Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Προγραμματισμός Επιλύτη Εξισώσεων Euler σε Διδιάστατη Ροή με τη Μέθοδο των Τεμνόμενων Κυψελών (Cut-Cells) σε Επεξεργαστές Καρτών Γραφικών

Διπλωματική Εργασία

Νικόλαος Μ. Αμαρτωλός

Επιβλέπων: Κ. Χ. Γιαννάκογλου, Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

Προγραμματισμός Επιλύτη Εξισώσεων Euler σε Διδιάστατη Ροή με τη Μέθοδο των Τεμνόμενων Κυψελών (Cut-Cells) σε Επεξεργαστές Καρτών Γραφικών

Διπλωματική Εργασία
Νικόλαος Μ. Αμαρτωλός

Επιβλέπων: Κ. Χ. Γιαννάκογλου, Καθηγητής Ε.Μ.Π.

Οκτώβριος 2015

Περίληψη

Τα τελευταία χρόνια, η Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης (ΜΠΥΡ&Β) του Εργαστηρίου Θερμικών Στροβιλομηχανών του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ) δραστηριοποιείται συστηματικά στην αριθμητική επίλυση προβλημάτων Υπολογιστικής Ρευστοδυναμικής σε επεξεργαστές καρτών γραφικών (GPUs). Στο πλαίσιο αυτό, έχει πραγματοποιηθεί ανάπτυξη επιλυτών των εξισώσεων ροής (Euler, Navier-Stokes) στη γλώσσα προγραμματισμού CUDA C της εταιρίας NVIDIA σε δομημένα και μη-δομημένα πλέγματα επιτυγχάνοντας επιταχύνσεις μεγαλύτερες του x40 συγκριτικά με επιλύτες που εκτελούνται σε CPUs.

Στην παρούσα διπλωματική εργασία, διερευνάται η χρήση καρτεσιανού πλέγματος με τη μέθοδο τεμνόμενων κυψελών (cut-cells) για την επίλυση των εξισώσεων ροής σε GPUs. Για το σκοπό αυτό, αναπτύχθηκε κώδικας σε CUDA C, που επιλύει τις διδιάστατες εξισώσεις Euler μόνιμης ροής συμπιεστού ρευστού με τη μέθοδο τεμνόμενων κυψελών. Η ανάπτυξη του κώδικα πραγματοποιήθηκε με αφετηρία υπάρχοντα αντίστοιχο κώδικα, που εκτελείται σε CPU, της ΜΠΥΡ&Β/ΕΜΠ, με τον οποίο και συγκρίθηκε. Η εκτέλεση του κώδικα έγινε σε κάρτες γραφικών GeForce GTX 670 της NVIDIA. Η σύγκριση των αποτελεσμάτων και των χρόνων εκτέλεσης του κώδικα που αναπτύχθηκε, με τον κώδικα που εκτελείται σε CPU, πραγματοποιήθηκε σε ίδια πλέγματα και ίδιες συνθήκες εξωτερικής ροής. Υπήρξε πιστή αναπαραγωγή των αποτελεσμάτων, ενώ επιτεύχθηκε σε ορισμένες περιπτώσεις επιτάχυνση στο χρόνο εκτέλεσης μεγαλύτερη του x40.

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Laboratory of Thermal Turbomachines
Parallel CFD & Optimization Unit

Programming of a Solver for Euler Equations in 2D Flows using Cut-Cells Method in GPUs

Diploma Thesis

Nikolaos M. Amartolos

Advisor: K. C. Giannakoglou, Professor NTUA

October 2015

Abstract

In the last years, the Parallel CFD & Optimization Unit of the Laboratory of Thermal Turbomachines of the National Technical University of Athens is developing CFD methods-software running on GPUs. Solvers of the flow equations (Euler, Navier-Stokes) have been developed, written in CUDA C, a programming language by NVIDIA, using structured and unstructured grids, achieving accelerations greater than x40 comparing to solvers which run exclusively on CPUs.

In this diploma thesis, the use of a 2D Cartesian grid using cut-cells method was developed and used to solve the flow equations. To this purpose, a new code was developed, written in CUDA C, which solves the 2D, steady, compressible, Euler equations using cut-cells method. The code development was based and validated using the existing, in-house code which is executed in CPU. All runs are performed on NVIDIA GeForce GTX 670 GPUs. The comparison of the results and times of convergence between the developed code and the code executed on the CPUs is made on the same grids, with the same farfield flow conditions. The results of both solvers are identical and the speed-up of the GPU is greater than x40.

Ευχαριστίες

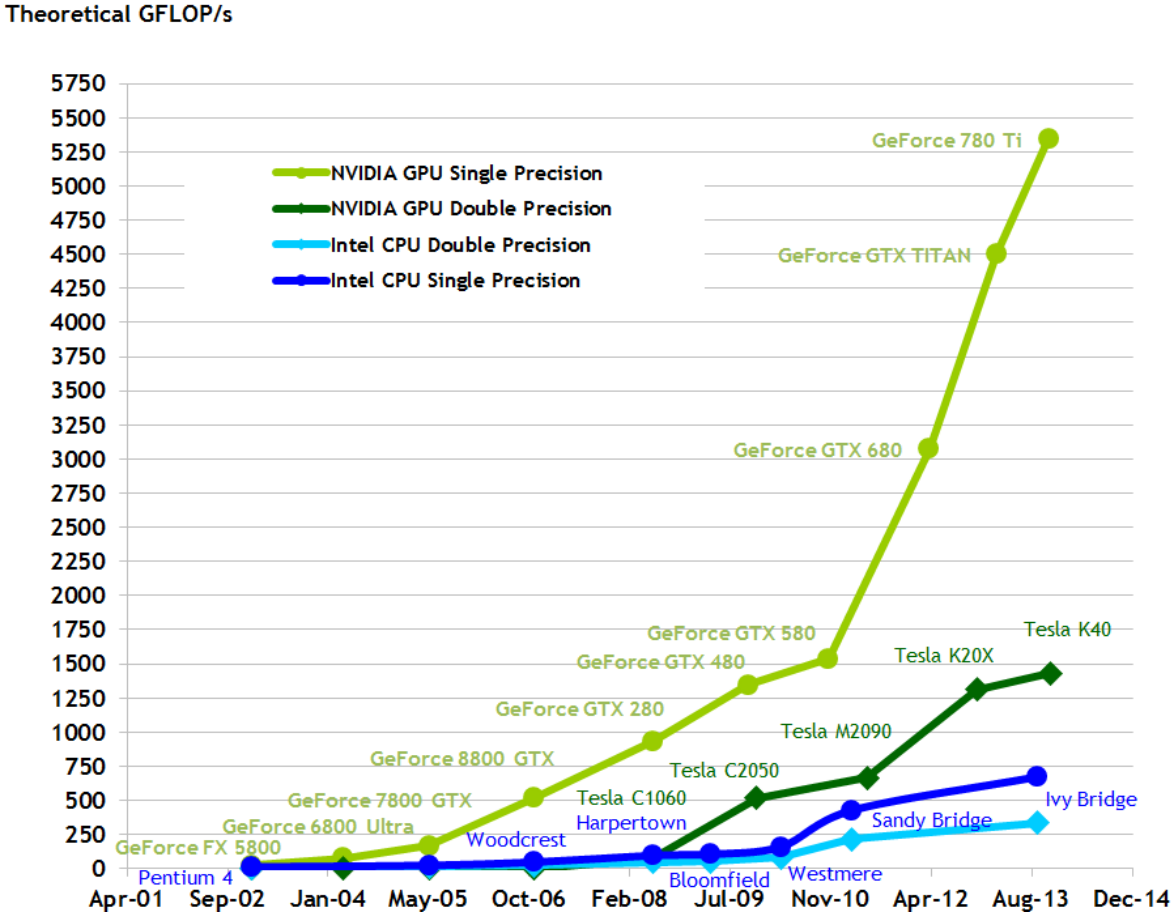
Με αφορμή την ολοκλήρωση της διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Κ.Χ. Γιαννάκογλου για την δυνατότητα που μου έδωσε να ασχοληθώ με το παρόν αντικείμενο, για την καθοδήγησή του καθώς και τη δυνατότητα χρήσης του τεχνικού εξοπλισμού της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης. Επίσης θα ήθελα να ευχαριστήσω τα μέλη της ερευνητικής ομάδας της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης για την πολύ χρήσιμη υποστήριξή τους. Συγκεκριμένα, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Κ. Σαμούχο για την βοήθεια, τις συμβουλές του και τον χρόνο που αφιέρωσε κατά την διάρκεια εκπόνησης της παρούσας εργασίας. Τέλος, θα ήθελα να ευχαριστήσω τον Δρ. Ξ. Τρομπούκη και τον υποψήφιο διδάκτορα Κ. Τσιάκα για τη βοήθειά τους σε θέματα σχετικά με τις κάρτες γραφικών και τη γλώσσα προγραμματισμού CUDA C.

Περιεχόμενα

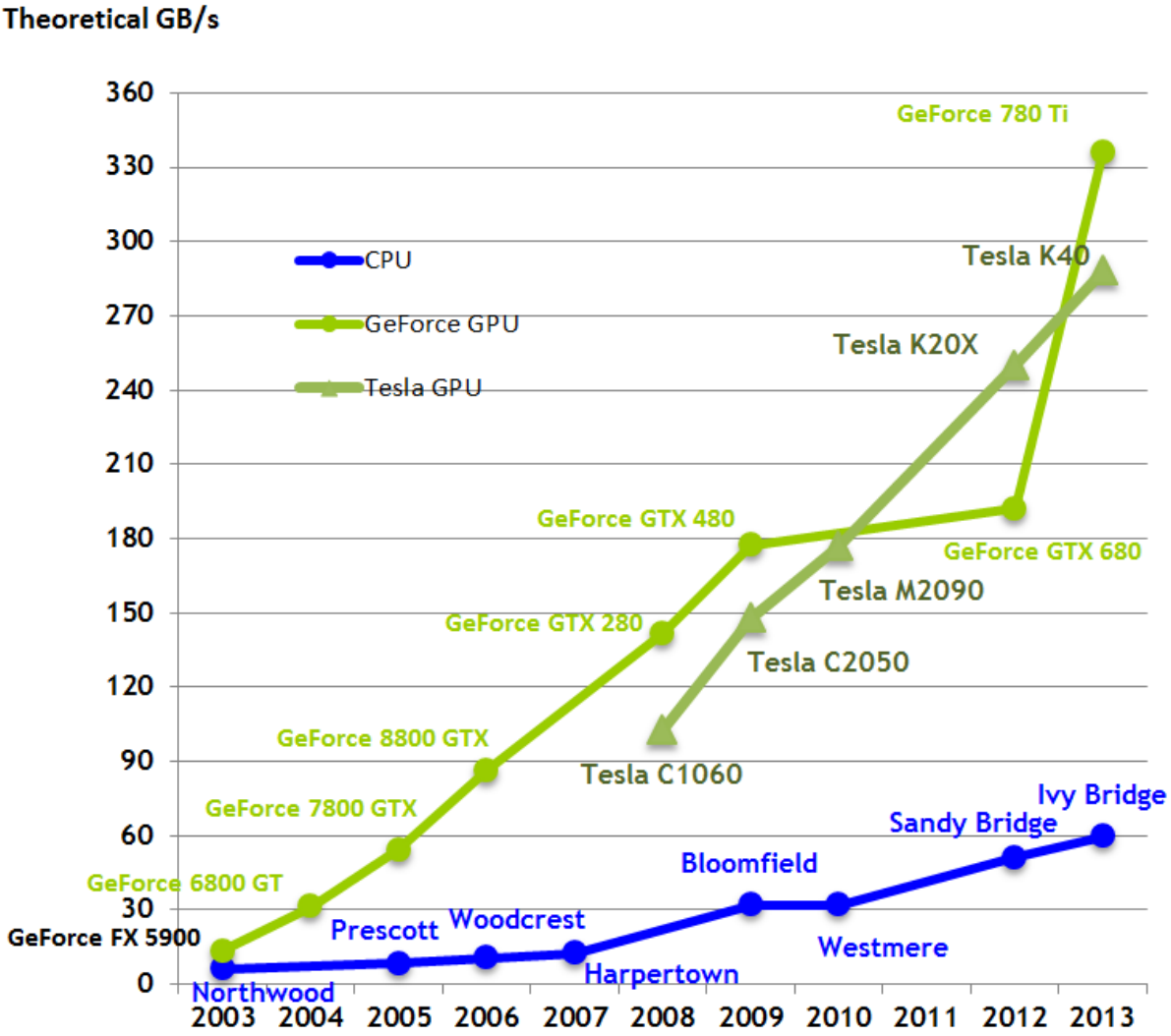
Κεφάλαιο 1

Εισαγωγή

Ένα από τα βασικά προβλήματα που αντιμετωπίζει η υπολογιστική ρευστοδυναμική, στο σημερινό της επίπεδο, είναι η έλλειψη υπολογιστικής ισχύος, καθώς η επίλυση των θεωρητικών μοντέλων προσομοίωσης ροής είναι χρονοβόρα και ο χρόνος που απαιτείται είναι ακόμα και απαγορευτικός, σε κάποιες περιπτώσεις, για τη βιομηχανία. Στην παρούσα εργασία, εξετάζεται η χρήση της κάρτας γραφικών (GPU, Graphics Processing Units), αντί του κεντρικού επεξεργαστή του υπολογιστή (CPU, Central Processing Unit) στην Υπολογιστική Ρευστοδυναμική, αφού οι σημερινές GPUs προσφέρουν υπολογιστική ισχύ μεγαλύτερη των CPU τουλάχιστον κατά μία τάξη μεγέθους, όπως φαίνονται και στα σχήματα 1.1 και 1.2.



Σχήμα 1.1: Ταχύτητα εκτέλεσης πράξεων κινητής υποδιαστολής. Παρατηρούμε ότι τα τελευταία χρόνια, με τη ραγδαία εξέλιξη των GPUs, η εκτέλεση των πράξεων απλής ακρίβειας γίνεται μέχρι και 7 φορές πιο γρήγορα σε GPUs από ότι σε CPUs και οι διπλής ακρίβειας πράξεις μέχρι και 5 φορές. Αξιοσημείωτο είναι το γεγονός πως η εκτέλεση πράξεων διπλής ακρίβειας σε GPUs γίνεται περίπου 2 φορές πιο γρήγορα από ότι σε CPUs για απλή ακρίβεια. [1]



Σχήμα 1.2: Εύρος Ζώνης (Bandwidth) της μνήμης των GPUs και CPUs τα τελευταία χρόνια. Παρατηρούμε ότι η νέα γενιά GPUs μπορεί και διαβάζει τουλάχιστον 5 φορές πιο γρήγορα δεδομένα που είναι αποθηκευμένα στη μνήμη της από ότι οι σύγχρονες CPUs. [1]

1.1 Οι Κάρτες Γραφικών σήμερα

Μέχρι πριν από λίγα χρόνια, η ισχύς ενός υπολογιστή εξαρτάτο από την ταχύτητα χρονισμού του πυρήνα της CPU και για την επίτευξη μεγαλύτερης υπολογιστικής ισχύος χρειαζόταν πιο γρήγορος χρονισμός (clock rate). Όμως, η ενέργεια που απαιτείται για τη λειτουργία του επεξεργαστή είναι ανάλογη του τετραγώνου της ταχύτητας χρονισμού του πυρήνα του. Έτσι φαίνεται πως η συνεχής αύξηση της ταχύτητας χρονισμού του πυρήνα του επεξεργαστή δεν είναι αποδεκτή λύση καθώς οδηγεί σε δυσανάλογα μεγάλη κατανάλωση ενέργειας. Γι' αυτόν το λόγο, οι κατασκευαστές ξεκίνησαν να αυξάνουν το πλήθος των πυρήνων ανά επεξεργαστή, αντί της ταχύτητας χρονισμού του κάθε πυρήνα ξεχωριστά. Η τεχνική αυτή έχει γίνει ευρέως αποδεκτή και, σήμερα, υπάρχουν στην παραγωγή

επεξεργαστές που αποτελούνται μέχρι και από 61 πυρήνες, ενώ ετοιμάζονται επεξεργαστές 80 πυρήνων. [19]

Την ίδια τεχνική χρήσης πολλών πυρήνων χρησιμοποιούν και οι GPUs, οι οποίες ακολουθούν πλήρως παράλληλη λογική και αποτελούνται από εκατοντάδες ή και χιλιάδες πυρήνες καθεμιά. Συγκριτικά με τους πυρήνες των CPUs, οι πυρήνες των GPUs είναι αισθητά πιο αργοί όμως, λόγω του πλήθους τους, οι σημερινές GPU πετυχαίνουν ταχύτητες επεξεργασίας δεδομένων πολλαπλάσιες των αντίστοιχων που επιτυγχάνονται από τις CPUs. Έτσι, τα τελευταία χρόνια αναπτύχθηκαν GPU αρχιτεκτονικών συμβατών με τα διεθνή πρότυπα εκτέλεσης πράξεων καθώς και μέθοδοι προγραμματισμού των GPUs αυτών βασισμένες σε γλώσσες προγραμματισμού όπως η C/C++, η FORTRAN, η Python και η OpenCL. Οι αρχιτεκτονικές αυτές είναι η CUDA, που αντιστοιχεί στα αρχικά των «Compute Unified Device Architecture» και η εξέλιξη της τεχνολογίας των Streams, οι οποίες αναπτύχθηκαν από τις εταιρίες NVIDIA και ATI αντίστοιχα. Η ATI ονομάζει την καινούρια αυτή τεχνολογία AMD APP, όπου τα αρχικά APP σημαίνουν Accelerated Parallel Processing. Και οι δύο αρχιτεκτονικές, αν και αναπτύχθηκαν ξεχωριστά, επιτρέπουν τον παράλληλο προγραμματισμό των GPUs για οποιαδήποτε εφαρμογή. Αυτό είναι γνωστό και ως General-Purpose Computing on Graphics Processing ή GPGPU.

Ακόμα ένα σημαντικό πλεονέκτημα των GPUs συγκριτικά με τις CPUs είναι το χαμηλό κόστος κτήσης. Η ραγδαία αύξηση των δυνατοτήτων των GPUs τα τελευταία χρόνια οφείλεται σε μεγάλο βαθμό στις απαιτήσεις της παιχνιδοβιομηχανίας για καλύτερα και πιο αληθοφανή γραφικά. Οπότε, δεν θα αρκούσε η δημιουργία πολύ ισχυρών GPUs αν το αγοραστικό κοινό δεν ήταν σε οικονομική θέση να της αποκτήσει. Στη σημερινή πραγματικότητα, μια GPU μπορεί να έχει εντυπωσιακές αποδόσεις, ενώ το κόστος της είναι ένα κλάσμα της τιμής κτήσης ενός αντίστοιχου υπολογιστικού συστήματος.

1.2 Ανάπτυξη εφαρμογών σε Κάρτες Γραφικών

Ο συνδυασμός της μεγάλης υπολογιστικής ισχύος, του εύκολου πλέον προγραμματισμού καθώς και του χαμηλού κόστους κτήσης οδήγησε την επιστημονική κοινότητα στη διερεύνηση των καινούριων δυνατοτήτων που προσφέρουν οι GPUs σε όλους τους τομείς της έρευνας και των εφαρμογών. Τα αποτελέσματα είναι εντυπωσιακά, καθώς παρατηρούνται επιταχύνσεις εφαρμογών από 10 έως και πάνω από 200 φορές, ανάλογα με το είδος της εκάστοτε εφαρμογής και τον βαθμό παραλληλοποίησης της. Ακόμη, σημαντικό είναι και η καινούρια αυτή τεχνολογία δεν απευθύνεται σε ένα μόνο τομέα, όπως είναι η υπολογιστική ρευστοδυναμική η οποία εξετάζεται στην παρούσα εργασία, αλλά σε οποιαδήποτε τομέα που χρειάζεται περισσότερη υπολογιστική ισχύ.

Ενδεικτικά αναφέρονται μερικές δημοσιεύσεις από τη βιβλιογραφία, μαζί με την επιτάχυνση που έχει επιτευχθεί στην εκάστοτε εφαρμογή, από διάφορους τομείς. Έχουν αναπτυχθεί:

- Λογισμικό επεξεργασίας φωτογραφιών αξονικού τομογράφου με χρήση GPU με χρονική επιτάχυνση 14 φορές.[2]
- Γεννήτρια ψευδοτυχαίων αριθμών με επιτάχυνση πάνω από 33 φορές.[3]

- Επιλύτης ροής υγρών κρυστάλλων με επιτάχυνση πάνω από 50 φορές.[4]
- Μοντέλο πρόβλεψης κινδύνου για την αποθήκευση CO_2 στο υπέδαφος με επιτάχυνση πάνω από 60 φορές.[5]
- Μοντέλο προσομοίωσης μεταφοράς νετρονίων με χρήση της μεθόδου Monte Carlo και επιλύτης μεταφοράς θερμότητας με επιτάχυνση μεγαλύτερη από 100 φορές.[6]

Ακόμα αναφέρονται και μερικές δημοσιεύσεις από την βιβλιογραφία από τον τομέα της υπολογιστικής ρευστοδυναμικής. Έχουν αναπτυχθεί:

- Επιλύτης εξισώσεων Navier-Stokes για μόνιμες και μη μόνιμες τυρβώδης ροές σε μη δομημένα και υβριδικά πλέγματα με επιτάχυνση έως και 46 φορές.[9]
- Επιλύτης εξισώσεων Euler για ροές γύρω από υπερηχητικό αεροσκάφος με επιτάχυνση πάνω από 40 φορές.[7]
- Προσομοιωτής ροών μέσα σε κτήρια με χρήση γρήγορων μοντέλων υπολογιστικής ρευστοδυναμικής με επιτάχυνση πάνω από 30 φορές.[8]
- Προσομοιωτής διδιάστατου μοντέλου παλίρροιας με επιτάχυνση έως και 88 φορές.[10]

1.3 Δομή της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη ενός επιλύτη εξισώσεων Euler για διδιάστατες ροές σε κάρτες γραφικών χρησιμοποιώντας την μέθοδο τεμνόμενων κυψελών για την πλεγματοποίηση του χωρίου.

Η εργασία είναι δομημένη στα ακόλουθα κεφάλαια:

- Στο **κεφάλαιο 2** παρουσιάζεται η μαθηματική διατύπωση των διδιάστατων εξισώσεων Euler για χρονικά μόνιμη ροή ατριβούς συμπιεστού ρευστού καθώς και η μέθοδος που χρησιμοποιήθηκε για τη δημιουργία του πλέγματος του υπολογιστικού χωρίου.
- Στο **κεφάλαιο 3** παρουσιάζεται η αρχιτεκτονική CUDA καθώς και τα χαρακτηριστικά της κάρτας γραφικών που χρησιμοποιήθηκε (Nvidia GeForce GTX 670).
- Στο **κεφάλαιο 4** παρουσιάζεται ο επιλύτης (σε GPUs) που αναπτύχθηκε και εξηγείται η λογική της ανάπτυξής του.
- Στο **κεφάλαιο 5** παρουσιάζονται τα αποτελέσματα που προέκυψαν από τη χρήση του παραπάνω επιλύτη και συγκρίνονται με τα αποτελέσματα αντίστοιχου πιστοποιημένου επιλύτη σχεδιασμένου να εκτελείται στη CPU.
- Στο **κεφάλαιο 6** γίνεται ανακεφαλαίωση των όσων έγιναν στην παρούσα διπλωματική εργασία και γίνονται κάποιες προτάσεις για μελλοντική βελτιστοποίηση του επιλύτη.

Κεφάλαιο 2

Διατύπωση - Διακριτοποίηση των Εξισώσεων Ροής

Στο κεφάλαιο αυτό παρουσιάζονται οι εξισώσεις ροής σε συντηρητική γραφή για χρονικά μόνιμη, δισδιάστατη ροή ατριβούς συμπιεστού ρευστού, δηλαδή οι εξισώσεις Euler[11]. Επιπλέον, παρουσιάζεται ο τρόπος αριθμητικής επίλυσης των εξισώσεων αυτών μέσω μεθόδου χρονοπροέλασης (time marching) με σκοπό να εφαρμοστεί σε καρτεσιανά πλέγματα που χρησιμοποιούνται στη μέθοδο τεμνόμενων κυψελών (cut-cells).

2.1 Συντηρητική Μορφή των Εξισώσεων Euler

Οι εξισώσεις Euler σε συντηρητική γραφή για χρονικά μόνιμη ροή συμπιεστού ρευστού, απουσία βαρυτικών δυνάμεων, στο καρτεσιανό σύστημα αναφοράς, για δισδιάστατες ροές είναι:

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \overrightarrow{f_x}}{\partial x} + \frac{\partial \overrightarrow{f_y}}{\partial y} = \vec{0} \qquad \textbf{(2.1)}$$

Τα διανύσματα των συντηρητικών (conservative) μεταβλητών $\vec{U}$ και της μη-συνεκτικής ροής (flux) ανά κατεύθυνση $\overrightarrow{f_r}$ είναι:

$$\overrightarrow{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad f_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(\rho E + p) \end{bmatrix} \quad f_y = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(\rho E + p) \end{bmatrix} \qquad \textbf{(2.2)}$$

όπου ρ η πυκνότητα του ρευστού, u και v οι συνιστώσες της ταχύτητας κατά τη x και y κατεύθυνση, αντίστοιχα, E η ολική ενέργεια ανά μονάδα μάζας, e η εσωτερική ενέργεια ανά μονάδα μάζας και p η πίεση του ρευστού. Ο χρονικός όρος που υπάρχει στις εξισώσεις αποτελεί έναν ψευδοχρονικό όρο και έχει προστεθεί για την εκμετάλλευση των ιδιοτήτων των υπερβολικών συστημάτων και για την εφαρμογή μιας μεθόδου χρονοπροέλασης. Η ολική ενέργεια ανά μονάδα μάζας εκφράζεται από την σχέση:

$$\mathbf{E} = \mathbf{e} + \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{u^2} + \mathbf{v^2}) \quad \textbf{(2.3)}$$

και σχετίζεται με την πίεση ως εξής:

$$\boldsymbol{E} = \frac{\boldsymbol{p}}{\boldsymbol{\rho(\gamma-1)}} + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{u^2} + \boldsymbol{v^2}) \quad \textbf{(2.4)}$$

Επίσης, η ολική ενθαλπία δίνεται από την σχέση:

$$\boldsymbol{H} = \boldsymbol{E} + \frac{\boldsymbol{p}}{\boldsymbol{\rho}} = \frac{\boldsymbol{\gamma p}}{\boldsymbol{\rho(\gamma-1)}} + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{u^2} + \boldsymbol{v^2}) \quad \textbf{(2.5)}$$

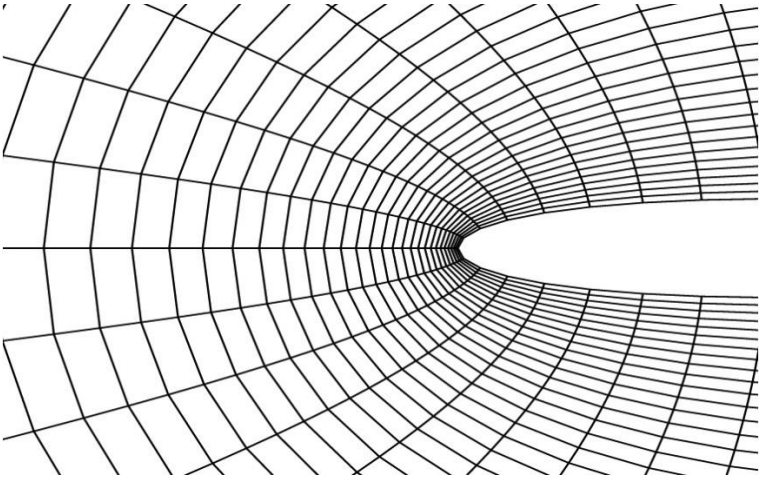
Τέλος, θεωρείται ότι το εργαζόμενο μέσο είναι τέλειο αέριο.

2.2 Διακριτοποίηση του Χωρίου Ροής

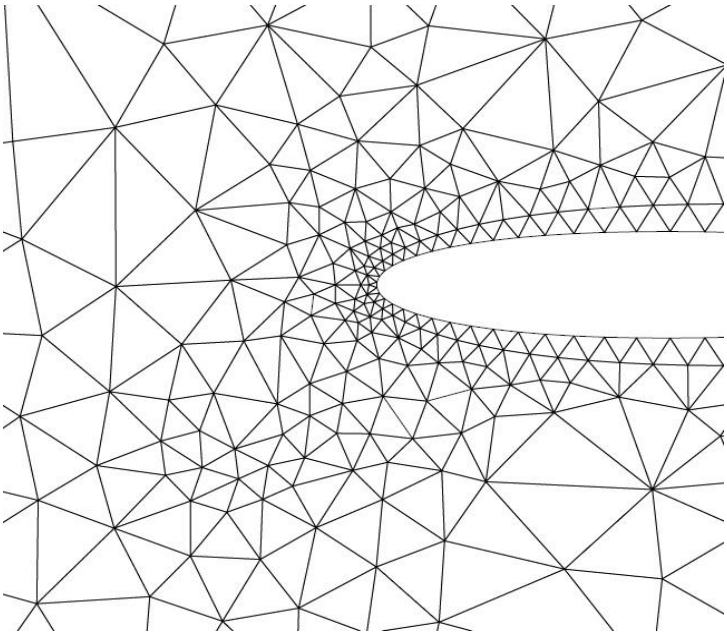
Προκειμένου να εφαρμοστεί η μέθοδος τεμνόμενων κυψελών για την επίλυση των εξισώσεων της σχέσης 2.1, απαιτείται η διακριτοποίηση του υπολογιστικού χωρίου. Ο πιο συνήθης τρόπος διακριτοποίησης είναι η δημιουργία πλέγματος που προσαρμόζεται γύρω από το σώμα, γύρω από το οποίο μελετάται η ροή (π.χ. αεροτομή, πτερύγιο συμπιεστή/στροβίλου). Γνωστοί τύποι πλεγμάτων που προκύπτουν από αυτόν τον τρόπο διακριτοποίησης είναι τα δομημένα (σχήμα 2.1α), τα μη-δομημένα (σχήμα 2.1β) και τα υβριδικά πλέγματα (συνδυασμός δομημένων και μη-δομημένων) (σχήμα 2.1γ).

Στις περισσότερες βιομηχανικές εφαρμογές η δημιουργία του πλέγματος είναι σύνθετη. Επίσης, στις περιπτώσεις κινούμενων σωμάτων σε κάθε χρονικό βήμα απαιτείται η δημιουργία ή αναπροσαρμογή του πλέγματος, κάτι που απαιτεί επιπλέον υπολογιστικό χρόνο.

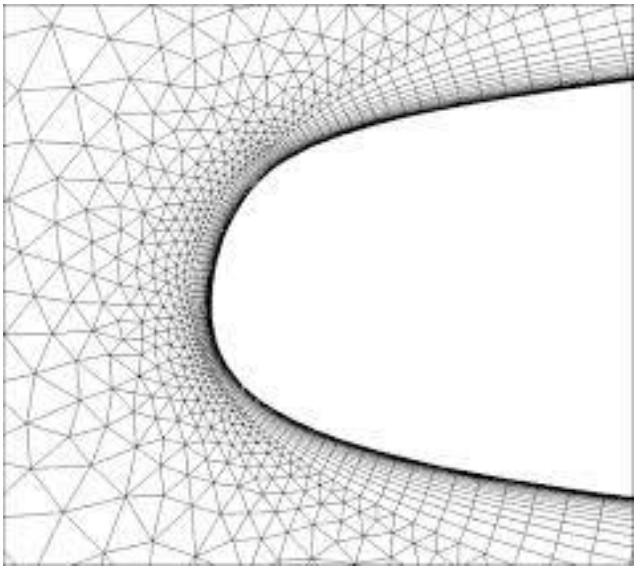
Για τους παραπάνω λόγους, προέκυψε ένας άλλος τρόπος διακριτοποίησης του υπολογιστικού χωρίου. Ο τρόπος αυτός ονομάζεται μέθοδος εμβαπτιζόμενων ορίων (immersed boundary method) και χρησιμοποιεί ένα καρτεσιανό πλέγμα που δεν προσαρμόζεται γύρω από το υπό μελέτη σώμα (σχήμα 2.2). Συνεπώς, κάποια από τις υπολογιστικές κυψέλες του πλέγματος τέμνονται από το σύνορο του σώματος. Έχουν αναπτυχθεί αρκετοί τρόποι μεταχείρισης αυτών των τεμνόμενων κυψελών. Αυτοί οι τρόπου μπορούν να κατηγοριοποιηθούν: α) στη συνεχή προσέγγιση (continuous forcing approach) και β) τη διακριτή προσέγγιση (discrete forcing approach).



(α)

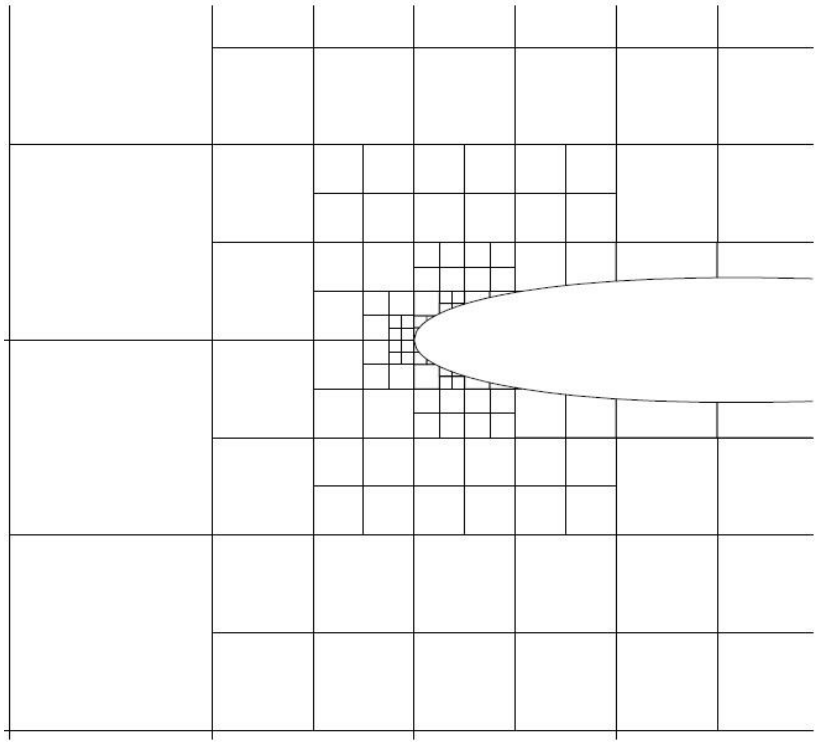


(β)



(γ)

Σχήμα 2.1: Διδιάστατα πλέγματα. (α) Δομημένο[24], (β) Μη-δομημένο[24], (γ) Υβριδικό πλέγμα[31].



Σχήμα 2.2: Καρτεσιανό διδιάσταστο πλέγμα [24].

2.2.1 Μέθοδος Εμβαπτιζόμενων Ορίων [25]

Από όσα έχουν αναφερθεί προηγουμένως, γίνεται αντιληπτό πως ένα από τα πλεονεκτήματα της μεθόδου εμβαπτιζόμενων ορίων είναι η ευκολία στη γένεση πλέγματος, καθώς δημιουργείται ένα καρτεσιανό πλέγμα, ακόμα και στην περίπτωση που το υπό μελέτη σώμα έχει πολύπλοκη γεωμετρία. Άλλο ένα πλεονέκτημα αυτής της μεθόδου είναι ότι μπορεί να χειρίζεται κινούμενα όρια, καθώς δεν απαιτείται το καρτεσιανό πλέγμα να «παραμορφωθεί».

Τα παραπάνω πλεονεκτήματα έχουν ως αποτέλεσμα, η μέθοδος αυτή να χρησιμοποιεί λιγότερη μνήμη και υπολογιστική ισχύ σε σχέση με τις συνήθεις μεθόδους. Πιο συγκεκριμένα, σε σύγκριση με τα δομημένα καμπυλόγραμμα πλέγματα, τα καρτεσιανά πλέγματα δεν απαιτούν τον υπολογισμό της συναλλοίωτης (ή της ανταλλοίωτης) διανυσματικής βάσης σε κάθε κόμβο του, με αποτέλεσμα τη μείωση των αριθμητικών υπολογισμών. Σε σχέση με τα μη-δομημένα καμπυλόγραμμα πλέγματα, τα καρτεσιανά πλέγματα αποτελούνται από κυψέλες απλούστερης γεωμετρίας (τετράγωνα). Κάτι τέτοιο τα καθιστά πιο εύχρηστα καθώς ο υπολογισμός των γεωμετρικών ποσοτήτων είναι απλούστερος.

Ένα μειονέκτημα της μεθόδου αυτής είναι η δυσκολία επιβολής των οριακών συνθηκών σε σύγκριση με τις παραδοσιακές μεθόδους. Η μη ακριβής επιβολή τους έχει επιπτώσεις στην ακρίβεια επίλυσης των εξισώσεων ροής. Επιπλέον, για την καλύτερη επιβολή τους, απαιτείται περαιτέρω πύκνωση του πλέγματος κοντά στο σύνορο του σώματος, με αποτέλεσμα την αύξηση του μεγέθους του πλέγματος.

2.2.2 Συνεχής Προσέγγιση (continuous forcing approach)

Σε αυτήν τη μέθοδο, δεν αρκεί μόνο η διακριτοποίηση των εξισώσεων ροής σε καρτεσιανό πλέγμα. Χρειάζεται να τροποποιηθούν οι εξισώσεις ροής προσθέτοντας μια συνάρτηση επιβολής (forcing function), η οποία αναπαράγει την επίδραση του ορίου.

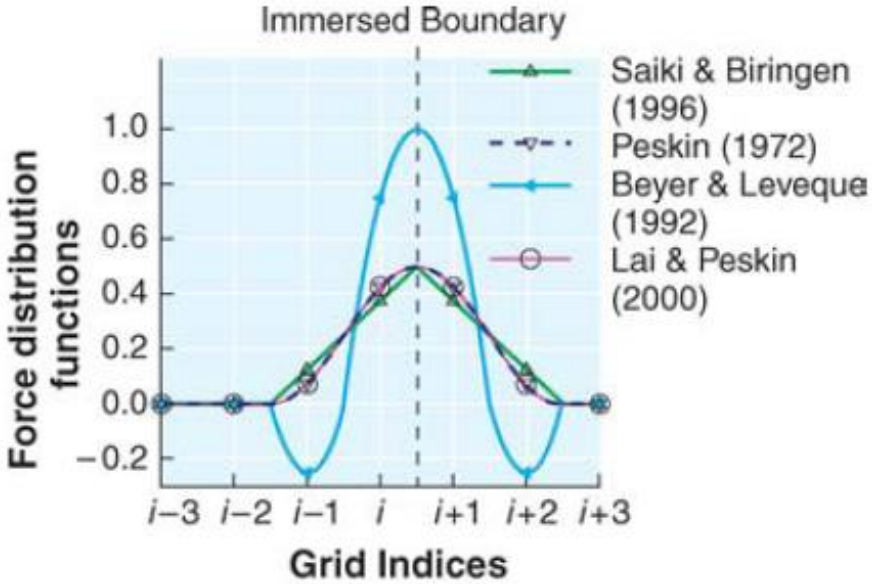
Η προσθήκη αυτής της συνάρτησης στις εξισώσεις ροής, έχει ως αποτέλεσμα να απαιτείται η επίλυση των εξισώσεων σε όλο το υπολογιστικό χωρίο. Στο σχήμα 2.3 φαίνονται διάφορες παραλλαγές κατανομών της συνάρτησης αυτής.

