροποιήθηκε σε $O(kn_s) = O(\frac{n_s^2}{P})$ κι αυτό μπορεί να δικαιολογηθεί από τις σειριακές ανακυκλώσεις της τάξεως k, οι οποίες εμφανίστηκαν κατά την μετάβαση στο σύστημα της σταθερής διάστασης.

Περίπτωση όπου $n_s = kP + v \ , \ 1 \leq v \leq P-1$

Στην περίπτωση αυτή γίνεται απεικόνιση των k διαδοχικών εικονικών επεξεργαστών σε καθένα από τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ και $k+1$ εικονικοί αντιστοιχίζονται στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$. Αυτό έχει ως αποτέλεσμα να χαθεί η ισοκατανομή του υπολογιστικού φορτίου στους επεξεργαστές κι έτσι αν και ο παράλληλος αλγόριθμος γίνεται περισσότερο πολύπλοκος είναι δυνατή η αποφυγή της πρόσθετης επιβάρυνσης σε υπολογιστικό κόστος, με αποτέλεσμα ο χρόνος κατά τον οποίο κάποιοι επεξεργαστές θα μείνουν υποχρεωτικά αδρανείς να ελαχιστοποιηθεί.

Πιο αναλυτικά για τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ ισχύουν τα ίδια με την περίπτωση όπου $n_s = kP$. Για στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ απεικονίζουμε τους $V_{(j-1)k+j-P+v+1}, \ldots, V_{jk+j-P+v+1}$ εικονικούς επεξεργαστές. Έτσι έχουμε να παρατηρήσουμε τα παρακάτω

- Όταν ο δείκτης k είναι *άρτιος*, τότε και οι δυο δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα είναι

  περιττοί όταν $j-P+v$ είναι *άρτιος*

  ενώ είναι

  άρτιοι όταν $j-P+v$ είναι *περιττός* .

  Έτσι, όταν το $j-P+v$ είναι *άρτιος* , τότε και οι δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *red (περιττοί)* κι έτσι στον επεξεργαστή $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ θα αντιστοιχούν $k+2$ *red* διανύσματα

  $$\boldsymbol{t}_l \ , \ l = (j-1)k+j-P+v, \ldots, jk+j-P+v+1$$

  και k *black* διανύσματα

  $$\boldsymbol{t}_{2p+l} \ , \ l = (j-1)k+j-P+v+1, \ldots, jk+j-P+v.$$

Ομοίως, όταν το $j-P+v$ είναι *περιττός*, τότε και δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *black (άρτιοι)* γι'αυτό στον επεξεργαστή $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ θα αντιστοιχούν k *red* διανύσματα

$$\boldsymbol{t}_l\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v$$

και $k+2$ *black* διανύσματα

$$\boldsymbol{t}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v+1.$$

- Όταν το k είναι *περιττός* για τους δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα ισχύει

  $(j-1)k+j-P+v+1$ θα είναι *άρτιος* ενώ $jk+j-P+v+1$ είναι *περιττός*.

  Άρα οι εικονικοί επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι αντίστοιχα *black (άρτιος)* και *red (περιττός)*. Έτσι σε καθένα επεξεργαστή $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ αντιστοιχούμε $k+1$ *red* διανύσματα

  $$\boldsymbol{t}_l\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v+1$$

  και $k+1$ *black* διανύσματα

  $$\boldsymbol{t}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v.$$

Από την παραπάνω ανάλυση προκύπτει ότι για k *άρτιο* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j$ $(j=1,\ldots,P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\cdots RB$, ενώ οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\cdots RB\ R$, όταν το j είναι *περιττό* και την $BR\,BR\cdots BR\,B$,όταν το j είναι *άρτιο*. Επίσης αφού το n_s και το k είναι *άρτιοι* θα είναι και το v *άρτιος*, δηλαδή το $P-v$ *περιττός (αντ. άρτιος)*, όταν το P είναι *περιττός (αντ. άρτιος)*.

Στη συνέχεια εμφανίζεται η γενική μορφή του αλγορίθμου κάθε παράλληλης διαδικασίας της SSOR preconditioned BiCGSTAB για *άρτιο* k

Red Cycle

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  else
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  endif
enddo

Black Cycle

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k + 1$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  else
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  endif
enddo

Για την περίπτωση όπου το k είναι *περιττό* οι πρώτοι $P - v - 1$ επεξεργαστές $\mathcal{P}_j \;\; (j = 1, \ldots, P - v - 1)$ ακολουθούν την αντιστοίχιση $RB\,RB \cdots RB\,R$ για j περιττό, ενώ για j άρτιο την $B\,RB \cdots RB$. Οπότε οι τελευταίοι $v + 1$ επεξεργαστές $\mathcal{P}_j \;\; (j = P - v, \ldots, P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB \cdots RB \;\; RB$. Επίσης, αφού το n_s είναι *άρτιος* και το k είναι *περιττό*, θα είναι το v *περιττός* για P *περιττός* και το v *άρτιος* για P *άρτιο*. Οπότε το $P - v - 1$ είναι *περιττός* για κάθε περίπτωση.

Στη συνέχεια εμφανίζεται η γενική μορφή κάθε παράλληλου αλγορίθμου της SSOR preconditioned BiCGSTAB για *περιττό* k

Red Cycle

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  elseif j odd *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  else
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  endif
enddo

Black Cycle

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  elseif j odd *then*
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  else
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  endif
enddo

Είναι εύκολο να παρατηρήσει κανείς στους παραπάνω αλγορίθμους προκύπτει ότι το υπολογιστικό παρέμεινε σταθερό με την περίπτωση $n_s = kP$, αν και ο αλγόριθμος στην περίπτωση $n_s = kP + v \quad , \quad 1 \leq v \leq P - 1$ είναι σημαντικά πιο πολύπλοκος και η υλοποίησή του σημαντικά δυσκολότερη. Έτσι είναι προτιμότερο, αν φυσικά είναι εφικτή η επιλογή της διαμέρισης n_s να επιλέγεται ως ακέραιο πολλαπλάσιο του αριθμού των επεξεργαστών του παράλληλου συστήματος.

Επίσης ο τρόπος απεικόνισης των αγνώστων στους επεξεργαστές συντελεί στην ελαχιστοποίηση των υπολογισμών. Αυτό συμβαίνει, γιατί απεικονίζοντας τους αγνώστους της δεξιάς και της αριστερής πλευράς κάθε στήλης πλέγματος, επιτυγχάνεται εκτέλεση των κοινών υπολογισμών μόνο μια φορά. Δηλαδή, για παράδειγμα στα βήματα S5 και S6 της διαδικασίας πολλαπλασιασμού του collocation πίνακα με το διάνυσμα $\boldsymbol{z}$, η κοινή πράξη $A_4 \boldsymbol{z}_1$ για τον υπολογισμό των $\boldsymbol{z}_{2p+1}$ και $\boldsymbol{z}_{2p+2}$ από τον ίδιο επεξεργαστή, θα εκτελεστεί μια φορά.

4.4.2 Υλοποίηση της επαναληπτικής μεθόδου BiCGSTAB στο παράλληλο υπολογιστικό σύστημα SGI Origin 2000

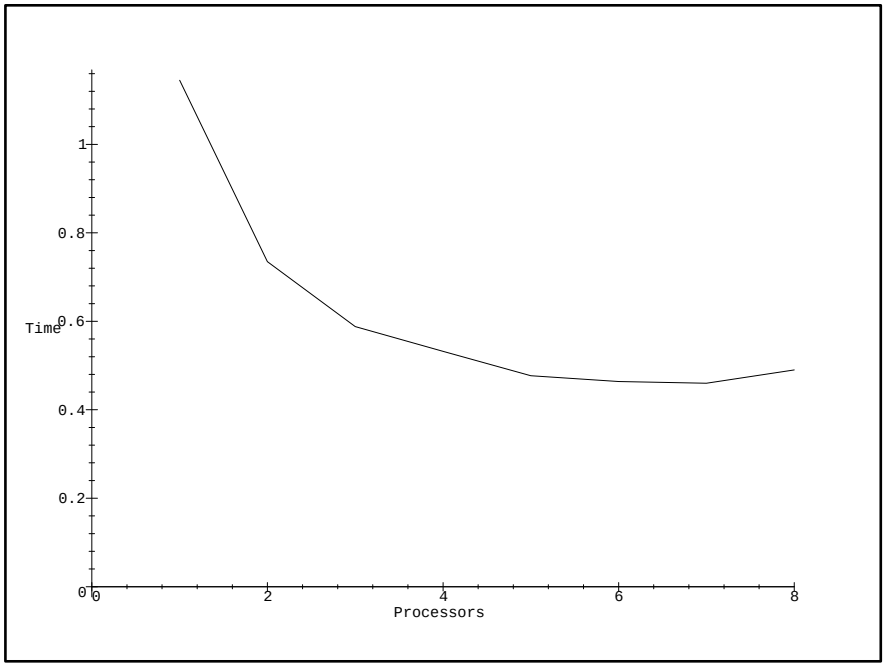
Ο παραλληλος αλγόριθμος της επαναληπτικής μεθόδου SSOR preconditioned BiCGSTAB για παράλληλα υπολογιστικά συστήματα κοινής μνήμης, ο οποίος περιγράφηκε στην προηγούμενη ενότητα, υλοποιήθηκε στο σύστημα του Πανεπιστημίου Πατρών SGI Origin 2000, όπως και ο αντίστοιχος αλγόριθμος της SOR. Χρησιμοποιήθηκαν ξανά οι μαθηματικές βιβλιοθήκες BLAS[73] και LAPACK[3], ενώ η τεχνική προγραμματισμού βασίστηκε ξανά σε OpenMP[25].

Σαν πρόβλημα μοντέλο χρησιμοποιήθηκε το ίδιο με στις προηγούμενες μετρήσεις και για τις ίδιες ακριβώς περιπτώσεις. Τα πειραματικά αποτελέσματα παρουσιάζονται στους παρακάτω πίνακες T26 για SSOR preconditioning και T27 για SGS. Εμφανίζεται ο συνολικός χρόνος εκτέλεσης της μεθόδου BiCGSTAB μετρημένος σε δευτερόλεπτα για κάθε δυνατή επιλογή αριθμού επεξεργαστών.

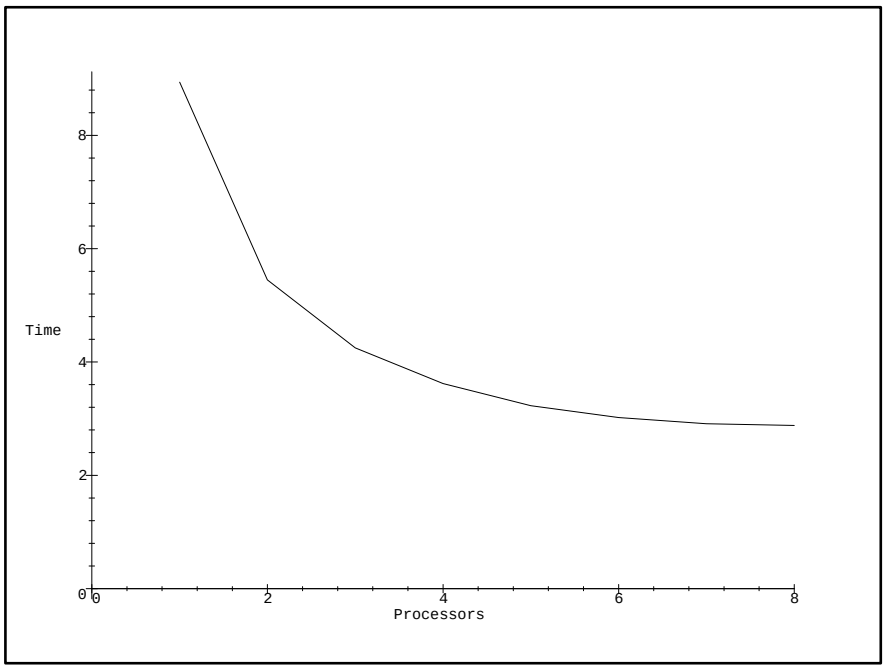
T26	Red-Black SSOR preconditioned BiCGSTAB							
n_s	1 proc.	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	1.67e-1	1.28e-1	1.13e-1	1.08e-1	1.13e-1	1.18e-1	1.23e-1	1.28e-1
32	1.15e0	7.35e-1	5.88e-1	5.32e-1	4.77e-1	4.64e-1	4.60e-1	4.90e-1
64	8.94e0	5.45e0	4.25e0	3.62e0	3.23e0	3.02e0	2.91e0	2.88e0
128	7.84e1	4.71e1	3.63e1	3.08e1	2.76e1	2.55e1	2.37e1	2.31e1
256	6.96e2	4.13e2	3.16e2	2.67e2	2.38e2	2.18e2	2.04e2	1.95e2

T27	Red-Black SGS preconditioned BiCGSTAB							
n_s	1 proc.	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	1.67e-1	1.28e-1	1.13e-1	1.08e-1	1.13e-1	1.18e-1	1.23e-1	1.28e-1
32	1.15e0	7.35e-1	5.88e-1	5.32e-1	4.77e-1	4.64e-1	4.60e-1	4.90e-1
64	9.60e0	5.84e0	4.55e0	3.88e0	3.47e0	3.24e0	3.18e0	3.03e0
128	8.58e1	5.16e1	3.98e1	3.39e1	3.03e1	2.80e1	2.60e1	2.53e1
256	7.09e2	4.20e2	3.23e2	2.72e2	2.42e2	2.22e2	2.08e2	1.99e2

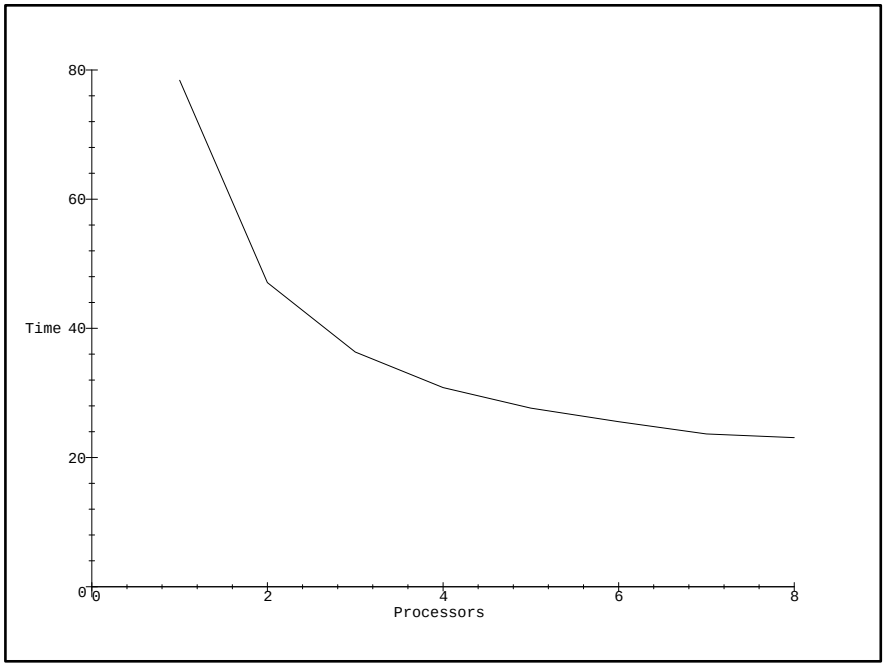
Οι επόμενες τέσσερις εικόνες εμφανίζουν τις γραφικές παραστάσεις της μείωσης του χρόνου ως προς την αύξηση του αριθμού των επεξεργαστών για διακριτοποίηση 32,64,128 και 256 για την SSOR preconditioned BiCGSTAB, ενώ οι αμέσως επόμενες τέσσερις εικόνες εμφανίζουν τις αντίστοιχες γραφικές παραστάσεις για την περίπτωση της SGS preconditioned BiCGSTAB.



