ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Διερεύνηση της απόδοσης κωδίκων υπολογιστικής ρευστοδυναμικής σε υπολογιστικά συστήματα με διαφορετικές κάρτες γραφικών

Φοιτήτρια: Λάμπα Ελένη
ΑΜ: 272017166

Επιβλέπων Καθηγητής: Ιωάννης Σαρρής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΣΕΠΤΕΜΒΙΟΣ 2022

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF MECHANICAL ENGINEERING

Diploma Thesis

Investigation of the performance of computational fluid dynamics codes in computer systems with different graphics cards

Student: Eleni Lampa
Registration Number: 272017166

Supervisor Professor: Ioannis Sarris

ATHENS-EGALEO, SEMPEMBER 2022

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Ιωάννης Σαρρής, Καθηγητής	Κων/νος Νίκας, Καθηγητής	Κων/νος-Ιωάν. Βασιλόπουλος, (βαθμίδα)
Ioannis Sarris Digitally signed by Ioannis Sarris Date: 2022.10.31 09:58:09 +02'00' (Υπογραφή)	Digitally signed by Konstantinos-Stefanos Nikas Date: 2022.10.31 11:45:28 +02'00' (Υπογραφή)	KONSTANTINOS-IOANNIS VASILOPOULOS Digitally signed by KONSTANTINOS-IOANNIS VASILOPOULOS Date: 2022.10.31 10:43:11 +02'00' (Υπογραφή)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Ελένη Λάμπα, Σεπτέμβριος, 2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Ελένη Λάμπα του Νικολάου, με αριθμό μητρώου 272017166 φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Η Δηλούσα,
Ελένη Λάμπα

Ευχαριστίες

Βιώνοντας αυτή τη γνωστική περιπέτεια όλα αυτά τα χρόνια στο τμήμα Μηχανολόγων Μηχανικών του ΠΑΔΑ, νιώθω την επιθυμία να ευχαριστήσω όλους τους καθηγητές μου, που με το παραπάνω, μας μοιραστήκαν τις γνώσεις τους και τα βιώματα τους. Από αυτή την εμπειρία μπορώ να δω τον εαυτό μου να προχωράει μπροστά τόσο επαγγελματικά όσο και πνευματικά.

Θέλω να δώσω θερμά ευχαριστήρια στον καθηγητή μου, κ. Ιωάννη Σαρρή, διότι μου έδωσε την ευκαιρία να θαυμάσω περισσότερο την επιστήμη της υπολογιστικής ρευστομηχανικής. Ως επιβλέπων καθηγητής της διπλωματικής μου διατριβής, ήταν πάντα πρόθυμος να με συμβουλέψει διευκολύνοντας έτσι το έργο μου. Η συνεργασία μου μαζί του ήταν μοναδική.

Βέβαια, δεν θα μπορούσα να παραλείψω και τον Σταύρο Δουλκερίδη, υποψήφιο διδάκτορα, που με βοήθησε να κατανοήσω το υπολογιστικό μέρος της διπλωματικής μου.

Ιδιαίτερα ευχαριστώ τους συμφοιτητές μου, με τις ιδιαίτερες προσωπικότητες τους. Από αυτούς είδα έναν κόσμο, ανοικτό σε όλα και γεμάτο αστεία.

Κλείνοντας, ευχαριστώ θερμότατα τους ανθρώπους που αγκαλιάζουν τις χαρές και τις λύπες μου. Η οικογένεια μου είναι ο λόγος που θέλω να συνεχίσω το επιστημονικό μου ενδιαφέρον.

Περίληψη

Σκοπός της διπλωματικής εργασίας αποτελεί η διεύρυνση της τεχνογνωσίας πάνω στο τομέα της υπολογιστικής ρευστοδυναμικής. Η ανάπτυξη της υπολογιστικής ισχύς προσδίδει ιδιαίτερο πλεονέκτημα στην σημερινή εποχή. Εφαρμογές με άπειρα δεδομένα και δύσκολες μεθόδους επίλυσης στοχεύουν επαναστατικές εξελίξεις.

Αντικείμενο της παρούσας διατριβής αποτελεί η εκτέλεση υπολογιστικών προσομοιώσεων ρευστοδυναμικής σε σύγχρονα αναπτυγμένα υπερυπολογιστικά συστήματα υψηλών αποδόσεων. Κατ' αυτόν τον τρόπο, η ερευνητική κοινότητα κατορθώνει να έχει γρήγορα και ακριβή αποτελέσματα που βελτιώνουν τις ατέλειες μιας κατασκευής. Στην παρούσα διπλωματική, προσομοιώθηκε στη γλώσσα προγραμματισμού Julia, η ροή αέρα γύρω από ένα κυκλικό σώμα, όπου εξάχθηκαν αποτελέσματα για διάφορα φυσικά μεγέθη, καθώς επίσης συγκρίθηκαν οι χρόνοι εκτέλεσης των προσομοιώσεων σε πολλαπλά υπολογιστικά συστήματα.

Στην αρχή της διατριβής, διατυπώνονται τα κεφάλαια της ρευστομηχανικής. Αναγκαίο καλό είναι το θεωρητικό υπόβαθρο που βοηθά τον αναγνώστη να κατανοήσει καλύτερα το πρόβλημα. Περιγράφεται ο ρόλος της ρευστομηχανικής και συγχρόνως οι πολλαπλές δυνατότητες που μπορεί η υπολογιστική ρευστοδυναμική να προσφέρει στις καθημερινές εφαρμογές. Στα επόμενα κεφάλαια διατυπώνονται τα βήματα μιας CFD λύσης που χρειάζεται για υλοποίηση ολοκληρωμένων αποτελεσμάτων. Αμέσως μετά, αναλύεται ο τρόπος εκτέλεσης του κώδικα του προβλήματος από τον παράλληλο προγραμματισμό. Συγχρόνως, παρατίθεται η αρχιτεκτονική του υπολογιστή, ο τρόπος διάταξης της μνήμης και τα βασικά εξαρτήματα που υλοποιείται η μελέτη περίπτωσης, δηλαδή οι κάρτες γραφικών GPU. Συνεχίζοντας, αναπτύσσονται τα μοντέλα του παράλληλου προγραμματισμού, ώστε να επιλεγεί το πιο αξιόλογο μοντέλο για την επίλυση του κώδικα πάνω σε μια κάρτα γραφικών σύμφωνα με τη γλώσσα προγραμματισμού Julia που θα χρησιμοποιηθεί. Ταυτόχρονα, περιγράφονται οι βασικές εξισώσεις και τα μεγέθη που θα βοηθήσουν στην σύνταξη και τη λογική του υπό μελέτη κώδικα. Τέλος, αξιολογούνται τα τελικά αποτελέσματα των προσομοιώσεων καθώς και ο χρόνος εκτέλεσης των προσομοιώσεων για κάθε διαφορετική περίπτωση που μελετάται.

Λέξεις – κλειδιά

CFD και υπολογιστής, Εξισώσεις Navier – Stokes, Μέθοδος Διακριτοποίησης, Δομημένο πλέγμα, Παράλληλος Προγραμματισμός, Μοντέλο CUDA, Γλώσσα Προγραμματισμού Julia, JuliaGPU.

Abstract

The aim of the thesis is to expand the know-how in the field of Computational Fluid Dynamics. The development of computing power confers a particular advantage in today's world. Applications with infinite data and difficult solution methods target revolutionary developments.

The subject of this thesis is the execution of Computational Fluid Dynamics Simulations in contemporary, high-performance, supercomputing systems. In this way, the research community manages to get fast and accurate results that improve the imperfections of a particular construction. In this thesis, the airflow around a circular body was simulated in the Julia programming language, where results for various physical sizes were extracted, as well as the execution times of simulations in multiple computer systems were compared.

At the beginning of the thesis, the chapters of fluid mechanics are formulated. The theoretical background is a necessity that helps the reader to better understand the problem. The role of fluid mechanics, and at the same time, the multiple possibilities that computational fluid dynamics can offer in everyday applications are described. The following chapters outline the steps of a CFD solution that is needed to implement comprehensive results. Immediately afterwards, the way the problem code is executed by parallel programming is analyzed. At the same time, the architecture of the computer, the way the memory is arranged and the basic components that the case study is implemented, i.e. the GPU graphics cards, are listed. Furthermore, the models of parallel programming are developed in order to select the most valuable model for solving the code on a graphics card according to the Julia programming language to be used. At the same time, the basic equations and magnitudes are described, which will help to compile the code under study logically. Finally, the final results of the simulations are evaluated as well as the execution time of the simulations for each different case studied.

Keywords

CFD and computer, Navier – Stokes equations, Discretization method, Structured grid, Parallel Programming, CUDA model, Julia programming language, JuliaGPU.

Περιεχόμενα

1 ΚΕΦΑΛΑΙΟ 1ο : ΡΕΥΣΤΟΜΗΧΑΝΙΚΗ

Η Ρευστομηχανική, εδώ και πολλές δεκαετίες, είναι μια από τις σημαντικότερες επιστημονικές κατηγορίες, όπου επικεντρώνεται στην κίνηση και στη κατάσταση των ρευστών. Προχωρώντας με τα χρόνια, η επιστήμη αυτή έχει διακλαδωθεί σε τρεις κλάδους. Ο πρώτος κλάδος ή όπως αλλιώς και Θεωρητική Ρευστομηχανική, στηρίζει τη φυσική και μαθηματική διερεύνηση των ροϊκών φαινομένων. Ο δεύτερος κλάδος, διαφορετικά Πειραματική Ρευστομηχανική, διατυπώνει τις ρευστοθερμικές και πειραματικές εφαρμογές στις πολλαπλές εφαρμοσμένες επιστήμες (όπως στην μηχανολογία, μετεωρολογία, χημική μηχανική, αεροναυπηγική κα). Ο τρίτος, με τους θεωρητικούς προβληματισμούς της ρευστομηχανικής εφαρμόστηκαν πάνω στους υπολογιστές προς την επίλυση τους, αναπτύσσοντας έτσι τον κλάδο της Υπολογιστικής Ρευστομηχανικής. Μέσα από τις προσομοιώσεις, τα θεωρητικά και πειραματικά αποτελέσματα που προκύπτουν, αντανακλούν στα φυσικά φαινόμενα της φύσης. Έτσι επιδιώκεται μια έγκυρη μελλοντική εφαρμογή, διά της πρόγνωσης της συμπεριφοράς των ροϊκών φαινομένων.

Η μαθηματική εξέταση των ροϊκών φαινομένων από την Θεωρητική ρευστομηχανική, αντιστοιχούν περίπου με τις πραγματικές ροές σύμφωνα με την πολυπλοκότητα τους. Η ανικανότητα λύσης των μαθηματικών εξισώσεων του φαινομένου, οδηγεί στην απλοποίηση του φαινομένου, όπου αυτό απωθείτε από το ρεαλιστικό αποτέλεσμα και καταλήγει στις προσεγγιστικές λύσεις. Συχνά διορθώνονται με την υποστήριξη των πειραματικών αποτελεσμάτων της Πειραματικής ρευστομηχανικής και αναπτύσσονται σε σχέσεις ως ημιεμπειρικές. Η απλότητα των προσεγγίσεων αυτών περιορίζεται στην ακριβή λύση του φαινομένου, γι’ αυτό το λόγο η σύγχρονη Υπολογιστική ρευστομηχανική αψηφά αυτές τις θεωρητικές δυσκολίες από την επαλήθευση των πειραματικών αποτελεσμάτων και εφαρμογών.

1.1 Ο Ρόλος της Ρευστομηχανικής

Βασικός ρόλος της ρευστομηχανικής είναι η προσπάθεια ανάπτυξης καινοτόμων μεθόδων αξιολόγησης ερευνητικών αποτελεσμάτων. Αυτό φαίνεται κυρίως στους διάφορους κλάδους όπου μπορούμε να εφαρμόσουμε αυτή την επιστήμη. Για παράδειγμα στην μηχανολογία, με εφαρμογές μετατροπής ενέργειας (αντλίες, στροβιλοκινητήρες, κινητήρες εσωτερικής καύσης κα). Στην μετεωρολογία, με φαινόμενα του αέρα στην ατμόσφαιρα (κίνηση ανέμου κα). Στη βιολογία, στη ροή του κυκλοφορικού κα. Στην τεχνολογία χημικών αντιδράσεων, ροή των ρευστών και μειγμάτων σε χημικές βιομηχανικές εγκαταστάσεις, πετροχημική παραγωγή κα. Στην επιστήμη πολιτικών μηχανικών, που περιλαμβάνει υδραυλικά έργα, ανεμοδυνάμεις στα κτίρια και κατασκευές, ροή αέρα και νερού με το έδαφος κα. Και στις μεταφορές, με πνευματικά και υδραυλικά συστήματα μεταφοράς, υποβρύχια κίνηση κα.

Όπως βλέπουμε από τις παραπάνω καθημερινές πρακτικές εφαρμογές, καταλαβαίνουμε την μεγάλη χρησιμότητα της ρευστομηχανικής.

1.2 Υπολογιστική Ρευστοδυναμική

Από τα αρχαία χρόνια μέχρι και τα μέσα του 20ου αιώνα η επιστήμη της ρευστοδυναμικής αποτελούσε μόνο καθαρά πειράματα και θεωρίες. Με το πέρας των χρόνων, προχώρησε αυτή η επιστήμη στην Υπολογιστική ρευστοδυναμική, εφαρμόζοντας με νέες τεχνικές μελέτης και αξιολόγησης αυτών των φυσικών φαινομένων.

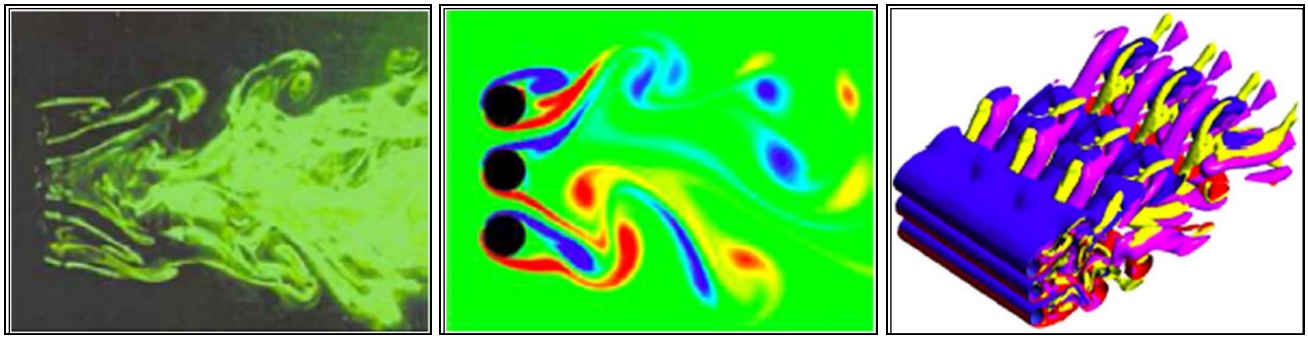
Η Υπολογιστική Ρευστοδυναμική (Computational Fluid Dynamics-CFD) είναι η μέθοδος της μαθηματικής μοντελοποίησης ενός φυσικού υπαρκτού φαινομένου που περιέχει τη ροή ρευστού και την αριθμητική επίλυση του. Χαρακτηρίζεται και ως η τέχνη της αντικατάστασης των μερικών παραγώγων με διακριτοποιημένες αλγεβρικές εξισώσεις και ολοκληρωμάτων, μέσω των οποίων εξάγονται σημειακές αριθμητικές τιμές των ιδιοτήτων του πεδίου ροής για κάθε σημείο του χώρου και για κάθε χρονική στιγμή. Αυτή η διαδικασία προσδίδει στην ανάλυση και βελτιστοποίηση της αεροδυναμικής κατά κύριο λόγο ενός αντικειμένου. Η υπολογιστική ικανότητα του λογισμικού CFD, την καθιστά ένα πολύ χρήσιμο εργαλείο, δίνει λύσεις σύμφωνα με τις πραγματικές φυσικές ιδιότητες των διάφορων μεγεθών που αλληλεπιδρούν στην ροή του ρευστού, αυτά τα μεγέθη μπορεί να είναι το ιξώδες, η πίεση, η θερμοκρασία, η ταχύτητα και η πυκνότητα. Παρέχει την δυνατότητα να εξετάζει το φαινόμενο που δημιουργείται και τα μεγέθη ταυτόχρονα. Το τελικό αντικείμενο του CFD, είναι ένα πακέτο αριθμητικών αποτελεσμάτων που αποτυπώνονται σε έναν ηλεκτρονικό υπολογιστή υψηλών προδιαγραφών.

Τα βασικά εργαλεία του λογισμικού της CFD περιέχει ένα μαθηματικό μοντέλο και μια αριθμητική μέθοδος της πραγματικής περίπτωσης. Οι εξισώσεις Navier – Stokes ορίζονται ως το μαθηματικό μοντέλο της περίπτωσης. Αναλύουν τις διάφορες μεταβολές των φυσικών ιδιοτήτων και τη μεταφορά της ενέργειας μέσα στη ροή του ρευστού. Εκτός αυτού, ο προσδιορισμός των κατάλληλων αριθμητικών μεθόδων είναι σημαντικοί για το σχηματισμό μιας αξιόπιστης λύσης. Η ανάλυση CFD αποτελεί βασικό στοιχείο για τη δημιουργία μιας ανάπτυξης προϊόντων, καθώς ο αριθμός των φυσικών πρωτοτύπων μπορεί να μειωθεί απότομα. Εξαιτίας της μεγάλης αφιέρωσης χρόνου και χρήματος για την ακριβή εκπόνηση των υπολογισμών των φαινομένων των ροών, μηχανικοί και επιστήμονες δημιούργησαν μια μέθοδο με ταχύτερη εξέταση, που τους επιτρέπει να συνδυάσουν το μαθηματικό μοντέλο και την αριθμητική μέθοδο με έναν υπολογιστή, όπου θα διευκρινιστεί στα προσεχώς κεφάλαια.

1.2.1 Εφαρμογές Υπολογιστικής Ρευστοδυναμικής – CFD

Όπου υπάρχει υγρό και αέρας, υπάρχει CFD. Έχοντας αναφέρει προηγουμένως, το αρχικό στάδιο για τη διεξαγωγή προσομοίωσης CFD καθορίζεται κατάλληλα από ένα πραγματικό μαθηματικό μοντέλο. Οι προσεγγίσεις και οι υποθέσεις δίνουν κατεύθυνση μέσω διαδικασιών λύσης για την εξέταση της υπόθεσης στον υπολογιστικό τομέα. Ασχολείται επίσης με μεθόδους που είναι σχεδιασμένοι να επιλύουν εξισώσεις της κίνησης του ρευστού σε δύο ή και τρεις διαστάσεις. Είναι πάνω σε μη τυποποιημένες εφαρμογές, όπου δεν μπορούν να βρεθούν σε εγχειρίδια. Καθώς αυτοί οι μέθοδοι έχουν χρησιμοποιηθεί στη μηχανική υψηλής τεχνολογίας (για παράδειγμα, αστροναυτική και αεροναυπηγική) εξ’ αρχής, χρησιμοποιούνται συχνότερα σε τομείς της μηχανικής όπου η γεωμετρία είναι περίπλοκη ή κάποιο σημαντικό χαρακτηριστικό που δεν μπορεί να αντιμετωπιστεί με τυπικές μεθόδους.

Γίνεται αντιληπτό ότι η υπολογιστική ρευστομηχανική μπορεί να χρησιμοποιηθεί με αμέτρητους τρόπους σε κάθε κλάδο ξεχωριστά. Αρχικά, ως ένα ερευνητικό εργαλείο, για την καλύτερη σαφήνεια των περιβαλλοντολογικών γεγονότων ροής (μη γραμμική). Αυτά ταυτίζονται με τη αλληλεπίδραση των οριακών στρωμάτων, τη μεταφορά, τις αναταράξεις, τη διάχυση και τον διασκεδασμό. Στην **Εικόνα 1α**, βλέπουμε ένα στιγμιότυπο μιας ασταθής ροής κατάντη των τριών πλαϊνών κυλίνδρων. Στην **Εικόνα 1β**, βλέπουμε τον μοντελοποιημένο CFD αριθμητικό υπολογισμό, μοντέλου δινορρευμάτων. Επίσης, παρατηρούμε περισσότερες λεπτομέρειες της ροής τρισδιάστατα, **Εικόνα 1γ**. Σαφώς αυτό το παράδειγμα επιβεβαιώνει την καλύτερη κατανόηση των μεγεθών κατά την προσομοίωση, αντί του εργαστηριακού πειράματος.



(α) **(β)** **(γ)**

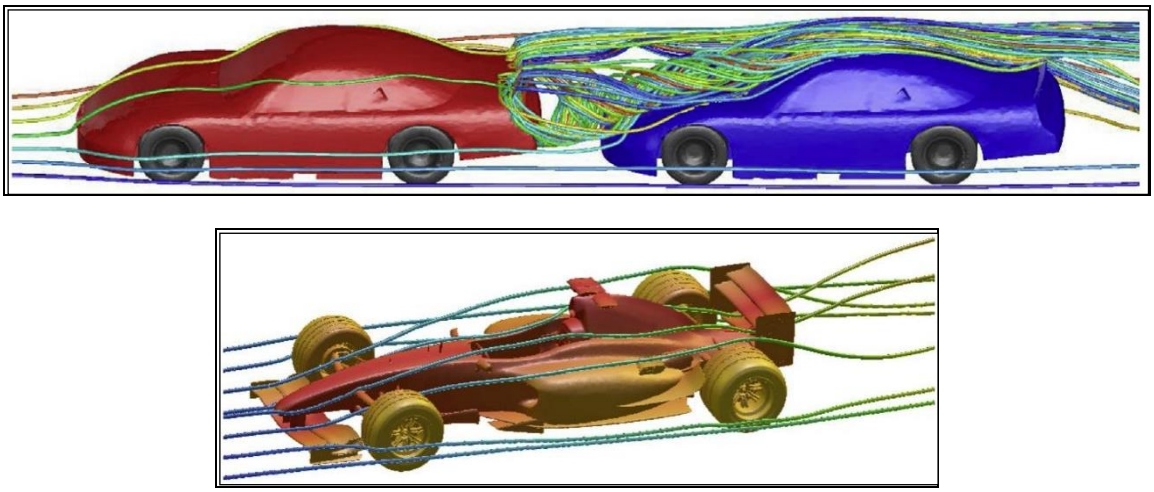
Εικόνα 1: Αριθμητικό πείραμα CFD ροής κατάντη τριών πλαϊνών κυλίνδρων.
(α) εργαστηριακή παρατήρηση, (β) αριθμητική προσομοίωση 2D, (γ) αριθμητική προσομοίωση 3D.

Σε συνέχεια, εφόσον κατέχει άπειρες πληροφορίες η υπολογιστική ρευστομηχανική, συμπεραίνεται ότι είναι ένα πολύ χρήσιμο εκπαιδευτικό εργαλείο. Ενθαρρύνοντας τον εκπαιδευόμενο άνθρωπο να μοντελοποιεί φυσικά φαινόμενα, είτε μέσω θεωρητικής είτε πρακτικής σκοπιάς. Προχωρώντας, από τα δύο προηγούμενα εργαλεία, αναπόσπαστο είναι ότι πολλές βιομηχανικές εταιρείες αναζητούν καινοτόμες λύσεις για την βελτιστοποίηση και την πρόβλεψη των αναγκών. Οι λογισμικές εφαρμογές παρέχουν αριθμητικές λύσεις και αναλύσεις στα σχετικά προβλήματα, με την απλή χρήση συμβατικών υπολογιστών. Ως εργαλείο σχεδιασμού, η CFD προσφέρει ουσιαστικές γνώσεις όσον αφορά τα χαρακτηριστικά της ροής εντός του εξοπλισμού και των διαδικασιών που χρειάζονται για την βελτίωση της μακροζωίας, αύξηση της παραγωγής και μείωση των αποβλήτων. Η διαρκείς ανάπτυξη της υπολογιστής ισχύς επαναστατεί στη χρήση CFD στις βιομηχανίες. Αυτές οι βιομηχανικές λύσεις αναπτύσσονται για να καταδείξουν περαιτέρω την ευρεία εφαρμογή αυτής της συγκεκριμένης τεχνολογίας στην πράξη.

Η δύναμη της CFD βρίσκεται πίσω στα παρασκήνια της κάθε εφαρμογής, η οποία έχει αλλάξει τον τρόπο και την οπτική των πραγμάτων. Παρουσιάζονται παρακάτω μερικές εφαρμογές από κάθε τομέα.

A. Μηχανική Οχημάτων

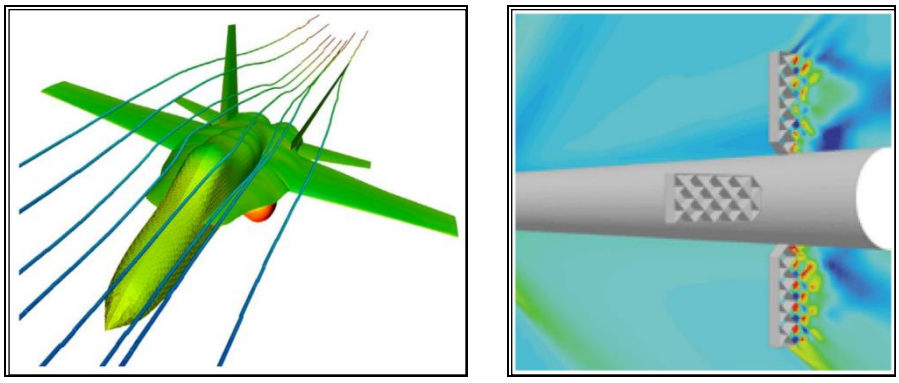
Ο συνδυασμός μηχανικής και υπολογιστή βρίσκεται στην πρώτη γραμμή της δημιουργία νέων εσωτερικών συστημάτων που ενισχύει τη συνολική οδηγητική εμπειρία, την άνεση και την ασφάλεια των επιβατών και προωθεί στην οικονομία καυσίμου. Η αεροδυναμική που απεικονίζεται στην **Εικόνα 2**, μειώνει την κατανάλωση καυσίμου.



Εικόνα 2: Αεροδυναμική οχημάτων.

B. Αεροναυπηγική

Η προοπτική βελτιστοποίησης της πτήσης είναι λόγος για ανταγωνισμό. Η προσομοίωση των γραμμών ρευστού ενός τζετ F18, **Εικόνα 3α** και η πρόβλεψη περιγραμμάτων συντελεστή πίεσης σε γωνία προσβολής 10^{o} γύρω από ένα υπερηχητικό σύστημα πυραύλου με πτερύγια πλέγματος, **Εικόνα 3β**.



(α) **(β)**

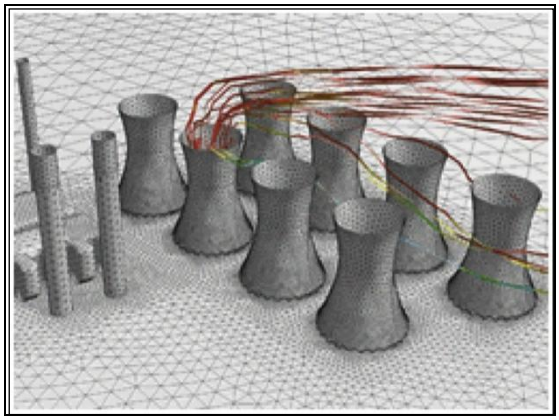
Εικόνα 3: (α) προσομοίωση γραμμών ρευστού σε F18, **(β)** περίγραμμα συντελεστή πίεσης σε γωνία προσβολής 10^{o} γύρω από υπερηχητικό σύστημα πυραύλου.

Υπάρχουν επίσης και άλλες εφαρμογές που η CFD έχει χρησιμοποιηθεί για την επίλυση ορισμένων σύνθετων λειτουργικών προβλημάτων σε σχέδια αεροσκαφών, όπως η μελέτη των επιπτώσεων των στροφών που ακολουθούν στην ασφαλή λειτουργία διαδοχικών αεροσκαφών που απογειώνονται και

προσγειώνονται στον αεροδιάδρομο, παράλληλα και η βελτίωση του εξαερισμού, της θέρμανσης και της ψύξης της καμπίνας.

C. Πολιτική και Περιβαλλοντική Μηχανική

Πολλές περιπτώσεις έχουν ως επίκεντρο την επίλυση των περιβαλλοντικών. Για παράδειγμα, το CFD έχει χρησιμοποιηθεί για την πρόβλεψη του ρύπου που διασκορπίζεται από έναν πύργο ψύξης που υπόκειται σε συνθήκες ανέμου όπως απεικονίζεται στην **Εικόνα 4**, παρακάτω.



Εικόνα 4: Παράδειγμα εφαρμογής CFD για διασπορά λοφίων από πύργο ψύξης.