Μερικές μέθοδοι που βασίζονται στην συνεχή προσέγγιση είναι:

- Ελαστικό όριο (Elastic boundary), εισήχθηκε από τον Peskin το 1972[25].
- Άκαμπτο όριο (Rigid boundary), χρησιμοποιήθηκε πρώτη φορά από τον Goldstein[25].
- Μέθοδος Κατανεμημένου Πολλαπλασιαστή Lagrange (Distributed Lagrange multiplier method - DLM), προτάθηκε από τον Glowinski[25].
- Μέθοδος Εμβαπτιζόμενης Διεπαφής (Immersed interface method - IIM), χρησιμοποιήθηκε σε προβλήματα με ελαστικές μεμβράνες από τον Lee[25].

Ένα πλεονέκτημα της συνεχούς προσέγγισης είναι ότι οι παραπάνω μέθοδοι είναι ανεξάρτητες από τη χωρική διακριτοποίηση κάτι που έχει ως συνέπεια την εύκολη προσαρμογή της σε έναν ήδη υπάρχοντα επιλύτη π.χ. εξισώσεων Navier-Stokes.

Ένα μειονέκτημα της συνεχούς προσέγγισης είναι ότι η εξομάλυνση της συνάρτησης επιβολής οδηγεί εκ φύσεως στην μη αποτελεσματική αναπαράσταση του συνόρου, με αποτέλεσμα οι μέθοδοι αυτές να μην είναι κατάλληλες για ροές με υψηλό αριθμό Reynolds. Άλλο ένα μειονέκτημα, που αναφέρθηκε και προηγουμένως, είναι η απαίτηση επίλυσης των τροποποιημένων εξισώσεων ροής τόσο στις κυψέλες που βρίσκονται εντός του πεδίου ροής όσο και σε αυτές που βρίσκονται εντός του στερεού σώματος.



Σχήμα 2.3: Κατανομές συναρτήσεων επιβολής όπως αναπτύχθηκαν με την πάροδο των χρόνων για τη συνεχή προσέγγιση. [25]

2.2.3 Διακριτή Προσέγγιση (discrete forcing approach)

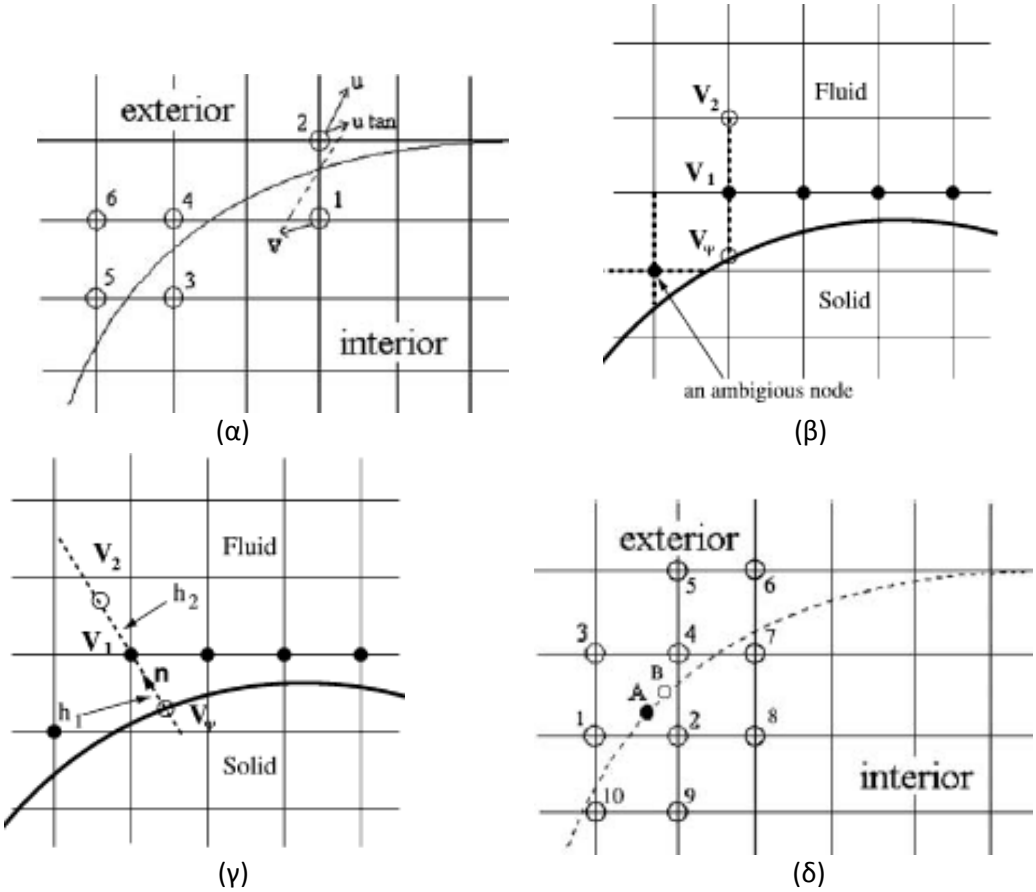
Σε αντίθεση με τη συνεχή προσέγγιση, η διακριτή είναι πιο κατάλληλη για ροές με υψηλό αριθμό Reynold, γιατί επιβάλλεται η ταχύτητα στο όριο χωρίς να χρειάζεται να υπολογιστεί κάποιος άλλος όρος για να προστεθεί στις συνοριακές συνθήκες.

Διάφορες μέθοδοι που αναπτύχθηκαν στηριζόμενες στη διακριτή προσέγγιση είναι:

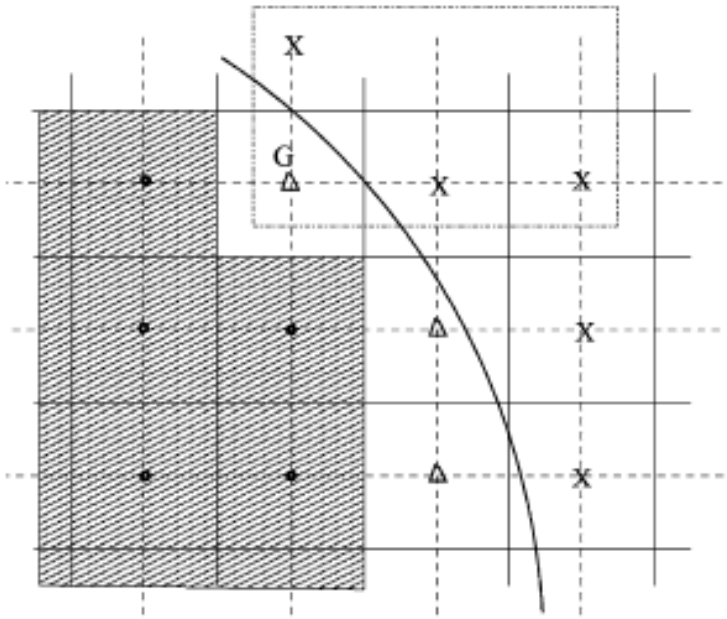
- Μέθοδος άμεσης επιβολής (direct forcing), αναπτύχθηκε από τον Mohd-Yusof και η συνάρτηση επιβολής καθορίζεται από την διαφορά μεταξύ της ταχύτητας στο όριο μετά από παρεμβολή και της επιθυμητής ταχύτητας. Παραλλαγές της μεθόδου αυτής έχουν αναπτυχθεί από τους Fadlun, Balaras, Gilmanov και Zhang. Στο σχήμα 2.4 παρουσιάζονται μερικά από τα σχήματα παρεμβολής που αναφέρθηκαν[25].
- Ο Ikeno ανέπτυξε ένα σχήμα για την συνοχή μεταξύ της πίεσης και της ταχύτητας, ώστε να επιτυγχάνεται η επιθυμητή συνθήκη πάνω στον τοίχο, όπως η συνθήκη μη-ολίσθησης ή η μη-μηδενική ταχύτητα[25].
- Ghost-cell (κυψέλη «φάντασμα»), αναπτύχθηκε από τον Tseng. Με τη μέθοδο αυτή, επιχειρείται υψηλότερης τάξης αναπαράσταση του ορίου χρησιμοποιώντας μια ζώνη με κυψέλες «φάντασμα» μέσα στο σώμα. Κυψέλη «φάντασμα» καλείται η κυψέλη που βρίσκεται μέσα στο στερεό σώμα και έχει μία τουλάχιστον γειτονική κυψέλη που βρίσκεται μέσα στο ρευστό. Για κάθε τέτοια κυψέλη πρέπει να γίνει ένα σχήμα παρεμβολής, το οποίο ενσωματώνει έμμεσα τις συνοριακές συνθήκες. Στο σχήμα 2.5 παρουσιάζεται αυτή η μέθοδος[25].

Εκτός του ότι μπορεί να χειριστεί καλύτερα ροές με υψηλό αριθμό Reynolds, η διακριτή προσέγγιση δεν εισάγει επιπλέον περιορισμούς σταθερότητας στην αναπαράσταση των στερεών σωμάτων, καθώς δεν απαιτείται από τον χρήστη να προσδιορίσει παραμέτρους για τις συναρτήσεις επιβολής. Ένα μειονέκτημα, σε σχέση με τη συνεχή προσέγγιση, είναι ότι εξαρτάται σε μεγάλο βαθμό από τη μέθοδο διακριτοποίησης που χρησιμοποιείται. Όμως, η επιλογή της μεθόδου διακριτοποίησης επιτρέπει τον έλεγχο της ακρίβειας της αριθμητικής μεθόδου και της ευστάθειας του επιλύτη.

Τέλος, ένα σημαντικό μειονέκτημα, τόσο της διακριτής όσο και της συνεχούς μεθόδου, είναι το γεγονός πως αυτές οι δύο μέθοδοι δεν είναι συντηρητικές, καθώς ένα μέρος της ροής καταφέρνει να διαπεράσει το στερεό τοίχωμα, παρά την επιβολή κατάλληλων συνθηκών στις κυψέλες που τέμνονται από αυτό. Δηλαδή, δεν διατηρείται πλήρως η μάζα στο στερεό τοίχωμα παρά τις συναρτήσεις επιβολής, με αποτέλεσμα μέρος της ροής να εισέρχεται στο στερεό. Το μειονέκτημα αυτό έρχεται να καλύψει η μέθοδος τεμνόμενων κυψελών (cut-cells), που αναλύεται στην παράγραφο 2.3.



Σχήμα 2.4: Σχήματα παρεμβολής για τη διακριτή προσέγγιση. (α) Mohd-Yusof, (β) Fadlun, (γ) Balaras, (δ) Zhang. [25]



Σχήμα 2.5: Υπολογιστικό χωρίο διακριτοποιημένο με τη μέθοδο ghost-cell. Χ, το κέντρο των κυψελών εντός του ρευστού και Δ, το κέντρο των κυψελών «φάντασμα». [25]

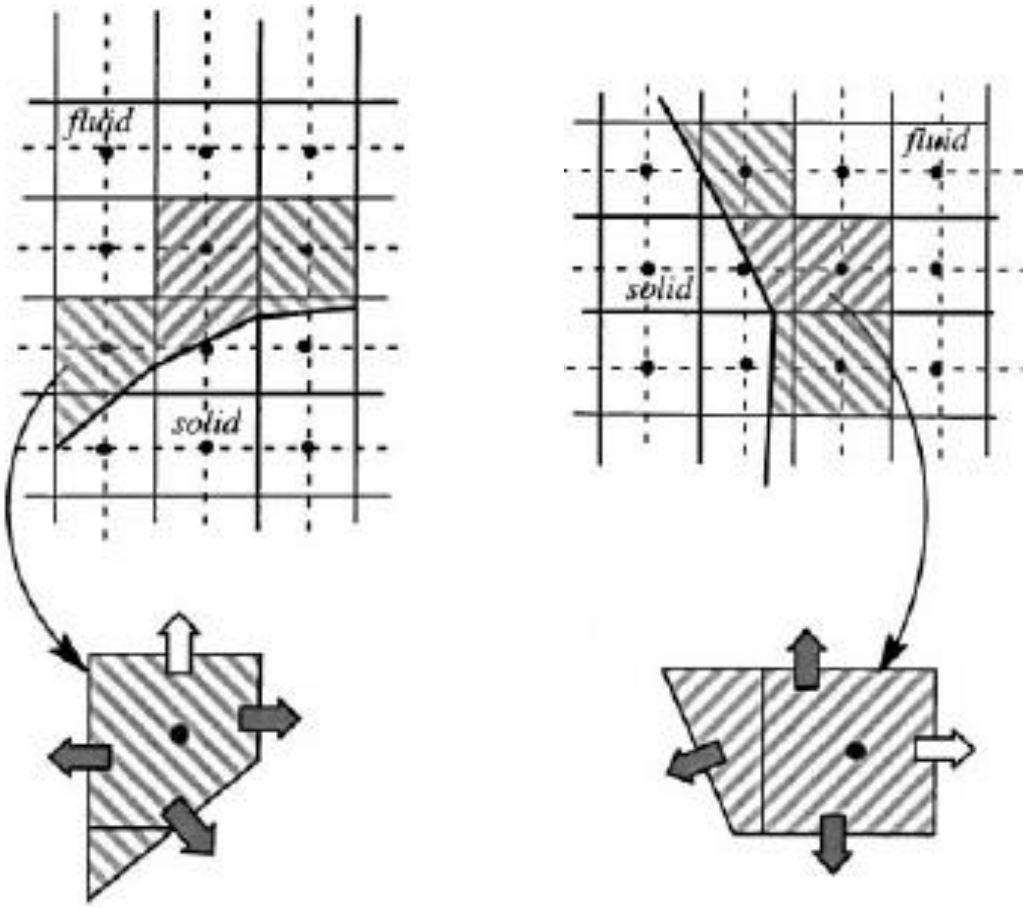
2.3 Cut-cells Method (Μέθοδος Τεμνόμενων Κυψελών)

Σε αντίθεση με τη συνεχή, στη μέθοδο τεμνόμενων κυψελών δεν επιβάλλεται κάποια συνάρτηση επιβολής για τις οριακές συνθήκες. Αντιθέτως, απαιτείται η περικοπή των καρτεσιανών κυψελών (η μέθοδος αυτή παλιά ονομαζόταν μέθοδος καρτεσιανού πλέγματος - Cartesian grid method) στο όριο, προκειμένου να δημιουργηθούν κυψέλες με το σχήμα της επιφάνειας του ορίου. Η αναμόρφωση αυτή, όμως, έχει ως αποτέλεσμα τη δημιουργία πολύ μικρών κυψελών, κάτι που έχει αντίκτυπο στην ευστάθεια του αριθμητικού σχήματος που θα χρησιμοποιηθεί.[25] Μια λύση σε αυτό το πρόβλημα είναι η συγχώνευση αυτών των κυψελών με γειτονικά τους. Στο σημείο αυτό επισημαίνεται πως η λύση αυτή δεν υιοθετείται στην παρούσα εργασία.

Η μέθοδος αυτή προτάθηκε από τον Ye[25] και χρησιμοποιεί σχήμα κεντρικών διαφορών για τις παρεμβολές γύρω από το όριο κάτι που δίνει δεύτερης τάξης ακρίβεια στη χωρική παράγωγο. Στην περίπτωση που το κέντρο της κυψέλης βρίσκεται εντός στερεού, επιλέγεται συνήθως να ενώνεται το τμήμα που βρίσκεται εντός του ρευστού με γειτονική κυψέλη. Όλη αυτή η αναμόρφωση έχει ως αποτέλεσμα να δημιουργούνται κυψέλες πολυγωνικού σχήματος. Στο σχήμα 2.6 παρουσιάζονται οι δύο παραπάνω περιπτώσεις αναμόρφωσης του καρτεσιανού πλέγματος.

Η πρόταση του Ye, για τον υπολογισμό των ροών (fluxes) στην πλευρά της κυψέλης που είναι στο όριο, προτείνει μια πολυωνυμική κατανομή συναρτήσει των συντεταγμένων x και y. Ο Udaykumar[25] προτείνει μια Eulerian-Lagrangian προσέγγιση για την περίπτωση κινούμενου συνόρου. Άλλες παραλλαγές της μεθόδου αυτής περιγράφονται στη μεταπτυχιακή διπλωματική εργασία [25] του πανεπιστημίου του Groningen. Στην παρούσα εργασία, οι ροές στις πλευρές των κυψελών υπολογίζονται με τα μεγέθη που βρίσκονται αποθηκευμένα στο βαρύκεντρο κάθε κυψέλης, χωρίς να πραγματοποιείται κάποια παρεμβολή (σχήμα πρώτης τάξης ακρίβειας).

Ένα βασικό πλεονέκτημα της μεθόδου τεμνόμενων κυψελών είναι ότι οι οριακές συνθήκες επιβάλλονται με μεγαλύτερη ακρίβεια σε σχέση με άλλες μεθόδους εμβαπτιζόμενων ορίων. Επιπλέον, η μέθοδος αυτή βασίζεται στη μέθοδο πεπερασμένων όγκων καθώς εγγυάται την διατήρηση της μάζας και της ορμής ακόμα και στην περιοχή γύρω από το όριο. Όμως, η εφαρμογή των οριακών συνθηκών σε ακανόνιστες κυψέλες απαιτεί ειδικούς χειρισμούς, κάτι που οδηγεί σε πολύπλοκους προγραμματιστικούς κώδικες. Τέλος, όταν χρησιμοποιείται η μέθοδος αυτή, πρέπει να αποφεύγεται ο σχηματισμός μικρών κυψελών, κάτι που όπως έχει ήδη τονιστεί, οδηγεί σε προβλήματα ευστάθειας.



(α) (β)

Σχήμα 2.6: Αναμόρφωση καρτεσιανού πλέγματος στη μέθοδο τεμνόμενων κυψελών όταν το κέντρο της κυψέλης που κόβεται από το όριο βρίσκεται (α) εντός ρευστού και (β) εντός στερεού. [25]

2.4 Διακριτοποίηση των Εξισώσεων Ροής

Χρησιμοποιώντας τη μέθοδο τεμνόμενων κυψελών, στο υπολογιστικό χωρίο έχουν δημιουργηθεί καρτεσιανές κυψέλες όπως φαίνονται στο σχήμα 2.7. Ολοκληρώνοντας τις εξισώσεις 2.1 στο εμβαδόν τυχαίας καρτεσιανής κυψέλης για μόνιμη ροή και εμβαδόν Ω, προκύπτει:

$$\int_{\Omega} \frac{\partial \overrightarrow{f_k}}{\partial k} d\Omega = 0 \qquad \textbf{(2.6)}$$

όπου k=x ή y, δηλαδή για κάθε καρτεσιανή κατεύθυνση. Χρησιμοποιώντας το θεώρημα Green-Gauss, το παραπάνω επιφανειακό ολοκλήρωμα μετατρέπεται στο επικαμπύλιο ολοκλήρωμα:

$$\oint_S \overrightarrow{f_\kappa} \cdot n_k \cdot dS = 0 \qquad \textbf{(2.7)}$$

όπου n_k είναι το κάθετο προς τα έξω μοναδιαίο διάνυσμα σε κάθε πλευρά της καρτεσιανής κυψέλης. Η 2.7 προσεγγίζεται από την:

$$\sum_j^{\pi\lambda\varepsilon\upsilon\rho\acute{\varepsilon}\varsigma} \overrightarrow{\boldsymbol{f_{k_j}}} \cdot \boldsymbol{n_{k_j}} \cdot \boldsymbol{\Delta S_j} = \boldsymbol{0},\ \forall \boldsymbol{A} \qquad \textbf{(2.8)}$$

Από την εξίσωση 2.8 παρατηρούμε ότι πρέπει να είναι γνωστές οι ροές πάνω στην κάθε πλευρά της κυψέλης. Για το λόγο αυτό χρησιμοποιείται το σχήμα Roe για να υπολογιστούν οι ροές πάνω σε κάθε πλευρά.

L και R είναι τα σημεία ακριβώς αριστερά και δεξιά της κοινής πλευράς, αντίστοιχα, όπως φαίνεται και στο σχήμα 2.8. Στην κοινή πλευρά εφαρμόζεται η σχέση:

$$\overrightarrow{\boldsymbol{f_k^{ROE}}} = \frac{\boldsymbol{1}}{\boldsymbol{2}}\left(\overrightarrow{\boldsymbol{f_k^R}} + \overrightarrow{\boldsymbol{f_k^L}}\right) - \frac{\boldsymbol{1}}{\boldsymbol{2}}\left|\widetilde{\underline{\boldsymbol{A_k}}}\right|\left(\overrightarrow{\boldsymbol{U_k^R}} - \overrightarrow{\boldsymbol{U_k^L}}\right) \qquad \textbf{(2.9)}$$

όπου A είναι το Ιακωβιανό μητρώο του συστήματος. Η ύπαρξη της περισπωμένης πάνω από το μητρώο τονίζει ότι στον υπολογισμό του θα χρησιμοποιηθούν τα μέσα κατά Roe μεγέθη, για τα οποία θα αναφερθούμε στη συνέχεια. Όπως είναι γνωστό, για κάθε στοιχείο του Ιακωβιανού μητρώου, ισχύει η σχέση:

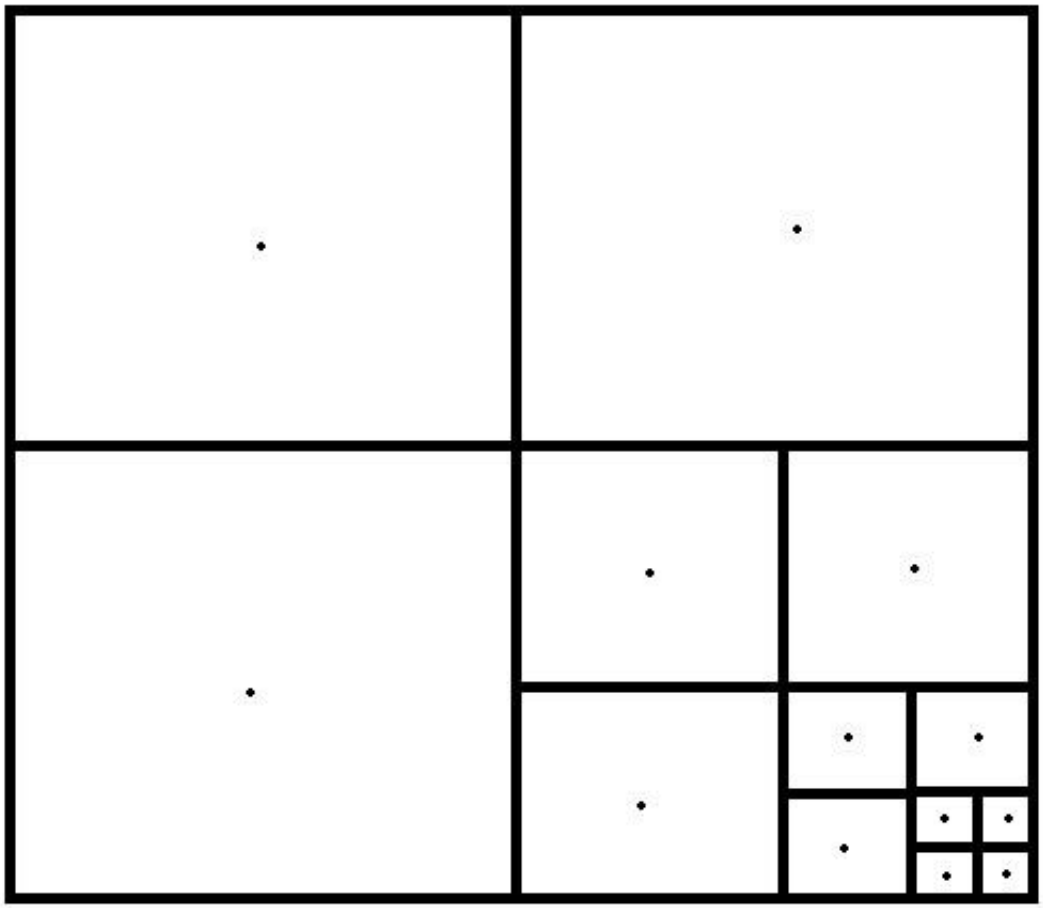
$$\boldsymbol{A_{ijk}} = \frac{\boldsymbol{\partial f_{ik}}}{\boldsymbol{\partial U_j}}, \boldsymbol{i} \in [\boldsymbol{1}, \boldsymbol{4}]\ \ \boldsymbol{j}\ \in [\boldsymbol{1}, \boldsymbol{4}]\ \ \boldsymbol{k} \in [\boldsymbol{x}, \boldsymbol{y}] \qquad \textbf{(2.10)}$$

Αφού κάνουμε τις απαραίτητες πράξεις, καταλήγουμε στις σχέσεις υπολογισμού του Ιακωβιανού μητρώου για την x και y κατεύθυνση.

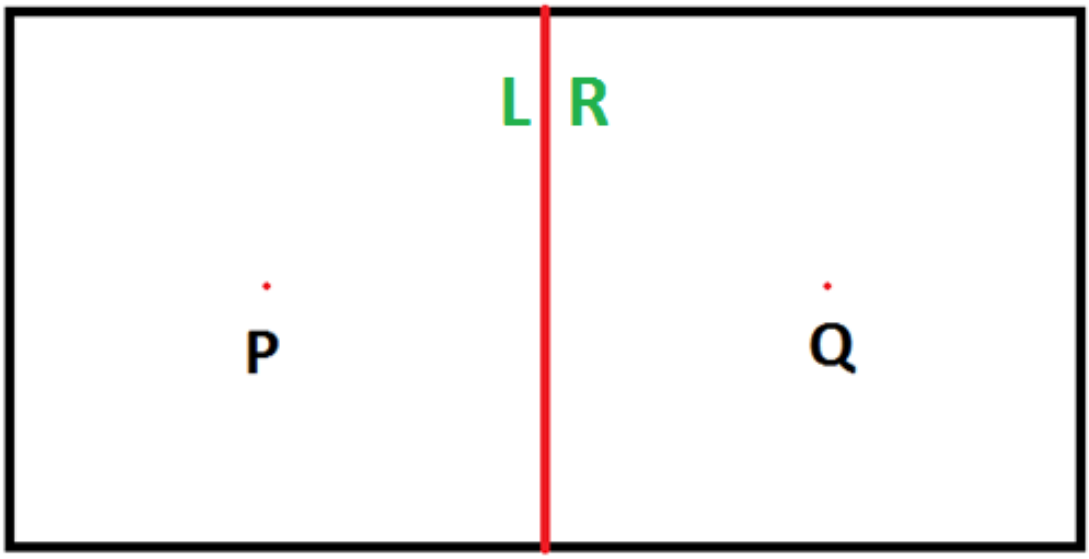
$$\boldsymbol{A_x} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\ \frac{\gamma-3}{2}\cdot\boldsymbol{u^2} + \frac{\gamma-1}{2}\cdot\boldsymbol{v^2} & (\boldsymbol{3}-\boldsymbol{\gamma})\boldsymbol{u} & -(\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{v} & \boldsymbol{\gamma}-\boldsymbol{1} \\ -\boldsymbol{u}\cdot\boldsymbol{v} & \boldsymbol{v} & \boldsymbol{u} & \boldsymbol{0} \\ -\boldsymbol{\gamma}\cdot\boldsymbol{u}\cdot\boldsymbol{E} + (\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{u}(\boldsymbol{u^2}+\boldsymbol{v^2}) & \boldsymbol{\gamma E} - \frac{\gamma-1}{2}\cdot(\boldsymbol{3u^2}+\boldsymbol{v^2}) & -(\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{uv} & \boldsymbol{\gamma}\cdot\boldsymbol{u} \end{bmatrix}$$

και

$$\boldsymbol{A_y} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \\ -\boldsymbol{u}\cdot\boldsymbol{v} & \boldsymbol{v} & \boldsymbol{u} & \boldsymbol{0} \\ \frac{\gamma-3}{2}\cdot\boldsymbol{v^2} + \frac{\gamma-1}{2}\cdot\boldsymbol{u^2} & -(\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{u} & (\boldsymbol{3}-\boldsymbol{\gamma})\boldsymbol{v} & \boldsymbol{\gamma}-\boldsymbol{1} \\ -\boldsymbol{\gamma}\cdot\boldsymbol{v}\cdot\boldsymbol{E} + (\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{v}(\boldsymbol{u^2}+\boldsymbol{v^2}) & -(\boldsymbol{\gamma}-\boldsymbol{1})\boldsymbol{uv} & \boldsymbol{\gamma E} - \frac{\gamma-1}{2}\cdot(\boldsymbol{u^2}+\boldsymbol{3v^2}) & \boldsymbol{\gamma}\cdot\boldsymbol{v} \end{bmatrix} \qquad \textbf{(2.11)}$$



Σχήμα 2.7: Το υπολογιστικό χωρίο έτσι όπως έχει σχηματιστεί με τη μέθοδο τεμνόμενων κυψελών.



Σχήμα 2.8: Δύο γειτονικές καρτεσιανές κυψέλες, όπου υπολογίζονται οι ροές μάζας, ορμής και ενέργειας στην κόκκινη κοινή πλευρά τους. P και Q είναι τα βαρύκεντρα των δύο αυτών κυψελών και L και R τα σημεία ακριβώς αριστερά και δεξιά της κοινής πλευράς, αντίστοιχα.

Τα μέσα κατά Roe [26] μεγέθη στη διεπιφάνεια των δύο κυψελών υπολογίζονται από τις σχέσεις 2.12 και χρησιμοποιούνται για τον υπολογισμό του $\tilde{A}$ (σχέση 2.10).

$$\tilde{\rho} = \sqrt{\rho_L \cdot \rho_R}$$

$$\tilde{u} = \frac{u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\tilde{v} = \frac{v_L\sqrt{\rho_L} + v_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\tilde{H} = \frac{H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\tilde{c} = \sqrt{(\gamma - 1)\left(\tilde{H} - \frac{\tilde{u}^2+\tilde{v}^2}{2}\right)} \qquad \textbf{(2.12)}$$

Υπενθυμίζεται η σχέση $c = \sqrt{\gamma\frac{p}{\rho}}$.

Όπως επισημάνθηκε προηγουμένως, οι μεταβλητές της ροής είναι γνωστές στο βαρύκεντρο κάθε κυψέλης και όχι πάνω στις πλευρές του. Οπότε, για να μπορέσουμε να χρησιμοποιήσουμε το σχήμα Roe, πρέπει να προσδιορίσουμε τις τιμές των μεταβλητών στα σημεία L και R. Αυτό μπορεί να συμβεί χρησιμοποιώντας σχήματα πρώτης, δεύτερης ή ανώτερης τάξης ακρίβειας. Για πρώτης τάξης ακρίβεια ισχύει: $\overrightarrow{U_L} = \overrightarrow{U_P}$, ενώ για δεύτερης τάξης ισχύει: $\overrightarrow{U_L} = \overrightarrow{U_P} + \frac{\partial\overrightarrow{U_P}}{\partial x}\Delta x + \frac{\partial\overrightarrow{U_P}}{\partial y}\Delta y$. Αντίστοιχες σχέσεις ισχύουν και για το σημείο R. Ο επιλύτης της παρούσας εργασίας χρησιμοποιεί πρώτης τάξης ακρίβεια για τον προσδιορισμό των τιμών των μεταβλητών ροής στα σημεία L και R.

Τέλος, από τη σχέση 2.10 προκύπτει η: $f_{ik} = A_{ijk} \cdot U_j$ **(2.13)**

Επομένως, η σχέση 2.9 μπορεί να γραφτεί στη μορφή:

$$\overrightarrow{f_k^{ROE}} = \frac{1}{2}\left(\underline{A^R} \cdot \overrightarrow{U_k^R} + \underline{A^L} \cdot \overrightarrow{U_k^L}\right) - \frac{1}{2}\left|\widetilde{\underline{A_k}}\right|\left(\overrightarrow{U_k^R} - \overrightarrow{U_k^L}\right) \qquad \textbf{(2.14)}$$

$$\Rightarrow \vec{f}\cdot\vec{n} = \frac{1}{2}\left(\underline{A_n^R} \cdot \overrightarrow{U_R} + \underline{A_n^L} \cdot \overrightarrow{U_L}\right) - \frac{1}{2}|\widetilde{\underline{A}}|\left(\overrightarrow{U_R} - \overrightarrow{U_L}\right) \qquad \textbf{(2.15)}$$

2.5 Αριθμητική Επίλυση

Έχοντας διακριτοποιήσει τις εξισώσεις ροής στη μορφή 2.8, πρέπει να υπολογιστεί το πεδίο $\overrightarrow{\boldsymbol{U}^*}$, το οποίο τις ικανοποιεί. Το αριστερό μέλος της 2.8 ονομάζεται $\overrightarrow{\boldsymbol{R_\iota}} = \sum \overrightarrow{\boldsymbol{f_{\iota j}}} \cdot \vec{\boldsymbol{n}} \cdot \boldsymbol{\Delta S_j}$, όπου i ο μετρητής των κυψελών. Επομένως ισχύει: $\overrightarrow{\boldsymbol{R_\iota}}\left(\overrightarrow{\boldsymbol{U}^*}\right) = \boldsymbol{0}$. **(2.16)**

Έστω $\overrightarrow{\boldsymbol{U}}$ μια αρχική εκτίμηση του πεδίου ροής, επομένως $\overrightarrow{\boldsymbol{R_\iota}}\left(\overrightarrow{\boldsymbol{U}}\right) \neq \boldsymbol{0}$ και $\overrightarrow{\boldsymbol{U}^*} = \overrightarrow{\boldsymbol{U}} + \overrightarrow{\boldsymbol{\Delta U}}$. Εφαρμόζοντας ανάπτυγμα Taylor στην σχέση 2.16 έχουμε:

$$\overrightarrow{\boldsymbol{R_\iota}}\left(\overrightarrow{\boldsymbol{U}^*}\right) = \boldsymbol{0} \Leftrightarrow \overrightarrow{\boldsymbol{R_\iota}}\left(\overrightarrow{\boldsymbol{U}} + \overrightarrow{\boldsymbol{\Delta U}}\right) = \boldsymbol{0} \Leftrightarrow \overrightarrow{\boldsymbol{R_\iota}}|_{\vec{\boldsymbol{U}}} + \frac{\partial \overrightarrow{\boldsymbol{R_\iota}}}{\partial \vec{\boldsymbol{U}}} \cdot \boldsymbol{\Delta \vec{U}} + \boldsymbol{O}(\boldsymbol{\Delta U^2}) = \boldsymbol{0} \quad \textbf{(2.17)}$$

Η σχέση 2.17 είναι ένα σύστημα 4xn εξισώσεων, όπου n ο αριθμός των κυψελών με αγνώστους τη διόρθωση $\Delta\vec{U}$. Η σχέση αυτή για τη κυψέλη με κέντρο P και για πρώτης τάξης ακρίβεια γράφεται:

$$\overrightarrow{\boldsymbol{R_P}} + \frac{\partial \overrightarrow{\boldsymbol{R_P}}}{\partial \overrightarrow{\boldsymbol{U_P}}} \cdot \boldsymbol{\Delta \overrightarrow{U_P}} + \sum_{\boldsymbol{Q=1}}^{\boldsymbol{\pi\lambda\epsilon\upsilon\rho\acute{\epsilon}\varsigma}} \frac{\partial \overrightarrow{\boldsymbol{R_P}}}{\partial \overrightarrow{\boldsymbol{U_Q}}} \cdot \boldsymbol{\Delta \overrightarrow{U_Q}} = \boldsymbol{0}$$

$$\Leftrightarrow \boldsymbol{\Delta \overrightarrow{U_P}} = \left[\frac{\partial \overrightarrow{\boldsymbol{R_P}}}{\partial \overrightarrow{\boldsymbol{U_P}}}\right]^{\boldsymbol{-1}} \cdot \left(-\overrightarrow{\boldsymbol{R_P}} - \sum_{\boldsymbol{Q=0}}^{\boldsymbol{\pi\lambda\epsilon\upsilon\rho\acute{\epsilon}\varsigma}} \frac{\partial \overrightarrow{\boldsymbol{R_P}}}{\partial \overrightarrow{\boldsymbol{U_Q}}} \cdot \boldsymbol{\Delta \overrightarrow{U_Q}}\right) \quad \textbf{(2.18)}$$

Όπως αναφέρθηκε και στην παράγραφο 2.1, για την επίλυση των εξισώσεων ροής χρησιμοποιούμε μέθοδο χρονοπροέλασης. Οπότε ολοκληρώνοντας την 2.1, αυτή την φορά και τον χρονικό όρο, και διακριτοποιώντας τις χωρικές παραγώγους για την νέα χρονική στιγμή (έμμεση μέθοδος), έχουμε:

$$\int_{\boldsymbol{\Omega}} \frac{\partial \vec{\boldsymbol{U}}_{\boldsymbol{i}}^{\boldsymbol{n}}}{\partial \boldsymbol{t}} \boldsymbol{d\Omega} + \int_{\boldsymbol{\Omega}} \frac{\partial \overrightarrow{\boldsymbol{f}_{\boldsymbol{k}}^{\boldsymbol{n+1}}}}{\partial \boldsymbol{x_k}} \boldsymbol{d\Omega} = \boldsymbol{0} \Leftrightarrow$$

$$\frac{\partial \vec{\boldsymbol{U}}_{\boldsymbol{i}}^{\boldsymbol{n}}}{\partial \boldsymbol{t}} \int_{\boldsymbol{\Omega}} \boldsymbol{d\Omega} + \oint_{\boldsymbol{S}} \overrightarrow{\boldsymbol{f}_{\boldsymbol{k}}^{\boldsymbol{n+1}}} \cdot \boldsymbol{n_k} \cdot \boldsymbol{dS} \cong \boldsymbol{0} \Leftrightarrow$$

$$\frac{\vec{\boldsymbol{U}}_{\boldsymbol{i}}^{\boldsymbol{n+1}} - \vec{\boldsymbol{U}}_{\boldsymbol{i}}^{\boldsymbol{n}}}{\boldsymbol{\Delta t_i}} \cdot \boldsymbol{\Omega_i} + \sum_{j}^{\pi\lambda\epsilon\upsilon\rho\acute{\epsilon}\varsigma} \overrightarrow{\boldsymbol{f}_{\boldsymbol{jk}}^{\boldsymbol{n+1}}} \cdot \boldsymbol{n_k} \cdot \boldsymbol{\Delta S_j} \cong \boldsymbol{0} \quad \textbf{(2.19)}$$

Ο τύπος από τον οποίο προκύπτει το ψευδοχρονικό βήμα σε κάθε κυψέλη είναι:

$$\boldsymbol{\Delta t_i} = \boldsymbol{CFL} \frac{\boldsymbol{\Omega_i}}{(|\boldsymbol{u}|+\boldsymbol{c})\boldsymbol{\Omega_{ix}} + (|\boldsymbol{v}|+\boldsymbol{c})\boldsymbol{\Omega_{iy}}} \quad \textbf{(2.19)}$$

όπου CFL είναι ο αριθμός Courant-Friedrichs-Lewy[30] και επιλέγεται από τον χρήστη και τα Ω_{ix} και Ω_{iy} είναι οι προβολές του όγκου Ω_i στους άξονες x και y, αντίστοιχα.

