ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΑΡΙΘΜΗΤΙΚΩΝ ΜΕΘΟΔΩΝ ΣΕ ΣΕΙΡΙΑΚΟ ΚΑΙ ΠΑΡΑΛΛΗΛΟ ΠΕΡΙΒΑΛΛΟΝ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ 2ης ΤΑΞΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΕΩΡΓΙΟΥ ΜΑΤΛΗ
Α.Ε.Μ 4109

Επιβλέπων

ΙΩΑΝΝΗΣ ΤΟΥΛΟΠΟΥΛΟΣ

Καστορία

5 Σεπτεμβρίου 2023

Ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στον επιβλέπων της πτυχιακής εργασίας μου κ.Τουλόπουλο Ιωάννη για την καθοδήγηση και την υποστήριξη του σε όλη τη διάρκεια αυτής της ερευνητικής εργασίας. Η τεχνογνωσία και η ενθάρρυνσή του ήταν πολύτιμη για να με βοηθήσουν να ανταποκριθώ στις προκλήσεις της διπλωματικής εργασίας μου και να αναπτύξω τις δεξιότητές μου ως ερευνητής.

Θα ήθελα επίσης να ευχαριστήσω τον κ.Δημόκα και τον κ.Βαρδάκα, τα μέλη της επιτροπής της πτυχιακής εργασίας μου, καθώς και τον κ.Σίσια, για τα σχόλια και τις γνώσεις τους, που βοήθησαν στη διαμόρφωση και την ολοκλήρωση αυτής της εργασίας.

Θα ήθελα να εκφράσω την εκτίμηση μου στους συναδέλφους και τους συμμαθητές μου, οι οποίοι παρείχαν υποστήριξη, ενθάρρυνση και πολύτιμη ανατροφοδότηση σε όλη τη διάρκεια των προπτυχιακών μου σπουδών.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την αμέριστη στήριξη και ενθάρρυνση. Η αγάπη, η κατανόηση και η ενθάρρυνση τους με στήριξαν στις προκλήσεις των σπουδών μου και έκαναν αυτό το επίτευγμα δυνατό.

Σας ευχαριστώ όλους για τη συμβολή σας στην πραγματοποίηση αυτής της διατριβής.

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει μια ολοκληρωμένη διερεύνηση της αλγοριθμικής υλοποίησης αριθμητικών μεθόδων για την επίλυση διαφορικών εξισώσεων δεύτερης τάξης τόσο σε σειριακά όσο και σε παράλληλα υπολογιστικά περιβάλλοντα. Ξεκινάμε διερευνώντας και κατανοώντας τις δυνατότητες του OpenMP για παραλληλισμό, στη συνέχεια αναλύουμε σε βάθος τις σειριακές μεθόδους Jacobi και Gauss-Seidel, αναλύοντας τις ιδιότητες σύγκλισης και την υπολογιστική τους αποτελεσματικότητα. Στη συνέχεια, η μελέτη μετατοπίζεται σε παράλληλες εφαρμογές, όπου παρουσιάζουμε και αξιολογούμε την απόδοση των παράλληλων υλοποιήσεων των μεθόδων Jacobi και Gauss-Seidel μέσα σε μια αρχιτεκτονική κοινής μνήμης. Εισάγουμε το πρόβλημα δύο συνοριακών σημείων και αναλύουμε την ακρίβεια και τη σταθερότητα της μεθόδου των πεπερασμένων διαφορών, καθώς σχετίζεται με την διακριτοποίηση διαφορικών εξισώσεων. Συζητάμε στρατηγικές διαμέρισης χωρίου με εστίαση στη μέθοδο Additive Schwarz ως ζωτικής σημασίας εργαλείο για τη διαμέριση του χωρίου σύνθετων προβλημάτων και τον συντονισμό της παράλληλης εκτέλεσης, ενισχύοντας έτσι την ικανότητά μας να χειριζόμαστε αποτελεσματικά μεγάλης κλίμακας προβλήματα συνοριακών σημείων. Αυτή η διατριβή προσφέρει κρίσιμες γνώσεις σε ακαδημαϊκούς και επαγγελματίες για αποτελεσματικές λύσεις σε προκλητικές διαφορικές εξισώσεις δεύτερης τάξης σε μια ποικιλία επιστημονικών και μηχανικών πεδίων. Προσφέρει διορατικές αναλύσεις στις αλγοριθμικές πολυπλοκότητες των αριθμητικών προσεγγίσεων.

Λέξεις Κλειδιά - OpenMP, Επαναληπτικές Μέθοδοι, Jacobi, Gauss-Seidel, Μέθοδος Πεπερασμένων στοιχείων, Πρόβλημά Δύο Συνοριακών Σημείων, Διαμέριση Χωρίου, Additive Schwarz

Abstract

This thesis presents a comprehensive investigation into the algorithmic implementation of numerical methods for solving second-order differential equations within both serial and parallel computing environments. We start by exploring and understanding the capabilities of OpenMP for parallelization, then analyze the serial Jacobi and Gauss-Seidel methods in depth, analyzing their convergence properties and computational effectiveness. The study then shifts to parallel applications, where we present and evaluate the performance of parallel implementations of the Jacobi and Gauss-Seidel methods within a shared memory architecture. We introduce the two-point boundary value problem and analyze the accuracy and stability of the finite difference method as it pertains to the discretization of differential equations. We discuss domain decomposition methods with a focus on the Additive Schwarz method as a vital tool for dividing the domain of complex problems and coordinating parallel execution, thereby enhancing our capacity to effectively handle large-scale two-point boundary value problems. This thesis offers critical knowledge to academics and practitioners for effective solutions to challenging second-order differential equations in a variety of scientific and engineering fields. It offers insightful analyses of the algorithmic complexities of numerical approaches.

Keywords - OpenMP, Iterative Methods, Jacobi, Gauss-Seidel, Finite Difference Method, Two-Point BVP, Domain Decomposition, Additive Schwarz

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Πινάκων

Λίστα Αλγορίθμων

Πρόλογος

Στην σημερινή εποχή, συχνά προκειμένου να κατανοήσουμε και να περιγράψουμε διάφορα φυσικά φαινόμενα και προβλήματα μηχανικής, οδηγούμαστε στην μαθηματική μοντελοποίηση αυτών, [1]. Κατά την μαθηματική μοντελοποίηση, εφαρμόζουμε θεμελιώδεις νόμους της φυσικής, όπως η αρχή διατήρησης της ενέργειας και της ύλης, και έπειτα, με τη χρήση τεχνικών και θεωρημάτων διαφορικού και ολοκληρωτικού λογισμού, μπορούμε να παράγουμε το μαθηματικό μοντέλο που περιγράφει το πρόβλημα. Τα μαθηματικά μοντέλα συνήθως καταλήγουν σε Μερικές Διαφορικές Εξισώσεις (Μ.Δ.Ε), δηλαδή σε εξισώσεις οι οποίες περιέχουν ως άγνωστες μεταβλητές τις ποσότητες που θέλουμε να μελετήσουμε, και συμπληρώνονται με κατάλληλες αρχικές συνθήκες, οι οποίες προκύπτουν από το φυσικό πρόβλημα. Οι Μ.Δ.Ε που περιγράφουν καθημερινά προβλήματα της φυσικής και της μηχανικής, είναι συνήθως αδύνατο να επιλυθούν με αναλυτικές τεχνικές μαθηματικών, και έτσι προσφεύγουμε στην αριθμητική επίλυση τους, δηλαδή πρώτα στην διακριτοποίηση τους, και στη συνέχεια στην παραγωγή ενός αλγορίθμου, από τον οποίο θα πάρουμε μια αριθμητική προσεγγιστική λύση.

Οι πιο γνωστές και συχνά χρησιμοποιούμενες μέθοδοι αριθμητικής επίλυσης Μ.Δ.Ε, είναι η μέθοδος των πεπερασμένων στοιχείων και η μέθοδος των πεπερασμένων διαφορών [4]. Κατά την μελέτη περίπλοκων συστημάτων με ακανόνιστη γεωμετρία ή οριακές συνθήκες, η μέθοδος των πεπερασμένων στοιχείων προτιμάται συχνά έναντι των πεπερασμένων διαφορών. Από την άλλη πλευρά, όταν η γεωμετρία είναι απλή και οι συνοριακές συνθήκες είναι σαφείς, η μέθοδος πεπερασμένων διαφορών επιλέγεται πιο συχνά. Επιπλέον, οι πεπερασμένες διαφορές είναι πιο απλές στην κατασκευή και απαιτούν λιγότερους υπολογιστικούς πόρους. Η κοινή ιδιότητα και των δύο μεθόδων, είναι η δυνατότητα τους να μετατρέπουν ένα φυσικό πρόβλημα σε ένα διακριτό ανάλογο του, το οποίο μας οδηγεί τελικά σε ένα γραμμικό σύστημα αγνώστων που

επιλύουμε με αριθμητικές επαναληπτικές μεθόδους.

Επαναληπτικές μέθοδοι για την επίλυση γραμμικών συστημάτων υπάρχουν πολλές, αλλά οι δύο από τις πιο συχνά χρησιμοποιούμενες είναι οι επαναληπτική μέθοδος Jacobi και Gauss-Seidel [5]. Η μέθοδος Jacobi μπορεί να συγκλίνει στη πραγματική λύση του προβλήματος για διάφορους πίνακες αγνώστων, και είναι απλή στην κατανόηση και στην εφαρμογή. Επίσης, χρησιμοποιεί περισσότερους υπολογιστικούς πόρους από τη μέθοδο Gauss-Seidel, και μπορεί να συγκλίνει αργά για συγκεκριμένους πίνακες. Συνήθως για συμμετρικά συστήματα, η μέθοδος Gauss-Seidel συγκλίνει γρηγορότερα από τη μέθοδο Jacobi, διότι αξιοποιεί περισσότερες πληροφορίες στην επανάληψη, και για αυτό το λόγο, καταναλώνει λιγότερους υπολογιστικούς πόρους. Ωστόσο, η ιδιότητα της μεθόδου Gauss-Seidel να χρησιμοποιεί περισσότερες πληροφορίες σε κάθε της επανάληψη, την καθιστά δύσχρηστη για παράλληλες υλοποιήσεις.

Σκοπός αυτής της εργασίας είναι η μελέτη και υλοποίηση της μεθόδου των πεπερασμένων διαφορών για το πρόβλημα συνοριακών τιμών δύο σημείων, καθώς και των επαναληπτικών μεθόδων Jacobi και Gauss-Seidel, με σειριακές και παράλληλες τεχνικές. Αρχικά, θα περιγράψουμε τη διαδικασία της παράλληλης επεξεργασίας με το πρότυπο OpenMP, το οποίο χρησιμοποιείται σε αρχιτεκτονικές διαμοιραζόμενης μνήμης, και θα εξηγήσουμε τις διάφορες δυνατότητες του. Στη συνέχεια, θα μελετήσουμε τα χαρακτηριστικά των επαναληπτικών μεθόδων Jacobi και Gauss-Seidel, και θα περιγράψουμε την σειριακή και παράλληλη υλοποίηση τους. Χαρακτηριστικό της μεθόδου πεπερασμένων διαφορών, είναι η διαδικασία της προσέγγισης των παραγώγων με πολυώνυμα Taylor, με την οποία προσεγγίζουμε τις παραγώγους που εμφανίζονται στην λύση της διαφορικής εξίσωσης, και με κατάλληλες τεχνικές, μπορούμε να αυξήσουμε την ακρίβεια της. Επιπλέον, θα κάνουμε μια εισαγωγή στις πιο κοινές τεχνικές αποσύνθεσης χωρίων, οι οποίες περιγράφουν τη διαμέριση του χωρίου που ορίζεται μια Μ.Δ.Ε. Η εισαγωγή γίνεται όσο το δυνατόν πιο απλή, με ιδιαίτερη έμφαση στον τρόπο με τον οποίο ορίζονται

αυτές οι μέθοδοι ως προς τις διαφορικές εξισώσεις, και τους αραιούς πίνακες που προκύπτουν από τις διακριτοποιήσεις τους. Τέλος, θα εφαρμόσουμε τις παραπάνω τεχνικές σε ένα απλό ακαδημαϊκό πρόβλημα και θα συγκρίνουμε τα αποτελέσματα τους.

Κεφάλαιο 1

Μαθηματικό Υπόβαθρο

1.1 Νόρμες και Πίνακες

Ορισμός 1.1.1 *(Διανυσματική νόρμα) Αν $x \in \mathbb{R}^n$, νόρμα του διανύσματος x είναι μια πραγματική συνάρτηση με τις παρακάτω ιδιότητες*

- $||x|| \geq 0$ *και* $||x|| = 0$ *αν και μόνο αν* $x = 0$.
- $||\alpha x|| = |\alpha| \, ||x||$ *για κάθε συντελεστή* α.
- $||x + y|| \leq ||x|| + ||y||, \; x, y \in \mathbb{R}$.

Συχνά χρησιμοποιούμενες νόρμες διανυσμάτων είναι οι

- $||x||_{\inf} = \max_{1 \leq i \leq n} |x_i|$.
- $||x||_1 = \sum_{i=1}^{n} |x_i|$.
- $||x||_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$.
- $||x||_p = (\sum_{i=1}^{n} x_i^p)^{1/p}$.

Ορισμός 1.1.2 *(Φυσική νόρμα) Έστω ο πίνακας $A \in \mathbb{R}^{n \times m}$. Η ποσότητα*

$$||A|| = \sup_{x \in \mathbb{R}^n - \{0\}} \frac{||Ax||}{||x||} = \sup_{||x|| \leq 1} ||Ax|| = \sup_{||x|| = 1} ||Ax||$$

όπου με sup συμβολίζουμε το μικρότερο άνω φράγμα μιας έκφρασης. Η νόρμα που ορίζεται στη παραπάνω σχέση ονομάζεται φυσική νόρμα του A που αντιστοιχεί στη διανυσματική νόρμα $||.||$. Η φυσική νόρμα ικανοποιεί τις ιδιότητες της διανυσματικής νόρμας. Κάποιες ιδιότητες της φυσικής νόρμας που χρησιμοποιούνται συχνά είναι

- $||Ax|| \leq ||A||\ |x|,\ \forall x \in \mathbb{R}^n$.
- $||A + B|| \leq ||A|| + ||B||$.
- $||AB|| \leq ||A||\ ||B||$.

Συχνά χρησιμοποιούμενες νόρμες πινάκων είναι οι

- $||A||_{\inf} = \max i\ \sum j|a_{ij}|$.
- $||A||_1 = \max j\ \sum i|a_{ij}|$.
- $||x||_2 = \left(\sum_{1\leq i,j\leq n} a_{ij}^2\right)^{1/2}$.

Ορισμός 1.1.3 *Ένας συμμετρικός $n \times n$ πίνακας A είναι θετικά ορισμένος αν ισχύει η ακόλουθη σχέση*

$$x^T Ax > 0,$$

για κάθε n-διάστατο διάνυσμα $x \neq 0$.

1.2 Μαθηματική Μοντελοποίηση και Προσομοίωση

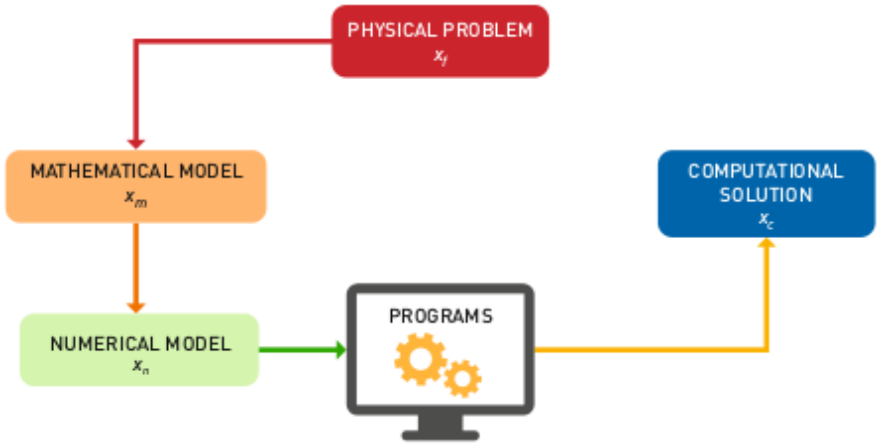

Figure 1.1: Διάγραμμα ροής μαθηματικής μοντελοποίησης. Εικόνα: https://link.springer.com/book/10.1007/978-3-030-44541-6, [1]

Τα διάφορα προβλήματα του πραγματικού κόσμου, οι αλληλεπιδράσεις στα φυσικά φαινόμενα και οι ιδιότητες τους, μπορούν πολλές φορές να περιγράφουν με μαθηματικά μοντέλα. Ο στόχος της μαθηματικής μοντελοποίησης είναι να μετατρέψει προβλήματα που προκύπτουν από τον φυσικό κόσμο σε μαθηματικά προβλήματα, δηλαδή σε Μ.Δ.Ε, και και να παρέχει λύσεις σε αυτά. Ένα μαθηματικό μοντέλο μπορεί να χρησιμοποιηθεί για πολλά διαφορετικά προβλήματα που έχουν κοινά χαρακτηριστικά. Τα μαθηματικά μοντέλα συνήθως ομαδοποιούνται ανάλογα με το είδος των προβλημάτων που μπορούν να αντιμετωπίσουν. Υπάρχουν μαθηματικά μοντέλα για τη μελέτη της κίνησης των ρευστών, της παραμόρφωσης των στερεών κατασκευών, της δυναμικής των ζωικών πληθυσμών που ανταγωνίζονται για την επιβίωση, της εξέλιξης των χρηματοπιστωτικών αγορών, της εξάπλωσης των επιδημιών και πολλών άλλων φαινομένων.

Ας επισημάνουμε ότι υπάρχουν ακόμη πιο γενικά μαθηματικά μοντέλα που είναι ικανά να λύνουν προβλήματα πολύ διαφορετικής φύσης. Ένα μεμονωμένο μαθηματικό μοντέλο μπορεί για παράδειγμα να βοηθήσει στην κατανόηση του πώς διαδίδεται η θερμότητα στους τοίχους ενός κτιρίου, ποιο είναι το ηλεκτρικό

δυναμικό που δημιουργείται από μια ορισμένη κατανομή της αλλαγής στον περιβάλλοντα χώρο ή πώς μια διαλυτή ουσία διασπείρεται σε ένα διάλυμα. Ειδικότερα, η προσομοίωση ενός φυσικού προβλήματος είναι μια διαδικασία που έχει τους εξής στόχους

- Να μεταφράσει ένα πραγματικό πρόβλημα σε ένα μαθηματικό μοντέλο.
- Να δημιουργήσει ένα διακριτό ανάλογο του που να προσεγγίζει το πραγματικό.
- Να υπολογίζει το διακριτό ανάλογο στον υπολογιστή.
- Να επιβεβαιώσει την ορθότητα του μαθηματικού μοντέλου και του διακριτού του μέσα από μια σειρά επιστημονικών υπολογισμών.

Στη μελέτη πολύπλοκων μαθηματικών προβλημάτων, τις περισσότερες φορές είναι αδύνατον να βρεθεί η πραγματική λύση. Αυτό συμβαίνει κυρίως για δύο λόγους

- Παρόλο που γνωρίζουμε τον μαθηματικό τύπο για την λύση της Μ.Δ.Ε του μοντέλου, σε ορισμένες περιπτώσεις η εφαρμογή του θα απαιτούσε τεράστιο αριθμό υπολογισμών και θα διαρκούσε πάρα πολύ χρόνο (χρόνια ή δεκαετίες).
- Σε άλλες καταστάσεις (τη συντριπτική πλειοψηφία), δεν μπορούμε να επιλύσουμε την Μ.Δ.Ε του μαθηματικό μοντέλου, κάτι που μας εμποδίζει να διατυπώσουμε άμεσα τη λύση του προβλήματος.

Η αριθμητική μοντελοποίηση μας βοηθάει να ξεπεράσουμε αυτές τις δυσκολίες.

Αριθμητική Μοντελοποίηση

Τα αριθμητικά μοντέλα είναι διακριτές έννοιες μαθηματικών μοντέλων, όπου η λύση τους διατυπώνεται με αριθμητικές μεθόδους που υπολογίζονται από εκτελέσιμα προγράμματα στον υπολογιστή. Στην εργασία αυτή, θα ονομάζουμε πολλές φορές την αριθμητική λύση που δίνει ο υπολογιστής ως υπολογιστική λύση.

Δεδομένα-Μοντέλο-Αποτελέσματα

Ο τρόπος με τον οποίο ερμηνεύουμε ένα μαθηματικό μοντέλο M είναι σαν ένα εργαλείο που επιδρά σε κάποια δεδομένα d_m(προσεγγίσεις των πραγματικών δεδομένων d_f του προβλήματος) και παράγει μια λύση, δηλαδή το αποτέλεσμα x_m. Το m αναφέρεται στο μαθηματικό μοντέλο.

Τα δεδομένα d_m είναι προσεγγίσεις των πραγματικών δεδομένων d_f, με τον ίδιο τρόπο που τα αποτελέσματα x_m είναι προσεγγίσεις των πραγματικών αποτελεσμάτων x_f.

Figure 1.2: Διάγραμμα ροής δεδομένων και μαθηματικού μοντέλου. Εικόνα: https://link.springer.com/book/10.1007/978-3-030-44541-6, [1]

Η αποτελεσματικότητα του μοντέλου M είναι ότι μπορεί να δεχθεί διαφορετικά δεδομένα και να μην αλλάξει η ουσία του. Αυτή η έννοια ονομάζεται ευστάθεια του μαθηματικού μοντέλου. Βέβαια τα αποτελέσματα θα εξαρτηθούν και από τα δεδομένα, αλλά και από το μαθηματικό μοντέλο.

Αντίθετα το αριθμητικό μοντέλο N είναι ένα εργαλείο που επιδρά σε μια προσέγγιση d_n των δεδομένων d_m, και παράγει την αριθμητική λύση x_n, που είναι μια προσέγγιση της λύσης x_m.

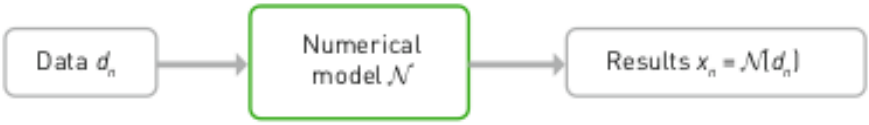

Figure 1.3: Διάγραμμα ροής δεδομένων και αριθμητικού μοντέλου. Εικόνα: https://link.springer.com/book/10.1007/978-3-030-44541-6, [1]

Τέλος, η υπολογιστική λύση που δημιουργεί ο υπολογιστής, μέσω του προγράμματος που υλοποιεί την αριθμητική μέθοδο, συνήθως συμβολίζεται με x_c. Δεδομένου ότι, ο υπολογιστής αναπόφευκτα εισάγει σφάλματα, γενικά η αριθμητική (θεωρητική) λύση και η υπολογιστική λύση δεν είναι ίδιες.

1.3 Γενικά περί Σφαλμάτων

Όταν για ένα πραγματικό πρόβλημα κατασκευάζουμε ένα μαθηματικό μοντέλο, συνήθως κάνουμε κάποιες παραδοχές και έτσι εισάγουμε ένα σφάλμα. Η ασυμφωνία μεταξύ της πραγματικής λύσης x_f του πραγματικού προβλήματος, και της λύσης x_m του μαθηματικού μοντέλου, ονομάζεται ως το σφάλμα του μοντέλου.

$$e_m = x_f - x_m$$

Ένα δεύτερο σφάλμα είναι το αριθμητικό σφάλμα e_n, το οποίο μετρά τη διαφορά μεταξύ της λύσης του μαθηματικού x_m και αριθμητικού x_n μοντέλου,

$$e_n = x_m - x_n$$

Τα αριθμητικά σφάλματα προκαλούνται κατά την προσέγγιση του μαθηματικού μοντέλου με το διακριτό ανάλογο του.
Το εκτελέσιμο πρόγραμμα που χρησιμοποιείται για την εφαρμογή της αριθμητικής μεθόδου σε έναν υπολογιστή θα εισάγει περαιτέρω λάθη, που ονομάζονται σφάλματα στρογγυλοποίησης e_r, λόγω του γεγονότος ότι οι αλγεβρικές πράξεις που εκτελούνται από τον υπολογιστή επηρεάζονται και αυτές από σφάλματα. Έτσι, το σφάλμα στρογγυλοποίησης ενός μοντέλου θα ισούται με τη διαφορά της αριθμητικής λύσης και της υπολογιστικής λύσης.

$$e_r = x_n - x_c$$

Τέλος, το άθροισμα των αριθμητικών σφαλμάτων και των σφαλμάτων στρογγυλοποίησης παράγει υπολογιστικά σφάλματα όπως

$$e_c = e_n + e_r = (x_m - x_n) + (x_n - x_c) = x_m - x_c$$

Μέσω της μαθηματικής ανάλυσης, μπορούμε να κρατήσουμε υπό έλεγχο το μέγεθος του σφάλματος. Αν ακολουθήσουμε προσεκτικά την μεθοδολογία διακριτοποίησης με υψηλή ακρίβεια, και εφαρμόσουμε σωστή υλοποίηση, τα υπόλοιπα σφάλματα είναι μικρά και επικρατεί μόνο το σφάλμα προσέγγισης, οπότε η υπολογιστική λύση θα μοιάζει πολύ με την πραγματική.

$$x_c \approx x_f,$$

Η διαδικασία σύγκρισης της υπολογιστικής λύσης και της πραγματικής λύσης, δηλαδή ο έλεγχος του σφάλματος

$$x_f - x_c = (x_f - x_m) + (x_m - x_c) = e_m + e_c,$$

μας βοηθάει να ελέγξουμε την αποδοτικότητα και καταλληλότητα του υπολογιστικού μοντέλου. Με τον όρο αξιολόγηση της αριθμητικής λύσης εννοούμε τη σύγκριση των λύσεων x_n και x_m που ισοδυναμεί με τον έλεγχο του αριθμητικού σφάλματος.

1.4 Εξάρτηση Σφαλμάτων

Τα διαφορετικά είδη σφαλμάτων εξαρτώνται από παράγοντες που σχετίζονται με την διακριτοποίηση. Με λίγα λόγια, για το αριθμητικό σφάλμα μπορούμε να υποθέσουμε ότι

$$e_n = e_n(h)$$

δηλαδή ότι το σφάλμα e_n εξαρτάται από μια παράμετρο $h > 0$ που υποδηλώνει το μέγεθος διακριτοποίησης.
Ομοίως, για το σφάλμα στρογγυλοποίησης μπορούμε να υποθέσουμε

$$e_r = e_r(v)$$

έτσι ώστε το σφάλμα e_r να εξαρτάται από μια παράμετρο v, που ονομάζεται μονάδα στρογγυλοποίησης, η οποία μετρά πόσο ακριβής είναι ο υπολογιστής μας στην προσέγγιση πραγματικών αριθμών και στην εκτέλεση αλγεβρικών πράξεων.

Το σφάλμα $e_r(u)$ είναι ένα μέτρο που περιγράφει πόσο εξαρτάται ο υπολογισμός από τον ίδιο τον υπολογιστή. Αυτή η ακρίβεια είναι πεπερασμένη, επειδή η μηχανή δεν μπορεί να λειτουργήσει χρησιμοποιώντας ακριβή αριθμητική (όπως θα κάναμε, ιδανικά).

