Παράλληλα Αριθμητικά Σχήματα για Υδροδυναμικές Ροές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ Μ.Δ.Ε.

Ευτύχιος Π. Πετράκης

Επιβλέπων:
Εμμανουήλ Ν. Μαθιουδάκης

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ - ΓΕΝΙΚΟ ΤΜΗΜΑ

Χανιά, 17 Μαΐου 2006.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους σύμβουλούς μου καθηγητές, το Λέκτορα Μαθιουδάκη Εμμανουήλ και το Λέκτορα Δελή Ανάργυρο, οι οποίοι μου παρείχαν την άρτια επιστημονική καθοδήγηση για την ολοκλήρωση της παρούσας διατριβής.

Τους καθηγητές του Γενικού Τμήματος και ιδιαίτερα την Αναπληρώτρια Καθηγήτρια Παπαδοπούλου Έλενα για τη συμμετοχή της στην τριμελή επιτροπή.

Τον διευθυντή του Εργαστηρίου Εφαρμοσμένων Μαθηματικών και Ηλεκτρονικών Υπολογιστών Καθηγητή Σαριδάκη Ιωάννη για την παραχώριση της υλικοτεχνικής υποδομής του εργαστηρίου.

Τέλος ευχαριστώ τους συναδέλφους μου μεταπτυχιακούς φοιτητές και υποψήφιους διδάκτορες για την ηθική συμπαράσταση που μου παρείχαν.

Περίληψη

Στην παρούσα εργασία αναπτύσσονται παράλληλοι αλγόριθμοι για την επίλυση προβλημάτων εξισώσεων ρηχών υδάτων (SW) στις δύο διαστάσεις. Τα αριθμητικά σχήματα που χρησιμοποιούνται είναι το Roe TVD και το MacCormack TVD. Οι τεχνικές παραλληλοποίησης που εφαρμόζονται βασίζονται στα πρότυπα OpenMP και MPI για αρχιτεκτονικές κοινής και μη-κοινής μνήμης αντίστοιχα. Σκοπός είναι η ανάπτυξη επιλυτών προβλημάτων εξισώσεων SW στις δύο διαστάσεις, οι οποίοι θα είναι σε θέση να χειριστούν *μεγάλα*, σε μέγεθος, προβλήματα σε ρεαλιστικό χρόνο. Για να γίνει μελέτη της απόδοσης των παράλληλων αλγορίθμων δοκιμάζονται οκτώ διαφορετικά προβλήματα δοκιμής σε δύο παράλληλα υπολογιστικά συστήματα, ένα ενιαίο κατανεμημένης-κοινής μνήμης και ένα δικτυακό σύστημα.

Κεφάλαιο 1

Εισαγωγή

Η μελέτη της ροής υδάτων μέσω της μοντελοποίησής τους από τις εξισώσεις ρηχών υδάτων (SW) έχει προσελκύσει τα τελευταία χρόνια το ερευνητικό ενδιαφέρον της επιστημονικής κοινότητας στον τομέα των Εφαρμοσμένων Μαθηματικών αλλά και της Μηχανικής και του Περιβάλλοντος. Μία σημαντική κατηγορία προβλημάτων που παρουσιάζουν πρακτικό ενδιαφέρον αποτελούν οι παλιρροιακές ροές στις εκβολές και τις παράκτιες περιοχές, η πλημμυρίδα στους ποταμούς, η κατάρρευση φραγμάτων κτλ. Απώτερος σκοπός της παρούσας εργασίας είναι η αριθμητική επίλυση του συστήματος εξισώσεων SW στις δύο διαστάσεις

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{w})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{w})}{\partial y} = \mathbf{R}, \tag{1.0.1}$$

όπου

$$\mathbf{w} = \begin{bmatrix} h \\ uh \\ vh \end{bmatrix}, \quad \mathbf{F}(\mathbf{w}) = \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \quad \mathbf{G}(\mathbf{w}) = \begin{bmatrix} vh \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}$$

και

$$\mathbf{R} = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix},$$

για δεδομένη χρονική στιγμή t, με $h(x, y, t)$ να συμβολίζει το ύψος του νερού πάνω από τον πυθμένα του καναλιού, $B(x, y)$ να είναι το ύψος του πυθμένα, g η επιτάχυνση της βαρύτητας και $u(x, y, t)$ και $v(x, y, t)$ να αντιπροσωπεύουν τις ταχύτητες ως προς την x και y κατεύθυνση αντίστοιχα. Η επίλυση του συστήματος αυτού βασίζεται στη χρήση κατάλληλων αριθμητικών σχημάτων πεπερασμένων όγκων. Για το λόγο αυτό χρησιμοποιήθηκαν δύο από τα πλέον δημοφιλή αριθμητικά σχήματα επίλυσης συστημάτων εξισώσεων SW αυτά του Roe και του MacCormack.

Η αριθμητική μοντελοποίηση προβλημάτων SW εδώ και καιρό στηρίζεται στη χρήση σειριακών υπολογιστών. Τα τελευταία χρόνια η επιστημονική κοινότητα

μετατοπίζεται στη χρήση παράλληλων συστημάτων. Η κίνηση αυτή έχει να κάνει με την ανάγκη για επίλυση προβλημάτων σε πραγματικές διαστάσεις χωρίς να απαιτείται απαγορευτικά μεγάλος χρόνος.

Για να επιτευχθεί ένα αποδεκτό επίπεδο ρεαλιστικότητας, οι μεγάλες εξομοιώσεις απαιτούν μεγάλο υπολογιστικό χρόνο και τεράστια αποθέματα μνήμης. Αυτό συμβαίνει γιατί, για να προσομειωθεί καλλίτερα η φυσική της ροής, το πρόβλημα θα πρέπει να διακριτοποιηθεί σε ένα αρκετά εκλεπτυνσμένο υπολογιστικό πλέγμα (grid). Για το λόγο αυτό υπάρχει μία σαφής τάση για την κατασκευή παράλληλων αλγορίθμων. Οι υπολογιστικοί πόροι που υπάρχουν διαθέσιμοι, όσο αφορά τους παράλληλους υπολογιστές, είναι ελκυστικοί για τον προγραμματιστή, και αν χρησιμοποιηθούν σωστά μπορούν να αποδώσουν σε υψηλό βαθμό και να ξεπεραστούν οι περιορισμοί των σειριακών υπολογιστικών συστημάτων. Επιπλέον η εξέλιξη της τεχνολογίας επιτρέπει σήμερα την απόκτηση παράλληλων υπολογιστικών συστημάτων με μικρό οικονομικό κόστος.

Τα πλεονεκτήματα που προκύπτουν από την κατασκευή και την υλοποίηση ενός παράλληλου αλγορίθμου είναι πολλά. Η σωστή υλοποιήση του αλγορίθμου σε παράλληλες υπολογιστικές αρχιτεκτονικές με βάση επιλεγμένων προτύπων έχει ως βασικό πλεονέκτημα την ραγδαία μείωση του χρόνου εκτέλεσης, ανάλογα με των αριθμό των επεξεργαστών που χρησιμοποιούνται.

Η παραλληλοποίηση επιτρέπει σε ένα πρόγραμμα να εκτελεστεί από όσους επεξεργαστές υπάρχουν διαθέσιμοι ταυτόχρονα, ούτως ώστε ο χρόνος που απαιτείται για να βρεθεί η αριθμητική λύση ενός προβλήματος, στην περίπτωσή μας η προσομοίωση ροών ελεύθερης επιφάνειας, να μειωθεί δραματικά. Η μείωση αυτή έχει ως αποτέλεσμα την αύξηση της παραγωγικότητας και επιτρέπει στο χρήστη να κάνει ακόμα περισσότερες δοκιμές ούτως ώστε να βελτιώσει την φυσική του προβλήματος μέσα στον κώδικα. Οι αβεβαιότητες που είναι πιθανόν να υπάρχουν για την αριθμητική επίλυση ενός προβλήματος SW έχοντας χρησιμοποιήσει μία μικρή διακριτοποίηση του χωρίου σε ένα σειριακό σύστημα μπορούν να εξαλειφθούν αυξάνοντας το υπολογιστικό πλέγμα (grid) και εκτελώντας το σε παράλληλη μορφή χωρίς επίπτωση στο χρόνο εκτέλεσης.

Για την υλοποίηση των παράλληλων αλγορίθμων έγινε χρήση των προτύπων OpenMP και MPI, τα οποία μαζί με την High Performance Fortran, αποτελούν την πλέον διαδεδομένη λύση για την παραλληλοποίηση προγραμμάτων. Για την εκτέλεση των προγραμμάτων χρησιμοποιήθηκαν δύο παράλληλα υπολογιστικά συστήματα. Το πρώτο αποτελείται από μία συστάδα τεσσάρων Sun Fire V240 όπου το κάθε ένα περιέχει δύο επεξεργαστές UltraSPARC IIIi στα 1.5 GHz και 2GB RAM. Η συστάδα συνδέεται μέσω ενός δικτύου Ethernet στο 1Gbit. Το δεύτερο αποτελείται από ένα σύστημα SGI Origin 350 με οκτώ επεξεργαστές MIPS R16000 εφοδιασμένο με 4GB RAM. Αμφότερα υπολογιστικά συστήματα αποτελούν υποδομή του Εργαστηρίου Εφαρμοσμένων Μαθηματικών και Ηλεκτρονικών Υπολογιστών του Γενικού Τμήματος του Πολυτεχνείου Κρήτης.

Αυτή τη στιγμή η έρευνα και ανάπτυξη στο χώρο των παράλληλων διεργασιών όσο αφορά τους επιλύτες Εξισώσεων SW βρίσκονται σε πολύ αρχικό στάδιο. Πλέον

της δημοσίευσης του Rao [30] δεν υπάρχουν ανάλογες που να αναφέρονται σε άμεσες στο χρόνο (explicit) αριθμητικές μεθόδους αλλά υπάρχουν κάποιες που αφορούν έμμεσα (implicit) σχήματα και έχουν να κάνουν κατά κύριο λόγο με την παραλληλοποίηση κατά το στάδιο της επίλυσης του γραμμικού συστήματος που προκύπτει σε κάθε χρονικό βήμα. Κύριες αναφορές το άρθρο των Paglieri, Ambrosi, Formaggia, Quarteroni και Scheinine [29] και το άρθρο των Chunbo, Kai, Ning και Qinghai [5].

Συγκριτικά με την δημοσίευση του Pradada Rao [30], στην παρούσα εργασία, εκτός από το αριθμητικό σχήμα MacCormack παραλληλοποιήθηκε και το σχήμα του Roe. Επιπρόσθετα οι υλοποιήσεις πραγματοποιήθηκαν και σε OpenMP. Πλέον του προβλήματος που παρουσιάζει ο Pradada Rao στο [30] δοκιμάστηκαν και άλλα επτά κλασικά προβλήματα δοκιμής (benchmark problems).

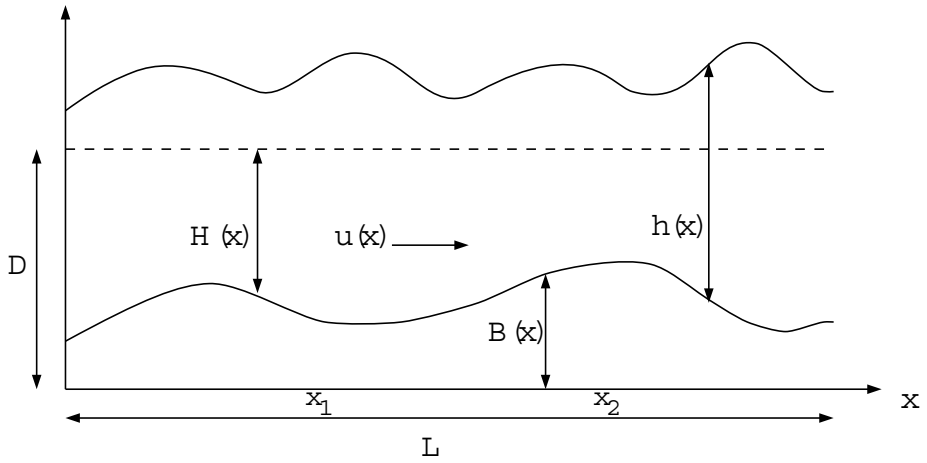
Η εργασία αυτή ακολουθεί την εξής δομή. Αρχικά παρουσιάζονται οι εξισώσεις SW στη μία και στις δύο διαστάσεις. Στο τρίτο κεφάλαιο παρατίθονται ορισμένα γνωστά σχήματα για τη μία και τις δύο διαστάσεις καθώς επίσης και ορισμένα προβλήματα στη μία διάσταση. Στο επόμενο κεφάλαιο γίνεται αναφορά στα προβλήματα δύο διαστάσεων, τα οποία θα επιχειρήσουμε να λύσουμε με παράλληλους αλγόριθμους. Εν συνεχεία περιγράφονται οι αρχιτεκτονικές των παράλληλων συστημάτων και θα κατασκευαστούν οι παράλληλοι αλγόριθμοι των μεθόδων. Τέλος παρουσιάζονται τα αποτελέσματα μετρήσεων και συγκρίσεων των υλοποιήσεων σε διάφορες παράλληλες αρχιτεκτονικές.

Κεφάλαιο 2

Εξισώσεις Ρηχών Υδάτων σε μία και δύο διαστάσεις

2.1 Νόμοι Διατήρησης

Πριν ξεκινήσουμε την εισαγωγή στις εξισώσεις SW ή αλλιώς σύστημα Saint Venant κρίνεται απαραίτητο να γίνει μία μικρή αλλά πλέον βασική αναφορά στους Νόμους Διατήρησης της Μάζας και της Ορμής. Για να γίνει αυτό θεωρούμε μία περιοχή μεταξύ του x_1 και του x_2 στο μονοδιάστατο κανάλι του Σχήματος 2.1. Στο διάστημα αυτό πραγματοποιείται ροή ύδατος βάθους $h(x,t)$ με ταχύτητα $u(x,t)$ πάνω από τοπογραφία ύψους $B(x)$, για τη δεδομένη χρονική στιγμή t.



Σχήμα 2.1: Το χωρίο των SW.

2.1.1 Νόμος Διατήρησης της Μάζας

Στο διάστημα από x_1 έως x_2 θεωρούμε ότι

> Η καθαρή ποσότητα του υγρού στην περιοχή από x_1 έως x_2 ισούται με το ρυθμό μεταβολής της ολικής ποσότητας του υγρού στην περιοχή από x_1 έως x_2.

Η ολική ποσότητα του υγρού στο διάστημα $[x_1, x_2]$ είναι

$$\int_{x_1}^{x_2} \int_{B}^{h+B} dx\, dy = \int_{x_1}^{x_2} (h + B - B)\, dx = \int_{x_1}^{x_2} h\, dx,$$

όπου παραγωγίζοντας ως προς t παίρνουμε ότι

$$\text{Ρυθμός μεταβολής ολικής μάζας υγρού στο } [x_1, x_2] = \frac{d}{dt} \int_{x_1}^{x_2} h\, dx.$$

Επιπρόσθετα έχουμε ότι η ποσότητα του υγρού η οποία εισέρχεται στο σημείο x_1 είναι $(uh)_{x_1}$ καθώς επίσης η ποσότητα η οποία εξέρχεται από το σημείο x_2 ισούται με $(uh)_{x_2}$. Γνωρίζοντας λοιπόν τα παραπάνω προκύπτει ότι

$$\text{Καθαρή ποσότητα υγρού στο } [x_1, x_2] = (uh)_{x_1} - (uh)_{x_2}.$$

Οπότε έχουμε την ολοκληρωτική μορφή της εξίσωσης της διατήρησης της μάζας

$$\frac{d}{dt} \int_{x_1}^{x_2} h\, dx + [uh]_{x_1}^{x_2} = 0. \tag{2.1.1}$$

Για να πάρουμε τη διαφορική μορφή παραγωγίζουμε την (2.1.1) ως προς το χρόνο t στο διάστημα $[t_1, t_2]$, για $t_2 > t_1$,

$$\int_{x_1}^{x_2} h(x, t_2)\, dx - \int_{x_1}^{x_2} h(x, t_1)\, dx + \int_{t_1}^{t_2} [uh]_{x_1}^{x_2}\, dt = 0.$$

Υποθέτοντας ότι οι συναρτήσεις $h(x,t)$ και $u(x,t)$ είναι παραγωγίσιμες και χρησιμοποιώντας τις σχέσεις

$$h(x, t_2) - h(x, t_1) = \int_{t_1}^{t_2} \frac{\partial h}{\partial t}\, dt \quad \text{και} \quad [uh]_{x_1}^{x_2} = \int_{x_1}^{x_2} \frac{\partial (uh)}{\partial x}\, dx,$$

παίρνουμε ότι

$$\int_{t_1}^{t_2} \int_{x_1}^{x_2} \left\{ \frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} \right\} dx\, dt = 0.$$

Εφόσον x_1, x_2 και t_1, t_2 είναι τυχαία, προκύπτει η διαφορική μορφή της εξίσωσης της διατήρησης της μάζας,

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} = 0. \tag{2.1.2}$$

2.1.2 Νόμος Διατήρησης της Ορμής

Στο διάστημα από x_1 έως x_2 θεωρούμε ότι,

> Ο συνολικός ρυθμός μεταβολής της ορμής στη x κατεύθυνση ισούται με την ασκούμενη δύναμη προς τη x κατεύθυνση, δηλαδή

$$\begin{aligned}\text{Συνολικός ρυθμός μεταβολής της ορμής στη } x \text{ κατεύθυνση} &= \frac{d}{dt}\int_{x_1}^{x_2}\int_{B}^{h+B} u\,dx\,dy + (hu^2)_{x_2} - (hu^2)_{x_1} \\ &= \frac{d}{dt}\int_{x_1}^{x_2} uh\,dx + (hu^2)_{x_2} - (hu^2)_{x_1}.\end{aligned}$$

Επιπρόσθετα,

$$\begin{aligned}\text{Δύναμη πίεσης στα άκρα} &= g\left[\int_{B}^{h+B}\big(y-(h+B)\big)\,dy\right]_{x_1}^{x_2} \\ &= g\left[\left[\frac{1}{2}y^2-(h+B)y\right]_{B}^{h+B}\right]_{x_1}^{x_2} \\ &= \left[-\frac{1}{2}gh^2\right]_{x_1}^{x_2}\end{aligned}$$

και

$$\text{Δύναμη πίεσης από τοπογραφία} = -g\int_{C_{B1,2}} h\,dy,$$

όπου το $C_{B1,2}$ δηλώνει την καμπύλη του πυθμένα (τοπογραφία) στην περιοχή μεταξύ x_1 και x_2. Θεωρώντας ότι δεν υπάρχουν ασυνέχειες στον πυθμένα

$$\text{Δύναμη πίεσης από τοπογραφία} = -g\int_{x_1}^{x_2} h\frac{dB}{dx}\,dx,$$

οπότε προκύπτει ότι

$$\text{Δύναμη ασκούμενη στη } x \text{ κατεύθυνση} = \left[-\frac{1}{2}gh^2\right]_{x_1}^{x_2} - g\int_{x_1}^{x_2} h\frac{dB}{dx}\,dx.$$

Άρα,

$$\frac{d}{dt}\int_{x_1}^{x_2} uh\,dx + (hu^2)_{x_2} - (hu^2)_{x_1} = \left[-\frac{1}{2}gh^2\right]_{x_1}^{x_2} - g\int_{x_1}^{x_2} h\frac{dB}{dx}\,dx,$$

όπου με μία αναδιάταξη των όρων προκύπτει η ολοκληρωτική μορφή της εξίσωσης για τη διατήρηση της ορμής,

$$\frac{d}{dt}\int_{x_1}^{x_2} uh\,dx + \left[hu^2+\frac{1}{2}gh^2\right]_{x_1}^{x_2} = -g\int_{x_1}^{x_2} h\frac{dB}{dx}\,dx. \tag{2.1.3}$$

Για να πάρουμε τη διαφορική μορφή της (2.1.3) χρησιμοποιούμε την ίδια λογική με αυτή που εφαρμόσαμε στο νόμο διατήρησης της μάζας και υποθέτουμε ότι $h(x,t)$ και $u(x,t)$ αποτελούν παραγωγίσιμες συναρτήσεις οπότε προκύπτει ότι

$$\frac{\partial(uh)}{\partial t} + \frac{\partial[hu^2 + \frac{1}{2}gh^2]}{\partial x} = -ghB_x. \tag{2.1.4}$$

2.2 Οι εξισώσεις SW στη μία διάσταση

Έχοντας δει παραπάνω τους νόμους διατήρησης της μάζας και της ορμής, είμαστε σε θέση να κάνουμε μία συνοπτική παρουσίαση των Εξισώσεων SW στη μία διάσταση.

Θεωρούμε την ύπαρξη νερού σε ένα κανάλι με μοναδιαίο πλάτος και υποθέτουμε ότι η κατακόρυφη ταχύτητα είναι αμελητέα και η οριζόντια $u(x,t)$ σταθερή σε κάθε σημείο του καναλιού. Αυτό αληθεύει στην περίπτωση μικρού μεγέθους κυμάτων σε ένα υγρό το οποίο είναι ρηχό σε σχέση με το μήκος κύματος.

Τώρα υποθέτουμε ότι το υγρό είναι ασυμπίεστο, ούτως ώστε η πυκνότητά του ρ να παραμένει σταθερή. Αντίθετα επιτρέπουμε στο βάθος του υγρού να ποικίλλει, το οποίο και ονομάζουμε ύψος $h(x,t)$.

Το παραπάνω υγρό ικανοποιεί το νόμο διατήρησης της μάζας που αναλύσαμε στην προηγούμενη ενότητα στη γνωστή του μορφή (2.1.2) τον οποίο θα διατυπώσουμε χρησιμοποιώντας και έναν πιο πρακτικό συμβολισμό

$$h_t + (uh)_x = 0. \tag{2.2.1}$$

Επίσης ακολουθεί το νόμο διατήρησης της ορμής όπως τον είδαμε στην εξίσωση (2.1.4), οπότε ξαναγράφοντάς τον έχουμε

$$(uh)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0. \tag{2.2.2}$$

Στο στάδιο αυτό έχουμε τη δυνατότητα να συνδιάσουμε τις εξισώσεις (2.2.1), (2.2.2) οι οποίες αναφέρθηκαν παραπάνω σε ένα σύστημα *μονοδιάστατων εξισώσεων SW*.

$$\begin{bmatrix} h \\ uh \end{bmatrix}_t + \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}_x = 0, \tag{2.2.3}$$

είτε πιο απλά

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{w})}{\partial x} = 0, \tag{2.2.4}$$

όπου,

$$\mathbf{w}(x,t) = \begin{bmatrix} h \\ uh \end{bmatrix},\ \mathbf{F}(\mathbf{w}) = \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}.$$

Το $\mathbf{F}(\mathbf{w})$ ονομάζεται *συνάρτηση ροής* (*flux function*).

Έως τώρα έχουμε κάνει αναφορά στις εξισώσεις SW για τις περιπτώσεις κατά τις οποίες ο πυθμένας του καναλιού δεν έχει κάποια τοπογραφία αλλά είναι επίπεδος. Οι αλλαγές που προκύπτουν στις εξισώσεις σε περιπτώσεις όπου η τοπογραφία είναι υπαρκτή έχουν να κάνουν με την προσθήκη του λεγόμενου *πηγαίου όρου τοπογραφίας (source term topography)* στο δεξί μέλος του συστήματος των εξισώσεων. Η (2.2.4) λοιπόν γίνεται

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{w})}{\partial x} = \mathbf{R}, \tag{2.2.5}$$

όπου

$$\mathbf{R} = \begin{bmatrix} 0 \\ -ghB_x \end{bmatrix}.$$

Ο όρος $\mathbf{B}$ αποτελεί συνάρτηση του x και περιγράφει την εκάστοτε τοπογραφία. Θα μπορούσαν να περιληφθούν και άλλοι όροι όπως π.χ. τριβή, δύναμη Coriolis, κλπ. αλλά δεν είναι στο σκοπό της παρούσας διατριβής.

Μερικά από τα αριθμητικά σχήματα που θα αναφέρουμε αργότερα απαιτούν τον υπολογισμό του Ιακωβιανού πίνακα του $\mathbf{F}(\mathbf{w})$ ο οποίος είναι

$$\frac{\partial \mathbf{F}}{\partial \mathbf{w}} = \mathbf{A}(\mathbf{w}) = \begin{bmatrix} 0 & 1 \\ gh - u^2 & 2u \end{bmatrix},$$

που έχει ιδιοτιμές τις

$$\lambda_1 = u - \sqrt{gh} \ \text{ και } \ \lambda_2 = u + \sqrt{gh}.$$

Τα αντίστοιχα ιδιοδιανύσματα είναι

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ u - \sqrt{gh} \end{bmatrix} \ \text{ και } \ \mathbf{e}_2 = \begin{bmatrix} 1 \\ u + \sqrt{gh} \end{bmatrix}.$$

Για να περιγράψουμε τη ροή χρησιμοποιούμε τον αριθμό Froude (F). Ο αριθμός F είναι ένας αδιάστατος αριθμός και ισούται με το λόγο της ταχύτητας ροής u ως προς την ταχύτητα των κυμάτων SW $c = \sqrt{gh}$ δηλαδή έχουμε ότι

$$F = \frac{u}{\sqrt{gh}}.$$

Ανάλογα με την τιμή του διακρίνονται τρεις περιπτώσεις

- Αν $F > 1$ η ροή ονομάζεται υπερκρίσιμη (supercritical), δυνάμεις αδρανείας υπερέχουν των δυνάμεων της βαρύτητας.
- Αν $F < 1$ υποκρίσιμη (subritical).
- Αν συνδιάζει και τα δύο μετακρίσιμη (transcritical).

Ορισμός 2.2.1 *Θεωρούμε το γραμμικό σύστημα*

$$u_t + Au_x = 0$$
$$u(x,0) = u_0(x)$$

όπου $u : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^m$ *και* $A \in \mathbb{R}^{m\times m}$ *είναι ένας σταθερός πίνακας. Το παραπάνω αποτελεί ένα σύστημα νόμων διατήρησης με συνάρτηση ροής (flux function)* $f(u) = Au$*. Το σύστημα αυτό ονομάζεται υπερβολικό αν ο* A *είναι διαγωνοποιήσιμος με πραγματικές ιδιοτιμές, οπότε γράφεται*

$$A = R\Lambda R^{-1},$$

όπου $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ *είναι ο διαγώνιος πίνακας των ιδιοτιμών και* $R = [r_1|r_2|\cdots|r_m]$ *είναι ο πίνακας των αντίστοιχων ισιοδιανυσμάτων. Το σύστημα ονομάζεται αυστηρά υπερβολικό όταν οι ιδιοτιμές είναι διακεκριμένες.*

Παρατίθεται το ακόλουθο θεώρημα (η απόδειξη του οποίου παραλείπεται και βασίζεται στην κλασική θεωρία των υπερβολικών εξισώσεων [7] και σε απλούς αλγεβρικούς υπολογισμούς).

Θεώρημα 2.2.1 *Το σύστημα SW είναι αυστηρά υπερβολικό για* $h > 0$*. Η μαθηματική εντροπία, η οποία είναι φυσική ενέργεια, ορίζεται*

$$E(h,u,B) = \frac{h}{2}u^2 + \frac{g}{2}h^2 + ghB \tag{2.2.6}$$

και ικανοποιεί την εντροπική συνθήκη

$$\frac{\partial E}{\partial t} + \mathrm{div}\big[u(E + \frac{g}{2}h^2)\big] \leq 0. \tag{2.2.7}$$

2.3 Οι εξισώσεις SW στις δύο διαστάσεις

Στην προηγούμενη ενότητα επιχειρήσαμε μία συνοπτική αναφορά στις εξισώσεις SW στη μία διάσταση. Η επέκτασή του στις δύο διαστάσεις θα αποτελέσει το θέμα ανάπτυξης σε αυτήν την ενότητα.

Στις δύο διαστάσεις, επιπρόσθετα από ότι συμβαίνει στη μία, υπάρχει ροή του υγρού και ως προς το πλάτος. Οι εξισώσεις που παράγονται έχοντας κατά νου το παραπάνω αποκτούν τη μορφή

$$\begin{bmatrix} h \\ uh \\ vh \end{bmatrix}_t + \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} vh \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix}, \tag{2.3.1}$$

είτε πιο απλά

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{w})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{w})}{\partial y} = \mathbf{R}, \tag{2.3.2}$$

όπου

$$\mathbf{w} = \begin{bmatrix} h \\ uh \\ vh \end{bmatrix}, \quad \mathbf{F}(\mathbf{w}) = \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \quad \mathbf{G}(\mathbf{w}) = \begin{bmatrix} vh \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}$$

και

$$\mathbf{R} = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix}.$$

με $h(x, y, t)$ να συμβολίζει το ύψος του νερού πάνω από τον πυθμένα του καναλιού, $B(x, y)$ να είναι το ύψος του πυθμένα και $u(x, y, t)$ και $v(x, y, t)$ να αντιπροσωπεύουν τις ταχύτητες ως προς την x και y κατεύθυνση αντίστοιχα. Τα $\mathbf{F}(\mathbf{w})$ και $\mathbf{G}(\mathbf{w})$ αποτελούν τις *συναρτήσεις ροής (flux functions)* ως προς την x και y κατεύθυνση αντίστοιχα. Το $\mathbf{R}$ αποτελεί τον πηγαίο όρο και μπορεί να γραφτεί και ως

$$\mathbf{R} = \mathbf{f} + \mathbf{g}, \tag{2.3.3}$$

όπου τα $\mathbf{f}$ και $\mathbf{g}$ περιέχουν τους παράγωγους όρους του πηγαίου όρου ως προς x και y αντίστοιχα.

Σε αντιστοιχία με τη μία διάσταση θα υπολογίσουμε τους Ιακωβιανούς πίνακες των συναρτήσεων-ροής μιας και θα μας χρειαστούν στα αριθμητικά σχήματα που θα παρουσιάσουμε παρακάτω.

$$\mathbf{A}(\mathbf{w}) = \frac{\partial \mathbf{F}}{\partial \mathbf{w}} = \begin{bmatrix} 0 & 1 & 0 \\ c^2 - u^2 & 2u & 0 \\ -uv & v & u \end{bmatrix} \quad \text{και} \quad \mathbf{B}(\mathbf{w}) = \frac{\partial \mathbf{G}}{\partial \mathbf{w}} = \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ c^2 - v^2 & 0 & 2v \end{bmatrix},$$

όπου $c = \sqrt{gh}$ όπως ορίσαμε πριν. Οι ιδιοτιμές του πίνακα $\mathbf{A}$ είναι

$$\lambda_1^F = u - c, \quad \lambda_2^F = u \quad \text{και} \quad \lambda_3^F = u + c,$$

και για τον πίνακα $\mathbf{B}$ είναι

$$\lambda_1^G = v - c, \quad \lambda_2^G = v \quad \text{και} \quad \lambda_3^G = v + c.$$

Τα αντίστοιχα ιδιοδιανύσματα για τον $\mathbf{A}$ είναι

$$\mathbf{e}_1^F = \begin{bmatrix} 1 \\ u - c \\ v \end{bmatrix}, \quad \mathbf{e}_2^F = \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix} \quad \text{και} \quad \mathbf{e}_3^F = \begin{bmatrix} 1 \\ u + c \\ v \end{bmatrix},$$

καθώς επίσης και για τον $\mathbf{B}$

$$\mathbf{e}_1^G = \begin{bmatrix} 1 \\ u \\ v - c \end{bmatrix}, \quad \mathbf{e}_2^G = \begin{bmatrix} 0 \\ -c \\ 0 \end{bmatrix} \quad \text{και} \quad \mathbf{e}_3^G = \begin{bmatrix} 1 \\ u \\ v + c \end{bmatrix}.$$

Η επίλυση των παραπάνω εξισώσεων παρουσιάζει δυσκολίες οι οποίες κατά κύριο λόγο οφείλονται

- στην εμφάνιση ασυνεχειών·
- στη μη γραμμικότητα του συστήματος·
- και στην ύπαρξη πηγαίων όρων σε ορισμένα από τα προβλήματα.

Κεφάλαιο 3

Αριθμητικά Σχήματα

Στην παρούσα ενότητα θα γίνει παρουσίαση των αριθμητικών σχημάτων που χρησιμοποιήθηκαν για την επίλυση των (2.2.5) και (2.3.2) στη μία και τις δύο διαστάσεις αντίστοιχα.

Υπάρχει πληθώρα αριθμητικών σχημάτων για την προσέγγιση των λύσεων των παραπάνω συστημάτων, τα οποία βασίζονται σε μεθόδους πεπερασμένων διαφορών και πεπερασμένων όγκων. Στην εργασία αυτή θα αναφερθούμε σε σχήματα που έχουν ως βάση τις πεπερασμένους όγκους.

Θα παρουσιαστούν αριθμητικά σχήματα πρώτης και δευτέρας τάξης. Είναι γνωστό ότι τα σχήματα πρώτης τάξης παρουσιάζουν *αριθμητική διάχυση* ενώ τα σχήματα δευτέρας παρουσιάζουν *αριθμητική διασπορά*. Διάχυση προκαλείται όταν το μέγεθος του κύματος ελαττώνεται με αποτέλεσμα η αριθμητική λύση να «απλώνει». Διασπορά προκαλείται όταν το κύμα ταξιδεύει με διαφορετικές ταχύτητες με αποτέλεσμα την εμφάνιση ταλαντώσεων στο αριθμητικό αποτέλεσμα. Η διάχυση και η διασπορά μπορούν να προκαλέσουν ανακριβή αποτελέσματα όπως συμβαίνει στο Σχήμα 3.3 στις κλασικές μεθόδους των Lax-Friedrichs και MacCormack.

Ένας τρόπος για να μειωθούν τα παραπάνω φαινόμενα για σχήματα δευτέρας τάξης είναι να κάνουμε χρήση μίας μεθόδου η οποία θα ικανοποιεί την ιδιότητα της *Ολικής Ελάττωσης Μεταβολής (Total Variational Diminishing property)*, ή αλλιώς την ιδιότητα TVD [16], [32]. Οι μέθοδοι *περιοριστών-ροής (flux-limiter)* ικανοποιούν την ιδιότητα TVD και επιτυγχάνουν μία εναλλαγή μεταξύ προσέγγισης δευτέρας τάξης όταν η περιοχή λύσης είναι ομαλή και πρώτης τάξης όταν βρίσκονται κοντά σε ασυνέχειες.

3.1 Αριθμητικά σχήματα 1D

Μία προσέγγιση με ευρεία χρήση για την αριθμητική επίλυση της (2.2.5) είναι η μέθοδος πεπερασμένων όγκων. Η μέθοδος αυτή περιλαμβάνει αντικατάσταση των μερικών παραγώγων της (2.2.5) με προσεγγίσεις που προέρχονται από

πεπερασμένους όγκους, π.χ.

$$\frac{\partial \mathbf{w}}{\partial t} = \frac{\mathbf{w}_i^{n+1} - \mathbf{w}_i^n}{\Delta t},$$

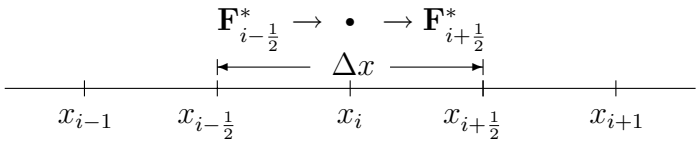
η οποία αποτελεί μία προσέγγιση στο χρόνο με προς τα εμπρός διαφορά και αποτελεί μία άμεση (explicit) μέθοδο. Θα πρέπει να υπάρξει μεγάλη προσοχή όταν χρησιμοποιούμε πεπερασμένες διαφορές για την κατασκευή των σχημάτων καθώς θα πρέπει να διασφαλίσουμε την ισχύ των νόμων διατήρησης. Ένα σχήμα πεπερασμένων διαφορών το οποίο δεν είναι συντηρητικό μπορεί να διαδώσει ασυνέχειες με λάθος ταχύτητα κύματος δίνοντας ανακριβή αριθμητικά αποτελέσματα. Για το λόγο αυτό κατασκευάζουμε σχήματα της μορφής

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^n - \frac{\Delta t}{\Delta x}\left[\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}}\right], \tag{3.1.1}$$

όπου

$$\mathbf{w}_i^n = \frac{1}{\Delta x}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{w}_i(x,t)\,dx$$

όπως αναφέρονται στο [19].



Σχήμα 3.1: Διακριτοποίηση για τη μία διάσταση.

Το να επιτύχουμε ένα συντηρητικό σχήμα όταν υπάρχει πηγαίος όρος μπορεί να είναι εξαιρετικά δύσκολο, κυρίως όταν ο πηγαίος όρος είναι πολύπλοκος. Παρόλα αυτά υπάρχει μία ποικιλία τακτικών προσέγγισης του πηγαίου όρου οι οποίες χωρίζονται σε δύο κύριες κατηγορίες:

- μία σημειακή προσέγγιση, όπου προσθέτουμε έναν προσεγγιστικό πηγαίο όρο στη (3.1.1)

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^n - \frac{\Delta t}{\Delta x}\left[\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}}\right] + \Delta t \mathbf{R}_i^n, \tag{3.1.2}$$

- μία φυσική προσέγγιση, όπου παίρνουμε μέσες διακριτές τιμές του πηγαίου όρου και κάνουμε upwind

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^n - \frac{\Delta t}{\Delta x}\left[\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}}\right] + \Delta t \mathbf{R}_i^*. \tag{3.1.3}$$

  με διακριτοποίηση ως προς τα $\mathbf{F}^*_{i\pm\frac{1}{2}}$ ώστε να ικανοποιείται η C-Ιδιότητα την οποία θα αναφέρουμε στην επόμενη παράγραφο.

3.1.1 Η C-Ιδιότητα

Θεωρούμε τις εξισώσεις SW για την περίπτωση μίας στάσιμης κατάστασης, δηλαδή

$$u(x,t) \equiv 0 \ \text{ και } \ h(x,t) \equiv D - B(x,t) \ \ \forall (x,t).$$

Για αυτή τη σταθερή περίπτωση, $\mathbf{w}_t = 0$, η συνάρτηση ροής και ο πηγαίος όρος έρχονται σε ισορροπία:

$$\mathbf{F}(\mathbf{w})_x = \mathbf{R}.$$

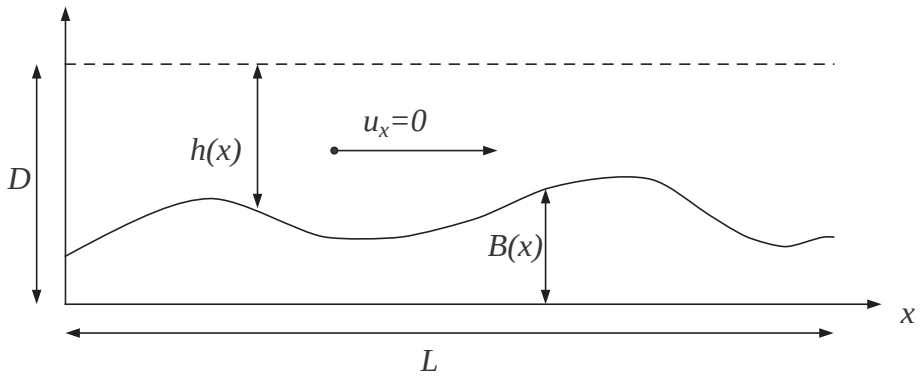
Ένα αριθμητικό σχήμα θα πρέπει να καταφέρνει να φέρει σε ισορροπία την αριθμητική ροή με την προσέγγιση του πηγαίου όρου,

$$\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}} = \mathbf{R}^*_i.$$

Τα παραπάνω θα μπορούσαν να γενικευτούν στην περίπτωση, όχι μόνο της ήρεμης ροής, αλλά γενικότερα των στάσιμων ροών με $u \neq 0$. Δηλαδή θα μπορούσαμε να είχαμε ότι

$$(u(x,t))_x \equiv 0 \ \text{ και } \ h(x,t) \equiv D - B(x,t) \ \ \forall (x,t),$$

όπως αυτή παρουσιάζεται στο Σχήμα 3.2.



Σχήμα 3.2: Η περίπτωση σταθερής κατάστασης: $u_x = 0$ και $h = D - B$.

Ακριβώς ανάλογα με τη μία διάσταση όλα τα παραπάνω μπορούν να επεκταθούν και στις δύο διαστάσεις. Ομοίως θεωρούμε τις εξισώσεις SW σε μια ήρεμη ροή,

$$u(x,y,t) \equiv 0, \ \ v(x,y,t) \equiv 0 \ \text{ και } \ h(x,y,t) \equiv D - B \ \ \forall (x,y,t).$$

Πάλι, για αυτή τη σταθερή περίπτωση, $\mathbf{w}_t = 0$, η συνάρτηση ροής και ο πηγαίος όρος έρχονται σε ισορροπία:

$$\mathbf{F}(\mathbf{w})_x + \mathbf{G}(\mathbf{w})_y = \mathbf{R}.$$

Στη μία διάσταση, οι όροι που χρειάστηκε να έρθουν σε ισορροπία ήταν δύο. Εδώ όμως οι όροι που πρέπει να ισορροπίσουν είναι τρεις, πράγμα που είναι πιο δύσκολο να πραγματοποιηθεί αριθμητικά. Για το λόγο αυτό ξαναγράφουμε τον πηγαίο όρο όπως στην (2.3.3) και διασπάμε την εξίσωση σε δύο εξισώσεις, οπότε έχουμε

$$\mathbf{F}(\mathbf{w})_x = \mathbf{f} \ \text{ και } \ \mathbf{G}(\mathbf{w})_y = \mathbf{g}.$$

Και εδώ η ισορροπία μεταξύ των αριθμητικών ροών και της προσέγγισης του πηγαίου όρου πρέπει να υπάρχει,

$$\mathbf{F}^*_{i+\frac{1}{2},j} - \mathbf{F}^*_{i-\frac{1}{2},j} = \mathbf{f}^*_{i,j} \ \text{ και } \ \mathbf{G}^*_{i+\frac{1}{2},j} - \mathbf{G}^*_{i-\frac{1}{2},j} = \mathbf{g}^*_{i,j}.$$

Αντίστοιχα με τη μία διάσταση μπορεί να υπάρξει μία γενίκευση των παραπάνω για σταθερές ροές, δηλαδή

$$(u(x,y,t))_x \equiv 0, \ \ (v(x,y,t))_y \equiv 0 \ \text{ και } \ h(x,y,t) \equiv D - B \ \ \forall (x,y,t).$$

Και για τις δύο περιπτώσεις, της μίας και των δύο διαστάσεων, έχουμε ότι αν υπάρχει αυτή η ισορροπία το αριθμητικό μας σχήμα ικανοποιεί:

- την προσεγγιστική C-Ιδιότητα (Ιδιότητα διατηρησιμότητας), αν το αριθμητικό σχήμα έχει ακρίβεια τάξης $O(\Delta x^2)$ όταν εφαρμοστεί σε περίπτωση ήρεμης ή σταθερής ροής·
- την ακριβή C-Ιδιότητα, αν το αριθμητικό σχήμα είναι ακριβές όταν εφαρμοστεί σε περίπτωση ήρεμης ή σταθερής ροής.

3.1.2 Σχήμα Lax-Friedrichs πρώτης τάξης

Το πρώτο σχήμα που θα αναφέρουμε είναι το Lax-Friedrichs. Το σχήμα αυτό έχει προσθήκη του προσεγγιστικού πηγαίου όρου για να επιτύχει τη λύση της (2.2.5) και έχει τη μορφή

$$\mathbf{w}_i^{n+1} = \frac{\mathbf{w}_{i+1}^n + \mathbf{w}_{i-1}^n}{2} - \frac{\Delta t}{2\Delta x}\left(\mathbf{F}_{i+1}^n - \mathbf{F}_{i-1}^n\right) + \frac{\Delta t}{\Delta x}\mathbf{R}_i^n, \tag{3.1.4}$$

όπου

$$\mathbf{w}_i^n = \begin{bmatrix} h_i^n \\ h_i^n u_i^n \end{bmatrix}, \ \mathbf{F}_i^n = \begin{bmatrix} h_i^n u_i^n \\ h_i^n (u_i^n)^2 + \dfrac{1}{2} g (h_i^n)^2 \end{bmatrix} \ \text{ και } \ \mathbf{R}_i^n = \begin{bmatrix} 0 \\ -g h_i^n (B_i - B_{i-1}) \end{bmatrix}.$$

Παρατηρούμε ότι το σχήμα αυτό έχει τη μορφή της (3.1.2) και ο Ιακωβιανός πίνακας του συστήματος (2.2.5) δεν χρειάζεται να προσεγγιστεί. Παρόλα αυτά όμως το σχήμα Lax-Friedrichs μπορεί να υποφέρει από διάχυση η οποία όμως μερικές φορές μπορεί να περιοριστεί κάνοντας χρήση μικρού χωρικού βήματος και αναφέρεται ως μέτρο σύγκρισης.