Έχοντας λάβει υπόψη τον χρονικό όρο, ορίζεται ξανά ως $\overrightarrow{R_i} = \sum \overrightarrow{f_{ij}} \cdot \vec{n} \cdot \Delta S_j + \frac{\Delta \vec{U}_i}{\Delta t_i} \cdot \Omega_i$ και χρησιμοποιώντας την 2.15, υπολογίζονται οι παράγωγοι που χρειάζονται στο σύστημα εξισώσεων 2.18 (στην νέα χρονική στιγμή), είναι:

$$\frac{\partial \overrightarrow{R_P}}{\partial \vec{U}_P^{n+1}} = \frac{1}{2} \sum_{j=1}^{\pi\lambda\varepsilon\upsilon\rho\acute{\varepsilon}\varsigma} \left(\underline{A_n^P} + \widetilde{|\underline{A}|} \right) \Delta S_j + \frac{\partial}{\partial \vec{U}_P^{n+1}} \left[\frac{\vec{U}_P^{n+1} - \vec{U}_P^n}{\Delta t_P} \cdot \Omega_P \right]$$

$$= \frac{1}{2} \sum_{j=1}^{\pi\lambda\varepsilon\upsilon\rho\acute{\varepsilon}\varsigma} \left(\underline{A_n^P} + \widetilde{|\underline{A}|} \right) \Delta S_j + \frac{\Omega_P}{\Delta t_P} \cdot I \qquad \textbf{(2.20)}$$

και

$$\frac{\partial \overrightarrow{R_P}}{\partial \overrightarrow{U_Q}} = \frac{1}{2} \left(\underline{A_n^Q} - \widetilde{|\underline{A}|} \right) \Delta S_Q \qquad \textbf{(2.21)}$$

Από την σχέση 2.19 προκύπτει ότι: $\frac{\Omega_P}{\Delta t_P} = \frac{(|u|+c)\Omega_{Px} + (|v|+c)\Omega_{Py}}{CFL}$ **(2.22)**

Έχοντας διευκρινίσει όλους τους όρους της σχέσης 2.18, παρατηρούμε ότι είναι ένα σύστημα εξισώσεων συναρτήσει των μεταβλητών ροής της τωρινής και της επόμενης χρονικής στιγμής. Το γραμμικοποιημένο αυτό σύστημα εξισώσεων επιλύεται με τη μέθοδο Jacobi, μιας και παραλληλίζεται πιο εύκολα, άρα είναι κατάλληλη για επιλύτες που τρέχουν σε κάρτες γραφικών. Έχει επιλεγεί να γίνονται 15 επαναλήψεις για την ανανέωση του $\Delta \vec{U}_i^{n+1}$ χωρίς απαραίτητα να επιτυγχάνεται σύγκλιση της μεθόδου Jacobi. Η λύση του συστήματος ($\Delta \vec{U}$) χρησιμοποιείται για την ανανέωση του πεδίου ροής. Η διαδικασία επαναλαμβάνεται μέχρι να βρεθεί εκείνο το πεδίο που ικανοποιεί τις εξισώσεις 2.8. Ο αριθμός των εξωτερικών επαναλήψεων επιλέγεται από τον χρήστη, ανάλογα με το μέγεθος του πλέγματος και την πολυπλοκότητα της γεωμετρίας, προκειμένου να επιτευχθεί η σύγκλιση της μεθόδου. Η επιλογή της μη ακριβούς επίλυσης του συστήματος 2.18 δικαιολογείται από το γεγονός ότι προκειμένου να συγκλίνει η όλη διαδικασία, δεν απαιτείται σε κάθε ψευδοχρονικό βήμα η ακριβής εύρεση του $\Delta \vec{U}^{n+1}$ και κατ' επέκταση του $\vec{U}^{n+1}$. Το κέρδος από αυτήν την επιλογή είναι η μείωση του υπολογιστικού κόστους χωρίς να επηρεάζεται η ακρίβεια των αποτελεσμάτων.

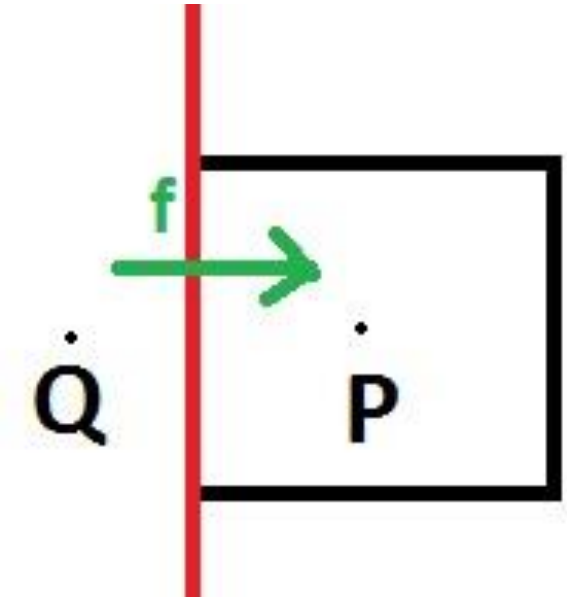
2.6 Οριακές συνθήκες

Στις προηγούμενες παραγράφους αναλύθηκε η διακριτοποίηση των εξισώσεων ροής (σχέση 2.1) για μια κυψέλη, της οποίας οι γειτονικές κυψέλες είναι εντός της ροής. Στην περίπτωση που η κυψέλη βρίσκεται στα άκρα του υπολογιστικού χωρίου (σχήμα 2.9) ή τέμνεται από το υπό μελέτη σώμα (σχήμα 2.10), πρέπει να εισαχθούν οι οριακές συνθήκες. Στην πρώτη περίπτωση εφαρμόζονται οι συνθήκες στο επ' άπειρο όριο (farfield conditions) και στη δεύτερη περίπτωση οι συνθήκες στερεού τοιχώματος (wall conditions).

2.6.1 Συνθήκες στο επ' άπειρο όριο (Farfield Conditions)

Στην περίπτωση αυτή, όπως φαίνεται και στο σχήμα 2.9, η κυψέλη βρίσκεται στο σύνορο του υπολογιστικού χωρίου. Σύμφωνα με την σχέση 2.8, σε κάθε πλευρά οποιασδήποτε κυψέλης απαιτείται ο υπολογισμός του $\vec{f} \cdot \vec{n}$. Σύμφωνα με το σχήμα Roe, χρειάζεται να είναι γνωστές και οι συντηρητικές μεταβλητές των γειτονικών κυψελών. Στην περίπτωση, όμως, που η πλευρά μιας κυψέλης είναι το όριο του υπολογιστικού χωρίου, δεν υπάρχει γειτονική κυψέλη. Γιατί την αντιμετώπιση του προβλήματος αυτού, ορίζεται ο ψευδόκομβος Q (όπως φαίνεται και στο σχήμα 2.9) και επιβάλλονται όλα τα πρωτεύοντα μεγέθη που περιγράφουν τις συνθήκες της ροής μακριά από το υπό μελέτη σώμα (οριακές συνθήκες τύπου Dirichlet). Οπότε, τώρα μπορεί να εφαρμοστεί το σχήμα Roe μεταξύ του κόμβου P και του ψευδόκομβου Q. Επειδή στον ψευδόκομβο δεν επιβάλλεται κάποια γεωγραφική θέση, ο υπολογισμός της ροής στην πλευρά αυτή γίνεται μόνο με σχήμα πρώτης τάξης ακρίβειας.

Με την παραπάνω λύση, η οποία χρησιμοποιείται και στον επιλύτη της παρούσας εργασίας, οι κυψέλες, που βρίσκονται στο σύνορο, δέχονται τις συνοριακές συνθήκες ως ροή.



Σχήμα 2.9: Η διακριτοποίηση των εξισώσεων ροής γίνεται για τη κυψέλη με κέντρο το P. Η κόκκινη γραμμή ορίζει το σύνορο του υπολογιστικού χωρίου. Q είναι ο ψευδόκομβος που χρησιμοποιούμε για τον υπολογισμό της ροής στη διεπιφάνεια κυψέλης και συνόρου.

2.6.2 Συνθήκες Στερεού Τοιχώματος (Wall Conditions)

Ακόμα και στην περίπτωση που η κυψέλη τέμνεται από το στερεό (σχήμα 2.10), απαιτείται ο υπολογισμός του $\vec{f} \cdot \vec{n}$ σε όλες τις πλευρές της. Στη διεπιφάνεια μεταξύ κυψέλης και στερεού σώματος, ορίζεται $\overrightarrow{f_w} = \vec{f} \cdot \vec{n}$. Για τον υπολογισμό του $\overrightarrow{f_w}$ δεν χρησιμοποιείται το σχήμα Roe.

Γενικά, ισχύει:

$$\vec{\mathbf{f}} \cdot \vec{\mathbf{n}} = \overrightarrow{\mathbf{f}_x}\mathbf{n}_x + \overrightarrow{\mathbf{f}_y}\boldsymbol{n}_y = \begin{bmatrix} \boldsymbol{\rho U_n} \\ \boldsymbol{\rho U_n u + p n_x} \\ \boldsymbol{\rho U_n v + p n_y} \\ \boldsymbol{U_n(\rho E + p)} \end{bmatrix} \quad \textbf{(2.23)}$$

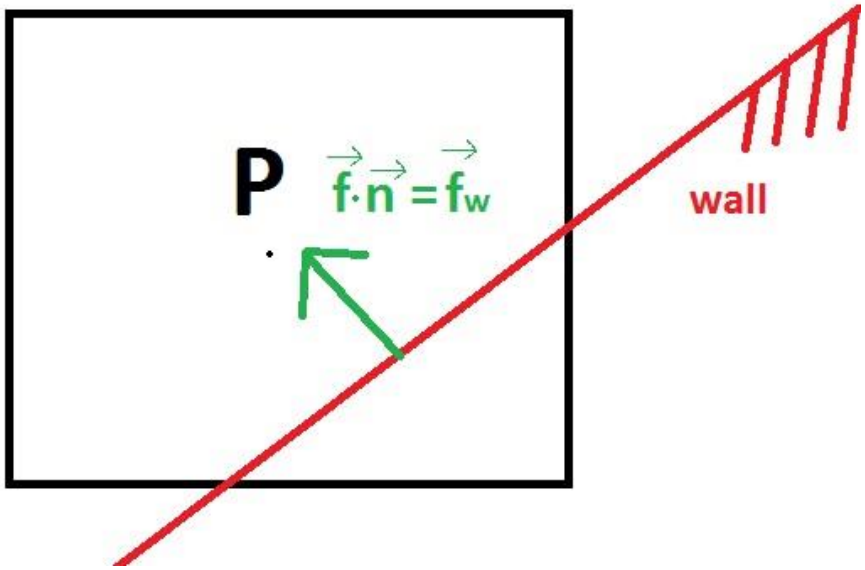
όπου $\boldsymbol{U_n = \vec{U} \cdot \vec{n} = u n_x + v n_y}$.

Πάνω στον τοίχο, δηλαδή στη διεπιφάνεια κυψέλης και στερεού σώματος, επιβάλλεται $\boldsymbol{\vec{U} \cdot \vec{n} =, U_n = 0}$, ώστε να εξασφαλίζεται η μη-εισχώρηση ρευστού μέσα στο στερεό.

Με αυτήν τη συνθήκη πάνω στον τοίχο, η σχέση 2.23 ισοδυναμεί με:

$$\overrightarrow{\boldsymbol{f_w}} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{p_P n_x} \\ \boldsymbol{p_P n_y} \\ \mathbf{0} \end{bmatrix} \quad \textbf{(2.24)}$$

Επομένως, η σχέση 2.24 πρέπει να ληφθεί υπ' όψη στον υπολογισμό των παραγώγων που χρειάζονται για το σύστημα 4xn εξισώσεων που έχει προκύψει από την διακριτοποίηση των εξισώσεων Euler.



Σχήμα 2.10: Η διακριτοποίηση των εξισώσεων ροής γίνεται για τη τραπεζοειδής κυψέλη με κέντρο το P. Η κόκκινη γραμμή ορίζει το περίγραμμα του υπό μελέτη στερεού σώματος.

Κεφάλαιο 3

Η αρχιτεκτονική παράλληλης επεξεργασίας CUDA

Η αρχιτεκτονική CUDA (Compute Unified Device Architecture) αναπτύσσεται από την NVIDIA και αποτελεί την αρχή κατασκευής των προγραμματιζόμενων καρτών γραφικών της εταιρίας. Οι Tesla, Fermi, Kepler και Maxwell αποτελούν μερικές ενδεικτικές εκδόσεις αρχιτεκτονικών CUDA τη χρονική περίοδο 2006-2015.

Κάθε προγραμματιζόμενη GPU της NVIDIA φέρει έναν αριθμό που χαρακτηρίζει την υπολογιστική δυνατότητα της κάρτας. Για παράδειγμα, οι GPUs υπολογιστικής δυνατότητας 1.0 ή 1.1 δεν υποστηρίζουν την αριθμητική διπλής ακρίβειας. Στο κεφάλαιο αυτό περιγράφεται η δομή της αρχιτεκτονικής Kepler που έχει υπολογιστική δυνατότητα 3.0 και χρησιμοποιήθηκε σε αυτήν τη διπλωματική εργασία.

Περισσότερες πληροφορίες για την αρχιτεκτονική CUDA μπορεί να βρει κάποιος και σε παρόμοιες διπλωματικές εργασίες που πραγματοποιήθηκαν στη Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών ΕΜΠ [13], [14], [15], [16], [17], [18] καθώς και στη Διδακτορική Διατριβή [12].

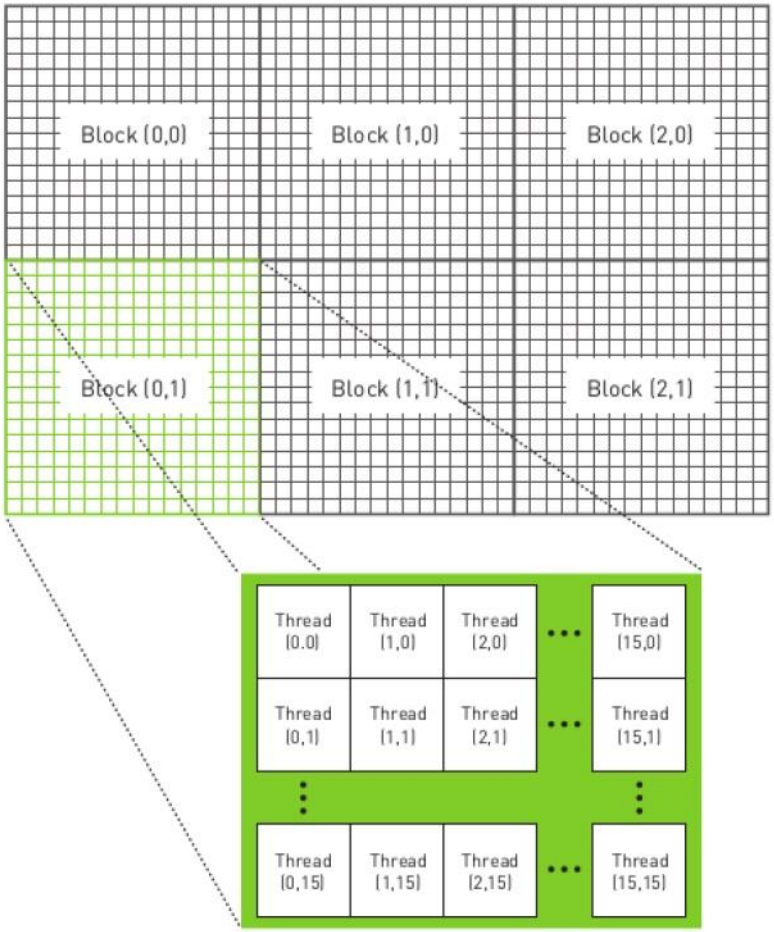
3.1 Οργάνωση των threads σε μία GPU

Τη βασική μονάδα επεξεργασίας των GPUs αποτελεί το thread. Κάθε GPU μπορεί να χειρίζεται τόσα threads όσα επιτρέπει η υπολογιστική της δυνατότητα και αρχιτεκτονική της. Για τη διαχείρισή τους εφαρμόζεται η λογική SIMT (Single Instruction Multiple Thread). Έτσι το ίδιο τμήμα κώδικα (kernel) εκτελείται από μία ομάδα από threads, με κάθε thread να διαχειρίζεται διαφορετικό όγκο δεδομένων.

Τα threads οργανώνονται σε blocks και εκτελούνται στους διαθέσιμους πολυεπεξεργαστές της GPU. Τα blocks που εκτελούν το ίδιο kernel, σχηματίζουν ένα grid από blocks. Κάθε πολυεπεξεργαστής μπορεί να αναλάβει την εκτέλεση έως και 8 blocks. Στην περίπτωση που δεν επαρκούν οι διαθέσιμοι πολυεπεξεργαστές της GPU, κάποια blocks μένουν ανενεργά μέχρι την ολοκλήρωση της εκτέλεσης ορισμένων εκ των ενεργών. Το γεγονός αυτό αποτελεί ένα μεγάλο πλεονέκτημα της CUDA, καθώς ο ίδιος κώδικας εκτελείται σε κάρτες γραφικών διαφορετικού αριθμού πολυεπεξεργαστών, χωρίς να απαιτείται η τροποποίηση ή προσαρμογή του. Ένα block μπορεί να αποτελείται το πολύ από 512 ή 1024 threads για GPUs υπολογιστικής δυνατότητας 1.x ή 2.x και άνω, αντίστοιχα.

Η εκτέλεση των threads γίνεται στους διαθέσιμους πολυεπεξεργαστές της GPU. Για την ακρίβεια, κάθε πολυεπεξεργαστής ομαδοποιηθεί τα threads σε ομάδες των 32, που ονομάζονται warps, ανεξάρτητα από την υπολογιστική δυνατότητα της GPU. Ο αριθμός των warps που μπορεί να χειρίζεται ταυτόχρονα ο πολυεπεξεργαστής εξαρτάται από την υπολογιστική δύναμη της GPU. Τα threads του ίδιου warp εκτελούν την ίδια σειρά εντολών ταυτόχρονα. Γι' αυτό το λόγο, ο προγραμματιστής οφείλει να εξασφαλίζει κάθε γραμμή του κώδικα να εκτελείται από την πλειοψηφία των threads του ίδιου warp. Στην περίπτωση που κάποιο thread δεν εκτελεί μια σειρά εντολών (πιο πιθανή αιτία είναι να έχει προηγηθεί κάποια συνθήκη if), τότε το thread αυτό περιμένει τα υπόλοιπα να εκτελέσουν τις εντολές αυτές και μόλις ολοκληρωθούν, αρχίζει την εκτέλεση των επόμενων εντολών του. Αυτό το χαρακτηριστικό του warp ελήφθη υπόψη κατά τον προγραμματισμό του επιλύτη και αναλύεται στο κεφάλαιο 4.

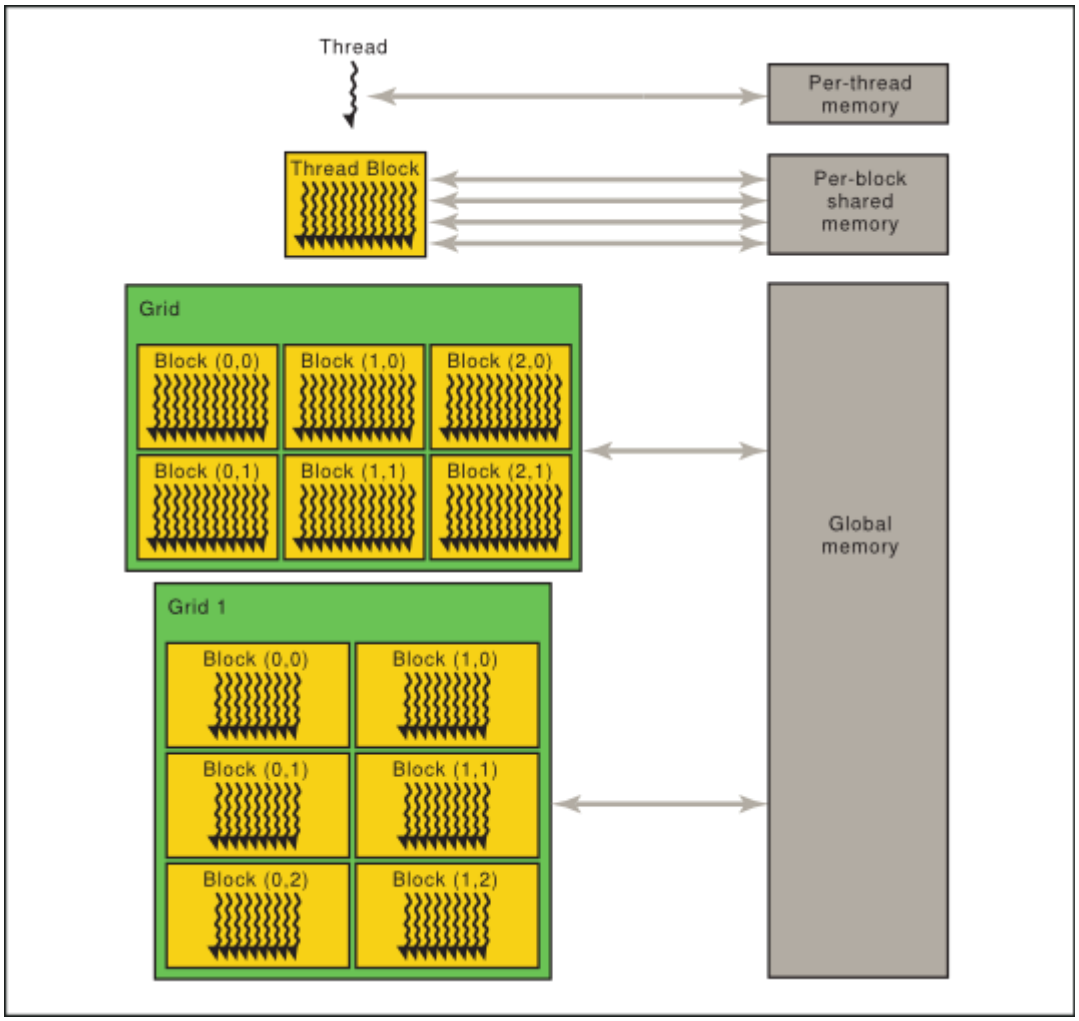
Κάθε thread χαρακτηρίζεται από ένα τοπικό αύξοντα αριθμό στο block που αυτό ανήκει και, αντίστοιχα, κάθε block από τον αύξοντα αριθμό του στο grid. Έτσι, κάθε thread μπορεί να συσχετιστεί με συγκεκριμένες θέσεις μνήμης με βάση τον αύξοντα αριθμό του στο block και τον αύξοντα αριθμό του block στο grid. Επομένως, κάθε thread επεξεργάζεται διαφορετικά δεδομένα.



Σχήμα 3.1: Διάταξη από blocks και threads σε 2 διαστάσεις. [20]

Ο προγραμματιστής επιλέγει ανάμεσα σε 1Δ, 2Δ ή 3Δ διάταξη των threads στα blocks και των blocks στο grid (Σχήμα 3.1). Σημειώνεται ότι οι GPUs αρχιτεκτονικής Kepler υποστηρίζουν την 3Δ διάταξη των blocks στο grid.

Η διάταξη και η οργάνωση των threads σε blocks και των blocks σε grid φαίνεται στο σχήμα 3.2. Στο σχήμα αυτό φαίνονται επιπλέον οι μνήμες της GPU στις οποίες έχει πρόσβαση κάθε thread. Οι μνήμες αυτές παρουσιάζονται στην παράγραφο 3.3.



Σχήμα 3.2: Οργάνωση των threads σε blocks και των blocks σε grid. Εδώ, ο προγραμματιστής έχει επιλέξει 1Δ κατανομή των threads σε blocks, και 2Δ κατανομή των blocks στο grid. Κάθε thread έχει τη δική του τοπική μνήμη. Τα threads του ίδιου block μοιράζονται δεδομένα μέσω της κοινής γρήγορα προσπελάσιμης shared μνήμης. Όλα τα threads του grid έχουν πρόσβαση στην κεντρική μνήμη της GPU, την texture και την constant. [1]

3.2 Η αρχιτεκτονική Kepler [21][22]

Οι πολυεπεξεργαστές των καρτών γραφικών που βασίζονται στην αρχιτεκτονική Kepler (σχήμα 3.4) συγκροτούνται σε GPCs (Graphic Processor Clusters). Κάθε GPC περιέχει:

- 8 πολυεπεξεργαστές

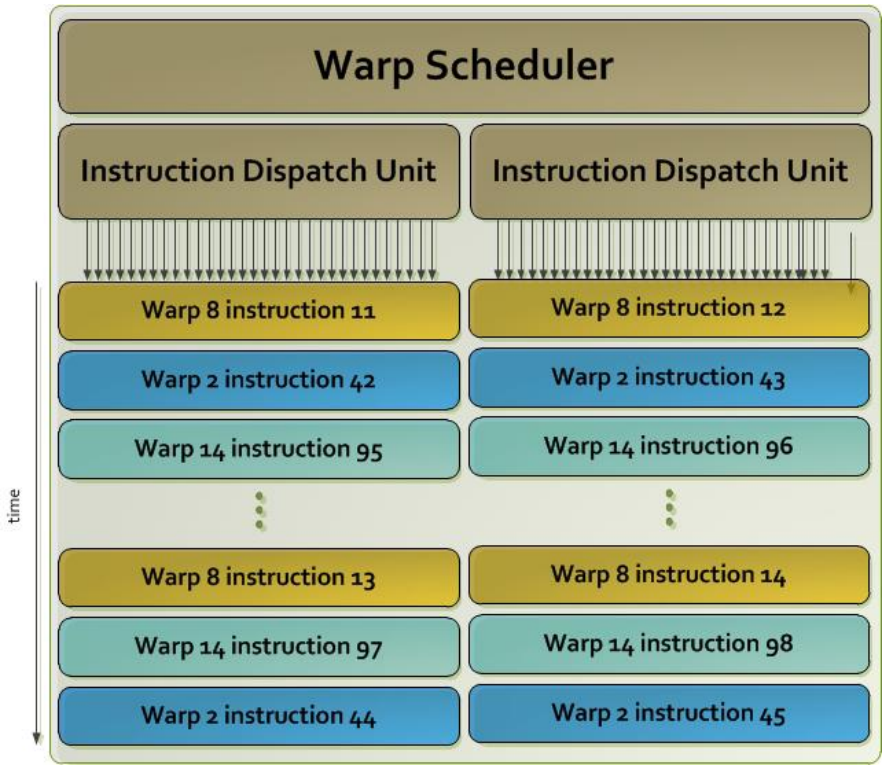
- 1 ενδιάμεση μνήμη (L2 cache)

Κάθε πολυεπεξεργαστής αποτελείται από:

- 192 πυρήνες
- 32 ειδικές μονάδες εκτέλεσης μαθηματικών συναρτήσεων (Special Function Units, SFU)
- 4 warp schedulers
- 2 ενδιάμεσες μνήμες (constant cache, texture cache)
- 1 ενδιάμεση μνήμη (L1 cache)
- 1 γρήγορα προσπελάσιμη μνήμη (shared μνήμη)
- 1 Read-Only Data μνήμη
- 65536 32-bit καταχωρητές (registers)

Ο αριθμός των πυρήνων ενός πολυεπεξεργαστή και εκείνος των SFUs δεν εξαρτάται από την υπολογιστική δυνατότητα της GPU αρχιτεκτονικής Kepler. Έτσι, κάρτες αρχιτεκτονικής Kepler (είτε υπολογιστικής δυνατότητας 3.0 είτε 3.5) έχουν 192 πυρήνες και 32 SFUs.

Οι warp schedulers ορίζουν στα warps τις εργασίες που θα εκτελεστούν. Τα warps που εκτελούνται σε έναν πολυεπεξεργαστή χαρακτηρίζονται από έναν τοπικό αύξοντα αριθμό. Δηλαδή τα threads 0 έως 31 ανήκουν στο warp 0 κ.ο.κ. Έτσι ο πρώτος από τους 4 warp schedulers χειρίζεται τα warps με αριθμό $4 \cdot \kappa$, ο δεύτερος με $4 \cdot \kappa + 1$ κ.ο.κ. Επίσης, κάθε πολυεπεξεργαστής περιέχει και 8 instruction dispatch units, επιτρέποντας να εκτελούνται έτσι 4 warps ταυτόχρονα. Κάθε warp scheduler σε κάρτες γραφικών αρχιτεκτονικής Kepler επιλέγει 4 warps και 2 ανεξάρτητες μεταξύ τους εντολές που μπορούν να εκτελεστούν σε κάθε κύκλο. Στις κάρτες υπολογιστικής δυνατότητας 3.5 μπορούν εντολές διπλής ακρίβειας να εκτελούνται παράλληλα με άλλες εντολές, κάτι που δεν συνέβαινε στις παλαιότερες κάρτες.



Σχήμα 3.3: Η χρονική πορεία εκτέλεσης των εντολών ενός warp scheduler. [21]

Οι ενδιάμεσες μνήμες constant, texture cache επιταχύνουν την ανάγνωση δεδομένων από την constant και την texture μνήμη, αντίστοιχα, που βρίσκονται στον ίδιο χώρο με την κεντρική μνήμη της GPU. Η διάσταση της texture cache εξαρτάται από τον τύπο της κάρτας γραφικών και είναι ανάμεσα σε 12 και 48 KB ανά πολυεπεξεργαστή. Η constant cache έχει διάσταση 8 KB και η constant μνήμη 64 KB.

Οι ενδιάμεσες μνήμες L1, L2 cache επιταχύνουν την πρόσβαση στην κεντρική μνήμη της GPU και στις τοπικές ανά thread. Υπενθυμίζεται ότι σε κάθε thread αντιστοιχεί συγκεκριμένος χώρος της κεντρικής μνήμης. Ο χώρος αυτός ανά thread αποτελεί την τοπική ανά thread μνήμη. Στις κάρτες γραφικών αρχιτεκτονικής Kepler το μέγεθος της τοπικής ανά thread μνήμης είναι 512 KB. Το συνολικό μέγεθος της L2 cache για όλους τους πολυεπεξεργαστές είναι 768 KB για κάρτες υπολογιστικής δυνατότητας 3.0, ενώ είναι 1536 KB για κάρτες υπολογιστικής δυνατότητας 3.5. Η L1 βρίσκεται στον ίδιο χώρο με τη shared μνήμη και δίνεται η δυνατότητα στον προγραμματιστή να επιλέξει ανάμεσα σε 16 KB L1 και 48 KB shared (αποτελεί προεπιλογή) ή ανάποδα ή και οι δύο να έχουν από 32 KB. Η επιλογή γίνεται σύμφωνα με τις απαιτήσεις του kernel σε shared μνήμη. Για παράδειγμα, στην περίπτωση που ένα kernel χρησιμοποιεί λιγότερα των 16KB από τη shared μνήμη ανά πολυεπεξεργαστή, συμφέρει ο προγραμματιστής να μοιράσει τα συνολικά 64 KB ανά πολυεπεξεργαστή σε 16 KB για τη shared και 48 KB για την L1 cache.

Σχήμα 3.4: Σχεδιάγραμμα της αρχιτεκτονικής Kepler. Σε αυτή, 15 πολυεπεξεργαστές μοιράζονται την ίδια L2 cache που επιταχύνει την προσπέλαση της κεντρικής μνήμης της κάρτας γραφικών και της τοπικής ανά thread που βρίσκεται στον ίδιο χώρο. Ο δίαυλος επικοινωνίας (PCI Express 3.0 Host Interface) επιτρέπει τη μεταφορά δεδομένων από την κεντρική μνήμη της κάρτας γραφικών σε εκείνη του υπολογιστή και ανάποδα. Μια ειδική μονάδα (GigaThread) διανέμει τα blocks στους διαθέσιμους πολυεπεξεργαστές. Κάθε πολυεπεξεργαστής έχει 192 πυρήνες, 32 ειδικές μονάδες εκτέλεσης μαθηματικών συναρτήσεων, 4 μονάδες χειρισμού των warps, registers των 32-bit και 64KB shared και L1 cache. [21]

Κάθε πολυεπεξεργαστής περιέχει 65536 32-bit registers. Με βάση τις απαιτήσεις του kernel σε registers και KB shared μνήμης ανά πολυεπεξεργαστή, καθορίζεται ο αριθμός των blocks που διανέμεται στους πολυεπεξεργαστές. Ο μέγιστος αριθμός threads που μπορεί γενικά να χειριστεί ένας πολυεπεξεργαστής είναι 2048.

3.3 Περιγραφή των ειδών μνήμης της GPU

Το μέγεθος των ενδιάμεσων (cache) μνημών των GPUs είναι πολύ μικρότερο σε σχέση με εκείνο των σύγχρονων CPUs. Ενδεικτικά αναφέρεται ότι κάθε blade server του cluster διασυνδεδεμένων GPUs της ΜΠΥΠ&Β/ΕΜΠ 2 quad core CPUs. Κάθε πυρήνας μιας CPU έχει 12288 KB cache μνήμη. Το μέγεθος αυτό είναι πολύ μεγαλύτερο σε σχέση με το συνολικό μέγεθος των cache μνημών ανά πολυεπεξεργαστή μιας GPU. Μια κάρτα γραφικών αρχιτεκτονικής Kepler έχει 8 KB constant cache, 12 KB texture cache, 64 KB για τη shared και την L1 cache 48 KB Read-Only Data cache και 96 KB L2 cache ανά πολυεπεξεργαστή. Συνολικά έχει 228 KB cache μνημών.

Ακολουθούν μερικές βασικές πληροφορίες για τις διάφορες μνήμες που διαθέτει μια GPU και έχουν αναφερθεί μέχρι στιγμής στην παρούσα εργασία.

3.3.1 Κεντρική μνήμη (global memory)

Η κεντρική μνήμη της GPU (global memory) αποτελεί τη μεγαλύτερη σε έκταση μνήμη της κάρτας γραφικών και είναι προσπελάσιμη από όλα τα threads. Χαρακτηρίζεται, όμως, από υψηλό χρόνο προσπέλασης. Συνεπώς, η χρήση της γρήγορα προσπελάσιμης shared μνήμης ή των cached constant και texture μνημών αντί της κεντρικής, φυσικά όταν αυτό γίνεται, αυξάνει κατακόρυφα την απόδοση του GPU-κώδικα. Επιπλέον η χρήση της κεντρικής μνήμης πρέπει να περιορίζεται, κατά το δυνατόν σε ένα kernel.

Οι GPUs αντιμετωπίζουν την κεντρική μνήμη ως μία αλληλουχία από τμήματα μήκους 32, 64 ή 128 Bytes. Αυτό πρακτικά σημαίνει ότι ο δίαυλος μεταφοράς δεδομένων από την κεντρική μνήμη στους registers των πολυεπεξεργαστών μεταφέρει τμήματα της κεντρικής μνήμης μήκους 32, 64 ή 128 Bytes. Επομένως, όσο μικρότερο είναι το πλήθος των τμημάτων της κεντρικής μνήμης, στα οποία έχουν πρόσβαση τα threads ενός warp, τόσο λιγότερες φορές θα μεταφέρει δεδομένα ο δίαυλος επικοινωνίας.

3.3.2 Constant μνήμη

Η constant μνήμη βρίσκεται στον ίδιο χώρο με την κεντρική μνήμη της GPU. Αντίθετα, όμως, με την κεντρική μνήμη, είναι cached, δηλαδή η πρόσβαση σε αυτήν επιτυγχάνεται από την constant cache, και επιτρέπει στα threads που ε-

κτελούνται στη GPU μόνο να διαβάζουν (read-only) από αυτή. Επομένως, ενδείκνυται η αποθήκευση σταθερών ποσοτήτων στη μνήμη αυτή. Η αποθήκευση των ποσοτήτων αυτών από τη στιγμή που τα threads της GPU δεν μπορούν να γράψουν σε αυτή, γίνεται από τη CPU. Υπενθυμίζεται ότι η διάσταση της constant μνήμης είναι 64 KB και εκείνη της constant cache 8 KB ανεξάρτητα της υπολογιστικής δυνατότητας της GPU.

Η ανάγνωση δεδομένων που είναι προσωρινά αποθηκευμένα στην constant cache (cache hit) γίνεται πρακτικά ακαριαία, αφού ο χρόνος προσπέλασης μιας cache μνήμης είναι πολύ μικρός. Αν τα δεδομένα δεν βρίσκονται στην constant cache (cache miss), τότε εκείνα μεταφέρονται από την constant μνήμη στην constant cache (παίρνοντας τη θέση κάποιων άλλων) και διαβάζονται από τα threads μέσω της constant cache. Δηλαδή, πρακτικά, ο χρόνος ανάγνωσης της τιμής μιας σταθεράς που δεν βρίσκεται στην constant cache είναι ίδιος με το χρόνο προσπέλασης της κεντρικής μνήμης.

3.3.3 Texture μνήμη

Η texture μνήμη είναι ακόμα ένα είδος μνήμης μόνο για ανάγνωση, που πρέπει να αυξήσει την απόδοση όταν οι αναγνώσεις στη μνήμη ακολουθούν συγκεκριμένα μοτίβα. Παρά το ότι αρχικά είχε σχεδιαστεί για παραδοσιακές εφαρμογές γραφικών, μπορεί να χρησιμοποιηθεί και σε υπολογιστικές εφαρμογές με εξαιρετικά αποτελέσματα. Η παρούσα εργασία δεν εκμεταλλεύεται την texture μνήμη και για το λόγο αυτό δε θα αναλυθεί παραπάνω.