Το πιο περίπλοκο σφάλμα για ποσοτικοποίηση είναι το σφάλμα e_m του μοντέλου, γιατί εξαρτάται από το φυσικό πρόβλημα και από το πόσο καλά το υιοθετημένο μαθηματικό μοντέλο είναι σε θέση να περιγράψει πιστά το φυσικό πρόβλημα. Μόνο η τελική επικύρωση, δηλαδή η σύγκριση της υπολογιστικής λύσης με τις μετρήσεις παρατηρήσιμων φυσικών μεγεθών (για παράδειγμα, η θερμοκρασία που προβλέπεται από ένα μοντέλο πρόγνωσης καιρού και η πραγματική θερμοκρασία που ανιχνεύεται) θα μας πει πόσο πιστό και αξιόπιστο είναι το μαθηματικό μοντέλο και η διακριτοποίηση αυτού.

Κεφάλαιο 2

Παράλληλη Επεξεργασία και OpenMP

2.1 Παράλληλη Επεξεργασία

Ένας από τους πιο κρίσιμους παράγοντες που πρέπει να λάβει υπόψη ένας επιστημονικός κώδικας όταν πρόκειται να επιτύχει υπολογισμούς υψηλής απόδοσης είναι η δυνατότητα παραλληλισμού του [6]. Αυτό συμβαίνει επειδή, στις περισσότερες περιπτώσεις, ο χρόνος εκτέλεσης ενός επιστημονικού κώδικα χρησιμοποιείται για την επίλυση μεγάλων γραμμικών συστημάτων. Δυστυχώς, αυτό είναι συνήθως το πιο δύσκολο κομμάτι του προγραμματισμού, και επομένως είναι σημαντικό να πραγματοποιηθεί η ανάπτυξη αποτελεσματικών παράλληλων αλγορίθμων.

Αν και ένας υπολογιστής κοινόχρηστης μνήμης μπορεί να μας παρέχει ικανοποιητικό αριθμό παράλληλων υπολογισμών που εκτελούνται στους βρόχους του προγράμματος, εξακολουθεί να είναι σημαντικό να λάβουμε υπόψη τους διάφορους παράγοντες που επηρεάζουν την απόδοση του συστήματος. Ένας από αυτούς τους παράγοντες, είναι η διαθεσιμότητα υψηλής ποιότητας χωρικών και χρονικών δεδομένων. Αυτό οφείλεται στο γεγονός ότι, ενώ είναι δυνατή η εκτέλεση πολλών διαδικασιών σε πολλαπλούς επεξεργαστές, δεν αρκεί απλώς να τους αναθέσουμε να τις εκτελούν ταυτόχρονα. Αντίθετα, πρέπει να

διασφαλίσουμε ότι τα δεδομένα που χειρίζονται αποθηκεύονται στην κρυφή μνήμη του επεξεργαστή.

Η χρονική και χωρική εντοπιότητα των δεδομένων σε ένα σύστημα κατανεμημένης μνήμης, όχι μόνο διασφαλίζει την απόδοση του προγράμματος, αλλά γίνεται επίσης μια απαραίτητη προϋπόθεση για τη χρήση πολλών υπολογιστικών κόμβων ταυτόχρονα. Κάθε ένας από αυτούς τους κόμβους εκτελεί στην πραγματικότητα μια ξεχωριστή διαδικασία και μπορεί να έχει πρόσβαση σε πληροφορίες μόνο στην τοπική του μνήμη. Επομένως, ένα πρόγραμμα πρέπει να χωριστεί σε ανεξάρτητες διεργασίες, καθεμια από τις οποίες χρησιμοποιεί τα δικά της δεδομένα, προκειμένου να υλοποιήσει την παράλληλη επεξεργασία. Οι ανταλλαγές δεδομένων μεταξύ των διαφόρων προγραμμάτων πραγματοποιούνται μέσω ενός δικτύου, όταν χρειάζεται να επικοινωνήσουν μεταξύ τους. Ως αποτέλεσμα, η προσέγγιση προγραμματισμού είναι εντελώς διαφορετική. Τα υπάρχων προγράμματα είναι όσα και οι επεξεργαστές. Αυτό έρχεται σε αντίθεση με το μοντέλο παράλληλου προγραμματισμού κοινόχρηστης μνήμης, στο οποίο υπάρχει ένα μόνο πρόγραμμα που υποβάλλεται σε παράλληλη επεξεργασία. Η χρήση βιβλιοθηκών διέλευσης μηνυμάτων επιτρέπει την επικοινωνία δεδομένων μεταξύ προγραμμάτων και τον διαμοιρασμό του προγράμματος σε ξεχωριστές διεργασίες. Ωστόσο, οι πηγαίοι κώδικες για τις διάφορες διεργασίες είναι συνήθως οι ίδιοι στην πράξη.

Χρησιμοποιώντας τις δυνατότητες πολλαπλών εργασιών του λειτουργικού συστήματος, ένα πρόγραμμα που δημιουργήθηκε για ένα κατανεμημένο περιβάλλον μπορεί επίσης να εκτελεστεί σε έναν υπολογιστή κοινόχρηστης μνήμης. Έτσι, η μετάδοση μηνυμάτων θα επιτευχθεί με τη δημιουργία απλών αντιγράφων στη μνήμη.

Οι υβριδικές αρχιτεκτονικές κατανεμημένης και κοινόχρηστης μνήμης χρησιμοποιούνται από πολύ μεγάλα συστήματα υπολογιστών. Κάθε κόμβος έχει πολλαπλούς επεξεργαστές και οι διαφορετικοί κόμβοι συνδέονται μέσω ενός δικτύου επικοινωνιών. Αυτό επιτρέπει τον προγραμματισμό μετάδοσης μηνυμάτων

για κατανεμημένες διεργασίες που χρησιμοποιούν φυσικούς επεξεργαστές ή κατανεμημένες διεργασίες από κόμβους που χρησιμοποιούν πολλαπλούς επεξεργαστές για την επίτευξη τοπικών υπολογισμών.
Επί του παρόντος, οι παραλληλισμοί για κατανεμημένη και κοινόχρηστη μνήμη χρησιμοποιούνται συχνά σε επιστημονικούς κώδικες. Σε περιβάλλοντα κατανεμημένης μνήμης, είναι συνήθως πολύ δύσκολο να προγραμματιστεί η επανεξισορρόπηση φορτίου κατά το χρόνο εκτέλεσης του προγράμματος. Μπορούμε όμως να μειώσουμε τον αριθμό των διεργασιών MPI και να δημιουργήσουμε μια δυναμική ισορροπία φορτίου σε επίπεδο κόμβου χάρη στο προγραμματιστικό μοντέλο του OpenMP.

2.2 Μετρικά Απόδοσης

Βαθμός Παραλληλισμού Ο αριθμός των εργασιών που μπορούν να ολοκληρωθούν ταυτόχρονα αναφέρεται ως βαθμός παραλληλισμού ενός προγράμματος. Η αξιολόγηση των τμημάτων που μπορούν και δεν μπορούν να εφαρμοστούν παράλληλα είναι η παραδοσιακή προσέγγιση του προβληματισμού για την παραλληλοποίηση των επιστημονικών προγραμμάτων. Αυτό περιγράφεται ως το ποσοστό του συνολικού χρόνου εκτέλεσης T και εκφράζεται ως το άθροισμα των παράλληλων και σειριακών χρόνων εκτέλεσης T_t και T_s. Εάν υποθέσουμε ότι το παράλληλο τμήμα του κώδικα διαιρείται τέλεια μεταξύ των διαφόρων νημάτων, έτσι ώστε η παράλληλη εκτέλεση να μην επιφέρει πρόσθετο κόστος, αυτό οδηγεί σε έναν τύπο γνωστό ως νόμο του Amdahl, ο οποίος χρησιμοποιείται για τον προσδιορισμό του ελάχιστου χρόνου εκτέλεσης σε έναν παράλληλο υπολογιστή με nt νήματα. Έτσι προκύπτει η παρακάτω εξίσωση:

$$T_{nt} = T_s + \frac{T_t}{nt}$$

Η επιτάχυνση του προγράμματος αναπαριστάται ως η αναλογία μεταξύ του

σειριακού και παράλληλου χρόνου εκτέλεσης του προγράμματος.

$$speedup(S_{nt}) = \frac{T_s}{T_{nt}} \tag{2.2.1}$$

Η αποδοτικότητα ενός προγράμματος αναπαριστάται ως η αναλογία μεταξύ της επιτεύξιμης επιτάχυνσης και της μέγιστης επιτάχυνσης η οποία είναι ίση με τον αριθμό των νημάτων που χρησιμοποιούνται.

$$efficiency = \frac{S_{nt}}{nt} \tag{2.2.2}$$

Ισχυρή κλιμάκωση Όταν το μέγεθος N ενός προβλήματος παραμένει σταθερό ενώ ο αριθμός των νημάτων αυξάνεται, μια μέθοδος λέγεται ότι είναι ισχυρά κλιμακούμενη. Με άλλα λόγια, μπορούμε να ελπίζουμε ότι η ύπαρξη nt νημάτων θα επιταχύνει τον χρόνο υπολογισμού σε σύγκριση με το να έχουμε μόλις ένα νήμα. Ορίζοντας ως $T(N)$, τον χρόνο που χρειάζεται για να ολοκληρωθεί ένας αλγόριθμος με μέγεθος N, η ισχυρή κλιμάκωση αναπαρίσταται ως εξής:

$$strong\ scalability\ efficiency = \frac{T_1(N)}{nt \times T_{nt}(N)} \tag{2.2.3}$$

2.3 OpenMP

Το OpenMP [7] (Open Specifications for Multi Processing) είναι μια Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface, API) η οποία δημιουργήθηκε από κατασκευαστές υλικού και λογισμικού. Το πρότυπο αυτό προσφέρει στους προγραμματιστές ένα σύνολο από οδηγίες προς το μεταγλωττιστή, οι οποίες ενσωματώνονται στον κώδικα ενός προγράμματος, και έτσι μπορεί ο μεταγλωττιστής να επιτύχει παραλληλισμό στο πρόγραμμα αυτό, σύμφωνα πάντα με τις οδηγίες και παραμέτρους που έχει θέσει ο προγραμματιστής. Το API αποτελείται κυρίως από τρία βασικά συστατικά

- Οδηγίες προς τον μεταγλωττιστή
- Βιβλιοθήκη με υποπρογράμματα
- Μεταβλητές περιβάλλοντος

Το πρότυπο αυτό έχει σχεδιαστεί για τις γλώσσες προγραμματισμού C/C++ και Fortran. Ένα πρόγραμμα που χρησιμοποιεί τις δυνατότητες του OpenMP είναι μεταφέρσιμο και υποστηρίζεται για αρκετές πλατφόρμες, μέσα στις οποίες περιλαμβάνονται διάφορες διανομές Linux καθώς και Windows. Εάν μια πλατφόρμα δεν υποστηρίζει τις δυνατότητες του OpenMP, ο μεταγλωττιστής αγνοεί τις οδηγίες, με αποτέλεσμα το πρόγραμμα να εκτελεστεί σειριακά. Έτσι, ο προγραμματιστής δεν χρειάζεται να μεταβάλει τα περιεχόμενα του κώδικα. Ο παραλληλισμός που προσφέρει αφορά την αρχιτεκτονική συστημάτων διαμοιραζόμενης μνήμης, οπότε δεν είναι δυνατόν να εφαρμοστεί σε συστήματα κατανεμημένης μνήμης. Επίσης οι εκάστοτε υλοποιήσεις δεν είναι απαραίτητα υλοποιημένες τέλεια, και δεν μπορεί να αξιοποιήσει την διαμοιραζόμενη μνήμη με τον καλύτερο δυνατό τρόπο, αλλά για τις ανάγκες αυτής της έρευνας οι δυνατότητες που μας προσφέρει είναι αρκετές.

Στόχοι του OpenMP Ο στόχος του OpenMP είναι να παρέχει ένα απλό και περιορισμένο σύνολο από οδηγίες και υποπρογράμματα με τα οποία θα μπορεί να γίνεται σημαντική παραλληλοποίηση του κώδικα. Πιο συγκεκριμένα μας παρέχει:

- Τη δυνατότητα παραλληλοποίησης ενός προγράμματος σταδιακά, σε αντίθεση με τη βιβλιοθήκη MPI [8] που κάθε φορά συμπεριλαμβάνει όλο το περιεχόμενο της.
- Τη δυνατότητα τόσο αδρομερούς (coarse-grain) όσο και λεπτομερούς (fine-grain) παραλληλισμού.

Τέλος, ο κυριότερος στόχος του OpenMP είναι να υποστηρίζεται από διάφορες

πλατφόρμες και να είναι μεταφέρσιμο.

Το μοντέλο προγραμματισμού του OpenMP Το OpenMP βασίζεται στο πολυνηματικό μοντέλο παραλληλισμού. Αυτό πρακτικά σημαίνει ότι η εφαρμογή του προγραμματιστή ξεκινάει την εκτέλεση της χρησιμοποιώντας μόνο το κύριο νήμα, η αλλιώς master thread, και στην πορεία όταν συναντήσει ένα παράλληλο τμήμα(parallel region) ορισμένο από τον προγραμματιστή, δημιουργεί αρκετά νήματα και εκτελεί αυτό το τμήμα παράλληλα(μοντέλο fork-join). Μόλις ολοκληρωθεί η εκτέλεση αυτού του τμήματος, τα νήματα τερματίζουν και συνεχίζει την εκτέλεση το κύριο νήμα. Ο συγχρονισμός των νημάτων, δηλαδή με ποια σειρά θα ξεκινήσουν τα νήματα την εκτέλεση του τμήματος κώδικα της παράλληλης περιοχής, και με πια σειρά θα τερματίσουν, είναι ένα πρόβλημα το οποίο δεν εγγυάται το OpenMP και είναι ευθύνη του προγραμματιστή.

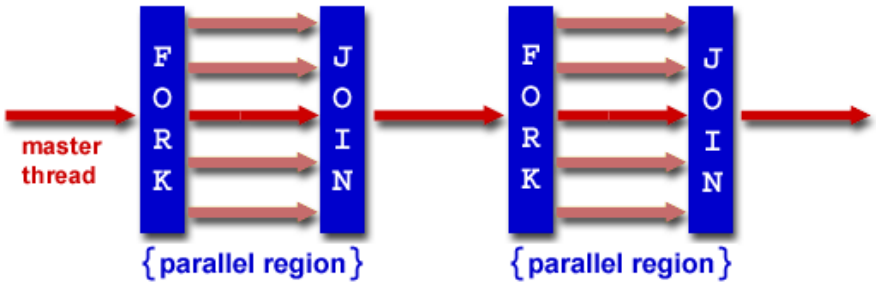

Figure 2.1: Το κύριο νήμα (master thread) δημιουργεί ένα σύνολο νημάτων όταν συναντήσει μια παράλληλη περιοχή, και συνεχίζει την εκτέλεση του σειριακά. Εικόνα: https://users.fit.cvut.cz/~soch/mi-par/openmp/OpenMP_soubory/fork_join1.gif

Επειδή το OpenMP κάνει χρήση της διαμοιραζόμενης μνήμης, τα νήματα μπορούν να έχουν πρόσβαση στη κύρια μνήμη, αλλά και να έχουν την δική τους ιδιωτική μνήμη(thread-private memory) όπου τα υπόλοιπα νήματα δεν έχουν πρόσβαση. Τα νήματα μπορεί να έχουν μια προσωρινή όψη της μνήμης έτσι ώστε να μην γίνονται συνεχόμενες προσπελάσεις στην κύρια μνήμη. Γενικά ισχύουν τα εξής:

- Σε μια οδηγία παραλληλισμού μπορεί να ορίζονται διαμοιραζόμενες και ιδι-

ωτικές μεταβλητές. Κάθε αναφορά σε διαμοιραζόμενη μεταβλητή είναι μια αναφορά στην ίδια την μεταβλητή, ενώ μια αναφορά σε ιδιωτική μεταβλητή είναι μια αναφορά σε ένα τοπικό αντίγραφο στην ιδιωτική μνήμη του νήματος.

- Για την προσπέλαση των διαμοιραζόμενων μεταβλητών από διαφορετικά νήματα απαιτείται συγχρονισμός.
- Είναι πιθανό σε κάποιες υλοποιήσεις η πρόσβαση σε διαμοιραζόμενες μεταβλητές να είναι ατομική ενέργεια, και να μην χρειάζεται παρέμβαση από τον προγραμματιστή.
- Όταν έχουμε εμφωλευμένες παράλληλες οδηγίες, ο ρόλος μιας μεταβλητής σε μια παράλληλη περιοχή μπορεί να είναι διαφορετικός από τον ρόλο σε μια άλλη. Για παράδειγμα, μια μεταβλητή που είναι ιδιωτική στην εξωτερική παράλληλη περιοχή, μπορεί να είναι διαμοιραζόμενη στην εσωτερική παράλληλη περιοχή, εκτός αν έχει ορισθεί διαφορετικά.

Οδηγίες OpenMP

Γενικά όταν συντάσσουμε οδηγίες που μας παρέχει το OpenMP, θα πρέπει πριν κάθε οδηγία, να συμπεριλάβουμε το #pragma omp, το οποίο δηλώνει πως οι οδηγίες και οι παράμετροι που ακολουθούν θα χρησιμοποιηθούν από το OpenMP για να εκτελεστεί το τμήμα κώδικα παράλληλα. Ορισμένοι κανόνες που πρέπει να ισχύουν για κάθε οδηγία είναι:

- Να είναι case sensitive.
- Να ακολουθούν τις συμβάσεις της C++ για τις οδηγίες προς τον μεταγλωττιστή.
- Να υπάρχει μόνο ένα όνομα οδηγίας σε κάθε οδηγία.
- Η οδηγία έχει ισχύ μόνο στο τμήμα εντολών που ακολουθεί.

Οδηγία parallel Η οδηγία parallel είναι η πιο βασική οδηγία από όλες, γιατί αυτή είναι που ορίζει την περιοχή κώδικα που θέλουμε να εκτελεστεί παράλληλα. Η σύνταξη της είναι

```
#pragma omp parallel [clause ...] newline
                     if (scalar_expression)
                     private (list)
                     shared (list)
                     default (shared | none)
                     firstprivate (list)
                     reduction (operator: list)
                     copyin (list)
                     num_threads (integer-expression)
```

Όταν το κύριο νήμα φτάσει σε μια οδηγία παράλληλης περιοχής, δημιουργεί μια ομάδα από νήματα και το ίδιο γίνεται το master της ομάδας. Το master νήμα συμμετέχει στη εκτέλεση του τμήματος όπως και τα άλλα νήματα. Το τμήμα κώδικα της παράλληλης περιοχής αντιγράφεται για κάθε νήμα και εκτελείται για όλα τα νήματα που έχουν οριστεί από τον προγραμματιστή. Γενικά δεν υπάρχει συγχρονισμός μεταξύ των νημάτων, οπότε κάθε νήμα μπορεί να φτάσει σε οποιοδήποτε σημείο μέσα στη παράλληλη περιοχή, σε τυχαία χρονική στιγμή. Στο τέλος της παράλληλης περιοχής πρέπει να υπάρχει ένα φράγμα, πέρα από το οποίο κανένα νήμα δεν θα μπορεί να συνεχίσει παρά μόνο το master νήμα. Το OpenMP παρέχει τη δυνατότητα δημιουργίας παράλληλης περιοχής μέσα σε μια άλλη παράλληλη περιοχή. Η εμφωλευμένη αυτή παράλληλη περιοχή, καταλήγει στη δημιουργία μιας καινούριας ομάδας από νήματα η οποία αποτελείται εξ' ορισμού από ένα νήμα. Ωστόσο, διάφορες υλοποιήσεις μπορούν να επιτρέψουν παραπάνω από ένα νήμα σε μια εμφωλευμένη παράλληλη περιοχή.

Ο αριθμός των νημάτων που δημιουργούνται κατά την είσοδο σε μια παράλληλη περιοχή μπορεί να καθοριστεί από τρεις παράγοντες, οι οποίοι κατά σειρά

προτεραιότητας είναι:

- Η χρήση της συνάρτησης omp_set_num_threads().
- Η τιμή της μεταβλητής περιβάλλοντος OMP_NUM_THREADS.
- Η χρήση της φράσης default, η οποία είναι και η προκαθορισμένη.

Επίσης, υπάρχει η δυνατότητα δυναμικής προσαρμογής του αριθμού των νημάτων για μια συγκεκριμένη παράλληλη περιοχή. Αυτό μπορεί να επιτευχθεί με τους εξής δύο τρόπους:

- Η χρήση της συνάρτησης omp_set_dynamic().
- Η τιμή της μεταβλητής περιβάλλοντος MP_DYNAMIC.

Η αρίθμηση των νημάτων ξεκινάει από τον αριθμό 0, ο οποίος δίνεται στον master κάθε ομάδας και φτάνει στον αριθμό N-1 όπου N ο αριθμός των νημάτων. Η ταυτότητα κάθε νήματος μπορεί να βρεθεί χρησιμοποιώντας τη συνάρτηση omp_get_thread_num(), ενώ ο συνολικός αριθμός των νημάτων μπορεί να βρεθεί με τη χρήση της συνάρτησης omp_get_num_threads(). Αν και τα νήματα εκτελούν το ίδιο τμήμα κώδικα, ωστόσο, θα θέλαμε να εκτελέσουν διαφορετικά μονοπάτια αυτού του κώδικα. Αυτό επιτυγχάνεται με το να δίνουμε διαφορετικό κομμάτι κώδικα σε κάθε νήμα χρησιμοποιώντας συνθήκες if-else και την ταυτότητα του νήματος.

Οδηγίες Διαμοιρασμού Εργασίας Μια οδηγία διαμοιρασμού εργασίας κατανέμει την εκτέλεση του κώδικα που περικλείεται στην παράλληλη περιοχή μεταξύ των νημάτων της περιοχής. Οι οδηγίες διαμοιρασμού εργασίας δεν δημιουργούν καινούρια νήματα και δεν υπάρχει κάποιος συγχρονισμός κατά την είσοδο σε μια τέτοια οδηγία, υπάρχει όμως συγχρονισμός κατά την έξοδο

(εισάγεται φράγμα). Υπάρχουν τρεις τύποι οδηγιών διαμοιρασμού εργασίας:

- **FOR**: Διαμοιράζει τις επαναλήψεις ενός βρόχου for στα νήματα της τρέχουσας παράλληλης περιοχής. Αντιστοιχεί στο μοντέλο Παραλληλισμού Δεδομένων.
- **SECTIONS**: Διαμοιράζει την εργασία σε διακριτά δομικά τμήματα. Κάθε τμήμα εκτελείται από ένα νήμα. Αντιστοιχεί στο μοντέλο Συναρτησιακού Παραλληλισμού.
- **SINGLE**: Σειριοποιεί ένα τμήμα του κώδικα ώστε να εκτελεστεί υποχρεωτικά από ένα νήμα.

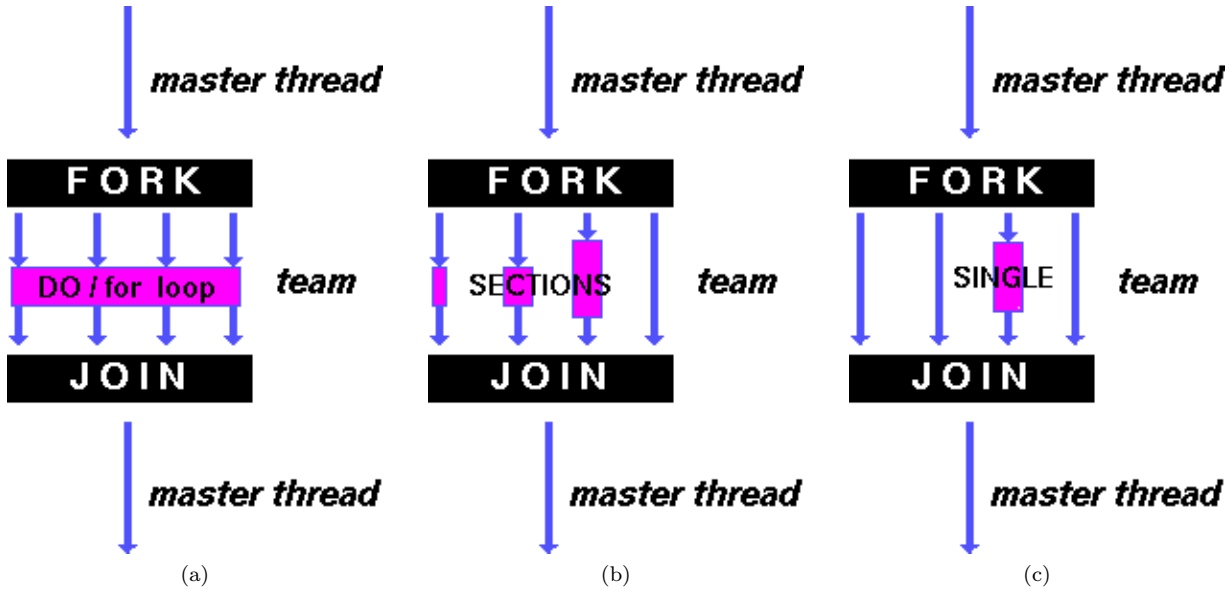

Figure 2.2: Οδηγίες διαμοιρασμού εργασίας: (a) FOR (b) SECTION (c) SINGLE. Εικόνα: https://hpc-tutorials.llnl.gov/openmp/images/work_share1.gif

βέβαια υπάρχουν και ορισμένοι περιορισμοί οι οποίοι είναι

- Για να εκτελεσθεί παράλληλα μια οδηγία διαμοιρασμού εργασίας, θα πρέπει να εσωκλείεται σε μια παράλληλη περιοχή.

- Οι περιοχές διαμοιρασμού εργασίας θα πρέπει να διαμοιράζουν τα δεδομένα (ή την εργασία) σε όλα τα νήματα ή σε ένα.
- Διαδοχικές περιοχές διαμοιρασμού εργασίας θα πρέπει να προσπελάζονται από τα μέλη μιας ομάδας με την ίδια σειρά.

Οδηγία for Χρησιμοποιώντας την οδηγία αυτή μπορούμε να επιτύχουμε παραλληλισμό των δεδομένων. Όλα τα νήματα θα εκτελέσουν το ίδιο τμήμα κώδικα, αλλά σε διαφορετικά δεδομένα. Προϋποθέτει ότι βρισκόμαστε μέσα σε μια παράλληλη περιοχή. Η σύνταξη της οδηγίας είναι

```
#pragma omp for [clause ...]  newline
                schedule (type [,chunk])
                ordered
                private (list)
                firstprivate (list)
                lastprivate (list)
                shared (list)
                reduction (operator: list)
                collapse (n)
                nowait

    for_loop
```

Η οδηγία χρησιμοποιείται για την εκτέλεση του βρόχου for παράλληλα. Με την schedule μπορούμε να ορίσουμε πώς θα γίνει ο διαμοιρασμός των επαναλήψεων του βρόχου for στα διαθέσιμα νήματα. Τα ορίσματα που μπορεί να πάρει είναι τα παρακάτω:

- **static:** Οι επαναλήψεις του βρόχου χωρίζονται σε ομάδες με μέγεθος chunk, και μοιράζονται στατικά στα νήματα. Δηλαδή, πριν αρχίσει η εκτέλεση του βρόχου for, κάθε νήμα γνωρίζει πόσες και ποίες επαναλήψεις

θα εκτελέσει. Αν δεν οριστεί το μέγεθος της ομάδας, τότε ο διαμοιρασμός των επαναλήψεων γίνεται αυτόματα στα διαθέσιμα νήματα.