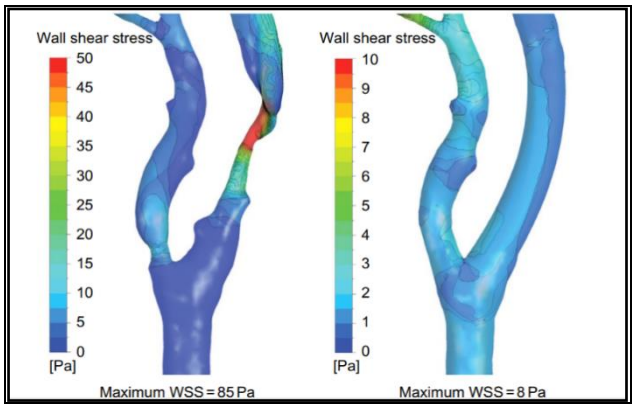
D. Ανανεώσιμες Πηγές και Παραγωγή Ενέργειας

Στην αγορά ενέργειας εξακολουθούν οι βιομηχανίες να χρησιμοποιούν τις παραδοσιακές πηγές ενέργειας, μεγιστοποιώντας όμως την απόδοση της επένδυσης μέσω CFD. Η βελτίωση των πτερυγίων της ανεμογεννήτριας βοηθά στην παραγωγή σταθερής ισχύς ανεξαρτήτως των συνθηκών ανέμου και των κλίσεων του εδάφους, καθώς στηρίζεται από μια 3D προσομοίωση, **Εικόνα 5**. Οι εκτιμήσεις της CFD έδωσαν την ευκαιρία στους μηχανικούς να μελετήσουν καλύτερα την οικονομική βιωσιμότητα των αιολικών πάρκων όπου χρειάζονται ακριβή αποτελέσματα ώστε να μειωθεί ο οικονομικός κίνδυνος.

Εικόνα 5: Παράδειγμα εφαρμογής CFD στην πρόβλεψη της ροής αέρα πάνω σε μια ανεμογεννήτρια σε αιολικό πάρκο.

E. Βιοϊατρική Επιστήμη και Μηχανική

Οι ιατρικοί ερευνητές βασίζονται σήμερα σε εργαλεία προσομοίωσης για να βοηθήσουν στην πρόβλεψη της συμπεριφοράς της κυκλοφοριακής ροής αίματος μέσα στο ανθρώπινο σώμα. Οι υπολογιστικές προσομοιώσεις μπορούν να παρέχουν ανεκτίμητες πληροφορίες που είναι εξαιρετικά δύσκολο να ληφθούν πειραματικά. Η **Εικόνα 6** δείχνει μια από τις πολλαπλές ικανότητες του CFD στην ιατρική, για την πρόβλεψη της τάσης διάτμησης τοιχώματος (WSS) σε στενωτική αρτηρία και έπειτα από την προσθήκη stent.

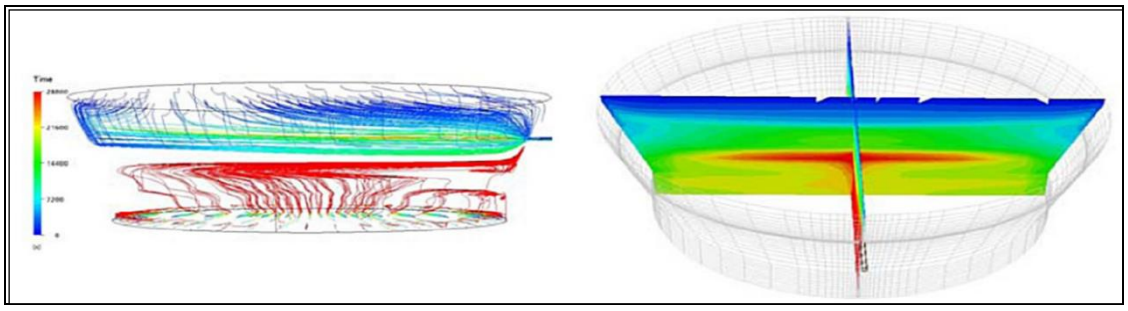


Εικόνα 6: Παράδειγμα πρόβλεψης CFD της τάσης διάτμησης τοιχώματος (WSS) σε στενωτική αρτηρία (αριστερά) και σε αρτηρία με stent (δεξιά).

Οι ιατρικές προσομοιώσεις των κυκλοφοριακών λειτουργιών προσφέρουν πολλά οφέλη. Μειώνουν τις πιθανότητες μετεγχειρητικών επιπλοκών, βοηθούν στην ανάπτυξη καλύτερων χειρουργικών επεμβάσεων και παρέχουν καλή κατανόηση των βιολογικών διεργασιών και αποτελεσματικότερου και λιγότερο καταστροφικού ιατρικού εξοπλισμού. Επιπλέον, η CFD χρησιμοποιείται όλο και περισσότερο μέσω εικονικών πρωτοτύπων για να προτείνει τον καλύτερο σχεδιασμό για χειρουργικές ανακατασκευές.

F. Μεταλλουργική

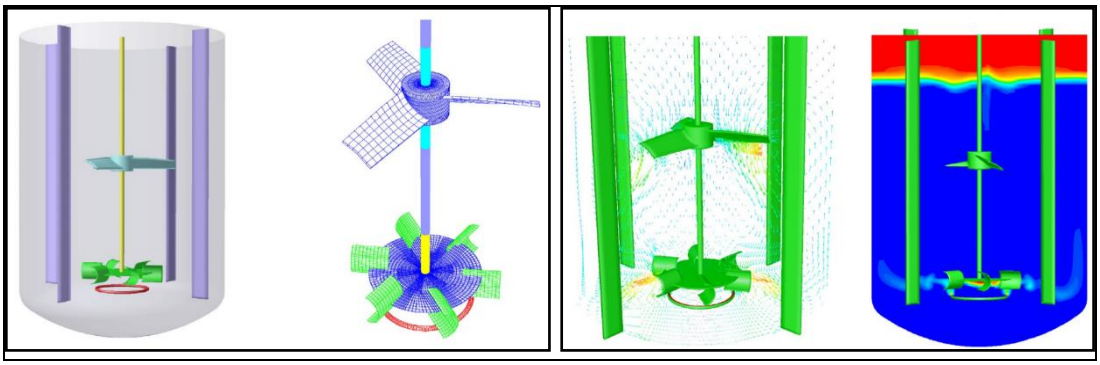
Η χρήση του CFD στην προσομοίωση ετερογενούς και αργής διάλυσης σωματιδίων κώδικα συσκευασμένης κλίνης φαίνεται στην **Εικόνα 7**. Εδώ, το προβλεπόμενο μοτίβο ροής λιωμένου σιδήρου και διάλυσης άνθρακα στην εστία υψικαμίνου λαμβάνεται μέσω του μοντέλου ισορροπίας πληθυσμού σε συνδυασμό με το μοντέλο ροής.



Εικόνα 7: Παράδειγμα εφαρμογής CFD για την πρόβλεψη ροής λιωμένου σιδήρου και διάλυσης άνθρακα στην εστία υψικαμίνου, (αριστερά, χρονοδιάγραμμα), (δεξιά, συγκέντρωση).

G. Χημική και Ορυκτή Επεξεργασία

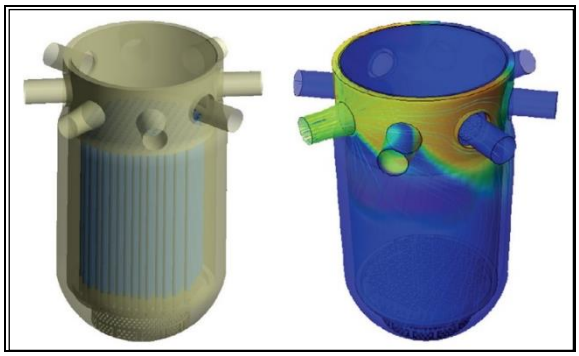
Βασική ανάγκη μερικών βιομηχανιών είναι η επεξεργασία των χημικών και των ορυκτών. Ποσά ενέργειας παραμορφώνουν τις πρώτες ύλες και διαμορφώνουν τα διάφορα προϊόντα, ανάλογα με τις ανάγκες του κόσμου. Από αυτό καταλαβαίνουμε ότι οι λειτουργικές διαδικασίες πρέπει να είναι πιο αποδοτικές από ενεργειακής, ασφαλιστικής και αντιρρυπαντικής άποψης. Η αύξηση της απόδοσης των αντιδραστήρων αερίου έχουν μεγάλη σημασία στη χημική βιομηχανία. Η **Εικόνα 8** διατυπώνει το περιγράμματα της πολυβάθμιας κατανομής των φυσαλίδων εντός της αναδευόμενης δεξαμενής συνοδευόμενη από την τοπική συμπεριφορά της ροής, που υποδεικνύεται από διανύσματα ταχύτητας, γύρω από μία από τις περιστρεφόμενες λεπίδες. Οι πληροφορίες για τη μεταφορά ρευστών μέσω CFD διασφαλίζει την καλύτερη διαθεσιμότητα δεδομένων προκειμένου να αυξηθεί η απόδοση βελτιώνοντας τις ροές των ρευστών, μειώνοντας έτσι το λειτουργικό κόστος και αυξάνοντας την αποδοτικότητα του συστήματος.



Εικόνα 8: Παράδειγμα εφαρμογής CFD, προσομοίωση αντιδραστήρα αναδευόμενης δεξαμενής αερίου.

H. Πυρηνική Ασφάλεια

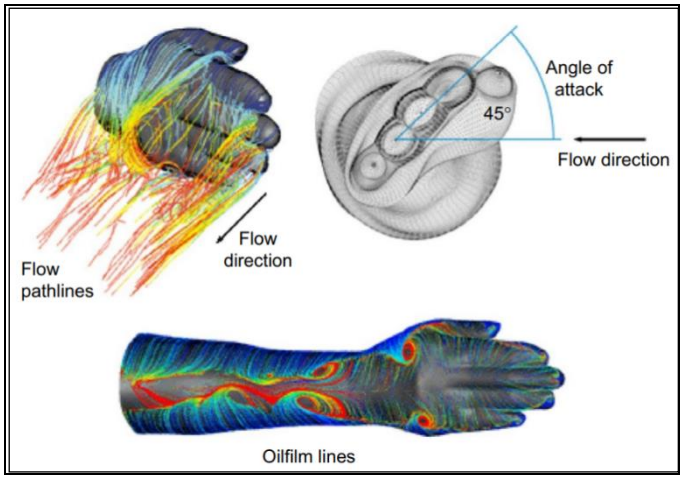
Η ανάλυση ασφάλειας των πυρηνικών εγκαταστάσεων κεντρίζει πολύ το ενδιαφέρον για την CFD να προβλέψει εφικτά μια σειρά σημαντικών φαινομένων ροής, που λογικά χωρίς την παρακολούθηση τους θα προκαλούσαν προβλήματα. Η **Εικόνα 9** δείχνει την εκκίνησης της αντλίας λόγω ισχυρής ροής με ώθηση στο ακροφύσιο εισόδου όπου το οριζόντιο τμήμα της ροής κυριαρχεί στο κάτω σε αντιδραστήρα υπό πίεση νερού.



Εικόνα 9: Παράδειγμα εφαρμογής CFD στην πρόβλεψη της τυρβώδους ανάμειξης στην εγκατάσταση δοκιμής δωματίου κατά την διάρκεια μεταβατικών αραιώσεων βορίου (εκκίνηση της πρώτης αντλίας ψυκτικού μέσου).

I. Αθλητισμός

Πολύ πρόσφατα, η CFD στον αθλητικό χώρο είναι να επιτύχει τη μέγιστη προωθητική απόδοση για κάθε άθλημα. Συγκεκριμένα στην κολύμβηση όπως αποδεικνύεται από το παράδειγμα στην **Εικόνα 10**. Οι έρευνες CFD μπορούν να αξιολογήσουν τη ροή γύρω από το χέρι και το αντιβράχιο ενός κολυμβητή κατά τη διάρκεια των φάσεων πρόωσης στο ελεύθερο και στη πεταλούδα. Οι δυνάμεις ανύψωσης και έλξης σταθερής κατάστασης του χέρι και του βραχίονα προσδιορίζονται μέσω της εφαρμογής εξελιγμένου μοντέλου στροβιλισμού και προσαρμοστικού πλέγματος. Για αυτό το παράδειγμα, οι συντελεστές δύναμης, που αξιολογήθηκαν σε γωνίες επίθεσης που κυμαίνονται από 15 έως 195 μοίρες και για τις διάφορες καταστάσεις αναταραχής νερού, βρέθηκαν να συγκρίνονται πολύ καλά με συντελεστές που αναπτύχθηκαν πειραματικά σε αεροδυναμική σήραγγα, δεξαμενή ρυμούλκησης και αγωγού. Η επιτυχής σύγκριση των προσομοιωμένων αποτελεσμάτων με πειραματικά δεδομένα επικυρώνει έτσι την επιλεγμένη τεχνική μοντελοποίησης CFD.



Εικόνα 10: Παράδειγμα εφαρμογής CFD για το σχεδιασμό της βέλτιστης διαδρομής. (Ευγενική προσφορά της κολύμβησης των ΗΠΑ.

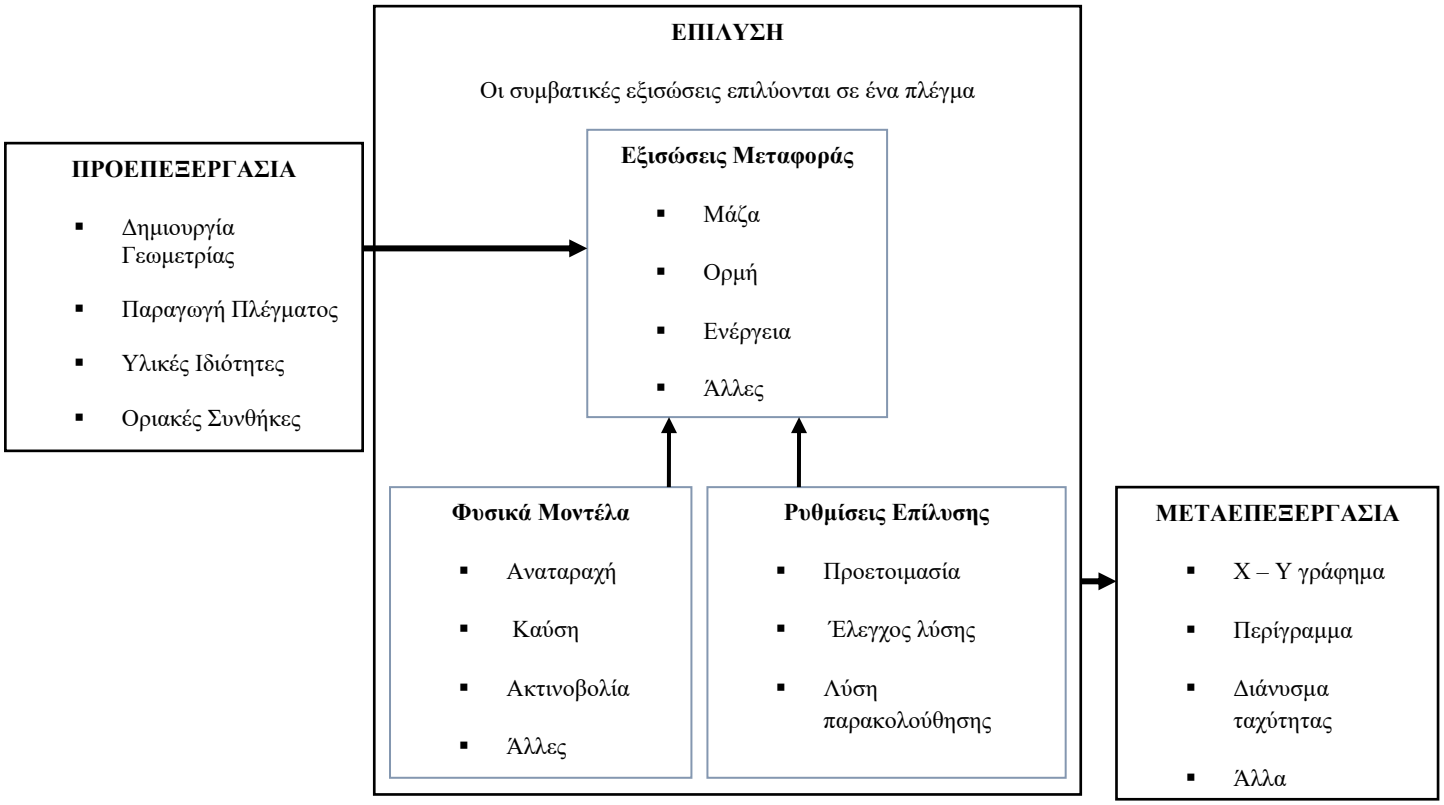
Η σύνταξη αυτού του κεφαλαίου πραγματοποιήθηκε με στόχο να φανερώσει το CFD από τα παρασκήνια των μεγάλων και πρωτότυπων εφαρμογών. Η βελτιστοποίηση της απόδοσης, της προστασίας και οικονομίας έχει πρωταρχικό ρόλο στην καθημερινή ζωή του ανθρώπου. Γίνεται λοιπόν φανερό πως αυτές οι λίγες εφαρμογές που αναφέρθηκαν, μπορούν να εξελιχθούν ταχύτερα, με έναν υπολογιστή.

2 ΚΕΦΑΛΑΙΟ 2ο : ΜΕΘΟΔΟΛΟΓΙΑ ΛΥΣΗΣ CFD

Σε αυτό το κεφάλαιο θα περιγραφεί η διαδικασία που ακολουθεί μια CFD λύση. Με την μετατροπή των βασικών CFD εξισώσεων σε υπολογιστικούς κώδικες προς την επίλυση προβλημάτων ροής ρευστών. Οι κώδικες αποτελούν την ολοκληρωμένη ανάλυση CFD που παρέχει τρία κύρια στάδια:

- **Προεπεξεργασία – Preprocess**
- **Επίλυση – Solver**
- **Μεταεπεξεργασία – Postprocess**

Συγκεκριμένα, προορίζονται τα βήματα **Σχήμα 1**, των τριών προαναφερθέντων στοιχείων στο πλαίσιο της ανάλυσης CFD, όπου θα αναπτυχθούν και στις επόμενες ενότητες.



Σχήμα 1: Οι συνιστώσες των τριών κύριων στοιχείων για την διαδικασία CFD.

Η ανάπτυξη νέων κωδίκων προσφέρει στους ερευνητές σημαντική πρόοδο στην έρευνα τους. Στη συνέχεια θα παρατεθούν κατά σειρά, οι συνιστώσες που απαιτούνται για τη λύση της αριθμητικής μεθόδου, πιο λεπτομερώς στη συνέχεια.

2.1 1ο Στάδιο : Προεπεξεργασία – Preprocess

2.1.1 Μαθηματικό Μοντέλο

Το σημείο έναρξης της κάθε αριθμητικής μεθόδου είναι το μαθηματικό μοντέλο, δηλαδή το σύνολο μερικών διαφορικών ή ολοκληρο-διαφορικών εξισώσεων και οριακών συνθηκών. Γίνεται η επιλογή του σωστού μοντέλου για την εφαρμογή (ατριβές, ασυμπίεστο, τυρβώδες, δισδιάστατο, τρισδιάστατο κα). Ο ορισμός και η δημιουργία γεωμετρίας της περιοχής ροής για τους υπολογισμούς CFD είναι να επιτρέψει τη δυναμική της ροής να αναπτυχθεί επαρκώς σε όλο το μήκος αυτών των υπολογιστικών τομέων. Μπορεί να διαθέτει επίσης νόμους διατήρησης, όμως είναι συνήθως σχεδιασμένο για συγκεκριμένο σύνολο εξισώσεων.

2.1.2 Μέθοδος Διακριτοποίησης

Μετά την κατάλληλη επιλογή του μαθηματικού μοντέλου, πρέπει να επιλεχθεί και η κατάλληλη μέθοδος διακριτοποίησης, δηλαδή μια μέθοδος προσέγγισης των διαφορικών εξισώσεων από ένα σύστημα αλγεβρικών εξισώσεων για τις μεταβλητές σε κάποιο σύνολο διακριτών θέσεων στο χώρο – χρόνο.

A. Μέθοδος Πεπερασμένων Διαφορών

Η μέθοδος των πεπερασμένων διαφορών (Finite Difference Method-FDM) προσφέρει έναν τρόπο υπολογισμού ενός πολυώνυμου με τη χρήση των τιμών του σε διάφορα διαδοχικά σημεία. Είναι μια καλή προσέγγιση για την εύρεση του γενικού όρου σε ένα μοτίβο, αν ακολουθεί μια πολυωνυμική μορφή. Επίσης είναι σταθερή μέθοδος, ακριβείς, ταχείας σύγκλισης και απλή στην επίλυση μερικών διαφορικών εξισώσεων (Partial Differential Equations-PDEs) προβλημάτων. Οι διαφορικοί όροι της εξίσωσης μετατρέπονται σε γραμμική αλγεβρική εξίσωση, την λεγόμενη εξίσωση πεπερασμένων διαφορών και τα εξαρτώμενες μεταβλητές μελετούνται μόνο σε διακριτά σημεία.

Εφόσον η μέθοδος FDM είναι μία από τις μεθόδους που χρησιμοποιείται για την επίλυση διαφορικών εξισώσεων που είναι δύσκολο να λυθούν αναλυτικά. Ο υποκείμενος τύπος είναι:

$$\frac{\partial p}{\partial x} = \lim_{\Delta x \to 0} \frac{p(x) - p(x - \Delta x)}{\Delta x} \qquad [1]$$

Από την παραπάνω εξίσωση, διακριτοποιείται, δηλαδή πλεγματοποιείται το ροϊκό πεδίο, μια εξίσωση μερικής διαφοράς μια αριθμητική μέθοδο για την επίλυση του PDE.

Το FDM είναι μια πολύ ευέλικτη τεχνική μοντελοποίησης, η οποία είναι σχετικά εύκολη για τους χρήστες να κατανοήσουν και να εφαρμόσουν. Το FDM είναι μια τεχνική χρονικού τομέα, η οποία μπορεί να βρει τη συγκέντρωση χρωστικής παντού στον υπολογιστικό τομέα σε ένα δεδομένο χρονικό πλαίσιο.

B. Μέθοδος Πεπερασμένων Στοιχείων

Αρχικά αναπτύχθηκε κυρίως για την επίλυση προβλημάτων στερεάς κατάστασης, αλλά έχει βρει σε ευρεία εφαρμογή και σε όλους τους τομείς της Υπολογιστικής Φυσικής και της μηχανικής, καθώς και στην CFD. Η FEM είναι μακράν η πιο ευέλικτη μέθοδος όλων των μεθόδων που έχει υπάρξει και μπορεί να προσαρμοστεί σε ένα μεγάλο φάσμα αριθμητικών εφαρμογών. Αυτό καθιστά FEM ως καθολικό εργαλείο για την επίλυση διαφορικών εξισώσεων αριθμητικά.

Η μέθοδος πεπερασμένων στοιχείων FEM επιτρέπει τη διακριτοποίηση του συνεχούς σε έναν πεπερασμένο αριθμό τμημάτων (ή στοιχείων) και τονίζει ότι τα χαρακτηριστικά του συνεχούς τομέα μπορούν να εκτιμηθούν συγκεντρώνοντας τις παρόμοιες ιδιότητες των διακριτοποιημένων στοιχείων ανά κόμβο. Ως αποτέλεσμα, το FEM έχει εφαρμοστεί αυστηρά για την επίλυση μιας ευρείας ποικιλίας προβλημάτων στην εφαρμοσμένη επιστήμη και τη μηχανική και έχει αναπτυχθεί γρήγορα με την πάροδο των χρόνων. Η βασική έννοια του FEM μπορεί να θεωρηθεί ως διάσπαση, σε μεμονωμένα μικρό κώδικα και εύρεση τοπικών επιλύσεων που ικανοποιούν τη διαφορική εξίσωση εντός των ορίων αυτής της έκδοσης κώδικα. Με τη σύνδεση των μεμονωμένων λύσεων σε αυτό το κώδικα, μπορεί να επιτευχθεί μια κοινή λύση.

C. Μέθοδος Πεπερασμένων Όγκων

Η αριθμητική διακριτοποίηση περιλαμβάνει την απαίτηση της διατήρησης της ορμής, μάζας και ενέργειας, αλλιώς μια συναρτησιακή μέθοδος, δεν εμφανίζει κάποια δυσλειτουργία σύγκλισης, κατά συνέπεια οι επαναληπτική υπολογισμοί να μειώνονται άρα έχει ως αποτέλεσμα την μείωση του υπολογιστικού χρόνου. Η μέθοδος αυτή είναι κατάλληλη και για τρισδιάστατα ροϊκά πεδία μεγάλων διαστάσεων. Στις σύνθετες όμως γεωμετρίες του ροϊκού πεδίου παρουσιάζουν δυσκολίες, με δομημένη πλεγματοποίηση. Για την ακρίβεια στην τρισδιάστατη περίπτωση πρέπει να είναι στοιχειώδεις όγκοι τοπολογικά καρτεσιανού συστήματος με έξη ακμές και πλευρές.

2.1.3 Σύστημα Συντεταγμένων και Βάσεων

Οι εξισώσεις διατήρησης μπορούν να γραφτούν σε πολλές διαφορετικές μορφές, ανάλογα με το σύστημα συντεταγμένων και τους φορείς βάσης που χρησιμοποιούνται. Για παράδειγμα, μπορεί κανείς να επιλέξει καρτεσιανά, κυλινδρικά, σφαιρικά, καμπυλόγραμμα ορθογώνια ή μη ορθογώνια συστήματα συντεταγμένων, τα οποία μπορεί να είναι σταθερά ή κινούμενα. Η επιλογή εξαρτάται από τη ροή που θέλουμε να πετύχουμε και μπορεί να επηρεάσει τη μέθοδο διακριτοποίησης και τον τύπο πλέγματος που θα χρησιμοποιηθεί.

Κάποιος πρέπει επίσης να επιλέξει τη βάση στην οποία θα οριστούν διανύσματα και τανυστές (σταθερά ή μεταβλητά, συνδιακύμανση ή αντίθεση, κα.). Σύμφωνα με αυτήν την επιλογή, το διάνυσμα ταχύτητας και ο τανυστής τάσης μπορούν να εκφραστούν σε όρους, π.χ. καρτεσιανά, συνδιακύμανση ή αντιπαραβολή, φυσικά ή μη φυσικά συστατικά προσανατολισμένα στις συντεταγμένες. Στην προκείμενη περίπτωση μελέτης της διπλωματικής εργασίας έχει επιλεγεί το Καρτεσιανό σύστημα συντεταγμένων.

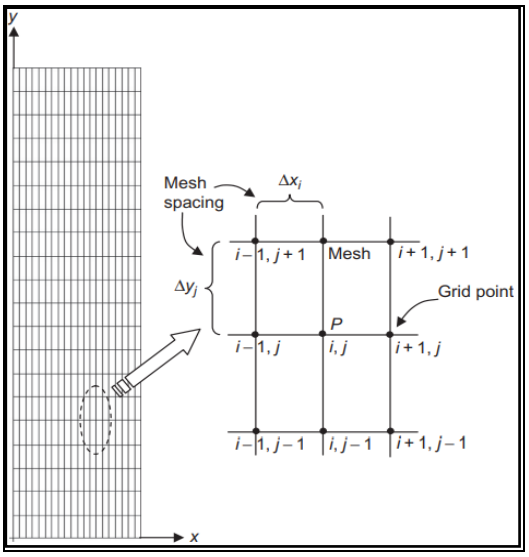
2.1.4 Αριθμητικό Πλέγμα

Το αριθμητικό πλέγμα πρόκειται να υπολογίσει τις μεταβλητές από τις διακριτές θέσεις, είναι ουσιαστικά μια διακριτή αναπαράσταση της γεωμετρίας που λύνεται το πρόβλημα. Η συγκέντρωση των σχημάτων στο εσωτερικό ή και μερικώς εσωτερικό της μελετώμενης γεωμετρίας CFD καλείται πλέγμα. Διαχωρίζει τον τομέα λύσης σε έναν πεπερασμένο αριθμό υποτομέων (στοιχεία, τόμοι ελέγχου κ.λπ.). Υπάρχουν πολλοί διαφορετικοί τύποι σχημάτων πλέγματος / κελιών, όπως θα δούμε πιο αναλυτικά.

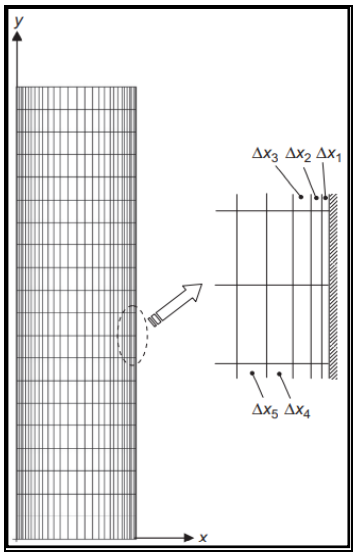
Σε ένα δισδιάστατο δομημένο πλέγμα, τα σημεία πλέγματος αντιμετωπίζονται κανονικά από τους δείκτες (i, j) όπου αντιπροσωπεύει αντίστοιχα τους δείκτες (x, y) σημείων κατεύθυνσης. Με την κατανομή κατάλληλων διακριτών τιμών για Δxi και Δyi, οι συντεταγμένες στην κατεύθυνση X και την κατεύθυνση Y μέσα στον φυσικό χώρο μπορούν πλέον να προσδιοριστούν σταδιακά, με αποτέλεσμα ένα ορθογώνιο πλέγμα που καλύπτει ολόκληρο τον τομέα. Για ομοιόμορφα κατανεμημένα σημεία πλέγματος, η απόσταση του Δxi ή Δyi είναι μια ενιαία αντιπροσωπευτική τιμή στην κατεύθυνση X ή Y κατεύθυνση, **Εικόνα 11**. Για ανομοιόμορφα κατανεμημένα σημεία πλέγματος, **Εικόνα 12**, η απόσταση των Δxi ή Δyi μπορεί ωστόσο να πάρει αποτελεσματικά έναν αριθμό διακριτών τιμών. Ως εκ τούτου, θα μπορούσαμε εύκολα να φτιάξουμε ένα πλέγμα άνισων αποστάσεων στην X ή Y κατεύθυνση, έχουμε πιο χονδροειδές πλέγμα μακριά από τα κάθετα όρια τοίχων, στον x άξονα και ένα λεπτότερο πλέγμα συγκεντρωμένο κοντά στα κάθετα όρια τοίχων στον y άξονα.