3.1.3 Λύτες Riemann και το σχήμα του Roe

Ο Roe [31] κατασκεύασε μία μέθοδο η οποία επιλύει συστήματα νόμων διατήρησης χρησιμοποιώντας μία κλαδική σταθερή προσέγγιση

$$\mathbf{w}(x,t_n)=\begin{cases}\mathbf{w}_L & \text{αν } x_L-\dfrac{\Delta x}{2}<x<x_L+\dfrac{\Delta x}{2}\\[2ex] \mathbf{w}_R & \text{αν } x_R-\dfrac{\Delta x}{2}<x<x_R+\dfrac{\Delta x}{2}\end{cases}$$

όπου τα $\mathbf{w_L}$ και $\mathbf{w_R}$ αποτελούν τις κλαδικές σταθερές καταστάσεις στο χρόνο t_n, και καθορίζουν την ακριβή λύση ενός γραμμικοποιημένου προβλήματος Riemann το οποίο συσχετίζεται με το (2.2.5)

$$\frac{\partial \mathbf{w}}{\partial t}+\tilde{\mathbf{A}}(\mathbf{w}_L,\mathbf{w}_R)\frac{\partial \mathbf{w}}{\partial x}=0, \tag{3.1.5}$$

όπου

$$\tilde{\mathbf{A}}(\mathbf{w}_L,\mathbf{w}_R)\approx\frac{\partial \mathbf{F}}{\partial \mathbf{w}}$$

ειναι ο γραμμικοποιημένος Ιακωβιανός πίνακας και η τίλδα (˜) ονομάζεται ο *μέσος του Roe*. Οι ιδιοτιμές και τα ιδιοδιανύσματα του $\tilde{\mathbf{A}}$ είναι $\tilde{\lambda}$ και $\tilde{\mathbf{e}}$ αντίστοιχα. Έχοντας υπολογίσει τις ιδιοτιμές, τα ιδιοδιανύσματα και τα wave strengths, το σχήμα του Roe [31] παίρνει τη μορφή

$$\mathbf{w}_i^{n+1}=\mathbf{w}_i^n-\frac{\Delta t}{\Delta x}\big(\mathbf{F}^*_{i+\frac{1}{2}}-\mathbf{F}^*_{i-\frac{1}{2}}\big), \tag{3.1.6}$$

όπου

$$\mathbf{F}^*_{i+\frac{1}{2}}=\frac{1}{2}\big(\mathbf{F}^n_{i+1}+\mathbf{F}^n_i\big)-\frac{1}{2}\Big[\sum_{k=1}^{M}\tilde{\alpha}_k|\tilde{\lambda}_k|\big(1-\phi_k(1-|\nu_k|)\big)\tilde{\mathbf{e}}_k\Big]_{i+\frac{1}{2}},$$

$$\nu_k=\frac{\Delta t}{\Delta x}\tilde{\lambda}_k,\quad \theta_k=\frac{(\tilde{\alpha}_k)_{I\pm\frac{1}{2}}}{(\tilde{\alpha}_k)_{i\pm\frac{1}{2}}},\quad I=i-\operatorname{sgn}\big((\nu_k)_{i\pm\frac{1}{2}}\big)$$

και το ϕ μπορεί να είναι οποιοσδήποτε περιοριστής-ροής του Πίνακα 3.1.

Όπως έχει παρουσιαστεί έως τώρα, το σχήμα του Roe υπολογίζει τη λύση ομογενών συστημάτων. Στην παρούσα φάση θα επιχειρήσουμε μία επέκταση του (3.1.6) ούτως ώστε να προσεγγίσουμε αριθμητικά τις εξισώσεις SW όταν υπάρχει πηγαίος όρος. Ένας τρόπος για να πραγματοποιηθεί αυτό είναι να προσθέσουμε απευθείας τον πηγαίο όρο στην (3.1.6)

$$\mathbf{w}_i^{n+1}=\mathbf{w}_i^n-\frac{\Delta t}{\Delta x}\big(\mathbf{F}^*_{i+\frac{1}{2}}-\mathbf{F}^*_{i-\frac{1}{2}}\big)+\frac{\Delta t}{\Delta x}\mathbf{R}_i^n \tag{3.1.7}$$

όπου

$$\mathbf{F}^*_{i+\frac{1}{2}} = \frac{1}{2}\big(\mathbf{F}^n_{i+1} + \mathbf{F}^n_i\big) - \frac{1}{2}\Big[\sum_{k=1}^{M} \tilde{\alpha}_k |\tilde{\lambda}_k| \big(1 - \phi_k(1 - |\nu_k|)\big)\tilde{\mathbf{e}}_k\Big]_{i+\frac{1}{2}},$$

$$\nu_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}_k, \;\; \theta_k = \frac{(\tilde{\alpha}_k)_{I\pm\frac{1}{2}}}{(\tilde{\alpha}_k)_{i\pm\frac{1}{2}}}, \;\; I = i - \mathrm{sgn}\big((\nu_k)_{i\pm\frac{1}{2}}\big)$$

και

$$\mathbf{R}^n_i = \begin{bmatrix} 0 \\ -gh^n_i(B_i - B_{i-1}) \end{bmatrix}.$$

Τώρα πρέπει να υπολογιστούν οι ιδιοτιμές, τα ιδιοδιανύσματα και τα wave strengths. Για τις εξισώσεις SW έχουμε

$$\tilde{\lambda}_1 = \tilde{u} + \tilde{c}, \;\; \tilde{\lambda}_2 = \tilde{u} - \tilde{c},$$

$$\tilde{\mathbf{e}}_1 = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c} \end{bmatrix}, \;\; \tilde{\mathbf{e}}_2 = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{c} \end{bmatrix},$$

$$\tilde{\alpha}_1 = \frac{1}{2}\Delta h + \frac{1}{2\tilde{c}}\big(\Delta(hu) - \tilde{u}\Delta h\big), \;\; \tilde{\alpha}_1 = \frac{1}{2}\Delta h - \frac{1}{2\tilde{c}}\big(\Delta(hu) - \tilde{u}\Delta h\big)$$

όπου

$$\tilde{u} = \frac{\sqrt{h_R}u_R + \sqrt{h_L}u_L}{\sqrt{h_R} + \sqrt{h_L}} \;\; \text{και} \;\; \tilde{c} = \sqrt{g\frac{h_R + h_L}{2}}.$$

Επιπρόσθετα θα μπορούσαμε να χρησιμοποιήσουμε μία μετατροπή του Hubbard η οποία εμπεριέχει μία TVD προσέγγιση του πηγαίου όρου, δηλαδή

$$\mathbf{w}^{n+1}_i = \mathbf{w}^n_i - \frac{\Delta t}{\Delta x}\big(\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}}\big) + \frac{\Delta t}{\Delta x}\big(\mathbf{R}^-_{i+\frac{1}{2}} + \mathbf{R}^+_{i-\frac{1}{2}}\big) \tag{3.1.8}$$

όπου

$$\mathbf{F}^*_{i+\frac{1}{2}} = \frac{1}{2}\big(\mathbf{F}^n_{i+1} + \mathbf{F}^n_i\big) - \frac{1}{2}\Big[\sum_{k=1}^{M} \tilde{\alpha}_k |\tilde{\lambda}_k| \big(1 - \phi_k(1 - |\nu_k|)\big)\tilde{\mathbf{e}}_k\Big]_{i+\frac{1}{2}},$$

$$\mathbf{R}^{\pm}_{i+\frac{1}{2}} = \frac{1}{2}\left[\sum_{k=1}^{2} \tilde{\beta}_k \tilde{\mathbf{e}}_k \Big(1 \pm \mathrm{sgn}(\tilde{\lambda}_k)\big(1 - \phi_k(1 - |\nu_k|)\big)\Big)\right]_{i+\frac{1}{2}},$$

$$\nu_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}_k, \;\; \theta_k = \frac{(\tilde{\alpha}_k)_{I\pm\frac{1}{2}}}{(\tilde{\alpha}_k)_{i\pm\frac{1}{2}}}, \;\; I = i - \mathrm{sgn}\big((\nu_k)_{i\pm\frac{1}{2}}\big),$$

$$\tilde{\beta}_1 = \frac{\tilde{c}\Delta H}{2} \;\; \text{και} \;\; \tilde{\beta}_2 = -\frac{\tilde{c}\Delta H}{2}.$$

Για ϕ_k μπορούμε να χρησιμοποιήσουμε οποιοδήποτε περιοριστή-ροής.

3.1.4 Σχήμα MacCormack

Στο [21] οι LeVeque και Yee παρουσίασαν μία μετατροπή του σχήματος MacCormack η οποία έχει ως εξής:

$$\mathbf{w}_i^{n+1} = \frac{1}{2}\big(\mathbf{w}_i^n + \mathbf{w}_i^{(1)}\big) - \frac{\Delta t}{2\Delta x}\big(\mathbf{F}_i^{(1)} - \mathbf{F}_{i-1}^{(1)}\big) + \frac{\Delta t}{2\Delta x}\mathbf{R}_i^{(1)}, \tag{3.1.9}$$

όπου

$$\mathbf{w}_i^{(1)} = \mathbf{w}_i^n - \frac{\Delta t}{\Delta x}\big(\mathbf{F}_{i+1}^n - \mathbf{F}_i^n\big) + \frac{\Delta t}{\Delta x}\mathbf{R}_i^n,$$

$$\mathbf{w}_i^n = \begin{bmatrix} h_i^n \\ h_i^n u_i^n \end{bmatrix}, \quad \mathbf{F}_i^n = \begin{bmatrix} h_i^n u_i^n \\ h_i^n (u_i^n)^2 + \dfrac{1}{2} g (h_i^n)^2 \end{bmatrix}$$

και

$$\mathbf{R}_i^n = \begin{bmatrix} 0 \\ -\dfrac{1}{2} g (h_{i+1}^n + h_i^n)(B_i - B_{i-1}) \end{bmatrix}.$$

Δυστυχώς το σχήμα αυτό υποφέρει από αριθμητική διασπορά με αποτέλεσμα την εμφάνιση ταλαντώσεων στην αριθμητική λύση όταν υπάρχουν ασυνέχειες. Οι LeVeque και Yee εξάλειψαν τις ταλαντώσεις προσαρμόζοντας την (3.1.9) και κάνοντας χρήση *περιοριστών-κλίσης (slope-limiters)* έτσι ώστε το σχήμα να ικανοποιεί την ιδιότητα TVD, οπότε έχουμε

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^{(2)} + \mathbf{D}_{i+\frac{1}{2}}^n - \mathbf{D}_{i-\frac{1}{2}}^n \tag{3.1.10}$$

Το $\mathbf{w}_i^{(2)}$ είναι η αριθμητική προσέγγιση η οποία προκύπτει από το σχήμα (3.1.9), δηλαδή

$$\mathbf{w}_i^{(2)} = \frac{1}{2}\big(\mathbf{w}_i^n + \mathbf{w}_i^{(1)}\big) - \frac{\Delta t}{2\Delta x}\big(\mathbf{F}_i^{(1)} - \mathbf{F}_{i-1}^{(1)}\big) + \frac{\Delta t}{2\Delta x}\mathbf{R}_i^{(1)},$$

και

$$\mathbf{D}_{i+\frac{1}{2}}^n = \frac{1}{2}\sum_{k=1}^{p}\big[|\nu_k|(1-|\nu_k|)(\alpha_k - Q_k)\mathbf{e}_k\big]_{i+\frac{1}{2}}^n$$

όπου

$$Q_{i+\frac{1}{2}}^k = \text{min mod}(\alpha_{i-\frac{1}{2}}^k, \alpha_{i+\frac{1}{2}}^k, \alpha_{i+\frac{3}{2}}^k),$$

$$\text{min mod}(a,b,c) = \begin{cases} d\min\big(|a|,|b|,|c|\big) & \text{αν } \ d = \text{sgn}(a) = \text{sgn}(b) = \text{sgn}(c) \\ 0 & \text{αλλιώς} \end{cases}$$

$$\begin{bmatrix} \alpha_{i+\frac{1}{2}}^1 \\ \alpha_{i+\frac{1}{2}}^2 \end{bmatrix} = \mathbf{X}_{i+\frac{1}{2}}^{-1}\big(\mathbf{w}_{i+1}^n - \mathbf{w}_i^n\big) \ \text{ και } \ \nu_{i+\frac{1}{2}}^k = \frac{\Delta t}{\Delta x}\lambda_{i+\frac{1}{2}}^k.$$

όπου το $\mathbf{X}$ αποτελεί τον πίνακα που περιέχει τα ιδιοδιανύσματα του Ιακωβιανού πίνακα. Το k παριστάνει τον k-οστό όρο του διανύσματος, λ και α οι ιδιοτιμές και τα wave strengths του συστήματος (2.2.5) αντίστοιχα.

Για τις εξισώσεις SW η προσέγγιση του πηγαίου όρου όπως αυτή δώθηκε παραπάνω επιτρέπει στο σχήμα MacCormack να ικανοποιεί τη C-Ιδιότητα για το δευτέρας τάξης σχήμα (3.1.9). Δυστυχώς η έκδοση του σχήματος με τη χρήση περιοριστών-κλίσης (3.1.11) δεν ικανοποιεί τη C-Ιδιότητα, αλλά το σχήμα αποκτά την δυνατότητα αυτή αν χρησιμοποιηθούν περιοριστές-ροής. Οπότε η εκδοχή του σχήματος MacCormack με περιοριστές-ροής είναι

$$\mathbf{w}_i^{n+1} = \mathbf{w}_i^{(2)} + \mathbf{D}_{i+\frac{1}{2}}^n - \mathbf{D}_{i-\frac{1}{2}}^n \tag{3.1.11}$$

όπου ομοίως με προηγουμένως

$$\mathbf{w}_i^{(2)} = \frac{1}{2}\big(\mathbf{w}_i^n + \mathbf{w}_i^{(1)}\big) - \frac{\Delta t}{2\Delta x}\big(\mathbf{F}_i^{(1)} - \mathbf{F}_{i-1}^{(1)}\big) + \frac{\Delta t}{2\Delta x}\mathbf{R}_i^{(1)},$$

αλλά ο όρος $\mathbf{D}$ μεταβάλεται και γίνεται

$$\mathbf{D}_{i+\frac{1}{2}}^n = \frac{\Delta t}{2\Delta x}\sum_{k=1}^{p}\big[(\alpha_k|\lambda_k| - \beta_k \mathrm{sgn}(\lambda_k))(1-|\nu_k|)(1-\phi(\theta_k))\mathbf{e}_k\big]_{i+\frac{1}{2}}^n.$$

Το ϕ μπορεί να είναι οποιοσδήποτε περιοριστής-ροής [32] του Πίνακα 3.1.

Πίνακας 3.1: Περιοριστές-ροής δευτέρας τάξης

Περιοριστής-ροής	$\phi(\theta)$
Minmod	$\phi(\theta) = \max\big(0, \min(1,\theta)\big)$
Superbee	$\phi(\theta) = \max\big(0, \min(2\theta,1), \min(\theta,2)\big)$
van Leer	$\phi(\theta) = \dfrac{\lvert\theta\rvert + \theta}{1+\lvert\theta\rvert}$
van Albada	$\phi(\theta) = \dfrac{\theta^2+\theta}{1+\theta^2}$

Τα β_k μπορούν να βρεθούν από την

$$\sum_{k=1}^{n}[\beta_k \mathbf{e}_k]_{i+\frac{1}{2}}^n = \mathbf{R}_{i+\frac{1}{2}}^n,$$

οπότε

$$(\beta_{1,2})_{i+\frac{1}{2}}^n = \pm\frac{1}{2}\Big(\sqrt{gh}\Delta B\Big)_{i+\frac{1}{2}}^n \quad \text{όπου} \quad \Delta B_{i+\frac{1}{2}}^n = B_{i+1}^n - B_i^n.$$

3.1.5 Η συνθήκη CFL για τη μία διάσταση

Μία από τις πρώτες δημοσιεύσεις που γράφτηκαν για μεθόδους πεπερασμένων διαφορών σε Μερικές Διαφορικές Εξισώσεις ήταν το 1928 από τους Courant, Friedrichs και Lewy [6]. Χρησιμοποίησαν μεθόδους πεπερασμένων διαφορών για να αποδείξουν την ύπαρξη λύσης συγκεκριμένων ΜΔΕ. Τότε διαπίστωσαν την αναγκαιότητα μίας συνθήκης η οποία θα εξασφαλίζει την ευστάθεια της αριθμητικής μεθόδου που χρησιμοποιούσαν. Η ποσότητα

$$\nu = \max_p \left| \frac{\lambda_p \Delta t}{\Delta x} \right| \tag{3.1.12}$$

ονομάζεται αριθμός Courant ή αριθμός CFL. Μία αναγκαία συνθήκη για την ευστάθεια των άμμεσων σχημάτων που παρουσιάστηκαν παραπάνω είναι ο αριθμός αυτός να μην ξεπερνάει τη μονάδα.

Με βάση τον αριθμό Courant προσδιορίζεται και το χρονικό βήμα το οποίο υπολογίζεται κάθε χρονική στιγμή από τη σχέση

$$\Delta t = \frac{\nu \Delta x}{\max_i \big(|\lambda_k| \big)}. \tag{3.1.13}$$

3.2 Προβλήματα στη μία διάσταση

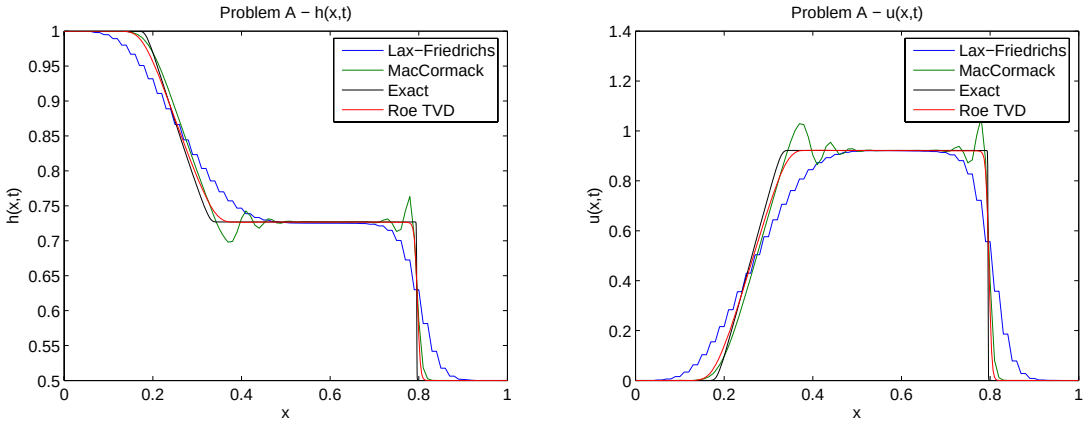
Σε αυτήν την ενότητα θα παρουσιάσουμε ορισμένα προβλήματα τα οποία θα επιλύσουμε κάνοντας χρήση κάποιων από τις μεθόδους που προαναφέραμε. Επιλέξαμε την Lax-Friedrichs ως μέθοδο πρώτης τάξης, την MacCormack ως μέθοδο δευτέρας τάξης και την Roe TVD ως μία μέθοδο η οποία ικανοποιεί την ιδιότητα TVD. Για την Roe TVD χρησιμοποιήσαμε για περιοριστή-ροής τον Superbee.

3.2.1 Πρόβλημα A - Κατάρρευση φράγματος

Το παρόν πρόβλημα δεν εμπεριέχει πηγαίο όρο. Θεωρούμε τις αρχικές συνθήκες

$$u(x,0) = 0 \ \text{ και } \ h(x,0) = \begin{cases} 1 & \text{αν } \ 0 \le x \le \dfrac{1}{2} \\ \phi_0 & \text{αν } \ \dfrac{1}{2} < x \le 1 \end{cases}$$

οι οποίες περιγράφουν το πρόβλημα της ολικής κατάρρευσης φράγματος. Στο σημείο $x = 0.5$ υπάρχει τείχος το οποίο καταρρέει ακαριαία και το οποίο χωρίζει το ρευστό σε δύο διαφορετικά βάθη. Η ροή που πραγματοποιείται είναι ελεύθερη.



Σχήμα 3.3: Πρόβλημα Α για $t = 0.1\,\mathrm{sec}$, $\mathrm{CFL} = 0.7$ και grid από 100 σημεία.

Το σχήμα των Lax-Friedriechs πάσχει από απότομες μεταβολές υπό τη μορφή «σκαλοπατιών». Το σχήμα MacCormack χωρίς TVD διόρθωση έχει προφανές πρόβλημα διασποράς ενώ το σχήμα Roe TVD με περιοριστή-ροής Superbee συμπεριφέρεται πολύ καλά.

3.2.2 Πρόβλημα Β - Κατάρρευση φράγματος με τοπογραφία

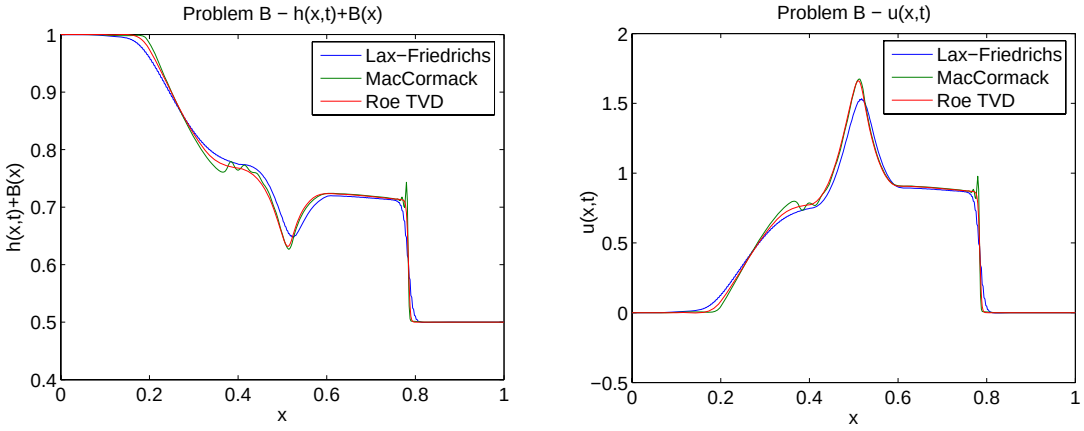
Αυτό το πρόβλημα είναι παρόμοιο με το πρόβλημα Α, με εξαίρεση ότι σε αυτή την περίπτωση υπάρχει τοπογραφία η οποία περιγράφεται ως εξής

$$B(x) = \begin{cases} \frac{1}{8}\Big(\cos\big(10\pi(x-\frac{1}{2})\big)+1\Big) & \text{αν } \frac{2}{5} \le x \le \frac{3}{5} \\ 0 & \text{αλλιώς} \end{cases}$$

με αρχικές συνθήκες

$$u(x,0) = 0 \;\; \text{και} \;\; h(x,0) = \begin{cases} 1 - B(x) & \text{αν } 0 \le x \le \frac{1}{2} \\ \phi_0 - B(x) & \text{αν } \frac{1}{2} < x \le 1 \end{cases}$$

Ομοίως με το Πρόβλημα Α υπάρχει τείχος στο σημείο $x = 0.5$ το οποίο καταρρέει και στις άκρες του χωρίου η ροή είναι ελεύθερη. Λόγο της ύπαρξης του πηγαίου όρου είναι εύκολο να υπάρξουν δυσκολίες στην αριθμητική προσέγγιση της λύσης του προβλήματος όπως φαίνεται και στο Σχήμα 3.4.



Σχήμα 3.4: Πρόβλημα B για $t = 0.1\,\mathrm{sec}$, $\mathrm{CFL} = 0.7$ και grid από 500 σημεία.

Έχοντας κάνει πολύ μεγαλύτερη διαμέριση του χωρίου από ότι στο προηγούμενο πρόβλημα η μεγάλη αδυναμία που είχε εμφανίσει το Lax-Friedriechs έχει σχεδόν αφανιστεί. Παρόλα αυτά δείχνει να πάσχει από αριθμητική διάχυση. Το σχήμα MacCormack εμφανίζει αριθμητική διασπορά.

3.2.3 Πρόβλημα C - Όπως περιγράφεται στο [20] από τον LeVeque

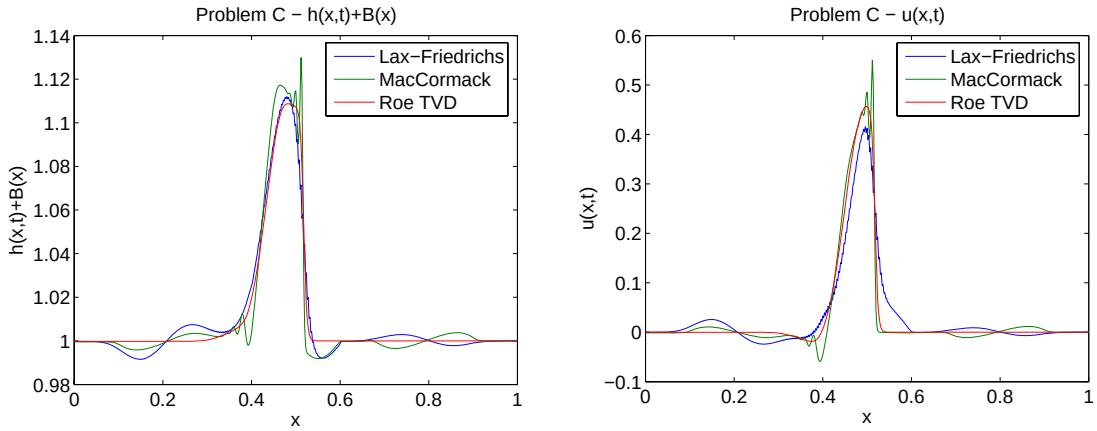
Σε αυτό το πρόβλημα υπάρχει τοπογραφία μιας και ο πυθμένας είναι μεταβλητού ύψους και περιγράφεται από τον τύπο

$$B(x) = \begin{cases} \frac{1}{4}\Big(\cos\big(10\pi(x-\frac{1}{2})\big)+1\Big) & \text{αν } \frac{2}{5} \le x \le \frac{3}{5} \\ 0 & \text{αλλιώς} \end{cases}$$

και οι αρχικές συνθήκες είναι

$$u(x,0) = 0 \;\text{ και }\; h(x,0) = \begin{cases} 1 - B(x) & \text{αν } x < 0.1 \\ 1.2 - B(x) & \text{αν } 0.1 \le x \le 0.2 \\ 1 - B(x) & \text{αν } x > 0.2 \end{cases}$$

Το πρόβλημα αναπαριστά έναν αρχικό παλμό ο οποίος «σπάει» σε δύο κύματα τα οποία κινούνται σε αντίθετες κατευθύνσεις. Το κύμα το οποίο κινείται προς τα δεξιά περνάει το ύψωμα της τοπογραφίας και ανακλάται μερικώς. Στο πρόβλημα αυτό δεν υπάρχουν τείχη στις άκρες με αποτέλεσμα να μην παρουσιάζεται ανάκλαση στα σημεία αυτά.



Σχήμα 3.5: Πρόβλημα C για $t = 0.1\,\text{sec}$, $\text{CFL} = 0.7$ και grid από 500 σημεία.

Παρόλο που η διαμέριση είναι αρκετά πυκνή, το σχήμα Lax-Friedriechs παρουσιάζει «σκαλοπάτια» και διασπορά. Το σχήμα MacCormack χωρίς διόρθωση εμφανίζει διασπορά ενώ το Roe TVD είναι μακράν το καλλίτερο.

3.2.4 Πρόβλημα D - Διάδοση παλιρροιακού κύματος σε τοπογραφία

Και αυτό το πρόβλημα παρουσιάζει πηγαίο όρο ο οποίος δίδεται από την συνάρτηση

$$H(x) = 50.5 - \frac{40x}{L} + 10\sin\big(\pi(\frac{4x}{L} + \frac{1}{2})\big),$$

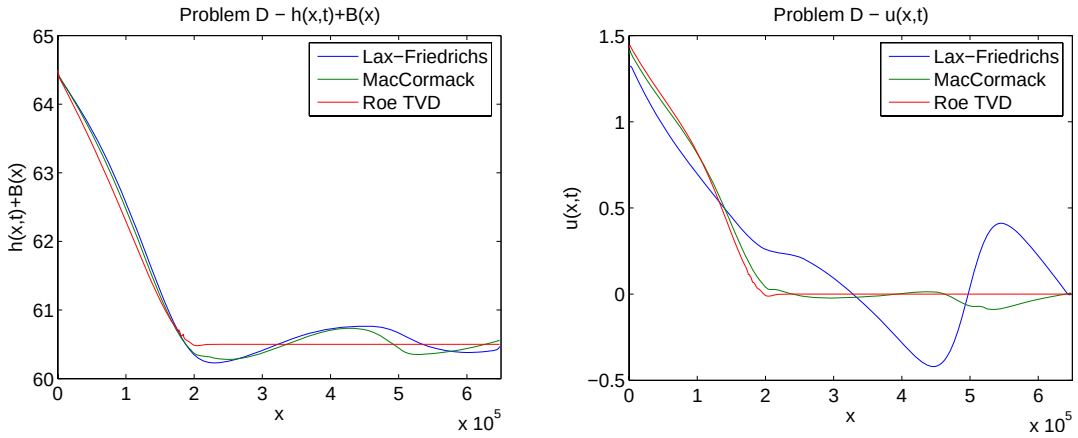
με αρχικές συνθήκες

$$u(x,0) = 0 \quad \text{και} \quad h(x,0) = H(x)$$

και συνοριακές συνθήκες

$$h(0,t) = \begin{cases} 64.5 + 4\cos\big(\pi(\frac{4t}{86400} - \frac{1}{2})\big) & \text{αν} \;\; t \leq 43200 \\ 60.5 & \text{αν} \;\; t > 43200 \end{cases}$$

Το πρόβλημα αυτό αναπαριστά ένα παλιρροιακό κύμα το οποίο διαδίδεται σε έναν πυθμένα μεταβαλλόμενου ύψους. Το χωρίο στο οποίο πραγματοποιείται το φαινόμενο έχει μήκος $L = 648000\,m$.

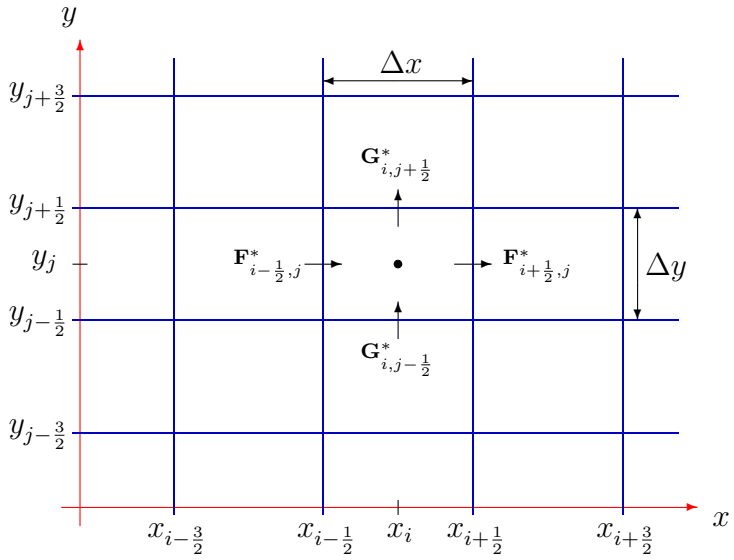


Σχήμα 3.6: Πρόβλημα D για $t = 10800\,\text{sec}$, $\text{CFL} = 0.7$ και grid από 500 σημεία.

Και σε αυτήν την περίπτωση τα αποτελέσματα που προκύπτουν από τα σχήματα Lax-Friedriechs και MacCormack είναι εντελώς ανακριβή.

3.3 Αριθμητικά σχήματα 2D

Έχοντας ήδη δει μερικά από τα πλέον γνωστά σχήματα στη μία διάσταση, στην ενότητα αυτή θα αναφερθούμε σε δύο πολύ διαδεδομένα σχήματα για τις δύο διαστάσεις. Τα σχήματα που θα παρουσιάσουμε είναι τα αντίστοιχα Roe TVD και MacCormack TVD για τις δύο διατάσεις, τα οποία θα μας απασχολήσουν για το υπόλοιπο της παρούσας εργασίας.



Σχήμα 3.7: Πλέγμα διακριτοποίησης για τις δύο διαστάσεις.

3.3.1 Σχήμα Roe 2D

Στις δύο διαστάσεις, η έκδοση του σχήματος του Roe προκύπτει διακριτοποιώντας το σύστημα ξεχωριστά για κάθε μία από τις δύο κατευθύνσεις.

$$\mathbf{w}_{i,j}^{n+1} = \mathbf{w}_{i,j}^{n} - \frac{\Delta t}{\Delta x}\big(\mathbf{F}^*_{i+\frac{1}{2},j} - \mathbf{F}^*_{i-\frac{1}{2},j}\big) - \frac{\Delta t}{\Delta y}\big(\mathbf{G}^*_{i,j+\frac{1}{2}} - \mathbf{G}^*_{i,j-\frac{1}{2}}\big) + \Delta t \mathbf{R}^*_{i,j}, \quad (3.3.1)$$

με

$$\Delta t \mathbf{R}^*_{i,j} = \frac{\Delta t}{\Delta x}\mathbf{f}^*_{i,j} + \frac{\Delta t}{\Delta y}\mathbf{g}^*_{i,j},$$

όπου $\mathbf{f}^*_{i,j}$ και $\mathbf{g}^*_{i,j}$ είναι οι προσεγγίσεις του πηγαίου όρου προς τη x και y κατεύθυνση. Οπότε οι αριθμητικές ροές του σχήματος του Roe με περιοριστή-ροής είναι

$$\mathbf{F}^*_{i+\frac{1}{2},j} = \frac{1}{2}\big(\mathbf{F}^n_{i+1,j} + \mathbf{F}^n_{i,j}\big) - \frac{1}{2}\sum_{k=1}^{p}\Big[\tilde{\alpha}^F_k|\tilde{\lambda}^F_k|\big(1-\phi(\theta^F_k)(1-|\nu^F_k|)\big)\tilde{\mathbf{e}}^F_k\Big]_{i+\frac{1}{2},j}, \quad (3.3.2)$$

και

$$\mathbf{G}^*_{i,j+\frac{1}{2}} = \frac{1}{2}\big(\mathbf{G}^n_{i,j+1} + \mathbf{G}^n_{i,j}\big) - \frac{1}{2}\sum_{k=1}^{p}\Big[\tilde{\alpha}^G_k|\tilde{\lambda}^G_k|\big(1-\phi(\theta^G_k)(1-|\nu^G_k|)\big)\tilde{\mathbf{e}}^G_k\Big]_{i,j+\frac{1}{2}}, \quad (3.3.3)$$

όπου

$$\nu^F_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}^F_k, \quad \theta^F_k = \frac{(\tilde{\alpha}^F_k)_{I+\frac{1}{2},j}}{(\tilde{\alpha}^F_k)_{i+\frac{1}{2},j}}, \quad I = i - \mathrm{sgn}\big((\nu^F_k)_{i+\frac{1}{2},j}\big),$$

$$\nu^G_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}^G_k, \quad \theta^G_k = \frac{(\tilde{\alpha}^G_k)_{i,J+\frac{1}{2}}}{(\tilde{\alpha}^G_k)_{i,j+\frac{1}{2}}}, \quad J = j - \mathrm{sgn}\big((\nu^G_k)_{i,j+\frac{1}{2}}\big),$$

και το ϕ_k μπορεί να είναι οποιοσδήποτε περιοριστής-ροής από τον Πίνακα 3.1. Οι τιμές των $\tilde{\lambda}_k$ και $\tilde{\mathbf{e}}_k$ είναι οι ιδιοτιμές και τα ιδιοδιανύσματα των Ιακωβιανών πινάκων $\tilde{\mathbf{A}}$ και $\tilde{\mathbf{B}}$ και $\tilde{\alpha}_k$ είναι τα wave strengths. Οι δείκτες F και G δηλώνουν αν οι όροι είναι για τον Ιακωβιανό πίνακα του $\tilde{\mathbf{A}}$ ή του $\tilde{\mathbf{B}}$ αντίστοιχα.

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 \\ \tilde{c}^2 - \tilde{u}^2 & 2\tilde{u} & 0 \\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} \end{bmatrix} \text{ και } \tilde{\mathbf{B}} = \begin{bmatrix} 0 & 0 & 1 \\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} \\ \tilde{c}^2 - \tilde{v}^2 & 0 & 2\tilde{v} \end{bmatrix}$$

όπου $\tilde{c} = \sqrt{g\tilde{h}}$. Οι ιδιοτιμές για τον πίνακα $\tilde{\mathbf{A}}$ είναι

$$\tilde{\lambda}^F_1 = \tilde{u} - \tilde{c}, \quad \tilde{\lambda}^F_2 = \tilde{u} \text{ και } \tilde{\lambda}^F_3 = \tilde{u} + \tilde{c},$$

και για τον $\tilde{\mathbf{B}}$

$$\tilde{\lambda}^G_1 = \tilde{v} - \tilde{c}, \quad \tilde{\lambda}^G_2 = \tilde{v} \text{ και } \tilde{\lambda}^G_3 = \tilde{v} + \tilde{c}.$$

Τα αντίστοιχα ιδιοδιανύσματα είναι, για τον $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{e}}_1^F = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{c} \\ \tilde{v} \end{bmatrix}, \quad \tilde{\mathbf{e}}_2^F = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{και} \quad \tilde{\mathbf{e}}_1^F = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c} \\ \tilde{v} \end{bmatrix},$$

και για τον $\tilde{\mathbf{B}}$

$$\tilde{\mathbf{e}}_1^G = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} - \tilde{c} \end{bmatrix}, \quad \tilde{\mathbf{e}}_2^G = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{και} \quad \tilde{\mathbf{e}}_1^G = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} + \tilde{c} \end{bmatrix}.$$

Τα wave strengths για τον $\tilde{\mathbf{A}}$ είναι

$$\tilde{\alpha}_{1,3}^F = \frac{1}{2}\Delta h \pm \frac{1}{2\tilde{c}}\big(\tilde{u}\Delta h - \Delta(uh)\big) \quad \text{και} \quad \tilde{\alpha}_2^F = \frac{1}{\tilde{c}}\big(\Delta(vh) - \tilde{v}\Delta h\big),$$

όπου $\Delta\mathbf{w} = \mathbf{w}_{R,j} - \mathbf{w}_{L,j}$, και για τον $\tilde{\mathbf{B}}$

$$\tilde{\alpha}_{1,3}^G = \frac{1}{2}\Delta h \pm \frac{1}{2\tilde{c}}\big(\tilde{v}\Delta h - \Delta(vh)\big) \quad \text{και} \quad \tilde{\alpha}_2^G = \frac{1}{\tilde{c}}\big(\Delta(uh) - \tilde{u}\Delta h\big),$$

όπου $\Delta\mathbf{w} = \mathbf{w}_{i,R} - \mathbf{w}_{i,L}$. Οι μέσοι του Roe είναι

$$\tilde{u} = \frac{\sqrt{h_R}u_R + \sqrt{h_L}u_L}{\sqrt{h_R} + \sqrt{h_L}}, \quad \tilde{v} = \frac{\sqrt{h_R}v_R + \sqrt{h_L}v_L}{\sqrt{h_R} + \sqrt{h_L}} \quad \text{και} \quad \tilde{h} = \frac{1}{2}(h_R + h_L),$$

με R και L να δηλώνουν τις τιμές δεξιά και αριστερά του προβλήματος Riemann.

Για την προσέγγιση του πηγαίου όρου, ομοίως διακριτοποιούμε ξεχωριστά και παίρνουμε

$$\mathbf{f}_{i,j}^* = \mathbf{f}_{i+\frac{1}{2},j}^- + \mathbf{f}_{i-\frac{1}{2},j}^+ \quad \text{και} \quad \mathbf{g}_{i,j}^* = \mathbf{g}_{i,j+\frac{1}{2}}^- + \mathbf{g}_{i,j-\frac{1}{2}}^+, \tag{3.3.4}$$

όπου

$$\mathbf{f}_{i+\frac{1}{2},j}^{\pm} = \frac{1}{2}\sum_{k=1}^{p}\Big[\tilde{\beta}_k^F\tilde{\mathbf{e}}_k^F\big(1 \pm \operatorname{sgn}(\tilde{\lambda}_k^F)(1 - \phi_k(1 - |\nu_k^F|))\big)\Big]_{i+\frac{1}{2},j},$$

$$\mathbf{g}_{i,j+\frac{1}{2}}^{\pm} = \frac{1}{2}\sum_{k=1}^{p}\Big[\tilde{\beta}_k^G\tilde{\mathbf{e}}_k^G\big(1 \pm \operatorname{sgn}(\tilde{\lambda}_k^G)(1 - \phi_k(1 - |\nu_k^G|))\big)\Big]_{i,j+\frac{1}{2}}.$$

Για τις εξισώσεις SW έχουμε ότι

$$\tilde{\beta}_1 = \frac{\tilde{c}\Delta B}{2}, \quad \tilde{\beta}_2 = 0 \quad \text{και} \quad \tilde{\beta}_3 = -\frac{\tilde{c}\Delta B}{2}$$

και για το $\mathbf{F}$ και για το $\mathbf{G}$. Με την παραπάνω προσέγγιση του πηγαίου όρου, το σχήμα ικανοποιεί τη C-Ιδιότητα.

3.3.2 Σχήμα MacCormack

Στην παρούσα παράγραφο παρουσιάζεται το σχήμα MacCormack για τις δύο διαστάσεις. Θα βασιστούμε στο αντίστοιχο σχήμα της μίας διάστασης στο οποίο είχαμε χρησιμοποιήσει περιοριστές-ροής και το οποίο έχει την δυνατότητα να ικανοποιεί τη C-Ιδιότητα.

$$\mathbf{w}_{i,j}^{n+1} = \mathbf{w}_{i,j}^{(2)} + \big(\mathbf{D}^F_{i+\frac{1}{2},j} - \mathbf{D}^F_{i-\frac{1}{2},j}\big) + \big(\mathbf{D}^G_{i,j+\frac{1}{2}} - \mathbf{D}^G_{i,j-\frac{1}{2}}\big) \tag{3.3.5}$$

με

$$\mathbf{D}^F_{i+\frac{1}{2},j} = \frac{\Delta t}{2\Delta x}\sum_{k=1}^{p}\big[(\alpha_k^F|\lambda_k^F| - \beta_k^F \mathrm{sgn}(\lambda_k^F))(1-|\nu_k^F|)(1-\phi(\theta_k^F))\mathbf{e}_k^F\big]$$

$$\mathbf{D}^G_{i,j+\frac{1}{2}} = \frac{\Delta t}{2\Delta y}\sum_{k=1}^{p}\big[(\alpha_k^G|\lambda_k^G| - \beta_k^G \mathrm{sgn}(\lambda_k^G))(1-|\nu_k^G|)(1-\phi(\theta_k^G))\mathbf{e}_k^G\big]$$

Το ϕ μπορεί να είναι οποιοσδήποτε περιοριστής-ροής του Πίνακα 3.1. Οι τιμές των λ_k και $\mathbf{e}_k$ είναι οι ιδιοτιμές και τα ιδιοδιανύσματα των Ιακωβιανών πινάκων $\mathbf{A}$ και $\mathbf{B}$ και α_k είναι τα wave strengths. Οι πίνακες $\mathbf{A}$ και $\mathbf{B}$ είναι

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ c^2-u^2 & 2u & 0 \\ -uv & v & u \end{bmatrix} \quad \text{και} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ c^2-v^2 & 0 & 2v \end{bmatrix}$$

όπου $c = \sqrt{gh}$. Οι ιδιοτιμές για τον πίνακα $\mathbf{A}$ είναι

$$\lambda_1^F = u - c, \quad \lambda_2^F = u \quad \text{και} \quad \lambda_3^F = u + c,$$

και για τον $\mathbf{B}$

$$\lambda_1^G = v - c, \quad \lambda_2^G = v \quad \text{και} \quad \lambda_3^G = v + c.$$

Τα αντίστοιχα ιδιοδιανύσματα είναι, για τον $\mathbf{A}$

$$\mathbf{e}_1^F = \begin{bmatrix} 1 \\ u-c \\ v \end{bmatrix}, \quad \mathbf{e}_2^F = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{και} \quad \mathbf{e}_1^F = \begin{bmatrix} 1 \\ u+c \\ v \end{bmatrix},$$

και για τον $\mathbf{B}$

$$\mathbf{e}_1^G = \begin{bmatrix} 1 \\ u \\ v-c \end{bmatrix}, \quad \mathbf{e}_2^G = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{και} \quad \mathbf{e}_1^G = \begin{bmatrix} 1 \\ u \\ v+c \end{bmatrix}.$$

Τα wave strengths για τον $\mathbf{A}$ είναι

$$\alpha_{1,3}^F = \frac{1}{2}\Delta h \pm \frac{1}{2c}\big(u\Delta h - \Delta(uh)\big) \quad \text{και} \quad \alpha_2^F = \frac{1}{c}\big(\Delta(vh) - v\Delta h\big),$$

όπου $\Delta\mathbf{w} = \mathbf{w}_{i+\frac{1}{2},j} - \mathbf{w}_{i,j}$, και για τον $\mathbf{B}$

$$\alpha^G_{1,3} = \frac{1}{2}\Delta h \pm \frac{1}{2c}\big(v\Delta h - \Delta(vh)\big) \;\text{ και }\; \alpha^G_2 = \frac{1}{c}\big(\Delta(uh) - u\Delta h\big),$$

όπου $\Delta\mathbf{w} = \mathbf{w}_{i,j+\frac{1}{2}} - \mathbf{w}_{i,j}$. Ακόμα έχουμε ότι

$$\beta^F_1 = \frac{1}{2c}(B_{i+1,j} - B_{i,j}),\;\; \beta^F_2 = 0 \;\text{ και }\; \beta^F_3 = -\frac{1}{2c}(B_{i+1,j} - B_{i,j}),$$

$$\beta^G_1 = \frac{1}{2c}(B_{i,j+1} - B_{i,j}),\;\; \beta^G_2 = 0 \;\text{ και }\; \beta^G_3 = -\frac{1}{2c}(B_{i,j+1} - B_{i,j}),$$

$$\nu^F_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}^F_k,\;\; \theta^F_k = \frac{(\tilde{\alpha}^F_k)_{I+\frac{1}{2},j}}{(\tilde{\alpha}^F_k)_{i+\frac{1}{2},j}},\;\; I = i - \operatorname{sgn}\big((\nu^F_k)_{i+\frac{1}{2},j}\big),$$

$$\nu^G_k = \frac{\Delta t}{\Delta x}\tilde{\lambda}^G_k,\;\; \theta^G_k = \frac{(\tilde{\alpha}^G_k)_{i,J+\frac{1}{2}}}{(\tilde{\alpha}^G_k)_{i,j+\frac{1}{2}}},\;\; J = j - \operatorname{sgn}\big((\nu^G_k)_{i,j+\frac{1}{2}}\big).$$

Το $\mathbf{w}^{(2)}_{i,j}$ είναι μία αρχική αριθμητική προσέγγιση η οποία προκύπτει από το σχήμα

$$\mathbf{w}^{(2)}_{i,j} = \frac{1}{2}\big(\mathbf{w}^n_{i,j} + \mathbf{w}^{(1)}_{i,j}\big) - \frac{\Delta t}{\Delta x}\big(\mathbf{F}^{(1)}_{i+1,j} - \mathbf{F}^{(1)}_{i,j}\big) - \frac{\Delta t}{\Delta y}\big(\mathbf{G}^{(1)}_{i,j+1} - \mathbf{G}^{(1)}_{i,j}\big) + \Delta t\mathbf{R}^{(1)}_{i,j},$$

όπου

$$\mathbf{w}^{(1)}_{i,j} = \mathbf{w}^n_{i,j} - \frac{\Delta t}{\Delta x}\big(\mathbf{F}^n_{i+1,j} - \mathbf{F}^n_{i,j}\big) - \frac{\Delta t}{\Delta y}\big(\mathbf{G}^n_{i,j+1} - \mathbf{G}^n_{i,j}\big) + \Delta t\mathbf{R}^n_{i,j}.$$

Ως προσέγγιση του πηγαίου όρου επιλέγουμε τον

$$\mathbf{R}^n_{i,j} = \begin{bmatrix} 0 \\ -g(2\Delta x)^{-1}h^n_{i,j}(B_{i+1,j} - B_{i-1,j}) \\ -g(2\Delta y)^{-1}h^n_{i,j}(B_{i,j+1} - B_{i,j-1}) \end{bmatrix}.$$

3.3.3 Η συνθήκη CFL για τις δύο διαστάσεις

Σε πλήρη αντιστοιχία με τη σχέση 3.1.12, ο αριθμός Courant για τις δύο διαστάσεις ορίζεται ως

$$\nu = \frac{\max\big(|\lambda^F|, |\lambda^G|\big)\Delta t}{\min(\Delta x, \Delta y)}, \tag{3.3.6}$$

οπότε το χρονικό βήμα γίνεται

$$\Delta t = \frac{\nu \min(\Delta x, \Delta y)}{\max_{i,j}\big(|\lambda^F|, |\lambda^G|\big)} \tag{3.3.7}$$

Για τα σχήματα Roe και MacCormack που αναφέρθηκαν παραπάνω ο αριθμός CFL πρέπει να είναι μικρότερος του 1 και του 0.5 αντίστοιχα.