- **dynamic:** Αυτή η παράμετρος είναι παρόμοια με την static. Οι επαναλήψεις χωρίζονται σε ομάδες με μέγεθος chunk και σε αυτή την περίπτωση μοιράζονται δυναμικά στα διαθέσιμα νήματα. Αν ο προγραμματιστής δεν ορίσει το μέγεθος της ομάδας, τότε η κάθε ομάδα θα έχει μέγεθος ένα, οπότε θα περιέχει μια επανάληψη του βρόχου. Όταν ένα νήμα ολοκληρώσει την εκτέλεση των επαναλήψεων μιας ομάδας, στο ίδιο νήμα, γίνεται δυναμική ανάθεση των επαναλήψεων μιας άλλης ομάδας.
- **guided:** Το μέγεθος κάθε ομάδας μειώνεται εκθετικά, καθώς γίνονται οι αναθέσεις των κομματιών του χώρου επανάληψης. Η τιμή chunk ορίζει το μικρότερο κομμάτι στο οποίο μπορεί να σπάσει. Η προκαθορισμένη τιμή είναι 1.
- **runtime:** Η απόφαση σχεδιασμού αναβάλλεται μέχρι να αρχίσει να εκτελείται το πρόγραμμα. Με αυτήν την παράμετρο δεν επιτρέπεται να οριστεί το chunk.

Για τη σωστή λειτουργία της οδηγίας for, θα πρέπει η μεταβλητή βήματος του βρόχου for να αυξάνεται με έναν σταθερό ρυθμό και να μην προκύπτει ως αποτέλεσμα μιας πράξης που περιλαμβάνει μεταβλητές. Επίσης, η τιμή της συνθήκης τερματισμού του βρόχου πρέπει να είναι σταθερή, ώστε ο βρόχος να τερματίζει μετά από έναν πεπερασμένο αριθμό επαναλήψεων.

Οδηγία barrier Η οδηγία barrier συγχρονίζει όλα τα νήματα της ομάδας, απαιτώντας από κάθε νήμα να σταματήσει προσωρινά την εκτέλεσή του, στο σημείο όπου υπάρχει η οδηγία barrier, μέχρις ότου όλα τα νήματα φτάσουν σε αυτό το σημείο. Στη συνέχεια, όλα τα νήματα ξεκινούν παράλληλα, από εκείνο το σημείο, την εκτέλεσή του κώδικα που ακολουθεί. Η σύνταξη της οδηγίας barrier είναι η παρακάτω:

```
#pragma omp barrier
```

Προκειμένου να εφαρμοστεί σωστά η οδηγία barrier, πρέπει να αντιμετωπιστεί από όλα τα νήματα ή από κανένα, και η σειρά με την οποία τα νήματα προσπελαύνουν τις οδηγίες διαμοιρασμού εργασίας και τις οδηγίες barrier σε μια παράλληλη περιοχή πρέπει να είναι ίδια για κάθε νήμα της ομάδας.

Οδηγία master Η οδηγία αυτή ορίζει ένα τμήμα κώδικα το οποίο θα εκτελεστεί από το master thread μόνο. Δεν υπάρχει κάποιο φράγμα στο τέλος του για τα υπόλοιπα νήματα, τα οποία απλά προσπερνούν την οδηγία αυτή. Η σύνταξη της οδηγίας master είναι η παρακάτω:

```
#pragma omp master
```

Συναρτήσεις συστήματος εκτέλεσης

omp_set_num_threads Η συνάρτηση αυτή ορίζει το πλήθος των νημάτων που θα δημιουργηθούν σε μια παράλληλη περιοχή (θετικός ακέραιος). Πρέπει να κληθεί σε μέρη του κώδικα που εκτελούνται σειριακά (από το master thread). Η συνάρτηση αυτή υπερέχει της μεταβλητής περιβάλλοντος OMP_NUM_THREADS. Επίσης, αν η δυνατότητα για δυναμική δημιουργία νημάτων είναι ενεργοποιημένη, η συνάρτηση δεν ορίζει τον αριθμό των νημάτων που θα δημιουργηθούν, αλλά το μέγιστο πλήθος που μπορούν να δημιουργηθούν.

```
#include <omp.h>
void omp_set_num_threads(int num_threads)
```

omp_get_num_threads Η συνάρτηση αυτή επιστρέφει το πλήθος των νημάτων που εκτελούνται εκείνη τη στιγμή. Σε μια παράλληλη περιοχή επιστρέφει τον αριθμό των νημάτων που δημιουργήθηκαν για την περιοχή αυτή, ενώ σε μια σειριακή περιοχή επιστρέφει την τιμή 1.

```
#include <omp.h>
int omp_get_num_threads(void)
```

omp_get_max_threads Η συνάρτηση αυτή επιστρέφει την μέγιστη τιμή που η προηγούμενη συνάρτηση (OMP_GET_NUM_THREADS) μπορεί να επιστρέψει. Πρακτικά επιστρέφει την τιμή της μεταβλητής περιβάλλοντος OMP_NUM_THREADS ή την τιμή του τέθηκε από τη συνάρτηση omp_set_num_threads().

```
#include <omp.h>
int omp_get_max_threads(void)
```

omp_get_thread_num Η συνάρτηση αυτή επιστρέφει τον αριθμό του νήματος που την καλεί. Το master thread έχει την τιμή 0. Αν κληθεί σε μια σειριακή περιοχή ή σε μια εμφωλευμένη παράλληλη περιοχή θα επιστρέψει την τιμή 0.

```
#include <omp.h>
int omp_get_thread_num(void)
```

omp_get_thread_limit Αυτό το υποπρόγραμμα μάς επιστρέφει τον μέγιστο αριθμό των νημάτων που είναι διαθέσιμα στο πρόγραμμα.

```
#include <omp.h>
int omp_get_thread_limit (void)
```

omp_get_num_procs Αυτό το υποπρόγραμμα μάς επιστρέφει τον αριθμό των διαθέσιμων επεξεργαστών στο πρόγραμμα.

```
#include <omp.h>
int omp_get_num_procs(void)
```

Φράσεις Εμβέλειας Δηλώσεων Δεδομένων

Ονομάζονται και φράσεις διαμοιρασμού δεδομένων και χρησιμοποιούνται σε συνδυασμό με διάφορες οδηγίες όπως η parallel, η for, και η sections, για τον

ορισμό διαμοιραζόμενων και ιδιωτικών μεταβλητών αλλά και της εμβέλειας τους. Οι περισσότερες μεταβλητές είναι εξ' ορισμού διαμοιραζόμενες εφόσον το OpenMP είναι υλοποιημένο για συστήματα διαμοιραζόμενης μνήμης, με εξαίρεση τα ορίσματα συναρτήσεων που καλούνται μέσα σε παράλληλες περιοχές, και τους μετρητές βρόχων for που είναι στην ουσία ιδιωτικές μεταβλητές. Οι φράσεις εμβέλειας που χρησιμοποιήθηκαν σε αυτήν την έρευνα είναι η private και η shared, οι οποίες ορίζουν ποιες μεταβλητές της σειριακής περιοχής μεταφέρουν τις τιμές τους προς/από τη παράλληλη περιοχή και με ποίον τρόπο. Επίσης, ορίζουν ποιες μεταβλητές θα είναι ορατές σε όλα τα νήματα και ποιες θα ανήκουν σε συγκεκριμένο νήμα. Τέλος, οποιαδήποτε φράση εμβέλειας και να χρησιμοποιήσουμε, θα έχει ισχύει μόνο στα όρια των οδηγιών που τη συνοδεύουν.

Φράση private Με τη φράση αυτή μπορούμε να ορίσουμε μεταβλητές ιδιωτικές για κάθε νήμα. Αυτό σημαίνει ότι κάθε νήμα έχει ένα δικό του αντίγραφο της μεταβλητής στην ιδιωτική του μνήμη. Όλες οι αναφορές στις μεταβλητές private αλλάζουν και μετατρέπονται σε αναφορές στα αντίγραφα των μεταβλητών. Τα αντίγραφα των μεταβλητών δεν αρχικοποιούνται στη δημιουργία των νημάτων, εκτός και αν χρησιμοποιηθεί η φράση firstprivate.

```
private (list)
```

Φράση shared Σε αντίθεση με την προηγούμενη φράση, η φράση shared ορίζει μεταβλητές ως διαμοιραζόμενες σε όλα τα νήματα. Αυτό σημαίνει ότι αν ένα νήμα αλλάξει την τιμή μιας τέτοιας μεταβλητής, η αλλαγή θα είναι ορατή σε όλα τα νήματα. Εδώ προκύπτει το φαινόμενο της συνθήκης ανταγωνισμού, από το οποίο δημιουργείται το πρόβλημα της χαμένης ενημέρωσης, αν περισσότερα από ένα νήματα γράψουν στην ίδια περιοχή μνήμης. Τα νήματα δεν έχουν τοπικό αντίγραφο της μεταβλητής αυτής, οπότε χρησιμοποιούν την θέση της στη διαμοιραζόμενη μνήμη. Ο προγραμματιστής είναι υπεύθυνος για να φροντίσει να μη δημιουργηθούν προβλήματα ταυτόχρονης ανάγνωσης και εγγραφής της

ίδιας μεταβλητής από πολλά νήματα. Οι μεταβλητές shared διατηρούν την τιμή τους κατά την έξοδο από παράλληλη περιοχή.

```
shared (list)
```

Κεφάλαιο 3

Αριθμητικές Μέθοδοι Γραμμικών Συστημάτων

Κατά την αριθμητική διακριτοποίηση των μαθηματικών μοντέλων, συνήθως οδηγούμαστε σε γραμμικά συστήματα υψηλής τάξης n, δηλαδή $Ax = b$, όπου ο $A \in \mathbb{R}^{n \times n}$ είναι ο πίνακας συντελεστών, $x \in \mathbb{R}^n$ είναι το διάνυσμα αγνώστων μεταβλητών, και $b \in \mathbb{R}^n$ το δεξιό μέλος του συστήματος. Τέτοια γραμμικά συστήματα συνήθως προκύπτουν έπειτα από την διακριτοποίηση διαφόρων προβλημάτων της φυσικής, τα οποία καθώς αυξάνεται η πολυπλοκότητα τους, αυξάνεται και η τάξη του εκάστοτε γραμμικού συστήματος. Για αυτό το λόγο, χρειαζόμαστε αποδοτικές αριθμητικές μεθόδους που αξιοποιούν κατά τον καλύτερο δυνατό τρόπο τις δυνατότητες των υπολογιστών.

3.1 Εισαγωγή στις Γενικές Επαναληπτικές Μεθόδους

Γενικά στην αριθμητική επίλυση γραμμικών συστημάτων, υπάρχουν δύο κατηγορίες αριθμητικών μεθόδων, οι άμεσες και οι επαναληπτικές. Οι άμεσες μέθοδοι, λόγω της αργής τους επίδοσης, εφαρμόζονται σε γραμμικά συστήματα μικρής τάξης στα οποία τα περισσότερα στοιχεία του πίνακα συντελεστών A είναι μη μηδενικά, και συχνά απαιτούν ένα πεπερασμένο πλήθος πράξεων για τον υπολογισμό της λύσης του συστήματος, η ακρίβεια της οποίας, είναι υψηλότερη

από αυτή των επαναληπτικών μεθόδων. Οι επαναληπτικές μέθοδοι, αντιθέτως, χρησιμοποιούνται κυρίως όταν ο πίνακας συντελεστών A είναι μεγάλης τάξης και τα περισσότερα στοιχεία του είναι μηδενικά. Σε ορισμένες περιπτώσεις, οι επαναληπτικές μέθοδοι εφαρμόζονται και όταν γνωρίζουμε εκ των προτέρων ότι η προσεγγιστική λύση τους θα βρεθεί γρήγορα.
Παρακάτω, θα περιγράψουμε την γενική μεθοδολογία, κατά την οποία μπορούν να κατασκευαστούν οι επαναληπτικές μέθοδοι. Θα κάνουμε ιδιαίτερη ανάλυση σε δύο από αυτές, οι οποίες είναι οι Jacobi και Gauss Seidel.
Θεωρούμε το παρακάτω γραμμικό σύστημα

$$Ax = b,\ A \in \mathbb{R}^{n\times n},\ x, b \in \mathbb{R}^n \tag{3.1.1}$$

Η γενική ιδέα των επαναληπτικών μεθόδων είναι να ξεκινήσουμε από μια αρχική προσεγγιστική λύση $x^{(0)}$, να κατασκευάσουμε μέσω μιας διαδικασίας μια ακολουθία διανυσμάτων $x^{(k)},\ k = 1, .., ,$ η οποία αν πληρεί κάποια κριτήρια, θα συγκλίνει στην ακριβή λύση x του (3.1.1). Ο πίνακας A μπορεί να γραφεί σαν τη διαφορά δύο άλλων πινάκων M και N

$$A = M - N \tag{3.1.2}$$

Κάποιοι περιορισμοί που θα πρέπει να ισχύουν για τον πίνακα M είναι οι

- Θα πρέπει ο πίνακας M να είναι αντιστρέψιμος. Αυτό σημαίνει ότι θα υπάρχει ο αντίστροφος του M^{-1}.
- Θα πρέπει το σύστημα $Mx = b$ να λύνεται με λιγότερες πράξεις σε σύγκριση με το γραμμικό σύστημα $Ax = b$.

Η εξίσωση (3.1.1), με εισαγωγή της εξίσωσης (3.1.2) μπορεί να γραφεί με τις παρακάτω δύο μορφές

$$(M - N)x = b \ \text{ ή } \ Mx = Nx + b.$$

Έχοντας το αρχικό διάνυσμα $x^{(0)}$, μπορεί να δημιουργηθεί μια ακολουθία διανυσμάτων με τον επαναληπτικό τύπο

$$Mx^{(k+1)} = Nx^{(k)} + b,\ k = 0, 1, ...$$

Πολλαπλασιάζοντας με τον αντίστροφο του πίνακα M, ο παραπάνω επαναληπτικός τύπος μετατρέπεται στον τύπο

$$x^{(k+1)} = Bx^{(k)} + c \tag{3.1.3}$$

όπου $B = M^{-1}N, c = M^{-1}b$ και $x^{(0)}$ το δοθέν αρχικό διάνυσμα.

Θεώρημα 3.1.1 *Ο επαναληπτικός τύπος (3.1.3) ορίζει μια ακολουθία διανυσμάτων $x^{(k)},\ k = 1, ...$ που αν συγκλίνει, θα συγκλίνει στην λύση του γραμμικού συστήματος (3.1.1), [9]*

Απόδειξη Παίρνοντας τα όρια του επαναληπτικού τύπου (3.1.3) και έχοντας ότι $y = \lim_{k\to\infty} x^{(k)}$, θα έχουμε

$$y = M^{-1}Ny + M^{-1}b$$

το οποίο μπορεί να γραφεί ως εξής

$$(M - N)y = b$$

όπου $(M - N) = A$. Επειδή η εξίσωση (3.1.1) έχει μια και μοναδική λύση, το όριο y της ακολουθίας διανυσμάτων $x^{(k)}$ συμπίπτει με αυτή.

□

Θεώρημα 3.1.2 *Ικανή και αναγκαία συνθήκη για τη σύγκλιση του επαναληπτικού τύπου (3.1.3) στην ακριβή λύση του γραμμικού συστήματος (3.1.1) είναι*

$$\rho(B) < 1$$

όπου $B = M^{-1}N$ *και με* ρ *συμβολίζεται η φασματική ακτίνα του πίνακα* B*, δες [9].*

Σημείωση *Η φασματική ακτίνα ενός πίνακα είναι μια κλιμακωτή τιμή που αντιπροσωπεύει τη μεγαλύτερη απόλυτη τιμή οποιασδήποτε ιδιοτιμής του πίνακα. Με άλλα λόγια, εάν ο πίνακας* A *είναι τετραγωνικός, η φασματική ακτίνα είναι το μέγιστο μέγεθος μεταξύ των απόλυτων τιμών όλων των ιδιοτιμών του* A*. Δηλαδή αν,* $\lambda_1, \lambda_2, ..., \lambda_n$ *είναι οι ιδιοτιμές του πίνακα* A*, τότε η φασματική ακτίνα εκφράζεται με τον ακόλουθο τρόπο*

$$\rho(A) = \max_{1 \le i \le n} |\lambda_i|.$$

Απόδειξη Έχουμε $e^{(k)} = x^{(k)} - x$ το σφάλμα μιας τυχαίας επανάληψης k και x η ακριβής λύση του (3.1.1). Αφαιρώντας τις σχέσεις (3.1.2) και (3.1.3) και πολλαπλασιάζοντας επί M^{-1}, παίρνουμε

$$x^{(k+1)} - x = M^{-1}N(x^{(k)} - x)$$

από την οποία προκύπτει η σχέση $e^{(k+1)} = Be^{(k)}$, όπου επαγωγικά μπορούμε να πάρουμε την $e^{(k)} = B^k e^{(0)}$. Για να συγκλίνει ο επαναληπτικός τύπος (3.1.3) στη ακριβή λύση του γραμμικού συστήματος (3.1.1), θα πρέπει να ισχύει $\lim_{k\to\infty} x^{(k)} = A^{-1}b$, όπου $A^{-1}b$ η ακριβής λύση. Αν στη σχέση $e^{(k)} = B^k e^{(0)}$, αντί για $e^{(0)}$ θέσουμε τις στήλες του πίνακα I, τότε παίρνουμε τις στήλες του πίνακα B^k. Επειδή ισχύει $\lim_{k\to\infty} e^{(k)} = 0$, τότε οι στήλες του πίνακα B^k θα τείνουν στο μηδενικό διάνυσμα. Η τελευταία σχέση ισχύει μονό αν $\rho(B) < 1$ και έτσι το θεώρημα αποδείχθηκε.

□

Πόρισμα 3.1.1 *μια ικανή συνθήκη [9] για την σύγκλιση του επαναληπτικού τύπου (3.1.3) στην ακριβή λύση του γραμμικού συστήματος, είναι για οποιαδήποτε νόρμα του πίνακα B (μέγιστη νόρμα, ευκλείδεια νόρμα) να ισχύει*

$$||B|| < 1$$

Επειδή ισχύει ότι $\rho(B) \leq ||B||$ το πόρισμα αποδείχθηκε από το θεώρημα (??).

Πόρισμα 3.1.2 *Αν για κάποια νόρμα και για κάποιο θετικό ακέραιο k για τον πίνακα B ισχύει ότι*

$$||B^k|| < 1$$

τότε ο επαναληπτικός τύπος (3.1.3) συγκλίνει, δες [9].

Απόδειξη Επειδή ισχύει ότι

$$\rho^k(B) = \rho(B^k) \leq ||B^k|| < 1$$

άρα $\rho(B) < 1$ και επομένως ο επαναληπτικός τύπος συγκλίνει.
□

Εκτίμηση σφάλματος

Για οποιαδήποτε νόρμα διανύσματος, και την φυσική νόρμα πίνακα, έχουμε την εκτίμηση σφάλματος

$$||x^{(k)} - x|| \leq ||B^k||\ ||x^{(0)} - x|| \leq ||B||^k\ ||x^{(0)} - x||$$

Αν $||B|| \leq 1$ τότε μπορεί να ισχύει

$$||x^{(k)} - x|| \leq \frac{||B||}{1 - ||B||} ||x^{(k)} - x^{(k-1)}||$$

Από την προηγούμενη σχέση προκύπτει η παρακάτω σχέση

$$||x^{(k)} - x|| \leq \frac{||B||^k}{1 - ||B||}||x^{(1)} - x^{(0)}||$$

Οι δύο τελευταίες εκτιμήσεις είναι χρήσιμες στην πράξη, διότι οι νόρμες διανυσμάτων $||x^{(k)} - x^{(k+1)}||$ και $||x^{(1)} - x^{(0)}||$ μπορούν εύκολα να υπολογιστούν.

3.2 Γενική Επαναληπτική μέθοδος Jacobi

Έστω το $n \times n$ γραμμικό σύστημα $Ax = b$. Ο πίνακας A είναι ομαλός, δηλαδή υπάρχει πίνακας A^{-1} για τον οποίο ισχύει $A \cdot A^{-1} = A^{-1} \cdot A = I$, και ικανοποιεί τους περιορισμούς $a_{ii} \neq 0,\ i = 1, 2, ..., n$. Επίσης, είναι αυστηρά διαγώνια υπέρτερος κατά γραμμή, ιδιότητα που όπως θα δούμε στη συνέχεια, για κάθε αρχικό διάνυσμα $x^{(0)}$, η επαναληπτική μέθοδος συγκλίνει.

Θεώρημα 3.2.1 *Αν ο πίνακας συντελεστών $A \in \mathbb{R}^{n \times n}$ είναι αυστηρά διαγώνια υπέρτερος κατά γραμμές [3], δηλαδή ισχύει*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, i = 1, 2, ..., n.$$

τότε η επαναληπτική μέθοδος Jacobi θα συγκλίνει στην ακριβή λύση του συστήματος (3.1.1).

Ακολουθώντας το (3.1.2) ο πίνακας A μπορεί να γραφεί στην μορφή

$$A = D + (L + U),$$

όπου ο D είναι ο διαγώνιος πίνακας με στοιχεία τα διαγώνια στοιχεία του A, και ο $L,\ U$ οι αυστηρά κάτω και ο άνω τριγωνικοί πίνακες του A. Πιο αναλυτικά, μπορούμε να τους γράψουμε ως εξής

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix},\ L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix},\ U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Η επαναληπτική προσεγγιστική μέθοδος Jacobi είναι η

$$x^{(k+1)} = Bx^{(k)} + c, k = 0, 1, ... \tag{3.2.1}$$

όπου

$$B = -D^{-1}(L + U),\ c = D^{-1}b$$

και $x^{(0)}$ το αρχικό διάνυσμα της λύσης. Ο πίνακας B αποτελείται από

$$D^{-1} = \begin{bmatrix} a_{11}^{-1} & 0 & \cdots & 0 \\ 0 & a_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{-1} \end{bmatrix},\ -(L+U) = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix}$$

Ο παραπάνω επαναληπτικός τύπος γράφεται με την παρακάτω αναλυτική μορφή,

$$x^{(k+1)} = \begin{bmatrix} 0 & -\frac{a_1 2}{a_1 1} & \cdots & -\frac{-a_1 n}{a_1 1} \\ -\frac{a_2 1}{a_2 2} & 0 & \cdots & -\frac{-a_2 n}{a_2 2} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_n 1}{a_n n} & -\frac{a_n 2}{a_n n} & \cdots & 0 \end{bmatrix} \cdot x^{(k)} + \begin{bmatrix} \frac{b_1}{a_1 1} \\ \frac{b_2}{a_2 2} \\ \vdots \\ \frac{b_n}{a_n n} \end{bmatrix}$$

Ένας εναλλακτικός τρόπος που μπορούμε να εκφράσουμε την μέθοδο Jacobi, ώστε με κατάλληλες πράξεις να μπορούμε να υπολογίσουμε τις άγνωστες

μεταβλητές της λύσης $x_i,\ i = 1, 2, ..., n$ είναι ο

$$Dx^{(k+1)} = -(L + U)x^{(k)} + b,\ k = 0, 1, ...$$

Έτσι προκύπτει η παρακάτω αλγεβρική σχέση

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}), i = 1, 2, ..., n \tag{3.2.2}$$
$$k = 0, 1, 2, ...\ \ x^{(0)} \text{ το δοθέν αρχικό διάνυσμα}$$

Βάση του πορίσματος (3.1.1), ικανή συνθήκη για την σύγκλιση της μεθόδου Jacobi είναι να ισχύει για κάθε φυσική νόρμα πίνακα η ανισότητα

$$|| - D^{-1}(L + U)|| < 1$$

ενώ βάση του θεωρήματος (3.1.2), αναγκαία και ικανή συνθήκη για την σύγκλιση είναι

$$\rho(-D^{-1}(L + U)) < 1$$

Στην πράξη, δεν είναι υπολογιστικά αποδοτικό να υπολογίζουμε τη φασματική ακτίνα ενός πίνακα, ώστε να ελέγξουμε αν η επαναληπτική μέθοδος συγκλίνει. Αντιθέτως, μπορούμε να υπολογίσουμε αν ο πίνακας συντελεστών A είναι αυστηρά διαγώνια υπέρτερος. Αναφέραμε νωρίτερα ότι, αν ο πίνακας A έχει αυστηρά διαγώνια υπεροχή κατά γραμμή, τότε θα συγκλίνει για κάθε αρχικό διάνυσμα $x^{(0)}$.

3.3 Γενική Επαναληπτική μέθοδος Gauss Seidel

Θεωρούμε το $n \times n$ γραμμικό σύστημα $Ax = b$, όπου ο πίνακας A είναι ομαλός και ικανοποιεί τους περιορισμούς $a_{ii} \neq 0,\ i = 1, 2, ..., n$. Επίσης, έχει αυστηρά διαγώνια υπεροχή κατά γραμμή, οπότε όπως είδαμε στην μέθοδο Jacobi η επαναληπτική μέθοδος συγκλίνει για κάθε αρχικό διάνυσμα $x^{(0)}$. Έτσι ισχύει

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|,\ i = 1, 2, ..., n.$$

Ο πίνακας A μπορεί να γραφεί στην παρόμοια με τη μέθοδο Jacobi μορφή

$$x^{(k+1)} = Bx^{(k)} + c,\ k = 0, 1, ...$$

όπου

$$B = -(L + D)^{-1}U,\ c = (L + D)^{-1}b$$

και $x^{(0)}$ δοθέν αρχικό διάνυσμα της λύσης. Ο παραπάνω επαναληπτικός τύπος μπορεί να γραφεί με μια αντίστοιχη εναλλακτική μορφή, ώστε να μπορούμε να υπολογίσουμε τις άγνωστες μεταβλητές της λύσης. Έτσι με κατάλληλες πράξεις θα έχουμε

$$(L + D)x^{(k+1)} = -Ux^{(k+1)} + b,\ k = 0, 1, ...$$

οπότε προκύπτει η αλγεβρική σχέση

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}),\ i = 1, 2, ..., n \tag{3.3.1}$$

$k = 0, 1, 2, ...$ $x^{(0)}$ δοθέν αρχικό διάνυσμα

Ικανή συνθήκη για την σύγκλιση της μεθόδου Gauss Seidel, είναι να ισχύει για κάθε φυσική νόρμα πίνακα η ανισότητα

$$|| - (L + D)^{-1}U|| < 1$$

ενώ αναγκαία και ικανή συνθήκη για την σύγκλιση είναι

$$\rho(-(L + D)^{-1}U) < 1$$

Σε πολλές περιπτώσεις, η μέθοδος Gauss Seidel είναι πιο αποτελεσματική

από την μέθοδο Jacobi ως προς την σύγκλιση της προσεγγιστικής λύσης. Αυτό συμβαίνει διότι η μέθοδος Gauss Seidel αξιοποιεί όλες τις διαθέσιμες πληροφορίες της τρέχουσας και προηγούμενης επανάληψης, ενώ η μέθοδος Jacobi όχι. Δηλαδή, όπως θα εξετάσουμε και στις παράλληλες μεθόδους, η Gauss Seidel χρησιμοποιεί τις προσεγγιστικές λύσεις της τρέχων επανάληψης, αλλά και της προηγούμενης. Ωστόσο, υπάρχουν γραμμικά συστήματα στα οποία η μέθοδος Jacobi συγκλίνει ενώ η μέθοδος Gauss Seidel αποκλίνει και αντιστρόφως.