1. Δομημένο Πλέγμα (Structured Mesh)

Τα στοιχεία εδώ, είναι ευθυγραμμισμένα με συγκεκριμένο τρόπο ή αλλιώς ακολουθούν ένα δομημένο μοτίβο.



Εικόνα 11: Ομοιόμορφο ορθογωνικό πλέγμα.



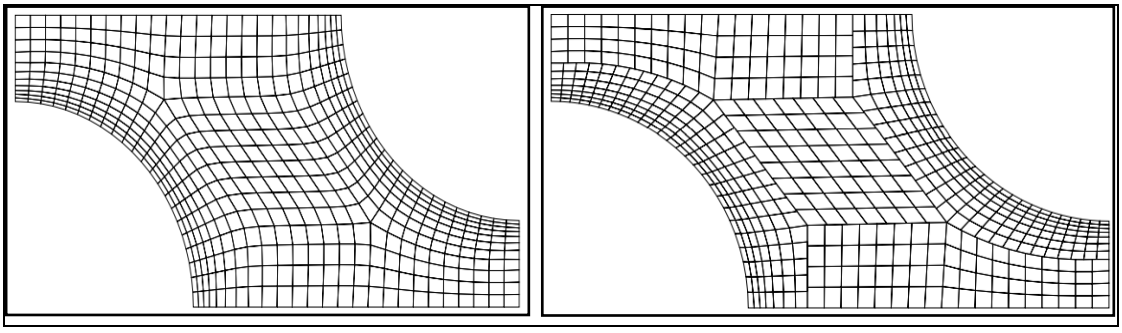
Εικόνα 12: Ανομοιόμορφο ορθογωνικό πλέγμα.

Για την κάλυψη της μελετώμενης γεωμετρίας χρειάζεται μόνο ένα μπλοκ ή με πολλαπλά μπλοκ, αναλόγως αν υλοποιηθεί σε απλές γεωμετρίες ή σύνθετες. Έτσι προκύπτει το Πλέγμα Μονο-Μπλοκ (Block-structured) και το Πλέγμα Πολλαπλών Μπλοκ (Multblock mesh), με πιο λεπτομέρεια παρακάτω:

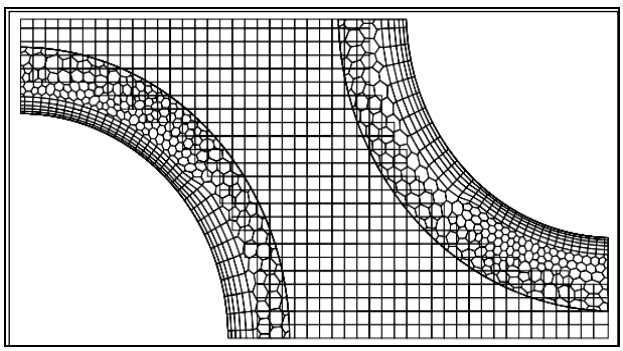
- **Block-Structured**

Ένα δομημένο πλέγμα μπλοκ με σύμμορφες διεπαφές, **Εικόνα 13α.** Οι διεπαφές των μπλοκ ήταν αρχικά κανονικές επιφάνειες (τμήματα κυλίνδρων ή αεροπλάνων), αλλά με ορισμένες εργασίες εξομάλυνσης που σχεδιάστηκαν για τη βελτίωση της ποιότητας του πλέγματος, έγιναν καμπύλες επιφάνειες. Με καλό σχεδιασμό δομής μπλοκ, μπορεί κανείς να δημιουργήσει ένα πλέγμα με καλή ποιότητα αλλά σε μεγάλο χρόνο.

Στην **Εικόνα 13β** έχουμε ένα πλέγμα δομημένο με μπλοκ με μη συμμορφούμενες διεπαφές, εκτός από το ότι το πλέγμα είναι λεπτότερο σε μπλοκ γύρω από σωλήνες. Αυτό το είδος πλέγματος είναι πιο ευέλικτο από τα προηγούμενα, χρησιμοποιεί λεπτότερα πλέγματα στις περιοχές που απαιτούν μεγαλύτερη ανάλυση. Η μη συμμορφούμενη διεπαφή μπορεί να αντιμετωπιστεί με πλήρως συντηρητικό τρόπο ή με τη χρήση "κρεμαστών κόμβων". Εδώ η εξομάλυνση πλέγματος μπορεί να εφαρμοστεί μόνο σε κάθε μπλοκ: οι διεπαφές διατηρούν συνήθως το αρχικό τους σχήμα, όπως μπορεί να φανεί συγκρίνοντας τα πλέγματα στην **Εικόνα 13**. Ο προγραμματισμός είναι πιο δύσκολος. Οι λύτες για δομημένα πλέγματα μπορούν να εφαρμοστούν κατά μπλοκ και πιο περίπλοκοι τομείς ροής μπορούν να αντιμετωπιστούν με αυτά τα πλέγματα. Η τοπική βελτίωση είναι δυνατή κατά μπλοκ (δηλαδή, το πλέγμα μπορεί να βελτιωθεί σε ορισμένα μπλοκ).



(α) **(β)**



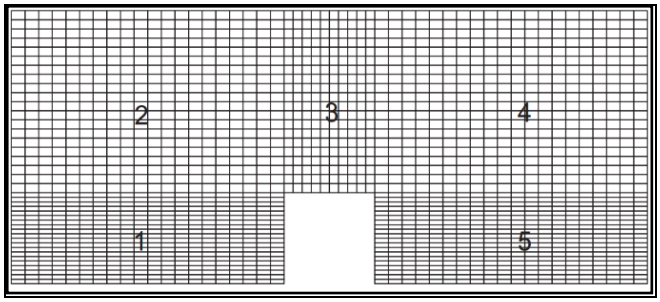
(γ)

Εικόνα 13: (α) Block-structured πλέγμα σε σύμμορφες διεπαφές, **(β)** Block-structured πλέγμα με μη σύμμορφες διεπαφές, **(γ)** Σύνθετο πλέγμα.

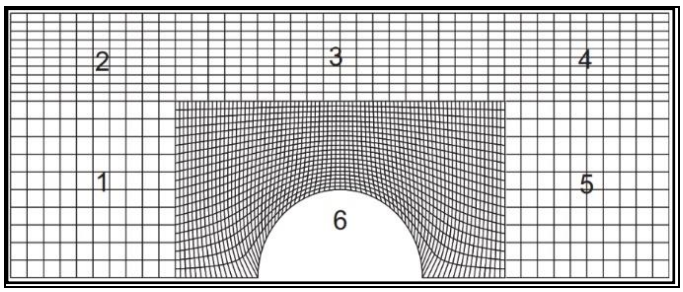
Τα δομημένα με μπλοκ πλέγματα με επικαλυπτόμενα μπλοκ ονομάζονται μερικές φορές σύνθετα ή χίμαιρα πλέγματα (composite ή Chimera grids), φαίνεται στην **Εικόνα 13γ**. Στην περιοχή επικάλυψης, οι οριακές συνθήκες για ένα μπλοκ λαμβάνονται με παρεμβολή του διαλύματος από το άλλο (επικαλυπτόμενο) μπλοκ. Το μειονέκτημα αυτών των πλεγμάτων είναι ότι η διατήρηση δεν επιβάλλεται εύκολα στα όρια των μπλοκ. Και τα πλεονεκτήματα είναι ότι οι σύνθετοι τομείς αντιμετωπίζονται ευκολότερα και μπορούν να χρησιμοποιηθούν για να ακολουθήσουν κινούμενα σώματα: ένα μπλοκ συνδέεται με το σώμα και κινείται μαζί του, ενώ ένα στάσιμο πλέγμα καλύπτει το περιβάλλον.

- **Πολλαπλό Πλέγμα (Multiblock Mesh)**

Μέσα στο δομημένο πλέγμα μπορούμε να χρησιμοποιούμε και τη μέθοδο πολυ-μπλοκ για να δημιουργήσουμε μια σύνθετη γεωμετρία. Θεωρούμε ότι στον πυρήνα του κυλίνδρου έχουμε δημιουργήσει ένα άλλο μικρότερο πλέγμα ή διαφορετικού μοτίβου σε σχέση με την εξωτερική περιοχή, για να τα διαχωρίσουμε τοποθετούμε περισσότερα από ένα πλαίσια οριοθέτησης (bounding box). Μέσω αυτής της προσέγγισης, το πλέγμα του εν λόγω τομέα συναρμολογείται τώρα από έναν αριθμό δομημένων μπλοκ που συνδέονται μεταξύ τους. Τα συνημμένα κάθε όψης των παρακείμενων μπλοκ μπορεί να είναι κανονικά (δηλαδή να έχουν συμφωνίες των κυτταρικών όψεων) **Εικόνα 14α** ή αυθαίρετα (δηλαδή να έχουν μη συμφωνίες των κυτταρικών διεπαφών) όπως φαίνεται στην **Εικόνα 14β**. Η δημιουργία πλεγμάτων ειδικά με διεπαφές κυττάρων που δεν ταιριάζουν είναι σίγουρα πολύ απλούστερη από τη δημιουργία ενός πλέγματος ενός μπλοκ προσαρμοσμένου σε ολόκληρο τον τομέα και για να παρακάμψει την αύξηση του μη ορθογωνικού πλέγματος ή της ασυμμετρίας των κυψελών πλέγματος. Μια τέτοια προσέγγιση προσφέρει μεγάλη ευελιξία στον προσδιορισμό της καλύτερης τοπολογίας πλέγματος για κάθε ένα από τα υποδιαιρεμένα μπλοκ.



(α)

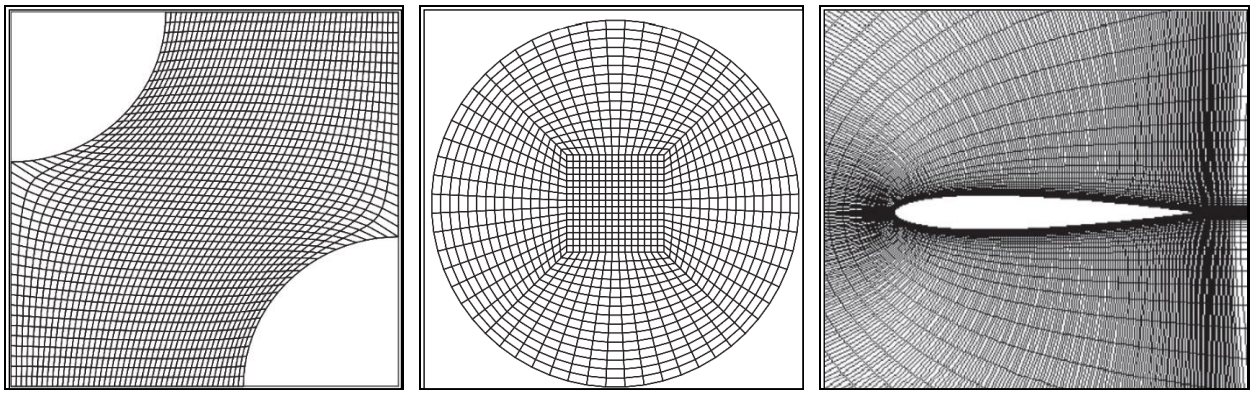


(β)

Εικόνα 11: (α) Multiblock πλέγμα με συμφωνίες των κυτταρικών όψεων, **(β)** Multiblock πλέγμα με μη συμφωνίες των κυτταρικών διεπαφών.

Αντίστοιχα και σε μια σύνθετη γεωμετρία, δεν γίνεται να την καλύψουμε μόνο με ένα πλαίσιο. Έτσι, δημιουργούμε διαφορετικό πλαίσιο με κάθε διαφορετικά σχήματα μέσα στο περιβάλλον του CFD και ακολουθώντας συνδυάζουμε αυτά τα μπλοκ και εμφανίζεται ένα ενιαίο δομημένο πλέγμα. Σημειώνοντας, ότι υπάρχουν πολλά μπλοκ που συμμετέχουν στη διαδικασία δημιουργίας πλέγματος αυτό καλείται δομημένο πλέγμα πολυ-μπλοκ.

Ο προγραμματιστής μπορεί να διαλέξει την κατάλληλη τοπολογία πλέγματος με βάση το πόσο «γεμίζει» το κάθε μπλοκ. Τα δομημένα πλέγματα υπάρχουν και σε Η, Ο ή C τύπους. Η **Εικόνα 15α** δείχνει ένα πλέγμα Η – τύπου το οποίο, όταν χαρτογραφείται σε ένα ορθογώνιο, έχει ξεχωριστά ανατολικά, δυτικά, βόρεια και νότια όρια. Σε ένα Ο – τύπου πλέγμα, τα δύο αντίθετα όρια συνδέονται μεταξύ τους (π.χ. ανατολικά προς δυτικά ή νότια προς βορρά). Ένα παράδειγμα είναι ένα πλέγμα γύρω από έναν κυκλικό κύλινδρο (**Εικόνα 15β**), εάν το τοίχωμα του κυλίνδρου είναι το νότιο όριο και το εξωτερικό άκρο είναι το βόρειο όριο, τότε τα δυτικά και ανατολικά όρια συγχωνεύονται για να δημιουργήσουν ατελείωτες γραμμές πλέγματος γύρω από τον κύλινδρο (ή όχι κυκλικές). Η μέτρηση για τον δείκτη i ξεκινά από μια αυθαίρετη ακτινική γραμμή που αντιπροσωπεύει τη διεπαφή μεταξύ των δυτικών και των ανατολικών ορίων. Όσον αφορά το Ο πλέγμα, η διεπαφή μεταξύ δύο τμημάτων του νότιου ορίου που βρίσκονται σε επαφή μπορεί να είναι είτε σύμμορφη είτε μη σύμμορφη. Στον C – τύπου πλέγματος, ένα όριο πέφτει εν μέρει στον εαυτό του, ένα σύνολο γραμμών πλέγματος τυλίγεται γύρω από την αεροτομή το σώμα, ενώ το άλλο κομμάτι είναι (σχεδόν) ορθογώνιο σε αυτό όπως φαίνεται στην **Εικόνα 15γ**, εάν το τοίχωμα αεροτομής είναι το νότιο όριο, από ό, τι εκτείνεται από το πίσω άκρο μέχρι κάποια απόσταση πίσω από το σώμα: η έξοδος είναι το δυτικό όριο κάτω και το ανατολικό όριο πάνω από τη διπλή νότια γραμμή που εκτείνεται από το πίσω άκρο, ενώ το βόρειο όριο καλύπτει την κάτω, αριστερή και πάνω πλευρά του τομέα λύσης (και έχει χρησιμοποιείται για υπολογισμούς DNS).



(α) **(β)** **(γ)**

Εικόνα 15: (α) Δομημένο πλέγμα Η-τύπου, για τον υπολογισμό της ροής στο ασύμμετρο τμήμα μιας κλιμακωτής τράπεζας ενός σωλήνα, **(β)** Δομημένο πλέγμα Ο-τύπου, σε κυκλικό κύλινδρο, **(γ)** Δομημένο πλέγμα C-τύπου, γύρω από μια αεροτομή.

Προσδίδει στη δημιουργία τετραγώνων, όπου είναι και η πιο απλή μορφή πλέγματος, δυστυχώς δεν συλλαμβάνει ακριβώς το σχήμα. Τα αποτελέσματα που δίνει δεν είναι πολύ ακριβή και χρησιμοποιείται για απλές γεωμετρίες και γι' αυτό χρησιμοποιείται σπάνια και δείχνει ότι βασίζονται σε Καρτεσιανά συστήματα συντεταγμένων που έχουν περιορισμούς στις ακανόνιστες γεωμετρίες.

Στην παρούσα διατριβή, η επίλυση του προβλήματος πραγματοποιήθηκε με τη χρήση δομημένου πλέγματος και γι' αυτό είναι εφικτό να αναφερθούν μερικά από τα πλεονεκτήματα του Δομημένου Πλέγματος:

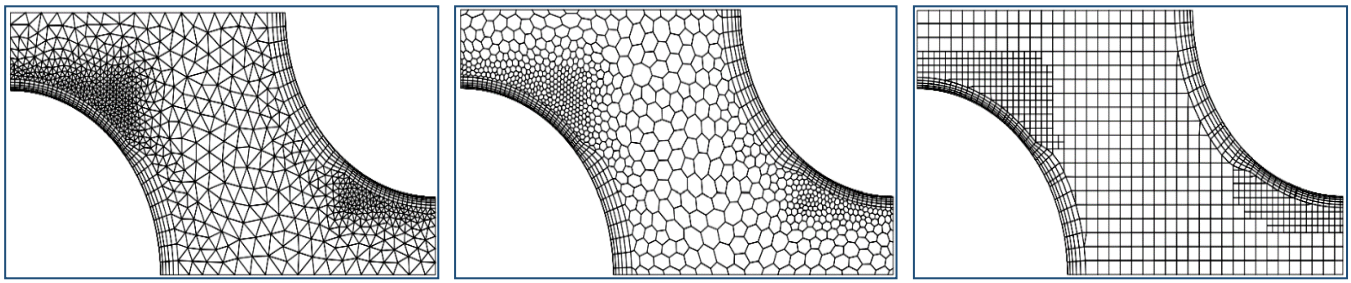
- Υψηλός βαθμός ελέγχου και ποιότητας: ο υψηλότερος βαθμός ελέγχου σημαίνει ότι η παραγωγή του ακριβώς επιθυμητού πλέγματος κατορθώνεται. Ο χρήστης έχει καλύτερο έλεγχο για τα μεγέθη και τις θέσεις των εσωτερικών κόμβων και για τη τοποθέτηση εσωτερικών κόμβων που είναι άμεσα συνδεδεμένη με τους εξωτερικούς κόμβους.
- Καλύτερη ευθυγράμμιση / σύγκλιση: γενικά, τα δομημένα πλέγματα ευθυγραμμίζονται στην κατεύθυνση ροής, έχοντας έτσι πιο ακριβή αποτελέσματα και καλύτερη σύγκλιση σε λύτες CFD. Η ευθυγράμμιση επιτυγχάνεται σχεδόν ακριβώς, διότι οι γραμμές πλέγματος και η ροή ακολουθούν τα περιγράμματα της γεωμετρίας.
- Λιγότερη απαίτηση μνήμης και χρόνου: δεν έχει ανάγκη αποθήκευσης οποιουδήποτε πίνακα συνδεσιμότητας, καθώς το πλέγμα ορίζεται σύμφωνα με ένα καθορισμένο μοτίβο. Ως αποτέλεσμα, ο χρόνος υπολογισμού CFD μειώνεται επίσης με δομημένα πλέγματα λόγω μειωμένου αριθμού κελιών.
- Η τοποθέτηση δεδομένων: αυτό σχετίζεται με τη διάταξη της μνήμης. Ο περιορισμός του εύρους ζώνης μνήμης σε μαζικά παράλληλες αρχιτεκτονικές GPU στο CFD σε αντίθεση με την υπολογιστική ισχύ. Από τον τρόπο σχεδιασμού τα δεδομένα για στοιχεία που είναι κοντά γεωμετρικά είναι επίσης κοντά στη μνήμη. Ως συνέπεια, λιγότερες απώλειες της κρυφής μνήμης άρα και καλύτερη χρήση του εύρους ζώνης μνήμης.
- Διαθέσιμοι αλγόριθμοι λύσεων: επιτρέπουν τη χρήση αλγορίθμων λύσεων που ας πούμε ότι δεν μπορούν να εφαρμοστούν σε μη δομημένα δεδομένα.
- Οριστικοί κανονικοί: η εφαρμογή των οριακών συνθηκών και των μοντέλων αναταράξεων λειτουργεί καλά όταν υπάρχει μια καλά καθορισμένη υπολογιστική κατεύθυνση κανονική σε ένα χαρακτηριστικό όπως ένας τοίχος. Οι εγκάρσιοι κανονικοί ορίζονται εύκολα σε ένα δομημένο πλέγμα.

Όπως έχει επισημανθεί από τα αναφερόμενα, η αποτελεσματικότητα των δομημένων πλεγμάτων τους καθιστά μια σταθερή επιλογή για έρευνα.

2. Μη Δομημένο – Αδόμητο Πλέγμα (Unstructured Mesh)

Το Μη Δομημένο είναι το ακριβώς αντίθετο είδος πλέγματος από το Δομημένο, όπου τα στοιχεία είναι διατεταγμένα με τυχαίο τρόπο ή τυχαία ονομάζεται αδόμητο/μη δομημένο πλέγμα. Το σύνολο των διακριτών κελιών που δημιουργείται μέσα στο χώρο CFD είναι τυχαία διαχωρισμένα και δεν ακολουθούν κανένα ενιαίο μοτίβο. Θα δούμε και τους αντίστοιχους τύπους αυτού του πλέγματος στη συνέχεια. Αυτό, για υπολογιστικές επιλύσεις, καθιστά τη χρήση μονάδων επεξεργασίας γραφικών (GPU) λιγότερο αποτελεσματική από ό, τι στην περίπτωση των δομημένων πλεγμάτων (το ίδιο ισχύει και με την χρήση διανυσματικών επεξεργαστών).

Οι τύποι μη δομημένου πλέγματος φαίνονται στα παρακάτω **Εικόνα 16**:



(α) **(β)** **(γ)**

Εικόνα 126: (α) Μη δομημένο πλέγμα τετραεδρικού τύπου, **(β)** Μη δομημένο πλέγμα πολυεδρικού τύπου, **(γ)** Μη δομημένο πλέγμα εξαεδρικού τύπου.

Βασικός λόγος χρήσης πλεγμάτων σε υπολογιστικές μεθόδους είναι η επίλυση των φυσικομαθηματικών θεωρημάτων μέσα σε κάποιες κυψέλες ή κόμβους. Οι θεωρητικοί μέθοδοι που χρησιμοποιούνται για κάθε μελέτη CFD (FDM, FEM, και FVM) επιλύουν πραγματικά τη μεταβλητή σε αυτά τα διακριτά κελιά/κόμβους. Η ποιότητα της συνολικής ανάλυσης του πλέγματος προέρχεται ξεκάθαρα από την ανάλογη επιλογή του τύπου πλέγματος. Το ποσοστό σύγκλισης αυξάνεται από την καλή ποιότητα του πλέγματος. Στα πλούσια στοιχεία, αυξάνεται τοπικά η αριθμητική σειρά, προωθείται η σταθερότητα και η σύγκλιση της λύσης. Επιπλέον, η καλύτερη ποιότητα πλέγματος προσφέρει μια ακριβέστερη λύση. Μια καλή παρατήρηση που μπορεί να τεθεί είναι η ικανότητα βελτίωσης του πλέγματος πάνω σε κάποιες συγκεκριμένες περίπλοκες περιοχές της γεωμετρίας, προσδίδει καλύτερη πιστότητα λύσης πάνω στην περιοχή. Συμπληρώνοντας μια παρατήρηση εδώ, αφού αποφασιστεί η ποιότητα του πλέγματος, πρέπει να συμπεριλάβουμε και το χρόνο της CPU. Πάνω σε ένα ιδιαίτερο πλέγμα, όπου ο αριθμός των κελίων ανά μονάδα επιφάνειας είναι μεγάλος, συνίσταται αντίστοιχα να είναι και ο χρόνος της CPU μεγάλος.

2.1.5 Φυσική και Ιδιότητες Ροής

Οι απαιτήσεις των βιομηχανιών για λύσεις διαδικασιών ροής είναι περίπλοκες. Με λίγα λόγια ο συνυπολογισμός των φυσικών ιδιοτήτων της ροής είναι σημαντική προϋπόθεση για την σωστή μοντελοποίηση του τύπου ροής. Διαπιστώνεται παράλληλα από αυτό ότι δημιουργούνται πολλές διαταραχές που επηρεάζουν το ρευστό μέσα στη ροή, για παράδειγμα η μεταφορά θερμότητας επηρεάζει την συνολική δυναμική του ρευστού.

Επομένως, ο χρήστης πρέπει αρχικά να ορίσει την φύση του ρευστού, ώστε οι προσομοιώσεις να ανταποκρίνονται στις σωστές λύσεις. Η πρώτη κίνηση είναι να έχει προσδιορίσει το αν είναι ασυμπίεστη ή συμπιεστή η ροή. Μετά, προσδιορίζει την κατηγορία των ρευστών (αμελητέο ή όχι το ιξώδες), ρυθμίζει μάλιστα το είδος της ροής (στρωτή ή τυρβώδης) και τη τακτοποίηση των εσωτερικών και εξωτερικών ροών. Αυτά τα βήματα βοηθάνε στη λύση του προβλήματος.

2.1.6 Οριακές Συνθήκες

Οι οριακές συνθήκες προβλέπονται για τις διάφορες συμπεριφορές που προκαλεί η ροή του ρευστού. Με το έκτο βήμα της προεπεξεργασίας, ο προγραμματιστής CFD καλείται να καθορίσει τις απαραίτητες συνθήκες που αντανακλούν στην πραγματική αναπαράσταση του φαινομένου. Εφόσον υπάρχουν όρια εισροής – εκροής της ροής, εγκαθίστανται οι οριακές συνθήκες για τη προσαρμογή της συμπεριφοράς του ρευστού. Μπορεί κάλλιστα, να γίνει οριοθέτηση της γεωμετρίας της ροής και των γύρω τοιχωμάτων από τα πιθανά εσωτερικά εμπόδια μέσα σε εξωτερικούς σταθερούς στερεούς τοίχους.

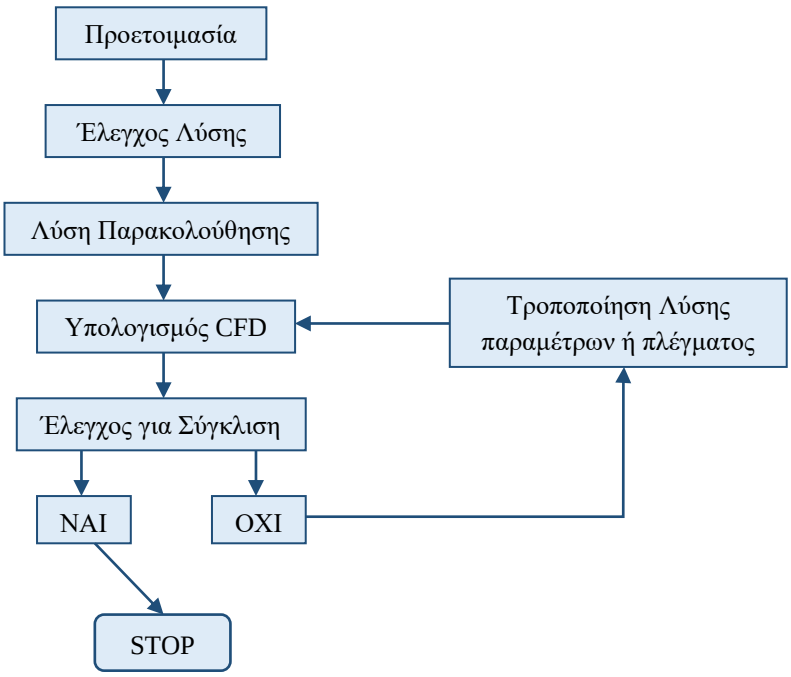
Η πλειοψηφία των εμπορικών κωδικών προβάλλουν αξιώσεις και διαθέτουν την ικανότητα επίλυσης όλων των καταστάσεων ροής ρευστού (υποηχητικών, διηχητικών και/ή υπερηχητικών). Συνήθως, αποδίδουν αποτελεσματικότερα σε υποηχητικές ταχύτητες, για τον λόγο ότι οι οριακές συνθήκες εκεί είναι λιγότερο απαιτητικές και καλύπτουν ευκολότερα το φάσμα της ροής.

2.2 2ο Στάδιο : Επίλυση – Solver

2.2.1 Έλεγχος Λύσης

Η διακριτοποίηση αποδίδει ένα μεγάλο σύστημα μη γραμμικών αλγεβρικών εξισώσεων. Η μέθοδος λύσης εξαρτάται από το πρόβλημα. Για ασταθείς ροές, χρησιμοποιούνται μέθοδοι που βασίζονται σε εκείνες που χρησιμοποιούνται για προβλήματα αρχικής τιμής για συνήθεις διαφορικές εξισώσεις (πορεία στο χρόνο). Σε κάθε βήμα πρέπει να λυθεί ένα ελλειπτικό πρόβλημα. Τα προβλήματα σταθερής ροής συνήθως επιλύονται με ψευδο-χρονική πορεία ή ισοδύναμο σχήμα επανάληψης. Επειδή οι εξισώσεις είναι μη γραμμικές, χρησιμοποιείται ένα σχήμα επανάληψης για την επίλυσή τους. Αυτές οι μέθοδοι χρησιμοποιούν διαδοχική γραμμικοποίηση των εξισώσεων και τα προκύπτοντα γραμμικά συστήματα επιλύονται σχεδόν πάντα με επαναληπτικές τεχνικές.

