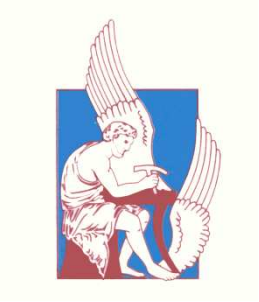
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΓΕΝΙΚΟ ΤΜΗΜΑ



ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΦΑΡΜΟΣΜΕΝΕΣ ΕΠΙΣΤΗΜΕΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΔΙΑΤΡΙΒΗ ΜΕΤΑΠΤΥΧΙΑΚΟΥ
ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ ΚΑΤΕΥΘΥΝΣΗ :
≪ΕΦΑΡΜΟΣΜΕΝΑ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΑ ΜΑΘΗΜΑΤΙΚΑ≫

ΑΡΙΘΜΗΤΙΚΗ ΕΠΙΛΥΣΗ ΝΟΜΩΝ ΔΙΑΤΗΡΗΣΗΣ ΜΕ ΧΡΗΣΗ ΜΗ-ΔΟΜΗΜΕΝΩΝ ΜΕΘΟΔΩΝ ΠΕΠΕΡΑΣΜΕΝΩΝ ΟΓΚΩΝ ΣΕ ΠΑΡΑΛΛΗΛΗ ΥΛΟΠΟΙΗΣΗ

ΠΑΝΑΓΙΩΤΗΣ Σ. ΒΑΒΙΛΗΣ

Επιβλέπων : **Επικ. Καθηγητής Ανάργυρος Δελής**

ΧΑΝΙΑ, 2012

επιτροπή

Επικ. Καθηγητής Ανάργυρος Δελής

Επικ. Καθηγητής Εμμανουήλ Μαθιουδάκης

Επικ. Καθηγητής Ιωάννης Νικολός

Πρόλογος

Το κεντρικό θέμα της εργασίας είναι η μελέτη και ανάπτυξη παράλληλου αλγορίθμου για την αριθμητική επίλυση των μαθηματικών εξισώσεων που περιγράφουν προβλήματα ροής νερού με ελεύθερη επιφάνεια κάτω από την επίδραση της βαρύτητας. Το υπολογιστικό κόστος των παραπάνω προβλημάτων είναι μεγάλο όταν πρέπει να προσομοιωθεί κάποιο φυσικό φαινόμενο, πχ κατάρρευση φράγματος, σε πραγματικό υπολογιστικό πεδίο αρκετών τετραγωνικών χιλιομέτρων. Αφού κατανοηθεί η αριθμητική μέθοδος και ο τρόπος που χρησιμοποιείται το υπολογιστικό πλέγμα αναλύεται ο κώδικας και σχεδιάζεται η παράλληλη μέθοδος. Δίνεται ιδιαίτερη σημασία στον κατακερματισμό του υπολογιστικού πλέγματος καθώς επίσης και στον τρόπο επικοινωνίας των επεξεργαστών ώστε να έχουμε το καλύτερο δυνατό αποτέλεσμα. Ο κώδικας δοκιμάζεται και συγκρίνεται με τον σειριακό με σκοπό την επαλήθευση των αποτελεσμάτων για διάφορα προβλήματα και υπολογιστικά πλέγματα. Τέλος μετράμε την επιτάχυνση και απόδοση εξάγοντας συμπεράσματα για την συμπεριφορά του κώδικα καθώς και της δυνατότητας κλιμάκωσης που έχει.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την τριμελή επιτροπή μου και ιδιαίτερα τον κ. Δελή για την άμεση ανταπόκριση και βοήθεια, την Μαρία Καζολέα για την πολύτιμη βοήθεια στην κατανόηση του σειριακού κώδικα και στην διόρθωση προβλημάτων του παράλληλου και τον Σωτήρη Σαρακήνο για τις χρήσιμες πληροφορίες που μου έδωσε σχετικά με την δική του παράλληλη υλοποίηση σε αντίστοιχο κώδικα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Οι ροές νερού με ελεύθερη επιφάνεια κάτω από την επίδραση της βαρύτητας αποτελούν μια μεγάλη κλάση προβλημάτων επιστημονικού και πρακτικού ενδιαφέροντος. Φυσικά φαινόμενα όπως πλημμύρες, κατάρρευση φραγμάτων, διάδοση κυμάτων σε ποταμούς και παράκτιες περιοχές απασχολούν εδώ και αρκετά χρόνια την επιστημονική κοινότητα λόγο του μεγάλου πρακτικού ενδιαφέροντος που παρουσιάζουν.Για φαινόμενα ροής ρευστών τα μαθηματικά μοντέλα αποτελούνται από συστήματα μερικών διαφορικών εξισώσεων με αρχικές και συνοριακές συνθήκες. Το επόμενο βήμα είναι η κατασκευή και επιλογή κατάλληλων αριθμητικών μεθόδων για την επίλυση αυτών.

Μέχρι στιγμής για την προσομοίωση φαινομένων, όπως αναρρίχηση κύματος στην ακτή, πλημμύρα ακτών και θραύση κυματισμών, έχουν αναπτυχθεί και χρησιμοποιούνται ευρέως διάφοροι επιχειρησιακοί κώδικες. Μερικοί από αυτούς είναι: HEC-RAS [6] ο οποίος βασίζεται στις εξισώσεις St. Venant μίας διάστασης, TELEMAC-2D [5], [4] ο οποίος βασίζεται στις εξισώσεις ρηχών υδάτων δύο διαστάσεων και οι οποίες επιλύονται με πεπερασμένα στοιχεία και LISFLOOD-FP [4] όπου οι εξισώσεις ρηχών υδάτων δύο διαστάσεων επιλύονται με πεπερασμένες διαφορές. Σε ακαδημαϊκό επίπεδο οι αριθμητικές μέθοδοι που έχουν χρησιμοποιηθεί είναι μέθοδοι πεπερασμένων διαφορών, πεπερασμένων στοιχείων [56] και τα τελευταία χρόνια μέθοδοι πεπερασμένων όγκων.

Το χαρακτηριστικό της αριθμητικής μεθόδου είναι η δυνατότητα εύρεσης ικανοποιητικής ακρίβειας λύσης με σχετικά μικρούς μεγέθους υπολογιστικό

πλέγμα. Τα πράγματα όμως είναι διαφορετικά όταν πρέπει να προσομοιωθεί ένα πραγματικό φυσικό φαινόμενο όπου το υπολογιστικό πλέγμα αντιστοιχεί σε περιοχή αρκετών KM^2. Σε αυτή την περίπτωση το υπολογιστικό πλέγμα μπορεί να αποτελείται από εκατοντάδες χιλιάδες ή ακόμα και μερικά εκατομμύρια στοιχεία. Το υπολογιστικό κόστος καθώς και το μέγεθος των δεδομένων είναι πλέον μεγάλο και υπολογίσιμο ακόμα και για τους ποιο σύγχρονους επεξεργαστές.

Η παραλληλοποίηση των αριθμητικών μεθόδων είναι πλέον μονόδρομος χάρις των δυνατοτήτων που προσφέρουν οι σύγχρονες υπολογιστικές μονάδες. Σήμερα υπάρχουν δύο θεμελιώδης παράλληλες αρχιτεκτονικές. (1) τα συστήματα διαμοιραζόμενης μνήμης (shared memory systems) όπως οι διάφοροι σταθμοί εργασίας με τους πολύ πύρηνους επεξεργαστές πάνω στην ίδια μητρική κάρτα ή συστήματα με τους πολύ πλυθείς επεξεργαστές παραγωγής γραφικών GPUs κατάλληλα ρυθμισμένους για παράλληλους υπολογισμούς. Και (2) τα συστήματα κατανεμημένης μνήμης distributed memory systems οπού αρκετοί αυτόνομοι σταθμοί εργασίας είναι συνδεδεμένοι μεταξύ τους μέσω ενός γρήγορου δικτύου που τους επιτρέπει την γρήγορη ανταλλαγή μηνυμάτων μεταξύ τους.

Τα τελευταία χρόνια υπάρχει ιδιαίτερο ενδιαφέρον για την 1η αρχιτεκτονική παραλληλοποίησης, ο λόγος είναι ότι πλέον οι επεξεργαστές είναι πολύ πήρηνοι και ο καθένας μπορεί να έχει ακόμα και στο φορητό του υπολογιστή την δυνατότητα των τεσσάρων ή και παραπάνω πυρήνων. Ο προγραμματισμός σε αυτά τα συστήματα γίνεται με την χρήση της βιβλιοθήκης OpenMP όπου είναι συμβατή με τις περισσότερες γλώσσες προγραμματισμού και αρκετά κατανοητή στους προγραμματιστές. Πολύ τελευταία είναι ιδιαίτερα διαδεδομένος ο προγραμματισμός των επεξεργαστών για γραφικά όπου μπορεί να προσφέρει υπό συνθήκες ιδιαίτερα μεγάλες επιδόσεις. Το πρόβλημα σε αυτή την περίπτωση είναι οι αρκετοί περιορισμοί στις γλώσσες προγραμματισμού και στις ιδιαίτερες προγραμματιστικές γνώσεις για να κάνεις έναν κώδικα συμβατό με αυτό το περιβάλλον. Γενικά σε αυτή την αρχιτεκτονική παραλληλοποίησης το βασικό πρόβλημα είναι το μέγεθος της μνήμης η οποία είναι η ίδια για όλους τους επεξεργαστές.

Αντίθετα στην 2η αρχιτεκτονική ο περιορισμός είναι στο πλήθος των επε-

ξεργαστών και στην επικοινωνία μεταξύ τους και όχι την μνήμη αφού ο κάθε κόμβος διαθέτει την δική του. Η επικοινωνία μεταξύ των κόμβων γίνεται με την χρήση της βιβλιοθήκης του MPI όπου το προγραμματιστικό βάρος πέφτει στην μείωση της επιμέρους επικοινωνίας μεταξύ των επεξεργαστών. Σε αυτή την αρχιτεκτονική ο κάθε επεξεργαστής τρέχει μια δική του διεργασία όπου ανάλογα με τον εκάστοτε αλγόριθμο έχει οριστεί μια τοπολογία πάνω στην οποία και γίνεται η επικοινωνία μεταξύ των επεξεργαστών. Κατά την διάρκεια της επικοινωνίας οι επεξεργαστές βρίσκονται σε κατάσταση αναμονής μέχρι την ολοκλήρωσή της.

Η απόδοση ενός παράλληλου κώδικα μετριέται με την επιτάχυνση speedup S και την απόδοση efficiency E, όπου εξαρτάτε από το μέγεθος του προβλήματος και το πλήθος των επεξεργαστών n_p. Στην περίπτωσή μας το μέγεθος του προβλήματος ορίζεται από το πλήθος των υπολογιστικών κελιών του πλέγματος n_c, οπότε η επιτάχυνση και η απόδοση ορίζονται ως,

$$S(n_c, n_p) = \frac{T(n_c, 1)}{T(n_c, n_p)}, \quad E(n_c, n_p) = \frac{T(n_c, 1)}{n_p T(n_c, n_p)} \tag{1.1}$$

όπου $T(n_c, n_p)$ είναι ο χρόνος εκτέλεσης του προγράμματος και $n_p = 1$ ορίζει την διαδοχική υλοποίηση. Ο νόμος τους Amdahl ορίζει την μέγιστη επιτάχυνση S_{max} και την μέγιστη απόδοση E_{max} ως,

$$S_{max}(n_p) = \frac{1}{1 - \theta_p + \theta_p / n_p}, \quad E_{max}(n_p) = \frac{1}{n_p(1 - \theta_p) + \theta_p} \tag{1.2}$$

όπου θ_p είναι το κομμάτι του κώδικα που μπορεί να εκτελεστεί παράλληλα. Το μέγιστο $\theta_p = 1$ με $S_{max} = n_p$ και $E_{max} = 1$. Γενικά όταν το $\theta_p < 1$ η απόδοση του παράλληλου κώδικα μειώνεται αυξάνοντας των αριθμό των επεξεργαστών n_p.

Σε αυτή την εργασία η παραλληλοποίηση του κώδικα έγινε βάση της κατανεμημένης μνήμης αρχιτεκτονικής χρησιμοποιώντας την βιβλιοθήκη του MPI. Μετά από σχετική μελέτη των αλγορίθμων του κώδικα αποφασίστηκε η παραλληλοποίηση να γίνει με την λογική του SPMD, Single Program Multiple Data. Αφού το κύριο πρόβλημα είναι ο όγκος των δεδομένων και παραλληλοποίηση των επιμέρους ρουτινών θα είχε τεράστιο υπολογιστικό αλλά και προγραμματιστικό, όσον αφορά την υλοποίηση, κόστος. Σύμφωνα με το SPMD μοντέλο ο κάθε επεξεργαστής τρέχει τον ίδιο κώδικα με διαφορετική

είσοδο δεδομένων (input) και διαφορετική έξοδο αποτελεσμάτων (output). Ο καταρκεματισμός του μη δομημένου πλέγματος (unstructure grid) είναι στατικός και έγινε με τις ρουτίνες του Metis v4x [34]. Η περίπτωση δυναμικού κατακερματισμού θα αύξανε την πολυπλοκότητα καθώς επίσης φεύγει από την λογική του SPMD αφού θα ήταν απαραίτητη συνεχώς η μεταφορά μεγάλου όγκου δεδομένων μεταξύ των επεξεργαστών. Ανάλογες εργασίες είναι [40],[10].

Κεφάλαιο 2

Μαθηματικό μοντέλο

Οι μη-γραμμικές εξισώσεις ρηχών υδάτων (NSWE) σε δύο χωρικές διαστάσεις παράγονται από την ολοκλήρωση κατά το βάθος των Navier-Stokes εξισώσεων, αγνοώντας την κάθετη επιτάχυνση των σωματιδίων του νερού και λαμβάνοντας την κατανομή της πίεσης ως υδροστατική. Αγνοώντας τα φαινόμενα Coriolis, επιφανειακής τάσης, οι εξισώσεις δίνονται στη διανυσματική μορφή νόμων διατήρησης ως:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathcal{H}(\mathbf{U}) = \mathcal{L}(\mathbf{U}, x, y) \quad \text{στο} \quad \Omega \times [0, t] \subset \mathbb{R}^2 \times \mathbb{R}^+, \tag{2.1}$$

όπου $\Omega \times [0, t]$ είναι χώρο-χρόνος στο Καρτεσιανό επίπεδο πάνω στο οποίο λύνονται οι εξισώσεις, το διάνυσμα των συντηρητικών μεταβλητών του νόμου διατήρησης και των συναρτήσεων ροής στην $x-$ και $y-$ διάσταση είναι

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathcal{H}(\mathbf{U}) = [\mathbf{F} \quad \mathbf{G}] = \begin{bmatrix} hu & hv \\ hu^2 + \frac{1}{2}gh^2 & huv \\ huv & hv^2 + \frac{1}{2}gh^2 \end{bmatrix},$$

όπου $\mathbf{u} = [u, v]^{\mathrm{T}}$ είναι το διάνυσμα του πεδίου ταχυτητας, $h(x, y, t)$ το βάθος της ροής (απόσταση μεταξύ του πυθμένα και της ελεύθερης επιφάνειας) και g η επιτάχυνση του πεδίου βαρύτητας. Ο πηγαίος όρος $\mathcal{L}(\mathbf{U}) = [\mathbf{R} + \mathbf{S}]$ μοντελοποιεί την επίδραση του ανάγλυφου του πυθμένα και της τριβής στη ροή. Το ανάγλυφο του πυθμένα ορίζεται από τη συνάρτηση $B(x, y)$, ενώ οι γεωμετρικοί όροι δίνονται από $\mathbf{R} = \mathbf{R}^1 + \mathbf{R}^2$ όπου

$$\mathbf{R}^1 = \begin{bmatrix} 0 & -gh\frac{\partial B(x,y)}{\partial x} & 0 \end{bmatrix}^{\mathrm{T}} \quad \text{και} \quad \mathbf{R}^2 = \begin{bmatrix} 0 & 0 & -gh\frac{\partial B(x,y)}{\partial y} \end{bmatrix}^{\mathrm{T}}.$$

Το στοιχείο $\mathbf{S}$ του πηγαίου όρου περιέχει την τάση της τριβής στον πυθμένα ως

$$\mathbf{S} = \begin{bmatrix} 0 & -\frac{\tau_{bx}}{\rho} & -\frac{\tau_{by}}{\rho} \end{bmatrix}^{\mathrm{T}},$$

όπου ρ είναι η πυκνότητα του νερού. Οι όροι της τάσης στον πυθμένα τ_{bx} και τ_{by} αντιπροσωπεύουν την διάχυση της ενέργειας, που οφείλεται στην τραχύτητα του πυθμένα πάνω στην ροή και προσεγγίζεται από την σχέση

$$\tau_{bx} = \rho C_f u\sqrt{u^2+v^2} \quad \text{και} \quad \tau_{by} = \rho C_f v\sqrt{u^2+v^2}.$$

Ο συντελεστής της τραχύτητας του πυθμένα υπολογίζεται από $C_f = gn_m^2/h^{1/3}$, με n_m να είναι ο συντελεστής τραχύτητας Manning.

Οι Ιακωβιανοί πίνακες του συστήματος (2.1) δίνονται ως

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

με $c = \sqrt{gh}$ να είναι η ταχύτητα του κύματος.

Η ολοκληρωτική μορφή των εξισώσεων πάνω σε ένα χωρίο Ω είναι

$$\frac{\partial}{\partial t}\iint\limits_{\Omega} \mathbf{U}d\Omega + \iint\limits_{\Omega}\Big(\nabla\cdot\mathcal{H}\Big)d\Omega = \iint\limits_{\Omega}\mathcal{L}d\Omega, \tag{2.2}$$

και από το θεώρημα απόκλισης του Gauss στο ολοκλήρωμα της ροής έχουμε

$$\frac{\partial}{\partial t}\iint\limits_{\Omega} \mathbf{U}d\Omega + \oint\limits_{\Gamma}\Big(\mathcal{H}\cdot\widetilde{\mathbf{n}}\Big)d\Gamma = \iint\limits_{\Omega}\mathcal{L}d\Omega, \tag{2.3}$$

όπου Γ είναι το σύνορο του χωρίου και $\widetilde{\mathbf{n}} = [\widetilde{n}_x, \widetilde{n}_y]^{\mathrm{T}}$ είναι το μοναδιαίο κάθετο προς τα έξω διάνυσμα. Μια διακριτή προσέγγιση της εξίσωσης (2.3) εφαρμόζεται σε κάθε υπολογιστικό κελί του $\Omega = \bigcup \Omega_P, P = 1, ..., N$, η χωρική ολοκλήρωση αντιπροσωπεύει τα ολοκληρώματα στο $|\Omega_P|$ του κελιού και τα επιφανειακά ολοκληρώματα αντιπροσωπεύουν την συνολική ροή στα όρια των κελιών. Το $\mathbf{U}_P$ υποδηλώνει την προσέγγιση όρων διατήρησης πάνω στο χωρίο την δεδομένη χρονική στιγμή, από την εξίσωση (2.3) η εξίσωση διατήρησης σε κάθε κελί μπορεί να γραφεί ως

$$\frac{\partial \mathbf{U}_P}{\partial t} = -\frac{1}{|\Omega_P|}\oint\limits_{\Gamma_P}\Big(\mathbf{F}\widetilde{n}_x + \mathbf{G}\widetilde{n}_y\Big)d\Gamma = \iint\limits_{\Omega_P}\mathcal{L}d\Omega. \tag{2.4}$$

Ο Ιακωβιανός πίνακας, $\mathbf{J}$,της συνάρτησης ροής $(\mathcal{H} \cdot \mathbf{n})$, επαληθεύετε ως

$$\mathbf{J} = \frac{\partial(\mathcal{H} \cdot \mathbf{n})}{\partial \mathbf{U}} = \mathbf{A}n_x + \mathbf{B}n_y$$

και μπορεί να εκφραστεί ως

$$\mathbf{J} = \begin{bmatrix} 0 & n_x & n_y \\ (c^2 - u^2)n_x - uvn_y & 2un_x + vn_y & un_y \\ -uvn_x + (c^2 - v^2)n_y & vn_x & un_x + 2vn_y \end{bmatrix}. \tag{2.5}$$

Οι ιδιοτιμές του $\mathbf{J}$ δίνονται από

$$\lambda_1 = un_x + vn_y - c, \quad \lambda_2 = un_x + vn_y, \quad \lambda_3 = un_x + vn_y + c, \tag{2.6}$$

με τα αντίστοιχα ιδιοδιανύσματα

$$\mathbf{r}_{1,3} = \begin{bmatrix} 1 & u \pm cn_x & v \pm cn_y \end{bmatrix}^{\mathrm{T}}, \quad \mathbf{r}_2 = \begin{bmatrix} 0 & -cn_y & cn_x \end{bmatrix}^{\mathrm{T}} \tag{2.7}$$

Από τα παραπάνω ιδιοδιανύσματα δύο πίνακες $\mathbf{P}$και $\mathbf{P}^{-1}$μπορούν να κατασκευαστούν με την ιδιότητα ότι διαγωνοποιούν τον Ιακωβιανό πίνακα $\mathbf{J}$ ως

$$\mathbf{J} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1},$$

όπου $\mathbf{\Lambda}$ είναι ο διαγώνιος πίνακας τις ιδιοτιμές στην κύρια διαγώνιο.

Κεφάλαιο 3

Κεντροθετιμένη Διακριτοποίηση Πεπερασμένων Όγκων

3.1 Πεπερασμένοι όγκοι

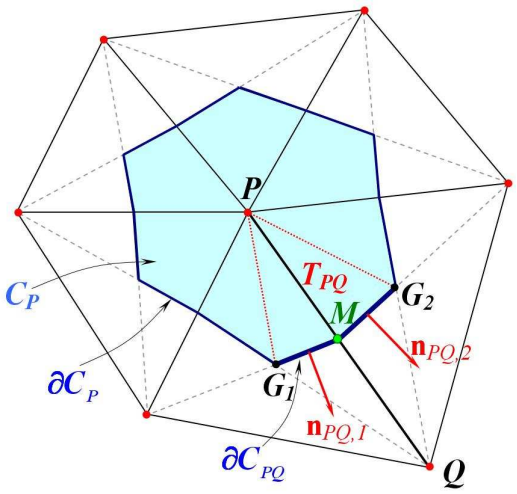
Η αρχική διακριτοποίηση του υπολογιστικού χωρίου γίνεται με τριγωνισμούς. Το σύνορο ∂C_P του χωρίου ελέγχου C_P γύρω από κάθε εσωτερικό κόμβο P του πλέγματος ορίζεται από την ένωση των βαρύκεντρων των τριγώνων που έχουν τον κόμβο P ως κοινή κορυφή με τα μέσα των αντίστοιχων ακμών που τέμνονται στον κόμβο P, Median Dual-approach,βλ. Σχήμα (3.1). Συνεπώς οι άγνωστες ποσότητες ορίζονται στο σημείο (κόμβο) P (Node-Centered).

Με $\partial C_{PQ} = \overline{G_1 M G_2}$ ορίζουμε το κοινό τμήμα του ∂C_P και ∂C_Q, όπου M είναι το μέσο της ακμής PQ. Το προς τα έξω κάθετο διάνυσμα στο ∂C_{PQ} είναι $\mathbf{n}_{PQ} = [n_{PQx}, n_{PQy}]^{\mathrm{T}}$ και $\widetilde{\mathbf{n}}_{PQ} = [\widetilde{n}_{PQx}, \widetilde{n}_{PQy}]^{\mathrm{T}}$ είναι το αντίστοιχο μοναδιαίο διάνυσμα. Αν $\mathbf{n}_{PQ,1}$ είναι το κάθετο στο $\overline{G_1 M}$ (με νόρμα ίση με το μήκος του $\overline{G_1 M}$), και $\mathbf{n}_{PQ,2}$ είναι το κάθετο στο $\overline{M G_2}$ (με νόρμα ίση με το μήκος του $\overline{M G_2}$) τότε

$$\mathbf{n}_{PQ} = \int_{\partial C_{PQ}} \widetilde{\mathbf{n}}\, dl = \mathbf{n}_{PQ,1} + \mathbf{n}_{PQ,2},$$

όπου dl είναι υπολογισμένο κατά μήκος του ∂C_{PQ}. Το υπό-κελί T_{PQ} είναι η ένωση των τριγώνων $G_1 M P$ και $M G_2 P$. Με A_P ορίζουμε το εμβαδόν του υπολογιστικού κελιού C_P.