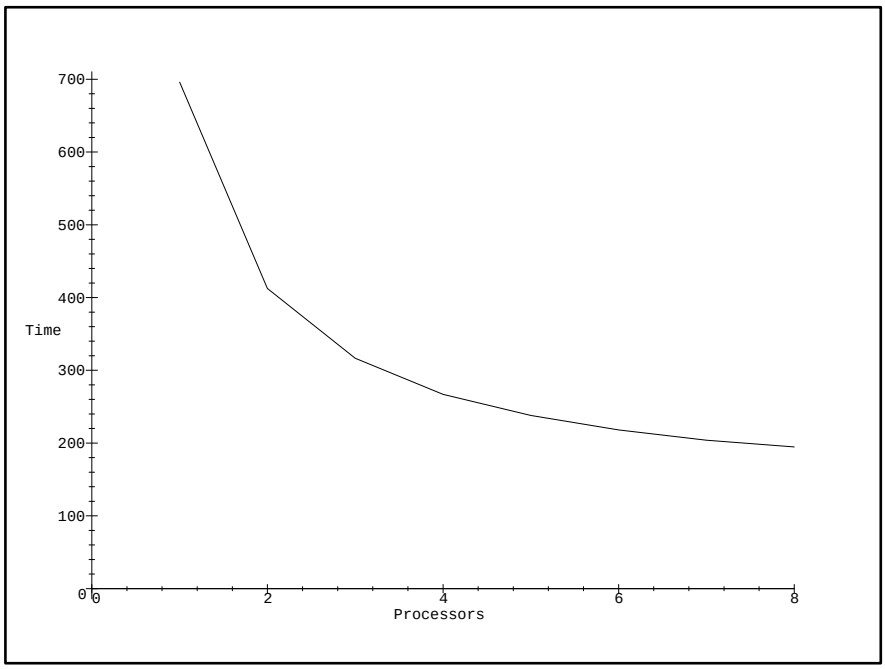
Εικόνα 63 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 32$ στην block red-black SSOR



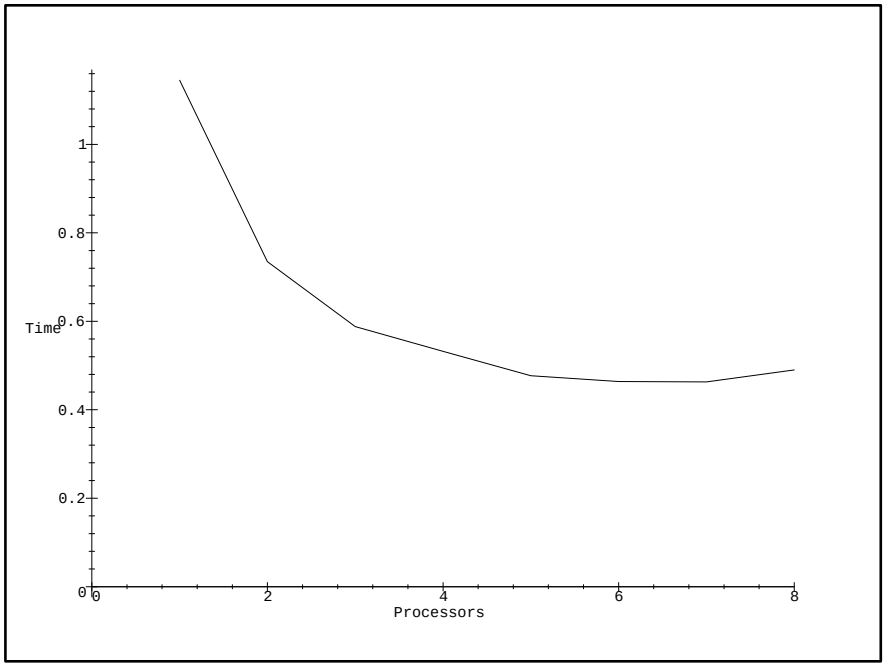
Εικόνα 64 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 64$ στην block red-black SSOR .



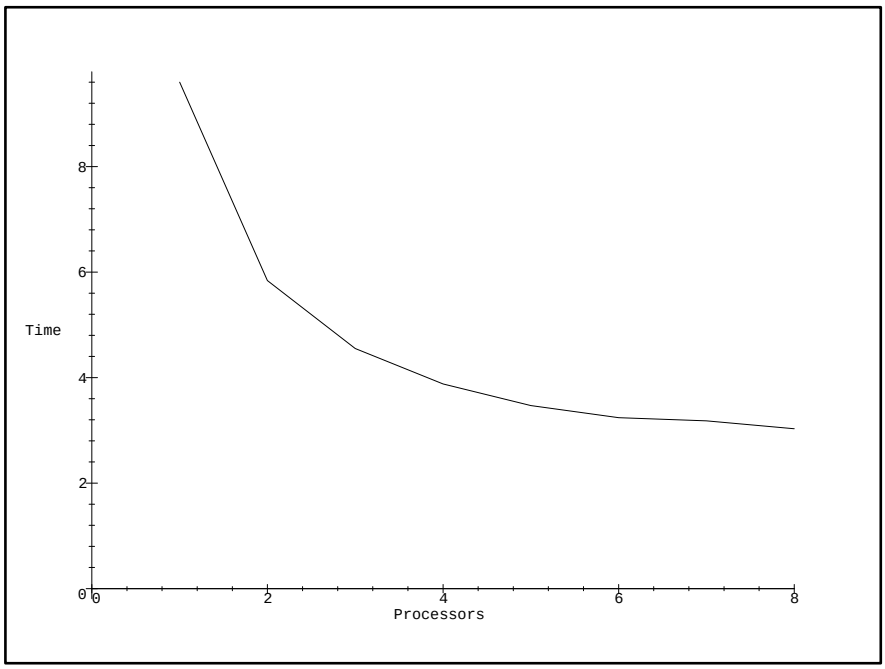
Εικόνα 65 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 128$ στην block red-black SSOR.



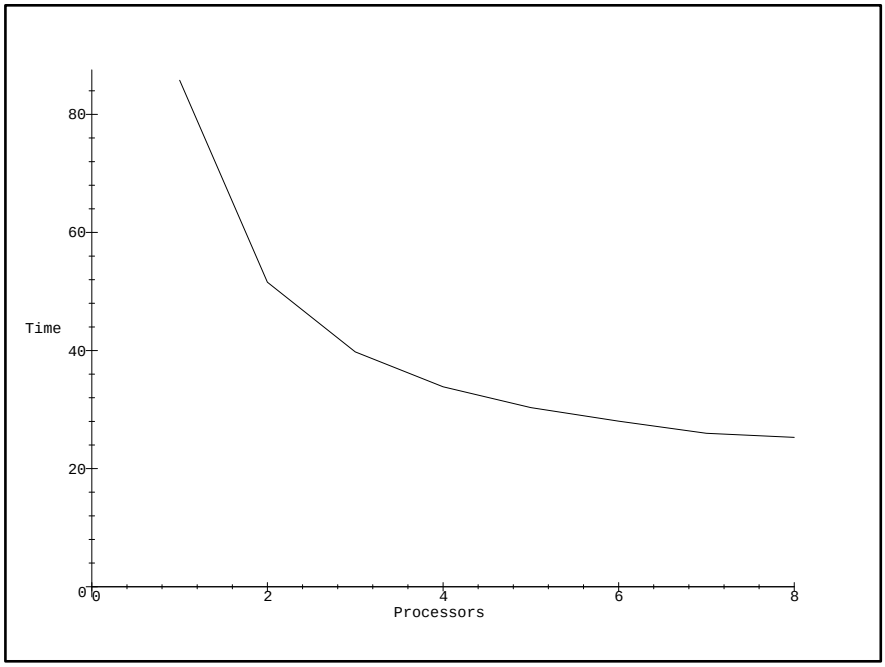
Εικόνα 66 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 256$ στην block red-black SSOR.



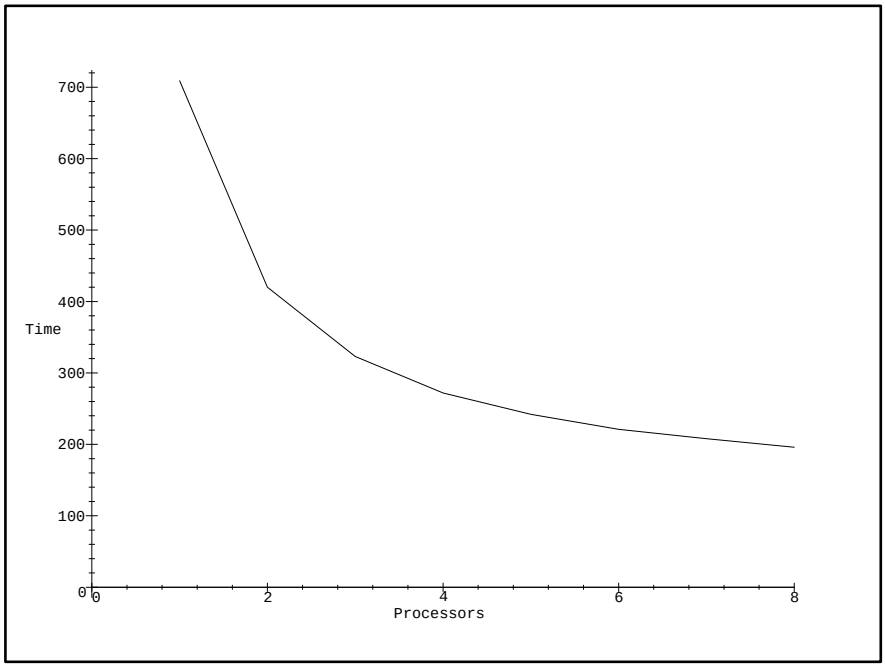
Εικόνα 67 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 32$ στην block red-black SGS.



Εικόνα 68 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 64$ στην block red-black SGS.



Εικόνα 69 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 128$ στην block red-black SGS.



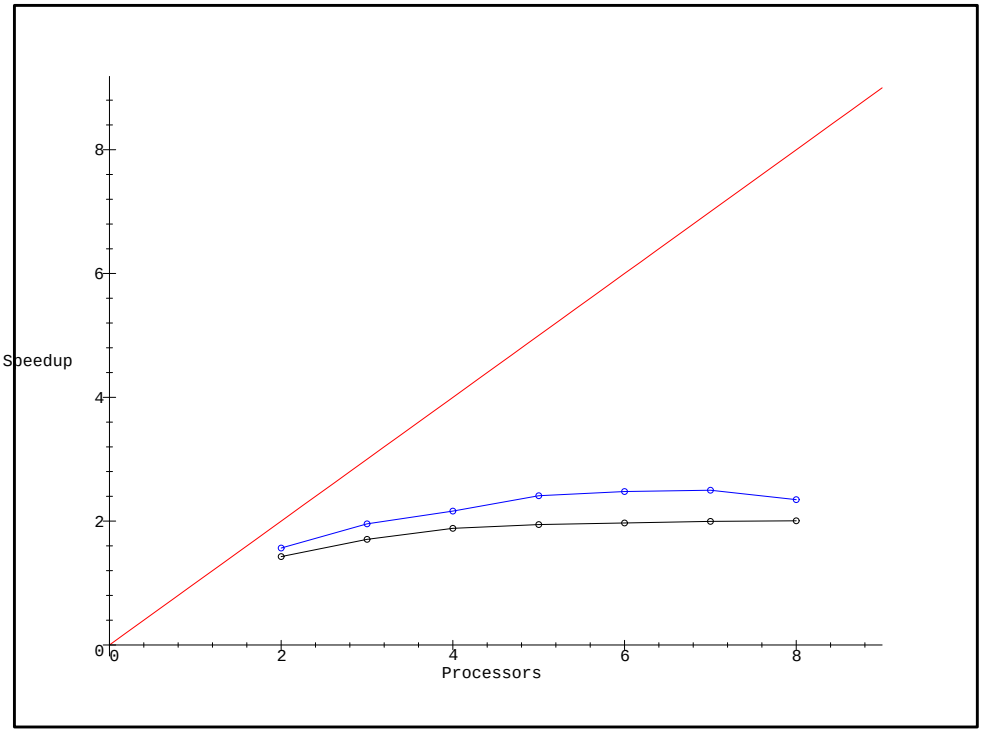
Εικόνα 70 : Μείωση του συνολικού χρόνου ως προς τον αριθμό επεξεργαστών για $n_s = 256$ στην block red-black SGS.

Για την μελέτη της απόδοσης των παράλληλων αλγορίθμων θα χρησιμοποιήσουμε ξανά την τιμή του speedup $speedup_{rb}$. Οι παρακάτω πίνακες T28 και T29 εμφανίζουν τις τιμές αυτές για SSOR precondition και για SGS αντίστοιχα.

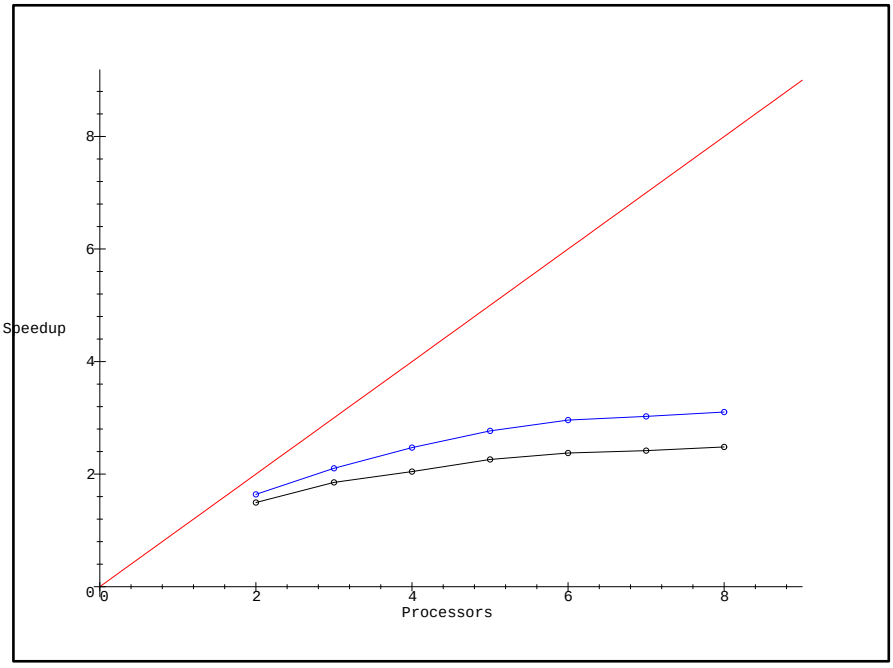
T28	$Speedup_{rb}$ for SSOR preconditioned BiCGSTAB						
n_s	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	1.305	1.478	1.546	1.478	1.415	1.358	1.305
32	1.565	1.956	2.162	2.411	2.478	2.500	2.347
64	1.640	2.104	2.470	2.768	2.960	3.027	3.104
128	1.665	2.160	2.545	2.841	3.075	3.308	3.394
256	1.685	2.203	2.607	2.924	3.193	3.412	3.569

T29	$Speedup_{rb}$ for SGS preconditioned BiCGSTAB						
n_s	2 proc.	3 proc.	4 proc.	5 proc.	6 proc.	7 proc.	8 proc.
16	1.305	1.478	1.546	1.478	1.415	1.358	1.305
32	1.565	1.956	2.162	2.411	2.478	2.500	2.347
64	1.644	2.110	2.474	2.766	2.963	3.019	3.168
128	1.663	2.156	2.531	2.832	3.064	3.300	3.391
256	1.688	2.195	2.607	2.930	3.194	3.409	3.563

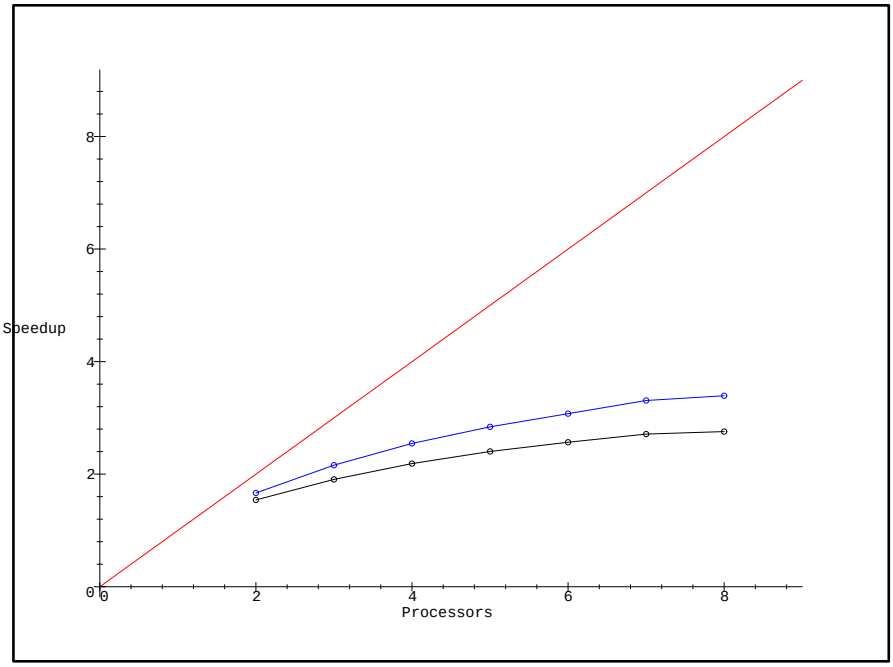
Από την μελέτη των γραφικών παραστάσεων και τις μετρήσεις των τιμών του speedup προκύπτει, ότι από την προσθήκη του δεύτερου επεξεργαστή αρχίζει να εμφανίζεται σημαντική μείωση του χρόνου εκτέλεσης της BiCGSTAB. Επίσης αξιόλογη απόδοση εμφανίζεται για διακριτοποιήσεις μεγαλύτερες του 16. Οι τέσσερις επόμενες Εικόνες εμφανίζουν τα γραφήματα των τιμών του speedup για την SSOR preconditioned BiCGSTAB και την SOR. Σε όλα τα γραφήματα με κόκκινο χρώμα εμφανίζεται η βέλτιστη τιμή του speedup, με μπλε η τιμή του $speedup_{rb}$ για την BiCGSTAB, ενώ με μαύρο η τιμή του $speedup_{rb}$ της SOR.