Κεφάλαιο 4

Προβλήματα στις δύο διαστάσεις

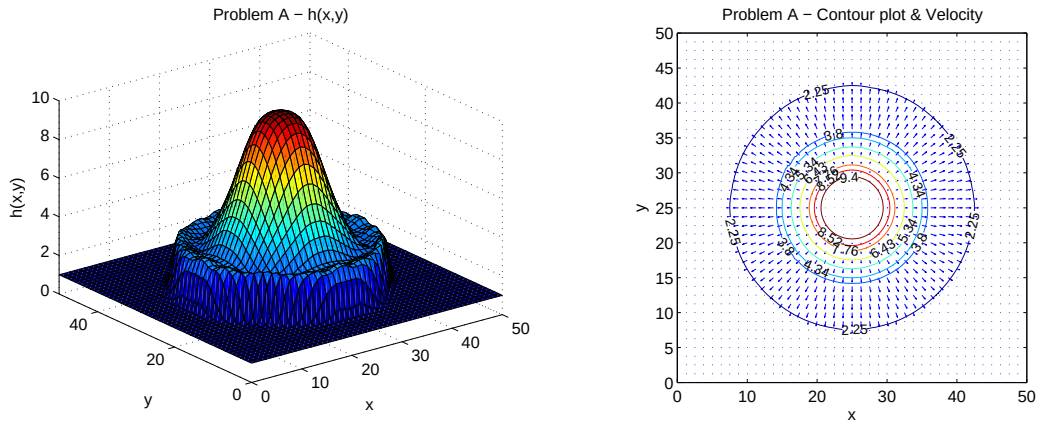
Στο κεφάλαιο αυτό γίνεται μία εκτενής παρουσίαση οκτώ δοκιμαστικών προβλημάτων στις δύο διαστάσεις. Στα προβλήματα αυτά στη συνέχεια θα δοκιμάσουμε και θα μετρήσουμε τους παράλληλους αλγόριθμους που αποτελεί και το ουσιαστικότερο κομμάτι της παρούσας εργασίας.

4.1 Πρόβλημα Α - Κατάρρευση κυκλικού φράγματος

Το πρόβλημα αυτό παρουσιάστηκε από τους Alcrudo και Garcia-Navarro [2]. Θεωρούμε μία τετράγωνη περιοχή διαστάσεων $50\,\mathrm{m} \times 50\,\mathrm{m}$. Στο κέντρο αυτής υπάρχει κυκλικό φράγμα ακτίνας $11\,\mathrm{m}$. Το νερό μέσα στο φράγμα έχει ύψος $10m$ και έξω από αυτό το ύψος του είναι $1\,\mathrm{m}$.

$$h(x,y,0) = \begin{cases} 10 & \text{αν } (x-5)^2 + (y-5)^2 \leq 11^2 \\ 1 & \text{αλλιώς} \end{cases}$$

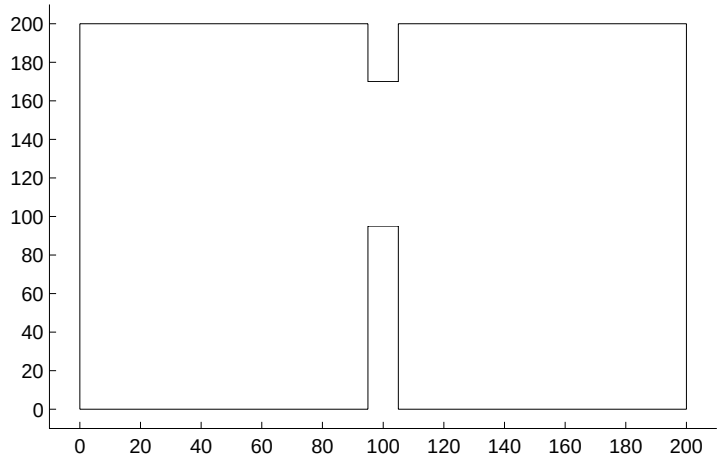
Ακαριαία το κυλινδρικό τείχος καταρρέει και το νερό αρχίζει να κινείται. Το πρόβλημα αυτό έχει σκοπό να δοκιμάσει τη συμμετρικότητα του αριθμητικού σχήματος. Η αριθμητική λύση του προβλήματος φαίνεται στο Σχήμα 4.1. Για την επίλυσή του χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid 100×100 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.



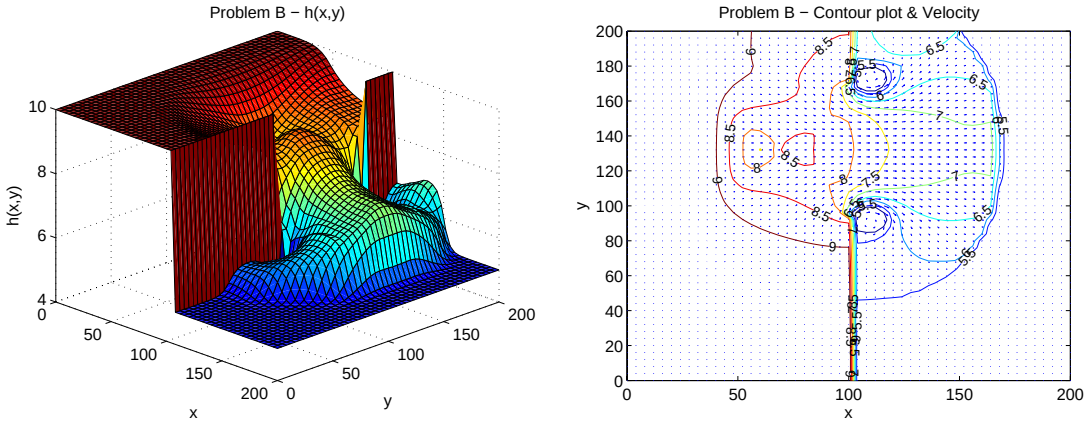
Σχήμα 4.1: Πρόβλημα Α: Αριθμητική λύση για το χρόνο $t = 0.69\,\mathrm{sec}$.

4.2 Πρόβλημα Β - Μερική κατάρρευση φράγματος

Το πρόβλημα αυτό είναι ένα παράδειγμα που χρησιμοποιήθηκε αρχικά από τους Fennema και Chaudhry [12]. Θεωρούμε μία τετράγωνη περιοχή $200\,\mathrm{m} \times 200\,\mathrm{m}$. Τα τείχη που οριοθετούν την περιοχή αυτή είναι αδιαπέραστα οπότε παρουσιάζεται το φαινόμενο της ανάκλασης όταν το νερό προσκρούσει σε αυτά. Στο κέντρο $x = 100\,\mathrm{m}$ υπάρχει τείχος $200\,\mathrm{m}$ παράλληλο στον άξονα των y. Το πάχος του τείχους αυτού θα το θεωρήσουμε αμελητέο. Αριστερά του φράγματος το νερό έχει ύψος $10\,\mathrm{m}$ και δεξιά $5\,\mathrm{m}$. Στο σημείο από $y = 100\,\mathrm{m}$ έως $y = 175\,\mathrm{m}$ το φράγμα καταρρέει. Μία παρουσίαση του φράγματος φαίνεται στο Σχήμα 4.2, καθώς επίσης η αριθμητική λύση του προβλήματος φαίνεται στο Σχήμα 4.3.



Σχήμα 4.2: Πρόβλημα Β: Παρουσίαση τειχών μετά την κατάρρευση.

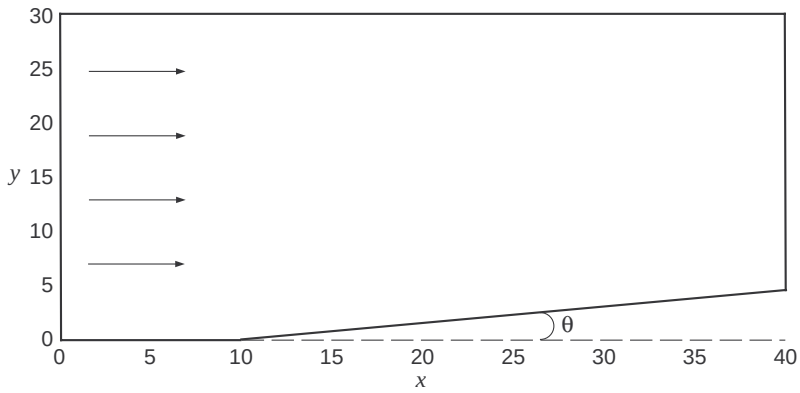


Σχήμα 4.3: Πρόβλημα B: Αριθμητική λύση για το χρόνο $t = 7.2\,\mathrm{sec}$.

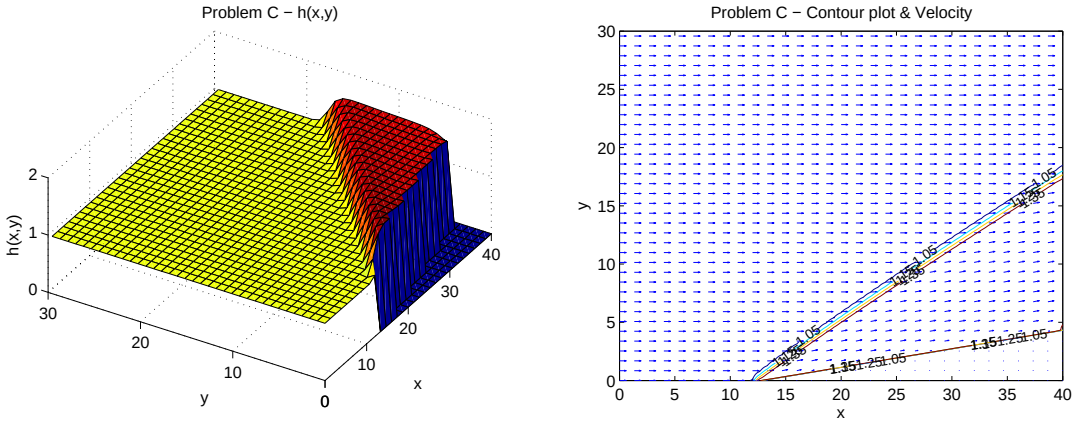
Για την επίλυσή του χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid 100×100 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.

4.3 Πρόβλημα C - Κανάλι με τείχος υπό κλίση

Θεωρούμε ένα κανάλι μήκους $40\,\mathrm{m}$ και πλάτους $30\,\mathrm{m}$. Στο σημείο $x = 10\,\mathrm{m}$ παρουσιάζεται κλίση γωνίας $\theta = 8.95°$. Το νερό αρχικά έχει ύψος $1\,\mathrm{m}$ και η είσοδος του νερού είναι συνεχής και με ταχύτητα $8.57\,\mathrm{m/s}$. Το χωρίο στο οποίο λαμβάνει μέρος το φαινόμενο παρουσιάζεται στο Σχήμα 4.4. Το σύστημα μετά από κάποιο χρόνο έρχεται σε ισορροπία όπως φαίνεται στην αριθμητική λύση στο Σχήμα 4.5. Για την επίλυσή του χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid 60×280 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.



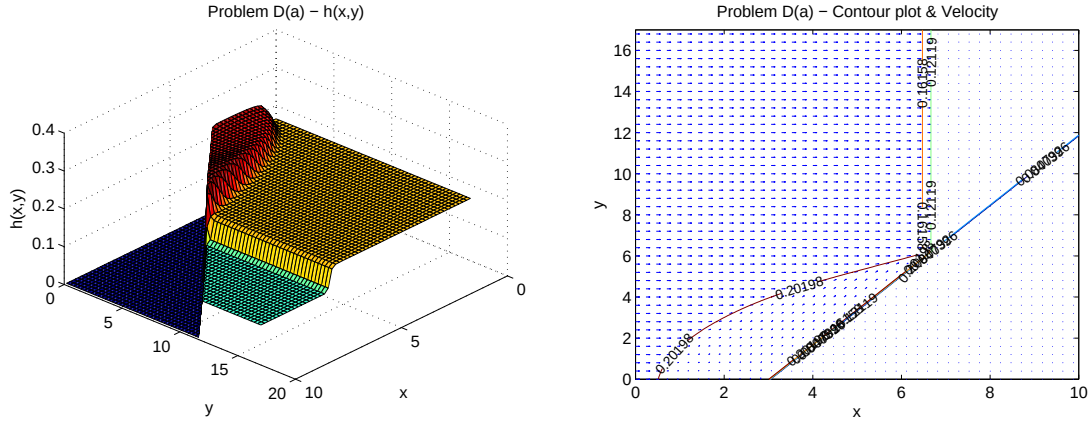
Σχήμα 4.4: Πρόβλημα C: Παρουσίαση καναλιού με τείχος υπό κλίση.



Σχήμα 4.5: Πρόβλημα C: Αριθμητική λύση για το χρόνο $t = 10\,\mathrm{sec}$.

4.4 Πρόβλημα D - Mach Bores [33]

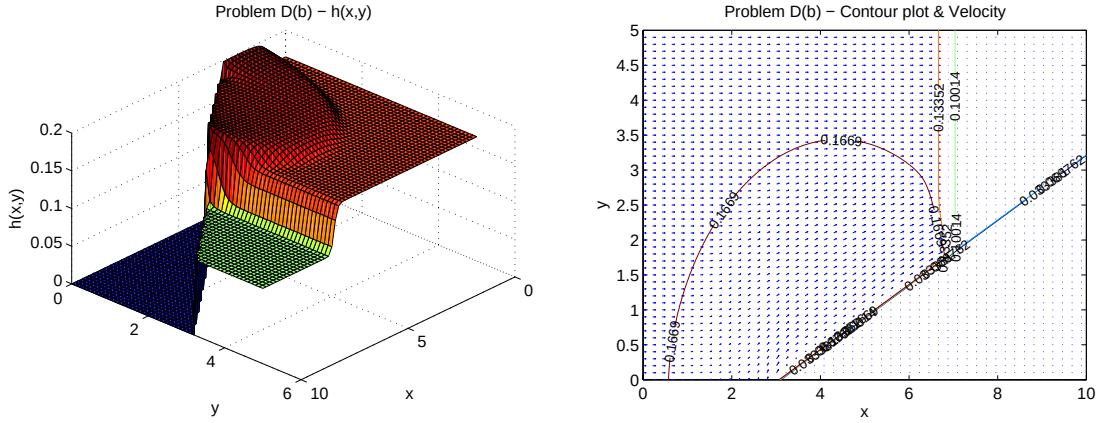
Το δοκιμαστικό πρόβλημα αυτό εκτελείται σε ένα κανάλι μήκους $10\,\mathrm{m}$ και πλάτους $5\,\mathrm{m}$ για την περίπτωση (α) και $17\,\mathrm{m}$ για την περίπτωση (β). Στο σημείο $x = 3\,\mathrm{m}$, στη μία πλευρά του καναλιού παρουσιάζεται αντίστοιχα με το Πρόβλημα C κλίση του τοιχώματος. Η γωνία είναι $\theta_1 = 60^\circ$ για την πρώτη περίπτωση και $\theta_2 = 25^\circ$ για τη δεύτερη. Στο χρόνο $t = 0\,\mathrm{sec}$, αριστερά από το σημείο $x = 1.5\,\mathrm{m}$ το νερό έχει ύψος $h_0 = 0.153\,\mathrm{m}$ και ταχύτητα $\frac{2}{3}\,\mathrm{m/sec}$ με συνεχή ροή, ενώ δεξιά από αυτό, το νερό έχει ύψος $h_1 = 0.1\,\mathrm{m}$ και μηδενική ταχύτητα. Το φαινόμενο εξελίσεται και η αριθμητική του λύση για τη χρονική στιγμή $t = 3.5\,\mathrm{sec}$ για την περίπτωση (α) παρουσιάζεται στο Σχήμα 4.6, ενώ για την περίπτωση (β) η λύση φαίνεται στο Σχήμα 4.7.



Σχήμα 4.6: Πρόβλημα D(a): Αριθμητική λύση για το χρόνο $t = 3.5\,\mathrm{sec}$.

Για την επίλυσή του χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid

170×170 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.



Σχήμα 4.7: Πρόβλημα D(b): Αριθμητική λύση για το χρόνο $t = 3.5\,\mathrm{sec}$.

Ομοίως χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid 186×200 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.

Εν γένει υπάρχουν τρεις πιθανές ανακλάσεις κυμάτων κατά την πρόσκρουσή τους σε τείχος υπό γωνία.

1. Η *κανονική ανάκλαση* για γωνίες θ μεγαλύτερες από $35^\circ - 45^\circ$·

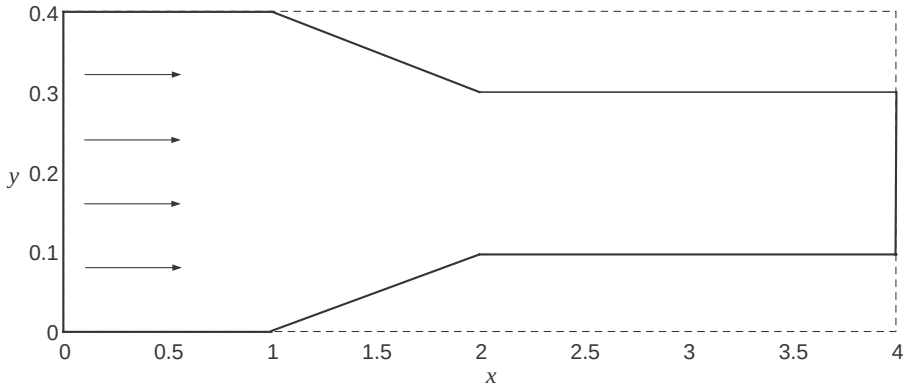
2. Η *ανάκλαση Mach* για γωνίες θ μεγαλύτερες από 20° και μικρότερες από $40^\circ - 45^\circ$·

3. Η *ειδική περίπτωση ανάκλασης Mach* για γωνίες θ μικρότερες από 20°.

Για την περίπτωσή μας, το πρώτο πρόβλημα αντιστοιχεί στην πρώτη κατηγορία ανακλάσεων και το δεύτερο αντιστοιχεί στην δεύτερη.

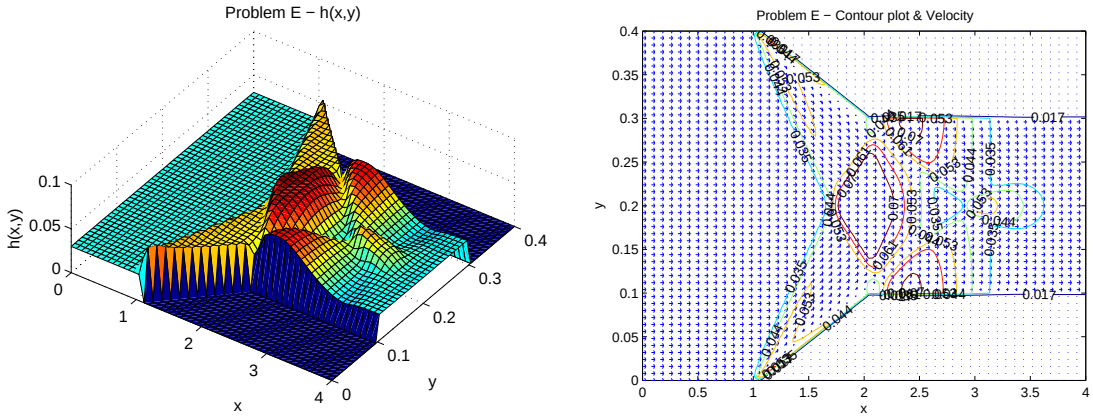
4.5 Πρόβλημα Ε - Κανάλι που στενεύει

Το πρόβλημα αυτό παρουσιάστηκε από τον Rao στο [30]. Αποτελεί από μία πιο σύνθετη εκδοχή του Προβλήματος C. Ένα κανάλι αρχικού πλάτους $0.4\,\mathrm{m}$ και μήκους $4\,\mathrm{m}$ στενεύει στο σημείο $x = 1\,\mathrm{m}$ έως το σημείο $x = 2\,\mathrm{m}$ και το πλάτος του μεταβάλεται και γίνεται $x = 0.2\,\mathrm{m}$. Στο Σχήμα 4.8 παρουσιάζεται ένα σχεδιάγραμμα της τοπολογίας του καναλιού.



Σχήμα 4.8: Πρόβλημα E: Παρουσίαση καναλιού που στενεύει.

Για $t = 0\,\text{sec}$ το ύψος του νερού είναι $h_0 = 0.03\,\text{m}$. Το υγρό εισέρχεται συνεχώς με ταχύτητα $3.3\,\text{m/sec}$. Βασικός σκοπός του δοκιμαστικού αυτού προβλήματος είναι να εξετάσει κατά πόσο το αριθμητικό μας σχήμα μπορεί να διατηρήσει τη συμμετρία. Μία αριθμητική λύση του προβλήματος για τη χρονική στιγμή $t = 0.5\,\text{sec}$ φαίνεται στο Σχήμα 4.9 όπως αυτή προέκυψε από το αριθμητικό σχήμα του Roe σε ένα grid 100×100 με περιοριστή-ροής τον Superbee και $\text{CFL} = 0.4$.



Σχήμα 4.9: Πρόβλημα E: Αριθμητική λύση για το χρόνο $t = 0.5\,\text{sec}$.

4.6 Πρόβλημα F - Διάδοση κύματος σε τοπογραφία

Το πρόβλημα αυτό αποτελείται από έναν πυθμένα με τοπογραφία και ένα κύμα το οποίο κινείται κατά μήκος του καναλιού διαστάσεων $1\,\text{m} \times 1\,\text{m}$. Η εισροή και η εκροή του υγρού στο κανάλι είναι ελεύθερη. Σκοπός του προβλήματος αυτού είναι να δοκιμάσει τα αριθμητικά σχήματα στην ακρίβειά τους όσο αφορά πολύ μικρές μεταβολές. Παρουσιάστηκε από τον LeVeque [20] και τους Hubbard και

Garcia-Navarro [17]. Πιο συγκεκριμένα, οι αρχικές συνθήκες του προβλήματος είναι

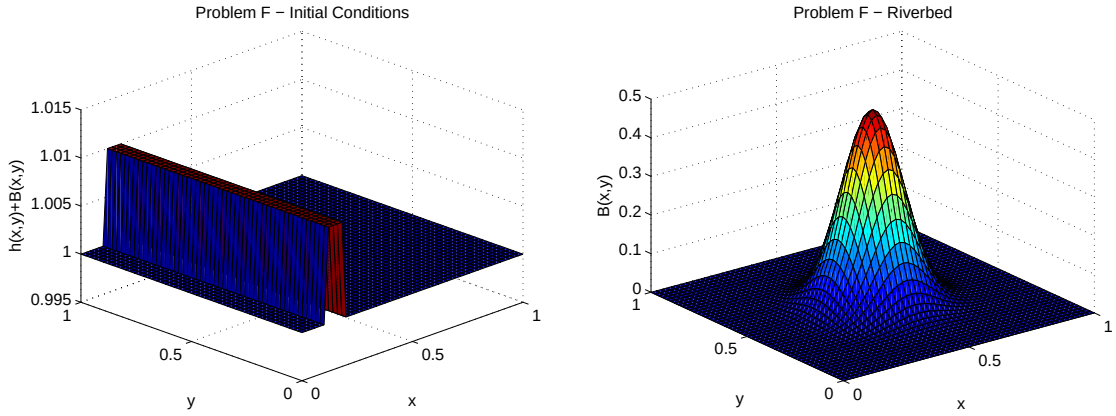
$$h(x,y,0) = \begin{cases} 1.01 - B(x,y) & \text{αν } \ 0.1 < x < 0.2, \ \ 0 \le y \le 1 \\ 1 - B(x,y) & \text{αλλιώς} \end{cases}$$

$$u(x,y,0) = 0, \quad v(x,y,0) = 0$$

Η τοπογραφία του πυθμένα περιγράφεται από τη συνάρτηση

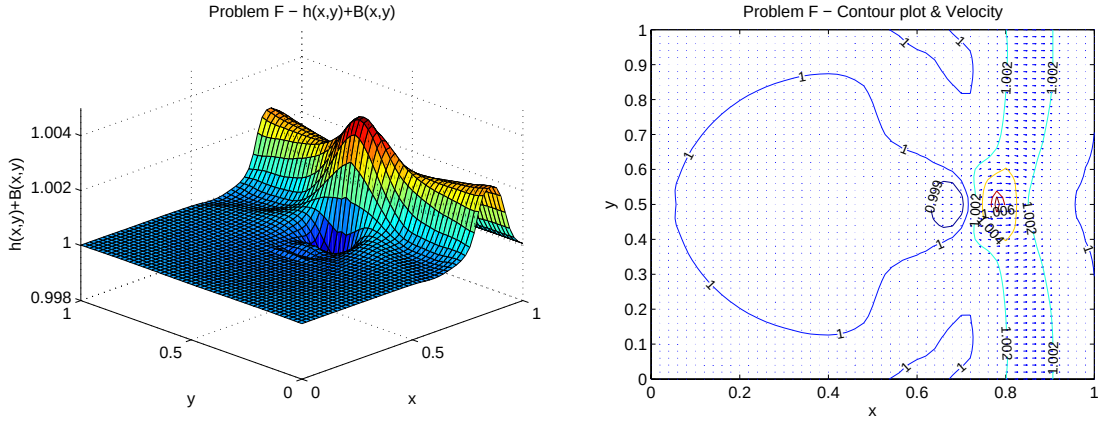
$$B(x,y) = 0.5 \exp\big(-50\big((x-0.5)^2 + (y-0.5)^2\big)\big),$$

όπως φαίνονται στο Σχήμα 4.10.



Σχήμα 4.10: Πρόβλημα F: Αρχικές συνθήκες.

Η επιτάχυνση της βαρύτητας για το πρόβλημα αυτό ορίζεται $g = 1\,\mathrm{m/sec^2}$. Η αριθμητική του λύση για το χρόνο $t = 0.7\,\mathrm{sec}$ φαίνεται στο Σχήμα 4.14. Για την επίλυσή του χρησιμοποιήθηκε το αριθμητικό σχήμα του Roe σε ένα grid 100×100 με περιοριστή-ροής τον Superbee και $\mathrm{CFL} = 0.4$.



Σχήμα 4.11: Πρόβλημα F: Αριθμητική λύση για το χρόνο $t = 0.7\,\mathrm{sec}$.

4.7 Πρόβλημα G - Ανακλάσεις κύματος σε τοπογραφία

Το παρόν πρόβλημα αποτελεί παραλλαγή του προηγούμενου προβλήματος. Πραγματοποιείται σε ένα κλειστό χωρίο διαστάσεων $2000\,\mathrm{m} \times 2000\,\mathrm{m}$. Τα τείχη ανακλούν τα προσπίπτοντα κύματα. Η επιτάχυνση της βαρύτητας για το πρόβλημα αυτό ορίζεται $g = 1\,\mathrm{m/sec^2}$. Το φαινόμενο αφήνεται να εξελιχθεί για 15 λεπτά. Για την περιγραφή των αρχικών συνθηκών έχουμε ότι

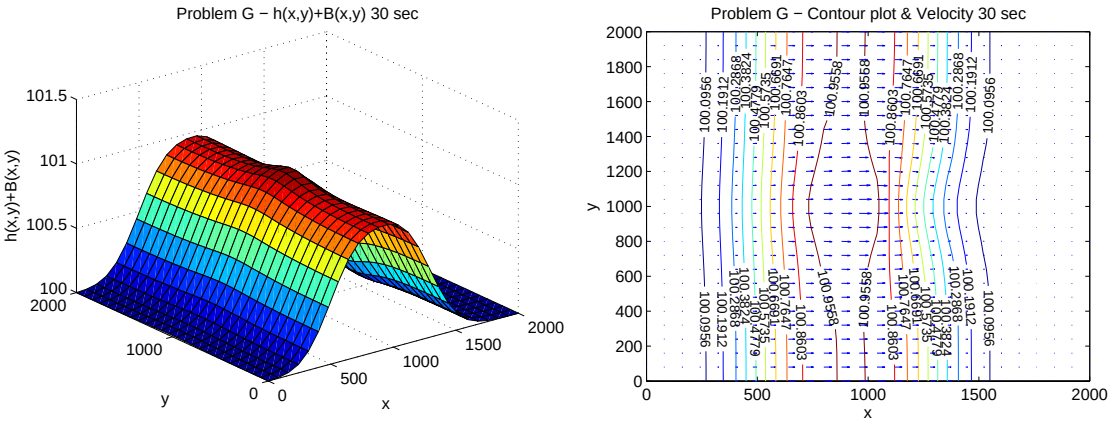
$$h(x,y,0) = \begin{cases} 110 - B(x,y) & \text{αν } \ 100 < x < 200, \ \ 0 \le y \le 2000 \\ 100 - B(x,y) & \text{αλλιώς} \end{cases}$$

$$u(x,y,0) = 0, \quad v(x,y,0) = 0$$

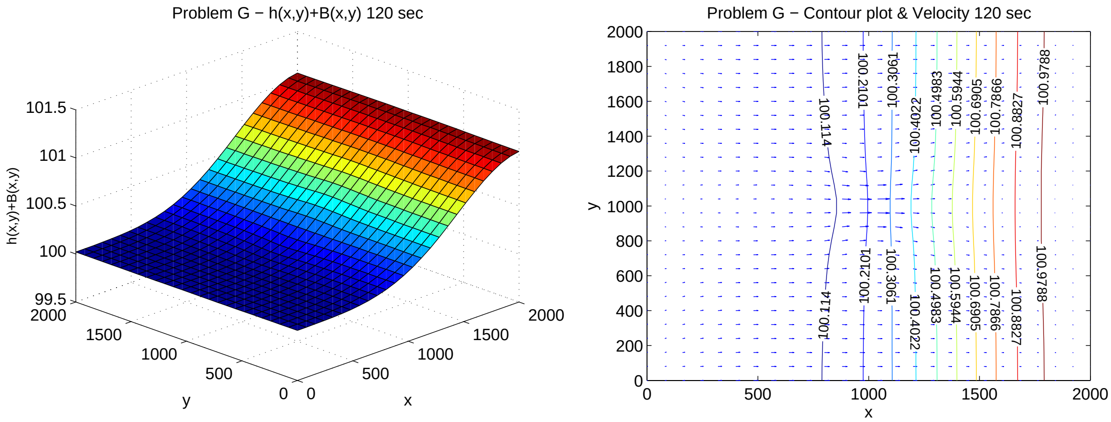
Η τοπογραφία του πυθμένα περιγράφεται από τη συνάρτηση

$$B(x,y) = 50\exp\bigl(-0.00002\bigl((x-1000)^2 + (y-1000)^2\bigr)\bigr).$$

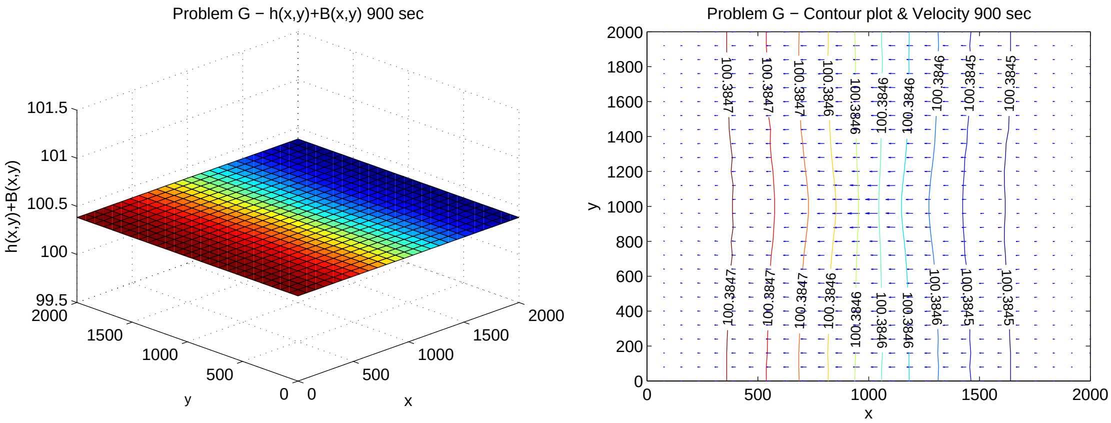
Η περιγραφή αποτελεί ένα ρεαλιστικό πρόβλημα το οποίο σκοπό έχει την αποδοτικότητα και ταυτόχρονα την χρησιμότητα των παράλληλων υπολογισμών.



Σχήμα 4.12: Πρόβλημα G: Αριθμητική λύση για το χρόνο $t = 30\,\mathrm{sec}$.



Σχήμα 4.13: Πρόβλημα G: Αριθμητική λύση για το χρόνο $t = 120\,\mathrm{sec}$.



Σχήμα 4.14: Πρόβλημα G: Αριθμητική λύση για το χρόνο $t = 900\,\mathrm{sec}$.

Στα παραπάνω σχήματα παρουσιάζεται η εξέλιξη του φαινομένου στα 30, 120 και $900\,\mathrm{sec}$ όπως αυτή προκύπτει απο το αριθμητικό σχήμα του Roe σε ένα grid 50×50 με περιοριστή-ροής τον Minmod και $\mathrm{CFL} = 0.7$. Μετά από 15 λεπτά το νερό έχει επέρθει σε ένα σχεδόν ομοιόμορφο ύψος (στάσιμη κατάσταση).

Κεφάλαιο 5

Παράλληλοι Υπολογισμοί

Έχοντας θέσει ως βασικότερο θέμα στην παρούσα εργασία την επίλυση προβλημάτων SW σε παράλληλες αρχιτεκτονικές, κάνοντας χρήση των σχημάτων όπως αυτά παρουσιάστηκαν στις προηγούμενες ενότητες, θεωρείται σκόπιμο να γίνει μία βασική εισαγωγή στους Παράλληλους Επιστημονικούς Υπολογισμούς.

Παραδοσιακά ο εκάστοτε αλγόριθμος αναπτύσσεται σε σειριακή μορφή, είτε ο αλγόριθμος αυτός έχει να κάνει με επιστημονικούς υπολογισμούς, όπως στην περίπτωσή μας, είτε αφορά ανάπτυξη λογισμικού ευρείας χρήσης. Οι αλγόριθμοι αυτοί αφού αναπτυχθούν μέσω κάποιας γλώσσας προγραμματισμού θα εκτελεστούν σε *έναν* υπολογιστή, με *μία* Κεντρική Μονάδα Επεξεργασίας (CPU). Το λογισμικό εκτελεί μία διεργασία η οποία αποτελείται από μία αλληλουχία εντολών-οδηγιών εκτελουμένων η μία μετά την άλλη. Μόνο μία οδηγία μπορεί να εκτελεστεί σε κάθε χρονική στιγμή.

Ο πιο απλός τρόπος για να αντιληφθεί κανείς την έννοια του παράλληλου υπολογισμού είναι να τη σκεφτεί ως την *ταυτόχρονη* χρήση *πολλών* υπολογιστικών πόρων για την επίλυση ενός υπολογιστικού προβλήματος. Οι υπολογιστικοί πόροι μπορούν να αποτελούνται από

- έναν υπολογιστή με πολλούς επεξεργαστές·
- πολλούς υπολογιστές συνδεμένους σε ένα δίκτυο·
- συνδυασμό και των δύο.

Το κατά πόσο μπορεί, ένα συγκεκριμένο πρόβλημα, να παραλληλοποιηθεί, έχει να κάνει αποκλειστικά με τα χαρακτηριστικά του. Το υπολογιστικό πρόβλημα, συνήθως, παρουσιάζει χαρακτηριστικά όπως την ικανότητα να

- «σπάει» σε διακριτά και ανεξάρτητα κομμάτια τα οποία μπορούν να λυθούν ταυτόχρονα·
- εκτελεί πολλές οδηγίες σε κάθε χρονική στιγμή·
- επιλύεται σε μικρότερο χρονικό διάστημα με τη χρήση πολλών πόρων.

Ένα απλό ερώτημα που θα μπορούσε να θέσει κάποιος είναι ο λόγος για τον οποίο χρησιμοποιούμε παράλληλους υπολογισμούς. Η απάντηση αποτελείται από δύο κύριους λόγους. Κάνοντας χρήση παράλληλων διεργασιών μπορεί να επιτευχθεί μεγάλη εξοικονόμιση του χρόνου εκτέλεσης μίας εφαρμογής καθώς επίσης αποκτάμε την δυνατότητα να επιλύουμε μεγαλύτερα προβλήματα. Πέρα όμως από τους δύο παραπάνω βασικούς λόγους υπάρχουν και πολλοί άλλοι αρκετά σημαντικοί. Μπορούμε να εκμεταλλευτούμε υπολογιστικούς πόρους οι οποίοι είναι απομακρισμένοι μέσω ενός ευρύτερου δικτύου, ή ακόμα και μέσω του διαδικτύου. Ένας ακόμη παράγοντας είναι και ο οικονομικός, μιας και αποκτάμε τη δυνατότητα χρήσης πολλών και φθηνών υπολογιστών αντί των πολλαπλάσια ακριβών υπερυπολογιστών. Επιπρόσθετα αποφεύγονται φαινόμενα περιορισμού μνήμης αφού μπορούμε να έχουμε πολλές μονάδες μνήμης χρησιμοποιώντας την μνήμη πολλών υπολογιστών. Πλέον το πυρίτιο, το μέσο με το οποίο κατασκευάζονται οι μικροεπεξεργαστές, έχει φτάσει σε πλατώ όσο αφορά τον αριθμό τρανζίστορ που υπάρχουν ανά τετραγωνική μονάδα μήκους. Όλοι οι παραπάνω λόγοι δείχνουν την αναγκαιότητα χρήσης διαδικασιών, οι οποίες βασίζονται σε παράλληλους αλγορίθμους, ώστε να επιτευχθεί η μείωση του χρόνου εκτέλεσης των εφαρμογών με ταυτόχρονη αύξηση του μεγέθους των προβλημάτων που επιλύονται.

5.1 Η ταξινόμιση του Flynn

Υπάρχουν πολλοί τρόποι για να ταξινομίσει κανείς τους παράλληλους αριθμητικούς υπολογισμούς. Η πιο γνωστή σε χρήση από το 1966 ονομάζεται *ταξινόμιση του Flynn*. Η ταξινόμιση του Flynn ορίζει τις διάφορες αρχιτεκτονικές των παράλληλων διεργασιών ανάλογα με το πως αυτές κατηγοριοποιούνται βάσει δύο ανεξάρτητων μεταβλητών, της *Οδηγίας* και του *Δεδομένου*. Κάθε μία από αυτές τις μεταβλητές μπορεί να πάρει μόνο δύο τιμές, *Ένας* και *Πολλοί*. Βάσει των παραπάνω διακρίνονται τέσσερις κατηγορίες.

SISD Μία Οδηγία, Ένα Δεδομένο. Αποτελεί την κλασική δομή ενός σειριακού αλγορίθμου. Μόνο ένα σύνολο Οδηγιών μπορεί να εκτελεστεί από τον υπολογιστή ανά ένα κύκλο του ρολογιού του επεξεργαστή του, καθώς επίσης και ένα σύνολο Δεδομένων χρησιμοποιείται κάθε φορά. Το μοντέλο αυτό είναι το αρχαιότερο και αυτό είχε τη μεγαλύτερη χρήση ως τώρα.

SIMD Μία Οδηγία, Πολλά Δεδομένα. Κάθε επεξεργαστής εκτελεί το ίδιο σύνολο Οδηγιών ανά ένα κύκλο του ρολογιού του αλλά με διαφορετικά Δεδομένα ο καθένας. Το μοντέλο αυτό είναι από τα πλέον δημοφιλή.

MISD Πολλές Οδηγίες, Ένα Δεδομένο. Πολύ λίγα παραδείγματα υπάρχουν για αυτό το μοντέλο. Ένα από αυτά είναι η χρήση πολλών κρυπτογραφικών αλγορίθμων για το «σπάσιμο» ενός κωδικοποιημένου μηνύματος.

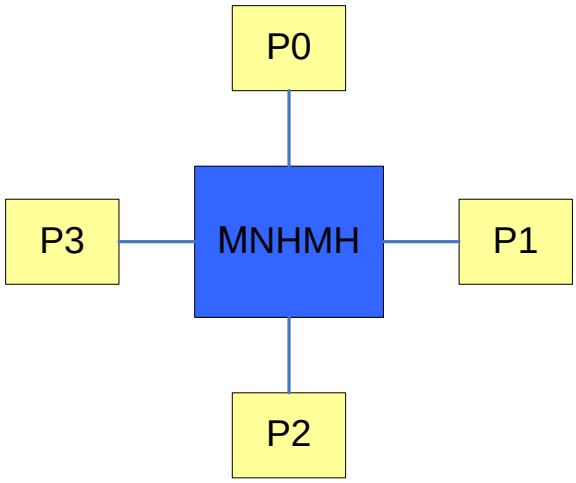
MIMD Πολλές Οδηγίες, Πολλά Δεδομένα. Πλέον διαδεδομένο μοντέλο. Διαφορετικά σύνολα Οδηγιών ανά επεξεργαστή, με διαφορετικά δεδομένα ο καθένας. Ο κάθε επεξεργαστής μπορεί να επιλύει ένα τελείως ανεξάρτητο τμήμα ενός προβλήματος, του οποίου τα τμήματα ενώνονται (συχρονίζονται) μόνο όταν αυτό είναι απολύτως απαραίτητο. Στο μοντέλο αυτό βασίζεται την εργασία μας.

5.2 Είδη μνήμης

Όπως είναι ήδη γνωστό ένα παράλληλο υπολογιστικό σύστημα χαρακτηρίζεται από το πλήθος των επεξεργαστών που διαθέτει καθώς επίσης και από τον τρόπο σύνδεσής τους. Ένα εξίσου βασικό χαρακτηριστικό αποτελεί και η αρχιτεκτονική που καθορίζει τον τρόπο πρόσβασης στη μνήμη του συστήματος, η οποία διακρίνεται σε τρεις βασικές κατηγορίες.

5.2.1 Αρχιτεκτονικές μνήμης κοινής διαχείρισης

Οι παράλληλοι υπολογιστές αρχιτεκτονικής με μνήμη κοινής διαχείρισης αποτελούν ένα μεγάλο κομμάτι του συνόλου των παράλληλων υπολογιστών. Όλοι οι επεξεργαστές λειτουργούν κάτω από ένα λειτουργικό σύστημα το οποίο επιτρέπει να έχουν πρόσβαση από κοινού στη μνήμη όπως φαίνεται και στο Σχήμα 5.1.



Σχήμα 5.1: Αρχιτεκτονική μνήμης κοινής διαχείρισης.

Ο κάθε επεξεργαστής λειτουργεί ανεξάρτητα από τους υπόλοιπους αλλά όλοι μαζί μοιράζονται την ίδια μνήμη. Κάθε αλλαγή που προκαλείται στις διευθύνσεις της μνήμης από έναν επεξεργαστή είναι ορατή από τους υπόλοιπους. Τα συστήματα μνήμης κοινής διαχείρισης χωρίζονται σε δύο κύριες κατηγορίες:

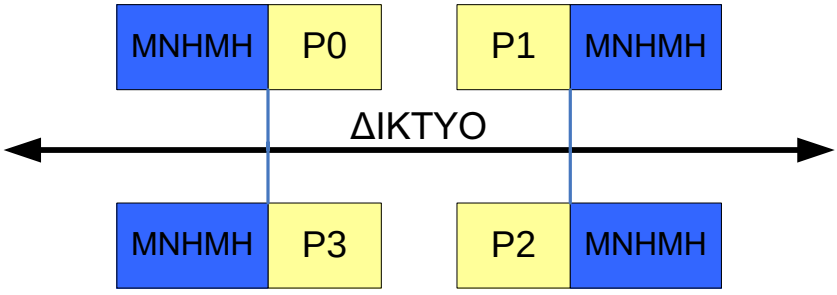
Ομοιόμορφης Πρόσβασης Μνήμης Αποτελούνται κυρίως από Συμμετρικά Πολυεπεξεργαστικά Μηχανήματα (ΣΠΜ). Τα συστήματα αυτού του τύπου έχουν όμοιους επεξεργαστές και η πρόσβαση στη μνήμη από τον καθένα γίνεται εξίσου.

Μη-Ομοιόμορφης Πρόσβασης Μνήμης Συνήθως αποτελούνται από φυσική ένωση δύο ή περισσοτέρων ΣΠΜ. Κάθε ΣΠΜ έχει άμμεση πρόσβαση στη μνήμη του άλλου. Δεν έχουν όλοι οι επεξεργαστές ίσους χρόνους πρόσβασης σε όλες τις μνήμες. Προφανώς η πρόσβαση στη μνήμη μέσω της φυσικής σύνδεσης είναι μικρότερης ταχύτητας.