3.4 Γενική Παράλληλη Μέθοδος Jacobi

Στην υποενότητα (3.2) μελετήσαμε την επαναληπτική μέθοδο Jacobi, και είδαμε ότι μπορούμε να εκφράσουμε τον επαναληπτικό τύπο της χρησιμοποιώντας είτε τον τύπο (3.2.1), είτε τον αλγεβρικό τύπο (3.2.2). Υπάρχουν διάφορες σειριακές τεχνικές υλοποίησης της μεθόδου Jacobi χρησιμοποιώντας τον τύπο (3.2.2), ωστόσο για μια αυθαίρετη προσεγγιστική λύση $x^{(k)}$, ξεκινώντας και λύνοντας για $x_1^{(k)}$, μετά από $n-1$ επαναλήψεις θα καταλήξουμε στη προσεγγιστική λύση της άγνωστης μεταβλητής $x_n^{(k)}$. Έτσι, για να υπολογίσουμε μια προσεγγιστική λύση της άγνωστης μεταβλητής $x_n^{(k)}$, θα πρέπει πρώτα να υπολογιστούν σειριακά οι λύσεις των $x_i^{(k)},\ i = 1, ..., n-1$. Εξετάζοντας τον τύπο (3.2.2), μπορούμε να δούμε ότι η προσεγγιστική λύση $x^{(k+1)}$, εξαρτάται από την προσεγγιστική λύση της προηγούμενης επανάληψης $x^{(k)}$. Αυτό σημαίνει ότι αν υπολογίσουμε τη λύση για κάποια άγνωστη μεταβλητή x_i του διανύσματος $x^{(k+1)}$, αυτή δεν θα επηρεάσει τις υπόλοιπες λύσεις των άλλων άγνωστων μεταβλητών $x_j,\ j \neq i$. Αυτή η παρατήρηση μας οδηγεί στο συμπέρασμα ότι μπορούμε να υπολογίσουμε τις προσεγγιστικές λύσεις $x^{(k)}$ με παράλληλο τρόπο. Ξαναγράφουμε το $n \times n$ γραμμικό σύστημα $Ax = b$ ως $AX = B$, όπου

$$A = \begin{bmatrix} a_{11} & a_{22} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

όπου ο A είναι αντιστρέψιμος, είναι αυστηρά διαγώνια υπέρτερος, και ικανοποιεί τους περιορισμούς που αναφέρθηκαν στην υποενότητα (3.2). Εφόσον θέλουμε να υπολογίσουμε τις λύσεις των άγνωστων μεταβλητών παράλληλα, θα πρέπει με κάποιο τρόπο να διασπάσουμε το γραμμικό σύστημα $AX = B$ σε διακριτές ομάδες άγνωστων μεταβλητών, και να αναθέσουμε τον υπολογισμό των άγνωστων μεταβλητών της κάθε ομάδας σε ένα ξεχωριστό νήμα. Ο αριθμός των ομάδων που θα δημιουργηθούν θα πρέπει να είναι ίσος με τον αριθμό των νημάτων που θα επιλέξει ο προγραμματιστής. Το πλήθος των άγνωστων μεταβλητών της κάθε ομάδας θα είναι ίσο με το πηλίκο της διάστασης n του πίνακα A με τον αριθμό των νημάτων, με την προϋπόθεση ότι η διάσταση του πίνακα A διαιρείται ακριβώς με τον αριθμό των νημάτων. Σε περιπτώσεις που η διάσταση του πίνακα A δεν διαιρείται ακριβώς με τον αριθμό των νημάτων που είναι διαθέσιμα προς χρήση, τότε προσθέτουμε τις άγνωστες μεταβλητές που έχουν απομείνει στην τελευταία ομάδα άγνωστων μεταβλητών. Μετά την διάσπαση του γραμμικού συστήματος σε διακριτές ομάδες άγνωστων μεταβλητών, το $AX = B$ θα έχει την εξής μορφή

$$A_p X_p = B_p,\ p = 1, ..., \textit{NUMBER OF THREADS}, \tag{3.4.1}$$

όπου, $g = \frac{n}{\text{NUMBER OF THREADS}}$, A_p ένας $g \times n$ πίνακας, X_p και B_p διανύσματα $g \times 1$, και p ο δείκτης της ομάδας αγνώστων μεταβλητών. Πιο αναλυτικά η (3.4.1) μπορεί να γραφεί ως εξής

$p = 1$

$$A_1X_1 = B_1 = \begin{bmatrix} a_{1,1} & a_{2,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{g,1} & a_{g,2} & \cdots & a_{g,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_g \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_g \end{bmatrix} \tag{3.4.2}$$

$p = 2$

$$A_2X_2 = B_2 = \begin{bmatrix} a_{g+1,1} & a_{g+1,2} & \cdots & a_{g+1,n} \\ a_{g+2,1} & a_{g+2,2} & \cdots & a_{g+2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{g*2,1} & a_{g*2,2} & \cdots & a_{g*2,n} \end{bmatrix} \begin{bmatrix} x_{g+1} \\ x_{g+2} \\ \vdots \\ x_{g*2} \end{bmatrix} = \begin{bmatrix} b_{g+1} \\ b_{g+2} \\ \vdots \\ b_{g*2} \end{bmatrix}$$

$p = 3$

$$A_3X_3 = B_3 = \begin{bmatrix} a_{(g*2)+1,1} & a_{(g*2)+1,2} & \cdots & a_{(g*2)+1,n} \\ a_{(g*2)+2,1} & a_{(g*2)+2,2} & \cdots & a_{(g*2)+2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(g*2)+g,1} & a_{(g*2)+g,2} & \cdots & a_{(g*2)+g,n} \end{bmatrix} \begin{bmatrix} x_{(g*2)+1} \\ x_{(g*2)+2} \\ \vdots \\ x_{(g*2)+g} \end{bmatrix} = \begin{bmatrix} b_{(g*2)+1} \\ b_{(g*2)+2} \\ \vdots \\ b_{(g*2)+g} \end{bmatrix}$$

Και γενικά για p

$$A_pX_p = B_p = \begin{bmatrix} a_{g*(p-1)+1,1} & \cdots & a_{g*(p-1)+1,n} \\ a_{g*(p-1)+2,1} & \cdots & a_{g*(p-1)+2,n} \\ \vdots & \ddots & \vdots \\ a_{g*(p-1)+g,1} & \cdots & a_{g*(p-1)+g,n} \end{bmatrix} \cdot \begin{bmatrix} x_{g*(p-1)+1} \\ x_{g*(p-1)+2} \\ \vdots \\ x_{g*(p-1)+g} \end{bmatrix} = \begin{bmatrix} b_{g*(p-1)+1} \\ b_{g*(p-1)+2} \\ \vdots \\ b_{g*(p-1)+g} \end{bmatrix}$$

και

$$a_{ij} = \begin{cases} a_{ij} & i = 1, ..., g,\ j = 1, ..., n \text{ για } p = 1 \\ a_{g*(p-1)+i,j} & i = 1, ..., g,\ j = 1, ..., n \text{ για } p > 1 \end{cases}$$

Βάση του γραμμικού συστήματος (3.4.1), με τον επαναληπτικό αλγεβρικό τύπο Jacobi (3.2.2), μπορούμε να υπολογίσουμε για μια επανάληψη $k,\ k = 1, ...,$ την προσεγγιστική λύση $X^{(k+1)}$ για μια ομάδα p άγνωστων μεταβλητών. Οπότε, ο τύπος (3.2.2) θα παραμείνει όπως είναι, αλλά θα εφαρμόζεται για διαφορετικές ομάδες p του γραμμικού συστήματος. Αρχικά, ο σειριακός τύπος ξεκινούσε από τον υπολογισμό της πρώτης μεταβλητής $x_1^{(k+1)}$ της επανάληψης k, και τελείωνε με τον υπολογισμό της μεταβλητής $x_n^{(k+1)}$ της ίδιας επανάληψης. Τώρα, βάση των παραμέτρων p και g, ο τύπος (3.2.2) θα είναι

$$X_{p,i}^{(k+1)} = \frac{1}{A_{p,ii}}(B_{p,i} - \sum_{\substack{j=1 \\ j \neq i}}^{n} A_{p,ij} X_{p,j}^{(k)}),\ i = g * (p-1) + 1, ..., g * (p-1) + g$$

$k = 0, 1, 2, ...$ δοσμένου του αρχικού διανύσματος $X^{(0)}$

(3.4.3)

όπου $X_{p,i}$ η $X_{p,j}$, η άγνωστη μεταβλητή i, j του διανύσματος Q της προσεγγιστικής λύσης της ομάδας p, $A_{p,ii}$ η $A_{p,ij}$, το αντίστοιχο στοιχείο του πίνακα A της ομάδας p.

Πλέον ο (3.4.3), μπορεί να χρησιμοποιηθεί για την παραλληλοποίηση του επαναληπτικού αλγορίθμου της μεθόδου Jacobi. Όταν ολοκληρωθεί ο παράλληλος υπολογισμός $X_p^{(k+1)}$ της κάθε ομάδας, συνενώνουμε το διάνυσμα της προσεγγιστικής λύσης $X^{(k+1)}$. Έπειτα μπορεί να υπολογιστεί το σφάλμα της προσεγγιστικής λύσης που εκφράζεται ως

$$||X^{(k+1)} - X^{(k)}|| < TOL$$

όπου *TOL* είναι μια τιμή ανοχής ορισμένη από τον προγραμματιστή. Αν το σφάλμα ξεπεράσει αυτή την τιμή, τότε η μέθοδος τερματίζεται.

3.5 Υλοποίηση Γενικής Παράλληλης Μεθόδου Jacobi

Στην υποενότητα (3.4) εξετάσαμε θεωρητικά και περιγράψαμε τον παράλληλο αλγόριθμο Jacobi. Παρακάτω θα μελετήσουμε πως υλοποιείται αυτή η παράλληλη μέθοδος προγραμματιστικά. Πιο συγκεκριμένα, θα δούμε πως διασπάται το γραμμικό σύστημα $AX = B$ διαμοιράζοντας τις άγνωστες μεταβλητές της λύσης σε ομάδες, και πως εκτελείται ο επαναληπτικός τύπος (3.2.2) παράλληλα με το πρότυπο OpenMP. Αρχικά, ας δούμε πως διαμοιράζουμε τις άγνωστες μεταβλητές της προσεγγιστικής λύσης $X^{(k+1)}$ σε ομάδες.

ΑΛΓΟΡΙΘΜΟΣ 1 Διαμοιρασμός Άγνωστων Μεταβλητών σε Ομάδες

Input N, NUMBER_OF_THREADS
Output groups

procedure CREATE_GROUPS()

```
1:  group_size = floor(N/NUMBER_OF_THREADS)
2:  groups = [ ][ ]
3:  for i = 1 to NUMBER_OF_THREADS do
4:    group = [ ]
5:    for j = group_size*(i-1) to group_size*i-1 do
6:      insert_variable(j, group)
7:    end for
8:    insert_group(group, groups)
9:  end for
10: for j = NUMBER_OF_THREADS*group_size to N do
11:   insert_variable_to_last_group(j, groups)
12: end for
13:
14: return  groups
```

Ο παραπάνω αλγόριθμος παίρνει σαν ορίσματα τη διάσταση του πίνακα και τον αριθμό των νημάτων που έχει ορίσει ο προγραμματιστής.

Στην γραμμή 1, διαιρείται η διάσταση n του πίνακα με τον αριθμό των νημάτων $NUMBER\_OF\_THREAD$ και με τη συνάρτηση floor() επιστρέφεται η μεγαλύτερη δυνατή ακέραια τιμή που είναι μικρότερη ή ίση με το δεδομένο όρισμα. Η συνάρτηση floor() είναι πολύ σημαντική, γιατί όπως θα δούμε μέσω ενός παραδείγματος, εξασφαλίζει την ακεραιότητα του διαμοιρασμού των άγνωστων μεταβλητών σε ομάδες. Έστω ότι η διάσταση του πίνακα είναι $n = 14$ και ο αριθμός των νημάτων είναι $NUMBER\_OF\_THREAD = 3$. Άρα και το πλήθος των ομάδων θα είναι 3. Το αποτέλεσμα που θα μας δώσει η διαίρεση θα έχει το ρόλο της μεταβλητής g, η οποία αναπαριστά το πλήθος των άγνωστων μεταβλητών κάθε ομάδας, το οποίο δεν μπορεί να είναι αριθμός κινητής υποδιαστολής. Εκτός αυτού, δεν μπορεί να είναι ούτε ο μικρότερος ακέραιος που είναι μεγαλύτερος η ίσος του αποτελέσματος της διαίρεσης. Αν παίρναμε τον μικρότερο ακέραιο που είναι μεγαλύτερος η ίσος του $\frac{14}{3} = 4.6$, δηλαδή το 5, εφόσον έχουμε 3 ομάδες, συνολικά θα είχαμε 15 άγνωστες μεταβλητές, ενώ η διάσταση του πίνακα είναι 14. Αντιθέτως, αν παίρναμε τον μεγαλύτερο δυνατό ακέραιο αριθμό που είναι μικρότερος ή ίσος του 4.6, δηλαδή το 4, συνολικά θα είχαμε 12 άγνωστες μεταβλητές, και τις υπόλοιπες 2 θα τις προσθέταμε στη τελευταία ομάδα. Γενικά, με τη συνάρτηση floor() εξαλείφονται τέτοια προβλήματα και διασφαλίζουμε την ακεραιότητα.

Στην γραμμή 2 δηλώνουμε τη μεταβλητή groups, η οποία θα περιέχει τις ομάδες των άγνωστων μεταβλητών.

Στις γραμμές 3 μέχρι 9, εισάγουμε τις άγνωστες μεταβλητές αρχικά σε μια ομάδα, και έπειτα εισάγουμε την ομάδα στο γενικό σύνολο των ομάδων *groups*. Δίνουμε ένα παράδειγμα για να περιγράψουμε αυτό το σημείο καλύτερα. Έστω ότι η διάσταση του πίνακα είναι $n = 15$ και ο αριθμός των νημάτων που θέλουμε να χρησιμοποιήσουμε είναι $NUMBER\_OF\_THREAD = 3$. Οι ομάδες που θα δημιουργηθούν θα είναι 3 και το πλήθος των μεταβλητών κάθε ομάδας θα είναι $g = \frac{n}{NUMBER\_OF\_THREAD} = 5$ η αλλιώς, $group\_size = 5$. Αρχικά, για $i = 1$, δημιουργούμε την ομάδα *group* η οποία είναι ένας κενός πίνακας, και

στη συνέχεια ξεκινώντας από $j = group\_size * (i-1) = 5 * (1-1) = 0$, και φτάνοντας μέχρι το $group\_size * i - 1 = 5 * 1 - 1 = 4$, εισάγουμε στην ομάδα *group* το $j,\ j = 0, 1, ..., 4$, το οποίο αναπαριστά την άγνωστη μεταβλητή. Μόλις ολοκληρωθεί η εισαγωγή για κάθε ομάδα *group*, εισάγουμε την *group* στο σύνολο των ομάδων *groups*. Για $i = 2$, δημιουργούμε ξανά μια κενή ομάδα *group* και αυτή τη φορά ξεκινώντας από $j = group\_size*(i-1) = 5*(2-1) = 5$, και φτάνοντας μέχρι το $group\_size * i - 1 = 5 * 2 - 1 = 9$, εισάγουμε στην ομάδα *group* το $j,\ j = 5, 6, ..., 9$. Τέλος, για $i = 3$, ξεκινώντας από $j = group\_size * (i-1) = 5 * (3-1) = 10$, και φτάνοντας μέχρι το $group\_size * i - 1 = 5 * 3 - 1 = 14$, εισάγουμε στην τελευταία ομάδα το $j,\ j = 10, 11, ..., 14$.

Στις γραμμές 10 μέχρι 12, εισάγουμε στην τελευταία ομάδα τις μεταβλητές που έχουν απομείνει στη περίπτωση που η διαίρεση $\frac{n}{NUMBER\_OF\_THREADS}$ αφήσει υπόλοιπο. Για παράδειγμα, αν $n = 14$, $NUMBER\_OF\_THREADS = 3$ και $g = \frac{n}{NUMBER\_OF\_THREADS} = \frac{14}{3} = 4.6$ η $group\_size = 4.6$. Μέσω της συνάρτησης floor(), η *group_size* θα πάρει τη τιμή 4, οπότε θα είναι $group\_size \times NUMBER\_OF\_THREADS = 12$ που αφήνει υπόλοιπο 2. Έτσι, εκτελώντας τον παραπάνω αλγόριθμο μέχρι και τη γραμμή 10, θα προκύψουν 3 ομάδες με συνολικά 12 μεταβλητές. Σε αυτή την περίπτωση, αν εκτελέσουμε τις γραμμές 10 μέχρι 12, ξεκινώντας από $j = NUMBER\_OF\_THREADS * group\_size = 3 * 4 = 12$ και φτάνοντας στο $n = 14$, θα εισαχθούν οι τελευταίες 2 μεταβλητές.

Στη συνέχεια δίνουμε τις κύριες γραμμές του κώδικα παραλληλοποίησης της μεθόδου Jacobi και θα εξετάσουμε την μεθοδολογία του αλγορίθμου.

ΑΛΓΟΡΙΘΜΟΣ 2 Παράλληλη Μέθοδος Jacobi

Input X, B, A, *TOL*, max_iterations, N, NUMBER_OF_THREADS, groups
Output X

```
procedure PARALLEL_JACOBI()
 1: for i = 1 to max_iterations do
 2:   prev_X = X
 3:   local_error = [N]
 4:   START_PARALLEL_REGION
 5:   for g = 0 to groups.size()-1 do
 6:     for unknown_variable in groups[g] do
 7:       sum = 0.0
 8:       for j=0 to N do
 9:         if j != unknown_variable then
10:           sum += A[unknown_variable][j]*prev_X[j]
11:         end if
12:       end for
13:       X[unknown_variable]=(1/A[unknown_variable][unknown_variable])*
          (B[unknown_variable]-sum)
14:       local_error[unknown_variable]=abs(X[unknown_variable]-prev_X[unknown_variable])
15:     end for
16:   end for
17:   END_PARALLEL_REGION
18:   max_error=get_max_error(local_error)
19:   if max_error < TOL then
20:     finish()
21:   end if
22: end for
23:
24: return  X
```

Στην γραμμή 1, έχουμε έναν εξωτερικό βρόχο For, ο οποίος θα σταματήσει τις επαναλήψεις μόλις φτάσουμε τον μέγιστο αριθμό επαναλήψεων, η το σφάλμα προσέγγισης γίνει μικρότερο από την τιμή ανοχής (TOL). Ο υπολογισμός του σφάλματος γίνεται στην γραμμή 18, και έπειτα συγκρίνεται με την τιμή ανοχής (TOL) στις γραμμές 19 μέχρι 21.

Στην γραμμή 2, αντιγράφουμε το διάνυσμα X άγνωστων μεταβλητών στη μεταβλητή $prev\_X$ γιατί όπως θα δούμε σε επόμενα βήματα του αλγορίθμου, το $prev\_X$ θα έχει το ρόλο του $x^{(k)}$ και το X θα είναι το $x^{(k+1)}$.

Από τη γραμμή 4 μέχρι και την 17, θα εκτελεστεί το παράλληλο τμήμα του αλγορίθμου χωρίζοντας τις επαναλήψεις του δεύτερου εμφωλευμένου βρόχου For στα νήματα που έχουμε ορίσει στη παράμετρο $NUMBER\_OF\_THREADS$. Ο αριθμός των επαναλήψεων του δεύτερου εμφωλευμένου For, θα είναι ίσος με τον αριθμό των ομάδων των άγνωστων μεταβλητών και άρα θα είναι ίσος με τον αριθμό των νημάτων. Από την γραμμή 6 μέχρι και την 13 εκτελείται ο επαναληπτικός παράλληλος αλγεβρικός τύπος Jacobi (3.4.3) για κάθε άγνωστη μεταβλητή στην ομάδα. Για κάθε άγνωστη μεταβλητή, ενημερώνουμε την τιμή του διανύσματος X το οποίο αναπαριστά το $X^{(k+1)}$ στη γραμμή 13, και χρησιμοποιούμε το διάνυσμα $prev\_X$ ως το $X^{(k)}$ για να υπολογίσουμε το άθροισμα του επαναληπτικού τύπου (3.4.3) στις γραμμές 8 μέχρι 12. Τέλος, υπολογίζουμε το απόλυτο σφάλμα μεταξύ του διανύσματος X και του $prev\_X$ για κάθε άγνωστη μεταβλητή στη γραμμή 14, και βρίσκουμε το μέγιστο σφάλμα στη γραμμή 18. Από την γραμμή 17 και παραπέρα, ο αλγόριθμος εκτελείται σειριακά μέχρι να ξανά συναντήσει το παράλληλο τμήμα στη γραμμή 4.

3.6 Γενική Παράλληλη Μέθοδος Jacobi-Gauss Seidel

Στις δύο παραπάνω ενότητες είδαμε ότι ο μαθηματικός αλγόριθμος της γενικής επαναληπτικής μεθόδου Jacobi έχει υψηλό βαθμό παραλληλισμού, διότι οι προσεγγιστικές λύσεις της εξαρτώνται μόνο από τις προσεγγιστικές λύσεις της προηγούμενης επανάληψης. Μελετώντας την επαναληπτική μέθοδο Gauss Seidel (3.3.1) που εξετάσαμε στην ενότητα (3.3) βλέπουμε ότι, κατά την επανάληψη $k+1,\ k = 1, 2, ...,$ η τιμή της προσεγγιστικής συνιστώσας $x_i^{(k+1)},\ i = 1, 2, ..., n,$ εξαρτάται όχι μόνο από την προηγούμενη λύση $x^{(k)}$, αλλά και από τις συνιστώσες της τρέχων λύσης $x^{(k+1)}$. Πιο αναλυτικά, η επαναληπτική μέθοδο Gauss Seidel (3.3) μπορεί να διασπαστεί σε δύο μέρη. Το πρώτο μέρος είναι

$$\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)},\ i = 1, 2, ..., n,$$

το οποίο εξαρτάται από την τιμή της τρέχων λύσης $x_j^{(k+1)},\ j = 1, ..., i-1$. Στην (3.3), η $x_j^{(k+1)},\ j = 1, ..., i-1$ είναι όλες οι συνιστώσες από το πρώτο στοιχείο της γραμμής $i,\ i = 1, ..., n$, μέχρι και το στοιχείο $j = i-1$. Το δεύτερο μέρος της επαναληπτικής μεθόδου Gauss Seidel (3.3) είναι

$$\sum_{j=i+1}^{n} a_{ij} x_j^{(k)}$$

το οποίο εξαρτάται μόνο από τις συνιστώσες της προηγούμενης λύσης $x_j^{(k)}$.
Εξαιτίας του πρώτου μέρους της επαναληπτικής μεθόδου Gauss Seidel, της έλλειψης συγχρονισμού των νημάτων, και της μορφής του πίνακα συντελεστών A, που στις γενικές περιπτώσεις είναι πυκνός, είναι δύσκολο να επιτευχθεί η παραλληλοποίηση της μεθόδου χρησιμοποιώντας την αρχιτεκτονική διαμοιραζόμενης μνήμης. Στην βιβλιογραφία, έχουν προταθεί αρκετές παράλληλες υλοποιήσεις της μεθόδου Gauss Seidel, η οποίες χρησιμοποιούν την αρχιτεκτονική κατανεμημένης μνήμης, και είναι αρκετά αποτελεσματικές για πυκνούς πίνακες, αλλά όχι και τόσο για αραιούς [10]. Παρακάτω θα δούμε μέσω ενός παραδείγματος, πως μπορούμε να συνδυάσουμε τα χαρακτηριστικά της μεθόδου Jacobi και αυτά της Gauss Seidel για να επιτύχουμε τον παραλληλισμό και γρηγορότερη σύγκλιση για γενικούς πίνακες συντελεστών A.
Υποθέτουμε πως θέλουμε να χρησιμοποιήσουμε 5 νήματα για να υπολογίσουμε την προσεγγιστική λύση για ένα $n \times n$ γραμμικό σύστημα $AX = B$ με διάσταση πίνακα $n = 100$. Αν διασπάσουμε το γραμμικό σύστημα σε μικρότερα υπο-συστήματα, θα έχει την μορφή

$$A_p X_p = B_p$$

όπου p ο δείκτης της ομάδας. Να υπενθυμίσουμε εδώ ότι ο συνολικός αριθμός των ομάδων άγνωστων μεταβλητών θα είναι ίσος με τον αριθμό των νημάτων που έχουμε ορίσει, δηλαδή στην περίπτωση μας 5. Η κάθε ομάδα θα έχει g άγνωστες μεταβλητές, όπου $g = \frac{N}{\text{NUMBER\_OF\_THREADS}}$. Λαμβάνοντας υπόψη την έλλειψη συγχρονισμού των νημάτων, η σειρά υπολογισμού των ομάδων άγνωστων μεταβλητών μπορεί να γίνει με διαφορετική σειρά από αυτή που περιμένουμε, και έτσι η $A_pX_p = B_p$ μπορεί έχει την μορφή

$$A_3X_3 = B_3$$
$$A_1X_1 = B_1$$
$$A_5X_5 = B_5$$
$$A_2X_2 = B_2$$
$$A_4X_4 = B_4$$

οπότε πρώτα θα υπολογιστεί η 3η ομάδα, μετά η 1η ομάδα κ.ο.κ. Η σειρά εκτέλεσης μπορεί να είναι διαφορετική κάθε φορά που θέλουμε να υπολογίσουμε την προσεγγιστική λύση.