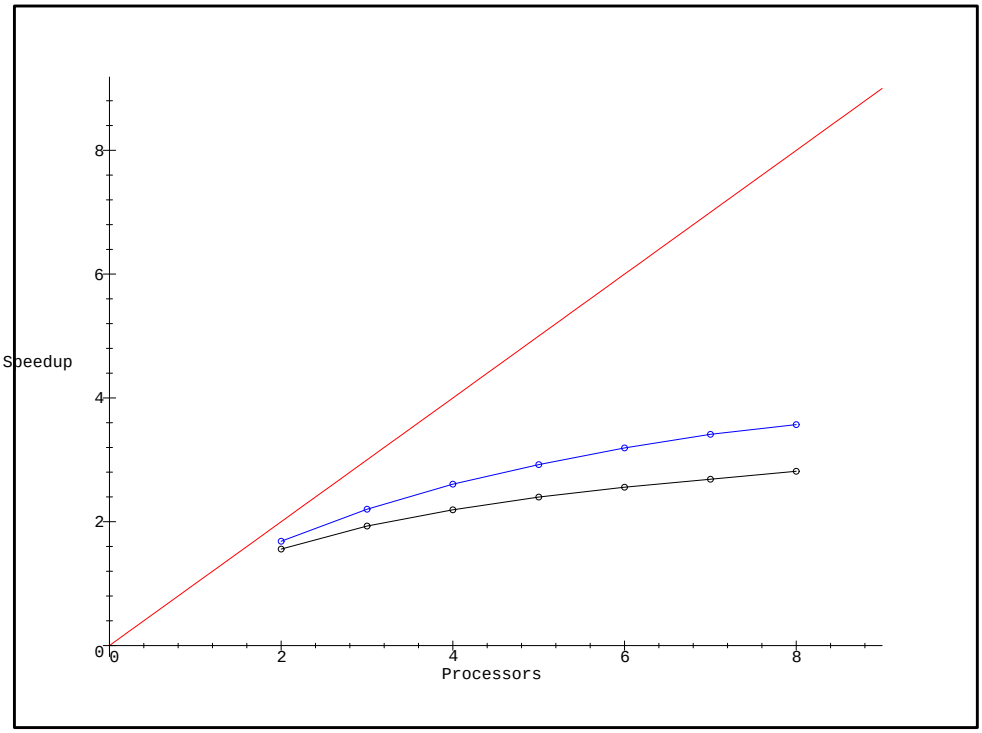
Εικόνα 71 : Οι τιμές των speedup για τις επαναληπτικές μεθόδους SOR και BiCGSTAB για $n_s = 32$.



Εικόνα 72 : Οι τιμές των speedup για τις επαναληπτικές μεθόδους SOR και BiCGSTAB για $n_s = 64$.



Εικόνα 73 : Οι τιμές των speedup για τις επαναληπτικές μεθόδους SOR και BiCGSTAB για $n_s = 128$.



Εικόνα 74 : Οι τιμές των speedup για τις επαναληπτικές μεθόδους SOR και BiCGSTAB για $n_s = 256$.

Παρατηρούμε από τις μετρήσεις των τιμών του speedup, ότι η συμπεριφορά των

δυο μεθόδων είναι παρόμοια. Έτσι είναι φανερό ότι στους δυο επεξεργαστές οι τιμές του speedup είναι πολύ κοντά στο θεωρητικό βέλτιστο, ενώ όσο προστίθενται περισσότεροι επεξεργαστές εμφανίζεται μια σταθερή απομάκρυνση από τη βέλτιστη τιμή. Αυτό οφείλεται στην αρχιτεκτονική σύνδεσης των επεξεργαστών του συγκεκριμένου υπολογιστικού συστήματος. Δηλαδή στην καθυστέρηση επικοινωνίας των επεξεργαστών με την κοινή μνήμη, λόγω της τοποθέτησης τους ανά ζεύγη και της επικοινωνίας μεταξύ των ζευγών μέσω router. Παρολαυτά για διαμερίσεις με μικρό βήμα διακριτοποίησης εξακολουθεί να υπάρχει τάση αύξησης των τιμών του speedup.

4.4.3 Εφαρμογή της επαναληπτικής μεθόδου BiCGSTAB σε παράλληλες αρχιτεκτονικές διανεμημένης μνήμης

Ο σχεδιασμός παράλληλου αλγορίθμου της BiCGSTAB για ένα υπολογιστικό σύστημα διανεμημένης μνήμης - *distributed memory*[14] - με τεχνικά χαρκτηριστικά όμοια με αυτό της υλοποίησης της SOR, έχει περισσότερες απαιτήσεις επικοινωνίας από τον αντίστοιχο της SOR, στον οποίο έγινε μια αρχική κατανομή των δεδομένων στους επεξεργαστές και στη συνέχεια ο καθένας τους τα αναβάθμιζε παίρνοντας κάποια από τα δεδομένα των γειτονικών επεξεργαστών. Αυτό συμβαίνει γιατί η μέθοδος BiCGSTAB σε κάθε επαναληπτικό βήμα της, όπως προκύπτει από τον αλγόριθμο της στην Εικόνα 15, δημιουργεί τρεις σταθερές και υπολογίζει δυο νόρμες διανυσμάτων. Οι σταθερές αυτές είναι οι ρ_{i-1}, α_i και ω_i, ενώ οι νόρμες είναι, του διανύσματος $\| s \|$ και του κριτηρίου σύγκλισης $\| M^{-1}r^{(i)} \|$. Έτσι όλες οι παραπάνω τιμές θα πρέπει να είναι γνωστές σε όλους τους επεξεργαστές και σε κάθε επαναληπτικό βήμα. Επίσης, όπως προέκυψε και στον παράλληλο αλγόριθμο για συστήματα κοινής μνήμης, στην τεχνική του SSOR precondition και ιδιαίτερα στην επίλυση των τριγωνικών γραμμικών συστημάτων ο κάθε επεξεργαστής χρειαζόταν κάποια από τα δεδομένα άλλων συγκεκριμένων επεξεργαστών. Έχοντας υπόψη

αυτές τις δυο βασικές παρατηρήσεις στην κατασκευή παράλληλου αλγορίθμου της BiCGSTAB θα πρέπει :

1. κάποιοι επεξεργαστές να βρεθούν όσο το δυνατόν πιο κοντά, για να ανταλλάσουν τα κατάλληλα δεδομένα γρήγορα και χωρίς να καθυστερούν τους υπόλοιπους

2. τόσο για τον υπολογισμό, όσο και για την μεταφορά των τιμών των προαναφερθέντων σταθερών σε όλους τους επεξεργαστές η αρχιτεκτονική σύνδεσης τους θα είναι αυτή με το λιγότερο επικοινωνιακό κόστος.

Για να ικανοποιηθούν οι δυο παραπάνω συνθήκες θα πρέπει να χρησιμοποιηθούν δυο διαφορετικές τοπολογίες - εικονικές αρχιτεκτονικές - σύνδεσης ανάμεσα στους επεξεργαστές για κάθε επαναληπτικό βήμα. Η πρώτη θα είναι αυτή του δυαδικού δέντρου - $binary\ tree$ - με κόστος $O(log_2 P)$ μεταφοράς δεδομένων σε όλους τους P το πλήθος επεξεργαστές, και η δεύτερη θα είναι αυτή της σύνδεσης σε σειρά - $pipeline$ - για την μεταφορά των δεδομένων γειτονικών επεξεργαστών κατά τις διαδικασίες επίλυσης των τριγωνικών συστημάτων στη μέθοδο του SSOR preconditioning.

Ο παρακάτω αλγόριθμος εμφανίζει την παραπάνω διαδικασία

Παράλληλος αλγόριθμος SSOR preconditioned BiCGSTAB
για υπολογιστικά συστήματα διανεμημένης μνήμης

Επιλογή αρχικής προσέγγισης $\boldsymbol{x}^{(0)}$ *της λύσης* $\boldsymbol{x}$

Δημιουργία αρχιτεκτονικής σύνδεσης δυαδικού δέντρου ανάμεσα στους επεξεργαστές

Block διαμέριση των διανυσμάτων $\boldsymbol{x}^{(0)}$ *και* $\boldsymbol{b}$ *- Κατανομή τους στους επεξεργαστές μαζί με τους βασικούς πίνακες* A_1, A_2, A_3 *και* A_4.

Δημιουργία αρχιτεκτονικής σύνδεσης σε σειρά ανάμεσα στους επεξεργαστές

Παράλληλος Υπολογισμός $M^{-1}\boldsymbol{b}$

Δημιουργία αρχιτεκτονικής σύνδεσης δυαδικού δέντρου ανάμεσα στους επεξεργαστές

Παράλληλος Υπολογισμός $\| M^{-1}\boldsymbol{b} \|_2$

for $j = 1, 2, ..., P$ **do in parallel**

$r_j^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

Επιλογή αρχικής τιμής $\hat{r}_j$ *(συνήθως* $\hat{r}_j = r_j{}^{(0)}$*)*

enddo

for $i = 1, 2, ...$

Παράλληλος Υπολογισμός $\rho_{i-1} = \hat{r}^T r^{(i-1)}$

if $\rho_{i-1} = 0$ *η μέθοδος αποτυγχάνει*

for $j = 1, 2, ..., P$ **do in parallel**

if $i = 1$

$p_j^{(1)} = r_j^{(0)}$

else

$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \frac{\alpha_{i-1}}{\omega_{i-1}}$

$p_j^{(i)} = r_j^{(i-1)} + \beta_{i-1} (p_j^{(i-1)} - \omega_{i-1} v_j^{(i-1)})$

endif

enddo

Δημιουργία αρχιτεκτονικής σύνδεσης σε σειρά ανάμεσα στους επεξεργαστές

Παράλληλη Επίλυση του $\mathrm{M}\,\hat{p} = p^{(i)}$

Δημιουργία αρχιτεκτονικής σύνδεσης δυαδικού δέντρου ανάμεσα στους επεξεργαστές

for $j = 1, 2, ..., P$ **do in parallel**

Παράλληλος υπολογισμός του $v_j^{(i)} = A\,\hat{p}_j$

enddo

Παράλληλος υπολογισμός του $\alpha_i = \frac{\rho_{i-1}}{\hat{r}^T v^{(i)}}$

for $j = 1, 2, ..., P$ **do in parallel**

$s_j = r_j^{(i-1)} - \alpha_i v_j^{(i)}$

enddo

Παράλληλος υπολογισμός του $\| s \|_2$

for $j = 1, 2, ..., P$ **do in parallel**

if $\| s \|_2$ *αρκετά μικρό* **then**

$\boldsymbol{x}_j^{(i)} = \boldsymbol{x}_j^{(i-1)} + \alpha_i \hat{p}_j$ *έξοδος*

enddo

Δημιουργία αρχιτεκτονικής σύνδεσης σε σειρά ανάμεσα στους επεξεργαστές

Παράλληλη Επίλυση του $\mathrm{M}\, z = s$

Δημιουργία αρχιτεκτονικής σύνδεσης δυαδικού δέντρου ανάμεσα στους επεξεργαστές

for $j = 1, 2, ..., P$ **do in parallel**

Παράλληλος υπολογισμός του $t_j = A\, z_j$

enddo

Παράλληλος υπολογισμός του $\omega_i = \frac{s^T t}{t^T t}$

for $j = 1, 2, ..., P$ **do in parallel**

$\boldsymbol{x}_j^{(i)} = \boldsymbol{x}_j^{(i-1)} + \alpha_i \hat{p}_j + \omega_i z_j$

if $\omega_i = 0$ *έξοδος*

$r_j^{(i)} = s_j - \omega_i t_j$

enddo

Δημιουργία αρχιτεκτονικής σύνδεσης σε σειρά ανάμεσα στους επεξεργαστές

Παράλληλος υπολογισμός του $\mathrm{M}^{-1} r^{(i)}$

Δημιουργία αρχιτεκτονικής σύνδεσης δυαδικού δέντρου ανάμεσα στους επεξεργαστές

Παράλληλος υπολογισμός του $\| \mathrm{M}^{-1} r^{(i)} \|_2$ *και Ελεγχος σύγκλισης*

end

Παράλληλη συγκέντρωση των τμημάτων της λύσης $\boldsymbol{x}^{(i)}$ *από όλους τους επεξεργαστές*

Στη συνέχεια θα γίνει απεικόνιση του παραπάνω αλγορίθμου σε ένα εικονικό παράλληλο σύστημα διανεμημένης μνήμης με απεριόριστο αριθμό επεξεργαστών και αμέσως μετά θα τροποποιηθεί κατάλληλα για πραγματικές αρχιτεκτονικές σταθερής διάστασης.

Απεικόνιση του αλγορίθμου σε ένα εικονικό παράλληλο σύστημα διανεμημένης μνήμης

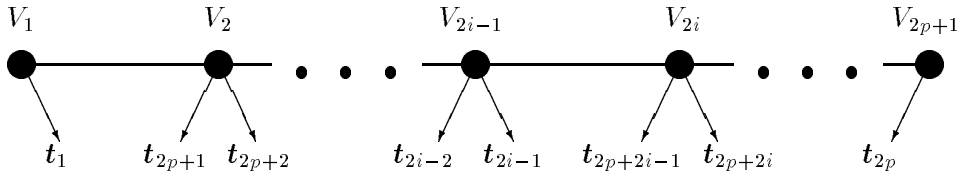
Αν δεχτούμε ότι έχει γίνει διαμέριση του χωρίου Ω σε $n_s = 2p$ το πλήθος υποδιαστήματα ως προς την x και την y κατεύθυνση, δηλαδή υπάρχουν $2p+1$ κάθετες και άλλες τόσες οριζόντιες γραμμές πλέγματος, μπορούμε να απεικονίσουμε τους αγνώστους που αντιστοιχούν σε κάθε κάθετη γραμμή πλέγματος σε καθένα επεξεργαστή. Έτσι θα έχει γίνει η ίδια block διαμέριση των διανυσμάτων $\boldsymbol{b}$ και $\boldsymbol{x}^{(0)}$, όπως και στον αντίστοιχο αλγόριθμο της SOR καθώς και η κατανομή των τμημάτων τους επεξεργαστές.

Όσο αφορά την σύνδεση των επεξεργαστών με την αρχιτεκτονική του δυαδικού δέντρου, ο πρώτος επεξεργατής που βρίσκεται στη ρίζα του θα αναλάβει την συγκέντρωση και τον υπολογισμό των σταθερών και της νόρμας για το κριτήριο σύγκλισης. Επίσης θα αποφασίζει αν θα γίνει το επόμενο επαναληπτικό βήμα και θα ενημερώνει τους υπόλοιπους επεξεργαστές. Ουσιαστικά η μέθοδος επικοινωνίας θα ακολουθεί το μοντέλο $master$ - $slave$. Αν θεωρήσουμε ότι κάθε επεξεργαστής δεν μπορεί να λάβει ή και να στέλνει ταυτόχρονα σε άλλους επεξεργαστές, τότε αν ένας αριθμός χρειάζεται χρόνο t_c για να μεταφερθεί από τον ένα στον άλλο επεξεργαστή, ο αριθμός αυτός χρειάζεται χρόνο $2t_c ln(2p+1)$ για να φτάσει στη ρίζα του δυαδικού δέντρου. Έτσι συνολικά για να υπολογιστεί κάθε σταθερά ή νόρμα και να ενημερωθούν όλοι οι επεξεργαστές χρειάζεται χρόνος $4t_c ln(2p+1)$ και επειδή αυτή η διαδικασία εμφανίζεται πέντε φορές σε κάθε επαναληπτικό βήμα θα χρειαστεί συνολικά $20t_c ln(2p+1)$.