Η αρχιτεκτονική αυτή παρουσιάζει πλεονεκτήματα και μειονεκτήματα. Αναφορικά στα πλεονεκτήματα πρέπει να τονιστεί η ευκολία που έχει ο χρήστης στο να προγραμματίσει τέτοιου είδους συστήματα καθώς επίσης και στο γεγονός ότι η ανταλλαγή δεδομένων είναι γρήγορη και ομοιογενής. Σε αντίθεση, στα μειονεκτήματα πρέπει να περιλάβει κανείς το κόστος, μιας και η σχεδίαση τέτοιων συστημάτων είναι ιδιαίτερα ακριβή. Επιπρόσθετα, κατά την προσθήκη νέων επεξεργαστών η κίνηση στο κανάλι επικοινωνίας μεταξύ CPU και μνήμης αυξάνεται γεωμετρικά.

5.2.2 Αρχιτεκτονική μνήμης μη-κοινής διαχείρισης

Συνεχίζοντας με τις αρχιτεκτονικές των συστημάτων σχετικά με την πρόσβαση στη μνήμη σειρά έχει αυτή της μη-κοινής διαχείρισης. Σε αυτόν τον τύπο, ο κάθε επεξεργαστής έχει πρόσβαση μέσω του λειτουργικού συστήματος στη δική του μνήμη άμεσης πρόσβασης. Μία διάταξη ενός τέτοιου πολυεπεξεργαστικού συστήματος φαίνεται στο Σχήμα 5.2.



Σχήμα 5.2: Μνήμη μη-κοινής διαχείρισης.

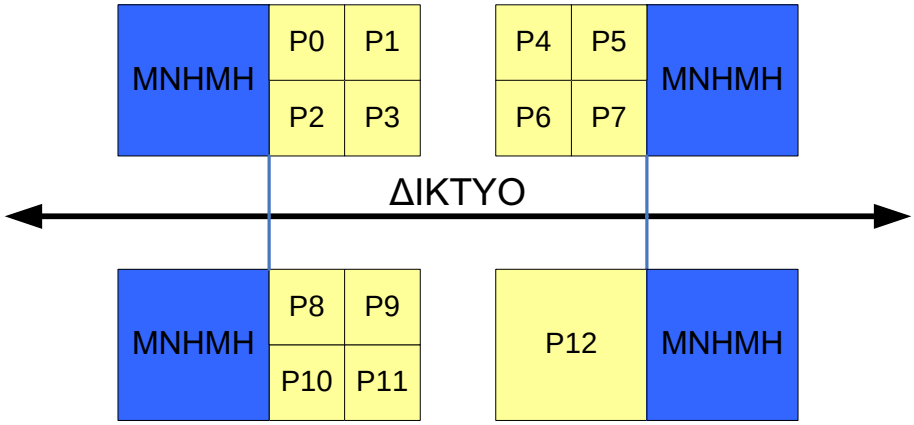
Κάθε επεξεργαστής έχει τη δική του αποκλειστική μνήμη. Οι διευθύνσεις της μνήμης είναι τοπικές και όχι γενικές για όλους τους επεξεργαστές. Μιας και ο κάθε επεξεργαστής έχει τη δική του μνήμη, δρα ανεξάρτητα με αποτέλεσμα

οι μεταβολές που προκαλούνται στη δική του μνήμη να μην επηρεάζουν τους υπόλοιπους. Όταν ένας επεξεργαστής χρειαστεί δεδομένα από τη μνήμη κάποιου άλλου, ο προγραμματιστής θα πρέπει να ορίσει με υψηλή ακρίβεια το πως θα πραγματοποιηθεί η επικοινωνία, αφού το λειτουργικό σύστημα κάθε κεντρικής μονάδας επεξεργασίας είναι ανεξάρτητο από των υπολοίπων οπότε δεν είναι εφικτή η πραγματοποίηση αυτής της διαδικασίας μέσω του λειτουργικού συστήματος μόνο. Το υλικό από το οποίο είναι κατασκευασμένο το δίκτυο ποικίλλει και μπορεί να είναι τόσο απλό όσο ένα συνηθισμένο δίκτυο Ethernet.

Στα πλεονεκτήματα της αρχιτεκτονικής αυτής πρέπει να τονίσουμε το γεγονός ότι η τυχούσα αύξηση του αριθμού των επεξεργαστών αυξάνει και το ποσό της μνήμης αναλογικά. Ο κάθε επεξεργαστής έχει ταχύτατη πρόσβαση στην μνήμη του. Αντιθέτως στα αρνητικά το κυριότερο είναι η ευθύνη του προγραμματιστή στο συγχρονισμό και τη διαχείριση της επικοινωνίας μεταξύ των επεξεργαστών. Επίσης αρκετές φορές είναι δύσκολο να χωριστούν δομές δεδομένων ολικής διευθυνσιολόγησης σε δομές τοπικής διευθυνσιολόγησης.

5.2.3 Αρχιτεκτονική υβριδικής μνήμης

Τέλος θα αναφέρουμε τα συστήματα με υβριδική μνήμη. Αποτελούν συνδιασμό των δύο παραπάνω τύπων όπως παρουσιάζεται στο Σχήμα 5.3.



Σχήμα 5.3: Υβριδική μνήμη.

Τα μεγαλύτεα και ισχυρότερα υπολογιστικά συστήματα (Supercomputers) αποτελούνται από συνδυασμό μνήμης κοινής και μη-κοινής διαχείρισης. Συνήθως το τμήμα που χαρακτηρίζεται ως κοινής διαχείρισης αποτελείται από ένα ΣΠΜ. Όλα τα ΣΠΜ συνδεμένα μαζί με διάφορα ετερογενή υπολογιστικά συστήματα σε ένα δίκτυο αποτελούν ένα σύστημα μη-κοινής μνήμης. Το μοντέλο αυτό περιγράφει την αρχιτεκτονική της υβριδικής μνήμης. Προφανώς στα πλεονεκτήματα και

τα μειονεκτήματα περιλαμβάνονται αυτά που είναι κοινά και στους δύο τύπους, μη-κοινής και κοινής διαχείρισης μνήμης.

5.3 Πρότυπα υλοποίησης σε παράλληλες αρχιτεκτονικές

Στην παρούσα εργασία, για την παραλληλοποίηση των αλγορίθμων που παρουσιάσαμε στο δεύτερο κεφάλαιο, έγινε χρήση των προτύπων OpenMP και MPI. Τα πρότυπα αυτά χαρακτηρίζονται από εντελώς διαφορετική φιλοσοφία το ένα με το άλλο. Η επιλογή τους έγινε για δύο λόγους. Αρχικά γιατί είναι τα πλέον διαδεδομένα, και κατά δεύτερο λόγο γιατί θέλαμε να αναδείξουμε τις διαφορές τους και να μελετηθεί η συμπεριφορά του ίδιου αριθμητικού σχήματος όταν αυτός υλοποιηθεί μέσω παράλληλων αλγορίθμων.

5.3.1 OpenMP

Μία προσέγγιση στον προγραμματισμό συστημάτων μνήμης κοινής διαχείρισης είναι να χρησιμοποιήσουμε πιο δομημένες κατασκευές όπως παράλληλους βρόγχους ούτως ώστε να παρουσιαστούν δυνατότητες παράλληλης εκτέλεσης. Η παραπάνω προσέγγιση ακολουθείται από το δημοφιλές OpenMP, ένα σύνολο από οδηγίες του μεταγλωττιστή, βιβλιοθήκες από υποπρογράμματα και μεταβλητές περιβάλλοντος τα οποία μπορούν να χρησιμοποιηθούν για να προσδιορίσουν παραλληλοποίηση αρχιτεκτονικής κοινής μνήμης σε κώδικες προγραμμάτων Fortran και C/C++. Το OpenMP παρόλο που αποτελεί ένα σύγχρονο στάνταρ (1997), θεωρείται εξέλιξη μίας μακράς ιστορίας μοντέλων κοινής μνήμης. Κατά κύριο λόγο επικεντρώνεται στην εκμετάλλευση όσο αφορά την παραλληλοποίηση των βρόγχων. Ένα βασικό προσόν του είναι το γεγονός ότι μπορεί να επιτρέψει στον χρήστη, έαν και εφόσον αυτός το επιθυμεί, να μεταγλωττήσει και να εκτελέσει το πρόγραμμά του σε απόλυτη σειριακή μορφή. Εάν για παράδειγμα ο χρήστης μετατρέψει ένα υπάρχον πρόγραμμά του σε παράλληλο, το νέο του πρόγραμμα διατηρεί την σειριακότητά του. Το γεγονός αυτό απλοποιεί την ανάπτυξη ενός προγράμματος, του debugging του, και της συντήρησής του. Παρόλα αυτά, αν ο προγραμματιστής ορίσει έναν βρόγχο ως παράλληλο ενώ συντρέχουν λόγοι εξάρτησης, το πρόγραμμα θα παράξει εσφαλμένα αποτελέσματα, των οποίων η προέλευση είναι πολύ δύσκολο να εντοπιστεί.

Η απλότητα χρήσης του OpenMP αποτελεί το μεγάλο του προτέρημα. Η υλοποίηση του παράλληλου αλγορίθμου επιτυγχάνεται με προσθήκη εντολών για την εκτέλεση παράλληλων διαδικασιών, οι οποίες αποτελούν επιλογή του προγραμματιστή με βάση τα κριτήρια της απόδοσης.

5.3.2 MPI

Το πρότυπο MPI (Message Passing Interface) αποτελείται από ένα σύνολο από υποπρογράμματα για τη διαχείριση της διακίνησης των δεδομένων μεταξύ των επικοινωνούντων επεξεργαστών. Κυρίως χρησιμοποιείται στις γλώσσες C, Fortran και C++, αλλά πρόσφατα έχει αναπτυχθεί και για άλλους μεταγλωττιστές. Τα υποπρογράμματα του MPI επιτρέπουν την επικοινωνία δύο επεξεργαστών σημείο-προς-σημείο, διαδικασίες συλλογής δεδομένων μεταξύ των διεργασιών καθώς επίσης και παράλληλο I/O. Ο προγραμματιστής δηλαδή έχει την ευθύνη της διαχείρισης της επικοινωνίας οπότε στο MPI θα πρέπει να ορίσει το είδος των δεδομένων τα οποία συναλλάσονται, επιτρέποντας στις υλοποιήσεις που βασίζονται στο MPI να βελτιστοποιούνται για μη-συνεχή δεδομένα στη μνήμη, καθώς επίσης και για ετερογενή συστήματα.

Το MPI σήμερα αποτελεί μία κορυφαία επιλογή όσο αφορά τη αποτελεσματικότητά του. Ο κώδικας που βασίζεται σε υλοποιήσεις μέσω του MPI χαρακτηρίζεται από έναν μεγάλο βαθμό φορητότητας ο οποίος δεν μπορεί να συγκριθεί με κάποια άλλη ανάλογη τεχνολογία. Οι δυνατότητες ελέγχου που παρέχει στον προγραμματιστή έχουν ως αποτέλεσμα προγράμματα υψηλής απόδοσης. Σε αντίθεση όμως με το OpenMP, αυτό φέρει και την ανάλογη δυσκολία στην ανάπτυξη του κώδικα. Επιπρόσθετα, ο ήδη υπάρχον σειριακός κώδικας πρέπει να επανασχεδιαστεί εξ ολοκλήρου από την αρχή.

Κεφάλαιο 6

Παράλληλα διδιάστατα αριθμητικά σχήματα

Σε προηγούμενο κεφάλαιο είχαμε κάνει μία περιγραφή των σχημάτων Roe και MacCormack στις δύο διαστάσεις. Στην παρούσα ενότητα θα κατασκευάσουμε παράλληλους αλγορίθμους των παραπάνω σχημάτων, οι οποίοι θα βασίζονται στα πρότυπα OpenMP και MPI. Σκοπός μας είναι να συγκρίνουμε τα αποτελέσματα τα οποία θα προκύψουν και να γίνει μία μελέτη της συμπεριφοράς τους σε παράλληλες αρχιτεκτονικές κοινής και διανεμημένης μνήμης.

Αξίζει να αναφερθεί ότι οι σύγχρονοι μεταγλωττιστές παρέχουν τη δυνατότητα αυτοματοποιημένης παραλληλοποίησης (Automatic Parallelization Option) η οποία αρχικά εφαρμόστηκε στους σειριακούς αλγορίθμους των αριθμητικών σχημάτων. Όμως επειδή οι αλγόριθμοι περιλαμβάνουν και ανακυκλώσεις που περιέχουν μικρό πλήθος εντολών-οδηγιών, των οποίων η παραλληλοποίηση δεν είναι αποδοτική, ο αυτοματισμός έχει ως αποτέλεσμα την επιβράδυνση του χρόνου εκτέλεσης της εφαρμογής κατά την αύξηση του πλήθους των επεξεργαστών. Ενδεικτικά αναφέρουμε ότι κατά τη χρήση της αυτοματοποιημένης επιλογής παραλληλοποίησης του μεταγλωττιστή MIPSPro® της εταιρίας SGI προέκυψε επιβάρυνση 2.3% στο χρόνο εκτέλεσης του σειριακού αλγορίθμου κάνοντας χρήση 8 επεξεργαστών. Είναι φανερό ότι για την επιτάχυνση της διαδικασίας επίλυσης χρειάζεται κατασκευή παράλληλων αλγορίθμων [23], [25], [24], [22].

6.1 Αρχιτεκτονικές κοινής μνήμης

Έχει γίνει ήδη αναφορά στο γεγονός ότι το πρότυπο OpenMP κατά κύριο λόγο επικεντρώνεται στην εκμετάλλευση όσο αφορά την παραλληλοποίηση κατά κύριο λόγο των βρόγχων για παράλληλες αρχιτεκτονικές κοινής διαχείρισης μνήμης. Η διαδικασία αυτή στηρίζεται στο γεγονός ότι κάθε διαδικασία ανά πάσα στιγμή έχει πρόσβαση στα δεδομένα της ενιαίας μνήμης.

Θεωρούμε ένα παράλληλο υπολογιστικό σύστημα κοινής μνήμης το οποίο

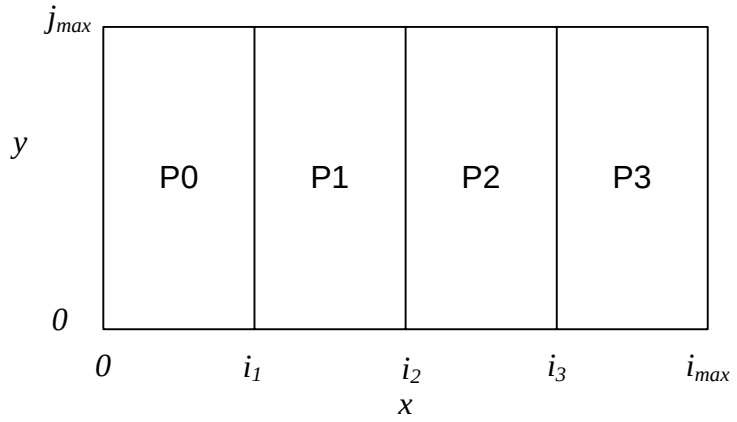
διαθέτει τα παρακάτω χαρακτηριστικά:

- Ο τύπος των επεξεργαστών είναι ίδιος και κάθε ένας τους διαθέτει ανεξάρτητη μνήμη άμεσης προσπέλασης.
- Κάθε επεξεργαστή διαθέτει σημαντική υπολογιστική ισχύ.
- Κάθε επεξεργαστής μπορεί να χρησιμοποιεί για ανάγνωση τα κοινά δεδομένα την ίδια χρονική στιγμή με οποιονδήποτε άλλο, αρκεί να μην τροποποιεί κοινή θέση μνήμης ταυτόχρονα με άλλο επεξεργαστή.
- Κάθε επεξεργαστής έχει τον ίδιο χρόνο πρόσβασης στην κοινή μνήμη, ενώ το επικοινωνιακό κόστος δε διαχειρίζεται από το χρήστη.
- Δεν πρέπει να υπάρχουν αδρανείς επεξεργαστές, δηλαδή χρειάζεται σωστή κατανομή του υπολογιστικού κόστους.

Λαμβάνοντας υπόψιν τα παραπάνω θεωρούμε ένα ορθογώνιο υπολογιστικό χωρίο του οποίου η διαμέριση αποτελείται, ως προς τον άξονα των x, από $i_{\max}$ διακριτοποιήσεις. Έστω ότι υπάρχουν διαθέσιμοι p επεξεργαστές και ισχύει ότι $i_{\max} = kp + \upsilon$, $k \in \mathbb{Z}$. Τότε κάθε επεξεργαστής P_i, $i = 0, \ldots, p-1$, αναλαμβάνει την επίλυση δικριτοποιημένου χωρίου μεγέθους:

$$P_i \longleftarrow \begin{cases} k+1 & \text{αν } i < \upsilon \\ k & \text{αλλιώς.} \end{cases}$$

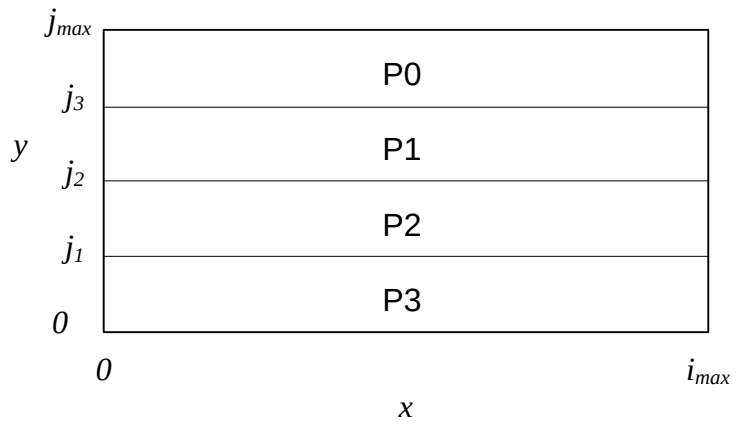
Το Σχήμα 6.1 εμφανίζει την περίπτωση των τεσσάρων επεξεργαστών.



Σχήμα 6.1: Αντιστοίχιση υποχωρίων στους επεξεργαστές ως προς το x άξονα.

Εναλλακτικά μπορεί να χρησιμοποιηθεί η διακριτοποίηση ως προς τον y άξονα για την αντιστοίχιση των υποχωρίων στους επεξεργαστές όπως αυτό παρουσιάζεται

στο Σχήμα 6.2. Κάτι τέτοιο όμως οδηγεί σε αύξηση του επικοινωνιακού κόστους κάθε επεξεργαστή με την κύρια μνήμη λόγω μεγαλύτερης αλληλοεξάρτησης στα σύνορα των υποχωρίων.



Σχήμα 6.2: Αντιστοίχιση υποχωρίων στους επεξεργαστές ως προς το y άξονα.

Αντίστοιχα θα μπορούσε να είναι δύο, οκτώ, δεκαέξι κτλ. Έστω λοιπόν ότι θα χρησιμοποιήσουμε τέσσερις επεξεργαστές. Η υλοποίηση των δύο αλγορίθμων βασίζεται ως επί το πλείστον σε διπλούς βρόγχους μέσω των οποίων διατρέχεται το διακριτοποιημένο χωρίο κατά γραμμές, δηλαδή ως προς τον άξονα των x. Η παραλληλοποίηση αφορά το μοίρασμα του υπολογιστικού χωρίου στους επεξεργαστές. Για να δούμε πως γίνεται αυτό θεωρούμε το παρακάτω παράδειγμα κώδικα.

```
C$OMP PARALLEL DO DEFAULT(SHARED), PRIVATE(var1, var2)
      do i = 0, imax
         do j = 0, jmax

            <code>

         end do
      end do
C$OMP END PARALLEL DO
```

Η χρήση της εντολής `C$OMP PARALLEL DO` προκαλεί το χώρισμα του ορθογωνίου εξίσου ανάλογα με το πλήθος των επεξεργαστών οι οποίοι βρίσκονται εν χρήση, στην περίπτωσή μας τέσσερις, και αντιστοιχεί κάθε κομμάτι σε μία κεντρική μονάδα επεξεργασίας. Τοποθετώντας το πριν το βρόγχο που αφορά το i επιτυγχάνεται το χώρισμα του χωρίου κατά τον άξονα των x. Με αυτόν τον τρόπο

έχουμε αναθέσει στον πρώτο επεξεργαστή το κομμάτι $[0, i_1 - 1] \cap \mathbb{Z}$, στο δεύτερο το $[i_1, i_2 - 1] \cap \mathbb{Z}$ κτλ, όπου τα i_1, i_2, i_3 υπολογίζονται από το OpenMP.
Όπως έχει ήδη αναφερθεί στο προηγούμενο κεφάλαιο, το OpenMP απευθύνεται σε αρχιτεκτονικές μνήμης κοινής διαχείρισης. Ως εκ τούτου, κάθε επεξεργαστής έχει πρόσβαση σε οποιοδήποτε κομμάτι μνήμης χρειαστεί. Πολλές φορές ορισμένες μεταβλητές μπορεί να αφορούν τον κάθε επεξεργαστή ξεχωριστά. Η δήλωση `DEFAULT(SHARED)` ορίζει ότι εν γένει οι μεταβλητές που βρίσκονται μέσα στο `PARALLEL DO` είναι κοινές για κάθε CPU, ενώ αντίθετα με τη δήλωση `PRIVATE(var1, var2)` ορίζονται οι μεταβλητές οι οποίες αφορούν τον κάθε επεξεργαστή αποκλειστικά. Κάθε `C$OMP PARALLEL DO` κλείνει με τη χρήση της εντολής `C$OMP END PARALLEL DO` .

Ανάλογες τακτικές υλοποίησης παράλληλων διαδικασιών με αυτές που εφαρμόστηκαν στο κυρίως πρόγραμμα υλοποιήθηκαν και στο υποπρόγραμμα το οποίο είναι υπεύθυνο για τον υπολογισμό των συνοριακών συνθηκών. Η ενέργεια αυτή είναι απαραίτητη μιας και η εν λόγω ρουτίνα καλείται σε κάθε χρονικό βήμα της κάθε μεθόδου.

Στη συνέχεια παρουσιάζονται οι δύο παράλληλοι αλγόριθμοι των αριθμητικών σχημάτων Roe-TVD και MacCormack.

Παράλληλος αλγόριθμος σχήματος Roe-TVD σε OpenMP

```
program Roe_OpenMP
read data
C$OMP PARALLEL DEFAULT(SHARED)
read ics
C$OMP END PARALLEL
do while (time < tfinal)
C$OMP PARALLEL DEFAULT(SHARED)
   read bcs
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED) PRIVATE(uaver, vaver, haver, c)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute uaver, vaver, haver, c
         compute eigenvalues, eigenvectors
         compute b, a
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED) PRIVATE(indices, thetas)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute n, indices, thetas, phi
         compute f+, f-, g+, g-
      end do
   end do
C$OMP END PARALLEL
```

```
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute F, G
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = 0, imax
      do j = 0, jmax
         compute w
         compute u, v, h
      end do
   end do
C$OMP END PARALLEL
C$OMP DO
   do i = 1, imax
      do j = 1, jmax
         compute max_eigenvalue
      end do
   end do
C$OMP END DO
   compute time_step
end do
C$OMP PARALLEL DEFAULT(SHARED)
read bcs
C$OMP END PARALLEL
write results
```

Παράλληλος αλγόριθμος σχήματος MacCormack σε OpenMP

```
program MacCormack_OpenMP
read data
C$OMP PARALLEL DEFAULT(SHARED)
read ics
C$OMP END PARALLEL
do while (time < tfinal)
C$OMP PARALLEL DEFAULT(SHARED)
   read bcs
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED) PRIVATE(c)
   do i = -5, imax + 5
      do j = -5, jmax + 5c
         compute eigenvalues, eigenvectors
         compute b, a, R, c, n
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
```

```
         compute D
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute F, G
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute w(1)
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute F(1), G(1), R(1)
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute w(2)
      end do
   end do
C$OMP END PARALLEL
C$OMP PARALLEL DEFAULT(SHARED)
   do i = 1, imax
      do j = 1, jmax
         compute w
         compute u, v, h
      end do
   end do
C$OMP END PARALLEL
C$OMP DO
   do i = 1, imax
      do j = 1, jmax
         compute max_eigenvalue
      end do
   end do
C$OMP END DO
   compute time_step
end do
C$OMP PARALLEL DEFAULT(SHARED)
read bcs
```

```
C$OMP END PARALLEL
write results
```

6.2 Αρχιτεκτονικές μη-κοινής μνήμης

Το MPI (Message Passing Interface), όπως δηλώνει το όνομά του, αφορά κατά κύριο λόγο τη μετάδοση και λήψη μηνυμάτων/δεδομένων σε αρχιτεκτονικές μη-κοινής μνήμης. Για το λόγο αυτό θα οριστεί ως *πομπός* ο επεξεργαστής ο οποίος αποστέλλει μηνύματα και ως *δέκτης* αυτός ο οποίος λαμβάνει. Θα θεωρηθεί ότι στη γενική περίπτωση υπάρχουν p το πλήθος επεξεργαστές οι οποίοι είναι ισοδύναμοι. Κάθε ένας από αυτούς έχει ένα δείκτη ο οποίος τον χαρακτηρίζει ξεκινώντας από το 0 για τον πρώτο και φτάνοντας στο $p-1$ για τον p-οστό. Το δείκτη αυτόν θα τον ονομάσουμε *rank*.

Κατά την εκτέλεση ενός αλγορίθμου ο οποίος είναι υλοποιημένος με τη χρήση του MPI ο κάθε επεξεργαστής έχει το δικό του αποκλειστικό τμήμα μνήμης το οποίο εκμεταλλεύεται. Το γεγονός αυτό είναι προφανές μιας και η παραλληλοποίηση κατά αυτόν τον τρόπο βασίζεται σε αρχιτεκτονική μη-κοινής μνήμης. Η κάθε κεντρική μονάδα επεξεργασίας εκτελεί τον κώδικα όπως ακριβώς αυτός έχει. Εάν λοιπόν σε ένα σειριακό πρόγραμμα, τοποθετηθούν οι απαραίτητες εντολές έναρξης και τερματισμού του MPI χωρίς άλλες παρεμβάσεις το αποτέλεσμα θα είναι να εκτελεστεί ο κώδικας από τον κάθε επεξεργαστή ξεχωριστά χωρίς να υπάρχει κάποια σχέση μεταξύ τους. Για να κατασκευαστεί λοιπόν ένα παράλληλο πρόγραμμα βασισμένο στο MPI απαιτούνται ειδικές οδηγίες οι οποίες αναφέρονται στο rank του κάθε επεξεργαστή.

Κατά την υλοποίηση του παράλληλου κώδικα επιθυμούμε όλοι οι επεξεργαστές να διαβάσουν τα αρχικά δεδομένα του εκάστοτε προβλήματος του οποίου ζητείται η αριθμητική λύση. Για το λόγο αυτό το αρχικό τμήμα του κώδικα, το οποίο αναφέρεται στη εισαγωγή δεδομένων εκτελεστεί από όλες τις κεντρικές μονάδες επεξεργασίας.

Στη συνέχεια έχουμε το κυρίως τμήμα των μεθόδων του Roe και του MacCormack. Στο σημείο αυτό κρίνεται απαραίτητο να γίνει ο χωρισμός της περιοχής που εξελίσεται το πρόβλημα. Εν αντιθέσει με το OpenMP, στο MPI ο χωρισμός του χωρίου ορίζεται από το χρήστη. Πάλι η λογική που θα ακολουθηθεί είναι ανάλογη με αυτή του Σχήματος 6.1. Με τους δείκτες `ibegin` και `iend` οριοθετείται το κάθε υποχωρίο που αντιστοιχεί σε κάθε έναν από τους επεξεργαστές. Να σημειωθεί ότι ο εκάστοτε αλγόριθμος εκτελείται σε ένα grid $[0, imax] \cap \mathbb{Z} \times [0, jmax] \cap \mathbb{Z}$. Οπότε ο κάθε επεξεργαστής εργάζεται στο κομμάτι $[ibegin, iend] \cap \mathbb{Z} \times [0, jmax] \cap \mathbb{Z}$. Για να πραγματοποιηθεί λοιπόν το χώρισμα και η ανάθεση των τμημάτων του προβλήματος στους επεξεργαστές χρησιμοποιούμε τον αλγόριθμο

```
ibegin = (rank*imax)/p + 1
iend = ((rank + 1)*imax)/p
```

```
if (rank .eq. p - 1) iend = imax
```

Να τονιστεί ότι οι δείκτες `ibegin` και `iend` έχουν διαφορετική τιμή σε κάθε έναν επεξεργαστή μιας και γίνεται χρήση αρχιτεκτονικής μνήμης μη-κοινής διαχείρισης. Με τον παραπάνω τρόπο λοιπόν, επιτυγχάνεται ο πλέον ίσος καταμερισμός φόρτου για την κάθε κεντρική μονάδα επεξεργασίας. Μία σύγκριση μεταξύ σειριακού και παράλληλου κώδικα παρουσιάζεται στο Σχήμα 6.3.

Σειριακός Κώδικας

```
do i = 0, imax
   do j = 0, jmax

      <code>

   end do
end do
```

Παράλληλος Κώδικας

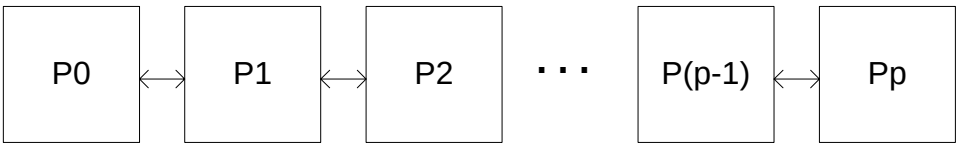
```
ibegin = (rank*imax)/p+1
iend = ((rank+1)*imax)/p
if (rank .eq. 0) ibegin = 0
if (rank .eq. p-1) iend = imax
do i = ibegin, iend
   do j = 0, jmax

      <code>

   end do
end do
```

Σχήμα 6.3: Σύγκριση σειριακού και παράλληλου κώδικα

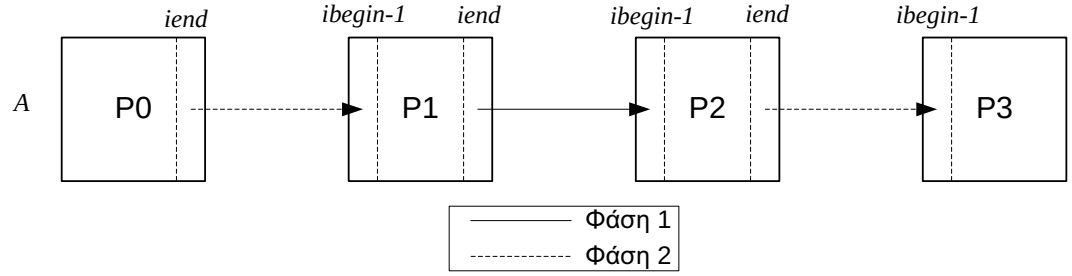
Για την υλοποίηση των δύο σχημάτων είναι απαραίτητη η «σύνδεση» των συνόρων των γειτονικών υποχωρίων για ανταλλαγή πληροφορίας είτε προς τα εμπρός, είτε προς τα πίσω. Ορισμένα δεδομένα τα οποία βρίσκονται στην πρώτη ή στην τελευταία στήλη του διακριτοποιημένου υποχωρίου ενός επεξεργαστή πρέπει να μεταδοθούν στον προηγούμενο ή επόμενο αντίστοιχα ούτως ώστε η κάθε CPU να έχει όλα τα δεδομένα που απαιτούνται για τους υπολογισμούς που εκτελεί. Αυτή η μορφή επικοινωνίας απαιτεί σύνδεση των επεξεργαστών σε σειρά όπως αυτή του Σχήματος 6.4.



Σχήμα 6.4: Σύνδεση επεξεργαστών για αρχιτεκτονικές μη-κοινής μνήμης.

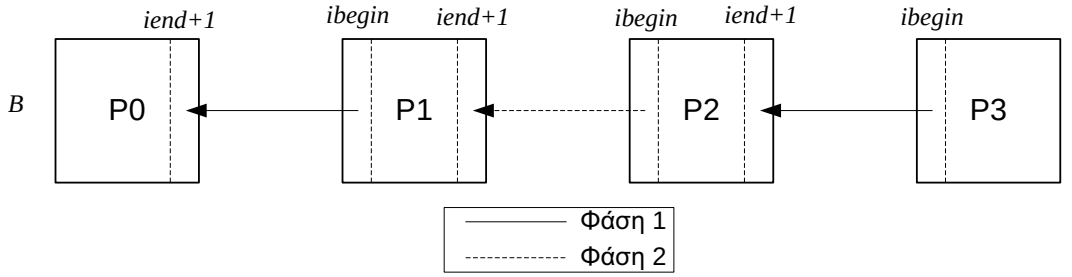
Ας θεωρήσουμε ότι θέλουμε να στείλουμε την τελευταία στήλη ενός πίνακα A που αντιστοιχεί σε κάθε επεξεργαστή στον επόμενο. Ένας ασφαλής τρόπος για

να πραγματοποιηθεί αυτό είναι να το χωρίσουμε σε δύο φάσεις. Κατά την πρώτη φάση οι επεξεργαστές με περιττό rank γίνονται πομποί και στέλνουν δεδομένα ενώ οι επεξεργαστές με άρτιο rank είναι δέκτες και δέχονται δεδομένα. Αντίστοιχα στη δεύτερη φάση πομποί γίνονται οι επεξεργαστές με άρτιο rank και δέκτες οι επεξεργαστές με περιττό. Μία τέτοια διαδικασία αποστολής-λήψης φαίνεται στο Σχήμα 6.5.



Σχήμα 6.5: Προς τα εμπρός αποστολή δεδομένων με τη μέθοδο περιττών-άρτιων.

Αναλόγως γίνεται και η προς τα πίσω αποστολή δεδομένων. Στο Σχήμα 6.6 παπουσιάζεται η αποστολή της πρώτης στήλης ενός πίνακα B που αντιστοιχεί σε κάθε επεξεργαστή στον προηγούμενό του. Και σε αυτήν την περίπτωση γίνεται χρήση της μεθόδου συγχρονισμού περιττών-άρτιων.

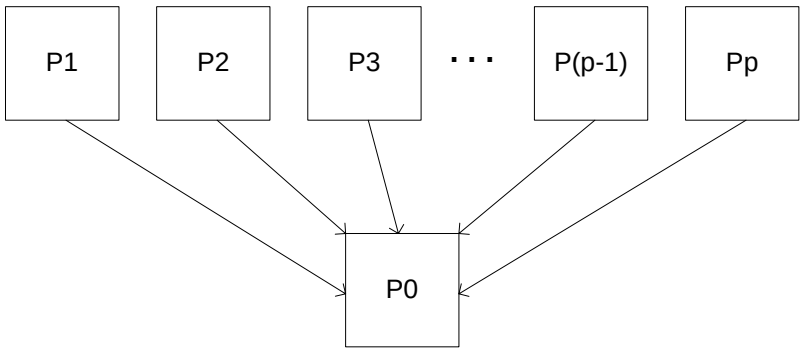


Σχήμα 6.6: Προς τα πίσω αποστολή δεδομένων με τη μέθοδο περιττών-άρτιων.

Ο λόγος για τον οποίο γίνεται χρήση της παραπάνω μεθόδου αποστολής-λήψης δεδομένων είναι για να διασφαλιστεί η μετάδοση του μηνύματος. Σε περίπτωση στην οποίο ο χρήστης ζητούσε τον ορισμό ενός επεξεργαστή ως πομπού και δέκτη ταυτόχρονα, η ευστάθια της μετάδοσης θα κινδύνευε από θέματα συγχρονισμού με αποτέλεσμα είτε την καθυστέρηση στη εκτέλεση είτε στη χειρότερη των περιπτώσεων σε λάθος τελική αριθμητική λύση του προβλήματος.

Μετά το πέρας της εκτέλεσης του αλγορίθμου τα τελικά αποτελέσματα θα πρέπει να συλλεχθούν. Τη διαδικασία τις περισυλλογής αναλαμβάνει η κεντρική

μονάδα επεξεργασίας με rank 0. Όλοι οι επεξεργαστές εκπέμπουν τα αποτελέσματα του υποχωρίου το οποίο έχουν αναλάβει και ο επεξεργαστής με rank 0 συλλέγει τα δεδομένα. Η διαδικασία αυτή παρουσιάζεται στο Σχήμα 6.7.



Σχήμα 6.7: Επικοινωνία επεξεργαστών κατά τη διάρκεια συλλογής τελικών αποτελεσμάτων.

Κάνοντας κανείς μία σύνοψη όσο αφορά την παράλληλη υλοποίηση των δύο αλγορίθμων με τη χρήση του MPI παρατηρεί ότι σε ένα μεγάλο βαθμό βασίζεται στη λογική του SIMD. Αυτό γιατί ο κάθε επεξεργαστής αναλαμβάνει ένα ξεχωριστό υποχωρίο και το επιλύει ακολουθώντας τις ίδιες εντολές. Όμως κατά το τέλος της επίλυσης τα δεδομένα του κάθε επεξεργαστή συλλέγονται σε έναν. Αυτή η διαφοροποίηση κατατάσσει το πρόγραμμα σε MIMD.

Ακολουθούν οι δύο παράλληλοι αλγόριθμοι των αριθμητικών σχημάτων Roe-TVD και MacCormack για αρχιτεκτονικές μη-κοινής μνήμης.

Παράλληλος αλγόριθμος σχήματος Roe-TVD σε MPI

```
program Roe_MPI
initialize MPI
read data
read ics
do while (time < tfinal)
   read bcs
   MPI_send u(:,ibegin), v(:,ibegin), h(:,ibegin) to p-1
   MPI_receive u(:,iend+1), v(:,iend+1), h(:,iend+1) from p+1
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute uaver, vaver, haver, c
         compute eigenvalues, eigenvectors
         compute b, a
      end do
   end do
   do i = -5, imax + 5
```

```
      do j = -5, jmax + 5
         compute n, indices, thetas, phi
         compute f+, f-, g+, g-
      end do
   end do
   MPI_send f+(:,iend) to p+1
   MPI_receive f+(:ibegin-1) from p-1
   do i = -5, imax + 5
      do j = -5, jmax + 5
         compute F, G
      end do
   end do
   MPI_send F(:,iend) to p+1
   MPI_receive F(:ibegin-1) from p-1
   do i = 1, imax
      do j = 1, jmax
         compute w
         compute u, v, h
      end do
   end do
   do i = 1, imax
      do j = 1, jmax
         compute max_eigenvalue
      end do
   end do
   if myrank = 0 then
      MPI_collect max_eigenvalues
      compute max_eigenvalue
      compute time_step
      MPI_broadcast time_step
   end if
end do
read bcs
if myrank = 0 then
   MPI_collect u, v, h
   write results
end if
```

Παράλληλος αλγόριθμος σχήματος MacCormack σε MPI

```
program MacCormack_MPI
initialize MPI
read data
read ics
do while (time < tfinal)
   read bcs
   MPI_send u(:,ibegin), v(:,ibegin), h(:,ibegin) B(:,ibegin) to p-1
   MPI_receive u(:,iend+1), v(:,iend+1), h(:,iend+1) B(:,iend+1)from p+1
   MPI_send B(:,iend) to p+1
   MPI_receive B(:ibegin-1) from p-1
```

```
do i = -5, imax + 5
   do j = -5, jmax + 5c
      compute eigenvalues, eigenvectors
      compute b, a, R, c, n
   end do
end do
do i = -5, imax + 5
   do j = -5, jmax + 5
      compute D
   end do
end do
do i = -5, imax + 5
   do j = -5, jmax + 5
      compute F, G
   end do
end do
MPI_send F(:,iend) to p+1
MPI_receive F(:ibegin-1) from p-1
do i = -5, imax + 5
   do j = -5, jmax + 5
      compute w(1)
   end do
end do
do i = -5, imax + 5
   do j = -5, jmax + 5
      compute F(1), G(1), R(1)
   end do
end do
MPI_send F(1)(:,ibegin) to p-1
MPI_receive F(1)(:,iend+1)from p+1
do i = -5, imax + 5
   do j = -5, jmax + 5
      compute w(2)
   end do
end do
MPI_send D(:,ibegin) to p-1
MPI_receive D(:,iend+1)from p+1
do i = 1, imax
   do j = 1, jmax
      compute w
      compute u, v, h
   end do
end do
do i = 1, imax
   do j = 1, jmax
      compute max_eigenvalue
   end do
end do
if myrank = 0 then
   MPI_collect max_eigenvalues
   compute max_eigenvalue
```

```
      compute time_step
      MPI_broadcast time_step
   end if
end do
read bcs
if myrank = 0 then
   MPI_collect u, v, h
   write results
end if
```

6.3 Μέτρηση απόδοσης παράλληλης υλοποίησης αλγορίθμου

Οι δείκτες οι οποίοι χρησιμοποιούνται συνήθως για τη μέτρηση της απόδοσης ενός παράλληλου κώδικα είναι δύο. Το Speedup (S), το οποίο είναι ένα μέτρο που εκφράζει το όφελος που αποκομίζεται με τη χρήση παράλληλης επεξεργασίας ενός προβλήματος και ορίζεται ως, [18]

$$S = \frac{\text{απαιτούμενος χρόνος για έναν επεξεργαστή}}{\text{απαιτούμενος χρόνος για } p \text{ επεξεργαστές}}. \tag{6.3.1}$$

Το άλλο μέτρο το οποίο χρησιμοποιείται είναι η Αποδοτικότητα (A) η οποία μετράει το ποσοστό του χρόνου που αφιερώνει ο κάθε επεξεργαστής στο υπολογιστικό τμήμα και ορίζεται ως ο λόγος του Speedup προς το πλήθος των επεξεργαστών p, άρα

$$A = \frac{S}{p}. \tag{6.3.2}$$

Προφανώς τα όρια για το S είναι μεταξύ 0 και p ενώ το A παίρνει τιμές από 0 έως 1 στην καλλίτερη περίπτωση. Με βάση τους δύο παραπάνω δείκτες, του Speedup και της Αποδοτικότητας, θα γίνει η σύγκριση των παράλληλων υλοποιήσεων των αλγορίθμων των σχημάτων του Roe και του MacCormack στις δύο διαστάσης που πραγματοποιήθηκαν για την εκπόνηση της παρούσας εργασίας.

6.4 Διαδικασία εκτέλεσης παράλληλων αλγορίθμων

Στη διάθεσή μας είχαμε ένα παράλληλο υπολογιστικό σύστημα της εταιρίας SGI και μοντέλο Origin 350. Το Origin 350 διαθέτει 8 επεξεργαστές τεχνολογίας MIPSPro® κοινής μνήμης 8 GByte με συχνότητα χρονισμού τα 600 MHz με 4 MByte μνήμης άμεσης προσπέλασης ο καθένας. Το λειτουργικό σύστημα ήταν το IRIX 6.5 και οι μεταγλωττιστές ήταν MIPSPro® 7.4. Στο μηχάνημα αυτό εκτελέστηκαν οι αλγόριθμοι σε OpenMP και MPI με χρήση 2, 4 και 8 επεξεργαστών.

Πέραν του SGI δοκιμάστηκε και μία συστάδα υπολογιστικών συστημάτων το οποίο αποτελούνταν από τέσσερις κόμβους Sun Fire V240. Το κάθε Sun Fire

V240 διέθεται δύο επεξεργαστές τεχνολογίας UltraSPARC IIIi. Το ρολόι του κάθε επεξεργαστή είναι συγχρονισμένο στα 1.5 GHz με 1 MByte μνήμης άμεσης προσπέλασης ο καθένας. Η μνήμη σε κάθε μηχάνημα είναι 2 GByte κοινής διαχείρισης οπότε τα προβλήματα δοκιμάστηκαν στο OpenMP σε 2 επεξεργαστές. Η εκτέλεση των αλγορίθμων στο MPI έγινε σε 2, 4 και 8 επεξεργαστές έχοντας συνδέσει τα V240 με δίκτυο Ethernet είτε 1Gbit είτε 100Mbit.

SGI Origin 350
- Σειριακά
- OpenMP: 2, 4, 8 επεξεργαστές
- MPI: 2, 4, 8 επεξεργαστές

Sun Fire V240
- Σειριακά
- OpenMP: 2 επεξεργαστές
- MPI 1Gbit: 2, 4, 8 επεξεργαστές
- MPI 100Mbit: 2, 4, 8 επεξεργαστές

Σχήμα 6.8: Συνδυασμοί εκτέλεσης αλγορίθμων.

Οι δείκτες που μετρήθηκαν ήταν ο χρόνος εκέλεσης, το Speedup, η Αποδοτικότητα και ο *χρόνος Επικοινωνίας* μεταξύ των επεξεργαστών[1].

[1] Ο χρόνος επικοινωνίας μεταξύ των επεξεργαστών είναι δυνατόν να μετρηθεί μόνο στην περίπτωση του MPI.

6.4.1 Πρόβλημα A - SGI Origin OpenMP/MPI

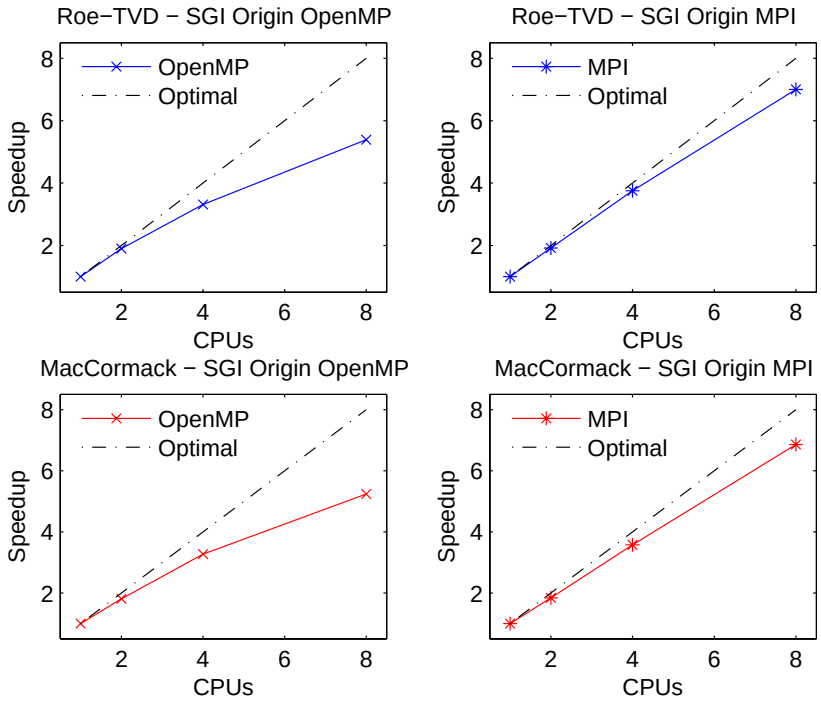
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.1: Πρόβλημα A: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	42.12	1.00	100.00	42.12	1.00	100.00	0.00
2	22.18	1.90	94.95	21.98	1.92	95.81	0.03
4	12.74	3.31	82.65	11.22	3.75	93.85	0.06
8	7.81	5.39	67.41	6.02	7.00	87.46	0.12

Πίνακας 6.2: Πρόβλημα A: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	34.31	1.00	100.00	34.31	1.00	100.00	0.00
2	18.91	1.81	90.45	18.59	1.84	92.01	0.02
4	10.45	3.27	81.84	9.58	3.57	89.27	0.04
8	6.53	5.24	65.49	4.99	6.86	85.70	0.10



Σχήμα 6.9: Πρόβλημα A: Αποτελέσματα στο SGI Origin.