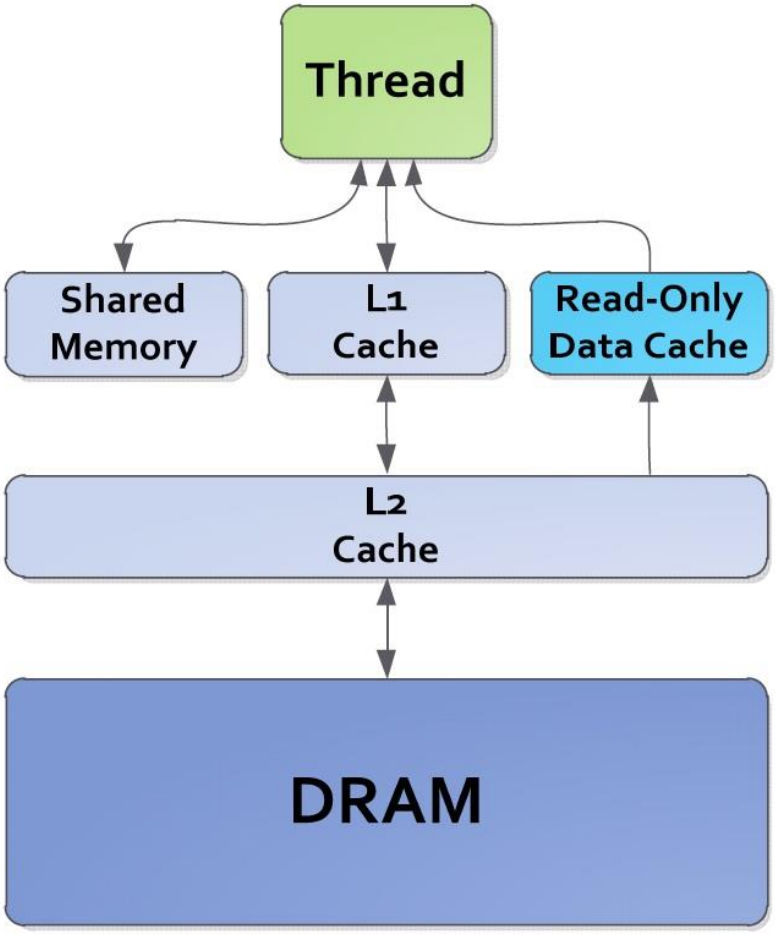
3.3.4 Shared μνήμη

Η shared μνήμη είναι μνήμη ταχείας προσπέλασης και επιτρέπει την επικοινωνία μεταξύ των threads του ίδιου block. Στις GPUs υπολογιστικής δυνατότητας 2.x, ο προγραμματιστής επιλέγει ανάμεσα σε 48 KB shared μνήμης και 16 KB L1 cache ανά πολυεπεξεργαστή, ή αντίστροφα, ενώ στις 3.x μπορεί να την μοιράσει και σε 32 KB shared μνήμης και 32 KB L1 cache ανά πολυεπεξεργαστή. Η ταχύτητα προσπέλασης της shared μνήμης είναι περίπου ίδια με εκείνη μιας cache μνήμης.

3.3.5 Τοπική μνήμη (local memory)

Σε κάθε thread αντιστοιχεί ένα τμήμα της κεντρικής μνήμης, η λεγόμενη τοπική μνήμη (local memory). Ως τμήμα της κεντρικής μνήμης, η τοπική χαρακτηρίζεται από υψηλούς χρόνους προσπέλασης, ενώ χρησιμοποιείται κυρίως από την ίδια την κάρτα γραφικών όταν οι απαιτήσεις ενός kernel σε registers υπερβαίνουν τους διαθέσιμους ανά πολυεπεξεργαστή. Είναι δομημένη με τέτοιο τρόπο ώστε, σε περίπτωση χρήσης της από τα threads του ίδιου warp, εκείνα να διαβάζουν/αποθηκεύουν σε θέσεις του ίδιου τμήματος μνήμης, όπως δηλαδή υπαγο-

ρεύει το βέλτιστο πρότυπο προσπέλασης της κεντρικής μνήμης. Έτσι επιτυγχάνεται ο ελάχιστος δυνατός χρόνος προσπέλασης. Η διάσταση της τοπικής ανά thread μνήμης είναι 512 KB για GPUs υπολογιστικής δυνατότητας 2.x και άνω.



Σχήμα 3.5: Ιεραρχία μνημών σε κάρτες γραφικών αρχιτεκτονικής Kepler [22].

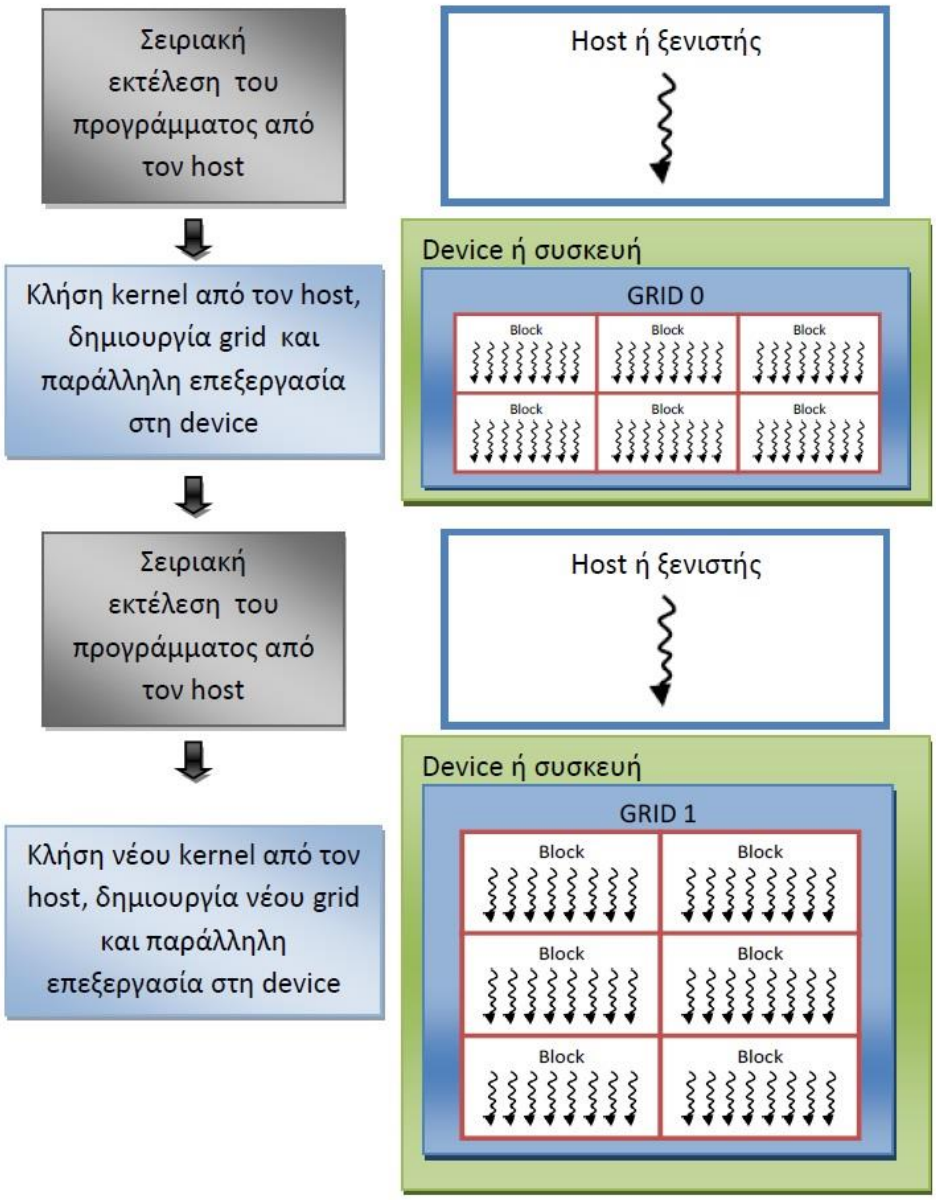
3.4 Συνεργασία CPU-GPU

Το προγραμματιστικό μοντέλο της CUDA ορίζει τη κεντρική μονάδα επεξεργασίας (CPU) ως host και τη κάρτα γραφικών ως device ή συσκευή. Η συσκευή συνεργάζεται με τον host για να εκτελέσουν ένα πρόγραμμα. Έτσι, η CPU εκτελεί σειριακά το πρόγραμμα και καλεί, όπου απαιτείται, τα kernels, τα οποία εκτελούνται στη κάρτα γραφικών. Στο σχήμα 3.7 παρουσιάζεται αυτός ο τρόπος λειτουργίας μεταξύ CPU και GPU.

Η κλήση ενός kernel από τη CPU είναι ασύγχρονη. Δηλαδή, ο έλεγχος επιστρέφει στη CPU πριν την ολοκλήρωση της εκτέλεσης του kernel. Αυτό πρακτικά σημαίνει ότι η CPU μπορεί να εκτελεί τμήμα ενός κώδικα χαμηλού ή ακόμα και μηδενικού βαθμού παραλληλίας, επεξεργαζόμενη δεδομένα που βρίσκονται στην κεντρική μνήμη του υπολογιστή, ταυτόχρονα με την εκτέλεση ενός kernel στη GPU. Έτσι επιτυγχάνεται η αξιοποίηση όλων των διαθέσιμων υπολογιστικών πό-

ρων και ως εκ τούτου, αυξάνεται η παράλληλη απόδοση του κώδικα. Στην περίπτωση όμως που ο κώδικας που εκτελείται στη CPU χρειάζεται δεδομένα από το kernel που εκτελείται στη GPU, τότε πρέπει να γίνει συγχρονισμός, δηλαδή να επιστραφεί ο έλεγχος στη CPU μετά την ολοκλήρωση της εκτέλεσης του kernel.

Ο διαχωρισμός μεταξύ host και device επιβάλλει, συν τοις άλλοις, κάθε μονάδα να διαθέτει τη δική της ξεχωριστή μνήμη. Αυτό σημαίνει ότι η μνήμη της CPU δεν είναι προσπελάσιμη από τα threads της GPU και αντίστροφα. Συνεπώς, για την επεξεργασία των δεδομένων από την GPU απαιτείται πρώτα η μεταφορά τους από τη μνήμη της CPU και, στη συνέχεια, η εκτέλεση του αντίστοιχου προγράμματος (kernel). Μόλις εκτελεστούν τα kernels, τα αποτελέσματα πρέπει πάλι να μεταφερθούν στην κεντρική μονάδα επεξεργασίας. Η διαδικασία μεταφοράς δεδομένων είναι ιδιαίτερα χρονοβόρα. Έτσι, η μείωση του όγκου των δεδομένων, που μεταφέρονται, επιταχύνει την εφαρμογή.



Σχήμα 3.7: Ροή ενός προγράμματος C με τη χρήση της μεθόδου CUDA. [13]

3.5 Μεταγλωττιστής nvcc

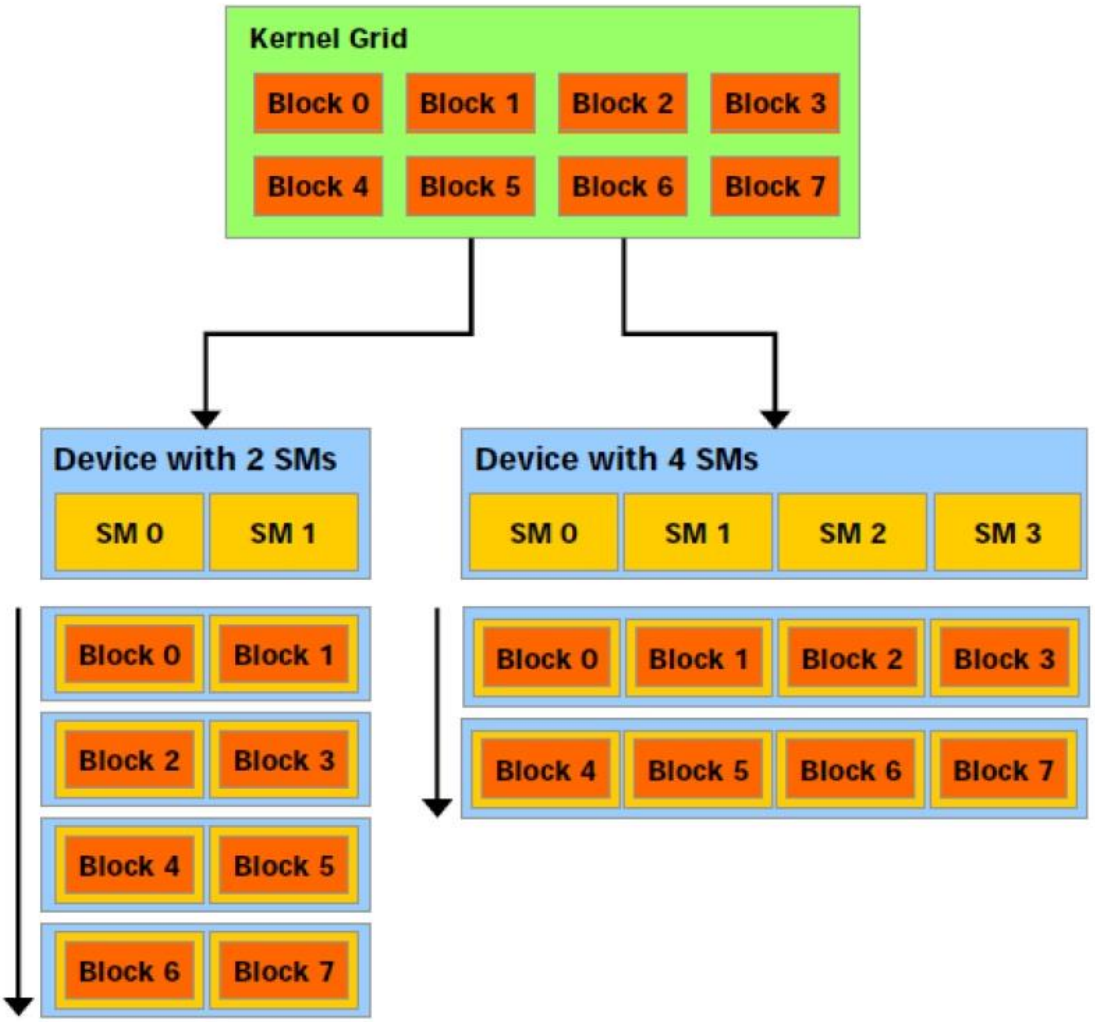
Για να γίνει εκτελέσιμος ο πηγαίος κώδικας, τα kernels πρέπει να μεταφραστούν, σε δυαδικό κώδικα από τον μεταγλωττιστή ή αλλιώς compiler της CUDA, τον nvcc. Ο μεταγλωττιστής nvcc, προκειμένου να απλοποιήσει τη διαδικασία μεταγλώττισης, χρησιμοποιεί εντολές που είναι ήδη οικίες για τους προγραμματιστές της C.

Ο πηγαίος κώδικας περιλαμβάνει τμήματα κώδικα γραμμένα για τη CPU και τμήματα κώδικα που προορίζονται για εκτέλεση στη GPU. Στόχος του μεταγλωττιστή nvcc είναι να ξεχωρίσει τον κώδικα που αναφέρεται στη GPU και τον κώδικα που αναφέρεται στη CPU και έπειτα να μεταγλωττίσει το κώδικα της GPU σε δυαδική μορφή. Υπάρχει δυνατότητα να μεταγλωττιστεί και σε γλώσσα assembly αλλά δεν χρειάζεται σε εφαρμογές γλώσσας C και γενικά γλωσσών προγραμματισμού υψηλού επιπέδου. Το κομμάτι του κώδικα που αναφέρεται στη CPU είναι δυνατό να μεταγλωττιστεί από τον ίδιο το μεταγλωττιστή nvcc ή καλώντας έναν άλλο μεταγλωττιστή. Έτσι, παράγεται το τελικό εκτελέσιμο αρχείο.

Ο μεταγλωττιστής επεξεργάζεται τον πηγαίο κώδικα βάσει των κανόνων σύνταξης της γλώσσας C++. Για αυτό το λόγο, το μέρος του κώδικα που εκτελείται στη CPU μπορεί να είναι εξ ολοκλήρου γραμμένο σε C++ και να χρησιμοποιεί όλα τα πλεονεκτήματα της γλώσσας. Το κομμάτι όμως που αναφέρεται στη GPU πρέπει να χρησιμοποιεί εντολές της C, οι οποίες είναι ένα υποσύνολο των εντολών της C++. Αυτό το πλεονέκτημα εκμεταλλεύεται η παρούσα εργασία. Συνεπώς, οι ευκολίες της C++ χάρη στον αντικειμενοστραφή προγραμματισμό συνδυάζονται άψογα με την παράλληλη επεξεργασία που προσφέρει το προγραμματιστικό μοντέλο της CUDA.

Ορισμένα χαρακτηριστικά της CUDA είναι διαθέσιμα μόνο για κάρτες γραφικών με συγκεκριμένη υπολογιστική δυνατότητα. Η υπολογιστική δυνατότητα της κάρτας δηλώνεται με δύο αριθμούς και εξαρτώνται από την αρχιτεκτονική της. Ο αριθμός αυτός κυμαίνεται από 1.0 μέχρι 5.3, αλλά το ανώτατο όριο μπορεί να ξεπεραστεί όσο κατασκευάζονται νέες κάρτες γραφικών με βελτιωμένη αρχιτεκτονική. Για να ενεργοποιηθούν κάποια χαρακτηριστικά της CUDA είναι αναγκαίο όχι μόνο να διατίθεται κάρτα γραφικών με επαρκή υπολογιστική δυνατότητα, αλλά και να δηλωθεί στο μεταγλωττιστή. Για παράδειγμα, οι πράξεις με αριθμούς διπλής ακρίβειας είναι διαθέσιμες για υπολογιστική δυνατότητα 1.3 και άνω.

Η εντολή μετατροπής ενός πηγαίου κώδικα (example.cu) σε εκτελέσιμο αρχείο (example.exe) είναι: nvcc –arch sm_30 example.cu –o example.exe –lcuda. Το λογισμικό έχει δυνατότητα για πράξεις διπλής ακρίβειας, αφού έχει δηλωθεί υπολογιστική ικανότητα -arch sm_303. Τέλος το όρισμα –lcuda δηλώνει στο μεταγλωττιστή ότι πρέπει να συμπεριληφθεί και η βιβλιοθήκη της CUDA. Αξίζει να σημειωθεί ότι δεν είναι απαραίτητο να συμπεριληφθεί ένα μόνο αρχείο για μεταγλώττιση. Αν ένα πρόγραμμα συνδέεται με περισσότερα αρχεία ή βιβλιοθήκες, αυτά δηλώνονται στο μεταγλωττιστή. Τα παραπάνω παραδείγματα αναφέρονται σε λειτουργικό UNIX.



Σχήμα 3.8: Λειτουργία πολυεπεξεργαστών (multiprocessors).[1]

3.6 NVIDIA GeForce GTX 670

Η Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ προκειμένου να εκτελέσει τους απαιτητικούς σε υπολογιστική ισχύ κώδικες που αναπτύσσει, έχει στην κατοχή της την πλατφόρμα υψηλής απόδοσης "VELOS". Αυτή αποτελείται από τρία PC clusters (συγκροτήματα υπολογιστών) με συνολική υπολογιστική ισχύ 44 Tera-Flop. Τα επιμέρους χαρακτηριστικά του κάθε cluster είναι:

- Το πρώτο αποτελείται από 32 διασυνδεδεμένους επεξεργαστές (CPUs) αρχιτεκτονικής 64-bit και συνολικά 80 πυρήνες. Η συνολική του μνήμη RAM είναι 110GB και το λογισμικό που χρησιμοποιείται είναι το Linux (Fedora και CentOS). Κάποιοι επεξεργαστές είναι εξοπλισμένοι με κάρτες γραφικών της NVIDIA (GTX 280, 285, 580 & 670).
- Το δεύτερο περιλαμβάνει 58 διακομιστές τύπου blade (blade servers), που ο καθένας διαθέτεις δύο τετραπύρηνους ή 2 εξαπύρηνους Xeon επεξεργαστές. Συνολικά το cluster διαθέτει 528 πυρήνες. Κάθε επεξεργαστής

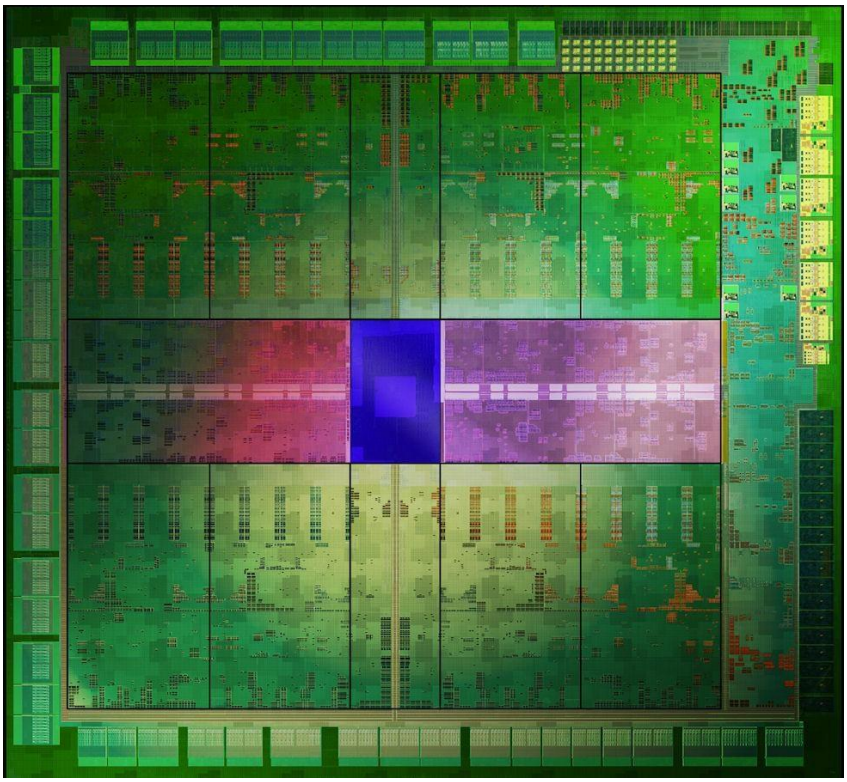
διαθέτει μνήμη RAM από 8 GB μέχρι 64 GB. Το λειτουργικό σύστημα είναι Linux (CentOS 5, 6 & 7).

- Το τρίτο περιλαμβάνει 6 διακομιστές τύπου blade (blade servers) με δύο τετραπύρηνους και 2 εξαπύρηνους Xeon επεξεργαστές, που ο κάθε επεξεργαστής έχει 16 GB μνήμη RAM. Οι 4 διακομιστές έχουν 3 κάρτες γραφικών NVIDIA Tesla M2050, με 3 GB μνήμη η καθεμιά κάρτα και οι άλλοι 2 διακομιστές έχουν 2 κάρτες γραφικών NVIDIA Tesla K20, με 5 GB μνήμη η καθεμιά (συνολικά 12 Tesla M2050, 4 K20, 56 GB μνήμη). Το λειτουργικό σύστημα είναι Linux (CentOS 7).

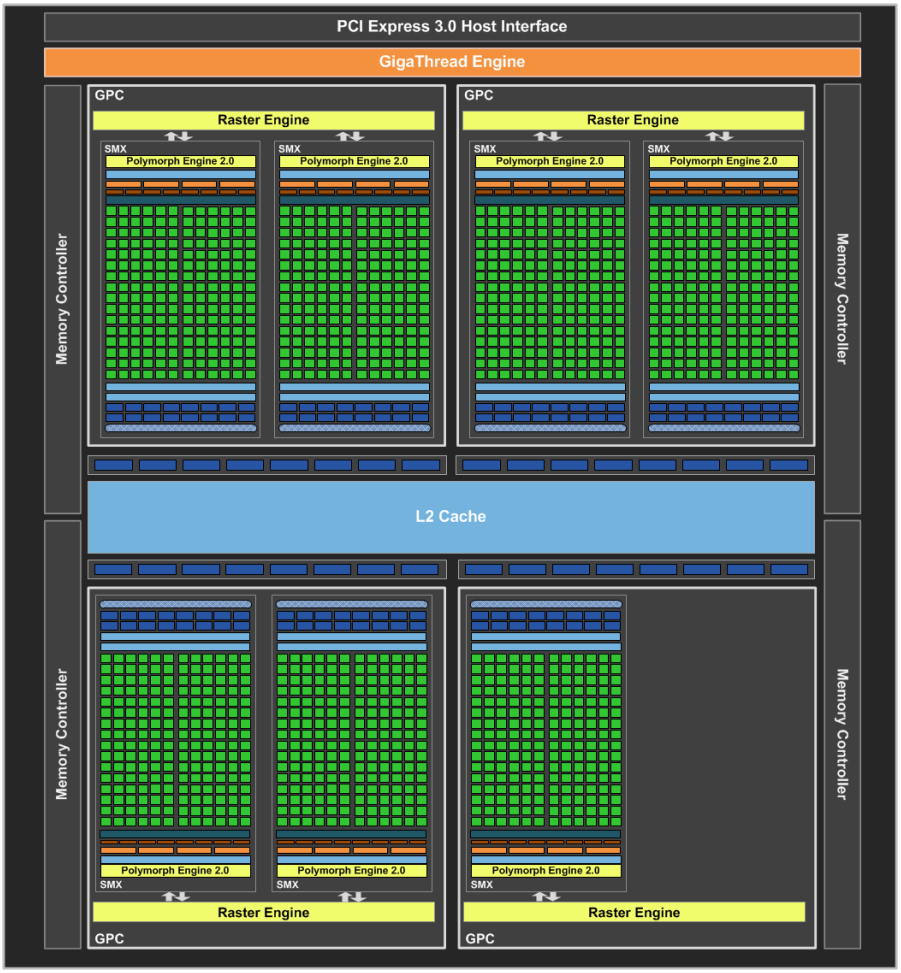
Οι απαραίτητες υπηρεσίες, που απαιτούνται για την επικοινωνία και την αποθήκευση δεδομένων, γίνονται από δύο αποκλειστικούς διακομιστές (ο ένας για τους δύο πρώτους clusters και ο άλλος για το τρίτο cluster) σύμφωνα με τα πρωτόκολλα Network Information Service (NIS[28]) και Network File System Service (NFS[27]).

Η κάρτα γραφικών που χρησιμοποιήθηκε για την εκτέλεση του επιλύτη που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας είναι η GeForce GTX 670 της εταιρίας NVIDIA και επεξεργαστή τον τετραπύρηνο Intel Core i5-3750[29] στα 3,4 GHz με μνήμη cache 6 MB. Η κάρτα αυτή είναι αρχιτεκτονικής Kepler, υπολογιστικής δυνατότητας 3.0 και η μέθοδος της λιθογραφίας στα 28nm. Στην κάρτα αυτή συναντάμε 4 GPC, 7 νέας γενιάς Streaming Multiprocessors (SMX) και 4 ελεγκτές μνήμης (memory controllers). Καθεμιά από τις μονάδες SMX ενσωματώνει 192 πυρήνες CUDA και καθώς οι μονάδες SMX είναι επτά, οι CUDA πυρήνες είναι $7 \times 192 = 1344$.

Τα σχήματα 3.9 και 3.10 αποτελούν εικόνες από τον «εσωτερικό» κόσμο της GTX 670.



Σχήμα 3.9: Διάγραμμα που δείχνει τα μικροπλακίδια (die) που αποτελείται μια κάρτα γραφικών αρχιτεκτονικής Kepler. [23]



Σχήμα 3.10: Το block διάγραμμα της GeForce GTX 670. [23]

Κεφάλαιο 4

Προγραμματισμός Επιλύτη σε GPUs

Στο κεφάλαιο αυτό παρουσιάζεται ο επιλύτης των εξισώσεων Euler, όπως αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, καθώς και η λογική πίσω από την ανάπτυξη του με σκοπό την καλύτερη αξιοποίηση των πόρων της GPU και την επίτευξη όσο τον δυνατόν μικρότερων χρόνων εκτέλεσης. Ο επιλύτης αυτός βασίστηκε σε ήδη υπάρχοντα επιλύτη της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών του Ε.Μ.Π., ο οποίος εκτελείται από τη CPU.

Ο επιλύτης, όπως έχει ήδη αναφερθεί στο κεφάλαιο 2, επιλύει διδιάστατες, χρονικά μόνιμες ροές συμπιεστού ρευστού, απουσία βαρυτικών δυνάμεων, χρησιμοποιώντας τη μέθοδο τεμνόμενων κυψελών (cut-cells). Ο επιλύτης που αναπτύχθηκε, αντικαθιστά το ήδη υπάρχον λογισμικό στο τμήμα που αφορά μόνο την επίλυση των εξισώσεων ροής, διατηρώντας τις υπόλοιπες λειτουργίες του, όπως είναι η πλεγματοποίηση του υπολογιστικού χωρίου. Επειδή χρησιμοποιήθηκε υπάρχων κώδικας για την πλεγματοποίηση, ο επιλύτης έπρεπε να αλλάξει τον τρόπο αποθήκευσης των γεωμετρικών πληροφοριών για την καλύτερη εκμετάλλευση μερικών δυνατοτήτων των GPUs.

4.1 Αλλαγή Αρίθμησης Κυψελών

Όπως αναφέρθηκε και στην εισαγωγή αυτού του κεφαλαίου, για την πλεγματοποίηση του υπολογιστικού χωρίου σύμφωνα με την μέθοδο τεμνόμενων κυψελών, έγινε χρήση υπάρχοντος κώδικα. Μετά την εκτέλεση του, στη μνήμη της CPU αποθηκεύονται δεδομένα για κάθε κυψέλη, όπως είναι ο αριθμός της (ID), οι συντεταγμένες της, οι διαστάσεις της, οι γειτονικές κυψέλες σε κάθε πλευρά της και άλλα γεωμετρικά χαρακτηριστικά. Οι κυψέλες αυτές χωρίζονται σε ενεργές (σε εκείνες όπου θα λυθούν οι εξισώσεις ροής) και στις μη-ενεργές (σε εκείνες που είναι εξολοκλήρου εντός στερεού, οπότε δεν χρειάζεται να λυθούν οι εξισώσεις). Οι ενεργές κυψέλες, με τη σειρά τους, χωρίζονται σε εκείνες που κόβονται από το στερεό τοίχωμα και εκείνες που δεν κόβονται. Σύμφωνα με την ανάλυση που έχει γίνει στο κεφάλαιο 2, αυτά τα 3 διαφορετικά είδη κυψελών χρήζουν διαφορετικής αντιμετώπισης κατά την επίλυση της ροής.

Στην παράγραφο 3.1 έχει τονιστεί πως κάθε πολυεπεξεργαστής της GPU ομαδοποιεί τα threads σε warps (ομάδες 32 threads) και ότι τα threads του ίδιου warp εκτελούν την ίδια σειρά εντολών ταυτόχρονα. Κάθε thread αντιστοιχεί σε μία κυψέλη και καλείται να υπολογίσει τα μητρώα της εξίσωσης 2.18. Οπότε, αν

χρησιμοποιούταν μια τυχαία αρίθμηση των κυψελών, όπως αυτή που εξάγει ο πλεγματοποιητής που χρησιμοποιήθηκε, τότε θα υπήρχαν στο ίδιο warp threads/κυψέλες που θα ήταν είτε μη-ενεργές είτε θα κόβονταν από το στερεό τοίχωμα είτε θα βρίσκονταν μέσα στη ροή, με αποτέλεσμα να μην εκτελούν παράλληλα τις ίδιες εντολές.

Προκειμένου να επιτευχθεί όσο το δυνατόν καλύτερη παρραλληλοποίηση του επιλύτη για κάρτες γραφικών, επιλέχθηκε η λύση της αλλαγής της αρίθμησης των κυψελών, η οποία θα εξαρτάται από το «είδος» της κυψέλης και όχι από την σειρά με την οποία τη δημιούργησε ο κώδικας πλεγματοποίησης.

Επίσης, επειδή σχηματίζονται ομάδες των 32 threads, το τμήμα του επιλύτη που εκτελείται στην GPU, θεωρεί πως ο αριθμός των κυψελών του πλέγματος είναι ακέραιο πολλαπλάσιο του 32. Αυτή η θεώρηση, δεν επηρεάζει τη δομή του κώδικα, γιατί οι «εικονικές» κυψέλες που δημιουργούνται στην κάρτα γραφικών, τις αντιμετωπίζει ο κώδικας ως να είναι μη-ενεργές κυψέλες, δηλαδή δεν χρειάζεται να λύσει τις εξισώσεις σε αυτές. Ο συνολικός αριθμός κυψελών που θεωρεί η GPU ότι έχει το υπολογιστικό χωρίο είναι $nCellsGPU = 32 * nWarps$, με $nWarps = int(\frac{nCellsCPU+32-1}{32})$. Η επιλογή αυτή δεν αυξάνει σημαντικά τις απαιτήσεις σε μνήμη στη GPU γιατί, στη χειρότερη περίπτωση, η κάρτα γραφικών θεωρεί ότι υπάρχουν 31 παραπάνω κυψέλες.

Επειδή οι εικονικές κυψέλες έχουν ID>=[αριθμός πραγματικών κυψελών], επιλέγεται η αρίθμηση των κυψελών να ξεκινήσει αρχικά με τις ενεργές κυψέλες, στη συνέχεια με τις κυψέλες που κόβονται από το στερεό τοίχωμα και, τέλος, με τις μη-ενεργές. Η επιλογή αυτή έγινε έτσι ώστε στο τελευταίο warp να αποτελείται από «εικονικές» και μη-ενεργές threads/κυψέλες, οι οποίες δεν συμμετέχουν στους υπολογισμούς.

Το αποτέλεσμα όλης αυτής της διαδικασία αλλαγής της αρίθμησης των κυψελών είναι να υπάρχουν μόνο 2 warps, ανεξάρτητα από τον αριθμό των κυψελών, τα οποία έχουν threads που αντιστοιχούν σε διαφορετική κατηγορία κυψελών, οι οποίες χρήζουν διαφορετικής αντιμετώπισης και εκτελούν διαφορετικές εντολές. Για την ακρίβεια, υπάρχουν δύο υποομάδες threads στα warps αυτά, με την κάθε υποομάδα να εκτελεί τις ίδιες εντολές. Η κάθε υποομάδα εκτελεί τις εντολές της παράλληλα, απλώς η δεύτερη υποομάδα περιμένει την πρώτη να εκτελέσει τις δικές της εντολές, για να αρχίσει μετά εκείνη τις δικές της.

4.2 Διαχείριση Γεωμετρικών Ποσοτήτων

Σύμφωνα με την ανάλυση που έγινε στο κεφάλαιο 2, για την λύση της εξίσωσης 2.18 χρειάζεται ο υπολογισμός γεωμετρικών ποσοτήτων για κάθε κυψέλη, όπως το μήκος κάθε πλευράς του, οι προβολές του όγκου στις κατευθύνσεις x και y και το κάθετο διάνυσμα στην πλευρά που κόβεται από το στερεό τοίχωμα.

Στον επιλύτη που εκτελείται από τη CPU, επιλέχθηκε να γίνονται οι παραπάνω γεωμετρικοί υπολογισμοί για κάθε κυψέλη σε κάθε ψευδοχρονική επανάληψη, ώστε να μην αποθηκεύονται πολλά δεδομένα στην μνήμη. Μια τέτοια αντιμετώπιση, όμως, απαιτεί συνεχή δέσμευση και αποδέσμευση τμήματος της μνήμης σε κάθε επανάληψη. Η διαδικασία αυτή επιβραδύνει πολύ περισσότερο την GPU απ’

ότι τη CPU. Έτσι, σε αντίθεση με το υπάρχον λογισμικό επιλέχθηκε να γίνουν όλοι οι απαραίτητοι υπολογισμοί στη CPU μια φορά πριν ξεκινήσουν οι ψευδοχρονικές επαναλήψεις, καθώς οι γεωμετρικές ποσότητες δεν αλλάζουν, αφού το πλέγμα παραμένει το ίδιο. Στη συνέχεια, οι πληροφορίες αυτές αποθηκεύονται στην κεντρική μνήμη της GPU, από την οποία αντλούνται κατευθείαν οι τιμές τους χωρίς να χρειάζεται να γίνονται οι ίδιοι υπολογισμοί σε κάθε επανάληψη. Ο τρόπος με τον οποίο αποθηκεύονται οι πληροφορίες αυτές στην κεντρική μνήμη της κάρτας γραφικών αναλύεται στην επόμενη παράγραφο.

4.3 Αποθήκευση Πινάκων Δύο ή Περισσότερων Διαστάσεων

Όπως σε όλα τα προβλήματα της Υπολογιστικής Ρευστοδυναμικής, έτσι και σε αυτό είναι απαραίτητη η αποθήκευση πινάκων δύο ή περισσότερων διαστάσεων στην κεντρική μνήμη της κάρτας γραφικών. Πίνακας δύο διαστάσεων χρειάζεται για την αποθήκευση των πρωτευουσών μεγεθών της ροής σε κάθε κυψέλη, της διόρθωσης $\Delta\vec{U}$ που υπολογίζεται με τη μέθοδο Jacobi, των γεωμετρικών χαρακτηριστικών που χρειάζονται στους υπολογισμούς κάθε κυψέλης (έγινε αναφορά στην παράγραφο 4.2) και του υπολοίπου $\vec{R}$. Οι πίνακες αυτοί (εκτός των γεωμετρικών ποσοτήτων) έχουν διάσταση 4x(αριθμός κυψελών στη GPU), ενώ για τα γεωμετρικά χαρακτηριστικά έχει διάσταση (αριθμός απαραίτητων γεωμετρικών χαρακτηριστικών για κάθε κυψέλη)x(αριθμός κυψελών στη GPU). Πίνακες περισσοτέρων διαστάσεων χρειάζονται για την αποθήκευση των μητρώων $\frac{\partial\overrightarrow{R_P}}{\partial\overrightarrow{U_P}}$ και $\frac{\partial\overrightarrow{R_P}}{\partial\overrightarrow{U_Q}}$. Για το πρώτο μητρώο ο πίνακας, που θέλουμε να αποθηκευτεί, έχει διάσταση 4x4x(αριθμός κυψελών στη GPU), ενώ για το δεύτερο μητρώο 4x4x(αριθμός γειτόνων κυψέλης)x(αριθμός κυψελών στη GPU).

