ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΓΕΝΙΚΟ ΤΜΗΜΑ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΦΑΡΜΟΣΜΕΝΕΣ ΕΠΙΣΤΗΜΕΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΔΙΑΤΡΙΒΗ ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ
ΚΑΤΕΥΘΥΝΣΗ : «ΕΦΑΡΜΟΣΜΕΝΑ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΑ ΜΑΘΗΜΑΤΙΚΑ»

ΑΡΙΘΜΗΤΙΚΗ ΜΕΘΟΔΟΣ ΠΕΠΕΡΑΣΜΕΝΩΝ ΔΙΑΦΟΡΩΝ ΣΥΜΠΑΓΩΝ ΣΧΗΜΑΤΩΝ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΕΛΛΕΙΠΤΙΚΩΝ ΠΡΟΒΛΗΜΑΤΩΝ ΣΕ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΔΙΚΤΥΑΚΩΝ ΥΠΟΛΟΓΙΣΜΩΝ

ΑΝΤΩΝΗΣ ΑΠΟΣΤΟΛΟΥ

Επιβλέπων : Επίκ. Καθηγητής **Εμμανουήλ Μαθιουδάκης**

ΧΑΝΙΑ , 2012

Η διατριβή αυτή εξετάστηκε επιτυχώς από τη παρακάτω Τριμελή Επιτροπή

- Επίκ. Καθηγητή Εμμανουήλ Μαθιουδάκη
- Καθηγήτρια Έλενα Παπαδοπούλου
- Επίκ. Καθηγητή Ανάργυρο Δελή

η οποία ορίστηκε κατά τη 2η/10-11-2011 συνεδρίαση της Γενικής Συνέλευσης Ειδικής Σύνθεσης του Γενικού Τμήματος του Πολυτεχνείου Κρήτης.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Επίκουρο καθηγητή Εμμανουήλ Μαθιουδάκη, ο οποίος ως επιβλέπων καθηγητής μου παρείχε την επιστημονική καθοδήγηση ώστε να γίνει δυνατή η ολοκλήρωση της διατριβής αυτής.

Ευχαριστώ τους καθηγητές του τομέα Μαθηματικών του Γενικού τμήματος για τις γνώσεις που μου προσέφεραν κατά την φοίτηση μου στο μεταπτυχιακό πρόγραμμα.

Τέλος ευχαριστώ τον υποψήφιο διδάκτορα Βασίλη Μάνδικα για την κατασκευή του γραμμικού συστήματος της μεθόδου.

Περίληψη

Στην διατριβή αυτή παρουσιάζεται η επίλυση Προβλημάτων Συνοριακών Τιμών (ΠΣΤ) σε παραλληλόγραμμα χωρία με μεθόδους Συμπαγών Σχημάτων Πεπερασμένων Διαφορών υψηλής ακρίβειας απευθείας σε σημεία διαφορετικά απο τους κόμβους του πλέγματος χωρίς την χρήση κάποιας μεθόδου παρεμβολής. Για πρώτη φορά η παραγόμενη αριθμητική μέθοδος υλοποιείται σε παράλληλες αρχιτεκτονικές δικτυακών υπολογισμών, λαμβάνοντας υπόψη την μελέτη της συμπεριφοράς σύγκλισης της μεθόδου της σειριακής υλοποίησης.

Ως πρόβλημα μοντέλο χρησιμοποιείται το ΠΣΤ τύπου Helmholtz, το οποίο έχει μια πολύ χρήσιμη εφαρμογή στην υλοποίηση της συνθήκης ασυμπιεστότητας κατά την επίλυση των εξισώσεων Navier-Stokes για προβλήματα ασυμπίεστων ροών σε εναλλασσόμενα πλέγματα (staggered).

Ιδιαίτερη μέριμνα δίνετε στην αύξηση των παράλληλων ιδιοτήτων του αραιού γενικού γραμμικού συστήματος της αριθμητικής μεθόδου, ώστε να γίνει εφικτή η κατασκευή αποδοτικού παράλληλου αλγορίθμου της επαναληπτικής μεθόδου Schur complement.

Η εργασία αυτή αποτελείται απο πέντε κεφάλαια:

Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στα Προβλήματα Συνοριακών Τιμών ελλειπτικού τύπου καθώς και στα πεδία εφαρμογών τους.

Στο δεύτερο κεφάλαιο παρουσιάζεται αναλυτικά η κατασκευή του αριθμητικού σχήματος 4ης τάξης ακρίβειας που επιλύει το πρόβλημα μοντέλο modified Helmholtz. Με την επιλογή αρίθμησης των αγνώστων λευκό-μαύρο (zebra coloring scheme) και τις κατάλληλες συνοριακές συνθήκες παράγεται το γενικό αραιό γραμμικό σύστημα. Στο τέλος

του κεφαλαίου υπάρχουν τα αποτελέσματα και ο σχολιασμός τους απο τις μετρήσεις σε σειριακή υλοποίηση.

Στο τρίτο κεφάλαιο γίνεται αναφορά στις πιο συνηθισμένες αρχιτεκτονικές πολυεπεξεργαστικών συστημάτων και τις αρχές που τις διέπουν. Στην συνέχεια παρουσιάζεται η διαδικασία κατασκευής του παράλληλου αλγορίθμου.

Στο τέταρτο κεφάλαιο υπάρχει η μελέτη της συμπεριφοράς και της απόδοσης σύμφωνα με τα αποτελέσματα των μετρήσεων σε δύο δικτυακά υπολογιστικά συστήματα. Στο τέλος παρουσιάζονται τα συμπεράσματα απο τη μελέτη της συμπεριφοράς κατά την εκτέλεση της εφαρμογής και επισημαίνονται τα ανοικτά ζητήματα.

Το πέμπτο κεφάλαιο παρουσιάζει τη συνολική εκτίμηση της αριθμητικής μεθόδου σύμφωνα με τα συμπεράσματα που προκύπτουν απο την εφαρμογή της υλοποίησης.

Στο παράρτημα υπάρχουν οι κώδικες σε γλώσσα Fortran απο τις υλοποιήσεις των αλγορίθμων.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Κατηγορίες προβλημάτων Συνοριακών τιμών

Η γενική μορφή των προβλημάτων συνοριακών τιμών (ΠΣΤ) στις δύο διαστάσεις έχει τη μορφή $Lu = f$ σε κάποιο χωρίο $\Omega \epsilon \mathbb{R}^2$, όπου:

$$Lu = a_{11}u_{xx} + a_{12}u_{xy} + a_{22}u_{yy} + a_1u_x + a_2u_y + a_0u \tag{1.1}$$

Οι συντελεστές a_{ij}, a_i, a_0 και το δεξί μέλος f είναι τέτοιες συναρτήσεις , ώστε στη γενική περίπτωση να υπάρχει εξάρτηση απο τα x, y, u, u_x, u_y (ημιγραμμική περίπτωση). Θα θεωρήσουμε στη συνέχεια ότι η $Lu = f$ είναι γραμμική διαφορική εξίσωση και κατά συνέπεια οι συντελεστές a_{ij}, a_i, a_0 καθώς και το δεξί μέλος f θα εξαρτώνται μόνο απο τις χωρικές μεταβλητές (x, y).
Ο τελεστής L ονομάζεται:

- Ελλειπτικός στην περίπτωση όπου ισχύει ότι $4a_{11}a_{22} > a_{12}^2$
- Υπερβολικός στην περίπτωση όπου ισχύει ότι $4a_{11}a_{22} < a_{12}^2$
- Παραβολικός στην περίπτωση όπου ισχύει ότι $4a_{11}a_{22} = a_{12}^2$

Οι πιο συνηθισμένες μορφές των παραπάνω κατηγοριών διαφορικών εξισώσεων είναι:

- εξίσωση Poisson $-\Delta u = f$ με $\Delta u = u_{xx} + u_{yy}$
- εξίσωση κύματος $u_{xx} - u_{yy} = 0$
- εξίσωση θερμότητας $u_{xx} - u_y = 0$
- εξίσωση Helmholtz $u_{xx} + u_{yy} - \lambda u = f$, όπου $\lambda \epsilon \mathbb{R}$

Στην διατριβή αυτή εφαρμόσουμε την αριθμητική μέθοδο συμπαγών πεπερασμένων διαφορών στο πρόβλημα Helmholtz θεωρώντας ότι η παράμετρος $\lambda > 0$, το οποίο αποτελεί την ειδική κατηγορία διαφορικών εξισώσεων modified Helmholtz προβλημάτων και είναι γενίκευση της διαφορικής εξίσωσης Poisson [11]. Η επίλυση αυτής της απλής σε μορφή διαφορικής εξίσωσης εμφανίζεται συχνά ως επιμέρους διαδικασία επίλυσης γενικότερων και σημαντικά πιο πολύπλοκων διαφορικών εξισώσεων. Χαρακτηριστικό παράδειγμα αποτελούν οι εξισώσεις Navier-Stokes, οι οποίες μοντελοποιούν ασυμπίεστες ροές [3, 5, 22]. Κατά την αριθμητική επίλυση τους [1] και ιδιαίτερα στην εφαρμογή της συνθήκης ασυμπιεστότητας χρειάζεται η αριθμητική επίλυση αρκετών προβλημάτων διόρθωσης της παραμέτρου της πίεσης. Η διόρθωση της πίεσης για κάθε χρονικό βήμα πραγματοποιείται με την επίλυση Poisson προβλημάτων για την περίπτωση καρτεσιανών συντεταγμένων [9, 12, 13], ενώ για πιο πολύπλοκες διακριτοποιήσεις πλέγματος , όπως για παράδειγμα σε καμπυλόγραμμες συντεταγμένες, απαιτείται η αριθμητική επίλυση προβλημάτων συνοριακών τιμών τύπου Helmholtz ή ακόμα και πιο γενικευμένης μορφής [3, 5]. Επειδή η διόρθωση της πίεσης πραγματοποιείται τουλάχιστον μια φορά σε κάθε χρονικό βήμα είναι απαραίτητη η επίλυση των προβλημάτων αυτών με βέλτιστο τρόπο, ώστε ο συνολικός αλγόριθμος της μεθόδου να θεωρηθεί αποδοτικός. Έτσι για την υλοποίηση ρεαλιστικών εφαρμογών είναι απαραίτητη η χρήση παράλληλων αρχιτεκτονικών διεξαγωγής των υπολογισμών για την επιτάχυνση της αριθμητικής μεθόδου [2, 4, 15, 16].

Τέτοιου τύπου αρχιτεκτονικές υπολογισμών σήμερα είναι διαδεδομένες , αφού η ανάπτυξη της τεχνολογίας έχει κάνει προσιτά τόσο τα πολυεπεξεργαστικά μηχανήματα, όσο και τη γρήγορη διασύνδεση μεταξύ τους, διότι τα σημερινά περιβάλλοντα διεξαγωγής επιστημονικών υπολογισμών υψηλών επιδόσεων αποτελούνται απο δικτυακά υπολογιστικά συστήματα και έτσι μπορούν να θεωρηθούν ως υπολογιστικές αρχιτεκτονικές κατανεμημένης μνήμης.

Κεφάλαιο 2

Συμπαγείς προσεγγίσεις τέταρτης τάξης

2.1 Το πρόβλημα Συνοριακών Τιμών Modified Helmholtz

Στην περίπτωση μίας χωρικής διάστασης το πρόβλημα μοντέλο Modified Helmholtz εκφράζεται ως:

$$u_{xx} - \lambda u = f \ , \ \lambda > 0 \ , \ x \epsilon \Omega \equiv [0, L_x] \tag{2.1}$$

με κατάλληλες συνοριακές συνθήκες. Χρησιμοποιώντας ομοιόμορφη διαμέριση μήκους Δx του χωρίου Ω , προκύπτουν $N_x = L_x/\Delta x$ υποδιαστήματα με κόμβους $x_i = i\Delta x$ με $0 \leq i \leq N_x$. Η δεύτερη παράγωγος u_{xx} σε κάθε κόμβο x_i του πλέγματος μπορεί να προσεγγιστεί απο τον τελεστή της κεντρικής διαφοράς

$$u_{xx} = \delta_x^2 u - \frac{\Delta x^2}{12} u_{xxxx} + O(\Delta x^4). \tag{2.2}$$

όπου u_{xxxx} θεωρούμε την τέταρτη παράγωγο στο τυχαίο σημείο διακριτοποίησης x_i. Παραλείποντας τους δύο τελευταίους όρους του δεξιού μέλους της παραπάνω εξίσωσης , έχουμε τη κεντρική διαφορά δεύτερης τάξης:

$$\delta_x^2 u = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta_x^2}$$

Η ιδέα για την επίτευξη υψηλότερης ακρίβειας είναι να προσεγγίσουμε τον όρο u_{xxxx} με τάξη δύο, επιτυνχάνοντας έτσι συνολική ακρίβεια τέταρτης τάξης. Παραγωγίζοντας δύο φορές την σχέση (2.1) προκύπτει ότι:

$$u_{xxxx} = f_{xx} + \lambda u_{xx}. \tag{2.3}$$

Εφαρμόζοντας στη παραπάνω σχέση τη κεντρική διαφορά για τους όρους u_{xx} και f_{xx} έχουμε :

$$u_{xx} = \delta_x^2 u + O(\Delta x^2) \ , \ f_{xx} = \delta_x^2 f + O(\Delta x^2) \tag{2.4}$$

οπότε με αντικατάσταση στην (2.3) παίρνουμε :

$$u_{xxxx} = \delta_x^2 f + \lambda \delta_x^2 u + O(\Delta x^2). \tag{2.5}$$

Αν αντικαστήσουμε στην (2.2) θα έχουμε :

$$u_{xx} = \delta_x^2 u - \frac{\Delta x^2}{12}(\delta_x^2 f + \lambda \delta_x^2 u) + O(\Delta x^4)$$

δηλαδή,

$$u_{xx} = \left(1 - \lambda \frac{\Delta x^2}{12}\right) \delta_x^2 u - \frac{\Delta x^2}{12} \delta_x^2 f + O(\Delta x^4).$$

οπότε για το αριθμητικό τελικό σχήμα ακρίβειας τέταρτης τάξης θα ισχύει για το πρόβλημα (2.1) η παρακάτω σχέση :

$$\left(1 - \lambda \frac{\Delta x^2}{12}\right) \delta_x^2 u - \frac{\Delta x^2}{12} \delta_x^2 f - \lambda u = f + O(\Delta x^4). \tag{2.6}$$

Παρατηρούμε ότι η μοναδική διαφορά απο το αντίστοιχο σχήμα της δεύτερης τάξης είναι η εμπλοκή υπολογισμού προσέγγισης παραγώγων του δεξιού μέλους της διαφορικής εξίσωσης . Η εξίσωση (2.6) μπορεί να έχει την μορφή :

$$\left(1+\frac{\Delta x^2}{12}\delta_x^2\right)^{-1}\left[\left(1-\lambda\frac{\Delta x^2}{12}\right)\delta_x^2-\lambda\right]u=f+O(\Delta x^4), \tag{2.7}$$

όπου ο τελεστής $T_x^{-1}\equiv\left(1+\frac{\Delta x^2}{12}\delta_x^2\right)^{-1}$ εκφράζεται μόνο ως μια συμβολική μορφή αναπαράστασης.

2.1.1 Εφαρμογή του προβλήματος μοντέλου σε δύο διαστάσεις

Η εφαρμογή αντίστοιχης μεθοδολογίας [11, 12, 13, 23] μπορεί να οδηγήσει στην παραγωγή αριθμητικών μεθόδων υψηλής ακρίβειας προσεγγίσεων της λύσης για το πρόβλημα μοντέλο δύο διαστάσεων Modified Helmholtz , το οποίο εκφράζεται ως Πρόβλημα Συνοριακών Τιμών (ΠΣΤ) στη μορφή :

$$u_{xx}(x,y)+u_{yy}(x,y)-\lambda u(x,y) \;=\; f(x,y) \quad , \quad (x,y)\in\Omega \ , \ \lambda>0 \tag{2.8}$$

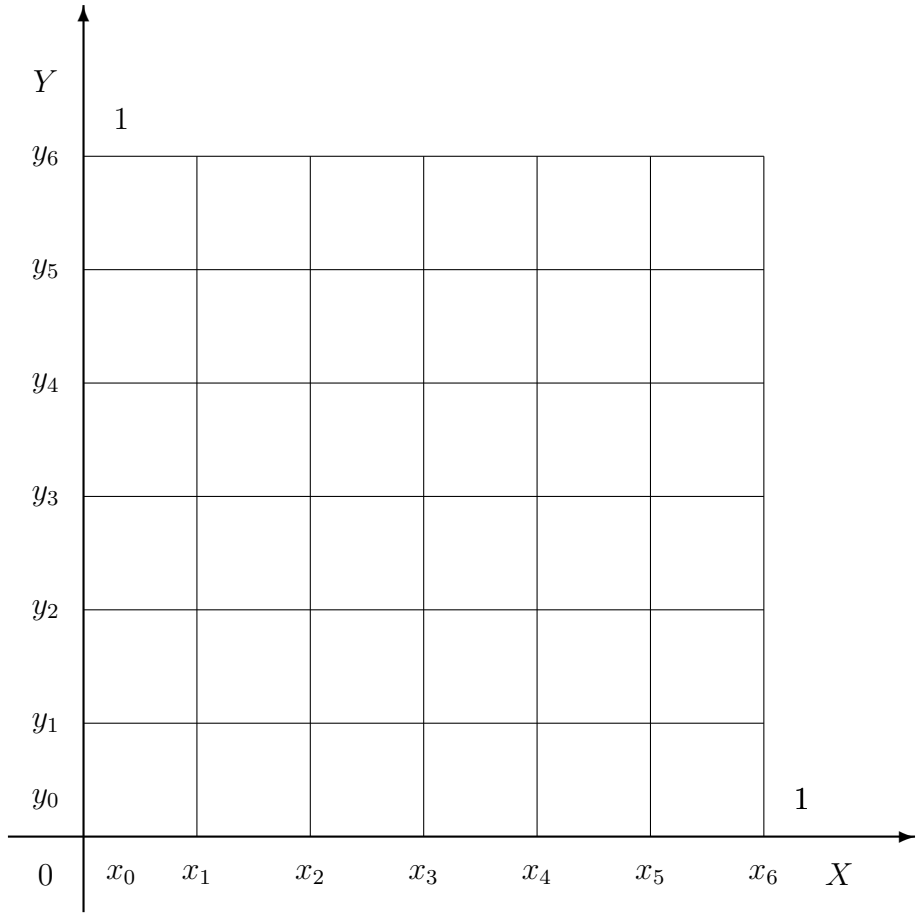
όπου το Ω είναι ένα ορθογώνιο χωρίο , με κατάλληλες συνοριακές συνθήκες. Η πραγματική λύση $u(x,y)$ και η συνάρτηση $f(x,y)$ θεωρούμε ότι είναι αρεκετά ομάλες με συνεχείς μερικές παραγώγους. Έστω το ορθογώνιο χωρίο $\Omega\equiv[0,L_x]\times[0,L_y]$ στο οποίο ορίζεται το ΠΣΤ. Διακριτοποιούμε το Ω με ομοιόμορφη διαμέρηση Δx και Δy ως προς κάθε χωρική κατεύθυνση με το πλήθος των υπολογιστικών κελιών να είναι $N_x=L_x/\Delta x$ και $N_y=L_y/\Delta y$ αντίστοιχα. Το σχήμα 2.1 εμφανίζει την περίπτωση αυτής της διακριτοποίησης για $L_x=L_y=1$ σε έξι υπολογιστικά κελιά ως προς κάθε χωρική διάσταση.

Τα σημεία του πλέγματος (υπολογιστικοί κόμβοι) είναι (x_i,y_i) με x_i και y_j , $0\le i\le N_x$, $0\le j\le N_y$. Η κεντρική διαφορά δεύτερης τάξης εκφράζεται ως :

$$\delta_x^2u_{i,j}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}, \quad \delta_y^2u_{i,j}=\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2}$$

Η εξίσωση (2.8) μπορεί να διακριτοποιηθεί σε κάποιο σημείο (x_i,y_i) του πλέγματος ως εξής:

$$\delta_x^2u_{i,j}+\delta_y^2u_{i,j}-\lambda u_{i,j}=f_{i,j}+O(\Delta x^4), \tag{2.9}$$



Σχήμα 2.1: Γεωμετρική απεικόνιση της διαμέρισης του πεδίου Ω

όπου η έκφραση $O(\Delta^2)$ περιλαμβάνει όρους δεύτερης τάξης της μορφής $O(\Delta x^2 + \Delta y^2)$ που παραλείπονται. Εργαζόμαστε ανάλογα για την μεταβλητή y ορίζοντας αντίστοιχα τα σύμβολα και παραγοντοποιώντας κατάλληλα έχουμε

$$u_{xx} - \lambda u + u_{yy} - \lambda u = f - \lambda u$$

Χρησιμοποιώντας τον συμβολισμό κατά αντιστοιχία με το μονοδιάστατο πρόβλημα προκύπτει η παρακάτω συμβολική σχέση:

$$\left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)^{-1} \left[\left(1 - \lambda\frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right] u + \left(1 + \frac{\Delta y^2}{12}\delta_y^2\right)^{-1} \left[\left(1 - \lambda\frac{\Delta y^2}{12}\right)\delta_y^2 - \lambda\right] u = f - \lambda u + O(\Delta^4) \tag{2.10}$$

όπου $O(\Delta^4)$ περιλαμβάνει όρους τάξης $O(\Delta x^4 + \Delta y^4)$. Εφαρμόζοντας τους συμβολικούς

τελεστές T_x^{-1} και T_y^{-1} θα έχουμε:

$$\left(1+\frac{\Delta y^2}{12}\delta_y^2\right)\left[\left(1-\lambda\frac{\Delta x^2}{12}\right)\delta_x^2-\lambda\right]u$$
$$+\left(1+\frac{\Delta x^2}{12}\delta_x^2\right)\left[\left(1-\lambda\frac{\Delta y^2}{12}\right)\delta_y^2-\lambda\right]u$$
$$=\left(1+\frac{\Delta y^2}{12}\delta_y^2\right)\left(1+\frac{\Delta x^2}{12}\delta_x^2\right)(f-\lambda u)+O(\Delta^4)$$

ή

$$T_y\left[\left(1-\lambda\frac{\Delta x^2}{12}\right)\delta_x^2-\lambda\right]u+T_x\left[\left(1-\lambda\frac{\Delta y^2}{12}\right)\delta_y^2-\lambda\right]u=T_xT_y(f-\lambda u)+O(\Delta^4) \quad (2.11)$$

Σημειώνουμε ότι στον όρο $O(\Delta^4)$ περιέχονται όροι μορφής $O(\Delta x^2\cdot\Delta y^2)$. Οπότε το τελικό σχήμα τέταρτης τάξης που προκύπτει είναι της μορφής:

$$(\delta_x^2+\delta_y^2)u+\frac{1}{12}(\Delta x^2+\Delta y^2)\delta_x^2\delta_y^2u-\lambda u-\frac{\lambda}{12}(\Delta x^2\delta_x^2+\Delta y^2\delta_y^2)u=f+\frac{1}{12}(\Delta x^2\delta_x^2+\Delta y^2\delta_y^2)f. \quad (2.12)$$

Αν ισχύει ότι $\gamma=\Delta_x/\Delta_y$ τότε η σχέση (2.12) γράφεται σε διακριτοποιημένη μορφή σε όλους τους κόμβους ως:

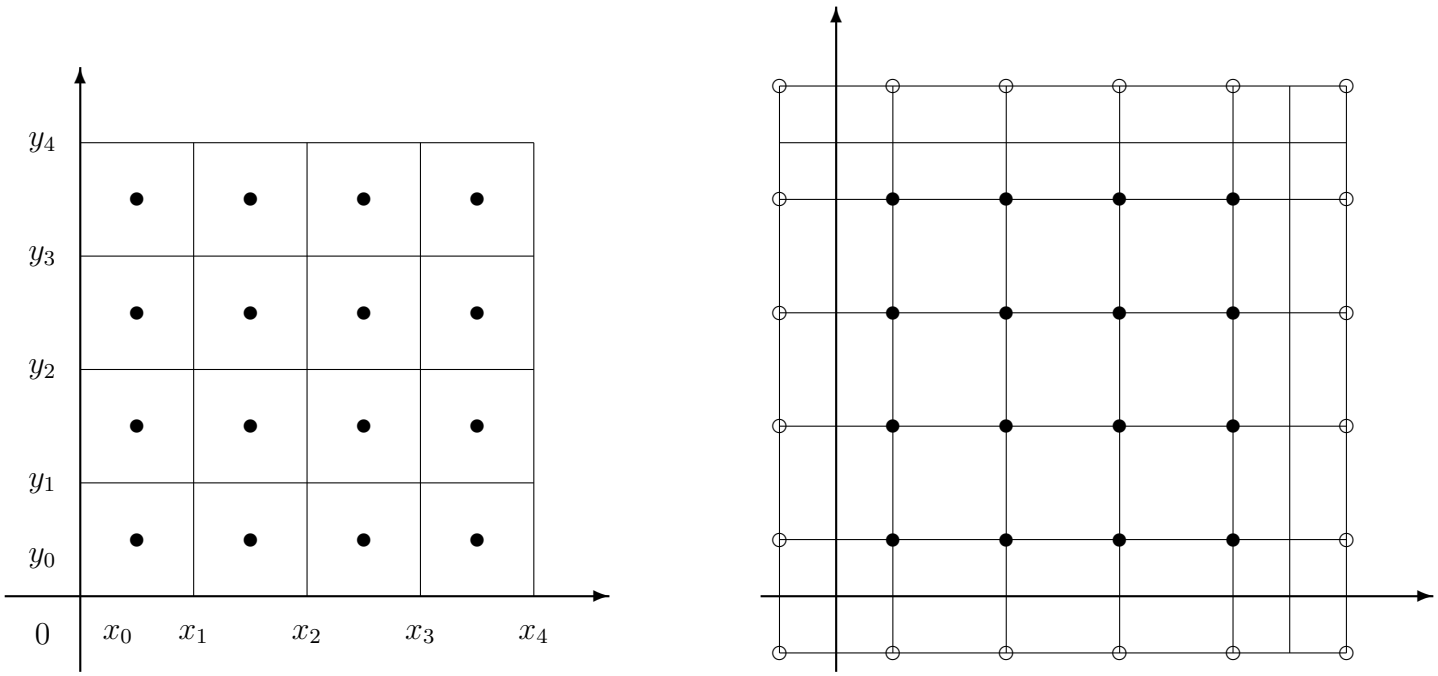
$$d(u_{i+1,j+1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i-1,j-1})+b(u_{i+1,j}+u_{i-1,j})+c(u_{i,j+1}+u_{i,j-1})-a_{i,j}u_{i,j}$$
$$=\frac{\delta x^2}{2}(8f_{i,j}+f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1}), \quad (2.13)$$

με τους συντελεστές :

$$a=10(1+\gamma^2)+4\lambda\Delta x^2,\quad b=5-\gamma^2-\tfrac{\lambda\Delta x^2}{2},\quad c=5\gamma^2-1-\tfrac{\lambda\Delta x^2}{2},\quad d=(1+\gamma^2)/2$$

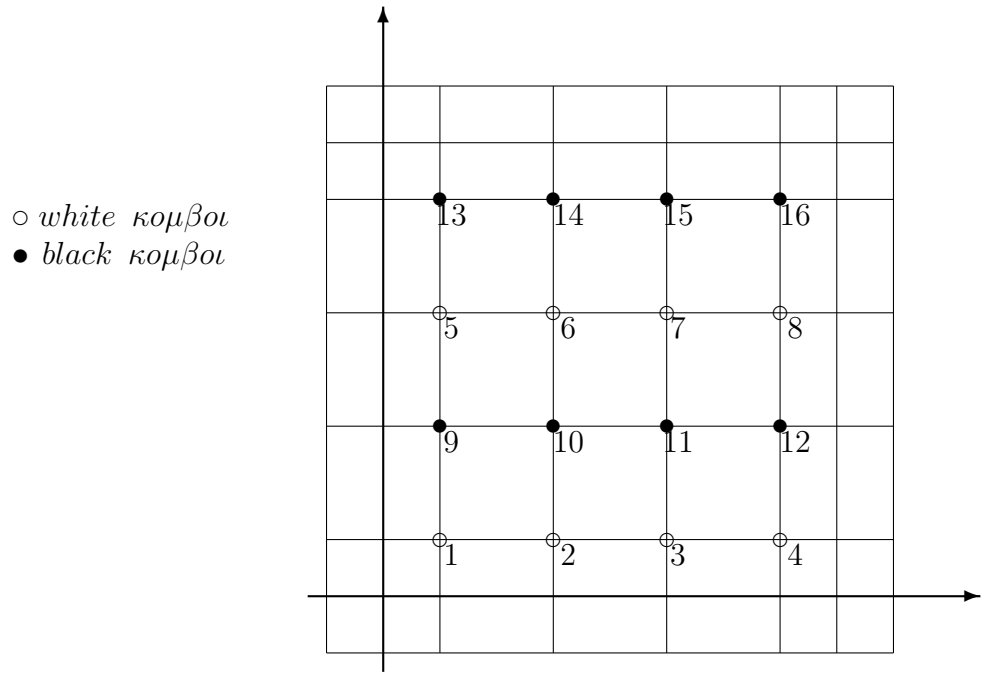
Για την υλοποίηση της παραπάνω μεθόδου και χωρίς περιορισμό της γενικότητας θεωρούμε το πεδίο $\Omega\equiv[0,1]\times[0,1]$, το οποίο διακριτοποιήται ομοιόμορφα.

Η αριθμητική μέθοδος πεπερασμένων συμπαγών διαφορών [10, 23] που βασίζεται στο σχήμα [2.12] αναζητά τη προσέγγιση της λύσης του ΠΣΤ στο κεντρικό κόμβο (x_i, y_i)



Σχήμα 2.2: Διακριτοποίηση σε 16 κόμβους (αριστερά) και το αντίστοιχο νέο υπολογιστικό πλέγμα (δεξιά)

κάθε υπολογιστικού κελιού, με συντεταγμένες $x_i = (i - \frac{1}{2})\Delta x$ και $y_i = (i - \frac{1}{2})\Delta y$, για $i = 0, 1, \ldots, N_x + 1$ και $j = 0, 1, \ldots, N_y + 1$. Τα εξωτερικά σημεία του νέου υπολογιστικού πλέγματος που προκύπτει για $i = 0, N_x + 1$ και $j = 0, N_y + 1$ είναι πρόσθετοι εικονικοί κόμβοι , οι οποίοι βρίσκονται εκτός του χωρίου Ω. Στο σχ. 2.2 παρουσιάζεται η περίπτωση της αρχικής διακριτοποίησης του χωρίου για $n = 4$ υπολογιστικά κελιά σε κάθε κατεύθυνση μαζί με τους κόμβους των κέντρων σε κάθε υπολογιστικό κελί. Δεξιά παρουσιάζεται το αντίστοιχο νέο υπολογιστικό πλέγμα του οποίου οι κόμβοι είναι τα κέντρα κάθε αρχικού υπολογιστικού κελιού. Σημειώνουμε ότι η μέθοδος [2.12] μπορεί να εφαρμοστεί για οποιόδήποτε σημείο του χωρίου, εκτός του κέντρου των αρχικών κελιών διακριτοποίησης που έχουμε επιλέξει.