6.4.2 Πρόβλημα A - Sun Fire V240 MPI

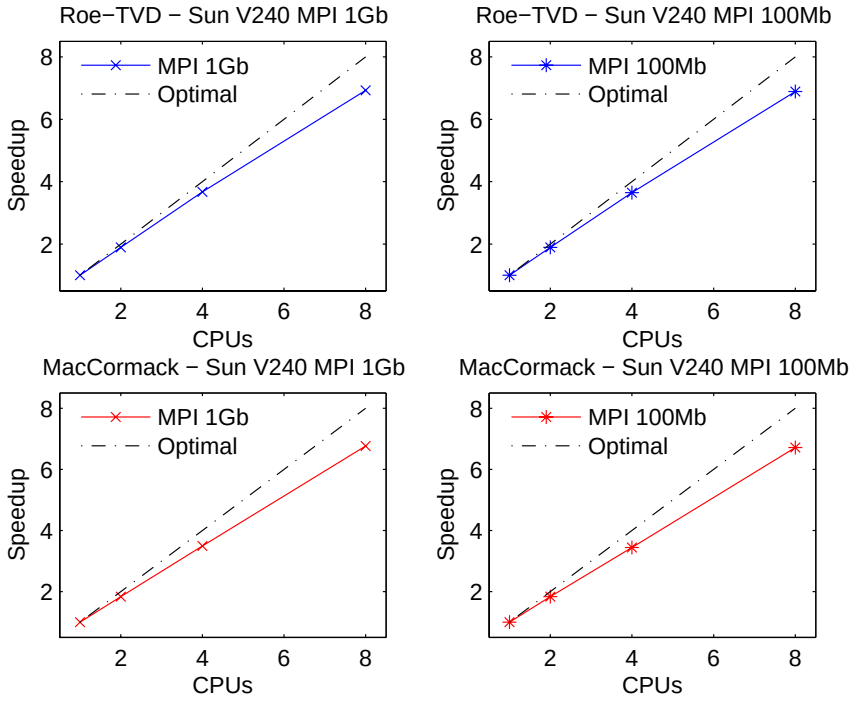
Δεδομένα: Grid: 300×300, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.3: Πρόβλημα A: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	54.98	1.00	100.00	0.00	54.98	1.00	100.00	0.00
2	29.13	1.89	94.37	0.03	29.13	1.89	94.37	0.03
4	14.97	3.67	91.82	0.08	15.06	3.65	91.27	0.10
8	7.93	6.93	86.66	0.17	7.98	6.89	86.12	0.23

Πίνακας 6.4: Πρόβλημα A: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	46.21	1.00	100.00	0.00	46.21	1.00	100.00	0.00
2	25.27	1.83	91.43	0.03	25.27	1.83	91.43	0.03
4	13.24	3.49	87.25	0.07	13.44	3.44	85.96	0.08
8	6.84	6.76	84.45	0.15	6.89	6.71	83.84	0.20



Σχήμα 6.10: Πρόβλημα A: Αποτελέσματα MPI στο Sun Fire V240.

6.4.3 Πρόβλημα A - Sun Fire V240 OpenMP

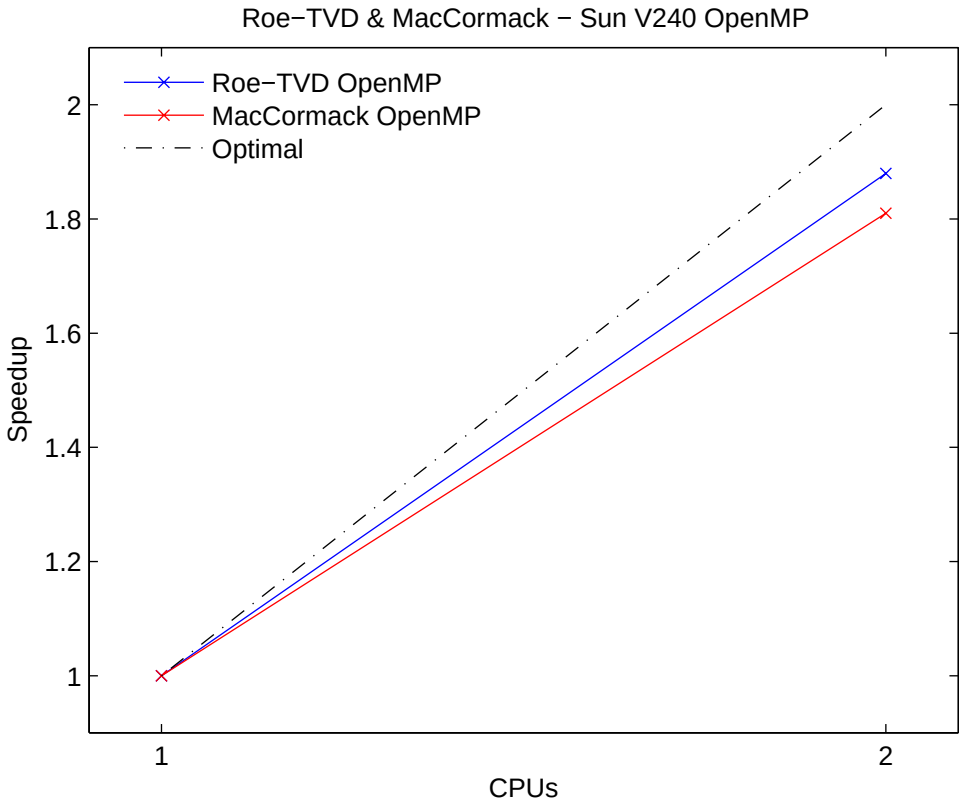
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.5: Πρόβλημα A: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	54.98	1.00	100.00
2	29.23	1.88	94.05

Πίνακας 6.6: Πρόβλημα A: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	46.21	1.00	100.00
2	25.54	1.81	90.47



Σχήμα 6.11: Πρόβλημα A: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.4 Πρόβλημα Β - SGI Origin OpenMP/MPI

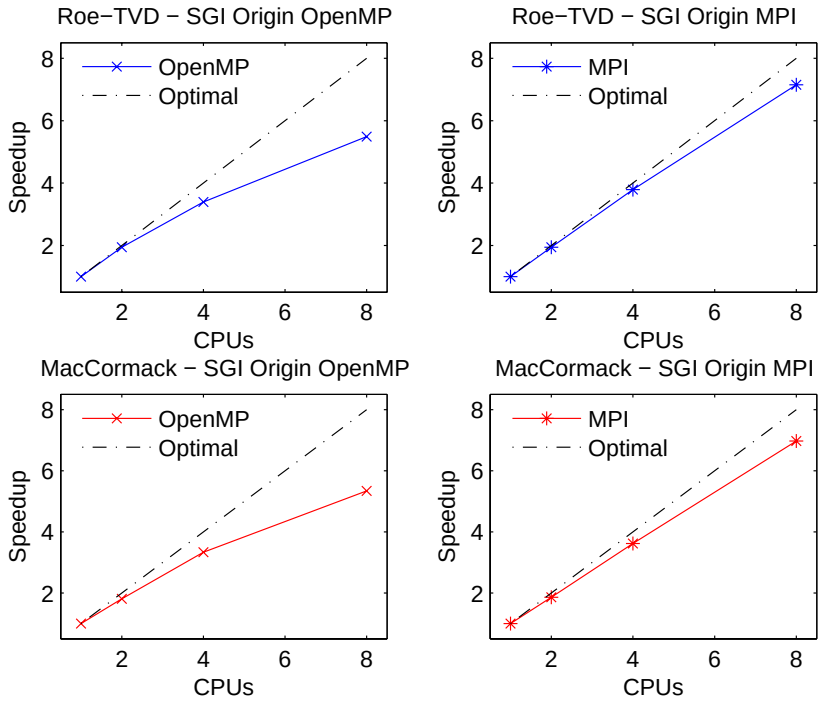
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.7: Πρόβλημα Β: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	110.89	1.00	100.00	110.89	1.00	100.00	0.00
2	57.15	1.94	97.02	57.09	1.94	97.12	0.07
4	32.73	3.39	84.70	29.23	3.79	94.84	0.14
8	20.21	5.49	68.59	15.51	7.15	89.37	0.31

Πίνακας 6.8: Πρόβλημα Β: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	89.16	1.00	100.00	89.16	1.00	100.00	0.00
2	47.66	1.87	93.54	47.88	1.86	97.12	0.04
4	26.76	3.33	83.30	24.60	3.62	94.84	0.12
8	16.69	5.34	66.78	12.80	6.97	87.07	0.25



Σχήμα 6.12: Πρόβλημα Β: Αποτελέσματα στο SGI Origin.

6.4.5 Πρόβλημα B - Sun Fire V240 MPI

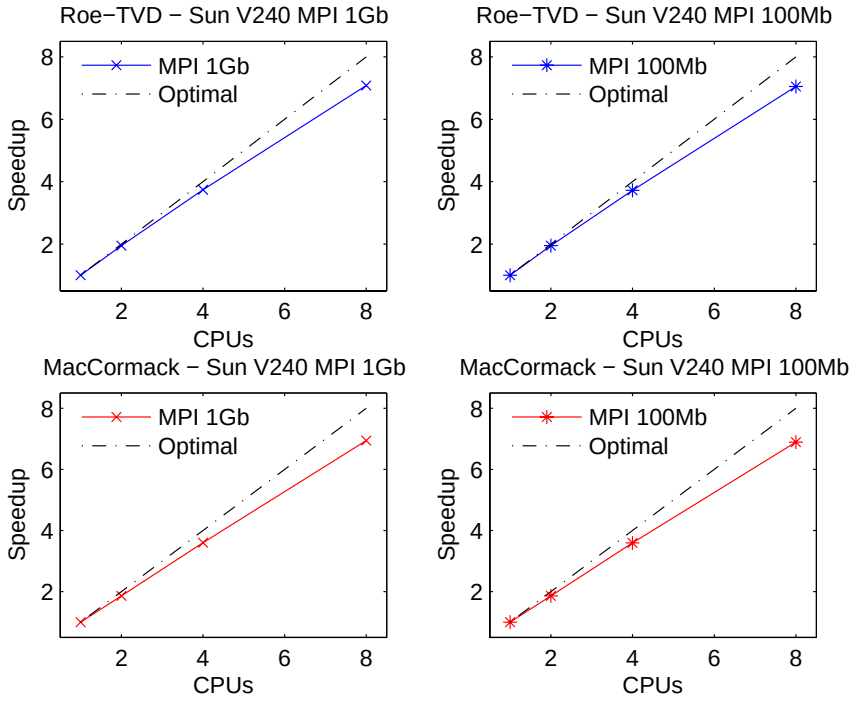
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.9: Πρόβλημα B: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	134.11	1.00	100.00	0.00	134.11	1.00	100.00	0.00
2	68.82	1.95	97.44	0.08	68.82	1.95	97.44	0.08
4	35.87	3.74	93.47	0.20	39.06	3.72	92.98	0.24
8	18.95	7.08	88.46	0.42	19.01	7.05	88.18	0.57

Πίνακας 6.10: Πρόβλημα B: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	99.14	1.00	100.00	0.00	99.14	1.00	100.00	0.00
2	53.32	1.86	92.97	0.06	53.32	1.86	92.97	0.06
4	27.55	3.60	89.96	0.15	27.60	3.59	89.80	0.18
8	14.28	6.94	86.78	0.31	14.38	6.89	86.18	0.42



Σχήμα 6.13: Πρόβλημα B: Αποτελέσματα MPI στο Sun Fire V240.

6.4.6 Πρόβλημα B - Sun Fire V240 OpenMP

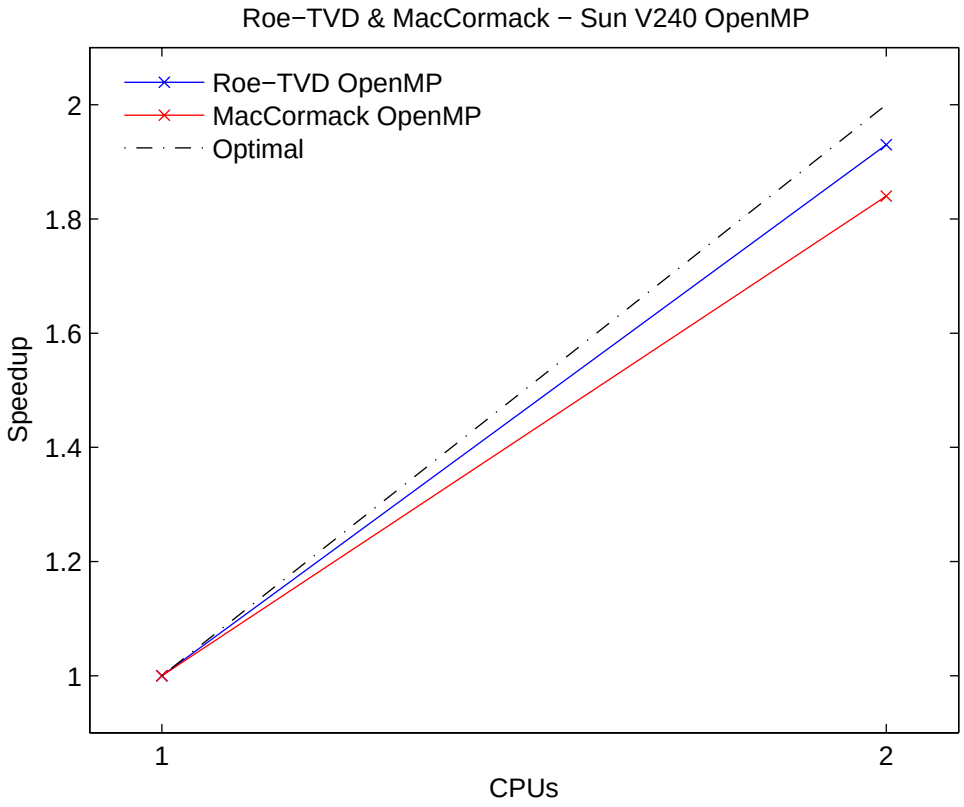
Δεδομένα: Grid: 300×300, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.11: Πρόβλημα B: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	134.11	1.00	100.00
2	69.53	1.93	96.44

Πίνακας 6.12: Πρόβλημα B: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	99.14	1.00	100.00
2	53.98	1.84	93.94



Σχήμα 6.14: Πρόβλημα B: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.7 Πρόβλημα C - SGI Origin OpenMP/MPI

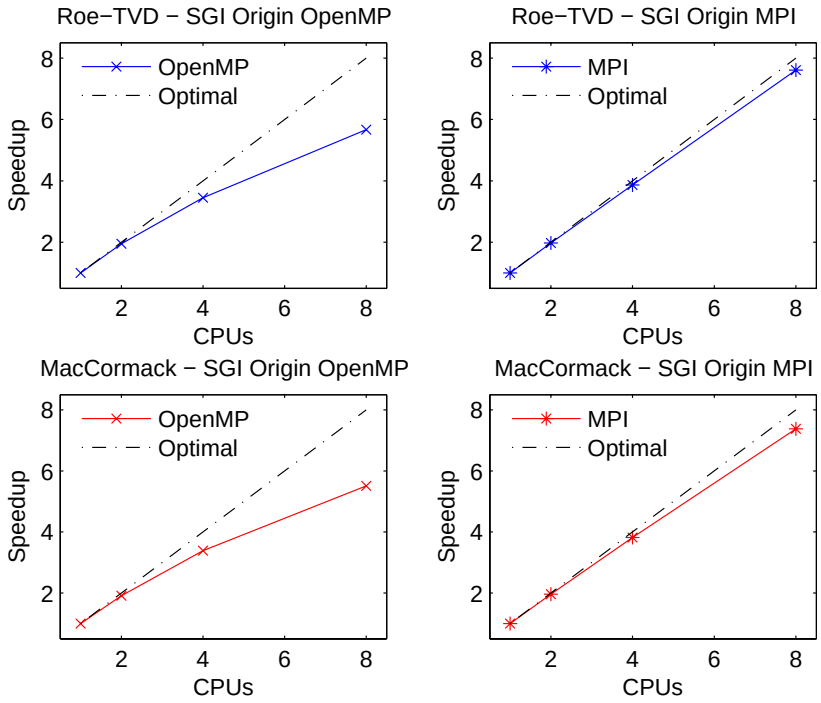
Δεδομένα: Grid: 60×280, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.13: Πρόβλημα C: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	590.21	1.00	100.00	590.21	1.00	100.00	0.00
2	302.66	1.95	97.50	298.15	1.98	98.98	0.26
4	171.07	3.45	86.25	152.51	3.87	96.75	0.58
8	104.12	5.67	70.86	77.56	7.61	93.52	1.05

Πίνακας 6.14: Πρόβλημα C: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	472.34	1.00	100.00	472.34	1.00	100.00	0.00
2	247.32	1.91	95.49	241.12	1.96	97.95	0.26
4	139.41	3.39	84.70	123.77	3.82	95.41	0.47
8	85.73	5.51	68.87	63.98	7.38	92.28	1.13



Σχήμα 6.15: Πρόβλημα C: Αποτελέσματα στο SGI Origin.

6.4.8 Πρόβλημα C - Sun Fire V240 MPI

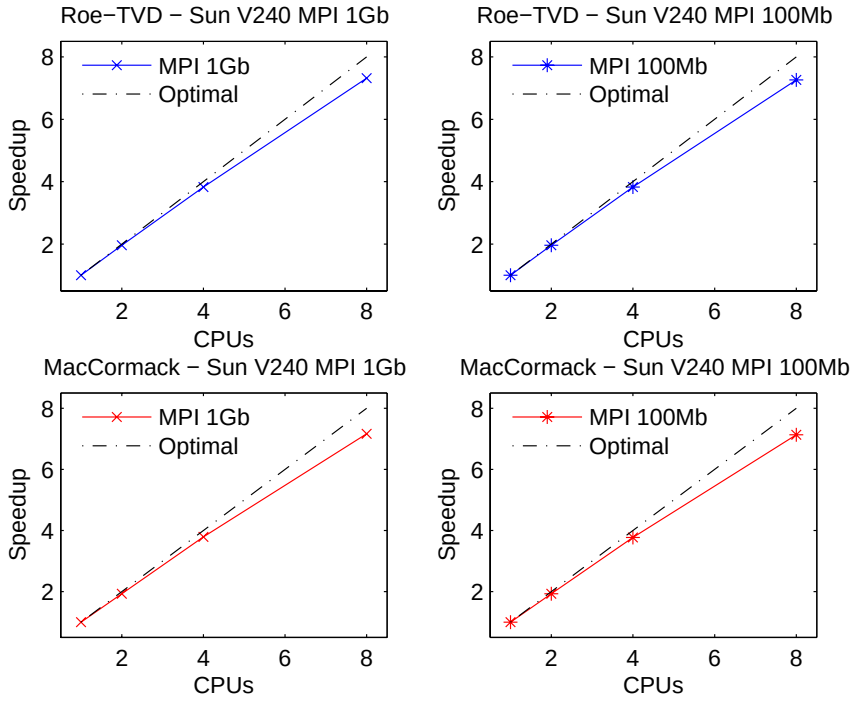
Δεδομένα: Grid: 60×280, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.15: Πρόβλημα C: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	715.23	1.00	100.00	0.00	715.23	1.00	100.00	0.00
2	364.36	1.96	98.15	0.33	364.36	1.96	98.15	0.33
4	186.55	3.83	95.85	0.95	186.59	3.83	95.83	1.09
8	97.77	7.32	91.44	1.82	98.45	7.26	90.81	2.01

Πίνακας 6.16: Πρόβλημα C: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	529.87	1.00	100.00	0.00	529.87	1.00	100.00	0.00
2	274.21	1.93	96.62	0.22	274.21	1.93	96.62	0.22
4	139.87	3.79	94.71	0.70	140.60	3.77	94.22	0.76
8	73.98	7.16	89.53	1.65	74.33	7.13	89.11	1.84



Σχήμα 6.16: Πρόβλημα C: Αποτελέσματα MPI στο Sun Fire V240.

6.4.9 Πρόβλημα C - Sun Fire V240 OpenMP

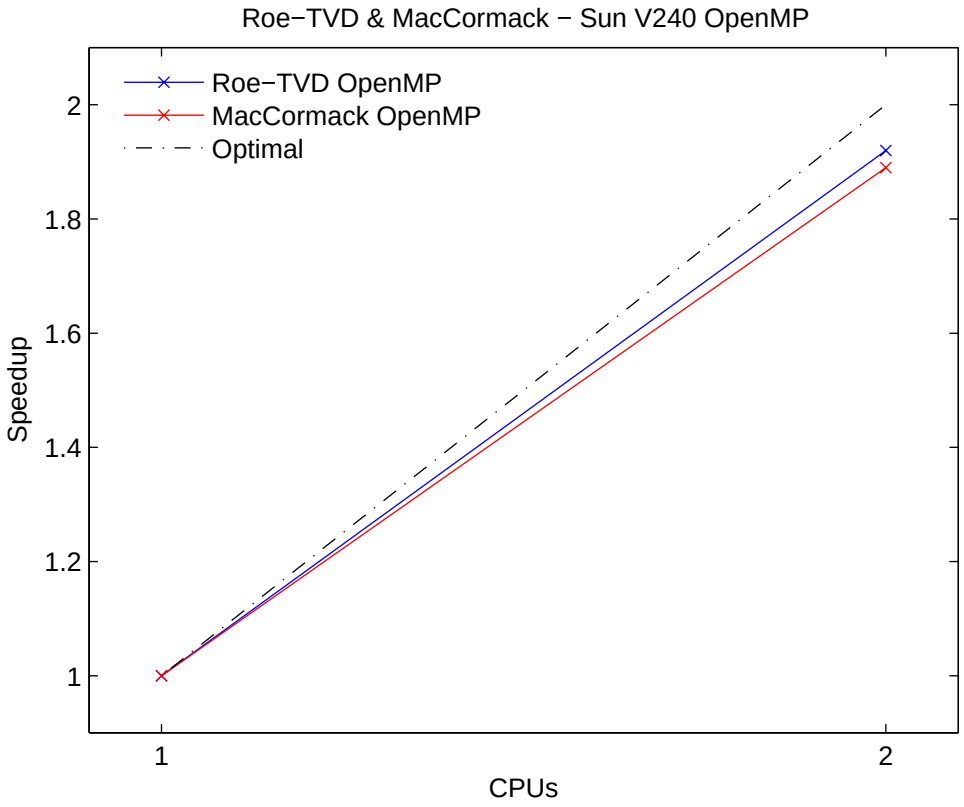
Δεδομένα: Grid: 60×280, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.17: Πρόβλημα C: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	715.23	1.00	100.00
2	371.59	1.92	96.24

Πίνακας 6.18: Πρόβλημα C: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	529.87	1.00	100.00
2	280.33	1.89	94.51



Σχήμα 6.17: Πρόβλημα C: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.10 Πρόβλημα D(a) - SGI Origin OpenMP/MPI

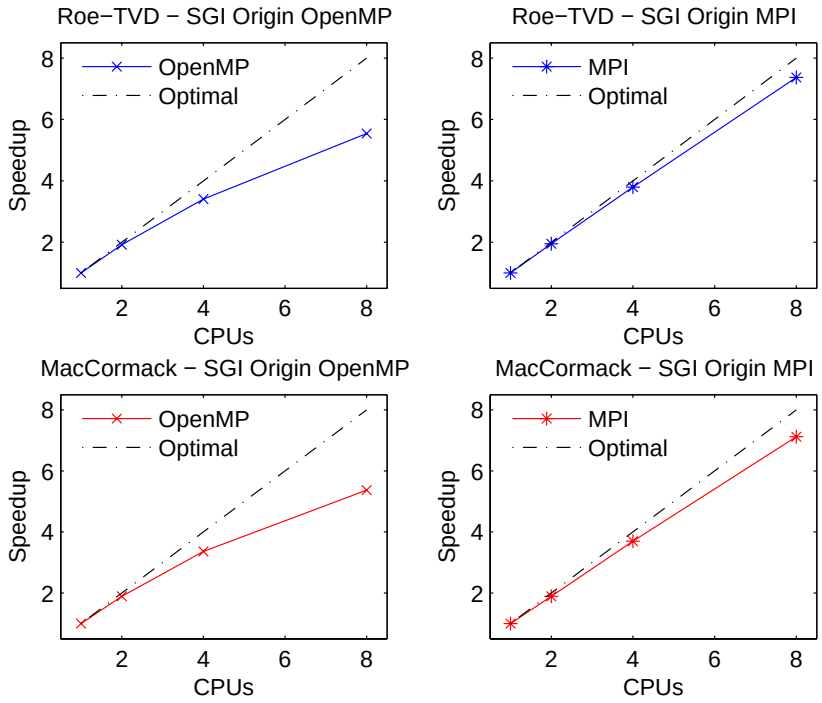
Δεδομένα: Grid: 170×170, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.19: Πρόβλημα D(a): Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	90.74	1.00	100.00	90.74	1.00	100.00	0.00
2	47.25	1.92	96.02	46.61	1.95	97.34	0.05
4	26.59	3.41	85.31	23.86	3.80	95.08	0.15
8	16.37	5.54	69.29	12.31	7.37	92.14	0.24

Πίνακας 6.20: Πρόβλημα D(a): MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	71.48	1.00	100.00	71.48	1.00	100.00	0.00
2	38.44	1.86	92.98	37.81	1.89	94.53	0.04
4	21.26	3.36	84.05	19.34	3.70	92.40	0.09
8	13.32	5.37	67.08	10.04	7.12	88.99	0.20



Σχήμα 6.18: Πρόβλημα D(a): Αποτελέσματα στο SGI Origin.

6.4.11 Πρόβλημα D(a) - Sun Fire V240 MPI

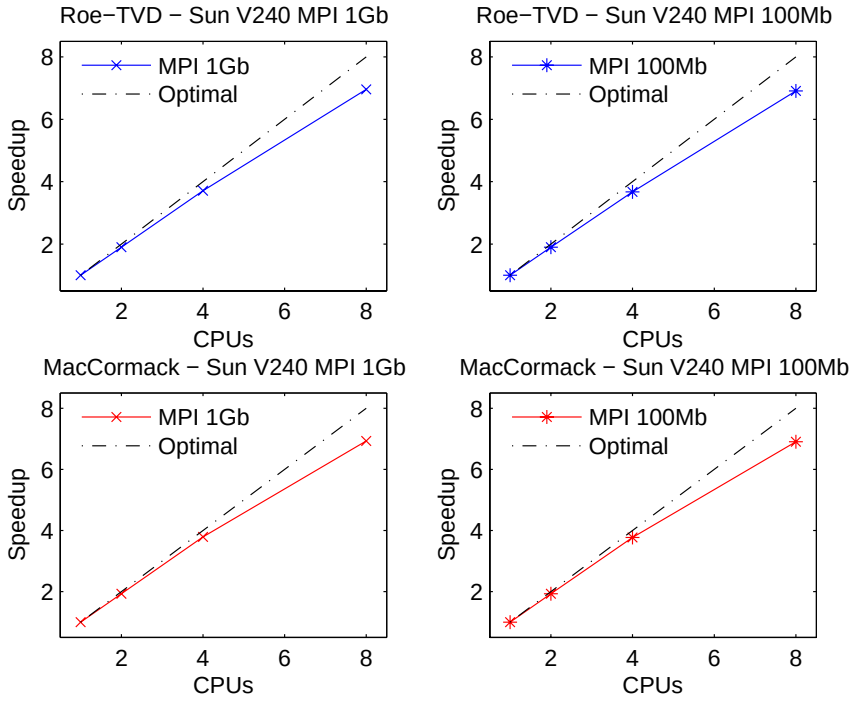
Δεδομένα: Grid: 170×170, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.21: Πρόβλημα D(a): Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	121.65	1.00	100.00	0.00	121.65	1.00	100.00	0.00
2	63.99	1.90	95.05	0.07	63.99	1.90	95.05	0.07
4	32.77	3.71	92.81	0.19	33.17	3.67	91.69	0.22
8	17.47	6.96	87.04	0.38	17.60	6.91	86.40	0.52

Πίνακας 6.22: Πρόβλημα D(a): MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	79.65	1.00	100.00	0.00	79.65	1.00	100.00	0.00
2	42.24	1.89	94.28	0.05	42.24	1.89	94.28	0.05
4	21.75	3.66	91.55	0.12	21.79	3.66	91.38	0.15
8	11.50	6.93	86.58	0.25	11.55	6.90	87.72	0.34



Σχήμα 6.19: Πρόβλημα D(a): Αποτελέσματα MPI στο Sun Fire V240.

6.4.12 Πρόβλημα D(a) - Sun Fire V240 OpenMP

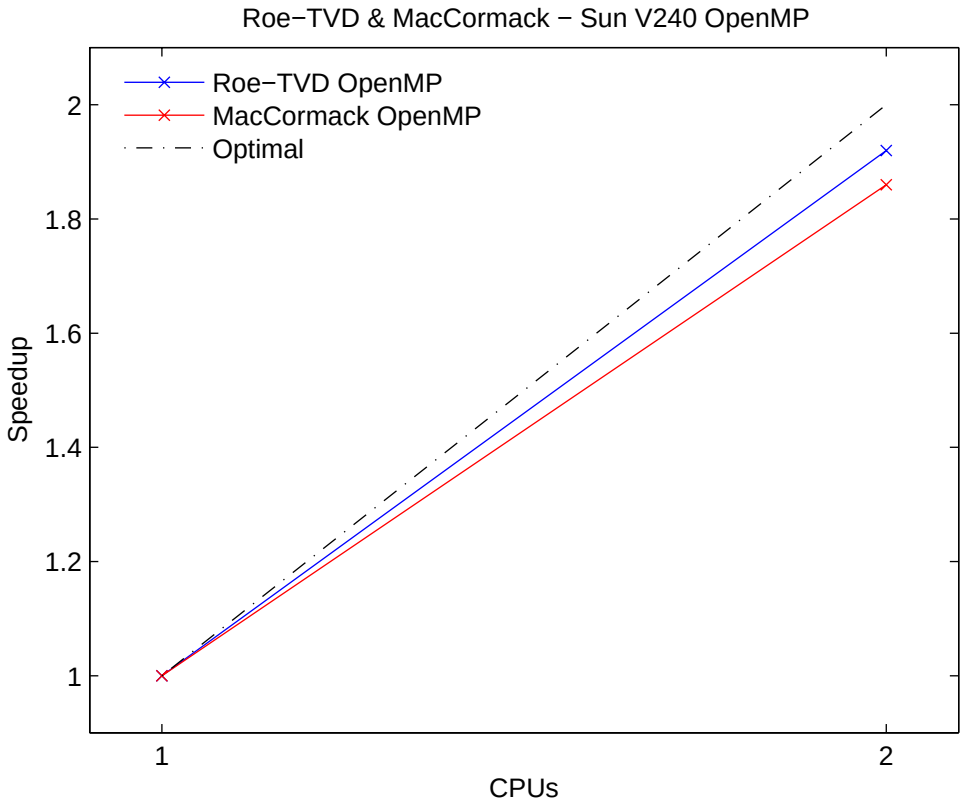
Δεδομένα: Grid: 170×170, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.23: Πρόβλημα D(a): Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	121.65	1.00	100.00
2	63.37	1.92	95.98

Πίνακας 6.24: Πρόβλημα D(a): MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	79.65	1.00	100.00
2	42.86	1.86	92.92



Σχήμα 6.20: Πρόβλημα D(a): Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.13 Πρόβλημα D(b) - SGI Origin OpenMP/MPI

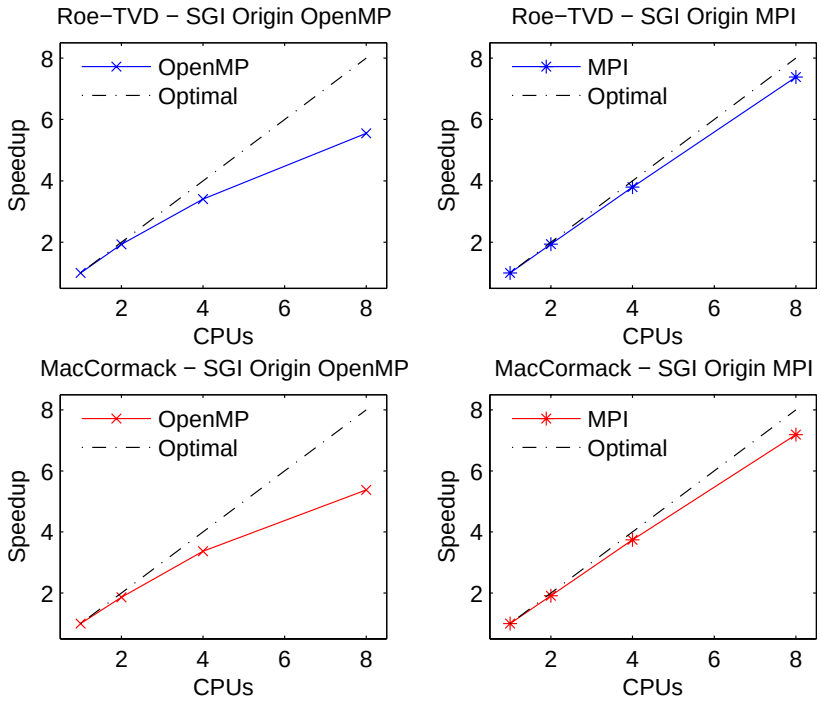
Δεδομένα: Grid: 186×200, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.25: Πρόβλημα D(b): Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	114.33	1.00	100.00	114.33	1.00	100.00	0.00
2	59.35	1.93	96.32	59.01	1.94	96.87	0.07
4	33.51	3.41	85.30	30.12	3.80	94.90	0.15
8	20.59	5.55	69.41	15.49	7.38	92.26	0.30

Πίνακας 6.26: Πρόβλημα D(b): MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	88.64	1.00	100.00	89.11	1.00	100.00	0.00
2	47.56	1.86	93.19	46.39	1.91	95.54	0.05
4	26.34	3.37	84.13	23.72	3.74	93.42	0.12
8	16.49	5.38	67.19	12.32	7.19	89.94	0.25



Σχήμα 6.21: Πρόβλημα D(b): Αποτελέσματα στο SGI Origin.

6.4.14 Πρόβλημα D(b) - Sun Fire V240 MPI

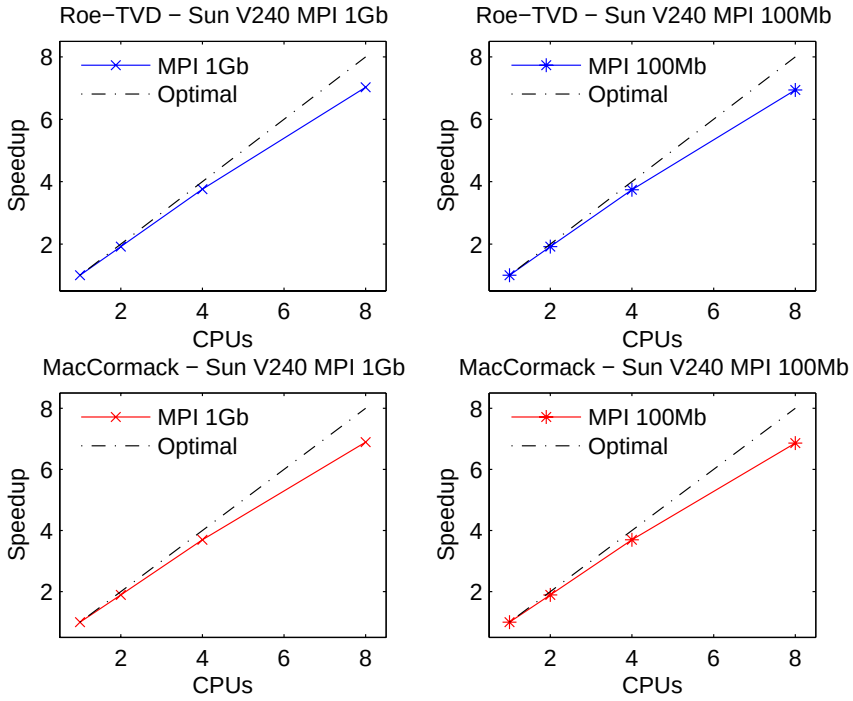
Δεδομένα: Grid: 186×200, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.27: Πρόβλημα D(b): Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	137.98	1.00	100.00	0.00	137.98	1.00	100.00	0.00
2	71.75	1.92	96.15	0.07	71.75	1.91	95.69	0.08
4	36.70	3.76	93.99	0.19	36.92	3.74	93.43	0.25
8	19.62	7.03	87.91	0.38	19.87	6.94	86.80	0.58

Πίνακας 6.28: Πρόβλημα D(b): MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	98.54	1.00	100.00	0.00	98.54	1.00	100.00	0.00
2	52.14	1.89	94.50	0.08	52.14	1.89	94.50	0.08
4	26.69	3.69	92.30	0.15	26.71	3.69	92.23	0.18
8	14.30	6.89	86.14	0.31	14.37	6.86	85.72	0.42



Σχήμα 6.22: Πρόβλημα D(b): Αποτελέσματα MPI στο Sun Fire V240.

6.4.15 Πρόβλημα D(b) - Sun Fire V240 OpenMP

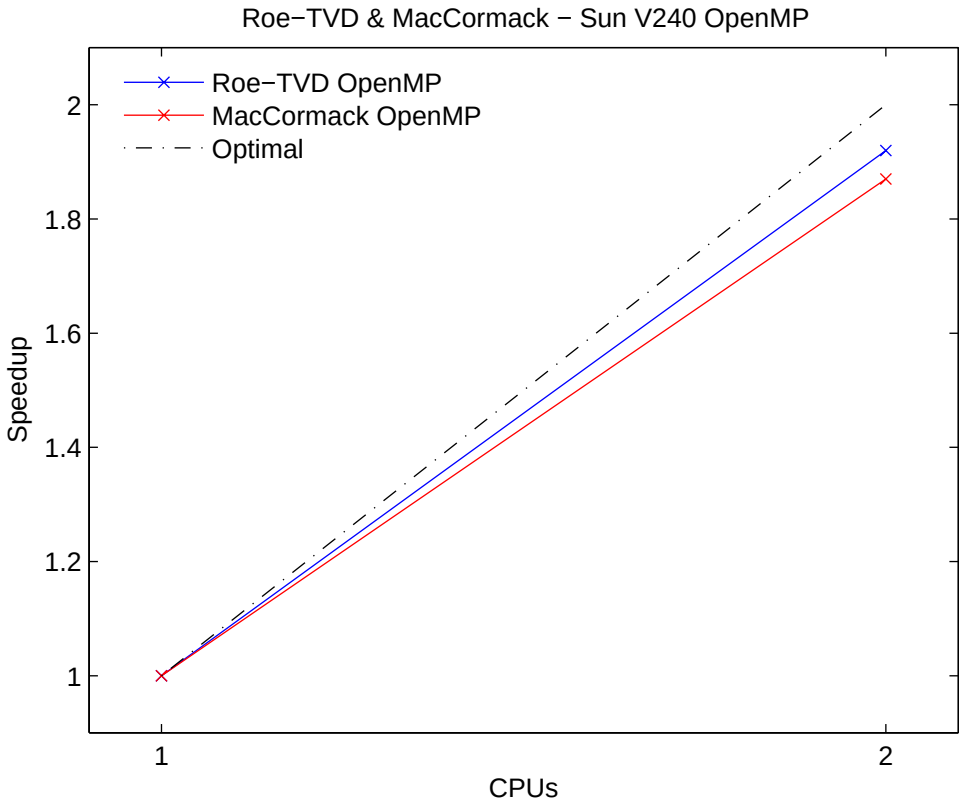
Δεδομένα: Grid: 186×200, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.29: Πρόβλημα D(b): Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	138.69	1.00	100.00
2	72.26	1.92	95.97

Πίνακας 6.30: Πρόβλημα D(b): MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	99.12	1.00	100.00
2	53.06	1.87	93.40



Σχήμα 6.23: Πρόβλημα D(b): Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.16 Πρόβλημα Ε - SGI Origin OpenMP/MPI

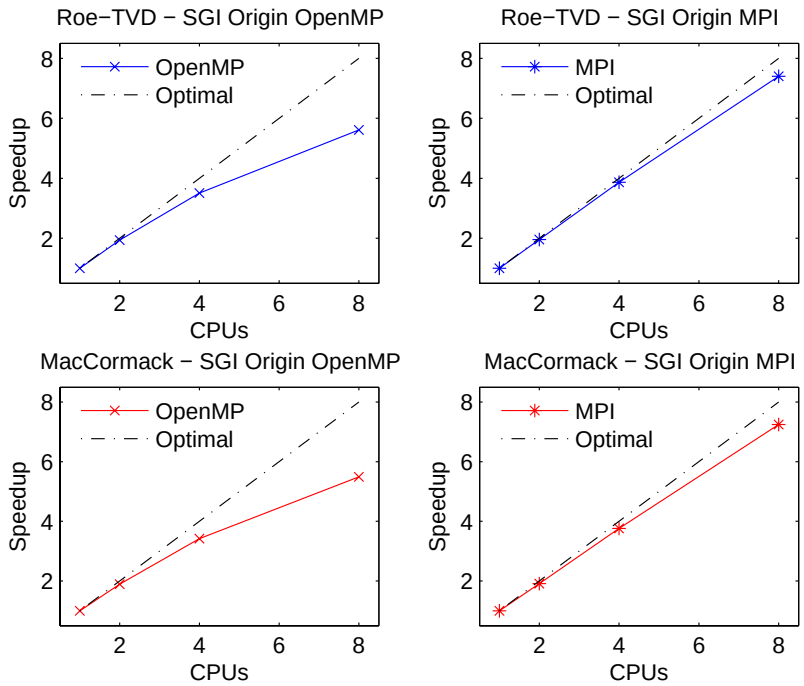
Δεδομένα: Grid: 200×200, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.31: Πρόβλημα Ε: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	285.44	1.00	100.00	285.44	1.00	100.00	0.00
2	147.11	1.94	97.02	145.85	1.96	97.85	0.17
4	81.36	3.51	87.71	73.67	3.87	96.86	0.37
8	50.84	5.61	70.18	38.55	7.40	92.56	0.79

Πίνακας 6.32: Πρόβλημα Ε: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	229.31	1.00	100.00	229.31	1.00	100.00	0.00
2	121.35	1.89	94.48	119.90	1.91	95.63	0.14
4	67.09	3.42	85.45	61.06	3.76	93.89	0.31
8	41.78	5.49	68.61	31.61	7.25	90.68	0.65



Σχήμα 6.24: Πρόβλημα Ε: Αποτελέσματα στο SGI Origin.

6.4.17 Πρόβλημα Ε - Sun Fire V240 MPI

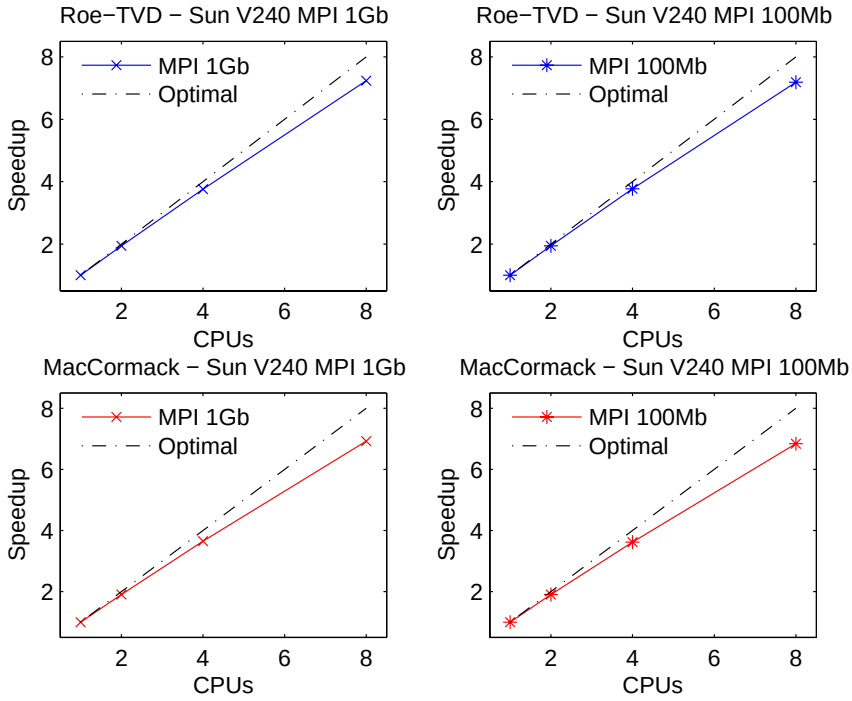
Δεδομένα: Grid: 200×200, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.33: Πρόβλημα Ε: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	345.21	1.00	100.00	0.00	345.21	1.00	100.00	0.00
2	178.12	1.94	96.90	0.21	178.12	1.94	96.90	0.21
4	91.86	3.76	93.95	0.52	91.92	3.77	94.20	0.63
8	47.71	7.24	90.44	1.08	47.98	7.19	89.94	1.49

Πίνακας 6.34: Πρόβλημα Ε: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	253.45	1.00	100.00	0.00	253.45	1.00	100.00	0.00
2	133.37	1.90	95.02	0.15	133.37	1.90	95.02	0.15
4	69.47	3.65	91.21	0.38	69.98	3.62	90.54	0.46
8	36.64	6.92	86.47	0.79	37.05	6.84	85.51	1.07



Σχήμα 6.25: Πρόβλημα Ε: Αποτελέσματα MPI στο Sun Fire V240.