Ανεξάρτητα από τη διάσταση του πίνακα που πρέπει να αποθηκευτεί, είτε είναι δύο ή τριών ή περισσοτέρων διαστάσεων, επιλέγεται να γίνει η αποθήκευση στη κεντρική μνήμη της κάρτας γραφικών ως διάνυσμα, δηλαδή ως πίνακας μιας μόνο διάστασης. Στο σημείο αυτό υπενθυμίζεται πως οι GPUs αντιμετωπίζουν την κεντρική μνήμη ως μία αλληλουχία από τμήματα μήκους 32, 64 ή 128 Bytes. Δηλαδή είναι δυνατό την ίδια στιγμή να διαβαστούν από την κεντρική μνήμη 32 συνεχόμενες θέσεις μνήμης με αποθηκευμένο πραγματικό αριθμό.

Με την αλλαγή αρίθμησης των κυψελών, όπως περιγράφτηκε στην παράγραφο 4.1, έχει επιτευχθεί όλα τα threads/κυψέλες του ίδιου warp να εκτελούν την ίδια εντολή. Επομένως, κατά τη διάρκεια ενός υπολογισμού, και τα 32 threads θα ζητούν την τιμή για παράδειγμα της πυκνότητας που είναι αποθηκευμένη στη κεντρική μνήμη και δεν θα ζητά κάποιο thread την τιμή της πίεσης. Σε συνδυασμό με τον τρόπο ανάγνωσης δεδομένων από την κεντρική μνήμη, επιλέγεται η αποθήκευση του πίνακα δύο διαστάσεων στην κάρτα γραφικών να γίνει αποθηκεύοντας αρχικά τις πυκνότητες των 32 threads του πρώτου warp, μετά τη ταχύτητα u κατά την x κατεύθυνση, έπειτα η ταχύτητα v κατά την y κατεύθυνση και τέλος η πίεση. Στη συνέχεια, αποθηκεύεται η πυκνότητα των 32 threads του δεύτερου warp κ.ο.κ., μέχρις ότου να αποθηκευτούν και τα τέσσερα πρωτεύοντα μεγέθη για όλες τις κυψέλες της GPU. Επειδή σε κάποιους υπολογισμούς χρειάζονται τα

συντηρητικά μεγέθη, γίνεται αντικατάσταση του πρωτεύοντος με το αντίστοιχο συντηρητικό μέγεθος στην ίδια θέση μνήμης, οπότε δεν χρειάζεται να δεσμευτούν παραπάνω θέσεις μνήμης. Στο σχήμα 4.1 φαίνεται γραφικά ο παραπάνω τρόπος αποθήκευσης, ενώ στο σχήμα 4.2 ο συνήθης τρόπος αποθήκευσης πινάκων 2 διαστάσεων στη CPU. Η ίδια διαδικασία ακολουθείται και για τους άλλους πίνακες δύο διαστάσεων. Επομένως, το i πρωτεύον μέγεθος της ID κυψέλης είναι αποθηκευμένο στην θέση $index = 4 * 32 * iWarp + i * 32 + iThread$ του διανύσματος που αποθηκεύτηκε στην κεντρική μνήμη της GPU, όπου $iWarp = ID/32$ (το πηλίκο της Ευκλείδειας διαίρεσης), $iThread = ID\%32$ (το υπόλοιπο της Ευκλείδειας διαίρεσης), $i = \{0,1,2,3\}$ και $ID = \{0,1,2,\dots,N-1\}$.

πυκνότητα $0^{ου}$ thread $0^{ου}$ warp	πυκνότητα $1^{ου}$ thread $0^{ου}$ warp	**…**	πυκνότητα $31^{ου}$ thread $0^{ου}$ warp	ταχύτητα u $0^{ου}$ thread $0^{ου}$ warp	**…**	πίεση $31^{ου}$ thread $0^{ου}$ warp	πυκνότητα $0^{ου}$ thread $1^{ου}$ warp	**…**
πυκνότητα $31^{ου}$ thread $1^{ου}$ warp	ταχύτητα u $0^{ου}$ thread $1^{ου}$ warp	**…**	πίεση $31^{ου}$ thread $1^{ου}$ warp	πυκνότητα $0^{ου}$ thread $2^{ου}$ warp	**…**	πίεση $31^{ου}$ thread $2^{ου}$ warp	**…**	πίεση $31^{ου}$ thread $(N-1)^{ου}$ warp

Σχήμα 4.1: Γραφική απεικόνιση αποθήκευσης του μητρώου των πρωτευουσών μεγεθών ως διάνυσμα στη GPU. (ακολουθείται η αρίθμηση πινάκων της C/C++)

πυκνότητα $0^{ης}$ κυψέλης	ταχύτητα u $0^{ης}$ κυψέλης	ταχύτητα v $0^{ης}$ κυψέλης	πίεση $0^{ης}$ κυψέλης
πυκνότητα $1^{ης}$ κυψέλης	ταχύτητα u $1^{ης}$ κυψέλης	ταχύτητα v $1^{ης}$ κυψέλης	πίεση $1^{ης}$ κυψέλης
πυκνότητα $2^{ης}$ κυψέλης	ταχύτητα u $2^{ης}$ κυψέλης	ταχύτητα v $2^{ης}$ κυψέλης	πίεση $2^{ης}$ κυψέλης
πυκνότητα $3^{ης}$ κυψέλης	ταχύτητα u $3^{ης}$ κ κυψέλης	ταχύτητα v $3^{ης}$ κυψέλης	πίεση $3^{ης}$ κυψέλης
…	**…**	**…**	**…**
πυκνότητα $(N-1)^{ης}$ κυψέλης	ταχύτητα u $(N-1)^{ης}$ κυψέλης	ταχύτητα v $(N-1)^{ης}$ κυψέλης	πίεση $(N-1)^{ης}$ κυψέλης

Σχήμα 4.2: Γραφική απεικόνιση συνήθους τρόπου αποθήκευσης του μητρώου των πρωτευουσών μεγεθών ως πίνακα δύο διαστάσεων στη CPU. (ακολουθείται η αρίθμηση πινάκων της C/C++)

Παρόμοια διαδικασία ακολουθείται και για την αποθήκευση πινάκων τριών διαστάσεων, όπως είναι το μητρώο $\frac{\partial \overrightarrow{R_P}}{\partial \overrightarrow{U_P}}$. Στην περίπτωση αυτή, αρχικά αποθηκεύουμε το στοιχείο (0,0) και για τα 32 threads του πρώτου warp, στη συνέχεια το (0,1) και κ.ο.κ.. Η ίδια διαδικασία ακολουθείται και για τα υπόλοιπα warps. Το στοιχείο (i,j) της ID κυψέλης είναι αποθηκευμένο στην θέση $index = 4 * 4 * 32 * iWarp + (i * 4 + j) * 32 + iThread$, όπου iWarp και iThread υπολογίζονται όπως και στην περίπτωση πίνακα δύο διαστάσεων, $i = \{0,1,2,3\}$, $j = \{0,1,2,3\}$ και $ID = \{0,1,2,\dots,N-1\}$. Στο σχήμα 4.3 φαίνεται γραφικά ο τρόπος αποθήκευσης ενός πίνακα τριών διαστάσεων.

Το μητρώο $\frac{\partial \overrightarrow{R_P}}{\partial \overrightarrow{U_Q}}$ είναι ένας πίνακας τεσσάρων διαστάσεων. Η δυσκολία με την αποθήκευση του μητρώου ως διάνυσμα στην κάρτα γραφικών, σε σύγκριση με τις προηγούμενες περιπτώσεις, είναι στο ότι δεν έχει κάθε κυψέλη τον ίδιο αριθμό γειτόνων. Η κάθε κυψέλη μπορεί να έχει το πολύ 8 γειτονικές κυψέλες, σύμφωνα με τη διαδικασία πλεγματοποίησης που ακολουθεί ο κώδικας που χρησιμοποιήθηκε. Οπότε για να διατηρηθεί η ομοιομορφία και στα 32 threads/κυψέλες του κάθε warp, θα μπορούσε να θεωρηθεί πως όλες οι κυψέλες έχουν 8 γείτονες αφήνοντας κενές τις θέσεις σε περίπτωση που μία κυψέλη έχει λιγότερους γείτονες. Κάτι τέτοιο θα οδηγούσε σε μεγάλη δέσμευση μνήμης στην κάρτα γραφικών. Προκειμένου να μειωθεί όσο το δυνατόν η απαίτηση σε μνήμη και ο επιλύτης να μπορεί να χρησιμοποιηθεί και σε μεγαλύτερα πλέγματα, επιλέχθηκε να υπολογίζεται ο μέγιστος αριθμός γειτόνων των 32 threads/κυψελών κάθε warp. Έτσι αν σε κάποιο warp ο μέγιστος αριθμός γειτόνων είναι το 6, τότε εξοικονομούνται $(8-6)*4*4*32*8\,Bytes = 8192\,Bytes = 8\,KB$ μνήμης. Υπενθυμίζεται ότι ένας πραγματικός αριθμός τύπου double καταλαμβάνει 8 Bytes μνήμης.

Έχοντας γίνει η παραπάνω απαραίτητη προεργασία, μπορεί να γίνει αποθήκευση του μητρώου ακολουθώντας παρόμοια λογική με αυτή για πίνακες τριών διαστάσεων. Αρχικά θα αποθηκευτεί το στοιχείο (0,0) του πρώτου γείτονα και για τα 32 threads του πρώτου warp, στη συνέχεια το (0,1) και αφού αποθηκευτεί και το στοιχείο (3,3) και των 32 threads του warp για τον πρώτο γείτονα, αποθηκεύεται το στοιχείο (0,0) του δεύτερου γείτονα, ύστερα τα 16 στοιχεία για τον τρίτο γείτονα κ.ο.κ.. Μόλις αποθηκευτούν όλα τα στοιχεία για όλους τους γείτονες των 32 threads του πρώτου warp, συνεχίζεται η ίδια διαδικασία και για τα υπόλοιπα warps. Το στοιχείο (i,j) της ID κυψέλης για τον nb γείτονα είναι αποθηκευμένο στην θέση $index = 4*4*32*SumWarpNeighbours[iWarp-1] + nb*4*4*32 + (i*4+j)*32 + iThread$, όπου iWarp και iThread υπολογίζονται όπως και πριν, $i = \{0,1,2,3\}$, $j = \{0,1,2,3\}$, SumWarpNeighbours[iWarp-1] είναι το άθροισμα του μέγιστου αριθμού γειτόνων όλων των προηγούμενων warps και $ID = \{0,1,2,\dots,N-1\}$.

Η εντολή για τη δέσμευση των απαραίτητων θέσεων μνήμης των παραπάνω διανυσμάτων στην κεντρική μνήμη της κάρτα γραφικών είναι η *cudaMalloc()*. Περισσότερες πληροφορίες για τη σύνταξή της υπάρχουν στη βιβλιογραφία [32].

D(0,0) $0^{ου}$ thread $0^{ου}$ warp	D(0,0) $1^{ου}$ thread $0^{ου}$ warp	…	D(0,0) $31^{ου}$ thread $0^{ου}$ warp	D(0,1) $0^{ου}$ thread $0^{ου}$ warp	…	D(0,1) $31^{ου}$ thread $0^{ου}$ warp	D(0,2) $0^{ου}$ thread $0^{ου}$ warp	…
D(0,2) $31^{ου}$ thread $0^{ου}$ warp	D(0,3) $0^{ου}$ thread $0^{ου}$ warp	…	D(0,3) $31^{ου}$ thread $0^{ου}$ warp	D(1,0) $0^{ου}$ thread $0^{ου}$ warp	…	D(1,0) $31^{ου}$ thread $0^{ου}$ warp	D(1,1) $0^{ου}$ thread $0^{ου}$ warp	…
D(1,1) $31^{ου}$ thread $0^{ου}$warp	D(1,2) $0^{ου}$ thread $0^{ου}$warp	…	D(3,3) $31^{ου}$ thread $0^{ου}$ warp	D(0,0) $0^{ου}$ thread $1^{ου}$ warp	…	D(3,3) $31^{ου}$ thread $1^{ου}$ warp	…	D(3,3) $31^{ου}$ thread $(N-1)^{ου}$ warp

Σχήμα 4.3: Γραφική απεικόνιση αποθήκευσης του πίνακα D τριών διαστάσεων ως διάνυσμα στη GPU. (ακολουθείται η αρίθμηση πινάκων της C/C++)

4.4 Διαχωρισμός σε Kernels

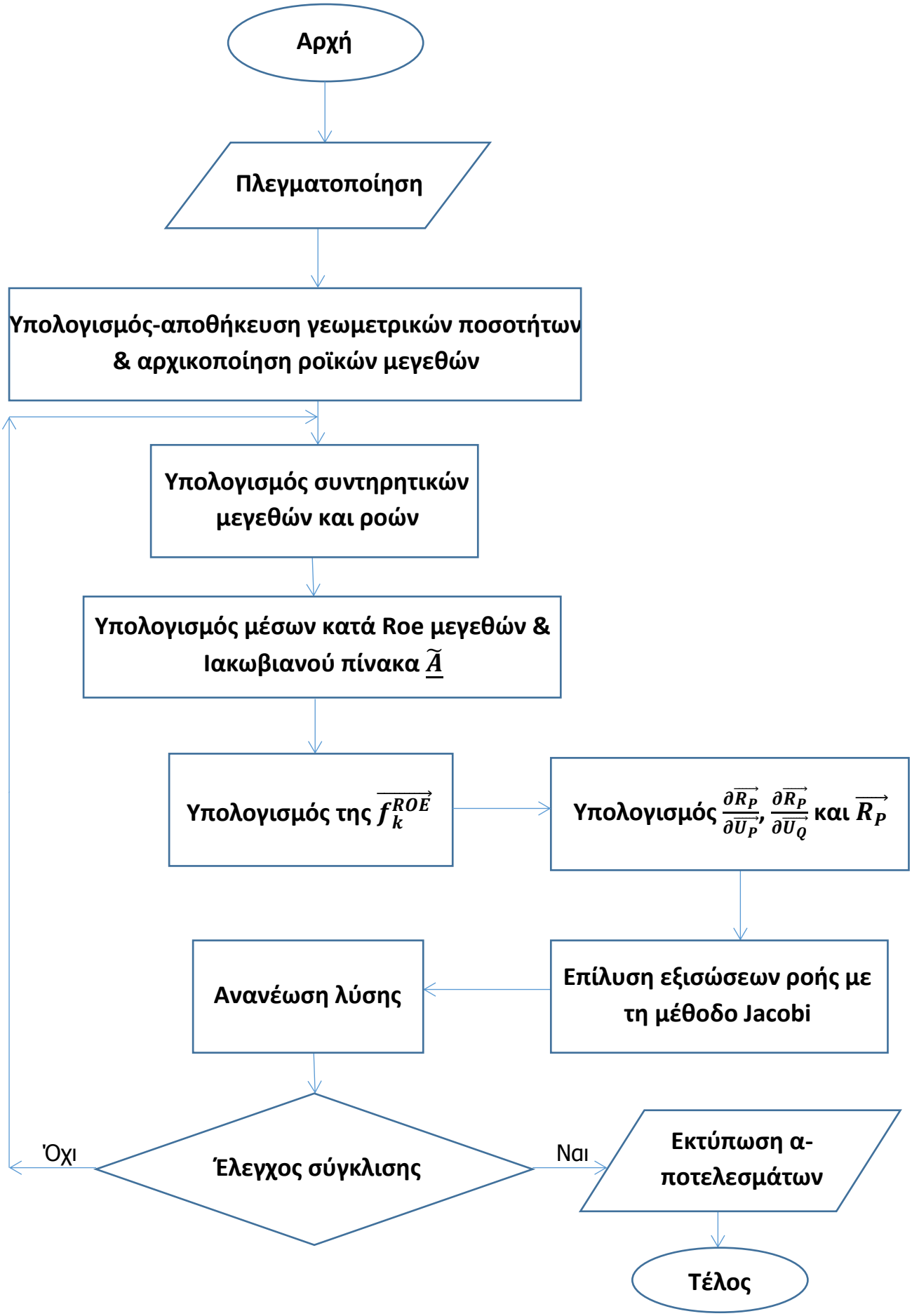
Αφού έχουν ακολουθηθεί οι απαραίτητες ενέργειες που περιγράφτηκαν προηγουμένως, στη κεντρική μνήμη της GPU έχουν δεσμευτεί θέσεις μνήμης για να αποθηκεύονται εκεί τα πρωτεύοντα μεγέθη της ροής για κάθε κυψέλη, η διόρθωση $\Delta\vec{U}$, το υπόλοιπο $\vec{R}$ και τα στοιχεία των μητρώων $\frac{\partial\vec{R_P}}{\partial\vec{U_P}}$ και $\frac{\partial\vec{R_P}}{\partial\vec{U_Q}}$ καθώς επίσης και τα γεωμετρικά χαρακτηριστικά που χρειάζονται στους υπολογισμούς κάθε κυψέλης. Πλέον βρίσκονται όλα τα απαραίτητα μεγέθη που χρειάζονται για τους υπολογισμούς αποθηκευμένα στην κάρτα γραφικών και έχουν δεσμευτεί οι απαραίτητες θέσεις μνήμης για την αποθήκευση των τιμών των μητρώων της εξίσωσης 2.18, οπότε μπορεί να ξεκινήσει η επίλυση. Ο αλγόριθμος επίλυσης εκτελείται από δύο kernels. Το πρώτο είναι υπεύθυνο για τον υπολογισμό των μητρώων του συστήματος (σχέση 2.18) και το δεύτερο για την επίλυσή του.

Στο πρώτο kernel, αρχικά, υπολογίζονται τα συντηρητικά μεγέθη της ροής $\overrightarrow{\boldsymbol{U_P}}$, τη ροή $\overrightarrow{\boldsymbol{f_k^P}}$ και ο Ιακωβιανός πίνακας $\underline{\boldsymbol{A^P}}$. Στη συνέχεια προστίθεται στα διαγώνια στοιχεία του μητρώου $\frac{\partial\overrightarrow{\boldsymbol{R_P}}}{\partial\overrightarrow{\boldsymbol{U_P}}}$ ο όρος $\frac{\boldsymbol{\Omega_P}}{\boldsymbol{\Delta t_P}}$. Κατόπιν, σε κάθε πλευρά της κυψέλης, υπολογίζονται τα συντηρητικά μεγέθη της ροής των γειτονικών κυψελών $\overrightarrow{\boldsymbol{U_Q}}$ και η ροή τους $\overrightarrow{\boldsymbol{f_k^Q}}$. Κατόπιν υπολογίζονται τα μέσα κατά Roe μεγέθη στη διεπιφάνεια των δύο κυψελών και ο Ιακωβιανός πίνακας $\underline{\widetilde{\boldsymbol{A}}}$. Πλέον είναι υπολογισμένα όλα τα απαραίτητα μεγέθη της σχέση 2.9, οπότε υπολογίζεται η $\overrightarrow{\boldsymbol{f_k^{ROE}}}$. Μετά τους παραπάνω υπολογισμούς, σε κάθε πλευρά της κυψέλης, σύμφωνα με τις σχέσεις 2.20 και 2.21, έχουν υπολογιστεί και τα μητρώα $\frac{\partial\overrightarrow{\boldsymbol{R_P}}}{\partial\overrightarrow{\boldsymbol{U_P}}}$ και $\frac{\partial\overrightarrow{\boldsymbol{R_P}}}{\partial\overrightarrow{\boldsymbol{U_Q}}}$ και το υπόλοιπο $\overrightarrow{\boldsymbol{R_P}}$. Στο τέλος του πρώτου kernel υπολογίζεται το αντίστροφο μητρώο $\left[\frac{\partial\overrightarrow{\boldsymbol{R_P}}}{\partial\overrightarrow{\boldsymbol{U_P}}}\right]^{-1}$.

Στο δεύτερο kernel γίνεται η επίλυση του συστήματος 2.18 σύμφωνα με την μέθοδο Jacobi. Σε κάθε επανάληψη της μεθόδου υπολογίζεται νέα διόρθωση $\Delta\vec{U}$, η οποία χρησιμοποιείται στην επόμενη επανάληψη. Μετά το πέρας 15 επαναλήψεων, η τελική διόρθωση που έχει υπολογιστεί, χρησιμοποιείται για την ανανέωση των συντηρητικών μεγεθών $\vec{U}$.

Η παραπάνω διαδικασία επαναλαμβάνεται για όσες ψευδοχρονικές επαναλήψεις έχει επιλέξει ο χρήστης, λαμβάνοντας υπόψη το πλήθος των κυψελών που απαρτίζουν το πλέγμα και το βαθμό σύγκλισης της λύσης που θέλει να επιτύχει. Μετά τη σύγκλιση του επιλύτη, τα πρωτεύοντα μεγέθη της ροής αποθηκεύονται στη μνήμη της CPU ώστε να μπορούν να αποθηκευτούν σε αρχείο και να γίνει η γραφική απεικόνιση του πεδίου.

Στο σχήμα 4.4 φαίνεται το λογικό διάγραμμα του παραπάνω αλγόριθμου, στον οποίο στηρίχτηκε ο επιλύτης της παρούσας εργασίας.



Σχήμα 4.4: Λογικό διάγραμμα του αλγορίθμου επίλυσης της ροής.

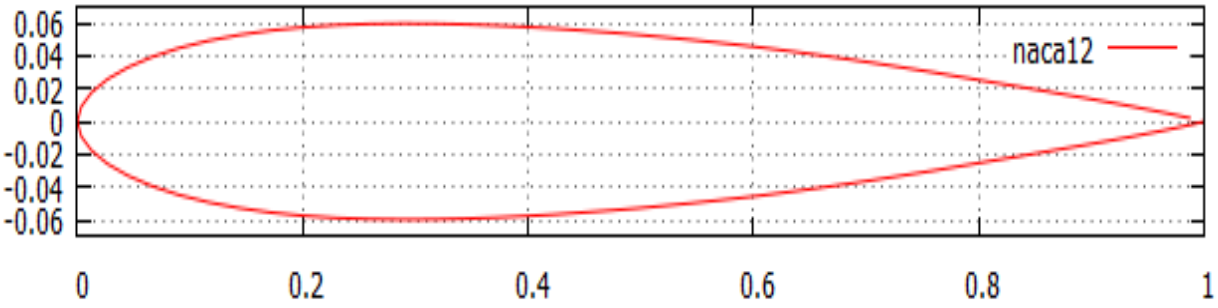
Κεφάλαιο 5

Αποτελέσματα και Συγκριτικές Επιδόσεις

Στο παρόν κεφάλαιο παρουσιάζονται τα αποτελέσματα που προέκυψαν από την εκτέλεση του προγραμματισθέντος επιλύτη. Χρησιμοποιήθηκαν διαφορετικές αεροτομές, διαφορετική πύκνωση του πλέγματος και διαφορετικές συνθήκες ροής (αριθμός Mach και γωνία προσβολής). Εκτός από τον έλεγχο της πιστής αναπαραγωγής αποτελεσμάτων με εκείνα του επιλύτη που εκτελείται στη CPU, εξετάζεται και ο χρόνος εκτέλεσης του και γίνεται σύγκριση μεταξύ των δύο επιλυτών. Υπενθυμίζεται πως ο επιλύτης της παρούσας εργασίας εκτελέστηκε στο server της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών του ΕΜΠ, ο οποίος είχε τον τετραπύρηνο επεξεργαστή Intel Core i5-3570, χρονισμένο στα 3.4GHz και κάρτα γραφικών την NVIDIA GeForce GTX 670 (περισσότερες πληροφορίες στην παράγραφο 3.6).

5.1 Μεμονωμένη Αεροτομή NACA 0012

Η πρώτη αεροτομή που χρησιμοποιήθηκε για την εκτέλεση του επιλύτη ήταν η NACA 0012 (σχήμα 5.1). Μελετήθηκε για ροή μηδενικής γωνίας προσβολής και ταχύτητας V=200m/s. Πρόκειται για μια συμμετρική αεροτομή με απλή γεωμετρία. Λόγω της συμμετρίας της και της μηδενικής γωνίας προσβολής, γνωρίζουμε πως το πεδίο ροής που θα σχηματιστεί γύρω από αυτή πρέπει να είναι συμμετρικό. Επίσης, δοκιμάστηκαν διάφορες πυκνώσεις του πλέγματος, ώστε να συγκριθούν οι 2 επιλύτες και να βγουν διάφορα συμπεράσματα για τον χρόνο εκτέλεσης του επιλύτη καθώς αυξάνεται το μέγεθος του πλέγματος.



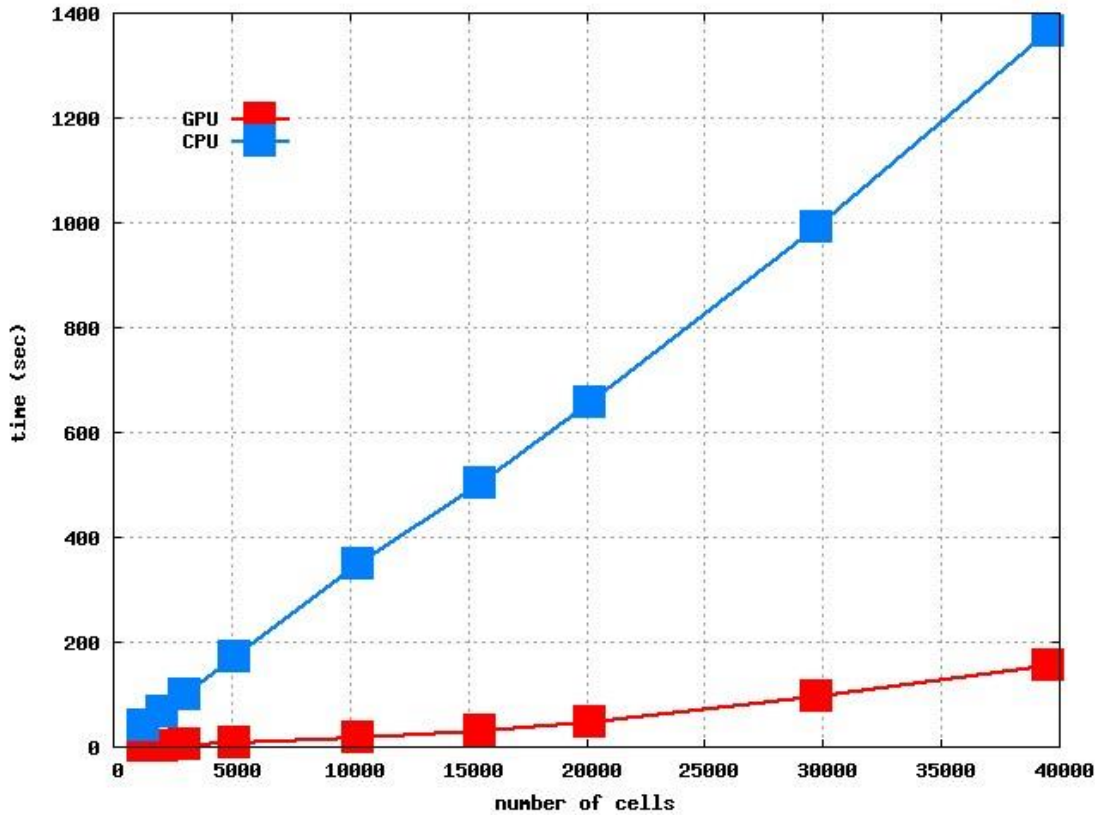
Σχήμα 5.1: Αεροτομή NACA 0012.

Η περίπτωση αυτή του προβλήματος επιλύθηκε 9 φορές και από τους δύο επιλύτες για διαφορετικής πύκνωσης πλέγματα. Και στα 9 διαφορετικά πλέγματα επιλέχτηκε να γίνουν 800 ψευδοχρονικά βήματα. Στη μέτρηση του χρόνου εκτέλεσης τόσο του CPU όσο και του GPU επιλύτη έχει εξαιρεθεί ο χρόνος που απαιτείται για τη δημιουργία του πλέγματος, καθώς ο χρόνος αυτός είναι ο ίδιος και στα δύο λογισμικά. Στον επιλύτη που εκτελείται στη GPU υπολογίζεται ο συνολικός χρόνος επίλυσης, αλλά και ο χρόνος που απαιτείται για την δέσμευση μνήμης για τους πίνακες (βλ. παράγραφο 4.3) στην κεντρική μνήμη της κάρτας γραφικών. Τα αποτελέσματα αυτά παρουσιάζονται στον πίνακα 5.1 και στα σχήματα 5.2, 5.3, 5.4 και 5.5.

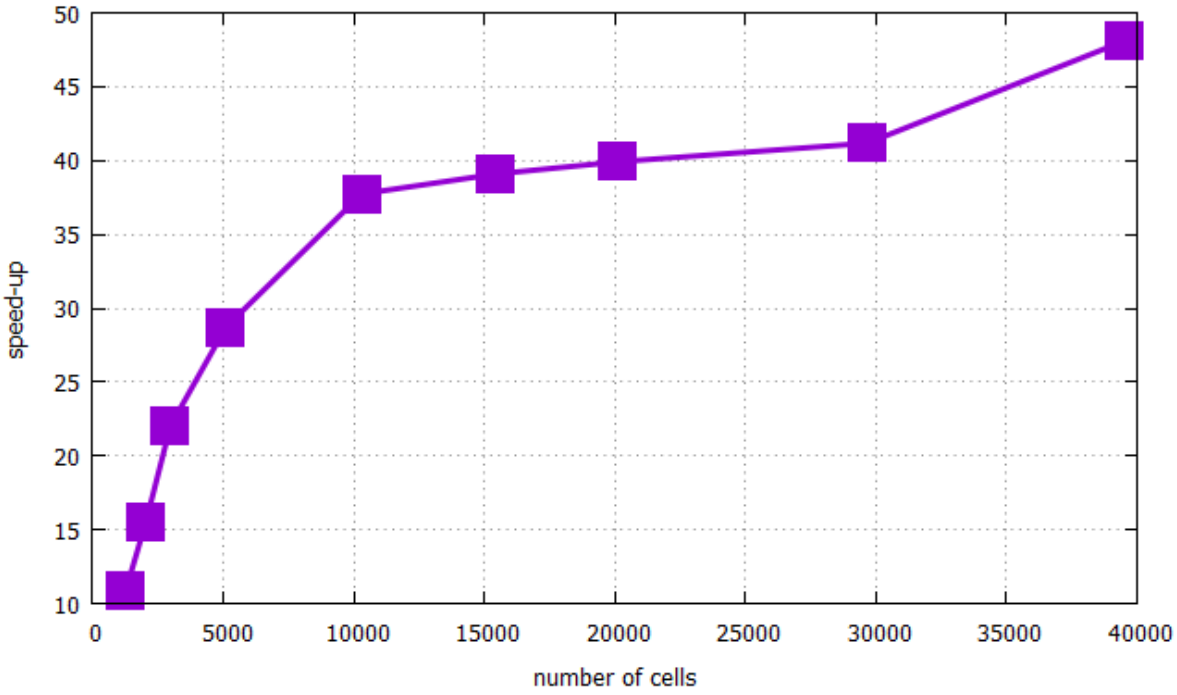
Επισημαίνεται πως ο επιλύτης της CPU εκτελέστηκε στον τετραπύρηνο επεξεργαστή Intel Xeon E5620[33] χρονισμένο στα 2.4 GHz.

Αριθμός Κυψελών	Χρόνος Εκτέλεσης Επιλύτη GPU (sec)	Χρόνος Δέσμευσης Μνήμης Πινάκων GPU (sec)	Χρόνος Εκτέλεσης Επιλύτη CPU (sec)
1264	4.1	0.14	42.99
2044	4.88	0.38	69.95
2980	5.37	0.82	100.18
5104	8.34	2.31	173.21
10312	18.29	8.96	352.15
15448	33.57	20.64	505.42
20080	50.23	33.71	659.28
29668	97.6	73.46	994.07
39508	157.47	129.02	1368.33

Πίνακας 5.1: Χρόνοι εκτέλεσης των δύο επιλυτών για διαφορετικής πύκνωσης πλέγματα.



Σχήμα 5.2: Διάγραμμα χρόνου εκτέλεσης επιλύτη σε CPU και GPU συναρτήσει του αριθμού των κυψελών του πλέγματος. Όσο μεγαλύτερο είναι το πλέγμα, τόσο μεγαλύτερη η διαφορά μεταξύ των δύο χρόνων εκτέλεσης.



Σχήμα 5.3: Διάγραμμα επιτάχυνσης χρόνου εκτέλεσης επιλύτη στη GPU συναρτήσει του αριθμού των κυψελών του πλέγματος. Εδώ συγκρίνεται μόνο ο χρόνος που απαιτείται για τις αριθμητικές πράξεις και δεν λαμβάνεται υπόψη ο χρόνος που απαιτείται για την αποθήκευση των πινάκων στη κάρτα γραφικών.

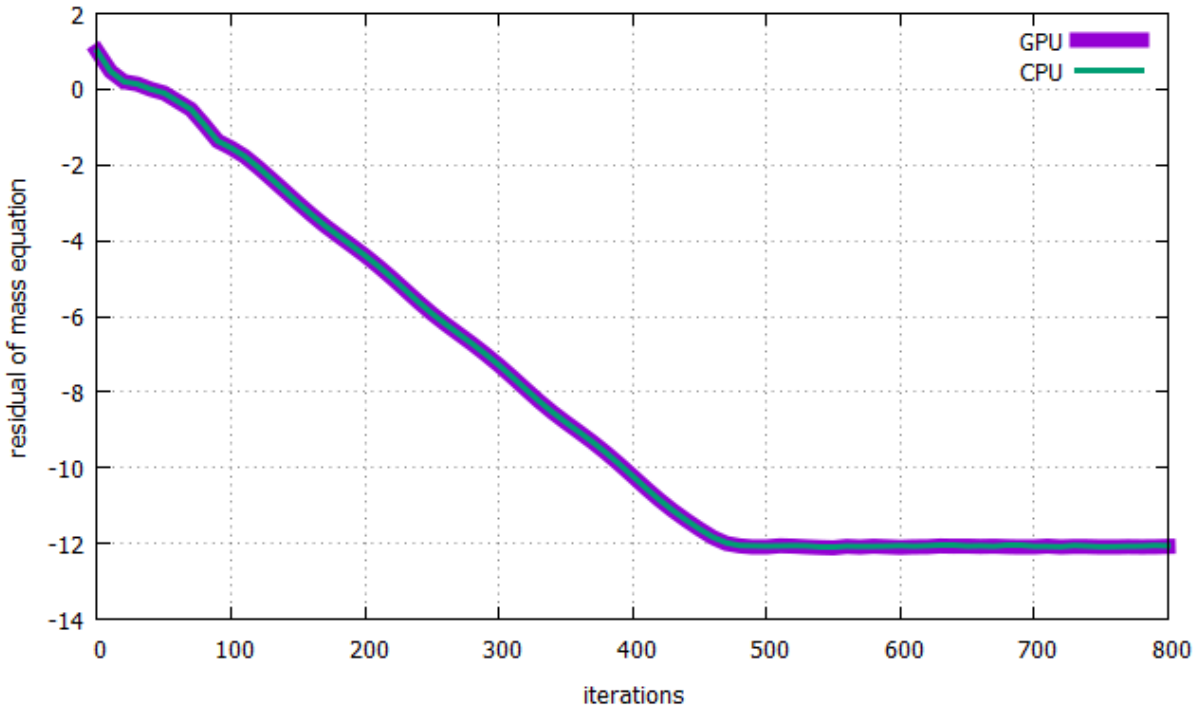
$$[speed-up = \frac{Χρόνος\ Εκτέλεσης\ Επιλύτη\ στη\ CPU}{(Χρόνος\ Εκτέλεσης\ Επιλύτη\ στη\ GPU)-(Χρόνος\ Δέσμευσης\ Μνήμης\ στη\ GPU)}]$$

Όπως ήταν αναμενόμενο, παρατηρείται από τα παραπάνω σχήματα ότι ο επιλύτης εκτελείται πιο γρήγορα στη GPU απ' ότι στη CPU. Από το σχήμα 5.2 παρατηρούμε πως ο χρόνος εκτέλεσης του στη CPU αυξάνεται γραμμικά με την αύξηση των αριθμών των κυψελών του πλέγματος, ενώ ο αντίστοιχος χρόνος στη GPU αυξάνεται με πολύ μικρότερο ρυθμό.

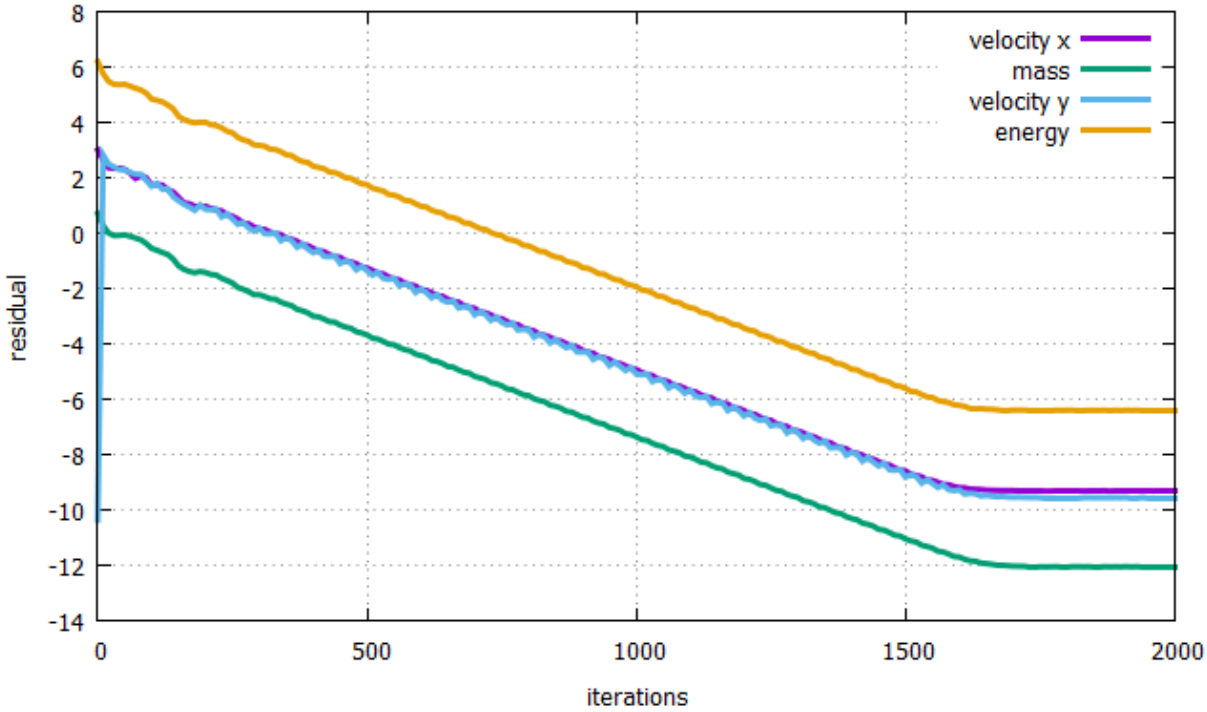
Στα σχήμα 5.3 παρουσιάζεται η επιτάχυνση που επιτεύχθηκε στον χρόνο εκτέλεσης του επιλύτη. Για τον υπολογισμό της επιτάχυνσης δεν λαμβάνεται υπόψη ο χρόνος δέσμευσης μνήμης για τους πίνακες, δηλαδή γίνεται σύγκριση του χρόνου που απαιτείται για την εκτέλεση των αριθμητικών πράξεων. Επομένως, από το σχήμα 5.3 καταλήγουμε στο ότι ο επιλύτης της παρούσας εργασίας πέτυχε επιτάχυνση πάνω από 40 φορές. Να τονιστεί πως στη παρούσα εργασία δόθηκε έμφαση στην πιστή αναπαραγωγή των αποτελεσμάτων και την επιτάχυνση των υπολογισμών και όχι τόσο στη σωστή δέσμευση μνήμης.