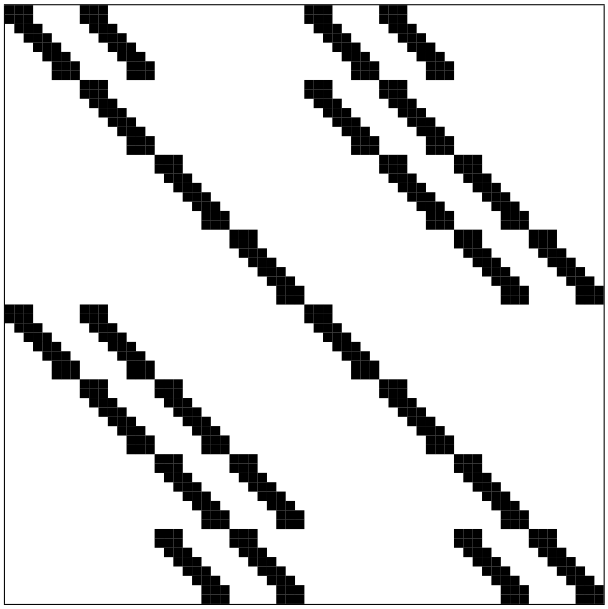
Σχήμα 2.3: Αρίθμηση αγνώστων-εξισώσεων σύμφωνα με το zebra coloring scheme

Οι εικονικοί κόμβοι εκτός του χωρίου μπορούν να απαληφθούν με την επιλογή κατάλληλων συνθηκών στο φυσικό σύνορο. Επομένως για συνοριακές συνθήκες τύπου Dirichlet μπορούμε να χρησιμοποιήσουμε τον τύπο προσέγγισης τέταρτης τάξης σφάλματος [6] ως προς μία διάσταση :

$$u_i = \frac{16}{5}u_{i+\frac{1}{2}} - 3u_{i+1} + u_{i+2} - \frac{1}{5}u_{i+3} \tag{2.14}$$

ενώ, για συνοριακές συνθήκες του είδους Neumann ο αντίστοιχος τύπος τέταρτης τάξης σφάλματος σε μία διάσταση είναι :

$$u_i = -\frac{12}{11}u_{i+\frac{1}{2}}^{'} + \frac{17}{22}u_{i+1} + \frac{9}{22}u_{i+2} - \frac{5}{22}u_{i+3} + \frac{1}{22}u_{i+4} \tag{2.15}$$



Σχήμα 2.4: Δομή του Block πίνακα συντελεστών

Οι αριθμητικές σχέσεις (2.14) και (2.15) μπορούν να συνδιαστούν κατάλληλα για τις περιπτώσεις μικτών και τύπου Robin συνοριακών συνθηκών , ώστε τελικά να χρειαστεί ο προσδιορισμός μόνο των εσωτερικών αγνώστων-τιμών της λύσης του ΠΣΤ.

Για να αυξηθούν οι ιδιότητες παραλληλοποίησης του παραγόμενου γραμμικού συστήματος θα χρησιμοποιήσουμε ένα διχρωματικό σχήμα αρίθμησης αγνώστων και εξισώσεων (βλ. σχήμα 2.3 - zebra coloring scheme). Η διαδικασία αρίθμησης ξεκινά με τη δημιουργία δυο χρωματικών ομάδων άσπρου και μαύρου χρώματος. Σε κάθε ομάδα ανήκουν όλοι οι άγνωστοι που αντιστοιχούν σε κάθε οριζόντια γραμμή πλέγματος με τον περιορισμό της μη γειτονείας ίδιου χρώματος αγνώστων ως προς την κάθετη διεύθυνση του πλέγματος. Στη συνέχεια γίνεται η αρίθμηση των αγνώστων σύμφωνα με τη λεξικογραφική μέθοδο από κάτω προς τα πάνω για την πρώτη ομάδα χρώματος και στη συνέχεια για την επόμενη. Με τον ίδιο τρόπο γίνεται και η αρίθμηση των εξισώσεων. Στο σχήμα 2.4 εμφανίζεται η δομή του παραγόμενου block πίνακα συντελεστών των αγνώστων. Κάθε block του πίνακα αντιστοιχεί σε συντελεστές αγνώστων του ίδιου χρώματος

για κάθε οριζόντια γραμμή πλέγματος . Ειδικότερα με αυτήν την αρίθμηση ο πίνακας συνελεστών $A \epsilon R^{nxn}$ της τέταρτης τάξης κεντρικών συμπαγών διαφορών μπορεί να γραφεί στην ακόλουθη μορφή :

$$A = \begin{bmatrix} A_1 & A_2 & 0 & \cdots & 0 & 0 & 0 & A_3 & A_4 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_5 & 0 & \cdots & 0 & 0 & 0 & A_6 & A_6 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & A_5 & \cdots & 0 & 0 & 0 & 0 & A_6 & A_6 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_5 & 0 & 0 & 0 & 0 & 0 & \cdots & A_6 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_5 & 0 & 0 & 0 & 0 & \cdots & A_6 & A_6 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & A_5 & 0 & 0 & 0 & \cdots & 0 & A_6 & A_6 \\ A_6 & A_6 & 0 & \cdots & 0 & 0 & 0 & A_5 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_6 & A_6 & \cdots & 0 & 0 & 0 & 0 & A_5 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & A_6 & \cdots & 0 & 0 & 0 & 0 & 0 & A_5 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_6 & A_6 & 0 & 0 & 0 & 0 & \cdots & A_5 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_6 & A_6 & 0 & 0 & 0 & \cdots & 0 & A_5 & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_4 & A_3 & 0 & 0 & 0 & \cdots & 0 & A_2 & A_1 \end{bmatrix} \tag{2.16}$$

όπου 0 είναι ο μηδενικός πίνακας κατάλληλης διάστασης , ενώ η δομή των πινάκων $A_i \, \epsilon \, R^{N_x \times N_x}$ για $i = 1, 2, 3, 4, 5, 6$ που αντιστοιχούν στην ίδια ομάδα αγνώστων οριζόντιας γραμμής στο πλέγμα διακριτοποίησης θα είναι:

$$A_i = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & \cdots & 0 & 0 & 0 & 0 \\ a_5 & a_6 & a_5 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_5 & a_6 & a_5 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_5 & a_6 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \vdots & a_6 & a_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & \vdots & a_5 & a_6 & a_5 & 0 \\ 0 & 0 & 0 & 0 & \vdots & 0 & a_5 & a_6 & a_5 \\ 0 & 0 & 0 & 0 & \vdots & a_4 & a_3 & a_2 & a_1 \end{bmatrix} \tag{2.17}$$

Για συνοριακές συνθήκες τύπου Dirichlet τα στοιχεία των πινάκων δίνονται απο τον παρακάτω πίνακα :

-	a_1	a_2	a_3	a_4	a_5	a_6
A_1	$-\frac{35}{2}s-e$	$u-e$	$\frac{e-u}{10}$	0	$\frac{u-e}{2}$	$v-\frac{5}{2}e$
A_2	$\frac{e-u^{'}}{10}$	$-\frac{s}{5}$	$\frac{s}{50}$	0	$-\frac{s}{10}$	$\frac{e-2t^{'}}{10}e$
A_3	$u^{'}-e$	$2s$	$-\frac{s}{5}$	0	s	$2t^{'}-e$
A_4	0	0	0	0	0	0
A_5	$v^{'}-\frac{5}{2}e$	$2t-e$	$\frac{e-2t}{10}$	0	$\frac{2t-e}{2}$	$-10s-4e$
A_6	$\frac{u^{'}-e}{2}$	s	$-\frac{s}{10}$	0	$\frac{s}{2}$	$\frac{2t^{'}-e}{2}$

με $s=1+\gamma^2$, $t=5-\gamma^2$, $u=7-5\gamma^2$, $v=-7-25\gamma^2$, $t^{'}=5\gamma^2-1$, $u^{'}=7\gamma^2-5$, $v^{'}=-7\gamma^2-25$, $e=\lambda\Delta x^2$

Ενώ για την περίπτωση συνοριακών συνθηκών τύπου Neumann οι αντίστοιχοι συντελεστές είναι :

-	a_1	a_2	a_3	a_4	a_5	a_6
A_1	$-6399s-\frac{105}{22}e$	$u-\frac{31}{44}e$	$-z+\frac{5}{44}e$	$w-\frac{1}{44}e$	$22w-\frac{e}{2}$	$v-\frac{193}{44}e$
A_2	$-z^{'}+\frac{5}{44}e$	$-155s$	$25s$	$-5s$	$-110s$	$-\frac{5}{22}t^{'}+\frac{5}{44}e$
A_3	$u^{'}-\frac{31}{44}e$	$961s$	$-155s$	$31s$	$682s$	$\frac{31}{22}t^{'}-\frac{31}{44}e$
A_4	$w^{'}-\frac{1}{44}e$	$31s$	$-5s$	s	$22s$	$\frac{t^{'}}{22}-\frac{e}{44}$
A_5	$v^{'}-\frac{193}{44}e$	$\frac{31}{22}t-\frac{31}{44}e$	$-\frac{5}{22}t+\frac{5}{44}e$	$\frac{t}{22}-\frac{e}{44}$	$t-\frac{e}{2}$	$-9680s-4e$
A_6	$22w^{'}-\frac{e}{2}$	$682s$	$-110s$	$22s$	$484s$	$t^{'}-\frac{e}{2}$

με :

$s=\frac{1+\gamma^2}{968}$ $\quad t=5-\gamma^2$ $\quad u=\frac{7347-837\gamma^2}{968}$

$w=\frac{237-27\gamma^2}{968}$ $\quad z=\frac{1185-135\gamma^2}{968}$ $\quad t^{'}=5\gamma^2-1$

$v^{'}=\frac{-237\gamma^2-135}{22}$ $\quad w^{'}=\frac{237\gamma^2-27}{968}$ $\quad z^{'}=\frac{1185\gamma^2-135}{968}$

$v=\frac{-237-135\gamma^2}{22}$ $\quad u^{'}=\frac{7347\gamma^2-837}{968}$ $\quad e=\lambda\Delta x^2$

2.2 Επαναληπτική επίλυση του γραμμικού συστήματος

Η εφαρμογή της αριθμητικής μεθόδου των συμπαγών πεπερασμένων διαφορών τέταρτης τάξης , έχει ως αποτέλεσμα τη παραγωγή ενός μη-συμμετρικού αραιού γραμμικού σύστηματος του οποίου η αποδοτική επίλυση μας υποδεικνύει την χρησιμοποίηση επαναληπτικών μεθόδων. Το γραμμικό σύστημα θα έχει την μορφή:

$$Au = b \tag{2.18}$$

Σύμφωνα με την αρίθμηση αγνώστων και εξισώσεων που χρησιμοποιήσαμε ο πίνακας των συντελεστών A θα έχει τη παρακάτω block μορφή:

$$A = \begin{bmatrix} D_W & H_B \\ H_W & D_B \end{bmatrix}$$

όπου,

$$H_W = \begin{bmatrix} A_6 & A_6 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_6 & A_6 & \cdots & 0 & 0 & 0 \\ 0 & 0 & A_6 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_6 & A_6 & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_6 & A_6 \\ 0 & 0 & 0 & \cdots & 0 & A_4 & A_3 \end{bmatrix} , \; H_B = \begin{bmatrix} A_3 & A_4 & 0 & \cdots & 0 & 0 & 0 \\ A_6 & A_6 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_6 & A_6 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_6 & 0 & 0 \\ 0 & 0 & 0 & \cdots & A_6 & A_6 & 0 \\ 0 & 0 & 0 & \cdots & 0 & A_6 & A_6 \end{bmatrix}$$

και

$$D_W = diag \left[\tilde{A}_1, \underbrace{A_5 \cdots A_5}_{\frac{N_y}{2}-2} \right] \;\; , \;\; D_B = diag \left[\underbrace{A_5 \cdots A_5}_{\frac{N_y}{2}-2}, \tilde{A}_2 \right]$$

$$\tilde{A}_1 = \begin{bmatrix} A_1 & A_2 \\ 0 & A_5 \end{bmatrix} \;\; , \;\; \tilde{A}_2 = \begin{bmatrix} A_5 & 0 \\ A_2 & A_1 \end{bmatrix}$$

Θεωρούμε την διάσπαση για τον πίνακα συντελεστών $A = D_A - L_A - U_A$, όπου

$$D_A = \begin{bmatrix} D_W & 0 \\ 0 & D_B \end{bmatrix} \;\; , \;\; L_A = \begin{bmatrix} 0 & 0 \\ -H_W & 0 \end{bmatrix} \;\; , \;\; U_A = \begin{bmatrix} 0 & -H_B \\ 0 & 0 \end{bmatrix}$$

καθώς και την αντίστοιχη διαμέρηση των διανυσμάτων της λύσης u όπως και του δεξιού μέλους b

$$u = \begin{bmatrix} u_W \\ u_B \end{bmatrix} \quad , \quad b = \begin{bmatrix} b_W \\ b_B \end{bmatrix}$$

Για την επαναληπτική επίλυση του γραμμικού συστήματος zebra coloring scheme (2.18) μπορούν να εφαρμοσθούν κλασσικές (stationery) επαναληπτικές μέθοδοι [7, 8, 18, 19, 20], μέθοδοι υπόχωρων Krylov (non-stationery) και η διαδικασία Shur complement [14, 15, 16, 18, 21].

Ειδικότερα, αν θεωρήσουμε την διάσπαση $A = M - N$ απο την κατασκευή του επαναληπτικού σχήματος

$$Mu^{(m+1)} = Nu^{(m)} + b, \;\; m = 0, 1, 2, ... \tag{2.19}$$

για κλασσικές στατικές μεθόδους προκύπτει η μέθοδος :

- Jacobi όταν $M = D_A$ και $N = L_A + U_A$
- Gauss-Seidel όταν $M = D_A - L_A$ και $N = U_A$
- optimal SOR όταν $M = D_A - \omega L_A$ και $N = \omega U_A + (1 - \omega) D_A \quad , \; \omega \epsilon (0, 2)$

Απο τις μεθόδους υπόχωρων Krylov, θεωρούμε τις GMRES και Bi-CGSTAB, με ή χωρίς προρύθμιση. Αν κάνουμε χρήση αριστερής προρύθμισης με τον αντίστοιχο πίνακα προρύθμισης (M),τότε το γραμμικό σύστημα θα έχει τη μορφή :

$$M^{-1}Au = M^{-1}b$$

Αν θεωρήσουμε τον πίνακα M ως τον πίνακα διάσπασης των μεθόδων Jacobi και Gauss-Seidel τότε για :

$$M = D_A$$

$$M = D_A - L_A$$

$$M = (D_A - L_A)D_A^{-1}(D_A - U_A)$$

προκύπτουν οι προρυθμίσεις Jacobi , Gauss-Seidel , συμμετρική Gauss-Seidel (SGS) αντίστοιχα.

Ο αλγόριθμος επίλυσης Schur Complement για γραμμικά συστήματα που προκύπτουν απο διχρωματικές αριθμήσεις της μορφής :

$$\begin{bmatrix} D_W & H_B \\ H_W & D_B \end{bmatrix} \begin{bmatrix} u_W \\ u_B \end{bmatrix} = \begin{bmatrix} b_W \\ b_B \end{bmatrix}$$

περιγράφεται απο τα παρακάτω βήματα:

1. Επίλυση του διαγώνιου συστήματος $D_W \widetilde{b}_W = b_W$
2. Υπολογισμός του διανύσματος $\widetilde{b}_B = b_B - H_W \widetilde{b}_W$
3. Επαναληπτική επίλυση $Su_B = \widetilde{b}_B$ όπου $S = D_B - H_W D_W^{-1} H_B$
4. Υπολογισμός του διανύσματος $\widetilde{u}_B = H_B u_B$
5. Επίλυση του διαγώνιου συστήματος $D_W \widetilde{u}_W = \widetilde{u}_B$
6. Υπολογισμός του διανύσματος $u_w = \widetilde{b}_W - \widetilde{u}_W$

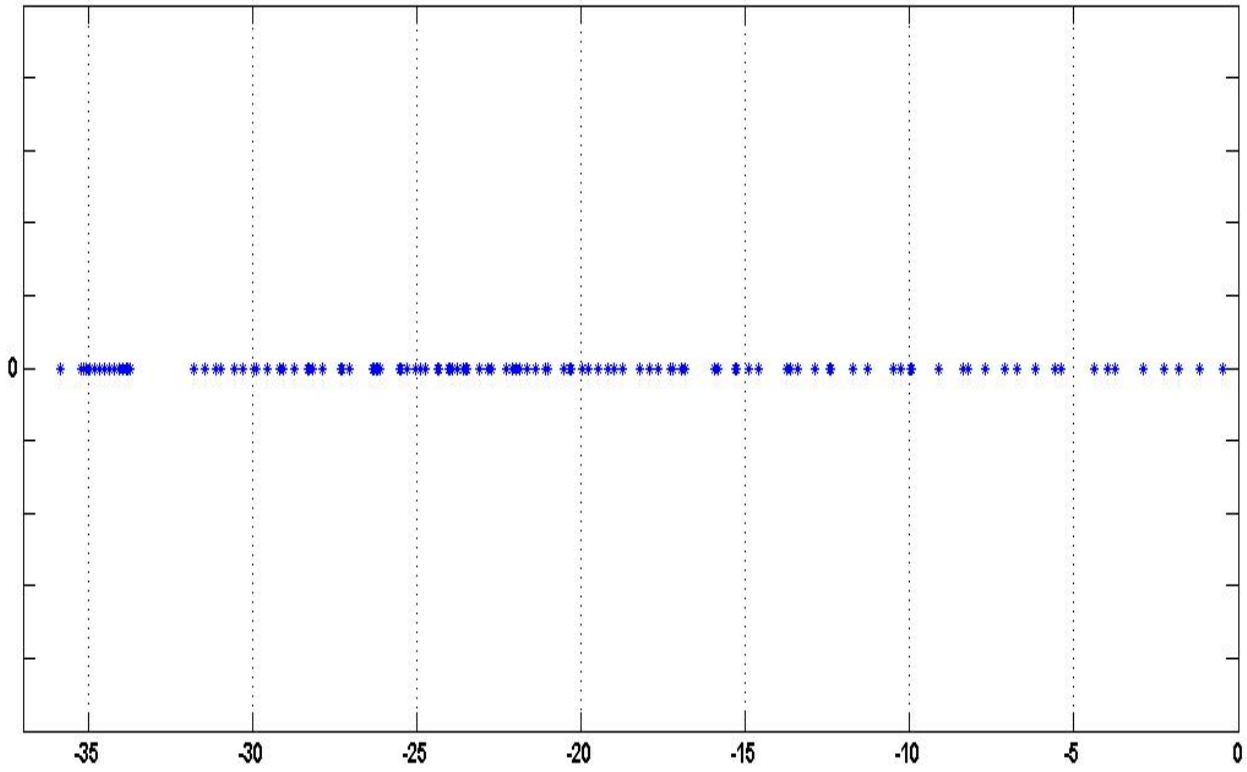
Στο τρίτο βήμα του αλγορίθμου για την επίλυση του γραμμικού συστήματος $Su_B = \widetilde{b}_B$, όπου S ο πίνακας Schur complement του A, θα πρέπει να επιλεγεί μια επαναληπτική διαδικασία η οποία περιλαμβάνει και το μεγαλύτερο υπολογιστικό κόστος του αλγορίθμου. Αυτό ισχύει διότι οι υπόλοιπες διαδικασίες εκτελούνται μια φορά και περιλαμβάνουν διαδικασίες γραμμικής άλγεβρας τμημάτων του A. Όμως η υλοποίηση αυτή μειώνει τον χρόνο εκτέλεσης σημαντικά διότι η επαναληπτική διαδικασία εμπλέκει

αγνώστους μόνο του ενός χρώματος, και ακόμα η διαδικασία Schur complement αποτελεί ένα είδος προρύθμισης του γραμμικού συστήματος. Έτσι η επαναληπτική επίλυση θα περιλαμβάνει μια μέθοδο χωρίς τη διαδικασία προρύθμισης.

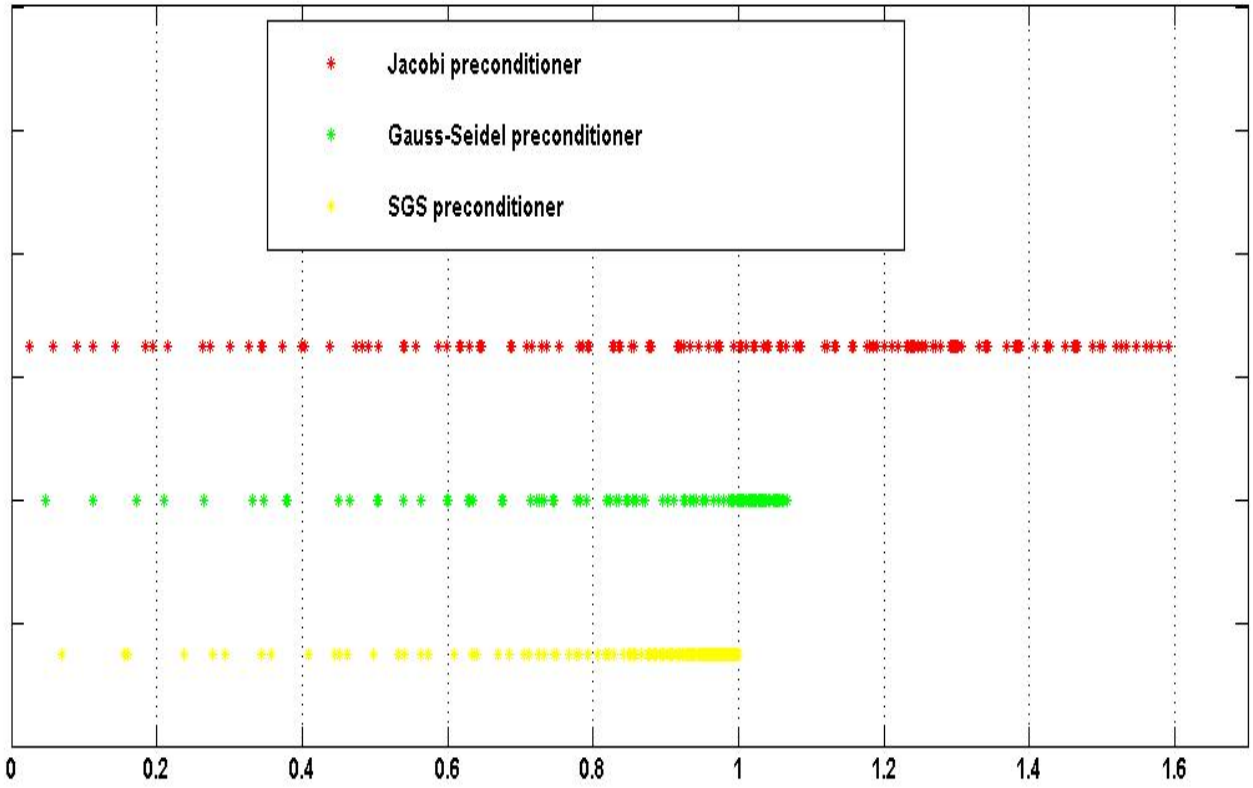
Για την περίπτωση επίλυσης, με κάποια μέθοδο υπόχωρων Krylov, η αναζήτηση των ιδιοτιμών του πίνακα συντελεστών (προρυθμισμένου ή μη) θα μπορούσε να δικαιολογήσει τη συμπεριφορά σύγκλισης.

Στο γράφημα (α) του σχήματος 2.5 αναπαρίστανται οι ιδιοτιμές του πίνακα συντελεστών A του γραμμικού συστήματος για $N_x = N_y = 16$. Όπως είναι φανερό η εφαρμογή μιας διαδικασίας προρύθμισης κρίνεται απαραίτητη για την επίλυση με χρήση μεθόδων Krylov , αφού οι ιδιοτιμές του βρίσκονται αρκετά μακριά απο αυτές του μοναδιαίου πίνακα. Στο γράφημα (β) έχουν σχεδιαστεί οι ιδιοτιμές με χρήση της τεχνικής προρύθμισης. Με κόκκινο χρώμα εμφανίζονται οι ιδιοτιμές με την εφαρμογή της προρύθμισης Jacobi. Η προρύθμιση αυτή είχε ως αποτέλεσμα τη συσσώρευση των ιδιοτιμών στο διάστημα $(0\,,\,1.6)$ με αρκετές απο αυτές να βρίσκονται γειτονικά της μονάδας και να παραμένουν όλες πραγματικές. Με πράσινο χρώμα η χρήση προρύθμισης Gauss-Siedel που έχει ως αποτέλεσμα τη μεγαλύτερη συσσώρευση των ιδιοτιμών κοντά στην μονάδα. Τέλος με κίτρινο χρώμα εμφανίζονται οι ιδιοτιμές που προκύπτουν με χρήση της προρύθμισης SGS. Σε αυτή την περίπτωση οι ιδιοτιμές περιορίζονται στο διάστημα $(0\,,\,1)$. Σύμφωνα με αυτά τα δεδομένα αναμένουμε η προρύθμιση GS να είναι η αποδοτικότερη αφού οι ιδιοτιμές παρουσιάζουν την καλύτερη συσσώρευση γύρω απο την μονάδα.

Επειδή σε κάθε περίπτωση οι ιδιοτιμές είναι πραγματικές , η χρήση της προρυθμισμένης μεθόδου GMRES θα μειώνει πολύ αργά το σφάλμα του υπολοίπου $r = b - M^{-1}Ax^m$ σε κάθε επαναληπτικό βήμα m [18]. Έτσι η μέθοδος Bi-CGSTAB με την ίδια προρύθμιση αναμένεται να συγκλίνει ταχύτερα.



(α) - Ιδιοτιμές του πίνακα συντελεστών A



(β) - Ιδιοτιμές του πίνακα συντελεστών με χρήση προρύθμισης

Σχήμα 2.5: Γράφημα ιδιοτιμών του πίνακα συντελεστών

Στην συνέχεια παρουσιάζονται τα πειραματικά αποτελέσματα της μελέτης της συμπεριφοράς σύγκλισης με την χρήση της αριθμητικής επίλυσης του ΠΣΤ. Στη μελέτη αυτή γίνεται διευρεύνηση της απόδοσης σε σχέση με το χρόνο επίλυσης κάθε μεθόδου και του υπολοίπου της παραγόμενης λύσης.

2.2.1 Αριθμητική επίλυση προβλήματος μοντέλου

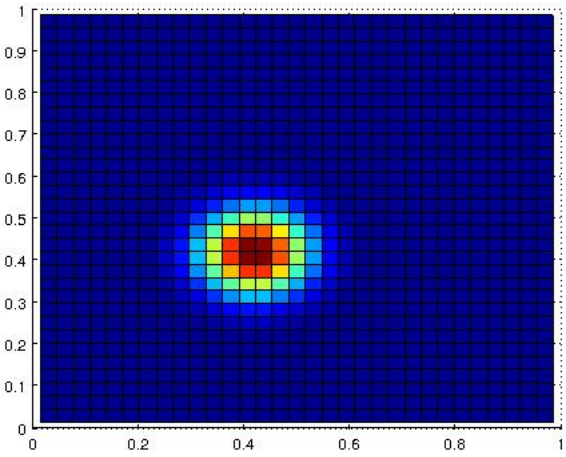
Θεωρούμε το Modified Helmholtz ΠΣΤ

$$\begin{cases} u_{xx}(x,y) + u_{yy}(x,y) - \lambda u(x,y) &= f(x,y) \ , \ (x,y) \in \Omega \equiv [0,1] \times [0,1] \\ u(x,y) &= 0 \ , \ (x,y) \in \partial\Omega \end{cases} \tag{2.20}$$

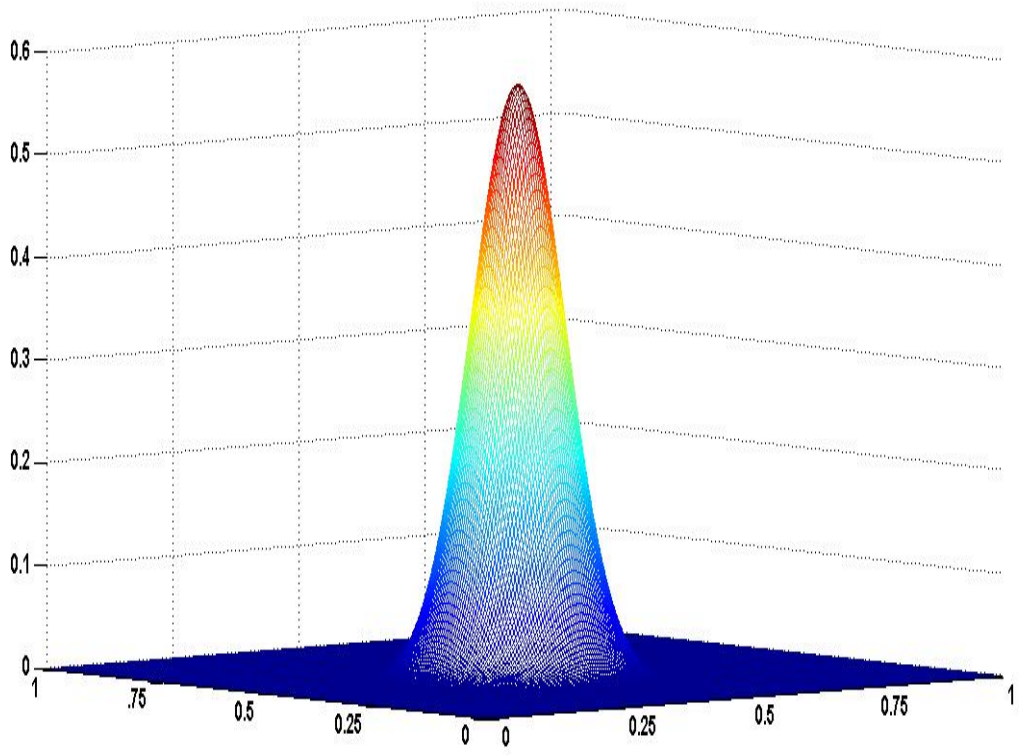
που έχει σαν λύση τη συνάρτηση :

$$\begin{cases} u(x,y) = 10\,\phi(x)\,\phi(y) \\ \\ \phi(x) = (x^2 - x)e^{-100(x-\alpha)^2} \quad , \alpha \in [0,1] \\ \phi(y) = (y^2 - y)e^{-100(y-\beta)^2} \quad , \beta \in [0,1] \end{cases} \tag{2.21}$$

η οποία για $\alpha = \beta = 0.4$ παρουσιάζεται γραφικά στα σχήματα 2.6-2.7, ως πρόβλημα δοκιμής .