Η επιλογή του λύτη εξαρτάται από τον τύπο πλέγματος και τον αριθμό των κόμβων που εμπλέκονται σε κάθε αλγεβρική εξίσωση. Για την καλύτερη κατανόηση των αναφερόμενων, μπορείτε να δείτε το αναλυτικό **Σχήμα 2**.



Σχήμα 2: Επισκόπηση της λύσης.

Η διαδικασία επανάληψης εμπεριέχει γενικά όλες τις διακριτές τιμές των ιδιοτήτων ροής όπως τη ταχύτητα, τη θερμοκρασία, τη πίεση και τις άλλες παραμέτρους μεταφοράς που αρχικοποιούνται πριν από τον υπολογισμό της λύσης. Θεωρητικά, οι αρχικές συνθήκες μπορεί να είναι καθαρά αυθαίρετες. Πρακτικά, υπάρχουν πολλά πλεονεκτήματα για την έξυπνη επιβολή αρχικών συνθηκών. Οι σωστές αρχικές συνθήκες έχουν μεγάλη σημασία για την διαδικασία επανάληψης. Πρώτον, αν προσεγγίζουν πολύ στην τελική λύση της σταθερής κατάστασης, θα συγκλίνει πιο γρήγορα η επαναληπτική διαδικασία άρα θα έχουμε μικρότερο χρόνο υπολογισμού. Δεύτερον, αν απέχουν από την πραγματικότητα, οι υπολογισμοί θα εκτελούν περισσότερους υπολογισμούς για την επίτευξη της

επιθυμητής σύγκλισης. Μάλιστα, οι ακατάλληλες αρχικές συνθήκες μπορεί να οδηγήσουν στην απόκλιση της επαναληπτικής διαδικασίας.

2.2.2 Κριτήρια Σύγκλισης

Τέλος, πρέπει να καθοριστούν τα κριτήρια σύγκλισης για την επαναληπτική μέθοδο. Περιλαμβάνει τις λειτουργίες: λύση παρακολούθησης, υπολογισμός CFD και έλεγχος σύγκλισης. Συνήθως, υπάρχουν δύο επίπεδα επαναλήψεων: εσωτερικές επαναλήψεις, εντός των οποίων επιλύονται οι γραμμικές εξισώσεις, και εξωτερικές επαναλήψεις, που ασχολούνται με τη μη γραμμικότητα και τη σύζευξη των εξισώσεων.

Η απόφαση για το πότε θα σταματήσει η επαναληπτική διαδικασία σε κάθε επίπεδο είναι σημαντική, από πλευρά ακρίβειας και αποτελεσματικότητας.

Λαμβάνεται υπόψη επίσης, ότι οι πτυχές όπως η σύγκλιση, τα κριτήρια σύγκλισης ή οι τιμές ανοχής, τα υπολείμματα, η σταθερότητα, τα σφάλματα, οι παράγοντες υποβάθμισης και η ανεξαρτησία του δικτύου στηρίζουν σε μεγάλο βαθμό τις πολλές αριθμητικές επιλύσεις για την προσομοίωση ενός προβλήματος CFD.

2.3 3ο Στάδιο : Μεταεπεξεργασία – Postprocess

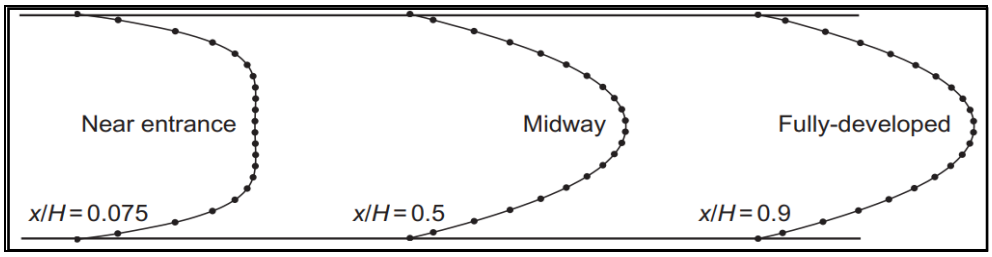
2.3.1 Αναφορά και Οπτικοποίηση Αποτελεσμάτων

Είναι γνωστό πως το CFD έχει την δυνατότητα να αναπαριστά ζωντανές γραφικές εικόνες των αποτελεσμάτων. Απεικονίζονται τα δεδομένα και τα φυσικά χαρακτηριστικά του προβλήματος που έχει προγραμματιστεί να επιλύει, μέσω των κωδικών που περιέχει. Ταξινομούνται σε διάφορες πλατφόρμε.

A. X –Y Γραφήματα

Αυτά τα γραφήματα είναι συνήθως δισδιάστατα γραφήματα που αντιπροσωπεύουν τη διακύμανση μιας εξαρτώμενης μεταβλητής μεταφοράς έναντι μιας άλλης ανεξάρτητης μεταβλητής. Οι X-Y γραφικές παραστάσεις είναι ο πιο ακριβής και ποσοτικός τρόπος παρουσίασης των αριθμητικών δεδομένων. Επομένως, είναι ένας τρόπος άμεσης σύγκρισης των δεδομένων με τις πειραματικές τιμές και είναι πολύ ευανάγνωστα. Επιπροσθέτως, οι λογαριθμικές κλίμακες επιτρέπουν τον εντοπισμό σημαντικών επιδράσεων ροής που συμβαίνουν ειδικά κοντά στα στερεά όρια. Αυτά τα γραφήματα χρησιμοποιούνται ευρέως για την παρουσίαση προφίλ γραμμής ταχύτητας και για πεδία επιφανειακών ποσοτήτων, όπως πίεση και συντελεστής επιφανειακής τριβής. Ένα παράδειγμα, ενός παραβολικού χαρακτήρα προφίλ είναι η ρεαλιστική κίνηση της ροής ρευστού που ρέει μέσα σε ένα παράλληλο κανάλι πλάκας. Εναλλακτικά, μέσω της χρήσης διαδοχικών δισδιάστατων γραφικών προφίλ όπως φαίνεται στην **Εικόνα 17**. Η κατανομή ροής μεταβάλλεται σταδιακά από ένα ομοιόμορφο προφίλ που

καθορίζεται στο όριο εισόδου (αριστερά) σε ένα παραβολικό προφίλ καθώς ταξιδεύει κατάντη προς το όριο εξόδου καναλιού (δεξιά).



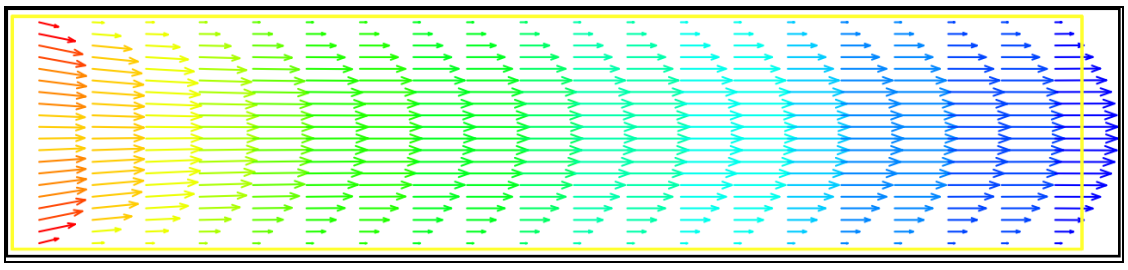
Εικόνα 17: Χ - Υ πλάνο παραβολικού προφίλ ταχύτητας (στρωτή) στην πλήρως αναπτυγμένη περιοχή.

Β. Διανυσματικά Γραφήματα

Το διανυσματικό γράφημα περιέχει τα μέσα με τα οποία εμφανίζεται στα διακριτά σημεία, μια διανυσματική ποσότητα (συνήθως η ταχύτητα) με βέλος (ο προσανατολισμός της κατεύθυνσης) του αντίστοιχου μεγέθους. Γενικά, παρουσιάζει τη προοπτική του πεδίου ροής σε δύο διαστάσεις.

Στο τρισδιάστατο πεδίο ροής, παρουσιάζεται σε διαφορετικά κομμάτια δισδιάστατων επιπέδων που περιέχουν τις διανυσματικές ποσότητες που δημιουργούν τους διαφορετικούς προσανατολισμούς για να ελεγχθούν καλύτερα τα φυσικά φαινόμενα. Εάν η πυκνότητα των πλεγμάτων είναι πιο υψηλή, ο χρήστης είναι σε θέση είτε να παρεμβάλλει είτε να μειώσει τον αριθμό των θέσεων εξόδου για να αποτρέψει την ομαδοποίηση αυτών των βελών "εξαλείφοντας" τη γραφική παράσταση.

Για να γίνουν κατανοητά τα προηγούμενα μπορούμε να θέσουμε το προηγούμενο απλό παράδειγμα της **Εικόνας 17**. Η **Εικόνα 18**, δείχνει μια τυπική γραφική παράσταση διανύσματος ταχύτητας που αντιπροσωπεύει τη ροή ρευστού κατά μήκος του καναλιού παράλληλης πλάκας. Η εναλλακτική οπτική της αναπτυσσόμενης ροής που προβλεπόταν στην **Εικόνα 17**. Τα διαφορετικά χρώματα που αντιστοιχούν στο διανυσματικό γράφημα απεικονίζουν τη σύνθετη της ταχύτητας διανύσματα με μια άλλη εξαρτημένη μεταβλητή. Το εύρος των χρωμάτων που δείχνει το γράφημα απεικονίζει την κατανομή μεταξύ υψηλών (κόκκινων) και χαμηλών (μπλε) πιέσεων που είναι αποτέλεσμα μέσα στη ροή ρευστού. Έτσι, ο χρήστης επιλέγει ελεύθερα τις άλλες μεταβλητές μεταφοράς που τονίζουν καλύτερα σημαντικές φυσικές πτυχές των φαινομένων ροής. Η μεταφορά θερμότητας, παράδειγμα, όπου μπορεί να είναι σημαντική, οι φορείς της ταχύτητας μπορούν να συνδυαστούν με την κατανομή θερμοκρασίας για να απεικονίσουν τη μεταφορά θερμού ρευστού εντός του τομέα ροής.

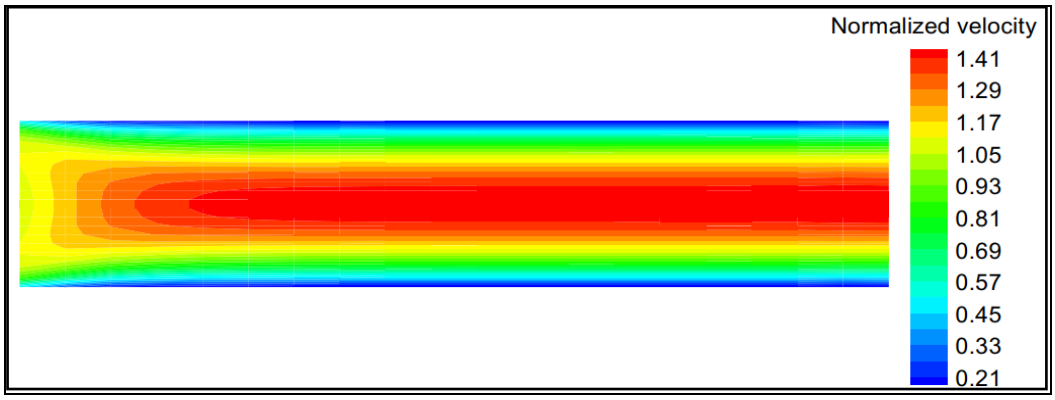


Εικόνα 18: Διανύσματα ταχύτητας που δείχνουν την ανάπτυξη ροής κατά μήκος του καναλιού της παράλληλης πλάκας.

C. Γράφημα Περιγράμματος

Η δημιουργία περιγράμματος διαπιστώνει άλλη μια χρήσιμη τεχνική προβολής αποτελεσμάτων CFD. Η γραμμή περιγράμματος (γνωστή ως isoline) μπορεί να περιγραφεί ως μια γραμμή ενδεικτική κάποιας ιδιότητας που είναι σταθερή στο διάστημα. Η ισοδύναμη αναπαράσταση στις τρεις διαστάσεις είναι μια ισοεπιφάνεια – isosurface. Σε αντίθεση με τα X–Y γραφήματα, τα περιγράμματα όπως τα διανυσματικά παρέχουν μια παγκόσμια περιγραφή της ροής ρευστού που είναι ενθυλακωμένη σε μία προβολή. Γενικά, τα περιγράμματα σχεδιάζονται έτσι ώστε η διαφορά μεταξύ της αριθμητικής τιμής της εξαρτημένης μεταβλητής μεταφοράς από μία γραμμή περιγράμματος σε μια γειτονική γραμμή περιγράμματος να διατηρείται σταθερή. Η χρήση γραφημάτων περιγράμματος συνήθως δεν στοχεύει στην ακριβή αξιολόγηση των αριθμητικών τιμών μεταξύ των γραμμών περιγράμματος. Στην πράξη, τα περιγράμματα είναι συνήθως γραμμικά κλιμακωτά. Ωστόσο, για να καταγραφούν καλύτερα οι μη ορατές λεπτομέρειες σε ορισμένες περιοχές εντός του πεδίου ροής, ο χρήστης μπορεί να χρησιμοποιήσει αδρανώς άλλους τύπους επιλογών κλιμάκωσης για να αποκαλύψει αυτές τις απομονωμένες συμπεριφορές ροής.

Από το προηγούμενο παράδειγμα (**Εικόνας 17**) στην περίπτωση αυτή η σταθερή ιδιότητα πεδίου ροής οποιασδήποτε μεταβλητής μεταφοράς υποδηλώνεται από τη σταθερή ένταση της σκίασης χρώματος. Ο χρωματικός χάρτης (κλίμακας "ουράνιου τόξου") χρησιμοποιείται για να απεικονίσει τις κατανομές της αδιάστατης ταχύτητας που προκύπτει, σε σχέση με την ταχύτητα εισόδου εντός του τομέα ροής. Στην **Εικόνα 19**, τα μεταβαλλόμενα χρωματισμένα περιγράμματα κοντά στην είσοδο (αριστερό μέρος) επιβεβαιώνουν την εξέλιξη της ροής ρευστού όπως παρατηρήθηκε προηγουμένως από τα διαδοχικά προφίλ ταχύτητας στο διάγραμμα X–Y στην **Εικόνα 17** και η γραφική παράσταση του διανύσματος ταχύτητας της **Εικόνας 18**. Αντιθέτως, δεν έγινε αντιληπτή η μεταβολή της ταχύτητας κοντά στην έξοδο (δεξί μέρος).



Εικόνα 19: Πλήρες περίγραμμα σε χρωματιστό χάρτη για την κατανομή της ταχύτητας.

3 ΚΕΦΑΛΑΙΟ 3ο : ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Με τον όρο Παράλληλος Προγραμματισμός ορίζουμε την ταυτόχρονη χρήση διαφόρων υπολογιστικών στοιχείων με στόχο την επίλυση μιας εφαρμογής. Η δυσκολία μιας εφαρμογής είναι ότι περιέχει σχετικά πολλά δεδομένα. Με την βοήθεια ενός πολυεπεξεργαστής ή πολυυπολογιστή, οι επιδόσεις μεγιστοποιούνται και ο χρόνος υλοποίησης ελαχιστοποιείται. Ουσιαστικά παραλληλοποιείται ο σειριακός αλγόριθμος. Η παράλληλη επεξεργασία είναι μια πρόσφατη μέθοδος στο χώρο της υπολογιστικής επιστήμης και στόχος της είναι η βελτιστοποίησης της ταχύτητας εκτέλεσης χωρίς την αλλαγή του υλικού. Για να προκύψει κάτι τέτοιο χρειάζονται οι επεξεργαστές να είναι φυσικά εγκατεστημένοι πολύ στενά μεταξύ τους, ώστε η επικοινωνία τους να είναι γρήγορη και έτσι οι προγραμματιστές να εκμεταλλευτούν τα ανάλογα παράλληλα προγραμματιστικά εργαλεία.

Ο παράλληλος προγραμματισμός αποτελείται από τρεις τεχνικές, την διασωλήνωση (pipelinig), τους υπερβαθμωτούς επεξεργαστές (superscalar processors) και τους επεξεργαστές ανυσμάτων (vector processors), οι οποίες τεχνικές αυξάνουν τον αριθμό των μικροεντολών ανά δευτερόλεπτο (MIPS) από έναν CPU (κεντρική μονάδα επεξεργασίας, επεξεργαστής), σε συνδυασμό με έναν μεταγλωττιστή η εικονική μηχανή (virtual machine) παραμένει ακολουθιακή. Από τα προλεγόμενα προέκυψε η βελτιστοποίηση της απόδοσης των δυνατοτήτων των ακολουθιακών προγραμμάτων. Μπορούμε να τονίσουμε ότι στους πολυεπεξεργαστές πραγματοποιείται μόνο ένα λειτουργικό σύστημα, όπου εκτελεί τις παράλληλες διεργασίες σε πολλαπλούς επεξεργαστές, αντίστοιχα στους πολυυπολογιστές κάθε κόμβος εκτελεί τον δικό του πυρήνα, αφού αποτελεί κατανεμημένα συστήματα και εφόσον αποτελεί το κατάλληλο λογισμικό.

Τα παράλληλα συστήματα διαχωρίζονται στα ομοιογενή και στα ανομοιογενή. Τα ομοιογενή συστήματα περιλαμβάνουν όμοια αρχιτεκτονική με τους επεξεργαστές σε αντίθεση με τα ανομοιογενή που έχουν ετερόκλητους επεξεργαστές. Μια άλλη οπτική διαχωρισμού είναι από το μηχανισμό επικοινωνίας δηλαδή με τη χρήση κοινής ή κατανεμημένης μνήμης (shared or distributed memory systems). Αυτοί οι μηχανισμοί επικοινωνίας έχουν ιδιαιτερότητες στην επίδοση των επεξεργαστών, γι’ αυτό θα αναπτυχθούν στα επόμενα κεφάλαια.

3.1 Υπολογιστική Ρευστομηχανική και Υπολογιστής

Η εξέλιξη των ηλεκτρονικών υπολογιστών, τα τελευταία πενήντα χρόνια, έχει μεγάλη συνεισφορά στους διάφορους επιστημονικούς τομείς. Στην μηχανική των ρευστών και στην αεροδυναμική πρόσφερε σημαντικά αγαθά. Μέχρι και τη δεκαετία του 1960 τα αποτελέσματα μιας έρευνας γινόταν σε πραγματικές προσομοιώσεις με αεροσήραγγα ή με επίλυση απλοποιημένων εξισώσεων. Η λύση ενός προβλήματος μέσα από τη χρήση ενός υπολογιστή αποκτά αξιοπιστία και δεν είναι αναγκαία η επιβεβαίωση με την πειραματική μέθοδο πλέον. Η υπολογιστική δύναμη που κατέχει, καταφέρνει και εκτελεί αριθμητικές επιλύσεις. Για παράδειγμα, η τυρβώδης ροή γύρω από μια αεροτομή σε μεγάλη γωνία πρόσπτωσης, με εμφάνιση καθολικής αποκόλλησης της ροής. Επίσης, μπορούμε να αναλύσουμε τρισδιάστατα πεδία ροής, όπως της δυναμικής ροής γύρω από ένα αεροσκάφος και της τρισδιάστατης ροής που αγκαλιάζει ένα αυτοκίνητο. Με την αξιοπιστία που έχει ο ηλεκτρονικός υπολογιστής μπορεί να επιλύει τα πεδία ροής με πολύ μεγαλύτερες δυνατότητες σε σχέση με τη συμβατή προσομοίωση, για τον λόγο ότι το πεδίο ροής στην αεροσήραγγα δεν μπορεί να είναι πλήρης.

Υπάρχουν φαινόμενα που αλληλεπίδρασης ροής με τα τοιχώματα της αεροσήραγγας, ή και δυσκολία συγχρονισμού διατύπωσης της ροής με τους αριθμούς Mach και Reynolds. Ο υπολογιστής εδώ πέρα είναι σίγουρο ότι δεν αντιμετωπίζει τέτοια φαινόμενα. Ο μόνος περιορισμός που εμφανίζει, είναι η ταχύτητα εκτέλεσης των υπολογισμών και η διαθεσιμότητα της μνήμης. Με το πέρας των χρόνων αυτό φαίνεται να έχει αντιμετωπιστεί με την εξέλιξη των υπολογιστικών εξαρτημάτων, όπως οι κάρτες γραφικών.

3.2 Αρχιτεκτονική Υπολογιστή

Ο ηλεκτρονικός υπολογιστής διαθέτει το υλικό κομμάτι (υλισμικό – hardware), που περιλαμβάνει τα μηχανικά μέρη, τον δίσκο, την οθόνη, τα ολοκληρωμένα κυκλώματα κα. και από τα προγράμματα ελέγχου της λειτουργίας (λογισμικό – software), περιλαμβάνοντας το λειτουργικό σύστημα, την γλώσσα προγραμματισμού κα.

Τα βασικά μέρη του υπολογιστή είναι τα εξής:

1. **Μονάδα Εισόδου**: είναι η άμεση οδός επικοινωνίας του χρήστη με τη μηχανή.
2. **Μονάδα Μνήμης**: η μνήμη της μηχανής, όπου αποθηκεύονται όλες οι πληροφορίες από τη μονάδα εισόδου, ως πρόγραμμα ή ως δεδομένα. Επιπρόσθετα, αποθηκεύονται κατά τη διάρκεια εκτέλεσης του προβλήματος οι πληροφορίες.
3. **Μονάδα Εξόδου**: η επικοινωνία μεταξύ υπολογιστή και περιφερειακών εξαρτημάτων (πχ εκτυπωτής ή και εμφάνιση αποτελεσμάτων πάνω στην οθόνη).
4. **Κεντρική Υπολογιστική Μονάδα**: είναι η καρδία της μηχανής, επικοινωνεί με την μνήμη και η μνήμη με τις μονάδες εισόδου και εξόδου, ώστε να εκτελεστεί η διεργασία.
5. **Μεταγλωττιστής**: πολλές φορές τα προγράμματα αριθμητικής επίλυσης των εφαρμογών είναι γραμμένα σε γλώσσα του ίδιου του προγραμματιστή και όχι από τον υπολογιστή. Έτσι η μονάδα εισόδου και η μνήμη διαθέτουν μια μονάδα που λέγεται μεταγλωττιστής (Compiler), όπου μεταγλωττίζει τα προγράμματα του προγραμματιστή στη γλώσσα του υπολογιστή.

Τα πιο κύρια μέρη του υπολογιστή που απαιτούν περαιτέρω ανάλυση για την προκείμενη έρευνα της διπλωματικής, είναι η Κεντρική Μονάδα Επεξεργασίας – CPU και η Μονάδα Επεξεργασίας Γραφικών – GPU.

3.2.1 Κεντρική Μονάδα Επεξεργασίας – CPU (Intel)

Σημαντικό εργαλείο για την υλοποίηση της παρούσας έρευνας είναι ο επεξεργαστής (Κεντρική Μονάδα Επεξεργασίας – CPU) της Intel. Αρχικά είναι ένα στοιχείο του ηλεκτρονικού υπολογιστή που ελέγχει όλες τις αριθμητικές και λογικές λειτουργίες του υπολογιστή και διευθύνει τις εργασίες των υπόλοιπων εξαρτημάτων του υπολογιστή. Οι σύγχρονοι επεξεργαστές στηρίζονται από τα τρανζίστορς, από αυτό οφείλεται και η υψηλή τους ταχύτητα.

Εικόνα 20: Πραγματική μορφή μιας CPU της Intel.

Κύρια στοιχεία της CPU:

- **Μονάδα ελέγχου (Control Unit – CU)**: παραδίδει, αποκωδικοποιεί και εκτελεί τις διάφορες οδηγίες, μεταφέρει σήματα ελέγχου που ελέγχουν το υλικό και μετακινεί δεδομένα γύρω από το σύστημα.

- **Αριθμητική Μονάδα Λογικής (Arithmetic Logic Unit – ALU)**: εκτελεί τις μαθηματικολογικές πράξεις (αποφάσεις), δηλαδή εκεί γίνονται οι υπολογισμοί και όπου λαμβάνονται οι αποφάσεις.

- **Καταχωρητής (Register)**: είναι μικρές ποσότητες μνήμης υψηλής ταχύτητας μέσα στην CPU. Χρησιμοποιούνται από τον επεξεργαστή για την αποθήκευση μικρών ποσοτήτων δεδομένων που απαιτούνται κατά την επεξεργασία, όπως: η διεύθυνση της επόμενης εντολής που θα εκτελεστεί, η τρέχουσα οδηγία που αποκωδικοποιείται και τα αποτελέσματα των υπολογισμών.

Άλλοι επεξεργαστές έχουν διαφορετικούς αριθμούς καταχωρητών για άλλους σκοπούς, αλλά οι περισσότεροι έχουν μερικά ή όλα τα ακόλουθα: ο μετρητής του προγράμματος (Program Counter), μητρώο διευθύνσεων μνήμης (Memory Address Register – MAR), κατάλογος στοιχείων μνήμης (Memory Data Register – MDR), τρέχων κατάλογος οδηγίας (Current Instruction Register – CIR) και συσσωρευτής (Accumulator – ACC).

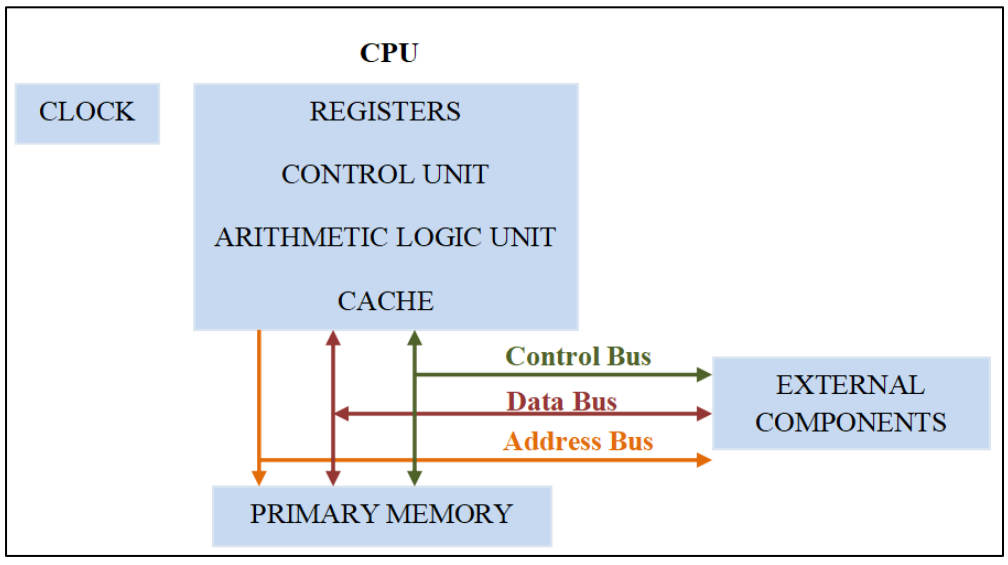
- **Κρυφή Μνήμη (Cache)**: Η προσωρινή μνήμη είναι μικρή ποσότητα μιας υψηλής ταχύτητας, μνήμη τυχαίας προσπέλασης (Random Access Memory-RAM) απευθείας μέσα κατασκευασμένη στον επεξεργαστή. Χρησιμεύει για την προσωρινή διατήρηση δεδομένων και οδηγιών που ο επεξεργαστής είναι πιθανό να επαναχρησιμοποιήσει. Από αυτό επιδιώκει την ταχύτερη επεξεργασία, μιας και ο επεξεργαστής δεν χρειάζεται να περιμένει να ληφθούν τα δεδομένα και οι οδηγίες από τη μνήμη RAM.

- **Clock**: ένα ρολόι-clock που περιέχει η CPU και χρησιμοποιείται για το συντονισμό όλων των στοιχείων του υπολογιστή. Το ρολόι στέλνει έναν κανονικό ηλεκτρικό παλμό που συγχρονίζει όλα τα εξαρτήματα. Η συχνότητα των παλμών είναι γνωστή ως ταχύτητα ρολογιού (σε Hertz). Όσο υψηλότερη είναι η συχνότητα, τόσο περισσότερες οδηγίες μπορούν να εκτελεστούν σε οποιαδήποτε δεδομένη χρονική στιγμή. Στη σημερινή εποχή, οι επεξεργαστές λειτουργούν συνήθως με ρυθμό 3 GHz έως 5 GHz, δηλαδή 3 δισεκατομμύρια έως 5 δισεκατομμύρια παλμούς ή κύκλους ανά δευτερόλεπτο.

- **Buses**: Ένα Bus είναι μια εσωτερική σύνδεση υψηλής ταχύτητας. Χρησιμοποιούνται για την αποστολή σημάτων ελέγχου και δεδομένων μεταξύ του επεξεργαστή και άλλων εξαρτημάτων.