Γενικά, για την προσέγγιση των μεταβλητών ενός υποσυστήματος, χρειάζονται οι προσεγγιστικές λύσεις όλων των προηγούμενων μεταβλητών της τρέχων επανάληψης, οι οποίες μπορεί να ανήκουν στο σύνολο άλλων υποσυστημάτων. Ας δούμε δύο παραδείγματα. Αν υποθέσουμε ότι η 3η ομάδα, με $p = 3$, υπολογίζεται πρώτη λόγω της έλλειψης συγχρονισμού των νημάτων, τότε η πρώτη άγνωστη μεταβλητή της, η οποία μπορεί να είναι η X_i, για αυθαίρετο i, θα χρειαστεί τις τρέχων προσεγγιστικές λύσεις $X_j,\ j = 1, ..., i-1$ της ομάδας 1 και 2, οι οποίες όμως δεν έχουν ακόμη υπολογιστεί, και ως αποτέλεσμα, θα πρέπει να χρησιμοποιήσουμε μόνο τις προσεγγιστικές λύσεις που έχουμε από την προηγούμενη επανάληψη. Σε αυτή την περίπτωση, η μέθοδος Gauss Seidel

θα μετατραπεί στην Jacobi. Έστω τώρα ότι θέλουμε να υπολογίσουμε την πρώτη άγνωστη μεταβλητή της 2ης ομάδας. Πριν ξεκινήσει ο υπολογισμός της, κάποιες από τις αρχικές μεταβλητές της 1ης ομάδας έχουν ήδη υπολογιστεί για την τρέχων επανάληψη. Όταν ξεκινήσει ο υπολογισμός της πρώτης μεταβλητής της 2ης ομάδας, θα χρησιμοποιηθούν οι προσεγγιστικές λύσεις της 1ης ομάδας της τρέχων επανάληψης που έχουν ήδη υπολογιστεί, αλλά και αυτές που δεν έχουν υπολογιστεί. Οπότε, το πρώτο μέρος της μεθόδου Gauss Seidel (3.3.1) θα αποτελείται από λύσεις της τρέχων επανάληψης $X^{(k+1)}$ και της προηγούμενης $X^{(k)}$. Σε αυτή την περίπτωση, χρησιμοποιούνται και δύο μέθοδοι για την εύρεση της προσεγγιστικής λύσης μιας μεταβλητής.
Όπως βλέπουμε, η παράλληλη αυτή μέθοδος είναι κυρίως ίδια με τη παράλληλη Jacobi, με τη διαφορά ότι αξιοποιούμε περισσότερες πληροφορίες για την προσεγγιστική λύση όταν μας δίνεται η δυνατότητα.

3.7 Υλοποίηση Γενικής Παράλληλης Μεθόδου Jacobi-Gauss Seidel

Στην ενότητα (3.6) εξετάσαμε θεωρητικά την γενική παράλληλη μέθοδο Jacobi-Gauss Seidel. Παρακάτω θα μελετήσουμε πως υλοποιείται αυτή η παράλληλη μέθοδος προγραμματιστικά. Η υλοποίηση της διάσπασης του γραμμικού συστήματος και της ανάθεσης των άγνωστων μεταβλητών σε ομάδες έχουν περιγραφεί στην ενότητα (3.5). Εδώ θα περιγράψουμε τον παράλληλο αλγόριθμο Jacobi-Gauss Seidel.

ΑΛΓΟΡΙΘΜΟΣ 3 Παράλληλη Μέθοδος Jacobi-Gauss Seidel

Input X, B, A, *TOL*, max_iterations, N, NUMBER_OF_THREADS, groups
Output X

procedure PARALLEL_JACOBI_GAUSS_SEIDEL()

```
1:  for i = 1 to max_iterations do
2:    prev_X = X
3:    local_error = [N]
4:    START_PARALLEL_REGION
5:    for g = 0 to groups.size()-1 do
6:      for unknown_variable in groups[g] do
7:        sum = 0.0
8:        for j=0 to N do
9:          if j != unknown_variable then
10:           sum += A[unknown_variable][j]*X[j]
11:         end if
12:       end for
13:       X[unknown_variable]=(1/A[unknown_variable][unknown_variable])*
          (B[unknown_variable]-sum)
14:       local_error[unknown_variable]=abs(X[unknown_variable]-
          prev_X[unknown_variable])
15:     end for
16:   end for
17:   END_PARALLEL_REGION
18:   max_error=get_max_error(local_error)
19:   if max_error < TOL then
20:     finish()
21:   end if
22: end for
23:
24: return  X
```

Ο αλγόριθμος είναι ο ίδιος με αυτόν που περιγράψαμε στην υποενότητα (3.5), αλλά εδώ χρησιμοποιούμε το διάνυσμα X στη γραμμή 10 και όχι το $prev\_X$. Με το διάνυσμα $prev\_X$ χρησιμοποιούσαμε τις προσεγγιστικές λύσεις μόνο της προηγούμενης επανάληψης $X^{(k)}$, ενώ εδώ χρησιμοποιούμε τις λύσεις της τρέχων και της προηγούμενης επανάληψης ανάλογα με την περίπτωση. Ακολουθώντας αυτή την μέθοδο, προκύπτει το ερώτημα της εξάρτησης των δεδομένων. Θα μπορούσαμε να υποθέσουμε, εξαιτίας της ανάγνωσης δεδομένων από διαμοιραζόμενες θέσεις μνήμης, οι οποίες χρησιμοποιούνται από άλλα

νήματα, ότι η παράλληλη μέθοδος θα μας έδινε λάθος αποτελέσματα. μια τέτοια υπόθεση είναι λανθασμένη διότι στη γραμμή 10, κάνουμε εγγραφή σε θέσεις μνήμης οι οποίες είναι διαμοιραζόμενες στα νήματα, αλλά τροποποιούνται από ένα νήμα μόνο, και η ανάγνωση στις θέσεις $X[j]$ δεν επηρεάζει την ακεραιότητα της λύσης. Έτσι, έχουμε επιτύχει την σχεδίαση μιας παράλληλης μεθόδου η οποία συνδυάζει τα χαρακτηριστικά της Jacobi και της Gauss Seidel, και δεν αυξάνει την πολυπλοκότητα του αλγορίθμου.

Κεφάλαιο 4

Αριθμητική Παραγώγιση

Η Αριθμητική Παραγώγιση είναι μια διαδικασία στην οποία προσπαθούμε να προσεγγίσουμε την τιμή της παραγώγου μιας συνάρτησης σε ένα σύνολο σημείων. Υπάρχουν περιπτώσεις όπου χρειαζόμαστε τις τιμές των παραγώγων πολύπλοκων συναρτήσεων, όπου είναι επίπονο να τις υπολογίσουμε αναλυτικά. Τότε καταφεύγουμε στην προσέγγιση τους μέσω τεχνικών αριθμητικής παραγώγισης. μια τεχνική είναι να υπολογίσουμε τις παραγώγους συναρτήσεων μέσω των πολυωνύμων προσέγγισης των συναρτήσεων.

4.1 Αριθμητική Παραγώγιση με Σειρές Taylor

Γνωρίζουμε από τον Απειροστικό Λογισμό, ότι η παράγωγος μιας συνάρτησης στο σημείο x_0 ορίζεται ως

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \tag{4.1.1}$$

Παρακάτω, θα εξετάσουμε εφαρμογές του θεωρήματος Taylor με ισαπέχοντα σημεία, οι οποίες μας βοηθούν να προσεγγίσουμε το λόγο $\frac{f(x+h)-f(x)}{h}$ της εξίσωσης (4.1.1).

Το θεώρημα του Taylor μας λέει ότι, αν γνωρίζουμε την τιμή των παραγώγων

της συνάρτησης σε ένα οποιοδήποτε σημείο του πεδίου ορισμού της, τότε μπορούμε να υπολογίσουμε προσεγγιστικά την τιμή των παραγώγων της συνάρτησης σε ένα γειτονικό σημείο.

Θεώρημα Taylor Αν η συνάρτηση $f(x)$ είναι συνεχής και n φορές παραγωγίσημη στο διάστημα $[x_a, x_b]$ και η παράγωγος της $f^{(n+1)}$ υπάρχει σε αυτό το διάστημα, τότε υπάρχει $\xi \in (x_a, x_b)$ τέτοιο ώστε

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + ... + \frac{(x-a)^{n+1}}{(n+1)!}f^{n+1}(\xi), \quad (4.1.2)$$

όπου $a \in (x_a, x_b)$. Ο τελευταίος όρος της παραπάνω εξίσωσης

$$Res_{n+1} = \frac{(x-a)^{n+1}}{(n+1)!}f^{n+1}(\xi) \quad (4.1.3)$$

ονομάζεται υπόλοιπο κατά Lagrange, το οποίο θα μας δώσει τις εκφράσεις των σφαλμάτων για τους διάφορους τύπους της αριθμητικής παραγώγισης. Αν γνωρίζουμε την τιμή της συνάρτησης f σε ένα σημείο x και στο γειτονικό του σημείο $x + h$ στο διάστημα $[x_a, x_b]$, τότε αντικαθιστώντας στον τύπο (4.1.2) το $a = x,\ x = x + h,\ (x - a) = (x + h - x) = h$ μπορούμε να υπολογίσουμε προσεγγιστικά την παράγωγο της συνάρτησης για το σημείο x και να πάρουμε αντίστοιχα την έκφραση του σφάλματος. Έτσι από τον τύπο (4.1.2) έχουμε

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(\xi),\ \xi \in [x, x+h]. \quad (4.1.4)$$

Λύνοντας ως προς την παράγωγο $f'(x)$, παίρνουμε

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi), \quad (4.1.5)$$

όπου ο προσεγγιστικός τύπος είναι ο

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

και η εκτίμηση σφάλματος

$$e(x) = f'(x) - \frac{f(x+h) - f(x)}{h} = \frac{h}{2} f''(\xi),$$

που προκύπτει από την διαφορά της προσεγγιστικής και της πραγματικής τιμής η οποία δίνεται από την τιμή της $f'(x)$ και είναι τάξης ένα. Ο τύπος (4.1.5) ονομάζεται τύπος αριθμητικής παραγώγισης με προς τα εμπρός διαφορές διότι για τον υπολογισμό της παραγώγου στο σημείο x χρησιμοποιούμε την τιμή της συνάρτησης στο σημείο $(x+h)$. Με αντίστοιχο τρόπο μπορούμε να πάρουμε και τον τύπο αριθμητικής παραγώγισης με προς τα πίσω διαφορές αντικαθιστώντας στον τύπο (4.1.2) το $a = x,\ x = x - h,\ (x - a) = (x - h - x) = -h$. Έτσι έχουμε

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!} f''(\xi),\ \xi \in [x-h, x]. \tag{4.1.6}$$

Λύνοντας ως προς την παράγωγο $f'(x)$, παίρνουμε

$$f'(x) = \frac{f(x) - f(x-h)}{h} - \frac{h}{2} f''(\xi), \tag{4.1.7}$$

όπου ο προσεγγιστικός τύπος είναι ο

$$f'(x) \approx \frac{f(x) - f(x-h)}{h},$$

και η εκτίμηση σφάλματος

$$e(x) = f'(x) - \frac{f(x) - f(x-h)}{h} = \frac{h}{2} f''(\xi).$$

Αν αφαιρέσουμε κατά μέλη τις σχέσεις (4.1.4) και (4.1.6) προκύπτει ένας πιο ακριβής προσεγγιστικός τύπος με τον οποίο υπολογίζουμε την τιμή της παραγώγου σε ένα σημείο x έχοντας τις τιμές της συνάρτησης στα σημεία $(x-h)$, και

$(x+h)$.

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3!}[f'''(\xi_1) + f'''(\xi_{-1})].$$

Λύνοντας ως προς την ζητούμενη παράγωγο $f'(x)$ παίρνουμε

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{12}[f'''(\xi_1) + f'''(\xi_{-1})],$$

όπου ο προσεγγιστικός τύπος είναι ο

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h},$$

και η εκτίμηση σφάλματος είναι

$$e(x) = f'(x) - \frac{f(x+h) - f(x-h}{2h} = \frac{h^2}{12}[f'''(\xi_1) + f'''(\xi_{-1})].$$

η οποία είναι τάξης δύο. Το άνω φράγμα του σφάλματος είναι

$$|e(x)| \leq \frac{h^2}{6} B_3$$

όπου $B_3 = \max_{x-h \leq \xi \leq x+h} |f'''(\xi)|$. Ο παραπάνω τύπος ονομάζεται τύπος αριθμητικής παραγώγισης με κεντρικές διαφορές.

Αντίστοιχη διαδικασία μπορούμε να ακολουθήσουμε για τον υπολογισμό της δευτέρας παραγώγου με κεντρικές διαφορές. Έτσι, προσθέτοντας κατά μέλη τους τύπους

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(\xi_1),$$
$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(\xi_{-1}),$$

και λύνοντας ως προς την παράγωγο $f''(x)$ παίρνουμε τον προσεγγιστικό τύπο

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2},$$

με εκτίμηση σφάλματος

$$e(x) = f''(x) - \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = \frac{h^2}{24}[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})]$$

με άνω φράγμα

$$|e(x)| \leq \frac{h^2}{12} B_4.$$

όπου $B_4 = \max_{x-h \leq \xi \leq x+h} |f^{(4)}(\xi)|$.

4.2 Διαδικασία του Richardson

Συχνά στους υπολογισμούς με τον υπολογιστή, ζητείται να αυξήσουμε την ακρίβεια της μεθόδου αριθμητικής παραγώγισης ώστε να έχουμε καλύτερες προσεγγίσεις της τιμής της παραγώγου της πραγματικής συνάρτησης. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας μεγαλύτερη τάξη σφάλματος αποκοπής του τύπου του Taylor. Είδαμε προηγουμένως πως ο τύπος των κεντρικών διαφορών για την δεύτερη παράγωγο $f''(x)$ της συνάρτησης $f(x)$ μας δίνει ένα σφάλμα τάξης δύο, το οποίο θα αυξήσουμε σε αυτή την ενότητα σε σφάλμα αποκοπής τάξης 4 χρησιμοποιώντας την διαδικασία του Richardson.
Χρησιμοποιώντας τις σχέσεις (4.1.4) και (4.1.6), θα σχηματίσουμε τα εξής αναπτύγματα Taylor

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) + \frac{h^5}{5!}f^{(5)}(x) + \frac{h^6}{6!}f^{(6)}(\xi_1), \quad \xi_1 \in [x, x+h],$$

και

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{(4)}(x) - \frac{h^5}{5!}f^{(5)}(x) + \frac{h^6}{6!}f^{(6)}(\xi_{-1}),$$
$$\xi_{-1} \in [x-h, x].$$

Προσθέτοντας τις παραπάνω σχέσεις έχουμε

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + \frac{h^2}{12}f^{(4)}(x) + \frac{h^4}{720}[f^{(6)}(\xi_1) + f^{(6)}(\xi_{-1})]. \quad (4.2.1)$$

Αν αντικαταστήσουμε στην σχέση (4.2.1) το h με το vh, και αν υποθέσουμε ότι έχουμε τις τιμές της συνάρτησης $f(x)$ στα σημεία $f(x+vh)$ και $f(x-vh)$, τότε παίρνουμε

$$\frac{f(x+vh) - 2f(x) + f(x-vh)}{v^2h^2} = f''(x) + \frac{v^2h^2}{12}f^{(4)}(x) + \frac{v^4h^4}{720}[f^{(6)}(\xi_1^*) + f^{(6)}(\xi_{-1}^*)],$$
$$\xi_1^*, \xi_{-1}^* \in [x-vh, x+vh]. \quad (4.2.2)$$

Τέλος, πολλαπλασιάζοντας τα δύο μέλη της σχέσης (4.2.1) με v^2 και αφαιρώντας την σχέση (4.2.2) από το αποτέλεσμα, τότε αντικαθιστώντας το v με 2 παίρνουμε τον τύπο κεντρικών διαφορών που περιγράψαμε στο κεφάλαιο της αριθμητικής παραγώγισης με τάξη σφάλματος 4.

$$f''(x) = \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x+2h)}{12h^2} +$$
$$\frac{h^4}{540}[f^{(6)}(\xi_1) + f^{(6)}(\xi_{-1}) - 4f^{(6)}(\xi_1^*) - 4f^{(6)}(\xi_{-1}^*)] \quad (4.2.3)$$

Με αυτή την διαδικασία μπορούμε να αυξήσουμε το σφάλμα αποκοπής για διάφορους τύπους αριθμητικής παραγώγισης που κατασκευάζουμε. Ο γενικός τύπος της διαδικασίας Richardson δεν περιγράφεται με λεπτομέρεια στο παρόν

κεφάλαιο, δες [11], αλλά παρατίθεται ένας πίνακας που συνοψίζονται οι συντελεστές των τύπων αριθμητικής παραγώγισης από πρώτη έως τέταρτη παράγωγο για τάξη σφάλματος 2 και 4.

Τάξη Σφάλμ.	$f(x-3h)$	$f(x-2h)$	$f(x-h)$	$f(x)$	$f(x+h)$	$f(x+2h)$	$f(x+3h)$	$=$
h^2	0	0	-1	0	1	0	0	$2hf'(x)$
h^2	0	0	1	-2	1	0	0	$h^2f''(x)$
h^2	0	-1	2	0	-2	1	0	$2h^3f^{(3)}(x)$
h^2	0	1	-4	6	-4	1	0	$h^4f^{(4)}(x)$
h^4	0	1	-8	0	8	-1	0	$12hf'(x)$
h^4	0	-1	16	30	16	-1	0	$12h^2f''(x)$
h^4	1	-8	13	0	-13	8	-1	$8h^3f^{(3)}(x)$
h^4	-1	12	-39	56	-39	12	-1	$6h^4f^{(4)}(x)$

Table 4.1: Συντελεστές τύπων αριθμητικής παραγώγισης με κεντρικές διαφορές, [3].

Κεφάλαιο 5

Μέθοδος Πεπερασμένων Διαφορών

Η μέθοδος των πεπερασμένων διαφορών βασίζεται σε τοπικές προσεγγίσεις των μερικών παραγώγων που εμφανίζονται σε μια Μ.Δ.Ε, οι οποίες προκύπτουν από σειρές Taylor. Η μέθοδος είναι αρκετά απλή στον ορισμό και μάλλον εύκολη στην εφαρμογή. Επίσης, είναι ιδιαίτερα χρήσιμη για απλές περιοχές, όπως ορθογώνια και ομοιόμορφα πλέγματα. Οι πίνακες των συστημάτων που προκύπτουν από τις διακριτοποιήσεις των Μ.Δ.Ε είναι συχνά συμμετρικοί και αραιοί, και συνήθως αποτελούν μη μηδενικά διαγώνια στοιχεία. Στο παρόν κεφάλαιο θα περιγράψουμε το πρόβλημα δύο σημείων.

Το πρόβλημα δύο σημείων, που συνήθως αναφέρεται ως το πρόβλημα των δύο συνοριακών σημείων, είναι ένα είδος προβλήματος που εφαρμόζεται σε διάφορα φυσικά φαινόμενα, συμπεριλαμβανομένων των προβλημάτων μεταφοράς θερμότητας, δυναμικής ρευστών, δομικής μηχανικής και βέλτιστου ελέγχου. Αποτελεσματικές και ακριβείς λύσεις σε προβλήματα οριακών τιμών δύο σημείων διαδραματίζουν κρίσιμο ρόλο στην κατανόηση και την πρόβλεψη της συμπεριφοράς των φυσικών συστημάτων και στη βελτιστοποίηση της απόδοσής τους. Σε αυτή τη περίπτωση το μαθηματικό μοντέλο, δηλαδή μια Μ.Δ.Ε, είναι ένα μοντέλο διάχυσης-αντίδρασης και καλούμαστε να βρούμε μια λύση αυτής που να ικανοποιεί τις συνοριακές συνθήκες σε δύο διαφορετικά σημεία.

Αυτές οι συνθήκες μπορεί να είναι προκαθορισμένες τιμές, η αλλιώς συνοριακές συνθήκες Dirichlet, τιμές παραγώγων, η αλλιώς συνοριακές συνθήκες Neumann, ή συνδυασμό και των δύο. Η εύρεση μιας λύσης που ικανοποιεί ταυτόχρονα τη Μ.Δ.Ε, και τις δεδομένες συνοριακές συνθήκες, είναι δύσκολη όταν αντιμετωπίζουμε προβλήματα δύο σημείων. Η λύση πρέπει να πληροί τόσο τους καθολικούς περιορισμούς που ορίζουν οι συνοριακές συνθήκες όσο και την τοπική συμπεριφορά της Μ.Δ.Ε.

5.1 Εισαγωγή στο Πρόβλημα Δύο Σημείων

Ζητείται μια συνάρτηση $u \in C^2[a,b]$, τέτοια ώστε

$$\begin{gathered} -u''(x) + q(x)u(x) = f(x),\ x \in [a,b],\ \text{με} \\ u(a) = u(b) = 0, \end{gathered} \tag{5.1.1}$$

όπου $a, b \in \mathbb{R},\ q, f \in C[a,b]$ και $q(x) > 0$, για κάθε $x \in [a,b]$.
Η εξίσωση (5.1.1) ορίζει το πρόβλημα δύο σημείων για μια Μ.Δ.Ε δεύτερης τάξης.
Έστω τώρα πως έχουμε την συνάρτηση w, για την οποία ισχύει $w \in C^4[a,b],\ x \in (a,b)$ και βήμα $h > 0$ τέτοιο ώστε $x+h, x-h \in [a,b]$. Αναπτύσσοντας την συνάρτηση w με τις σειρές Taylor για $x+h, x-h$ θα έχουμε

$$\begin{aligned} w(x+h) &= w(x) + hw'(x) + \frac{h^2}{2}w''(x) + \frac{h^3}{6}w'''(x) + \frac{h^4}{24}w^{(4)}(\xi_1), \\ w(x-h) &= w(x) - hw'(x) + \frac{h^2}{2}w''(x) - \frac{h^3}{6}w'''(x) + \frac{h^4}{24}w^{(4)}(\xi_{-1}), \end{aligned}$$

όπου $\xi_1, \xi_{-1} \in (x-h, x+h)$. Αν προσθέσουμε τις $w(x+h)$ και $w(x-h)$ θα πάρουμε

$$w(x-h) - 2w(x) + w(x+h) = h^2w''(x) + \frac{h^4}{24}[w^{(4)}(\xi_1) + w^{(4)}(\xi_{-1})]$$

Αν διαιρέσουμε με h^2 το αριστερό και δεξί μέλος του παραπάνω τύπου, και έπειτα μεταφέρουμε το $w''(x)$ στο αριστερό μέλος θα καταλήξουμε στον τύπο

$$\left|\frac{w(x-h)-2w(x)+w(x+h)}{h^2}-w''(x)\right| \leq h^2\frac{1}{24}\max_{a\leq\xi_1,\xi_{-1}\leq b}|w^{(4)}(\xi_1)+w^{(4)}(\xi_{-1})|. \tag{5.1.2}$$

Όταν η τιμή του βήματος h είναι αρκετά μικρή, η δεύτερη παράγωγος της συνάρτησης w σε ένα σημείο x του διαστήματος $[a,b]$, προσεγγίζεται με αρκετά καλή ακρίβεια από τη πεπερασμένη διαφορά $\frac{1}{h^2}[w(x-h)-2w(x)+w(x+h)]$. Το βήμα h ορίζει τη διαμέριση του χωρίου όπου ορίζεται η εξίσωση (5.1.1). Η διαμέριση ενός χωρίου αναφέρεται στην αποτελεσματική διαίρεση του χωρίου σε μικρότερα υποσύνολα, τα οποία θα είναι πεπερασμένα.
Έτσι οδηγούμαστε στην εξής μέθοδο πεπερασμένων διαφορών

$$-\frac{U_{i-1}-2U_i+U_{i+1}}{h^2}+q(x_i)U_i=f(x_i),\ i=1,...,J. \tag{5.1.3}$$

όπου $J\in\mathbb{N}$, $h=\frac{b-a}{J+1}$, το βήμα διαμέρισης του χωρίου $[a,b]$, και $x_i=a+ih,\ i=0,...,J+1$. Το $U=(U_0,U_1,...,U_{J+1})^T\in\mathbb{R}_0^{J+2}$ είναι η προσέγγιση της συνάρτησης u στο (5.1.1) στα σημεία $u(x_0),u(x_1),...,u(x_{J+1})$. Στην μέθοδο (5.1.3) καταλήξαμε προσεγγίζοντας την τιμή της δευτέρας παραγώγου $u''(x_i)$ της λύσης u του προβλήματος (5.1.1) με τη πεπερασμένη διαφορά $(u(x_{i-1})-2u(x_i)+u(x_{i+1}))/h^2$. Η προσεγγιστική λύση ικανοποιεί τις συνοριακές συνθήκες του προβλήματος (5.1.1). Δηλαδή έχουμε $U_0=U_{J+1}=0$. Η μέθοδος (5.1.3) μπορεί να μετατραπεί σε ένα γραμμικό σύστημα J x J με αγνώστους το διάνυσμα $[U_1,...,U_J]^T$ που να έχει την μορφή

$$A \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{J-1} \\ U_J \end{bmatrix} = h^2 \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{J-1}) \\ f(x_J) \end{bmatrix} \tag{5.1.4}$$

με τον A για πίνακα συντελεστών,

$$A = \begin{bmatrix} 2+h^2q(x_1) & -1 & & & 0 \\ -1 & 2+h^2q(x_2) & -1 & & \\ \ddots & \ddots & \ddots & & \\ & -1 & 2+h^2q(x_{n-1}) & -1 & \\ 0 & & -1 & 2+h^2q(x_n) & \end{bmatrix}$$

Ο πίνακας συντελεστών A του παραπάνω γραμμικού συστήματος είναι συμμετρικός γιατί ισχύει $a_{ij} = a_{ji}$, και τριδιαγώνιος επειδή $a_{ij} = 0$ για $|i-j| > 1$. Ο A είναι αντιστρέψιμος, που σημαίνει ότι η προσεγγιστική λύση U της συνάρτησης u υπάρχει και είναι μοναδική, και είναι θετικά ορισμένος μόνο όταν η συνάρτηση $q(x)$ λαμβάνει θετικές τιμές, για όλα τα $x \in [a,b]$. Μια άλλη σημαντική ιδιότητα του πίνακα A είναι πως είναι αυστηρά διαγώνια υπέρτερος, πράγμα που σημαίνει πως αν εφαρμόσουμε μια επαναληπτική μέθοδο για την επίλυση του, αυτή η μέθοδος θα συγκλίνει στην ακριβή λύση της συνάρτησης u στο πρόβλημα (5.1.1). Μπορούμε να γράψουμε τον A ως άθροισμα δύο $J \times J$ πινάκων, $A = T + Q$, όπου T ο τριδιαγώνιος πίνακας με στοιχεία $[-1, 2, -1]$ και Q ο διαγώνιος πίνακας με μη αρνητικά στοιχεία $Q_{ii} = h^2 q(x_i)$. Για να αποδείξουμε πως ο A είναι θετικά ορισμένος, ορίζουμε το διάνυσμα $y = [y_1, ..., y_J]^T \in \mathbb{R}^J$, και υπολογίζουμε το $y^T T y$ ως εξής

$$y^T T y = (y_1, ..., y_J)(2y_1 - y_2, -y_1 + 2y_2 - y_3, ..., -y_J + 2y_J)^T = \\ y_1^2 + (y_1 - y_2)^2 + \cdots + (y_{J-1} - y_J)^2 + y_J^2.$$

Από αυτή την σχέση συμπεραίνουμε ότι ο T είναι θετικά ορισμένος και άρα ο A είναι θετικά ορισμένος.