Για έναν συνοριακό κόμβο P του πλέγματος ο ορισμός του κελιού ελέγχου



Σχήμα 3.1: *Ορισμός του κελιού ελέγχου για ένα εσωτερικό κόμβο του υπολογιστικού πλέγματος*

περιγράφεται στο Σχήμα (3.2). Το προς τα έξω κάθετο διάνυσμα στο $\overline{M_1PM_2}$ είναι $\mathbf{n}_P = [n_{Px}, n_{Py}]^{\mathrm{T}}$, όπου το $\widetilde{\mathbf{n}}_P = [\widetilde{n}_{Px}, \widetilde{n}_{Py}]^{\mathrm{T}}$ είναι το αντίστοιχο μοναδιαίο διάνυσμα. Αν $\mathbf{n}_{P,1}$ είναι κάθετο στο $\overline{M_1P}$ (με νόρμα ίση με το μήκος του $\overline{M_1P}$), όπου $\mathbf{n}_{P,2}$ είναι κάθετο στο $\overline{PM_2}$ (με νόρμα ίση με το μήκος του $\overline{PM_2}$) τότε:

$$\mathbf{n}_P = \int\limits_{\overline{M_1PM_2}} \widetilde{\mathbf{n}}\, dl = \mathbf{n}_{P,1} + \mathbf{n}_{P,2},$$

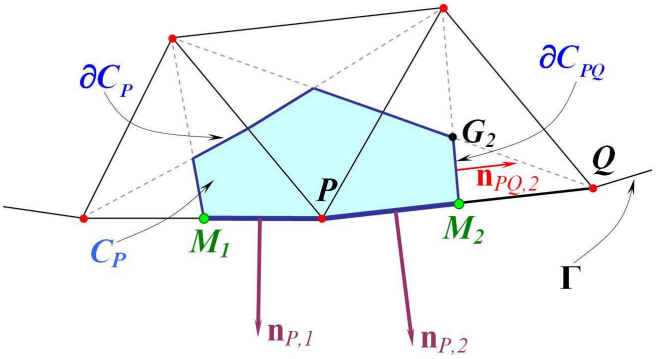
όπου dl είναι υπολογισμένο στο $\overline{M_1PM_2}$. Επίσης, στην περίπτωση ενός συνοριακού κόμβου P, $\partial C_{PQ} = \overline{M_2G_2}$ και αν $\mathbf{n}_{PQ,2}$ είναι κάθετο στο $\overline{MG_2}$ (με νόρμα ίση με το μήκος του $\overline{MG_2}$) τότε:

$$\mathbf{n}_{PQ} = \int\limits_{\partial C_{PQ}} \tilde{\mathbf{n}}\, dl = \mathbf{n}_{PQ,2},$$

όπου το *dl* είναι υπολογισμένο κατά μήκος του ∂C_{PQ}.

Αν Γ είναι σύνορο του χωρίου και K_P η ομάδα γειτονικών κόμβων του P, τότε ∂C_P περιγράφεται ως:

$$\partial C_P = \bigcup_{Q \in K_P} \partial C_{PQ} + (\partial C_P \cap \Gamma)\,.$$



Σχήμα 3.2: *Ορισμός του υπολογιστικού κελιού για ένα συνοριακό κόμβο*

3.2 Ολοκλήρωση των εξισώσεων και ορισμός της αριθμητικής ροής

Ακολουθώντας την εξίσωση (2.3) η ολοκληρωτική μορφή της εξίσωσης στο δημιουργημένο node-centered όγκο ελέγχου δίνεται από

$$\iint\limits_{C_P} \frac{\partial \mathbf{U}}{\partial t} dxdy + \oint\limits_{\partial C_P} \Big(\mathbf{F}\widetilde{n}_x + \mathbf{G}\widetilde{n}_y\Big) dl = \iint\limits_{C_P} \mathcal{L}\, dxdy$$

ή

$$\iint\limits_{C_P} \frac{\partial \mathbf{U}}{\partial t} dxdy + \sum_{Q \in K_P} \left\{ \int\limits_{\partial C_{PQ}} \Big(\mathbf{F}\widetilde{n}_x + \mathbf{G}\widetilde{n}_y\Big) dl \right\} + \int\limits_{\partial C_P \cap \Gamma} \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big) dl = \iint\limits_{C_P} \mathcal{L}\, dxdy.$$

Χωρίζουμε τον πηγαίο όρο του ολοκληρώματος σε άθροισμα των ολοκληρωμάτων στα υπό-κελιά T_{PQ}, $Q \in K_P$,

$$\iint\limits_{C_P} \frac{\partial \mathbf{U}}{\partial t} dxdy + \sum_{Q \in K_P} \left\{ \int\limits_{\partial C_{PQ}} \Big(\mathbf{F}\widetilde{n}_x + \mathbf{G}\widetilde{n}_y\Big) dl \right\} + \int\limits_{\partial C_P \cap \Gamma} \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big) dl = \sum_{Q \in K_P} \left\{ \iint\limits_{T_{PQ}} \mathcal{L}\, dxdy \right\}. \tag{3.1}$$

Για την ολοκλήρωση της χρονικής παραγώγου του διανύσματος των μεταβλητών του νόμου διατήρησης, υποθέτουμε μια ομοιόμορφη κατανομή των άγνωστων ποσοτήτων στο C_P, ίση με την τιμή του στον κόμβο P

$$\iint\limits_{C_P} \frac{\partial \mathbf{U}}{\partial t} dxdy = \frac{\partial \mathbf{U}_P}{\partial t} \iint\limits_{C_P} dxdy = \frac{\partial \mathbf{U}_P}{\partial t} A_P.$$

Στη συνέχεια εισάγουμε τα διανύσματα των συναρτήσεων ροής ως

$$\mathbf{\Phi}_{PQ} = \int\limits_{\partial C_{PQ}} \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big) dl \quad \text{και} \quad \mathbf{\Phi}_{P,out} = \int\limits_{\partial C_P \cap \Gamma} \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big) dl.$$

ως εκ τούτου η εξίσωση (3.1) γίνεται

$$\frac{\partial \mathbf{U}_P}{\partial t} A_P + \sum_{Q \in K_P} \mathbf{\Phi}_{PQ} + \mathbf{\Phi}_{P,out} = \sum_{Q \in K_P} \left\{ \iint\limits_{T_{PQ}.} \mathcal{L}\, dxdy \right\} \tag{3.2}$$

Για όλες τις ακμές του μη δομημένου πλέγματος η συνάρτηση ροής $\mathbf{\Phi}_{PQ}$ θα πρέπει να υπολογιστεί και να προστεθεί (με το κατάλληλο πρόσημο) στο άθροισμα της ροής των παρακείμενων κελιών C_P και C_Q αντίστοιχα. Η συνάρτηση ροής μπορεί, υποθέτοντας ομοιόμορφη κατανομή της στο ∂C_{PQ}, να προσσεγγιστεί από τον κανόνα του μέσου των επικαμπύλιων ολόκληρωμάτων του $\mathcal{H}$ στο ∂C_{PQ}, ίσο με την τιμή του στο μέσο M της ακμής PQ, έτσι

$$\mathbf{\Phi}_{PQ} = \int\limits_{\partial C_{PQ}} \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big) dl \approx \Big(\mathbf{F}\tilde{n}_x + \mathbf{G}\tilde{n}_y\Big)_M \|\mathbf{n}_{PQ}\| = \Big(\mathbf{F}n_{PQx} + \mathbf{G}n_{PQy}\Big)_M.$$

Προκειμένου να υπολογίσουμε το βαθμωτό γινόμενο $\mathbf{Z} = \mathcal{H} \cdot \tilde{\mathbf{n}} = \mathbf{F}n_{PQx} + \mathbf{G}n_{PQy}$ στο M και αντίστοιχα το διάνυσμα ροής, υποθέτουμε το μονοδιάστατο πρόβλημα Riemann μεταξύ της αριστερής (L) και δεξιάς (R) κατάστασης στις δύο πλευρές του σημείου M, και που ορίζεται από τα διανύσματα $\mathbf{U}^L_{PQ}$ και $\mathbf{U}^R_{PQ}$ αντίστοιχα, το οποίο επιλύεται με την προσέγγιση του Roe (Roe's approximate Riemman solver)[48]. Λαμβάνοντας υπόψη ένα απλοποιημένο πρόβλημα Riemann το οποίο λύνεται με ακριβή τρόπο το σχήμα του Roe χρησιμοποιείται ευρέως. Η λύση στηρίζεται στο γεγονός ότι ο Ιακωβιανός πίνακας (2.5) είναι σταθερός και υπολογίζεται με συνέπεια στους νόμους διατήρησης κατά μήκος του PQ. Έτσι,

$$\mathbf{\Phi}_{PQ} = \frac{1}{2}\Big\{\mathbf{Z}\left(\mathbf{U}^L_{PQ}, \mathbf{n}_{PQ}\right) + \mathbf{Z}\left(\mathbf{U}^R_{PQ}, \mathbf{n}_{PQ}\right)\Big\} - \frac{1}{2}\left|\widetilde{\mathbf{J}}_{PQ}\right|\Big(\mathbf{U}^R_{PQ} - \mathbf{U}^L_{PQ}\Big), \tag{3.3}$$

όπου $\widetilde{\mathbf{J}}_{PQ}$ είναι ο Ιακωβιανός πίνακας που υπολογίζεται με την προσέγγιση (μέσες τιμές) του Roe στις μεταβλητές $\mathbf{W} = [h, u, v]^{\mathrm{T}}$ και $\left|\widetilde{\mathbf{J}}_{PQ}\right|$ ορίζεται ως

$$\left|\widetilde{\mathbf{J}}_{PQ}\right| = \left(\widetilde{\mathbf{P}}\left|\widetilde{\mathbf{\Lambda}}\right|\widetilde{\mathbf{P}}^{-1}\right)_{PQ},$$

με $|\mathbf{\Lambda}|$ να είναι ο διαγώνιος πίνακας που περιέχει τις απόλυτες τιμές των ιδιοτιμών του $\mathbf{J}$. Το $\widetilde{\ }$ υποδηλώνει ότι οι πίνακες υπολογίστηκαν με τις μέσες τιμές του Roe που δίνονται ως,

$$\widetilde{h} = \sqrt{h^L \cdot h^R},\ \ \widetilde{u} = \frac{\sqrt{h^L}u^L + \sqrt{h^R}u^R}{\sqrt{h^L} + \sqrt{h^R}},\ \ \widetilde{v} = \frac{\sqrt{h^L}v^L + \sqrt{h^R}v^R}{\sqrt{h^L} + \sqrt{h^R}}$$

και $\widetilde{c} = \sqrt{g\frac{h^L + h^R}{2}}$.
Η εξίσωση (3.3) μπορεί εναλλακτικά να γραφεί με την παρακάτω μορφή, η οποία και χρησιμοποιείται στην υλοποίηση

$$\mathbf{\Phi}_{PQ} = \mathbf{Z}\left(\mathbf{U}^L_{PQ}, n_{PQ}\right) + \widetilde{\mathbf{J}}^-_{PQ}\Big(\mathbf{U}^R_{PQ} - \mathbf{U}^L_{PQ}\Big), \tag{3.4}$$

όπου

$$\widetilde{\mathbf{J}}^-_{PQ} = \left(\widetilde{\mathbf{P}}\ \ \widetilde{\mathbf{\Lambda}}^-\ \widetilde{\mathbf{P}}^{-1}\right)_{PQ}\ ,\quad \widetilde{\mathbf{\Lambda}}^- = \mathrm{diag}\left\{\widetilde{\lambda}^-_i\right\}\ ,\quad \widetilde{\lambda}^-_i = \min\left(\widetilde{\lambda}_i, 0\right),\ i = 1, 2, 3,$$

και

$$\mathbf{Z}\left(\mathbf{U}^L_{PQ}, n_{PQ}\right) = \mathbf{F}^L n_{PQx} + \mathbf{G}^L n_{PQy} = \begin{bmatrix} h\Big(u\, n_{PQx} + v\, n_{PQy}\Big) \\ hu\Big(u\, n_{PQx} + v\, n_{PQy}\Big) + \dfrac{1}{2}gh^2 n_{PQx} \\ hv\Big(u\, n_{PQx} + v\, n_{PQy}\Big) + \dfrac{1}{2}gh^2 n_{PQy} \end{bmatrix}^L.$$

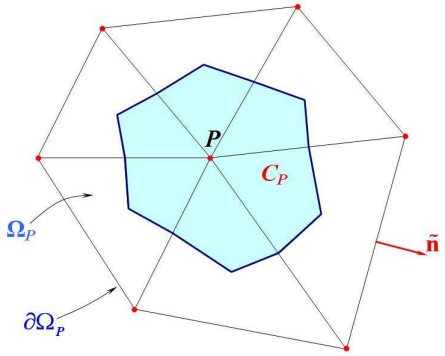
Το πρώτης τάξης ακρίβειας (στο χώρο) αριθμητικό σχήμα προκύπτει εάν οι τιμές $\mathbf{U}^L_{PQ} = \mathbf{U}_P$ και $\mathbf{U}^R_{PQ} = \mathbf{U}_Q$, δηλαδή υποθέτουμε μια κατά τμήματα σταθερή προσέγγιση των αγνώστων ποσοτήτων στους όγκους ελέγχου.

3.3 Δεύτερης τάξης σχήμα αριθμητικής ροής

Προκειμένου να βελτιωθεί η χωρική τάξη ακρίβειας του αριθμητικού σχήματος περισσότεροι κόμβοι θα πρέπει να λαμβάνονται υπόψη στον υπολογισμό της αριθμητικής ροής πάνω σε ένα αριθμητικό κελί. Ως εκ τούτου εφαρμόζεται ένα δεύτερης τάξης σχήμα βασισμένο στην γραμμική ανακατασκευή

MUSLC [54] των βασικών μεταβλητών σε κάθε όγκο ελέγχου χρησιμοποιώντας έναν περιορισμό στην κλίση που ελέγχει το συνολική μεταβολή στο υπο ανακατασκευή πεδίο.

Για το λόγο αυτό η κλίση $(\nabla \mathbf{W})_P$ πρέπει να υπολογιστεί σε κάθε όγκο ελέγχου P του υπολογιστικού πλέγματος εφαρμόζοντας το Green-Gauss θεώρημα στη περιοχή Ω_P, βλ. Σχήμα (3.3), το οποίο περιγράφεται από την ένωση όλων των τριγώνων που μοιράζονται την κορυφή P, ακολουθώντας το [3]. Το $(\nabla \mathbf{W})_P$ υπολογίζεται ως μέσος όρος λαμβάνοντας υπόψη ότι η διακριτή λύση του $(\nabla \mathbf{W})_P$ μεταβάλλεται γραμμικά, το οποίο σημαίνει ότι η κλίση είναι σταθερή (Green-Gauss γραμμική αναπαράσταση). Για κάθε συνηστώσα του $\mathbf{W}$ έχουμε



Σχήμα 3.3: *Ορισμός της περιοχής όπου υπολογίζεται το διάνυσμα κλίσης των βασικών μεταβλητών*

$$\iint_{\Omega_P} \nabla w_i dA = \oint_{\partial\Omega_P} w_i \hat{\mathbf{n}}\, dl,$$

από την οποία αποδυκνείεται ότι

$$(\nabla w_i)_P = \frac{1}{A_P} \sum_{Q \in K_P} \frac{1}{2}\Big(w_{i,P} + w_{i,Q}\Big)\mathbf{n}_{PQ}.$$

Στη περίπτωση που ο κόμβος P είναι συνοριακός βλ. Σχήμα (3.2) η προηγούμενη σχέση μετασχηματίζεται ώς:

$$(\nabla w_i)_P = \frac{1}{A_P}\left\{ \sum_{Q \in K_P} \frac{1}{2}\Big(w_{i,P} + w_{i,Q}\Big)\mathbf{n}_{PQ} + w_{i,P}\Big(\mathbf{n}_{P,1} + \mathbf{n}_{P,2}\Big) \right\}.$$

Σε αυτό το σημείο είναι σημαντικό να πούμε ότι, οι παραπάνω σχέσεις είναι βασισμένες στις ακμές του υπολογιστκού πλέγματος (τριγωνισμού) και είναι συμβατές με την γενική δομή της υλοποίησης μας. Στο μη-δομημένο υπολογιστικό πλέγμα το σχήμα MUSCL εφαρμόζεται σε κάθε ακμή. Δεξιά και αριστερά οι βασικές μεταβλητές στο μέσο M της ακμής PQ προσεγγίζονται (ανακατασκευάζονται γραμμικά) ως

$$\begin{aligned} w^L_{i,PQ} &= w_{i,P} + \frac{1}{2}\mathbf{r}_{PQ}\cdot(\nabla w_i)_P\,, \\ w^R_{i,PQ} &= w_{i,Q} - \frac{1}{2}\mathbf{r}_{PQ}\cdot(\nabla w_i)_Q\,, \end{aligned}$$

όπου $\mathbf{r}_{PQ}$ είναι το διάνυσμα που συνδέει τους κόμβους P και Q και από εδώ και στο εξής, οι εκθέτες θα δηλώνουν τις ανακατασκευασμένες τιμές, εκτός και αν ορίζεται διαφορετικά. Για την πρόληψη ταλαντώσεων από το αριθμητικό σχήμα στη ανακατασκευή της λύσης εφαρμόζεται αυστήρη μονοτονία χρησιμοποιώντας τον Albada-van Leer περιοριστή (limiter) κλίσης [53] με αποτέλεσμα

$$\begin{aligned} w^L_{i,PQ} &= w_{i,P} + \frac{1}{2}\mathrm{LIM}\,(a_P, b)\,; \\ w^R_{i,PQ} &= w_{i,Q} - \frac{1}{2}\mathrm{LIM}\,(a_Q, b)\,, \end{aligned}$$

όπου

$$\begin{aligned} b &= w_{i,Q} - w_{i,P}, \\ a_P &= 2\Big(\Big(\nabla w_i\Big)_P\cdot\mathbf{r}_{PQ}\Big) - b, \\ a_Q &= 2\Big((\nabla w_i)_Q\cdot\mathbf{r}_{PQ}\Big) - b \end{aligned}$$

και

$$\mathrm{LIM}\,(a,b) = \begin{cases} \dfrac{(a^2+e)\,b + (b^2+e)\,a}{a^2+b^2+2e}, & \text{if } \; ab > 0 \\ 0, & \text{if } \; ab \le 0, \end{cases}$$

όπου $0 < e << 1$, χρησιμοποιείται για την αποφυγή διάχυσης λόγο του μηδέν ($e = 10^{-16}$ στην υλοποίηση μας). Μπορούν επίσης να εφαρμοστούν και άλλοι περιοριστές αλλά στην παρούσα εργασία έχει χρησιμοποιήθει ο παραπάνω. Αξίζει να σημειωθεί εδώ ο η ίδια ανακατασκευή χρησιμοποιείται και για τον υπολογισμό της κλίσης της βαθυμετρίας $B(x,y)$, η οποία χρειάζεται για τον υπολογισμό του πηγαίου όρου της κλίσης του πυθμένα.

3.4 Διακριτοποίηση πηγαίων όρων κλίσης πυθμένα

Η ολοκλήρωση των πηγαίων όρων της κλίσης τους πυθμένα **R(U)** πάνω στον όγκο ελέγχου C_P υπολογίζεται ως

$$\iint_{C_P} \mathbf{R}\left(\mathbf{U}\right) dxdy = \iint_{C_P} \left\{\mathbf{R}^1 + \mathbf{R}^2\right\} dxdy =$$
$$\iint_{C_P} \left\{ \begin{bmatrix} 0 \\ -gh \\ 0 \end{bmatrix} \frac{\partial B}{\partial x} + \begin{bmatrix} 0 \\ 0 \\ -gh \end{bmatrix} \frac{\partial B}{\partial y} \right\} dxdy.$$

Υπολογίζουμε τον όρο τοπογραφίας ως διάνυσμα ροής

$$\mathbf{\Psi}_{PQ} = \iint_{T_{PQ}} \left\{ \begin{bmatrix} 0 \\ -gh \\ 0 \end{bmatrix} \frac{\partial B}{\partial x} + \begin{bmatrix} 0 \\ 0 \\ -gh \end{bmatrix} \frac{\partial B}{\partial y} \right\} dxdy, \quad Q \in K_P,$$

ως εκ τούτου,

$$\iint_{C_P} \mathbf{R}\left(\mathbf{U}\right) dxdy = \sum_{Q \in K_P} \mathbf{\Psi}_{PQ}.$$

Για όλες τις ακμές, του μη δομημένου πλέγματος, το διάνυσμα $\mathbf{\Psi}_{PQ}$ θα πρέπει να υπολογίζεται έτσι ώστε να προστεθεί σωστά στην ροή το άθροισμα των δύο γειτονικών κελιών C_P και C_Q αντίστοιχα. Το διάνυσμα $\mathbf{\Psi}_{PQ}$ προσεγγίζεται στην διακριτή μορφή $\widetilde{\mathbf{\Psi}}_{PQ}$. Το διακριτό διάνυσμα ροής των πηγαίων όρων εξαρτάται από τις τιμές των μεταβλητών στο σύνορο ∂C_{PQ} του κελιού ελέγχου, και στο αντίστοιχο κάθετο διάνυσμα. Όπως έχει αποδειχθεί στο [8] και [7], ένα upwind σχήμα χρησιμοποιείται για την προσέγγιση του πηγαίου όρου της τοπογραφίας ώστε να αποφευχθούν μη φυσικές ταλαντώσεις στην λύση ικανοποιώντας την λεγόμενη ιδιότητα-$\mathcal{C}$ της στάσιμης κατάστασης σε ηρεμία, $u = v = 0$, πάνω από τοπογραφία. Για να επιτευχθεί αυτό η ολοκλήρωση του πηγαίου όρου προβάλλεται στα ιδιοδιανύσματα του $\tilde{\mathbf{J}}$ και η γραμμικοποιημένοι μορφή της μπορεί να γραφτεί ως:

$$\widetilde{\mathbf{\Psi}}_{PQ} = \left(\widetilde{\mathbf{P}}\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ}.$$

Προκειμένου να επιτευχθεί η αριθμητική κατάσταση ισσοροπίας (ως εκ τούτου ικανοποιείται η ιδιότητα-$\mathcal{C}$) ο πηγαίος όρος της τοπογραφίας πρέπει να γραμμικοποιηθεί με τον ίδιο τρόπο (Roe-averaged state) όπως ο όρος της αριθμητικής ροής $\mathbf{\Phi}$ προηγούμενα. Η upwind διακριτοποίηση του πηγαίου όρου παρέχει τους παρακάτω δύο όρους οι οποίοι θα προστεθούν στον πηγαίο όρο για τους αντίστοιχους όγκους ελέγχου γύρω από τα σημεία P και Q αντίστοιχα:

$$\widetilde{\mathbf{\Psi}}^{-}_{PQ} = \left(\widetilde{\mathbf{P}}\mathbf{I}^{-}\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ}, \;\; \widetilde{\mathbf{\Psi}}^{+}_{PQ} = \left(\widetilde{\mathbf{P}}\mathbf{I}^{+}\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ} = \widetilde{\mathbf{\Psi}}^{-}_{QP},$$

όπου $\mathbf{I}^{\pm} = \widetilde{\mathbf{\Lambda}}^{\pm}\widetilde{\mathbf{\Lambda}}^{-1}$. όπου με κατάλληλες αλγεβρικές πράξεις έχουμε:

$$\widetilde{\mathbf{\Psi}}^{-}_{PQ} = \frac{1}{2}\left(\widetilde{\mathbf{P}}\left(\mathbf{I} - \left|\widetilde{\mathbf{\Lambda}}\right|\widetilde{\mathbf{\Lambda}}^{-1}\right)\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ} \tag{3.5}$$

και:

$$\widetilde{\mathbf{\Psi}}^{+}_{PQ} = \frac{1}{2}\left(\widetilde{\mathbf{P}}\left(\mathbf{I} + \left|\widetilde{\mathbf{\Lambda}}\right|\widetilde{\mathbf{\Lambda}}^{-1}\right)\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ}. \tag{3.6}$$

Ο όρος $(\widetilde{\mathbf{\Psi}})_{PQ}$ στην εξίσωση (3.5) και (3.6) προσεγγίζεται με τον παρακάτω τρόπο ώστε να εξισορροπηθούν οι αντίστοιχοι όροι ροής στις υδραστατικές συνθήκες,

$$(\widetilde{\mathbf{\Psi}})_{PQ} = \begin{bmatrix} 0 \\ -g\dfrac{h^L + h^R}{2}\left(B^R - B^L\right)n_{PQx} \\ -g\dfrac{h^L + h^R}{2}\left(B^R - B^L\right)n_{PQy} \end{bmatrix}_{PQ}$$

όπου, για το 1ης τάξης σχήμα, το αποτέλεσμα είναι

$$(\widetilde{\mathbf{\Psi}})_{PQ} = \begin{bmatrix} 0 \\ -g\dfrac{h_P + h_Q}{2}\left(B_Q - B_P\right)n_{PQx} \\ -g\dfrac{h_P + h_Q}{2}\left(B_Q - B_P\right)n_{PQy} \end{bmatrix}_{PQ}.$$

Η συνεισφορά της αριθμητικής ροής πρέπει τώρα να είναι ίση με αυτή του πηγαίου όρου για τις υδροστατικές συνθήκες:

$$\mathbf{Z}\left(\mathbf{U}^L_{PQ}, n_{PQ}\right) + \left(\widetilde{\mathbf{P}}\widetilde{\mathbf{\Lambda}}^{-}\widetilde{\mathbf{P}}^{-1}\right)\left(\mathbf{U}^R_{PQ} - \mathbf{U}^L_{PQ}\right) = \left(\frac{1}{2}\widetilde{\mathbf{P}}\left(\mathbf{I} - \left|\widetilde{\mathbf{\Lambda}}\right|\widetilde{\mathbf{\Lambda}}^{-1}\right)\widetilde{\mathbf{P}}^{-1}\widetilde{\mathbf{\Psi}}\right)_{PQ}. \tag{3.7}$$