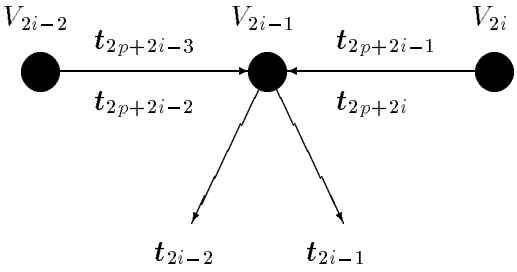
Οι υπόλοιπες βασικές πράξεις γραμμικής άλγεβρας μεταξύ των τμημάτων των διανυσμάτων, που έχουν αντιστοιχιστεί σε κάθε επεξεργαστή, μαζί με τους πολλαπλασιασμούς τους με το αντίστοιχο τμήμα του collocation πίνακα γίνονται εντελώς παράλληλα σε κάθε επεξεργαστή. Ιδιαίτερα για τις πράξεις του πολλαπλασιασμού διανύσματος, είτε με τον collocation πίνακα, είτε με το διαγώνιο του block στη

διαδικασία του preconditioning, μπορούν να χρησιμοποιηθούν οι αλγόριθμοι του παρουσιάστηκαν στην προηγούμενη ενότητα και αναφέρονται σε παράλληλα συστήματα κοινής μνήμης.

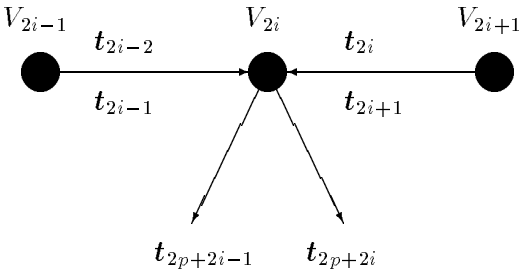
Για την αρχιτεκτονική σε σειρά, που θα εφαρμόζεται στα τρια σημεία δράσης του precondioner πίνακα σε κάθε επαναληπτικό βήμα, έχει γίνει αντιστοίχηση σε κάθε έναν από τους $V_j\ ,\ j = 1, \ldots, 2p+1$ επεξεργαστές των δυο τμημάτων της λύσης του γραμμικού συστήματος, που αναφέρονται στους αγνώστους από την δεξιά και αριστερή πλευρά κάθε κάθες γραμμής πλέγματος. Η επόμενη εικόνα εμφανίζει την σύνδεσή τους και την απεικόνιση των δυο τμημάτων της λύσης $\boldsymbol{t}$ του άνω ή κάτω τριγωνικού συστήματος της τεχνικής του preconditioning.



Είναι φανερό, ότι στους *περιττούς* επεξεργαστές έχουν απεικονισθεί οι red γραμμές πλέγματος και στους *άρτιους* οι $black$. Σύμφωνα με την block διαμέριση μεγέθους $2p$ όλων των διανυσμάτων της προηγούμενης ενότητας, σε κάθε περιττό επεξεργαστή $V_{2i-1}\ ,\ i = 1, \ldots, p+1$, έχει απεικονισθεί ο υπολογισμός των $\boldsymbol{t}_{2i-2}$ και $\boldsymbol{t}_{2i-1}$ τμημάτων της λύσης, ενώ σε κάθε άρτιο επεξεργαστή $V_{2i}\ ,\ i = 1, \ldots, p$, έχουν απεικονισθεί τα τμήματα $\boldsymbol{t}_{2p+2i-1}$ και $\boldsymbol{t}_{2p+2i}$. Η εμφανιζόμενη ανομοιομορφία στον πρώτο V_1 και τελευταίο V_{2p+1} επεξεργαστή είναι αποτέλεσμα των συνοριακών συνθηκών της διαφορικής εξίσωσης.



Παρατηρούμε επίσης, ότι για να υπολογίσει κάθε περιττός (red) επεξεργαστής V_{2i-1} στην επίλυση του άνω τριγωνικού συστήματος τη τιμή του τμήματος $\boldsymbol{t}_{2i-2}$ και $\boldsymbol{t}_{2i-1}$ της λύσης, σύμφωνα με τα βήματα του αλγορίθμου BiCGSTAB προηγούμενης ενότητας, θα χρειαστεί να επικοινωνήσει με τους γειτονικούς $black$ V_{2i-2} και V_{2i} για να λάβει από αυτούς τα διανύσματα $\boldsymbol{t}_{2p+2i-3}$, $\boldsymbol{t}_{2p+2i-2}$ και $\boldsymbol{t}_{2p+2i-1}$, $\boldsymbol{t}_{2p+2i}$ αντίστοιχα.



Ομοίως, οι άρτιοι $black$ επεξεργαστές V_{2i} για να υπολογίσουν τα τμήματα $\boldsymbol{t}_{2p+2i-1}$ και $\boldsymbol{t}_{2p+2i}$ της λύσης του κάτω τριγωνικού συστήματος της τεχνικής του SSOR precondition, σύμφωνα με τον αλγορίθμο της BiCGSTAB, θα πρέπει να λάβουν από τους γειτονικούς red V_{2i-1} και V_{2i+1} επεξεργαστές τα ζεύγη διανυσμάτων $\boldsymbol{t}_{2i}$, $\boldsymbol{t}_{2i-1}$ και $\boldsymbol{t}_{2i}$, $\boldsymbol{t}_{2i+1}$ αντίστοιχα.

Συνδυάζοντας τα παραπάνω με την αρχιτεκτονική της εικονικής παράλληλης μηχανής και το επαναληπτικό βήμα της SSOR preconditioned BiCGSTAB σε παράλληλη μορφή, οι δυο παρακάτω αλγόριθμοι περιγράφουν τις δυο επιμέρους διαδικασίες της τεχνικής του SSOR preconditioning αυτές της επίλυσης των άνω και κάτω τριγωνικών συστημάτων. Κάθε διαδικασία αποτελείται από ένα Red και ένα Black Cycle και ειδικότερα για την επίλυση του κάτω τριγωνικού συστήματος εκτελείται πρώτα η Red και μετά η Black Cycle, ενώ στην επίλυση του άνω γίνεται αντίστροφα. Οι

διαδικασίες αυτές εμφανίζονται παρακάτω :

Red Cycle για την επίλυση του κάτω τριγωνικού συστήματος

Computation Phase

for $i = 1$ *to* $p+1$ *do in parallel*
V_{2i-1} *computes* $\boldsymbol{t}_{2i-2}$ and $\boldsymbol{t}_{2i-1}$
enddo

Black Cycle για την επίλυση του κάτω τριγωνικού συστήματος

Communication Phase

for $i = 1$ *to* p *do in parallel*
V_{2i-1} *sends to* V_{2i} the vectors
$\boldsymbol{t}_{2i-2}$, $\boldsymbol{t}_{2i-1}$
enddo

for $i = 1$ *to* p *do in parallel*
V_{2i+1} *sends to* V_{2i} the vectors
$\boldsymbol{t}_{2i}$, $\boldsymbol{t}_{2i+1}$
enddo

Computation Phase

for $i = 1$ *to* p *do in parallel*
V_{2i} *computes* $\boldsymbol{t}_{2p+2i-1}$ and $\boldsymbol{t}_{2p+2i}$
enddo

Black Cycle για την επίλυση του άνω τριγωνικού συστήματος

Computation Phase

for $i = 1$ *to* p *do in parallel*
V_{2i} *computes* $\boldsymbol{t}_{2p+2i-1}$ and $\boldsymbol{t}_{2p+2i}$
enddo

Red Cycle για την επίλυση του άνω τριγωνικού συστήματος

Communication Phase

for $i = 2$ *to* $p+1$ *do in parallel*
V_{2i-2} *sends to* V_{2i-1} the vectors
$\boldsymbol{t}_{2p+2i-3}$, $\boldsymbol{x}_{2p+2i-2}$
enddo

for $i = 1$ *to* p *do in parallel*
V_{2i} *sends to* V_{2i-1} the vect ors
$\boldsymbol{t}_{2p+2i-1}$, $\boldsymbol{t}_{2p+2i}$
enddo

Computation Phase

for $i = 1$ *to* $p+1$ *do in parallel*
V_{2i-1} *computes* $\boldsymbol{t}_{2i-2}$ and $\boldsymbol{t}_{2i-1}$
enddo

Από τον παράλληλο αλγόριθμο της SSOR preconditioned BiCGSTAB προκύπτει, ότι το υπολογιστικό κόστος $t_{comp}^{(m)}$ στο επαναληπτικό βήμα m της είναι $O(n_s)$. Αυτό ισχύει, γιατί οι βασικές πράξεις γραμμικής άλγεβρας για τον υπολογισμό κάθε τμήματος της λύσης, περιλαμβάνουν κυρίως πολλαπλασιασμούς των πινάκων ζώνης A_3 και A_4 διαστάσεων $5 \times 2n_s$ καθώς επίσης και μπρος πίσω αντικαταστάσεις με τους παραγοντοποιημένους ήδη πίνακες ζώνης A_1 και A_2.

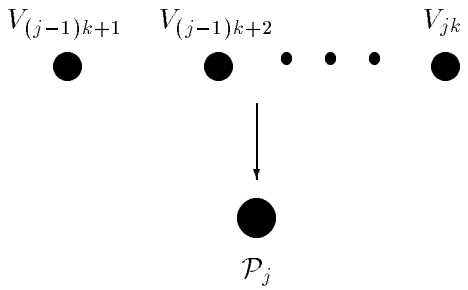
Στο ήδη υπολογισμένο $20ln(2p+1)$ κόστος επικοινωνίας $t_{comm}^{(m)}$ κάθε επαναληπτικού βήματος θα πρέπει να προστεθεί και ο χρόνος μεταφοράς από έναν επεξεργαστή στον γειτονικό του συνολικά 24 διανυσμάτων (4 κατά την διάρκεια της Red Cycle διαδικασίας και 4 κατά την διάρκεια της Black Cycle) μεγέθους $4p = 2n_s$ το καθένα σε τρεις διαδικασίες, ή πιο σωστά ο χρόνος μεταφοράς 12 διανυσμάτων (6 κατά την διάρκεια της Red Cycle διαδικασίας και 6 κατά την Black Cycle) μεγέθους $8p = 4n_s$ το καθένα.

Απεικόνιση του αλγορίθμου σε ένα παράλληλο σύστημα διανεμημένης μνήμης και σταθερής διάστασης

Ο παράλληλος αλγορίθμο της SSOR preconditioned BiCGSTAB της προηγούμενης ενότητας θα απεικονιστεί σε ένα παράλληλο σύστημα διανεμημένης μνήμης σταθερής διάστασης με έναν προκαθορισμένο αριθμό P επεξεργαστών. Αυτό άλλωστε ισχύει και σε ένα πραγματικό παράλληλο υπολογιστικό σύστημα , δηλαδή απαρτίζεται από P επεξεργαστές $\mathcal{P}_j \quad , \quad (j = 1, \ldots, P)$, οι οποίοι είναι συνδεμένοι σε μορφή δυαδικού δέντρου, ενώ κατά τις διαδικασίες επίλυσης των preconditioned γραμμικών συστημάτων αυτοί συνδέονται σε σειρά. Έτσι το υπολογιστικό φορτίο θα πρέπει να ανακατενεμηθεί, με άλλα λόγια θα γίνει κάποια ομαδοποίηση των εικονικών επεξεργαστών και θα αντιστοιχιστεί κάθε ομάδα τους σε ένα πραγματικό επεξεργαστή. Οι δυνατές περιπτώσεις είναι δύο και αναλύονται παρακάτω.

Περίπτωση όπου $n_s = k\,P$

Στην περίπτωση αυτή γίνεται ομαδοποίηση k το πλήθος διαδοχικών εικονικών επεξεργαστών οι οποίοι αντιστοιχίζονται σε καθένα πραγματικό επεξεργαστή $\mathcal{P}_j \quad , \quad (j = 1, \ldots, P)$. Έτσι υπάρχει ισοκατανομή του υπολογιστικού φορτίου όπως εμφανίζεται στο παρακάτω σχήμα.



Κάθε άλλη δυνατή αντιστοίχιση αυξάνει το κόστος επικοινωνίας. Δηλαδή αν γίνει απευθείας απεικόνιση των αγνώστων στους επεξεργαστές, με πιθανό αποτέλεσμα τον υπολογισμό των δυο τμημάτων της προσεγγιστικής λύσης, τα οποία ανήκουν

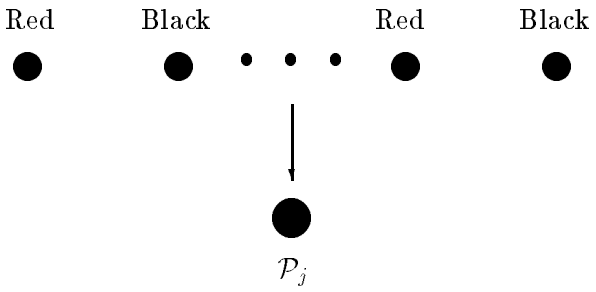
στην ίδια κάθετη γραμμή πλεγματος και έχουν αντιστοιχιστεί σε ένα εικονικό επεξεργαστή, από διαφορετικούς πραγματικούς, το κόστος επικοινωνίας σε μια τέτοια περίπτωση θα είναι πολλαπλάσιο, γιατί απαιτείται η μεταφορά περισσότερων διανυσμάτων από κάθε επεξεργαστή κατά την επίλυση των τριγωνικών συστημάτων στην τεχνική του precondition. Άρα η αντιστοίχιση του παραπάνω σχήματος είναι η **βέλτιστη** και σε κάθε παργματικό επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί οι k εικονικοί επεξεργαστές $V_{(j-1)k+1}, \ldots, V_{jk}$, εκτός από τον τελευταίο $\mathcal{P}_P$, στον οποίο θα αντιστοιχιστούν $k+1$ εικονικοί επεξεργαστές, αφού το εικονικό παράλληλο σύστημα απαρτίζεται από $n_s + 1$ επεξεργαστές.

Για την κατανομή των δεδομένων στους επεξεργαστές υπάρχουν οι εξής παρατηρήσεις:

- Αν ο αριθμός k είναι άρτιος τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

  $(j-1)k+1$ είναι *περιττός* ενώ ο jk είναι *άρτιος*.

  Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι αντίστοιχα *red (περιττοί)* και *black (άρτιοι)* επεξεργαστές σύμφωνα με το παρακάτω σχήμα



  Άρα στον τυχαίο επεξεργαστή $\mathcal{P}_j$ έχουν αντιστοιχιστεί k *red* διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k, \ldots, jk-1$$

  και k *black* διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk \, .$$

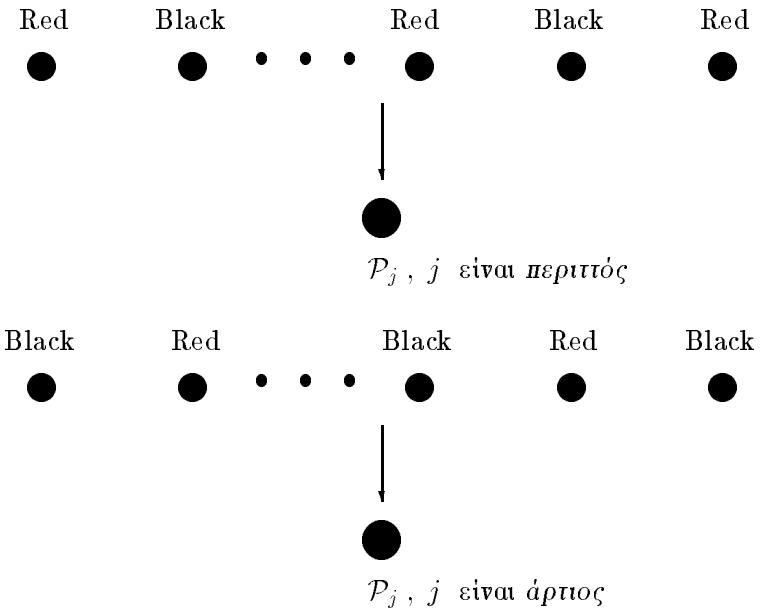
- Αν ο αριθμός k είναι περιττός τότε για τους δείκτες $(j-1)k+1$ και jk ισχύει

$$(j-1)k+1 \text{ και } jk \text{ είναι } \textit{περιττά}, \text{ όταν το } j \text{ είναι } \textit{περιττό}$$

ενώ τα

$$(j-1)k+1 \text{ και } jk \text{ είναι } \textit{άρτια}, \text{ όταν το } j \text{ είναι } \textit{άρτιο}.$$

Έτσι οι εικονικοί επεξεργαστές $V_{(j-1)k+1}$ και V_{jk} θα είναι και οι δύο *red (περιττοί)* όταν το j είναι *περιττό*, ενώ θα είναι και οι δύο *black (άρτιοι)* όταν το j είναι *άρτιος*. Αυτό απεικονίζεται σχηματικά παρακάτω.