Από τα όσα παρουσιάστηκαν προηγουμένως, παρατηρήθηκε ότι ο ένας στόχος της παρούσας εργασίας επιτεύχθηκε, δηλαδή να μειωθεί ο χρόνος εκτέλεσης σε σχέση με τον επιλύτη που εκτελείται στη CPU. Ο άλλος στόχος ήταν να επιτευχθεί πιστή αναπαραγωγή των αποτελεσμάτων, κάτι το οποίο φαίνεται από τους ταυτόσημους ρυθμούς σύγκλισης στις δύο περιπτώσεις (σχήμα 5.4).

Στο σχήμα 5.5 παρουσιάζεται ο ρυθμός σύγκλισης και των 4 εξισώσεων που διέπουν την ροή. Πρόκειται για την περίπτωση πλέγματος 39508 κυψελών κατά την διάρκεια 2000 επαναλήψεων. Στις 2 εξισώσεις διατήρησης της ορμής πετυχαίνεται ο ίδιος βαθμός σύγκλισης.

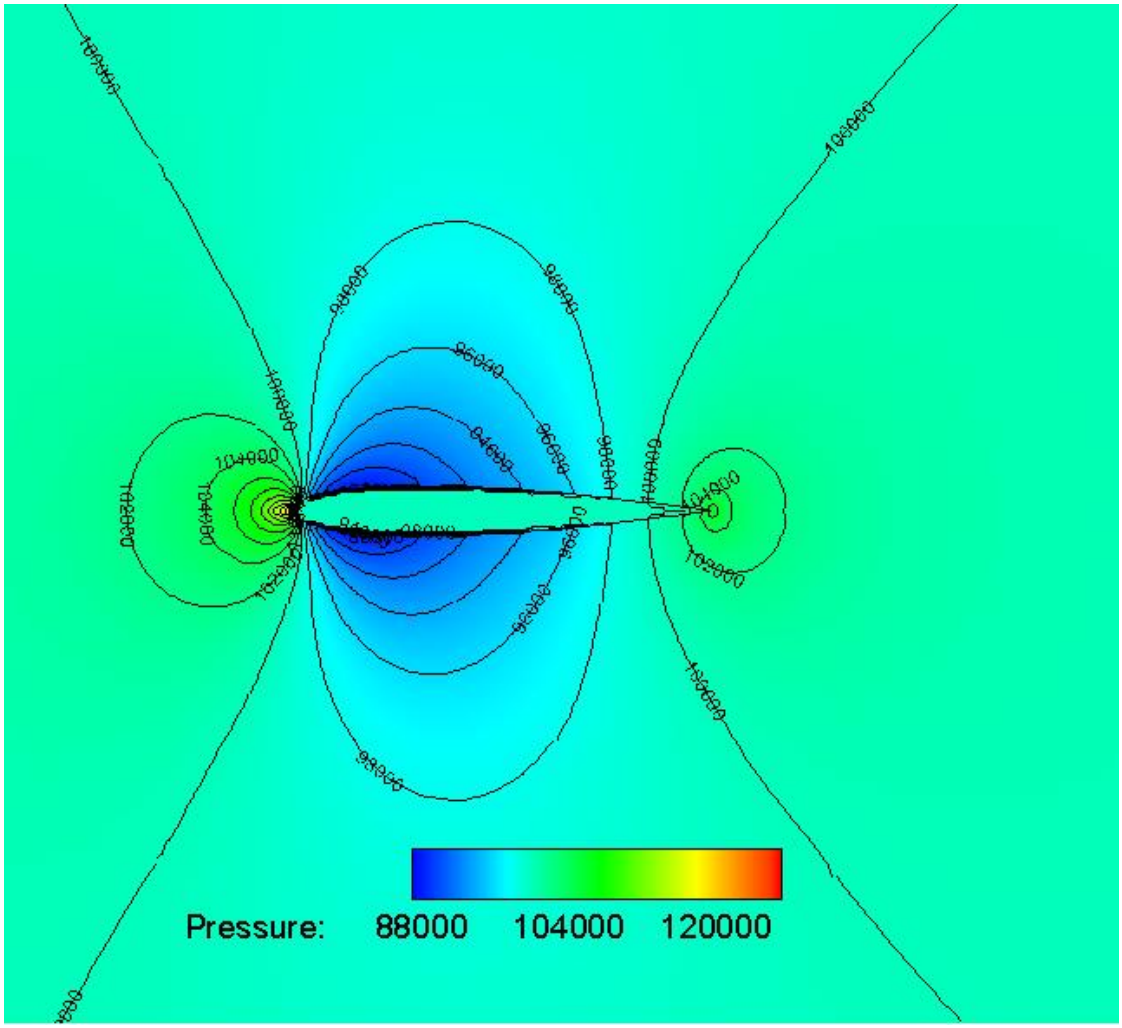


Σχήμα 5.4: Διάγραμμα ρυθμού σύγκλισης του επιλύτη για CPU και GPU για περίπτωση πλέγματος 5104 κυψελών και 800 ψευδοχρονικών επαναλήψεων. Από την ταύτιση των δύο καμπυλών συμπεραίνεται η πιστή αναπαραγωγή αποτελεσμάτων από τον επιλύτη της παρούσας εργασίας.

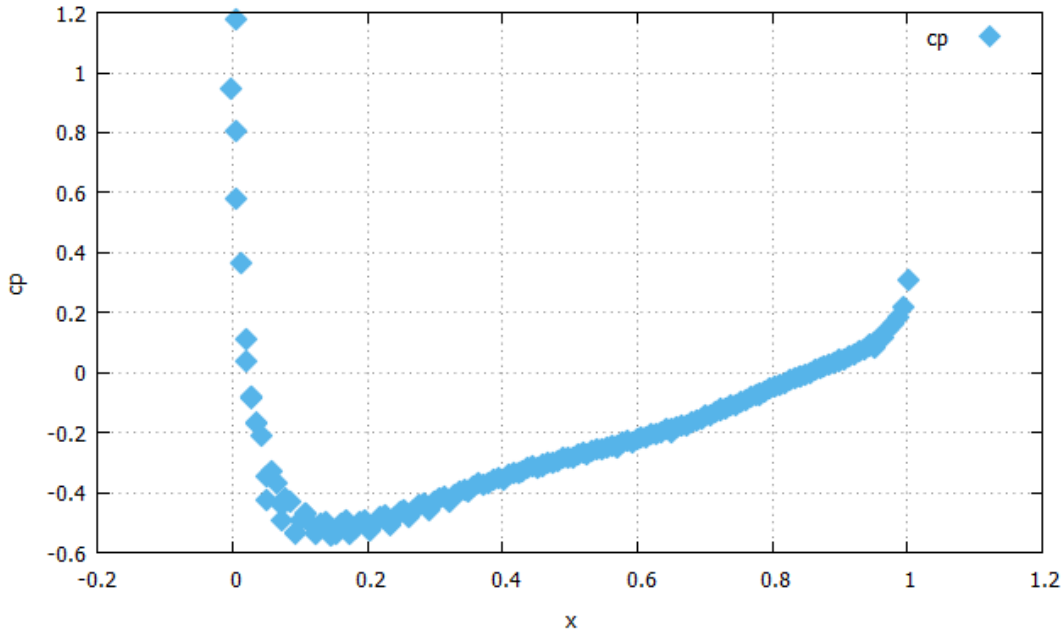


Σχήμα 5.5: Διάγραμμα ρυθμού σύγκλισης των τεσσάρων εξισώσεων για την περίπτωση πλέγματος 39508 κυψελών για την αεροτομή NACA 0012.

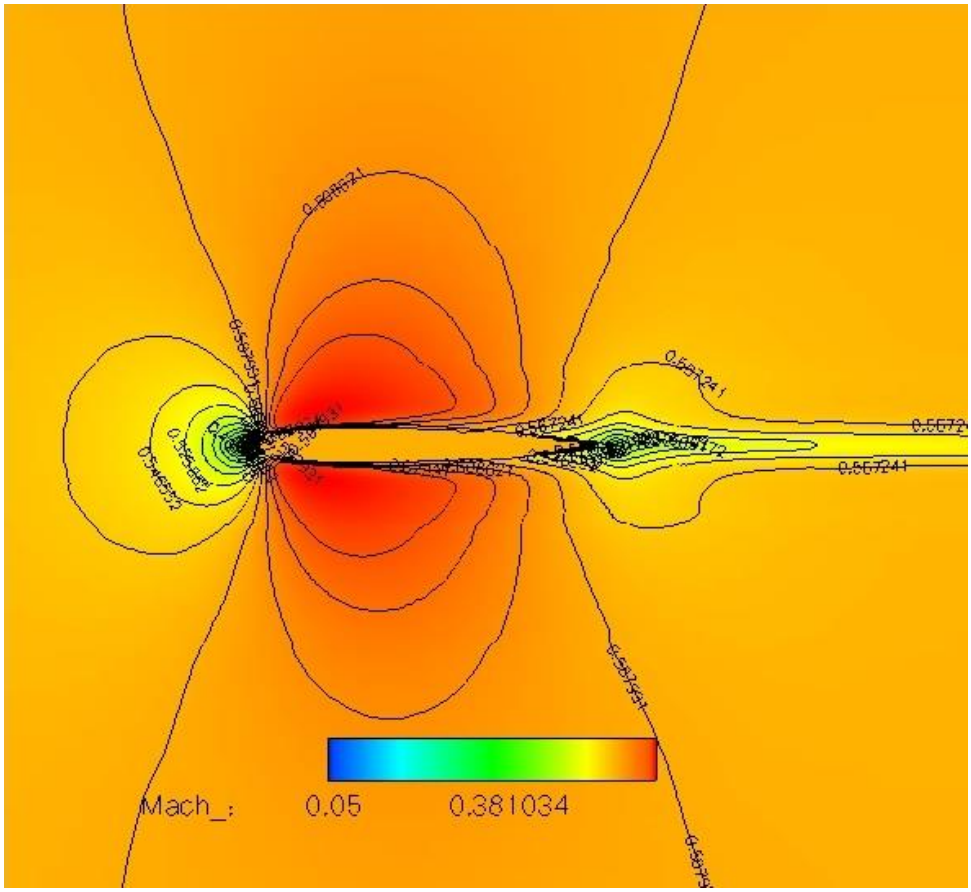
Στα επόμενα σχήμα που ακολουθούν (5.6 – 5.8) παρουσιάζονται τα αποτελέσματα της επίλυσης στο χωρίο γύρω από την αεροτομή. Πρόκειται για την περίπτωση που έχει σχηματιστεί πλέγμα 39508 κυψελών (σχήμα 5.9) και πραγματοποιήθηκαν 2000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση. Στο σχήμα 5.6 παρουσιάζεται η πίεση στο ρευστό γύρω από την αεροτομή. Λόγω της συμμετρίας της αεροτομής και της μηδενικής γωνίας προσβολής, η πίεση είναι συμμετρικά κατανεμημένη στο υπολογιστικό χωρίο. Η μέγιστη τιμή πίεσης παρατηρείται στην ακμή προσβολής που είναι σημείο αποκοπής. Το ίδιο αποτέλεσμα βλέπουμε και στο σχήμα 5.7 όπου παρουσιάζεται ο συντελεστής πίεσης γύρω από την αεροτομή. Στο σχήμα 5.8 βλέπουμε τον αριθμό Mach. Όπως είναι γνωστό από την θεωρία λεπτών αεροτομών, στο εμπρός τμήμα της αεροτομής έχουμε επιτάχυνση της ροής και στη συνέχεια επιβράδυνση.



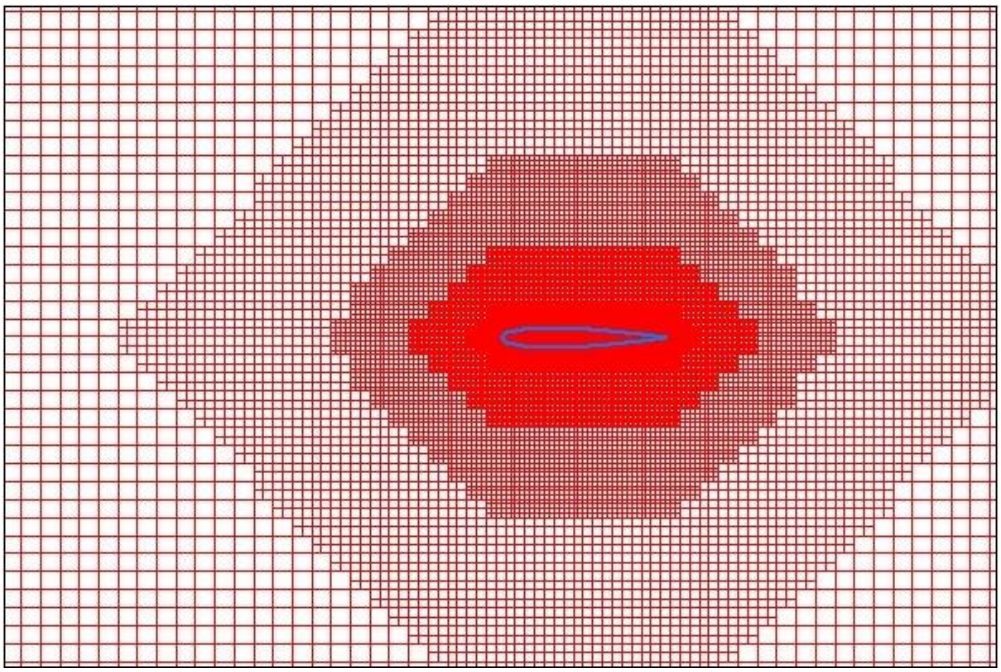
Σχήμα 5.6: Διάγραμμα αναπαράστασης της πίεσης στο χωρίο γύρω από την αεροτομή NACA 0012. Το αποτέλεσμα αυτό προέκυψε μετά από 2000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 39508 κυψελών. Στην ακμή προσβολής παρατηρείται η μέγιστη τιμή της πίεσης καθώς εκεί βρίσκεται το σημείο αποκοπής, ενώ στο εμπρός τμήμα της αεροτομής έχουμε πτώση της, καθώς εκεί επιταχύνεται η ροή.



Σχήμα 5.7: Διάγραμμα κατανομής του συντελεστή πίεσης γύρω από την συμμετρική αεροτομή NACA 0012. Επειδή η ροή έχει μηδενική γωνία πρόσπτωσης, όπως είναι γνωστό, δεν δημιουργείται άνωση, οπότε η πίεση (άρα και ο συντελεστής πίεσης) είναι συμμετρικά κατανεμημένη γύρω από αυτήν.



Σχήμα 5.8: Διάγραμμα αναπαράστασης του αριθμού Mach στο χωρίο γύρω από την αεροτομή NACA 0012. Το αποτέλεσμα αυτό προέκυψε μετά από 2000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 39508 κυψελών. Γίνεται αντιληπτό το σημείο αποκοπής στην ακμή προσβολής, αλλά και η επιτάχυνση της ροής στο εμπρός μέρος της αεροτομής.



Σχήμα 5.9: Το πλέγμα 39508 κυψελών, όπως δημιουργήθηκε γύρω από την αεροτομή NACA 0012 από τον κώδικα πλεγματοποίησης. Χαρακτηριστική είναι η πύκνωση του πλέγματος κοντά στην αεροτομή, προκειμένου να επιτευχθεί μεγαλύτερη ακρίβεια στον υπολογισμό των μεγεθών της ροής.

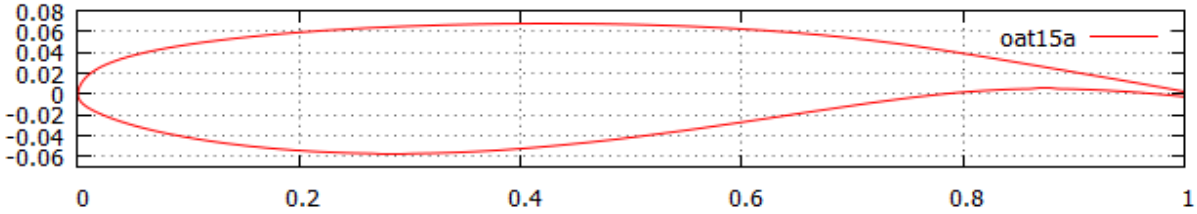
5.2 Αεροτομή OAT 15A

Η δεύτερη αεροτομή που χρησιμοποιήθηκε για την εκτέλεση του επιλύτη ήταν η OAT 15A (σχήμα 5.10). Μελετήθηκε για ροή με γωνία προσβολής $\boldsymbol{\alpha_\infty = 2.5^o}$ και Mach $\boldsymbol{M_\infty = 0.73}$. Πρόκειται για μια μη-συμμετρική αεροτομή σε σχέση με την προηγούμενη περίπτωση και σε αυτές τις συνθήκες ροής αναμένεται από πειράματα που έχουν γίνει (βλ. βιβλιογραφία [34]) να εμφανιστεί κύμα κρούσης στην άνω πλευρά της στη θέση x=0.6 περίπου.

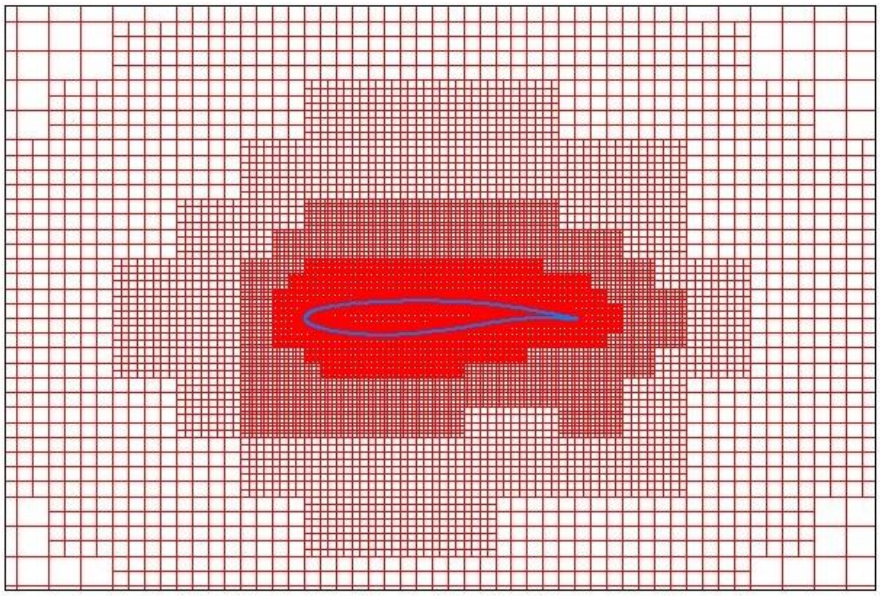
Το πλέγμα που δημιουργήθηκε για την επίλυση της ροής γύρω από αυτήν την αεροτομή αποτελείται από 30328 κυψελών (σχήμα 5.11), ενώ επιλέχθηκε να γίνουν 3000 ψευδοχρονικές επαναλήψεις για να επιτευχθεί ικανοποιητική σύγκλιση. Στο σχήμα 5.12 παρουσιάζεται ο ρυθμός σύγκλισης για το προαναφερθέν πλέγμα όταν ο επιλύτης εκτελείται στη CPU και στη GPU. Παρατηρείται ότι στη CPU έχουμε σύγκλιση στις 1500 επαναλήψεις, ενώ στη GPU μετά από άλλες 200 περίπου επαναλήψεις. Από τον πίνακα 5.2, όπου παρουσιάζονται οι συνολικοί χρόνοι εκτέλεσης των δύο επιλυτών, συμπεραίνουμε ο επιλύτης σε GPU θα έφτανε σε σύγκλιση πάλι αρκετά γρηγορότερα σε πραγματικό χρόνο.

Στο σχήμα 5.13 παρουσιάζεται ο ρυθμός σύγκλισης και των 4 εξισώσεων που διέπουν τη ροή. Στις 2 εξισώσεις διατήρησης της ορμής πετυχαίνεται ο ίδιος βαθμός σύγκλισης.

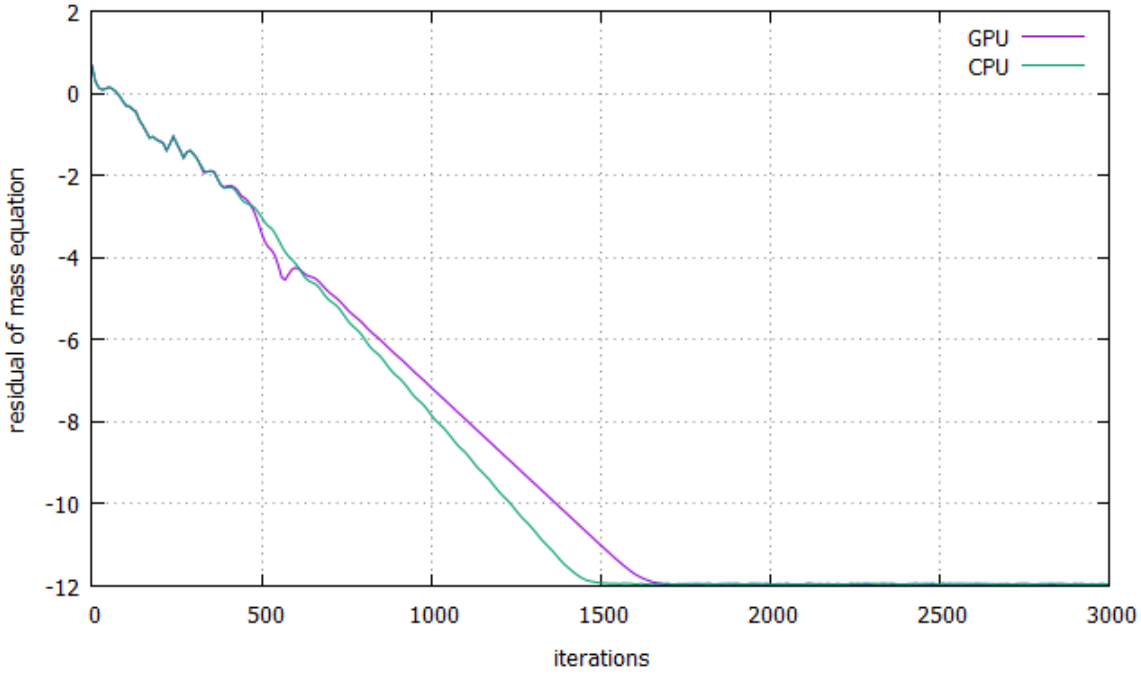
Σημειώνεται ότι ο πιστοποιημένος επιλύτης εκτελέστηκε στον τετραπύρηνο επεξεργαστή Intel Xeon E5620[33] χρονισμένο στα 2.4 GHz.



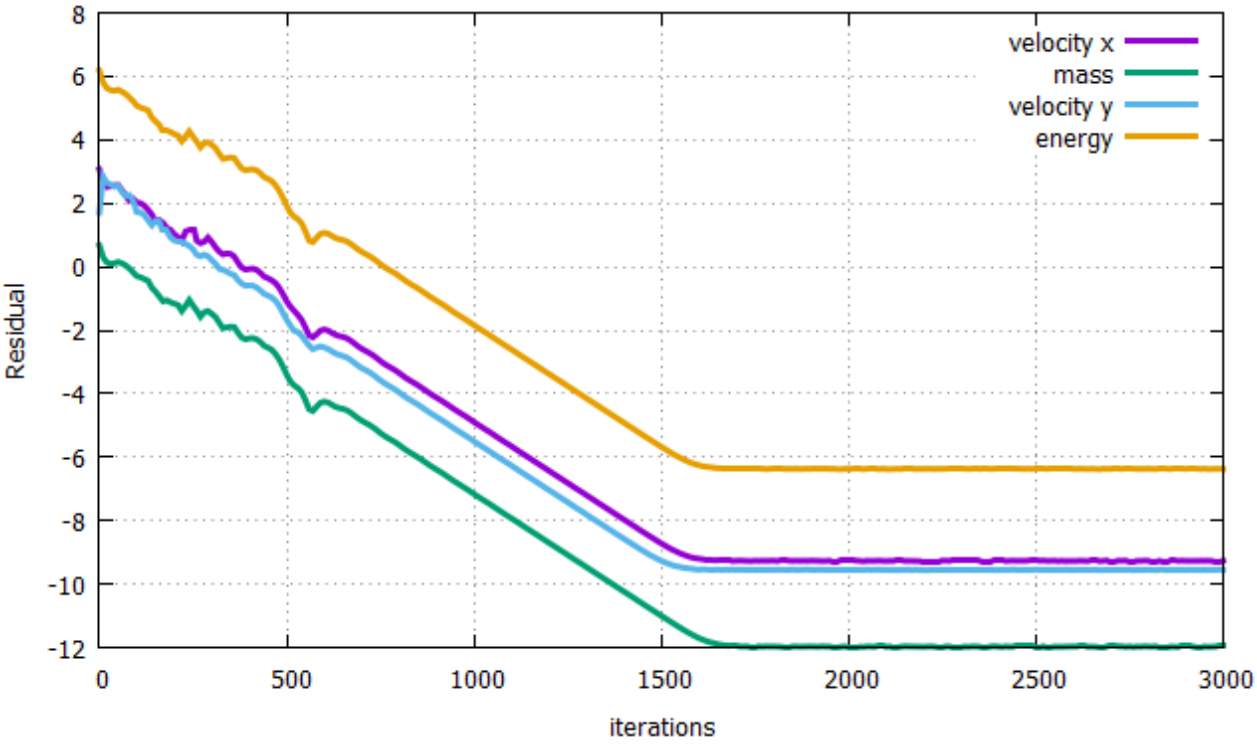
Σχήμα 5.10: Αεροτομή OAT 15A.



Σχήμα 5.11: Το πλέγμα 30328 κυψελών, όπως δημιουργήθηκε από τον κώδικα πλεγματοποίησης γύρω από την αεροτομή OAT 15A. Χαρακτηριστική είναι η πύκνωση του πλέγματος κοντά στην αεροτομή, προκειμένου να επιτευχθεί μεγαλύτερη ακρίβεια στον υπολογισμό των μεγεθών της ροής.



Σχήμα 5.12: Διάγραμμα ρυθμού σύγκλισης του επιλύτη για CPU και GPU για πλέγμα 30328 κυψελών και 3000 ψευδοχρονικών επαναλήψεων. Παρατηρείται μια μικρή απόκλιση των δύο καμπύλων, όμως στα τελικά αποτελέσματα της επίλυσης της ροής υπάρχει πλήρης ταύτιση.



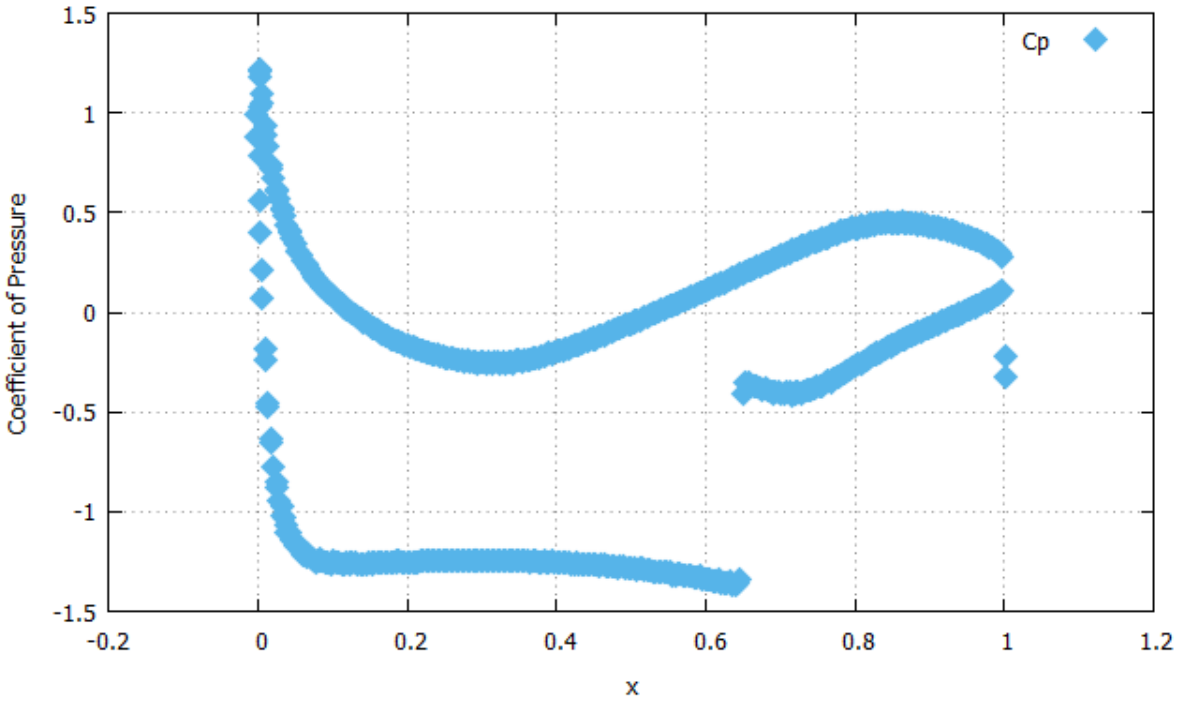
Σχήμα 5.13: Διάγραμμα ρυθμού σύγκλισης των τεσσάρων εξισώσεων για την περίπτωση πλέγματος 30328 κυψελών για την αεροτομή OAT 15A.

Στον πίνακα 5.2 παρατηρούμε ότι για τη συγκεκριμένη αεροτομή για το πλέγμα του σχήματος 5.11, επετεύχθη επιτάχυνση στο συνολικό χρόνο εκτέλεσης του κώδικα πάνω από 20 φορές. Μάλιστα η επίδοση αυτή είναι καλύτερη από την περίπτωση της αεροτομής NACA 0012 για ίδιο βαθμό πύκνωσης του πλέγματος. Αυτό οφείλεται στο ότι έγιναν σε αυτή την περίπτωση περισσότερες επαναλήψεις από όσες χρειαζόταν για να υπάρξει σύγκλιση, σε αντίθεση με την προηγούμενη περίπτωση όπου ο επιλύτης δεν είχε προλάβει να συγκλίνει στις 800 επαναλήψεις. Όσον αφορά τον χρόνο που απαιτείται μόνο για τους υπολογισμούς, δηλαδή αφαιρώντας τον χρόνο που απαιτείται για τη δέσμευση μνήμης για τους πίνακες, παρουσιάστηκε επιτάχυνση μεγαλύτερη από 37 φορές. Και σε αυτήν την περίπτωση ο χρόνος αυτός καταλαμβάνει ένα σημαντικό μέρος του συνολικού χρόνου εκτέλεσης, συγκεκριμένα το 46%.

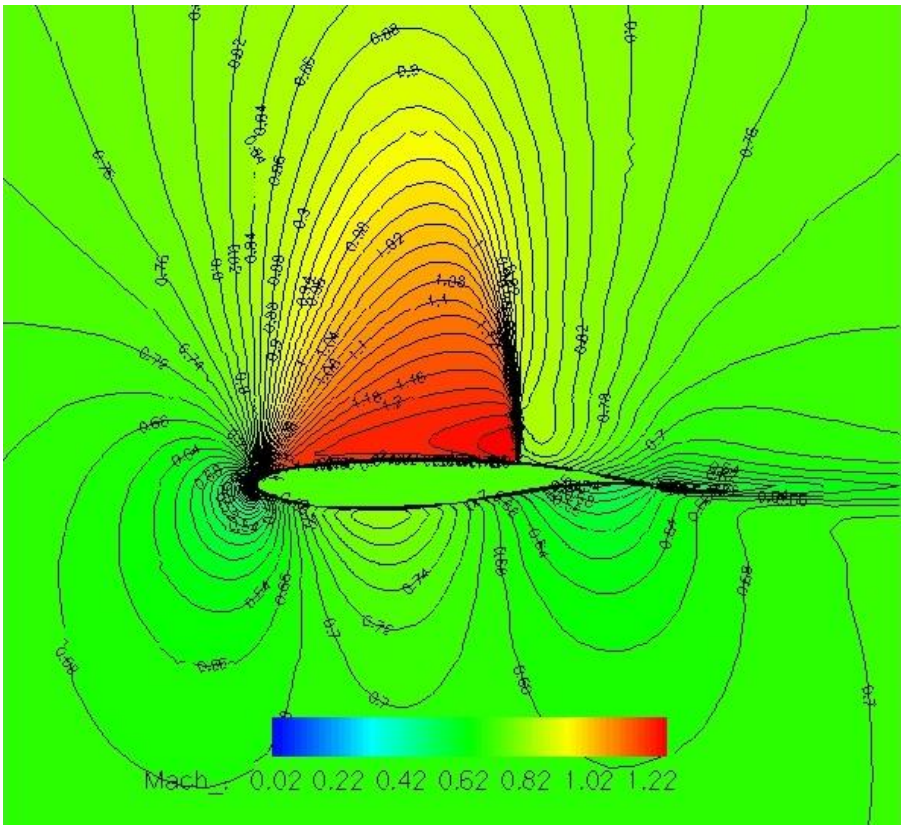
Στη συνέχεια, παρουσιάζονται τα αποτελέσματα που προέκυψαν από τον επιλύτη που εκτελείται στη GPU.

Χρόνος Εκτέλεσης Επιλύτη GPU (sec)	Χρόνος Δέσμευσης Μνήμης Πινάκων GPU (sec)	Χρόνος Εκτέλεσης Επιλύτη CPU (sec)
169.39	77.81	3464.66

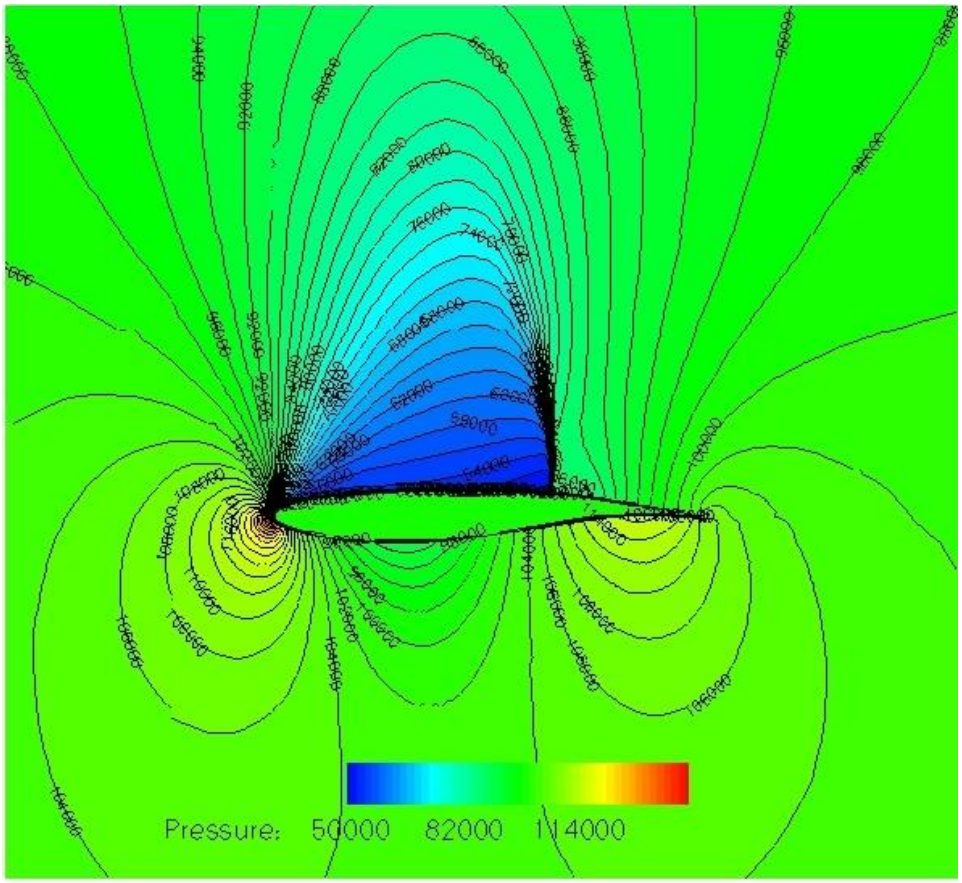
Πίνακας 5.2: Χρόνοι εκτέλεσης των δύο επιλυτών για πλέγμα 30328 κυψελών για την περίπτωση της αεροτομής OAT 15A. Παρατηρείται επιτάχυνση της επίλυσης πάνω από 37 φορές.



Σχήμα 5.14: Διάγραμμα κατανομής του συντελεστή πίεσης γύρω από την αεροτομή OAT 15A. Η ασυνέχεια που παρατηρείται στο διάγραμμα στη θέση x=0.65 οφείλεται στη δημιουργία κάθε-του κύματος κρούσης, κάτι που αναμενόταν λόγω των συνθηκών ροής (βλ. βιβλιογραφία [34]).