Υπενθυμίζουμε ότι,

$$\widetilde{c} = \sqrt{g\frac{h^L + h^R}{2}}$$

και όσον αφορά τις υδροστατικές συνθήκες έχουμε:

$$u = v = 0, \quad B^R - B^L = -\left(h^R - h^L\right).$$

Η ισότητα της εξίσωσης (3.7) δίνει το ακόλουθο:

$$\begin{bmatrix} \frac{-\widetilde{c}\|\mathbf{n}_{PQ}\|}{2}\left(h^R - h^L\right) \\ \frac{\widetilde{c}^2 n_{PQx}}{2}\left(h^R - h^L\right) \\ \frac{\widetilde{c}^2 n_{PQy}}{2}\left(h^R - h^L\right) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{2}g\left(h^L\right)^2 n_{PQx} \\ \frac{1}{2}g\left(h^L\right)^2 n_{PQy} \end{bmatrix} = \begin{bmatrix} -\widetilde{c}\frac{\left(h^R - h^L\right)}{2}\|\mathbf{n}_{PQ}\| \\ \widetilde{c}^2\frac{\left(h^R - h^L\right)}{2}n_{PQx} \\ \widetilde{c}^2\frac{\left(h^R - h^L\right)}{2}n_{PQy} \end{bmatrix}. \tag{3.8}$$

Στην περίπτωση 1ης τάξης σχήμα η εξίσωση (3.8) δίνει:

$$\begin{bmatrix} \frac{-\widetilde{c}\|\mathbf{n}_{PQ}\|}{2}\left(h_Q - h_P\right) \\ \frac{\widetilde{c}^2 n_{PQx}}{2}\left(h_Q - h_P\right) \\ \frac{\widetilde{c}^2 n_{PQy}}{2}\left(h_Q - h_P\right) \end{bmatrix} + \frac{1}{2}gh_P^2\begin{bmatrix} 0 \\ n_{PQx} \\ n_{PQy} \end{bmatrix} = \begin{bmatrix} -\widetilde{c}\frac{\left(h_Q - h_P\right)}{2}\|\mathbf{n}_{PQ}\| \\ \widetilde{c}^2\frac{\left(h_Q - h_P\right)}{2}n_{PQx} \\ \widetilde{c}^2\frac{\left(h_Q - h_P\right)}{2}n_{PQy} \end{bmatrix},$$

και μόνο το αριστερό μέλος είναι αυτό που δεν μπορεί να ακυρωθεί απευθείας από κάποιον παρόμοιο όρο στο δεξί μέλος της ισότητας. Αν λάβουμε όμως υπόψη ότι όλες οι ροές αθροίζονται σε όλες τις ακμές που έχουν κοινή κορυφή P (κατά μήκος ∂C_P), ο δεύτερος όρος εξαφανίζεται. Αυτό οφείλεται στο γεγονός ότι

$$\oint_{\partial C_P} \mathbf{n}dl = 0, \quad \oint_{\partial C_P}\left(\mathbf{n}\cdot\hat{\mathbf{i}}\right)dl = 0 \Rightarrow \oint_{\partial C_P} n_x dl = 0 \quad \text{και} \quad \oint_{\partial C_P}\left(\mathbf{n}\cdot\hat{\mathbf{j}}\right)dl = 0 \Rightarrow \oint_{\partial C_P} n_y dl = 0.$$

Τότε

$$\sum_{Q\in K_P}\frac{1}{2}g\left(h^L\right)^2\begin{bmatrix} 0 \\ n_{PQx} \\ n_{PQy} \end{bmatrix} = \frac{1}{2}gh_P^2\sum_{Q\in K_P}\begin{bmatrix} 0 \\ n_{PQx} \\ n_{PQy} \end{bmatrix} = \frac{1}{2}gh_P^2\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Στην περίπτωση της 2ης τάξης MUSCL διακριτοποίησης, η συνεισφορά της αριθμητικής ροής δεν εξισώνεται με την διακριτοποίηση του πηγαίου όρου για τις υδροστατικές συνθήκες, ακόμα και στην περίπτωση που είναι 2ης τάξης και η διακριτοποίηση της κλίσης του πυθμένα (όπως είχε τονιστεί προηγουμένος, η ίδια διαδικασία ανακατασκευής χρησιμοποιείται για να υπολογιστεί η κλισης της τοπογραφίας $B(x,y)$). Ο λόγος είναι ότι ο δεύτερος όρος της εξίσωσης (3.8) δεν εξαφανίζεται λόγω του ότι το ανακατασκευασμένο h^L έχει διαφορετική τιμή για κάθε ένα από τα διαφορετικά $\partial C_{PQ},\ Q \in K_P$ και ως εκ τούτου

$$\sum_{Q\in K_P} \frac{1}{2} g \left(h^L\right)^2 \begin{bmatrix} 0 \\ n_{PQx} \\ n_{PQy} \end{bmatrix} \neq 0.$$

Σε αυτή την περίπτωση, ένας όρος θα πρέπει να προστεθεί στον πηγαίο όρο $\widetilde{\mathbf{\Psi}}^{-}_{PQ}$, προκειμένου να εξισώσει τους αντίστοιχους μη μηδενικούς όρους της αριθμητικής ροής, σύμφωνα με την εργασία [32, 45], έχοντας κατά νού ότι όλες οι τιμές έχουν ανακατασκευστεί. Ο όρος που πρέπει να προστεθεί είναι:

$$\begin{bmatrix} 0 \\ -g\dfrac{h^L+h_P}{2}\left(B^L-B_P\right)n_{PQx} \\ -g\dfrac{h^L+h_P}{2}\left(B^L-B_P\right)n_{PQy} \end{bmatrix}.$$

Ο παραπάνω όρος απαλήφεται για το 1ης τάξης σχήμα ως $B^L = B_P$. Για τις υδροστατικές συνθήκες έχουμε:

$$B^L - B_P = -\left(h^L - h_P\right)$$

και έτσι ο νέος όρος είναι γραμμένος για τις υδροστατικές συνθήκες ως:

$$\begin{bmatrix} 0 \\ -g\dfrac{h^L+h_P}{2}\left(B^L-B_P\right)n_{PQx} \\ -g\dfrac{h^L+h_P}{2}\left(B^L-B_P\right)n_{PQy} \end{bmatrix} = \begin{bmatrix} 0 \\ -g\dfrac{h^L+h_P}{2}\left(h_P-h^L\right)n_{PQx} \\ -g\dfrac{h^L+h_P}{2}\left(h_P-h^L\right)n_{PQy} \end{bmatrix} =$$

$$= \begin{bmatrix} 0 \\ -g\dfrac{1}{2}\left(h_P^2-\left(h^L\right)^2\right)n_{PQx} \\ -g\dfrac{1}{2}\left(h_P^2-\left(h^L\right)^2\right)n_{PQy} \end{bmatrix} = \begin{bmatrix} 0 \\ -g\dfrac{1}{2}h_P^2 n_{PQx} \\ -g\dfrac{1}{2}h_P^2 n_{PQy} \end{bmatrix} + \begin{bmatrix} 0 \\ g\dfrac{1}{2}\left(h^L\right)^2 n_{PQx} \\ g\dfrac{1}{2}\left(h^L\right)^2 n_{PQy} \end{bmatrix}. \tag{3.9}$$

Ο δεύτερος όρος του δεξιού μέλους της εξίσωσης (3.9) τώρα εξισώνεται με τον αντίστοιχο όρο της αριθμητικής ροής, ο δεύτερος προστέθηκε στο δεξί μέλος της εξίσωσης (3.8) ενώ ο πρώτος όρος απαλήφεται, οπότε έχουμε:

$$\sum_{Q\in K_P} \frac{1}{2} g h_P^2 \begin{bmatrix} 0 \\ n_{PQx} \\ n_{PQy} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Ο όρος που προστήθεται στον πηγαίο όρο $\widetilde{\mathbf{\Psi}}^{+}_{PQ} = \widetilde{\mathbf{\Psi}}^{-}_{QP}$ (για τον υπολογισμό της ροής στο Q) ομοίος δίνεται ως:

$$\begin{bmatrix} 0 \\ -g\dfrac{h^R + h_Q}{2}\left(B^R - B_Q\right)(-n_{PQx}) \\ -g\dfrac{h^R + h_Q}{2}\left(B^R - B_Q\right)(-n_{PQy}) \end{bmatrix} = \begin{bmatrix} 0 \\ -g\dfrac{h^R + h_Q}{2}\left(B_Q - B^R\right) n_{PQx} \\ -g\dfrac{h^R + h_Q}{2}\left(B_Q - B^R\right) n_{PQy} \end{bmatrix}.$$

Τονίζουμε εδώ ότι, οι ανωτέρω υψηλή τάξης διόρθωση για τους πηγαίους όρους δίνει μια ακριβή ισορροπία μεταξύ της αριθμητικής και του πηγαίου όρου σε υδροστατικές συνθήκες (ροή σε κατάσταση ηρεμίας), με αποτέλεσμα ένα πλήρως 2ης τάξης σχήμα.

3.5 Στεγνό-υγρό μέτωπο, αντιμετώπιση για την διατήρηση της μάζας

Στις περισσότερες πραγματικές εφαρμογές υπάρχει κινούμενο σύνορο μεταξύ των στεγνών και υγρών περιοχών του υπολογιστικού πλέγματος κατά την μεταβολή της στάθμης του νερού. Αυτό το σύνορο χαρακτηρίζεται από μια υγρή-στεγανή κατάσταση και ένας ειδικός χειρισμός είναι απαραίτητος για τον ακριβή υπολογισμό του κινούμενου συνόρου. Το υπολογιστικό πλέγμα είναι συγκεκριμένο και σταθερό με τους υπολογιστικούς όγκους να χαρακτηρίζονται ως υγρό ή όχι κατά την προσέγγιση της λύσης. Με σκοπό να οριστεί πότε ένα κελί είναι στεγνό και πότε υγρό έχει οριστεί μια παράμετρος ε_{wd} που το καθορίζει. Αν η στάθμη του νερού είναι χαμηλότερη από το ε_{wd} σε ένα όγκο ελέγχου του πλέγματος τότε το χωρίο ελέγχου χαρακτηρίζεται ως στεγνό. Η επιλογή του ε_{wd} δεν είναι πάντα εύκολη υπόθεση και εξαρτάται

από το είδος του προβλήματος που μελετάμε, την ακρίβεια της μηχανής και το μέγεθως του υπολογιστικού πλέγματος. Παρακάτω αναλύονται τα βήματα που ακολουθούνται στην προσέγγιση του θγρού-στεγανού μετώπου.

Ας υποθέσουμε οτι έχουμε την περίπτωση υγρού-στεγανού μεταξύ των κόμβων P και Q, που συνδέονται από την ακμή PQ. Χωρίς περιορισμό της γενικότητας, υποθέτουμε ότι ο κόμβος Q είναι στεγανός και ο κόμβος P είναι υγρός. Με βάση την παράμετρο υγρού-στεγανού έχουμε

$$h_Q \leq \varepsilon_{wd} \quad \text{και} \quad h_P > \varepsilon_{wd}$$

και ορίζουμε $\mathbf{W}_Q = [0 \quad 0 \quad 0]^{\mathrm{T}}$. Αυτό αποτελεί κοινή πρακτική στους υπολογισμούς με τις εξισώσεις ρηχών υδάτων, παράδειγμα [30].

Σύμφωνα με την εργασία [17], ένα αριθμητικό σχήμα θεωρείται ότι είναι καλά ισορροπημένο εάν μπορεί να υπολογίζει ακριβώς σε στάσιμη κατάσταση λύσεις που αντιστοιχούν σε ροή σε ηρεμίας, ανεξάρτητα της ύπαρξης υγρών-στεγανών μετώπων ή όχι, προκειμένου να ικανοποιήσει την εκτεταμένη ιδιότητα-C . Για να επιτευχθεί αυτό πρέπει να επαναπροσδιορίσουμε τη βαθυμετρία στο σημείο Q. Ο λόγος για τον επανακαθορισμό της βαθυμετρίας είναι να επιτευχθεί η ακριβής αριθμητική ισορροπία μεταξύ της κλίσης του πυθμένα και των υδροστατικών όρων,[15]. Ο επαναπροσδιορισμός της βαθυμετρίας στο σημείο Q ορίζεται ως:

$$\Delta B_{PQ} = \begin{cases} -\left(h_Q - h_P\right), & \text{αν } h_P > \varepsilon_{wd} \text{ και } h_Q \leq \varepsilon_{wd} \text{ και } h_P < \Big(B_Q - B_P\Big), \\ B_Q - B_P & \text{αλλιώς}, \end{cases}$$

στον υπολογισμό του $(\widetilde{\mathbf{\Psi}})_{PQ}$ στην εξίσωση (3.5) και (3.6). Επιπλέον, για την ακμή PQ (όπου υπάρχει υγρό-στεγανό μεταξύ των σημείων) εφαρμόζουμε πάντοτε την χρήση ενώς πρώτης τάξεως σχήματος για την διακριτοποίηση της πρώτης εξίσωσης για τον υπολογισμό της στάθμης του νερού, ένα πρώτης τάξης σχήμα επίσης χρησιμοποιείται για όλες τις στεγανές περιοχές στη διακρητοποίηση. Αυτό εισάγεται με σκοπό να έχουμε μια τμηματικά σταθερή κατανομή της βαθυμετρίας και του βάθους στη περιοχή που ορίζεται ως υγρό-στεγανό μέτωπο. Το υβριδικό πρώτης-δεύτερης τάξης σχήμα έχει δώσει πολύ καλά αποτελέσματα στα διάφορα υπολογιστικά πειράματα. Στην περίπτωση που και οι δύο κορυφές της ακμής είναι υγρές, το δεύτερης τάξης σχήμα

χρησιμοποιείται και για τις τρείς εξισώσεις, εκτός και αν έχει επιλεγεί πρώτης τάξης σχήμα για όλο το υπολογιστικά πλέγμα.

Ενώ με τον παραπάνω επαναπροσδιορισμό του διακριτοποιημένης τοπογραφίας μπορεί κανείς να αντιμετωπίζει τις καταστάσεις ανύψωσης τοπογραφίας για μια ροή σε κατάσταση ηρεμίας περαιτέρω τροποποιήσεις πρέπει να γίνουν για μια ροή στην κίνηση. Για την ακμή PQ με ένα υγρό-στεγανό μέτωπο μεταξύ των κορυφών της, επιβάλουμε τον ακόλουθο περιορισμό για τον υπολογισμό των αντίστοιχων ροών, ακολουθώντας τις εργασίες [17, 19, 21],

$$An \left[h_P > \varepsilon_{wd} \text{ και } h_Q \leq \varepsilon_{wd} \text{ και } h_P < (B_Q - B_P) \right]$$

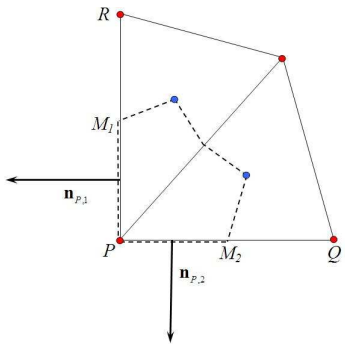
τότε $u^L = u^R = v^L = v^R = 0$.
Να σημειωθεί εδώ ότι το πεδίο ταχυτήτων δεν είναι μηδέν στο υγρό κελί P, αλλά η παραπάνω σχέση επιβάλεται προσωρινά όταν υπολογίζεται η αριθμητική ροή και συνεισφορά των πηγαίων όρων. Η υπολογιστική λύση είναι η ίδια αν ο Q είναι υγρός κόμβος και ο P στεγανός. Όπως έχει σημειωθεί στο [17] έχουμε καλύτερη προσέγγιση της λύσης στα υγρά σημεία όπου η ορμή είναι μικρότερη κατά απόλυτη τιμή.

Στην περίπτωση όπου η βαθυμετρία ενός στεγανού κελιού είναι μικρότερη ή ίση με αυτή στο υγρό κελί η ροή θα συνεχίσει και θα κατακλίσει το στεγανό κελί και συνήθως δεν είναι ανάγκη να μεταβάλουμε την κλίση του πυθμένα. Στην περίπτωση όμως που ο πυθμένας έχει απότομη κλίση, περισσότερο νερό από το κανονικό μπορεί να υπολογιστεί σαν αυτό που έχει κατακλίσει το στεγανό κελί, αυτό έχει σαν συνέπεια το βάθος στο υγρού κελί να γίνει αρνητικό και το αριθμητικό σχήμα να γίνει ασταθές. Σε αυτή την εργασία ακολουθώντας το [14], που παρουσιαστεί το παραπάνω φαινόμενο και προκειμένου να διατηρηθεί η ευστάθεια του αριθμητικού σχήματος στα κελιά που έχει υπολογιστεί αρνητικό βάθος μετά από ένα χρονικό βήμα η στάθμη του νερού και οι ταχύτητες μηδενίζονται. Αυτό προσθέτει νερό στο γενίκο σύστημα. Στη συνέχεια ο ίδιος όγκος νερού αφαιρείται από τα υγρά κελιά με σκόπο να διατηρηθεί η συνολική μάζα. Επιπλέον και προκειμένου να διατηρηθούν τα στοιχεία της ταχύτητας, οι διατηρητέες μεταβλητές uh και vh στα κελία όπου έχει μεταβληθεί ο όγκος του νερού αλλάζουν ανάλογα ώστα τα στοιχεία της ταχύτητας u και v να παραμείνουν τα ίδια στο ίδιο χρονικό

βήμα. Να σημειωθεί εδώ ότι η ίδια διαδικασία ακολουθείται και στα κελιά που είναι σχεδόν στεγνα (σύμφωνα με την παράμετρο ε_{wd}), και το βάθος του νερού εκεί ορίζεται μηδέν.

3.6 Συνοριακές συνθήκες

Στην node-based προσεγγίση οι βαθμοί ελευθερίας (άγνωστοι) ορίζονται κατευθείαν στο σύνορο, κατα συνέπεια η node-centered προσέγγιση είναι βολική για συνοριακές συνθήκες τύπου Dirichlet. Ωστόσο ένα θεμελιώδες πρόβλημα εμφανίζεται όταν δύο γειτονικές ακμές, με συνοριακό κόμβο στην ενωσή τους, έχουν διαφορετικού τύπου συνοριακές συνθήκες [43]. Οι συνοριακές συνθήκες που βασίζονται στις ακμές αντί τις κορυφές εφαρμόζονται καλύτερα σε ασθενής προσεγγίσεις όταν οι συνοριακές συνθήκες εισάγονται στα υπόλοιπα της ροής στο σύνορο, βλ. Σχήμα (3.4). Σε αυτή την περίπτωση οι συνοριακές συνθήκες είναι διατυπωμένες με παρόμοιο τρόπο και στη node-centered και στη cell-centered διακριτοποίηση (με το πλεονέκτημα της μη χρήσης ιδεατών (ghost) κελιών στην node-centered προσέγγιση.



Σχήμα 3.4: *Υλοποίηση συνοριακών συνθηκών για την node-centered διακριτοποίηση.*

Ας υποθέσουμε ότι η επιφάνεια PQ στο Σχήμα (3.4) ανήκει σε σύνορο τύπου τοίχου. Για την καλύτερη προσομοίωση του συνόρου τύπου τοίχου η ροή στο σύνορο PM_2 του εμπλεκόμενου όγκου ελέγχου υπολογίζεται με το κάθετο διάνυσμα στο πεδίο των ταχυτήτων ίσο με το μηδέν. Το $\mathbf{n}_{P,2}$ είναι το κάθετο διάνυσμα στη επιφάνεια PM_2, βλ. Σχήμα (3.4). Τότε σύμφωνα με την εξίσωση (3.4) της εμπλεκόμενης ροής (που προστίθεται στον όγκο ελέγχου)

υπολογίζεται ως

$$\mathbf{\Phi}_{P,2} = \mathbf{Z}\Big(\mathbf{U}_{P,2}, \mathbf{n}_{P,2}\Big) = \left[\begin{array}{c} h\Big(u\, n_{P,2x} + v\, n_{P,2y}\Big) \\ hu\Big(u\, n_{P,2x} + v\, n_{P,2y}\Big) + \dfrac{1}{2}gh^2 n_{P,2x} \\ hv\Big(u\, n_{P,2x} + v\, n_{P,2y}\Big) + \dfrac{1}{2}gh^2 n_{P,2y} \end{array}\right]_P .$$

Η υλοποίηση του μηδενικού κάθετου διανύσματος στο πεδίο των ταχυτήτων απαιτεί ότι

$$\Big(u\, n_{P,2x} + v\, n_{P,2y}\Big) = 0$$

με αποτέλεσμα

$$\mathbf{\Phi}_{P,2} = \left[\begin{array}{c} 0 \\ \dfrac{1}{2}gh_P^2\, n_{P,2x} \\ \dfrac{1}{2}gh_P^2\, n_{P,2y} \end{array}\right].$$

Αν υποθέσουμε ότι η επιφάνεια PR στο Σχήμα (3.4) είναι σύνορο τύπου εκροής (outflow) και $\mathbf{n}_{P,1}$ είναι το κάθετο στη επιφάνεια PM_1 διάνυσμα. Τότε, σύμφωνα με την εξίσωση (3.4) της εμπλεκόμενης ροής (που προστήθετε στον όγκο ελέγχου P) υπολογίζεται ως

$$\mathbf{\Phi}_{P,1} = \mathbf{Z}\Big(\mathbf{U}_{P,1}, \mathbf{n}_{P,1}\Big) = \left[\begin{array}{c} h\Big(u\, n_{P,1x} + v\, n_{P,1y}\Big) \\ hu\Big(u\, n_{P,1x} + v\, n_{P,1y}\Big) + \dfrac{1}{2}gh^2 n_{P,1x} \\ hv\Big(u\, n_{P,1x} + v\, n_{P,1y}\Big) + \dfrac{1}{2}gh^2 n_{P,1y} \end{array}\right]_P .$$

Ο δεύτερος όρος στην εξίσωση (3.4) εξαφανίζεται καθώς δεν προβλέπεται καμία διαφοροποίηση των συντηρητικών μεταβλητών που λαμβάνουν μέρος στο σύνορο της εκροής.

Για συνοριακές συνθήκες εισροής το διάνυσμα των συντηρητικών μεταβλητών επιβάλεται στους συνοριακούς κόμβους σε κάθε χρονικό βήμα, το οποίο χρησιμοποιείται για τον υπολογισμό της ροής στους εμπλεκόμενους όγκους ελέγχου. Στην περίπτωση συνόρου τύπου εκροής έχουμε την ίδια αντιμετώπιση. Η παραπάνω αντιμετώπιση των συνοριακών συνθηκών διατηρεί την ακρίβεια δεύτερης τάξης του σχήματος στους συνοριακούς κόμβους, που χρειάζονται για τον υπολογισμό της ροής στο σύνορο. Η θέση των βαθμών ελευθερίας απευθείας στο σύνορο στη node-centered διακριτοποίηση απλοποιεί την υλοποίηση για διαφόρου τύπου συνοριακές συνθήκες. Επιπλέον,

συνοριακές συνθήκες βασισμένες στις επιφάνειες του πλέγματος είναι συμβατές με την edge-based υλοποίηση.

3.7 Χρονική διακριτοποίηση

Στις προηγούμενες ενότητες εξετάσαμε την χωρική διακριτοποίηση σε πεπερασμένους όγκους. Προκειμένου να επιτευχθεί ένα πλήρως διακριτό σχήμα πρέπει να διακριτοποιήσουμε και τον χρονικό τελεστή. Για την χρονική διακριτοποίηση επιλέγουμε την άμεση Runge-Kutta (RK) τεσσάρων σταδίων, λόγω της ενισχυμένης περιοχής ευστάθειάς της [35, 36]. Η διακριτοποιημένη μορφή της εξίσωσης (3.2) γράφεται ως

$$\frac{\partial \mathbf{U}_P}{\partial t} = \text{RHS}\left(\mathbf{U}\right),$$

με RHS ορίζουμε την χωρική διακριτοποίηση.

Το RK σχήμα περιγράφεται παρακάτω

$$\begin{aligned}
\mathbf{U}_P^{(0)} &= \mathbf{U}_P^n; \\
\mathbf{U}_P^{(k)} &= \mathbf{U}_P^{(0)} + a_k\,\Delta t^n\, RHS\left(\mathbf{U}_P^{(k-1)}\right), \text{ για } k = 1, ..., 4; \\
\mathbf{U}_P^{n+1} &= \mathbf{U}_P^{(4)},
\end{aligned}$$

όπου $\Delta t^n = t^{n+1} - t^n$ είναι το χρονικό βήμα. Η βέλτιστες τιμές (λαμβάνοντας υπόψη την CFL συνθήκη) για a_k έχουμε, [35, 36]:

$$a_1 = 0.11, \;\; a_2 = 0.26, \;\; a_3 = 0.5 \quad \text{και} \quad a_4 = 1.0.$$

Όπως αναφέρεται στο [35, 36], όταν χρησιμοποιούμε την παράμετρο $a_3 = 0.5$, μπορεί να αποδειχθεί ότι η αντίστοιχη μέθοδος RK είναι δεύτερης τάξης ακρίβειας στο χρόνο. Το παραπάνω σχήμα μειώνει την Euler ολοκλήρωση για $k = 1$ και $a_1 = 1$.