  Χρησιμοποιούνται τρεις τύποι: Πρώτος, Δίαυλος διευθύνσεων-μεταφέρει διευθύνσεις μνήμης από τον επεξεργαστή σε άλλα στοιχεία, όπως η κύρια μνήμη και οι συσκευές εισόδου/εξόδου. Δεύτερος, Δίαυλος δεδομένων-μεταφέρει τα πραγματικά δεδομένα μεταξύ του επεξεργαστή και άλλων στοιχείων. Τρίτος, Δίαυλος ελέγχου-μεταφέρει σήματα ελέγχου από τον επεξεργαστή σε άλλα εξαρτήματα. Ο δίαυλος ελέγχου μεταφέρει επίσης τους παλμούς του clock.

Στο παρακάτω **Σχήμα 3** φαίνονται τα στοιχεία της CPU όλα μαζί να δουλεύουν για την επεξεργασία και τον έλεγχο του συστήματος:



Σχήμα 3: Τα στοιχεία CPU.

3.2.2 Μονάδα Επεξεργασίας Γραφικών – GPU (NVIDIA)

Βασικό εργαλείο για την διεξαγωγή αυτής της διπλωματικής εργασίας είναι οι κάρτες γραφικών της Nvidia. Τα πιο σημαντικά τμήματα της κάρτας γραφικών είναι η μονάδα επεξεργασίας γραφικών (Graphics Processing Unit – GPU). Για αρχή είναι ένα εξαρτήματα του υπολογιστή, μια κάρτα η οποία δέχεται δεδομένα από έναν επεξεργαστή CPU, ώστε να μετατρέψει αυτά τα δεδομένα σε εικόνα στην οθόνη του υπολογιστή. Μάλιστα πιο συγκεκριμένα είναι μια πλακέτα κυκλωμάτων που αποτελείται από έναν επεξεργαστή, κυκλώματα μνήμης RAM και ένα μικροκύκλωμα (chip) εισόδου/εξόδου (BIOS). Ο επεξεργαστής της κάρτας γραφικών λέγεται μονάδα επεξεργασίας γραφικών. Η οποία αυτή μονάδα GPU είναι σχεδιασμένη για να εκτελεί πολύπλοκους μαθηματικούς και γεωμετρικούς υπολογισμούς, όπου προσφέρουν καλύτερη απόδοση στις κάρτες.

Εικόνα 21: Πραγματική μορφή μιας GPU της Nvidia.

Η κάρτα γραφικών από μόνη της δεν λειτουργεί, χρειάζεται την βοήθεια και των άλλων εξαρτημάτων. Κάποια από αυτά τα εξαρτήματα του υπολογιστή είναι: Μητρική Κάρτα που αυτή παίρνει δεδομένα για να τα επεξεργαστεί και τροφοδοτείται από αυτή (μπορεί να τροφοδοτηθεί κατευθείαν από το τροφοδοτικό αν είναι μεγαλύτερων απαιτήσεων), Κεντρική Μνήμη, Κεντρική Μονάδα Επεξεργασίας που περιέχει δεδομένα για κάθε ένα pixel και Οθόνη.

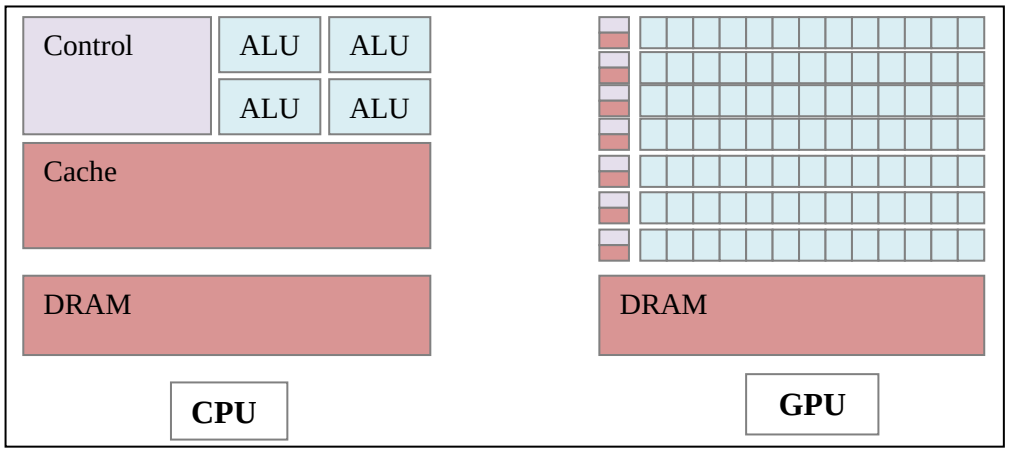
Διαφορές μεταξύ CPU – GPU

Ο φόρτος επεξεργασίας δεδομένων που δέχεται μια GPU και μια CPU για την εκτέλεση των υπολογισμών θα μπορούσε να χαρακτηριστεί ίδιος. Όμως σε μεταξύ τους αυτά τα εξαρτήματα διαφέρουν σε κάποια σημεία.

i. **Στην εκτέλεση των υπολογισμών**. Η GPU είναι έχει πιο εξειδικευμένες λειτουργίες. Και οι πολλοί πυρήνες του επεξεργαστή της κάρτας γραφικών εκτελεί πιο αποδοτικά έναν αλγόριθμο μέσα σε μεγάλο πλήθος δεδομένων. Από αυτή την ικανότητα της GPU είναι φανερό πως χρησιμοποιείται σε εφαρμογές με τεράστιες υπολογιστικές ανάγκες.

ii. **Στην αρχιτεκτονική και στη σχεδίαση**. Μια CPU αποτελείται από τις μονάδες ελέγχου της που καταλαμβάνουν το μεγαλύτερο μέρος και επίσης από τα κυκλώματα εκτέλεσης των υπολογισμών που βρίσκονται στο υπόλοιπο τμήμα. Σε αντίθεση με μια GPU, το μεγαλύτερο μέρος το κατακτούν τα τρανζίστορ για τους πολλαπλούς παράλληλους υπολογισμούς, το μικρότερο τμήμα το κατέχουν τα κυκλώματα και οι μονάδες ελέγχου.

iii. **Στον τρόπο οργάνωσης και διαχείρισης της μνήμης**. Αυτό γίνεται κατανοητό από τις δύο αρχιτεκτονικές, πολυπύρηνων επεξεργαστών Nehalem και μιας κάρτας γραφικών με εκτέλεση CUDA. Στους πολυπύρηνους επεξεργαστές Nehalem, διαθέτουμε τέσσερις πυρήνες με κάθε έναν να έχει την δική του L1 και L2 cache (μερικά KB μέγεθος). Οι πυρήνες έχουν πρόσβαση στην κοινή μνήμη cache L3 (μεγέθους 8MB) και επικοινωνούν μεταξύ τους μέσω αυτής. Το κύκλωμα της κοινής cache δίνει σηματοφόρους που αποκαθιστούν την διάταξη των δεδομένων μεταξύ αυτής και των cache ανώτερων επιπέδων των πυρήνων. Σε έναν πολυεπεξεργαστή κάρτας το νήμα μπορεί να διαβάσει και να γράψει δεδομένα στον καταχωρητή, την τοπική, την κοινόχρηστη μνήμη και την καθολική μνήμη της συσκευής. Μπορεί όμως μόνο να διαβάσει από τη μνήμη σταθερών (constant memory) και την μνήμη υφών (texture memory). Πρόσβαση για ανάγνωση και καταχώρηση δεδομένων στην καθολική μνήμη της συσκευής έχει και το κεντρικό σύστημα host.



Σχήμα 4: Διαφορές σχεδίασης μιας CPU και μιας GPU.

Οι GPUs πρωτό-σχεδιάστηκαν για τη εμφάνιση εικόνων για γραφικά υπολογιστών και κονσόλες βιντεοπαιχνιδιών, από στις αρχές όμως της δεκαετίας του 2010, μπορούν επίσης να χρησιμοποιηθούν για την ταχύτητα των υπολογισμών που περιλαμβάνουν τεράστιες ποσότητες δεδομένων.

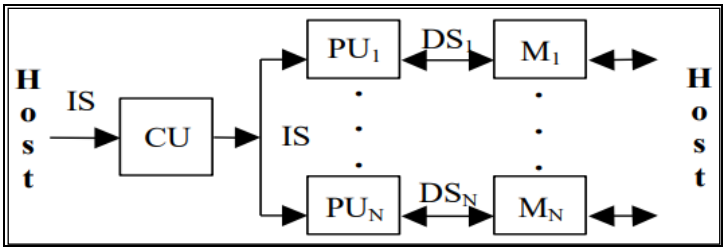
Η CPU δεν αντικαθίσταται τελείως από μια GPU. Απλά συμπληρώνει την αρχιτεκτονική της CPU, εκτελώντας παράλληλους υπολογισμούς μιας εφαρμογής καθώς το κύριο πρόγραμμα συνεχίζει να εκτελείται στην CPU. Επίσης, η CPU συντονίζει ένα μεγάλο φάσμα υπολογιστικών διεργασιών γενικής χρήσης και η GPU εκτελεί μικρότερο φάσμα, πιο εξειδικευμένων εργασιών (συνήθως μαθηματικών). Ο παραλληλισμός προσφέρει στην GPU να ολοκληρώνει περισσότερες εργασίες την ίδια χρονική διάρκεια σε σχέση με την CPU, λόγω και της μεγάλης διαφοράς έχουν στους πυρήνες, η GPU να διαθέτει από 4 έως 8 πυρήνες μιας CPU.

Όσον αφορά την αρχιτεκτονική του υπολογιστή, μια από τις βασικότερες αρχιτεκτονικές είναι η ταξινόμηση κατά Flynn. Διακρίνεται από το πλήθος και το είδος των εντολών, αλλιώς την ακολουθία εντολών Instruction Streams και την επεξεργασία των δεδομένων Data Streams.

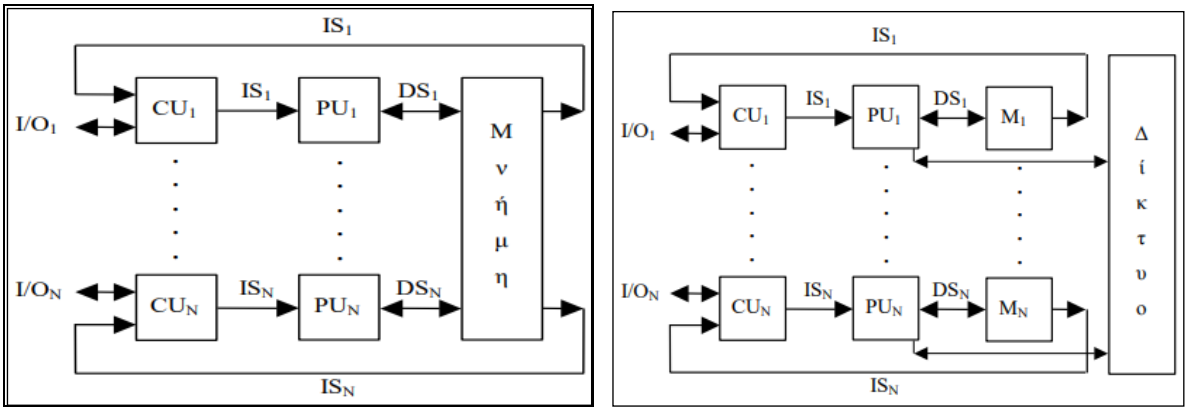
Υπάρχουν οι ακόλουθες τέσσερις κατηγορίες κατά Flynn:

- SISD (Single Instruction Stream – Single Data Stream)
- SIMD (Single Instruction Stream – Multiple Data Stream)
- MISD (Multiple Instruction Stream – Single Data Stream)
- MIMD (Multiple Instruction Stream – Multiple Data Stream)

Αναφορικά, για να περιγράψουμε τα παράλληλα προγράμματα απαιτούνται δύο από τα σημαντικότερα μοντέλα κατά την ταξινόμηση Flynn, το **SIMD** (Single Instruction Stream / Multiple Data Stream) και το **MIMD** (Multiple Instruction Stream / Multiple Data Stream). Το SIMD περιέχει πολλαπλές μονάδες επεξεργασίας όπου τρέχουν την ίδια αρχική εντολή απλά σε διαφορετικά δεδομένα. Από την άλλη το MIMD η κάθε μία από τις διάφορες εντολές τρέχουν σε διαφορετικά δεδομένα ξεχωριστά και δεν χρειάζεται απαιτητικός εξοπλισμός. Γι' αυτό, χρησιμοποιείται περισσότερο στην παρούσα εποχή, για την ευελιξία και την μη πολυπλοκότητα του μοντέλου, όπως θα περιγράψουμε περαιτέρω.



Σχήμα 5: Μοντέλο SIMD.



Σχήμα 6: (α) Μοντέλο MIMD με κοινό δίαυλο, **(β)** Μοντέλο MIMD με δίκτυο.

3.2.3 Ταξινόμηση MIMD

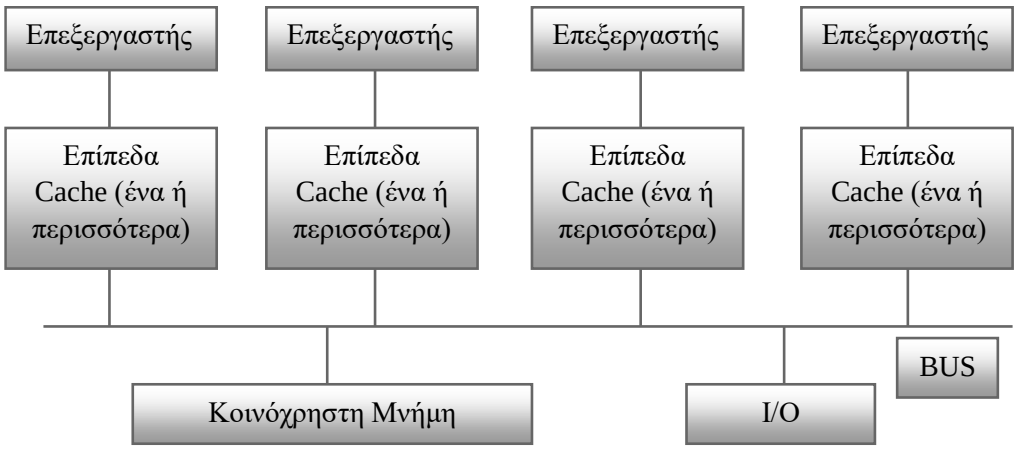
Το παρόν μοντέλο υλοποιείται κατά κύριο λόγο στον παράλληλο προγραμματισμό, αφού περιγράφει παράλληλα υπολογιστικά μοντέλα. Το μοντέλο μηχανής MIMD χωρίζεται σε δύο βασικές κατηγορίες, αναλόγως με την μέθοδο επικοινωνίας της μνήμης του συστήματός της. Αυτές οι δύο κατηγορίες υπολογιστών είναι οι εξής:

I. **Κοινής Μνήμης (Shared Memory) - Πολυεπεξεργαστές**: όπου η μνήμη διαχωρίζεται σε πολλά τμήματα ώστε να εκτελεί ταυτόχρονες προσπελάσεις από διαφορετικούς επεξεργαστές έχοντας ένα μέσο μετάδοσης, ένα κοινό δίαυλο (επικοινωνίας τύπου Bus). Για κάθε δεδομένη στιγμή ένας μόνο επεξεργαστής επικοινωνεί με ένα τμήμα μνήμης.

II. **Κατανεμημένης Μνήμης (Distributed Memory) - Πολυυπολογιστές**: πρόκειται για μια συλλογή διάφορων επεξεργαστών που επικοινωνούν μεταξύ τους μέσω ενός δικτύου διασύνδεσης.

I.Ταξινόμηση MIMD με Κοινή Μνήμη

Οι πολυεπεξεργαστές εμπεριέχουν πολλούς επεξεργαστές οι οποίοι δεν δουλεύουν ταυτόχρονα μεταξύ τους, αλλά έχουν έναν κοινό καθολικό χώρο φυσικών διευθύνσεων. Οι κάθε τροποποιήσεις των δεδομένων φαίνονται από όλους τους επεξεργαστές.



Σχήμα 7: Μοντέλο πολυεπεξεργαστών σε κοινό δίαυλο.

Οι πολυεπεξεργαστές διασυνδέονται μεταξύ τους και με τη μνήμη μέσω των δικτύων τύπου bus, ή/και με πολλαπλά bus ή δίκτυα διακοπτών crossbar, **Σχήμα 7**.

Από αυτή τη ταξινόμηση μνήμης διαπιστώνονται τα ακόλουθα πλεονεκτήματα:

- Εμφανίζεται ένα προγραμματιστικό μοντέλο και δεν έχει μεγάλες διαφορές με τα κλασικά μοντέλα σειριακού προγραμματισμού, άρα είναι εύκολο στη χρήση.

- Τα δεδομένα επικοινωνούν και συγχρονίζονται εύκολα μιας και είναι συμβατοί οι επεξεργαστές στον κοινό πόρο μνήμης.

Παρόλα αυτά διαπιστώνονται και τα ακόλουθα μειονεκτήματα:

- Η κλιμάκωση (scalability) είναι ελλιπής στο πλήθος των επεξεργαστών. Η κυκλοφορία του δίαυλου μνήμης αυξάνεται άρα και οι προσπελάσεις στην κρυφή μνήμη, αφού γίνεται προσθήκη κεντρικών επεξεργαστών μέσα στο σύστημα.

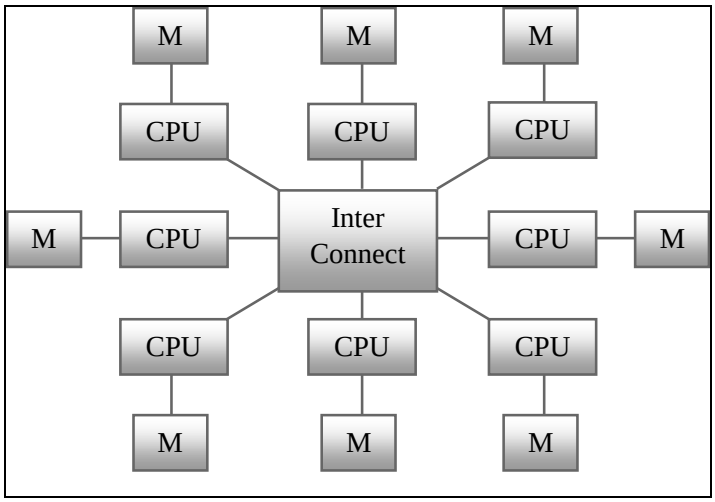
- Για να επιλυθεί το παραπάνω μειονέκτημα, χρειάζεται η παρουσία ειδικού υλικού (πχ. μεγαλύτερο εύρος ζώνης διαύλου) είτε με τη χρήση ειδικών διασυνδετικών δικτύων, όπου έτσι απαιτεί το σύστημα συντήρηση με μεγαλύτερο κόστος.

- Ο προγραμματισμός ζητάει ειδικές τεχνικές.

Η παράμετρος του μέσου χρόνου προσπέλασης μνήμης σε αυτό το μοντέλο είναι ίδιος. Η κοινή μνήμη επιτρέπει διέλευσης σε όλους τους επεξεργαστές, ανεξαρτήτως της διεύθυνσης του δεδομένου που είναι αποθηκευμένο στη μνήμη. Κι' αυτό ονομάζεται μοντέλο Ομοιόμορφης Προσπέλασης Μνήμης (Uniform Memory Access – UMA).

ΙΙ.Ταξινόμηση MIMD με Κατανεμημένη Μνήμη

Οι πολυυπολογιστές αποτελούνται από επεξεργαστές που διαθέτουν μια δική τους τοπική μνήμη. Η επικοινωνία των διασκορπισμένων μνήμων των υπόλοιπων επεξεργαστών πραγματοποιείται μέσω του δικτύου. Κάποιες γνωστές τεχνολογίες δικτύωσης είναι το *Ethernet* και ο τρόπος επικοινωνίας τους είναι είτε σημειακός είτε καθολικός. Εφόσον οι επεξεργαστές λειτουργούν με την δική τους "προσωπική" – κρυφή μνήμη, όταν μετασκευάζουν τα δεδομένα τους στη μνήμη δεν επιτρέπουν τροποποιήσεις στα δεδομένα των άλλων επεξεργαστών.



Σχήμα 8: Μοντέλο πολυυπολογιστών σε δίκτυο διασύνδεσης.

Από αυτό το μοντέλο ταξινόμησης μνήμης μπορούν να διατυπωθούν τα ακόλουθα πλεονεκτήματα:

- Η κλιμάκωση της προσθήκης περισσότερων επεξεργαστών και της αύξησης μεγέθους της μνήμης/επεξεργαστή είναι εύκολη.

- Μείωση της χρονική στιγμής της προσπέλασης στη μνήμη, αφού οι τοπικές μνήμες αφήνουν την διέλευση στα δεδομένα ανεμπόδιστα και πολύ γρήγορα. Δεν υπάρχει κεντρικός δίαυλος άρα δεν υπάρχει καθυστέρηση δεσμεύσεων.

- Αύξηση του Bandwidth (bits/second), όγκος δεδομένων που μπορούν να προσεγγίζουν σε μια συγκεκριμένη χρονική στιγμή.

- Μείωση του κόστους κατασκευής, συντήρησης και αναβάθμισης στο σύστημα, μιας και μπορούν να χρησιμοποιηθούν τεχνολογίες δικτύου και συμβατοί υπολογιστές.

Επίσης μπορούν να διατυπωθούν τα ακόλουθα μειονεκτήματα αυτού του μοντέλου:

- Η επικοινωνία με μηνύματα των επεξεργαστών μειώνει την επιτάχυνση του δικτύου. Διότι είναι πολύπλοκο το ίδιο το δίκτυο και υπάρχει καθυστέρηση στη μεταφορά του σήματος των μηνυμάτων (από τον πομπού στον δέκτη).

- Για την επαρκή επικοινωνία απαιτούνται σύνθετα μοντέλα προγραμματισμού.

- Μόνο σε μεγάλο πλήθος δεδομένων και σε εφαρμογές με περιορισμένη επικοινωνία μπορεί η κλιμάκωση να είναι αποτελεσματική.

Στη συνέχεια παρατίθεται ο τρόπος κατανομής των δύο κατηγοριών χρήσης μνήμης:

Πρώτον για στους πολυεπεξεργαστές υπάρχουν δύο τρόποι:

1. Χωρική κατανομή, έχει σχέση με την φυσική τοποθέτηση της μνήμης και επεξεργαστή.

2. Λογική κατανομή, έχει σχέση με τον τρόπο που αναγνωρίζει ο επεξεργαστής τη μνήμη.

Δεύτερον για τους πολυυπολογιστές αποκλειστικά για τη χρήση Χωρικά κατανεμημένης μνήμης υπάρχουν οι δύο παρακάτω περιπτώσεις:

1. Λογικά κατανεμημένη, οι επεξεργαστές προσπελαύνουν, ξεχωριστά, θέσεις μνήμης που περιλαμβάνονται μέσα στην προσωπική μνήμη τους.

2. Λογικά κοινή, οι επεξεργαστές προσπελαύνουν κάθε διεύθυνση μνήμης.

Στο μοντέλο πολυυπολογιστή εντοπίζονται δύο περιπτώσεις:

1. Μοντέλο Χωρικά και Λογικά κατανεμημένης μνήμης. Ο επεξεργαστής έχει διέλευση στα δεδομένα μόνο της δικής του μνήμης.

2. Μοντέλο Χωρικά/Λ ογικά κατανεμημένης/κοινής μνήμης (Κοινής Κατανεμημένης Μνήμης – Distributed Shared Memory). Εδώ, οι μνήμες είναι χωρικά κατανεμημένες, όπου οι επεξεργαστές τις αναγνωρίζουν ως ενιαίος χώρος διευθύνσεων. Από προγραμματιστικής μεριάς, αυτοί οι πολυυπολογιστές δεν έχουν διαφορές από το μοντέλο πολυεπεξεργαστών κοινής μνήμης, γιατί έχουμε τη δυνατότητα προσπέλασης σε κάθε διεύθυνση (απομακρυσμένης ή τοπικής).

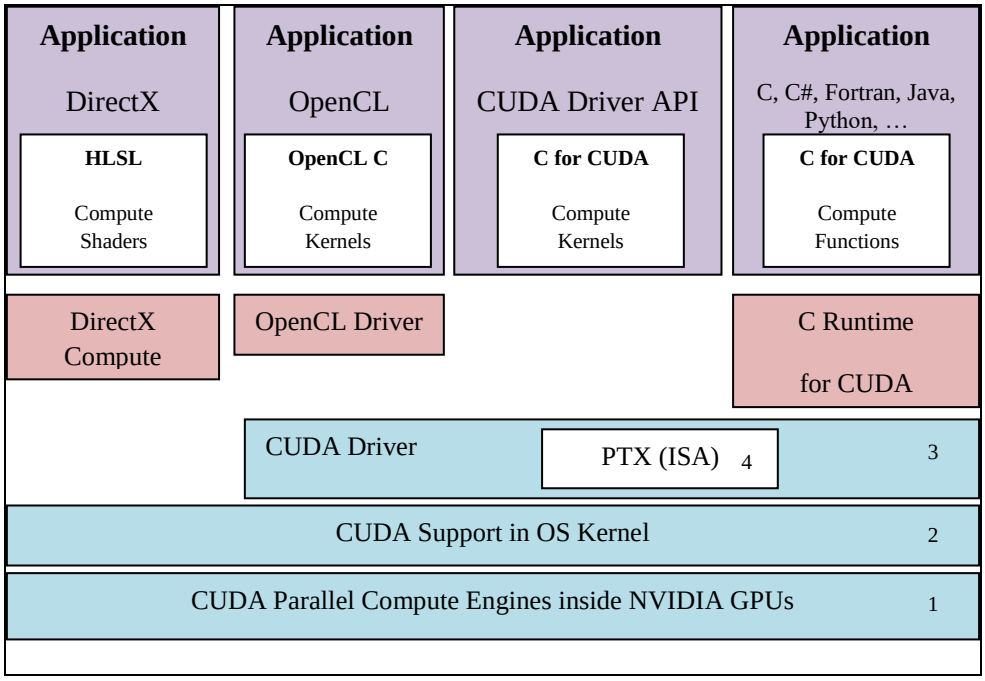
Η μεταβλητή που αλλάζει εδώ, είναι ο χρόνος προσπέλασης της μνήμης που είναι αλλιώς σε μια εντολή προσπέλασης διεύθυνσης μέσα σε μια χωρικά τοπική μνήμη και αλλιώς σε μια εντολή μέσα σε μια χωρικά απομακρυσμένη μνήμη. Εξαιτίας αυτού οι πολυυπολογιστές DSM λέγονται μηχανές Μη-Ομοιόμορφης Προσπέλασης Μνήμης (Non-Uniform Memory Access – NUNA) σε αντίθεση με τους πολυεπεξεργαστές UMA.

3.3 Μοντέλα Παράλληλου Προγραμματισμού

Για την σύνταξη και την υλοποίηση των παράλληλων επεξεργασιών τα μοντέλα διαχωρίζονται σε δύο κατηγορίες, σύμφωνα με τον τρόπο διάταξης της μνήμης του συστήματος. Τα δύο μοντέλα είναι τα μοντέλα Κοινής και Κατανεμημένης Μνήμης.

- Μοντέλο Κοινής Μνήμης: OpenMP, το OpenCL, το DirectCompute και το CUDA.

- Μοντέλο Κατανεμημένης Μνήμης: OpenMP, MPI, Hadoop και το MapReduce.



Σχήμα 9: Διαστρωμάτωση CUDA και άλλες προγραμματιστικές πλατφόρμες.

3.3.1 Μοντέλα Παράλληλου Προγραμματισμού – Κοινής Μνήμης

A. Μοντέλο OpenMP

Το OpenMP (Open Specification for Multi-Processing) είναι ένα πρότυπο της παράλληλης επεξεργασίας, όπου δίνει την δυνατότητα ανάπτυξης παράλληλων προγραμμάτων για συστήματα κοινής μνήμης. Από αυτό το πρότυπο ο χειριστής μπορεί να δώσει ένα σύνολο από εντολές και οδηγίες προς τον μεταγλωττιστή ώστε να ορίσει τα μέρη του προγράμματος που θέλει να προκύψουν παράλληλα με την απαίτηση ενός κόμβου από τον ορισμό της κοινής μνήμης.

Η διεπαφή εφαρμογών προγραμμάτων (Application Programm Interface-API) του OpenMP χρησιμοποιείται για τη μεταβίβαση δεδομένων μεταξύ εφαρμογών λογισμικού με έναν τυποποιημένο τρόπο. Το OpenMP δεν είναι σχεδιασμένο με την μέγιστη αποδοτικότητα στη χρήση κοινής μνήμης. Όμως είναι ένα μικρό και εύκολο πρότυπο, με το οποίο η παραλληλοποίηση μπορεί να πραγματοποιηθεί. Έχει την δυνατότητα σταδιακής παραλληλοποίησης ενός προγράμματος σε αντίθεση με το MPI και την παραλληλία χονδρού και λεπτού κόκκου.

Βασικότερος στόχος του OpenMP είναι η φορητότητα (portability). Υποστηρίζει Fortran (77, 90 και 95) και C/C++, Unix και WindowsNT.