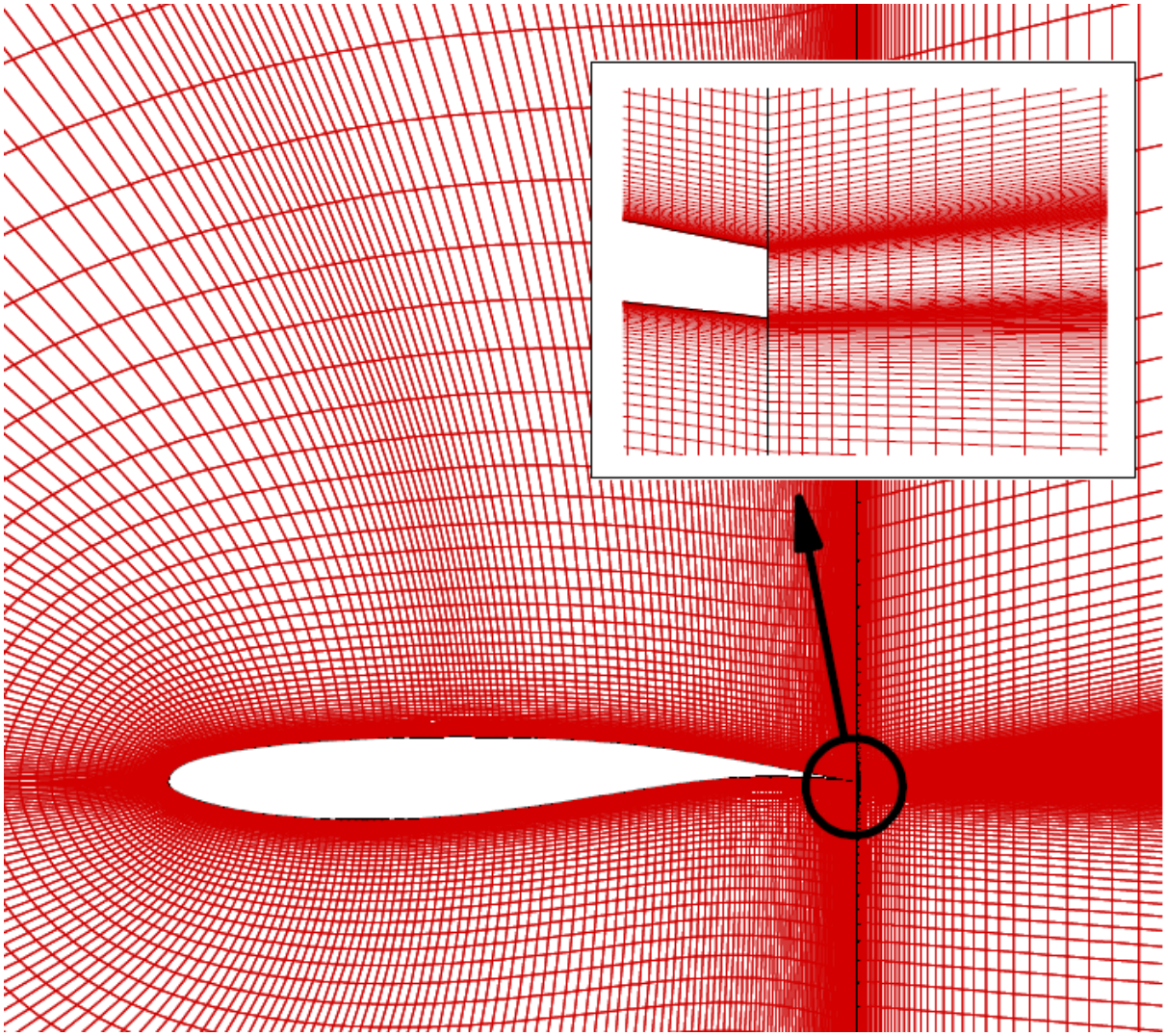
Σχήμα 5.15: Διάγραμμα αναπαράστασης του αριθμού Mach στο πεδίο γύρω από την αεροτομή OAT 15A. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 30328 κυψελών. Καθώς επιταχύνεται η ροή στο εμπρός της αεροτομής, στην πάνω πλευρά ξεπερνάει το M=1, οπότε στη θέση x=0.6 δημιουργείται κάθετο κύμα κρούσης.



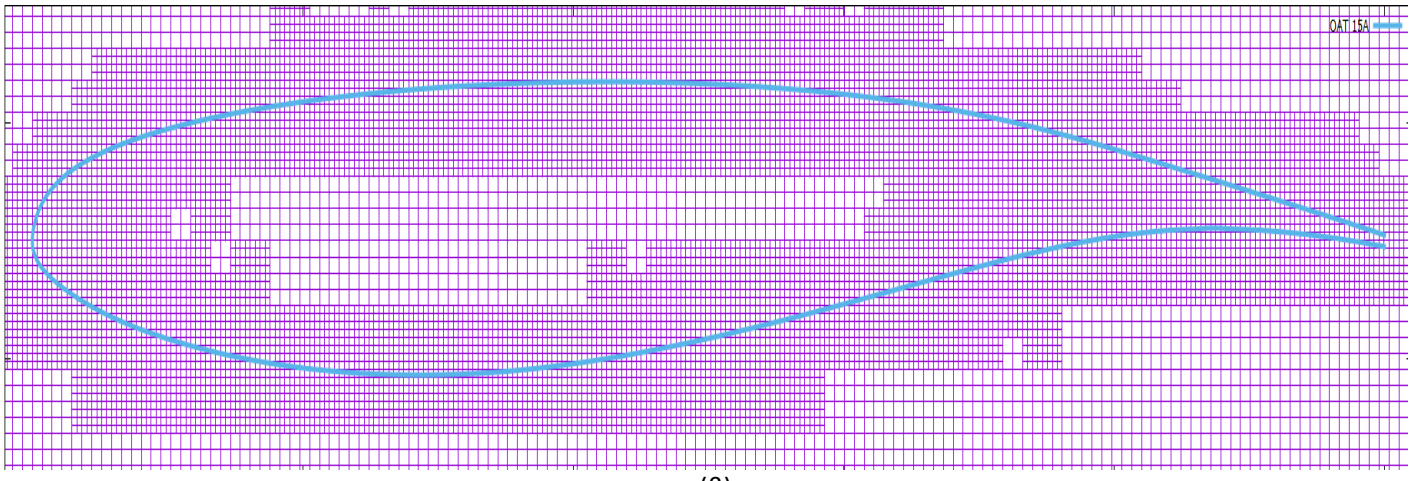
Σχήμα 5.16: Διάγραμμα αναπαράστασης της πίεσης στο χωρίο γύρω από την αεροτομή OAT 15A. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 30328 κυψελών. Στην κάτω πλευρά της αεροτομής κοντά στην ακμή προσβολής παρατηρείται η μέγιστη τιμή της πίεσης καθώς εκεί βρίσκεται το σημείο αποκοπής. Επίσης στην πάνω πλευρά παρατηρείται μεγάλη πτώση της πίεσης λόγω των υπερηχητικών ταχυτήτων που αναπτύσσονται και στο σημείο x=0.6 συμβαίνει απότομη αύξηση της (ασυνέχεια) λόγω του κάθετου κύματος κρούσης.

Στο σχήμα 5.14 παρουσιάζεται το διάγραμμα κατανομής του συντελεστή πίεσης γύρω από την αεροτομή. Στη θέση x=0.65 στην πάνω πλευρά της αεροτομής παρατηρείται μια ασυνέχεια. Αυτή οφείλεται στην ύπαρξη κάθετου κύματος κρούσης, όπως συμπεραίνουμε παρατηρώντας και τα σχήματα 5.15 και 5.16. Στο σχήμα 5.15 παρουσιάζεται ο αριθμός Mach στο πεδίο γύρω από την αεροτομή. Στο εμπρός τμήμα της αεροτομής έχουμε επιτάχυνση της ροής, η οποία αποκτά υπερηχητική ταχύτητα, μέχρι το σημείο x=0.65, όπου έχουμε απότομη μείωση της λόγω του κάθετου κύματος κρούσης και αντίστοιχα απότομη αύξηση της πίεσης στο σχήμα 5.16. Στο διάγραμμα αναπαράστασης της πίεσης παρατηρείται και το σημείο ανακοπής στη κάτω πλευρά της αεροτομής, κοντά στην ακμή προσβολής.

Τέλος, στο σχήμα 5.17β φαίνεται πιο λεπτομερώς το πλέγμα που προέκυψε από τη μέθοδο τεμνόμενων κυψελών (παράγραφος 2.3) γύρω από την αεροτομή. Στο σχήμα 5.17α παρουσιάζεται το σύνηθες δομημένο πλέγμα που χρησιμοποιείται για την επίλυση τέτοιου είδους αεροτομών (C-type).



(α)



(β)

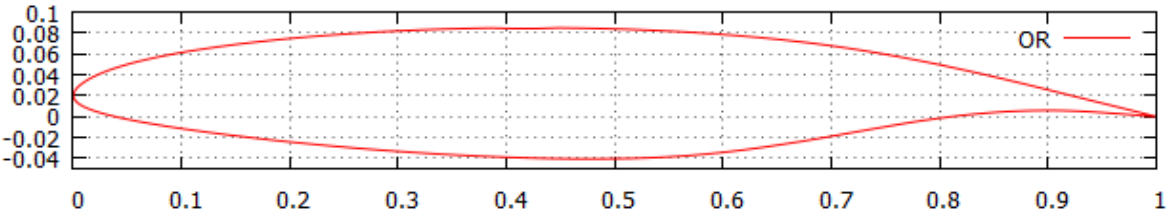
Σχήμα 5.17: Παρουσιάζονται δύο διαφορετικά είδη πλέγματος για την επίλυση της ροής γύρω από την αεροτομή OAT 15A. α) Δομημένο C-Type πλέγμα [34]. β) Καρτεσιανό πλέγμα βασισμένο στη μέθοδο τεμνόμενων κυψελών (cut-cells).

5.3 Αεροτομή OR με Μαζεμένο το Flap

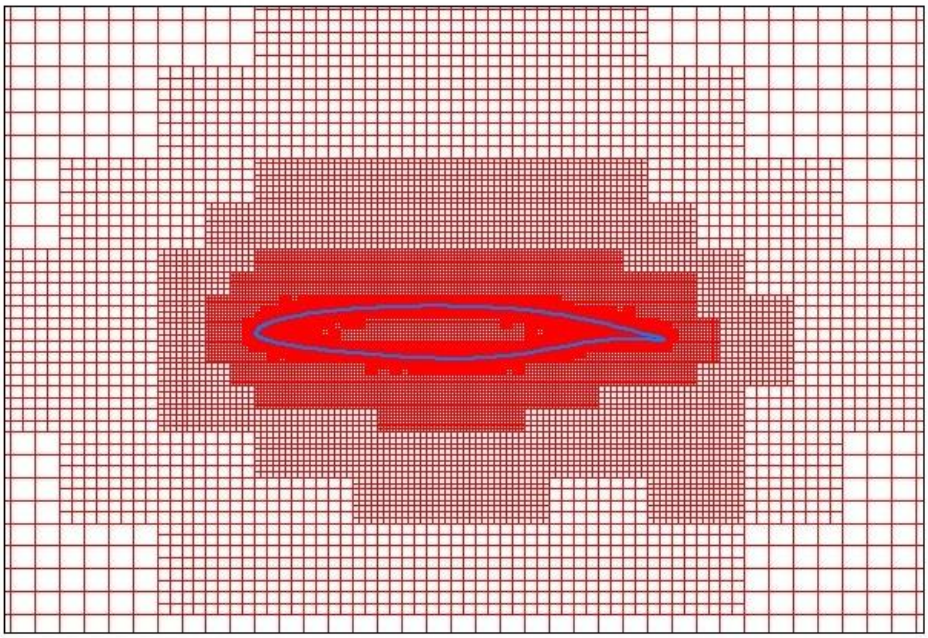
Η τρίτη αεροτομή που χρησιμοποιήθηκε για την εκτέλεση του επιλύτη ήταν η OR στην περίπτωση που είναι μαζεμένο το flap (σχήμα 5.18). Στην παράγραφο 5.4 θα μελετηθεί η περίπτωση που είναι ανοιχτό το flap. Μελετήθηκε για ροή μηδενικής γωνίας προσβολής και ταχύτητας V=70m/s. Η διαφορά με την περίπτωση της NACA 0012 είναι ότι δεν είναι συμμετρική αεροτομή και ότι μελετάται σε μικρότερη ταχύτητα ροής.

Το πλέγμα που δημιουργήθηκε για την επίλυση της ροής γύρω από αυτή την αεροτομή αποτελείται από 29176 κυψέλες (σχήμα 5.19), ενώ επιλέχθηκε να γίνουν 3000 ψευδοχρονικές επαναλήψεις για να επιτευχθεί ικανοποιητική σύγκλιση. Στο σχήμα 5.20 παρουσιάζεται ο ρυθμός σύγκλισης για το προαναφερθέν πλέγμα όταν ο επιλύτης εκτελείται στη CPU και στη GPU. Παρατηρείται ότι υπάρχει σύγκλιση στις 2300 επαναλήψεις. Στο σχήμα 5.21 παρουσιάζεται ο ρυθμός σύγκλισης και των 4 εξισώσεων που διέπουν την ροή. Στις 2 εξισώσεις διατήρησης της ορμής πετυχαίνεται ο ίδιος βαθμός σύγκλισης.

Σημειώνεται ότι ο πιστοποιημένος επιλύτης εκτελέστηκε στον τετραπύρηνο επεξεργαστή Intel Xeon E5620[33] χρονισμένο στα 2.4 GHz.



Σχήμα 5.18: Αεροτομή OR με μαζεμένο το flap.



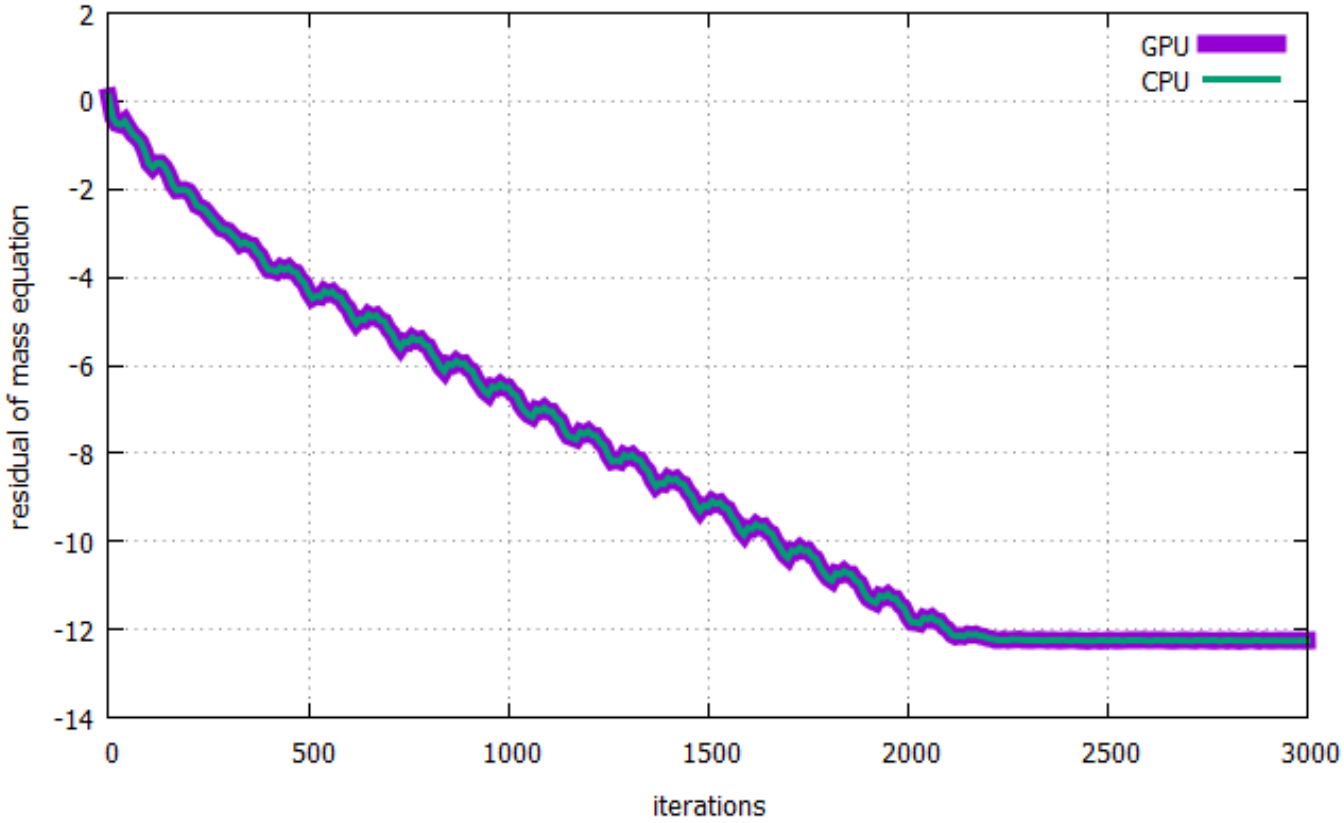
Σχήμα 5.19: Το πλέγμα 29176 κυψελών, όπως δημιουργήθηκε από τον κώδικα πλεγματοποίησης γύρω από την αεροτομή OR στην περίπτωση που είναι μαζεμένο το flap. Χαρακτηριστική είναι η πύκνωση του πλέγματος κοντά στην αεροτομή, προκειμένου να επιτευχθεί μεγαλύτερη ακρίβεια στον υπολογισμό των μεγεθών της ροής.

Από τον πίνακα 5.3 παρατηρούμε ότι για τη συγκεκριμένη αεροτομή για το πλέγμα του σχήματος 5.19, επετεύχθη επιτάχυνση στο συνολικό χρόνο εκτέλεσης του κώδικα πάνω από 22 φορές, καθώς επιλέχθηκε να γίνουν 3000 επαναλήψεις σε αντίθεση με την περίπτωση της αεροτομής NACA 0012 για ίδιο βαθμό πύκνωσης του πλέγματος, όπου έγιναν μόνο 800 επαναλήψεις, χωρίς να έχει υπάρξει απόλυτη σύγκλιση. Όσον αφορά τον χρόνο που απαιτείται μόνο για τους υπολογισμούς, δηλαδή αφαιρώντας τον χρόνο που απαιτείται για την δέσμευση μνήμης για την αποθήκευση των πινάκων, παρουσιάστηκε επιτάχυνση μεγαλύτερη από 42 φορές. Και σε αυτήν την περίπτωση γίνεται αισθητός ο χρόνος που απαιτείται για την δέσμευση μνήμης στη GPU, όπου αποτελεί το 49% του συνολικού χρόνου.

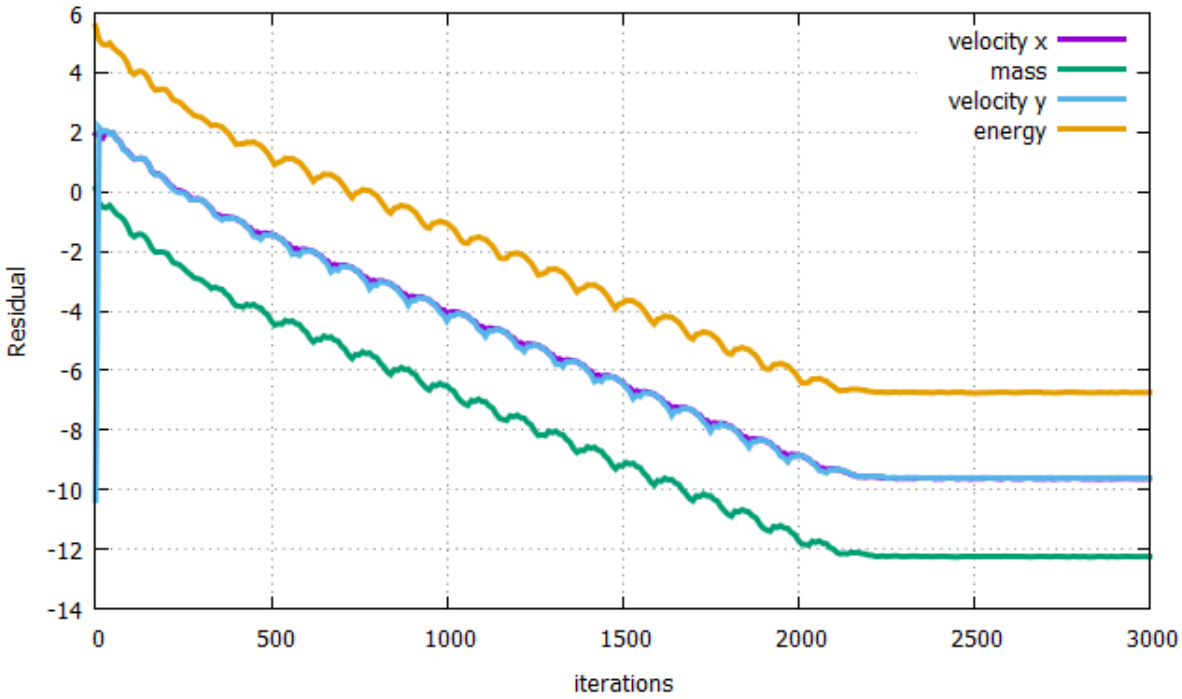
Στη συνέχεια παρουσιάζονται τα αποτελέσματα που προέκυψαν από τον επιλύτη που εκτελείται στη GPU.

Χρόνος Εκτέλεσης Επιλύτη GPU (sec)	Χρόνος Δέσμευσης Μνήμης Πινάκων GPU (sec)	Χρόνος Εκτέλεσης Επιλύτη CPU (sec)
149.58	72.57	3301.68

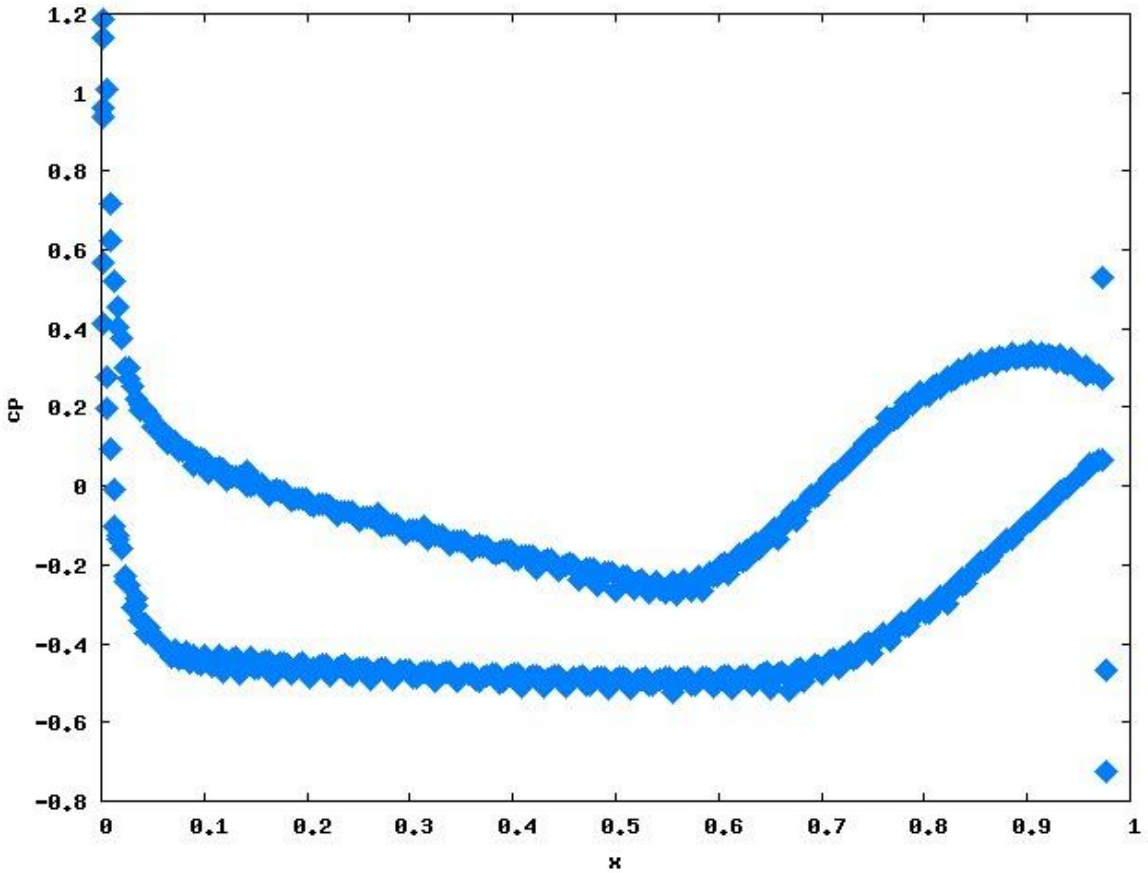
Πίνακας 5.3: Χρόνοι εκτέλεσης των δύο επιλυτών για πλέγμα 29176 κυψελών για την περίπτωση της αεροτομής OR με μαζεμένο το flap. Παρατηρείται επιτάχυνση της επίλυσης πάνω από 42 φορές.



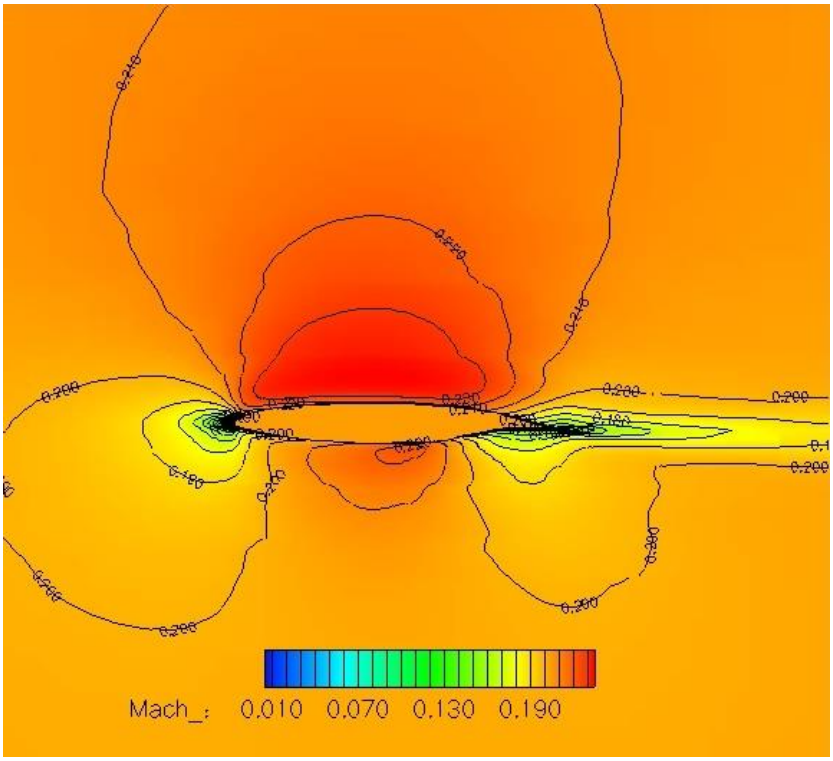
Σχήμα 5.20: Διάγραμμα ρυθμού σύγκλισης του επιλύτη για CPU και GPU για πλέγμα 29176 κυψελών και 3000 ψευδοχρονικών επαναλήψεων. Από την ταύτιση των δύο καμπυλών συμπεραίνεται η πιστή αναπαραγωγή αποτελεσμάτων από τον επιλύτη της παρούσας εργασίας.



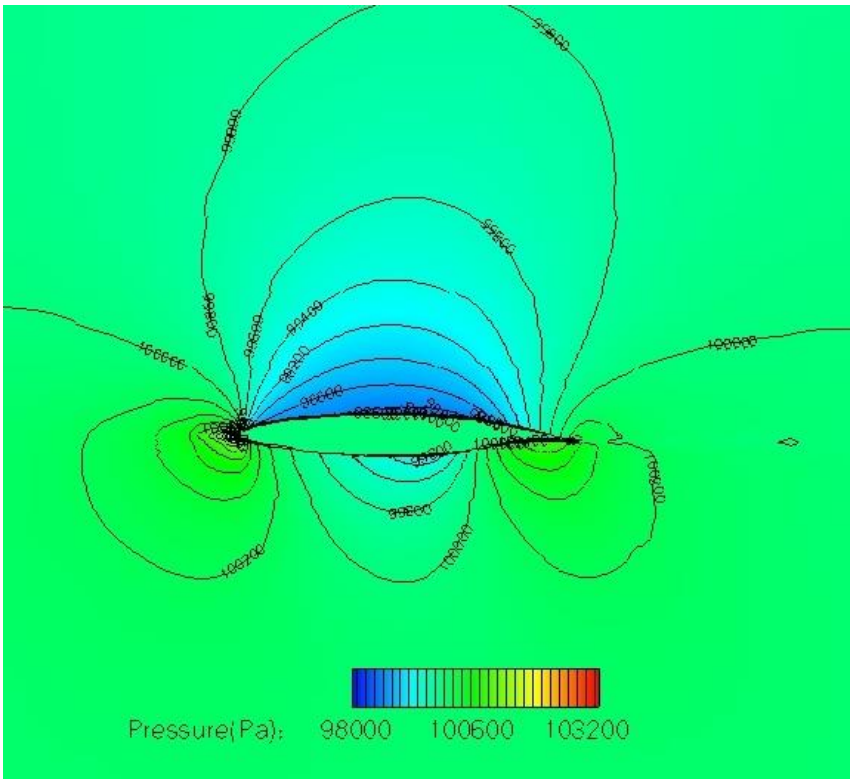
Σχήμα 5.21: Διάγραμμα ρυθμού σύγκλισης των τεσσάρων εξισώσεων για την περίπτωση πλέγματος 29176 κυψελών για την αεροτομή OR με μαζεμένο το flap.



Σχήμα 5.22: Διάγραμμα κατανομής του συντελεστή πίεσης γύρω από την αεροτομή OR με μαζεμένο το flap.



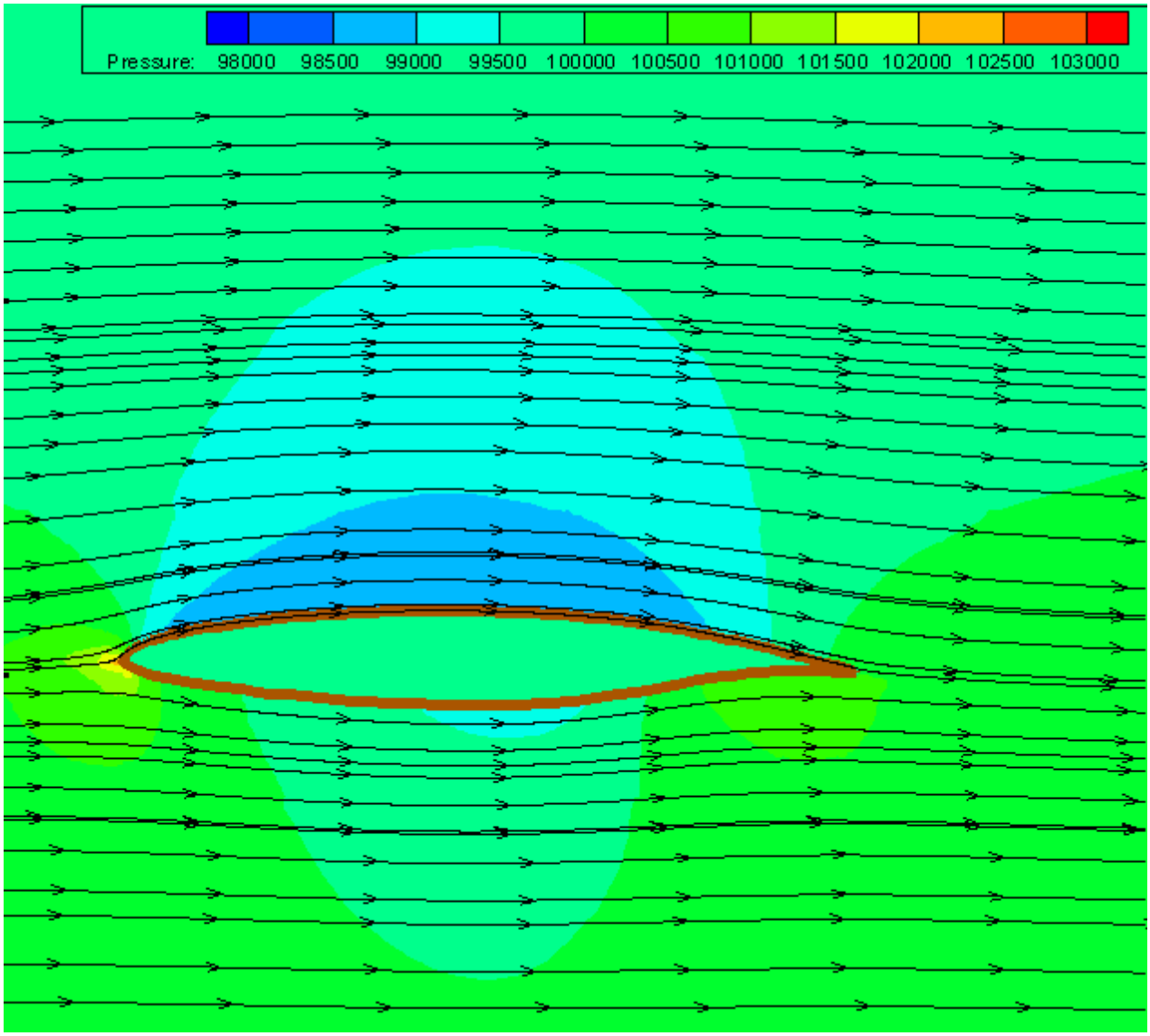
Σχήμα 5.23: Διάγραμμα αναπαράστασης του αριθμού Mach στο πεδίο γύρω από την αεροτομή OR με μαζεμένο το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 29176 κυψελών. Φαίνεται το σημείο ανακοπής στην περιοχή γύρω από την ακμή προσβολής, λόγω της μηδενικής γωνίας προσβολής.



Σχήμα 5.24: Διάγραμμα αναπαράστασης της πίεσης στο χωρίο γύρω από την αεροτομή OR με μαζεμένο το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 29176 κυψελών. Η πίεση παίρνει την μεγαλύτερη της τιμή κοντά στην ακμή προσβολής, καθώς εκεί βρίσκεται στο σημείο ανάκοπής.

Στο σχήμα 5.22 παρουσιάζεται το διάγραμμα κατανομής του συντελεστή πίεσης γύρω από την αεροτομή. Από το διάγραμμα αυτό παρατηρούμε ότι η ροή επιταχύνεται στο πρώτο μισό της (πτώση του συντελεστή πίεσης, επομένως και της ίδια της πίεση) και στην συνέχεια επιβραδύνεται καθώς αυξάνεται ο συντελεστής πίεσης. Παρόμοια συμπεράσματα βγάζουμε και από τα σχήματα 5.23 και 5.24 όπου παρουσιάζεται ο αριθμός Mach και η πίεση, αντίστοιχα, στο πεδίο γύρω από την αεροτομή. Επιπλέον, στο σχήμα 5.24 φαίνεται ότι η πίεση έχει μικρότερη τιμή στην πάνω πλευρά από ότι στην κάτω πλευρά, καθώς η αεροτομή δεν είναι συμμετρική.

Τέλος, στο σχήμα 5.25 παρουσιάζεται η κατεύθυνση της ροής γύρω από την αεροτομή. Έτσι ώστε να μπορεί να γίνει σύγκριση με το αντίστοιχο σχήμα (σχήμα 5.33) για την περίπτωση που είναι ανοιχτό το flap, ώστε να γίνει αισθητός ο τρόπος με τον οποίο επηρεάζεται η ροή με την παρουσία του flap.



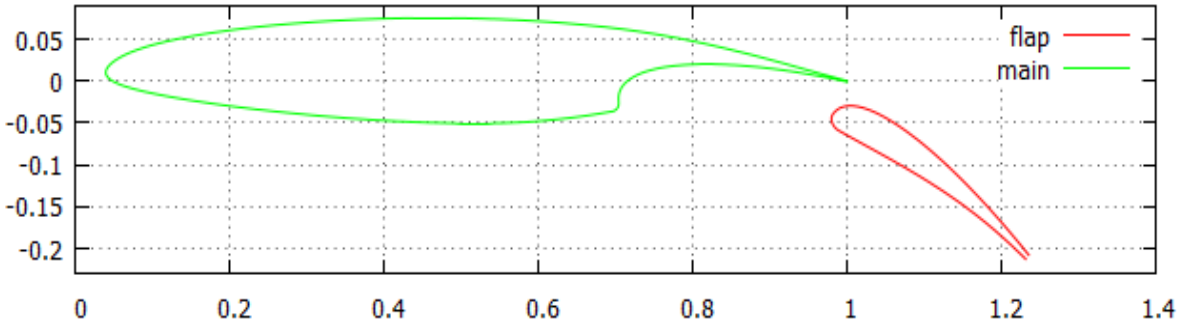
Σχήμα 5.25: Διάγραμμα αναπαράστασης της ροής γύρω από την αεροτομή OR με μαζεμένο το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 29176 κυψελών.

5.4 Αεροτομή OR με Ανοιχτό το Flap

Η τελευταία αεροτομή που χρησιμοποιήθηκε για την εκτέλεση του επιλύτη ήταν η OR στην περίπτωση που είναι ανοιχτό το flap (σχήμα 5.26). Μελετήθηκε για ροή μηδενικής γωνίας προσβολής και ταχύτητας V=70m/s. Η ιδιαιτερότητα αυτής της περίπτωσης σε σχέση με τις προηγούμενες είναι ότι το στερεό σώμα αποτελείται από 2 γεωμετρίες. Οπότε, με αυτή την περίπτωση προβλήματος δίνεται η ευκαιρία στον επιλύτη να δοκιμαστεί σε περίπτωση ύπαρξης περισσότερων του ενός σωμάτων μέσα στο πεδίο της ροής.

Το πλέγμα που δημιουργήθηκε για την επίλυση της ροής γύρω από αυτή την αεροτομή αποτελείται από 22630 κυψέλες (σχήμα 5.27 και 5.28), ενώ επιλέχθηκε να γίνουν 3000 ψευδοχρονικές επαναλήψεις για να επιτευχθεί ικανοποιητική σύγκλιση. Στο σχήμα 5.29 παρουσιάζεται ο ρυθμός σύγκλισης για το προαναφερθέν πλέγμα όταν ο επιλύτης εκτελείται στη CPU και στη GPU. Παρατηρείται ότι δεν υπάρχει σύγκλιση στις 3000 επαναλήψεις, οπότε θα μπορούσε να επιλεγεί λίγο μεγαλύτερος αριθμός επαναλήψεων. Στο σχήμα 5.30 παρουσιάζεται ο ρυθμός σύγκλισης και των 4 εξισώσεων που διέπουν την ροή. Παρατηρείται ότι και στις τέσσερις εξισώσεις δεν έχει επιτευχθεί σύγκλιση. Στις 2 εξισώσεις διατήρησης της ορμής πετυχαίνεται ο ίδιος βαθμός σύγκλισης.

Σημειώνεται ότι ο πιστοποιημένος επιλύτης εκτελέστηκε στον τετραπύρηνο επεξεργαστή Intel Xeon E5620[33] χρονισμένο στα 2.4 GHz.