Άρα, όταν το j είναι *περιττό*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k+1$ *red* διανύσματα

$$\boldsymbol{x}_l \ , \ l = (j-1)k, \ldots, jk$$

και τα $k-1$ *black* διανύσματα

$$\boldsymbol{x}_{2p+l} \ , \ l = (j-1)k+1, \ldots, jk-1 \ ,$$

ενώ, όταν το j είναι *άρτιο*, στον επεξεργαστή $\mathcal{P}_j$ αντιστιχοίζονται τα $k-1$ red διανύσματα

$$\boldsymbol{x}_l\ ,\ l=(j-1)k+1,\ldots,jk-1$$

και τα $k+1$ $black$ διανύσματα

$$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k,\ldots,jk\ .$$

Σε αυτό το σημείο θα πρέπει να τονίσουμε ότι το πρώτο red διάνυσμα $\boldsymbol{x}_0$ θα έχει υπολογιστεί από τον πρώτο επεξεργαστή $\mathcal{P}_1$, ενώ το τελευταίο $\boldsymbol{x}_{4p}$ αντίστοιχα από τον $\mathcal{P}_P$. Λαμβάνοντας υπόψη την παραπάνω ανάλυση, καταλήγουμε στις δυο παρακάτω παρατηρήσεις.

- όταν ο δείκτης k είναι *άρτιος* η απεικόνιση των k εικονικών επεξεργαστών θα γίνεται με την διάταξη $RB\,RB\cdots RB$, όπου R συμβολίζει "Red" και αντίστοιχα το B "Black" εικονικό επεξεργαστή. Έτσι κατά την διάρκεια της Red Cycle διαδικασίας της επίλυσης του άνω τριγωνικού συστήματος στην τεχνική του preconditioning, ο επεξεργαστής $\mathcal{P}_{\hat{j}}$ $(\hat{j}=1,\ldots,P-1)$, θα πρέπει να *στείλει* στον επεξεργαστή $\mathcal{P}_{\hat{j}+1}$ τα διανύσματα που σχετίζονται με τον τελευταίο του black εικονικό επεξεργαστή V_{jk}, ενώ κατά την διάρκεια της Black Cycle διαδικασίας της επίλυσης του κάτω τριγωνικού συστήματος στην τεχνική του preconditioning, ο επεξεργαστής $\mathcal{P}_{\hat{j}+1}$ θα πρέπει να *στείλει* στον $\mathcal{P}_{\hat{j}}$ τα διανύσματα που σχετίζονται με τον πρώτο του red εικονικό επεξεργαστή V_{jk+1}

- όταν ο δείκτης k είναι *περιττός* η απεικόνιση θα ακολουθεί τη διάταξη $RB\,RB\cdots RB\ \ R$, όταν ο δείκτης j είναι *περιττός* και $BR\,BR\cdots BR\,B$, όταν είναι *άρτιος*. Έτσι κατά τη διάρκεια της Red (αντίστοιχα Black) Cycle διαδικασίας της επίλυσης του άνω (αντ. κάτω) τριγωνικού συστήματος στην τεχνική του preconditioning, ο επεξεργαστής $\mathcal{P}_j$ $(j=\text{άρτιος})$ $(\text{αντ.}j=$

περιττός) θα πρέπει να *στείλει* στους επεξεργαστές $\mathcal{P}_{j-1}$ και $\mathcal{P}_{j+1}$ αντίστοιχα τα διανύσματα που σχετίζονται με το πρώτο και τελευταίο black (αντ. red) εικονικούς επεξεργαστές $V_{(j-1)k+1}$ και V_{jk}

Σύμφωνα με την παραπάνω ανάλυση ο παράλληλος αλγόριθμος της SSOR preconditioned BiCGSTAB για παράλληλα συστήματα σταθερής διάστασης δεν αλλάζει τη γενική του μορφή, αλλά υπάρχει κάποια διαφοροποίηση στις επιμέρους διαδικασίες της επίλυσης των άνω και κάτω τριγωνικών συστημάτων στην τεχνική του preconditioning, όπως αυτές περιγράφονται στον παρακάτω αλγόριθμο :

Αλγόριθμος επίλυσης **κάτω** *τριγωνικού συστήματος* $M_1\boldsymbol{t} = \boldsymbol{z}$

Red Cycle για άρτιο k

Computation Phase

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k$ *to* $jk-1$ *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
  endd.
enddo

Black Cycle για άρτιο k

Communication Phase

for $q = 1$ *to* $\hat{s}$ *do in parallel*
  $\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
    $\boldsymbol{t}_{2qk}$, $\boldsymbol{t}_{2qk+1}$
enddo

for $q = 1$ *to* s *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors
    $\boldsymbol{t}_{(2q-1)k}$, $\boldsymbol{t}_{(2q-1)k+1}$
enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  for $l = (j-1)k+1$ *to* jk *do*
    $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
  enddo
enddo

Red Cycle για περιττό k

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if j *odd* *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  else
    for $l = (j-1)k+1$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  endif
enddo

Black Cycle για περιττό k

Communication Phase

for $q = 1$ *to* s *do in parallel*
  $\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
    $\boldsymbol{t}_{(2q-1)k-1}$, $\boldsymbol{t}_{(2q-1)k}$
enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*
  $\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
    $\boldsymbol{t}_{2qk}$, $\boldsymbol{t}_{2qk+1}$
enddo

```
Computation Phase
for    j = 1  to  P  do in parallel
  if   j  odd    then
    for   l = (j − 1)k + 1  to  jk − 1  do
      P_j computes t_{2p+l}
    enddo
  else
    for   l = (j − 1)k  to  jk  do
      P_j computes t_{2p+l}
    enddo
  endif
enddo
```

Αλγόριθμος επίλυσης **άνω** *τριγωνικού συστήματος* $M_2\boldsymbol{t} = \boldsymbol{z}$

Black Cycle για άρτιο k

```
Computation Phase
for    j = 1  to  P  do in parallel
  for   l = (j − 1)k + 1  to  jk  do
    P_j computes t_{2p+l}
  enddo
enddo
```

Red Cycle για άρτιο k

Communication Phase

for $q = 1$ *to* s *do in parallel*
$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors
$\boldsymbol{t}_{2p+(2q-1)k-1}$, $\boldsymbol{t}_{2p+(2q-1)k}$
enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*
$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors
$\boldsymbol{t}_{2p+2qk-1}$, $\boldsymbol{t}_{2p+2qk}$
enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
for $l = (j-1)k$ *to* $jk-1$ *do*
$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
enddo
enddo

Black Cycle για περιττό k

Computation Phase

for $j = 1$ *to* P *do in parallel*
if j *odd* *then*
for $l = (j-1)k+1$ *to* $jk-1$ *do*
$\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
enddo
else
for $l = (j-1)k$ *to* jk *do*
$\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
enddo
endif
enddo

Red Cycle για περιττό k

Communication Phase

for $q = 1$ *to* s *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors
    $\boldsymbol{t}_{2p+(2q-1)k}$, $\boldsymbol{t}_{2p+(2q-1)k+1}$
enddo

for $q = 1$ *to* $\hat{s}$ *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors
    $\boldsymbol{t}_{2p+2qk-1}$, $\boldsymbol{t}_{2p+2qk}$
enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if j *odd* *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  else
    for $l = (j-1)k+1$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  endif
enddo

όπου οι παραπάνω ακέραιοι s και $\hat{s}$ ορίζονται ως εξής

$$s = \begin{cases} \frac{P-1}{2} & , \text{ αν } P \text{ περιττός} \\ \frac{P}{2} & , \text{ αν } P \text{ άρτιος} \end{cases} , \quad \hat{s} = \begin{cases} \frac{P-1}{2} & , \text{ αν } P \text{ περιττός} \\ \frac{P-2}{2} & , \text{ αν } P \text{ άρτιος} \end{cases} . \qquad (138)$$

Στον παραπάνω αλγόριθμο είναι προφανές ότι το κόστος επικοινωνίας δεν μεταβλήθηκε κατά την μετάβαση στο παράλληλο σύστημα σταθερής διάστασης από το εικονικό. Αντίθετα το υπολογιστικό κόστος από $O(n_s)$ διαφοροποιήθηκε σε $O(kn_s) = O(\frac{n_s^2}{P})$ κι αυτό μπορεί να δικαιολογηθεί από τις σειριακές ανακυκλώσεις της τάξεως k, οι οποίες εμφανίστηκαν κατά την μετάβαση στο σύστημα της σταθερής διάστασης,

όπως είχε συμβεί και στην ανάλογη περίπτωση της SOR.

Περίπτωση όπου $n_s = kP + v \ , \ 1 \leq v \leq P-1$

Στην περίπτωση αυτή ο αριθμός των υποδιαστημάτων της διακριτοποίησης δεν είναι ακέραιο πολλαπλάσιο του αριθμού των επεξεργαστών. Άρα γίνεται απεικόνιση των k διαδοχικών εικονικών επεξεργαστών σε καθένα από τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ και $k+1$ εικονικοί αντιστοιχίζονται στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$. Αυτό έχει ως αποτέλεσμα να χαθεί η ισοκατανομή του υπολογιστικού φορτίου στους επεξεργαστές κι έτσι αν και ο παράλληλος αλγόριθμος γίνεται περισσότερο πολύπλοκος, είναι δυνατή η αποφυγή αυξημένης πρόσθετης επιβάρυνσης σε υπολογιστικό και επικοινωνιακό κόστος.

Ειδικότερα για τους πρώτους $P-v-1$ επεξεργαστές $\mathcal{P}_j \ \ (j = 1, \ldots, P-v-1)$ ισχύουν τα ίδια με την περίπτωση όπου $n_s = kP$. Για στους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ απεικονίζουμε τους $V_{(j-1)k+j-P+v+1}, \ldots, V_{jk+j-P+v+1}$ εικονικούς επεξεργαστές. Έτσι έχουμε να παρατηρήσουμε τα παρακάτω

- Όταν ο δείκτης k είναι *άρτιος*, τότε και οι δυο δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα είναι

  περιττοί όταν $j-P+v$ είναι *άρτιος*

  ενώ είναι

  άρτιοι όταν $j-P+v$ είναι *περιττός* .

  Έτσι, όταν το $j-P+v$ είναι *άρτιος* , τότε και οι δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *red (περιττοί)* κι έτσι στον επεξεργαστή $\mathcal{P}_j \ \ (j = P-v, \ldots, P)$ θα αντιστοιχούν $k+2$ *red* διανύσματα

  $$\boldsymbol{x}_l \ , \ l = (j-1)k+j-P+v, \ldots, jk+j-P+v+1$$

και k *black* διανύσματα

$$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v.$$

Ομοίως, όταν το $j-P+v$ είναι *περιττός*, τότε και δυο επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι *black (άρτιοι)* γι'αυτό στον επεξεργαστή $\mathcal{P}_j\ (j=P-v,\ldots,P)$ θα αντιστοιχούν k *red* διανύσματα

$$\boldsymbol{x}_{l}\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v$$

και $k+2$ *black* διανύσματα

$$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v+1.$$

- Όταν το k είναι *περιττός* για τους δείκτες $(j-1)k+j-P+v+1$ και $jk+j-P+v+1$ θα ισχύει

  $(j-1)k+j-P+v+1$ θα είναι *άρτιος* ενώ $jk+j-P+v+1$ είναι *περιττός*.

  Άρα οι εικονικοί επεξεργαστές $V_{(j-1)k+j-P+v+1}$ και $V_{jk+j-P+v+1}$ θα είναι αντίστοιχα *black (άρτιος)* και *red (περιττός)*. Έτσι σε καθένα επεξεργαστή $\mathcal{P}_j\ (j=P-v,\ldots,P)$ αντιστοιχούμε $k+1$ *red* διανύσματα

  $$\boldsymbol{x}_{l}\ ,\ l=(j-1)k+j-P+v+1,\ldots,jk+j-P+v+1$$

  και $k+1$ *black* διανύσματα

  $$\boldsymbol{x}_{2p+l}\ ,\ l=(j-1)k+j-P+v,\ldots,jk+j-P+v.$$

Έτσι προκύπτει ότι για k *άρτιο* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j\ (j=1,\ldots,P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\,\cdots\,RB$, ενώ οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j\ (j=P-v,\ldots,P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\,\cdots\,RB\ R$, όταν το j είναι *περιττό* και την

$BR\,BR\cdots BR\,B$,όταν το j είναι *άρτιο*. Αυτό σημαίνει, ότι κατά τη διάρκεια των διαδικασιών Red και Black Cycle της επίλυσης των άνω και κάτω τριγωνικών συστημάτων στη τεχνική του preconditioning, ο τρόπος επικοινωνίας μεταξύ των πρώτων $P-v-1$ επεξεργαστών $\mathcal{P}_j\ \ (j=1,\ldots,P-v-1)$ θα είναι ίδιος με αυτόν της περίπτωσης $k=$ άρτιος για $n_s=kP$, ενώ για τους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j\ \ (j=P-v,\ldots,P)$ θα είναι ίδιος με αυτόν της περίπτωσης $k=$ περιττός. Επίσης αφού το n_s και το k είναι *άρτιοι* θα είναι και το v *άρτιος*, δηλαδή το $P-v$ *περιττός (αντ. άρτιος)*, όταν το P είναι *περιττός (αντ. άρτιος)*.

Στη συνέχεια εμφανίζονται οι δυο αλγόριθμοι επίλυσης των τριγωνικών συστημάτων για *άρτιο* k.