Σχήμα 2.6: Η ακριβής λύση $u = u(x,y)$ για το πρόβλημα δοκιμής



Σχήμα 2.7: Η ακριβής λύση $u = u(x, y)$ για το πρόβλημα δοκιμής

Στον πίνακα 2.1 εμφανίζονται τα αριθμητικά αποτελέσματα της επίλυσης του προβλήματος (2.20) για κάθε επαναληπτική μέθοδο για $\lambda = 1$. Το πλήθος των επαναλήψεων , οι μετρήσεις του χρόνου επίλυσης καθώς και η άπειρη νόρμα του υπολοίπου $\|r\|_\infty = \|\mathbf{b} - A\mathbf{x}\|_\infty$, για τις περιπτώσεις : $N_x = N_y = 16, 32, 64, 128$ παρουσιάζονται ξεχωριστά για κάθε επαναληπτική μέθοδο. Η χρήση του λογισμικού Matlab έγινε για την υλοποίηση όλων των μεθόδων σε ένα υπολογιστικό σύστημα τύπου HP DL180 με 64GB μνήμη , με 2 επεξεργαστές τύπου Xeon 6 πυρήνων συχνότητας 2.8GHz και λειτουργικό σύστημα Oracle Linux 6.2 . Η έκδοση του λογισμικού Matlab είναι η R2012a. Σε όλες τις μεθόδους το κριτήριο τερματισμού έχει σταθερά ελέγχου σύγκλισης 10^{-6} εκτός απο τις μεθόδους Krylov όπου είναι 10^{-9}. Η αρχική προσέγγιση της λύσης σε κάθε περίπτωση είναι $x^{(0)} = b$. Η παράμετρος επανεκκίνησης της μεθόδου GMRES είναι 50.

Όπως προκύπτει απο τη μελέτη της απόδοσης των παραπάνω μεθόδων η μέθοδος επίλυσης Shur Complement επιλύει το γραμμικό σύστημα ταχύτερα για διακριτοποιήσεις μεγαλύτερες απο $N_x = N_y = 128$. Γι' αυτό και στο επόμενο κεφάλαιο παρουσιάζεται η κατασκευή παράλληλου αλγορίθμου μόνο για αυτή τη μέθοδο , αφού αναμένεται να συγκλίνει και ταχύτερα σε παράλληλα περιβάλλοντα σε σχέση με τις υπόλοιπες επειδή οι βασικές πράξεις γραμμικής άλγεβρας , οι οποίες παρουσιάζουν ισχυρές ιδιότητες παραλληλοποίησης, είναι κοινές σε όλες τις μεθόδους.

-	Jacobi				Gauss-Seidel		
N_x	Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$		Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$
16	622	2.4945ε-02	1.1322ε-08		356	2.2846ε-02	1.8428ε-09
32	2223	1.7466ε-01	1.1069ε-08		1348	2.5841ε-01	9.8943ε-10
64	7907	1.8332ε+00	1.1459ε-08		5137	5.3719ε+00	4.9882ε-10
128	27905	2.2953ε+01	1.1545ε-09		19612	1.6216ε+02	2.4972ε-10
-	SGS				GMRES Jacobi preconditioner		
N_x	Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$		Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$
16	228	2.2071ε-02	5.3989ε-09		33	1.6022ε-01	1.1927ε-06
32	818	2.6603ε-01	5.6977ε-09		69	2.2845ε+00	4.2998ε-07
64	2932	6.9798ε+00	5.9296ε-09		307	3.6576ε+01	1.0327ε-07
128	10455	1.7362ε+02	5.9997ε-09		576	5.8439ε+02	2.0040ε-08
-	GMRES GS preconditioner				GMRES SGS preconditioner		
N_x	Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$		Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$
16	22	2.1479ε-01	3.3177ε-06		18	5.9214ε-01	1.3097ε-06
32	43	3.6047ε+01	1.5838ε-06		34	9.8256ε+00	5.0022ε-07
64	118	7.7712ε+01	1.9328ε-07		66	1.9294ε+02	1.6377ε-07
128	294	2.5477ε+03	6.3710ε-08		190	5.6757ε+03	4.7468ε-08
-	BiCGSTAB Jacobi preconditioner				BiCGSTAB GS preconditioner		
N_x	Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$		Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$
16	27	1.5355ε-01	7.0849ε-07		15	2.0963ε-01	1.0617ε-06
32	44	2.3252ε+00	1.2609ε-07		29	3.6546ε+00	3.1145ε-07
64	88	3.6245ε+01	6.9961ε-08		57	7.8152ε+01	1.4910ε-07
128	194	5.8695ε+02	1.9340ε-08		129	2.5311ε+03	6.7183ε-08
-	BiCGSTAB SGS preconditioner				Shur complement		
N_x	Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$		Iterations	Time	$\|\mathbf{b} - A\mathbf{x}\|_\infty$
16	12	6.1223ε-01	1.0051ε-06		28	7.8791ε-02	1.0072ε-9
32	26	9.7678ε+00	3.1570ε-07		43	8.1876ε-01	2.7787ε-9
64	42	1.9534ε+02	1.8759ε-07		82	1.5479ε+00	1.1594-10
128	83	5.6321ε+03	1.8591ε-08		166	1.8838ε+01	6.0081-10

Πίνακας 2.1: Αποτελέσματα επίλυσης του προβλήματος (2.20)

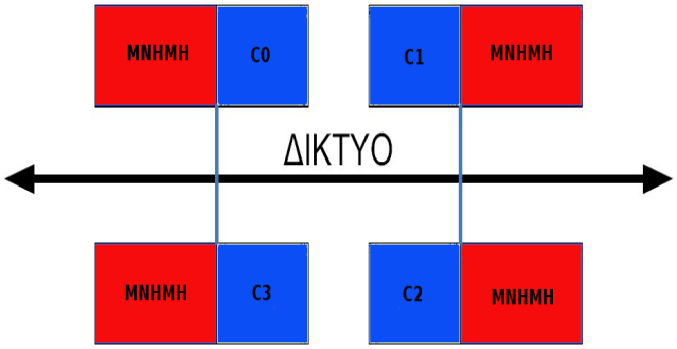
Κεφάλαιο 3

Επαναληπτική επίλυση σε δικτυακά υπολογιστικά περιβάλλοντα

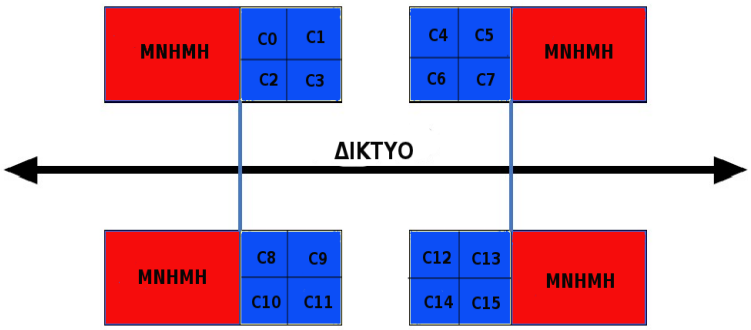
3.1 Δικτυακές υπολογιστικές αρχιτεκτονικές

Υπολογισμοί πλέγματος (grid computing) θεωρούνται οι επιστημονικοί υπολογισμοί, οι οποίοι αξιοποιούν πόρους που προέρχονται απο διαφορετικού τύπου υπολογιστικά συστήματα, τα οποία είναι συνδεδεμένα μεταξύ τους σε ένα δίκτυο. Το δίκτυο διασύνδεσης μπορεί να είναι εξιδικευμένο (τύπου infiniband,myrinet) ή απλά τύπου ethernet χαλκού και να είναι αυτό που χρησιμοποιείται για την σύνδεση στο διαδίκτυο. Αντίστοιχα υπάρχουν εξιδεικευμένες ή όχι κάρτες δικτύου οι οποίες διαχειρίζονται τη πρόσβαση στου αντίστοιχου τύπου δικτύου.

Τα δικτυακά υπολογιστικά συστήματα μπορεί δηλαδή να είναι εξεδικευμένα (συστάδες μηχανών σε ένα γρήγορο δίκτυο) ή διαφορετικού τύπου μηχανήματα, τα οποία προορίζονται για διαφορετικές χρήσεις και συνδέονται μεταξύ τους ώστε να λειτουργήσουν ως μια ολοκληρωμένη παράλληλη μηχανή. Τα μηχανήματα αυτά μπορεί να είναι εγκατεστημένα σε διαφορετικές γεωγραφικές τοποθεσίες και να λειτουργούν όταν οι συνθήκες το απαιτούν ως μέλη ενός ολοκληρωμένου υπολογιστικού περιβάλλοντος. Για παράδειγμα το Πολυτεχνείο Κρήτης διαθέτει ένα δικτυακό υπολογιστικό σύστημα τύπου συστάδας, το οποίο μπορεί να θεωρηθεί ως μέλος του συστήματος των ελληνικών υπολογιστικών συστημάτων (Hellas Grid), τα οποία στην συνέχεια αποτελούν υποστήματα του πανευρωπαϊκού υπολογιστικού πλέγματος (Euro Grid).



Σχήμα 3.1: Η απλούστερη αρχιτεκτονική δικτυακού υπολογιστή.



Σχήμα 3.2: Υβριδικό δικτυακό υπολογιστικό σύστημα.

Το σχήμα 3.1 παρουσιάζει την πιο απλή μορφή ενός δικτυακού υπολογιστικού συστήματος. Αποτελείται απο τέσσερα ανεξάρτητα συστήματα, τα οποία διαθέτουν απο ένα επεξεργαστή και τη δική του μνήμη. Έτσι για να έχει πρόσβαση ένα άλλο υπολογιστικό σύστημα στα δεδομένα της μνήμης ενός διαφορετικού υποσυστήματος θα χρειαστεί να γίνει μεταφορά τους μέσω του δικτύου. Η μεταφορά αυτή προγματοποιείται μέσω ανταλλαγής κατάλληλων μηνυμάτων.

Τα πολυεπεξεργαστικά υπολογιστικά συστήματα είναι σήμερα αρκετά διαδεδομένα, αφού η ανάπτυξη της τεχνολογίας έδωσε την δυνατότητα της κατασκευής τους με χαμηλό κόστος . Η συμμετοχή πολυεπεξεργαστικών συστημάτων έχει ως αποτέλεσμα τη δημιουργία υβριδικών δικτυακών υπολογιστικών συστημάτων κοινής και κατανεμημένης μνήμης (σχήμα 3.2).

Τα κύρια χαρακτηριστικά ενός δικτυακού υπολογιστικού περιβάλλοντος αποτελούν-

ται απο το είδος της αρχιτεκτονικής των επιμέρους συστημάτων καθώς και απο τον τρόπο διασύνδεσης τους. Τα χαρακτηριστικά αυτά λαμβάνονται υπόψη στην κατασκευή των παράλληλων αλγορίθμων των εφαρμογών, ώστε να υπάρξη η βέλτιστη αξιοποίηση των διατιθέμενων πόρων.

Η ανάπτυξη εφαρμογών σε τέτοιου τύπου συστήματα απαιτεί περισσότερα εργαλεία από αυτά που χρειάζονται για την ανάπτυξη σε μονοεπεξεργαστικά ή πολυεπεξεργαστικά συστήματα κοινής μνήμης . Έτσι ο προγραμματιστής θα μπορεί να διαχειριστεί το κόστος επικοινωνίας , το συγχρονισμό των επεξεργαστών αλλά και την εξισορρόπηση του υπολογιστικού κόστους. Αυτό επιτυγχάνεται με την χρήση διάφορων APIs (Application Programming Interface) στις υπάρχουσες υψηλού επιπέδου γλώσσες προγραμματισμού (Fortran , C). Δηλαδή με την προσθήκη βιβλιοθηκών συναρτήσεων μέσω των οποίων επιτυγχάνεται η λειτουργία όλων των επιμέρους υπολογιστικών συστημάτων ως μια ολοκληρωμένη υπολογιστική μηχανή. Το δημοφιλέστερο σήμερα API για τέτοιου τύπου υλοποιήσεις είναι το πρότυπο MPI (Message Passing Interface) [17].

3.2 Κατασκευή παράλληλου αλγόριθμου για αρχιτεκτονικές διανεμημένης μνήμης

Για την κατασκευή ενός αποδοτικού αλγόριθμου της επαναληπτικής μεθόδου Schur Complement για παράλληλες αρχιτεκτονικές διανεμημένης μνήμης θα πρέπει να ληφθούν υπόψη οι παράκατω βασικοί κανόνες:

- Να γίνει ομοιόμορφη κατανομή του υπολογιστικού φόρτου στους διαθέσιμους επεξεργαστές ανάλογα με τις δυνατότητες επεξεργασίας τους
- Ελαχιστοποίηση αδρανών επεξεργαστών
- Ελαχιστοποίηση κόστους επικοινωνίας

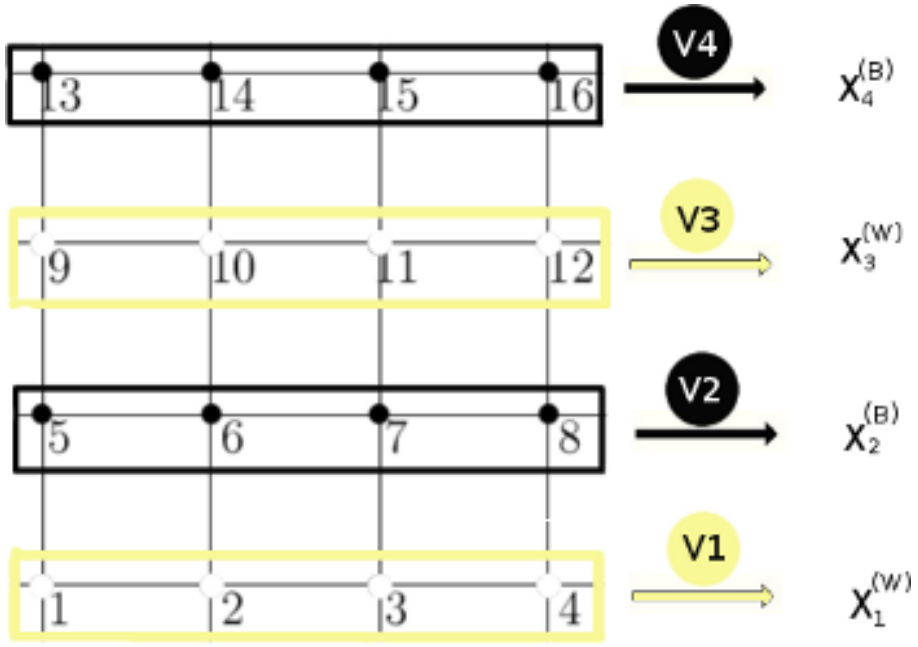
Λαμβάνοντας υπόψη τα παραπάνω, η κατασκευή του αλγορίθμου θα βασιστεί στο γεγονός ότι κάθε επεξεργαστής θα υπολογίσει κατάλληλες ομάδες αγνώστων, ώστε να υπάρξει

ελαχιστοποίηση του κόστους επικοινωνίας, αλλά και ομοιόμορφη κατανομή του φόρτου υπολογισμών, οπότε τα δεδομένα θα χρειαστεί να αντιστοιχιστούν κατάλληλα στους επεξεργαστές. Αρχικά γίνεται αντιστοίχιση ομάδων αγνώστων σε ένα εικονικό παράλληλο σύστημα με απεριόριστο πλήθος ίδιου τύπου επεξεργαστών, ενώ στη συνέχεια υπάρχει η απεικόνιση τους σε μία αρχιτεκτονική με προκαθορισμένο αριθμό επεξεργαστών, όπως άλλωστε ισχύει και στην πραγματικότητα.

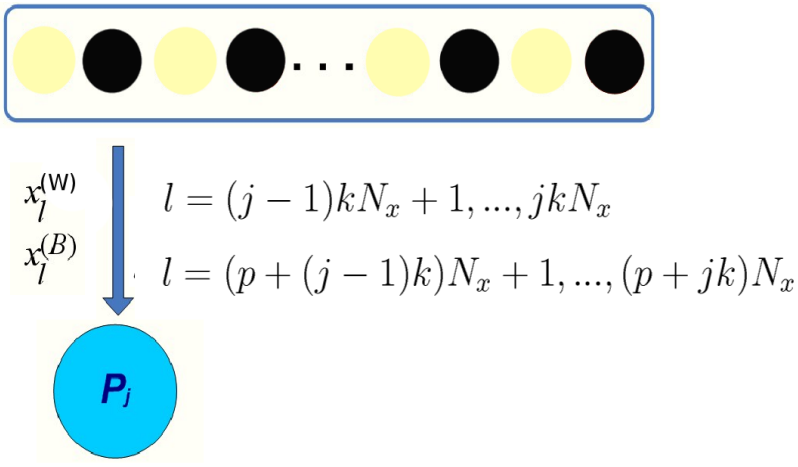
Θεωρούμε την διαμέριση του χωρίου Ω σε $N_y = 2p$ άρτιο το πλήθος υποδιαστήματα ως προς τη y κατεύθυνση και N_x ως προς τη x κατεύθυνση,χωρίς περιορισμό της γενικότητας. Αντιστοιχούμε ομάδες N_x αγνώστων σε πλήθος N_y εικονικούς επεξεργαστές V_j για $j = 1, 2, \ldots, N_y$ με το τρόπο που παρουσιάζεται στο Σχήμα 3.3. Κάθε οριζόντια ομάδα N_x αγνώστων αντιστοιχίζεται σε ένα εικονικό επεξεργαστή, καθώς και όλα τα δεδομένα που απαιτεί η αριθμητική μέθοδος για τον υπολογισμό της κάθε ομάδας αυτών των αγνώστων. Αυτό γίνεται, επειδή ο διαχωρισμός του υπολογισμού αγνώστων, οι οποίοι αντιστοιχούν σε μια οριζόντια ομάδα πλέγματος, θα είχε ως αποτέλεσμα την αύξηση του επικοινωνιακού κόστους, διότι σύμφωνα με το γενικό τύπου του αριθμητικού σχήματος (2.13) ο υπολογισμός κάθε αγνώστου απαιτεί πληροφορία απο όλους τους γειτονικούς αγνώστους του πλέγματος διακριτοποίησης. Για το λόγο αυτό ο διαχωρισμός και η αντιστοίχηση των αγνώστων στους επεξεργαστές περιορίστηκε μόνο ως προς την οριζόντια κατεύθυνση.

Στην περίπτωση τώρα που το παράλληλο υπολογιστικό σύστημα αποτελείται απο N πλήθους επεξεργαστών P_j, όπου $j = 1, \ldots, N$ και όπου το N να διαιρεί ακριβώς το p, το υπολογιστικό φορτίο μπορεί να κατανεμηθεί ομοιόμορφα σε αυτούς. Έτσι προκύπτει μια ομαδοποίηση των εικονικών επεξεργαστών και κάθε τέτοια ομάδα με $2k$ μέλη αντιστοιχίζεται σε ένα πραγματικό επεξεργαστή P_j. Ο τρόπος αυτός παρουσιάζεται στο Σχήμα 3.4, και προκύπτει ότι k άρτιος εφόσον $k = \frac{N_y}{N} = 2\frac{p}{N}$. Σε αυτή την περίπτωση αν $t = \frac{p}{N}$ κάθε επεξεργαστής P_j θα χρειαστεί να υπολογίσει $l = k \cdot N_x$ αγνώστους της ομάδας white και αγνώστους της ομάδας black, δηλαδή $2l$ συνολικούς αγνώστους ή $2k$

ομάδες αγνώστων μεγέθους N_x. Όμοια αντιμετωπίζονται οι υπόλοιπες περιπτώσεις.



Σχήμα 3.3: Απεικόνιση ομάδων αγνώστων στους εικονικούς επεξεργαστές



Σχήμα 3.4: Απεικόνιση $2k$ εικονικών επεξεργαστών.

Σύμφωνα την παραπάνω ανάλυση ο παράλληλος αλγόριθμος που υλοποιεί τη διαδικασία Schur Complement μπορεί να πάρει την παρακάτω μορφή για ένα παράλληλο υπολογιστικό σύστημα σταθερής διάστασης N επεξεργαστών.

Παράλληλος αλγόριθμος Schur Complement

βήμα 1: Απεικόνιση δεδομένων στους N επεξεργαστές

βήμα 2: Παράλληλη επίλυση του διαγώνιου συστήματος $D_W \widetilde{b}_W = b_W$

βήμα 3: Παράλληλος υπολογισμός του διανύσματος $\widetilde{b}_B = b_B - H_W \widetilde{b}_W$

$\leftrightarrow$ ανταλλαγή δεδομένων

βήμα 4: Επαναληπτική επίλυση $S u_B = \widetilde{b}_B$

$\leftrightarrow$ ανταλλαγή δεδομένων

βήμα 5: Παράλληλος υπολογισμός του διανύσματος $\widetilde{u}_B = H_B u_B$

$\leftrightarrow$ ανταλλαγή δεδομένων

βήμα 6: Παράλληλη επίλυση του διαγώνιου συστήματος $D_W \widetilde{u}_W = \widetilde{u}_B$

βήμα 7: Παράλληλος υπολογισμός του διανύσματος $u_w = \widetilde{b}_W - \widetilde{u}_W$

βήμα 8: Συλλογή αποτελεσμάτων απο N επεξεργαστές

Πιο αναλυτικά κάθε βήμα του παραπάνω αλγορίθμου περιγράφεται απο τις επόμενες διαδικασίες.

βήμα 1

for $j = 1, \ldots, N$ **do in parallel**
  if $j = 1$ then
    κατασκευή A_1, A_2, A_3, A_4, A_5, και A_6 πινάκων
    παραγοντοποίηση A_1, A_2 και A_5 πινάκων
    κατασκευή b δεξιού μέλους
    κατασκευή x αρχικής προσέγγισης
    for $j = 2, \ldots, N$ do
      P_1 στέλνει στον P_j επεξεργαστή τους πίνακες
      A_1, A_2, A_3, A_4, A_5, και A_6 και τους
      A_1, A_2 και A_5 παραγοντοποιημένους
      και τα κατάλληλα διανύσματα
      $b_l^{(W)}$, $b_l^{(B)}$, $x_l^{(W)}$ και $x_l^{(B)}$
    enddo
  else
    P_j λαμβάνει από τον P_1 τους πίνακες
    A_1, A_2, A_3, A_4, A_5, και A_6 και τους
    A_1, A_2 και A_5 παραγοντοποιημένους
    και τα κατάλληλα διανύσματα
    $b_l^{(W)}$, $b_l^{(B)}$, $x_l^{(W)}$ και $x_l^{(B)}$
  endif
enddo

βήμα 2

for $j = 1, \ldots, N$ **do in parallel**
  P_j υπολογίζει το διάνυσμα $\tilde{b}_l^{(W)}$
enddo

βήμα 3

Διαδικασία Επικοινωνίας

for $j = 2, \ldots, N$ **do in parallel**
  P_j στέλνει το διάνυσμα $\tilde{b}_{(j-1)kN_x+1}$ στον επεξεργαστή P_{j-1}
enddo

Διαδικασία Υπολογισμού

for $j = 1, \ldots, N$ **do in parallel**
  P_j υπολογίζει το διάνυσμα $\tilde{b}_l^{(B)}$
enddo

βήμα 4

for $j = 1, \ldots, N$ **do in parallel**
P_j υπολογίζει το διάνυσμα $u_l^{(B)}$ με την μέθοδο Bi-CGSTAB
Επιλογή αρχικής προσέγγισης $u_B^{(0)}$ της λύσης $u_l^{(B)}$
$r^0 = \tilde{b}_l^{(B)} - S u_B^{(0)}$ (βήμα 4α)
Επιλογή $\tilde{r} = r^{(0)}$
for $i = 1, 2, \ldots$
if $j = 1$
$\rho_{i-1} = \tilde{r}^T r^{(i-1)}$
if $\rho_{i-1} = 0$ η μέθοδος αποτυγχάνει
αποστολή ρ_{i-1} σε όλους του επεξεργαστές
endif
if $i = 1$
$\rho^{(1)} = r^{(0)}$
else
$\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \frac{\alpha_{i-1}}{\omega_{i-1}}$
$p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - \omega_{i-1} \upsilon^{(i-1)})$
endif
$\upsilon^{(i)} = S p^{(i)}$ (βήμα 4α)
if $j = 1$ $\alpha_i = \frac{\rho_{i-1}}{\tilde{r}^T \upsilon^{(i)}}$
$s = r^{(i-1)} - \alpha_i \upsilon^{(i)}$
αποστολή τιμής $r^{(i-1)}$ σε όλους τους επεξεργαστές
if $\| s \|$ είναι αρκετά μικρό then
$u_B^{(i)} = u_B^{(i-1)} + \alpha_i p^{(i)}$ stop
$u_l^{(B)} = u_B^{(i)}$
endif
$t = Ss$ (βήμα 4α)
if $j = 1$ $\omega_i = \frac{s^T t}{t^T t}$
αποστολή τιμής ω_i σε όλους τους επεξεργαστές
$u_B^{(i)} = u_B^{(i-1)} + \alpha_i p^{(i)} + \omega_i s$
$u_l^{(B)} = u_B^{(i)}$
Έλεγχος για σύγκλιση απο P_1 ενημέρωση υπόλοιπων επεξεργαστών
if $\omega_i = 0$ stop
$r^{(i)} = s - \omega_i t$
end
enddo

βήμα 4α - παράλληλος υπολογισμός $y = St$

Διαδικασία Επικοινωνίας

for $j = 1, \ldots, N-1$ **do in parallel**
P_j στέλνει το διάνυσμα $t(l - N_x + 1)$ διάστασης N_x
στον επεξεργαστή P_{j+1}
enddo

Διαδικασία Υπολογισμού

for $j = 1, \ldots, N$ **do in parallel**
P_j υπολογίζει $s = D_B^{-1}t$, $t = H_B t$ και $t = D_B^{-1}t$
enddo

Διαδικασία Επικοινωνίας

for $j = 2, \ldots, N$ **do in parallel**
P_j στέλνει το διάνυσμα $t_{(1)}$ διάστασης N_x
στον επεξεργαστή P_{j-1}
enddo

Διαδικασία Υπολογισμού

for $j = 1, \ldots, N$ **do in parallel**
P_j υπολογίζει τα διανύσματα $t = H_W t$ και $s = s - t$
enddo

βήμα 5

Διαδικασία Επικοινωνίας

for $j = 1, \ldots, N-1$ **do in parallel**
  P_j στέλνει το διάνυσμα $u^{(B)}_{(p+jk-1)N_x+1}$ στον επεξεργαστή P_{j+1}
enddo

Διαδικασία Υπολογισμού

for $j = 1, \ldots, N$ **do in parallel**
  P_j υπολογίζει το διάνυσμα $\tilde{u}^{(B)}_l$
enddo

βήμα 6

for $j = 1, \ldots, N$ **do in parallel**
  P_j υπολογίζει το διάνυσμα $\tilde{u}^{(W)}_l$
enddo

βήμα 7

for $j = 1, \ldots, N$ **do in parallel**
  P_j υπολογίζει το διάνυσμα $u^{(W)}_l$
enddo

βήμα 8

for $j = 1, \ldots, N$ **do in parallel**
  if $j \neq N$ then
    P_j στέλνει στον P_1 επεξεργαστή
      τα διανύσματα $u^{(W)}_l$ και $u^{(B)}_l$
  else
    for $j = 2, \ldots, N$ do
    P_1 λαμβάνει απο τον P_j επεξεργαστή
    τα κατάλληλα διανύσματα $u^{(W)}_l$ και $u^{(B)}_l$
  endif
enddo

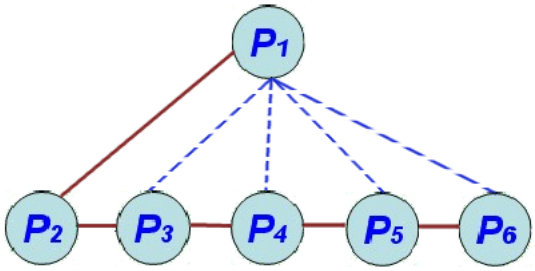
Ως προς την αρχιτεκτονική επικοινωνίας του παραπάνω αλγορίθμου θα είναι τύπου master-slave. Δηλαδή ο χρήστης αρχικά θα αναθέσει στο βήμα 1 το πρόβλημα σε ένα επεξεργαστή. Αυτός θα είναι ο ίδιος απο τον οποίο θα ζητηθούν τα αποτέλεσμα και έτσι θα έχει την αρχική κατανομή των δεδομένων στους υπόλοιπους επεξεργαστές. Επίσης στο βήμα 4 θα αποφασίζει και θα ενημερώνει τους υπόλοιπους αν η επαναληπτική διαδικασία έχει επιτύχει σύγκλιση ή όχι και φυσικά στο τέλος του βήματος 8 θα συλλέξει τα αποτελέσματα ώστε να είναι διαθέσιμα ενοποιημένα στο χρήστη. Το ρόλο αυτού του κύριου επεξεργαστή θα αναλάβει ο πρώτος σε αρίθμηση κόμβος P_1, ο οποίος θα πρέπει να σημειωθεί ότι θα πραγματοποιήσει και όλους τους υπολογισμούς που χρειάζονται στα υπόλοιπα βήματα του αλγορίθμου για τον υπολογισμό της ομάδας αγνώστων που του αναλογεί σε αντιστοιχία με τους υπόλοιπους επεξεργαστές.

Οπότε ο κύριος επεξεργαστής σύμφωνα με τον αλγόριθμο του βήματος 1 θα κατασκευάσει και θα διαμερίσει και θα στείλει τα αρχικά δεδομένα για κάθε επεξεργαστή. Δηλαδή τους βασικούς πίνακες A_i για $i = 1, 2, \dots, 6$ μαζί με τα τμήματα του δεξιού μέλους και της αρχικής προσέγγισης της λύσης του γραμμικού συστήματος (2.18) που χρειάζεται κάθε επεξεργαστής για τον υπολογισμό των αγνώστων που του αντιστοιχούν.