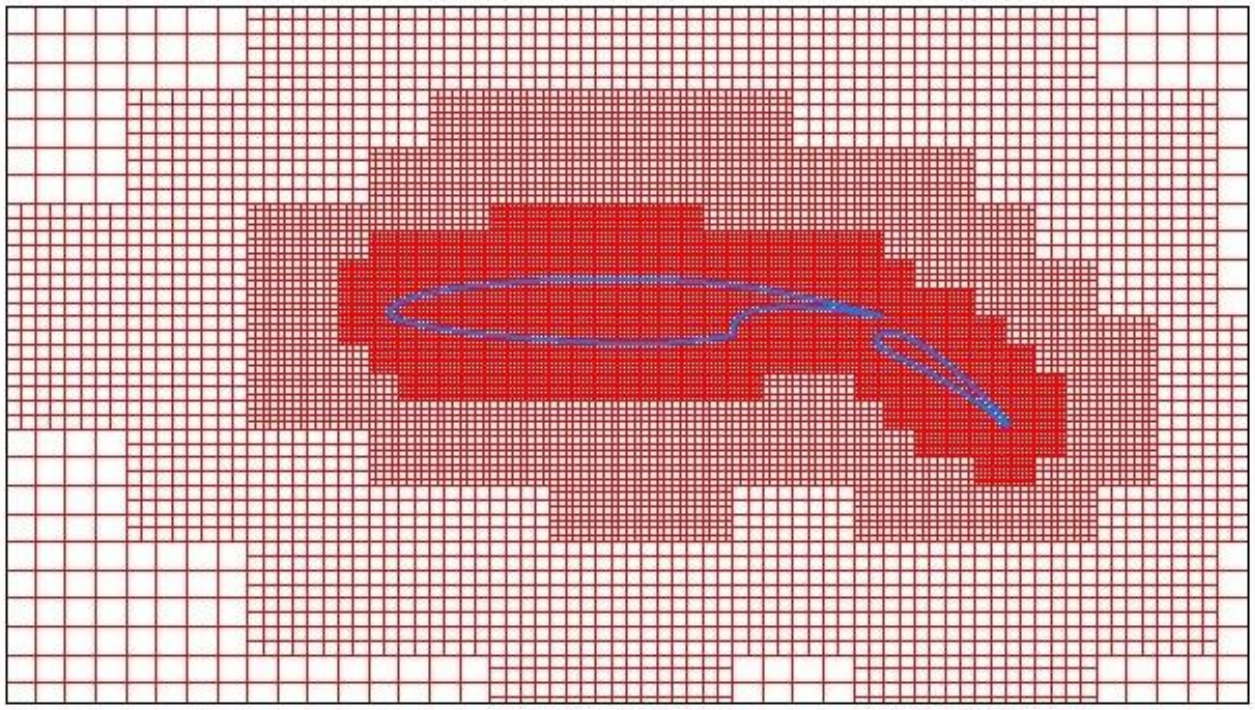


Σχήμα 5.26: Αεροτομή OR με ανοιχτό το flap.

Από τον πίνακα 5.4 παρατηρούμε ότι για τη συγκεκριμένη αεροτομή για το πλέγμα του σχήματος 5.27, επετεύχθη επιτάχυνση στο συνολικό χρόνο εκτέλεσης του κώδικα πάνω από 25 φορές. Όσον αφορά τον χρόνο που απαιτείται μόνο για τους υπολογισμούς, δηλαδή αφαιρώντας τον χρόνο που απαιτείται για τη δέσμευση μνήμης για τους πίνακες, παρουσιάστηκε επιτάχυνση μεγαλύτερη από 41 φορές. Σε αυτήν την περίπτωση η δέσμευση μνήμης κατέλαβε μόνο το 39% του συνολικού χρόνου εκτέλεσης του επιλύτη στη GPU.

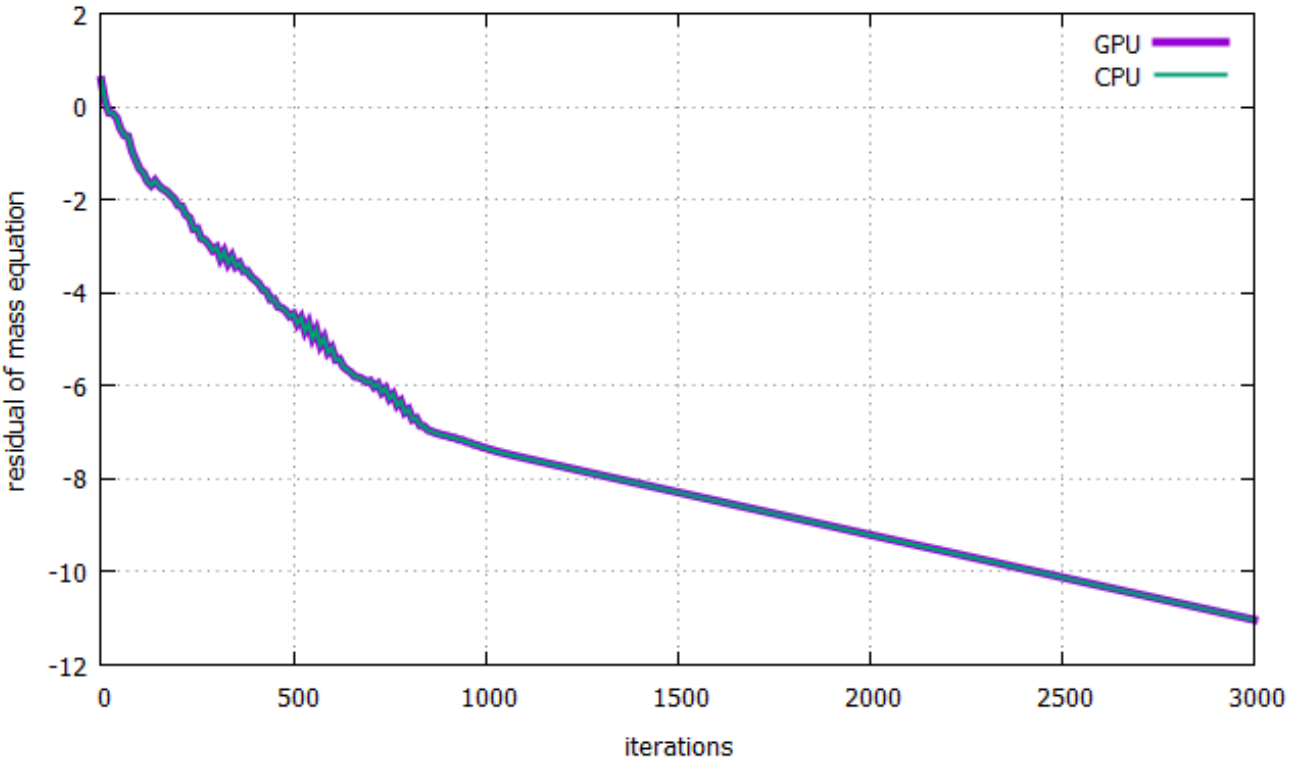
Χρόνος Εκτέλεσης Επιλύτη GPU (sec)	Χρόνος Δέσμευσης Μνήμης Πινάκων GPU (sec)	Χρόνος Εκτέλεσης Επιλύτη CPU (sec)
111.4	43.25	2808.71

Πίνακας 5.4: Χρόνοι εκτέλεσης των δύο επιλυτών για πλέγμα 22630 κυψελών για την περίπτωση της αεροτομής OR με ανοιχτό το flap. Παρατηρείται επιτάχυνση της επίλυσης πάνω από 41 φορές.

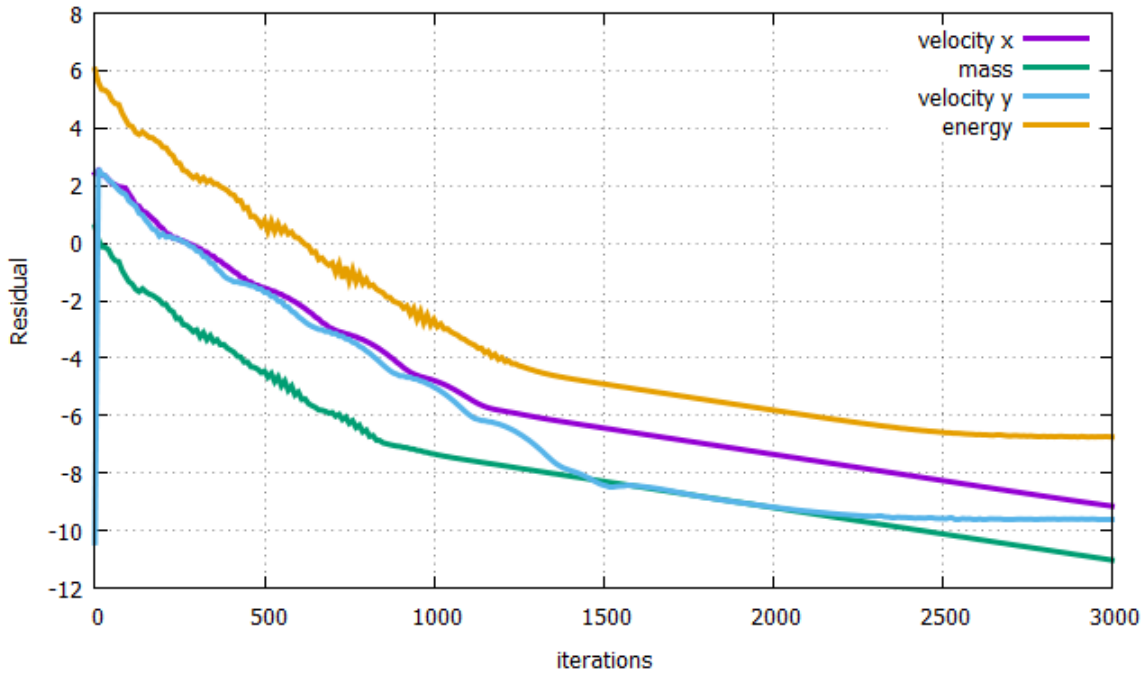


Σχήμα 5.27: Το πλέγμα 22630 κυψελών, όπως δημιουργήθηκε από τον κώδικα πλεγματοποίησης γύρω από την αεροτομή OR με ανοιχτό το flap. Χαρακτηριστική είναι η πύκνωση του πλέγματος κοντά στην αεροτομή, προκειμένου να επιτευχθεί μεγαλύτερη ακρίβεια στον υπολογισμό των μεγεθών της ροής.

Σχήμα 5.28: Το καρτεσιανό πλέγμα 22630 κυψελών γύρω από την αεροτομή OR με ανοιχτό το flap εστιασμένο στην περιοχή γύρω των στερεών σωμάτων.

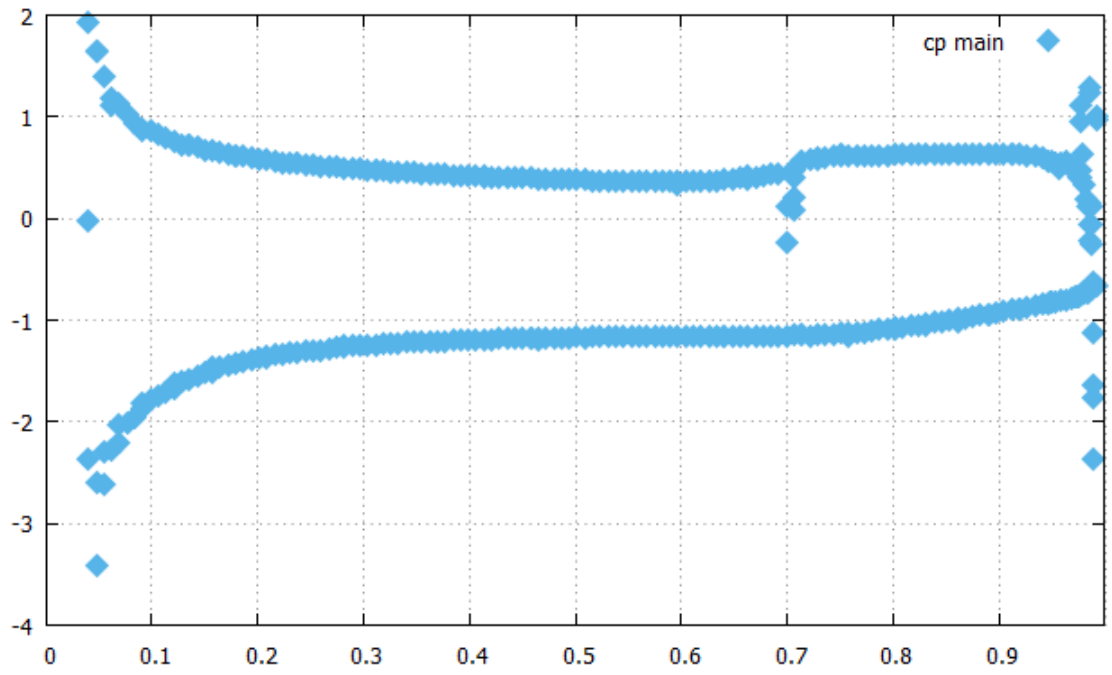


Σχήμα 5.29: Διάγραμμα ρυθμού σύγκλισης του επιλύτη για CPU και GPU για πλέγμα 22630 κυψελών και 3000 ψευδοχρονικών επαναλήψεων. Από την ταύτιση των δύο καμπυλών συμπεραίνεται η πιστή αναπαραγωγή αποτελεσμάτων από τον επιλύτη της παρούσας εργασίας.

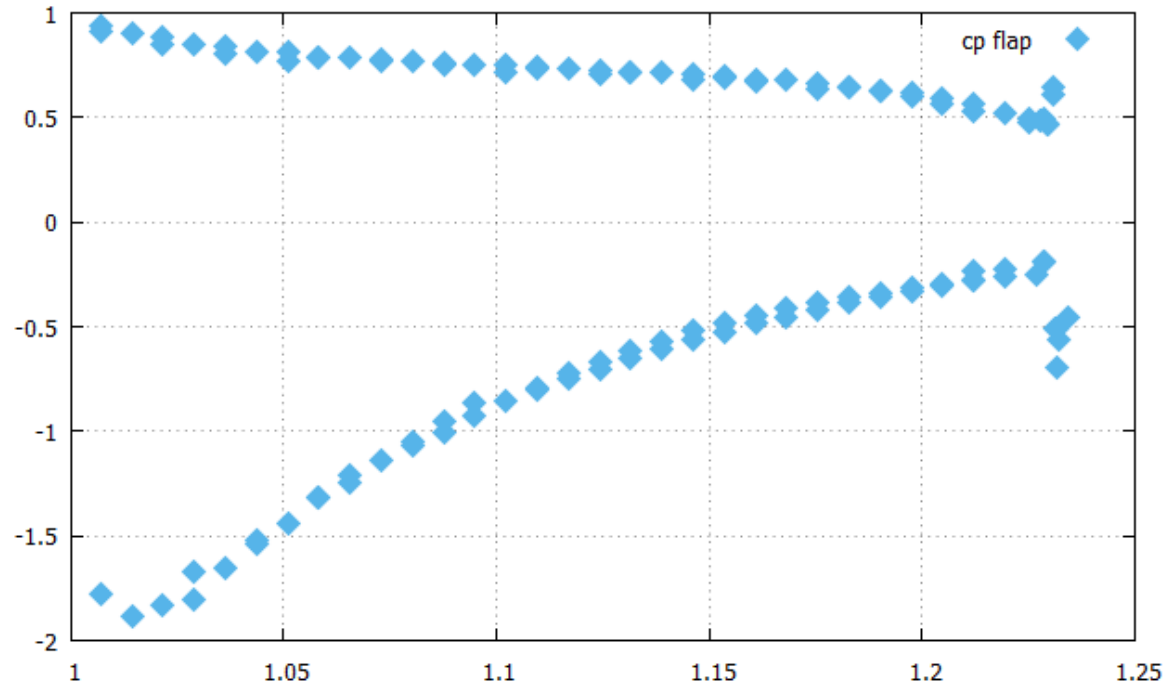


Σχήμα 5.30: Διάγραμμα ρυθμού σύγκλισης των τεσσάρων εξισώσεων για την περίπτωση πλέγματος 22630 κυψελών για την αεροτομή OR με ανοιχτό το flap 3000 ψευδοχρονικών επαναλήψεων. Παρατηρείται ότι μετά από 3000 επαναλήψεις δεν έχει επιτευχθεί πλήρης σύγκλιση.

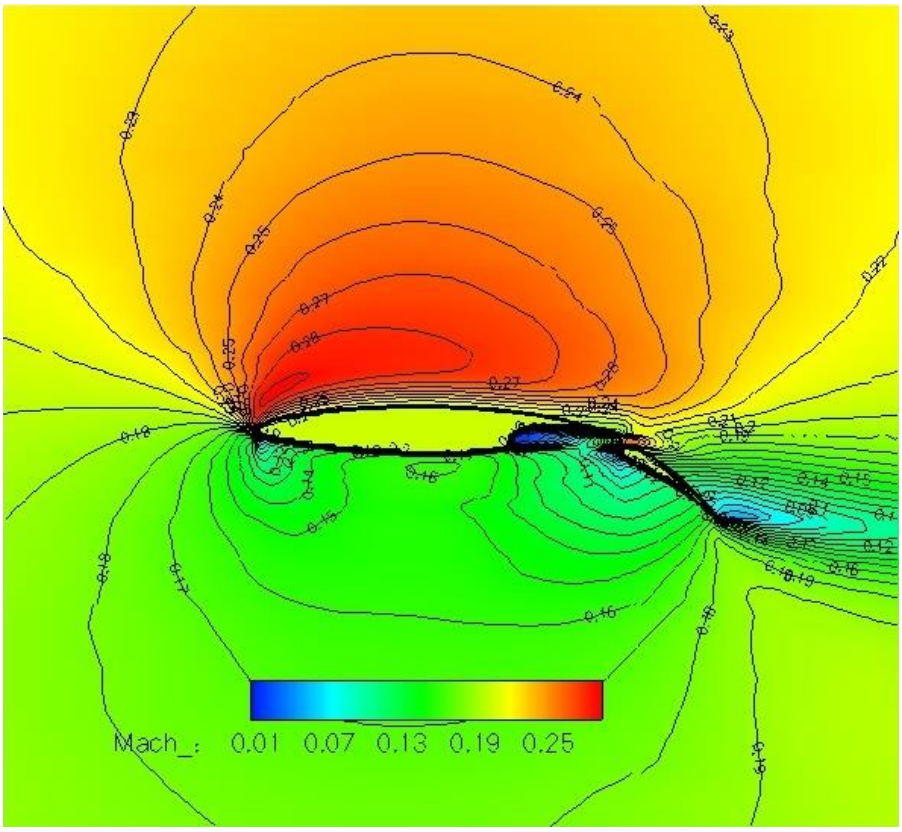
Στη συνέχεια παρουσιάζονται τα αποτελέσματα που προέκυψαν από τον επιλύτη που εκτελείται στη GPU.



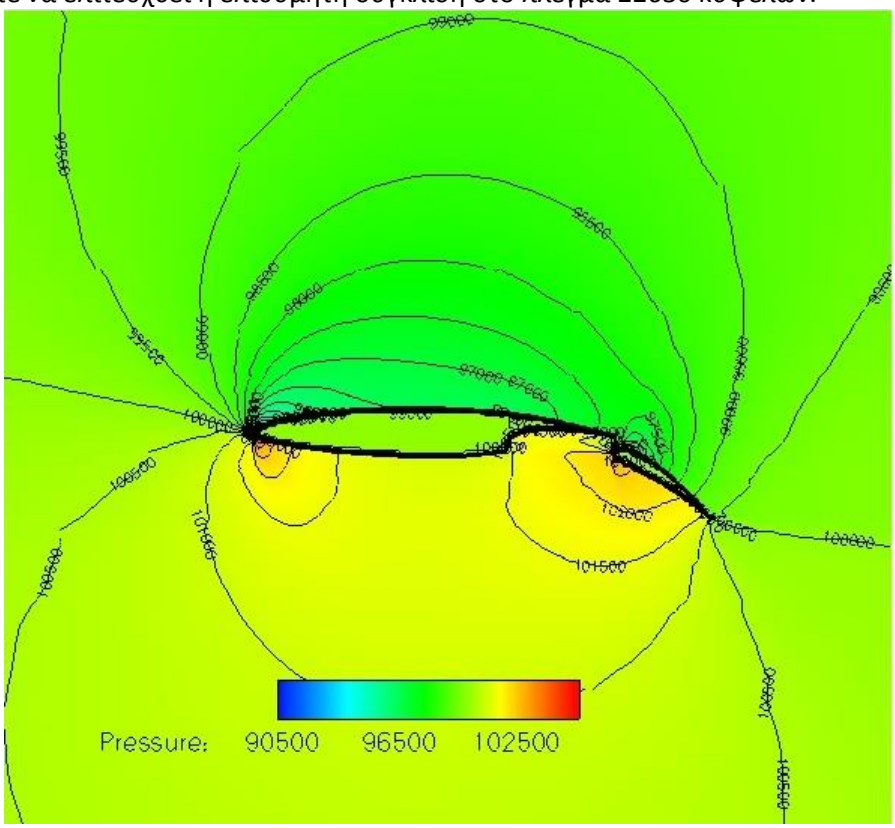
Σχήμα 5.31: Διάγραμμα κατανομής του συντελεστή πίεσης γύρω από το κυρίως τμήμα της αεροτομή OR. Παρατηρείται μια σημαντική διαφορά στην τιμή του συντελεστή πίεσης μεταξύ της πάνω και της κάτω πλευράς.



Σχήμα 5.32: Διάγραμμα κατανομής του συντελεστή πίεσης γύρω από το flap. Παρατηρείται μια σημαντική διαφορά στην τιμή του συντελεστή πίεσης μεταξύ της πάνω και της κάτω πλευράς του.



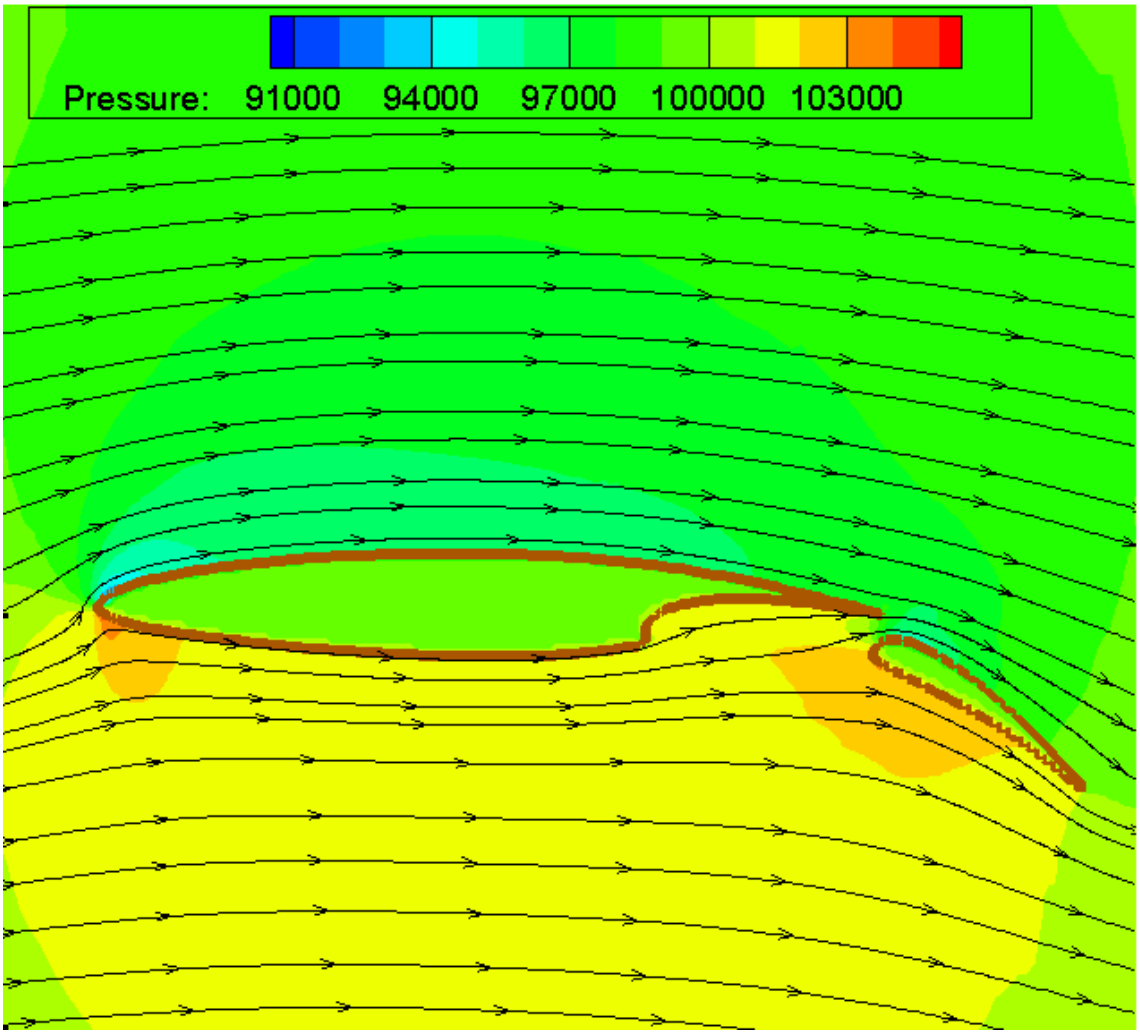
Σχήμα 5.33: Διάγραμμα αναπαράστασης του αριθμού Mach στο πεδίο γύρω από την αεροτομή OR με ανοιχτό το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 22630 κυψελών.



Σχήμα 5.34: Διάγραμμα αναπαράστασης της πίεσης στο χωρίο γύρω από την αεροτομή OR με ανοιχτό το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 22630 κυψελών.

Με τα παραπάνω σχήματα (5.31 - 5.34) που παρουσιάζονται, μπορούν να βγουν χρήσιμα συμπεράσματα για την χρησιμότητα των flaps. Μία από τις χρήσεις των flaps στην αεροναυπηγική είναι η αύξηση του συντελεστή άνωσης κατά την διάρκεια της απογείωσης του αεροσκάφους, καθώς αυτό κινείται με μικρή ταχύτητα. Στα σχήματα αυτά και ειδικά σε εκείνα που δείχνουν την πίεση, παρατηρείται διαφορά στην τιμή της στην περιοχή πάνω και κάτω από την αεροτομή. Αυτή η διαφορά πίεσης είναι εκείνη που προκαλεί την άνωση στην αεροτομή. Επίσης, φαίνεται ότι η ροή επιταχύνεται σε όλη την πάνω πλευρά του κυρίου τμήματος της αεροτομής.

Στο σχήμα 5.35 βλέπουμε πως επηρεάζεται η ροή με την παρουσία ενός δεύτερου σώματος σε σχέση με την ροή του ρευστού στο σχήμα 5.25, όπου δεν είναι ανοιχτό το flap.



Σχήμα 5.35: Διάγραμμα αναπαράστασης της ροής γύρω από την αεροτομή OR με ανοιχτό το flap. Το αποτέλεσμα αυτό προέκυψε μετά από 3000 ψευδοχρονικές επαναλήψεις ώστε να επιτευχθεί η επιθυμητή σύγκλιση στο πλέγμα 22630 κυψελών. Παρατηρείται η επιρροή του flap στη ροή γύρω από το κυρίως τμήμα της αεροτομής σε σχέση με την περίπτωση που ήταν κλειστό το flap.

Κεφάλαιο 6

Ανακεφαλαίωση και Συμπεράσματα

Η Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης, τα τελευταία χρόνια, ασχολείται συστηματικά με την αριθμητική επίλυση προβλημάτων Υπολογιστικής Ρευστοδυναμικής δημιουργώντας λογισμικό που εκμεταλλεύεται την παραλληλία που προσφέρουν οι κάρτες γραφικών. Στο πλαίσιο αυτό, έχουν δημιουργηθεί κώδικες που επιλύουν τις εξισώσεις ροής σε δομημένα και μη δομημένα πλέγματα [9]. Στόχος της παρούσας διπλωματικής εργασίας ήταν η δημιουργία κώδικα που επιλύει τις διδιάστατες εξισώσεις μη-συνεκτικής μόνιμης ροής (Euler) με τη μέθοδο των τεμνόμενων κυψελών (cut-cells) εκμεταλλευόμενος την παραλληλία των καρτών γραφικών.

Κατά την εκπόνηση της εργασίας, διατυπώθηκαν και διακριτοποιήθηκαν οι εξισώσεις ροής για καρτεσιανό πλέγμα και διατυπώθηκαν οι οριακές συνθήκες για την εφαρμογή της μεθόδου τεμνόμενων κυψελών. Η μέθοδος που επιλέχτηκε για την επίλυση των εξισώσεων ροής είναι αυτής της χρονοπροέλασης, ενώ το τελικό σύστημα εξισώσεων επιλύθηκε με τη μέθοδο Jacobi.

Στη συνέχεια, με βάση τα παραπάνω δεδομένα δημιουργήθηκε ο αντίστοιχος κώδικας που επιλύει τη ροή με γνώμονα τον αντίστοιχο κώδικα της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης που εκτελείται σειριακά σε CPU, χρησιμοποιώντας το ίδιο λογισμικό για την γένεση του πλέγματος. Ο κώδικας προγραμματίστηκε στη γλώσσα προγραμματισμού CUDA C και εκτελέστηκε σε κάρτες γραφικών GeForce GTX 670 της εταιρίας NVIDIA. Κατά τον προγραμματισμό, έγινε προσπάθεια εκμετάλλευσης των δυνατοτήτων που προσφέρουν οι κάρτες γραφικών με στόχο τη μείωση του χρόνου εκτέλεσης του λογισμικού.

Από τα αποτελέσματα που προέκυψαν από την εκτέλεση του κώδικα, διαπιστώθηκε η πιστή αναπαραγωγή τους με εκείνα που προκύπτουν από την εκτέλεση του αντίστοιχου υπάρχοντα κώδικα για τη CPU. Στη συνέχεια, έγινε σύγκριση των χρόνων εκτέλεσης των 2 κωδίκων για τα ίδια πλέγματα και τις ίδιες συνθήκες εξωτερικής ροής. Αποδείχθηκε, όπως ήταν αναμενόμενο, ότι ο κώδικας εκτελείται αρκετά πιο γρήγορα σε κάρτες γραφικών από ότι σε CPU. Μάλιστα, σε κάποιες περιπτώσεις επιτεύχθηκε επιτάχυνση στο χρόνο εκτέλεσης υπολογισμών μεγαλύτερη του x40. Επίσης, διαπιστώθηκε ότι καταναλώνεται σημαντικός χρόνος για τη δέσμευση μνήμης στη κάρτα γραφικών προκειμένου να αποθηκευτούν οι απαραίτητοι για τους υπολογισμούς πίνακες.

Ολοκληρώνοντας, ακολουθούν κάποιες προτάσεις για μελλοντική έρευνα. Θα μπορούσε να τροποποιηθεί ο κώδικας ώστε να χρειάζεται λιγότερες θέσεις μνήμης, καθώς σπαταλάται σημαντικός χρόνος για τη δέσμευσή τους επιτρέποντάς

του να επιλύει ακόμα μεγαλύτερα πλέγματα. Στη παρούσα εργασία, δόθηκε έμφαση στην πιστή αναπαραγωγή των αποτελεσμάτων και την επιτάχυνση των υπολογισμών και όχι τόσο στη σωστή δέσμευση μνήμης. Επίσης, θα μπορούσε να προστεθεί η δυνατότητα στον κώδικα να χρησιμοποιεί δεύτερης τάξης σχήμα για τη διακριτοποίηση των χωρικών όρων της εξίσωσης ροής και όχι μόνο πρώτης, δηλαδή να υπολογίζει τις χωρικές παραγώγους των ροϊκών μεγεθών. Τέλος, παράλληλα με την παρούσα εργασία, εξελισσόταν η διπλωματική εργασία του Σ. Κατσανούλη [35] με αντικείμενο τον προγραμματισμό λογισμικού για την επίλυση μη-μόνιμων ροών με τη μέθοδο τεμνόμενων κυψελών σε CPUs. Οπότε, έχοντας ως γνώμονα το λογισμικό αυτό, θα μπορούσε να επεκταθεί η χρήση του παρόντος κώδικα σε μη-μόνιμες ροές. Έτσι, θα μπορεί να γίνει χρήση ενός από τα σημαντικότερα πλεονεκτήματα της μεθόδου τεμνόμενων κυψελών, που αφορά την αποφυγή επαναπλεγματοποίησης του μεταβαλλόμενου χωρίου σε κάθε νέα χρονική στιγμή.

Βιβλιογραφία

1. NVIDIA. *NVIDIA CUDA C Programming Guide*, Version 7.0, March 2015.

2. Teng-Yi Huang, Yu-Wei Tang and Shiun-Ying Ju. *Accelerating image registration of MRI by GPU-based parallel computation*, Magnetic Resonance Imaging 29 (2011) 712-716.

3. Carolyn L. Phillips, Joshua A. Anderson and Sharon C. Glotzer. *Pseudo-random number generation for Brownian Dynamics and Dissipative Particle Dynamics simulations on GPU devices*, Journal of Computational Physics 230 (2011) 7191-7201.

4. Alfeus Sunarso, Tomohiro Tsuji and Shigeomi Chono. *GPU-accelerated molecular dynamics simulation for study of liquid crystalline flows*, Journal of Computational Physics 229 (2010) 5486-5497.

5. Yan Zhang, Panagiotis Vouzis and Nikolaos V. Sahinidis. *GPU simulations for risk assessment in CO_2 geologic sequestration*, Computers and Chemical Engineering 35 (2011) 1631-1644.

6. Adino Heimlich, Antonio C.A. Mol and Claudio M.N.A. Pereira. *GPU-based Monte Carlo simulation in neutron transport and finite di_erences heat equation evaluation*, Progress in Nuclear Energy 53 (2011) 229-239.

7. Erich Elsen, Patrick LeGresley and Eric Darve. *Large calculation of the flow over a hypersonic vehicle using a GPU*, Journal of Computational Physics 227(2008)10148-10161.

8. Wangda Zuo and Qingyan Chen. *Fast and informative flow simulations in a building by using fast fluid dynamics model on graphics processing unit*, Building and Environment 45(2010)747-757.

9. Varvara G. Asouti, Xenofon S. Trompoukis, Ioannis C. Kampolis and Kyriakos C. Giannakoglou. *Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units*, International Journal for Numerical Methods in Fluids, Vol. 67(2), pp. 232-246, 2011.

10. Alfed j. Kalyanapu, Siddharth Shankar, Eric R. Pardyjak, David R. Judi and Steven J. Burian. *Assessment of GPU computational enhancement to a 2D flood model*, Enviromental Modelling & Software 26(2011)1009-1016.

11. Κυριάκος Χ. Γιαννάκογλου. *Μέθοδοι Βελτιστοποίησης στην Αεροδυναμική*, Πανεπιστημιακές Εκδόσεις ΕΜΠ, 4η Έκδοση, 2006

12. Ξενοφών Σ. Τρομπούκης. *Αριθμητική επίλυση προβλημάτων αεροδυναμικής - αεροελαστικής σε επεξεργαστές καρτών γραφικών*, Διδακτορική Διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, 2012

13. Γεώργιος Βαλσαμάκης. *Αριθμητική επίλυση μη-μόνιμου πεδίου ροής σε κάρτες γραφικών με απεικόνιση του σε πραγματικό χρόνο*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Φεβρουάριος 2010

14. Γεώργιος Δ. Ρήγας. *Προσομοίωση και χαμηλού κόστους βελτιστοποίηση του ενεργητικού ελέγχου της ροής ρευστού γύρω από αεροτομή σε κάρτες γραφικών*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Ιούλιος 2010

15. Ελευθέριος Ι. Φούντης. *Προγραμματισμός σε Κάρτες Γραφικών και Εφαρμογή στην Αεροδυναμική Βελτιστοποίηση*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Οκτώβριος 2009

16. Ιωάννης Σ. Καββαδίας. *Προγραμματισμός επιλύτη 3Δ εξισώσεων ροής ατριβούς ρευστού σε δομημένα πλέγματα σε κάρτες γραφικών*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Οκτώβριος 2011

17. Γεώργιος Σ. Ελευθερίου. *Προγραμματισμός Παράλληλου Επιλύτη Εξισώσεων Euler για 3Δ Ροές σε Δομημένα Πλέγματα σε Συστοιχίες Καρτών Γραφικών*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Οκτώβριος 2012

18. Γεώργιος Δ. Ντανάκας. *Επίλυση Δισδιάστατης Ροής με Κεντροκυψελική Διατύπωση σε Μη-Δομημένα Πλέγματα. Προγραμματισμός σε Επεξεργαστές Καρτών Γραφικών*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Φεβρουάριος 2012

19. https://en.wikipedia.org/wiki/Multi-core_processor

20. Jason Sanders and Edward Kandrot. CUDA by example: An introduction to General-Purpose GPU Programming, Addison-Wesley, 2010.

21. http://gpu.cs.uct.ac.za/Slides/Kepler.pdf

22. https://en.wikipedia.org/wiki/Kepler_(microarchitecture)

23. http://gr.pcmag.com/nvidia-geforce-gtx-670/644/review/nvidia-geforce-gtx-670

24. William J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, University of Michigan, 1994.

25. Henry Bandringa. *Immersed boundary methods*, Master Thesis in Applied Mathematics, University of Groningen, August 2010.

26. Philip L. Roe. *Approximate Riemann Solvers, Parameter Vectors and Difference Schemes*, Journal of Computational Physics 43 (1981) 357-372.

27. https://en.wikipedia.org/wiki/Network_File_System

28. https://en.wikipedia.org/wiki/Network_Information_Service

29. http://ark.intel.com/products/65702/Intel-Core-i5-3570-Processor-6M-Cache-up-to-3_80-GHz

30. Richard Courant, Kurt Friedrichs and Hans Lewy. *Über die partiellen differenzengleichungen der mathematischen physik,* Mathematische Annalen 100 (1928) 32-74.

31. Ali E. Yilmaz, Emel Mahmutyazicioglu and Ġsmail H. Tuncer. *Turbulent flow solutions on 2D hybrid grids using an implicit multigrid algorithm*, 6. Ankara International Aerospace Conference AIAC-2011-143, September 2011

32. Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley, 2011

33. http://ark.intel.com/products/47925/Intel-Xeon-Processor-E5620-12M-Cache-2_40-GHz-5_86-GTs-Intel-QPI

34. Vincent Brunet, Sébastien Deck, Pascal Molton and Mylène Thiery. *A complete experimental and numerical study of the buffet phenomenon over the OAT15A airfoil*, 40ème Colloque Aérodynamique Appliquée, 2005

35. Στέργιος Κατσανούλης. *Προγραμματισμός Λογισμικού για την Επίλυση Μη-Μόνιμων Πεδίων Ροής με Κινούμενα Στερεά Όρια με τη Μέθοδο των Τεμνόμενων Κυψελών (Cut-Cell)*, Διπλωματική εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών ΕΜΠ, Οκτώβριος 2015