Αλγόριθμος επίλυσης **κάτω** *τριγωνικού συστήματος* $M_1\boldsymbol{t}=\boldsymbol{z}$

Red Cycle

Computation Phase
for $j=1$ *to* P *do in parallel*
  if $1\le j\le P-v-1$ *then*
    for $l=(j-1)k$ *to* $jk-1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  elseif $j-P+v$ even *then*
    for $l=(j-1)k+j-P+v$ *to* $jk+j-P+v+1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{\,l}$
    enddo
  else
    for $l=(j-1)k+j-P+v+1$ *to* $jk+j-P+v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{\,l}$
    enddo
  endif
enddo

Black Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{t}_{(2q-1)k}$, $\boldsymbol{t}_{(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{(2\hat{q}-1)(k+1)-P+v}$, $\boldsymbol{t}_{(2\hat{q}-1)(k+1)-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2qk}$, $\boldsymbol{t}_{2qk+1}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{t}_{2(k+1)\hat{q}-P+v+2}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{t}_{(2q-1)k}$, $\boldsymbol{t}_{(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{t}_{(2\hat{q}-1)k+2\hat{q}-P+v}$, $\boldsymbol{t}_{(2\hat{q}-1)k+2\hat{q}-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2qk}$, $\boldsymbol{t}_{2qk+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{t}_{2(k+1)\hat{q}-P+v}$, $\boldsymbol{t}_{2(k+1)\hat{q}-P+v+1}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k + 1$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  else
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  endif
enddo

Αλγόριθμος επίλυσης **άνω** *τριγωνικού συστήματος* $M_2\boldsymbol{t} = \boldsymbol{z}$

Black Cycle

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $1 \leq j \leq P - v - 1$ *then*
    for $l = (j-1)k + 1$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  elseif $j - P + v$ even *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  else
    for $l = (j-1)k + j - P + v$ *to* $jk + j - P + v + 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  endif
enddo

Red Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2p+(2q-1)k-1}$, $\boldsymbol{t}_{2p+(2q-1)k}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{t}_{2p+2\hat{q}(k+1)-k-P+v}$, $\boldsymbol{t}_{2p+2\hat{q}(k+1)-k-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-1}{2}$ *and for* $\hat{q} = \frac{P-v+1}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{t}_{2p+2qk-1}$, $\boldsymbol{t}_{2p+2qk}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v}$, $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+1}$

enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and* *for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2p+(2q-1)k-1}$, $\boldsymbol{t}_{2p+(2q-1)k}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{2p+(2\hat{q}-1)(k+1)-P+v}$, $\boldsymbol{t}_{2p+(2\hat{q}-1)(k+1)-P+v+1}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and* *for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{t}^{(m)}_{2p+2qk-1}$, $\boldsymbol{x}_{2p+2qk}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+2}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

if $1 \leq j \leq P-v-1$ *then*

for $l = (j-1)k$ *to* $jk-1$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$

enddo

elseif $j-P+v$ even *then*

for $l = (j-1)k+j-P+v$ *to* $jk+j-P+v+1$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$

enddo

else

for $l = (j-1)k+j-P+v+1$ *to* $jk+j-P+v$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_{\ l}$

enddo

endif

enddo

Όταν τώρα το k είναι *περιττό* οι πρώτοι $P-v-1$ επεξεργαστές $\mathcal{P}_j$ $(j=1,\ldots,P-v-1)$ ακολουθούν την αντιστοίχιση $RB\,RB\,\cdots\,RB\,R$ για j περιττό, ενώ για j άρτιο την $B\,RB\,\cdots\,RB$. Έτσι οι τελευταίοι $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ με αντιστοίχιση $k+1$ εικονικών επεξεργαστών θα ακολουθούν τη διάταξη $RB\,RB\,\cdots\,RB\;\;RB$. Αυτό σημαίνει, ότι κατά τη διάρκεια των διαδικασιών Red και Black Cycle της επίλυσης των άνω και κάτω τριγωνικών συστημάτων κατά την τεχνική του preconditioning, ο τρόπος επικοινωνίας μεταξύ των πρώτων $P-v-1$ επεξεργαστών $\mathcal{P}_j$ $(j=1,\ldots,P-v-1)$ θα είναι ίδιος με αυτόν της περίπτωσης $k=$ περιττός για $n_s=kP$, ενώ για τους υπόλοιπους $v+1$ επεξεργαστές $\mathcal{P}_j$ $(j=P-v,\ldots,P)$ θα είναι ίδιος με αυτόν της περίπτωσης $k=$ άρτιος. Επίσης, αφού το n_s είναι *άρτιος* και το k είναι *περιττό*, θα είναι το v *περιττός* για P *περιττός* και το v *άρτιος* για P *άρτιο*. Οπότε το $P-v-1$ είναι *περιττός* για κάθε περίπτωση.

Ακολουθούν οι αλγόριθμοι της επίλυσης των άνω και κάτω τριγωνικών συστημάτων κατά τη διαδικασία του SSOR preconditioning για *περιττό* k.

Αλγόριθμος επίλυσης **κάτω** *τριγωνικού συστήματος* $M_1\boldsymbol{t} = \boldsymbol{z}$

Red Cycle

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  elseif j odd *then*
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  else
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$
    enddo
  endif
enddo

Black Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*
  $\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{(2q-1)k}$, $\boldsymbol{t}_{(2q-1)k-1}$
  $\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{(2\hat{q}-1)(k+1)-P+v}$, $\boldsymbol{t}_{(2\hat{q}-1)(k+1)-P+v+1}$
enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*
  $\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2qk+2}$, $\boldsymbol{t}_{2qk+1}$
  $\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{t}_{2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{t}_{2(k+1)\hat{q}-P+v}$
enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q-1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{(2q-1)k}$, $\boldsymbol{t}_{(2q-1)k-1}$

$\mathcal{P}_{2\hat{q}-1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{(2\hat{q}-1)k+2\hat{q}-P+v-1}$, $\boldsymbol{t}_{(2\hat{q}-1)k+2\hat{q}-P+v-2}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q+1}$ *sends to* $\mathcal{P}_{2q}$ the vectors $\boldsymbol{t}_{2qk}$, $\boldsymbol{t}_{2qk+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}+1}$ the vectors $\boldsymbol{t}_{2(k+1)\hat{q}-P+v-1}$, $\boldsymbol{t}_{2(k+1)\hat{q}-P+v}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

  if $P - v \leq j \leq P$ *then*

    for $l = (j-1)k - P + v + 1$ *to* $jk + j - P + v$ *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$

    enddo

  elseif j odd *then*

    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$

    enddo

  else

    for $l = (j-1)k$ *to* jk *do*

      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$

    enddo

  endif

enddo

Αλγόριθμος επίλυσης **άνω** *τριγωνικού συστήματος* $M_2\boldsymbol{t} = \boldsymbol{z}$

Black Cycle

Computation Phase

for $j = 1$ *to* P *do in parallel*
  if $P - v \leq j \leq P$ *then*
    for $l = (j-1)k - P + v + 1$ *to* $jk + j - P + v$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  elseif j odd *then*
    for $l = (j-1)k + 1$ *to* $jk - 1$ *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  else
    for $l = (j-1)k$ *to* jk *do*
      $\mathcal{P}_j$ *computes* $\boldsymbol{t}_{2p+l}$
    enddo
  endif
enddo

Red Cycle

Communication Phase για περιττό P

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{t}_{2p+(2q-1)k}$, $\boldsymbol{t}_{2p+(2q-1)k+1}$
  $\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{t}_{2p+2\hat{q}(k+1)-k-P+v}$, $\boldsymbol{t}_{2p+2\hat{q}(k+1)-k-P+v+1}$
enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-1}{2}$ *do in parallel*
  $\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{t}_{2p+2qk-1}$, $\boldsymbol{t}_{2p+2qk}$
  $\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+1}$, $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+2}$
enddo

Communication Phase για άρτιο P

for $q = 1$ *to* $\frac{P-v}{2}$ *and for* $\hat{q} = \frac{P-v+2}{2}$ *to* $\frac{P}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q-1}$ the vectors $\boldsymbol{t}_{2p+(2q-1)k}$, $\boldsymbol{t}_{2p+(2q-1)k+1}$

$\mathcal{P}_{2\hat{q}}$ *sends to* $\mathcal{P}_{2\hat{q}-1}$ the vectors $\boldsymbol{t}_{2p+(2\hat{q}-1)(k+1)-P+v+1}$, $\boldsymbol{t}_{2p+(2\hat{q}-1)(k+1)-P+v+2}$

enddo

for $q = 1$ *to* $\frac{P-v-2}{2}$ *and for* $\hat{q} = \frac{P-v}{2}$ *to* $\frac{P-2}{2}$ *do in parallel*

$\mathcal{P}_{2q}$ *sends to* $\mathcal{P}_{2q+1}$ the vectors $\boldsymbol{t}_{2p+2qk-1}$, $\boldsymbol{t}_{2p+2qk-2}$

$\mathcal{P}_{2\hat{q}+1}$ *sends to* $\mathcal{P}_{2\hat{q}}$ the vectors $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+2}$, $\boldsymbol{t}_{2p+2(k+1)\hat{q}-P+v+3}$

enddo

Computation Phase

for $j = 1$ *to* P *do in parallel*

if $P - v \leq j \leq P$ *then*

for $l = (j-1)k + j - P + v + 1$ *to* $jk + j - P + v$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$

enddo

elseif j odd *then*

for $l = (j-1)k$ *to* jk *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$

enddo

else

for $l = (j-1)k + 1$ *to* $jk - 1$ *do*

$\mathcal{P}_j$ *computes* $\boldsymbol{t}_l$

enddo

endif

enddo

Μελετώντας την δομή των παραπάνω αλγορίθμων προκύπτει ότι το υπολογιστικό, αλλά και το επικοινωνιακό κόστος, παρέμεινε σταθερό με την περίπτωση $n_s = kP$, αν και ο αλγόριθμος στην περίπτωση $n_s = kP + v \;\;,\;\; 1 \leq v \leq P-1$ είναι σημαντικά πιο πολύπλοκος, γιατί δημιουργήθηκαν περισσότερες περιπτώσεις και έπρεπε να υπάρξει όσο τον δυνατόν ισοκατανομή του φορτίου για καθεμιά τους, ώστε να

εξασφαλιστεί η καλύτερη δυνατή απόδοση.

Επίσης ο τρόπος απεικόνισης των αγνώστων στους επεξεργαστές συντελεί, όμοια με την περίπτωση της αρχιτεκτονικής κοινής μνήμης, στην ελαχιστοποίηση των υπολογισμών.

4.4.4 Υλοποίηση της επαναληπτικής μεθόδου BiCGSTAB στο παράλληλο υπολογιστικό σύστημα Parsytec CC2

Έγινε υλοποίηση του παραλλήλου αλγορίθμου της επαναληπτικής μεθόδου SSOR preconditioned BiCGSTAB στο παράλληλο υπολογιστικό σύστημα διανεμημένης μνήμης στο οποίο έγινε και η αντίστοιχη υλοποίηση της SOR για το ίδιο ακριβώς πρόβλημα μοντέλο και για τις ίδιες πειραματικές μετρήσεις. Για κριτήριο σύγκλισης χρησιμοποιήθηκε το

$$\frac{M^{-1}\boldsymbol{r}^{(m)}}{M^{-1}\boldsymbol{b}} \leq tol$$

Η παράλληλη υλοποίηση του αλγορίθμου έγινε μέσω του προγραμματιστικού μοντέλου SPMD (Single Program Multiple Data), δηλαδή υπήρχε ένας κώδικας προγράμματος, όπου το κατάλληλο τμήμα του που αφορούσε τα συγκεκριμένα δεδομένα εκτελέστηκε στον ανάλογο επεξεργαστή. Οι περιπτώσεις που υλοποιήθηκαν είναι για $n_s = kP$, $k = \text{άρτιος}$, αφού η διαμέριση n_s επιλέχθηκε κάθε φορά ως δύναμη του 2.

Η σειριακή μορφή του red-black παράλληλου αλγορίθμου υλοποιήθηκε στον ένα από τους δυο επεξεργαστές. Τα πειραματικά αποτελέσματα εμφανίζονται στον παρακάτω πίνακα T30, όπου παρουσιάζεται ο συνολικός χρόνος εκτέλεσης της μεθόδου μετρημένος σε δευτερόλεπτα, καθώς και οι χρόνοι της διαδικασίας matvec που αφορά τον πολλαπλασιασμό διανύσματος με τον collocation πίνακα και της τεχνικής του precondition msolve. Ο πίνακας T31 παρουσιάζει τις ίδιες πειραματικές μετρήσεις για την περίπτωση της SGS preconditioned BiCGSTAB.

T30	Red-Black SSOR pr. BiCGSTAB		
ns	Matvec	Msolve	Total
4	4.43e-3	1.77e-2	2.34e-2
8	2.65e-2	1.04e-1	1.39e-1
16	1.69e-1	6.50e-1	8.55e-1
32	1.22e0	4.77e0	6.40e0
64	9.89e0	3.89e1	5.29e1

T31	Red-Black SGS pr. BiCGSTAB		
ns	Matvec	Msolve	Total
4	2.72e-3	9.86e-3	1.33e-2
8	2.12e-2	8.21e-2	1.10e-1
16	1.69e-1	6.50e-1	8.55e-1
32	1.22e0	4.77e0	6.40e0
64	1.05e1	4.19e1	5.71e1

Όπως προκύπτει από τις μετρήσεις το μεγαλύτερο υπολογιστικό κόστος έχει η τεχνική του preconditioning και ακολουθεί η πράξη του πολλαπλασιασμού με τον collocation πίνακα. Ο παρακάτω πίνακας T32 εμφανίζει τις μετρήσεις σε χρόνο εκτέλεσης του παράλληλου αλγορίθμου της SSOR preconditioned BiCGSTAB στους δυο διαθέσιμους επεξεργαστές. Παρουσιάζονται οι μετρήσεις για το συνολικό κόστος όλων των red και black διαδικασιών της τεχνικής του precondition, ενώ ξεχωριστά εμφανίζεται το υπολογιστικό και το επικοινωνιακό κόστος. Οι τρεις τελευταίες στήλες εμφανίζουν τις μετρήσεις του συνολικού υπολογιστικού και επικοινωνιακού κόστους της μεθόδου για κάθε διακριτοποίηση. Ο επόμενος πίνακας T33 εμφανίζει τις αντίστοιχες μετρήσεις για την SGS preconditioned BiCGSTAB.

T32	Parallel SSOR preconditioned BiCGSTAB							
ns	Red Solve		Black Solve					
ns	comp.	comm.	comp.	comm.	Total	Comp.	Comm.	Total
4	6.62e-3	2.75e-2	4.11e-3	5.15e-2	8.97e-2	8.01e-2	3.27e-3	1.17e-1
8	2.26e-2	5.02e-2	1.58e-2	8.07e-2	1.69e-1	8.60e-2	1.32e-1	2.18e-1
16	1.28e-1	1.33e-1	1.02e-1	2.09e-1	5.72e-1	3.89e-1	3.44e-1	7.33e-1
32	8.43e-1	5.23e-1	7.80e-1	8.12e-1	2.96e0	2.51e0	1.34e0	3.85e0
64	6.84e0	3.29e0	6.74e0	4.80e0	2.17e1	2.09e1	8.11e0	2.90e1

T33	Parallel SGS preconditioned BiCGSTAB							
ns	Red Solve		Black Solve					
ns	comp.	comm.	comp.	comm.	Total	Comp.	Comm.	Total
4	4.73e-3	1.76e-2	2.75e-3	3.28e-2	5.79e-2	2.41e-2	5.15e-2	7.56e-2
8	2.12e-2	4.76e-2	1.58e-2	8.01e-2	1.65e-1	8.80e-2	1.29e-1	2.17e-1
16	1.28e-1	1.33e-1	1.02e-1	2.09e-1	5.72e-1	3.89e-1	3.44e-1	7.33e-1
32	8.43e-1	5.23e-1	7.80e-1	8.12e-1	2.96e0	2.51e0	1.34e0	3.85e0
64	7.26e0	3.54e0	7.11e0	5.20e0	2.31e1	2.21e1	8.75e0	3.09e1

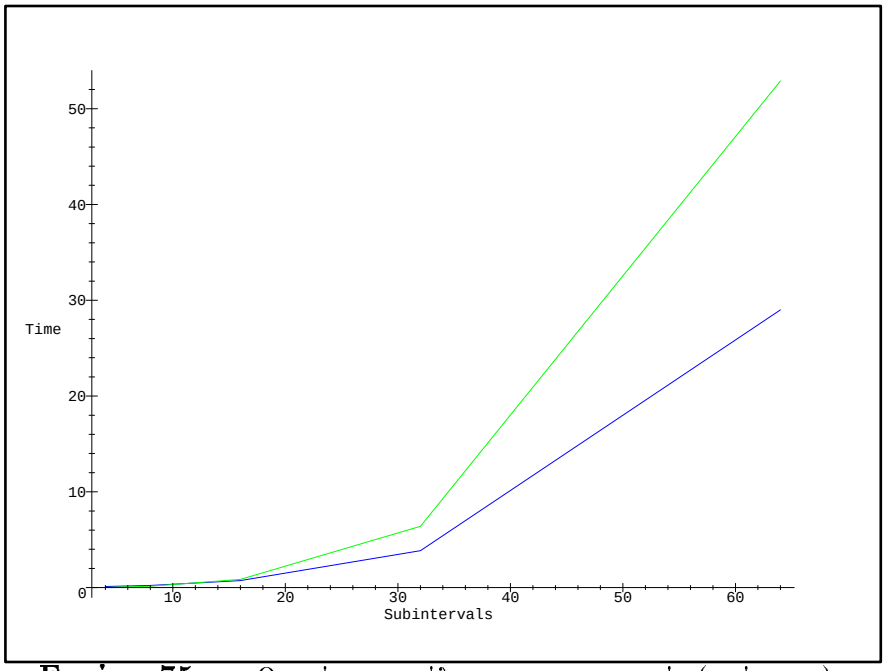
Παρατηρούμε ότι για διακριτοποιήσεις μικρότερες των $n_s = 36$ υποδιαστημάτων το επικοινωνιακό κόστος είναι μεγαλύτερο από το υπολογιστικό για την διαδικασία του preconditioning, ενώ αυτό ισχύει συνολικά στη μέθοδο για μικρότερες τιμές από $n_s = 16$. Το υπολογιστικό κόστος της Red διαδικασίας κατά την τεχνική του preconditioning είναι ελαφρά μειωμένο εξαιτείας του μειωμένου χρόνου υπολογισμού του πρώτου και τελευταίου τμήματος της Red λύσης σε σχέση με τα αντίστοιχα της Black λύσης. Στο επικοινωνικό κόστος έχει προστεθεί εκτός από το χρόνο επικοινωνίας των επεξεργαστών και ο χρόνος συγχρονισμού μεταξύ τους. Γι' αυτό στην Black διαδικασία εμφανίζεται λίγο αυξημένο από της Red. Στον υπολογιστικό χρόνο της Solve έχει προστεθεί και ο ανάλογος χρόνος της πράξης του διανύσματος με το block διαγώνιο τμήμα του collocation πίνακα.