Το συγκεκριμένο μοντέλο που αναλύουμε βασίζεται στο πολυνηματικό μοντέλο παραλληλισμού. Με άλλα λόγια είναι μια διεργασία κοινής μνήμης που αποτελείται από πολλά νήματα. Η διαδικασία μιας εφαρμογής σε OpenMP είναι ότι ξεκινά με ένα κύριο νήμα (master thread), εκτελεί σειριακά μέχρι να συναντήσει το πρώτο παράλληλο τμήμα. Όταν ο προγραμματιστής έχει ορίσει την περιοχή της παραλληλίας (parallel region), τότε δημιουργούνται τα απαιτούμενα νήματα (fork-join μοντέλο) και το κομμάτι του κώδικα που είναι μέσα στην παράλληλη περιοχή εκτελείται παράλληλα. Όταν τελειώσει ο υπολογισμός της παράλληλης περιοχής όλα τα νήματα τερματίζουν και συνεχίζει μόνο το κύριο νήμα.

B. Μοντέλο PThreads ή POSIX

Γενικό μοντέλο POSIX (PThreads/Portable Operating System Interface). Εκτελείται ως ένα αρχείο επικεφαλίδων (header file) και η βιβλιοθήκη προς την δημιουργία και το χειρισμό της κάθε μονάδας εκτέλεσης (worker) που λέγεται Threads. Τα νήματα (threads) είναι ένα σύνολο τύπων και κλήσεων διαδικασιών της γλώσσας προγραμματισμού C.

C. Μοντέλο CUDA

Η CUDA (Compute Unified Device Architecture) είναι μια υλική και λογισμική αρχιτεκτονική που είναι σχεδιασμένη να υποστηρίζει τον παράλληλο προγραμματισμό σε κάρτες γραφικών GPU. Περιλαμβάνει την επέκταση του προγραμματισμού σε γλώσσες C, C++, Fortran, OpenCL, DirectCompute κα. Καθώς προσφέρει την δυνατότητα παράλληλης εκτέλεσης σε προγράμματα σε αυτές τις γλώσσες.

Ειδικότερα για το μοντέλο OpenCL όπως αναγράφουμε πιο πάνω, μπορεί εναλλακτικά να προσφέρει λύση αντί το μοντέλο CUDA. Αποτελείται από προγραμματισμό χαμηλού επιπέδου. Πρωτοεμφανίστηκε το 2008 και είναι ένα από τα πρώτα μοντέλα ανοικτού προτύπου προγραμματισμού εφαρμογών. Η εκτέλεση του γίνεται σε ετερογενή παράλληλα συστήματα, με κάρτες γραφικών, πολυπύρηνους επεξεργαστές και διάφορα άλλα συστήματα επεξεργασίας. Χαρακτηριστικό του μοντέλου είναι η λειτουργικότητα του, ανεξαρτήτως του υλικού εκτέλεσης. Ο σχεδιασμός αυτός δίνει την ευκολία στις εφαρμογές να μεταφέρονται. Η σύνταξη του προγράμματος μέσα σε οποιοδήποτε υλικό, φαίνεται να είναι ένα σημαντικό πρότυπο και εκτελείται σε ένα μεγάλο εύρος συστημάτων που στηρίζουν διαφορετικά APIs.

Η σύνταξη του παράλληλου προγραμματισμού στο παρόν πρότυπο χρησιμοποιεί τη γλώσσα C99 και άλλες προσθήκες APIs. Πρώτα προσφέρουν τον κώδικα που θα εκτελέσει στην πλατφόρμα, και μετά τον εκτελούν χρησιμοποιώντας τις απαραίτητες συναρτήσεις. Ο παραλληλισμός του OpenCL έχει παραλληλοποίηση τόσο σε διεργασίες με επίπεδο task-based parallelism όσο και σε data-based parallelism.

Στη συνέχεια μπορούμε να δούμε κάποιες ομοιότητες και διαφορές των δύο μοντέλων CUDA και OpenCL στον παρακάτω **Πίνακα 1**:

Ομοιότητες	Διαφορές
Το OpenCL αντιμετωπίσει το υλικό ιεραρχικά θεωρώντας ότι υπάρχει μια κύρια διεργασία, που γίνεται στον host (host process) και οι υπόλοιπες δευτερεύουσες διεργασίες στις συσκευές (device processes).	Το OpenCL είναι ένα διασυστημικό πρότυπο οπότε μπορεί να υποστηρίζει ετερογενή συστήματα (όπως π.χ. GPU, CPU) αντιμετωπίζοντάς τα όλα αυτά σαν μια ενιαία πλατφόρμα.
Εκτέλεση που βασίζεται στην παραλληλία δεδομένων.	Από το παραπάνω, χρησιμοποιεί ένα πολύπλοκο μοντέλο για την διαχείριση των συσκευών και τη μεταγλώττιση και την εκκίνηση του πυρήνα.
Σύνθετη ιεραρχία μνήμης.	

Πίνακας 1: Ομοιοτήτων και Διαφορών της CUDA και της OpenCL.

D. Μοντέλο DirectCompute

Εμφανίστηκε αυτό το μοντέλο για τις κάρτες γραφικών της Microsoft όπου εφαρμόζεται στο DirectX API. Η σύνταξη του προγράμματος εξαρτάται στην .hlsl (γλώσσα προγραμματισμού για συναρτήσεις shader), αφού χρησιμοποιεί αντίστοιχες έννοιες προγραμματισμού με αυτές των υπολοίπων μοντέλων. Η χρήση του για συγγραφή παράλληλου κώδικα χαμηλού επιπέδου, αποτελεί σκοπό και για τον παράλληλο προγραμματισμό σε GPU που εκτελούνται σε συστήματα MS Windows.

Το DirectCompute συμβαδίζει με όλα τα GPU chip που παρέχουν εγκαταστημένη την Direct X11. Κατευθύνεται προς τον προγραμματισμό διαδραστικών εφαρμογών σε πραγματικό χρόνο (interactive, real-time applications) και γι' αυτό είναι άμεσα συνδεδεμένο με τα APIs του πακέτου Direct3D. Το συγκεκριμένο API μπορεί να εκμεταλλευτεί σε προβλήματα που χρησιμοποιούν παράλληλο κώδικα, για παράδειγμα σε προσομοιώσεις, σε κρυπτογράφηση, σε επεξεργασία εικόνας και καταγραφής και στον τομέα PC games.

3.3.2 Μοντέλα Παράλληλου Προγραμματισμού – Κατανεμημένης Μνήμης

A. Μοντέλο MPI

Το MPI είναι μια διεπαφή/βιβλιοθήκη μεταβίβασης μηνυμάτων για την επικοινωνία παράλληλης επεξεργασίας. Το βρίσκουμε σε συστήματα κατανεμημένης μνήμης, καθώς ο υπολογισμός κατανέμεται σε ένα σύνολο όμοιων διεργασιών, ακολουθιακά αριθμημένων, που γίνονται σε πολλούς κόμβους και ο κώδικάς επεξεργάζεται ανάλογα με τον σειριακό αριθμό τους.

Εργαλεία: Intel NX, IBM EUI, IBM CCL, PARMACS, OCCAM, PVM κα.

Αναπτύχθηκε για σχεδιαστικές ανάγκες. Αποτελεί τις βασικές λειτουργίες ανταλλαγής μηνυμάτων και τις σχετικές λειτουργικότητες (συλλογική επικοινωνία). Και υποστηρίζει την C και την Fortran.

B. Μοντέλο OpenMP

Αυτό το μοντέλο έχει οδηγίες ή συναρτήσεις για των μεταφραστή και τις περιβαλλοντικές μεταβλητές, όπου αυτές ανακαλούνται από άλλες γλώσσες υψηλού επιπέδου (C, C++, Fortran). Αυτό το πρόγραμμα αρχίζει με μια ακολουθιακή διεργασία δημιουργώντας ένα αρχικό νήμα. Μετά, η κεντρική διεργασία παράγει πιο πολλά νήματα με δικό τους μοναδικό αριθμό (εντολή fork). Τα νήματα αυτά εργάζονται παράλληλα και μπορούν να συνενωθούν σε οποιοδήποτε στάδιο της εκτέλεσης (εντολή join).

C. Μοντέλο UPC

Η UPC (Unified Parallel C) είναι ένα προγραμματιστικό μοντέλο που τροποποιεί τη σύνταξη μιας σειριακής γλώσσας. Διαφορετικά, είναι μια παράλληλη γλώσσα προγραμματισμού πάνω στο σύστημα κοινής ή και κατανεμημένης μνήμης. Κατέχει την έννοια της διαχωρισμένης μνήμης (partitioned memory), διότι ο χρήστης βλέπει το σύστημα ως ένα σφαιρικό διάστημα διευθύνσεων (global address space), όπου διακλαδώνονται σε πολλά λογικά διαστήματα διευθύνσεων ανά νήμα. Κάθε νήμα έχει δύο τύπους προσβάσεων άλλων νημάτων και χρησιμοποιούν την ίδια σύνταξη. Από τον όρο της συνάφειας νήματος (thread affinity), βελτιώνεται η απόδοση των προσβάσεων στη μνήμη μεταξύ του ενός νήματος και του διαστήματος διευθύνσεων ανά νήμα όπου έχει δεσμευθεί το νήμα.

D. Μοντέλο Hadoop

Το μοντέλο αυτό της Apache χρησιμοποιεί μια προγραμματιστική πλατφόρμα για παράλληλη επεξεργασία σε μεγάλου όγκου δεδομένων σε συστοιχίες υπολογιστικών συστημάτων, γραμμένη σε γλώσσα Java.

E. Μοντέλο MapReduce

Διαχωρίζει τον κύριο όγκο δεδομένων σε άλλα μικρότερα μέρη και τα θέτει στους αντίστοιχους κόμβους της συστοιχίας, είναι δηλαδή μια διασύνδεση. Βασίζεται σε δύο διεργασίες, όπου εκτελούνται για κάθε κόμβο μοναδικά:

- **Master JobTracker**: προγραμματίζει τους υπολογισμούς που είναι να εκτελεστούν και ξανά εκτελεί τις διεργασίες τις αποτυχημένες διεργασίες.

- **Slave TaskTracker**: εκτελεί τους αντίστοιχους υπολογισμούς στους κόμβους που του αναθέτει ο Master JobTracker.

Στην παρούσα διπλωματική, θα αναλυθεί περισσότερο, από τα παραπάνω μοντέλα παράλληλου προγραμματισμού, το μοντέλο CUDA, διότι διαθέτει την προγραμματιστική ικανότητα σε κάρτες γραφικών, όπου απασχολεί το αντικείμενο της έρευνας.

3.3.3 Βασικά Στοιχεία Μοντέλου CUDA

Στα τέλη του 2006 η Nvidia προώθησε το εξελιγμένο της μοντέλο προγραμματισμού CUDA που διαθέτει:

1. APIs για συγγραφή και εκτέλεση υπολογιστικών προγραμμάτων.

2. Υποστήριξη γλωσσών προγραμματισμού υψηλού επιπέδου (CUDA C).

3. Μεταφραστή για την μετάφραση των προγραμμάτων (nvcc compiler).

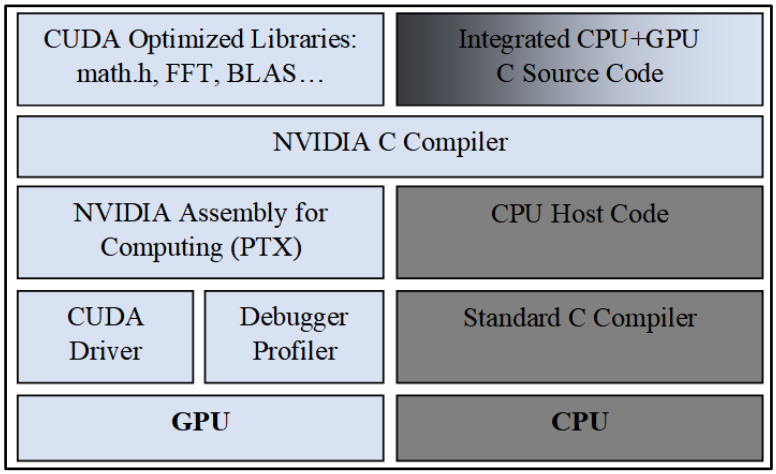
4. Στόχος αυτής της αρχιτεκτονικής ήταν να προσφέρει εργαλεία προγραμματισμού για εφαρμογές γενικού σκοπού (άλλης φύσης από τις εφαρμογές επεξεργασίας γραφικών) που θα καταφέρνουν να υλοποιούνται στα κυκλώματα των καρτών γραφικών.

Πέρα από τα προηγούμενα, παρέχεται μάλιστα και η δυνατότητα συμπληρωματικής-βοηθητικής χρήσης πρόσθετων βιβλιοθηκών όπως:

- CUFFT, που αποτελεί βιβλιοθήκη για ταχύτατους μετασχηματισμούς Fourier.

- CUBLAS, βιβλιοθήκη υποστήριξης συναρτήσεων γραμμικής άλγεβρας.

- NPP, βιβλιοθήκη συναρτήσεων για επεξεργασία δεδομένων με CUDA.
- Nsight ή άλλα προγραμματιστικά εργαλεία για αποσφαλμάτωση και βελτιστοποίηση του κώδικα.



Σχήμα 10: Προγραμματιστικά εργαλεία σε ένα σύστημα CUDA.

Η αρχιτεκτονική CUDA είναι ετερογενούς φύσης. Αυτό υποδηλώνει πως ο κώδικας εκτελείται από δύο ταυτόχρονα συστήματα, ένα από το σύστημα host, που περιλαμβάνεται από έναν ή περισσότερους επεξεργαστές και ένα σύστημα με πολλές GPU, μία ή περισσότερες κάρτες γραφικών. Όταν είναι ένα ετερογενές σύστημα προς εκτέλεση δεν γίνεται να εκτελείται σε ένα μόνο σύστημα. Απαιτείται οργάνωση σε τμήματα, δηλαδή κάποια στο host και κάποια στην κάρτα γραφικών. Αυτή η διαδικασία καλείται, γιατί το κάθε ένα σύστημα αποτελείται από άλλο διαμορφωμένο υλικό και με διαφορετικό είδος κώδικα λειτουργεί αποδοτικά.

Η κάρτα γραφικών λειτουργεί αποδοτικά μέσα από τις εξής ορισμένες συνθήκες:

- Οι υπολογισμοί που εργάζονται παράλληλα την ίδια χρονική στιγμή, μέσα σε τεράστιο πλήθος δεδομένων.
- Μερικές περιπτώσεις της προσπέλασης μνήμης διευκολύνουν το υλικό της κάρτας γραφικών να συνδέει τις προσπελάσεις στη μνήμη και να εκτελεί λειτουργίες σύνταξης και ανάγνωσης, σε πολλά δεδομένα ταυτόχρονα, με μία μόνο εντολή / λειτουργία.
- Η διαδικασία των υπολογισμών με τη λιγότερη απαίτηση αντιγραφής δεδομένων κάρτας προς τον host και αντίστροφα, στοιχίζει σε χρόνο και απόδοση.

Στην περίπτωση που οι πολλαπλοί πυρήνες εκτελούν λειτουργίες στα ίδια δεδομένα, τότε τα αποτελέσματα, μέχρι να κληθεί να τα χρησιμοποιήσει ο επόμενος πυρήνας, του κάθε πυρήνα αποθηκεύονται προσωρινά στη μνήμη της συσκευής. Η αντιγραφή των αποτελεσμάτων και μετά η επανα-προσκόμιση τους στην κάρτα, συνεπάγεται σε απώλεια χρόνου.

Στη περίπτωση πολύ περίπλοκων υπολογισμών η εκτέλεσή τους σε κάρτα CUDA διακρίνεται χρήσιμη για τη καλυτέρευση της απόδοσης, η απώλεια χρόνου στη μετάδοση των δεδομένων από το ένα σύστημα στο άλλο, ισοσταθμίζεται μέσω της ταχύτητας εκτέλεσης των υπολογισμών.

Η σύνταξη των προγραμμάτων με το CUDA μοντέλο αποτελείται με πολλές δυνατότητες ως προς τη γλώσσα σύνταξης. Μια γλώσσα σύνταξης χρησιμοποιεί ευρέως διαδεδομένες γλώσσες υψηλού επιπέδου (πχ. C, C#, Python και Java) προγραμματισμού. Πέρα από τις γλώσσες υψηλού επιπέδου (πχ. CUDA C), γίνεται να εκμεταλλευτούμε και άλλη διεπαφή προγραμματισμού, το Driver API για χαμηλού επιπέδου. Ο κώδικας που είναι γραμμένος από με τις βιβλιοθήκες του Driver API είναι πιο δύσκολος στην κατανόηση, αλλά ο έλεγχος των παραμέτρων των συναρτήσεων της κάρτας γραφικών είναι καλύτερος. Το μοντέλο CUDA βασίζεται στην ενοποιημένη γραμμή εντολών που χρησιμοποιείται από τις κάρτες για την επεξεργασία και την απεικόνιση γραφικών.

- **CUDA API**

Το μοντέλο CUDA προσφέρει δύο προγραμματιστικές διεπαφές:

- **CUDA Driver API**: είναι χαμηλού επιπέδου διεπαφή, που εκτελείται από τη βιβλιοθήκη nvcuda (πρόθεμα cu) και αποτελεί τη βάση και για άλλες εκτελέσεις υψηλότερου επιπέδου.
- **C Runtime API**: είναι υψηλού επιπέδου διεπαφή, χρησιμοποιώντας την δυναμική βιβλιοθήκη cudart (πρόθεμα cuda) και έχει ως βάση της εκτέλεσης το CUDA Driver API.

Οι διεπαφές και οι βιβλιοθήκες του μοντέλου CUDA έχουν συναρτήσεις για τις παρακάτω λειτουργίες:

- Διαχείριση της κάρτας γραφικών.
- Διαχείριση του context.
- Διαχείριση μνήμης.
- Έλεγχος εκτέλεσης εντολών.
- Διαχείριση αναφορών υφής (texture references).
- Διαχείριση των code modules.
- Διαλειτουργικότητα με τις βιβλιοθήκες της OpenGL και της Direct3D.

Διαφορές CUDA Driver API – C Runtime CUDA

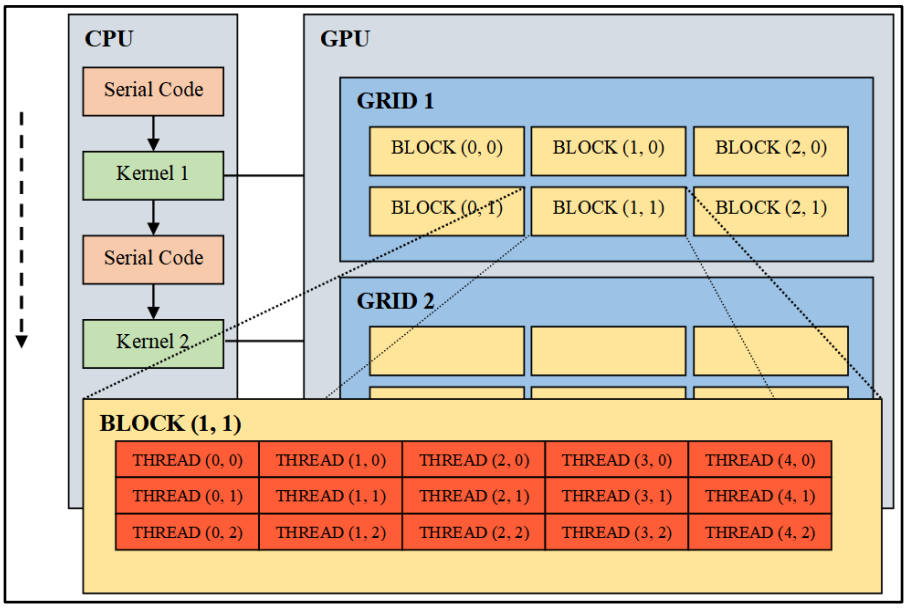
Οι δύο διεπαφές προγραμματισμού της CUDA είναι σαφώς διαφορετικές. Αν και μπορούν δύο αλγόριθμοι που βασίζονται σε διαφορετικές διεπαφές να συνεργαστούν, εντούτοις αυτό γίνεται σε μικρό βαθμό και με συγκεκριμένες μόνο συναρτήσεις.

Driver API	C Runtime
Μια διεπαφή με μεγάλη και περίπλοκη συγγραφή κώδικα. Η αποσφαλμάτωση είναι δύσκολη, αλλά έχει καλύτερο έλεγχο των λειτουργιών της κάρτας. Ο καθορισμός των παραμέτρων που δίνονται στους πυρήνες και η εκκίνησή τους είναι δυσκολότερη, διότι οι αρχικοποίηση γίνεται απευθείας μέσω συναρτήσεων και όχι έμμεσα μέσω ειδικής σύνταξης.	Χρησιμοποιείται σε μεγαλύτερο βαθμό αυτό το περιβάλλον της CUDA, παρέχει μεγαλύτερη ευκολία στον προγραμματισμό της κάρτας μέσω έμμεσης αρχικοποίησης, διαχείρισης περιβάλλοντος και ενότητας.

Πίνακας 2: Διαφορές μεταξύ Driver API και C Runtime.

Χρήση των εννοιών στο μοντέλο της CUDA

Από τα προγραμματιστικά εργαλεία CUDA ο συντασσόμενος κώδικας διαθέτει εντολές και είναι ενιαίος, ανάλογα το πρόθεμά εκτελούν τις εντολές είτε στους πυρήνες του κεντρικού επεξεργαστή, είτε στην κάρτα γραφικών.



Σχήμα 11: Οργάνωση νημάτων σε μια κάρτα γραφικών.

Επομένως, διαχωρίζεται ο ενιαίος κώδικας, ο οποίος χωρίζεται στον κώδικα προς εκτέλεση στον host και τον κώδικα προς εκτέλεση στη device του συστήματος.

Η επίτευξη της παράλληλης εκτέλεσης των διεργασιών στη συσκευή, με υποστήριξη της CUDA, απαιτεί οργάνωση με συγκεκριμένο τρόπο. Με βάση το μοντέλο της CUDA εισάγονται έννοιες που αφορούν την ομαδοποίηση των διεργασιών και την κατάλληλη διαχείρισή τους.

Η διεργασία είναι ένα στιγμιότυπο του προγράμματος, που εκτελείται μαζί με τα απαραίτητα συστατικά που χρειάζεται προς εκτέλεση, όπως οι καταχωρητές, ο μετρητής προγράμματος, απαραίτητοι πόροι του λειτουργικού συστήματος, το τμήμα της μνήμης που περιέχει τον κώδικα κτλ.

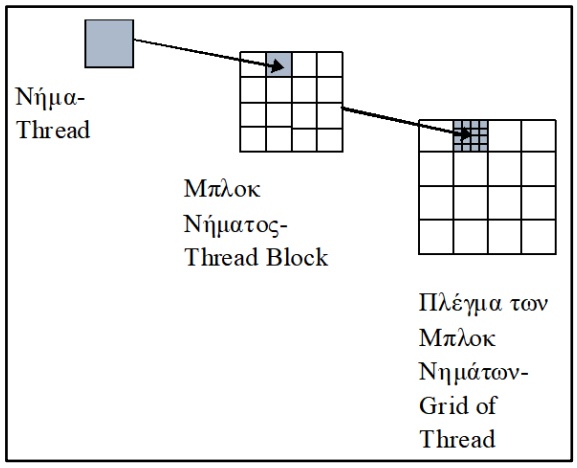
- Οι **πυρήνες (kernel)** είναι συναρτήσεις στη γλώσσα C και καθορίζουν τα τμήματα κώδικα, που θα εκτελεστούν στη συσκευή και δημιουργούν νήματα τα οποία εκτελούνται παράλληλα όσες φορές επιθυμεί ο προγραμματιστής μέσα από τις παραμέτρους που τους δίνει.

- Τα **μπλοκ (block)** δίνουν τη δυνατότητα της παράλληλης εκτέλεσης των νημάτων δίνεται μέσα από την οργάνωσή τους σε ομάδες. Δηλαδή τα νήματα τα οποία βρίσκονται μέσα στα μπλοκ εκτελούνται παράλληλα. Σε κάθε περίπτωση, το κάθε μπλοκ μπορεί να συλλάβει έως κάποιον αριθμό νημάτων, ο οποίος καθορίζεται πάντα από τις δυνατότητες της εκάστοτε κάρτας εκτέλεσης. Αναλυτικά, η ιδιότητα maxThreadsPerBlock, ορίζει το μέγιστο αριθμό νημάτων που μπορεί να φιλοξενήσει ένα μπλοκ (ίση με 512 νήματα). Επίσης τα μπλοκ οργανώνονται σε συστοιχίες για να αποτελέσουν ένα ή περισσότερα πλέγματα, ο αριθμός τους εξαρτάται από το πλήθος των δεδομένων και τον αριθμών των διαθέσιμων επεξεργαστών στην κάρτα γραφικών.

- Τα **πλέγματα (grid)** αποτελούνται από οργανωμένα σε συστοιχίες μπλοκ, ο αριθμός των οποίων εξαρτάται από το πλήθος των δεδομένων μας και τον αριθμών των διαθέσιμων επεξεργαστών στην κάρτα γραφικών.

- Τα **νήματα (threads)** εκτέλεσης στα οποία διασπάται ο κώδικας προκειμένου να μπορεί να εκτελεστεί παράλληλα από τις μονάδες επεξεργασίας της κάρτας.

Η διαμόρφωση νημάτων με βάση και τα παραπάνω, φαίνεται στο **Σχήμα 12**.



Σχήμα 12: Η διαμόρφωση του νήματος μέσα στο πλέγμα εκτέλεσης.

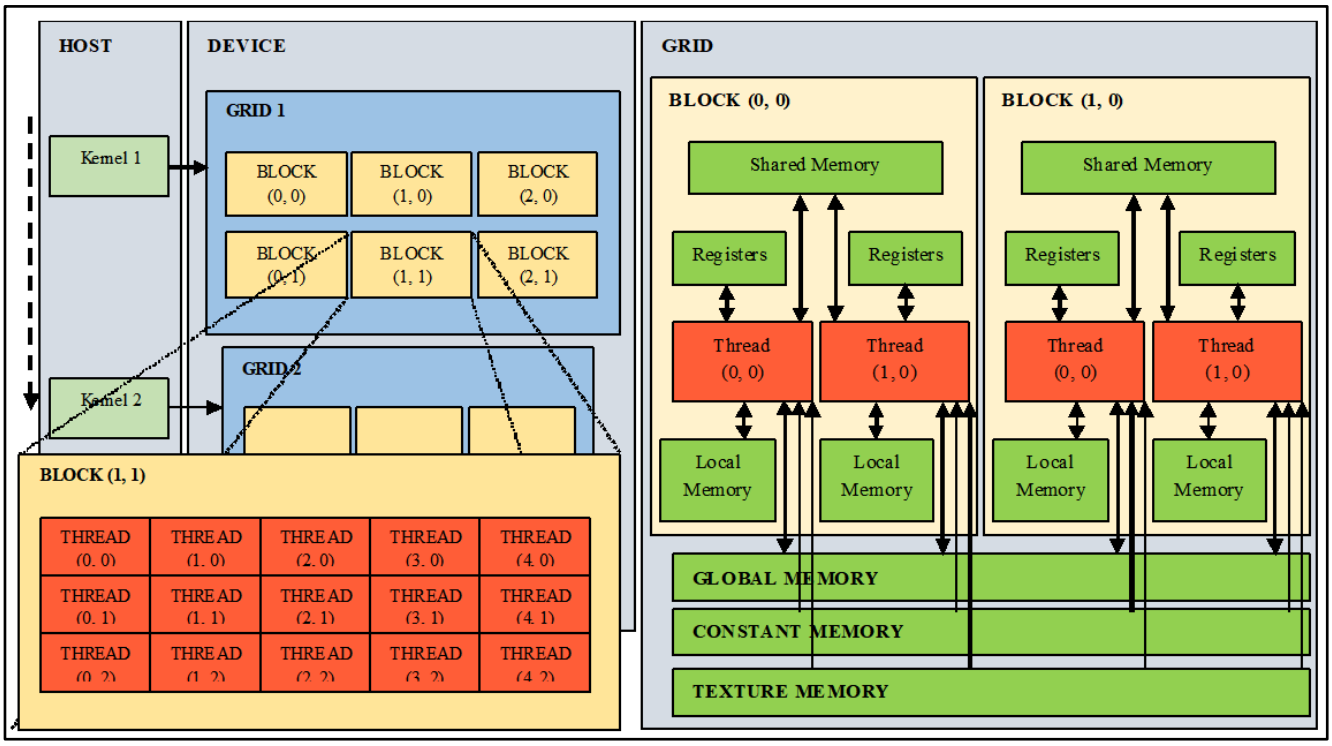
Το υπολογιστικό πλέγμα καλύπτει ένα ορισμένο χώρο και στα όρια του πλέγματος απαιτείται ο καθορισμός οριακών συνθηκών.

Ο προσδιορισμός των προηγούμενων εννοιών χρησιμεύουν για CUDA προγραμματισμό, χρησιμοποιώντας όμως ορισμένες ενσωματωμένες μεταβλητές. Ο προγραμματιστής μπορεί με αυτές τις μεταβλητές να καθορίσει επακριβώς τις διαστάσεις ενός μπλοκ ή ενός πλέγματος και να έχει πρόσβαση στους δείκτες που προσδιορίζουν μοναδικά κάθε μπλοκ ή κάθε νήμα μέσα σε ένα πλέγμα.