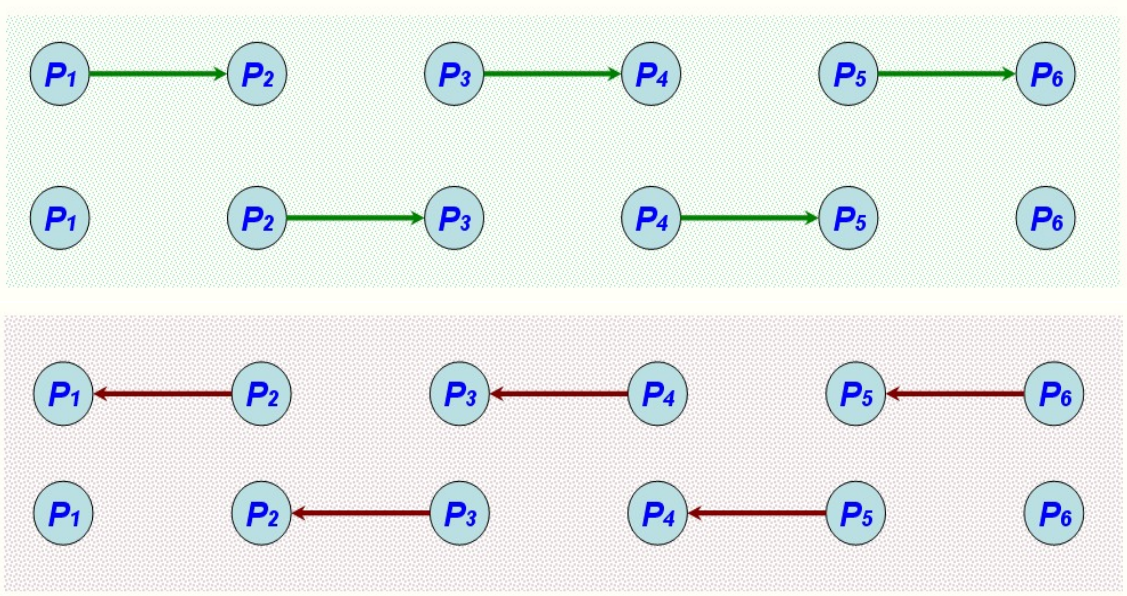
Στο βήμα 4 ο κύριος επεξεργαστής θα χρειαστεί να συλλέξει τις απαραίτητες πληροφορίες (τμήματα διανυσμάτων) για τον υπολογισμό των σταθερών της μεθόδου BiCGSTAB καθώς και την αποστολή τους πίσω σε όλους τους επεξεργαστές, ώστε να συνεχίσουν το κάθε βήμα της επαναληπτικής διαδικασίας. Η ίδια διαδικασία ακολουθείται και κατά τον υπολογισμό της νόρμας του κριτηρίου τερματισμού, όπου ο κύριος επεξεργαστής, ο οποίος θα είναι ο μόνος που θα την υπολογίσει, θα αποφασίσει και θα ενημερώσει κατάλληλα τους υπόλοιπους αν έχει επιτευχθεί σύγκλιση.

Στα βήματα 3 και 5 θα χρειαστεί να πραγματοποιηθεί ανταλλαγή δεδομένων ανάμεσα σε γειτονικούς σε αρίθμιση επεξεργαστές, λόγω της δομής των πινάκων H_W και H_B. Η ίδια διαδικασία επικοινωνίας υπάρχει και στο βήμα 4 για κάθε επαναληπτικό βήμα της μεθόδου BiCGSTAB, επειδή περιλαμβάνει δύο πολλαπλασιασμούς του πίνακα S με διά-

νυσμα. Στον υπολογισμό όμως του πίνακα S απαιτείται πολλαπλασιασμός διανύσματος με τους πίνακες H_W και H_B.



Σχήμα 3.5: Σχηματική απεικόνιση διασύνδεσης επεξεργαστών.



Σχήμα 3.6: Διαδικασία αμφίδρομης επικοινωνίας μεταξύ γειτονικών επεξεργαστών.

Έτσι η συνολική διασύνδεση των επεξεργαστών θα είναι μεικτού τύπου διασύνδεσης σε σειρά και δυαδικού δένδρου, ώστε να ελαχιστοποιείται η διαδικασία μεταφοράς δεδομένων στον κύριο επεξεργαστή απο τους υπόλοιπους και αντίστροφα. Στο σχήμα (3.5) παρουσιάζεται σχηματικά η μεικτή διασύνδεση των επεξεργαστών για τη περίπτωση $N = 6$. Η σύνδεση με τη συνεχή γραμμή χρησιμοποιείται για ανταλλαγή δεδομένων στις περιπτώσεις πολλαπλασιασμού των πινάκων H_B και H_W, ενώ η διακεκομμένη σύνδεση χρησιμοποιείται για την ανταλλαγή δεδομένων με το κύριο επεξεργαστή απο τους υπόλοιπους. Όπως παρατηρούμε είναι μεικτού τύπου αρχιτεκτονικής σύνδεσης σε σειρά και σε αστέρα.

Το σχήμα (3.6) εμφανίζει τη διαδικασία επικοινωνίας των γειτονικών επεξεργαστών. Στο πάνω μέρος παρουσιάζεται η περίπτωση πολλαπλασιασμού διανύσματος με τον πίνακα H_B και στο κάτω αντίστοιχα με τον H_W. Όπως φαίνεται η κάθε διαδικασία ανταλλαγής δεδομένων απαιτεί δύο φάσεις. Αρχικά οι περιττής αρίθμησης επεξεργαστές στέλνουν (λαμβάνουν) δεδομένα στους άρτιους, οι οποίοι λαμβάνουν (στέλνουν) τα δεδομένα για τη περίπτωση πολλαπλασιασμού του πίνακα H_B (H_W).

Στο επόμενο κεφάλαιο παρουσιάζεται η υλοποίηση του παραπάνω παράλληλου αλγορίθμου καθώς και η μελέτη της συμπεριφοράς του σε δύο δικτυακά υπολογιστικά μηχανήματα.

Κεφάλαιο 4

Υλοποίηση και μελέτη της συμπεριφοράς του παράλληλου αλγορίθμου

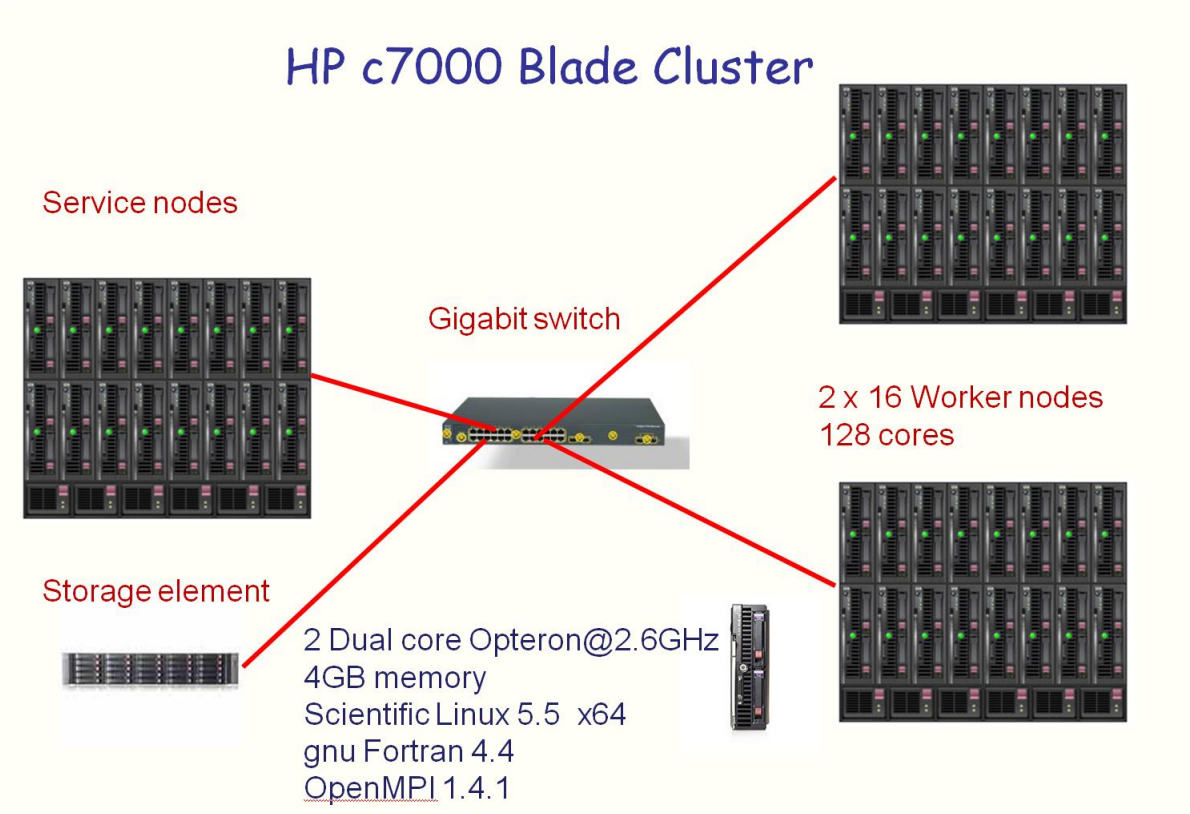
Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα απο τη μελέτη της συμπεριφοράς της υλοποίησης του παράλληλου αλγορίθμου της επαναληπτικής μεθόδου Schur Complement που παρουσιάστηκε στο προηγούμενο κεφάλαιο.

4.1 Περιγραφή των χρησιμοποιηθέντων πολυεπεξεργαστικών υπολογιστικών συστημάτων

Η ανάπτυξη της εφαρμογής έγινε με χρήση της γλώσσας προγραμματισμού Fortran για αριθμητική διπλής ακρίβειας,ενώ η υλοποίηση των παράλληλων τμημάτων του αλγορίθμου έγινε σύμφωνα με το πρότυπο MPI.

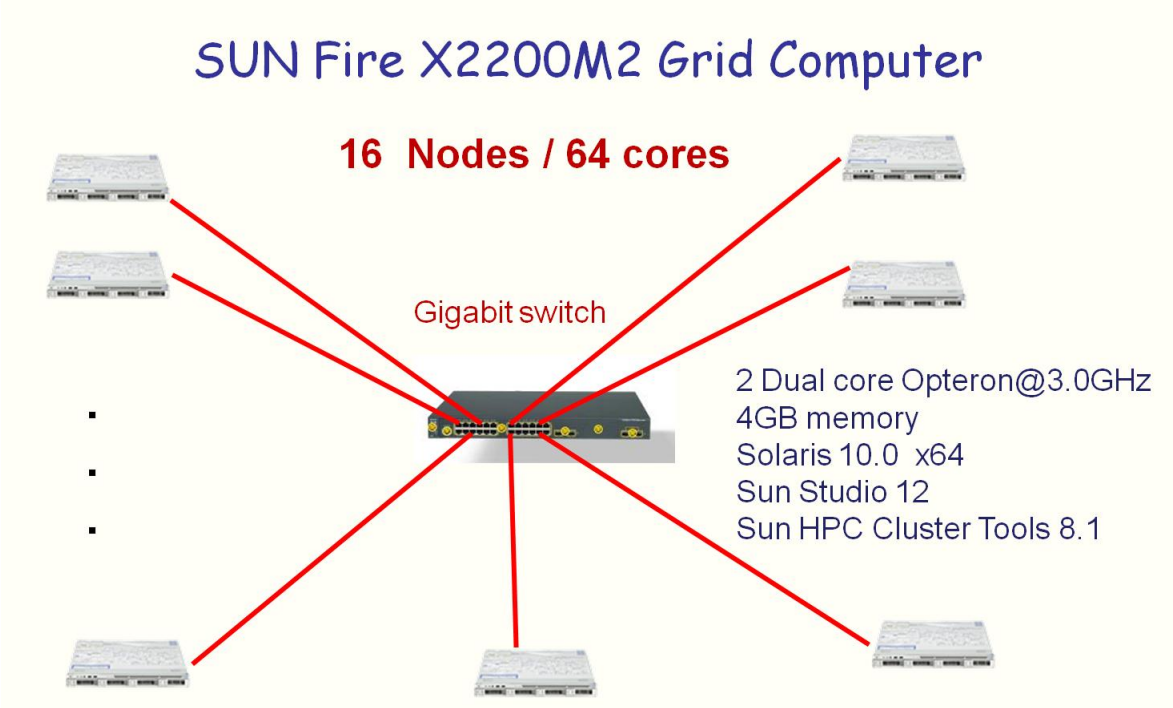
Η μελέτη της συμπεριφοράς της υλοποίησης πραγματοποιήθηκε σε δύο δικτυακά υπολογιστικά συστήματα. Το πρώτο μηχάνημα είναι ο υπολογιστής πλέγματος του *Πολυτεχνείου Κρήτης* το οποίο φέρει την ονομασία *Τάλως*. Αποτελείται απο τρείς συστάδες υπολογιστικών κόμβων οι οποίες επικοινωνούν μεταξύ τους καθεμία μέσω μιας κάρτας δικτύου ταχύτητας 1Gbps συνδεδεμένες σε δίκτυο χαλακού τύπου Ethernet (σχήμα 4.1). Κάθε συστάδα είναι τύπου HP c7000, ενώ κάθε υπολογιστικός της κόμβος είναι τύπου blade και διαθέτει δύο φυσικούς επεξεργαστές Opteron@2.6GHz, με δύο υπολογιστικούς

πυρήνες για τον καθένα και συνολικής μνήμης 4GB για κάθε υπολογιστικό κόμβο. Η ταχύτητα εποκοινωνίας υπολογιστικών κόμβων που ανήκουν στην ίδια συστάδα 10Gbps. Το σύστημα διαθέτει ένα εξειδικευμένο κόμβο ως κόμβο αποθήκευσης, τους δίσκους του οποίου χρησιμοποιούν όλοι οι υπολογιστικοί κόμβοι μέσω του προτύπου NFS. Σε αυτή την υλοποίηση χρησιμοποιήθηκαν οι πρώτοι τρείς υπολογιστικοί κόμβοι απο την πρώτη συστάδα, στη δυνέχεια οι επόμενοι δεκαέξι απο την δεύτερη συστάδα και τέλος οι επόμενοι δεκατρείς απο την τρίτη συστάδα για δοκιμές με χρήση 32 υπολογιστικών κόμβων ή 128 υπολογιστικών πυρήνων.



Σχήμα 4.1: Το πολυεπεξεργαστικό σύστημα *Τάλως*

Το λειτουργικό σύστημα αυτής της μηχανής είναι το Scientific Linux 5.5 x64,ενώ η υλοποίηση του κώδικα έγινε με χρήση της γλώσσας προγραμματισμού Fortran σε διπλή ακρίβεια και ειδικότερα απο την έκδοση GNU Fortran 4.4. Η ανταλλαγή μηνυμάτων υλοποιήθηκε μέσω του πρωτύπου MPI έκδοσης OpenMPI 1.4.1.



Σχήμα 4.2: Το πολυεπεξεργαστικό σύστημα *Πλειάδες*

Το δεύτερο δικτυακό υπολογιστικό μηχάνημα που χρησιμοποιήθηκε ανήκει στο *Εργαστήριο Εφαρμοσμένων Μαθηματικών και Η/Υ* του *Πολυτεχνείου Κρήτης* και ονομάζεται *Πλειάδες* (σχήμα 4.2). Αποτελείται απο δεκαέξι ανεξάρτητους πολυεπεξεργαστικούς κόμβους συνδεδεμένους μεταξύ τους σε δίκτυο χαλκού τύπου Ethernet μεταβαλόμενης ταχύτητας απο 10Mbps,100Mbps και 1Gbps.

Κάθε πολυεπεξεργαστικός κόμβος του συστήματος είναι τύπου SunFire X2200M2 με δύο διαθέσιμους επεξεργαστές των δύο πυρήνων τύπου Opteron 2222@3.0Ghz, μνήμης L2 cache 1 MB ανά πυρήνα. Η συνολική μνήμη κάθε κόμβου είναι 4GB, ενώ το λειτουργικό τους σύστημα είναι το Solaris 10. Το λογισμικό ανάπτυξης της εφαρμογής είναι ο μεταγλωττιστής Fortran από το βελτιστοποιημένο για αυτή την αρχιτεκτονική πακέτο

λογισμικού Sun Studio 12. Επίσης χρησιμοποιήθηκε το πρότυπο MPI και ειδικότερα η ελεύθερης χρήσης υλοποίηση του OpenMPI 1.2.5

4.2 Αποτελέσματα δοκιμών στο υπολογιστικό σύστημα *Τάλως*

Απο το μηχάνημα *Τάλως* χρησιμοποιήθηκαν 32 υπολογιστικοί κόμβοι, δηλαδή 128 πυρήνες, για προβλήματα διακριτοποίησης απο 128 μέχρι 4096 υπολογιστικά κελιά ανά κατεύθυνση.

Ο πίνακας Τ4.1 εμφανίζει τους χρόνους που χρειάστηκαν για να επιλυθούν όλα τα προβλήματα μετρημένους σε δευτερόλεπτα. Παρουσιάζονται τόσο οι συνολικοί χρόνοι εκτέλεσης,όσο και οι χρόνοι υπολογισμών μαζί με τους χρόνους επικοινωνίας και συγχρονισμού του παράλληλου αλγορίθμου για κάθε περίπτωση. Ο πίνακας Τ4.2 που ακολουθεί εμφανίζει τις μετρήσεις της επιτάχυνσης (speedup) του παράλληλου αλγορίθμου όπου ορίζεται να είναι

$$speedup = \frac{T_s}{T_p}$$

όπου T_s είναι ο χρόνος εκτέλεσης σε ένα επεξεργαστή και T_p ο αντίστοιχος χρόνος εκτέλεσης για p επεξεργαστές.

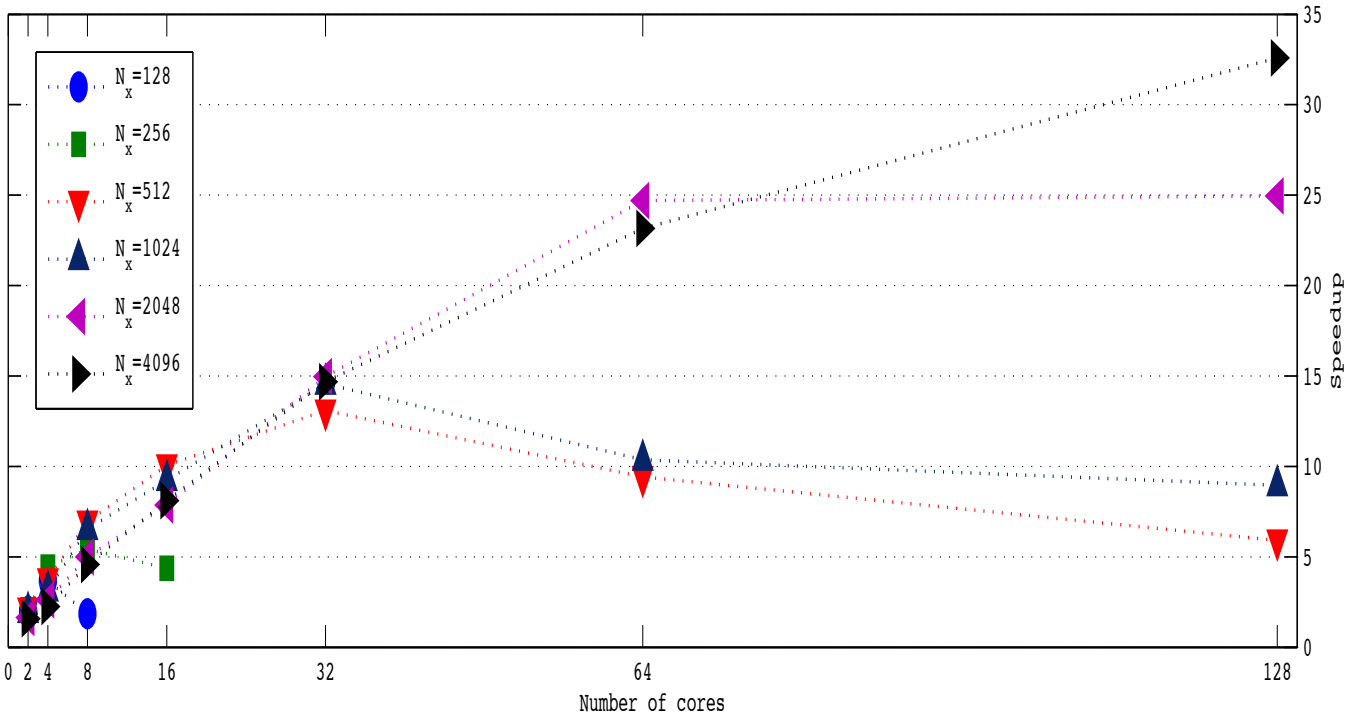
Το σχήμα 4.3 εμφανίζει το συνολικό γράφημα της επιτάχυνσης. Παρατηρούμε ότι για $N_x = 256$ και χρήση 4 υπολογιστικών πυρήνων εμφανίζεται υπεργραμμική επιτάχυνση εξαιτίας του μεγέθους του προβλήματος και της μνήμης cache.

N_x	cores	Communication Time	Computational Time	Total Time
128	1	-	3.630e-01	3.630e-01
	2	5.695e-03	1.765e-01	1.822e-01
	4	7.337e-03	9.118e-02	9.851e-02
	8	1.433e-01	5.241e-02	1.958e-01
256	1	-	3.337e+00	3.337e+00
	2	6.482e-02	1.541e+00	1.606e+00
	4	2.056e-02	7.318e-01	7.524e-01
	8	2.197e-01	3.949e-01	6.147e-01
	16	5.493e-01	2.154e-01	7.647e-01
512	1	-	2.966e+01	2.966e+01
	2	6.643e-01	1.323e+01	1.390e+01
	4	5.175e-01	7.365e+00	7.883e+00
	8	8.784e-01	3.390e+00	4.268e+00
	16	1.303e+00	1.643e+00	2.947e+00
	32	1.382e+00	8.843e-01	2.266e+00
1024	1	-	2.413e+02	2.413e+02
	2	7.112e+00	1.171e+02	1.242e+02
	4	2.448e+00	7.470e+01	7.715e+01
	8	5.874e+00	3.103e+01	3.690e+01
	16	1.033e+01	1.575e+01	2.609e+01
	32	8.985e+00	7.589e+00	1.657e+01
	64	1.930e+01	3.977e+00	2.327e+01
2048	1	-	2.638e+03	2.638e+03
	2	1.708e+00	1.610e+03	1.612e+03
	4	5.377e+01	9.553e+02	1.009e+03
	8	2.440e+01	5.045e+02	5.289e+02
	16	1.260e+02	2.095e+02	3.355e+02
	32	8.700e+01	8.924e+01	1.762e+02
	64	6.311e+01	4.366e+01	1.067e+02
	128	8.191e+01	2.378e+01	1.056e+02
4096	1	-	2.675e+04	2.675e+04
	2	5.936e+00	1.689e+04	1.690e+04
	4	8.950e+02	1.099e+04	1.189e+04
	8	3.840e+02	5.464e+03	5.848e+03
	16	2.728e+02	3.026e+03	3.299e+03
	32	8.428e+02	9.793e+02	1.822e+03
	64	3.990e+02	7.561e+02	1.155e+03
	128	4.955e+02	3.257e+02	8.213e+02

Table 4.1: Αριθμητικά αποτελέσματα για το σύστημα *Τάλως*

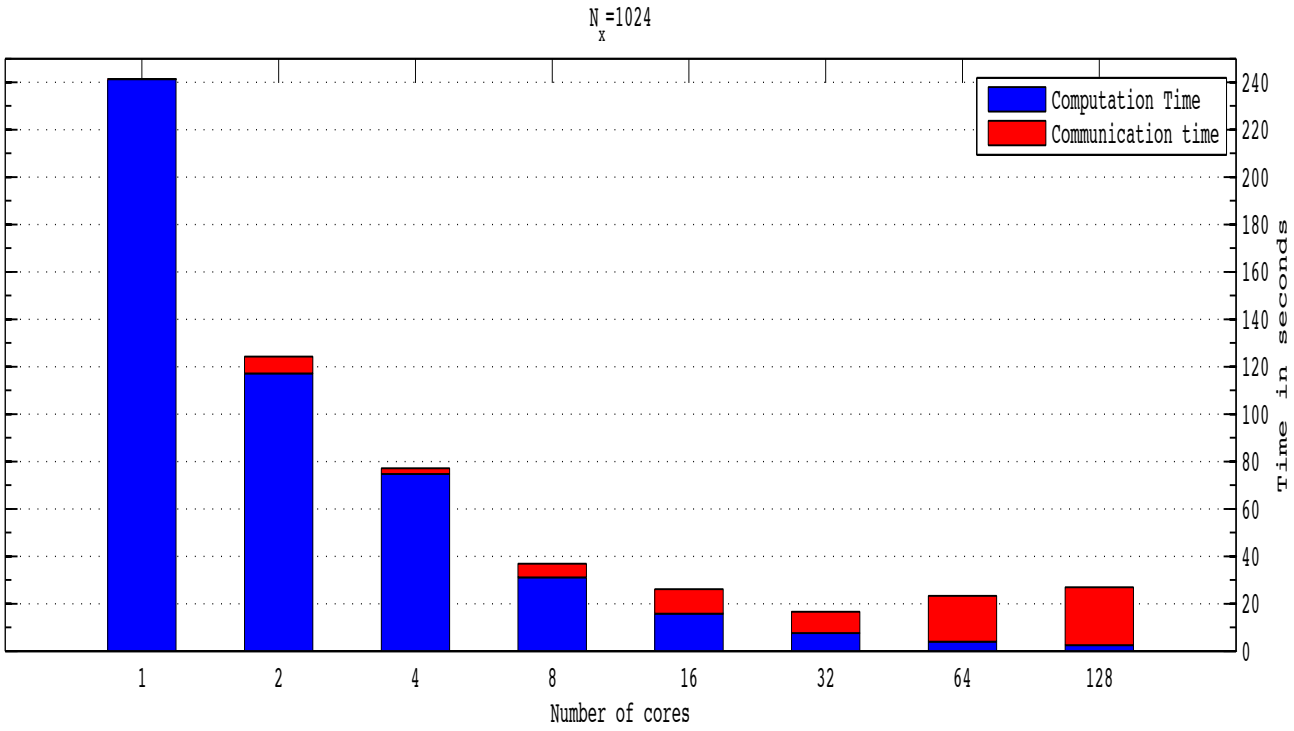
Τάλως speedup							
Number of Cores :	2	4	8	16	32	64	128
$N_x = 128$	1.99	3.68	1.85				
$N_x = 256$	2.07	4.43	5.42	4.36			
$N_x = 512$	2.13	3.76	6.94	10.06	13.08	9.42	5.91
$N_x = 1024$	1.94	3.12	6.54	9.24	14.56	10.37	8.95
$N_x = 2048$	1.63	2.61	4.98	7.86	14.96	24.70	24.95
$N_x = 4096$	1.58	2.24	4.57	8.11	14.68	23.16	32.58

Πίνακας 4.2: Λόγοι επιτάχυνσης για το σύστημα *Τάλως*

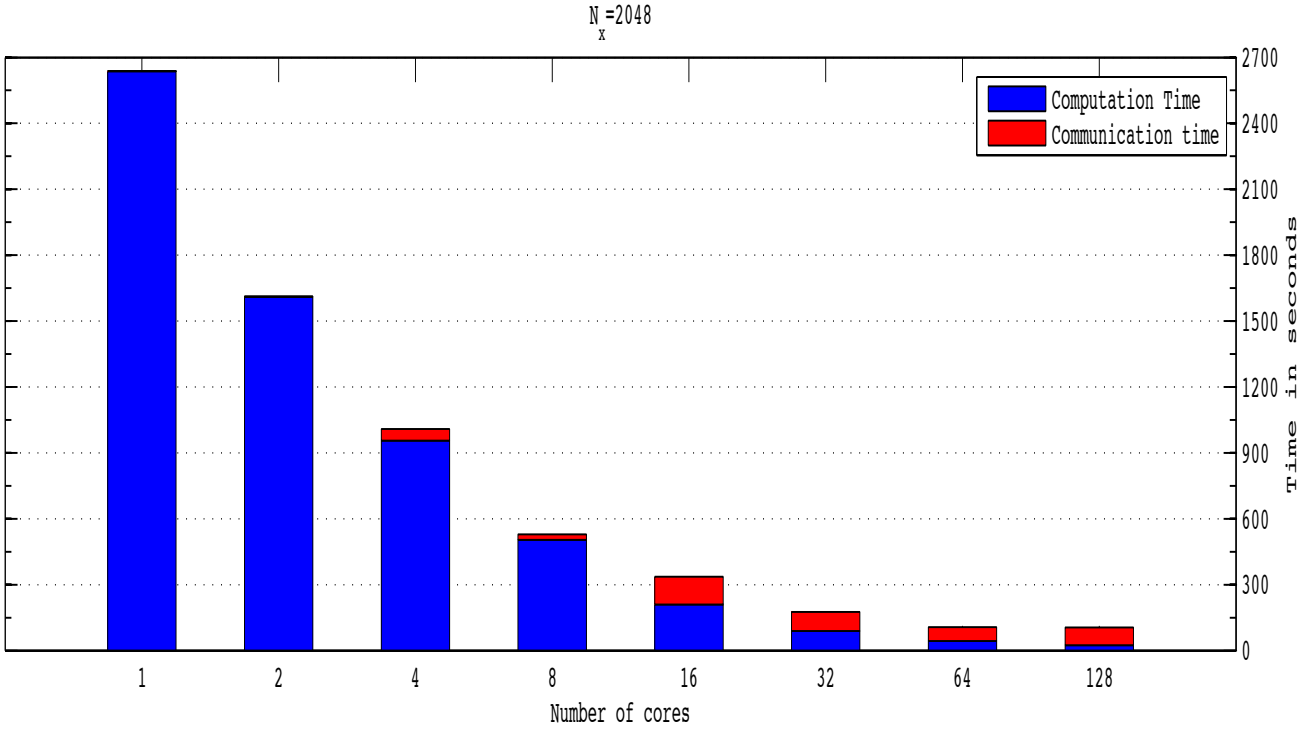


Σχήμα 4.3: Διάγραμμα λόγων επιτάχυνσης για το σύστημα *Τάλως*

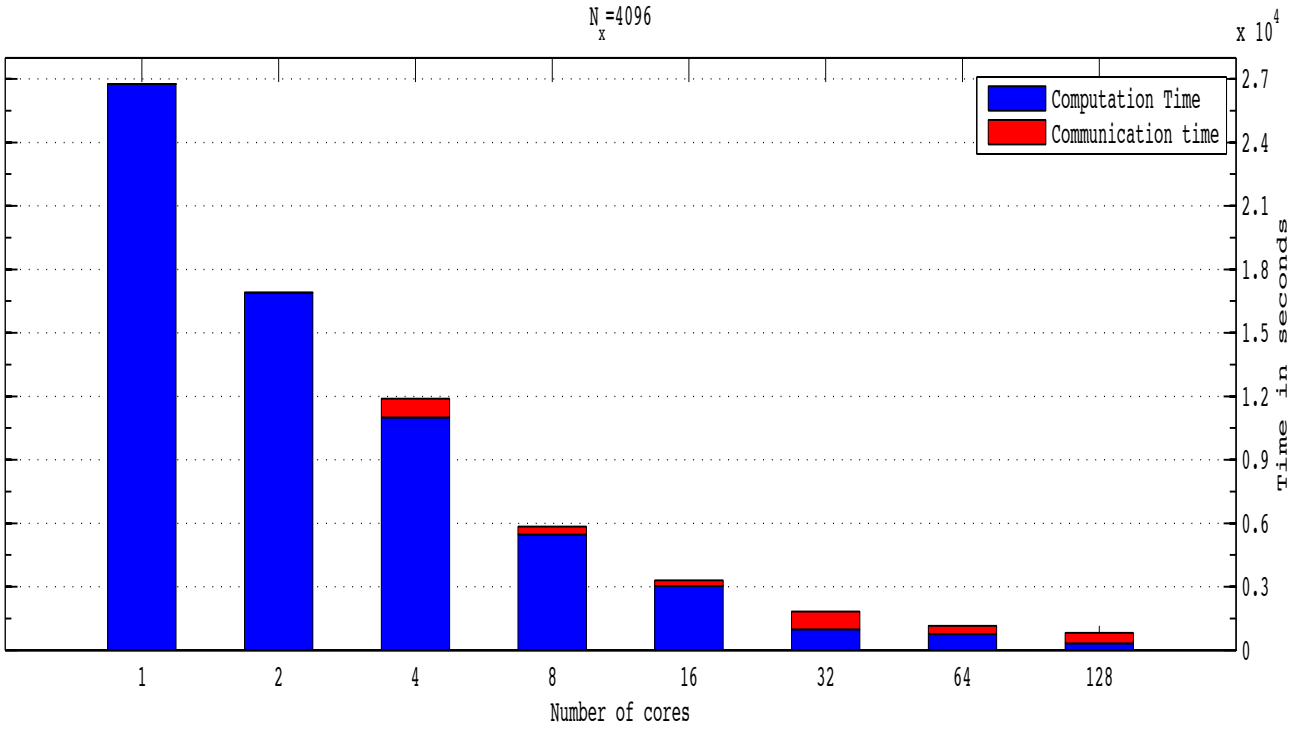
Τα επόμενα σχήματα 4.4,4.5 και 4.6 παρουσιάζουν σε μορφή διαγραμμάτων τη σχέση μεγέθους των χρόνων εκτέλεσης για υπολογισμούς και επικοινωνία ως προς το πλήθος των υπολογιστικών πυρήνων για διακριτοποίησεις μεγέθους $N_x = 1024, 2048$ και 4096 αντίστοιχα. Παρατηρούμε ότι ο χρόνος επικοινωνίας αυξάνει σημαντικά με τη χρήση περισσότερων υπολογιστικών πυρήνων, αλλά και απο τη μετάβαση στο εσωτερικό δίκτυο του μηχανήματατος για περιπτώσεις χρήσης πυρήνων πλήθους άνω των 4.