Τα παραπάνω αποτελέσματα ωστόσο είναι ενδεικτικά της συμπεριφοράς του παράλληλου αλγορίθμου, γιατί δεν υπήρχαν αρκετοί επεξεργαστές διαθέσιμοι, ώστε να φανεί η πραγματική διάσταση του επικοινωνιακού κόστους. Έτσι χρησιμοποιώντας ξανά ως ένδειξη απόδοσης του παράλληλου αλγορίθμου την τιμή του speedup, όπως αυτή ορίστηκε και στις μετρήσεις του αλγορίθμου στο υπολογιστικό σύστημα της κοινής μνήμης της προηγούμενης ενότητας, μπορούμε να περιμένουμε αποδόσεις πολύ κοντά στη θεωρητική βέλτιστη τιμή 2, δηλαδή τον αριθμό των επεξεργαστών. Ο παρακάτω πίνακας T34 παρουσιάζει τις τιμές του speedup για κάθε τιμή της διακριτοποίησης του προβλήματος της SSOR preconditioned BiCGSTAB. Αντίστοιχες μετρήσεις για την SGS preconditioned BiCGSTAB παρουσιάζονται στον πίνακα T35.

T34	Time for SSOR pr. BiCGSTAB		
n_s	Sequential	Parallel	$speedup_{rb}$
4	2.34e-2	1.17e-1	-
8	1.39e-1	2.18e-1	-
16	8.55e-1	7.33e-1	1.166
32	6.40e0	3.85e0	1.662
64	5.29e1	2.90e1	1.824

T35	Time for SGS pr. BiCGSTAB		
n_s	Sequential	Parallel	$speedup_{rb}$
4	1.33e-2	7.56e-2	-
8	1.10e-1	2.17e-1	-
16	8.55e-1	7.33e-1	1.166
32	6.40e0	3.85e0	1.662
64	5.71e1	3.09e1	1.848

Στην Εικόνα 75 εμφανίζεται η γραφική παράσταση του συνολικού χρόνου εκτέλεσης κάθε αλγορίθμου ως προς τη διακριτοποίηση με SSOR preconditioning.



Εικόνα 75 : Ο χρόνος εκτέλεσης της σειριακής (πράσσινο) και της παράλληλης (μπλε) SSOR preconditioned BiCGSTAB ως προς τη διακριτοποίηση.

Με πράσσινο χρώμα εμφανίζεται το γράφημα για την σειριακή υλοποίηση του red-black αλγορίθμου και με μπλε η παράλληλη υλοποίηση του. Είναι φανερό ότι ο χρόνος του παράλληλου αλγορίθμου για διακριτοποιήσεις με $n_s \geq 64$ μειώνεται περίπου στο μισό κατά την παράλληλη υλοποίηση.

5 ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα επιστημονικά αποτελέσματα που προέκυψαν από την παρούσα διατριβή θα μπορούσαν να χωριστούν σε δυο βασικές κατηγορίες. Στην πρώτη κατηγορία έχουμε αυτά που προκύπτουν από τη σειριακή μελέτη της συμπεριφοράς των non-stationary ή Krylov subspace μεθόδων κατά την επίλυση των γενικών γραμμικών συστημάτων μεγάλης τάξης. Τα γραμμικά αυτά συστήματα προκύπτουν από τη διακριτοποίηση του προβλήματος μοντέλου Poisson, το οποίο χρησιμοποιήσαμε, με την αριθμητική μέθοδο επίλυσης διαφορικών εξισώσεων collocation βασισμένη στα Hermite bi-cybic πεπερασμένα στοιχεία. Ως αρίθμηση αγνώστων και εξισώσεων χρησιμοποιήθηκε αυτή που παράγει τον collocation πίνακα σε block τριδιαγώνια μορφή και η διαμερισή του βασίστηκε στην πρόσφατα παρουσιαζόμενη ως αποδοτικότερη μεταξύ άλλων για την SOR και GMRES. Χρησιμοποιήθηκαν οι δημοφιλέστερες Krylov subspace επαναληπτικές μέθοδοι GMRES(m), BiCGSTAB και ORTHOMIN, για τις οποίες παράλληλα δοκιμάστηκαν τρεις τεχνικές preconditioning, οι οποίες βασίστηκαν στους πίνακες διάσπασης των SSOR, SGS και Jacobi, και ήταν άμεσα διαθέσιμοι χωρίς επιπλέον κόστος κατασκευής. Οι πειραματικές μετρήσεις περιλάμβαναν δυο μορφές αποθήκευσης του πίνακα συντελεστών και του preconditioner, αυτή της block και της αραιάς αποθήκευσης κατά στήλες. Αυτό συνέβει, γιατί ο παραγόμενος collocation

πίνακας και κατά συνέπεια και ο preconditioner, είναι αραιοί, block και μεγάλης διάστασης. Έτσι τα αποτελέσματα έδειξαν ότι :

- κατά την μέθοδο της αραιάς αποθήκευσης και πιο συγκεκριμένα, η επίδραση του σφάλματος από τη μέθοδο της ατελούς παραγοντοποίησης του preconditioner πίνακα, είναι ικανή, ώστε οι επαναληπτικές μέθοδοι με Jacobi preconditioning να μην επιτυγχάνουν σύγκλιση. Η ανάλυση των τιμών Ritz έδειξε και τα πειραματικά αποτελέσματα επαλήθευσαν, ότι για αυτή την περίπτωση είναι προτιμότερο να χρησιμοποιηθεί η τεχνική του SSOR ή και του SGS preconditioning με την BiCGSTAB επαναληπτική μέθοδο.

- όσο αφορά την περίπτωση της block αποθήκευσης των πινάκων, πάλι αποδοτικότερη προέκυψε η SSOR preconditioned BiCGSTAB επαναληπτική μέθοδος, σύμφωνα με τις πειραματικές δοκιμές, οι οποίες επαληθεύτηκαν και με την ανάλυση των τιμών Ritz.

- η σύγκριση των μεθόδων έδειξε ως καλύτερη επιλογή την SOR για διακριτοποιήσεις με μικρό αριθμό υποδιαστημάτων και την SSOR preconditioned BiCGSTAB για αυτές με μεγάλο, έχοντας πάντα σαν δεδομένο ότι είναι γνωστή η τιμή της βέλτιστης τιμής της παραμέτρου ω της SOR.

Στο άλλο τμήμα της παρούσας διατριβής έχοντας πλέον ως δεδομένες τις βέλτιστες μεθόδους optimal SOR από την κατηγορία των stationary επαναληπτικών μεθόδων και SSOR preconditioned BiCGSTAB από αυτή των non-stationary, έγινε προσπάθεια για την κατασκευή αποδοτικών αλγορίθμων τους για παράλληλες αρχιτεκτονικές κοινής και διανεμημένης μνήμης. Έτσι χρειάστηκε να γίνει επαναρίθμηση αγνώστων και εξισώσεων σύμφωνα με την μέθοδο red-black, δηλαδή ουσιαστικά να εφαρμοστεί ένας μετασχηματισμός ομοιότητας στο γραμμικό σύστημα. Αυτό είχε σαν αποτέλεσμα την κατασκευή του collocation πίνακα στην 2-cyclic κανονική του μορφή, ώστε να είναι εφικτός ο υπολογισμός των αγνώστων σε παράλληλες διαδικασίες. Επιπλέον

χρησιμοποιήθηκε ένας ακόμα μετασχηματισμός με τον οποίο διπλασιάστηκε ο παραλληλισμός των μεθόδων. Έτσι έγινε εφικτή η κατασκευή δυο παράλληλων αλγορίθμων για κάθε μέθοδο, για παράλληλα περιβάλλοντα κοινής και διανεμημένης μνήμης. Αυτοί οι αλγόριθμοι υλοποιήθηκαν με επιτυχία σε δυο παράλληλα υπολογιστικά συστήματα και όπως έδειξαν οι πειραματικές μετρήσεις ήταν αποδοτικοί επειδή :

- οι δυο μετασχηματισμοί που εφαρμόστηκαν δεν επηρέασαν την συμπεριφορά της BiCGSTAB κι έτσι εξακολουθεί να θεωρείται η αποδοτικότερη των non-stationary μεθόδων και για το red-black collocation γραμμικό σύστημα, με καλύτερη τεχνική preconditioning, από αυτές που δοκιμάστηκαν, την SSOR. Το αποτέλεσμα αυτό προέκυψε από την ανάλυση των τιμών Ritz. Όσο αφορά την SOR, όπως αποδείχτηκε θεωρητικά δεν αλλάζει η συμπεριφορά σύγκλισής της σε σχέση με την περίπτωση της block τριδιαγώνιας μορφής του collocation γραμμικού συστήματος. Το ίδιο ισχύει και για τις βέλτιστες τιμές της παραμέτρου της ω.

- η προτινόμενη απεικόνιση των αγνώστων στους επεξεργαστές συντελεί στην ελαχιστοποίηση των υπολογισμών, εξαιτείας της απεικόνισης όλων των αγνώστων που αντιστοιχούν σε κάθε στήλη του πλέγματος διακριτοποίησης, στον ίδιο επεξεργαστή. Αυτό ισχύει για τους αλγορίθμους για αρχιτεκτονικές κοινής και διανεμημένης μνήμης και για τις δυο επαναληπτικές μεθόδους.

- ο υπολογισμός από τον ίδιο επεξεργαστή της ομάδων red και black αγνώστων, οι οποίοι αντιστοιχούν σε γειτονικές στήλες του πλέγματος διακριτοποίησης, μειώνει το επικοινωνιακό κόστος στην περίπτωση των αλγορίθμων για αρχιτεκτονικές διανεμημένης μνήμης.

ΠΑΡΑΡΤΗΜΑ

Βιβλιογραφία

References

[1] Γ.Δ. Ακρίβης, Β.Α.Δουγαλής. Εισαγωγή στην Αριθμητική Ανάλυση - Πανεπιστημιακές Εκδόσεις Κρήτης, 1997.

[2] L. Adams and H. Jordan, " Is SOR color-blind?",*SIAM J. Sci. Statist. Comput.,* 7 (1986), pp. 490-506.

[3] E. Anderson, Z. Bai, C. Bischof, J .Demmel, J .Dongarra, J .Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov , and D. Sorensen. LAPACK Users' Guide, Second Edition. SIAM, Philadelphia, PA, 1995.

[4] W.E. Arnoldi. " The principle of minimized iteration in the solution of the matrix eigenproblem ". *Quart. Appl. Math.*, 9:17-29,1951.

[5] S. F. Ashby , T .A. Manteuffel, and P .E. Saylor, " A taxonomy for conjugate gradient methods ", *SIAM J. Numer. Anal.,* 27 (1990), pp. 1542-1568.

[6] C. Ashcraft and R. Grimes, " On vectorizing incomplete factorizations and SSOR preconditioners", *SIAM J. Sci. Statist. Comput.,* 9 (1988), pp. 122-15 1.

[7] 0. Axelsson, "Incomplete block matrix factorization preconditioning methods. The ultimate answer? ", *J. Comput. Appl. Math.,* 12 and 13 (1985), pp. 3-18.

[8] 0. Axelsson, "A general incomplete block-matrix factorization method",*Linear Algebra Appl.,* 74 (1986), pp. 179-190.

[9] 0. Axelsson and A. Barker, "Finite element solution of boundary value problems". Theory and computation, Academic Press, Orlando, Fl., 1984.

[10] 0. Axelsson and P. S. Vassilevski, " A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning". *J. Matrix Analysis and Applications,* 12(4):625-644, 1991.

[11] R.E. Bank and T.F. Chan. " An analysis of the composite step biconjugate gradient method ", *Numerische Mathematik,* 66:295-319, 1993.

[12] R. E. Bank and T.F. Chan, "A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems", *Numer. Alg.,* (1994), pp. 1-16.

[13] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst. " Templates for the Solution of Linear Systems Building Blocks for Iterative Methods ", SIAM, Philadelphia 1993.

[14] B.P. Bertsekas and J.N. Tsitsiklis, "Parallel and Distributed Computation", *Prentice Hall*, New Jersey, 1989

[15] G. Birkiff, M. H. Schultz and R. S. Varga, "Piecewise Hermite in one and two Variables with Applications to Partial Differential Equations", *Numer. Math.*, **11**, 232-256 (1968).

[16] C. de Boor and Swartz, "Collocation at Gaussian Points", *Siam J Numer. Anal.* **10**, 582-606 (1973).

[17] C. Brezinski and H. Sadok, "Avoiding breakdown in the CGS algorithm", *Numer. Alg.,* 1 (1991), pp. 199-206.

[18] C. Brezinski, M. Zaglia, and H. Sadok, "Avoiding breakdown and near breakdown in Lanczos type algorithms", *Numer. Alg.,* 1 (1991), pp. 261-284.

[19] C. Brezinski, M. Zaglia, and H. Sadok," A breakdown free Lanczos type algorithm for solving linear systems", *Numer. Math.,* 63 (1992), pp. 29-38.

[20] P. N .Brown," A theoretical comparison of the Arnoldi and GMRES algorithms", *SIAM J. Sci. Statist. Comput.,* 20 (1991), pp. 58-78.

[21] A. M. Bruaset. A Survey of Preconditioned Iterative Methods. Longman Scientific and Technical, Harlow , UK, 1995.

[22] G. Brussino and V. Sonnad, " A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations ", *Int. J. Numerical Methods in Engineering*, 28:801-815,1989.

[23] T. Chan, L. De Pillis, and H. van der Vorst, "A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems", Tech. Rep. CAM 91-17, UCLA, Dept. of Math., Los Angeles, CA 90024-1555, 1991.

[24] T. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. Tong, "A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems", *SIAM J. Sci. Comp.,* 15(2) (1994), pp. 338-347.

[25] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan and J. McDonald, "Parallel Programming in OpenMP', Morgan Kaufmann Publishers, ISBN 1-55860-671-8, 2000.

[26] P .Concus, G. Golub, and G. Meurant , " Block preconditioning for the conjugate gradient method", *SIAM J. Sci. Statist. Comput.,* 6 (1985), pp. 220-25 2.

[27] P. Concus, G. Golub, and D. O'Leary, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations", in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 309-332.

[28] . D. F. Cosgrove, J. C. Diaz, and A. Griewank. "Approximate inverse preconditionings for sparse linear systems". *Int J. Computer Mathematics,* 44:91-110, 1992.

[29] J. Cullum and A. Greenbaum, " Relations between Galerkin and norm-minimizing iterative methods for solving linear systems ", *SIAM J. Matrix Analysis and Applications*, 17:223-247, 1996.

[30] J.Dongarra, G.A.Geist, R.Manchek, V.S.Sunderam. Integrated PVM Framework Supports Heterogeneous Network Computing. Oak Ridge National Laboratory - Univ. of Tennessee, 1993.

[31] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst. " Numerical Linear Algebra for High-Performance Computers ", SIAM, Philadelphia 1998.

[32] J. Douglas and T. Dupont, *Collocation Methods for Parabolic Equations in a Single Space Variable*, Springer-Verlag Lecture Notes **385**, Berlin/New York,1974.

[33] D. Dubois, A. Greenbaum and G. Rodrigue," Approximating the inverse of a matrix for use in iterative algorithms on vector processors", *Computing*, 22:257-268, 1979.

[34] M. Eiermann and R. Varga, " Is the optimal ω best for the SOR iteration method ?", *Linear Algebra Appl.,* 182:257-277, 1993.

[35] V. Faber, W. Joubert, M. Knill, and T. Manteuffel," Minimal residual method stronger than polynomial preconditioning",*SIAM J. Matrix Anal. Appl.,* 17 (1996), pp. 707-729.