Αν R_P είναι η ελάχιστη απόσταση μεταξύ του P και του ∂C_P, τότε το γενικό χρονικό βήμα Δt^n υπολογίζεται βάση του CFL όρου ως

$$\Delta t^n = CFL \cdot \min_P \left(\frac{R_P}{\left(\sqrt{u^2 + v^2} + c\right)_P^n} \right).$$

3.8 Διακριτοποίηση του όρου της τριβής

Σύμφωνα με τις εργασίες [13, 44, 33, 1] μια σημειο-κεντρικά έμεση στο χρόνο διακριτοποίηση του όρου της τριβής μπορεί να προκαλέσει αριθμητική αστάθεια όταν ο συντελεστής τραχύτητας είναι υψηλός. Για να προσεγγίσουμε κατάλληλα τον όρο της τριβης ακολουθούμε τις εργασίες [13, 1, 33]. Ξεκινάμε με μια ξεχωριστή έμεση διατύπωση των όρων της τριβής που μπορεί να έχει κανείς για τις μεταβλητές της ορμής στο $P-$κελί.

$$(hu)_P^{n+1} = (hu)_P^* - (ghS_x^f)_P^{n+1}\Delta t^n \tag{3.10}$$

$$(hv)_P^{n+1} = (hv)_P^* - (ghS_y^f)_P^{n+1}\Delta t^n \tag{3.11}$$

όπου υπολογίζονται οι τιμές που έχουν σύμανση $*$, χωρίς να να συμπεριλαμβάνονται οι δυνάμεις τριβής, χρησιμοποιώντας οποιοδήποτε σχήμα πεπερασμένων χωρίων όπως έχει ήδη περιγραφεί στις προηγούμενες ενότητες. Ορίζοντας

$$R_f = \frac{n_m^2||\mathbf{u}||}{h^{\frac{4}{3}}}$$

οι εξισώσεις (3.11) γράφονται:

$$\begin{aligned} & (hu)_P^{n+1} = (hu)_P^* - (ghuR_f)_P^{n+1}\Delta t^n \\ = \; & (hu)_P^* - (ghu)_P^{n+1}\left[(1-\theta)(R_f)_P^{n+1} + \theta(R_f)_P^n\right]\Delta t^n \end{aligned}$$

διαχωρίζοντας τα αμέσα και έμεσα τμήματα έχουμε

$$\begin{aligned} & (hu)_P^{n+1}\left[1 + (1-\theta)g(R_f)_P^{n+1}\Delta t^n\right] \\ = \; & (hu)_P^* - g(hu)_P^n\theta(R_f)_P^n\Delta t^n. \end{aligned} \tag{3.12}$$

Υποθέτοντας ότι $(R_f)^{n+1} \simeq (R_f)^*$ μπορούμε να γράψουμε την εξίσωση (3.12) ως

$$(hu)_P^{n+1} = \frac{(hu)_P^* - \theta g(hu)_P^n(R_f)_P^n\Delta t^n}{1 + (1-\theta)g(R_f)_P^*\Delta t^n}.$$

Με τον ίδιο τρόπο έχουμε

$$(hv)_P^{n+1} = \frac{(hv)_P^* - \theta g(hv)_P^n(R_f)_P^n\Delta t^n}{1 + (1-\theta)g(R_f)_P^*\Delta t^n}.$$

Η παράμετρος θ είναι παράμετρος εμεσότητας. Όταν $\theta = 0$ ο όρος της τριβής υπολογίζεται ολικώς έμεσα και οταν $\theta = 1$ υπολογίζεται ολικώς άμεσα. Η παραπάνω αντιμετώπιση είναι απευθείας συμβατή με την χρονική διακριτοποίηση Runge-Kutta.

Σύμφωνα με [44] και [33] η άμεση διακριτοποίηση των όρων της τριβής επιδρά με το CFL και δίνει ένα εναλλακτικό μέγεθος του χρονικού βήματος για τις δύο πρώτης και δεύτερης τάξης προσεγγίσεις για το αρχικό υπολογιστικό πλέγμα αλλιώς για να επιτευχθεί σταθερότητα πρέπει να γίνει επενεξέταση του υπολογιστικού πλέγματος. Και οι δύο περιπτώσεις προσφέρουν ευστάθεια αλλά με μεγάλο υπολογιστικό κόστος ειδικά κοντά σε στεγανά-υγρά μέτωπα που χαρακτηρίζονται από μικρά βάθη και ο όρος της τριβής κυριαρχεί της κλίσης του πυθμένα. Από την άλλη, η άμμεση διακριτοποίηση των όρων της τριβής δεν εξερτάται από το χρόνο άρα και δεν απαιτεί περαιτέρω περιορισμούς στο χρονικό βήμα πέρα από το CFL.

Κεφάλαιο 4

Σειριακός Κώδικας

4.1 Αρχικός Κώδικας Fortran 77

4.1.1 Περιγραφή

Ο αρχικώς κώδικας γράφτηκε από τους Ι. Νικολός και Α. Δελής το Δεκέμβριο του 2009, στο κώδικα αυτόν προστέθηκαν τμήματα όσον αφορά το αριθμητικό σχήμα από την Μ. Καζολέα στα πλαίσια της μεταπτυχιακής της εργασίας αρχικά και της διδακτορικής της διατριβής στην συνέχεια. Ο κώδικας γράφτηκε σε Fortran 77 και το σύνολο των συναρτήσεων και υπορουτίνων υπήρχαν σε ένα αρχείο. Η δέσμευση μνήμης ήταν στατική στην αρχή του προγράμματος και οι κοινές μεταβλητές όπως τα διανύσματα του πλέγματος και της λύσης περνούσαν στα υποπρογράμματα μέσο των common blocks που προσφέρει η Fortran 77.

4.1.2 Κατακερματισμός και Fortran 90

Αρχικά ο κώδικας κατακερματίστηκε σε αρχεία όπου το κάθε ένα περιέχει την κάθε υπορουτίνα ή συνάρτηση, το όνομα του αρχείου αποτελείται από το όνομα του υποπρογράμματος που περιέχει με την κατάληξη .f. Στη συνέχεια φτιάχτηκε Makefile ώστε να γίνεται οργανωμένα η μεταγλώττιση του κώδικα. Ο κατακερματισμός του κώδικα έγινε με το εργαλείο fsplit το οποίο είναι ανοικτού κώδικα.

`http://people.sc.fsu.edu/~jburkardt/c_src/f77split/f77split.html`

Στο επόμενο βήμα έγινε ο μετασχηματισμός του κώδικα από Fortran 77 σε Fortran 90. Ο μετασχηματισμός κρίθηκε απαραίτητος διότι η Fortran 90 προσφέρει δυναμική διαχείριση της μνήμης πράγμα σημαντικό για την απόδοση και οργάνωση του παράλληλου κώδικα στην συνέχεια. Ο μετασχηματισμός έγινε με το λογισμό ανοικτού κώδικα tof90.f90 όπου με κατάλληλη τροποποίηση έφερε το επιθυμητό αποτέλεσμα. Στην συνέχεια αναλύθηκε η χρήση της μνήμης από τον κώδικα και προστέθηκε η υπορουτίνα memalloc.f90 όπου εκεί γίνονται πλέον οι αρχικές δεσμεύσεις μνήμης ανάλογα με το πρόβλημα κάθε φορά. Τέλος τα αρχικά common blocks μετασχηματίστηκαν σε modules στο αρχείο modules.f90 όπου και οργανώθηκαν ανάλογα με την χρήση της κάθε μεταβλητής.

```
http://jblevins.org/mirror/amiller/to_f90.f90
```

Με την ολοκλήρωση του μετασχηματισμού έγιναν αρκετές δοκιμές σε σύγκριση με τον αρχικό κώδικα ώστε να μην υπάρχει κάποια αμφιβολία για τυχόν λογικό σφάλμα που θα οδηγούσε σε λάθος αποτελέσματα.

Κεφάλαιο 5

Παράλληλος Κώδικας

5.1 Γενική Περιγραφή

Το μοντέλο παραλληλοποίησης που επιλέχτηκε είναι το SPMD, Single Program Multiple Data με στατικό κατακερματισμό του υπολογιστικού πλέγματος. Το αρχικό υπολογιστικό πλέγματα κατακερματίζεται σε n_p μικρότερα πλέγματα όπου $i_0, ..., n_{p-1}$ ανεξάρτητες διεργασίες τρέχουν στο κάθε ένα από αυτά. Οι p διεργασίες αρχίζουν και τελειώνουν αυτόνομα με δικά τους η κάθε μια δεδομένα εισόδου και εξόδου. Οι επικοινωνία που απαιτεί η αριθμητική μέθοδος γίνεται σε κάθε χρονικό βήμα με την αποστολή και λήψη μηνυμάτων που έχουν να κάνουν με τις τιμές στα εσωτερικά σύνορα.

Το μοντέλο SPMD υπερέχει του μοντέλου Master-Slave, όπου ένας επεξεργαστής (Master) συγκεντρώνει τα δεδομένα και οργανώνει τις διεργασίες των υπολοίπων επεξεργαστών (Slave), καθώς επίσης χρησιμοποιείται και σαν δίαυλος επικοινωνίας με τα δεδομένα εισόδου και εξόδου. Ένα σημαντικό πλεονέκτημα του μοντέλου SPMD είναι ότι το $\theta_p = 1$ στην εξίσωση (1.2) πράγμα που σημαίνει ότι επιτυγχάνεται υψηλή απόδοση αυξάνοντας το n_p. Σε αντίθεση με το Master-Slave μοντέλο όπου το $\theta_p < 1$. Επίσης ένα εξίσου σημαντικό πλεονέκτημα είναι ότι με το SMPD μοντέλο γίνεται η μικρότερη δυνατή παρέμβαση στον αρχικό σειριακό κώδικα.

Τέλος για την υλοποίηση της παραλληλοποίησης επιλέχτηκε η αρχιτεκτονική κατανεμημένης μνήμης με την χρήση της βιβλιοθήκης του MPI, η οποία είναι απολύτως συμβατή και με της διαμοιραζόμενες μνήμης αρχιτεκτονικές πράγμα που κάνει τον κώδικα ευέλικτο στην χρήση.

5.2 Υπολογιστικό Πλέγμα

5.2.1 Γενικά

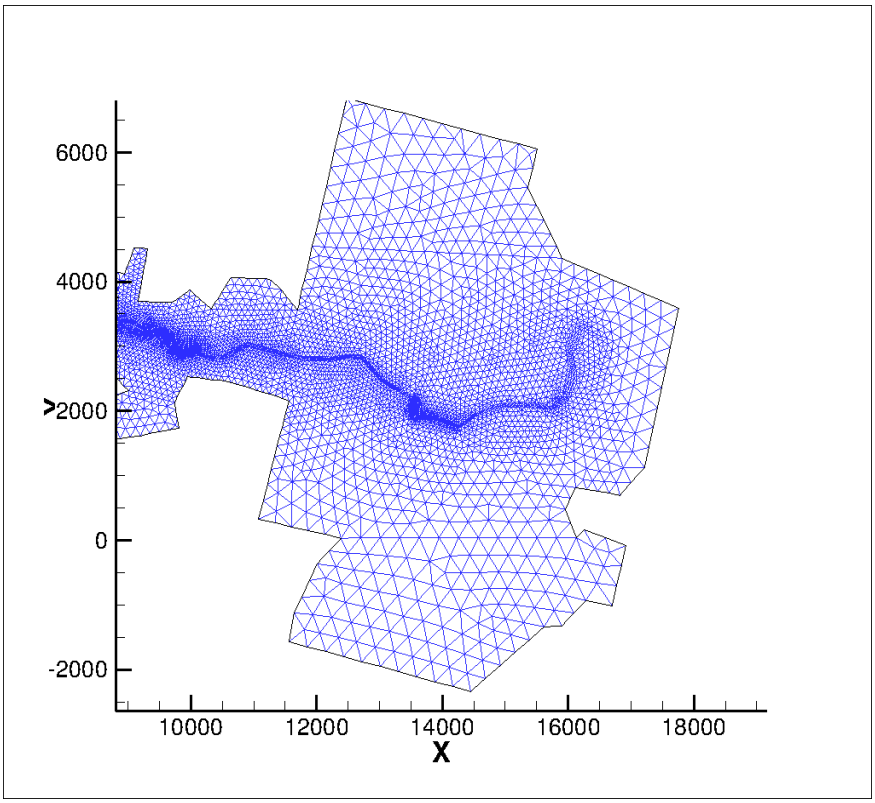
Το υπολογιστικό πλέγμα είναι μη δομημένο (unstructure) και έγινε βάση του τριγωνισμού Delaunay, μέσω του ανοικτού κώδικα λογισμικό Delaundo. Η μορφή του παραγόμενου αρχείου του υπολογιστικού πλέγματος είναι της μορφής delaundo όπου και ο κώδικας είναι σχεδιασμένος να δέχεται αυτή την μορφή σαν είσοδο του υπολογιστικού πλέγματος.

Η μορφή delaundo χωρίζει το αρχείο του υπολογιστικού πλέγματος σε τρία τμήματα. Το πρώτο τμήμα περιγράφει τα στοιχεία (elements) του πλέγματος βάση των κόμβων (nodes) που απαρτίζουν κάθε στοιχείο και τα γειτονικά σε αυτό στοιχεία. Επίσης περιγράφεται το είδος του στοιχείου, τρίγωνο στην περίπτωσή μας, καθώς επίσης και ο αύξων αριθμός του. Στο δεύτερο τμήμα περιγράφονται οι κόμβοι με τις x, y, z συντεταγμένους και τον αύξων αριθμό τους. Τέλος στο τρίτο τμήμα περιγράφονται στα σύνορα του πλέγματος. Το κάθε ένα σύνορο περιγράφεται από το πλήθος των στοιχείων που το αποτελούν, όπου για κάθε ένα στοιχείο του συνόρου δίνει τους κόμβους που είναι πάνω στο σύνορο και το είδος του συνόρου αυτού.

5.2.2 Κατακερματισμός

Ο κατακερματισμός του υπολογιστικού πλέγματος είναι το σημαντικότερο κομμάτι στο μοντέλο SPMD μιας και καθορίζει το υπολογιστικό φορτίου του κάθε επεξεργαστή καθώς επίσης και τον ομοιόμορφο καταμερισμός εργασίας (load balance) των επεξεργαστών. Στα πλήρως μη δομημένα πλέγματα ο κατακερματισμός βάση των καρτεσιανών συντεταγμένων [55] δεν έχει τα επιθυμητά αποτελέσματα όσον αφορά την ισορροπία και τον καταμερισμό της επεξεργαστικής ισχύς. Ο λόγος είναι ότι σε κάποια σημεία το πλέγμα μπορεί να είναι αρκετά πυκνό ενώ σε άλλα σημεία αρκετά ποιο αραιό, Σχήμα (5.1).

Για το λόγο αυτό χρησιμοποιήθηκε η βιβλιοθήκη του Metis [34], η οποία εφαρμόζοντας διάφορα εργαλεία σχετικά με τον κατακερματισμό γράφων εξασφαλίζει ένα ισορροπημένο καταμερισμό με την λιγότερη δυνατή επικοινωνία μεταξύ των υποπλεγμάτων με σχετικά εύκολο τρόπο. Η βιβλιοθήκη του Metis υποστηρίζει κατακερματισμό βασιζόμενο είτε στις κορυφές είτε στα κελιά του πλέγματος. Το Metis θεωρεί το πλέγμα των κορυφών συνδεδεμένο με ακμές μεταξύ των κελιών ως πλέγμα (nodal



Σχήμα 5.1: *Μη δομημένο Πλέγμα*

mesh) και το πλέγμα των κέντρων των κελιών συνδεδεμένων με το κέντρο των γειτονικών κελιών το οποίο είναι το dual mesh. Στην δικής μας περίπτωση το πλέγμα που δόθηκε αρχικά στο Metis είναι τύπου nodal. Το Metis με την επεξεργασία παράγει δύο αρχεία της μορφής $< filename > .npart.x$ και $< filename > .epart.x$ όπου $x = 2, ..., p$ με p το πλήθος των πλεγμάτων που θέλουμε να διαμεριστεί το αρχικό. Το npart αρχείο περιέχει σε μια στήλη την πληροφορία για το σε ποιο υποπλέγμα ανήκει ο κάθε κόμβος του αρχικού πλέγματος όπου η κάθε γραμμή του αρχείου npart αντιστοιχεί στον αύξων αριθμό του αρχικού. Ομοίως το αρχείο epart για τα στοιχεία του αρχικού πλέγματος.

5.2.3 Δημιουργία υποπλεγμάτων

Βασιζόμενοι στην λογική του SMPD πρέπει ο κώδικας να είναι ο ίδιος σε κάθε επεξεργαστή με τις λιγότερες δυνατές αλλαγές σε σχέση με τον αρχικό. Για το λόγο αυτό η μορφή του υπολογιστικού πλέγματος του κάθε επεξεργαστή πρέπει να είναι σύμ-

φωνη με την αρχική. Για να επιτευχθεί αυτό φτιάχτηκε ένας ανεξάρτητος κώδικας (post processing), το Parmesh,(Appendix A), σε ansi C το οποίο δημιουργεί αρχεία της μορφής $mesh - x.dpl$ όπου $x = 0, 1, 2, ..., p$ με p το πλήθος των διαθέσιμων επεξεργαστών. Το αρχεία αυτά έχουν ακριβώς την ίδια δομή με το αρχικό, δηλαδή μορφή delando, με λιγότερες εγγραφές.

Το Parmesh διαβάζει το αρχικό αρχείο πλέγματος delaundo καθώς επίσης και τα αρχεία που παράγει το Metis. Το κάθε στοιχείο (element) του πλέγματος αποθηκεύεται στην προσωρινή μνήμη μεσώ ενός πίνακα τύπου $element\_t$ που περιγράφει η παρακάτω δομή.

```
1 typedef struct element {
2         int tp,nds[3],nbrs[3],id;
3         int dnbrs[3];
4         int ndbc[2],bc;
5         int dom,ddom,did;
6 } element_t;
```

Στη γραμμή (2) είναι οι μεταβλητές που αποθηκεύονται τα δεδομένα που διαβάζονται από το αρχείο του αρχικού υπολογιστικού πλέγματος, ενώ οι υπόλοιπες γραμμές είναι διάφορες βοηθητικές μεταβλητές για την υλοποίηση. Ομοίως ο κάθε κόμβος του πλέγματος αποθηκεύεται στην προσωρινή μνήμη μέσω ενός πίνακα τύπου $node\_t$ που περιγράφει η παρακάτω δομή, όπου και εδώ οι γραμμές (2) και (3) είναι μεταβλητές για τα δεδομένα του αρχικού πλέγματος και οι υπόλοιπες είναι βοηθητικές μεταβλητές για την υλοποίηση.

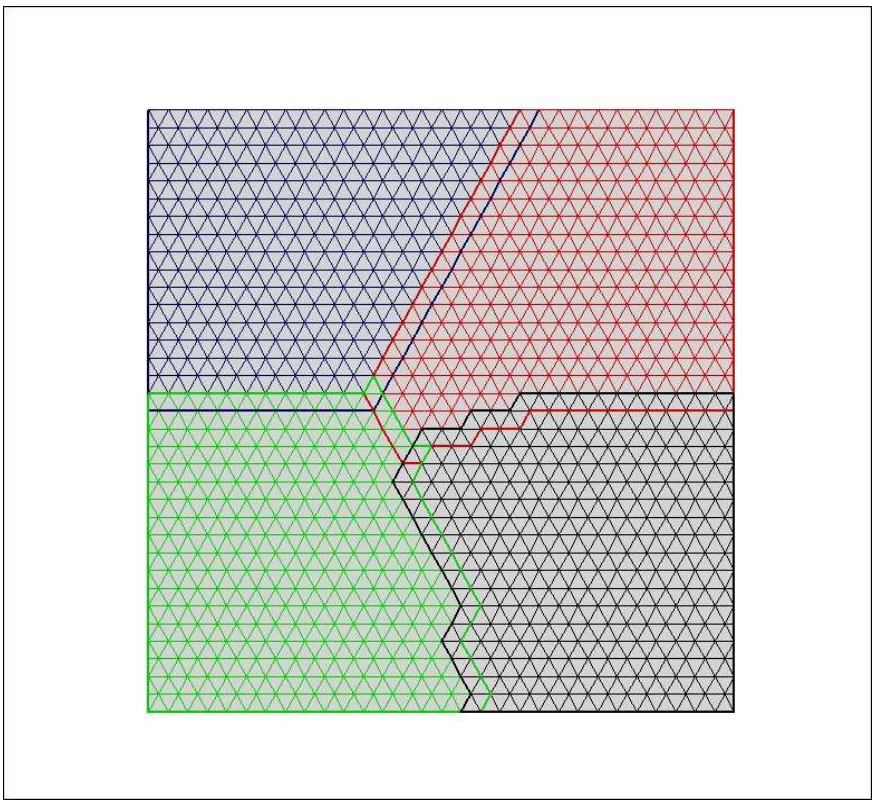
```
1 typedef struct node {
2         float x,y,z,icond[3];
3         int id;
4         int did,dom,ddom,dm;
5 } node_t;
```

Τέλος το τρίτο μέρος του πλέγματος με την πληροφορία για τα σύνορα αποθηκεύεται στην προσωρινή μνήμη ενός πίνακα τύπου $bc\_element\_t$ που περιγράφεται από την παρακάτω δομή.

```
1 typedef struct bc_element {
2         int nd1, nd2;
3         int id,tp;
```

```
4 } bc_element_t;
```

Μετά την εισαγωγή των δεδομένων από το αρχικό αρχείο του υπολογιστικού πλέγματος καθώς επίσης και της πληροφορίας του METIS οι κόμβοι και τα στοιχεία έχουν χωριστεί στους n κομμάτια όπου $n = 0, 1, ...n_p - 1$. Η πληροφορία αυτή έχει καταχωρηθεί στην μεταβλητή **dom**. Το κάθε τρίγωνο του πλέγματος αποτελείται από τρεις κόμβους, σύμφωνα με το METIS ένα τρίγωνο ανήκει στον **ι** επεξεργαστή αν δύο από τους κόμβους του τουλάχιστον ανήκουν στον επεξεργαστή αυτόν. Το παραπάνω δημιουργεί στο κάθε τμήμα του υπολογιστικού πλέγματος *δόντια* τα οποία δυσκολεύουν τον ορισμό των εσωτερικών συνοριακών συνθηκών που θα δούμε παρακάτω. Για το λόγο αυτό ο κώδικας δημιουργεί ευθύγραμμα εσωτερικά σύνορα τουλάχιστον δύο τριγώνων, Σχήμα (5.2), αλλάζοντας προσωρινά το τμήμα που ανήκει το κάθε τρίγωνο, η αλλαγή αυτή αποθηκεύεται στην μεταβλητή **ddom**. Έπειτα διατρέχουμε όλους τους κόμβους και κάνουμε τις αντίστοιχες αλλαγές και σε αυτούς.



Σχήμα 5.2: *Κατακερματισμένο Πλέγμα*

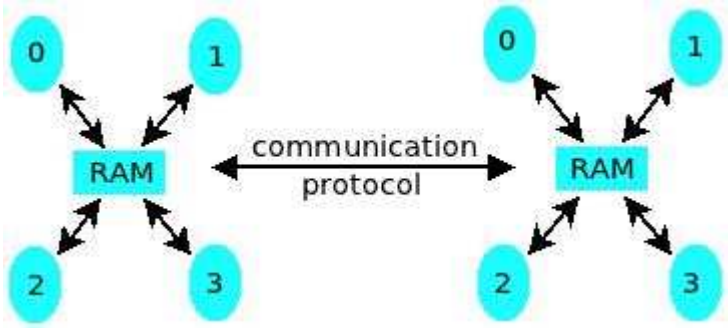
Η αριθμητική μέθοδος όπως έχουμε ήδη δείξει είναι βασισμένη στους κόμβους

του υπολογιστικού πλέγματος. Για το λόγο αυτό η αρχική πληροφορία του METIS για την προέλευση του κάθε κόμβου είναι πολύτιμη. Βάση αυτής της πληροφορίας ορίζονται οι κόμβοι που θα χαρακτηριστούν ως εσωτερικοί συνοριακοί και πάνω σε αυτούς θα γίνει σε κάθε χρονικό βήμα η ανταλλαγή πληροφορίας που θα δούμε παρακάτω. Η μέθοδος, όπως έχουμε δει, για το κάθε ένα νέο υπολογιστικό στοιχείο που δημιουργεί παίρνει πληροφορία από τους γειτονικούς κόμβους των γειτονικών τριγώνων, το βάθος αυτής της πληροφορίας είναι ένα τρίγωνο. Άρα το κάθε κομμάτι του αρχικού υπολογιστικού πλέγματος αρκεί να μπαίνει μέσα στο γειτονικό πλέγμα ένα τρίγωνο, Σχήμα (5.2). Αυτό σημαίνει ότι για το κάθε κομμάτι έχουμε το υπολογιστικό σύνορο, δηλαδή μέχρι που λύνει ο κάθε επεξεργαστής, και το πραγματικό εσωτερικό σύνορο του οποίου την πληροφορία χρειάζεται ο κάθε επεξεργαστής για να λύσει μέχρι το υπολογιστικό σύνορο. Το πραγματικό εσωτερικό σύνορο είναι υπολογιστικό σύνορο του γειτονικού επεξεργαστή. Βάση λοιπόν των παραπάνω ο κάθε επεξεργαστής πρέπει να ενημερωθεί για το ποιου κόμβοι είναι προς ανταλλαγή και με ποιόν επεξεργαστή. Ο κώδικας λοιπόν που παράγει τα κατακερματισμένα υπολογιστικά πλέγματα δημιουργεί επίσης και n_p αρχεία με όνομα scom-x.dpl, όπου $x = 0, ..., n_p - 1$. Ο κάθε ένας επεξεργαστής στην αρχή του προγράμματος μετά την εισαγωγή του υπολογιστικού πλέγματος διαβάζει το αντίστοιχο αρχείο και καθορίζει ποιους κόμβους θα στείλει και σε ποιόν. Σε αυτό το σημείο γίνεται επικοινωνία όλων των επεξεργαστών μεταξύ τους που καθορίζουν το μέγεθος, τον τρόπο επικοινωνίας και κάνουν τις αντιστοιχίσεις των κόμβων που ανταλλάσσουν. Η πληροφορία που στέλνεται και λαμβάνεται κάθε φορά αποθηκεύεται σε δύο διανύσματα προσωρινής μνήμης αντίστοιχα. Η διάσταση αυτών των διανυσμάτων καθορίζεται από το μεγαλύτερο μύνημα. Επείδη το υπολογιστικό πλέγμα είναι μη δομημένο το μέγεθος των μυνημάτων προς και από κάθε επεξεργαστή δεν είναι πάντα το ίδιο. Επιλέγεται το μεγαλύτερο ώστε να γίνει μια φορά δυναμική δέσμευση μνήμης. Η δυναμική δέσμευση μνήμης πλεονεκτεί έναντι της στατικής αλλά δεν παύει να είναι αργό κομμάτι του κώδικα μιας και γίνεται κλίση συστήματος (system call). Επίσης πολλαπλές δεσμεύσεις και αποδευσμέσεις μέσα στον κώδικα προκαλλούν κατακερματισμό της μνήμης (fragmentation) που είναι και αυτό καταστροφικό από θέμα απόδοσης, (ρουτίνα inpbnd.f90, Appendix B).