6.4.18 Πρόβλημα Ε - Sun Fire V240 OpenMP

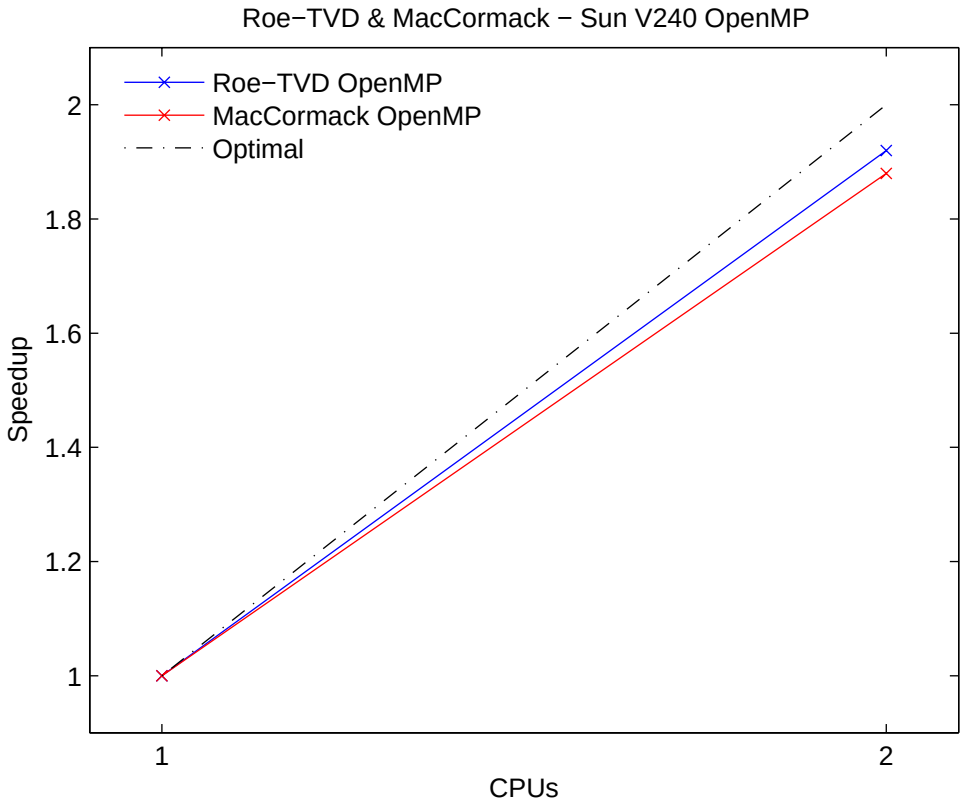
Δεδομένα: Grid: 200×200, $\mathrm{CFL} = 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.35: Πρόβλημα Ε: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	345.21	1.00	100.00
2	179.97	1.92	95.91

Πίνακας 6.36: Πρόβλημα Ε: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	253.45	1.00	100.00
2	135.12	1.88	93.79



Σχήμα 6.26: Πρόβλημα Ε: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.19 Πρόβλημα F - SGI Origin OpenMP/MPI

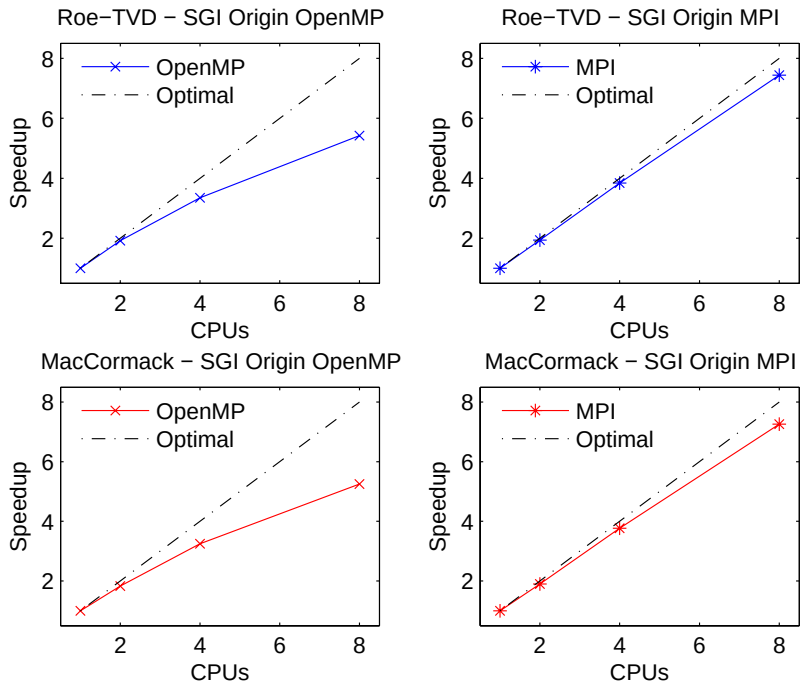
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.37: Πρόβλημα F: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	160.23	1.00	100.00	160.23	1.00	100.00	0.00
2	83.44	1.92	96.02	82.47	1.94	97.14	0.10
4	47.86	3.35	83.70	41.77	3.84	95.90	0.21
8	529.55	5.41	67.78	21.54	7.44	92.98	0.45

Πίνακας 6.38: Πρόβλημα F: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	125.55	1.00	100.00	125.55	1.00	100.00	0.00
2	68.60	1.83	91.51	67.77	1.90	95.18	0.08
4	38.64	3.35	81.23	34.23	3.77	94.22	0.17
8	23.90	5.25	65.66	17.77	7.26	90.75	0.36



Σχήμα 6.27: Πρόβλημα F: Αποτελέσματα στο SGI Origin.

6.4.20 Πρόβλημα F - Sun Fire V240 MPI

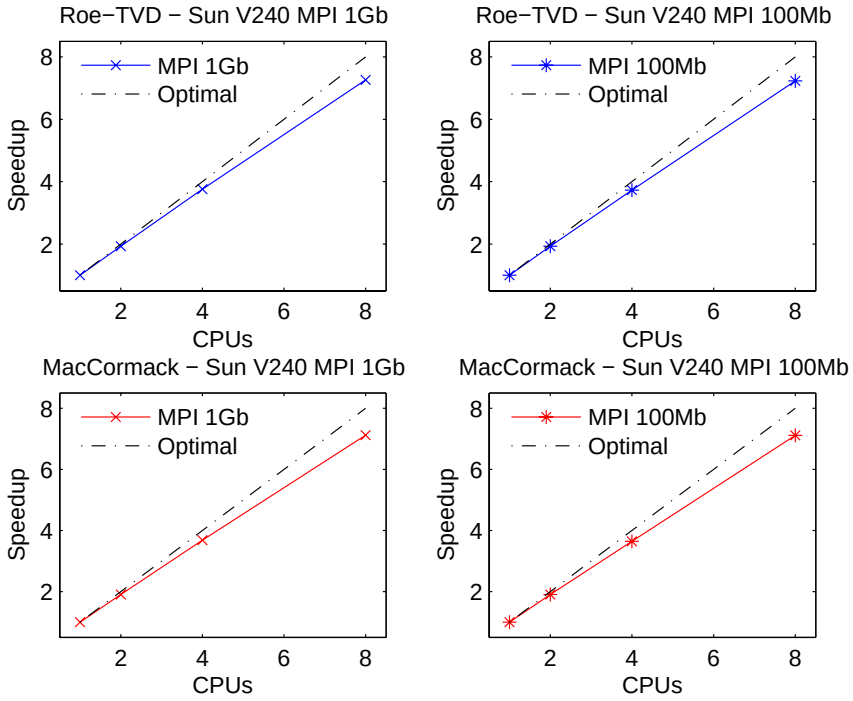
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.39: Πρόβλημα F: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	194.20	1.00	100.00	0.00	194.20	1.00	100.00	0.00
2	100.74	1.93	96.39	0.12	100.74	1.93	96.39	0.12
4	51.69	3.76	93.96	0.29	52.13	3.73	93.13	0.35
8	26.74	7.26	90.78	0.62	26.87	7.23	90.34	0.82

Πίνακας 6.40: Πρόβλημα F: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	144.02	1.00	100.00	0.00	144.02	1.00	100.00	0.00
2	75.66	1.90	95.18	0.09	75.66	1.90	95.18	0.09
4	39.12	3.68	92.04	0.22	39.59	3.64	90.94	0.26
8	20.23	7.12	88.99	0.45	20.26	7.11	88.86	0.61



Σχήμα 6.28: Πρόβλημα F: Αποτελέσματα MPI στο Sun Fire V240.

6.4.21 Πρόβλημα F - Sun Fire V240 OpenMP

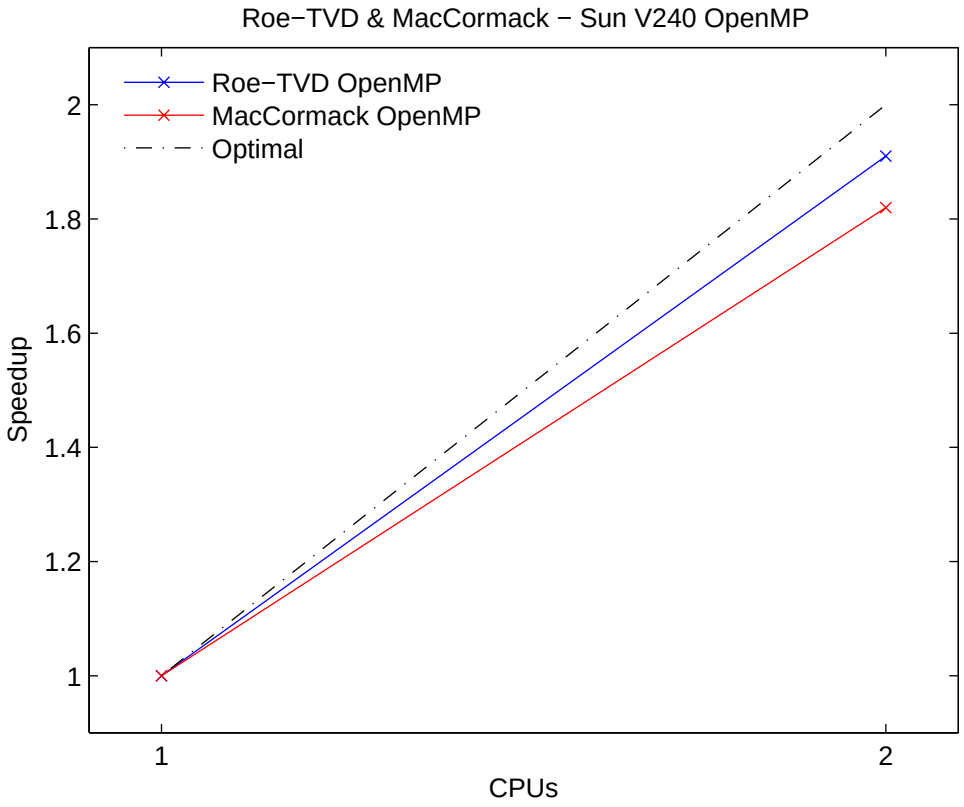
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Superbee περιοριστής-ροής

Πίνακας 6.41: Πρόβλημα F: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	194.20	1.00	100.00
2	101.69	1.91	95.49

Πίνακας 6.42: Πρόβλημα F: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	144.02	1.00	100.00
2	79.22	1.82	90.90



Σχήμα 6.29: Πρόβλημα F: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.22 Πρόβλημα G - SGI Origin OpenMP/MPI

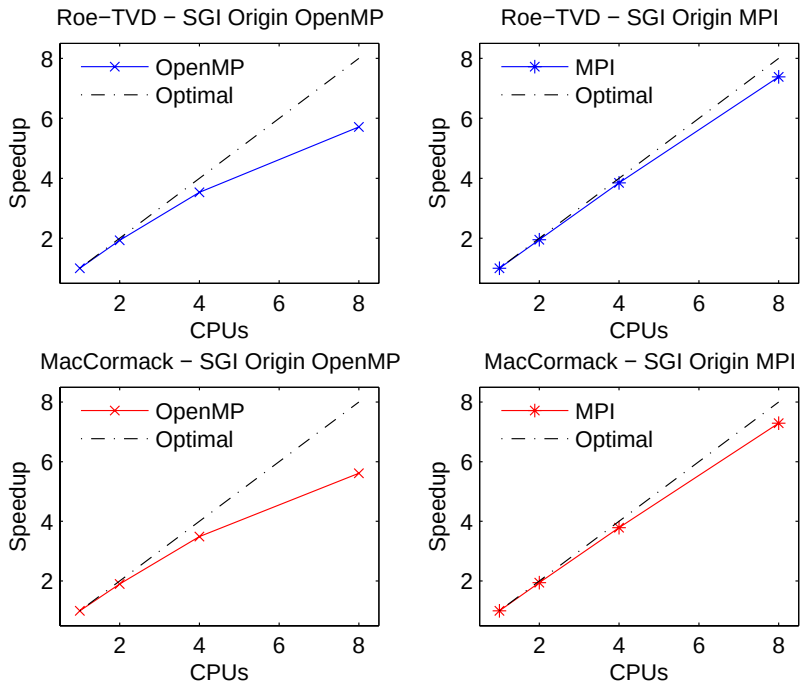
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.43: Πρόβλημα G: Roe-TVD, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	2018.68	1.00	100.00	2018.68	1.00	100.00	0.00
2	1044.98	1.93	96.59	1035.68	1.95	97.46	1.22
4	571.86	3.53	88.25	524.77	3.85	96.17	2.64
8	353.49	5.71	71.38	273.59	7.38	92.23	5.68

Πίνακας 6.44: Πρόβλημα G: MacCormack, SGI Origin OpenMP/MPI

	OpenMP			MPI			
CPUs	Χρόνος	SU	Αποδ. %	Χρόνος	SU	Αποδ. %	Επικ.
1	1619.28	1.00	100.00	1619.28	1.00	100.00	0.00
2	852.12	1.90	95.01	836.69	1.94	96.77	0.97
4	463.74	3.49	87.29	427.13	3.37	94.78	2.11
8	288.54	5.61	70.15	222.04	7.29	91.16	4.54



Σχήμα 6.30: Πρόβλημα G: Αποτελέσματα στο SGI Origin.

6.4.23 Πρόβλημα G - Sun Fire V240 MPI

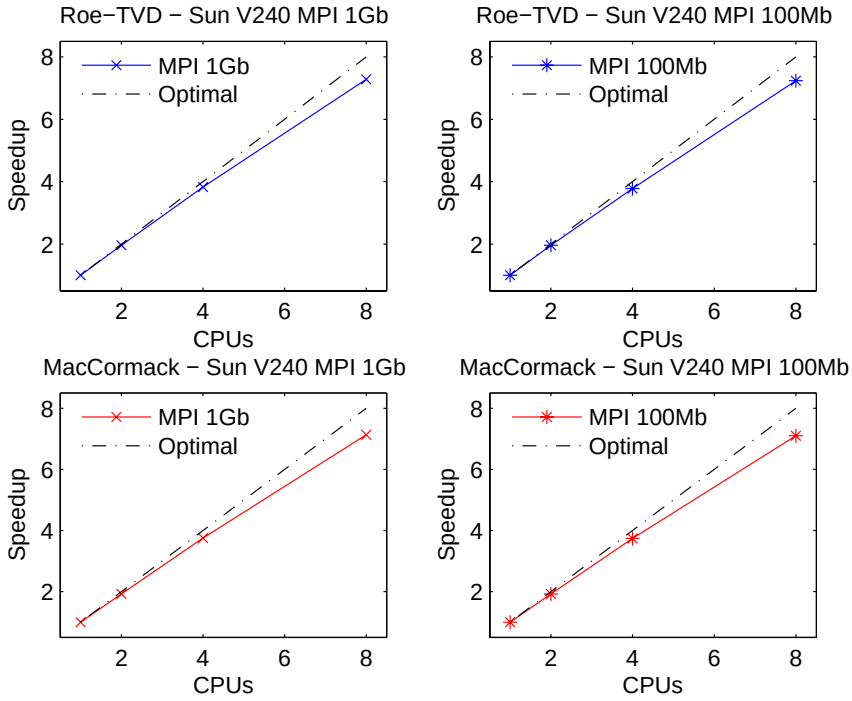
Δεδομένα: Grid: 300×300, CFL $= 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.45: Πρόβλημα G: Roe-TVD, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	2449.01	1.00	100.00	0.00	2449.01	1.00	100.00	0.00
2	1246.64	1.96	98.22	1.47	1246.64	1.96	98.22	1.47
4	639.24	3.83	95.78	3.68	647.78	3.78	94.52	4.42
8	336.31	7.28	91.03	7.61	338.17	7.24	90.51	10.31

Πίνακας 6.46: Πρόβλημα G: MacCormack, Sun Fire V240 MPI

	MPI 1Gbit				MPI 100Mbit			
CPUs	Χρόνος	SU	Αποδ. %	Επικ.	Χρόνος	SU	Αποδ. %	Επικ.
1	1752.32	1.00	100.00	0.00	1752.32	1.00	100.00	0.00
2	911.02	1.92	96.17	1.05	911.02	1.92	96.17	1.05
4	467.32	3.75	93.74	2.63	468.89	3.74	93.43	3.16
8	245.93	7.13	89.07	5.44	246.78	7.10	88.76	7.38



Σχήμα 6.31: Πρόβλημα G: Αποτελέσματα MPI στο Sun Fire V240.

6.4.24 Πρόβλημα G - Sun Fire V240 OpenMP

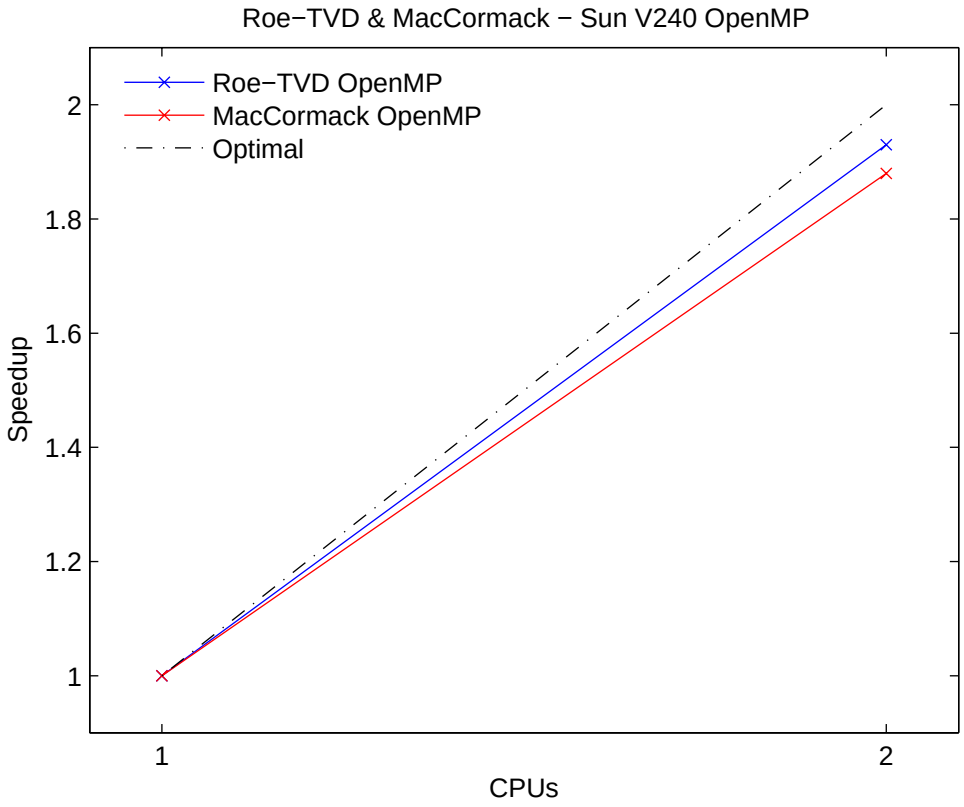
Δεδομένα: Grid: 300×300, $\mathrm{CFL} = 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.47: Πρόβλημα G: Roe-TVD, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	2830.31	1.00	100.00
2	1465.32	1.93	96.58

Πίνακας 6.48: Πρόβλημα G: MacCormack, Sun Fire V240 OpenMP

	OpenMP		
CPUs	Χρόνος	SU	Αποδ. %
1	1752.32	1.00	100.00
2	932.78	1.88	93.93



Σχήμα 6.32: Πρόβλημα G: Αποτελέσματα OpenMP στο Sun Fire V240.

6.4.25 Πρόβλημα G - SGI Origin OpenMP (Διάφορα grid)

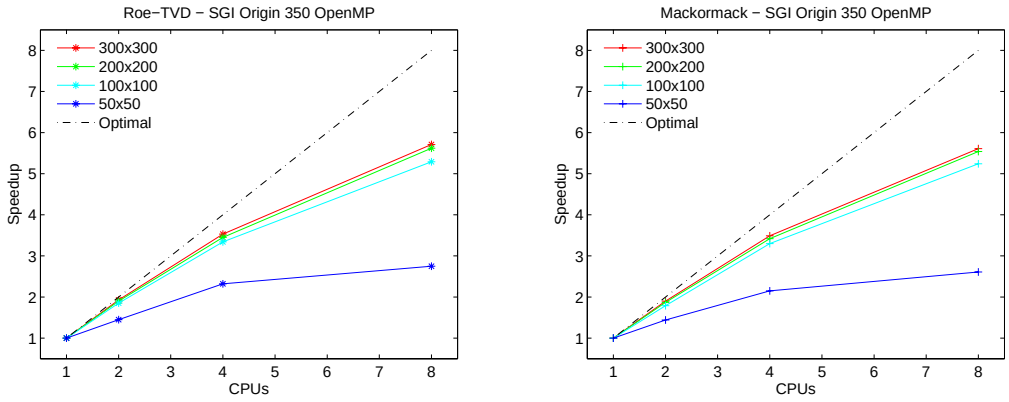
Δεδομένα: Grid: Διάφορα, $\mathrm{CFL} = 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.49: Πρόβλημα G: Roe-TVD, SGI Origin OpenMP (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	2018.62	1.00	507.03	1.00	45.42	1.00	3.64	1.00
2	1044.98	1.93	266.72	1.90	24.62	1.85	2.51	1.45
4	571.86	3.53	146.97	3.45	13.59	3.34	1.57	2.32
8	353.49	5.71	90.20	5.62	8.58	5.29	1.32	2.75

Πίνακας 6.50: Πρόβλημα G: MacCormack, SGI Origin OpenMP (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	1619.28	1.00	480.35	1.00	38.98	1.00	2.92	1.00
2	852.12	1.90	256.60	1.87	21.74	1.79	2.03	1.44
4	463.74	3.49	140.29	3.42	11.82	3.30	1.36	2.15
8	288.54	5.61	86.69	5.54	7.44	5.24	1.12	2.61



Σχήμα 6.33: Πρόβλημα G: Διάφορα grid σε OpenMP στο SGI Origin 350.

6.4.26 Πρόβλημα G - SGI Origin MPI (Διάφορα grid)

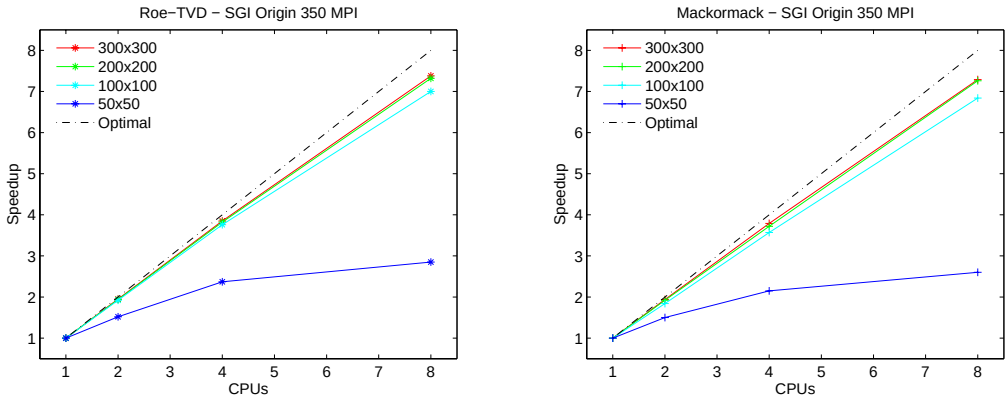
Δεδομένα: Grid: Διάφορα, $\mathrm{CFL} = 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.51: Πρόβλημα G: Roe-TVD, SGI Origin MPI (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	2018.62	1.00	507.03	1.00	45.42	1.00	3.64	1.00
2	1035.68	1.95	261.22	1.94	23.64	1.92	2.39	1.52
4	524.77	3.85	132.63	3.82	12.09	3.76	1.54	2.37
8	273.59	7.38	69.26	7.32	6.49	7.00	1.28	2.85

Πίνακας 6.52: Πρόβλημα G: MacCormack, SGI Origin MPI (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	1619.28	1.00	480.35	1.00	38.98	1.00	2.92	1.00
2	836.69	1.94	249.79	1.92	21.23	1.84	1.95	1.50
4	427.13	3.79	129.02	3.72	10.92	3.57	1.36	2.15
8	222.04	7.29	66.20	7.26	5.70	6.84	1.12	2.60



Σχήμα 6.34: Πρόβλημα G: Διάφορα grid σε MPI στο SGI Origin 350.

6.4.27 Πρόβλημα G - Sun Fire V240 MPI (Διάφορα grid)

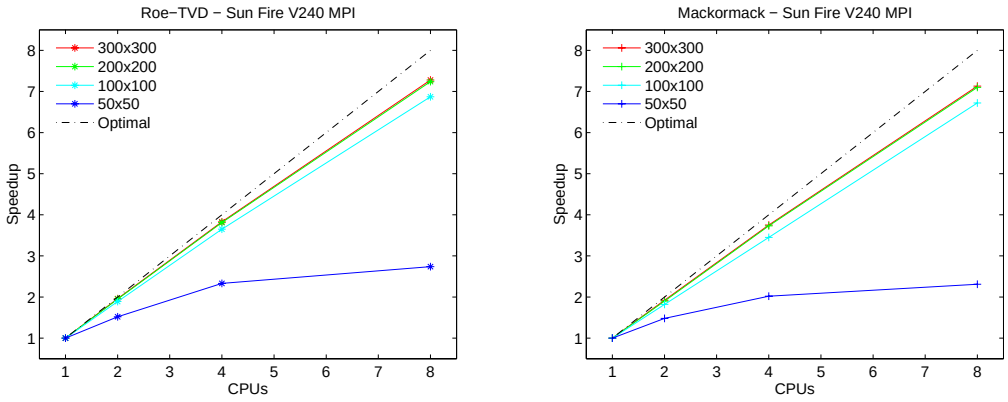
Δεδομένα: Grid: Διάφορα, $\mathrm{CFL} = 0.4$, Minmod περιοριστής-ροής

Πίνακας 6.53: Πρόβλημα G: Roe-TVD, Sun Fire V240 MPI (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	2449.01	1.00	617.13	1.00	56.48	1.00	4.42	1.00
2	1246.64	1.96	316.80	1.95	29.84	1.89	2.76	1.60
4	639.24	3.83	161.85	3.81	15.48	3.65	1.90	2.33
8	336.31	7.28	85.23	7.24	8.22	6.87	1.61	2.74

Πίνακας 6.54: Πρόβλημα G: MacCormack, Sun Fire V240 MPI (Διάφορα grid)

	300×300		200×200		100×100		50×50	
CPUs	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU	Χρόνος	SU
1	1752.32	1.00	520.04	1.00	52.30	1.00	4.09	1.00
2	911.02	1.92	273.42	1.90	28.72	1.82	2.71	1.51
4	467.32	3.75	139.38	3.73	15.15	3.45	2.03	2.02
8	245.93	7.13	73.22	7.10	7.78	6.72	1.77	2.31



Σχήμα 6.35: Πρόβλημα G: Διάφορα grid σε MPI στο Sun Fire V240.

Κεφάλαιο 7

Συμπεράσματα

Στα πλαίσια αυτής της εργασίας αρχικά έγινε μία πολύ σύντομη παρουσίαση των εξισώσεων ρηχών υδάτων (SW) στη μία και στις δύο διαστάσεις. Πραγματοποιήθηκε μία αναφορά σε σχήματα αριθμητικής επίλυσης των εξισώσεων SW τόσο για τη μία αλλά και για τις δύο διαστάσεις. Πιο συγκεκριμένα, για τη μία διάσταση, παρουσιάστηκαν τα σχήματα Lax-Friedrichs, MacCormack καθώς επίσης και το σχήμα Roe-TVD. Στις δύο διαστάσεις είδαμε τα αντοίστιχα σχήματα MacCormack και Roe-TVD.

Βασικό σκοπό αποτέλεσε η ανάπτυξη παράλληλων αλγορίθμων για τα δύο προαναφερθέντα σχήματα στις δύο διαστάσεις, όπου το υπολογιστικό κόστος είναι σημαντικό, και η δοκιμή τους όσο αφορά την αποδοτικότητά τους σε σύγκριση με τους αντοίστιχους σειριακούς. Για να επιτευχθεί αυτή η σύγκριση για πολλές και διαφορετικές περιπτώσεις προβλημάτων αναπτύχθηκε μια σειρά από οκτώ προβλήματα προς επίλυση. Τα προβλήματα αυτά δοκιμάστηκαν σε παράλληλες υλοποιήσεις των σχημάτων, βασισμένες στα λογισμικά πρότυπα OpenMP και MPI. Οι εκτελέσεις των προγραμμάτων πραγματοποιήθηκαν σε ένα πολυεπεξεργαστικό σύστημα SGI Origin 350 καθώς επίσης και σε μία συστάδα τεσσάρων Sun Fire V240 συνδεμένα είτε σε δίκτυο 1Gbit είτε σε δίκτυο 100Mbit.

Η προφανής διαπίστωση που προκύπτει από την παρούσα εργασία είναι ότι τα σχήματα αυτά κάθε αυτά ευνοούν την παραλληλοποίηση. Αυτό συμβαίνει κυρίως διότι αποτελούνται από τμηματικά ανεξάρτητες διαδικασίες και επιπλέον η πληροφορία η οποία πρέπει να μεταφερθεί από το ένα τμήμα στο άλλο είναι σχετικά μικρή. Οι δοκιμές έδειξαν ότι η υλοποίηση των αλγορίθμων για αρχιτεκτονικές μη-κοινής μνήμης απέφερε απόδοση η οποία προσεγγίζει τη θεωρητική βέλτιστη.

Θέλοντας να αναλύσει κανείς τα αποτελέσματα που προκύπτουν από τα δοκιμαστικά προβλήματα παρατηρεί τη σαφή υπεροχή της παραλληλοποίησης η οποία βασίζεται στο MPI. Σε όλες ανεξαιρέτως τις περιπτώσεις οι δείκτες του Speedup και της Αποδοτικότητας ήταν καλλίτεροι από τους αντίστοιχους του OpenMP. Το παραπάνω πόρισμα ισχύει και για τα δύο παράλληλα υπολογιστικά συστήματα που χρησιμοποιήθηκαν. Η διαφορά μάλιστα, θα μπορούσε να πει κανείς, είναι πάρα πολύ μεγάλη. Αυτό οφείλεται στις διαφορές των δύο προτύπων όπου από τη

μία το OpenMP προσφέρει την απόλυτη ευκολία κατά την υλοποίηση αλλά μηδενικό έλεγχο στον προγραμματιστή σε αντίθεση με το MPI όπου τα πάντα ορίζονται από αυτόν που αναπτύσει το πρόγραμμα. Αυτός ο έλεγχος τις επικοινωνίας και του συγχρονισμού είναι και ο λόγος της μεγάλης διαφοράς των δύο προτύπων στην απόδοση.

Μία άλλη παρατήρηση έχει να κάνει με τη σχέση που προκύπτει μεταξύ του χρόνου εκτέλεσης ενός προβλήματος και της απόδοσης της παράλληλης υλοποίησης. Μία σύγκριση που θα μπορούσε να κάνει ο αναγνώστης για να δει αυτήν την εξάρτηση είναι τα προβλήματα A και G. Και τα δύο προβλήματα εκτελούνται σε διαμέριση 300×300. Το πρόβλημα A όμως απαιτεί συγκριτικά με το πρόβλημα G πολύ μικρότερο χρόνο για την αριθμητική του επίλυση. Παρατηρούμε λοιπόν ότι στο πρόβλημα G και οι δύο δείκτες απόδοσης είναι καλλίτεροι. Για παράδειγμα στην υλοποίηση του σχήματος Roe-TVD σε MPI και εκτέλεση στο SGI Origin 350 σε 8 επεξεργαστές το πρόβλημα A απέφερε Speedup 7.38 έναντι 7.00 του προβλήματος G.

Ένα άλλο στοιχείο το οποίο επηρεάζει τους δείκτες απόδοσης των παράλληλων υλοποιήσεων είναι και η διαμέριση. Εάν ο χωρισμός του προβλήματος σε υποχωρία, τα οποία μοιράζονται οι επεξεργαστές, γίνει παράλληλα με τον άξονα του οποίου η διαμέριση είναι μικρότερη τότε η πληροφορία η οποία θα ανταλλάσεται μεταξύ των επεξεργαστών περιορίζεται στο ελάχιστο δυνατό. Για παράδειγμα ας εξετάσουμε την περίπτωση του προβλήματος C. Η διαμέριση που έχει χρησιμοποιηθεί είναι 60×280. Το πρόβλημα έχει χωριστεί παράλληλα προς την πλευρά την οποία έχουμε διαμερίσει σε μικρότερο αριθμό κομματιών, δηλαδή την πλευρά με τα 60 σημεία διαμέρισης. Με αυτόν τον τρόπο γίνεται μεταφορά μικρότερης πληροφορίας μιας και τα διανύσματα που πρέπει να σταλθούν από τον ένα επεξεργαστή στον άλλο είναι 60 θέσεων και όχι 280. Το πρόβλημα αυτό, επιπρόσθετα, απαιτεί μεγάλο χρόνο για την επίλυσή του. Συμβαδίζοντας λοιπόν και με την προηγούμενη παράγραφο, το πρόβλημα C αποφέρει τις καλλίτερες επιδόσεις για όλες τις περιπτώσεις παράλληλων υπολογιστικών συστημάτων και πακέτων παραλληλοποίησης.

Κατά τη χρήση διάφορων grid στο ίδιο πρόβλημα, παρατηρήθηκε καλλίτερη συμπεριφορά των δύο σχημάτων και στα δύο παράλληλα υπολογιστικά συστήματα στις περιπτώσεις των μεγάλων πλεγμάτων. Σε δοκιμή του προβλήματος G για πλέγματα 50×50, 100×100, 200×200 και 300×300 οι δείκτες απόδοσης αυξάνονται καθώς εκλεπτύνεται το αριθμητικό χωρίο. Το γεγονός αυτό οφείλεται στο ότι το υπολογιστικό κομμάτι, για μικρά grid, είναι μικρό σε σχέση με το κομμάτι της επικοινωνίας και του συγχρονισμού των επεξεργαστών.

Συγκριτικά για τα δύο αριθμητικά σχήματα και στις δύο υλοποιήσεις παρατηρείται μία ελαφρώς καλλίτερη συμπεριφορά του Roe-TVD σε σχέση με το σχήμα του MacCormack. Το γεγονός αυτό οφείλεται στην ανάγκη, όσο αφορά το MacCormack για συχνότερη αποστολή και λήψη δεδομένων. Τα διανύσματα τα οποία μεταφέρονται από τον ένα επεξεργαστή στον άλλο είναι πιο πολλά στο πλήθος. Αυτό έχει ως συνέπεια να απαιτείται συχνότερα συγχρονισμός των

επεξεργαστών με αποτέλεσμα να επιβραδύνεται η αριθμητική επίλυση των προβλημάτων.

Ένα γεγονός στο οποίο αξίζει να σταθεί κανείς είναι η σύγκριση των αποτελεσμάτων μεταξύ των δύο παράλληλων υπολογιστικών συστημάτων. Σε όλα τα δοκιμαστικά προβλήματα που παραθέσαμε στην παρούσα εργασία, οι δύο δείκτες απόδοσης, το Speedup και η Αποδοτικότητα, ήταν καλλίτερα στο SGI Origin 350 εν συγκρίσει με τα αντίστοιχα του grid των τεσσάρων Sun Fire V240. Το γεγονός αυτό είναι λογικό μιας και στο μηχάνημα της SGI και οι 8 επεξεργαστές βρίσκονται κάτω από το ίδιο λειτουργικό σύστημα. Στην περίπτωση των Sun Fire V240, κάθε ένα μηχάνημα έχει το δικό του λειτουργικό σύστημα και η σύνδεσή τους γίνεται από ένα απλό δίκτυο Ethernet.

Κοιτώντας κάποιος τα οικονομικά δεδομένα όσο αφορά το κόστος απόκτησης των παράλληλων υπολογιστών, η παραπάνω διαπίστωση, που αποτελεί ένα δεδομένο, δεν είναι το μοναδικό κριτήριο επιλογής μεταξύ ενός συστήματος τύπου SGI Origin 350 και ενός δικτυακού grid υπολογιστών. Όπως ειπώθηκε παραπάνω, η ανάπτυξη παράλληλου κώδικα βασισμένου στο πρότυπο MPI αποφέρει πολύ καλλίτερα αποτελέσματα από ότι αν αναπτυσσόταν σε OpenMP. Στη συστάδα με τα Sun Fire V240 οι επιδόσεις στο MPI υστερούν πολύ λίγο σε σχέση με το SGI Origin 350.

Θέλοντας να κλείσουμε την εργασία αυτή, στηριζόμενοι σε όλες τις παραπάνω διαπιστώσεις, καταλήγουμε στο συμπέρασμα ότι η ανάπτυξη παράλληλων αλγορίθμων, βασισμένων κατά κύριο λόγο στο MPI, για άμεσα (explicit) αριθμητικά σχήματα επίλυσης προβλημάτων εξισώσεων SW όπως το Roe-TVD και το MacCormack, αποτελεί μία πολύ αξιόλογη επιλογή έχοντας κανείς υπόψιν του τους δείκτες του Speedup και της Αποδοτικότητας. Τα μεγέθη των προβλημάτων που μπορούν να πραγματευτούν με τη χρήση παράλληλων υπολογιστικών συστημάτων μεγαλώνουν με ταυτόχρονη μείωση του χρόνου εκτέλεσης. Ως μελλοντική προοπτική θα μπορούσε κανείς να θέσει την παραλληλοποίηση και άλλων άμεσων σχημάτων όπως για παράδειγμα αυτό του Lax-Friedrichs καθώς επίσης να ανασυνταχθεί η έρευνα όσο αφορά τους παράλληλους αλγόριθμους έμμεσων (implicit) στο χρόνο σχημάτων επίλυσης με γνώμονα τα νέα μέσα της τεχνολογίας όσο αφορά τους παράλληλους αριθμητικούς υπολογισμούς.

Βιβλιογραφία

[1] Μαθιουδάκης Ε. *Επαναλληπτικές Μέθοδοι για την Επίλυση Μεγάλων Γραμμικών Συστημάτων σε Παράλληλες Αρχιτεκτονικές*. PhD Thesis, Πολυτεχνείο Κρήτης, 2001.

[2] Alcrudo, F. and Garcia-Navarro P. A High-Resolution Godunov-type Scheme in Finite Volumes for the 2D Shallow Water Equations. *Int. J. Numer. Math. Fluids*, 16:489–505, 1993.

[3] Caleffi V., Valiani A. and Zanni A. Finite Volume Method for Simulating Extreme Flood Events in Natural Channels. *Journal of Hydraulic Research*, 41(2):167–177, 2003.

[4] Chandra, R., Dagum L., Kohr, D., Maydan, D., McDonald, J. and Menon, R. *Parallel Programing in OpenMP*. Morgan Kaufmann Publishers, Inc., 2001.

[5] Chunbo J., Kai L., Ning L. and Qinghai Z. Implicit Parallel FEM Analysis of Shallow Water Equations. *Tsinghua Science & Technology*, 10(3):364–371, June 2005.

[6] Courant R., Friedrichs K. and Lewy H. Uber die Partiellen Differenzegleichungen der Mathematisches Physik. *Math. Ann.*, 100:32–74, 1928.

[7] Dafermos, C. *Hyperbolic Conservation Laws in Continuum Physics*. Grundlehrer Math., 2000.

[8] Delis A. Improved Application of the HLLE Rieman Solver for the Shallow Water Equations with Source Terms. *Comm. Numer. Meth. Engng*, 19:59–83, 2003.

[9] Delis A. and Katsaounis Th. Relaxation Schemes for the Shallow Water Equations. *Int. J. Numer. Meth. Fluids*, 41:695–719, 2003.

[10] Delis A., Skeels C. and Ryrie S. Implicit High-Resolution Methods for Modeling One-Dimensional Open Channel Flow. *Journal of Hydraulic Research*, 38(5):369–382, 2000.

[11] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A. *Sourcebook of Parallel Computing*. Morgan Kaufmann Publishers, Inc., 2003.

[12] Fennema, R. J. and Chaudhry, M. H. Explicit Methods for 2-D Transient Free-Surface Flows. *American Society of Civil Engineers, Journal of Hydraulic Engineering*, 116:1013–1034, 1990.

[13] Fiedler R. and Ramirez A. A Numerical Method for Simulating Discontinuous Shallow Flow over an Infiltrating Surface. *Int. J. Numer. Meth. Fluids*, 32:219–240, 2000.

[14] Güler I., Behr M. and Tezduyar T. Parallel Finite Element Computation of Free-Surface Flows. *Computational Mechanics*, 23:117–123, 1999.

[15] Gwo-Fong L., Jihn-Sung L. and Wen-Dar G. Finite-Volume Component-Wise TVD Schemes for 2D Shallow Water Equations. *Advances in Water Resources*, 26:861–873, 2003.

[16] Harten, A. High Resolution Schemes for Conservation Laws. *J. Comput. Phys.*, 49, 1983.

[17] Hubbard, M. E. and Garcia-Navarro P. Flux Difference Splitting and the Balancing of Source Terms and Flux Gradients. *Numerical Analysis Report*, 3/99, 1993.

[18] Kumar, V., Grama, A., Gupta, A. and Karypis, G. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamon Cummings Inc., 1994.

[19] LeVeque, R. J. *Numerical Methods for Conservational Laws*. Birkhäuser Verlag, 1992.

[20] LeVeque, R. J. Balancing Source Terms and Flux Gradients in High-Resolution Godunov Methods: The Quasi-Steady Wave-Propagation Algorithm. *J. Comput. Phys.*, 146, 1998.

[21] LeVeque, R. J. and Yee, H. C. A Study of Numerical Methods for Hyperbolic Conservation Laws with Stiff Source Terms. *J. Comput. Phys.*, 86:187–210, 1990.

[22] Mathioudakis, E. and Papadopoulou, E.. MPI Managment of Hermite Collocation Computation on a Distributed-Shared Memory System. *Procs of the 9th WSEAS Int. Conf. on Applied Mathematics*, Istanbul, Turkey, 2006.

[23] Mathioudakis, E., Papadopoulou, E. and Sadidakis, Y. Mapping Parallel Algorithm for PDE Computations on a Distributed Memory Computers. *Parallel Alg. and Appl.*, 8:141–154, 1996.

[24] Mathioudakis, E., Papadopoulou, E. and Sadidakis, Y. Bi-CGSTAB for Collocation Equations on Distributed Memory Parallel Computers. *Numerical Mathematics and Advanced Applications - ENUMATH 2001*, 957–966, 2003.

[25] Mathioudakis, E., Papadopoulou, E. and Sadidakis, Y. Iterative Solution of Elliptic Collocation Systems on a Cognitive Parallel Computer. *Computers and Mathematics with Applications*, 48:951–970, 2004.

[26] MPI website: `http://www.mpi-forum.org/`

[27] OpenMP website: `http://www.openmp.org/`

[28] Pacheco, P. S. *Parallel Programming with MPI*. Morgan Kaufmann Publishers, Inc., 1997.

[29] Paglieri L., Ambrosi D., Formaggia L., Quarteroni A. and Scheinine A. L. Parallel Computation for Shallow Water Flow: A Domain Decomposition Approach. *Parallel Computing*, 23(9):1261–1277, September 1997.

[30] Rao, P. A Parallel Hydrodynamic Model for Shallow Water Equations. *Appl. Math. Comput.*, 150(1):291–302, 2004.

[31] Roe, P. L. Approximate Rieman Solvers, Parameter Vectors and Difference Schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[32] Sweby, P. K. Source Terms and Conservation Laws: A Preliminary Discussion. *Numerical Analysis Report*, 6/89, 1989.

[33] Toro, E. *Riemann Solvers and Numerical Methods for Fluid Dynamiics*. Springer-Verlag, 1999.

[34] Zoppou C. and Roberts S. Numerical Solution of the Two-Dimensional Unsteady Dam Break. *Applied Mathematics Modeling*, 24:457–475, 2000.