Οι μεταβλητές αυτές που αναφέραμε είναι οι ακόλουθες:

- gridDim (τύπου dim3 που περιέχει τις διαστάσεις ενός πλέγματος)
- blockIdx (τύπου unint3 που προσδιορίζει κάθε μπλοκ μέσα σε ένα πλέγμα)
- blockDim (τύπου dim3 και προσδιορίζει τις διαστάσεις ενός μπλοκ)
- threadIdx (τύπου unint3 που προσδιορίζει μονοσήμαντα ένα νήμα μέσα στο πλέγμα)
- warpSize (τύπου int που προσδιορίζει το μέγεθος ενός σμήνους σε νήματα)

Σημειώνεται, ότι δεν μπορούμε να κάνουμε ανάθεση τιμής των μεταβλητών, ούτε επίσης να εξαγάγουμε τη διεύθυνση αποθήκευσης στη μνήμη αυτών. Δουλεύοντας με αυτές τις μεταβλητές μπορούμε να έχουμε πρόσβαση σε χρήσιμες ιδιότητες της κάρτας γραφικών, **Σχήμα 13**.



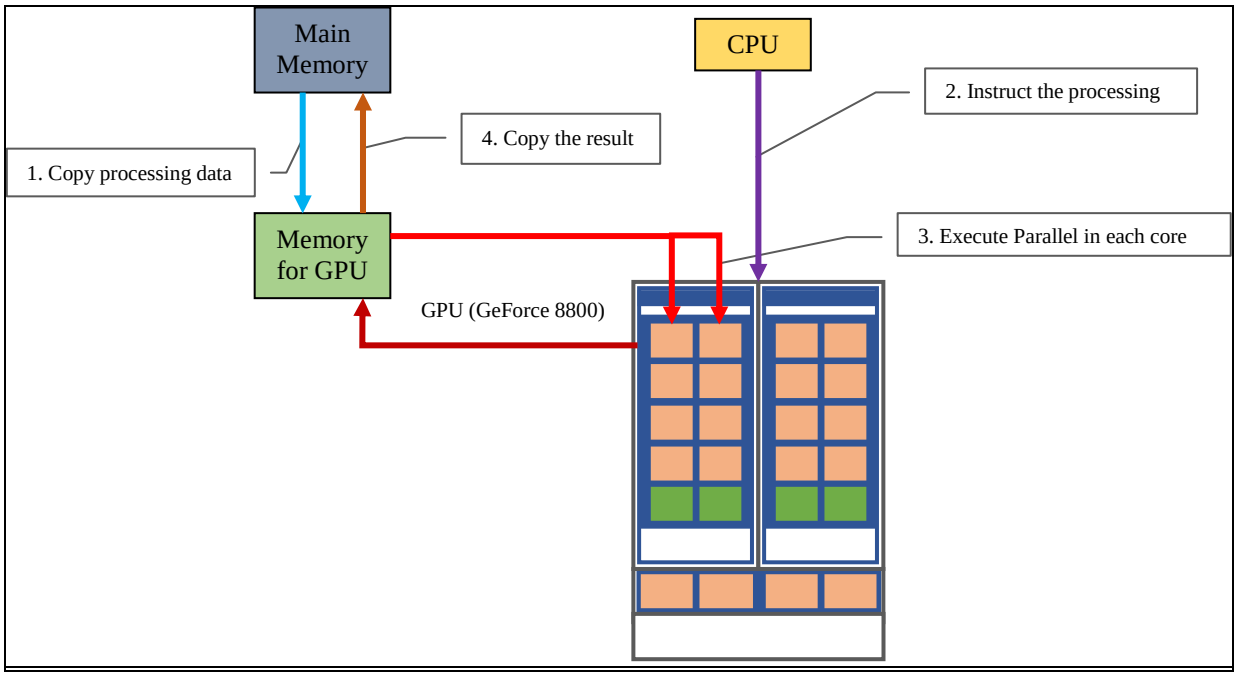
Σχήμα 13: Ιεραρχία μνήμης ανά μπλοκ και πλέγμα σε κάρτα γραφικών.

3.4 Εφαρμογή CUDA στις Κάρτες Γραφικών – GPU

Οι προγραμματιστές εφαρμογών αξιοποιούν την απόδοση της παράλληλης αρχιτεκτονικής GPU χρησιμοποιώντας ένα παράλληλο μοντέλο προγραμματισμού που εφευρέθηκε από τη NVIDIA που ονομαζόμενο CUDA. Όλες οι GPU της NVIDIA – GeForce®, Quadro® και Tesla®, υποστηρίζουν το μοντέλο παράλληλου προγραμματισμού NVIDIA® CUDA®.

Η πρόσφατη αρχιτεκτονική η CUDA (Compute Unified Device Architecture) της Nvidia επιτρέπει τη σύνταξη του κώδικα για τους επεξεργαστές στις κάρτες γραφικών και χρησιμοποιείται επίσης ως προγραμματιστικό εργαλείο σε γλώσσα C/C++.

Η υλοποίηση του κώδικα CUDA πάνω στην κάρτα γραφικών μπορεί να αναπαρασταθεί στο παρακάτω **Σχήμα 14**, έχοντας την ανάλογη ακολουθία εκτέλεσης.



Σχήμα 14: Ακολουθία εκτέλεσης του μοντέλου CUDA.

Η αρχιτεκτονική της CUDA υποστηρίζεται από τους Streaming Multiprocessors (SMs), όπου εκτελούν πάρα πολλά νήματα συγχρόνως. Αυτοί οι επεξεργαστές έχουν την δυνατότητα να εκτελούν με μια συγκεκριμένη αρχιτεκτονική, μεγάλο αριθμό παράλληλων νημάτων.

3.4.1 Αρχιτεκτονική Κάρτας Γραφικών που υποστηρίζει CUDA

Σημαντικό βήμα για την ανάλυση του μοντέλου CUDA, είναι να αναφέρουμε τον τρόπο οργάνωσης της GPU που θα υποστηρίζει. Η GPU CUDA έχει την παρακάτω οργάνωση:

1. **Επεξεργαστικά Στοιχεία**

- **SM:** Η συστοιχία από πολυεπεξεργαστές συνεχούς ροής, οι λεγόμενοι SM (Streaming Multiprocessors – SM), διαχειρίζεται έναν πολύ μεγάλο αριθμό νημάτων.

- **SP:** Βασικό στοιχείο κάθε SM είναι οι SP (Streaming Processors) ή διαφορετικά επεξεργαστές συνεχούς ροής που μοιράζονται την κρυφή μνήμη των εντολών και τη μονάδα ελέγχου (control unit). Αντίστοιχα στο μοντέλο της πολυπύρηνης αρχιτεκτονικής οι SP αποτελούν τους επεξεργαστές, οι οποίοι διαχειρίζονται σε έναν μεγάλο αριθμό νημάτων. Ο αριθμός των SM διαφέρει από τη μία γενιά GPU CUDA στην άλλη.

  Κάθε SP διαθέτει μία μονάδα πρόσθεσης και μία πολλαπλασιασμού. Παράλληλα, διαθέτει και εξειδικευμένες μονάδες που εκτελούν τις floating point λειτουργίες όπως υπολογισμούς τετραγωνικών ριζών κτλ. Μέσα στα SP τα νήματα κάθε εφαρμογής πρέπει να οργανώνονται μαζικά και να εκτελούνται με την αντίστοιχη μέθοδο, ώστε να επιτυγχάνεται όσο το δυνατόν μεγαλύτερος βαθμός παραλληλίας.

2. **Στοιχεία μνήμης**

- **Καθολική Μνήμη**: Η καθολική μνήμη χρησιμοποιείται για ανάγνωση και εγγραφή δεδομένων και υπάρχει στην κάρτα γραφικών ως buffer (χρήση προσωρινής μνήμης) για frames γραφικών για τις εφαρμογές γραφικών, ενώ λειτουργεί σαν off-chip μνήμη (με μεγάλο εύρος ζώνης και λανθάνοντα χρόνο) όταν πρόκειται για άλλες υπολογιστικές εργασίες. Ο μεγάλος λανθάνων χρόνος αντιζυγιάζεται εν μέρη με την προσθήκη μνήμης cache στις κάρτες με διπλάσια υπολογιστική δυνατότητα. Η GPU έχει έως 4 GB DRAM μνήμης τύπου διπλού αριθμού δεδομένων γραφικών (Graphics Double Data Rate – GDDR) η οποία αποτελεί την καθολική μνήμη της κάρτας και είναι μοναδική από την κεντρική μνήμη (που υπάρχει στην μητρική κάρτα του host) του υπολογιστή της κεντρικής επεξεργασίας.

- **Μνήμη Σταθερών**: Η μνήμη σταθερών αποτελεί επίσης μια αργή μνήμη στην οποία προστέθηκε ένα cache και χρησιμοποιείται μόνο για ανάγνωση από τα νήματα. Παρέχει την δυνατότητα χρήσης της εντολής LoaD Uniform – LDU.

- **Μνήμη Υφών**: Η μνήμη υφών χρησιμοποιείται και αυτή από τα νήματα για ανάγνωση και εξυπηρετεί ειδικά πρότυπα προσπέλασης δεδομένων.

- **Κοινόχρηστη Μνήμη** (48 KB/MP): Η κοινόχρηστη μνήμη είναι ένα γρήγορο στοιχείο μνήμης, αφού είναι χωρισμένη σε υποστοιχεία μνήμης (banks) και απαιτεί ιδιαίτερη προσοχή για τυχόν συγκρούσεις. Χρησιμοποιείται από τα νήματα εντός του ίδιου μπλοκ για ανταλλαγή δεδομένων.

- **Τοπική μνήμη**: Η τοπική μνήμη είναι αργή μνήμη που δεν έχει cache, χρησιμοποιεί αυτόματο συγκερασμό (coalesce) προσπελάσεων για ανάγνωση και εγγραφή δεδομένων. Κυρίως χρησιμεύει για την αποθήκευση των δεδομένων όταν υπάρχει έλλειψη χώρου στους καταχωρητές.

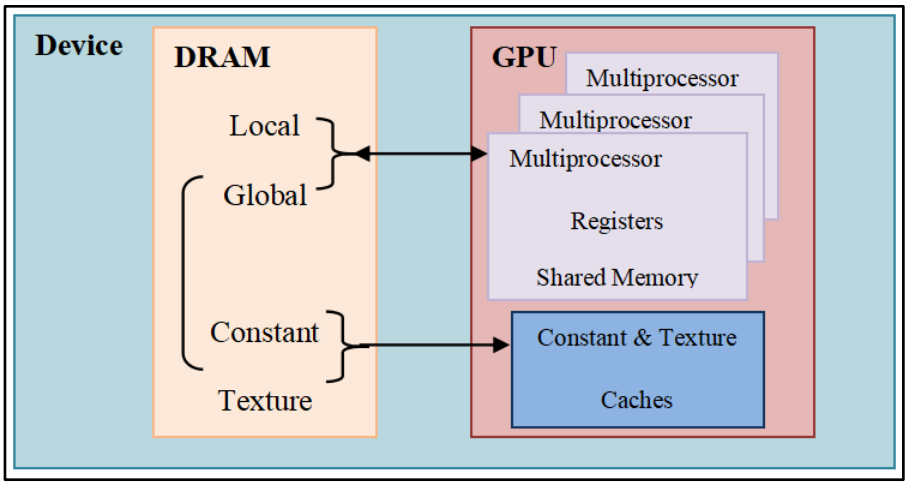
- **Καταχωρητές** (8192 – 16384 32-bit/MP): Οι καταχωρητές είναι τα πιο γρήγορα στοιχεία μνήμης και χρησιμοποιούνται από τα νήματα για ανάγνωση και εγγραφή (κάθε νήμα διαθέτει τον δικό του καταχωρητή).

3. Εύρος Ζώνης Επικοινωνίας / Μνήμης

- Το εύρος ζώνης επικοινωνίας μιας κάρτας γραφικών απασχολεί το διαθέσιμο εύρος για ανταλλαγή δεδομένων ανάμεσα της κάρτας και του κεντρικού συστήματος επεξεργασίας.
- Το εύρος ζώνης μνήμης αποτελεί το διαθέσιμο εύρος για ανταλλαγή δεδομένων της καθολικής μνήμης και των υπόλοιπων στοιχείων μέσα στα όρια της κάρτας γραφικών.
- Το εύρος ζώνης μνήμης είναι πολύ μεγαλύτερο από της επικοινωνίας.

4. Ιεραρχία Μνήμης

Το κεντρικό σύστημα εκτελεί τον κώδικα για κάθε νήμα που ορίζει η αρχιτεκτονική CUDA, με τα νήματα εκτελούνται σε μια κάρτα γραφικών, όπου διατηρεί τα δικά της ανεξάρτητα κυκλώματα μνήμης. Τα πιο γρήγορα στοιχεία μνήμης (η κοινή μνήμη και οι καταχωρητές) είναι τοποθετημένα στα κυκλώματα των SM (με χωρητικότητα μερικών KB). Τη μεγαλύτερη χωρητικότητα, της τάξης των GB και τους μεγαλύτερους χρόνους προσπέλασης διαθέτει η καθολική μνήμη, η οποία βρίσκεται πάνω στο κύκλωμα της κάρτας γραφικών, εκτός των ορίων SM.



Σχήμα 15: Οργάνωση της μνήμης σε κάρτα γραφικών.

4 ΚΕΦΑΛΑΙΟ 4ο: ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JULIA

Η Julia είναι μια γλώσσα προγραμματισμού με ανοικτό κώδικα, όπου σε υψηλό επίπεδο προγραμματισμού είναι απλή και σε χαμηλό επίπεδο αποδοτική. Αναφορικά είναι μια από της πιο σύγχρονες γλώσσες, που εμφανίστηκε στα μέσα του 2012, και διατηρεί σε πολύ καλό βαθμό τα διάφορα χαρακτηριστικά της ώστε να ανταποκρίνεται στις πολύπλοκες διεργασίες.

Η Julia είναι ένα από τα πιο ισχυρά εργαλεία που υπάρχουν και προσδίδει πολλές δυνατότητες σε κάθε επιστημονικό τομέα, όπως στην χρηματοοικονομική, ρομποτική, αεροδιαστημική, κα. Καθώς συνεισφέρει με την υψηλή αποδοτικότητα της, την υλοποίηση μεγάλων επιστημονικών δεδομένων και υπολογισμών. Έχει σχεδιαστεί έτσι ώστε να λειτουργεί σε κατανεμημένο και παράλληλο προγραμματισμό, μάλιστα αποτελείται από μια εκτενή βιβλιοθήκη μαθηματικών συναρτήσεων με μεγάλη αριθμητική ακρίβεια. Χαρακτηριστικό αυτής της γλώσσας είναι ότι είναι εύχρηστη για υπολογισμούς CFD και για παράλληλο προγραμματισμό και σε προγραμματισμό πάνω στις GPU. Από αυτή την ευελιξία αυτής της γλώσσας προγραμματισμού φαίνεται να αναπτύσσεται με μεγάλους ρυθμούς στην σημερινή ερευνητική εποχή.

Στην προκειμένη περίπτωση της διπλωματικής, η χρήση της γλώσσας Julia στην ενότητα της Υπολογιστικής Ρευστομηχανικής – CFD είναι η κατάλληλη. Προσφέρει τους απαραίτητους κώδικες, πολλές τεχνικές επίλυσης των προβλημάτων και την ευκολία σύνταξης του προγράμματος για υπολογιστικές εφαρμογές με μεγάλο πλήθος μετρήσεων και ακρίβειας. Οι κώδικες για CFD γράφονται σε Fortran και C/C#. Η γλώσσα της Julia προσεγγίζει στην απόδοση της Fortran μέσω μιας αυτόματης μετάφρασης των εξισώσεων σε έναν προσπελάσιμο κώδικα. Η δυνατότητα γλώσσας Julia να συγχωνεύει την υψηλή απόδοση με την παραγωγικότητα, την αντιπροσωπεύει ως την τέλεια επιλογή για ερευνητές που δουλεύουν σε διαφορετικούς επιστημονικούς κλάδους.

Συμβαδίζει με το πακέτο Pycall και του Matlab κα. για να ανταποκρίνεται στις λειτουργίες του κώδικα της Python. Οι βιβλιοθήκες και οι συναρτήσεις δεν απαιτούν API ή warper. Χρησιμοποιεί coroutines, cooperative, multitasking, ή one-shot continuations. Παράλληλα κατέχει ένα ισχυρό type system, έχει ικανοποιητική υποστήριξη στο Unicode, έχει ικανότητες μεταπρογραμματισμού και μακροεντολών τύπου Lisp και έχει ένα ενσωματωμένο διαχειριστή πακέτων (Pkg).

4.1 Πλατφόρμες Υποστήριξης και Εφαρμογές της Γλώσσας Julia

Υπάρχουν διάφορες πλατφόρμες στις οποίες οι κάρτες γραφικών υποστηρίζονται. Όλες οι εκδόσεις Julia είναι συμβατές με Linux και Windows και η λειτουργικότητα διατηρείται από τις πολλαπλές εφαρμογές και βιβλιοθήκες.

- NVIDIA CUDA: για προγραμματισμό πυρήνα σε χαμηλό και υψηλό επίπεδο πάνω σε συστοιχίες με διαθέσιμα πακέτα.

- Intel GPUs with oneAPI: μπορεί και προγραμματίζει τον πυρήνα αλλά δεν είναι διαθέσιμο ακόμα στις βιβλιοθήκες (oneMKL, oneDNN).

- AMD GPUs: είναι ακόμα σε πειραματικό στάδιο και εκτελείται πάνω στην στοίβα Rock. Προγραμματίζονται στη Julia σε επίπεδο πυρήνα αλλιώς χρησιμοποιούνται συστοιχίες.

Οι διάφορες εφαρμογές και βιβλιοθήκες της Julia στηρίζονται πάνω στις δυνατότητες των GPU:

- Flux.jl library: για μηχανική μάθηση.
- Yao.jl framework: για την έρευνα κβαντικών πληροφοριών.
- DiffEqGPU.jl ως μέρος των Differential Equations.jl ecosystem: για τη χρήση GPU σε απλές διαφορικές εξισώσεις.
- Oceananigans.jl: για την επιτάχυνση μιας μοντελοποιημένης εφαρμογής σε μη υδροστατικό ωκεανό.
- GPUifyLoops.jl και Kernel Abstractions.jl: για να δουλεύουν με CPU και GPU ομοίως να χρησιμοποιούν vendor-neutral abstractions.

Άλλες επίσης εφαρμογές και βιβλιοθήκες της Julia είναι μέσω ειδικών συστοιχιών για GPU, όπως το CuArray από το CUDA.jl ή το ROCArray από το AMDGPU.jl, με λογισμικό που χρησιμοποιεί της διεπαφές Julia array που μπορεί να εκτελεστεί σε μια κάρτα γραφικών.

4.2 Julia και Κάρτες Γραφικών – JuliaGPU

Βασικό εξάρτημα για την διεξαγωγή της διπλωματικής εργασίας είναι οι κάρτες γραφικών της NVIDIA. Όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο, μια κάρτα δέχεται δεδομένα από έναν CPU και τα μετατρέπει σε εικόνα. Η μονάδα GPU είναι σχεδιασμένη να εκτελεί πολύπλοκους μαθηματικούς και γεωμετρικούς υπολογισμούς για καλύτερη απόδοση.

Η γλώσσα προγραμματισμού Julia μέσα στις κάρτες γραφικών GPU, δείχνει μια πολύ καλή και λογική σκέψη. Η επίλυση του προβλήματος θα έχει κάθε φορά την βέλτιστη απόδοση. Αξιοποιεί τη σύνταξη υψηλού επιπέδου και έναν μεταγλωττιστή για τον προγραμματισμό των επιταχυντών.

5 ΚΕΦΑΛΑΙΟ 5ο : ΔΥΣΔΙΑΣΤΑΤΗ ΡΟΗ ΓΥΡΩ ΑΠΟ ΚΥΛΙΝΔΡΟ ΚΥΚΛΙΚΗΣ ΔΙΑΤΟΜΗΣ — ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

5.1 Βασικές Εξισώσεις CFD

Η κύρια ασχολία του CFD είναι η αριθμητική ανάλυση των φαινομένων δυναμικής ρευστού, η οποία αποτελεί διαφορικό λογισμό. Οι βασικές εξισώσεις που εντάσσονται στη δυναμική των ρευστών είναι οι εξισώσεις Navier – Stokes. Η CFD ουσιαστικά μετατρέπει τις μερικές διαφορικές εξισώσεις σε αλγεβρικές προς την αριθμητική επίλυση. Η πηγή αυτών των εξισώσεων προέρχεται από την υπόθεση ότι ένα σωματίδιο ρευστού παραμορφώνεται υπό διατμητική τάση. Αμέσως μετά, από τον δεύτερο νόμο κίνησης, η δυναμική του σωματιδίου περιγράφεται από τη μάζα, την ορμή και την ενέργεια του, καθώς αυτά τα σύνολα εξισώσεων αποτελούν τις εξισώσεις Navier – Stokes.

Η βάση των φυσικών ιδιοτήτων και φαινομένων των ρευστών προέρχεται από τη περιγραφή των εξισώσεων διατήρησης. Καθώς μπορούν και να διατυπωθούν σε πολλαπλές μορφές για κάθε περίπτωση: (1) για διάφορα συστήματα συντεταγμένων, (2) ως αναπόσπαστη μορφή για ένα πεπερασμένο όγκο ελέγχου, το οποίο χρησιμεύει ως σημείο εκκίνησης για αριθμητική μέθοδο, και (3) ως ολοκληρωματική μορφή ή και διαφορική (tensor) μορφή σε ένα Καρτεσιανό σύστημα αναφοράς. Οι εξισώσεις που αναφέρθηκαν, όπως θα αναπτυχθούν και παρακάτω, παραμένουν σταθερές μέσα στο κλειστό σύστημα:

1. Εξίσωση της Συνέχειας – Εξίσωση Διατήρησης της Μάζας

Σε αυτό το εδάφιο γίνεται η μελέτη της κίνησης του ρευστού. Εξεταζόμενο μέσα σε ένα μοντέλο σταθερού πεπερασμένου όγκου ελέγχου, η ροή διέρχεται μέσω αυτού. Από αυτό έχουμε ότι ο όγκος V και η επιφάνεια ελέγχου S δε μεταβάλλονται στο χρόνο, όπως η μάζα του ρευστού που μεταβάλλεται. Η ολοκληρωματική μορφή της εξίσωσης της Συνέχειας (Continuity Equation) είναι:

$$\frac{\partial}{\partial t}\iiint_V \rho \, dV + \iint_S \rho \left(\vec{V} * \overrightarrow{dS}\right) = 0 \quad [2]$$

Η Εξίσωση [1] είναι αποτέλεσμα της εφαρμογής της φυσικής μελετώμενης αρχής σε ακίνητο πεπερασμένο όγκο ελέγχου και χρησιμοποιείται σε πολλά ρευστοδυναμικά φαινόμενα σε πεπερασμένες επιφάνειες. Αυτή η μορφή εξίσωσης μπορεί να εφαρμοστεί σε οποιαδήποτε 3D μη μόνιμη ροή οποιοδήποτε ρευστού, σε ατριβή ή πραγματική και ασυμπίεστη ή συμπιεστή.

2. 2ος Νόμος του Νεύτωνα – Εξίσωση Διατήρησης της Ορμής

Στο εδάφιο αυτό διατυπώνεται ο 2ος Νόμος του Νεύτωνα, Εξίσωση [3], για να αναλυθεί ένας πεπερασμένος κι ακίνητος όγκο ελέγχου. Η έκφραση του ολικού ρυθμού μεταβολής της ορμής του ρευστού που διέρχεται από τον ακίνητο όγκο ελέγχου φαίνεται στην Εξίσωση [4].

$$\vec{F} = m\,\vec{a} = \frac{d}{dt}\left(m * \vec{V}\right) \quad [3]$$

$$\vec{F} = \frac{\partial}{\partial t} \iiint_V \rho \, \vec{V} \, dV + \oint\!\!\oint_S \left(\rho \, \vec{V} \, \overrightarrow{dS}\right)\vec{V} \quad [4]$$

$$\left[\frac{\partial}{\partial t} \iiint_V \rho \, \vec{V} \, dV\right] + \left[\oint\!\!\oint_S \left(\rho \, \vec{V} * \overrightarrow{dS}\right) \vec{V}\right] = \left[-\oint\!\!\oint_S p \, \overrightarrow{dS}\right] + \left[\iiint_V \rho \, \vec{f} \, dV\right] + \left[\vec{F}_{viscous}\right] \quad [5]$$

Η Εξίσωση [4] εκφράζει την ολοκληρωματική μορφή της Εξίσωσης της Ορμής που μπορεί να συνδέει αεροδυναμικά φαινόμενα σε μια πεπερασμένη περιοχή χώρου.

3. 1ος Νόμος της Θερμοδυναμικής – Εξίσωση Διατήρησης της Ενέργειας

Σε αυτό το εδάφιο, λαμβάνεται υπόψη ότι η πυκνότητα δεν είναι σταθερή.

$$\delta q + \delta w = de \quad [6]$$

Η Εξίσωση [6] αποτελεί την ολοκληρωματική μορφή του 1ου Θερμοδυναμικού Αξιώματος σε μια ροή ρευστού.

$$\left[\iiint_V (\dot{q}\,\rho)\, dV\right] + \left[\dot{Q}_{viscous}\right] - \left[\iint_S p\left(\vec{V} * \overrightarrow{dS}\right)\right] + \left[\iiint_V \left(\rho \, \vec{f} \, \vec{V}\right) dV\right] + \left[\dot{W}_{viscous}\right]$$
$$=$$
$$\frac{\partial}{\partial t}\left\{\iiint_V \left[\rho\left(e + \frac{V^2}{2}\right) dV\right]\right\} + \iint_S \left[\left(\rho\left(\vec{V}\,\overrightarrow{dS}\right)\right) * \left(e + \frac{V^2}{2}\right)\right] \quad [7]$$

4. Οι Εξισώσεις Navier-Stokes

Οι εξισώσεις Navier Stokes είναι ένα σύστημα μερικών διαφορικών εξισώσεων που περιγράφουν την κίνηση των ρευστών και προκύπτουν από την εφαρμογή νόμων διατήρησης για τη μάζα, την ορμή και την ενέργεια, λαμβάνοντας όμως υπόψη ότι η συνολική πίεση στο ρευστό προκύπτει από τη στατική πίεση και από τις διατμητικές τάσεις που υφίστανται λόγω της ιξώδους ροής.

Για μη συμπιεστά ρευστά και για ιξώδες μ σταθερό σε όλο το χώρο ισχύει η σχέση:

$$\rho \frac{D\vec{V}}{Dt} = \rho\vec{f} - \nabla p + \mu \nabla^2 \vec{V} \quad [8]$$

5.2 Αριθμός Prandtl – Pr

Η περιγραφή του μηχανισμού της μεταφοράς θερμότητας στα ρευστά πραγματοποιείται από τη θεωρία του οριακού στρώματος. Ο αριθμός Prandtl (Pr) ενός ρευστού χαρακτηρίζει το νόημα της ορμής του οριακού στρώματος στο θερμικό οριακό στρώμα, στη μεταφορά θερμότητας. Ο αριθμός Pr προσδιορίζεται αποκλειστικά από τις ιδιότητες του ρευστού.

Ο αριθμός Prandtl είναι μια αδιάστατη ποσότητα που θέτει το ιξώδες ενός υγρού σε συσχέτιση με τη θερμική αγωγιμότητα. Επομένως, αξιολογεί τη σχέση μεταξύ μεταφοράς ορμής και θερμικής ικανότητας μεταφοράς ενός ρευστού. Και ορίζεται ως εξής:

$$Pr = \frac{v}{a} = \frac{\eta}{\rho\,\alpha} = \frac{\eta\,c_p}{\lambda} = \frac{Μεταφορά\ Ορμής}{Μεταφορά\ Θερμότητας} \quad [9]$$

Με τη χρήση της θερμικής διάχυσης α που ορίζεται ως:

$$\alpha = \frac{\lambda}{\rho\,c_p} \quad [10]$$

Τα ρευστά με μικρούς αριθμούς Prandtl είναι υγρά ελεύθερης ροής με υψηλή θερμική αγωγιμότητα, Θερμοαγώγιμα υγρά. Στα προβλήματα θερμικής ροής, η μεταφορά θερμότητας ελέγχεται από το συνδυασμό του αριθμού Prandtl και του αριθμού Reynolds.

5.3 Αριθμός Reynolds – Re

Μία από τις πιο σημαντικές αδιαστασιακές παραμέτρους είναι ο αριθμός Reynolds (Re), που περιγράφει τα χαρακτηριστικά ροής και ορίζεται ως ο λόγος της δύναμης αδράνειας έναντι της δύναμης τριβής.