5.3 Τοπολογία επεξεργαστών

5.3.1 Γενικά

Η τοπολογία των επεξεργαστών σε ένα παράλληλο πρόβλημα αποτελεί σημαντικό παράγοντα τόσο για την επιτυχή υλοποίηση όσο για την απόδοση του κώδικα [29]. Η τοπολογία θα μπορούσε να χωριστεί σε δύο κατηγορίες, στην πραγματική τοπολογία των επεξεργαστών όπως την δίνει η αρχιτεκτονική και στην τοπολογία που απαιτεί η φύση του προβλήματος το οποίο λύνουμε παράλληλα. Η τεχνολογία των υπολογιστών σήμερα μας δίνει μια μεγάλη γκάμα αρχιτεκτονικών, με ιδιαίτερα διαδεδομένα τα πολύ-πύρηνα συστήματα όπου ένας ή και παραπάνω πολύ-πύρηνοι επεξεργαστές είναι στην ίδια μητρική κάρτα με συμμετρική (UMA) ή με μη συμμετρική (NUMA) πρόσβαση στην μνήμη, Σχήμα (5.3). Πολλά τέτοια συστήματα κατάλληλα συνδεμένα μεταξύ τους μας δίνουν την δυνατότητα για μεγάλο αριθμό διαθέσιμων επεξεργαστών.

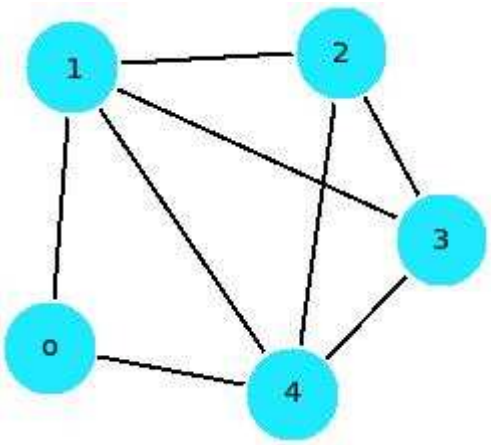


Σχήμα 5.3: *Συμμετρική και μη συμμετρική πρόσβαση στην μνήμη*

5.3.2 Τοπολογία προβλήματος

Το υπολογιστικό πλέγμα του προβλήματος, πάνω στο οποίο παραλληλοποιήθηκε ο κώδικας, είναι μη δομημένο (unstructure). Αυτό σημαίνει ότι και ο κατακερματισμός δεν ακολουθεί κάποια σειρά αλλά γίνεται με μόνο κριτήριο την όσον το δυνατόν καλύτερη κατανομή του υπολογιστικού φορτίου. Ο κάθε επεξεργαστής λοιπόν μπορεί να επικοινωνεί με τον οποιοδήποτε καθώς επίσης και δεν έχουν όλοι οι επεξεργαστές τον ίδιο αριθμό γειτόνων για επικοινωνία. Τον καθορισμό της επικοινωνίας, όπως έχει ήδη αναφερθεί, τον κανονίζει η ρουτίνα inpbnd.f90, μετά

το διάβασμα των αρχείων δημιουργείται ένας δυαδικός πίνακας ο **topo** διάστασης $n_p \times n_p$ όπου ο αύξων αριθμός της κάθε στήλης αντιστοιχεί με το id του επεξεργαστή που στέλνει και ο αύξων αριθμός της κάθε γραμμής αντιστοιχεί με το id του επεξεργαστή που παραλαμβάνει. Άρα όταν στο στοιχείο (i, j) του πίνακα είναι $(i, j) = 1$ τότε ο επεξεργαστής j στέλνει πληροφορία στον επεξεργαστή i και ο i την λαμβάνει από τον j. Ο πίνακας αυτός είναι γνωστός σε όλους τους επεξεργαστές. Αντίστοιχα δημιουργούνται δύο πίνακες οι **snd_nds** και **rcv_nds** με διάσταση $Nmax \times n_p$ όπου $Nmax$ ο μέγιστος αριθμός των κόμβων προς αποστολή και παραλαβή αντίστοιχα. Ο κάθε ένας επεξεργαστής έχει τους δικούς του πίνακες όπου η κάθε στήλη αντιστοιχή σε αύξων αριθμό επεξεργαστή. Άρα ο κάθε ένας ξέρει τους κόμβους που στέλνει και σε ποιόν όπως επίσης και τους κόμβους που λαμβάνει και από ποιόν. Από το παραπάνω προκύπτει ότι η προεπιλεγμένη τοπολογία που δημιουργεί η βιβλιοθήκη του MPI κατά την εκτέλεση του προγράμματος δεν είναι αρκετή αφού οι επεξεργαστές είναι στην σειρά ο ένας μετά των άλλων. Επίσης δεν εξυπηρετεί ούτε η δημιουργία καρτεσιανής τοπολογίας αφού δεν έχει νόημα το δεξιά,αριστερά, πάνω και κάτω. Αυτό που εύκολα αντιλαμβάνεται κανείς είναι ότι η επικοινωνία των επεξεργαστών είναι τυχαία κάθε φορά, Σχήμα (5.4). Από το Σχήμα (5.4) βλέπουμε ότι η επικοι-



Σχήμα 5.4: *Τυχαία επικοινωνία μεταξύ των επεξεργαστών*

νωνία αποτελεί ένα κατευθυνόμενο γράφο, άρα θα πρέπει στον κώδικα να ορίσουμε τοπολογία γράφου. Ο κατευθυνόμενος γράφος ορίζεται από το ζεύγος $D = (V, A)$, όπου V είναι οι κορυφές του γράφου και A οι ακμές που συνδέουν τις κορυφές. Στο κώδικα οι κορυφές είναι οι επεξεργαστές και οι ακμές η τυχόν επικοινωνία μεταξύ τους. Αυτή η πληροφορία υπάρχει στον πίνακα με την επικοινωνία που δημιουργεί η

ρουτίνα inpbnd.f90. Οπότε καλούμε την ρουτίνα graph_topo (Appendix C) η οποία δημιουργεί μια καινούργια τοπολογία γράφου βάση της οποίας γίνεται η αποστολή και παραλαβή των μηνυμάτων μεταξύ των επεξεργαστών. Εδώ αξίζει να τονιστεί ότι για την κατασκευή του γράφου υποθέσαμε ότι όλες οι ακμές έχουν το ίδιο βάρος, δηλαδή ότι το κόστος επικοινωνίας είναι το ίδιο για όλους τους επεξεργαστές. Αυτή η υπόθεση κάνει την υλοποίηση απλούστερη με κόστος μικρό μεν αλλά υπαρκτό δε στην απόδοση. Αυτό που θα έπρεπε να γίνει [28] είναι να οριστούν βάρη στις ακμές του γράφου, τα βάρη αυτά καθορίζονται από την αρχιτεκτονική πλέον. Παράδειγμα στο Σχήμα (5.3) η επικοινωνία που γίνεται μέσω του πρωτοκόλλου είναι ποιο αργή σε σχέση με την επικοινωνία μέσα στο κάθε κομμάτι.

5.4 Επικοινωνία στους εσωτερικούς συνοριακούς κόμβους

Έχοντας πλέον ορίσει την τοπολογία που περιγράφει την επικοινωνία μεταξύ των επεξεργαστών περνάμε στην δημιουργία της ρουτίνας εκείνης όπου σε κάθε χρονικό βήμα θα καλείται και θα ενημερώνει τους κόμβους που αποτελούν εσωτερικό σύνορο για τον κάθε ένα επεξεργαστή. Η επικοινωνία σε αυτό το σημείο είναι point to point ο κάθε επεξεργαστής στέλνει ένα συγκεκριμένο μήνυμα σε κάποιον άλλο και αντίστροφα. Ο σκοπός είναι να αποστέλλονται ταυτόχρονα όσον το δυνατόν περισσότερα μηνύματα χωρίς όμως να διατρέχουμε κίνδυνο για dead lock. Οι επιλογές που έχουμε για την point to point επικοινωνία είναι η blocking και η non-blocking επικοινωνία. Στην blocking επικοινωνία για να προχωρήσει ο κώδικας πρέπει για κάθε μια αποστολή (Send) πρέπει να έχει ολοκληρωθεί μια παραλαβή (Receive), αντίθετα στην non-blocking ο κώδικας προχωράει και ας υπάρχουν σε εκκρεμότητα παραλαβές ή αποστολές. Στην δεύτερη περίπτωση υπάρχει η δυνατότητα για συγχρονισμό των διεργασιών στα σημεία που θεωρούμε σκόπιμο.

Η non-blocking επικοινωνία σε ορισμένα προβλήματα επιτυγχάνει καλύτερη απόδοση, στο συγκεκριμένο όμως πρόβλημα η αποστολή και παραλαβή των δεδομένων είναι κρίσιμη διότι η αριθμητική μέθοδος απαιτεί καινούργια πληροφορία σε κάθε χρονικό βήμα για τους εσωτερικούς συνοριακούς κόμβους, άρα ο συγχρονισμός είναι απαραίτητος. Έχοντας λάβει υπόψη την καινούργια τοπολογία ο κάθε επεξεργαστής είναι ξεκάθαρο με ποιον επικοινωνεί οπότε επιλέγουμε ταυτόχρονη blocking επικοινωνία χωρίς να υπάρχει κίνδυνος για dead lock. Η ρουτίνα αυτή λέγεται bc_com.f90,

Appendix D και καλείται μια φορά στην αρχή και μετά σε κάθε χρονικό βήμα. Η ρουτίνα αυτή χρησιμοποιεί την πληροφορία που παράγει

5.5 Χρονικό βήμα δt^n

Όπως έχουμε δεί στην εξίσωση (3.7) το χρονικό βήμα υπολογίζεται δυναμικά βάση του CFL. Όταν το υπολογιστικό πλέγμα είναι ομοιόμορφο και έχει δομημένη μορφή τότε σε κάθε κομμάτι του έχουμε το ίδιο δt^n. Αυτό όμως δεν συμβαίνει σε πλήρως μη δομημένα πλέγματα, οπότε ο κάθε επεξεργαστής έχει το δικό του. Στην περίπτωση αυτή η υπολογιστική λύση του κάθε κομματιού προηγείται ή καθυστερεί έναντι των υπολοίπων. Επειδή σε κάθε χρονικό βήμα υπάρχει εξάρτηση μεταξύ των κομματιών πρέπει να συγχρονιστεί η λύση. Ο συγχρονισμός μπορεί να γίνει με δύο τρόπους, ο πρώτος τρόπος είναι να έχουμε αναμονή μέχρι να φτάσουν όλοι την ίδια χρονικη στιγμή πριν την επικοινωνία ενώ ο δεύτερος τρόπος είναι να υπολογίσει το δt^n ο κάθε επεξεργαστής και να γίνει μια προς όλους επικοινωνία με σκοπό να επικρατήσει το μικρότερο σε όλα τα κομμάτια.

```
!*** Collect all the dt1 and reduce the minimum one at dt2,
call mpi_allreduce(dt1,dt2,1,MPI_DOUBLE_PRECISION, &
MPI_MIN,MPI_COMM_WORLD, ierr)
```

5.6 Υγρό - στεγανό μέτωπο

Στο μοντέλο υπάρχει μια παράμετρο (threshold) που καθορίζει από ποιο σημείο και μέτα ένας κόμβος θα θεωρείται στεγανός. Η παράμετρος αυτή μπορεί να καθοριστεί από τον αρχείο που αρχικοποιεί το μοντέλο και να θεωρηθεί στην συνέχεια σταθερή ή να υπολογιστεί στην αρχή του κώδικα ανάλογα με το υπολογιστικό πλέγμα και τις αρχικές συνθήκες. Στην δεύτερη περίπτωση ακολοθείται μια διαδικασία όπως η παραπάνω ώστε να επικρατήσει το μικρότερο σε όλα τα κομμάτια.

Κεφάλαιο 6

Αριθμητικά αποτελέσματα

6.1 Γενικά

Σε αυτή την ενότητα παρουσιάζονται τρία αριθμητικά προβλήματα στα οποία δοκιμάστηκε ο παράλληλος κώδικας για την ορθότητα των αποτελεσμάτων καθώς και την απόδοσή του. Ιδιαίτερη έμφαση δόθηκε στην δοκιμή διαφόρων υπολογιστικών πλεγμάτων τόσο με δομημένα χαρακτηριστικά όσο και πλήρως μη δομημένα.

6.2 Thacker's axisymmetric

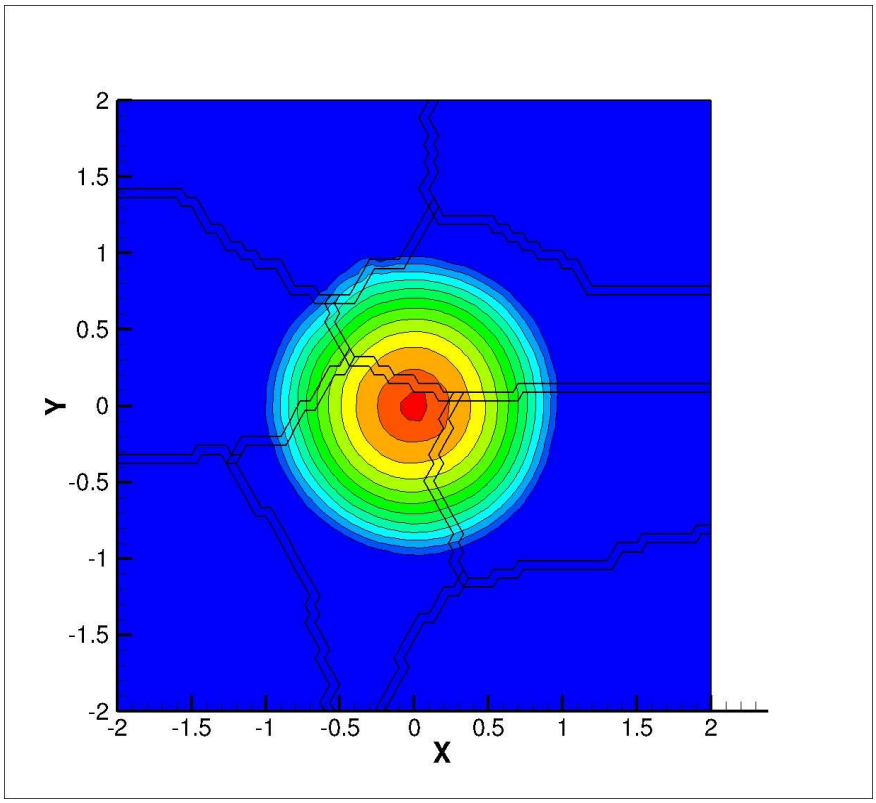
Αρκετές αναλυτικές λύσεις υπάρχουν για δύο διαστάσεων NSWE με ελεύθερο και κινούμενο σύνορο, εμπεριέχοντας και φαινόμενα υγρού-στεγανού μετώπου. Η δύο διαστάσεων αναλυτική λύση για NSWE που δοκιμάζουμε εδώ είναι το αξισυμμετρικό πρόβλημα του Thacker [49]. Το πρόβλημα αυτό έχει χρεσημοποιηθεί σε αρκετές επιστημονικές εργασίες με σκοπό να δοκιμαστούν διάφορα αριθμητικά μοντέλα [31, 39, 22, 18, 42, 16, 47, 2, 12, 41, 46, 45].

Στο πρόβλημα αυτό η αναλυτική λύση είναι:

$$h(x,y,t) = h_0 \left[\frac{(1-A^2)^{1/2}}{1-A\cos(\omega t)} - 1 - \frac{r^2}{a^2}\left(\frac{(1-A^2)^{1/2}}{(1-A\cos(\omega t))^2} - 1 \right) \right]$$

$$\mathbf{u} = \left[\frac{1}{1 - A\cos(\omega t)}\left(\frac{1}{2}\omega x A \sin(\omega t)\right), \right.$$
$$\left. \frac{1}{1 - A\cos(\omega t)}\left(\frac{1}{2}\omega y A \sin(\omega t)\right)\right]^{\mathrm{T}},$$

όπου η συχνότητα ω δίνεται από $\omega = \sqrt{8gh_0/a^2}$, r είναι η απόσταση από το κέντρο, r_0 είναι η αρχική απόσταση από το κέντρο$A = (a^2 - r_0^2)/(a^2 + r_0^2)$. Η αναλυτική λύση στο $t = 0$ χρησιμοποιείται σαν αρχική κατάσταση. Οι τιμές που χρησιμοποιούνται για τα αριθμητικά test είναι $a = 1$, $r_0 = 0.8m$ και $h_0 = 0.1m$. Η λύση είναι περιοδική με $T = 2\pi/\omega$, στην πρώτη μισή περίοδο ($t \in [0, T/2]$) το νερό κινείται προς τα έξω ενώ την δεύτερη μισή το νερό μαζεύει προς τα μέσα.



Σχήμα 6.1: *Thacker's Axisymmetric*

Στο Σχήμα (6.1) βλέπουμε την σύγκριση της αναλυτικής λύσης (contours με γραμμές) με την υπολογιστική (flood contours) για υπολογιστικό πλέγμα με αριθμό κόμβων $n_p = 4305$. Το υπολογιστικό πλέγμα χωρίστηκε σε οκτώ επεξεργαστές.

CPUs	1	2	4	6	8	12	36	48	60	72	84
Time (sec)	3327	1689	783	591	433	383	109	78	71	58	47

Πίνακας 6.1: *Υπολογιστικό πλέγμα 185493 κόμβων*

CPUs	1	2	4	8	12	36	48	60	72	92
Time (sec)	-	-	9761	4015	2455	733	521	459	353	268

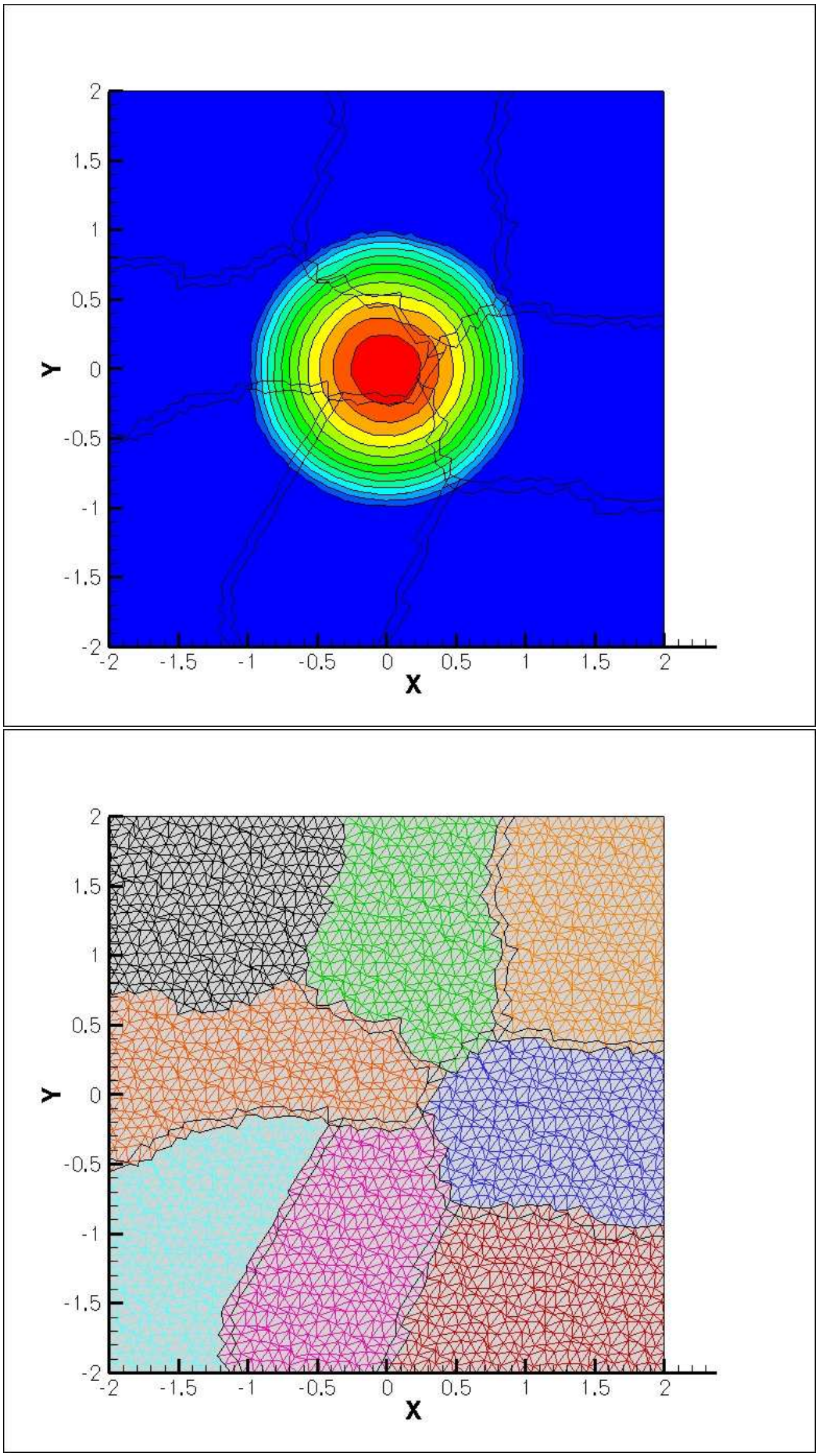
Πίνακας 6.2: *Υπολογιστικό πλέγμα 1156732 κόμβων*

6.3 Διάφορα υπολογιστικά πλέγματα

Σε αυτή την ενότητα παρουσιάζονται δοκιμές που έγιναν με διαφορετικά υπολογιστικά πλέγματα. Στον πίνακα (6.1) βλέπουμε το χρόνο εκτέλεσης, σε δευτερόλεπτα, σύμφωνα με το πλήθος των επεξεργαστών για ένα υπολογιστικό πλέγμα με 185493 κόμβους για χρόνο $t = 4T + T/2$ όπου T η περίοδος. Αντίστοιχα στον πίνακα (6.2) βλέπουμε το χρόνο εκτέλεσης με 1156732 κόμβους. Τέλος στο Σχήμα (6.2) βλέπουμε την παράλληλη υπολογιστική λύση (flood contours) σε σύγκριση με την σειριακή λύση (line contours) για distorted υπολογιστικό πλέγμα.