Θεώρημα 5.1.1 *Έστω $U \in \mathbb{R}_0^{J+2}$ η λύση του προβλήματος (5.1.3), με $U_0 = U_{J+1} = 0$. Τότε ισχύει η ακόλουθη ανισότητα,*

$$\max_{0 \le i \le J+1} |U_i| \le \max_{x \in [a,b]} |f(x_i)|. \tag{5.1.5}$$

Απόδειξη Από τη σχέση (5.1.3), εύκολα παίρνουμε,

$$(2 + h^2 q(x_i))U_i = U_{i+1} + U_{i-1} + f(x_i),\ 1 \le i \le J.$$

Η συνάρτηση $q(x)$ είναι συνεχής και θετική για $x \in [a, b]$, οπότε αν θέσουμε σαν ελάχιστη τιμή της συνάρτησης $q_{min} = \min_{x \in [a,b]} q(x)$, η παραπάνω ισότητα μας δίνει για κάθε $i = 1, ..., J$,

$$(2 + h^2 q_{min})|U_i| \le |U_{i+1}| + |U_{i-1}| + |f(x_i)| \\ \le 2 \max_{0 \le i \le J+1} |U_i| \le \max_{x \in [a,b]} |f(x_i)|.$$

Οπότε

$$(2 + h^2 q_{min}) \max_{1 \le i \le J} |U_i| \le 2 \max_{0 \le i \le J+1} |U_i| + \max_{x \in [a,b]} |f(x)|,$$

η οποία εύκολα δίνει τη ζητούμενη σχέση (5.1.5), [12]. □

Ευστάθεια Όταν μικρές μεταβολές στα δεδομένα του προβλήματος οδηγούν σε μικρές μεταβολές της αριθμητικής λύσης, τότε η αριθμητική μέθοδος λέγεται ευσταθής. Αντιθέτως, αν μεταβάλουμε τα δεδομένα του προβλήματος κατά ένα μικρό παράγοντα, και η αριθμητική λύση μεταβληθεί πολύ, τότε η αριθμητική μέθοδος είναι ασταθής. Στην ειδική περίπτωση που η διαφορική εξίσωση είναι

γραμμική, όπως είναι η (5.1.1), θέλουμε η αριθμητική λύση να φράσσεται με μια σταθερά επί τα δεδομένα, όπως η σχέση (5.1.5). Ανεξαρτήτως του προβλήματος που καλούμαστε να επιλύσουμε, η ευστάθεια της αριθμητικής μεθόδου είναι εσωτερική ιδιότητα του σχήματος. Από το Θεώρημα (5.1.1) μπορούμε να δείξουμε ότι το γραμμικό σύστημα που οδηγεί η σχέση (5.1.3) έχει μοναδική λύση. Αν θεωρήσουμε το αντίστοιχο ομογενές γραμμικό σύστημα, τότε από το Θεώρημα (5.1.1), οδηγούμαστε ότι η μοναδική λύση είναι η μηδενική λύση $U_i = 0,\ i = 0, ..., J+1$.

Συνέπεια Αν στη σχέση (5.1.3) αντικαταστήσουμε την προσέγγιση λύση U, με την ακριβή λύση u, τότε θα πάρουμε

$$-\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} + q(x_i)u(x_i) = f(x_i) + h^2\frac{1}{24}|u^{(4)}(\xi_1) + u^{(4)}(\xi_{-1})|,$$
$$i = 1, ..., J.$$

όπου $\xi_1, \xi_{-1} \in (x-h, x+h)$. Το διάνυσμα με συνιστώσες $u(x_i),\ i = 1, ..., J$, δεν θα ικανοποιεί τη σχέση (5.1.3) και θα υπάρχει σφάλμα, το οποίο αν τείνει στο μηδέν, καθώς το h τείνει στο μηδέν, τότε η μέθοδος θα είναι συνεπής.

Θεώρημα 5.1.2 *,[4], Έστω ότι η ακριβής λύση u του προβλήματος (5.1.1) είναι αρκετά ομαλή, δηλαδή $u \in \mathbb{C}^4[a,b]$. Τότε υπάρχει μια σταθερά c, η οποία είναι ανεξάρτητη του βήματος διαμέρισης h, τέτοια ώστε*

$$\max_{1\le i\le J} |u(x_i) - U_i| \le ch^2, \tag{5.1.6}$$

όπου $U \in \mathbb{R}_0^{J+2}$ ή λύση του (5.1.3).

Απόδειξη Ορίζουμε ως $E_i = u(x_i) - U_i,\ i = 0, ..., J+1$, όπου λόγω των σχέσεων $U_0 = u(a) = 0$ και $U_{J+1} = u(b) = 0$, έχουμε $E_0 = E_{J+1} = 0$. Αν

αφαιρέσουμε τις (5.1.3) και (5.1) κατά μέλη, παίρνουμε

$$E_{i+1} - (2 + q(x_i)h^2)E_i + E_{i-1} = h^2 r_i,\ i = 1, ..., N,$$

όπου $r_i = \frac{h^4}{24}[u^{(4)}(\xi_1) + u^{(4)}(\xi_{-1})]$, με $\xi_1, \xi_{-1} \in (x-h, x+h)$. Λόγω της σχέσης (5.1.2) θα έχουμε

$$\max_{1\le i\le J} |r_i| \le \frac{h^2}{12} \max_{a\le x\le b} |u^{(4)}(x)|.$$

Ορίζουμε στη συνέχεια $\tilde{E} = \max_{1\le i\le J} |E_i|, \tilde{r} = \max_{1\le i\le J} |r_i|$ και επειδή η συνάρτηση $q(x)$ είναι συνεχής και θετική, για $x \in [a, b], q_{min} = \min_{x\in[a,b]} q(x)$. Συνεπώς

$$(2 + q(x_i)h^2)E_i = E_{i+1} + E_{i-1} + h^2 r_i,$$

οπότε

$$(2 + q_{min}h^2)|E_i| \le 2\tilde{E} + h^2\tilde{r}.$$

Από όπου προκύπτει

$$q_{min}h^2 \max_{1\le i\le J} |E_i| \le h^2\tilde{r},$$

η οποία δίνει την ακόλουθη ανισότητα

$$\max_{1\le i\le J} |E_i| \le \tilde{r} \le ch^2$$

□

Βάση του θεωρήματος (5.1.2), η προσεγγιστική λύση συγκλίνει στην ακριβή λύση της συνάρτησης u καθώς το βήμα διαμέρισης h τείνει στο μηδέν. Επίσης, επειδή η εκτίμηση του σφάλματος είναι της τάξης του h^2, δηλαδή έχει τάξη ακρίβειας δύο, για αρκετά μικρό βήμα h, καθώς υποδιπλασιάζεται το h, το σφάλμα υποτετραπλασιάζεται.

Κεφάλαιο 6

Αριθμητικές Μέθοδοι Διαμέρισης Χωρίου

Η διαμέριση χωρίου [13] είναι μια τεχνική της αριθμητικής ανάλυσης που χρησιμοποιείται για την επίλυση μερικών διαφορικών εξισώσεων (Μ.Δ.Ε). Η κεντρική ιδέα είναι η διαμέριση του αρχικού χωρίου σε μικρότερα, απλούστερα χωρία, και λύνοντας τη Μ.Δ.Ε σε κάθε χωρίο ξεχωριστά. Στη συνέχεια, οι λύσεις συνδυάζονται για να ληφθεί η καθολική λύση της Μ.Δ.Ε στο αρχικό χωρίο. Έτσι, έχοντας μικρότερα χωρία, η Μ.Δ.Ε μπορεί να λυθεί ανεξάρτητα, και το προκύπτον σύστημα λύνεται με αριθμητικές μεθόδους όπως οι Jacobi και Gauss-Seidel, επιτυγχάνοντας γρηγορότερη σύγκλιση λόγω των μικρότερων συστημάτων. Αυτή η τεχνική επιτρέπει τη χρήση παράλληλων υπολογιστών, καθώς το υποπρόβλημα κάθε χωρίου μπορεί να λυθεί ανεξάρτητα σε διαφορετικό επεξεργαστή ή νήμα, γεγονός που μπορεί να οδηγήσει σε ταχύτερους χρόνους επίλυσης.

Υπάρχουν διάφοροι τύποι μεθόδων διαμέρισης του αρχικού χωρίου, όπως οι επικαλυπτόμενες και μη επικαλυπτόμενες μέθοδοι. Οι μέθοδοι επικάλυψης επιτρέπουν την επικάλυψη των χωρίων, δηλαδή το κάθε χωρίο μπορεί να περιλαμβάνει σημεία γειτονικών χωρίων, κάτι που μπορεί να βελτιώσει την ακρίβεια της λύσης, αλλά μπορεί επίσης να οδηγήσει σε πιο περίπλοκη επικοινωνία μεταξύ των επεξεργαστών. Οι μη επικαλυπτόμενες μέθοδοι δεν επιτρέπουν

επικαλυπτόμενα χωρία, γεγονός που απλοποιεί την επικοινωνία μεταξύ των επεξεργαστών, αλλά μπορεί να οδηγήσει σε λιγότερο ακριβείς λύσεις, και αυξημένο χρόνο εκτέλεσης.

Οι πιο συχνά χρησιμοποιούμενες μέθοδοι είναι οι Schwarz [2]. Οι μέθοδοι Schwarz διαιρούν το χωρίο σε επικαλυπτόμενα η μη επικαλυπτόμενα χωρία, και λύνουν τη Μ.Δ.Ε σε κάθε χωρίο ξεχωριστά. Η λύση σε κάθε χωρίο χρησιμοποιείται στη συνέχεια ως συνοριακή συνθήκη για τα γειτονικά χωρία, κάτι που βοηθά στη διασφάλιση της συνέχειας στα όρια του χωρίου. Παραδείγματα μεθόδων Schwarz περιλαμβάνουν τις Additive, Restricted και Multiplicative μεθόδους Schwarz.

Σε αυτό το κεφάλαιο εστιάζουμε το ενδιαφέρον μας στις μεθόδους διαμέρισης χωρίου Schwarz, και πιο συγκεκριμένα στην μη επικαλυπτόμενη Additive Schwarz μέθοδο. Θα εξετάσουμε την εναλλασσόμενη και την παράλληλη φιλοσοφία τους, θα δούμε πως περνάμε από το συνεχές επίπεδο, στο διακριτό ανάλογο τους, ώστε να είναι εφικτό να υλοποιηθούν προγραμματιστικά σε παράλληλο περιβάλλον.

6.1 Περιγραφή των Schwarz Μεθόδων

Εισαγωγικά για την εφαρμογή των μεθόδων

Έχουμε το πρόβλημα

$$\begin{aligned} -u'' + qu &= f \text{ στο } [0,1], \\ u(0) &= u(1) = 0, \end{aligned} \tag{6.1.1}$$

ορίζουμε ως $\Omega = [0,1]$, το οποίο διασπάται σε δύο επικαλυπτόμενα χωρία $\Omega_1 = [0,\beta]$ και $\Omega_2 = [\alpha,1]$, όπου $\alpha < \beta$ με διεπαφές τις $\Gamma_1 = \beta$ και $\Gamma_2 = \alpha$, και μια επικάλυψη $\beta - \alpha$, η οποία είναι ελάχιστη σε μήκος.

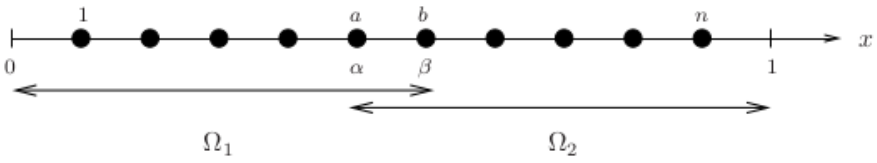

Figure 6.1: Το χωρίο Ω διασπάται σε δύο επικαλυπτόμενα χωρία Ω_1 και Ω_2. Εικόνα: https://www.unige.ch/~gander/Preprints/SchwarzHistorical.pdf, [2]

Επειδή η επικάλυψη είναι ελάχιστη, θα αναφέρεται ως μη επικαλυπτόμενη. Παρακάτω, οι διεπαφές $\Gamma_1 = \beta$ και $\Gamma_2 = \alpha$ θα χρησιμοποιούνται για την ανάκτηση των εσωτερικών συνοριακών συνθηκών στα δύο χωρία $\Omega_1,\ \Omega_2$, και οι προσεγγιστικές λύσεις του χωρίου Ω_1 δεν θα επικαλύπτονται από αυτές του χωρίου Ω_2 εξαιτίας της ελάχιστης επικάλυψης.

Έχοντας μια αρχική προσέγγιση $u_2^{(0)}$ της λύσης u στη διεπαφή Γ_1 (εσωτερικό συνοριακό σημείο του χωρίου Ω_1), υπολογίζουμε επαναληπτικά τις προσεγγιστικές λύσεις $u_1^{(n+1)}$ στο Ω_1 και $u_2^{(n+1)}$ στο Ω_2 σύμφωνα με τον παρακάτω αλγόριθμο

$$\begin{aligned} -\frac{d^2u_1^{(n+1)}}{dx^2} + qu_1^{(n+1)} &= f, \text{ στο } \Omega_1, \qquad -\frac{d^2u_2^{(n+1)}}{dx^2} + qu_2^{(n+1)} = f, \text{ στο } \Omega_2, \\ u_1^{(n+1)} &= u_2^{(n)}, \text{ στο } \Gamma_1, \qquad u_2^{(n+1)} = u_1^{(n+1)}, \text{ στο } \Gamma_2, \end{aligned} \tag{6.1.2}$$

ο οποίος περιγράφει την επαναληπτική εναλλασσόμενη μέθοδος Schwarz για το πρόβλημα (6.1.1).

Τα $u_1^{(n+1)}$ και $u_2^{(n+1)}$ ικανοποιούν τις συνθήκες Dirichlet της εξίσωσης (6.1.1) στα εξωτερικά όρια των χωρίων Ω_1 και Ω_2. Δηλαδή ισχύει πως $u_1^{(n+1)}(0) = 0$ και $u_2^{(n+1)}(1) = 0$.

Η εναλλασσόμενη μέθοδος Schwarz μπορεί εύκολα να επεκταθεί ώστε να περιλαμβάνει περισσότερα από δύο χωρία που προκύπτουν από το αρχικό χωρίο Ω. Ωστόσο, πρέπει να δοθεί προσοχή στη διατύπωση για να διασφαλιστεί ότι οι πιο πρόσφατες πληροφορίες χρησιμοποιούνται πάντα στις διεπαφές. Ο αλγόριθμος (6.1.2) μπορεί να γενικευτεί για J χωρία στον

$$\begin{aligned} -\frac{d^2u_j^{(n+1)}}{dx^2} + qu_j^{(n+1)} &= f, \qquad \text{στο } \Omega_j, \\ u_j^{(n+1)} &= u_k^{(n+1_{jk})}, \quad \text{στο } \Gamma_{jk} \end{aligned} \tag{6.1.3}$$

όπου το 1_{jk} ισούται με ένα εάν $j > k$ και μηδέν αλλιώς. Ο ορισμός του Γ_{jk} διασφαλίζει ότι η διεπαφή Γ_{jk} είναι το μέρος της διεπαφής του Ω_j στο Ω_k στο οποίο το Ω_k παρέχει την πιο πρόσφατη διαθέσιμη ενημέρωση για τον αλγόριθμο.

Εξετάζοντας τον αλγόριθμο (6.1.2), μπορούμε να διαπιστώσουμε πως οι λύσεις των χωρίων Ω_1 και Ω_2, μπορούν να επιλυθούν ταυτόχρονα αρκεί να τροποποιήσουμε τον αλγόριθμο με τον εξής τρόπο

$$\begin{aligned} -\frac{d^2u_1^{(n+1)}}{dx^2} + qu_1^{(n+1)} &= f, \text{ στο } \Omega_1, \qquad -\frac{d^2u_2^{(n+1)}}{dx^2} + qu_2^{(n+1)} = f, \text{ στο } \Omega_2, \\ u_1^{(n+1)} &= u_2^{(n)}, \text{ στο } \Gamma_1, \qquad\qquad u_2^{(n+1)} = u_1^{(n)}, \text{ στο } \Gamma_2, \end{aligned} \tag{6.1.4}$$

Έτσι καταλήγουμε στην παράλληλη μέθοδο Schwarz [14]. Η μόνη αλλαγή στον αλγόριθμο (6.1.4), είναι η επανάληψη στην δεύτερη συνθήκη, η οποία χρησιμοποιεί την προσεγγιστική λύση του χωρίου Ω_1 από την προηγουμένη επανάληψη. Χρησιμοποιώντας τις αρχικές προσεγγίσεις $u_1^{(0)}$ και $u_2^{(0)}$, μπορεί κανείς τώρα να υπολογίσει ταυτόχρονα και τις δύο λύσεις των χωρίων Ω_1 και Ω_2 για $n = 0, 1, \ldots$. Ωστόσο, δεν υπάρχει κανένα πλεονέκτημα σε αυτό το απλό σενάριο δύο χωρίων, επειδή η ακολουθία προσεγγιστικών λύσεων που υπολογίζεται σε κάθε δύο επαναλήψεις για το Ω_1, είναι η ίδια με την ακολουθία που υπολογίζεται σε μια επανάληψη, ξανά στο Ω_1, χρησιμοποιώντας την εναλλασσόμενη μέθοδο Schwarz. Όταν χρησιμοποιούνται πολυάριθμα χωρία, δεν υπάρχει πλέον η ισότητα των ακολουθιών προσεγγιστικών λύσεων μεταξύ των δύο μεθόδων, επομένως η παράλληλη επεξεργασία μπορεί να είναι επωφελής. Ωστόσο, υπάρχει ένα κρίσιμο ζήτημα στο σενάριο πολλών χωρίων που έχει

περιγραφεί στο [14], βάση του οποίου, για κάθε σημείο διεπαφής Γ_{jk} σε ένα χωρίο Ω_j, υπάρχει ένα γειτονικό χωρίο από όπου θα ανακτηθούν οι συνοριακές συνθήκες. μια γενίκευση του αλγορίθμου (6.1.4) για πολλά χωρία έχει τη μορφή

$$\begin{aligned} -\frac{d^2u_j^{(n+1)}}{dx^2} + qu_j^{(n+1)} &= f, \qquad \text{στο } \Omega_j, \\ u_j^{(n+1)} &= u_k^{(n)}, \quad \text{στο } \Gamma_{jk} \end{aligned} \tag{6.1.5}$$

Διακριτοποίηση μεθόδων Schwarz Αν διακριτοποιήσουμε την συνεχής εναλλασσόμενη μέθοδο Schwarz χρησιμοποιώντας την μέθοδο πεπερασμένων διαφορών, και επιβάλλουμε τις συνθήκες διεπαφής στο δεξιό μέλος, τότε καταλήγουμε στο παρακάτω γραμμικό σύστημα για το πρώτο χωρίο Ω_1.

$$A_1u_1^{(n+1)} = f_1 - A_{12}u_2^{(n)}$$

Ο πίνακας A_1 προκύπτει από την διακριτοποίηση της διαφορικής εξίσωσης, ενώ ο πίνακας A_{12} είναι μηδενικός, εκτός από τους αγνώστους στη διεπαφή Γ_1. Παρόμοια με το πρώτο, είναι και το δεύτερο χωρίο Ω_2, με τη διαφορά ότι αξιοποιεί τη προσεγγιστική λύση του πρώτου χωρίου για την τρέχων επανάληψη.

$$A_2u_2^{(n+1)} = f_2 - A_{21}u_1^{(n+1)}$$

Γενικεύοντας για πολλά χωρία έχουμε

$$A_ju_j^{(n+1)} = f_j - \sum_{k=1}^{j-1} A_{jk}u_k^{(n+1)} - \sum_{k=j+1}^{J} A_{jk}u_k^{(n)} \tag{6.1.6}$$

Όμοια, μπορούμε να διακριτοποιήσουμε και την παράλληλη συνεχής μέθοδο Schwarz, αρχικά για το πρώτο χωρίο Ω_1

$$A_1 u_1^{(n+1)} = f_1 - A_{12} u_2^{(n)}$$

το οποίο είναι ίδιο με το πρώτο χωρίο της εναλλασσόμενης μεθόδου. Για το Ω_2 έχουμε

$$A_2 u_2^{(n+1)} = f_2 - A_{21} u_1^{(n)},$$

και πλέον δεν χρησιμοποιούμε τη προσεγγιστική λύση της τρέχων επανάληψης από το πρώτο χωρίο, διότι αλλιώς δεν θα ήταν παραλληλοποιήσιμη. Για πολλά χωρία ο σχετικός τύπος είναι

$$A_j u_j^{(n+1)} = f_j - \sum_{k \neq j}^{J} A_{jk} u_k^{(n)} \tag{6.1.7}$$

6.2 Υλοποίηση και τεχνικά χαρακτηριστικά της μεθόδου Additive Schwarz

Προκειμένου να επιτευχθεί μια διαμέριση χωρίου για το διακριτό γραμμικό σύστημα $Au = f$, χρειάζεται να διαιρεθούν οι άγνωστες μεταβλητές στο διάνυσμα u σε υποσύνολα, με παρόμοιο τρόπο που διαιρέθηκε το αρχικό χωρίο σε μικρότερα μη επικαλυπτόμενα χωρία. Αυτό μπορεί να επιτευχθεί με τη χρήση απλών τελεστών περιορισμού. Εάν θέλουμε, για παράδειγμα, να χωρίσουμε τους αγνώστους σε ένα πρώτο και ένα δεύτερο μη επικαλυπτόμενο σύνολο, μπορούμε να χρησιμοποιήσουμε τους πίνακες περιορισμού R_1 και R_2.

$$R_1 = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}, \; R_2 = \begin{bmatrix} & 1 & \\ & & \ddots \\ & & & 1 \end{bmatrix}$$

Τα στοιχεία των R_1 και R_2 είναι παντού μηδέν, εκτός από τις θέσεις που υποδεικνύονται με ένα 1. Έχοντας τον πίνακα συντελεστών A διάστασης

$n \times n$, που προκύπτει από τη διακριτοποίηση του προβλήματος (6.1.1) στο χωρίο $\Omega = [0, 1]$, ο πίνακας R_1 εκφράζει το χωρίο $\Omega_1 = [0, \beta]$, που προκύπτει από τη διαμέριση του χωρίου Ω. Αντίστοιχα ισχύει και για τον πίνακα R_2. Με αυτούς τους πίνακες περιορισμού, το $R_1 u$ δίνει το πρώτο σύνολο αγνώστων, και το $R_2 u$ το δεύτερο. Μπορούμε επίσης να ορίσουμε έναν περιορισμό του πίνακα συντελεστών A στο πρώτο και στο δεύτερο σύνολο αγνώστων, χρησιμοποιώντας τους ίδιους περιοριστικούς πίνακες R_1 και R_2.

$$A_j = R_j A R_j^T,\ j = 1, 2$$

και

$$A_{jk} = (R_j A - A_j R_j) R_k^T,\ k, j = 1, 2.$$

Η Additive Schwarz μέθοδος ορίζεται από τον τύπο

$$u^{(n+1)} = u^{(n)} + (R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2)(f - A u^{(n)}), \tag{6.2.1}$$

από τον οποίο μπορούν να επιλυθούν παράλληλα οι λύσεις των χωρίων $\Omega_j,\ j = 1, 2$, όπως στον αλγόριθμο (6.1.5). Ο τύπος (6.2.1) προφανώς δεν μπορεί να χρησιμοποιηθεί στη πράξη λόγω των πολλών και πολύπλοκων πράξεων που περιέχει. Θα πρέπει να τον μετατρέψουμε σε μια κατάλληλη μορφή για να είναι εφικτή η υλοποίηση του προγραμματιστικά. Έτσι για το πρώτο χωρίο Ω_1 έχουμε την εξής απλοποίηση

$$\begin{aligned}
R_1(f - A u^{(n)}) &= f_1 - A_{12} u_2^{(n)} - A_1 u_1^{(n)}, \\
A_1^{-1} R_1(f - A u^{(n)}) &= A_1^{-1}(f_1 - A_{12} u_2^{(n)}) - u_1^{(n)}, \\
R_1^T A_1^{-1} R_1(f - A u^{(n)}) &= \begin{bmatrix} A_1^{-1}(f_1 - A_{12} u_2^{(n)}) - u_1^{(n)} \\ 0 \end{bmatrix}
\end{aligned}$$

Παρόμοια και για το δεύτερο χωρίο Ω_2

$$
\begin{aligned}
R_2(f - Au^{(n)}) &= f_2 - A_{21}u_1^{(n)} - A_2u_2^{(n)},\\
A_2^{-1}R_2(f - Au^{(n)}) &= A_2^{-1}(f_2 - A_{21}u_1^{(n)}) - u_2^{(n)},\\
R_2^TA_2^{-1}R_2(f - Au^{(n)}) &= \begin{bmatrix} 0 \\ A_2^{-1}(f_2 - A_{21}u_1^{(n)}) - u_2^{(n)} \end{bmatrix}
\end{aligned}
$$

καταλήγοντας στον τύπο

$$
u^{(n+1)} = \begin{bmatrix} u_1^{(n)} \\ u_2^{(n)} \end{bmatrix} + \begin{bmatrix} A_1^{-1}(f_1 - A_{12}u_2^{(n)}) - u_1^{(n)} \\ A_2^{-1}(f_2 - A_{21}u_1^{(n)}) - u_2^{(n)} \end{bmatrix}
$$

από τον οποίο τα $u_1^{(n)}$ και $u_2^{(n)}$ διαγράφονται. Οπότε ο τελικός τύπος είναι ο

$$
u^{(n+1)} = \begin{bmatrix} A_1^{-1}(f_1 - A_{12}u_2^{(n)}) \\ A_2^{-1}(f_2 - A_{21}u_1^{(n)}) \end{bmatrix} \tag{6.2.2}
$$

που γράφεται σε μορφή πινάκων

$$
\begin{aligned}
A_1u_1^{(n+1)} &= f_1 - A_{12}u_2^{(n)},\\
A_2u_2^{(n+1)} &= f_2 - A_{21}u_1^{(n)}.
\end{aligned} \tag{6.2.3}
$$

Η (6.2.3) είναι πανομοιότυπη με την (6.1.7) για δύο χωρία. Στην περίπτωση των μη επικαλυπτόμενων χωρίων, η μέθοδος Additive Schwarz μπορεί να αναπαρασταθεί εναλλακτικά με μια τροποποιημένη μέθοδο Jacobi που διασπάει ένα γραμμικό σύστημα σε μικρότερα συστήματα με τετραγωνικούς πίνακες συντελεστών. Η αναπαράσταση αυτής της μεθόδου είναι

$$
\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} u_1^{(n+1)} \\ u_2^{(n+1)} \end{bmatrix} = \begin{bmatrix} 0 & -A_{12} \\ -A_{21} & 0 \end{bmatrix} \begin{bmatrix} u_1^{(n)} \\ u_2^{(n)} \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}. \tag{6.2.4}
$$

Αν τα χωρία Ω_1 και Ω_2 είναι επικαλυπτόμενα, που σημαίνει πως ορισμένα στοιχεία του περιοριστικού πίνακα R_1 υπάρχουν στον R_2, τότε η διαγραφή των $u_1^{(n)}$ και $u_2^{(n)}$ που περιγράφηκε παραπάνω, δεν είναι πλέον εφικτή. Έτσι, βάση του [2], οι μη μηδενικοί όροι στο $A_1^{-1}(f_1 - A_{12}u_2^{(n)}) - u_1^{(n)}$ από τη λύση

του πρώτου χωρίου, και οι μη μηδενικοί όροι στο $A_2^{-1}(f_2 - A_{21}u_1^{(n)}) - u_2^{(n)}$ από τη λύση του δεύτερου χωρίου επικαλύπτονται, και τα $u_1^{(n)}$ και $u_2^{(n)}$ αφαιρούνται δύο φορές, και προστίθεται η καινούργια προσέγγιση και από τα δύο χωρία. Γενικεύοντας την (6.2.4) για περισσότερα από δύο μη επικαλυπτόμενα χωρία, θα καταλήξουμε στον γενικό τύπο

$$A_j u_j^{(n+1)} = -\sum_{k \neq j}^{J} A_{jk} u_k^{(n)} + f_j, \ j = 1, ..., J-1, \tag{6.2.5}$$

ο οποίος είναι ίδιος με τον γενικό παράλληλο τύπο (6.1.7).