Σχήμα 4.4: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 1024$



Σχήμα 4.5: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 2048$



Σχήμα 4.6: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 4096$

4.3 Αποτελέσματα δοκιμών στο υπολογιστικό σύστημα *Πλειάδες*

Το πολυεπεξεργαστικό σύστημα *Πλειάδες* χρησιμοποιήθηκε για την επίλυση του ίδιου μεγέθους προβλημάτων με αυτά των δοκιμών της προηγούμενης ενότητας. Έτσι είναι εφικτή η σύγκριση της συμπεριφοράς υλοποίησης του παράλληλου αλγορίθμου μέχρι και την περίπτωση χρήσης 64 υπολογιστικών πυρήνων που διαθέτει αυτό το μηχάνημα.

Ο πίνακας T4.3 περιέχει τους χρόνους εκτέλεσης σε δευτερόλεπτα για όλα τα μεγέθη των προβλημάτων στην περίπτωση όπου το δίκτυο διασύνδεσης των υπολογιστικών κόμβων ήταν ταχύτητας 10Mbps. Απο αυτές τις χρονομετρήσεις προκύπτει ο πίνακας επιταχύνσεων T4.4, ενώ το σχήμα 4.7 εμφανίζει γραφικά τα μεγέθη όλων των επιταχύνσεων για τη χρήση μέχρι και 64 υπολογιστικών πυρήνων.

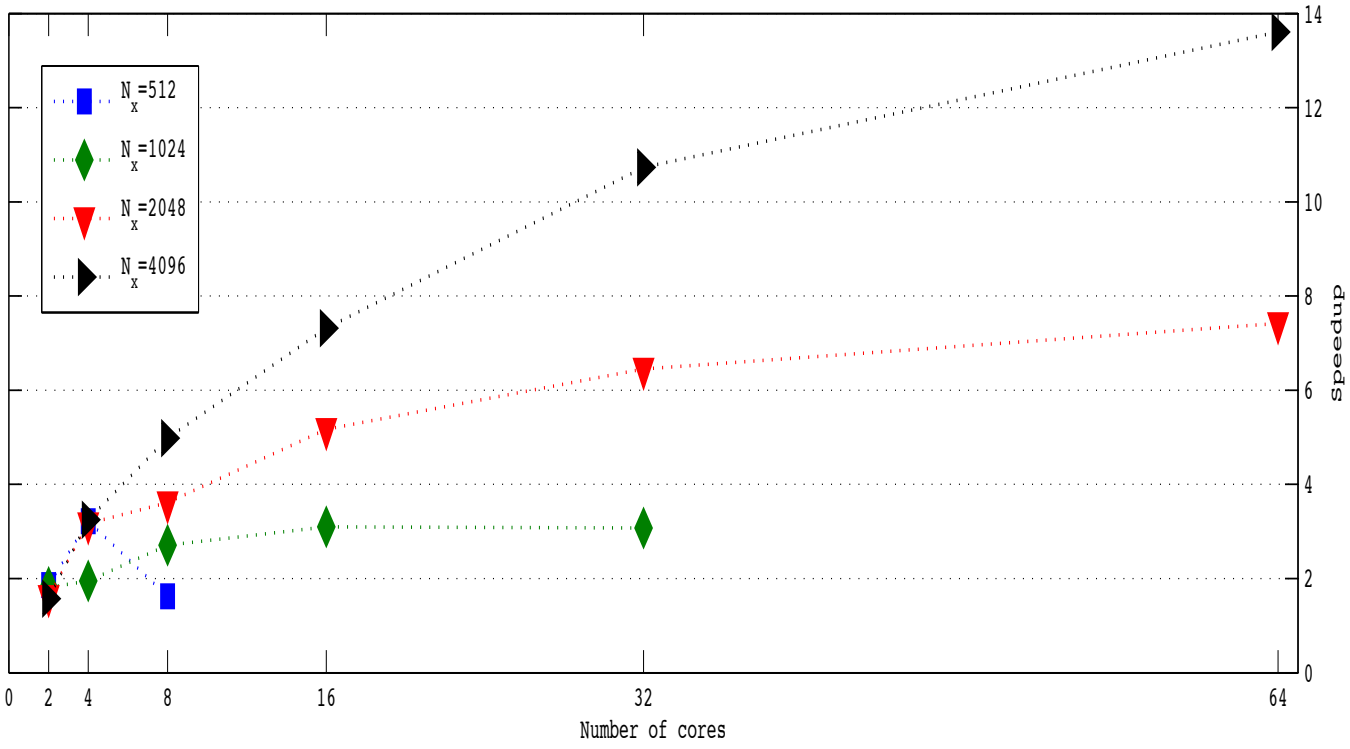
Τα σχήματα 4.8,4.9 και 4.10 παρουσιάζουν τις αναλογίες των χρόνων υπολογισμού και επικοινωνίας σε σχέση με το συνολικό χρόνο και το πλήθος των διαθέσιμων υπολογιστικών πυρήνων για προβλήματα μεγέθους $N_x = 1024, 2048$ και 4096. Παρατηρούμε ότι η χρήση αυτού του σχετικά αργού δικτύου των 10Mbps έχει ως αποτέλεσμα στη περίπτωση υλοποίησης σε μεγάλου πλήθους υπολογιστικών πυρήνων ο χρόνος επικοινωνίας να είναι μεγαλύτερος απο τον αντίστοιχο υπολογισμών.

N_x	cores	Communication Time	Computational Time	Total Time
128	1	-	4.526e-01	4.526e-01
	2	5.484e-03	2.762e-01	2.817e-01
	4	8.165e-03	1.676e-01	1.758e-01
	8	1.964e+00	1.180e-01	2.082e+00
256	1	-	3.974e+00	3.974e+00
	2	2.295e-02	2.181e+00	2.204e+00
	4	2.205e-02	1.119e+00	1.141e+00
	8	5.768e+00	6.745e-01	6.443e+00
512	1	-	3.506e+01	3.506e+01
	2	1.800e-01	1.859e+01	1.877e+01
	4	3.571e-01	1.052e+01	1.088e+01
	8	1.637e+01	5.266e+00	2.164e+01
	16	2.174e+01	2.752e+00	2.450e+01
1024	1	-	2.820e+02	2.820e+02
	2	1.790e+00	1.533e+02	1.550e+02
	4	4.520e+00	1.397e+02	1.443e+02
	8	5.797e+01	4.256e+01	1.039e+02
	16	6.948e+01	2.139e+01	9.087e+01
	32	7.997e+01	1.190e+01	9.187e+01
	64	8.394e+01	7.653e+00	9.160e+01
2048	1	-	3.005e+03	3.005e+03
	2	1.308e+01	1.835e+03	1.848e+03
	4	6.899e+00	9.398e+02	9.467e+02
	8	3.487e+02	4.822e+02	8.309e+02
	16	3.414e+02	2.393e+02	5.807e+02
	32	3.200e+02	1.457e+02	4.657e+02
	64	3.417e+02	6.276e+01	4.045e+02
4096	1	-	3.090e+04	3.090e+04
	2	2.492e+02	1.936e+04	1.961e+04
	4	1.515e+02	9.355e+03	9.507e+03
	8	1.553e+03	4.645e+03	6.198e+03
	16	1.825e+03	2.391e+03	4.216e+03
	32	1.297e+03	1.580e+03	2.877e+03
	64	1.461e+03	8.081e+02	2.270e+03

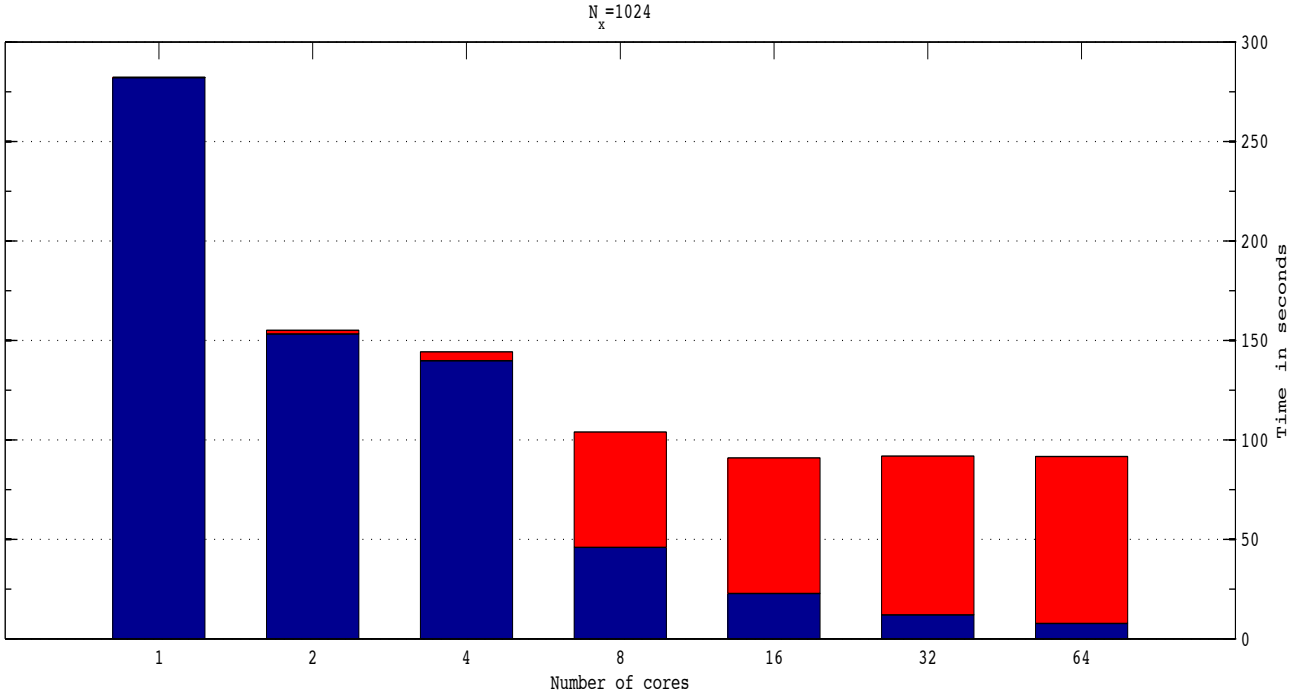
Table 4.3: Αριθμητικά αποτελέσματα για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 10 Mbps

Number of Cores :	2	4	8	16	32	64
$N_x = 512$	1.86	3.22	1.62			
$N_x = 1024$	1.81	1.95	2.71	3.10	3.07	
$N_x = 2048$	1.62	3.17	3.61	5.17	6.45	7.42
$N_x = 4096$	1.57	3.25	4.98	7.32	10.73	13.61

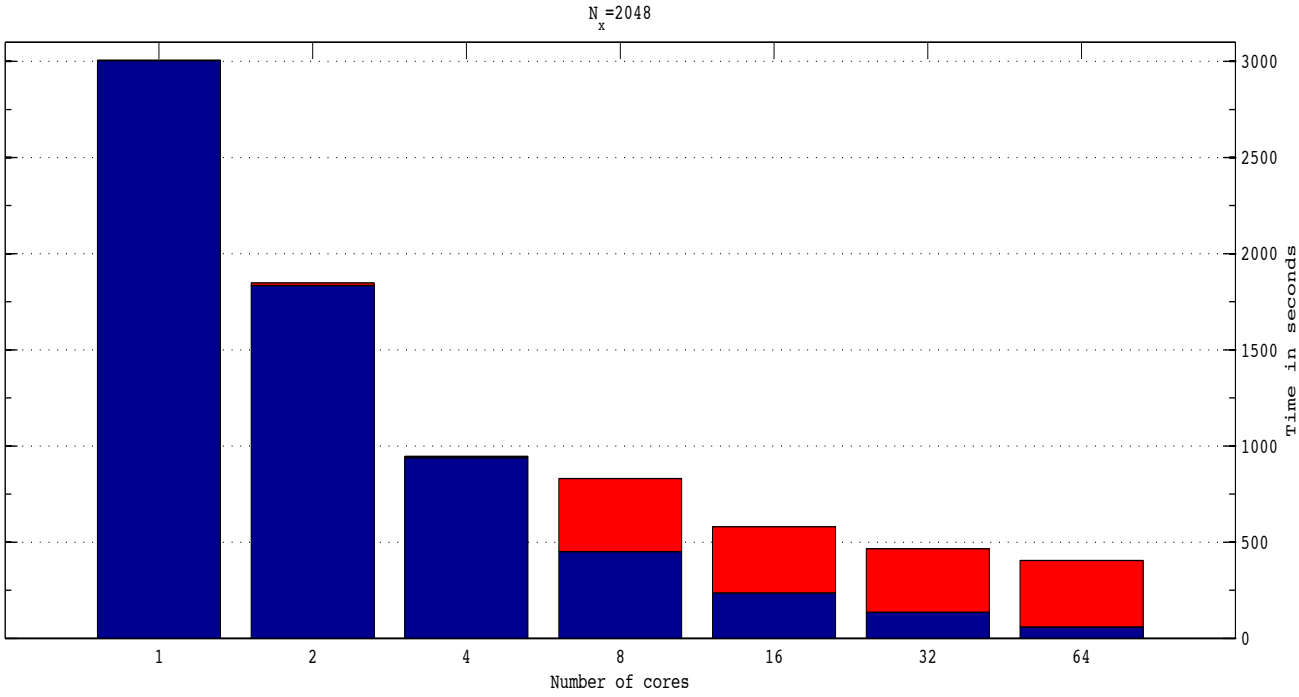
Πίνακας 4.4: Λόγοι επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 10Mbps



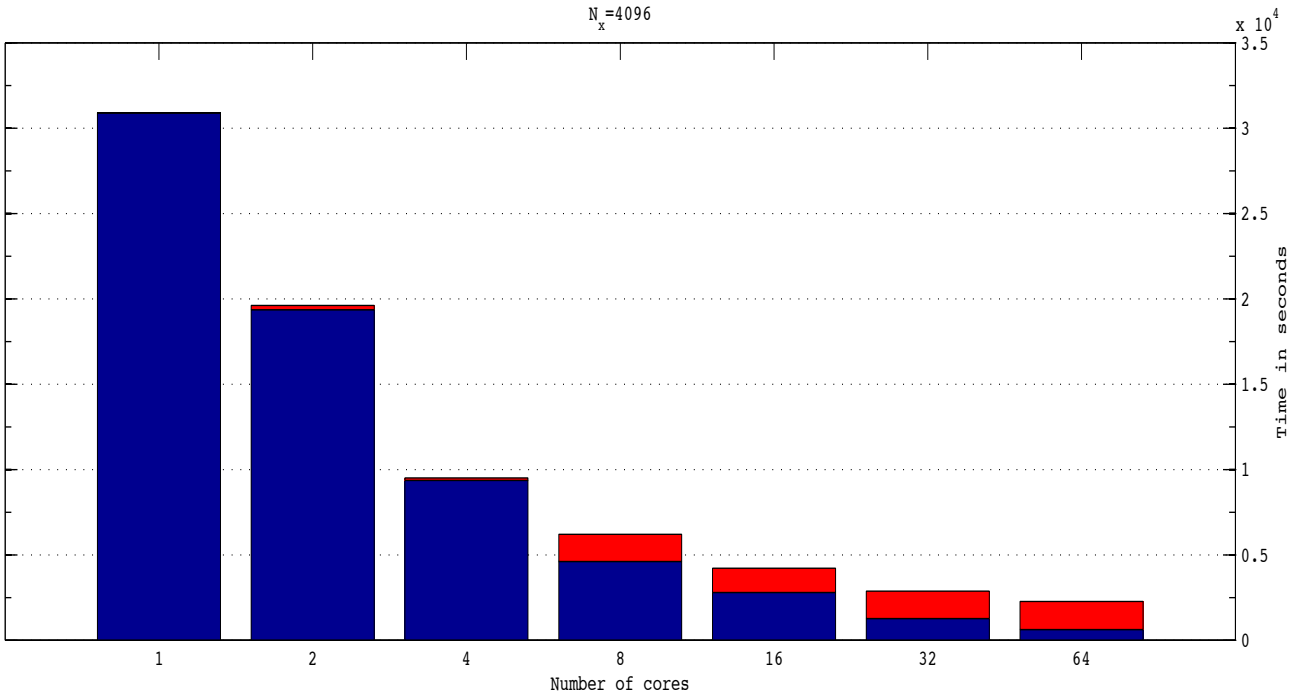
Σχήμα 4.7: Διάγραμμα λόγων επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 10Mbps



Σχήμα 4.8: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 1024$



Σχήμα 4.9: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 2048$



Σχήμα 4.10: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 4096$

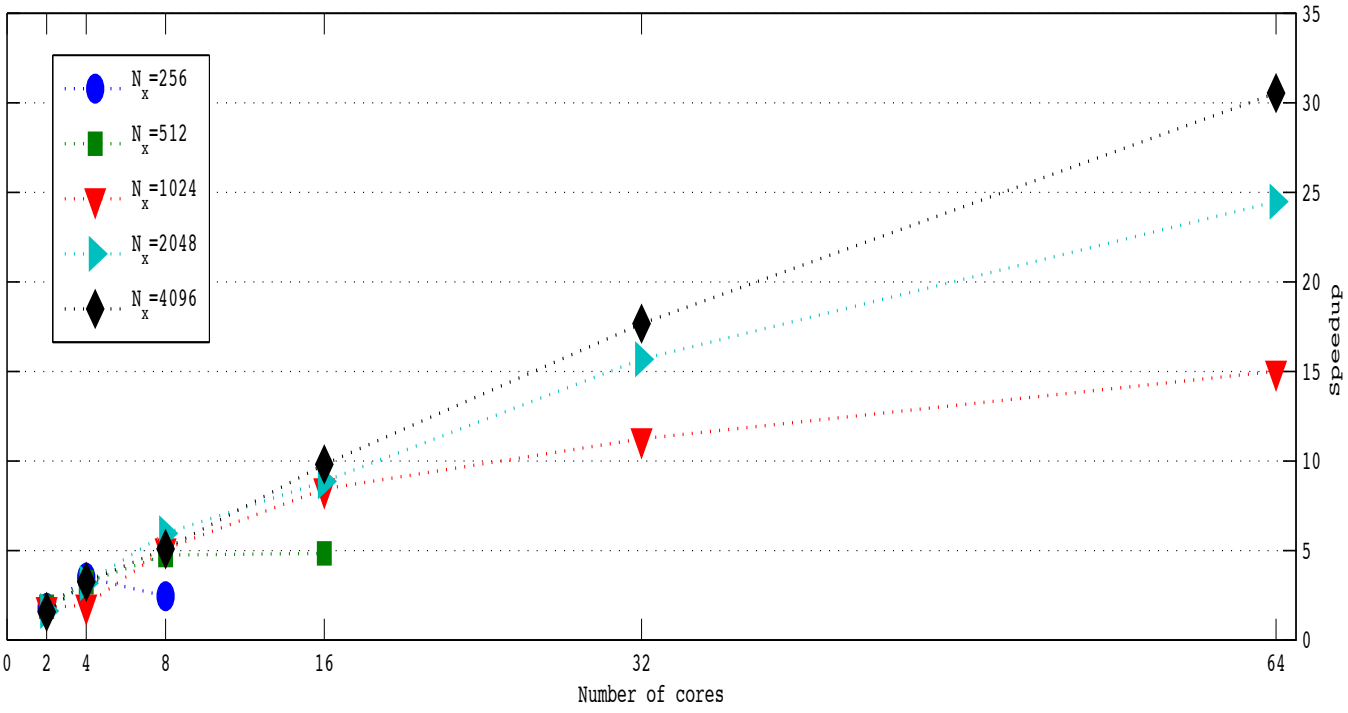
Ο πίνακας T4.5 περιλαμβάνει τις μετρήσεις χρόνων εκτέλεσης για τη περίπτωση όπου η ταχύτητα δικτύου διασύνδεσης των κόμβων του υπολογιστικού συστήματος ήταν 100Mbps. Έτσι οι μετρήσεις των χρόνων διαφοροποιούνται σε σχέση με τις αντίστοιχες μετρήσεις του δικτύου των 10Mbps για χρήση υπολογιστικών κόμβων πλήθους άνω των 4. Ο πίνακας T4.6 παρουσιάζει τις επιταχύνσεις και το σχήμα 4.11 τις αναπαριστά σε γράφημα. Τα σχήματα 4.12,4.13 και 4.14 εμφανίζουν την αναλογία των χρόνων υπολογισμού και επικοινωνίας για τις τρείς μεγαλύτερες σε μέγεθος δοκιμές που πραγματοποιήθηκαν.

N_x	cores	Communication Time	Computational Time	Total Time
128	1	-	4.526e-01	4.526e-01
	2	5.484e-03	2.762e-01	2.817e-01
	4	8.165e-03	1.676e-01	1.758e-01
	8	2.979e-01	1.180e-01	4.159e-01
256	1	-	3.974e+00	3.974e+00
	2	2.295e-02	2.181e+00	2.204e+00
	4	2.205e-02	1.119e+00	1.141e+00
	8	7.355e-01	6.745e-01	1.624e+00
512	1	-	3.506e+01	3.506e+01
	2	1.800e-01	1.859e+01	1.877e+01
	4	3.571e-01	1.052e+01	1.088e+01
	8	2.128e+00	5.266e+00	7.394e+00
	16	4.491e+00	2.752e+00	7.243e+00
1024	1	-	2.820e+02	2.820e+02
	2	1.790e+00	1.533e+02	1.550e+02
	4	4.520e+00	1.397e+02	1.443e+02
	8	1.291e+01	4.256e+01	5.547e+01
	16	1.201e+01	2.139e+01	3.340e+01
	32	1.318e+01	1.190e+01	2.508e+01
	64	1.112e+01	7.653e+00	1.878e+01
2048	1	-	3.005e+03	3.005e+03
	2	1.308e+01	1.835e+03	1.848e+03
	4	6.899e+00	9.398e+02	9.467e+02
	8	2.240e+01	4.822e+02	5.046e+02
	16	4.580e+01	2.932e+02	3.390e+02
	32	4.800e+01	1.437e+02	1.917e+02
	64	5.994e+01	6.276e+01	1.227e+02
4096	1	-	3.090e+04	3.090e+04
	2	2.492e+02	1.936e+04	1.961e+04
	4	1.515e+02	9.355e+03	9.507e+03
	8	1.416e+03	4.645e+03	6.061e+03
	16	7.600e+02	2.391e+03	3.151e+03
	32	1.690e+02	1.580e+03	1.749e+03
	64	2.029e+02	8.081e+02	1.011e+03

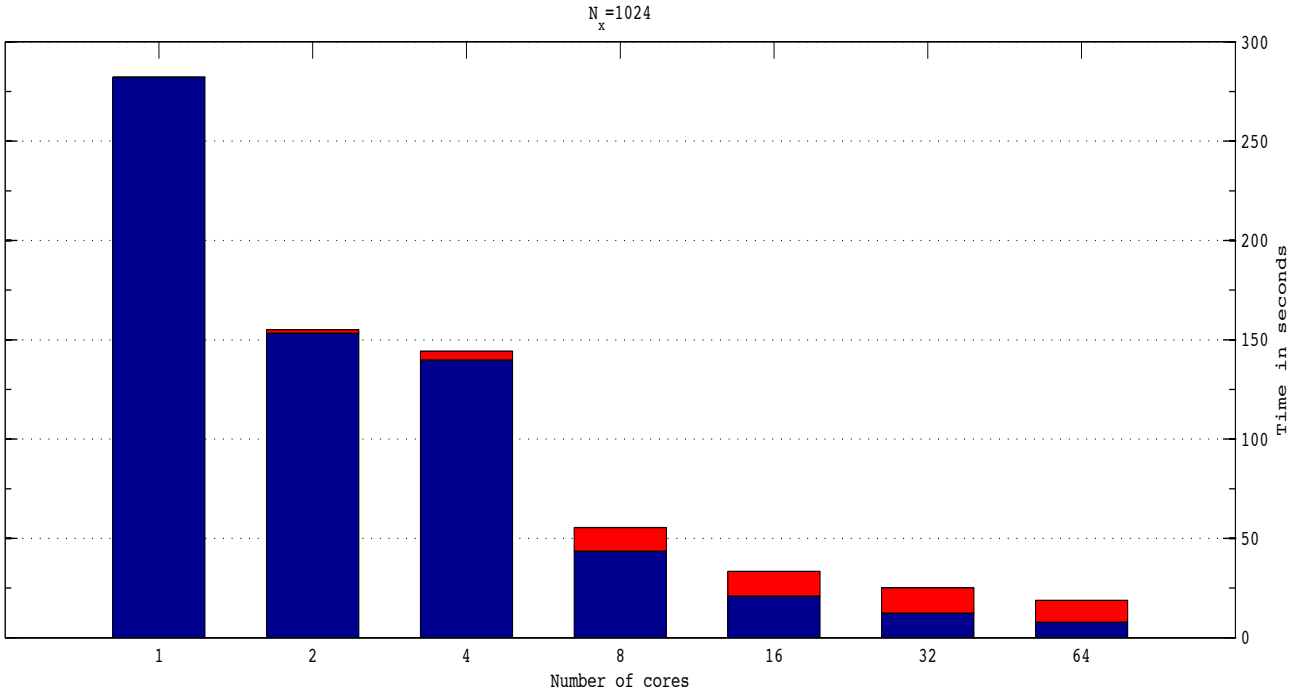
Table 4.5: Αριθμητικά αποτελέσματα για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 100 Mbps

Number of Cores :	2	4	8	16	32	64
$N_x = 256$	1.80	3.48	2.44			
$N_x = 512$	1.86	3.22	4.74	4.84		
$N_x = 1024$	1.81	1.95	5.08	8.44	11.24	15.01
$N_x = 2048$	1.62	3.17	5.95	8.86	15.67	24.48
$N_x = 4096$	1.57	3.25	5.09	9.80	17.66	30.54

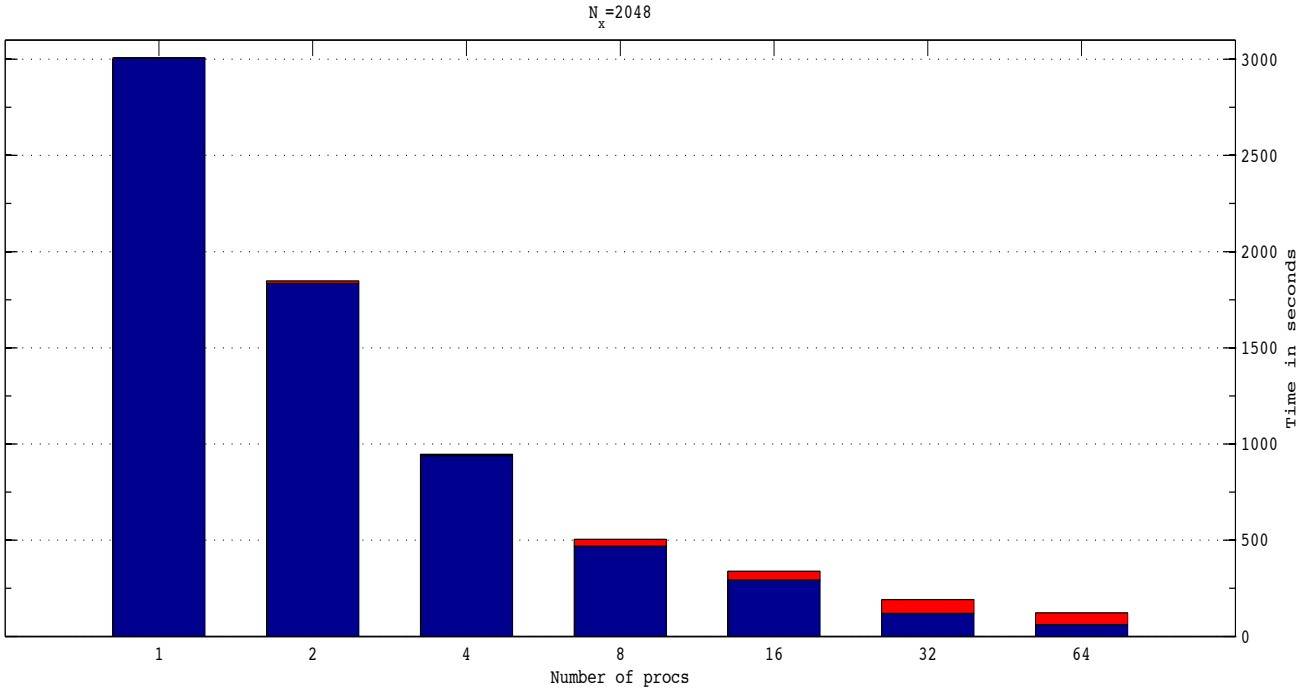
Πίνακας 4.6: Λόγοι επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 100Mbps



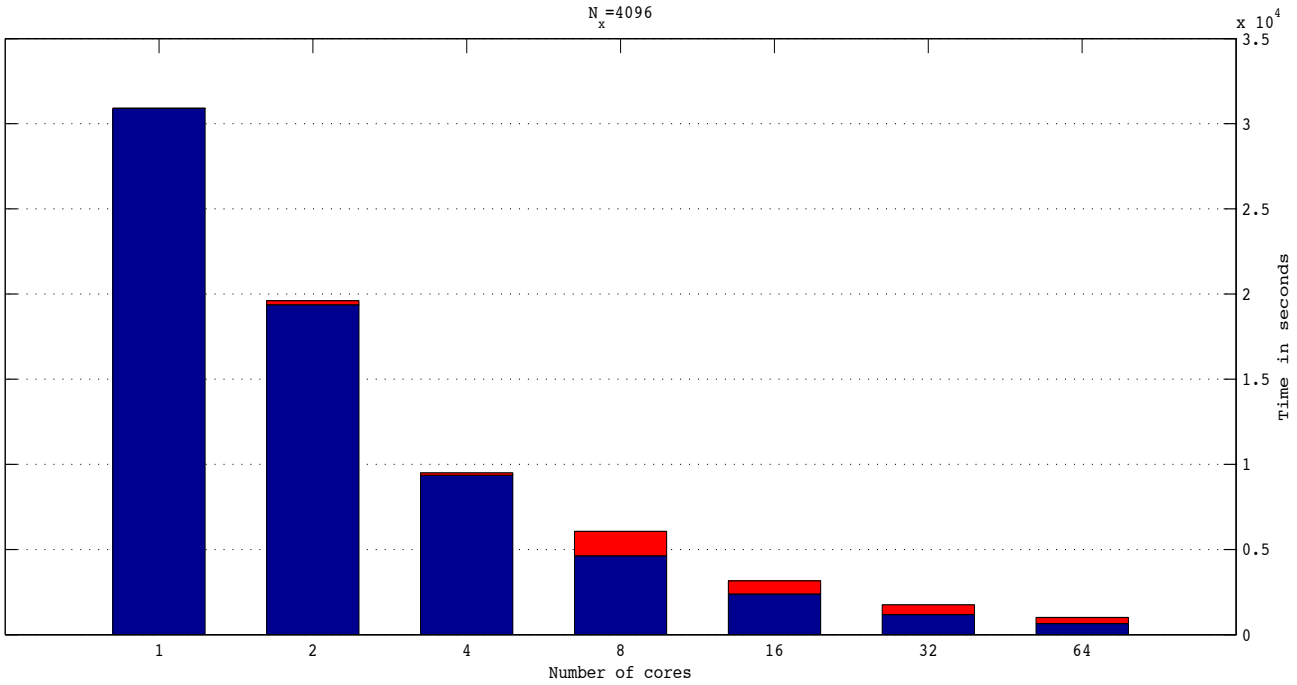
Σχήμα 4.11: Διάγραμμα λόγων επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 100Mbps



Σχήμα 4.12: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 1024$



Σχήμα 4.13: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 2048$



Σχήμα 4.14: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 4096$

Στην συνέχεια παρουσιάζονται τα αποτελέσματα των δοκιμών για τη περίπτωση χρήσης δικτύου ταχύτητας 1Gbps στην διασύνδεση των υπολογιστικών κόμβων. Ειδικότερα ο πίνακας T4.7 εμφανίζει τις χρονομετρήσεις των δύο φάσεων του παράλληλου αλγορίθμου, ενώ ο πίνακας T4.8 τις επιταχύνσεις από τη χρήση μέχρι 64 υπολογιστικών πυρήνων.