[36] V. Faber, T.A. Manteuffel. " Necessary and sufficient conditions for the existence of a conjugate gradient method ", *SIAM J. Numerical Analysis,* 21(2):352-362, 1984.

[37] R. W .Freund, "A transpose-free quasi-minimal residual algorithm for non- Hermitian linear systems",*SIAM J. Sci. Comput.,* 14 (1993), pp. 470-482.

[38] R. W. Freund and N. M. Nachtigal," QMR: A quasi-minimal residual method for non-Hermitian linear systems", *Numer. Math.,* 60 (1991), pp. 315-339.

[39] R. W. Freund and N. M. Nachtigal," An implementation of the QMR method based on coupled two-term recurrences",*SIAM J. Scientific Computing,* 15,pp 313-337, 1994.

[40] R. W. Freund, G. H. Golub, and N .M. Nachtigal, "Iterative solution of linear systems", *Acta Numerica*, (1992), pp. 57-100.

[41] R.W. Freund, M.H. Gutknecht and N.M. Nachtigal, " An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices ", *SIAM J. Scientific Computing,* 14:137-158, 1993.

[42] R.W. Freundand N.M. Nachtigal, " An implementati on of the look-ahead Lanczos algorithm for non-Hermitian matrices, part 2 ", Tech. Report 90.46, RIACS, NASA Ames Research Center, 1990.

[43] R.W. Freundand N.M. Nachtigal, " A quasi-minimal residual method for non-hermitian linear systems ", *Numerische Mathematik*, 60:315-339,1991.

[44] R.W. Freundand, " A transpose - free quasi-minimal residual algorithm for non-hermitian linear systems ", *SIAM J. Scientific Computing,* 14:470-482, 1993.

[45] A.Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Manchek and V.Sunderam. PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Network Parallel Computing. MIT 1995.

[46] A.Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Manchek and V.Sunderam. PVM 3 User's Guide and Reference Manual. Oak Ridge National Laboratory - Univ. of Tennessee, 1994.

[47] G. Golub and D. O'Leary, "Some history of the conjugate gradient and Lanczos methods", *SIAM Rev.,* 31 (1989), pp. 50-102.

[48] G. H. Golub and M. L. Overton, "The converyence of inexact Chebyshev and Richardson iterative methods for solving linear systems",*Numer. Math.,* 53 (1988), pp. 571-593.

[49] G.H. Golub and C.F. Van Loan. " Matrix Computations ". Third Edition. The Johns Hopkins University Press, Baltimore, 1996.

[50] A. Greenbaun. " Iterative methods for Solving Linear Systems ", SIAM - Frontiers in Applied Mathematics, Philadelphia 1997.

[51] A. Greenbaum, V. Ptak, and Z. Strakos," Any nonincreasing convergence curve is possible for GMRES", *SIAM J. Matrix Anal. Appl.,* 17 (1996), pp. 465-469.

[52] A. Greenbaum and Z. Strakos, "Matrices that generate the same Krylov residual spaces", in *Recent Advances in Iterative Methods,* G. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, Berlin, New York, 1994, pp. 95-118.

[53] M. J. Grote and T. Huckle. "Parallel preconditionings with sparse approximate inverses". *S J. Scientific Computing,* 18:838-853, 1997.

[54] I. Gustafsson and G. Lindskog. " A preconditioning technique based on element matrix factorizations". *Comput. Methods ppl. Mech. Eng.*, 55:201-220, 1986.

[55] M.H. Gutknecht, "A completed theory of the unsymmetric Lanczos process and related algorithms, part I", *SIAM J. Matrix Anal. Appl.,* 13 (1992), pp. 594-639.

[56] M.H. Gutknecht, " Variants of BICGSTAB for matrices with complex spectrum", *SIAM J. Scientific Computing*, 14:1020-1033, 1993.

[57] M.H. Gutknecht, "A completed theory of the unsymmetric Lanczos process and related algorithms, part II", *SIAM J. Matrix Anal. Appl.,* 15 (1994), pp. 15-58.

[58] A. Hadjidimos, T.S. Papatheodorou and Y. G. Saridakis, "Optimal Block Iterative Schemes for Certain Large Sparse and Non-symmetric Linear Systems", *Linear Algebra Appl.*, **110**, 285-318 (1988).

[59] L. A. Hageman and D. M. Young, " Applied Iterative Methods", *Academic Press*, New York, 1981

[60] M.R. Hestenes and E. Stiefel. " Methods of conjugate gradients for solving linear systems, *J. Res. Bur. Stand.*, 49:409-436, 1954.

[61] E. N. Houstis, "Application of the method of Collocation on Lines for Solving Nonlinear Hyperbolic Problems",*Math. Comp.*, **31**, 443 - 456 (1977).

[62] E. N. Houstis, W. Mitchell, J. R. Rice, "Collocation Software for Second Order Elliptic Partial Differential Equations", *ACM Trans. Math. Software*, **11**, 379-412 (1985).

[63] E. N. Houstis, R. E. Lynch, T. S. Papatheodorou and J. R. Rice, "Evaluation of Numerical Methods for Elliptic Partial Differential Equations", *J Comp. Phys.*, **27**, 323 - 350 (1978).

[64] Y. Huang, H.A. van der Vorst. " Some observations on the convergence behaviour of GMRES ", Technical Report 89-09, Delft University of Technology , Faculty of Tech. Math., 1989.

[65] 0. Johnson , C. Micchelli, and G. Paul, "Polynomial preconditioning for conjugate gradient calculation", *SIAM J. Numer. Anal.,* 20 (1983), pp. 362-3 76.

[66] W. Joubert , "Lanczos methods for the solution of nonsymmetric systems of linear equations", *SIAM J. Matrix Anal. Appl.,* 13 (1992), pp. 926-943.

[67] W. Kahan, " Gauss-Seidel methods for the solution of nonsymmetric systems of linear equations ", PhD thesis, University of Toronto, 1958.

[68] D. Kershaw, " The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations", *J. Comput. Phys.,* 26:43-65, 1978.

[69] S. K. Kim and A. T .Chronopoulos, "A class of Lanczos-like algorithms implemented on parallel computers", *Parallel Comput.,* 17 (1991), pp. 763-7 78.

[70] D.R. Kincaid, J.R. Respess, D.M. Young, R.G. Grimes. "ITPACK 2C: A Fortran package for solving large sparse linear systems by adaptive accelerated iterative methods ", *ACM Trans, Math. Soft.*, **8**,(1982) pp. 302-322, Algorithm 586.

[71] C. Lanczos. " Solution of systems of linear equations by minimized iterations", *J. Research of the National Bureau of Standards*, 49:33-53, 1952.

[72] Yu-Lin Lai, A. Hadjidimos, E. N. Houstis and J. R. Rice, "On the iterative solution of Hermite Collocation Equations", CSD-TR-92-094, Purdue University, 1992.

[73] C. Lawson, R. Hanson, D. Kincaid, and F .Krogh, "Basic Linear Algebra Subprograms for FORTRAN usage", *ACM Trans. Math. Soft.*, 5 (1979), pp. 308-325.

[74] Ε. Ν. Μαθιουδάκης, " Επιστημονικοί Υπολογισμοί σε Παράλληλα Περιβάλλοντα ", Διπλωματική Διατριβή Μεταπτυχιακού Διπλώματος Ειδίκευσης, Πολυτεχνείο Κρήτης, 1996.

[75] T. Manteuffel, " An incomplete factorization technique for positive definite linear systems", *Mathematics of Computation*, 34:473-497, 1980.

[76] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Iterative PDE Computations on a Transputer based Parallel Computer", Procs *2nd Hellenic Conference on Mathematics and Informatics (HERMIS '94)*, 425-431, 1994.

[77] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Mapping Parallel Iterative Algorithms for PDE Computations on a Distributed Memory Computers ", *Parallel Algorithms and Applications* , **8**, pp. 141-154,1996.

[78] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " AOR Iterative Methods for Collocation Equations on Real and Virtual Parallel Machines", Procs *3nd Hellenic Conference on Mathematics and Informatics (HERMIS '96)*, Athens, 1996.

[79] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Non-Stationary Iterative Schemes for the Solution of Elliptic Collocation Systems ", Procs *4nd Hellenic - European Research on Computer Mathematics and its Applications (HERCMA '98)*, Athens 1998.

[80] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Iterative Solution of Elliptic Collocation Systems on a Cognitive Parallel Computer ", accepted, *Advances in Partial Differential Equations*.

[81] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, " Stationary and Non-Stationary preconditioned Iterative Schemes for the Solution of Elliptic Collocation Systems ",Procs *5th IMACS conference on Iterative Methods in Scientific Computing*, Heraklion - Greece, 2001.

[82] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "BiCGSTAB for Collocation systems on distributed memory parallel architectures", Procs of *ENUMATH 2001, European conference on Numerical Mathematics and Advanced Applications*, Ischia - Italy, 2001.

[83] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "BiCGSTAB for Collocation systems on shared memory parallel architectures", Procs of *International Conference on Numerical Algorithms 2001*, Marrakesh - Morocco, 2001.

[84] N. Nachtigal, "A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non-Hermitian linear systems", Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[85] N.M. Nachtigal, S.C. Reddy and L.N. Trefethen," How fast are non-symmetric matrix iterations ?", *SIAM J. Matrix Analysis and Applications*, 13:778-795, 1992.

[86] N. M. Nachtigal, L. Reichel, and L. N. Trefethen. "A hybrid GMRES algorithm for nonsymmetric matrix iterations". Technical Report 90-7 , MIT , Cambridge, MA, 1990.

[87] D. O'Leary , " The block conjugate gradient algorithm and related methods" , *Linear Algebra Appl.,* 29 (1980), pp. 293-322.

[88] J.M. Ortega and R.G. Voigt, "Solution of Partial Differential Equations on Vector and Parallel Computers", *SIAM Review* **27***(2)*, 149-240, 1985.

[89] C.C. Paige and M.A. Saunders, " Solution of sparse indefinite systems of linear equations ", *SIAM J. Numerical Analysis,* 12:617-629, 1975.

[90] E. Papadopoulou, Y.G. Saridakis and T.S. Papatheodorou, "Orderings and Partitions of PDE Computations for a Fixed Size VLSI Architecture", Procs *of IEEE-ACM Fall Joint Computer Science Conference*, 366-374, Dallas, 1987.

[91] T.S. Papatheodorou, "Inverses for a Class of Banded Matrices and Applications to Piecewise Cubic Approximation", *J. Comp. Appl. Math.* **8***(4)*, 285-288, 1982.

[92] T.S. Papatheodorou, "Block AOR Iteration for Nonsymmetric Matrices" , *Math. Comp.* **41***(164)*, 511-525, 1983

[93] T.S. Papatheodorou and Y. G. Saridakis, " Parallel Algorithms and Architectures for Multisplitting Iterative Methods", *Parallel Computing*, 12:171-182,1989.

[94] B.N. Parlett, D.R. Taylor and Z.A. Liu, " A look ahead Lanczos algorithm for unsymmetric matrices ", *Mathematics of Computations*, 44:105-124, 1985.

[95] C. Pommerell and W. Fichtner, " PILS : An iterative linear solver package for ill-conditioned systems ", *In Supercomputing '91*, pp 588-599, IEEE Computer Society, Los Alamitos, CA, 1991.

[96] G. Radicati di Brozolo and Y. Robert, " Parallel conjugate gradient - like algorithms for solving sparse non-symmetric systems on a vector multiprocessor, *Parallel Computing*, 11:223-239, 1989.

[97] Y. Saad, "The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems", *SIAM J. Numer. Anal*, 19 (1982), pp. 485-506.

[98] Y. Saad, "Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.*, 5 (1984), pp. 203-228.

[99] Y. Saad, "Practical use of polynomial preconditionings for the conjugate gradient method", *SIAM J. Sci. Statist. Comput.*, 6 (1985), pp. 865-881.

[100] Y. Saad, "Preconditioning techniques for indefinite and nonsymmetric linear systems", *J. Comput. ppl. Math.*, 24 (1988), pp. 89-105.

[101] Y. Saad, "Krylov subspace methods on supercomputers", *SIAM J. Sci. Statist. Comput.*, 10 (1989), pp. 1200-1232.

[102] Y. Saad, " Numerical Methods for Large Eigenvalue Problems ", Manchester University Press, Manchester, UK, 1992.

[103] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm", *SIAM J. Sci. Comput.*, 14 (1993), pp. 461-469.

[104] Y. Saad and M.H. Schultz, " GMRES : a generalized minimal residual algorithm for solving nonsymetric linear systems ", *SIAM J. Scientific and Statistical Computing*, 7:856-869, 1986.

[105] Y.G. Saridakis, " Parallelism, Applicability and Optimality of Modern Iterative Methods", *Phd. Thesis*, Clarkson University, 1985.

[106] Y.G. Saridakis, "Optimally Repartitioned SOR Iterative Method for p-Cyclic Collocation Matrices",Procs *2nd Hellenic Conference on Mathematics and Info rmatics (HERMIS '94)*, 1994

[107] G.L.G. Sleijpen and D.R. Fokkema, " BICGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum", *ETNA*, 1:11-32, 1993.

[108] G.L.G. Sleijpen, H.A. van der Vorst and D.R. Fokkema, " BICGSTAB(l) and other hybrid Bi-CG methods", *Numerical Algorithms*, 7:75-109,1994.

[109] A. van der Sluis and H. van der Vorst , "The rate of convergence of conjugate gradients", *Numer. Math.,* 48 (1986), pp. 543-560.

[110] P. Sonneveld, " CGS : a fast Lanczos type solver for non-symmetric linear systems", *SIAM J. Scientific and Statistical Computing*, 10:36-52, 1989.

[111] K. C. Toh, " GMRES vs. ideal GMRES ", *SIAM J. Matrix Anal. Appl.,* 18 (1997), pp. 30-36.

[112] R.S. Varga, "Matrix Iterative Analysis",*Prentice Hall*, Englewood Cliffs, NJ, 1962

[113] P .Vassilevski, "Preconditioning nonsymmetric and indefinite finite element matrices",*J. Numer. AIg. Appl.,* 1 (1992), pp. 59-76.

[114] P. Vinsome, "Orthomin, an iterative method for solving sparse sets of simultaneous linear equations", *Proc. 4th symposium on Numerical Simulation of Reservoir Performance,* Society of Petroleum Engineers, pp 149-159,1976.

[115] H. A. van der Vorst. " Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems", *J. Comput. Phys.,* 44 (1981), pp. 1-19.

[116] H. A. van der Vorst. " High performance preconditioning ". *S J. Sc ientific and Statistical Computing,* 10:1174-1185, 1989.

[117] H.A. van der Vorst, " Bi-CGSTAB : A fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems", *SIAM J. Scientific and Statistical Computing*, 13:631-644, 1992.

[118] H. A. van der Vorst and C. Vuik. " The superlinear convergence behaviour of GMRES". *J. Comput. ppl. Math.,* 48:327-341, 1993.

[119] H. A. van der Vorst and C. Vuik. "GMRESR: A family of nested GMRES methods". *Numerical Linear Algebra with Applications,* 1:369-386, 1994.

[120] C. Vuik and H. A. van der Vorst. "A comparison of some GMRES-like methods". *Linear Algebra and its Applications,* 160:131-162, 1992.

[121] H. F. Walker, "Implementation of the GMRES method using Householder tansfor mations",*SIAM J. Sci. Statist. Comput.,* 9 (1988), pp. 152-163.

[122] T .Washio and K. Hayami. " Parallel block preconditioning based on SSOR and MILU" .*Numerical Linear Algebra with Applications,* 1:533-553, 1994.

[123] R. Weiss,"Convergence Behavior of Generalized Conjugate Gradient Methods", Ph.D. dissertation, University of Karlsruhe, Karlsruhe, Germany, 1990.

[124] 0. Widlund, "A Lanczos method for a class of non-symmetric systems of linear equations", *SIAM J. Numer. Anal.*, 15 (1978), pp. 801-812.

[125] D.M. Young, "Iterative Solution of Large Linear Systems", *Academic Press*, New York, 1981

[126] D.M. Young and K.C. Jea, "Generalized conjugate gradient acceleration of nonsym-metrizable iterative methods", *Linear Algebra Appl.*, 34:159-194,1980.

[127] L. Zhou and H.F. Walker, " Residual smoothing techniques for iterative methods", *SIAM J. Scientific and Statistical Computing*, 15:297-312,1994.