6.3 Υλοποίηση της Additive Schwarz Μεθόδου

Στις προηγούμενες δύο ενότητες αυτού του κεφαλαίου, περιγράφηκε η εναλλασσόμενη και παράλληλη μέθοδος Schwarz, στο συνεχές, αλλά και στο διακριτό επίπεδο. Βάση αυτών προέκυψε η Additive Schwarz (6.2.1) μέθοδος, η οποία είναι εκ φύσεως παράλληλη. Θεωρώντας το πρόβλημα (6.1.1), το διακριτό ανάλογο του, σύμφωνα με τη μέθοδο πεπερασμένων διαφορών, είναι το (5.1.5), από το οποίο προκύπτει το γραμμικό σύστημα $AU = F$. Για λόγους συμβατότητας με το παρόν κεφάλαιο, το $AU = F$ θα αναφέρεται ως $Au = f$. Επειδή ο πίνακας συντελεστών A είναι τριδιαγώνιος, τα $A_{jk}, \ j,k = 1, ..., J$ του γενικού τύπου (6.2.5) θα έχουν ένα μοναδικό μη μηδενικό στοιχείο με τιμή -1, που θα βρίσκεται σε σταθερές θέσεις στον πίνακα A, για κάθε χωρίο $\Omega_j, \ \ j = 1, ..., J$. Έτσι, το αποτέλεσμα του πολλαπλασιασμού του A_{jk} με τα διανύσματα των προσεγγιστικών λύσεων $u_k^{(n)}$, θα δίνει πάντα έναν πραγματικό αριθμό, που αναπαριστά την πληροφορία της διεπαφής μεταξύ των χωρίων. Έχοντας αυτό κατά νου, μπορούμε να παραλείψουμε τους άσκοπους πολλαπλασιασμούς, και αντί αυτών, να πολλαπλασιάσουμε το -1 με το αντίστοιχο στοιχείο του διανύσματος $u_k^{(n)}$.

Σημαντική προϋπόθεση για την διαμέριση του χωρίου, είναι το αποτέλεσμα

της διαίρεσης $\frac{n}{NUMBER\_OF\_DOMAINS}$ να είναι ακέραιος αριθμός, ώστε να είναι εφικτή η εύρεση των A_j χωρίς να χρειάζεται να υπολογιστεί μέσω του τύπου $R_j A R_j^T$. Παρακάτω υπάρχει η αναλυτική μορφή του αλγόριθμου.

ΑΛΓΟΡΙΘΜΟΣ 4 Παράλληλη Μέθοδος Additive Schwarz

Input U, F, A, *TOL*, max_iterations, N, NUMBER_OF_DOMAINS, Block_Size
Output U

procedure PARALLEL_ASM()

```
1:  for i = 1 to max_iterations do
2:    prev_U = U
3:    local_error = [N]
4:    START_PARALLEL_REGION
5:    for j = 0 to NUMBER_OF_DOMAINS do
6:      if j > 0 and j < NUMBER_OF_DOMAINS - 1 then
7:        GAUSS_SEIDEL(U, F, A, TOL, max_iterations, N, j*Block_Size, Block_Size*(j+1),
          NUMBER_OF_DOMAINS-1, j, [prev_U[Block_Size*j-1]*-1, prev_U[Block_Size*(j+1)]*-
          1])
8:      else if j = NUMBER_OF_DOMAINS - 1 then
9:        GAUSS_SEIDEL(U, F, A, TOL, max_iterations, N, j*Block_Size, Block_Size*(j+1),
          NUMBER_OF_DOMAINS-1, j, [prev_U[Block_Size*j-1]*-1, 0])
10:     else
11:       GAUSS_SEIDEL(U, F, A, TOL, max_iterations, N, j*Block_Size, Block_Size*(j+1),
          NUMBER_OF_DOMAINS-1, j, [0, prev_U[Block_Size*(j+1)]*-1])
12:     end if
13:     for i=Block_Size*j to Block_Size*(j+1) do
14:       local_error[i]=abs(U[i]-prev_U[i])
15:     end for
16:   end for
17:   END_PARALLEL_REGION
18:   max_error=get_max_error(local_error)
19:   if max_error < TOL then
20:     finish()
21:   end if
22: end for
23:
24: return X
```

Ξεκινώντας, ο βρόχος θα εκτελεστεί για max_iterations επαναλήψεις, οι προσεγγιστικές λύσεις του διανύσματος U θα αποθηκευτούν στην μεταβλητή prev_U, ώστε αργότερα να συγκριθούν με τις καινούργιες προσεγγιστικές λύσεις του U,

και να χρησιμοποιηθούν για την επίλυση των γραμμικών υποσυστημάτων που προκύπτουν από την διαμέριση του χωρίου Ω. Στην γραμμή 5, ο βρόχος θα εκτελεστεί NUMBER_OF_DOMAINS φορές, που υποδηλώνει τον αριθμό των χωρίων που θα διασπαστεί το αρχικό χωρίο Ω. Έπειτα, σε κάθε συνθήκη ελέγχου, θα συγκρίνεται ο αριθμός του τρέχων χωρίου, και αναλόγως του αποτελέσματος, υπάρχουν τρεις περιπτώσεις

- Ο αριθμός του χωρίου είναι 0, δηλαδή είναι το πρώτο χωρίο Ω_1.
- Ο αριθμός του χωρίου είναι NUMBER_OF_DOMAINS - 1, δηλαδή είναι το τελευταίο χωρίο Ω_J.
- Ο αριθμός του χωρίου είναι μεγαλύτερος από το μηδέν και από το NUMBER_OF_DOMAINS -1, δηλαδή είναι ένα ενδιάμεσο χωρίο.

Και στις τρεις περιπτώσεις εκτελείται μια τροποποιημένη σειριακή μέθοδος Gauss Seidel ξεκινώντας την εκτέλεση της από την γραμμή j*Block_Size, και τερματίζοντας στην Block_Size*(j+1). Έχοντας για κάθε χωρίο, τον αριθμό της γραμμής εκκίνησης, και τον αριθμό της γραμμής τερματισμού, μπορούμε και καθορίζουμε το A_j.
Η τελευταία παράμετρος της μεθόδου Gauss Seidel, η οποία είναι μια λίστα δύο στοιχείων, καθορίζει την πληροφορία στις διεπαφές του κάθε χωρίου. Η πρώτη συνθήκη ελέγχου αναπαριστά τα χωρία που είναι ενδιάμεσα από το πρώτο και το τελευταίο χωρίο, οπότε δέχονται πληροφορίες από το γειτονικό χωρίο στα δεξιά και από αυτό στα αριστερά. Έτσι, το πρώτο στοιχείο στη λίστα αναπαριστά την πληροφορία του αριστερού γειτονικού χωρίου, και το δεύτερο στοιχείο αναπαριστά την πληροφορία του δεξιού γειτονικού χωρίου. Αντίστοιχα, η ίδια διαδικασία εφαρμόζεται στο πρώτο, και στο τελευταίο χωρίο.
Στην γραμμή 13 μέχρι 15, υπολογίζεται το σφάλμα για την προσεγγιστική λύση $u_j^{(n+1)}$ του κάθε χωρίου $\Omega_j,\ j = 1, ..., J$. Τα επόμενα βήματα είναι ίδια

με αυτά των παράλληλων αλγορίθμων Jacobi και Jacobi-Gauss Seidel.

Κεφάλαιο 7

Εκτέλεση και Αποτελέσματα Αριθμητικών Μεθόδων

Θέλουμε να λύσουμε το πρόβλημα δύο σημείων που περιγράφηκε στο κεφάλαιο της μεθόδου πεπερασμένων διαφορών

$$\begin{aligned} -u''(x) + q(x)u(x) = f(x),\ x \in [a,b],\ \text{με} \\ u(a) = u(b) = 0, \end{aligned} \tag{7.1}$$

όπου $a, b \in \mathbb{R},\ q, f \in C[a,b]$ και $q(x) > 0$, για κάθε $x \in [a,b]$.
Χρησιμοποιώντας την μέθοδο πεπερασμένων διαφορών (5.1.5), καταλήγουμε στο διακριτό ανάλογο του παραπάνω προβλήματος, δηλαδή στο γραμμικό σύστημα $AU = F$.

$$A = \begin{bmatrix} 2 + h^2 q(x_1) & -1 & & & 0 \\ -1 & 2 + h^2 q(x_2) & -1 & & \\ \ddots & \ddots & \ddots & & \\ & -1 & 2 + h^2 q(x_{n-1}) & -1 & \\ 0 & & -1 & 2 + h^2 q(x_n) & \end{bmatrix}$$

$$U = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{J-1} \\ U_J \end{bmatrix} \text{ και } F = \begin{bmatrix} h^2 f(x_1) \\ h^2 f(x_2) \\ \vdots \\ h^2 f(x_{J-1}) \\ h^2 f(x_J) \end{bmatrix}$$

Έχοντας το πρόβλημα (7.1) στη διακριτή του μορφή, μπορούμε πλέον να εφαρμόσουμε τις επαναληπτικές μεθόδους Jacobi και Gauss Seidel, καθώς και την μέθοδο διαμέρισης χωρίου Additive Schwarz. Στα προηγούμενα κεφάλαια, περιγράφηκαν οι παράλληλοι αλγόριθμοι των Jacobi και Jacobi-Gauss Seidel, οι οποίοι εφαρμόζονται σε γενικούς πίνακες συντελεστών A, που μπορεί να είναι πυκνοί, η αραιοί. Η τυπική διαδικασία που ακολουθούν για την εύρεση μιας προσεγγιστικής λύσης, είναι η άθροιση των επιμέρους στοιχείων κάθε γραμμής, πολλαπλασιασμένων με το αντίστοιχο στοιχείο του διανύσματος της τρέχων, η της προηγούμενης προσεγγιστικής λύσης. Αυτή η διαδικασία, όσο χρήσιμη και να είναι στην γενική χρήση της, δεν θα χρησιμοποιηθεί για την εύρεση της προσεγγιστικής λύσης του προβλήματος (7.1). Επειδή ο πίνακας συντελεστών A του προβλήματος (7.1) είναι τριδιαγώνιος, δηλαδή τα μη μηδενικά στοιχεία του βρίσκονται μια θέση δεξιά και αριστερά του διαγώνιου στοιχείου, η άθροιση και ο πολλαπλασιασμός των επιμέρους στοιχείων μιας γραμμής i με το διάνυσμα της προσεγγιστικής λύσης $U^{(k)}, k = 1, ..., n$, περιορίζονται μόνο στα δύο στοιχεία, δηλαδή αυτά που βρίσκονται δεξιά και αριστερά του διαγώνιου στοιχείου. Έτσι, τροποποιώντας τους παράλληλους αλγορίθμους Jacobi και Jacobi-Gauss Seidel για να επιλύουν τριδιαγώνιους πίνακες, σχηματίζονται οι παρακάτω αλγόριθμοι

ΑΛΓΟΡΙΘΜΟΣ 5 Παράλληλη Μέθοδος Jacobi για Τριδιαγώνιους Πίνακες

Input U, F, A, *TOL*, max_iterations, N, NUMBER_OF_THREADS, groups
Output U

procedure PARALLEL_JACOBI()

```
1:  for i = 1 to max_iterations do
2:    prev_U = U
3:    local_error = [N]
4:    START_PARALLEL_REGION
5:    for g = 0 to groups.size()-1 do
6:      for variable in groups[g] do
7:        if variable > 0 and variable < N-1 then
8:          U[variable] = (1.0f / A[variable * N + variable]) * (F[variable] -
            (A[variable * N + (variable-1)]*prev_U[variable-1]) - (A[variable * N + (vari-
            able+1)]*prev_U[variable+1]))
9:        else if variable = N-1 then
10:         U[variable] = (1.0f / A[variable * N + variable]) * (F[variable] - (A[variable * N
            + (variable-1)]*prev_U[variable-1]))
11:       else
12:         X[variable] = (1.0f / A[variable * N + variable]) * (F[variable] - (A[variable * N
            + (variable+1)]*prev_U[variable+1]))
13:       end if
14:       local_error[variable]=abs(U[variable]-prev_U[variable])
15:     end for
16:   end for
17:   END_PARALLEL_REGION
18:   max_error=get_max_error(local_error)
19:   if max_error < TOL then
20:     finish()
21:   end if
22: end for
23:
24: return X
```

ΑΛΓΟΡΙΘΜΟΣ 6 Παράλληλη Μέθοδος Jacobi-Gauss Seidel για Τριδιαγώνιους Πίνακες

Input U, F, A, *TOL*, max_iterations, N, NUMBER_OF_THREADS, groups
Output U

procedure PARALLEL_JACOBI_GAUSS_SEIDEL()

```
1: for i = 1 to max_iterations do
2:   prev_U = U
3:   local_error = [N]
4:   START_PARALLEL_REGION
5:   for g = 0 to groups.size()-1 do
6:     for variable in groups[g] do
7:       if variable > 0 and variable < N-1 then
8:         U[variable] = (1.0f / A[variable * N + variable]) * (F[variable] - (A[variable * N
           + (variable-1)]*U[variable-1]) - (A[variable * N + (variable+1)]*U[variable+1]))
9:       else if variable = N-1 then
10:        U[variable] = (1.0f / A[variable * N + variable]) * (F[variable] - (A[variable * N
           + (variable-1)]*U[variable-1]))
11:      else
12:        U[variable] = (1.0f / A[variable * N + variable]) * (F[variable] - (A[variable * N
           + (variable+1)]*U[variable+1]))
13:      end if
14:      local_error[variable]=abs(U[variable]-prev_U[variable])
15:    end for
16:  end for
17:  END_PARALLEL_REGION
18:  max_error=get_max_error(local_error)
19:  if max_error < TOL then
20:    finish()
21:  end if
22: end for
23:
24: return X
```

Η διαφορά των παραπάνω παράλληλων αλγορίθμων με αυτών των γενικών αλγορίθμων που εξετάστηκαν στο κεφάλαιο των επαναληπτικών μεθόδων, είναι στο σώμα του εμφωλευμένου βρόχου for. Ειδικότερα, υπάρχουν τρεις περιπτώσεις

- Η γραμμή i που εξετάζεται στην τρέχων επανάληψη είναι $i > 0$ και $i < N - 1$ όπου N η τάξη του A.
- Η γραμμή i που εξετάζεται στην τρέχων επανάληψη είναι $i = N - 1$.

- Η γραμμή i που εξετάζεται στην τρέχων επανάληψη είναι $i = 0$.

Η πρώτη περίπτωση που υλοποιείται στη γραμμή 8, για τον παράλληλο αλγόριθμο Jacobi, μπορεί να αναπαρασταθεί μαθηματικά ως εξής

$$U_i^{(k+1)} = \frac{1}{a_{ii}}(F_i - (a_{ij-1}U_{j-1}^{(k)} - a_{ij+1}U_{j+1}^{(k)}), \text{όταν } i = j.$$

Από την πρώτη περίπτωση γίνεται προφανές πως για μια γραμμή $i, i = 1, ..., n$, χρησιμοποιείται το γινόμενο $a_{ij-1}x_{j-1}^{(k)}$, όπου το a_{ij-1} είναι το αριστερό στοιχείο του διαγώνιου στοιχείου a_{ii}, και αντίστοιχα το $a_{ij+1}x_{j+1}^{(k)}$, όπου το a_{ij+1} είναι το δεξιό στοιχείο του διαγώνιου στοιχείου a_{ii}.
Με ανάλογο τρόπο θα γίνει και η μαθηματική αναπαράσταση των άλλων δύο περιπτώσεων. Για την δεύτερη περίπτωση έχουμε

$$U_i^{(k+1)} = \frac{1}{a_{ii}}(F_i - a_{ij-1}U_{j-1}^{(k)}), \text{όταν } i = j,$$

και για την τρίτη

$$U_i^{(k+1)} = \frac{1}{a_{ii}}(F_i - a_{ij+1}U_{j+1}^{(k)}), \text{όταν } i = j.$$

Παρόμοια μπορούμε αναπαραστήσουμε και τις περιπτώσεις του αλγορίθμου Jacobi-Gauss Seidel αλλάζοντας, αναλόγως την περίπτωση, τα $U^{(k)}$ με $U^{(k+1)}$.
Ο αλγόριθμος της μεθόδου Additive Schwarz που περιγράφηκε στην υποενότητα (6.3) παραμένει ως έχει.
Εκτελώντας τις μεθόδους Jacobi και Jacobi-Gauss Seidel με 1, 2, 4, και 6 νήματα, και την μέθοδο Additve Schwarz με 1, 2, 4, 5, και 8 νήματα, για διάφορες τάξεις του πίνακα συντελεστών A, προκύπτουν τα παρακάτω γραφήματα στην εικόνα (7.1) με τους αντίστοιχους πίνακες τους. Γενικά θέλουμε να παρατηρήσουμε τη συμπεριφορά των μεθόδων και να περιγράψουμε τα αποτελέσματα τους για σχετικά "μεγάλη" τάξη γραμμικού συστήματος, καθώς η ακριβής διακριτοποίηση μιας Μ.Δ.Ε μας οδηγεί σε τέτοια γραμμικά συστήματα. Η επιλογή του πλήθους των νημάτων που χρησιμοποιήθηκαν για την εκτέλεση και

τον πειραματισμό των παραπάνω επαναληπτικών μεθόδων, σχετίζεται με την αποδοτικότητα τους που περιγράφεται παρακάτω.
Ως κριτήρια τερματισμού έχουμε

- Ο μέγιστος αριθμός επαναλήψεων των επαναληπτικών μεθόδων.
- μια αυθαίρετη τιμή ανοχής ορισμένη από τον χρήστη.

Τα κριτήρια τερματισμού, και ιδιαίτερα η τιμή ανοχής, επηρεάζουν σε μεγάλο βαθμό τον χρόνο ολοκλήρωσης των μεθόδων, διότι αναλόγως των απαιτήσεων ακρίβειας των μεθόδων, ο χρόνος ολοκλήρωσης μιας μεθόδου για μια δεδομένη τάξη N του γραμμικού συστήματος $Au = f$, μπορεί να είναι πιο αργός από τον αναμενόμενο.

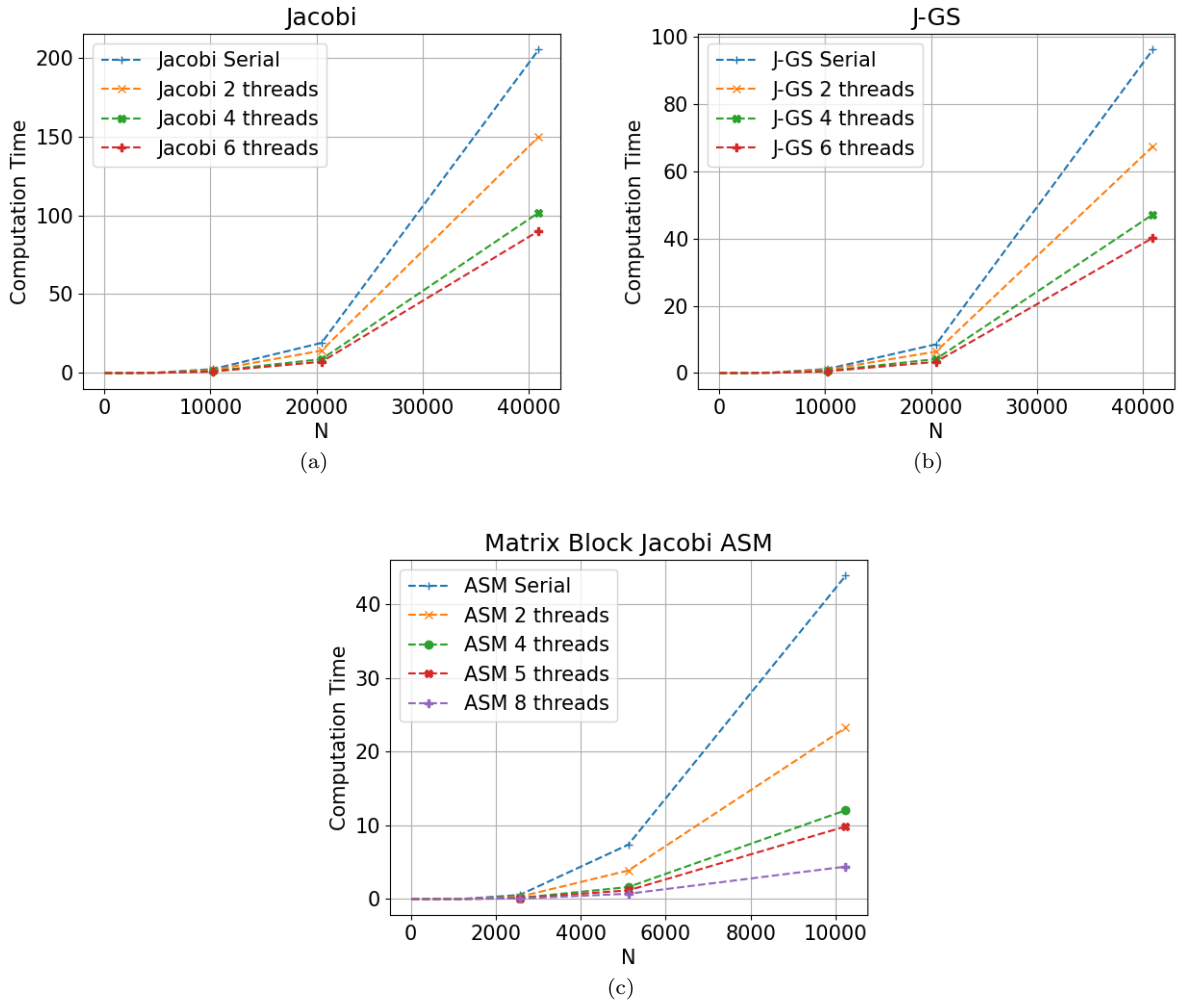

Figure 7.1: Αποτελέσματα εκτέλεσης των μεθόδων Jacobi, Jacobi-Gauss Seidel, και της Additive Schwarz μεθόδου. Στον x άξονα αναπαρίσταται η τάξη του γραμμικού συστήματος, και στον άξονα y ο χρόνος υπολογισμού.

Αποτελέσματα Εκτέλεσης Jacobi

N	1 Νήμα	2 Νήματα	4 Νήματα	6 Νήματα	Επαναλήψεις	Σφάλμα E
21	2.00E-06	0.000203	0.000336	0.001085	4	1.50
41	3.00E-06	0.000223	0.000335	0.001075	5	1.03
81	8.00E-06	0.000252	0.000378	0.000191	7	0.77
161	2.40E-05	0.000366	0.000481	0.000573	12	0.23
321	0.000102	0.000264	0.000247	0.000253	26	0.05
641	0.000571	0.000714	0.00066	0.000618	75	0.01
1281	0.004809	0.004668	0.003799	0.003344	257	3.72E-03
2561	0.040213	0.034356	0.020601	0.031529	921	9.31E-04
5121	0.304706	0.244915	0.149439	0.140728	3332	2.32E-04
10241	2.501331	1.912369	1.211718	0.9552	11995	5.81E-05
20481	19.115581	14.215506	8.797289	7.145356	42782	1.39E-05
40961	205.073114	149.860024	101.710744	90.082974	170000	3.14E-06

Table 7.1: Αποτελέσματα χρόνου εκτέλεσης της επαναληπτικής μεθόδου Jacobi με 1, 2, 4, και 6 νήματα.

Αποτελέσματα Εκτέλεσης Jacobi-Gauss Seidel

N	1 Νήμα	2 Νήματα	4 Νήματα	6 Νήματα	Επαναλήψεις	Σφάλμα E
21	0.000002	0.000193	0.000327	0.001437	4	1.50
41	3.00E-06	0.000219	0.000347	0.000557	5	1.03
81	7.00E-06	0.000255	0.000374	0.000467	6	0.77
161	2.00E-05	0.000334	0.000449	0.000967	10	0.23
321	7.10E-05	0.000187	0.000212	0.00023	18	0.05
641	0.000345	0.000559	0.001511	0.003976	44	0.01
1281	0.002669	0.002798	0.002461	0.002405	138	3.72E-03
2561	0.022015	0.017793	0.01188	0.008763	484	9.31E-04
5121	0.154548	0.129266	0.077448	0.065958	1734	2.32E-04
10241	1.235061	0.995226	0.630719	0.531857	6224	5.82E-05
20481	8.554399	6.388804	4.166344	3.323508	20000	1.30E-05
40961	96.260279	67.436694	47.184837	40.220563	78631	3.23E-06

Table 7.2: Αποτελέσματα χρόνου εκτέλεσης της επαναληπτικής μεθόδου Jacobi-Gauss Seidel με 1, 2, 4, και 6 νήματα.

Αποτελέσματα Εκτέλεσης ASM

N	1 Χωρίο	2 Χωρία	4 Χωρία	5 Χωρία	8 Χωρία	Επανα-λήψεις	Σφάλμα E
22	6.95E-06	0.000572803	0.005093547	0.003781847	-	4	1.54
42	1.22E-05	0.00269473	0.006135972	0.006275305	-	5	1.06
82	3.24E-05	0.006551226	0.001780578	0.004404544	0.001108121	7	0.78
162	0.000110652	0.002538583	0.005458087	0.005112489	0.000855317	10	0.23
322	0.000534559	0.000727338	0.004862713	0.00858845	0.000556775	16	0.06
642	0.004560123	0.006103339	0.002540582	0.022872965	0.001310353	30	0.01
1282	0.042903114	0.026805658	0.015796395	0.014587309	0.008104797	57	4.64E-03
2562	0.54477087	0.306212888	0.148768884	0.11168129	0.06994521	113	1.19E-03
5122	7.363915534	3.857077241	1.620106829	1.172035066	0.710841484	216	3.04E-04
10242	43.846706503	23.222138298	12.034713165	9.838531832	4.393086947	448	7.67E-05

Table 7.3: Αποτελέσματα χρόνου εκτέλεσης της Additive Schwarz μεθόδου με 1, 2, 4, 5, και 8 χωρία.

Συγκρίνοντας τα αποτελέσματα των μεθόδων Jacobi και Jacobi-Gauss Seidel για σειριακή και παράλληλη εκτέλεση, όπως φαίνεται στην εικόνα (7.2) και στους πίνακες παραπάνω, η Jacobi-Gauss Seidel συγκλίνει γρηγορότερα σε μια προσέγγιση της πραγματικής λύσης του προβλήματος (7.1), παρουσιάζοντας το ίδιο σφάλμα με την Jacobi για κάθε τάξη N του πίνακα συντελεστών A. Ως σφάλμα αξιολόγησης των μεθόδων, δηλαδή της προσεγγιστικής λύσης από την πραγματική, χρησιμοποιείται το

$$E = \left(\sum_{i=1}^{J} h|U_i - u(x_i)|^2 \right)^{\frac{1}{2}},$$

το οποίο είναι διαφορετικό από το σφάλμα $|U^{(k+1)} - U^{(k)}|$ που χρησιμοποιείται για τον έλεγχο υπέρβασης της τιμής ανοχής, και τον τερματισμό των επαναλήψεων των μεθόδων.