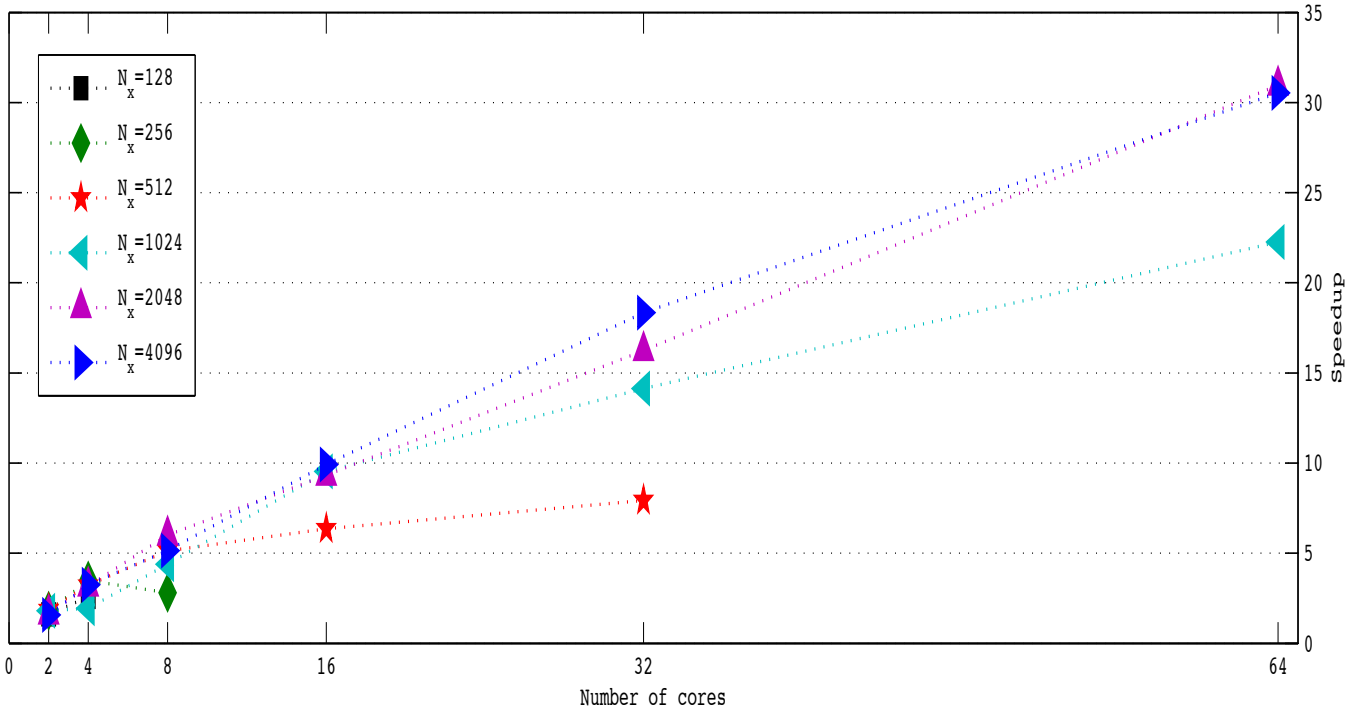
Το σχήμα 4.15 αναπαριστά γραφικά αυτές τις επιταχύνσεις για το σύνολο των δοκιμών. Τα γραφίματα 4.16,4.17 και 4.18 εμφανίζουν την συσχέτιση των χρόνων των δύο βασικών διαδικασιών του αλγορίθμου, δηλαδή της υπολογιστικής και επικοινωνιακής διαδικασίας για τη περίπτωση των τριών μεγαλύτερων σε μέγεθος προβλημάτων.

N_x	cores	Communication Time	Computational Time	Total Time
128	1	-	4.526e-01	4.526e-01
	2	5.484e-03	2.762e-01	2.817e-01
	4	8.165e-03	1.676e-01	1.758e-01
	8	3.482e-01	1.180e-01	4.663e-01
256	1	-	3.974e+00	3.974e+00
	2	2.295e-02	2.181e+00	2.204e+00
	4	2.205e-02	1.119e+00	1.141e+00
	8	7.363e-01	6.745e-01	1.410e+00
512	1	-	3.506e+01	3.506e+01
	2	1.800e-01	1.859e+01	1.877e+01
	4	3.571e-01	1.052e+01	1.088e+01
	8	1.562e+00	5.266e+00	6.828e+00
	16	2.769e+00	2.752e+00	5.522e+00
1024	1	-	2.820e+02	2.820e+02
	2	1.790e+00	1.533e+02	1.550e+02
	4	4.520e+00	1.397e+02	1.443e+02
	8	2.168e+01	4.256e+01	6.424e+01
	16	8.170e+00	2.139e+01	2.956e+01
	32	8.048e+00	1.190e+01	1.995e+01
	64	5.015e+00	7.653e+01	1.266e+01
2048	1	-	3.005e+03	3.005e+03
	2	1.308e+01	1.835e+03	1.848e+03
	4	6.899e+00	9.398e+02	9.467e+02
	8	1.743e+01	4.822e+02	4.997e+02
	16	8.245e+01	2.393e+02	3.218e+02
	32	4.119e+01	1.437e+02	1.849e+02
	64	3.424e+01	6.276e+01	9.701e+01
4096	1	-	3.090e+04	3.090e+04
	2	2.492e+02	1.936e+04	1.961e+04
	4	1.515e+02	9.355e+03	9.507e+03
	8	1.357e+03	4.645e+03	6.003e+03
	16	7.202e+02	2.391e+03	3.111e+03
	32	1.043e+02	1.580e+03	1.684e+03
	64	1.569e+02	8.081e+02	9.650e+02

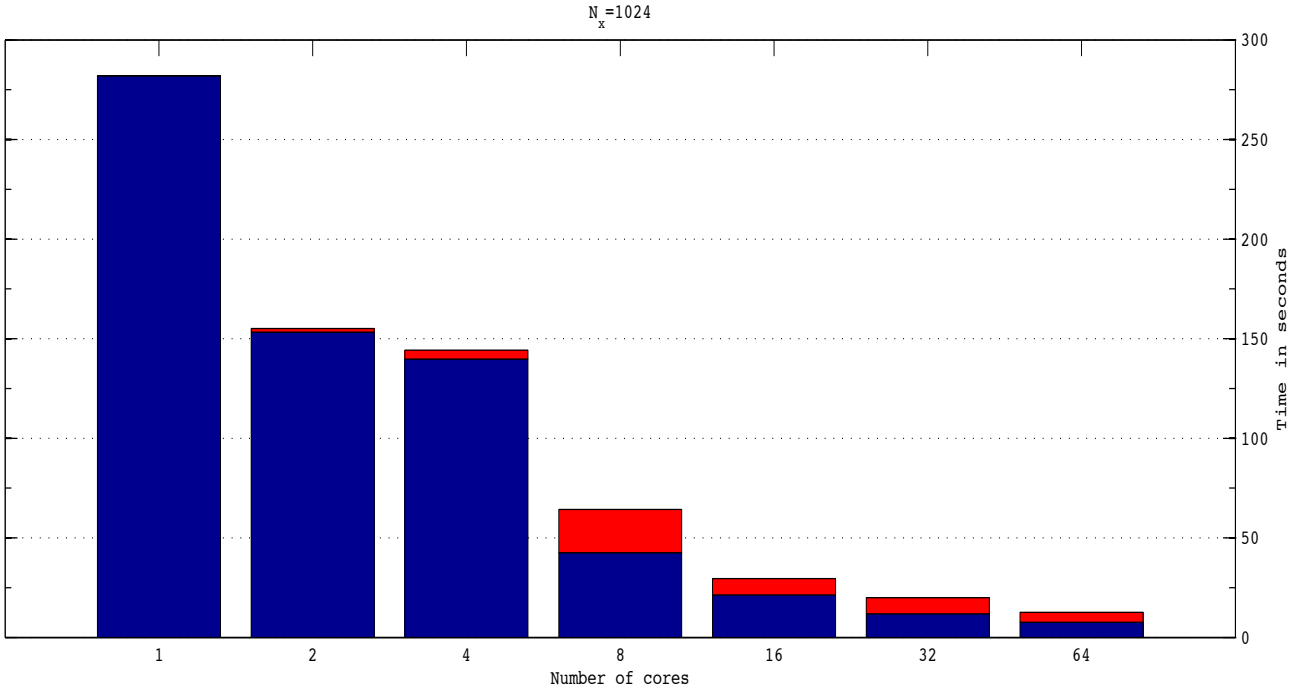
Table 4.7: Αριθμητικά αποτελέσματα για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 1Gbps

Number of Cores :	2	4	8	16	32	64
$N_x = 128$	1.60	2.57				
$N_x = 256$	1.80	3.48	2.81			
$N_x = 512$	1.86	3.22	5.13	6.35	7.92	
$N_x = 1024$	1.81	1.95	4.39	9.54	14.13	22.26
$N_x = 2048$	1.62	3.17	6.01	9.33	16.24	30.97
$N_x = 4096$	1.57	3.25	5.14	9.92	18.34	32.01

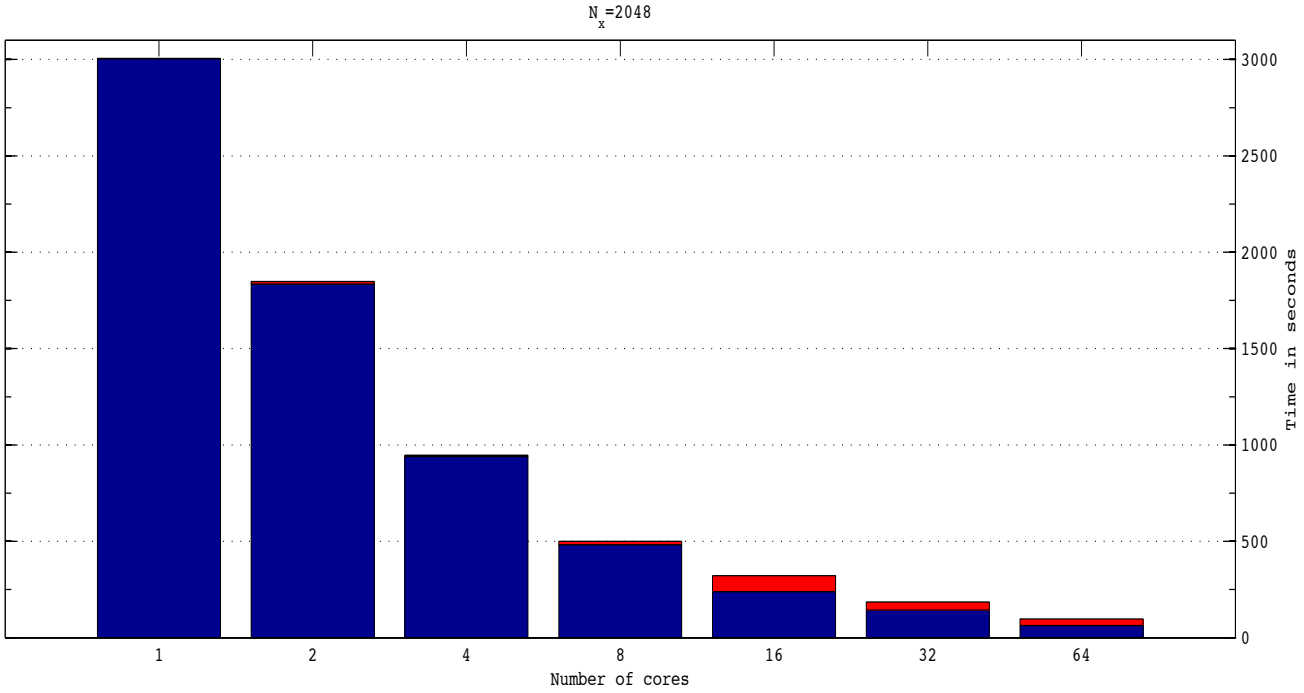
Πίνακας 4.8: Λόγοι επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 1Gbps



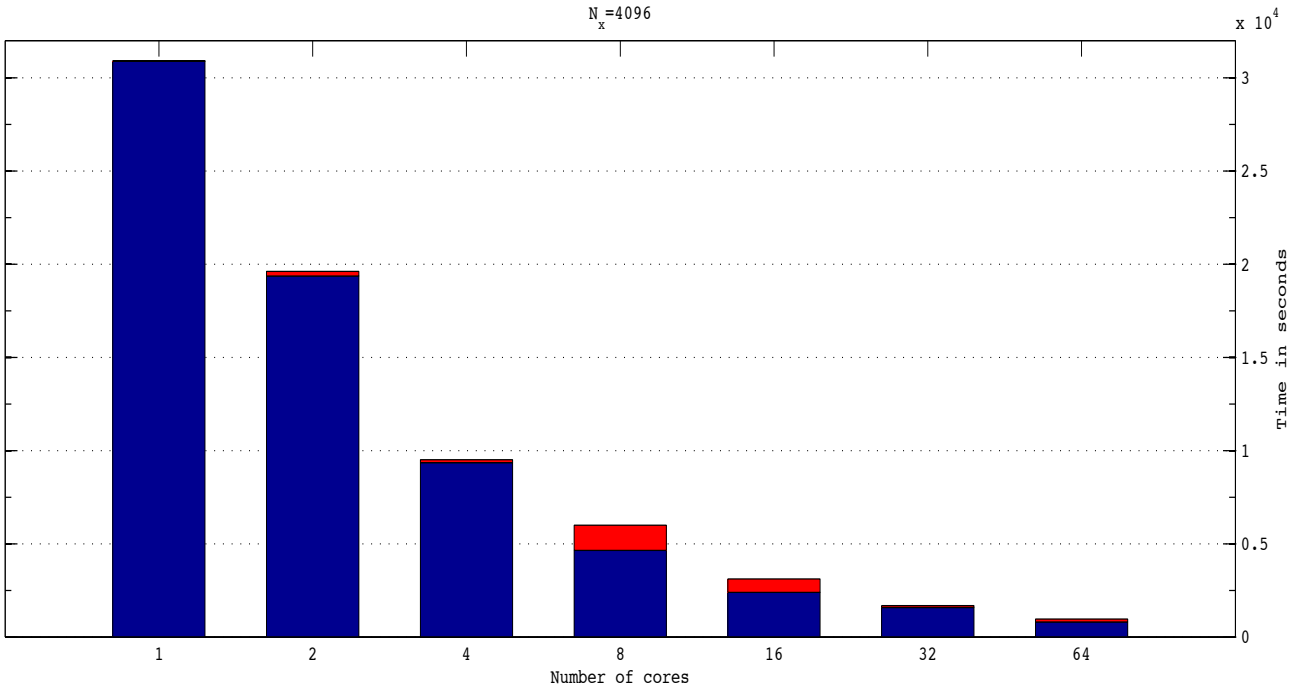
Σχήμα 4.15: Διάγραμμα λόγου επιτάχυνσης για το σύστημα *Πλειάδες* με ταχύτητα διασύνδεσης 1Gbps



Σχήμα 4.16: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 1024$



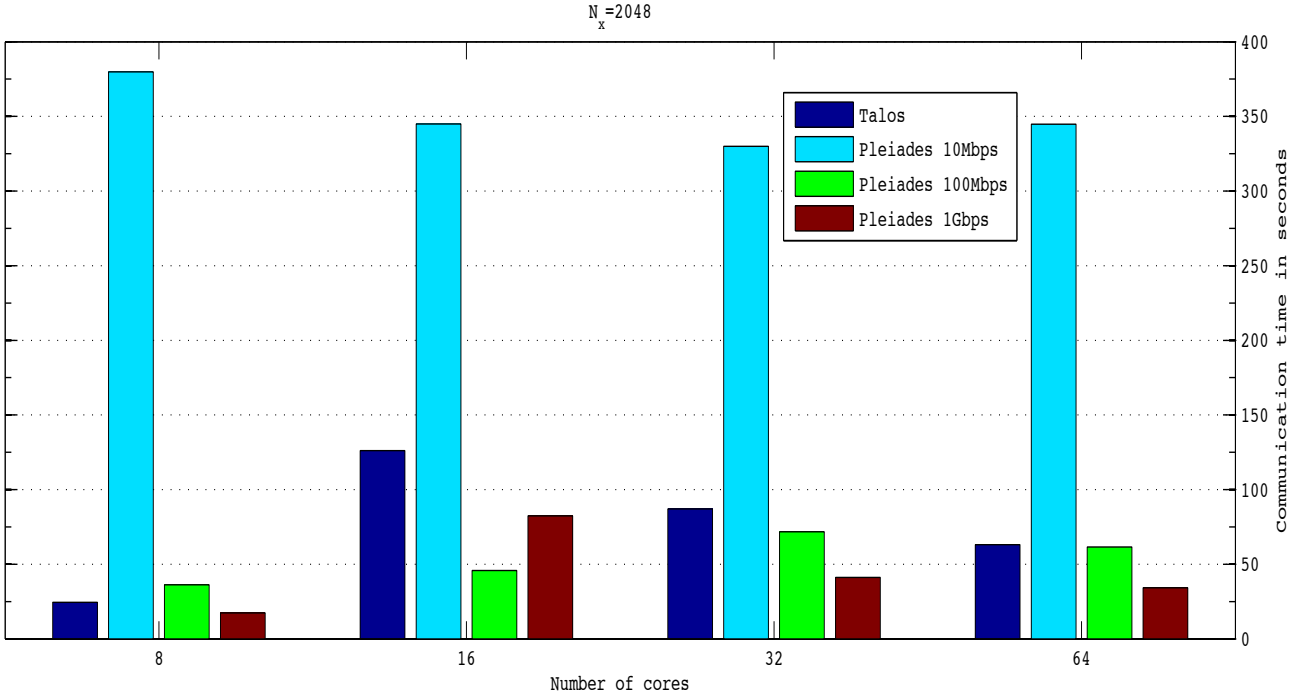
Σχήμα 4.17: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 2048$



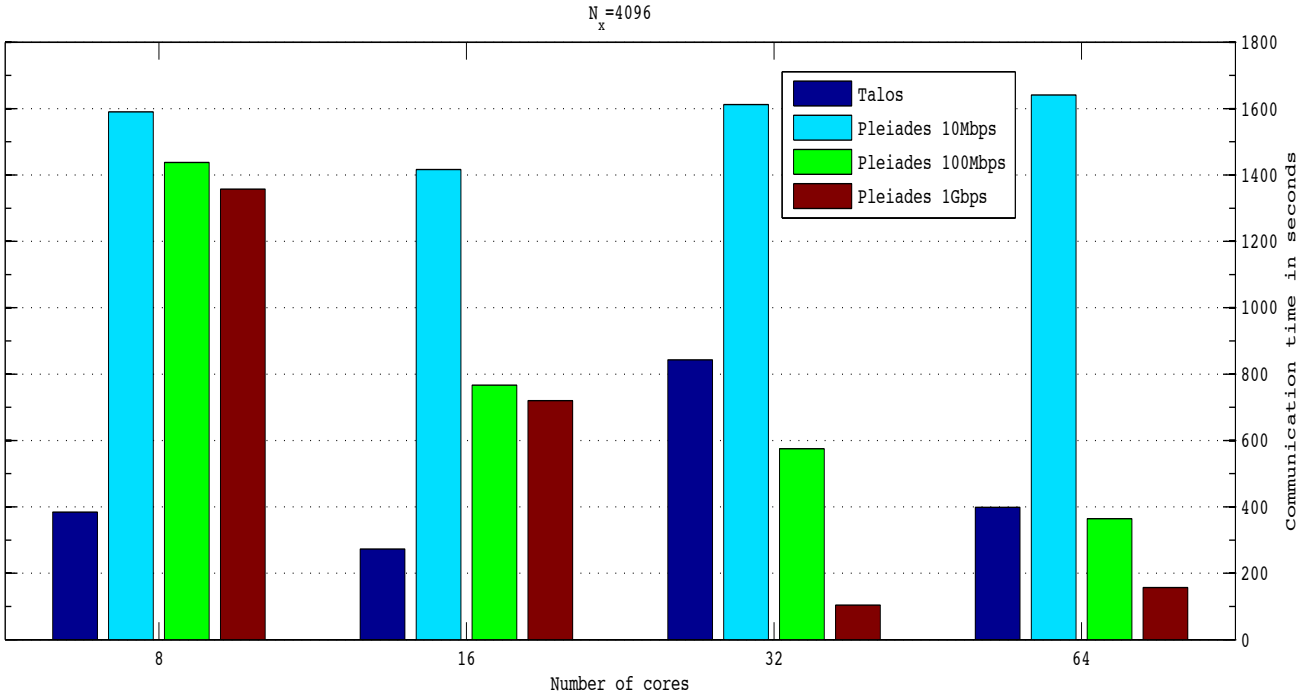
Σχήμα 4.18: Συγκριτικό διάγραμμα χρόνου επικοινωνίας/υπολογισμού $N_x = 4096$

Επειδή τα υπολογιστικά συστήματα *Τάλως* και *Πλειάδες* αποτελούνται απο σχεδόν ίδιου τύπου και χαρακτηριστικών επεξεργαστές, διαθέτουν ίδιας κατηγορίας λειτουργικό σύστημα, ενώ και η διασύνδεση τους είναι παραπλήσια, είναι εφικτή η σύγκριση των μετρήσεων ανάμεσα τους. Έτσι τα σχήματα 4.19 και 4.20 εμφανίζουν γραφικά την επιβάρυνση του παράλληλου αλγορίθμου με το επικοινωνιακό κόστος. Ειδικότερα το σχήμα 4.19 παρουσιάζει το χρόνο επικοινωνίας που μετρήθηκε για τη χρήση 8,16,32 και 64 υπολογιστικών πυρήνων στο πρόβλημα μεγέθους $N_x = 2048$ σε όλα τα υπολογιστικά συστήματα. Το σχήμα 4.20 είναι το αντίστοιχο για $N_x = 4096$. Παρατηρούμε ότι, όπως άλλωστε αναμενόταν, ότι η μεγαλύτερη τιμή υπάρχει στο πιο αργό δίκτυο δηλαδή των 10Mbps. Η εμφάνιση μη παραπλήσιας συμπεριφοράς στις χρονομετρήσεις του κόστους επικοινωνίας ανάμεσα στο δίκτυο ταχύτητας 1Gbps του συστήματος *Πλειάδες* και του συστήματος *Τάλως*, οφείλεται στο γεγονός ότι αρκετοί υπολογιστικοί του κόμβοι χρειάστηκε να επικοινωνήσουν με τους υπόλοιπους που βρίσκονταν σε άλλη συστοιχία και η επικοινωνία μεταξύ των συστοιχιών γινόντα από μόνο μία κάρτα δικτύου ταχύτητας 1Gbps, σε αντίθεση με το σύστημα *Πλειάδες* όπου κάθε υπολογιστικός κόμβος επικοινωνεί με τους υπόλοιπους μέσω δικής του κάρτας.

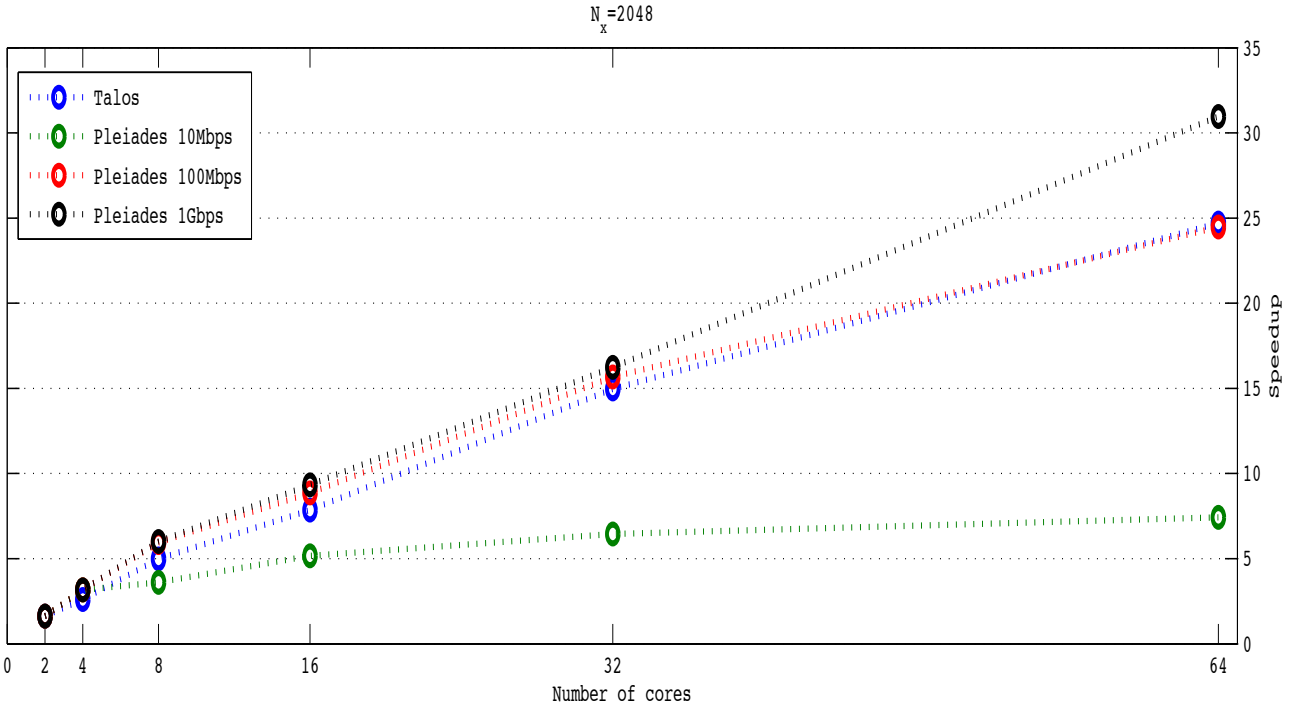
Τα σχήματα 4.21 και 4.22 εμφανίζουν τις επιταχύνσεις συνολικά και για τα δύο συστήματα και μέχρι 64 υπολογιστικούς πυρήνες που είναι δυνατή η σύγκριση ανάμεσα τους, για τις δύο μεγαλύτερες σε μέγεθος δοκιμές. Η ποσοστιαία επιτάχυνση κυμαίνεται κοντά στο θεωρητικό βέλτιστο για χρήση μέχρι 32 υπολογιστικών πυρήνων και λίγο πάνω απο το 50% για 64.