Παράρτημα Α΄

Πηγαίος Κώδικας

Α΄.1 Πηγαίος κώδικας του Roe OpenMP

```
program Roe2D_OMP

      implicit none
      integer xst, yst, ictype, bctype, FLAG, k, flim, i, j
      integer I1, I2, I3, J1, J2, J3, n, m
      parameter (n = 300)
      parameter (m = 300)
      real dtime, tm, ta(2)
      double precision ax, bx, ay, by, tfinal, CFL, Lx, Ly, pi, g, dx
      double precision dy, dt
      double precision u(-5:n+5,-5:m+5), h(-5:n+5,-5:m+5)
      double precision v(-5:n+5,-5:m+5)
      double precision uav, vav, havf, havg, cf, cg
      double precision lf1(-5:n+5,-5:m+5), lf2(-5:n+5,-5:m+5)
      double precision lf3(-5:n+5,-5:m+5), lg1(-5:n+5,-5:m+5)
      double precision lg2(-5:n+5,-5:m+5), lg3(-5:n+5,-5:m+5)
      double precision ef11, ef12(-5:n+5,-5:m+5)
      double precision ef13(-5:n+5,-5:m+5), ef21
      double precision ef22, ef23
      double precision eg11, eg12(-5:n+5,-5:m+5)
      double precision eg13(-5:n+5,-5:m+5), eg21
      double precision eg22, eg23
      double precision eg31, eg32(-5:n+5,-5:m+5)
      double precision eg33(-5:n+5,-5:m+5), ef31
      double precision ef32(-5:n+5,-5:m+5), ef33(-5:n+5,-5:m+5)
      double precision af1(-5:n+5,-5:m+5), af2(-5:n+5,-5:m+5)
      double precision af3(-5:n+5,-5:m+5), ag1(-5:n+5,-5:m+5)
      double precision ag2(-5:n+5,-5:m+5), ag3(-5:n+5,-5:m+5)
      double precision nf1(-5:n+5,-5:m+5), nf2(-5:n+5,-5:m+5)
      double precision nf3(-5:n+5,-5:m+5), ng1(-5:n+5,-5:m+5)
      double precision ng2(-5:n+5,-5:m+5), ng3(-5:n+5,-5:m+5)
      double precision thetaf1, thetaf2, thetaf3,thetag1,thetag2,thetag3
      double precision phif1(-5:n+5,-5:m+5), phif2(-5:n+5,-5:m+5)
```

```
      double precision phif3(-5:n+5,-5:m+5), phig1(-5:n+5,-5:m+5)
      double precision phig2(-5:n+5,-5:m+5), phig3(-5:n+5,-5:m+5)
      double precision bf1(-5:n+5,-5:m+5), bf2
      double precision bf3(-5:n+5,-5:m+5), bg1(-5:n+5,-5:m+5)
      double precision bg2, bg3(-5:n+5,-5:m+5)
      double precision fp1(-5:n+5,-5:m+5), fp2(-5:n+5,-5:m+5)
      double precision fp3(-5:n+5,-5:m+5), fm1(-5:n+5,-5:m+5)
      double precision fm2(-5:n+5,-5:m+5), fm3(-5:n+5,-5:m+5)
      double precision gp1(-5:n+5,-5:m+5), gp2(-5:n+5,-5:m+5)
      double precision gp3(-5:n+5,-5:m+5), gm1(-5:n+5,-5:m+5)
      double precision gm2(-5:n+5,-5:m+5), gm3(-5:n+5,-5:m+5)
      double precision fst1(-5:n+5,-5:m+5), fst2(-5:n+5,-5:m+5)
      double precision fst3(-5:n+5,-5:m+5), gst1(-5:n+5,-5:m+5)
      double precision gst2(-5:n+5,-5:m+5), gst3(-5:n+5,-5:m+5)
      double precision F1(-5:n+5,-5:m+5), F2(-5:n+5,-5:m+5)
      double precision F3(-5:n+5,-5:m+5), G1(-5:n+5,-5:m+5)
      double precision G2(-5:n+5,-5:m+5), G3(-5:n+5,-5:m+5)
      double precision w1(-5:n+5,-5:m+5), w2(-5:n+5,-5:m+5)
      double precision w3(-5:n+5,-5:m+5), B(-5:n+5,-5:m+5)
      double precision xind(n), yind(n), yytype, yy
      double precision time, timet, lfmax, lgmax, lim, eps, x, y, delta
      parameter (eps = 1e-8)

      open (20, file = 'input.dat')
      read (20,*) ax, bx, ay, by, xst, yst, tfinal, CFL, ictype,
     +  bctype, flim, g, yytype, delta
      close(20, status = 'keep')
      Lx = bx - ax
      Ly = by - ay
      dx = Lx/dble(xst)
      dy = Ly/dble(yst)
      pi = 4.0d0*datan(1.0d0)

C$OMP PARALLEL DEFAULT(SHARED)
      call ics(n, ax, dx, ay, dy, xst, yst, ictype, h, u, v, B)
C$OMP END PARALLEL

      dt = 0.0d0
      time = 0.0d0
      FLAG = 0
      k = 0

      tm = dtime(ta)

      ef11 = 1.0d0
      ef21 = 0.0d0
      ef22 = 0.0d0
      ef23 = 1.0d0
      ef31 = 1.0d0
      eg11 = 1.0d0
      eg21 = 0.0d0
```

```
      eg22 = 1.0d0
      eg23 = 0.0d0
      eg31 = 1.0d0

      bf2 = 0.0d0
      bg2 = 0.0d0

      do while ((time .le. tfinal) .and. (FLAG .ne. 2))
         k = k + 1
C$OMP PARALLEL DEFAULT(SHARED)
         call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)
C$OMP END PARALLEL

C$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(uav, vav, havf, havg)
C$OMP& PRIVATE(cf, cg)
         do i = -5, xst + 5
            do j = -5, yst + 5
               w1(i,j) = h(i,j)
               w2(i,j) = u(i,j)*h(i,j)
               w3(i,j) = v(i,j)*h(i,j)

               if (dsqrt(h(i+1,j)) + dsqrt(h(i,j)) .eq. 0.0d0)then
               uav = 0.0d0
               else
               uav = (dsqrt(h(i+1,j))*u(i+1,j) + dsqrt(h(i,j))*
     +           u(i,j))/(dsqrt(h(i+1,j)) + dsqrt(h(i,j)))
               end if
               if (dsqrt(h(i,j+1)) + dsqrt(h(i,j)) .eq. 0.0d0)then
               vav = 0.0d0
               else
               vav = (dsqrt(h(i,j+1))*v(i,j+1) + dsqrt(h(i,j))*
     +           v(i,j))/(dsqrt(h(i,j+1)) + dsqrt(h(i,j)))
               end if

               havf = 0.5d0*(h(i + 1,j) + h(i,j))
               havg = 0.5d0*(h(i,j + 1) + h(i,j))

               cf = dsqrt(g*havf)
               cg = dsqrt(g*havg)

               lf1(i,j) = uav - cf
               lf2(i,j) = uav
               lf3(i,j) = uav + cf

               ef12(i,j) = uav - cf
               ef13(i,j) = vav
               ef32(i,j) = uav + cf
               ef33(i,j) = vav

               lg1(i,j) = vav - cg
               lg2(i,j) = vav
```

```
               lg3(i,j) = vav + cg

               eg12(i,j) = uav
               eg13(i,j) = vav - cg
               eg32(i,j) = uav
               eg33(i,j) = vav + cg

               bf1(i,j) = 0.5d0*cf*(B(i+1,j) - B(i,j))
               bf3(i,j) = -0.5d0*cf*(B(i+1,j) - B(i,j))

               bg1(i,j) = 0.5d0*cg*(B(i,j+1) - B(i,j))
               bg3(i,j) = -0.5d0*cg*(B(i,j+1) - B(i,j))

               if (cf .eq. 0.0d0) then
                  af1(i,j) = 0.0d0
                  af2(i,j) = 0.0d0
                  af3(i,j) = 0.0d0
               else
                  af1(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) + (0.5d0/(cf))*
     +             (uav*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
                  af2(i,j) = (1.0d0/cf)*((v(i+1,j)*h(i+1,j) - v(i,j)*
     +             h(i,j)) - (vav*(h(i+1,j) - h(i,j))))
                  af3(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) - (0.5d0/(cf))*
     +             (uav*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
               end if

               if (cg .eq. 0.0d0) then
                  ag1(i,j) = 0.0d0
                  ag2(i,j) = 0.0d0
                  ag3(i,j) = 0.0d0
               else
                  ag1(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) + (0.5d0/(cg))*
     +             (vav*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
                  ag2(i,j) = (1.0d0/cg)*((u(i,j+1)*h(i,j+1) - u(i,j)*
     +             h(i,j)) - (uav*(h(i,j+1) - h(i,j))))
                  ag3(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) - (0.5d0/(cg))*
     +             (vav*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
               end if
            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(I1, I2, I3, J1, J2, J3)
C$OMP& PRIVATE(thetaf1, thetaf2, thetaf3, thetag1, thetag2, thetag3)
         do i = -5, xst + 5
            do j = -5, yst + 5
               nf1(i,j) = (dt/dx)*lf1(i,j)
```

```
nf2(i,j) = (dt/dx)*lf2(i,j)
nf3(i,j) = (dt/dx)*lf3(i,j)

ng1(i,j) = (dt/dy)*lg1(i,j)
ng2(i,j) = (dt/dy)*lg2(i,j)
ng3(i,j) = (dt/dy)*lg3(i,j)

if (nf1(i,j) .eq. 0.0d0) then
   I1 = i
else
   I1 = i - nf1(i,j)/dabs(nf1(i,j))
end if
if (nf2(i,j) .eq. 0.0d0) then
   I2 = i
else
   I2 = i - nf2(i,j)/dabs(nf2(i,j))
end if
if (nf3(i,j) .eq. 0.0d0) then
   I3 = i
else
   I3 = i - nf3(i,j)/dabs(nf3(i,j))
end if
if (ng1(i,j) .eq. 0.0d0) then
   J1 = j
else
   J1 = j - ng1(i,j)/dabs(ng1(i,j))
end if
if (ng2(i,j) .eq. 0.0d0) then
   J2 = j
else
   J2 = j - ng2(i,j)/dabs(ng2(i,j))
end if
if (ng3(i,j) .eq. 0.0d0) then
   J3 = j
else
   J3 = j - ng3(i,j)/dabs(ng3(i,j))
end if

thetaf1 = (af1(I1,j))/(af1(i,j) + eps)
thetaf2 = (af2(I2,j))/(af2(i,j) + eps)
thetaf3 = (af3(I3,j))/(af3(i,j) + eps)

thetag1 = (ag1(i,J1))/(ag1(i,j) + eps)
thetag2 = (ag2(i,J2))/(ag2(i,j) + eps)
thetag3 = (ag3(i,J3))/(ag3(i,j) + eps)

phif1(i,j) = lim(thetaf1, flim)
phif2(i,j) = lim(thetaf2, flim)
phif3(i,j) = lim(thetaf3, flim)

phig1(i,j) = lim(thetag1, flim)
```

```
            phig2(i,j) = lim(thetag2, flim)
            phig3(i,j) = lim(thetag3, flim)

            fp1(i,j) = 0.5d0*(bf1(i,j)*ef11*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef21*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef31*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fp2(i,j) = 0.5d0*(bf1(i,j)*ef12(i,j)*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef22*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef32(i,j)*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fp3(i,j) = 0.5d0*(bf1(i,j)*ef13(i,j)*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef23*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef33(i,j)*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm1(i,j) = 0.5d0*(bf1(i,j)*ef11*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef21*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef31*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm2(i,j) = 0.5d0*(bf1(i,j)*ef12(i,j)*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef22*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef32(i,j)*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm3(i,j) = 0.5d0*(bf1(i,j)*ef13(i,j)*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j))))) + bf2*ef23*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j))))) + bf3(i,j)*ef33(i,j)*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))
```

```
              gp1(i,j) = 0.5d0*(bg1(i,j)*eg11*(1.0d0 + lg1(i,j)/
     +          (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 - dabs
     +          (ng1(i,j)))))) + bg2*eg21*(1.0d0 + lg2(i,j)/
     +          (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 - dabs
     +          (ng2(i,j)))))) + bg3(i,j)*eg31*(1.0d0 + lg3(i,j)/
     +          (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 - dabs
     +          (ng3(i,j))))))

              gp2(i,j) = 0.5d0*(bg1(i,j)*eg12(i,j)*(1.0d0 + lg1(i,j)/
     +            (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +            (ng1(i,j)))))) + bg2*eg22*(1.0d0 + lg2(i,j)/
     +            (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +            (ng2(i,j)))))) + bg3(i,j)*eg32(i,j)*(1.0d0 + lg3(i,j)/
     +            (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +            (ng3(i,j))))))

              gp3(i,j) = 0.5d0*(bg1(i,j)*eg13(i,j)*(1.0d0 + lg1(i,j)/
     +            (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +            (ng1(i,j)))))) + bg2*eg23*(1.0d0 + lg2(i,j)/
     +            (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +            (ng2(i,j)))))) + bg3(i,j)*eg33(i,j)*(1.0d0 + lg3(i,j)/
     +            (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +            (ng3(i,j))))))

              gm1(i,j) = 0.5d0*(bg1(i,j)*eg11*(1.0d0 - lg1(i,j)/
     +          (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 - dabs
     +          (ng1(i,j)))))) + bg2*eg21*(1.0d0 - lg2(i,j)/
     +          (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 - dabs
     +          (ng2(i,j)))))) + bg3(i,j)*eg31*(1.0d0 - lg3(i,j)/
     +          (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 - dabs
     +          (ng3(i,j))))))

              gm2(i,j) = 0.5d0*(bg1(i,j)*eg12(i,j)*(1.0d0 - lg1(i,j)/
     +            (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +            (ng1(i,j)))))) + bg2*eg22*(1.0d0 - lg2(i,j)/
     +            (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +            (ng2(i,j)))))) + bg3(i,j)*eg32(i,j)*(1.0d0 - lg3(i,j)/
     +            (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +            (ng3(i,j))))))

              gm3(i,j) = 0.5d0*(bg1(i,j)*eg13(i,j)*(1.0d0 - lg1(i,j)/
     +            (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +            (ng1(i,j)))))) + bg2*eg23*(1.0d0 - lg2(i,j)/
     +            (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +            (ng2(i,j)))))) + bg3(i,j)*eg33(i,j)*(1.0d0 - lg3(i,j)/
     +            (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +            (ng3(i,j))))))
           end do
         end do
C$OMP END PARALLEL DO
```

```
C$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(yytype)
         do i = -5, xst + 5
            do j = -5 , yst + 5
               fst1(i,j) = fm1(i,j) + fp1(i-1,j)
               fst2(i,j) = fm2(i,j) + fp2(i-1,j)
               fst3(i,j) = fm3(i,j) + fp3(i-1,j)

               gst1(i,j) = gm1(i,j) + gp1(i,j-1)
               gst2(i,j) = gm2(i,j) + gp2(i,j-1)
               gst3(i,j) = gm3(i,j) + gp3(i,j-1)

               F1(i,j) = 0.5d0*(u(i+1,j)*h(i+1,j) + u(i,j)*h(i,j)) -
     +          0.5d0*(af1(i,j)*yy(lf1(i,j), delta, af1(i,j), yytype)*
     +          (1.0d0 -phif1(i,j)*(1.0d0 - dabs(nf1(i,j))))*ef11 +
     +          af2(i,j)*yy(lf2(i,j), delta, af2(i,j), yytype)*(1.0d0 -
     +          phif2(i,j)*(1.0d0 -dabs(nf2(i,j))))*ef21 +af3(i,j)*
     +          yy(lf3(i,j), delta, af3(i,j), yytype)*(1.0d0 -
     +          phif3(i,j)*(1.0d0 - dabs(nf3(i,j))))*ef31)

               F2(i,j) = 0.5d0*(h(i+1,j)*u(i+1,j)**2 + 0.5d0*g*
     +          h(i+1,j)**2 + h(i,j)*u(i,j)**2 + 0.5d0*g*h(i,j)**2) -
     +          0.5d0*(af1(i,j)*yy(lf1(i,j), delta, af1(i,j), yytype)*
     +          (1.0d0 - phif1(i,j)*(1.0d0 - dabs(nf1(i,j))))*
     +          ef12(i,j) + af2(i,j)*yy(lf2(i,j), delta, af2(i,j),
     +          yytype)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs(nf2(i,j))))*
     +          ef22 + af3(i,j)*yy(lf3(i,j), delta, af3(i,j),
     +          yytype)*(1.0d0 - phif3(i,j)*(1.0d0 - dabs(nf3(i,j))))*
     +          ef32(i,j))


               F3(i,j) = 0.5d0*(u(i+1,j)*v(i+1,j)*h(i+1,j) +u(i,j)*
     +           v(i,j)*h(i,j)) - 0.5d0*(af1(i,j)*yy(lf1(i,j), delta,
     +           af1(i,j), yytype)*(1.0d0 - phif1(i,j)*(1.0d0 -
     +           dabs(nf1(i,j))))*ef13(i,j) + af2(i,j)*yy(lf1(i,j),
     +           delta, af1(i,j), yytype)*(1.0d0-phif2(i,j)*(1.0d0 -
     +           dabs(nf2(i,j))))*ef23 + af3(i,j)*yy(lf1(i,j),
     +           delta, af1(i,j), yytype)*(1.0d0-phif3(i,j)*(1.0d0-
     +           dabs(nf3(i,j))))*ef33(i,j))

               G1(i,j) = 0.5d0*(v(i,j+1)*h(i,j+1) + v(i,j)*h(i,j)) -
     +           0.5d0*(ag1(i,j)*yy(lg1(i,j), delta, ag1(i,j), yytype)*
     +           (1.0d0- phig1(i,j)*(1.0d0 - dabs(ng1(i,j))))*eg11
     +           + ag2(i,j)*yy(lg2(i,j), delta, ag2(i,j), yytype)*(1.0d0
     +           - phig2(i,j)*(1.0d0 -dabs(ng2(i,j))))*eg21 +
     +           ag3(i,j)*yy(lg3(i,j), delta, ag3(i,j), yytype)*(1.0d0 -
     +           phig3(i,j)*(1.0d0 - dabs(ng3(i,j))))*eg31)

               G2(i,j) = 0.5d0*(u(i,j+1)*v(i,j+1)*h(i,j+1) + u(i,j)*
     +           v(i,j)*h(i,j)) - 0.5d0*(ag1(i,j)*yy(lg1(i,j), delta,
     +           ag1(i,j), yytype)*(1.0d0 - phig1(i,j)*(1.0d0 -
     +           dabs(ng1(i,j))))*eg12(i,j) + ag2(i,j)*yy(lg2(i,j),
```

```
     +           delta, ag2(i,j), yytype)*(1.0d0 - phig2(i,j)*(1.0d0 -
     +           dabs(ng2(i,j))))*eg22 + ag3(i,j)*yy(lg3(i,j),
     +           delta, ag3(i,j), yytype)*(1.0d0-phig3(i,j)*(1.0d0 -
     +           dabs(ng3(i,j))))*eg32(i,j))

               G3(i,j) = 0.5d0*(h(i,j+1)*v(i,j+1)**2 + 0.5d0*g*
     +          h(i,j+1)**2 + h(i,j)*v(i,j)**2 + 0.5d0*g*h(i,j)**2) -
     +          0.5d0*(ag1(i,j)*yy(lg1(i,j), delta, ag1(i,j), yytype)*
     +          (1.0d0 - phig1(i,j)*(1.0d0 - dabs(ng1(i,j))))*eg13(i,j)
     +          + ag2(i,j)*yy(lg2(i,j), delta, ag2(i,j), yytype)*(1.0d0
     +          - phig2(i,j)*(1.0d0 -dabs(ng2(i,j))))*eg23 +
     +          ag3(i,j)*yy(lg3(i,j), delta, ag3(i,j), yytype)*(1.0d0 -
     +          phig3(i,j)*(1.0d0 - dabs(ng3(i,j))))*eg33(i,j))
            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED)
         do i = 0, xst
            do j = 0, yst
               w1(i,j) = w1(i,j) - (dt/dx)*(F1(i,j) - F1(i-1,j)) -
     +          (dt/dy)*(G1(i,j) - G1(i,j-1)) + (dt/dx)*fst1(i,j) +
     +          (dt/dy)*gst1(i,j)

               w2(i,j) = w2(i,j) - (dt/dx)*(F2(i,j) - F2(i-1,j)) -
     +          (dt/dy)*(G2(i,j) - G2(i,j-1)) + (dt/dx)*fst2(i,j) +
     +          (dt/dy)*gst2(i,j)

               w3(i,j) = w3(i,j) - (dt/dx)*(F3(i,j) - F3(i-1,j)) -
     +          (dt/dy)*(G3(i,j) - G3(i,j-1)) + (dt/dx)*fst3(i,j) +
     +          (dt/dy)*gst3(i,j)

               h(i,j) = w1(i,j)
               u(i,j) = w2(i,j)/(w1(i,j))
               v(i,j) = w3(i,j)/(w1(i,j))
            end do
         end do
C$OMP END PARALLEL DO

         lfmax = 0.0d0
         lgmax = 0.0d0

C$OMP DO
         do i = 0, xst
            do j = 0, yst
               lfmax = dmax1(lfmax, dabs(lf1(i,j)), dabs(lf2(i,j)),
     +           dabs(lf3(i,j)))
               lgmax = dmax1(lgmax, dabs(lg1(i,j)), dabs(lg2(i,j)),
     +           dabs(lg3(i,j)))
             end do
          end do
```

```
C$OMP END DO

         dt = (CFL*dmin1(dx,dy))/(dmax1(lfmax, lgmax))
         time = time + dt
         if (time .gt. tfinal) then
            timet = time - dt
            dt = tfinal - timet
            time = timet + dt
            FLAG = FLAG + 1
         end if
      end do

C$OMP PARALLEL DEFAULT(SHARED)
      call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)
C$OMP END PARALLEL
      tm = dtime(ta)
print*, 'Running Time = ', tm, 'seconds'

      open(10, file = 'xind.dat')
      open(11, file = 'yind.dat')
      open(12, file = 'hind.dat')
      open(13, file = 'uind.dat')
      open(14, file = 'vind.dat')
      open(15, file = 'bind.dat')
      open(16, file = 'st.dat')

      write (16,*) xst
      write (16,*) yst
      do i = 0, xst
         xind(i) = ax + dble(i)*dx
         write (10,*) xind(i)
      end do

      do j = 0, yst
         yind(j) = ay + dble(j)*dy
         write (11,*) yind(j)
      end do

      do j = 0, yst
         do i = 0, xst
            write (12,*) h(i,j)
            write (13,*) u(i,j)
            write (14,*) v(i,j)
            write (15,*) B(i,j)
         end do
      end do

      close (10, status = 'keep')
      close (11, status = 'keep')
      close (12, status = 'keep')
      close (13, status = 'keep')
```

```
close (14, status = 'keep')
close (15, status = 'keep')
close (16, status = 'keep')

stop
end
```

Α΄.2 Πηγαίος κώδικας του MacCormack OpenMP

```
program Mc2D_OMP

implicit none
integer xst, yst, ictype, bctype, FLAG, k, slim, i, j, n, m
parameter (n = 325)
parameter (n = 325)
real dtime, tm, ta(2)
double precision ax, bx, ay, by, tfinal, CFL, Lx, Ly, pi, g, dx
double precision dy, dt, Qx, Qy
double precision u(-5:n+5,-5:m+5), h(-5:n+5,-5:m+5)
double precision v(-5:n+5,-5:m+5)
double precision c
double precision lf1(-5:n+5,-5:m+5), lf2(-5:n+5,-5:m+5)
double precision lf3(-5:n+5,-5:m+5), lg1(-5:n+5,-5:m+5)
double precision lg2(-5:n+5,-5:m+5), lg3(-5:n+5,-5:m+5)
double precision ef11, ef12(-5:n+5,-5:m+5)
double precision ef13(-5:n+5,-5:m+5), ef21
double precision ef22, ef23
double precision eg11, eg12(-5:n+5,-5:m+5)
double precision eg13(-5:n+5,-5:m+5), eg21
double precision eg22, eg23
double precision eg31, eg32(-5:n+5,-5:m+5)
double precision eg33(-5:n+5,-5:m+5), ef31
double precision ef32(-5:n+5,-5:m+5), ef33(-5:n+5,-5:m+5)
double precision af1(-5:n+5,-5:m+5), af2(-5:n+5,-5:m+5)
double precision af3(-5:n+5,-5:m+5), ag1(-5:n+5,-5:m+5)
double precision ag2(-5:n+5,-5:m+5), ag3(-5:n+5,-5:m+5)
double precision nf1(-5:n+5,-5:m+5), nf2(-5:n+5,-5:m+5)
double precision nf3(-5:n+5,-5:m+5), ng1(-5:n+5,-5:m+5)
double precision ng2(-5:n+5,-5:m+5), ng3(-5:n+5,-5:m+5)
double precision thetaf1, thetaf2, thetaf3,thetag1,thetag2,thetag3
double precision phif1(-5:n+5,-5:m+5), phif2(-5:n+5,-5:m+5)
double precision phif3(-5:n+5,-5:m+5), phig1(-5:n+5,-5:m+5)
double precision phig2(-5:n+5,-5:m+5), phig3(-5:n+5,-5:m+5)
double precision bf1(-5:n+5,-5:m+5), bf2(-5:n+5,-5:m+5)
double precision bf3(-5:n+5,-5:m+5), bg1(-5:n+5,-5:m+5)
double precision bg2(-5:n+5,-5:m+5), bg3(-5:n+5,-5:m+5)
double precision F1(-5:n+5,-5:m+5), F2(-5:n+5,-5:m+5)
double precision F3(-5:n+5,-5:m+5), G1(-5:n+5,-5:m+5)
double precision G2(-5:n+5,-5:m+5), G3(-5:n+5,-5:m+5)
double precision w1(-5:n+5,-5:m+5), w2(-5:n+5,-5:m+5)
```

```
      double precision w3(-5:n+5,-5:m+5), B(-5:n+5,-5:m+5)
      double precision Ft1(-5:n+5,-5:m+5), Ft2(-5:n+5,-5:m+5)
      double precision Ft3(-5:n+5,-5:m+5), Gt1(-5:n+5,-5:m+5)
      double precision Gt2(-5:n+5,-5:m+5), Gt3(-5:n+5,-5:m+5)
      double precision wt1(-5:n+5,-5:m+5), wt2(-5:n+5,-5:m+5)
      double precision wt3(-5:n+5,-5:m+5)
      double precision wtt1(-5:n+5,-5:m+5), wtt2(-5:n+5,-5:m+5)
      double precision wtt3(-5:n+5,-5:m+5)
      double precision Df1(-5:n+5,-5:m+5), Df2(-5:n+5,-5:m+5)
      double precision Df3(-5:n+5,-5:m+5)
      double precision Dg1(-5:n+5,-5:m+5), Dg2(-5:n+5,-5:m+5)
      double precision Dg3(-5:n+5,-5:m+5)
      double precision ut(-5:n+5,-5:m+5), ht(-5:n+5,-5:m+5)
      double precision vt(-5:n+5,-5:m+5)
      double precision R1(-5:n+5,-5:m+5), R2(-5:n+5,-5:m+5)
      double precision R3(-5:n+5,-5:m+5), Rt1(-5:n+5,-5:m+5)
      double precision Rt2(-5:n+5,-5:m+5), Rt3(-5:n+5,-5:m+5)
      double precision xind(n), yind(n)
      double precision time, timet, lmax, lfmax, lgmax, lim, eps, x, y
      double precision dBx(-5:n+5,-5:m+5), dBy(-5:n+5,-5:m+5)
      parameter (eps = 1e-8)

      open (20, file = 'input.dat')
      read (20,*) ax, bx, ay, by, xst, yst, tfinal, CFL, ictype,
     +  bctype, slim, g
      close(20, status = 'keep')
      Lx = bx - ax
      Ly = by - ay
      dx = Lx/dble(xst)
      dy = Ly/dble(yst)
      pi = 4.0d0*datan(1.0d0)

C$OMP PARALLEL DEFAULT(SHARED)
      call ics(n, ax, dx, ay, dy, xst, yst, ictype, h, u, v, B,!)
     + dBx, dBy)
C$OMP END PARALLEL

      dt = 0.0d0
      time = 0.0d0
      FLAG = 0
      k = 0
      ef11 = 1.0d0
      ef21 = 0.0d0
      ef22 = 0.0d0
      ef23 = 1.0d0
      ef31 = 1.0d0
      eg11 = 1.0d0
      eg21 = 0.0d0
      eg22 = 1.0d0
      eg23 = 0.0d0
      eg31 = 1.0d0
```

```
tm = dtime(ta)

      do while ((time .le. tfinal) .and. (FLAG .ne. 2))
         k = k + 1

C$OMP PARALLEL DEFAULT(SHARED)
         call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)
C$OMP END PARALLEL

C$OMP PARALLEL DO DEFAULT(SHARED), PRIVATE(c)
         do i = -2, xst + 2
            do j = -2, yst + 2

               w1(i,j) = h(i,j)
               w2(i,j) = u(i,j)*h(i,j)
               w3(i,j) = v(i,j)*h(i,j)

               c = dsqrt(g*h(i,j))

               lf1(i,j) = u(i,j) - c
               lf2(i,j) = u(i,j)
               lf3(i,j) = u(i,j) + c

               ef12(i,j) = u(i,j) - c
               ef13(i,j) = v(i,j)
               ef32(i,j) = u(i,j) + c
               ef33(i,j) = v(i,j)

               lg1(i,j) = v(i,j) - c
               lg2(i,j) = v(i,j)
               lg3(i,j) = v(i,j) + c

               eg12(i,j) = u(i,j)
               eg13(i,j) = v(i,j) - c
               eg32(i,j) = u(i,j)
               eg33(i,j) = v(i,j) + c

               bf1(i,j) = 0.5d0*c*(B(i+1,j) - B(i,j))
               bf2(i,j) = 0.0d0
               bf3(i,j) = -0.5d0*c*(B(i+1,j) - B(i,j))

               bg1(i,j) = 0.5d0*c*(B(i,j+1) - B(i,j))
               bg2(i,j) = 0.0d0
               bg3(i,j) = -0.5d0*c*(B(i,j+1) - B(i,j))

               if (c .eq. 0.0d0) then
                  af1(i,j) = 0.0d0
                  af2(i,j) = 0.0d0
                  af3(i,j) = 0.0d0
               else
                  af1(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) + (0.5d0/c)*
```

```
     +             (u(i,j)*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
                   af2(i,j) = (1.0d0/c)*((v(i+1,j)*h(i+1,j) - v(i,j)*
     +             h(i,j)) - v(i,j)*(h(i+1,j) - h(i,j)))
                   af3(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) - (0.5d0/c)*
     +             (u(i,j)*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
               end if

               if (c .eq. 0.0d0) then
                  ag1(i,j) = 0.0d0
                  ag2(i,j) = 0.0d0
                  ag3(i,j) = 0.0d0
               else
                  ag1(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) + (0.5d0/c)*
     +             (v(i,j)*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
                  ag2(i,j) = (1.0d0/c)*((u(i,j+1)*h(i,j+1) - u(i,j)*
     +             h(i,j)) - u(i,j)*(h(i,j+1) - h(i,j)))
                  ag3(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) - (0.5d0/c)*
     +             (v(i,j)*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
               end if

               R1(i,j) = 0.0d0
               R2(i,j) =-(g/(2.0d0*dx))*h(i,j)*(B(i+1,j)-B(i-1,j))
               R3(i,j) =-(g/(2.0d0*dy))*h(i,j)*(B(i,j+1)-B(i,j-1))

               nf1(i,j) = (dt/dx)*lf1(i,j)
               nf2(i,j) = (dt/dx)*lf2(i,j)
               nf3(i,j) = (dt/dx)*lf3(i,j)

               ng1(i,j) = (dt/dy)*lg1(i,j)
               ng2(i,j) = (dt/dy)*lg2(i,j)
               ng3(i,j) = (dt/dy)*lg3(i,j)

            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED), PRIVATE(slim)
         do i = -2, xst + 2
            do j = -2 , yst + 2

               Df1(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +          dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +          ef11 + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +          (af2(i,j) - Qx(n, af2, i, j, slim))* ef21 +
     +          dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +          Qx(n, af3, i, j, slim))* ef31)
```

```
               Df2(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +          dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +          ef12(i,j) + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +          (af2(i,j) - Qx(n, af2, i, j, slim))*ef22 +
     +          dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +          Qx(n, af3, i, j, slim))*ef32(i,j))

               Df3(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +          dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +          ef13(i,j) + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +          (af2(i,j) - Qx(n, af2, i, j, slim))*ef23 +
     +          dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +          Qx(n, af3, i, j, slim))*ef33(i,j))

               Dg1(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +          dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +          eg11 + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +          (ag2(i,j) - Qy(n, ag2, i, j, slim))* eg21 +
     +          dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +          Qy(n, ag3, i, j, slim))* eg31)

               Dg2(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +          dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +          eg12(i,j) + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +          (ag2(i,j) - Qy(n, ag2, i, j, slim))*eg22 +
     +          dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +          Qy(n, ag3, i, j, slim))*eg32(i,j))

               Dg3(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +          dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +          eg13(i,j) + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +          (ag2(i,j) - Qy(n, ag2, i, j, slim))*eg23 +
     +          dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +          Qy(n, ag3, i, j, slim))*eg33(i,j))

            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED)
         do i = -2, xst + 2
            do j = -2 , yst + 2
               F1(i,j) = u(i,j)*h(i,j)
               F2(i,j) = h(i,j)*(u(i,j)**2) + 0.5d0*g*(h(i,j)**2)
               F3(i,j) = u(i,j)*v(i,j)*h(i,j)
               G1(i,j) = v(i,j)*h(i,j)
               G2(i,j) = u(i,j)*v(i,j)*h(i,j)
               G3(i,j) = h(i,j)*(v(i,j)**2) + 0.5d0*g*(h(i,j)**2)
            end do
         end do
C$OMP END PARALLEL DO
```

```
C$OMP PARALLEL DO DEFAULT(SHARED)
         do i = -2, xst + 2
            do j = -2, yst + 2
               wt1(i,j) = w1(i,j) - (dt/dx)*(F1(i,j) - F1(i-1,j)) -
     &          (dt/dy)*(G1(i,j) - G1(i,j-1)) + dt*R1(i,j)

               wt2(i,j) = w2(i,j) - (dt/dx)*(F2(i,j) - F2(i-1,j)) -
     &          (dt/dy)*(G2(i,j) - G2(i,j-1)) + dt*R2(i,j)

               wt3(i,j) = w3(i,j) - (dt/dx)*(F3(i,j) - F3(i-1,j)) -
     &          (dt/dy)*(G3(i,j) - G3(i,j-1)) + dt*R3(i,j)

               if (wt1(i,j) .le. 0.0d0) then
                  print*, '----', wt1(i,j), i, j
                  wt1(i,j) = eps
               end if

               ht(i,j) = wt1(i,j)
               ut(i,j) = wt2(i,j)/wt1(i,j)
               vt(i,j) = wt3(i,j)/wt1(i,j)
            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED)
         do i = -2, xst + 2
            do j = -2, yst + 2

               Ft1(i,j) = ut(i,j)*ht(i,j)
               Ft2(i,j) = ht(i,j)*(ut(i,j)**2) + 0.5d0*g*(ht(i,j)**2)
               Ft3(i,j) = ut(i,j)*vt(i,j)*ht(i,j)
               Gt1(i,j) = vt(i,j)*ht(i,j)
               Gt2(i,j) = ut(i,j)*vt(i,j)*ht(i,j)
               Gt3(i,j) = ht(i,j)*(vt(i,j)**2) + 0.5d0*g*(ht(i,j)**2)
               Rt1(i,j) = 0.0d0
               Rt2(i,j)=-(g/(2.0d0*dx))*ht(i,j)*(B(i+1,j)-B(i-1,j))
               Rt3(i,j)=-(g/(2.0d0*dy))*ht(i,j)*(B(i,j+1)-B(i,j-1))


            end do
         end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED)
         do i = 0, xst
            do j = 0, yst
               wtt1(i,j) = 0.5d0*(w1(i,j) + wt1(i,j)-(dt/dx)*(Ft1(i+1,j)
     &           - Ft1(i,j)) - (dt/dy)*(Gt1(i,j+1) - Gt1(i,j)) +
     &           dt*Rt1(i,j))
               wtt2(i,j) = 0.5d0*(w2(i,j) + wt2(i,j)-(dt/dx)*(Ft2(i+1,j)
```

```
     &            - Ft2(i,j)) - (dt/dy)*(Gt2(i,j+1) - Gt2(i,j)) +
     &            dt*Rt2(i,j))
                wtt3(i,j) = 0.5d0*(w3(i,j) + wt3(i,j)-(dt/dx)*(Ft3(i+1,j)
     &            - Ft3(i,j)) - (dt/dy)*(Gt3(i,j+1) - Gt3(i,j)) +
     &            dt*Rt3(i,j))

                if (wtt1(i,j) .le. 0.0d0) then
                   print*, '***', w1(i,j), i, j
                   wtt1(i,j) = eps
                end if
             end do
          end do
C$OMP END PARALLEL DO

C$OMP PARALLEL DO DEFAULT(SHARED)
          do i = 0, xst
             do j = 0, yst
                w1(i,j) = wtt1(i,j) + Df1(i,j) - Df1(i-1,j) + Dg1(i,j)
     +           - Dg1(i,j-1)
                w2(i,j) = wtt2(i,j) + Df2(i,j) - Df2(i-1,j) + Dg2(i,j)
     +           - Dg2(i,j-1)
                w3(i,j) = wtt3(i,j) + Df3(i,j) - Df3(i-1,j) + Dg3(i,j)
     +           - Dg3(i,j-1)

             if (wt1(i,j) .le. 0.0d0) then
                   print*, '----', wt1(i,j), i, j
                   wt1(i,j) = eps
                end if

                h(i,j) = w1(i,j)
                u(i,j) = w2(i,j)/w1(i,j)
                v(i,j) = w3(i,j)/w1(i,j)
             end do
          end do
C$OMP END PARALLEL DO

lfmax = 0.0d0
lgmax = 0.0d0

C$OMP DO
          do i = 0, xst
             do j = 0, yst
                lfmax = dmax1(lfmax, dabs(lf1(i,j)), dabs(lf2(i,j)),
     +            dabs(lf3(i,j)))
                lgmax = dmax1(lgmax, dabs(lg1(i,j)), dabs(lg2(i,j)),
     +            dabs(lg3(i,j)))
              end do
           end do
C$OMP END DO

          dt = (CFL*dmin1(dx,dy))/(dmax1(lfmax, lgmax))
```

```
         time = time + dt
         if (time .gt. tfinal) then
            timet = time - dt
            dt = tfinal - timet
            time = timet + dt
            FLAG = FLAG + 1
         end if

      end do
C$OMP PARALLEL DEFAULT(SHARED)
      call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)
C$OMP END PARALLEL
tm = dtime(ta)
print*, 'Running Time = ', tm, 'seconds'

      open(10, file = 'xind.dat')
      open(11, file = 'yind.dat')
      open(12, file = 'hind.dat')
      open(13, file = 'uind.dat')
      open(14, file = 'vind.dat')
      open(15, file = 'bind.dat')
      open(16, file = 'st.dat')

      write (16,*) xst
      write (16,*) yst
      do i = 0, xst
         xind(i) = ax + dble(i)*dx
         write (10,*) xind(i)
      end do

      do j = 0, yst
         yind(j) = ay + dble(j)*dy
         write (11,*) yind(j)
      end do

      do j = 0, yst
         do i = 0, xst
            write (12,*) h(i,j)
            write (13,*) u(i,j)
            write (14,*) v(i,j)
            write (15,*) B(i,j)
         end do
      end do

      close (10, status = 'keep')
      close (11, status = 'keep')
      close (12, status = 'keep')
      close (13, status = 'keep')
      close (14, status = 'keep')
      close (15, status = 'keep')
```

```
      close (16, status = 'keep')

      stop
      end
```