6.4 2D πρόβλημα μονήρους κύματος πάνω σε κωνικό νησί

Τέτοιου είδους προβλήματα είχαν μεγάλο ενδιαφέρον ειδικά την δεκαετία του 90 λόγο των διαφόρων γεγονότων όπως tsunamis και διάφορα άλλα φαινόμενα κατακλυσμού μικρών νησιών. Με σκοπό την συλλογή τέτοιου είδους αποτελεσμάτων, τόσο υπολογιστικών όσο πειραματικών, μια σειρά από μεγάλης κλίμακας πειράματα έγιναν στο εργαστήριο μηχανικής υδάτων του στρατού των Ηνωμένων Πολιτειών της Αμερικής, στις εργασίες [11, 56] γίνεται εκτενής αναφορά. Το υπολογιστικό πλέγμα έχει διαστάσεις $30 \times 25m$ με ένα κωνικό νησί κοντά στο κέντρο. Η βάση του νησιού έχει διάμετρο 7.2μ και στην κορυφή 2.2μ, ενώ το ύψος του είναι 0.625μ. Το πρόβλημα αυτό έχει χρησιμοποιηθεί σε πολλές εργασίες [38, 50, 51, 39, 9, 30, 20] με σκοπό να δοκιμαστεί η συμπεριφορά διαφόρων υπολογιστικών μεθόδων. Ο αρχικός κώδι-

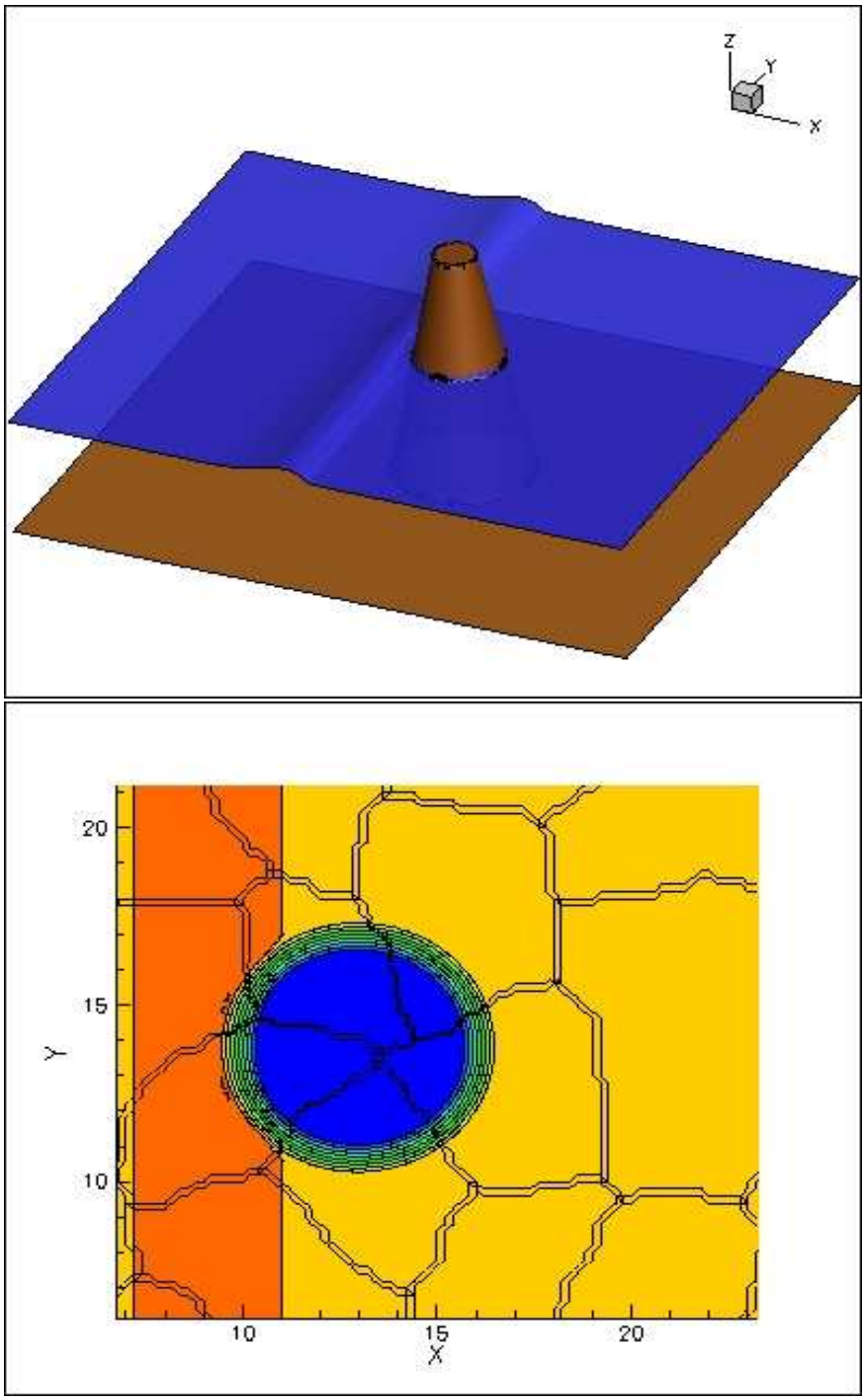


Σχήμα 6.2: *Thacker's Axisymmetric distorted mesh*

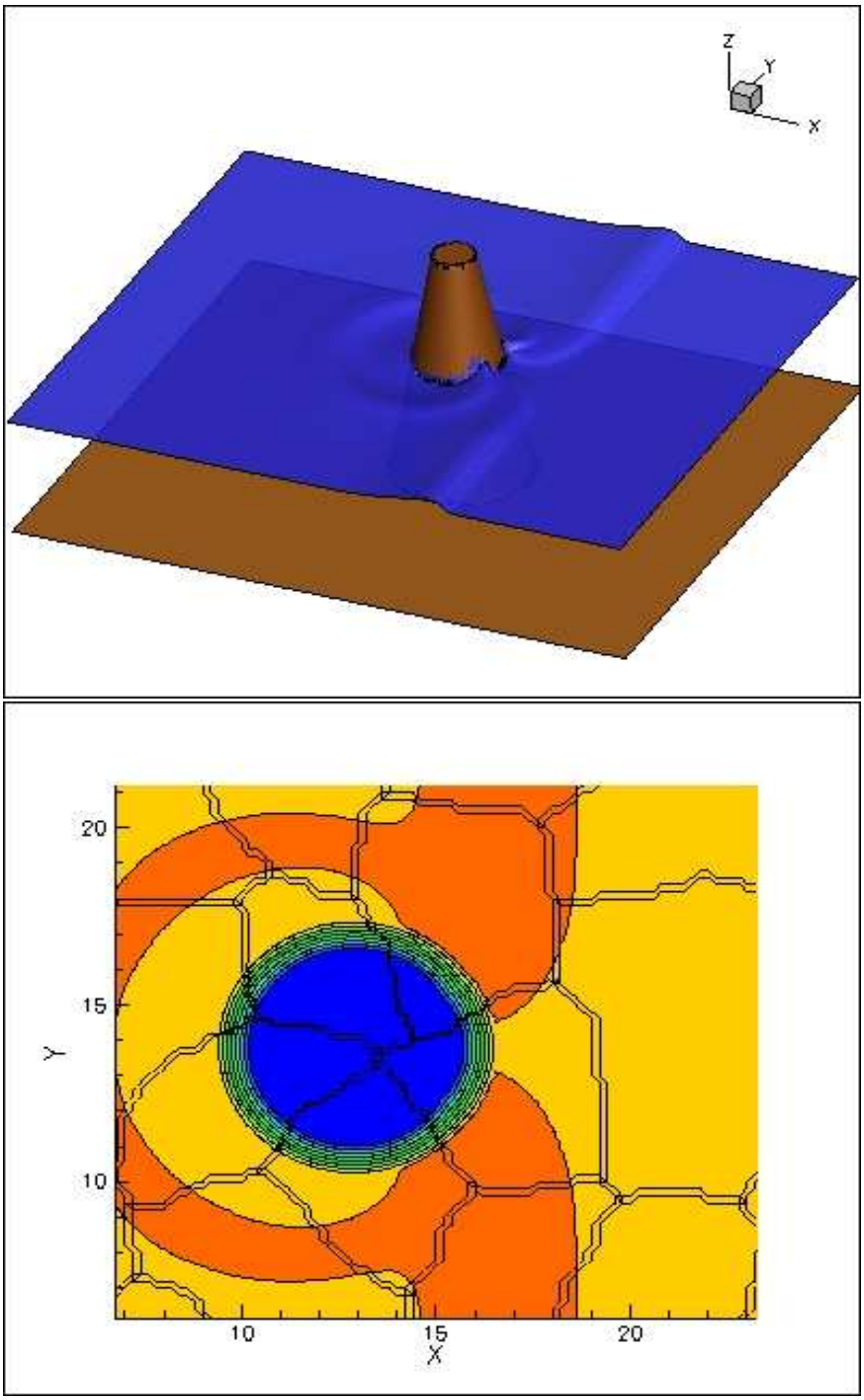
CPUs	1	2	4	8	12	36	48
Time (sec)	415	233	116	60	43	14	11

Πίνακας 6.3: *Υπολογιστικό πλέγμα 52191 κόμβων*

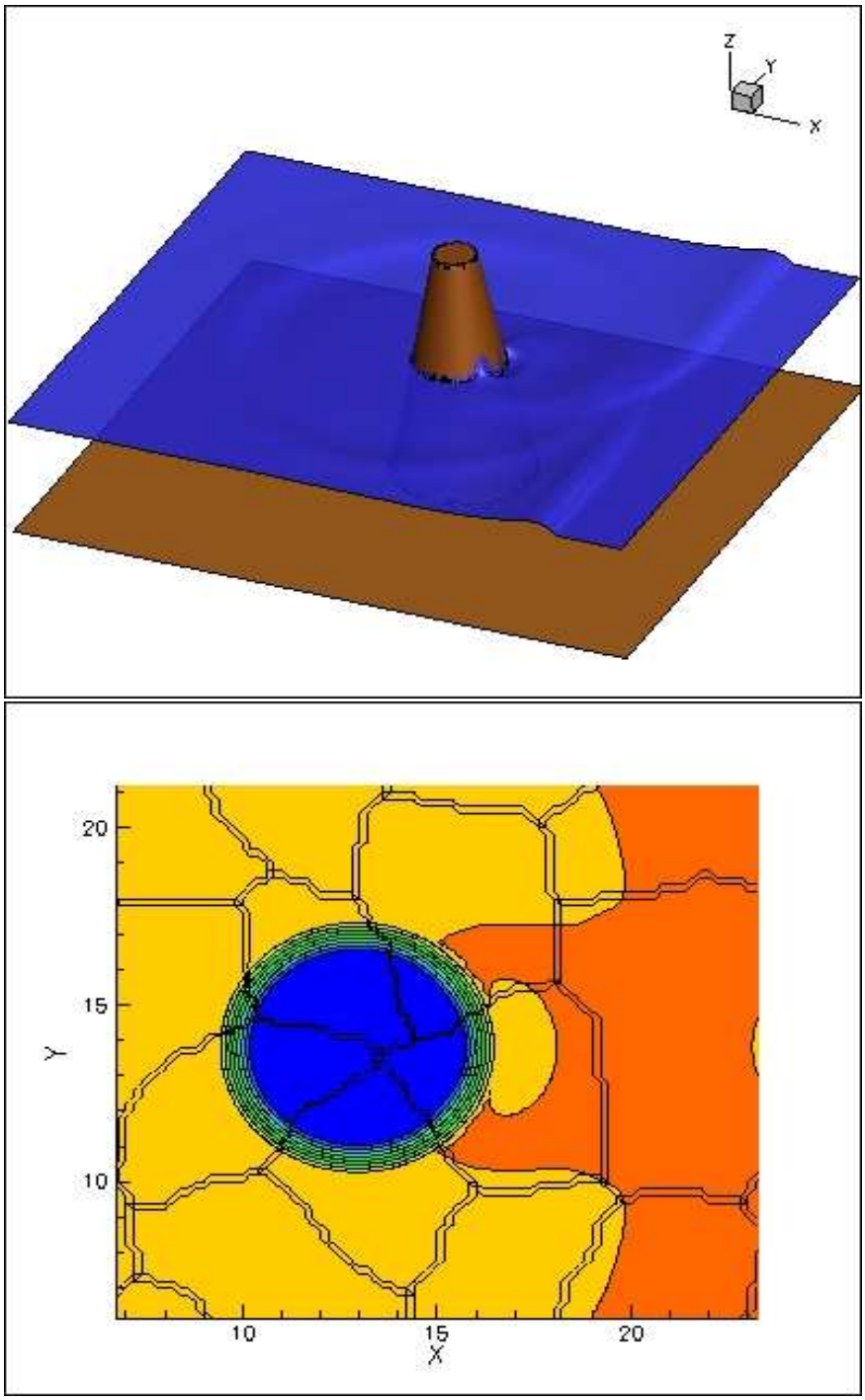
κας σύμφωνα με τις εργασίες [45] έχει ελεγχθεί ως προς την ακρίβεια της μεθόδου και της ορθότητας των αποτελεσμάτων, βάση αυτού γίνεται στο Σχήμα (6.3,6.4,6.5) σύγκριση μεταξύ των αποτελεσμάτων του σειριακού και του παράλληλου κώδικα για τα ίδια δεδομένα εισόδου την ίδια χρονική στιγμή. Στο πίνακα (6.3) βλέπουμε τον χρόνο εκτέλεσης,σε δευτερόλεπτα, με 52191 κόμβους.



Σχήμα 6.3: *Στιγμιότυπα από την λύση ενός μονήρους κύματος πάνω σε κωνικό υησί (αριστερά) και contour (δεξιά) με γραμμές για την λύση που δίνει ο αρχικός σειριακός κώδικας και flood που δίνει ο παράλληλος κώδικας για 24 επεξεργαστές τη χρονική στιγμή* $t = 4$



Σχήμα 6.4: *Στιγμιότυπα από την λύση ενός μονήρους κύματος πάνω σε κωνικό νησί (αριστερά) και contour (δεξιά) με γραμμές για την λύση που δίνει ο αρχικός σειριακός κώδικας και flood που δίνει ο παράλληλος κώδικας για 24 επεξεργαστές τη χρονική στιγμή* $t = 9$



Σχήμα 6.5: *Στιγμιότυπα από την λύση ενός μονήρους κύματος πάνω σε κωνικό νησί (αριστερά) και contour (δεξιά) με γραμμές για την λύση που δίνει ο αρχικός σειριακός κώδικας και flood που δίνει ο παράλληλος κώδικας για 24 επεξεργαστές τη χρονική στιγμή* $t = 12$

6.5 Το φράγμα του Malpasset

6.5.1 Ιστορικά στοιχεία

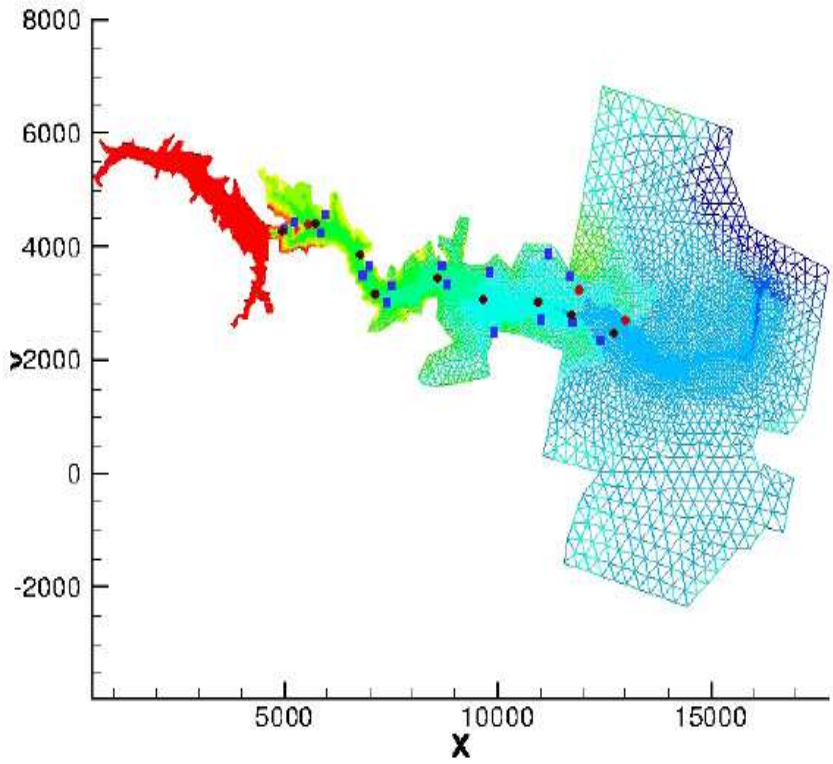
Το Malpasset ήταν ένα τόξο φράγμα στον Reyran ποταμό, κατασκευάστηκε περίπου 7 χλμ. βόρεια της Fréjus στην Κυανή Ακτή , νότια Γαλλία , στην Var διαμέρισμα . Κατέρρευσε στις 21:13 2 Δεκεμβρίου 1959, σκοτώνοντας 421 ανθρώπους από τις πλημμύρες. Διάφορες πηγές αναφέρουν αριθμούς θανάτων των 361, 400, 423, 429 ή 510. Η ζημία ανήλθε σε συνολικά 68 εκατομμύρια δολάρια. Ο ολόκληρος τοίχος

Σχήμα 6.6: *Το φράγμα του Malpasset το 1988, 29 χρόνια μετά την καταστροφή*

κατέρρευσε με μόνο μερικά τετράγωνα να παραμείνουν στη δεξιά πλευρά. Τα κομμάτια του φράγματος είναι ακόμα διάσπαρτα σε όλη την περιοχή. Η κατάρρευση δημιούργησε ένα τεράστιο κύμα με ύψος 40 μέτρα (130 πόδια) που κινούταν με 70 χιλιόμετρα (43 μίλια) ανά ώρα, καταστρέφοντας δύο μικρά χωριά - Malpasset και Bozon, όπου και τελικά κατέληξε στην θάλασσα. Το σπάσιμο του φράγματος και η διαδρομή του νερού έγινε πόλος έλξης πολλών ερευνητών τόσο στο να προσημειώσουν φυσικά υπό κλίμακα το φαινόμενο όσο και υπολογιστικά δοκιμάζοντας την συμπεριφορά διαφόρων μοντέλων.

6.5.2 Αριθμητικά αποτελέσματα

Στη βιβλιογραφία, όπως ήδη αναφέρθηκε, υπάρχουν πολλές αναφορές και μελέτες σχετικές με την φυσική αυτή καταστροφή (βλ. [27, 24, 25, 52, 57, 13, 37, 26, 23]). Δεδομένα για τη περιοχή του φαινομένου έχουμε από την Γαλλική εταιρεία ηλεκτρισμού (Electricite de France EDF). Η σχετική με αυτό εργασία προτάθηκε από την EDF στα πλαίσια του Ευρωπαϊκού προγράμματος CADAM. Οι διαστάσεις της περιοχής μελέτης είναι 17.5km×9km. Η υψομετρική της κοιλάδας κυμαίνεται από -20μ μέχρι +100μ, και εκτός της θάλασσας και του ταμιευτήρα ο πυθμένας είναι στεγνός. Το υπολογιστικό πλέγμα είναι πλήρως μη δομημένο και αποτελείται από 13.541 κόμβους και 26.000 τρίγωνα, όπως έχει παρουσιαστεί στην εργασία [25], βλ Σχήμα (6.7).

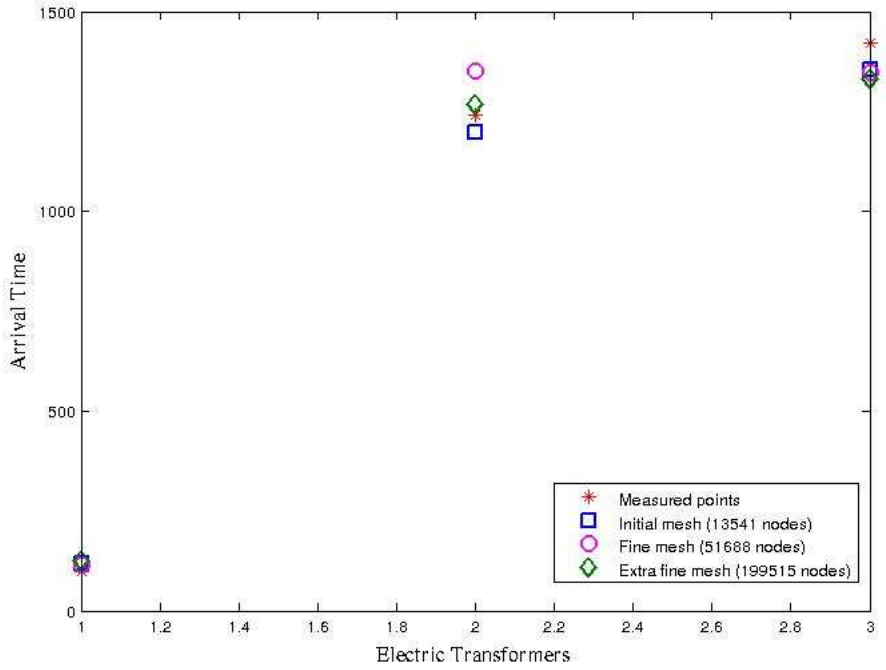


Σχήμα 6.7: *Μετασχηματιστές ♦, σημεία μετρήσεων της αστυνομίας ■ και μετρήσεις από φυσικό μουτέλο υπό κλίμακα●*

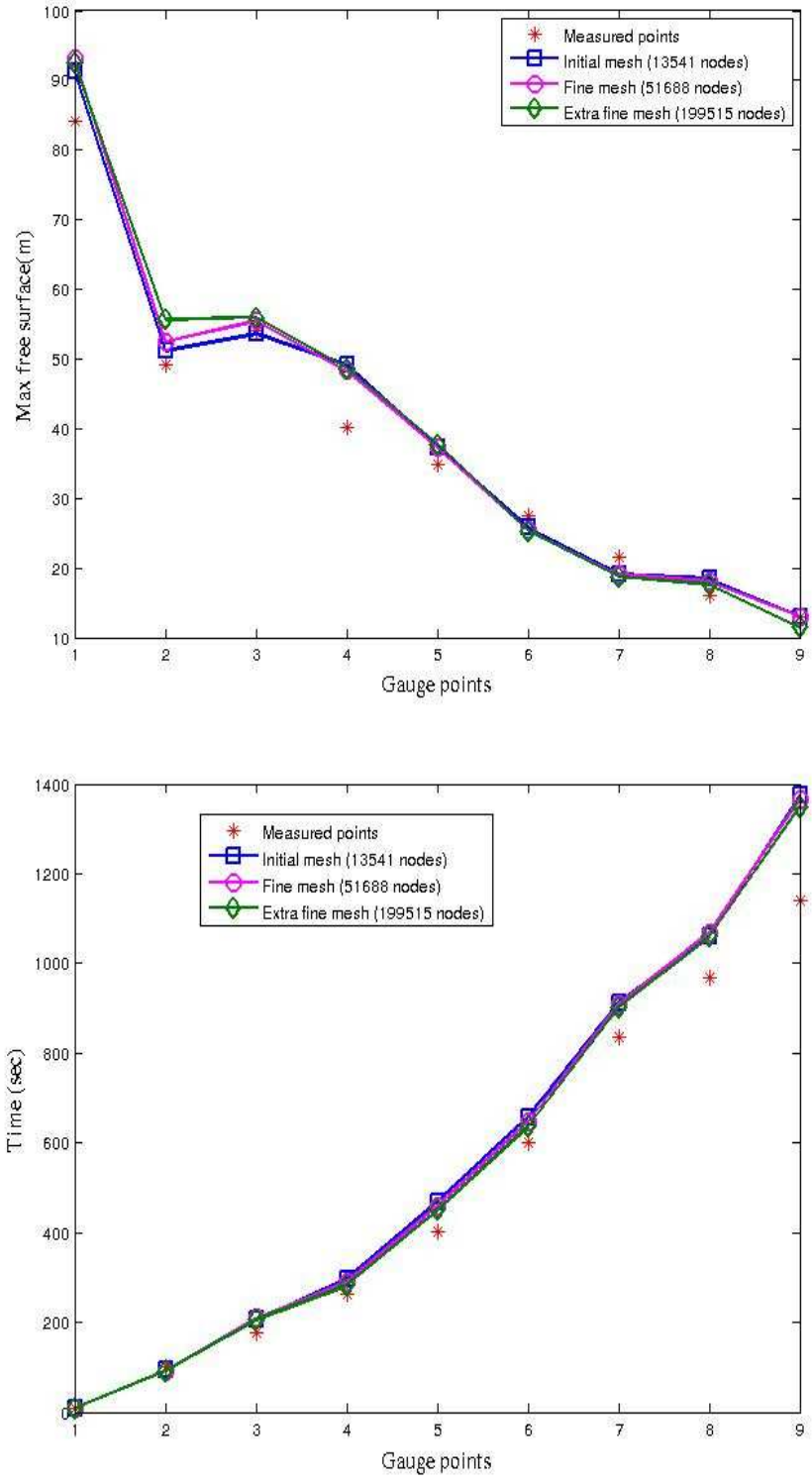
Το μόνο ανοικτό σύνορο είναι στην θάλασσα με σταθερό βάθος. Για τις αρχικές συνθήκες το βάθος του νερού μέσα στον ταμιευτήρα είναι 100μ.