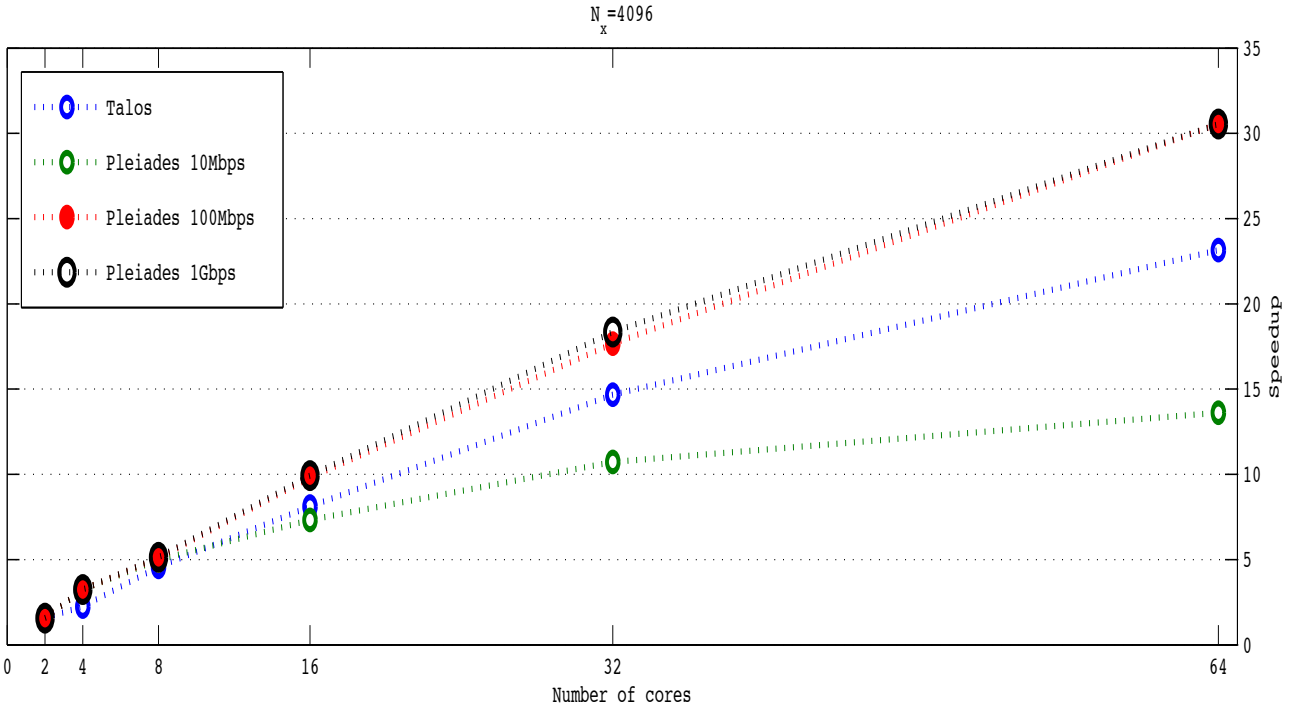
Σχήμα 4.19: Χρόνος διαδικασίας επικοινωνίας για $N_x = 2048$



Σχήμα 4.20: Χρόνος διαδικασίας επικοινωνίας για $N_x = 4096$



Σχήμα 4.21: Μετρήσεις speedup για $N_x = 2048$



Σχήμα 4.22: Μετρήσεις speedup για $N_x = 4096$

Κεφάλαιο 5

Συμπεράσματα

Σε αυτή τη διατριβή επιλύθηκε ένα ελλειπτικό πρόβλημα συνοριακών τιμών με την εφαρμογή της αριθμητικής μέθοδου συμπαγών πεπερασμένων διαφορών σε παραλληλογράμμου σχήματος χωρία. Πραγματοποιήθηκε μελέτη της συμπεριφοράς σύγκλισης των δημοφιλέστερων επαναληπτικών μεθόδων και η ταχύτερη υλοποιήθηκε σε δικτυακά υπολογιστικά περιβάλλοντα. Η υλοποίηση της μεθόδου βασίστηκε στην εφαρμογή τεχνικών αύξησης των παράλληλων ιδιοτήτων του παραγόμενου αραιού και γενικού γραμμικού συστήματος. Η κατασκευή του παράλληλου αλγορίθμου βασίστηκε στις απαιτήσεις των σύγχρονων δικτυακών αρχιτεκτονικών διεξαγωγής επιστημονικών υπολογισμών.

Η υλοποίηση της εφαρμογής πραγματοποιήθηκε σε δύο διαφορετικού τύπου δικτυακά μηχανήματα και έγινε μελέτη της συμπεριφοράς απόδοσης σε αυτά. Το πρώτο είναι ένα υπολογιστής πλέγματος ο οποίος διαθέτει κοινή θέση αποθήκευσης για όλους τους υπολογιστικούς κόμβους του, ενώ η διασύνδεση κόμβων που είναι εγκατεστημένοι στην ίδια συστάδα είναι πολύ γρήγορης ταχύτητας, σε αντίθεση με τη διασύνδεση συστάδων που είναι υποδεκαπλάσιας. Το δεύτερο μηχάνημα αποτελείται απο ανεξάρτητους υπολογιστικούς κόμβους οι οποίοι είναι συνδεδεμένοι σε ένα δίκτυο μεταβλητής ταχύτητας. Έτσι ήταν εφικτή η μελέτη της επίδρασης της ταχύτητας του δικτύου στο κόστος επικοινωνίας του παράλληλου αλγορίθμου.

Απο τη μελέτη της συμπεριφοράς της υλοποίησης τόσο του σειριακού,όσο και του παράλληλου αλγορίθμου προέκυψαν τα επόμενα συμπεράσματα :

- Η επαναληπτική μέθοδος του υπολοίπου Schur μπορεί να επιλύσει ταχύτερα το γραμμικό σύστημα της αριθμητικής μεθόδου των πεπερασμένων συμπαγών σχημάτων για την επίλυση ελλειπτικών προβλημάτων συνοριακών τιμών.

- Η αρίθμηση βασισμένη στο zebra (άσπρο-μαύρο) σχήμα οδηγεί στην αύξηση των παράλληλων ιδιοτήτων του αραιού γραμμικού συστήματος, επιτρέποντας την ανεξαρτητοποίηση ομάδων αγνώστων, οι οποίοι μπορούν να υπολογιστούν ανεξάρτητα μεταξύ τους μέσω του επαναληπτικού αλγορίθμου του υπολοίπου Schur.

- Η κατασκευή και υλοποίηση παράλληλου αλγορίθμου μπορεί να μειώσει το χρόνο επίλυσης πολύ κοντά στο θεωρητικό βέλτιστο για χρήση λίγων υπολογιστικών πυρήνων, ενώ με απόδοση που πλησιάζει το $50\% - 70\%$ για δεκάδες και το 25% για εκατοντάδες σε διακριτοποιήσεις της τάξης 1 προς 4000, όπου απαιτείται ο υπολογισμός 17 εκατομμυρίων (2^{24}) αγνώστων.

Απο τα παραπάνω συμπεράσματα προκύπτει ότι η χρήση δικτυακών υπολογιστικών συστημάτων επιτρέπει την επίλυση ρεαλιστικών προβλημάτων συνοριακών τιμών, τα οποία απαντούνται σε σύγχρονες εφαρμογές της μηχανικής, όπως στην αεροδιαστημική και στις μεταφορές (πλοία, τρένα υψηλών ταχυτήτων) σε αποδεκτό χρονικό διάστημα. Η επιπλέον χρήση σύγχρονων υπολογιστικών μέσων, όπως οι κάρτες γραφικών υπολογισμών (GPU's), σε αυτά τα δικτυακά περιβάλλοντα υπολογισμών θα μπορούσε να μειώσει ακόμα περισσότερο το χρόνο επίλυσης των γραμμικών συστημάτων και κατά συνέπεια των διαφορικών εξισώσεων. Για αυτό θα χρειαστεί να επεκταθεί και να τροποποιηθεί κατάλληλα ο παράλληλος επαναλαπτικός αλγόριθμος που κατασκευάστηκε σε αυτή τη διατριβή.

Παράρτημα

Α΄.1 Σειριακός Κώδικας

main.f

```
parameter (nx=4096,ny=nx,n=nx*ny,n2=n/2)
implicit real*8 (a-h,o-z)
include 'mpif.h'
real*8 b(n),x(n),a1(5,nx),a1t(5,nx),
+ a0(7,nx),a0h(5,nx),a0t(7,nx),
+ a0o(5,nx),a0to(5,nx),xi(nx),yi(ny),
+ xe(n),r(n2),rh(n2),pi(n2),ph(n2),s(n2),sh(n2),
+ resid,t(n2),error,temp,ui(n2),w(n),hl,MPI_Wtime
integer ipvta0(nx),ipvta0t(nx)

print*,'========================================== '
print*,' '
print*,'Serial Schur Complement  for  Nx=Ny= ',nx
tm=MPI_Wtime()
hl=1.0d0 ! parameter L
c ***domain parameters***
xL=1.0d0
yL=1.0d0
g=1.0d0 !(g=hx/hy)
iter=9159
resid=5.0d-18
call start(nx,ny,xL,yL,g,a0,a0h,a0t,a1,a1t,ipvta0,
+ ipvta0t,xi,yi,a0o,a0to,hl)
call makeb(nx,ny,xL,yL,g,b,xi,yi,hl)
call dcopy(n,b,1,x,1)
call Schur(nx,x,b,a1,a1t,a0o,a0to,a0h,a0t,
+ a0,resid,r,rh,pi,ph,t,s,sh,ui,iter,
+ ipvta0,ipvta0t,w,tm,t1,t2)
tm=MPI_Wtime()-tm
print*,'Iter. for Schur Comp. with Bi-CGSTAB =',iter
```

```
print*,'Error of Bi-CGSTAB =',resid
print*,'Time for 1-3 steps =',t1,'  seconds'
print*,'Time for Bi-CGSTAB =',t2,'  seconds'
print*,'Time for 5-7 steps =',tm,'  seconds'
print*,'Total Time =',tm+t1+t2,'  seconds'
call dcopy(n,x,1,b,1)
call inv_line_wb(nx,b,x)
k=1
do j=1,ny
  do i=1,nx
  xe(k)=u(xi(i),yi(j))
  k=k+1
  enddo
enddo
dm=0.0d0
do i=1,n
  xe(i)=dabs(xe(i)-x(i))
  if (xe(i).gt.dm) dm=xe(i)
enddo
open(11,file='x')
open(12,file='y')
open(13,file='ux')
open(14,file='dux')
333    format(f18.15)
do i=1,nx
  write(11,333) xi(i)
enddo
close(11)
k=1
do j=1,ny
  write(12,333) yi(j)
  do i=1,nx
    write(14,333) xe(k)
    write(13,333) x(k)
    k=k+1
  enddo
enddo
close(14)
close(12)
close(13)
call line_wb(nx,x,xe)
call matvec(nx,ny,xe,x,a1,a1t,a0h,a0o,a0to)
call makeb(nx,ny,xL,yL,g,b,xi,yi,hl)
call daxpy(n,-1.0d0,x,1,b,1)
dm=0.0d0
```

```
do i=1,n
  dc=dabs(b(i))
  if (dc.gt.dm) dm=dc
enddo
print*,' '
print*,'========================================= '
stop
end
```

Το υποπρόγραμμα start δημιουργεί τους πίνακες $a0, a0t, a0h, a0to, a1, a1t$

```
subroutine start(nx,ny,Lx,Ly,g,a0,a0h,a0t,
+ a1,a1t,ipvta0,ipvta0t,x,y,a0o,a0to,l)
implicit real*8 (a-h,o-z)
real*8 a1(5,*),a1t(5,*),a0(7,*),a0h(5,*),l,
+ a0t(7,*),Lx,Ly,x(*),y(*),a0o(5,*),a0to(5,*),c,A,B,G,D
integer ipvta0(*),ipvta0t(*)

hx=Lx/dfloat(nx)
hy=Ly/dfloat(ny)
g=hx/hy
c=.5d0*l*hx*hx
A=.5d0*(1.0d0+g*g)
B=5.0d0*g*g-1.0d0-c
E=5.0d0-g*g-c
D=8.0d0*c+10.0d0*(1.0d0+g*g)
do j=1,nx
  do i=1,7
    a0(i,j)=0.0d0
    a0t(i,j)=0.0d0
  enddo
enddo
do j=1,nx
  do i=1,5
    a0h(i,j)=0.0d0
    a1(i,j)=0.0d0
    a1t(i,j)=0.0d0
  enddo
enddo
do j=2,nx-1
  a0t(5,j)=-3.0d0*E-D
  a0h(3,j)=-0.2d0*E
```

```
  a1t(3,j)=2.0d0*E
  a0(5,j)=-D
  a1(3,j)=E
enddo
a0t(5,1)=9.0d0*a-3.0d0*B-3.0d0*E-D
a0t(5,nx)=a0t(5,1)
a0t(3,3)=0.6d0*a-0.2d0*B
a0t(7,nx-2)=a0t(3,3)
a0t(4,2)=-6.0d0*a+2.0d0*B
a0t(6,nx-1)=a0t(4,2)
a0h(3,1)=0.6d0*a-0.2d0*E
a0h(3,nx)=a0h(3,1)
a0h(2,2)=-0.4d0*a
a0h(4,nx-1)=-0.4d0*a
a0h(1,3)=0.04d0*a
a0h(5,nx-2)=0.04d0*a
a1t(3,1)=-6.0d0*a+2.0d0*E
a1t(3,nx)=-6.0d0*a+2.0d0*E
a1t(1,3)=-0.4d0*a
a1t(5,nx-2)=-0.4d0*a
a0(5,1)=-3.0d0*B-D
a0(5,nx)=-3.0d0*B-D
a0(3,3)=-0.2d0*B
a0(7,nx-2)=-0.2d0*B
a0(4,2)=2.0d0*B
a0(6,nx-1)=2.0d0*B
a1(3,1)=-3.0d0*a+E
a1(3,nx)=-3.0d0*a+E
a1(1,3)=-0.2d0*a
a1(5,nx-2)=-0.2d0*a
a1(2,2)=2.0d0*a
a1(4,nx-1)=2.0d0*a
do j=3,nx
  a0h(2,j)=-0.2d0*a
  a0h(4,j-2)=-0.2d0*a
  a0t(4,j)=-3.0d0*a+B
  a0t(6,j-2)=-3.0d0*a+B
  a0(4,j)=B
  a0(6,j-2)=B
  a1(2,j)=a
  a1(4,j-2)=a
enddo
do j=2,nx
  a1t(2,j)=2.0d0*a
  a1t(4,j-1)=2.0d0*a
```

```
enddo
a1t(2,2)=4.0d0*a
a1t(4,nx-1)=4.0d0*a
do i=1,5
  do j=1,nx
    a0o(i,j)=a0(i+2,j)
    a0to(i,j)=a0t(i+2,j)
  enddo
enddo
call dgbtrf(nx,nx,2,2,a0,7,ipvta0,info)
if (info.ne.0) then
  print*,' info for A0 =',info
  stop
endif
call dgbtrf(nx,nx,2,2,a0t,7,ipvta0t,info)
if (info.ne.0) then
  print*,' info for A0t =',info
  stop
endif
return
end
```

Το υποπρόγραμμα makeb δημιουργεί το δεξιό μέλος b

```
subroutine makeb(nx,ny,Lx,Ly,g,b,x,y,e)
implicit real*8 (a-h,o-z)
external bc,f
real*8 b(*),Lx,Ly,x(*),y(*),bc,f,e

hx=Lx/dfloat(nx)
hy=Ly/dfloat(ny)
g=hx/hy
hx2=hx*hx
x(1)=hx/2.0d0
do i=2,nx
  x(i)=x(i-1)+hx
enddo
y(1)=hy/2.0d0
do i=2,ny
  y(i)=y(i-1)+hy
enddo
hx22=hx*hx*0.5d0
```

```
c        ******first line ************
b(1)=hx*hx*(f(x(1),y(1),e)+f(x(2),y(1),e)+f(x(1),y(2),e)
+ -(f(x(3),y(1),e)+f(x(1),y(3),e))/10.0d0+8.0d0/5.0d0*
+ (f(0.0d0,y(1),e)+f(x(1),0.0d0,e)))-
+ 16.0d0*(1.0d0+g*g)/5.0d0*(bc(0.0d0,y(2))+
+ bc(x(2),0.0d0))
+ -(56.0d0/5.0d0-8.0d0*g*g-
+ 8.0d0*e*hx2/5.0d0)*bc(0.0d0,y(1))
+ -(56.0d0/5.0d0*g*g-8.0d0-
+ 8.0d0*e*hx2/5.0d0)*bc(x(1),0.0d0)
+ +8.0d0/25.0d0*(1.0d0+g*g)*(bc(0.0d0,y(3))
+ +bc(x(3),0.0d0))-
+ 128.0d0/25.0d0*(1.0d0+g*g)*bc(0.0d0,0.0d0)
do k=2,nx-1
  b(k)=(5.0d0*f(x(k),y(1),e)+f(x(k+1),y(1),e)+
  + f(x(k-1),y(1),e)+
  + 2.0d0*f(x(k),y(2),e)-f(x(k),y(3),e)/5.0d0+16.0d0/5.0d0*
  + f(x(k),0.0d0,e))*hx22-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(x(k-1),0.0d0)-(16.0d0*g*g-16.0d0/
  + 5.0d0-8.0d0*e*hx2/5.0d0)*bc(x(k),0.0d0)-
  +  8.0d0/5.0d0*(1.0d0+g*g)*bc(x(k+1),0.0d0)
enddo
b(nx)=hx*hx*(f(x(nx),y(1),e)+f(x(nx-1),y(1),e)+
+ f(x(nx),y(2),e)
+ -(f(x(nx-2),y(1),e)+f(x(nx),y(3),e))/10.0d0+8.0d0/5.0d0
+ *(f(x(nx),0.0d0,e)+f(x(nx)+x(1),y(1),e)))
+ -16.0d0*(1.0d0+g*g)/5.0d0*(bc(1.0d0,y(2))+bc(x(nx-1),0.0d0))
+ -(56.0d0/5.0d0-8.0d0*g*g-8.0d0*e*hx2/5.0d0)*bc(1.0d0,y(1))
+ -(56.0d0/5.0d0*g*g-8.0d0-8.0d0*e*hx2/5.0d0)*bc(x(nx),0.0d0)
+ +8.0d0/25.0d0*(1.0d0+g*g)*(bc(1.0d0,y(3))
+ +bc(x(nx-2),0.0d0))
+ -128.0d0/25.0d0*(1.0d0+g*g)*bc(1.0d0,0.0d0)
c       ******main body************
j=nx+1
do k=3,ny-1,2
  b(j)=hx22*(5.0d0*f(x(1),y(k),e)+2.0d0*f(x(2),y(k),e)+
  + f(x(1),y(k-1),e)+f(x(1),y(k+1),e)-f(x(3),y(k),e)/5.0d0+
  + 16.0d0/5.0d0*f(0.0d0,y(k),e))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(0.0d0,y(k-1))-
  + (16.0d0*g*g-16.0d0/5.0d0-8.0d0*e*hx2/5.0d0)*bc(0.0d0,y(k))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(0.0d0,y(k+1))
  j=j+1
  do i=2,nx-1
    b(j)=hx22*(8.0d0*f(x(i),y(k),e)+f(x(i+1),y(k),e)+
    + f(x(i-1),y(k),e)+f(x(i),y(k+1),e)+f(x(i),y(k-1),e))
```

```
    j=j+1
  enddo
  b(j)=hx22*(5.0d0*f(x(nx),y(k),e)+2.0d0*f(x(nx-1),y(k),e)+
  + f(x(nx),y(k+1),e)+f(x(nx),y(k-1),e)-f(x(nx-2),y(k),e)/5.0d0+
  + 16.0d0/5.0d0*f(x(nx)+x(1),y(k),e))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(1.0d0,y(k-1))-
  + (16.0d0*g*g-16.0d0/5.0d0-8.0d0*e*hx2/5.0d0)*bc(1.0d0,y(k))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(1.0d0,y(k+1))
  j=j+1
enddo
do k=2,ny-2,2
  b(j)=hx22*(5.0d0*f(x(1),y(k),e)+2.0d0*f(x(2),y(k),e)+
  + f(x(1),y(k-1),e)+f(x(1),y(k+1),e)-f(x(3),y(k),e)/5.0d0+
  + 16.0d0/5.0d0*f(0.0d0,y(k),e))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(0.0d0,y(k-1))-
  + (16.0d0*g*g-16.0d0/5.0d0-8.0d0*e*hx2/5.0d0)*bc(0.0d0,y(k))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(0.0d0,y(k+1))
  j=j+1
  do i=2,nx-1
    b(j)=hx22*(8.0d0*f(x(i),y(k),e)+f(x(i+1),y(k),e)+
    + f(x(i-1),y(k),e)+f(x(i),y(k+1),e)+f(x(i),y(k-1),e))
    j=j+1
  enddo
  b(j)=hx22*(5.0d0*f(x(nx),y(k),e)+2.0d0*f(x(nx-1),y(k),e)+
  + f(x(nx),y(k-1),e)+f(x(nx),y(k+1),e)-f(x(nx-2),y(k),e)/5.0d0+
  + 16.0d0/5.0d0*f(x(nx)+x(1),y(k),e))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(1.0d0,y(k-1))-
  + (16.0d0*g*g-16.0d0/5.0d0-8.0d0*e*hx2/5.0d0)*bc(1.0d0,y(k))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(1.0d0,y(k+1))
  j=j+1
enddo
c        ******last line ************
b(j)=hx*hx*(f(x(1),y(ny),e)+f(x(2),y(ny),e)+
+ f(x(1),y(ny-1),e)-
+ (f(x(3),y(ny),e)+f(x(1),y(ny-2),e))/10.0d0+8.0d0/5.0d0*
+ (f(x(1),y(ny)+y(1),e)+f(0.0d0,y(ny),e)))-
+ 16.0d0*(1.0d0+g*g)/5.0d0*(bc(0.0d0,y(ny-1))+
+ bc(x(2),1.0d0))
+ -(56.0d0/5.0d0-8.0d0*g*g-8.0d0*e*hx2/5.0d0)*bc(0.0d0,y(ny))
+ -(56.0d0/5.0d0*g*g-8.0d0-8.0d0*e*hx2/5.0d0)*bc(x(1),1.0d0)
+ +8.0d0/25.0d0*(1.0d0+g*g)*(bc(0.0d0,y(ny-2))
+ +bc(x(3),1.0d0))-
+ 128.0d0/25.0d0*(1.0d0+g*g)*bc(0.0d0,1.0d0)
j=j+1
do i=2,nx-1
```

```
  b(j)=hx22*(5.0d0*f(x(i),y(ny),e)+f(x(i+1),y(ny),e)+
  + f(x(i-1),y(ny),e)+ 2.0d0*f(x(i),y(ny-1),e)-
  + f(x(i),y(ny-2),e)/5.0d0+16.0d0/5.0d0*f(x(i),y(ny)+y(1),e))-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(x(i-1),1.0d0)-(16.0d0*g*g-16.0d0/
  + 5.0d0-8.0d0*e*hx2/5.0d0)*bc(x(i),1.0d0)-
  + 8.0d0/5.0d0*(1.0d0+g*g)*bc(x(i+1),1.0d0)
  j=j+1
enddo
b(j)=hx*hx*(f(x(nx),y(ny),e)+f(x(nx-1),y(ny),e)+
+ f(x(nx),y(ny-1),e)-
+ (f(x(nx-2),y(ny),e)+f(x(nx),y(ny-2),e))/10.0d0+8.0d0/5.0d0*
+ (f(x(nx),y(ny)+y(1),e)+f(x(nx)+x(1),y(ny),e)))-
+ 16.0d0*(1.0d0+g*g)/5.0d0*(bc(1.0d0,y(ny-1))+bc(x(nx-1),1.0d0))
+ -(56.0d0/5.0d0-8.0d0*g*g-8.0d0*e*hx2/5.0d0)*bc(1.0d0,y(ny))
+ -(56.0d0/5.0d0*g*g-8.0d0-8.0d0*e*hx2/5.0d0)*bc(x(nx),1.0d0)+
+ 8.0d0/25.0d0*(1.0d0+g*g)*(bc(1.0d0,y(ny-2))
+ +bc(x(nx-2),1.0d0))-
+ 128.0d0/25.0d0*(1.0d0+g*g)*bc(1.0d0,1.0d0)
return
end
```

Το υποπρόγραμμα f - πραγματική λύση

```
real*8 function f(x,y,e)
implicit real*8 (a-h,o-z)
real*8  x,y,e

c ***positioning parameters***
a=0.1d0
b=0.1d0
f=-0.4D4*exp(-0.1D3*(y-b)**2)*exp(-0.1D3*(x-a)**2)*(x**2-x)*(y**
+ 2-y)+0.1D2*exp(-0.1D3*(y-b)**2)*(-0.2D3*x+0.2D3*a)**2*exp(-0.1D3*
+ (x-a)**2)*(x**2-x)*(y**2-y)+0.2D2*exp(-0.1D3*(y-b)**2)*(-0.2D3*x+0
+ .2D3*a)*exp(-0.1D3*(x-a)**2)*(2*x-1)*(y**2-y)+0.2D2*exp(-0.1D3*(y-
+ b)**2)*exp(-0.1D3*(x-a)**2)*(y**2-y)+0.1D2*(-0.2D3*y+0.2D3*b)**2*e
+ xp(-0.1D3*(y-b)**2)*exp(-0.1D3*(x-a)**2)*(x**2-x)*(y**2-y)+0.2D2*(
+ -0.2D3*y+0.2D3*b)*exp(-0.1D3*(y-b)**2)*exp(-0.1D3*(x-a)**2)*(x**2-
+ x)*(2*y-1)+0.2D2*exp(-0.1D3*(y-b)**2)*exp(-0.1D3*(x-a)**2)*(x**2-x
+ )-e*0.1D2*exp(-0.1D3*(y-b)**2)*exp(-0.1D3*(x-a)**2)*(x**2-x)*(y**
+ 2-y)
return
end
```

Το υποπρόγραμμα Schur υλοποιεί την μέθοδο Schur Complement

```
subroutine Schur(nx,x,b,a1,a1t,a0o,a0to,a0h,a0t,
+ a0,error,r,rh,pi,ph,t,s,sh,ui,istep,
+ ipvta0,ipvta0t,xo,tm,t1,t2)
implicit real*8 (a-h,o-z)
real*8 x(*),b(*),a1(5,*),a1t(5,*),a0h(5,*),a0o(5,*),
+ a0(7,*),a0t(7,*),a0to(5,*),error,temp,r(*),rh(*),pi(*),ph(*),
+ t(*),xo(*),s(*),sh(*),ui(*),MPI_Wtime
integer ipvta0(*),ipvta0t(*)

n=nx*int(nx/2)
call solvedr(nx,nx,b,a0,a0h,a0t,ipvta0,ipvta0t)
call dcopy(n,b,1,r,1)
call matl(nx,nx,r,b,a1,a1t)
call daxpy(n,-1.0d0,r,1,b(n+1),1)
t1=MPI_Wtime()-tm
tm=MPI_Wtime()
call bicgstab(nx,x(n+1),b(n+1),xo,ipvta0,ipvta0t,
+ a0,a0t,a0h,a0o,a0to,a1,a1t,error,r,rh,pi,ph,
+ t,s,sh,ui,istep)
t2=MPI_Wtime()-tm
tm=MPI_Wtime()
call matu(nx,nx,x,x(n+1),a1,a1t)
call solvedr(nx,nx,x,a0,a0h,a0t,
+ ipvta0,ipvta0t)
call dscal(n,-1.0d0,x,1)
call daxpy(n,1.0d0,b,1,x,1)
return
end
```

Το υποπρόγραμμα bicgstab - επαναληπτική επίλυση γραμμικού συστήματος

```
subroutine bicgstab(nx,x,b,xo,ipvta0,ipvta0t,
+ a0,a0t,a0h,a0o,a0to,a1,a1t,error,r,rh,pi,ph,
+ t,s,sh,ui,istep)
implicit real*8 (a-h,o-z)
real*8 x(*),b(*),a1(5,*),a0(7,*),a0t(7,*),
+ a1t(5,*),a0h(5,*),a0o(5,*),xo(*),
+ rh(*),pi(*),ph(*),t(*),a0to(5,*),
```

```
+ s(*),sh(*),ui(*),error,r(*)
integer ipvta0(*),ipvta0t(*)

imaxstep=istep
tol=error
istep=0
n=nx*nx/2
dnrmb=dnrm2(n,b,1)
call dcopy(n,b,1,x,1)
call smatvec(nx,x,xo,r,a0,a0h,a0t,
+ a1,a1t,a0o,a0to,ipvta0,ipvta0t)
call dscal (n,-1.0d0,r,1)
call daxpy(n,1.0d0,b,1,r,1)
call dcopy(n,r,1,rh,1)
999 continue
istep=istep+1
if (istep.gt.1) roip2=roip1
roip1=ddot(n,rh,1,r,1)
if (roip1.eq.0.0d0) then
  print*,' BiCGSTAB Fails - Pi-1=0 '
  return
endif
if (istep.eq.1) then
  call dcopy(n,r,1,pi,1)
  else
  bi=(roip1/roip2)*(ai/wi)
  call daxpy(n,-wi,ui,1,pi,1)
  call dcopy(n,r,1,t,1)
  call daxpy(n,bi,pi,1,t,1)
  call dcopy(n,t,1,pi,1)
endif
call smatvec(nx,pi,xo,ui,a0,a0h,a0t,
+ a1,a1t,a0o,a0to,ipvta0,ipvta0t)
call dcopy(n,pi,1,ph,1)
ai=roip1/ddot(n,rh,1,ui,1)
call dcopy(n,r,1,s,1)
call daxpy(n,-ai,ui,1,s,1)
call dcopy(n,s,1,sh,1)
call smatvec(nx,s,xo,t,a0,a0h,a0t,
+ a1,a1t,a0o,a0to,ipvta0,ipvta0t)
wi=ddot(n,t,1,s,1)/ddot(n,t,1,t,1)
call daxpy(n,ai,ph,1,x,1)
call daxpy(n,wi,sh,1,x,1)
call daxpy(n,-wi,t,1,s,1)
call dcopy(n,s,1,r,1)
```

```
dnrmr=dnrm2(n,s,1)
error=dnrmr/dnrmb
if (wi.ne.0.0d0.and.error.gt.tol.and.istep.lt.imaxstep)
+ goto 999
return
end
```

Το υποπρόγραμμα smatvec - εκτελεί τον πολλαπλασιασμό $x = S * x_0$

```
subroutine smatvec(nx,xo,r,x,a0,a0h,a0t,
+ a1,a1t,a0o,a0to,ipvta0,ipvta0t)
implicit real*8 (a-h,o-z)
real*8 a0(7,*),a0o(5,*),a0t(7,*),a1(5,*),
+ a1t(5,*),a0h(5,*),x(*),r(*),xo(*),a0to(5,*)
integer ipvta0(*),ipvta0t(*)
n2=nx*nx/2
call matu(nx,nx,r,xo,a1,a1t)
call solvedr(nx,nx,r,a0,a0h,a0t,
+ ipvta0,ipvta0t)
call matl(nx,nx,x,r,a1,a1t)
call dscal(n2,-1.0d0,x,1)
do k=0,int(nx/2)-2
  CALL dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
  + 0.0d0,r(k*nx+1),1)
enddo
CALL dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(n2-2*nx+1),
+ 1,0.0d0,r(n2-nx+1),1)
CALL dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo(n2-nx+1),
+ 1,1.0d0,r(n2-nx+1),1)
call daxpy(n2,1.0d0,r,1,x,1)
return
end
```