Α΄.3 Πηγαίος κώδικας του Roe MPI

```
      program Roe2D_MPI

      implicit none
      include 'mpif.h'

integer myrank, p, source, dest, tag, ierr
integer status(MPI_STATUS_SIZE), left, right
integer ibegin, iend, jbegin, jend, ii, jj, np
parameter (np = 8)

      integer xst, yst, ictype, bctype, FLAG, k, flim, i, j
      integer I1, I2, I3, J1, J2, J3, n, m
      parameter (n = 310)
      parameter (m = 310)

double precision buff1(-5:n+5), buff2(-5:n+5), buff3(-5:n+5)

      real dtime, tm, ta(2)
      double precision ax, bx, ay, by, tfinal, CFL, Lx, Ly, pi, g, dx
      double precision dy, dt
      double precision u(-5:n+5,-5:m+5), h(-5:n+5,-5:m+5)
      double precision v(-5:n+5,-5:m+5)
      double precision uav, vav, havf, havg, cf, cg
      double precision lf1(-5:n+5,-5:m+5), lf2(-5:n+5,-5:m+5)
      double precision lf3(-5:n+5,-5:m+5), lg1(-5:n+5,-5:m+5)
      double precision lg2(-5:n+5,-5:m+5), lg3(-5:n+5,-5:m+5)
      double precision ef11, ef12(-5:n+5,-5:m+5)
      double precision ef13(-5:n+5,-5:m+5), ef21
      double precision ef22, ef23
      double precision eg11, eg12(-5:n+5,-5:m+5)
      double precision eg13(-5:n+5,-5:m+5), eg21
      double precision eg22, eg23
      double precision eg31, eg32(-5:n+5,-5:m+5)
      double precision eg33(-5:n+5,-5:m+5), ef31
      double precision ef32(-5:n+5,-5:m+5), ef33(-5:n+5,-5:m+5)
      double precision af1(-5:n+5,-5:m+5), af2(-5:n+5,-5:m+5)
      double precision af3(-5:n+5,-5:m+5), ag1(-5:n+5,-5:m+5)
      double precision ag2(-5:n+5,-5:m+5), ag3(-5:n+5,-5:m+5)
      double precision nf1(-5:n+5,-5:m+5), nf2(-5:n+5,-5:m+5)
      double precision nf3(-5:n+5,-5:m+5), ng1(-5:n+5,-5:m+5)
      double precision ng2(-5:n+5,-5:m+5), ng3(-5:n+5,-5:m+5)
      double precision thetaf1, thetaf2, thetaf3,thetag1,thetag2,thetag3
      double precision phif1(-5:n+5,-5:m+5), phif2(-5:n+5,-5:m+5)
```

```
      double precision phif3(-5:n+5,-5:m+5), phig1(-5:n+5,-5:m+5)
      double precision phig2(-5:n+5,-5:m+5), phig3(-5:n+5,-5:m+5)
      double precision bf1(-5:n+5,-5:m+5), bf2
      double precision bf3(-5:n+5,-5:m+5), bg1(-5:n+5,-5:m+5)
      double precision bg2, bg3(-5:n+5,-5:m+5)
      double precision fp1(-5:n+5,-5:m+5), fp2(-5:n+5,-5:m+5)
      double precision fp3(-5:n+5,-5:m+5), fm1(-5:n+5,-5:m+5)
      double precision fm2(-5:n+5,-5:m+5), fm3(-5:n+5,-5:m+5)
      double precision gp1(-5:n+5,-5:m+5), gp2(-5:n+5,-5:m+5)
      double precision gp3(-5:n+5,-5:m+5), gm1(-5:n+5,-5:m+5)
      double precision gm2(-5:n+5,-5:m+5), gm3(-5:n+5,-5:m+5)
      double precision fst1(-5:n+5,-5:m+5), fst2(-5:n+5,-5:m+5)
      double precision fst3(-5:n+5,-5:m+5), gst1(-5:n+5,-5:m+5)
      double precision gst2(-5:n+5,-5:m+5), gst3(-5:n+5,-5:m+5)
      double precision F1(-5:n+5,-5:m+5), F2(-5:n+5,-5:m+5)
      double precision F3(-5:n+5,-5:m+5), G1(-5:n+5,-5:m+5)
      double precision G2(-5:n+5,-5:m+5), G3(-5:n+5,-5:m+5)
      double precision w1(-5:n+5,-5:m+5), w2(-5:n+5,-5:m+5)
      double precision w3(-5:n+5,-5:m+5), B(-5:n+5,-5:m+5)
      double precision xind(n), yind(n), yytype, yy, tim1, tim2, timmpi
      double precision time, timet, lim, eps, x, y, delta
      double precision lfmax(0:np-1), lgmax(0:np-1), lmax
      parameter (eps = 1e-8)

call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, p, ierr)

if (myrank .eq. 0) then
       open (20, file = 'input.dat')
       read (20,*) ax, bx, ay, by, xst, yst, tfinal, CFL, ictype,
     +    bctype, flim, g, yytype, delta
       close(20, status = 'keep')
      end if
      timmpi = 0.d0
      tim1 = MPI_WTIME()

      call MPI_BCAST(ax, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(bx, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(ay, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(by, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(xst, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(yst, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(tfinal, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(CFL, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
```

```
       call MPI_BCAST(ictype, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
       call MPI_BCAST(bctype, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
       call MPI_BCAST(flim, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
       call MPI_BCAST(g, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
      + ierr)
       call MPI_BCAST(yytype, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
       call MPI_BCAST(delta, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
      +  ierr)

       tim2 = MPI_WTIME()
       timmpi = tim2 - time
       Lx = bx - ax
       Ly = by - ay
       dx = Lx/dble(xst)
       dy = Ly/dble(yst)
       pi = 4.0d0*datan(1.0d0)

       call ics(n, ax, dx, ay, dy, xst, yst, ictype, h, u, v, B)

       dt = 0.0d0
       time = 0.0d0
       FLAG = 0
       k = 0

if (myrank .eq. 0) then
       tm = dtime(ta)
end if

       ef11 = 1.0d0
       ef21 = 0.0d0
       ef22 = 0.0d0
       ef23 = 1.0d0
       ef31 = 1.0d0
       eg11 = 1.0d0
       eg21 = 0.0d0
       eg22 = 1.0d0
       eg23 = 0.0d0
       eg31 = 1.0d0

       bf2 = 0.0d0
       bg2 = 0.0d0

       do while ((time .le. tfinal) .and. (FLAG .ne. 2))
          k = k + 1

          call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -5
if (myrank .eq. p-1) iend = xst + 5
```

```
      tim1 = MPI_WTIME()

         if (mod(myrank,2) .eq. 1) then
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = h(ibegin,j)
buff2(j) = u(ibegin,j)
buff3(j) = v(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. p-1) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
               h(iend+1,j) = buff1(j)
               u(iend+1,j) = buff2(j)
               v(iend+1,j) = buff3(j)
end do
end if
else
if (myrank .ne. p-1) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
               h(iend+1,j) = buff1(j)
               u(iend+1,j) = buff2(j)
               v(iend+1,j) = buff3(j)
end do
   end if
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = h(ibegin,j)
buff2(j) = u(ibegin,j)
buff3(j) = v(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
```

```
     +   myrank-1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, ierr)
end if
end if

      tim2 = MPI_WTIME()
      timmpi = timmpi + tim2 - tim1

         do i = ibegin, iend
            do j = -5, yst + 5

               w1(i,j) = h(i,j)
               w2(i,j) = u(i,j)*h(i,j)
               w3(i,j) = v(i,j)*h(i,j)

               if (dsqrt(h(i+1,j)) + dsqrt(h(i,j)) .eq. 0.0d0)then
               uav = 0.0d0
               else
               uav = (dsqrt(h(i+1,j))*u(i+1,j) + dsqrt(h(i,j))*
     +           u(i,j))/(dsqrt(h(i+1,j)) + dsqrt(h(i,j)))
               end if
               if (dsqrt(h(i,j+1)) + dsqrt(h(i,j)) .eq. 0.0d0)then
               vav = 0.0d0
               else
               vav = (dsqrt(h(i,j+1))*v(i,j+1) + dsqrt(h(i,j))*
     +           v(i,j))/(dsqrt(h(i,j+1)) + dsqrt(h(i,j)))
               end if

               havf = 0.5d0*(h(i + 1,j) + h(i,j))
               havg = 0.5d0*(h(i,j + 1) + h(i,j))

               cf = dsqrt(g*havf)
               cg = dsqrt(g*havg)

               lf1(i,j) = uav - cf
               lf2(i,j) = uav
               lf3(i,j) = uav + cf

               ef12(i,j) = uav - cf
               ef13(i,j) = vav
               ef32(i,j) = uav + cf
               ef33(i,j) = vav

               lg1(i,j) = vav - cg
               lg2(i,j) = vav
               lg3(i,j) = vav + cg

               eg12(i,j) = uav
```

```
               eg13(i,j) = vav - cg
               eg32(i,j) = uav
               eg33(i,j) = vav + cg

               bf1(i,j) = 0.5d0*cf*(B(i+1,j) - B(i,j))
               bf3(i,j) = -0.5d0*cf*(B(i+1,j) - B(i,j))

               bg1(i,j) = 0.5d0*cg*(B(i,j+1) - B(i,j))
               bg3(i,j) = -0.5d0*cg*(B(i,j+1) - B(i,j))

               if (cf .eq. 0.0d0) then
                  af1(i,j) = 0.0d0
                  af2(i,j) = 0.0d0
                  af3(i,j) = 0.0d0
               else
                  af1(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) + (0.5d0/(cf))*
     +             (uav*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
                  af2(i,j) = (1.0d0/cf)*((v(i+1,j)*h(i+1,j) - v(i,j)*
     +             h(i,j)) - (vav*(h(i+1,j) - h(i,j))))
                  af3(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) - (0.5d0/(cf))*
     +             (uav*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +             u(i,j)*h(i,j)))
               end if

               if (cg .eq. 0.0d0) then
                  ag1(i,j) = 0.0d0
                  ag2(i,j) = 0.0d0
                  ag3(i,j) = 0.0d0
               else
                  ag1(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) + (0.5d0/(cg))*
     +             (vav*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
                  ag2(i,j) = (1.0d0/cg)*((u(i,j+1)*h(i,j+1) - u(i,j)*
     +             h(i,j)) - (uav*(h(i,j+1) - h(i,j))))
                  ag3(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) - (0.5d0/(cg))*
     +             (vav*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +             v(i,j)*h(i,j)))
               end if
            end do
         end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -5
if (myrank .eq. p-1) iend = xst + 5
         do i = ibegin, iend
            do j = -5, yst + 5
               nf1(i,j) = (dt/dx)*lf1(i,j)
               nf2(i,j) = (dt/dx)*lf2(i,j)
               nf3(i,j) = (dt/dx)*lf3(i,j)
```

```
ng1(i,j) = (dt/dy)*lg1(i,j)
ng2(i,j) = (dt/dy)*lg2(i,j)
ng3(i,j) = (dt/dy)*lg3(i,j)

if (nf1(i,j) .eq. 0.0d0) then
   I1 = i
else
   I1 = i - nf1(i,j)/dabs(nf1(i,j))
end if
if (nf2(i,j) .eq. 0.0d0) then
   I2 = i
else
   I2 = i - nf2(i,j)/dabs(nf2(i,j))
end if
if (nf3(i,j) .eq. 0.0d0) then
   I3 = i
else
   I3 = i - nf3(i,j)/dabs(nf3(i,j))
end if
if (ng1(i,j) .eq. 0.0d0) then
   J1 = j
else
   J1 = j - ng1(i,j)/dabs(ng1(i,j))
end if
if (ng2(i,j) .eq. 0.0d0) then
   J2 = j
else
   J2 = j - ng2(i,j)/dabs(ng2(i,j))
end if
if (ng3(i,j) .eq. 0.0d0) then
   J3 = j
else
   J3 = j - ng3(i,j)/dabs(ng3(i,j))
end if

thetaf1 = (af1(I1,j))/(af1(i,j) + eps)
thetaf2 = (af2(I2,j))/(af2(i,j) + eps)
thetaf3 = (af3(I3,j))/(af3(i,j) + eps)

thetag1 = (ag1(i,J1))/(ag1(i,j) + eps)
thetag2 = (ag2(i,J2))/(ag2(i,j) + eps)
thetag3 = (ag3(i,J3))/(ag3(i,j) + eps)

phif1(i,j) = lim(thetaf1, flim)
phif2(i,j) = lim(thetaf2, flim)
phif3(i,j) = lim(thetaf3, flim)

phig1(i,j) = lim(thetag1, flim)
phig2(i,j) = lim(thetag2, flim)
phig3(i,j) = lim(thetag3, flim)
```

```
            fp1(i,j) = 0.5d0*(bf1(i,j)*ef11*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef21*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef31*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fp2(i,j) = 0.5d0*(bf1(i,j)*ef12(i,j)*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef22*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef32(i,j)*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fp3(i,j) = 0.5d0*(bf1(i,j)*ef13(i,j)*(1.0d0 + lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef23*(1.0d0 + lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef33(i,j)*(1.0d0 + lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm1(i,j) = 0.5d0*(bf1(i,j)*ef11*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef21*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef31*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm2(i,j) = 0.5d0*(bf1(i,j)*ef12(i,j)*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef22*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef32(i,j)*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            fm3(i,j) = 0.5d0*(bf1(i,j)*ef13(i,j)*(1.0d0 - lf1(i,j)/
     +        (dabs(lf1(i,j)) + eps)*(1.0d0 - phif1(i,j)*(1.0d0 -dabs
     +        (nf1(i,j)))) + bf2*ef23*(1.0d0 - lf2(i,j)/
     +        (dabs(lf2(i,j)) + eps)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs
     +        (nf2(i,j)))) + bf3(i,j)*ef33(i,j)*(1.0d0 - lf3(i,j)/
     +        (dabs(lf3(i,j)) + eps)*(1.0d0 - phif3(i,j)*(1.0d0 -dabs
     +        (nf3(i,j))))))

            gp1(i,j) = 0.5d0*(bg1(i,j)*eg11*(1.0d0 + lg1(i,j)/
     +         (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 - dabs
```

```
     +          (ng1(i,j))))) + bg2*eg21*(1.0d0 + lg2(i,j)/
     +          (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 - dabs
     +          (ng2(i,j))))) + bg3(i,j)*eg31*(1.0d0 + lg3(i,j)/
     +          (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 - dabs
     +          (ng3(i,j))))))

               gp2(i,j) = 0.5d0*(bg1(i,j)*eg12(i,j)*(1.0d0 + lg1(i,j)/
     +           (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +           (ng1(i,j))))) + bg2*eg22*(1.0d0 + lg2(i,j)/
     +           (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +           (ng2(i,j))))) + bg3(i,j)*eg32(i,j)*(1.0d0 + lg3(i,j)/
     +           (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +           (ng3(i,j))))))

               gp3(i,j) = 0.5d0*(bg1(i,j)*eg13(i,j)*(1.0d0 + lg1(i,j)/
     +           (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +           (ng1(i,j))))) + bg2*eg23*(1.0d0 + lg2(i,j)/
     +           (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +           (ng2(i,j))))) + bg3(i,j)*eg33(i,j)*(1.0d0 + lg3(i,j)/
     +           (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +           (ng3(i,j))))))

               gm1(i,j) = 0.5d0*(bg1(i,j)*eg11*(1.0d0 - lg1(i,j)/
     +          (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 - dabs
     +          (ng1(i,j))))) + bg2*eg21*(1.0d0 - lg2(i,j)/
     +          (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 - dabs
     +          (ng2(i,j))))) + bg3(i,j)*eg31*(1.0d0 - lg3(i,j)/
     +          (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 - dabs
     +          (ng3(i,j))))))

               gm2(i,j) = 0.5d0*(bg1(i,j)*eg12(i,j)*(1.0d0 - lg1(i,j)/
     +           (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +           (ng1(i,j))))) + bg2*eg22*(1.0d0 - lg2(i,j)/
     +           (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +           (ng2(i,j))))) + bg3(i,j)*eg32(i,j)*(1.0d0 - lg3(i,j)/
     +           (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +           (ng3(i,j))))))

               gm3(i,j) = 0.5d0*(bg1(i,j)*eg13(i,j)*(1.0d0 - lg1(i,j)/
     +           (dabs(lg1(i,j)) + eps)*(1.0d0 - phig1(i,j)*(1.0d0 -dabs
     +           (ng1(i,j))))) + bg2*eg23*(1.0d0 - lg2(i,j)/
     +           (dabs(lg2(i,j)) + eps)*(1.0d0 - phig2(i,j)*(1.0d0 -dabs
     +           (ng2(i,j))))) + bg3(i,j)*eg33(i,j)*(1.0d0 - lg3(i,j)/
     +           (dabs(lg3(i,j)) + eps)*(1.0d0 - phig3(i,j)*(1.0d0 -dabs
     +           (ng3(i,j))))))

            end do
         end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
```

```
if (myrank .eq. 0) ibegin = -5
if (myrank .eq. p-1) iend = xst + 5

      tim1 = MPI_WTIME()

if (mod(myrank,2) .eq. 1) then
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = fp1(iend,j)
buff2(j) = fp2(iend,j)
buff3(j) = fp3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
fp1(ibegin-1,j) = buff1(j)
                fp2(ibegin-1,j) = buff2(j)
                fp3(ibegin-1,j) = buff3(j)
end do
end if
   else
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
fp1(ibegin-1,j) = buff1(j)
                fp2(ibegin-1,j) = buff2(j)
                fp3(ibegin-1,j) = buff3(j)
end do
end if
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = fp1(iend,j)
buff2(j) = fp2(iend,j)
buff3(j) = fp3(iend,j)
```

```
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, ierr)
end if
end if

      tim2 = MPI_WTIME()
      timmpi = timmpi + tim2 - tim1

         do i = ibegin, iend
            do j = -5 , yst + 5

               fst1(i,j) = fm1(i,j) + fp1(i-1,j)
               fst2(i,j) = fm2(i,j) + fp2(i-1,j)
               fst3(i,j) = fm3(i,j) + fp3(i-1,j)
               gst1(i,j) = gm1(i,j) + gp1(i,j-1)
               gst2(i,j) = gm2(i,j) + gp2(i,j-1)
               gst3(i,j) = gm3(i,j) + gp3(i,j-1)

               F1(i,j) = 0.5d0*(u(i+1,j)*h(i+1,j) + u(i,j)*h(i,j)) -
     +          0.5d0*(af1(i,j)*yy(lf1(i,j), delta, af1(i,j), yytype)*
     +          (1.0d0 -phif1(i,j)*(1.0d0 - dabs(nf1(i,j))))*ef11 +
     +          af2(i,j)*yy(lf2(i,j), delta, af2(i,j), yytype)*(1.0d0 -
     +          phif2(i,j)*(1.0d0 -dabs(nf2(i,j))))*ef21 +af3(i,j)*
     +          yy(lf3(i,j), delta, af3(i,j), yytype)*(1.0d0 -
     +          phif3(i,j)*(1.0d0 - dabs(nf3(i,j))))*ef31)

               F2(i,j) = 0.5d0*(h(i+1,j)*u(i+1,j)**2 + 0.5d0*g*
     +          h(i+1,j)**2 + h(i,j)*u(i,j)**2 + 0.5d0*g*h(i,j)**2) -
     +          0.5d0*(af1(i,j)*yy(lf1(i,j), delta, af1(i,j), yytype)*
     +          (1.0d0 - phif1(i,j)*(1.0d0 - dabs(nf1(i,j))))*
     +          ef12(i,j) + af2(i,j)*yy(lf2(i,j), delta, af2(i,j),
     +          yytype)*(1.0d0 - phif2(i,j)*(1.0d0 -dabs(nf2(i,j))))*
     +          ef22 + af3(i,j)*yy(lf3(i,j), delta, af3(i,j),
     +          yytype)*(1.0d0 - phif3(i,j)*(1.0d0 - dabs(nf3(i,j))))*
     +          ef32(i,j))

               F3(i,j) = 0.5d0*(u(i+1,j)*v(i+1,j)*h(i+1,j) +u(i,j)*
     +           v(i,j)*h(i,j)) - 0.5d0*(af1(i,j)*yy(lf1(i,j), delta,
     +           af1(i,j), yytype)*(1.0d0 - phif1(i,j)*(1.0d0 -
     +           dabs(nf1(i,j))))*ef13(i,j) + af2(i,j)*yy(lf1(i,j),
     +           delta, af1(i,j), yytype)*(1.0d0-phif2(i,j)*(1.0d0 -
     +           dabs(nf2(i,j))))*ef23 + af3(i,j)*yy(lf1(i,j),
     +           delta, af1(i,j), yytype)*(1.0d0-phif3(i,j)*(1.0d0-
     +           dabs(nf3(i,j))))*ef33(i,j))

               G1(i,j) = 0.5d0*(v(i,j+1)*h(i,j+1) + v(i,j)*h(i,j)) -
```

```
     +           0.5d0*(ag1(i,j)*yy(lg1(i,j), delta, ag1(i,j), yytype)*
     +           (1.0d0- phig1(i,j)*(1.0d0 - dabs(ng1(i,j))))*eg11
     +           + ag2(i,j)*yy(lg2(i,j), delta, ag2(i,j), yytype)*(1.0d0
     +           - phig2(i,j)*(1.0d0 -dabs(ng2(i,j))))*eg21 +
     +           ag3(i,j)*yy(lg3(i,j), delta, ag3(i,j), yytype)*(1.0d0 -
     +           phig3(i,j)*(1.0d0 - dabs(ng3(i,j))))*eg31)

               G2(i,j) = 0.5d0*(u(i,j+1)*v(i,j+1)*h(i,j+1) + u(i,j)*
     +           v(i,j)*h(i,j)) - 0.5d0*(ag1(i,j)*yy(lg1(i,j), delta,
     +           ag1(i,j), yytype)*(1.0d0 - phig1(i,j)*(1.0d0 -
     +           dabs(ng1(i,j))))*eg12(i,j) + ag2(i,j)*yy(lg2(i,j),
     +           delta, ag2(i,j), yytype)*(1.0d0 - phig2(i,j)*(1.0d0 -
     +           dabs(ng2(i,j))))*eg22 + ag3(i,j)*yy(lg3(i,j),
     +           delta, ag3(i,j), yytype)*(1.0d0-phig3(i,j)*(1.0d0 -
     +           dabs(ng3(i,j))))*eg32(i,j))

               G3(i,j) = 0.5d0*(h(i,j+1)*v(i,j+1)**2 + 0.5d0*g*
     +           h(i,j+1)**2 + h(i,j)*v(i,j)**2 + 0.5d0*g*h(i,j)**2) -
     +           0.5d0*(ag1(i,j)*yy(lg1(i,j), delta, ag1(i,j), yytype)*
     +           (1.0d0 - phig1(i,j)*(1.0d0 - dabs(ng1(i,j))))*eg13(i,j)
     +           + ag2(i,j)*yy(lg2(i,j), delta, ag2(i,j), yytype)*(1.0d0
     +           - phig2(i,j)*(1.0d0 -dabs(ng2(i,j))))*eg23 +
     +           ag3(i,j)*yy(lg3(i,j), delta, ag3(i,j), yytype)*(1.0d0 -
     +           phig3(i,j)*(1.0d0 - dabs(ng3(i,j))))*eg33(i,j))
            end do
         end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = 0

      tim1 = MPI_WTIME() !!!!!!!!!!!!!!

if (mod(myrank,2) .eq. 1) then
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = F1(iend,j)
buff2(j) = F2(iend,j)
buff3(j) = F3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
```

```
     +   myrank-1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
F1(ibegin-1,j) = buff1(j)
               F2(ibegin-1,j) = buff2(j)
               F3(ibegin-1,j) = buff3(j)
end do
end if
         else
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
F1(ibegin-1,j) = buff1(j)
               F2(ibegin-1,j) = buff2(j)
               F3(ibegin-1,j) = buff3(j)
end do
end if
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = F1(iend,j)
buff2(j) = F2(iend,j)
buff3(j) = F3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, ierr)
end if
end if

      tim2 = MPI_WTIME()
      timmpi = timmpi + tim2 - tim1

         do i = ibegin, iend
            do j = 0, yst
               w1(i,j) = w1(i,j) - (dt/dx)*(F1(i,j) - F1(i-1,j)) -
     +          (dt/dy)*(G1(i,j) - G1(i,j-1)) + (dt/dx)*fst1(i,j) +
     +          (dt/dy)*gst1(i,j)

               w2(i,j) = w2(i,j) - (dt/dx)*(F2(i,j) - F2(i-1,j)) -
     +          (dt/dy)*(G2(i,j) - G2(i,j-1)) + (dt/dx)*fst2(i,j) +
     +          (dt/dy)*gst2(i,j)
```

```
               w3(i,j) = w3(i,j) - (dt/dx)*(F3(i,j) - F3(i-1,j)) -
     +          (dt/dy)*(G3(i,j) - G3(i,j-1)) + (dt/dx)*fst3(i,j) +
     +          (dt/dy)*gst3(i,j)

               if (w1(i,j) .le. 0.0d0) then
                  print*, '***', w1(i,j), i, j
                  w1(i,j) = eps
               end if

               h(i,j) = w1(i,j)
               u(i,j) = w2(i,j)/(w1(i,j))
               v(i,j) = w3(i,j)/(w1(i,j))
            end do
         end do

         lfmax(myrank) = 0.0d0
         lgmax(myrank) = 0.0d0

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = 0
if (myrank .eq. p-1) iend = xst

         do i = ibegin, iend
            do j = 0, yst
               lfmax(myrank) = dmax1(lfmax(myrank), dabs(lf1(i,j)),
     +           dabs(lf2(i,j)), dabs(lf3(i,j)))
               lgmax(myrank) = dmax1(lgmax(myrank), dabs(lg1(i,j)),
     +           dabs(lg2(i,j)), dabs(lg3(i,j)))
            end do
         end do

       tim1 = MPI_WTIME()

if (myrank .eq. 0) then
do source = 1, p-1
            call MPI_RECV(lfmax(source), 1, MPI_DOUBLE_PRECISION,
     +        source, 0, MPI_COMM_WORLD, status, ierr)
            call MPI_RECV(lgmax(source), 1, MPI_DOUBLE_PRECISION,
     +        source, 1, MPI_COMM_WORLD, status, ierr)
end do
else
call MPI_SEND(lfmax(myrank), 1, MPI_DOUBLE_PRECISION,
     +   0, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(lgmax(myrank), 1, MPI_DOUBLE_PRECISION,
     +   0, 1, MPI_COMM_WORLD, ierr)
end if

       tim2 = MPI_WTIME()
       timmpi = timmpi + tim2 - tim1
```

```
         if (myrank .eq. 0) then
            lmax = 0.0d0
            do i = 0, p-1
               lmax = dmax1(lmax, lfmax(i), lgmax(i))
            end do
         end if

         call MPI_BCAST(lmax, 1, MPI_DOUBLE_PRECISION, 0,
     +     MPI_COMM_WORLD, ierr)

         dt = (CFL*dmin1(dx,dy))/lmax
         time = time + dt
         if (time .gt. tfinal) then
            timet = time - dt
            dt = tfinal - timet
            time = timet + dt
            FLAG = FLAG + 1
         end if
       end do

       call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)

if (myrank .eq. 0) then
        tm = dtime(ta)
print*, 'STEPS = ', k ,' Running Time = ', tm, 'seconds'
print*, 'MPI time = ', timmpi
       end if

do source = 0, p-1
ibegin = (source*xst/p) + 1
iend = ((source + 1)*xst/p)
if (source .eq. 0) ibegin = 0
do ii = ibegin, iend
do jj = 0, yst
call MPI_BCAST(h(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
call MPI_BCAST(u(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
call MPI_BCAST(v(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
end do
end do
end do

if (myrank .eq. 0) then
        open(10, file = 'xind.dat')
        open(11, file = 'yind.dat')
        open(12, file = 'hind.dat')
        open(13, file = 'uind.dat')
        open(14, file = 'vind.dat')
        open(15, file = 'bind.dat')
```

```
       open(16, file = 'st.dat')

       write (16,*) xst
       write (16,*) yst
       do i = 0, xst
          xind(i) = ax + dble(i)*dx
          write (10,*) xind(i)
       end do

       do j = 0, yst
          yind(j) = ay + dble(j)*dy
          write (11,*) yind(j)
       end do

       do j = 0, yst
          do i = 0, xst
             write (12,*) h(i,j)
             write (13,*) u(i,j)
             write (14,*) v(i,j)
             write (15,*) B(i,j)
          end do
       end do

       close (10, status = 'keep')
       close (11, status = 'keep')
       close (12, status = 'keep')
       close (13, status = 'keep')
       close (14, status = 'keep')
       close (15, status = 'keep')
       close (16, status = 'keep')
      end if

call MPI_FINALIZE(ierr)

      stop
      end
```

Α΄.4 Πηγαίος κώδικας του MacCormack MPI

```
      program Mc2D_MPI

      implicit none

include 'mpif.h'

integer myrank, p, source, dest, tag, ierr
integer status(MPI_STATUS_SIZE), left, right
integer ibegin, iend, jbegin, jend, ii, jj, np
parameter (np = 8)
```

```
      integer xst, yst, ictype, bctype, FLAG, k, slim, i, j, n, m
      parameter (n = 310)
      parameter (m = 310)

double precision buff1(-5:n+5), buff2(-5:n+5), buff3(-5:n+5)
double precision buff4(-5:n+5), buff5(-5:n+5)

real dtime, tm, ta(2)
      double precision ax, bx, ay, by, tfinal, CFL, Lx, Ly, pi, g, dx
      double precision dy, dt, Qx, Qy
      double precision u(-5:n+5,-5:m+5), h(-5:n+5,-5:m+5)
      double precision v(-5:n+5,-5:m+5)
      double precision c
      double precision lf1(-5:n+5,-5:m+5), lf2(-5:n+5,-5:m+5)
      double precision lf3(-5:n+5,-5:m+5), lg1(-5:n+5,-5:m+5)
      double precision lg2(-5:n+5,-5:m+5), lg3(-5:n+5,-5:m+5)
      double precision ef11, ef12(-5:n+5,-5:m+5)
      double precision ef13(-5:n+5,-5:m+5), ef21
      double precision ef22, ef23
      double precision eg11, eg12(-5:n+5,-5:m+5)
      double precision eg13(-5:n+5,-5:m+5), eg21
      double precision eg22, eg23
      double precision eg31, eg32(-5:n+5,-5:m+5)
      double precision eg33(-5:n+5,-5:m+5), ef31
      double precision ef32(-5:n+5,-5:m+5), ef33(-5:n+5,-5:m+5)
      double precision af1(-5:n+5,-5:m+5), af2(-5:n+5,-5:m+5)
      double precision af3(-5:n+5,-5:m+5), ag1(-5:n+5,-5:m+5)
      double precision ag2(-5:n+5,-5:m+5), ag3(-5:n+5,-5:m+5)
      double precision nf1(-5:n+5,-5:m+5), nf2(-5:n+5,-5:m+5)
      double precision nf3(-5:n+5,-5:m+5), ng1(-5:n+5,-5:m+5)
      double precision ng2(-5:n+5,-5:m+5), ng3(-5:n+5,-5:m+5)
      double precision thetaf1, thetaf2, thetaf3,thetag1,thetag2,thetag3
      double precision phif1(-5:n+5,-5:m+5), phif2(-5:n+5,-5:m+5)
      double precision phif3(-5:n+5,-5:m+5), phig1(-5:n+5,-5:m+5)
      double precision phig2(-5:n+5,-5:m+5), phig3(-5:n+5,-5:m+5)
      double precision bf1(-5:n+5,-5:m+5), bf2(-5:n+5,-5:m+5)
      double precision bf3(-5:n+5,-5:m+5), bg1(-5:n+5,-5:m+5)
      double precision bg2(-5:n+5,-5:m+5), bg3(-5:n+5,-5:m+5)
      double precision F1(-5:n+5,-5:m+5), F2(-5:n+5,-5:m+5)
      double precision F3(-5:n+5,-5:m+5), G1(-5:n+5,-5:m+5)
      double precision G2(-5:n+5,-5:m+5), G3(-5:n+5,-5:m+5)
      double precision w1(-5:n+5,-5:m+5), w2(-5:n+5,-5:m+5)
      double precision w3(-5:n+5,-5:m+5), B(-5:n+5,-5:m+5)
      double precision Ft1(-5:n+5,-5:m+5), Ft2(-5:n+5,-5:m+5)
      double precision Ft3(-5:n+5,-5:m+5), Gt1(-5:n+5,-5:m+5)
      double precision Gt2(-5:n+5,-5:m+5), Gt3(-5:n+5,-5:m+5)
      double precision wt1(-5:n+5,-5:m+5), wt2(-5:n+5,-5:m+5)
      double precision wt3(-5:n+5,-5:m+5)
      double precision wtt1(-5:n+5,-5:m+5), wtt2(-5:n+5,-5:m+5)
      double precision wtt3(-5:n+5,-5:m+5)
      double precision Df1(-5:n+5,-5:m+5), Df2(-5:n+5,-5:m+5)
```

```
      double precision Df3(-5:n+5,-5:m+5)
      double precision Dg1(-5:n+5,-5:m+5), Dg2(-5:n+5,-5:m+5)
      double precision Dg3(-5:n+5,-5:m+5)
      double precision ut(-5:n+5,-5:m+5), ht(-5:n+5,-5:m+5)
      double precision vt(-5:n+5,-5:m+5)
      double precision R1(-5:n+5,-5:m+5), R2(-5:n+5,-5:m+5)
      double precision R3(-5:n+5,-5:m+5), Rt1(-5:n+5,-5:m+5)
      double precision Rt2(-5:n+5,-5:m+5), Rt3(-5:n+5,-5:m+5)
      double precision xind(n), yind(n)
      double precision time, timet, lmax, lfmax, lgmax, lim, eps, x, y
      double precision lfmax(0:np-1), lgmax(0:np-1), lmax
      double precision dBx(-5:n+5,-5:m+5), dBy(-5:n+5,-5:m+5)
      parameter (eps = 1e-8)

call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, p, ierr)

if (myrank .eq. 0) then
         open (20, file = 'input.dat')
         read (20,*) ax, bx, ay, by, xst, yst, tfinal, CFL, ictype,
     +     bctype, slim, g
         close(20, status = 'keep')
      end if

      call MPI_BCAST(ax, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(bx, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(ay, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(by, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(xst, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(yst, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(tfinal, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(CFL, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     +  ierr)
      call MPI_BCAST(ictype, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(bctype, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(slim, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
      call MPI_BCAST(g, 1, MPI_DOUBLE_PRECISION, 0, MPI_COMM_WORLD,
     + ierr)

      Lx = bx - ax
      Ly = by - ay
      dx = Lx/dble(xst)
      dy = Ly/dble(yst)
      pi = 4.0d0*datan(1.0d0)
```

```
         call ics(n, ax, dx, ay, dy, xst, yst, ictype, h, u, v, B,
     +     dBx, dBy)

      dt = 0.0d0
      time = 0.0d0
      FLAG = 0
      k = 0

if (myrank .eq. 0) then
       tm = dtime(ta)
end if

      ef11 = 1.0d0
      ef21 = 0.0d0
      ef22 = 0.0d0
      ef23 = 1.0d0
      ef31 = 1.0d0
      eg11 = 1.0d0
      eg21 = 0.0d0
      eg22 = 1.0d0
      eg23 = 0.0d0
      eg31 = 1.0d0

      do while ((time .le. tfinal) .and. (FLAG .ne. 2))
         k = k + 1

         call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -2
if (myrank .eq. p-1) iend = xst + 2

         if (mod(myrank,2) .eq. 1) then
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = h(ibegin,j)
buff2(j) = u(ibegin,j)
buff3(j) = v(ibegin,j)
buff4(j) = B(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff4(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. p-1) then
```

```
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 6, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff4(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 7, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
h(iend+1,j) = buff1(j)
               u(iend+1,j) = buff2(j)
               v(iend+1,j) = buff3(j)
               B(iend+1,j) = buff4(j)
end do
end if
else
if (myrank .ne. p-1) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff4(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
h(iend+1,j) = buff1(j)
               u(iend+1,j) = buff2(j)
               v(iend+1,j) = buff3(j)
               B(iend+1,j) = buff4(j)
end do
   end if
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = h(ibegin,j)
buff2(j) = u(ibegin,j)
buff3(j) = v(ibegin,j)
buff4(j) = B(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 6, MPI_COMM_WORLD, ierr)
      call MPI_SEND(buff4(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 7, MPI_COMM_WORLD, ierr)
end if
end if

         if (mod(myrank,2) .eq. 1) then
```

```
if (myrank .ne. p-1) then
do j = -5, yst+5
buff5(j) = B(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
B(ibegin-1,j) = buff5(j)
end do
end if
         else
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
B(ibegin-1,j) = buff5(j)
end do
end if
if (myrank .ne. p-1) then
do j = -5, yst+5
buff5(j) = B(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, ierr)
end if
end if

         do i = ibegin, iend
            do j = -2, yst + 2

               w1(i,j) = h(i,j)
               w2(i,j) = u(i,j)*h(i,j)
               w3(i,j) = v(i,j)*h(i,j)

               c = dsqrt(g*h(i,j))

               lf1(i,j) = u(i,j) - c
               lf2(i,j) = u(i,j)
               lf3(i,j) = u(i,j) + c

               ef12(i,j) = u(i,j) - c
               ef13(i,j) = v(i,j)
               ef32(i,j) = u(i,j) + c
               ef33(i,j) = v(i,j)

               lg1(i,j) = v(i,j) - c
               lg2(i,j) = v(i,j)
```

```
            lg3(i,j) = v(i,j) + c

            eg12(i,j) = u(i,j)
            eg13(i,j) = v(i,j) - c
            eg32(i,j) = u(i,j)
            eg33(i,j) = v(i,j) + c

            bf1(i,j) = 0.5d0*c*(B(i+1,j) - B(i,j))
            bf2(i,j) = 0.0d0
            bf3(i,j) = -0.5d0*c*(B(i+1,j) - B(i,j))

            bg1(i,j) = 0.5d0*c*(B(i,j+1) - B(i,j))
            bg2(i,j) = 0.0d0
            bg3(i,j) = -0.5d0*c*(B(i,j+1) - B(i,j))

            if (c .eq. 0.0d0) then
               af1(i,j) = 0.0d0
               af2(i,j) = 0.0d0
               af3(i,j) = 0.0d0
            else
               af1(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) + (0.5d0/c)*
     +          (u(i,j)*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +          u(i,j)*h(i,j)))
               af2(i,j) = (1.0d0/c)*((v(i+1,j)*h(i+1,j) - v(i,j)*
     +          h(i,j)) - v(i,j)*(h(i+1,j) - h(i,j)))
               af3(i,j) = 0.5d0*(h(i+1,j) - h(i,j)) - (0.5d0/c)*
     +          (u(i,j)*(h(i+1,j) - h(i,j)) - (u(i+1,j)*h(i+1,j) -
     +          u(i,j)*h(i,j)))
            end if

            if (c .eq. 0.0d0) then
               ag1(i,j) = 0.0d0
               ag2(i,j) = 0.0d0
               ag3(i,j) = 0.0d0
            else
               ag1(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) + (0.5d0/c)*
     +          (v(i,j)*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +          v(i,j)*h(i,j)))
               ag2(i,j) = (1.0d0/c)*((u(i,j+1)*h(i,j+1) - u(i,j)*
     +          h(i,j)) - u(i,j)*(h(i,j+1) - h(i,j)))
               ag3(i,j) = 0.5d0*(h(i,j+1) - h(i,j)) - (0.5d0/c)*
     +          (v(i,j)*(h(i,j+1) - h(i,j)) - (v(i,j+1)*h(i,j+1) -
     +          v(i,j)*h(i,j)))
            end if

            R1(i,j) = 0.0d0
            R2(i,j) =-(g/(2.0d0*dx))*h(i,j)*(B(i+1,j)-B(i-1,j))
            R3(i,j) =-(g/(2.0d0*dy))*h(i,j)*(B(i,j+1)-B(i,j-1))

            nf1(i,j) = (dt/dx)*lf1(i,j)
            nf2(i,j) = (dt/dx)*lf2(i,j)
```

```
                nf3(i,j) = (dt/dx)*lf3(i,j)

                ng1(i,j) = (dt/dy)*lg1(i,j)
                ng2(i,j) = (dt/dy)*lg2(i,j)
                ng3(i,j) = (dt/dy)*lg3(i,j)

             end do
          end do

          ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -2
if (myrank .eq. p-1) iend = xst + 2

do i = ibegin, iend
             do j = -2 , yst + 2

                Df1(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +           dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +           ef11 + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +           (af2(i,j) - Qx(n, af2, i, j, slim))* ef21 +
     +           dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +           Qx(n, af3, i, j, slim))* ef31)

                Df2(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +           dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +           ef12(i,j) + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +           (af2(i,j) - Qx(n, af2, i, j, slim))*ef22 +
     +           dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +           Qx(n, af3, i, j, slim))*ef32(i,j))

                Df3(i,j) = 0.5d0*(dabs(nf1(i,j))*(1.0d0 -
     +           dabs(nf1(i,j)))*(af1(i,j) - Qx(n, af1, i, j, slim))*
     +           ef13(i,j) + dabs(nf2(i,j))*(1.0d0 - dabs(nf2(i,j)))*
     +           (af2(i,j) - Qx(n, af2, i, j, slim))*ef23 +
     +           dabs(nf3(i,j))*(1.0d0 - dabs(nf3(i,j)))*(af3(i,j) -
     +           Qx(n, af3, i, j, slim))*ef33(i,j))

                Dg1(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +           dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +           eg11 + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +           (ag2(i,j) - Qy(n, ag2, i, j, slim))* eg21 +
     +           dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +           Qy(n, ag3, i, j, slim))* eg31)

                Dg2(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +           dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +           eg12(i,j) + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +           (ag2(i,j) - Qy(n, ag2, i, j, slim))*eg22 +
     +           dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +           Qy(n, ag3, i, j, slim))*eg32(i,j))
```

```
                Dg3(i,j) = 0.5d0*(dabs(ng1(i,j))*(1.0d0 -
     +           dabs(ng1(i,j)))*(ag1(i,j) - Qy(n, ag1, i, j, slim))*
     +           eg13(i,j) + dabs(ng2(i,j))*(1.0d0 - dabs(ng2(i,j)))*
     +           (ag2(i,j) - Qy(n, ag2, i, j, slim))*eg23 +
     +           dabs(ng3(i,j))*(1.0d0 - dabs(ng3(i,j)))*(ag3(i,j) -
     +           Qy(n, ag3, i, j, slim))*eg33(i,j))

             end do
          end do

          ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -2
if (myrank .eq. p-1) iend = xst + 2

do i = ibegin, iend
             do j = -2 , yst + 2
                F1(i,j) = u(i,j)*h(i,j)
                F2(i,j) = h(i,j)*(u(i,j)**2) + 0.5d0*g*(h(i,j)**2)
                F3(i,j) = u(i,j)*v(i,j)*h(i,j)
                G1(i,j) = v(i,j)*h(i,j)
                G2(i,j) = u(i,j)*v(i,j)*h(i,j)
                G3(i,j) = h(i,j)*(v(i,j)**2) + 0.5d0*g*(h(i,j)**2)
             end do
          end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -2
if (myrank .eq. p-1) iend = xst + 2

if (mod(myrank,2) .eq. 1) then
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = F1(iend,j)
buff2(j) = F2(iend,j)
buff3(j) = F3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, status, ierr)
```

```
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
F1(ibegin-1,j) = buff1(j)
               F2(ibegin-1,j) = buff2(j)
               F3(ibegin-1,j) = buff3(j)
end do
end if
         else
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
F1(ibegin-1,j) = buff1(j)
               F2(ibegin-1,j) = buff2(j)
               F3(ibegin-1,j) = buff3(j)
end do
end if
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = F1(iend,j)
buff2(j) = F2(iend,j)
buff3(j) = F3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, ierr)
end if
end if
         do i = ibegin, iend
            do j = -2, yst+2
               wt1(i,j) = w1(i,j) - (dt/dx)*(F1(i,j) - F1(i-1,j)) -
     &          (dt/dy)*(G1(i,j) - G1(i,j-1)) + dt*R1(i,j)

               wt2(i,j) = w2(i,j) - (dt/dx)*(F2(i,j) - F2(i-1,j)) -
     &          (dt/dy)*(G2(i,j) - G2(i,j-1)) + dt*R2(i,j)

               wt3(i,j) = w3(i,j) - (dt/dx)*(F3(i,j) - F3(i-1,j)) -
     &          (dt/dy)*(G3(i,j) - G3(i,j-1)) + dt*R3(i,j)

               if (wt1(i,j) .le. 0.0d0) then
                  print*, '----', wt1(i,j), i, j
                  wt1(i,j) = eps
               end if
```

```
               ht(i,j) = wt1(i,j)
               ut(i,j) = wt2(i,j)/wt1(i,j)
               vt(i,j) = wt3(i,j)/wt1(i,j)
            end do
         end do

         ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = -2
if (myrank .eq. p-1) iend = xst + 2

do i = ibegin, iend
            do j = -2, yst + 2

               Ft1(i,j) = ut(i,j)*ht(i,j)
               Ft2(i,j) = ht(i,j)*(ut(i,j)**2) + 0.5d0*g*(ht(i,j)**2)
               Ft3(i,j) = ut(i,j)*vt(i,j)*ht(i,j)
               Gt1(i,j) = vt(i,j)*ht(i,j)
               Gt2(i,j) = ut(i,j)*vt(i,j)*ht(i,j)
               Gt3(i,j) = ht(i,j)*(vt(i,j)**2) + 0.5d0*g*(ht(i,j)**2)
               Rt1(i,j) = 0.0d0
               Rt2(i,j)=-(g/(2.0d0*dx))*ht(i,j)*(B(i+1,j)-B(i-1,j))
               Rt3(i,j)=-(g/(2.0d0*dy))*ht(i,j)*(B(i,j+1)-B(i,j-1))
            end do
         end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = 0

         if (mod(myrank,2) .eq. 1) then
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = Ft1(ibegin,j)
buff2(j) = Ft2(ibegin,j)
buff3(j) = Ft3(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. p-1) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
```

```
     +   myrank+1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
Ft1(iend+1,j) = buff1(j)
               Ft2(iend+1,j) = buff2(j)
               Ft3(iend+1,j) = buff3(j)
end do
end if
else
if (myrank .ne. p-1) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
Ft1(iend+1,j) = buff1(j)
               Ft2(iend+1,j) = buff2(j)
               Ft3(iend+1,j) = buff3(j)
end do
   end if
if (myrank .ne. 0) then
do j = -5, yst+5
buff1(j) = Ft1(ibegin,j)
buff2(j) = Ft2(ibegin,j)
buff3(j) = Ft3(ibegin,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, ierr)
end if
end if

         do i = ibegin, iend
            do j = 0 , yst
               wtt1(i,j) = 0.5d0*(w1(i,j) + wt1(i,j)-(dt/dx)*(Ft1(i+1,j)
     &           - Ft1(i,j)) - (dt/dy)*(Gt1(i,j+1) - Gt1(i,j)) +
     &           dt*Rt1(i,j))
               wtt2(i,j) = 0.5d0*(w2(i,j) + wt2(i,j)-(dt/dx)*(Ft2(i+1,j)
     &           - Ft2(i,j)) - (dt/dy)*(Gt2(i,j+1) - Gt2(i,j)) +
     &           dt*Rt2(i,j))
               wtt3(i,j) = 0.5d0*(w3(i,j) + wt3(i,j)-(dt/dx)*(Ft3(i+1,j)
     &           - Ft3(i,j)) - (dt/dy)*(Gt3(i,j+1) - Gt3(i,j)) +
     &           dt*Rt3(i,j))

               if (wtt1(i,j) .le. 0.0d0) then
                  print*, '***', w1(i,j), i, j
                  wtt1(i,j) = eps
```

```
               end if
             end do
          end do

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = 0

if (mod(myrank,2) .eq. 1) then
if (myrank .ne. p-1) then
do j = -5, yst+5
buff1(j) = Df1(iend,j)
buff2(j) = Df2(iend,j)
buff3(j) = Df3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 1, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 2, MPI_COMM_WORLD, ierr)
end if
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 3, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 4, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 5, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
Df1(ibegin-1,j) = buff1(j)
               Df2(ibegin-1,j) = buff2(j)
               Df3(ibegin-1,j) = buff3(j)
end do
end if
         else
if (myrank .ne. 0) then
call MPI_RECV(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 0, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 1, MPI_COMM_WORLD, status, ierr)
call MPI_RECV(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank-1, 2, MPI_COMM_WORLD, status, ierr)
do j = -5, yst+5
Df1(ibegin-1,j) = buff1(j)
               Df2(ibegin-1,j) = buff2(j)
               Df3(ibegin-1,j) = buff3(j)
end do
end if
if (myrank .ne. p-1) then
do j = -5, yst+5
```

```
buff1(j) = Df1(iend,j)
buff2(j) = Df2(iend,j)
buff3(j) = Df3(iend,j)
end do
call MPI_SEND(buff1(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 3, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff2(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 4, MPI_COMM_WORLD, ierr)
call MPI_SEND(buff3(-5), yst+11, MPI_DOUBLE_PRECISION,
     +   myrank+1, 5, MPI_COMM_WORLD, ierr)
end if
end if
         do i = ibegin, iend
            do j = 0, yst
               w1(i,j) = wtt1(i,j) + Df1(i,j) - Df1(i-1,j) + Dg1(i,j)
     +          - Dg1(i,j-1)
               w2(i,j) = wtt2(i,j) + Df2(i,j) - Df2(i-1,j) + Dg2(i,j)
     +          - Dg2(i,j-1)
               w3(i,j) = wtt3(i,j) + Df3(i,j) - Df3(i-1,j) + Dg3(i,j)
     +          - Dg3(i,j-1)

            if (wt1(i,j) .le. 0.0d0) then
                  print*, '----', wt1(i,j), i, j
                  wt1(i,j) = eps
               end if

               h(i,j) = w1(i,j)
               u(i,j) = w2(i,j)/w1(i,j)
               v(i,j) = w3(i,j)/w1(i,j)
            end do
         end do

         lfmax(myrank) = 0.0d0
         lgmax(myrank) = 0.0d0

ibegin = (myrank*xst/p) + 1
iend = ((myrank + 1)*(xst)/p)
if (myrank .eq. 0) ibegin = 0
if (myrank .eq. p-1) iend = xst

         do i = ibegin, iend
            do j = 0, yst
               lfmax(myrank) = dmax1(lfmax(myrank), dabs(lf1(i,j)),
     +           dabs(lf2(i,j)), dabs(lf3(i,j)))
               lgmax(myrank) = dmax1(lgmax(myrank), dabs(lg1(i,j)),
     +           dabs(lg2(i,j)), dabs(lg3(i,j)))
            end do
         end do

if (myrank .eq. 0) then
do source = 1, p-1
```

```
            call MPI_RECV(lfmax(source), 1, MPI_DOUBLE_PRECISION,
     +        source, 0, MPI_COMM_WORLD, status, ierr)
            call MPI_RECV(lgmax(source), 1, MPI_DOUBLE_PRECISION,
     +        source, 1, MPI_COMM_WORLD, status, ierr)
end do
else
call MPI_SEND(lfmax(myrank), 1, MPI_DOUBLE_PRECISION,
     +   0, 0, MPI_COMM_WORLD, ierr)
call MPI_SEND(lgmax(myrank), 1, MPI_DOUBLE_PRECISION,
     +   0, 1, MPI_COMM_WORLD, ierr)
end if

         if (myrank .eq. 0) then
            lmax = 0.0d0
            do i = 0, p-1
               lmax = dmax1(lmax, lfmax(i), lgmax(i))
            end do
         end if

            call MPI_BCAST(lmax, 1, MPI_DOUBLE_PRECISION, 0,
     +        MPI_COMM_WORLD, ierr)

         dt = (CFL*dmin1(dx,dy))/lmax
         time = time + dt
         if (time .gt. tfinal) then
            timet = time - dt
            dt = tfinal - timet
            time = timet + dt
            FLAG = FLAG + 1
         end if
       end do

       call bcs(n, xst, yst, bctype, ax, ay, dx, dy, h, u, v)

if (myrank .eq. 0) then
        tm = dtime(ta)
print*, 'STEPS = ', k ,' Running Time = ', tm, 'seconds'
       end if

do source = 0, p-1
ibegin = (source*xst/p) + 1
iend = ((source + 1)*xst/p)
if (source .eq. 0) ibegin = 0
do ii = ibegin, iend
do jj = 0, yst
call MPI_BCAST(h(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
call MPI_BCAST(u(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
call MPI_BCAST(v(ii,jj), 1, MPI_DOUBLE_PRECISION,
     +   source, MPI_COMM_WORLD, ierr)
```

```
end do
end do
end do

if (myrank .eq. 0) then
       open(10, file = 'xind.dat')
       open(11, file = 'yind.dat')
       open(12, file = 'hind.dat')
       open(13, file = 'uind.dat')
       open(14, file = 'vind.dat')
       open(15, file = 'bind.dat')
       open(16, file = 'st.dat')

       write (16,*) xst
       write (16,*) yst
       do i = 0, xst
          xind(i) = ax + dble(i)*dx
          write (10,*) xind(i)
       end do

       do j = 0, yst
          yind(j) = ay + dble(j)*dy
          write (11,*) yind(j)
       end do

       do j = 0, yst
          do i = 0, xst
             write (12,*) h(i,j)
             write (13,*) u(i,j)
             write (14,*) v(i,j)
             write (15,*) B(i,j)
          end do
       end do

       close (10, status = 'keep')
       close (11, status = 'keep')
       close (12, status = 'keep')
       close (13, status = 'keep')
       close (14, status = 'keep')
       close (15, status = 'keep')
       close (16, status = 'keep')
      end if

call MPI_FINALIZE(ierr)

      stop
      end
```