Κατά την διάρκεια της καταστροφής η τοπική αστυνομία έκανε μετρήσεις σε 17 σημεία, βλ Σχήμα (6.7), σχετικές με την ώρα άφιξης του νερού καθώς επίσης και της στάθμης. Αν και οι μετρήσεις της τοπικής αστυνομίας δεν είναι ακριβείς έχουμε ακριβή χρόνο άφιξης του νερού σε τρία σημεία όπου υπήρχαν μετασχηματιστές της εταιρείας ηλεκτρισμού και ξέρουμε την ακριβή ώρα που σταμάτησαν να λειτουργούν. Επίσης το 1964 έγινε φυσική προσομοίωση υπό κλίμακα 1:4000 με σκοπό την καλύτερη κατανόηση της ροής του νερού όπου και ορίστηκαν κάποια σημεία μέσα στην κοιλάδα σχετικά με την άφιξη και την στάθμη του νερού, [24]. Όλα τα παραπάνω δεδομένα χρησιμοποιήθηκαν στο να ελεγχθεί η αριθμητική μέθοδος και ο σειριακός κώδικας, βλ [45]. Όπως και στον σειριακό κώδικα έτσι και εδώ ο αριθμητικός όρος της τριβής ορίστηκε να είναι $\theta = 0$ που σημαίνει ότι χρησιμοποιείται έμμεσο σχήμα για τον υπολογισμό της.

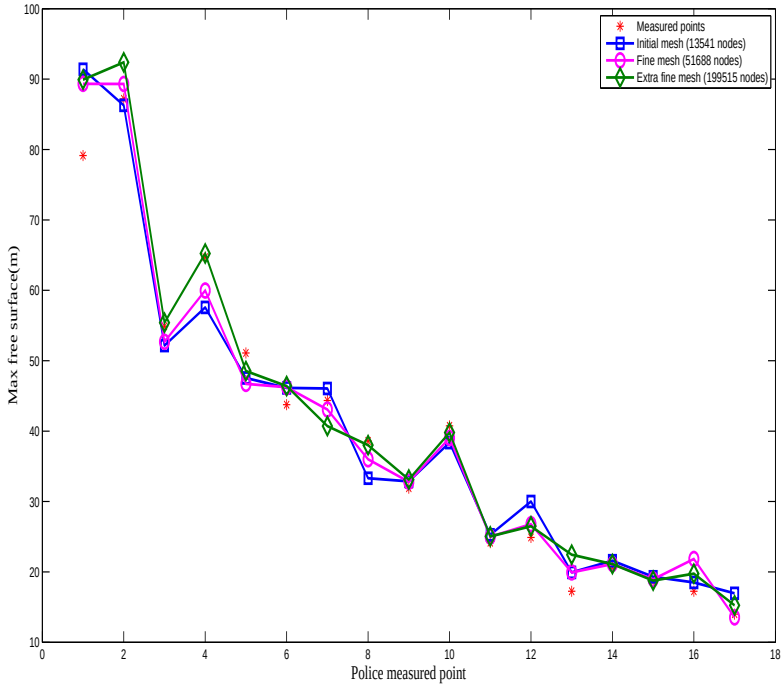
Όπως έχει ήδη αναφερθεί στην εργασία [45] το αριθμητικό μοντέλο προσεγγίζει ικανοποιητικά τα πειραματικά δεδομένα και τις μετρήσεις. Το σφάλμα που παρουσιάζεται δεν μπορούμε εύκολα να πούμε ότι οφείλεται σε σφάλμα της αριθμητικής μεθόδου ή των μετρήσεων. Κάνοντας χρήση του παράλληλου κώδικα μας δόθηκε η δυνατότητα να τρέξουμε το αριθμητικό μοντέλο για μεγαλύτερα πλέγματα ώστε να βγάλουμε κάποια συμπεράσματα για την προέλευση του σφάλματος που παρουσιάζεται.



Σχήμα 6.8: *Σύγκριση αριθμητικών αποτελεσμάτων με πειραματικά δεδομένα για διάφορα πλέγματα*



Σχήμα 6.9: *Σύγκριση αριθμητικών αποτελεσμάτων με πειραματικά δεδομένα για διάφορα πλέγματα*



Σχήμα 6.10: *Σύγκριση αριθμητικών αποτελεσμάτων με πειραματικά δεδομένα για διάφορα πλέγματα*

Σχήμα 6.11: *Στιγμιότυπα της κίνησης του νερού κατά την διάρκεια της καταστροφής τις χρονικές στιγμές* $t = 0, 400, 800, 1200, 1600, 2400$

Κεφάλαιο 7

Ανάλυση απόδοσης

7.1 Γενικά

Σε αυτή την ενότητα γίνεται ανάλυση του παράλληλου κώδικα όσον αφορά την επιτάχυνση και την απόδοση. Για το λόγο αυτό χρησιμοποιήθηκε το Beowulf Cluster του Ινστιτούτου Γενετικής και Βιολογίας (ΙΘΑΒΙΓ) του Ελληνικού Κέντρου Θαλασσίων Ερευνών (ΕΛΚΕΘΕ), το οποίο αποτελείται από 96 επεξεργαστές και 432GB RAM συνολικά. Αναλυτικότερα το cluster αποτελείται από 8 κόμβους όπου ο κάθε κόμβος έχει τα παρακάτω χαρακτηριστικά

```
2 x
Processor Number X5680
# of Cores 6
# of Threads 12
Clock Speed 3.33 GHz
Max Turbo Frequency 3.6 GHz
Intel Smart Cache 12 MB
Bus/Core Ratio 25
Intel QPI Speed 6.4 GT/s
# of QPI Links 2
Instruction Set 64-bit
Instruction Set Extensions SSE4.2
Embedded Options Available No
Lithography 32 nm
Max TDP 130 W
```

```
VID Voltage Range 0.750V-1.350V

Max Memory Size  48GB (7nodes)
Memory Types DDR3-1333 ECC
# of Memory Channels 3
Max Memory Bandwidth 32 GB/s

Max Memory Size  96GB (1nodes)
Memory Types DDR3-1333 ECC
# of Memory Channels 3
Max Memory Bandwidth 32 GB/s
```

Η διασύνδεση των παραπάνω κόμβων γίνεται με κάρτες dual 10Gb SFP+ ultra low latency πάνω σε ένα αντίστοιχο 10Gb οπτικό switch. Το λειτουργικού σύστημα είναι Linux x86_64 βασισμένο στην διανομή Gentoo με πυρήνα custom 2.6.35 ειδικά τροποποιημένο για βέλτιστη απόδοση. Ο compliler που χρησιμοποιήθηκε είναι ο intel ifort (IFORT) 12.0.2 20110112 και η βιβλιοθήκη mpich2-1.4.1p1 με τα παρακάτω optimization flags

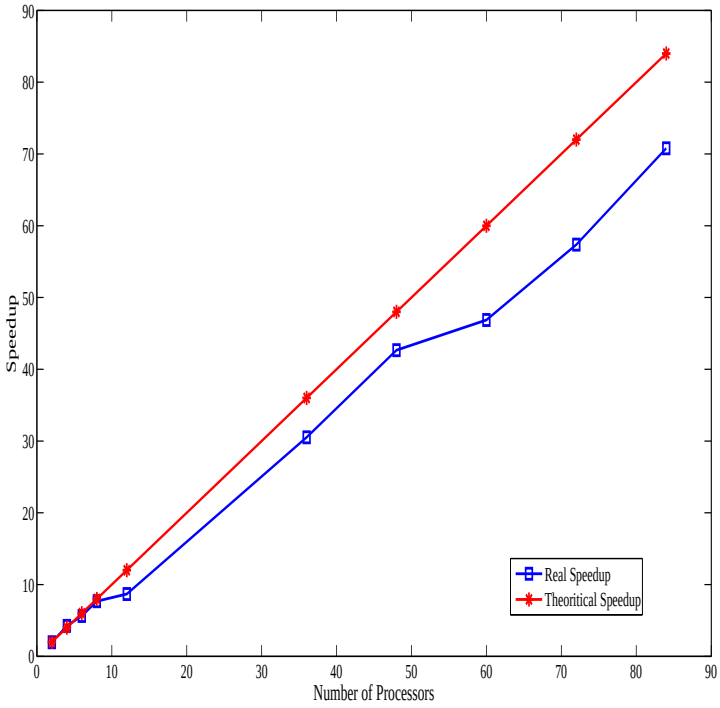
```
-O3 -ipo -xSSE4.2
```

7.2 Συντελεστής επιτάχυνσης

Ο συντελεστής επιτάχυνσης ορίζεται από την εξίσωση (1.1) και (1.2), όπου όπως έχουμε ήδη αναφέρει ο όρος θ_p της εξίσωσης (1.2) είναι το παράλληλο κομμάτι του κώδικα όπου στο συγκεκριμένο πρόβλημα είναι $\theta_p = 1$. Αυτό σημαίνει ότι ο συντελεστής επιτάχυνσης μπορεί σε ιδανικές συνθήκες να πάρει την μέγιστη τιμή.

Οι μετρήσεις έγιναν για υπολογιστικό πλέγμα 185493 κόμβων, όπου στο Σχήμα (7.1) βλέπουμε ότι ο συντελεστής επιτάχυνσης είναι γραμμικός και σε σχέση με τον ιδανικό έχουμε τρία σκαλοπάτια, στους 12, 48 και 60 επεξεργαστές. Η απόκλιση σε σχέση με τον ιδανικό έχει να κάνει με το κόστος επικοινωνίας όπου κρατώντας το μέγεθος του προβλήματος σταθερό αυξάνει κατά την αύξηση των επεξεργαστών. Τα σκαλοπάτια έχουν να κάνουν περισσότερο με την δομή του cluster.

Αν παρατηρήσει κανείς τον πίνακα (6.1) και το Σχήμα (7.2) θα διαπιστώσει ότι ο συντελεστής επιτάχυνσης είναι υπερ γραμμικός σε κάποιο σημείο. Η επιτάχυνση

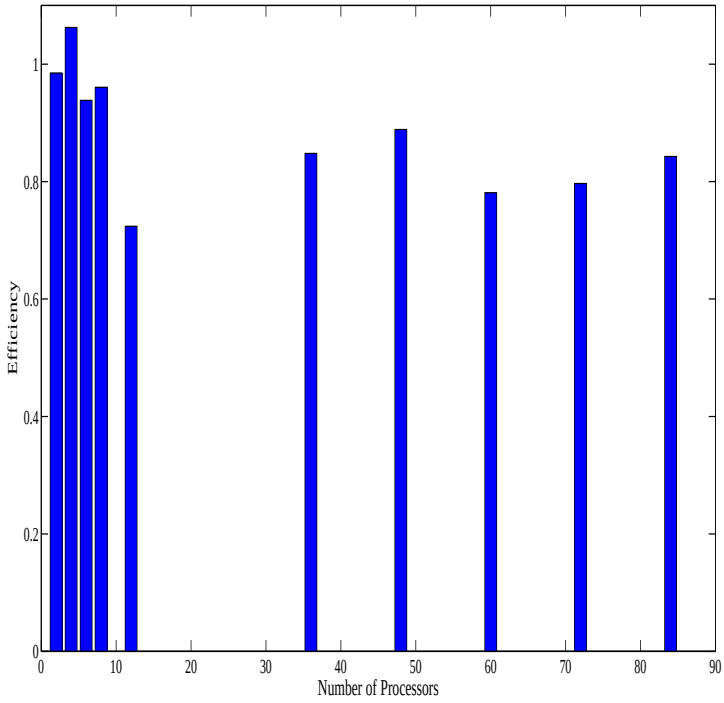


Σχήμα 7.1: *Συντελεστής επιτάχυνσης*

είναι υπερ γραμμική όταν είναι ποιο πάνω από την ιδανική θεωρητική. Αυτό το φαινόμενο συμβαίνει σπάνεια και μπερδεύει τους αρχάριους που περιμένουν κάτι λιγότερο ή ίσο του θεωρητικού. Η πιθανότερη αιτία της υπερ γραμμικής επιτάχυνσης είναι cache μνήμες και η ιεραρχία αυτών στα σύγχρονα υπολογιστικά συστήματα. Σε ένα παράλληλο κώδικα δεν αλλάζει μόνο ο αριθμός των επεξεργαστών αλλά η cache μνήμη. Αυτό σημαίνει ότι υπάρχει το ενδεχώμενο ένα μεγάλο κομμάτι δεδομένων να χωραέι στην cache οπότε επεξεργασία εκεί είναι εξαιρετικά ποιο γρήγορη.

7.3 Συντελεστής απόδοσης

Η απόδοση efficiency (1.1) μετράει το ποσοστό των πόρων του συστήματος που χρησιμοποιεί ο κώδικας, όπως είναι λογικό συνδέεται με το συντελεστή επιτάχυνσης. Στα σημεία που έχουμε γραμμικό και κοντά στο βέλτιστο συντελεστή επιτάχυνσης περιμένουμε η απόδοση να είναι κοντά στην μονάδα, βλ. Σχήμα (7.2).

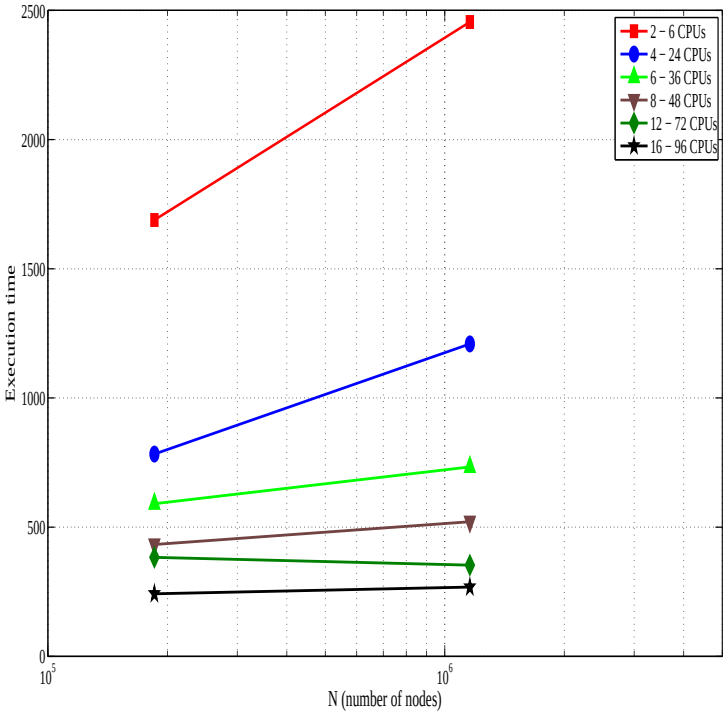


Σχήμα 7.2: *Συντελεστής απόδοσης*

7.4 Κλιμάκωση scaling

Η δυνατότητα κλιμάκωσης (scaling) είναι ένα από τα ζητούμενα ενός παράλληλου κώδικα. Όπως είδαμε στο Σχήμα (7.1) η επιτάχυνση ενός προβλήματος, όσο καλός και να είναι ο κώδικας, έχει κάποιο όριο όπου από και πέρα ο κώδικας δεν είναι το ίδιο αποδοτικός και δεν χρησιμοποιεί καλά του πόρους του συστήματος. Άρα λοιπόν για το κάθε μέγεθος πρόβλημα έχουμε έναν μέγιστο αριθμό επεξεργαστών και επιτάχυνσης, από εκεί και πέρα είναι σημαντικό να δούμε πως συμπεριφέρεται ο κώδικας σε μεγαλύτερο πρόβλημα με τους ίδιους πόρους. Σκοπός λοιπόν δεν είναι να επιταχύνουμε το πρόβλημα αλλά να λύσουμε ένα μεγαλύτερο πρόβλημα στον ίδιο χρόνο.

Αυξήσαμε λοιπόν το μέγεθος του προβλήματος κατά έξι περίπου φορές, οπότε όπως βλέπουμε στο Σχήμα (7.3) ο χρόνος είναι σχεδόν ο ίδιος για αντίστοιχα έξι φορές περισσότερους επεξεργαστές. Η κλίση που παρατηρείται οφείλεται στο γεγονός ότι μεγαλύτερο πρόβλημα αντιστοιχεί σε μεγαλύτερα μηνύματα προς αποστολή και



Σχήμα 7.3: *Κλιμάκωση*

παραλαβή. Για μεγάλο αριθμό επεξεργαστών η κλιμάκωση είναι σχεδόν 100% διότι τα μηνύματα είναι μεν πολλά αλλά είναι δε μικρά, δηλαδή πλησιάζουν το μέγεθος των μηνυμάτων του κατά 6 φορές μικρότερου προβλήματος.

7.5 Αντί επίλογου

Η παραλληλοποίηση μιας πλήρως μη δομημένης μεθόδου αποτελεί ένα δύσκολο πρόβλημα που υπόσχεται πολλά όσον αφορά την απόδοση. Εκτός την παραλληλοποίηση της μεθόδου απαιτείται ιδιαίτερη δουλειά για την οργάνωση και τον κατακερματισμό του μη δομημένου υπολογιστικού πλέγματος. Η κατασκευή του κώδικα Parmesh είχε σαν σκοπό να διαχωρίσει αυτά τα δύο πράγματα με σκοπό να κάνει, πέρα των άλλων, τον κώδικα ποιο ευέλικτο σε τυχόν διαφορετικές μορφές υπολογιστικού πλέγματος. Το αποτέλεσμα της εργασίας αυτής είναι αφενός η κατασκευή του προγράμματος που κατακερματίζει το υπολογιστικό πλέγμα και αφετέρου η κατασκευή των κατάλληλων υπορουτίνων που μπορούν εν δυνάμει να εφαρμοστούν στην επίλυση οποιουδήποτε προβλήματος με την ίδια μέθοδο, πχ εξισώσεις Euler.

Παρουσιάστηκαν αρκετά προβλήματα στον κατακερματισμό του υπολογιστικού πλέγματος, ιδιαίτερα όταν το πλέγμα παρουσίαζε τελείως μη δομημένη μορφή. Αν και αρχικά η πηγή των προβλημάτων ήταν διάφορες ατέλειες στον κώδικα (bugs) στην πορεία παρουσιάστηκαν και προβλήματα που πήγαζαν από το πως κατακερμάτιζε το Metis κάθε φορά το πλέγμα, ιδιαίτερα στην έκδοση 5. Αυτό ήταν και η βασική αιτία που το πρόβλημα του Malpasset δεν καταφέραμε να το τρέξουμε για παραπάνω από 8 επεξεργαστές σε αντίθεση με τα υπόλοιπα προβλήματα που δεν είχαμε κανένα περιορισμό.

Μελλοντική δουλειά που θα μπορούσε να γίνει είναι η κατασκευή του αρχικού πλέγματος σε γράφο τύπου dual και όχι nodal που είναι τώρα ώστε το Metis να δίνει καλύτερα αποτελέσματα καθώς επίσης και η πλήρως σύνδεσή του με τον κώδικα ώστε να λαμβάνεται υπόψη και η απόσταση των επεξεργαστών μεταξύ τους. Έτσι κατά την κατασκευή της τοπολογίας των επεξεργαστών να ορίζονται τα αντίστοιχα βάρη στις ακμές μεταξύ των κόμβων, ώστε να πετύχουμε ακόμα καλύτερη απόδοση.