Το υποπρόγραμμα matvec - εκτελεί τον πολλαπλασιασμό $x = A * x_0$

```
subroutine matvec(nx,ny,xo,x,
+ a1,a1t,a0h,a0o,a0to)
implicit real*8 (a-h,o-z)
real*8 a1(5,*),a1t(5,*),a0h(5,*),
+ x(*),xo(*),a0o(5,*),a0to(5,*)
```

```
n2y=int(ny/2)
n=nx*ny
n2=int(n/2)
call matu(nx,ny,x,xo(n2+1),a1,a1t)
call matl(nx,ny,x(n2+1),xo,a1,a1t)
call dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(nx+1),1,
+ 1.0d0,x,1)
call dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(n-2*nx+1),1,
+ 1.0d0,x(n-nx+1),1)
call dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo,1,
+ 1.0d0,x,1)
do k=n2y-1,1,-1
  call dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
  + 1.0d0,x(k*nx+1),1)
enddo
call dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo(n-nx+1),1,
+ 1.0d0,x(n-nx+1),1)
j=n2
do k=0,n2y-2
  CALL dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1+j),1,
  + 1.0d0,x(k*nx+1+j),1)
enddo
return
end
```

Το υποπρόγραμμα solvedr -επίλυση διαγώνιου συστήματος για white αγνώστους

```
subroutine solvedr(nx,ny,x,a0,a0h,a0t,
+ ipvta0,ipvta0t)
real*8 a0(7,*),a0h(5,*),a0t(7,*),x(*)
integer ipvta0(*),ipvta0t(*)

c *** Solve Dw*x = x
c *** where Dr:
c *** the white diagonal block of Coefficient matrix***

n2=int(ny/2)
do k=n2-1,1,-1
  call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
enddo
call dgbmv('n',nx,nx,2,2,-1.0d0,a0h,5,x(nx+1),1,
+ 1.0d0,x,1)
```

```
call dgbtrs('N',nx,2,2,1,a0t,7,ipvta0t,x,nx,info)
return
end
```

Το υποπρόγραμμα solvedb -επίλυση διαγώνιου συστήματος για black αγνώστους

```
subroutine solvedb(nx,ny,x,a0,a0h,a0t,
+ ipvta0,ipvta0t)
real*8 a0(7,*),a0h(5,*),a0t(7,*),x(*)
integer ipvta0(*),ipvta0t(*)

c *** Solve Db*x = x
c *** where Dr:
c *** the black diagonal block of Coefficient matrix***

n2=int(ny/2)
do k=0,n2-2
  call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
enddo
call dgbmv('n',nx,nx,2,2,-1.0d0,a0h,5,x((n2-2)*nx+1),1,
+ 1.0d0,x((n2-1)*nx+1),1)
call dgbtrs('N',nx,2,2,1,a0t,7,ipvta0t,x((n2-1)*nx+1),nx,info)
return
end
```

Το υποπρόγραμμα matd - πολλαπλασιασμός διαγώνιου πίνακα

```
subroutine matd(nx,ny,x,xo,a0o,a0to,a0h)
implicit real*8 (a-h,o-z)
real*8 x(*),a0o(5,*),a0h(5,*),a0to(5,*),xo(*)

call dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo,1,0.0d0,x,1)
call dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(nx+1),1,1.0d0,x,1)
do k=1,nx-2
  call dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
  + 0.0d0,x(k*nx+1),1)
enddo
call dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(nx*(nx-2)+1),
+ 1,0.0d0,x(nx*(nx-1)+1),1)
call dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo(nx*(nx-1)+1),
```

```
+ 1,1.0d0,x(nx*(nx-1)+1),1)
return
end
```

Το υποπρόγραμμα matl - πολλαπλασιασμός κάτω τριγωνικού πίνακα

```
subroutine matl(nx,ny,x,xo,a1,a1t)
implicit real*8 (a-h,o-z)
real*8 X(*),a1(5,*),a1t(5,*),xo(*)

n2=int(ny/2)
do k=1,n2-1
  call dcopy(nx,xo(k*nx+1),1,x(k*nx+1),1)
  call daxpy(nx,1.0d0,xo((k-1)*nx+1),1,x(k*nx+1),1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,x(k*nx+1),1,
  + 0.0d0,x((k-1)*nx+1),1)
enddo
call dgbmv('n',nx,nx,2,2,1.0d0,a1t,5,xo((n2-1)*nx+1),1,
+ 0.0d0,x((n2-1)*nx+1),1)
return
end
```

Το υποπρόγραμμα matu - πολλαπλασιασμός άνω τριγωνικού πίνακα

```
subroutine matu(nx,ny,x,xo,a1,a1t)
implicit real*8 (a-h,o-z)
real*8 x(*),a1(5,*),a1t(5,*),xo(*)

n2=int(ny/2)
do k=1,n2-1
  call dcopy(nx,xo(k*nx+1),1,x,1)
  call daxpy(nx,1.0d0,xo((k-1)*nx+1),1,x,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,x,1,
  + 0.0d0,x(k*nx+1),1)
enddo
call dgbmv('n',nx,nx,2,2,1.0d0,a1t,5,xo,1,0.0d0,x,1)
return
end
```

Το υποπρόγραμμα matdb - πολλαπλασιασμός διαγώνιου πίνακα,black κομμάτι

```
subroutine matdb(nx,ny,x,xo,a0o,a0to,a0h)
implicit real*8 (a-h,o-z)
real*8 x(*),xo(*),a0o(5,*),a0to(5,*),a0h(5,*)

nx2=int(nx/2)
do k=0,nx2-2
  call dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
  + 0.0d0,x(k*nx+1),1)
enddo
call dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo(nx*(nx2-2)+1),
+ 1,0.0d0,x(nx*(nx2-1)+1),1)
call dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo(nx*(nx2-1)+1),
+ 1,1.0d0,x(nx*(nx2-1)+1),1)
return
end
```

Το υποπρόγραμμα line_wb - επαναδιάταξη των αγνώστων σε αρίθμηση white-black

```
subroutine line_wb(ns,xold,xnew)
implicit real*8 (a-h,o-z)
real*8 xnew(*),xold(*)

n=ns*ns
ns2=2*ns
n2=n/2
iold=1
inew=1
c ***first the white lines***
do i=1,ns,2
  call dcopy(ns,xold(iold),1,xnew(inew),1)
  iold=iold+ns2
  inew=inew+ns
enddo
c ***second the black lines***
inew=n2+1
iold=ns+1
do i=2,ns,2
```

```
  call dcopy(ns,xold(iold),1,xnew(inew),1)
  iold=iold+ns2
  inew=inew+ns
enddo
return
end
```

Το υποπρόγραμμα inv_line_wb - επαναδιάταξη της λύσης απο αρίθμηση white-black σε κανονική σειρά

```
subroutine inv_line_wb(ns,xold,xnew)
implicit real*8 (a-h,o-z)
real*8 xnew(*),xold(*)

n=ns*ns
ns2=2*ns
n2=n/2
iold=1
inew=1
c ***first the white lines***
do i=1,ns,2
  call dcopy(ns,xold(iold),1,xnew(inew),1)
  iold=iold+ns
  inew=inew+ns2
enddo
c ***second the black lines***
inew=ns+1
iold=n2+1
do i=2,ns,2
  call dcopy(ns,xold(iold),1,xnew(inew),1)
  iold=iold+ns
  inew=inew+ns2
enddo
return
end
```

Makefile

```
FORTRAN  = mpif90
OPTS     =
LOADER   = mpif90
LOADOPTS = -O3 -o bcgs
LIBRARY  = -lmpi

FILES  =  makeb.o makematrices.o ieeeck.o line_wb.o inv_line_wb.o
f.o ddot.o  dgemv.o dgbtf2.o dcopy.o daxpy.o dgbmv.o dnrm2.o
dscal.o dgbtrf.o dgbtrs.o main.o matvec.o smatvec.o dger.o ilaenv.o
dgemm.o dtbsv.o dswap.o dtrsm.o lsame.o bicgstab.o dlaswp.o xerbla.o
idamax.o Schur.o matdb.o

OBJS   =

bcgs : Makefile $(FILES)
$(LOADER) $(LOADOPTS) $(FILES) $(OBJS) $(LIBRARY)
.f.o:
mpif90 -c -O3 $*.f
clean:
rm -f *.o *.inp core* a.out bcgs
```

Α΄.2 Παράλληλος Κώδικας

main.f

```
parameter (np=128,nx=4096,ny=nx,n=nx*ny,n2=n/2,ip=n2/np)
implicit real*8 (a-h,o-z)
include 'mpif.h'
real*8 b(n),x(n),xe(n),a1(5,nx),a1t(5,nx),r(ip),MPI_Wtime,
+ a0(7,nx),a0h(5,nx),a0t(7,nx),roip1pt,roip1,XX(ip),sh(ip),
+ a0o(5,nx),a0to(5,nx),xi(nx),yi(ny),ph(ip),pi(ip),s(ip),
+ brpart(ip),bbpart(ip),rtemp(nx),oldb(nx),ui(ip),rh(ip)
integer ipvta0(nx),ipvta0t(nx),status(MPI_STATUS_SIZE),p

call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD,myrank,ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD,p,ierr)
if (myrank.eq.0) then
  print*,'------ Parallel Schur Complement  ------------'
  print*,'   Nx = Ny    = ',nx
  print*,'   Processors    = ',np
  tcomp=0.0d0
  tcomm=0.0d0
  tm=MPI_Wtime()
  tmc=MPI_Wtime()
endif

e=1.0d0 ! parameter L
c ***domain parameters***
xL=1.0d0
yL=1.0d0
g=1.0d0 !(g=hx/hy)
resid=5.0d-18
iter=9159
call start(nx,ny,xL,yL,g,a0,a0h,a0t,a1,a1t,ipvta0,
+ ipvta0t,xi,yi,a0o,a0to,e)
istep=0
if (myrank.eq.0) then
  call makeb(nx,ny,xL,yL,g,b,xi,yi,e)
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_Scatter(b,ip,MPI_DOUBLE_PRECISION,brpart,ip,
+ MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
```

```
call MPI_Scatter(b(n2+1),ip,MPI_DOUBLE_PRECISION,bbpart,ip,
+ MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call drsolve(nx,ny,brpart,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
if (myrank.ne.0) call dcopy(nx,brpart,1,rtemp,1)
if ((mod(myrank,2).eq.0).and.(myrank.ne.0)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,106,
  + MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.np-1).and.(myrank.ne.0)) then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,106,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (mod(myrank,2).eq.1) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,206,
  + MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,206,
  + MPI_COMM_WORLD,status,ierr)
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call dcopy(ip,brpart,1,r,1)
call matl(nx,ny,r,oldb,a1,a1t,myrank,ip,np,rtemp)
call daxpy(ip,-1.0d0,r,1,bbpart,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tcomp13=tcomp
  tcomm13=tcomm
  tmc=MPI_Wtime()
endif
call dcopy(ip,bbpart,1,r,1)
c ***bicgstab start***
dnr=0.0d0
```

```
do i=1,ip
  dnr=dnr+bbpart(i)**2.0d0
enddo
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_REDUCE(dnr,dnrmb,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
  dnrmb=dsqrt(dnrmb)
  imaxstep=iter
  tol=resid
endif
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif

c ***SMATVEC***
if (myrank.ne.np-1) then
  call dcopy(nx,r(ip-nx+1),1,rtemp,1)
endif
if (mod(myrank,2).eq.0) then
  call MPI_SEND(rtemp(1),nx,MPI_DOUBLE_PRECISION,
  + myrank+1,116,MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,116,
  + MPI_COMM_WORLD,status,ierr)
endif
if ((mod(myrank,2).eq.1).and.(myrank.ne.np-1)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,
  + myrank+1,216,MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.0).and.(myrank.ne.np-1))  then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,216,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
```

```
call matu(nx,ny,r,oldb,a1,a1t,myrank,ip,np,rtemp)
call drsolve(nx,ny,r,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
if (myrank.ne.0) call dcopy(nx,r,1,rtemp,1)
  if ((mod(myrank,2).eq.0).and.(myrank.ne.0)) then
    call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,106,
    + MPI_COMM_WORLD,status,ierr)
  else
    if ((myrank.ne.np-1).and.(myrank.ne.0)) then
      call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,
      + 106,MPI_COMM_WORLD,status,ierr)
    endif
endif
if (mod(myrank,2).eq.1) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,206,
  + MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,206,
  + MPI_COMM_WORLD,status,ierr)
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matl(nx,ny,r,oldb,a1,a1t,myrank,ip,np,rtemp)
call dscal(ip,-1.0d0,r,1)
call matdb(nx,ny,XX,bbpart,a0o,a0to,a0h,myrank,ip,np)
call daxpy(ip,1.0d0,XX,1,r,1)
c ***SMATVEC END***
call dscal (ip,-1.0d0,r,1)
call daxpy(ip,1.0d0,bbpart,1,r,1)
dnrm=dnrm2(ip,r,1)
call dcopy(ip,r,1,rh,1)
roip1pt=ddot(ip,rh,1,r,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_REDUCE(roip1pt,roip1,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
```

```
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
istep=0
call dcopy(ip,r,1,pi,1)
999    continue
istep=istep+1
 call dcopy(ip,pi,1,ui,1)
c ***SMATVEC***
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
if (myrank.ne.np-1) then
  call dcopy(nx,ui(ip-nx+1),1,rtemp,1)
endif
if (mod(myrank,2).eq.0) then
  call MPI_SEND(rtemp(1),nx,MPI_DOUBLE_PRECISION,
  + myrank+1,116,MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,116,
  + MPI_COMM_WORLD,status,ierr)
endif
if ((mod(myrank,2).eq.1).and.(myrank.ne.np-1)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,
  + myrank+1,216,MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.0).and.(myrank.ne.np-1))  then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,216,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matu(nx,ny,ui,oldb,a1,a1t,myrank,ip,np,rtemp)
call drsolve(nx,ny,ui,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
if (myrank.ne.0) call dcopy(nx,ui,1,rtemp,1)
if ((mod(myrank,2).eq.0).and.(myrank.ne.0)) then
```

```
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,106,
  + MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.np-1).and.(myrank.ne.0)) then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,106,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (mod(myrank,2).eq.1) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,206,
  + MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,206,
  + MPI_COMM_WORLD,status,ierr)
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matl(nx,ny,ui,oldb,a1,a1t,myrank,ip,np,rtemp)
call dscal(ip,-1.0d0,ui,1)
call matdb(nx,ny,XX,pi,a0o,a0to,a0h,myrank,ip,np)
call daxpy(ip,1.0d0,XX,1,ui,1)
c ***SMATVEC END***
call dcopy(ip,pi,1,ph,1)
aip=ddot(ip,rh,1,ui,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_REDUCE(aip,aii,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0)  ai=roip1/aii
call MPI_BCAST(ai,1,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call dcopy(ip,r,1,s,1)
if (istep.gt.1) call dcopy(ip,sh,1,s,1)
call daxpy(ip,-ai,ui,1,s,1)
call dcopy(ip,s,1,sh,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
```

```
endif
c ***SMATVEC***
if (myrank.ne.np-1) then
  call dcopy(nx,s(ip-nx+1),1,rtemp,1)
endif
if (mod(myrank,2).eq.0) then
  call MPI_SEND(rtemp(1),nx,MPI_DOUBLE_PRECISION,
  + myrank+1,116,MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,116,
  + MPI_COMM_WORLD,status,ierr)
endif
if ((mod(myrank,2).eq.1).and.(myrank.ne.np-1)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,
  + myrank+1,216,MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.0).and.(myrank.ne.np-1))  then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,216,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matu(nx,ny,s,oldb,a1,a1t,myrank,ip,np,rtemp)
call drsolve(nx,ny,s,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
if (myrank.ne.0) call dcopy(nx,s,1,rtemp,1)
if ((mod(myrank,2).eq.0).and.(myrank.ne.0)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,106,
  + MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.np-1).and.(myrank.ne.0)) then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,106,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (mod(myrank,2).eq.1) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,myrank-1,206,
  + MPI_COMM_WORLD,status,ierr)
```

```
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank+1,206,
  + MPI_COMM_WORLD,status,ierr)
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matl(nx,ny,s,oldb,a1,a1t,myrank,ip,np,rtemp)
call dscal(ip,-1.0d0,s,1)
call matdb(nx,ny,XX,sh,a0o,a0to,a0h,myrank,ip,np)
call daxpy(ip,1.0d0,XX,1,s,1)
c ***SMATVEC END***
wia=ddot(ip,s,1,sh,1)
wib=ddot(ip,s,1,s,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_REDUCE(wia,wa,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
call MPI_REDUCE(wib,wb,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
  iter=iter+1
  wi=wa/wb
endif
call MPI_BCAST(wi,1,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call daxpy(ip,ai,ph,1,bbpart,1)
call daxpy(ip,wi,sh,1,bbpart,1)
call daxpy(ip,-wi,s,1,sh,1)
dnr=0.0d0
do i=1,ip
  dnr=dnr+sh(i)**2.0d0
enddo
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
```

```
call MPI_REDUCE(dnr,dnrmr,1,MPI_DOUBLE_PRECISION,
+ MPI_SUM,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
if (myrank.eq.0) then
  dnrmr=dsqrt(dnrmr)
  resid=dnrmr/dnrmb
  if (wi.ne.0.0d0.and.resid.gt.tol.and.istep.lt.imaxstep) then
    id=1
  else
    id=0
  endif
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_BCAST(id,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
if (id.eq.1)  then
  roip1p=ddot(ip,rh,1,sh,1)
  if (myrank.eq.0) then
    roip2=roip1
    tcomp=MPI_Wtime()-tmc+tcomp
    tmc=MPI_Wtime()
  endif
  call MPI_REDUCE(roip1p,roip1,1,MPI_DOUBLE_PRECISION,
  + MPI_SUM,0,MPI_COMM_WORLD,ierr)
  if (myrank.eq.0)  bi=(roip1/roip2)*(ai/wi)
  call MPI_BCAST(bi,1,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
  if (myrank.eq.0) then
    tcomm=MPI_Wtime()-tmc+tcomm
    tmc=MPI_Wtime()
  endif
  call daxpy(ip,-wi,ui,1,pi,1)
  call dcopy(ip,sh,1,s,1)
  call daxpy(ip,bi,pi,1,s,1)
  call dcopy(ip,s,1,pi,1)
  if (myrank.eq.0) then
    tcomp=MPI_Wtime()-tmc+tcomp
    tmc=MPI_Wtime()
  endif
```

```
  goto 999
endif
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tcomp4=tcomp-tcomp13
  tcomm4=tcomm-tcomm13
  tmc=MPI_Wtime()
endif
call dcopy(ip,bbpart,1,r,1)
if (myrank.ne.np-1) then
  call dcopy(nx,r(ip-nx+1),1,rtemp,1)
endif
if (mod(myrank,2).eq.0) then
  call MPI_SEND(rtemp(1),nx,MPI_DOUBLE_PRECISION,
  + myrank+1,116,MPI_COMM_WORLD,status,ierr)
else
  call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,116,
  + MPI_COMM_WORLD,status,ierr)
endif
if ((mod(myrank,2).eq.1).and.(myrank.ne.np-1)) then
  call MPI_SEND(rtemp,nx,MPI_DOUBLE_PRECISION,
  + myrank+1,216,MPI_COMM_WORLD,status,ierr)
else
  if ((myrank.ne.0).and.(myrank.ne.np-1))  then
    call MPI_recv(oldb,nx,MPI_DOUBLE_PRECISION,myrank-1,216,
    + MPI_COMM_WORLD,status,ierr)
  endif
endif
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tmc=MPI_Wtime()
endif
call matu(nx,ny,r,oldb,a1,a1t,myrank,ip,np,rtemp)
call drsolve(nx,ny,r,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
call dscal(ip,-1.0d0,r,1)
call daxpy(ip,1.0d0,r,1,brpart,1)
if (myrank.eq.0) then
  tcomp=MPI_Wtime()-tmc+tcomp
  tmc=MPI_Wtime()
endif
call MPI_Gather(brpart,ip,MPI_DOUBLE_PRECISION,x,ip,
+ MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
call MPI_Gather(bbpart,ip,MPI_DOUBLE_PRECISION,x(n2+1),ip,
+ MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
```

```
if (myrank.eq.0) then
  tcomm=MPI_Wtime()-tmc+tcomm
  tcomp57=tcomp-tcomp4-tcomp13
  tcomm57=tcomm-tcomm4-tcomm13
  tm=MPI_Wtime()-tm
  print*,'Comp. time for steps 1-3 = ',tcomp13,' seconds'
  print*,'Comm. time for steps 1-3 = ',tcomm13,' seconds'
  print*,'Comp. time for Bi-CGSTAB = ',tcomp4,' seconds'
  print*,'Comm. time for Bi-CGSTAB = ',tcomm4,' seconds'
  print*,'Comp. time for steps 5-7 = ',tcomp57,' seconds'
  print*,'Comm. time for steps 5-7 = ',tcomm57,' seconds'
  print*,'Total Computational time = ',tcomp,' seconds'
  print*,'Total Communication time = ',tcomm,' seconds'
  print*,'Total time = ',tcomp+tcomm,' seconds'
endif
if (myrank.eq.0) print*,'Bi-CGSTAB exits after ',istep,' steps.'
call MPI_FINALIZE(ierr)
stop
end
```

Το υποπρόγραμμα matdb - πολλαπλασιασμός διαγώνιου πίνακα,black κομμάτι

```
subroutine matdb(nx,ny,x,xo,a0o,a0to,a0h,myrank,ip,np)
implicit real*8 (a-h,o-z)
real*8 x(*),a0o(5,*),a0h(5,*),a0to(5,*),xo(*)

ny2=int(ny/2)
ns2=int(ny2/np)
if (myrank.eq.np-1) then
  do k=0,ns2-2
    CALL dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
    + 0.0d0,x(k*nx+1),1)
  enddo
  CALL dgbmv('n',nx,nx,2,2,1.0d0,a0h,5,xo((ns2-2)*nx+1),1,
  + 0.0d0,x((ns2-1)*nx+1),1)
  CALL dgbmv('n',nx,nx,2,2,1.0d0,a0to,5,xo((ns2-1)*nx+1),1,
  + 1.0d0,x((ns2-1)*nx+1),1)
else
  do k=0,ns2-1
    CALL dgbmv('n',nx,nx,2,2,1.0d0,a0o,5,xo(k*nx+1),1,
    + 0.0d0,x(k*nx+1),1)
  enddo
```

```
endif
return
end
```

Το υποπρόγραμμα matl - πολλαπλασιασμός κάτω τριγωνικού πίνακα

```
subroutine matl(nx,ny,xo,xold,a1,a1t,myrank,ip,np,t)
implicit real*8 (a-h,o-z)
real*8 a1(5,*),a1t(5,*),xo(*),xold(*),t(*)

n2=int(ip/nx)
do k=1,n2-1
  call dcopy(nx,xo((k-1)*nx+1),1,t,1)
  call daxpy(nx,1.0d0,xo(k*nx+1),1,t,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,t,1,
  + 0.0d0,xo((k-1)*nx+1),1)
enddo
if (myrank.eq.np-1) then
  call dcopy(nx,xo(ip-nx+1),1,t,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1t,5,t,1,
  + 0.0d0,xo(ip-nx+1),1)
else
  call daxpy(nx,1.0d0,xold,1,xo(ip-nx+1),1)
  call dcopy(nx,xo(ip-nx+1),1,t,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,t,1,
  + 0.0d0,xo(ip-nx+1),1)
endif
return
end
```

Το υποπρόγραμμα matu - πολλαπλασιασμός άνω τριγωνικού πίνακα

```
subroutine matu(nx,ny,xo,xold,a1,a1t,myrank,ip,np,t)
implicit real*8 (a-h,o-z)
real*8 a1(5,*),a1t(5,*),xo(*),xold(*),t(*)

n2=int(ip/ny)
do k=n2,2,-1
  call daxpy(nx,1.0d0,xo((k-2)*nx+1),1,xo((k-1)*nx+1),1)
  call dcopy(nx,xo((k-1)*nx+1),1,t,1)
```

```
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,t,1,
  + 0.0d0,xo((k-1)*nx+1),1)
enddo
if (myrank.eq.0) then
  call dcopy(nx,xo,1,t,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1t,5,t,1,
  + 0.0d0,xo,1)
else
  call daxpy(nx,1.0d0,xold,1,xo,1)
  call dcopy(nx,xo,1,t,1)
  call dgbmv('n',nx,nx,2,2,1.0d0,a1,5,t,1,
  + 0.0d0,xo,1)
endif
return
end
```

Το υποπρόγραμμα drsolve -επίλυση διαγώνιου συστήματος για white αγνώστους

```
subroutine drsolve(nx,ny,X,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
implicit real*8 (a-h,o-z)
real*8 X(*),a0(7,*),a0t(7,*),a0h(5,*)
integer ipvta0(*),ipvta0t(*),myrank,ip,np

c *** Solve Dw*x = x
c *** where Dr:
c *** the white diagonal block of Coefficient matrix***

ny2=int(ny/2)
ns2=int(ny2/np)
if (myrank.eq.0) then
  do k=ns2-1,1,-1
    call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
  enddo
  call dgbmv('n',nx,nx,2,2,-1.0d0,a0h,5,x(nx+1),1,
  + 1.0d0,x,1)
  call dgbtrs('N',nx,2,2,1,a0t,7,ipvta0t,x,nx,info)
else
  do k=ns2-1,0,-1
  call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
enddo
endif
```

```
return
end
```

Το υποπρόγραμμα dbsolve -επίλυση διαγώνιου συστήματος για black αγνώστους

```
subroutine dbsolve(nx,ny,X,a0,a0h,a0t,ipvta0,
+ ipvta0t,myrank,ip,np)
implicit real*8 (a-h,o-z)
real*8 X(*),a0(7,*),a0t(7,*),a0h(5,*)
integer ipvta0(*),ipvta0t(*),myrank,ip,np

c *** Solve Db*x = x
c *** where Dr:
c *** the black diagonal block of Coefficient matrix***

ny2=int(ny/2)
ns2=int(ny2/np)
if (myrank.eq.np-1) then
  do k=0,ns2-2
    call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
  enddo
  call dgbmv('n',nx,nx,2,2,-1.0d0,a0h,5,x((ns2-2)*nx+1),1,
  + 1.0d0,x((ns2-1)*nx+1),1)
  call dgbtrs('N',nx,2,2,1,a0t,7,ipvta0t,x((ns2-1)*nx+1),nx,info)
else
  do k=0,ns2-1
    call dgbtrs('N',nx,2,2,1,a0,7,ipvta0,x(k*nx+1),nx,info)
  enddo
endif
return
end
```

Βιβλιογραφία

[1] 0. Axelsson and A. Barker, "Finite element solution of boundary value problems". *Theory and computation*, Academic Press, Orlando, Fl., 1984.

[2] A. Delis and E. Mathioudakis, "A Finite Volume Method Parallelization for the Simulation of Free Surface Shallow Water Flows", *Maths and Computers in Simulation*, ELSEVIER, 79(11),pp. 3339-3359, 2009.

[3] M. Deville, P. Fischer, and E. Mund, *High-Order Methods for Incompressible Fluid Flow*, Cambridge Monographs on Applied and Computational Mathematics, 9, Cambridge Univ. Press, 2004.

[4] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Toczon and A. White, *SourceBook of Parallel Computing*, San Francisco,CA: Morgan Kaufmann Publishers, 2003.

[5] J. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*, Springer, 2002.

[6] Gaitonde, D. V. and Visbal, M. R., “High–order schemes for Navier–Stokes equations: Algorithms and implementation into FDL3DI," Air Vehicles Directorate, Air Force Research Laboratory, Write–Patterson Air Force Base, Ohio 45433-7913, AFRL-VA-WP-TR-1998-3060, August 1998.

[7] 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.

[8] L. A. Hageman and D. M. Young, " Applied Iterative Methods", *Academic Press*, New York, 1981.

[9] N.A. Kampanis and J. A. Ekaterinaris, "A staggered grid, high-order accurate method for the Incompressible Navier-Stokes equations", *J. Comp. Physics*,215,pp. 589-613,2006.

[10] Lele, S. K., ”Compact finite difference schemes with spectral–like resolution," *J. Comput. Physics*, **103**, 1992, pp. 16–42.

[11] V. Mandikas, E. Mathioudakis and N. Kampanis, "Multigrid techniques for high order cell centered compact scheme discretizations of the modified Helmholtz equation", submitted,2012.

[12] V. G. Mandikas, E.N. Mathioudakis, N. A. Kampanis and J. A. Ekaterinaris, " High-order accurate numerical pressure correction based on Geometric Multigrid schemes for the incompressible Navier Stokes equations ", Procs Conference in Numerical Analysis (NumAn ’10), pp 149-157, Chania 2010.

[13] E. Mathioudakis, V. Mandikas, J. Ekaterinaris and N. Kampanis, "High order pressure correction multigrid techniques applied on staggered grid finite-difference compact schemes for the incompressible Navier Stokes equations ", submitted, 2012.

[14] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Bi-CGSTAB for collocation equations on distributed memory parallel computers" , *Numerical Mathematics and advanced applications - ENUMATH 2001*, pp. 957-966, 2003.

[15] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Iterative Solution of Elliptic Collocation Systems on a Cognitive Parallel Computer", *Computers and Mathematics with applications*, vol. 48, pp. 951-970, 2004.

[16] E. N. Mathioudakis and E. P. Papadopoulou, "Grid computing for Bi-CGSTAB applied to the solution of modified Helmholtz equation", *INt. J. Applied Maths and comp. sciences*, vol. 4, no. 3, pp. 179-184, 2007.

[17] Open Message Passing Interface (OpenMPI) web page, *http://www.open-mpi.org*.

[18] Y. Saad, "Iterative Methods for sparse linear systems", SIAM, 2003.

[19] D.M. Young, "Iterative Solution of Large Linear Systems", *Academic Press*, New York, 1981

[20] R.S. Varga, "Matrix Iterative Analysis",*Prentice Hall*, Englewood Cliffs, NJ, 1962

[21] H. A. van der Vorst, "Bi-CGSTAB : A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.*, vol. 13, pp. 631-644, 1992.

[22] Wu, J. C., "Theory of aerodynamic force and moment in viscous flow," *AIAA Journal*, **19** 4, 1981, pp. 432–441.

[23] Jun Zhang, "Multigrid Method and fourth order compact difference scheme for 2D Poisson equation with unequal meshsize discretization", *J. Comp. Physics*, 179, pp. 170-179, 2002.