Βάση του θεωρήματος (5.1.2), για αρκετά μεγάλη τάξη του γραμμικού συστήματος $AU = F$ (η μεγάλη τάξη ενός γραμμικού συστήματος ισούται με μικρό βήμα διαμέρισης h), το σφάλμα προσέγγισης υποτετραπλασιάζεται, άρα η

προσεγγιστική λύση συγκλίνει στην πραγματική λύση της Μ.Δ.Ε u καθώς η τάξη του γραμμικού συστήματος αυξάνεται.

Τα αποτελέσματα είναι αναμενόμενα καθώς η Jacobi-Gauss Seidel χρησιμοποιεί τις προσεγγιστικές λύσεις $U^{(k+1)}$ της τρέχων επανάληψης, και έτσι, συνδυάζοντας την ιδιότητα αυτή με την παράλληλη υλοποίηση της, υπερισχύει σε όλες τις περιπτώσεις.

Τα αποτελέσματα εκτέλεσης της μεθόδου Additive Schwarz, για κάθε τάξη N του πίνακα συντελεστών A, και για κάθε πλήθος χωρίων Ω_i, είναι πολύ υψηλότερα από τα αντίστοιχα αποτελέσματα των μεθόδων Jacobi και Jacobi-Gauss Seidel. Ως αποτέλεσμα, η σύγκριση τους με οποιαδήποτε σειρά είναι μάταια. Ωστόσο, η αποδοτικότητα, η επιτάχυνση, και η κλιμάκωση των τριών μεθόδων, είναι συγκρίσιμα μετρικά επειδή περιγράφουν την γενική συμπεριφορά των μεθόδων και όχι τα αποτελέσματα τους βάση ορισμένων παραμέτρων, όπως η τάξη του γραμμικού συστήματος και το πλήθος των νημάτων. Παρακάτω, όταν συγκρίνουμε την επιτάχυνση, την αποδοτικότητα και την κλιμάκωση της μεθόδου Additive Schwarz, η σύγκριση θα γίνεται για τα N που είναι κοινά για τις τρεις μεθόδους. Η αργή σύγκλιση της μεθόδου, οφείλεται κατά κύριο λόγο στην εναλλαγή των συνοριακών τιμών μεταξύ των χωρίων, κατά την οποία εναλλάσσονται οι πραγματικές συνοριακές τιμές με προσεγγιστικές τιμές. Ως αποτέλεσμα, οι επαναλήψεις της μεθόδου Gauss Seidel, που εκτελείται για κάθε χωρίο $\Omega_i, i = 1, ...,$ δεν μειώνονται. Θεωρητικά, η μείωση των επαναλήψεων είναι επιθυμητό αποτέλεσμα, διότι σε κάθε χωρίο λύνουμε ένα μικρότερο σε τάξη N γραμμικό σύστημα. Από το πρόβλημα (7.1) γνωρίζουμε πως οι συνοριακές συνθήκες στο χωρίο $\Omega = [a, b]$, είναι οι $u(a) = u(b) = 0$, και άρα θα ισχύει $U(a) = U(b) = 0$, όπου U είναι το διάνυσμα της προσεγγιστικής λύσης. Αν χωρίσουμε το αρχικό χωρίο Ω σε δύο μη επικαλυπτόμενα χωρία Ω_1 και Ω_2, τότε οι συνοριακές συνθήκες του Ω_1, με βάση το διάνυσμα U, θα είναι οι $U(\Omega_{1a}) = u(a) = 0$ και $U(\Omega_{1b}) = U(\Omega_{2a})$, όπου $U(\Omega_{2a})$ είναι η προσωρινή προσεγγιστική λύση στο σημείο Ω_{2a}. Αντίστοιχα για το χωρίο Ω_2 θα ισχύει,

$U(\Omega_{2a}) = U(\Omega_{1b})$, όπου $U(\Omega_{1b})$ είναι η προσωρινή προσεγγιστική λύση στο σημείο Ω_{1b} και $U(\Omega_{2b}) = u(b) = 0$.

Επίσης, για ορισμένες τάξεις N του πίνακα συντελεστών A, το σφάλμα έχει αυξηθεί ελάχιστα συγκριτικά με τις μεθόδους Jacobi και Jacobi-Gauss Seidel, αλλά ταυτόχρονα έχουν μειωθεί πολύ οι επαναλήψεις. Οι επαναλήψεις του πίνακα (7.3), δεν είναι αυτές των μεθόδων Gauss Seidel που εκτελούνται για κάθε χωρίο, αλλά αυτές του εξωτερικού βρόχου for του αλγόριθμου (4). Το ελάχιστα αυξημένο σφάλμα είναι λογικό φαινόμενο της μεθόδου, καθώς αντί της πραγματικής λύσης των υποσυστημάτων, παίρνουμε μια προσέγγιση αυτών.

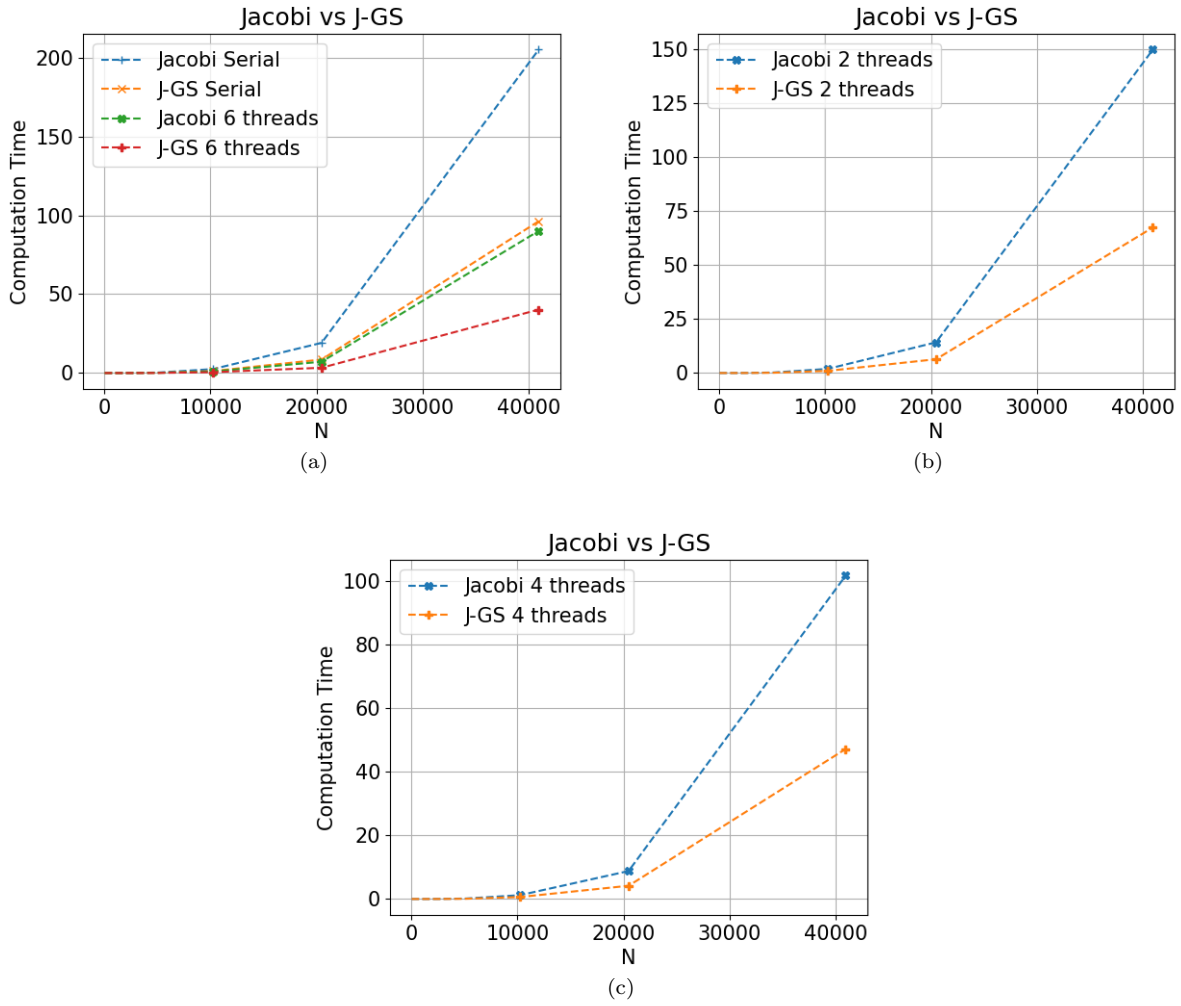

Figure 7.2: Σύγκριση αποτελεσμάτων εκτέλεσης των μεθόδων Jacobi και Jacobi-Gauss Seidel.

Στην εικόνα (7.3), φαίνεται η επιτάχυνση (δες ενότητα 2.2) των τριών μεθόδων. Για την Jacobi και Jacobi-Gauss Seidel, σύμφωνα με τον αριθμό νημάτων που χρησιμοποιήθηκαν, η επιτάχυνση δεν είναι η μέγιστη δυνατή. Για παράδειγμα, όταν χρησιμοποιούνται nt νήματα για την εκτέλεση μιας μεθόδου, θα περιμέναμε η επιτάχυνση να είναι nt φορές γρηγορότερη. Προφανώς αυτό δεν συμβαίνει για τις μεθόδους Jacobi και Jacobi-Gauss Seidel. Το ίδιο δεν ισχύει για την Additive Schwarz μέθοδο, καθώς για αρκετά μεγάλη τάξη του συστήματος, μας δίνει την μέγιστη δυνατή επιτάχυνση, με εξαίρεση την εκτέλεση της με οκτώ νήματα όπου η επιτάχυνση ξεπερνάει τη μέγιστη δυνατή, και γίνε-

ται 10 φορές γρηγορότερη. Βέβαια, υπάρχουν περιπτώσεις όπου η επιτάχυνση μειώνεται, όπως στην περίπτωση των τεσσάρων και πέντε νημάτων για τάξη συστήματος $N = 10242$. Παρ' όλα αυτά όμως, αυτή η μείωση είναι αρκετά μικρή, και έτσι συνεχίζει και βρίσκεται κοντά στην μέγιστη δυνατή επιτάχυνση.

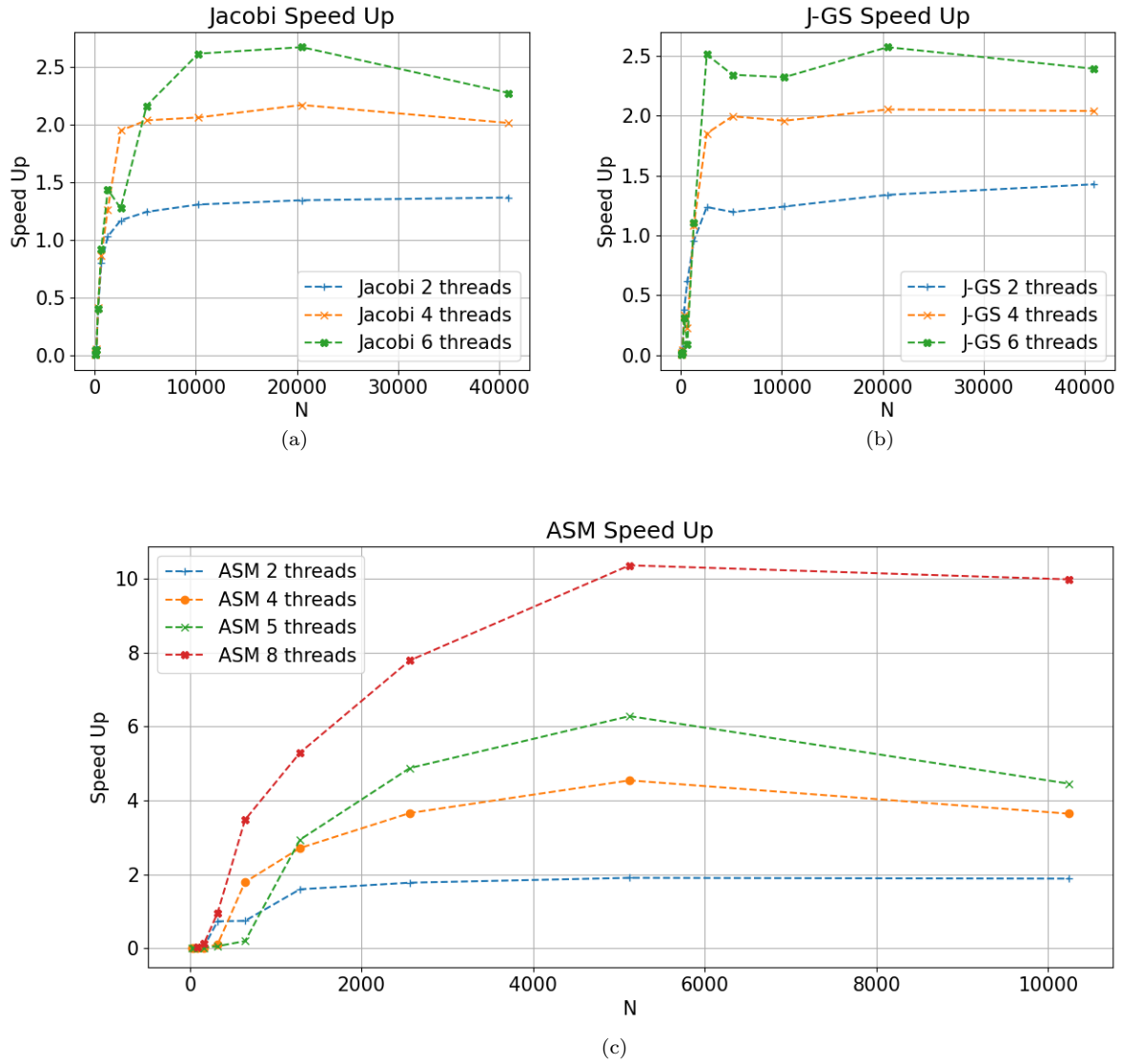

Figure 7.3: Επιτάχυνση των μεθόδων Jacobi, Jacobi-Gauss Seidel, και Additive Schwarz.

Ακολούθως, η αποδοτικότητα (2.2) των τριών μεθόδων, που βασίζεται στην επιτεύξιμη επιτάχυνση, φαίνεται στην εικόνα (7.4). Όπως αναφέρθηκε παραπάνω, η επιτάχυνση δεν είναι η μέγιστη δυνατή για τις μεθόδους Jacobi

και Jacobi-Gauss Seidel, και αυτό αποδεικνύεται από τα αντίστοιχα γραφήματα αποδοτικότητας. Στην εικόνα (7.4) φαίνεται πως η αποδοτικότητα των μεθόδων Jacobi και Jacobi-Gauss Seidel μειώνεται καθώς αυξάνουμε τα νήματα και η τάξη του συστήματος παραμένει σταθερή. Έτσι, αυτές οι δύο μέθοδοι δεν είναι ισχυρά κλιμακούμενες (2.2). Η αποδοτικότητα της Additive Schwarz μεθόδου είναι δυσκολότερο να ερμηνευτεί, διότι εξαιτίας της αργής σύγκλισης της μεθόδου και του αυξημένου χρόνου εκτέλεσης της, ο πειραματισμός με μεγαλύτερα N δεν ήταν δυνατός. Ωστόσο, σε ορισμένες περιπτώσεις η Additive Schwarz μέθοδος ξεπερνάει τη μέγιστη δυνατή επιτάχυνση, επομένως για αυτές τις περιπτώσεις είναι πολύ αποδοτική και επιτυγχάνει ισχυρή κλιμάκωση. Η μοναδική μείωση της αποδοτικότητας για αρκετά μεγάλη τάξη N, συμβαίνει όταν $N = 10242$. Όπως και με την επιτάχυνση, η μείωση της αποδοτικότητας είναι μικρή και έτσι δεν υπάρχει μεγάλη απώλεια στην επίδοση της μεθόδου.

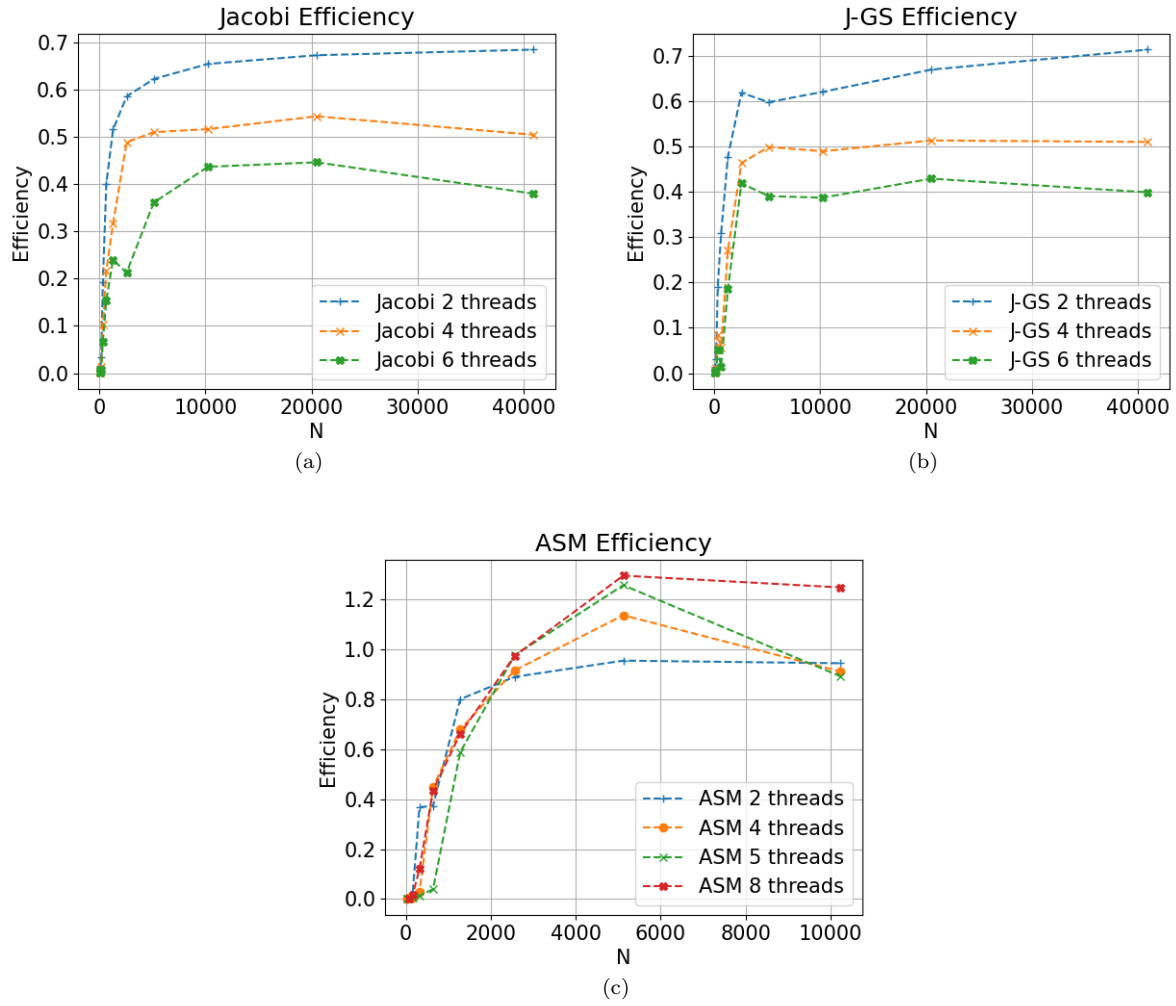

Figure 7.4: Απόδοση των μεθόδων Jacobi, Jacobi-Gauss Seidel, και Additive Schwarz.

Βάση των αποτελεσμάτων εκτέλεσης των προαναφερθέντων μεθόδων σε αρχιτεκτονική παράλληλης κοινόχρηστης μνήμης, η μέθοδος Jacobi-Gauss Seidel έχει την καλύτερη επίδοση όσον αφορά τον χρόνο σύγκλισης της προσεγγιστικής λύσης του προβλήματος (7.1) στην πραγματική. Αν και η Jacobi-Gauss Seidel υπερισχύει των άλλων μεθόδων στον χρόνο εκτέλεσης της, η Additive Schwarz παρήγαγε ενδιαφέροντα αποτελέσματα στα οποία είναι προφανές πως παρ' όλο της χαμηλής επίδοσης της, είναι γενικά πιο αποδοτική, και η επιτάχυνση της αυξάνεται καθώς αυξάνονται τα νήματα. Επιπλέον, η μέθοδος Additive Schwarz έχει την δυνατότητα να επεκταθεί και σε υβριδικές αρχιτεκ-

τονικές, ώστε το κάθε χωρίο $\Omega_i, i = 1, ...$ να εκτελείται σε ξεχωριστό υπολογιστή χρησιμοποιώντας την Jacobi-Gauss Seidel για την επίλυση του. Αυτό είναι εφικτό, διότι για μια δεδομένη τάξη $N, N > 320$ του πίνακα συντελεστών A, οι επαναλήψεις είναι πολύ λιγότερες από αυτές της μεθόδου Jacobi-Gauss Seidel, και έτσι το πλήθος επικοινωνιών μεταξύ των υπολογιστών θα είναι μικρότερο, που συμβάλει στην γρηγορότερη εκτέλεση της μεθόδου. Γενικότερα, η παρούσα υλοποίηση της Jacobi-Gauss Seidel περιορίζεται σε αρχιτεκτονικές διαμοιραζόμενης μνήμης, ενώ η υλοποίηση της Additive Schwarz έχει περισσότερες προοπτικές.

Επίλογος

Συμπερασματικά, η παρούσα εργασία διερεύνησε την αριθμητική επίλυση διαφορικών εξισώσεων χρησιμοποιώντας την μέθοδο πεπερασμένων διαφορών, εστιάζοντας συγκεκριμένα στις σειριακές και παράλληλες υλοποιήσεις των μεθόδων Jacobi, Gauss Seidel και Additive Schwarz. Η παραλληλοποίηση επιτεύχθηκε χρησιμοποιώντας το OpenMP, ένα ευρέως χρησιμοποιούμενο πρότυπο για παράλληλους υπολογιστές κοινόχρηστης μνήμης.

Μέσα από μια ολοκληρωμένη ανάλυση και σύγκριση αυτών των μεθόδων, προέκυψαν αρκετά βασικά ευρήματα. Πρώτον, παρατηρήθηκε ότι και οι δύο μέθοδοι Jacobi και Gauss Seidel είναι επαναληπτικές τεχνικές που συγκλίνουν σε μια προσεγγιστική λύση της πραγματικής λύσης μέσω διαδοχικών επαναλήψεων. Ωστόσο, η μέθοδος Gauss Seidel παρουσιάζει έναν ανώτερο ρυθμό σύγκλισης λόγω της χρήσης προσεγγιστικών τιμών κατά την ίδια επανάληψη, καθιστώντας την μια ευνοϊκή επιλογή για την επίλυση διαφορικών εξισώσεων.

Επιπλέον, οι παράλληλες υλοποιήσεις των μεθόδων Jacobi και Jacobi-Gauss Seidel με χρήση OpenMP επέδειξαν σημαντικές βελτιώσεις στην απόδοση. Με την κατανομή του φόρτου εργασίας σε πολλαπλά νήματα, ο υπολογιστικός χρόνος μειώθηκε αποτελεσματικά, επιτρέποντας ταχύτερη σύγκλιση και πιο αποτελεσματική δημιουργία λύσεων. Αυτό υπογραμμίζει το πλεονέκτημα της χρήσης τεχνικών παράλληλων υπολογιστών κατά την αντιμετώπιση προβλη-

μάτων μεγάλης κλίμακας, καθώς παρέχουν ένα μέσο για την αξιοποίηση της ισχύος των σύγχρονων επεξεργαστών πολλαπλών πυρήνων.

Η μέθοδος Additive Schwarz, από την άλλη πλευρά, εισήγαγε μια προσέγγιση διαμέρισης χωρίου που υποδιαίρεσε το πρόβλημα σε μικρότερα χωρία, τα οποία επιλύθηκαν ανεξάρτητα πριν συνδυαστούν. Αυτή η μέθοδος επέδειξε εξαιρετική επιτάχυνση και αποδοτικότητα, επιτρέποντας την αποτελεσματική παραλληλοποίηση σε πολλά νήματα.

Συνολικά, αυτή η έρευνα συμβάλλει στην κατανόηση αριθμητικών μεθόδων για την επίλυση διαφορικών εξισώσεων, ιδιαίτερα στο πλαίσιο των παράλληλων υπολογιστικών περιβαλλόντων.

Βιβλιογραφία

[1] Alfio Quarteroni and Paola Gervasio. *A Primer on Mathematical Modelling*, volume 121. Springer Nature, 2020.

[2] Martin J Gander et al. Schwarz methods over the course of time. *Electron. Trans. Numer. Anal*, 31(5):228–255, 2008.

[3] Ιωάννης Θ. Φαμέλης. *ΥΠΟΛΟΓΙΣΤΙΚΑ ΜΑΘΗΜΑΤΙΚΑ Αριθμητικές μέθοδοι και μέθοδοι βελτιστοποίησης με υλοποίηση σε MATLAB(Ocatve) και Python*. ΚΡΙΤΙΚΗ, 2021.

[4] ΔΟΥΓΑΛΗΣ ΒΑΣΙΛΕΙΟΣ ΑΚΡΙΒΗΣ ΓΕΩΡΓΙΟΣ. *Αριθμητικές μέθοδοι για συνήθεις διαφορικές εξισώσεις*. ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΕΚΔΟΣΕΙΣ ΚΡΗΤΗΣ, 2η edition, 2013.

[5] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[6] Guillaume Houzeaux Frédéric Magoules, François-Xavier Roux. *Parallel Scientific Computing*. Wiley-ISTE, 2016.

[7] Rohit Chandra, Leo Dagum, Ramesh Menon, David Kohr, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[8] Αθανάσιος Ι. Μάργαρης. *MPI Θεωρία Και Εφαρμογές*. ΕΚΔΟΣΕΙΣ ΤΖΙΟΛΑ, 2019.

[9] Uri M. Asher Chen Greif. *Εισαγωγή στις αριθμητικές μεθόδους*. ΚΛΕΙΔΑΡΙΘΜΟΣ, 2022.

[10] Yueqiang Shang. A distributed memory parallel gauss–seidel algorithm for linear algebraic systems. *Computers & Mathematics with Applications*, 57(8):1369–1376, 2009.

[11] Μιχαήλ Ν. Βραχάτης. *Αριθμητική Ανάλυση - Εισαγωγή*. ΚΛΕΙΔΑΡΙΘΜΟΣ, 2011.

[12] Παναγιώτης Χατζηπαντελίδης. Πεπερασμένες Διαφορές. http://users.math.uoc.gr/~p.chatzipa/index_files/mem253/2014b/book1.pdf.

[13] Frédéric Nataf Victorita Dolean, Pierre Jolivet. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. SIAM, 2015.

[14] Pierre-Louis Lions. On the schwarz alternating method. iii: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, 1990.