APPENDIX

.1 Ansi C:Parmesh Code

```
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef struct element {
int tp,nds[3],nbrs[3],id,did;
int dnbrs[3];
int ndbc[2],bc;
int dom,ddom;
} element_t;

typedef struct node {
double x,y,z,icond[3];
int id,dom;
int did,ddom,dm;
} node_t;

typedef struct bc_element {
int nd1, nd2;
int id,tp;
} bc_element_t;


int  main(int argc, char *argv[]) {
int nprocs;
FILE *fmesh,*fepart,*fnpart,**output,**routput,**soutput;
char in_epart[20], in_npart[20];
char **fout,**rout,**sout;
char title[100];
int bl1[3],nnodes;
```

```
register int i,j,k,m,l;
double domains[6];
int tm1,tm2,tm3,tm4;
int *ntri,*nnd,id=1,tmp0,tmp1,tmp;
int nbc,*nel,*bct,**dnel,pnbc,n1bc;
int foo,pan;

element_t *tri ;
bc_element_t **bc,*ibc;
node_t *nd, *nbuf;

if (argc < 3) {
 printf("This program requires at least two argument.\n");
 exit(5);
}
fmesh = fopen(argv[1],"r");
nprocs = atoi(argv[2]);

        sprintf(in_epart,"metis.msh.epart.%d",nprocs);
        sprintf(in_npart,"metis.msh.npart.%d",nprocs);

fepart = fopen(in_epart,"r");
fnpart = fopen(in_npart,"r");


/* number of elements for each domain */
ntri = (int *)malloc(nprocs*sizeof(int));
/* number of nodes for each domain */
        nnd = (int *)malloc(nprocs*sizeof(int));

        for (i=0; i<nprocs; i++) {
         ntri[i]=0;
         nnd[i]=0;
        }
```

```
/* First mesh block */
fscanf(fmesh,"%s",title);
fscanf(fmesh,"%d %d %d",&bl1[0],&bl1[1],&bl1[2]);
tri = (element_t *)malloc(bl1[0]*sizeof(element_t));

 for (i=0; i<bl1[0]; i++){
fscanf(fmesh,"%d %d %d %d %d %d %d %d",&tri[i].tp,&tri[i].nds[0],
&tri[i].nds[1],&tri[i].nds[2],
&tri[i].nbrs[0],&tri[i].nbrs[1],&tri[i].nbrs[2],&tri[i].id);
fscanf(fepart,"%d",&tri[i].dom);
ntri[tri[i].dom]++;
tri[i].did=0;
tri[i].ddom=-1;
tri[i].ndbc[0]=-1;
tri[i].ndbc[1]=-1;
tri[i].bc=-1;
tri[i].dnbrs[0]=0;
tri[i].dnbrs[1]=0;
tri[i].dnbrs[2]=0;
  }

/* Second mesh block */
fscanf(fmesh,"%d",&nnodes);
fscanf(fmesh,"%lf %lf %lf %lf %lf %lf",&domains[0],&domains[1],
        &domains[2],&domains[3],&domains[4],&domains[5]);
nd = (node_t *)malloc(nnodes*sizeof(node_t));
for (i=0; i<nnodes; i++){
fscanf(fmesh,"%lf %lf %lf %lf %lf %lf %d",&nd[i].x,
                &nd[i].y,&nd[i].z,&nd[i].icond[0],
        &nd[i].icond[1],&nd[i].icond[2],&nd[i].id);
fscanf(fnpart,"%d",&nd[i].ddom);
nnd[nd[i].ddom]++;
nd[i].did=nd[i].id;
nd[i].dom=-1;
nd[i].dm=nd[i].dom;
```

```
}

/* Third mesh block */

fscanf(fmesh,"%d",&nbc);
bc = (bc_element_t **)malloc(nbc*sizeof(bc_element_t *));
dnel = (int **)malloc(nprocs*sizeof(int *));
nel = (int *)malloc(nbc*sizeof(int));

for (i=0; i<nbc; i++) {
 fscanf(fmesh,"%d %d",&nel[i],&tmp);
 bc[i] = (bc_element_t *)malloc(nel[i]*sizeof(bc_element_t));
 for (j=0; j<nel[i]; j++) {
   fscanf(fmesh,"%d %d %d %d",&bc[i][j].nd1,
         &bc[i][j].nd2,&bc[i][j].id,&tmp);
   bc[i][j].tp=i;
 }
}



fclose(fmesh);
fclose(fnpart);
fclose(fepart);

/* printf("----EKSODOS----\n");
for (i=0; i<bl1[0]; i++)
printf("%d\n",tri[i].nds[0]);

return (1);
*/

/* output: domain mesh file & internal domain communication file */
fout = (char **)malloc(nprocs*sizeof(char *));
// rout = (char **)malloc(nprocs*sizeof(char *));
```

```
sout = (char **)malloc(nprocs*sizeof(char *));
        output = (FILE **)malloc(nprocs*sizeof(FILE *));
//        routput = (FILE **)malloc(nprocs*sizeof(FILE *));
        soutput = (FILE **)malloc(nprocs*sizeof(FILE *));
for (i=0; i<nprocs; i++){
               fout[i] = (char *)malloc(20*sizeof(char));
//               rout[i] = (char *)malloc(20*sizeof(char));
               sout[i] = (char *)malloc(20*sizeof(char));
               output[i] = (FILE *)malloc(20*sizeof(FILE));
//               routput[i] = (FILE *)malloc(20*sizeof(FILE));
               soutput[i] = (FILE *)malloc(20*sizeof(FILE));
               sprintf(fout[i],"mesh-%3.3d.dpl",i);
//               sprintf(rout[i],"rcom-%3.3d.dpl",i);
               sprintf(sout[i],"scom-%3.3d.dpl",i);
               output[i]=fopen(fout[i],"w");
//               routput[i]=fopen(rout[i],"w");
               soutput[i]=fopen(sout[i],"w");
               fprintf(output[i],"%s\n",title);
       fprintf(soutput[i],"Global ID\tDest ID\t\tDest Domain\n");
//        fprintf(routput[i],"Global ID\tLocal ID\t\tHost Domain\n");
        }


      for(k=0; k<nprocs; k++){
      printf("Domain:%d---prin----%d\n",k,ntri[k]);

 /* Kataxorisi ton trigonon tou domain sto buffer */
        for (i=0; i<bl1[0]; i++){
if (tri[i].dom==k){
tri[i].ddom=k;
}
}


/* Kataxorisi ton gitonon pou den einai sto domain sto buffer */
```

```
for (i=0; i<bl1[0]; i++){
if (tri[i].dom==k){
for (j=0; j<3; j++){
if (tri[tri[i].nbrs[j]-1].dom!=k ){
tri[tri[i].nbrs[j]-1].ddom=k;
}

}
}
}

 /* Kataxorisi ton kombon tou domain */
for (i=0; i<bl1[0]; i++){
if (tri[i].ddom==k){
for (j=0; j<3; j++){
nd[tri[i].nds[j]-1].dom=k;
}
}
}

/* Epanelenxos gia tixon dontia */
 for (i=0; i<bl1[0]; i++){
  if ( nd[tri[i].nds[0]-1].dom==k && nd[tri[i].nds[1]-1].dom==k
              && nd[tri[i].nds[2]-1].dom==k )
              if ( tri[i].ddom != k ){
tri[i].ddom=k;
printf("********##############\n");
}
}


for (i=0; i<bl1[0]; i++){
if (tri[i].ddom==k){
for (j=0; j<3; j++){
nd[tri[i].nds[j]-1].dom=k;
```

```
}
}
}



id=1;
ntri[k]=0;
for (j=0; j<bl1[0]; j++){
if(tri[j].ddom==k){
tri[j].did=id;
id++;
ntri[k]++;
}
}

      printf("Domain:%d---meta----%d\n",k,ntri[k]);

nnd[k]=0;
id=1;
for (j=0; j<nnodes; j++){
if(nd[j].dom==k){
nnd[k]++;
nd[j].id=id;
id++;
}
}

for (j=0; j<bl1[0]; j++){
if (tri[j].ddom==k){
  for (i=0; i<3; i++){
if ( tri[j].nbrs[i] == 0 )
   tri[j].dnbrs[i]=0;
else if ( tri[tri[j].nbrs[i]-1].ddom != k)
   tri[j].dnbrs[i]=0;
```

```
else
   tri[j].dnbrs[i]=tri[tri[j].nbrs[i]-1].did;
  }
}
}


tmp0=0;
tmp1=-1;
/* n1bc metraei tous eswterikous sinoriakous kombous */
n1bc=0;
/* Grapsimo tou block ton elements sto ka8e mesh file */
fprintf(output[k],"\t%d\t %d\t %d\t\n",ntri[k],bl1[1],bl1[2]);
for (i=0; i<bl1[0]; i++){
  if (tri[i].ddom==k){
    fprintf(output[k],"\t%d\t %d\t %d\t %d\t %d\t %d\t %d\t %d\n",
    tri[i].tp,nd[tri[i].nds[0]-1].id,nd[tri[i].nds[1]-1].id,
            nd[tri[i].nds[2]-1].id,tri[i].dnbrs[0],tri[i].dnbrs[1],
            tri[i].dnbrs[2],tri[i].did);
  }
       }

/* Grapsimo ton kombon tou ka8e domain */
fprintf(output[k],"\t%d\n",nnd[k]);
fprintf(output[k],"\t%lf\t %lf\t %lf\t %lf\t
  %lf\t %lf\n",domains[0],domains[1],
          domains[2],domains[3],domains[4],domains[5]);
for (i=0; i<nnodes; i++){
 if ( nd[i].dom == k )
  fprintf(output[k],"\t%lf\t %lf\t %lf\t %lf\t %lf\t %lf\t %d\t %d\n",
               nd[i].x,nd[i].y,nd[i].z,nd[i].icond[0],
            nd[i].icond[1],nd[i].icond[2],nd[i].id,nd[i].did);
if ( nd[i].dom == k && nd[i].ddom !=k ){
fprintf(soutput[nd[i].ddom],"%d\t\t%d\t\t%d\n",nd[i].did,
                      nd[i].id,nd[i].dom);
```

```
// fprintf(routput[k],"%d\t\t%d\t\t%d\n",nd[i].did,nd[nd[i].did-1].id,
                  nd[i].ddom);
  }
}

   /* Metrima ton sinoriakon kombon ana idos kai ana domain */
dnel[k]=(int *)malloc(nbc*sizeof(int));
for (i=0; i<nbc; i++) {
dnel[k][i]=0;
for (j=0; j<nel[i]; j++) {
if (tri[bc[i][j].id-1].ddom == k)
dnel[k][i]++;
}
}

for (i=0; i<nbc; i++)
printf("ID:%d,No:%d,Size:%d\n",k,i,dnel[k][i]);
pnbc=0;
for (i=0; i<nbc; i++)
if (dnel[k][i] != 0)
pnbc++;

/* Metrima ton esoteriko boundary komvon */
n1bc=0;
for (i=0; i<bl1[0]; i++) {
if (tri[i].ddom == k){
for (j=0; j<3; j++) {
if (tri[tri[i].nbrs[j]-1].ddom != k)
n1bc++;
}
}
}

/* Memory allocation kai kataxorisi ton eswteriko boundary nodes */
ibc = (bc_element_t *)malloc(n1bc*sizeof(bc_element_t));
```

```
l=0;
for (i=0; i<bl1[0]; i++) {
if (tri[i].ddom == k){
 for (j=0; j<3; j++) {
if (tri[tri[i].nbrs[j]-1].ddom != k) {
 if (j == 0) {
ibc[l].nd1=nd[tri[i].nds[1]-1].id;
ibc[l].nd2=nd[tri[i].nds[2]-1].id;
ibc[l].id=tri[i].did;
l++;
 } else if (j==1) {
ibc[l].nd1=nd[tri[i].nds[0]-1].id;
ibc[l].nd2=nd[tri[i].nds[2]-1].id;
ibc[l].id=tri[i].did;
l++;
 }else {
ibc[l].nd1=nd[tri[i].nds[0]-1].id;
ibc[l].nd2=nd[tri[i].nds[1]-1].id;
ibc[l].id=tri[i].did;
l++;
 }
      }

}
}
}

printf("ID:%d---->n1bc=%d\n",k,n1bc);
/* Grapsimo tou bc tou ka8e domain */
fprintf(output[k],"\t%d\n",pnbc+1);
fprintf(output[k],"\t%d\t %d\n",n1bc,tmp0);
/* for (i=0; i<bl1[0]; i++){
if (tri[i].ddom==k && tri[i].dom!=k){
fprintf(output[k],"\t");
foo=0;
```

```
for (j=0; j<3; j++){
if (nd[tri[i].nds[j]-1].ddom!=k){
fprintf(output[k],"%d\t",nd[tri[i].nds[j]-1].id);
foo++;
}
}
fprintf(output[k],"%d\t %d\n",tri[i].did,tmp0);
  }
}
*/
for (i=0; i<n1bc; i++) {
fprintf(output[k],"\t%d\t%d\t%d\t%d\n",ibc[i].nd1,ibc[i].nd2,
      ibc[i].id,tmp0);
}
for (i=0; i<nbc; i++) {
if (dnel[k][i] != 0) {
fprintf(output[k],"\t%d\t %d\n",dnel[k][i],i+1);
for (j=0; j<nel[i]; j++) {
if (tri[bc[i][j].id-1].ddom == k && bc[i][j].tp == i)
fprintf(output[k],"\t%d\t%d\t%d\t%d\n",nd[bc[i][j].nd1-1].id,
                 nd[bc[i][j].nd2-1].id,tri[bc[i][j].id-1].did,tmp0);
}
}
}

/* Init */
for (i=0; i<nnodes; i++)
nd[i].dom=-1;

for (i=0; i<bl1[0]; i++){
tri[i].ddom=-1;
tri[i].did=0;
tri[i].dnbrs[0]=0;
tri[i].dnbrs[1]=0;
tri[i].dnbrs[2]=0;
```

```
}

free(ibc);

      }


return (0);

}
```

.2 Fortran 90: inpbnd.f90

```
 subroutine inpbnd
INCLUDE 'PARAM.H'

INCLUDE 'mpif.h'


character (LEN=24) :: rfilein,sfilein
integer, allocatable :: ii(:),com_map(:),sr_nds(:,:),sbuf(:),rbuf(:)
integer :: scnt,rcnt,i,j,k,l,m,d1,d2,d3,tmp,kk
integer :: ierr,snmax1,psz,stat(MPI_STATUS_SIZE)
logical :: flag
integer, allocatable :: stats(:,:), reqs(:)


!----non-blocking variables memory allocation
psz = 100
allocate( stats(MPI_STATUS_SIZE,2), reqs(psz) )
```

```
!----Read the internal boundary nodes for each domain
write(frank,'(I3.3)') myid
write(sfilein,*) 'scom','-',frank,'.dpl'



allocate(rn(0:nprocs-1))
allocate(sn(0:nprocs-1))
allocate(ii(0:nprocs-1))

open (UNIT=10,FILE=sfilein)

rewind(10)
read(10,*)

!----Calculate kai measure the Sent Nodes
rn(:)=0
sn(:)=0
scnt=0
d2=-1
d1=d2
do
    read(10,*,END=200)tmp,tmp,d1
    sn(d1)=sn(d1)+1
    d2=d1
    scnt=scnt+1
end do
200 continue

!----Max size of Sent Nodes array
snmax=sn(0)
do i=1,nprocs-1
  if (sn(i) > snmax) snmax=sn(i)
end do
```

```
call mpi_allreduce(snmax,snmax1,1,MPI_INTEGER, &
MPI_MAX,MPI_COMM_WORLD, ierr)
snmax=snmax1
call mpi_barrier(MPI_COMM_WORLD,ierr)
!----Memory allocations of buffer arrays
allocate(snd_nds(snmax,0:nprocs-1), &
  rcv_nds(snmax,0:nprocs-1),sr_nds(snmax,0:nprocs-1))
allocate(qsnd(3,snmax),qrcv(3,snmax),sbuf(snmax),rbuf(snmax))

snd_nds(:,:)=0
rcv_nds(:,:)=0
sr_nds(:,:)=0

rewind(10)
read(10,*)

!----Read the Sent nodes array
ii(:)=1
do
    read(10,*,END=500)d2,d3,d1
    snd_nds(ii(d1),d1)=d2
    sr_nds(ii(d1),d1)=d3
    ii(d1)=ii(d1)+1
end do
500 continue

!---Search and Find the local index of Sent Nodes
do k=0,nprocs-1
   do i=1,sn(k)
    do j=1,nnode
     if (snd_nds(i,k) == indx(j)) snd_nds(i,k)=j
     end do
   end do
end do
```

```
!---create Communication Map
allocate(com_map(0:nprocs-1),topo(0:nprocs-1,0:nprocs-1))
com_map(:)=0

do i=0,nprocs-1
   if (sn(i) /= 0) com_map(i)=1
end do

close(10)

!----All Gather the communication topology
 call MPI_ALLGATHER(com_map,size(com_map),   &
               MPI_INTEGER,topo,size(com_map), &
               MPI_INTEGER, MPI_COMM_WORLD, ierr)

 call MPI_Barrier(MPI_COMM_WORLD,ierr)

!----Create graph topology
 call graph_topo

      do i=1,nneighbors
        k=neighbors(i)
        call MPI_Send(sn(k),1,MPI_INTEGER,k,1, &
                COMM_GRAPH,ierr)
        call MPI_Recv(rn(k),1,MPI_INTEGER,k,1, &
                COMM_GRAPH,stat,ierr)
      end do

      call MPI_TYPE_CONTIGUOUS(snmax,MPI_INTEGER, &
           columntype,ierr)
      call MPI_TYPE_COMMIT(columntype,ierr)

      do i=1,nneighbors
        k=neighbors(i)
```

```
        call MPI_Send(sr_nds(1,k),1,columntype,k,myid, &
                COMM_GRAPH,ierr)
        call MPI_Recv(rcv_nds(1,k),1,columntype,k,k, &
                COMM_GRAPH,stat,ierr)
      end do

        do i=0,nprocs-1
         if ( myid /= i ) then
          do l=1,rn(i)
           indexnod(rcv_nds(l,i))=-1
          end do
         end if
        end do

      return
end subroutine inpbnd
```

.3 Fortran 90: graph_top.f90

```
 subroutine graph_topo
include 'PARAM.H'
include 'mpif.h'

      integer   :: ierr,i,j,k,stat
      integer   :: nnodes,nedges
      integer   :: degree
      integer ,allocatable      :: indexl(:),edgs(:)
      logical   :: reorder

     allocate( indexl(nprocs) )
     degree=0
     indexl(:)=0
     do i=0,nprocs-1
     do j=0,nprocs-1
        if ( topo(i,j) == 1 ) degree=degree+1
```

```
     enddo
     indexl(i+1)=degree
     end do
     allocate (edgs(degree))

     k=1
     do i=0,nprocs-1
     do j=0,nprocs-1
        if (topo(i,j) == 1) then
                edgs(k) = j
                k=k+1
        end if
     end do
     end do


     reorder=.false.

     call MPI_GRAPH_CREATE(MPI_COMM_WORLD,nprocs, &
    indexl,edgs,reorder,COMM_GRAPH,ierr)

     call MPI_TOPO_TEST(COMM_GRAPH, stat,ierr)
     call MPI_GRAPHDIMS_GET(COMM_GRAPH,nnodes,&
  nedges,ierr)
     call MPI_GRAPH_GET(COMM_GRAPH, nnodes, nedges, &
  indexl, edgs,ierr)

     call MPI_GRAPH_NEIGHBORS_COUNT(COMM_GRAPH, myid, &
  nneighbors, ierr)
     allocate(neighbors(nneighbors))
     call MPI_GRAPH_NEIGHBORS(COMM_GRAPH, myid, nneighbors, &
  neighbors, ierr)

     return
end subroutine graph_topo
```

.4 Fortran 90: bc_com.f90

```
 subroutine bc_com
INCLUDE 'PARAM.H'

INCLUDE 'mpif.h'
      integer   :: i,j,k,l,ierr
      integer   :: stat(MPI_STATUS_SIZE)



      do i=1,nneighbors
        k=neighbors(i)
        do l=1,sn(k)
           qsnd(1,l)=q(1,snd_nds(l,k))
           qsnd(2,l)=q(2,snd_nds(l,k))
           qsnd(3,l)=q(3,snd_nds(l,k))
        end do

        call MPI_Send(qsnd,3*sn(k),MPI_DOUBLE_PRECISION,&
      k,myid,COMM_GRAPH,ierr)
        call MPI_Recv(qrcv,3*rn(k),MPI_DOUBLE_PRECISION,k, &
      k,COMM_GRAPH,stat,ierr)

        do l=1,rn(k)
           q(1,rcv_nds(l,k))=qrcv(1,l)
           q(2,rcv_nds(l,k))=qrcv(2,l)
           q(3,rcv_nds(l,k))=qrcv(3,l)
        end do

      end do

      return

end subroutine bc_com
```

Bibliography

[1] J. M. A. Serrano-Pacheco and P. Garcia-Navarro. Finite volume method for the simulation of the waves generated by landslides. *Journal of Hydrology*, pages 373–273, 2009.

[2] F. Aureli, P. Maranzoni, and C. Ziveri. A weighted surface-depth gradient method for the numerical integration of the 2D shallow water equations with topography. *Advances in Water Resources*, 31:962, 2008.

[3] T. J. Barth. Aspects of unstructured grids and finite volume solvers for the Euler and Navier-Stokes equations. In Special Course on Unstructured Grid Methods for advection Dominated Flows, AGARD report 787, 1992.

[4] P. D. Bates and A. P. J. De Roo. A simple raster-based model for floodplain inudation. *Journal of Hydrology*, 236:54, 2000.

[5] P. D. Bates and M. S. Horritt. Predicting floodplain inundation: raster-based modelling versus the finite element approach. *Hydrological Processes*, 15:825, 2001.

[6] P. D. Bates and M. S. Horritt. Evaluation of 1-d and 2-d numerical models for prediciting river flood inundation. *Journal of Hydrology*, 268:87, 2002.

[7] A. Bermudez, A. Dervieux, J. A. Desideri, and M. E. Vázquez. Upwind schemes for the two-dimensional shallow water equations with variable depth using unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, 155(1-2):49, 1998.

[8] A. Bermudez and M. Vázquez-Cendón. Upwind methods for hyperbolic conservation laws with source terms. *Computers and Fluids*, 23:1049, 1994.

[9] S. F. Bradford and B. F. Sanders. Finite-volume model for shallow-water flooding of arbitary topography. *J. Hydraul. Engrg-ASCE*, 128:289, 2002.

[10] R. L. D. Brett F. Sanders, Jochen E. Schubert. Parbrezo: A parallel, unstructured grid, godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. *Advances in Water Resources*, 115(12):1456–1467, 2010.

[11] M. J. Briggs, C. E. Synolakis, G. S. Harkins, and D. R. Green. Laboratoty experiments of tsunami runup on a circular island. *Pure Appl. Geophys.*, 144:569, 1995.

[12] M. Brocchini and N. Dodd. Nonlinear shallow water equations modeling for coastal engineering. *J. Waterw. Port, Coastal, Ocean Eng.*, 134:104, 2008.

[13] P. Brufau, P. García-Navarro, and M. Vázquez-Cendón. Zero mass error using unsteady wetting-drying coditions in shallow flows over dry irregular topography. *Int. J. Numer. Meth. Fluids*, 45:1047, 2004.

[14] P. Brufau, P. García-Navarro, and M. E. Vázquez-Cendón. Zero mass error using unsteady wetting-drying coditions in shallow flows over dry irregular topography. *Int. J. Numer. Meth. Fluids*, 45:1047, 2004.

[15] P. Brufau, M. Vázquez-Cendón, and P. Gracía-Navarro. A numerical model for the flooding and drying of irregular domain. *Int. J. Numer. Meth. Fluids*, 39:247, 2002.

[16] S. Bunya, E. J. Kubatko, J. J. Westerink, and C. Dawson. A wetting and drying treatment for the Runge-kutta discontinuous Galerkin solution tothe shallow water equations. *Comput. Methods Appl. Mech. Enrg.*, 198, 2009.

[17] M. J. Castro, A. M. Ferreiro, J. A. García-Rodriguez, J. M. González-Vida, J. Macías, C. Parés, and M. E. Vázquez-Cendón. The numerical treatment of wet/dry fronts in shallow flows: Application to one-layer and two-layer systems. *Mathematical and Computer Modelling*, 42:419, 2005.

[18] V. Casulli and P. Zanolli. Comparing analytical and numerical solution of nonlinear two and three-dimensional hydrostatic flows. *Int. J. Num. Meth. Fluids*, 53, 2007.

[19] L. Cea, J. Puertas, and M. E. Vázquez-Cendón. Depth averaged modelling of turbulent shallow water flow with wet-dry fronts. *Arch. Comput. Methods Eng.*, 14:303, 2007.

[20] B. H. Choi, D. C. Kim, E. Pelinovsky, and S. B. Woo. Three-dimensional simulation of tsunami run-up around conical island. *Coastal Engineering*, 54:618, 2007.

[21] A. I. Delis, M. Kazolea, and N. A. Kampanis. A robust high resolution finite volume scheme for the simulation of long waves over complex domain. *Int. J. Numer. Meth. Fluids*, 56:419, 2008.

[22] J. M. Gallardo, C. Parés, and M. Castro. On a well-balanced higher-order finite volume scheme for shallow water equations with topography and dry areas. *J. Comp. Phys.*, 227:574, 2007.

[23] D. L. George. Adaptive finite volume methods with well-balanced Riemann solvers for modeling floods in rugged terrain: Application to the Malpasset dam-break flood (France, 1959). *Int. J. for Num. Meth. Fluids*, In press.

[24] N. Goutal. The Malpasset dam failure - An overview and test case definition. *Proceeding of the 4rd CADAM workshop*, Zaragoza, 1999.

[25] J. M. Hervouet. A high-resolution 2-d dam-break model using parallelization. *Hydrological Processes*, 14:2211, 2000.

[26] J. M. Hervouet. *Hydrodynamics of free surface flows: Modelling with the finite element method*. Wiley, 2007.

[27] J.-M. Hervouet and A. Petitjean. Malpasset dam-break revisited with 2-dimensional computations. *J. of Hydraulic Res.*, 37:777, 1999.

[28] T. Hoefler, R. Rabenseifner, H. Ritzdorf, B. R. de Supinski, R. Thakur, and J. L. Traeff. The scalable process topology interface of mpi 2.2. *Concurrency and Computation: Practice and Experience*, 23(4):293–310, Aug. 2010.

[29] T. Hoefler and M. Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11)*, pages 75–85. ACM, Jun. 2011.

[30] M. E. Hubbard and N. Dodd. A 2D numerical model of wave runup and overtoping. *Coastal Eng.*, 47:1, 2002.

[31] M. E. Hubbard and N. Dodd. A 2D numerical model of wave runup and overtoping. *Coastal Eng.*, 47:1, 2002.

[32] M. E. Hubbard and P. Garcia-Navarro. Flux difference splitting and the balancing of source terms and flux gradients. *J. Comp. Phys.*, 165:2, 2000.

[33] P. G.-N. J. Murillo and J. Burguete. Time step restrictions for well-balanced shallow water solutions in non-zero velocity steady states. *Int. J. for Num. Meth. Fluids*, page 60:1351, 2009.

[34] K. Lab, 2009. Digitical Technology Center, University of Minnesota.

[35] M. H. Lallemand. Etude de schémas Runge-Kutta á 4 pas pour la résolution multigrille des équations d'Euler 2D. Raport de Recherche, INRIA, 1988.

[36] M. H. Lallemand. Dissipative properties of Runge-Kutta schemes with upwind spatial approximation for the Euler equations. Raport de Recherche No 1173, INRIA, 1990.

[37] D. Liang, L. Binliang, and R. A. Falconer. Simulation of rapidly varying flow using an efficient TVD-MacCormack scheme. *Int. J. Num. Meth. Fluids*, 53:811, 2007.

[38] P. L. F. Liu, Y. S. Cho, M. J. Briggs, U. Kanoglou, and C. E. Synolakis. Runup of solitary wave on a circular island. *J. Fluid Mech.*, 302:259, 1995.

[39] P. J. Lynett, T. R. Wu, and P. L. Liu. Modeling wave runup with depth integrated equations. *Coastal Eng.*, 46:89, 2002.

[40] M. D. L. A. J. M. M. J. M. G. Manuel J Castro, Sergio Ortega. Gpu computing for shallow water flow simulation based on finite volume schemes. *Comptes Rendus Mécanique*, 339(2-3):165–184, 2011.

[41] F. Marche. A simple well-balanced model for two-dimensional coastal engineering applications. In *Hyperbolic problems : theory, numerics, applications*, Springer Berlin Heidelberg, 271-283 2008.

[42] F. Marche, P. Bonneton, P. Fabrie, and N. Seguin. Evaluation of well-balanced bore-capturing schemes for 2D wetting and drying processes. *Int. J. Numer. Meth. Fluids*, 53:867, 2007.

[43] D. J. Mavriplis. Unstructured mesh discritizations and solvers for computational aerodynamics. 18th AIAA Computational Fluid Dynamics Conference, AIAA 2007-3955, 2007.

[44] J. Murillo, P. Garcia-Navarro, J. Burguete, and P. Brufau. The influence of source terms on stability, accuracy and conservation in two-dimensional shallow flow simulation using triangular finite volumes. *Int. J. for Numer. Methods in Fluids*, 54:543, 2007.

[45] I. K. Nikolos and A. I. Delis. An unstructured node-centered finite volume scheme for shallow water flows with wet/dry fronts over complex topography. *Comput. Methods Appl. Mech. Engrg*, 198:3723, 2009.

[46] M. Ricchiuto and A. Bollermann. Accuracy of stabilized residual distribution for shallow water flows including dry beds. In *Proceedings of the 12th international conference on hyperbolic problems : theory, numerics, applications (HYP08)*, Maryland, USA, June 2008.

[47] M. Ricchiuto and A. Bollermann. Stabilized residual distribution for shallow water simulations. *J. Comp. Phys.*, 1071:1115, 2009.

[48] P. L. Roe. Aprroximate Riemann solvers, parameter vectors, and difference schemes. *J. Comp. Phys.*, 43:357, 1981.

[49] W. C. Thacker. Some exact solutions to the nonlinear shallow water wave equations. *J. Fluid Mech.*, 107:499, 1981.

[50] V. V. Titov and C. E. Synolakis. Modeling of breaking and nonbreaking long-wave evolution and runup using VTCS-2. *J. Waterw. Port, Coastal, Ocean Eng.*, 121:308, 1995.

[51] V. V. Titov and C. E. Synolakis. Numerical modeling of tidal wave runup. *J. Waterw. Port, Coastal, Ocean Eng.*, 124:157, 1998.

[52] A. Valiani, V. Caleffi, and A. Zanni. Case study: Malpasset dam-break simulation using a two-dimensional finite volume method. *J. Hydraulic Engrg-ASCE*, 128:460, 2002.

[53] G. D. Van Albada, B. Van Leer, and W. W. Roberts. A comparative study of computational methods in cosmic gas dynamics. *Astron. Astrophysics*, 108:46–84, 1982.

[54] B. van Leer. Towards the ultimate conservative difference scheme V. A second order sequel to Godunov's method. *J. Comp. Phys.*, 32:101, 1979.

[55] P. Vavilis and J. A. Ekaterinaris. Science and supercomputing in europe. Technical report, 2005.

[56] H. Yeh, P. Liu, and C. Synolakis. *Long Wave Runup Models*. World Scientific Publishing Co., 1996.

[57] X. Ying, S. S. Y. Wang, and A. A. Khan. Numerical simulation of flood inundation due to dam and levee breach. In P. Bizier and P. DeBarry, editors, *World Water and Environmental Resources Congress 2003*, page 366. ASCE., 2003.