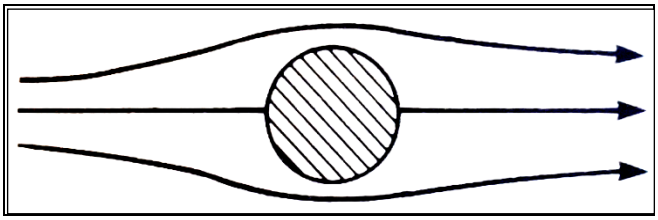
$$Re = \frac{Δύναμη\ Αδράνειας}{Δύναμη\ Τριβής} = \frac{\rho * V * D}{\mu} \quad [11]$$

Παρατηρείται πως ενσωματώνει την πυκνότητα, την ταχύτητα εισόδου και το δυναμικό ιξώδες. Η κύρια χρήση του αριθμού Reynolds είναι να αποδεικνύει εάν η ροή είναι στρωτή ή τυρβώδης. Σε χαμηλό αριθμό Reynolds < 2300, οι δυνάμεις αδράνειας είναι μικρότερες από τις ιξώδεις δυνάμεις και έτσι οι διαταραχές διαχέονται με τη ροή να είναι στρωτή. Σε υψηλό αριθμό Reynolds > 2300, οι δυνάμεις αδράνειας είναι αρκετά μεγάλες για να ενισχύσουν τις διαταραχές και συμβαίνει μια μετάβαση σε αναταράξεις. Εδώ, γίνεται ασταθής η κίνηση ακόμη και με σταθερές επιβαλλόμενες οριακές συνθήκες. Η ταχύτητα και όλες οι άλλες ιδιότητες ροής ποικίλλουν με τυχαίο και χαοτικό τρόπο, η ροή είναι τυρβώδης.

5.4 Απλή Εφαρμογή των Βασικών Εξισώσεων

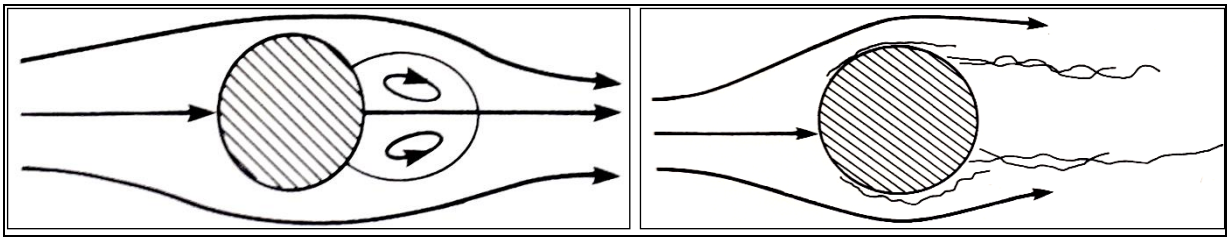
Στα πλαίσια μελέτης αυτής της εργασίας θα διατυπωθεί το φαινόμενο της ροής γύρω από έναν κύλινδρο μέσω της γλώσσας προγραμματισμού Julia. Πριν όμως εξεταστεί το υπολογιστικό μέρος της εργασίας θα αναλυθεί το θεωρητικό μέρος: πως δηλαδή επηρεάζεται η ροή γύρω από έναν εξεταζόμενο κύλινδρο.

Από τη θεωρία είναι γνωστό πως τα προαναφερόμενα μεγέθη εφαρμόζονται στην ροή γύρω από τον κύλινδρο, έτσι όπως αναμένεται. Αρχικά η ροή ενός μη συνεκτικού ρευστού γύρω από έναν κύλινδρο γίνεται με την μέθοδο της επαλληλίας, όπως απεικονίζεται στην **Εικόνα 22**. Το πεδίο της ροής είναι συμμετρικό στο πάνω και κάτω ήμισυ, όπως και η κατανομή της πίεσης εξισορροπείται στον οριζόντιο και κάθετο άξονα από το κέντρο του κυλίνδρου. Εφόσον η κατανομή της πίεσης στον οριζόντιο άξονα είναι συμμετρική δεν έχουμε επίδραση της δύναμης της Άντωσης (Lift). Αντίστοιχα για τον κατακόρυφο άξονα, η κατανομή της πίεσης αφού είναι συμμετρική, δεν υπάρχει άσκηση της Οπισθέλκουσας (Drag) δύναμης πάνω στο σώμα.



Εικόνα 22: Μέθοδος της Επαλληλίας εφαρμοσμένη γύρω από τον κύλινδρο.

Πέρα από το ιδεατό μέρος του φαινομένου αυτού, επιπλέον λαμβάνεται και η πραγματική συμπεριφορά των δυνάμεων συναρτήσει του αριθμού Reynolds. Όσο μεγαλύτερος είναι ο αριθμός Reynolds τόσο μικρότερη είναι το εύρος των επιδράσεων της τριβής. Αριθμοί Reynolds μικρότεροι από τη μονάδα, το εύρος της ροής, που εκεί επιδρά η συνεκτικότητα είναι μεγάλο. Έτσι δημιουργούνται και τα ανάλογα απορρεύματα στροβιλισμών (von Karmann vortex street). Πυκνοί στροβιλισμοί υπάρχουν ακριβώς πίσω από το σώμα και όσο απομακρυνόμαστε από αυτό, αραιώνουν, λόγω της ιξώδους διάχυσης. Όταν ο αριθμός Reynolds αυξάνεται, η μπροστινή περιοχή του κυλίνδρου μειώνεται αρκετά, όπως αποτυπώνεται στην **Εικόνα 23α**. Επιπρόσθετα στο ίδιο σχήμα παρατηρείται και το σημείο αποκόλλησης, με τις δυνάμεις αδράνειας να είναι πολύ μεγάλες, όπου τα σωματίδια του ρευστού δεν μπορούν να συνεχίσουν την ίδια διαδρομή της επιφάνειας του κυλίνδρου. Τέλος παρατηρείται μετά από μερική απόσταση από τον κύλινδρο δημιουργείται μια περιοχή με τα σωματίδια του ρευστού να κινούνται αντίθετα της φοράς της αρχικής ροής και μετά περιφερειακά μέσα της.

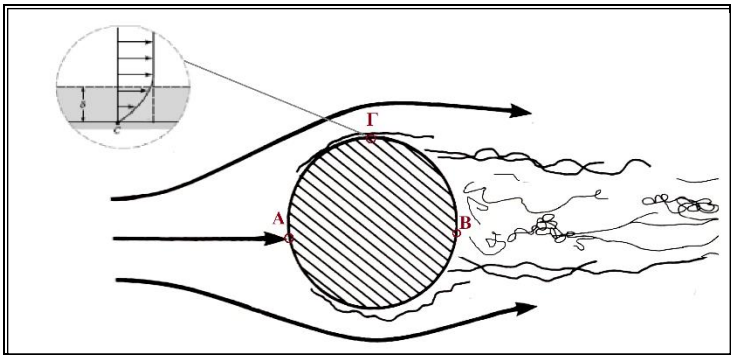


(α) **(β)**

Εικόνα 23: (α) δημιουργία δύο διακριτών δινών, **(β)** αποκόλληση ροής.

Φαίνεται πως η περιοχή του αριθμού Reynolds είναι αυξημένη στο μπροστινό μέρος του κυλίνδρου, οι συνεκτικές λοιπόν εκεί επιδράσεις είναι μειωμένες. Επίσης παρατηρείται πως το σημείο αποκόλλησης με τις δυνάμεις αδράνειας να είναι πολύ μεγάλες και τα σωματίδια του ρευστού δεν συνεχίζουν πάνω στην επιφάνεια του κυλίνδρου, **Εικόνα 23β**. Μετέπειτα, από κάποια απόσταση μακριά από το σώμα σχηματίζεται περιοχή με σωματίδια ρευστού που κινούνται αντίθετα και μετά περιφερειακά μέσα της περιοχής. Για πολύ ακόμα μεγαλύτερους αριθμούς Reynolds, η περιοχή των συνεκτικών επιδράσεων συμπυκνώνεται στην περιοχή κατάντη τον κύλινδρο, αφού είναι μια λεπτή περιοχή του οριακού στρώματος δ<<D, λόγω της τριβής πάνω στον κύλινδρο.

Στη παρακάτω **Εικόνα 24** διατυπώνεται ο μηχανισμός αποκόλλησης στην ροή του σωματιδίου του ρευστού πάνω στην επιφάνεια του κυλίνδρου.



Εικόνα 24: Μηχανισμός αποκόλλησης της ροής των σωματιδίων του ρευστού.

Το σωματίδιο του ρευστού κινείται μέσα στην περιοχή του οριακού στρώματος, από το Α σημείο προς Β σημείο, κατανέμει την ίδια πίεση με τα σωματίδια του ορίου του οριακού στρώματος, με άλλα λόγια η κατανομή της πίεσης της μη συνεκτικής ροής. Εφόσον, έχουμε στην περιοχή κοντά στην επιφάνεια του κυλίνδρου τριβή, το σωματίδιο ρευστού που κινείται μέσα στο οριακό στρώμα, έχει απώλειες της κινητικής του ενέργειας. Αυτό το έλλειμμα ενέργειας, έχει ως αποτέλεσμα το ρευστό να μην έχει αρκετή ενέργεια για να ξεπεράσει την διαρκεί αυξημένη στατική πίεση από το σημείο Γ έως το Β. Επίσης, η απώλεια κινητικής ενέργειας φαίνεται και στο προφίλ ταχύτητας στο Γ σημείο. Από αυτό, το σωματίδιο ρευστού δεν μπορεί να φτάσει μέχρι το Β σημείο. Με συνέπεια, να έχουμε αποκόλληση (separation) του ρευστού από την επιφάνεια του κυλίνδρου. Από αυτό αναπτύσσεται η πεπερασμένη μορφή οπισθέλκουσας στο σώμα (παράδοξο d'Alembert). Στη πίσω πλευρά του σώματος η μέση πίεση είναι αρκετά μικρότερη από τη μέση πίεση στη μπροστά πλευρά. Αρά δημιουργείται μια δύναμη αντίστασης (pressure Drag) από τη διαφορά πίεσης, ενώ η δύναμη αντίστασης λόγω τριβής (friction Drag) είναι πολύ μικρότερη. Από τις μικρές κλίσεις της ταχύτητας (velocity gradients), το ρευστό είναι συνεκτικό και η ροή ατριβής στο σημείο έξω από την συνεκτική περιοχή. Από τις πολύ μεγάλες κλίσεις τώρα, στην συνεκτική περιοχή εμφανίζουν διατμητικές τάσεις, όταν πολλαπλασιαστεί με το δυναμικό ιξώδες του ρευστού (παράδειγμα στην εξίσωση Navier Stokes). Σημαντικό είναι να τονιστεί ότι οι πιο πολλές πραγματικές ροές έχουν υψηλούς αριθμούς Reynolds. Για να μελετήσει κανείς αυτές τις ροές πρέπει να χωρίσει το περιβάλλον της ροής σε δύο πεδία, ώστε να είναι πιο απλή η μελέτη. Το πρώτο πεδίο καλύπτει την ροή με τις έντονες επιδράσεις συνεκτικών παραμέτρων. Το δεύτερο πεδίο, τη μη συνεκτική ροή. Η μεγάλη αυτή ανάπτυξη των ηλεκτρονικών υπολογιστών δίνει την δυνατότητα στην σημερινή εποχή μας, την εξερεύνηση όλης της ροής ως συνεκτική. Ο χρόνος επεξεργασίας και υλοποίησης και γενικά όλης της διαδικασίας είναι εξαιρετικά μειωμένος.

6 ΚΕΦΑΛΑΙΟ 6ο : ΔΙΣΔΙΑΣΤΑΤΗ ΡΟΗ ΓΥΡΩ ΑΠΟ ΚΥΛΙΝΔΡΟ ΚΥΚΛΙΚΗΣ ΔΙΑΤΟΜΗΣ — ΥΠΟΛΟΓΙΣΤΙΚΟ ΜΕΡΟΣ

Στη παρούσα εργασία πραγματοποιήθηκαν δισδιάστατες προσομοιώσεις ροής αέρα γύρω από κύλινδρο κυκλικής διατομής μέσα σε κανάλι ορθογωνικής διατομής. Σκοπός των προσομοιώσεων ήταν η μελέτη και η ανάδειξη ταχύτερων και ακριβέστερων μεθόδων επίλυσης των προσομοιώσεων σε σύγχρονα υπολογιστικά συστήματα.

Υλοποιήθηκε κατάλληλος προγραμματιστικός κώδικας σε γλώσσα προγραμματισμού Julia για την περιγραφή του μοντέλου των προσομοιώσεων και για την επίλυση των υπολογιστικών πλεγμάτων των αντίστοιχων γεωμετριών με τη μέθοδο των πεπερασμένων διαφορών.

Το δισδιάστατο μοντέλο των προσομοιώσεων περιγράφει τον υπολογισμό της Ταχύτητας Vy της ροής του αέρα, την Στροβιλότητα C του αέρα, τον αριθμό Prandtl Pr, καθώς το χρόνο και την απόδοση επίλυσης του κώδικα. Για την περιγραφή του δισδιάστατου μοντέλου χρησιμοποιήθηκαν οι εξισώσεις Navier-Stokes.

Επίσης ορίστηκαν οι οριακές συνθήκες του μοντέλου ως εξής, στα τοιχώματα του καναλιού και του κυλίνδρου ορίστηκε συνθήκη μη ολίσθησης, ορίστηκε στην είσοδο του καναλιού σταθερή και ομοιόμορφη ταχύτητα ροής, και στην έξοδο του καναλιού ορίστηκε μηδενική πίεση.

Ως δεδομένα του μοντέλου ορίστηκαν τα παρακάτω:

lx (m)	0.6
ly (m)	1
ρ (kg/m^3)	1
μ (kg/m sec)	1.00E-04

Για την μελέτη και την ανάδειξη ταχύτερων και ακριβέστερων μεθόδων επίλυσης των προσομοιώσεων σε σύγχρονα υπολογιστικά συστήματα πραγματοποιήθηκαν οι παρακάτω παραμετροποιήσεις με σταδιακή αύξηση του αριθμού Reynolds και επιπλέον με σταδιακή πύκνωση του πλέγματος του καναλιού.

Vin (m/sec)	0.01	1	100
Re	1.00E+02	1.00E+04	1.00E+06

ny	255	510	1020
nx	153	306	612

Οι προσομοιώσεις εκτελέστηκαν σε υπολογιστή του εργαστηρίου Ανάλυσης και Προσομοίωσης Ροών του τμήματος Μηχανολόγων Μηχανικών του ΠΑΔΑ. Ο υπολογιστής του εργαστηρίου διέθετε, λειτουργικό σύστημα Linux, επεξεργαστή Intel(R) Core(TM) i9-9900 CPU @ 3.10GHz με 8 πυρήνες και συνολικά 16 Threads καθώς επίσης 32GB μνήμης RAM και μία κάρτα γραφικών GPU NVIDIA GeForce RTX 2070 SUPER.

Οι παραμετροποιήσεις των προσομοιώσεων πραγματοποιήθηκαν για τις περιπτώσεις, 1 πυρήνα της CPU σειριακά, παράλληλη εκτέλεση σε 8 πυρήνες της CPU, και παράλληλη εκτέλεση στην κάρτα γραφικών GPU.

Οι επεξεργαστές πολλών πυρήνων ως GPU είναι συστήματα προσανατολισμένα στην απόδοση που χρησιμοποιούν τον τεράστιο παραλληλισμό τους για να κρύψουν την καθυστέρηση. Από την πλευρά των επιστημονικών εφαρμογών, οι περισσότεροι αλγόριθμοι απαιτούν μόνο λίγες πράξεις ή flops σε σύγκριση με τον αριθμό των αριθμών ή των byte που έχουν πρόσβαση από την κύρια μνήμη, και επομένως δεσμεύονται σημαντικά στη μνήμη. Η μέτρηση Flop/s δεν είναι πλέον η πλέον κατάλληλη για την αναφορά της απόδοσης της εφαρμογής πολλών σύγχρονων εφαρμογών. Αυτή η κατάσταση ανέπτυξε μια μέτρηση αξιολόγησης της απόδοσης με βάση τη διεκπεραίωση μνήμης T_eff, για να αξιολογηθεί η απόδοση επαναληπτικών stencil επιλυτών.

Η αποτελεσματική πρόσβαση στη μνήμη, A_{eff} [GB], είναι το άθροισμα του διπλάσιου αποτυπώματος μνήμης των άγνωστων πεδίων, D_u, (πεδία που εξαρτώνται από το δικό τους ιστορικό και που πρέπει να ενημερώνονται σε κάθε επανάληψη) και των γνωστών πεδίων, D_k, που δεν άλλαξε σε κάθε επανάληψη. Η πραγματική πρόσβαση στη μνήμη διαιρούμενη με τον χρόνο εκτέλεσης ανά επανάληψη, t_{it} [sec], ορίζει την ενεργή απόδοση μνήμης, T_{eff} [GB/sec].

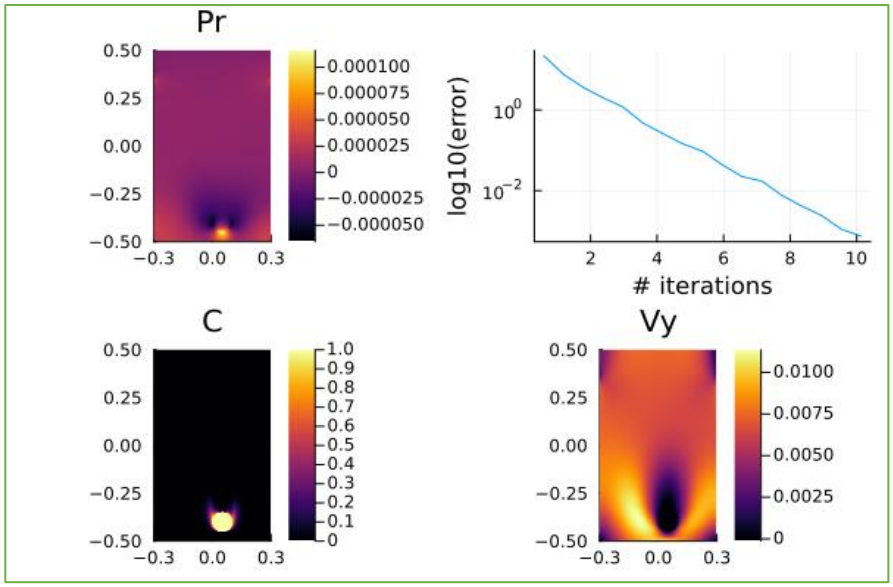
$$A_{eff} = 2D_u + D_k$$

$$T_{eff} = \frac{A_{eff}}{t_{it}}$$

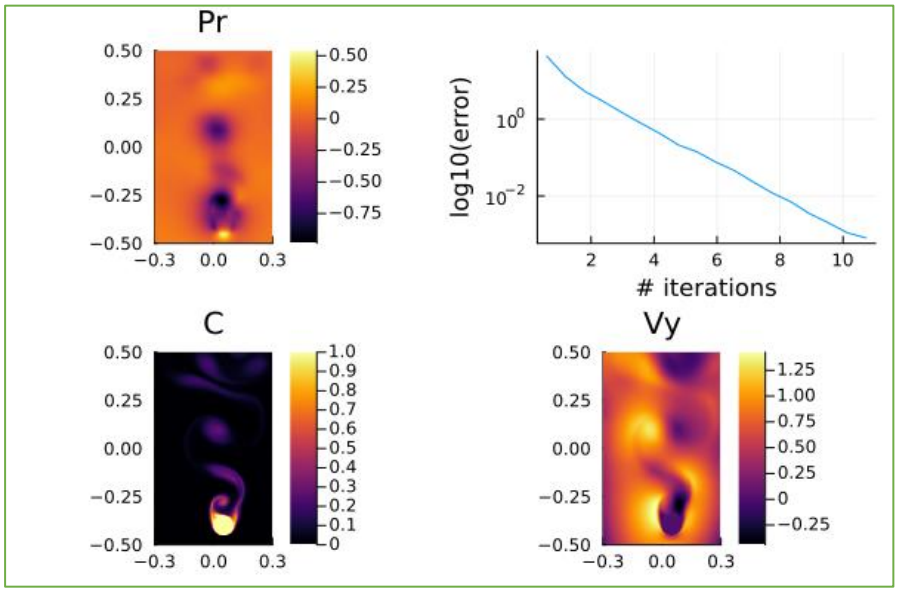
Ο προγραμματιστικός κώδικας που υλοποιήθηκε για τις προσομοιώσεις παρατίθεται στο ΠΑΡΑΡΤΗΜΑ Α΄ της παρούσας εργασίας.

ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

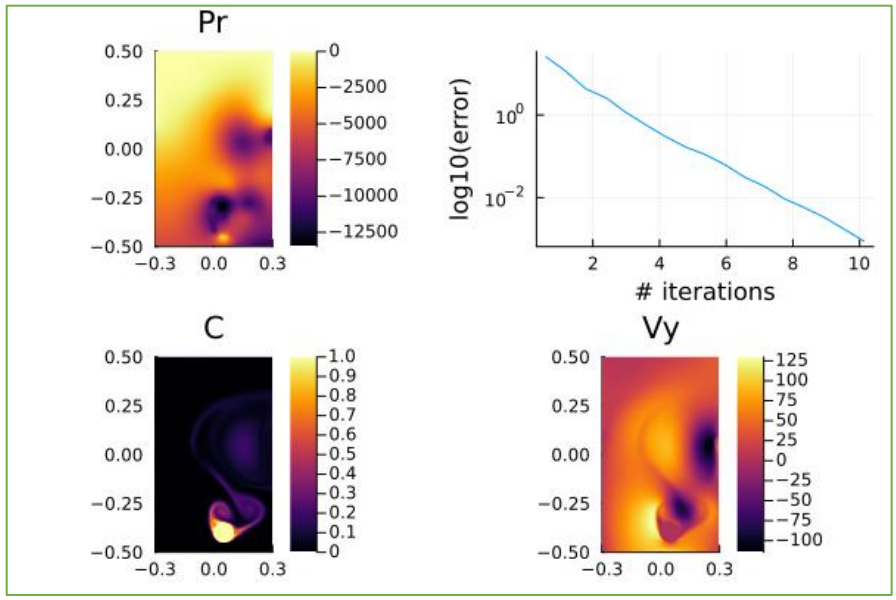
ny	255
nx	153
Re	1.00E+02



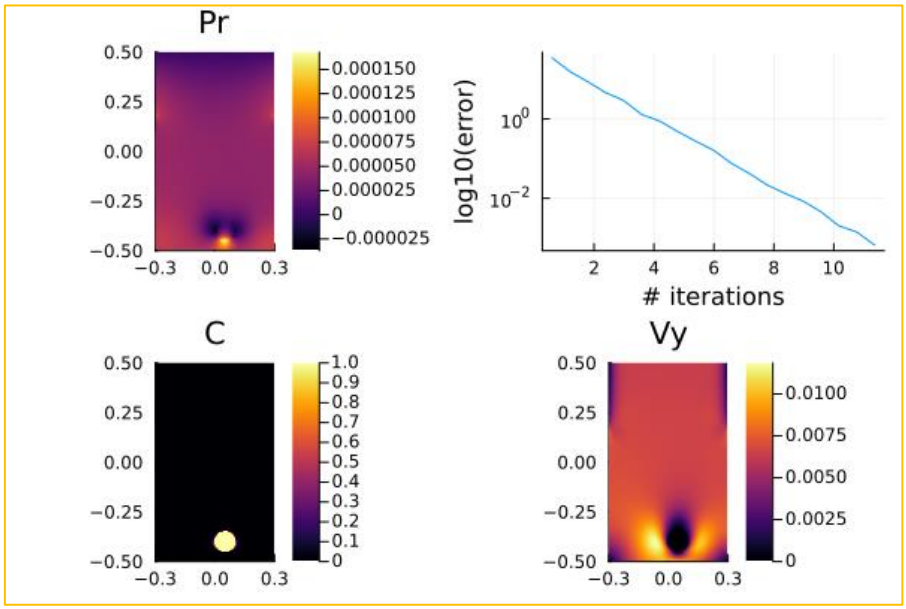
ny	255
nx	153
Re	1.00E+04



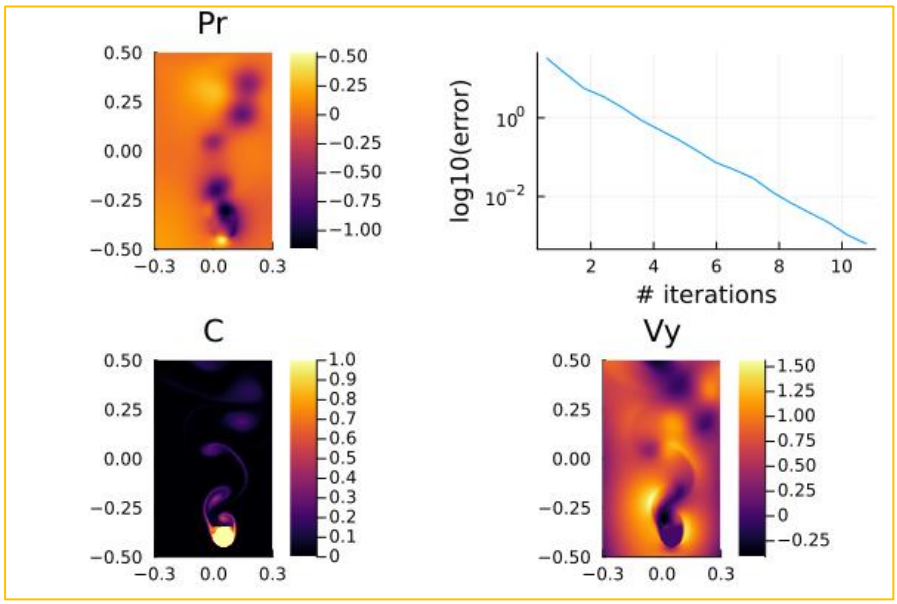
ny	255
nx	153
Re	1.00E+06



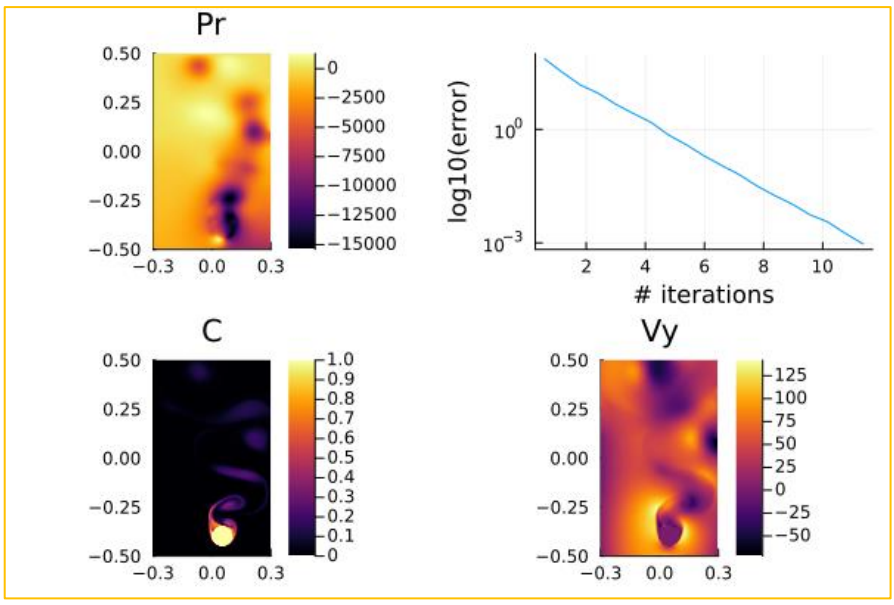
ny	510
nx	306
Re	1.00E+02



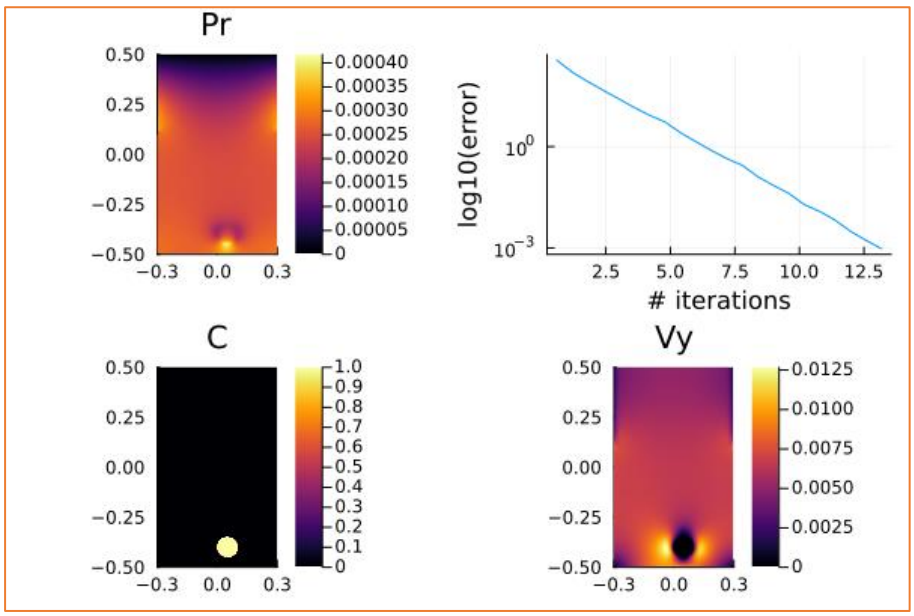
ny	510
nx	306
Re	1.00E+04



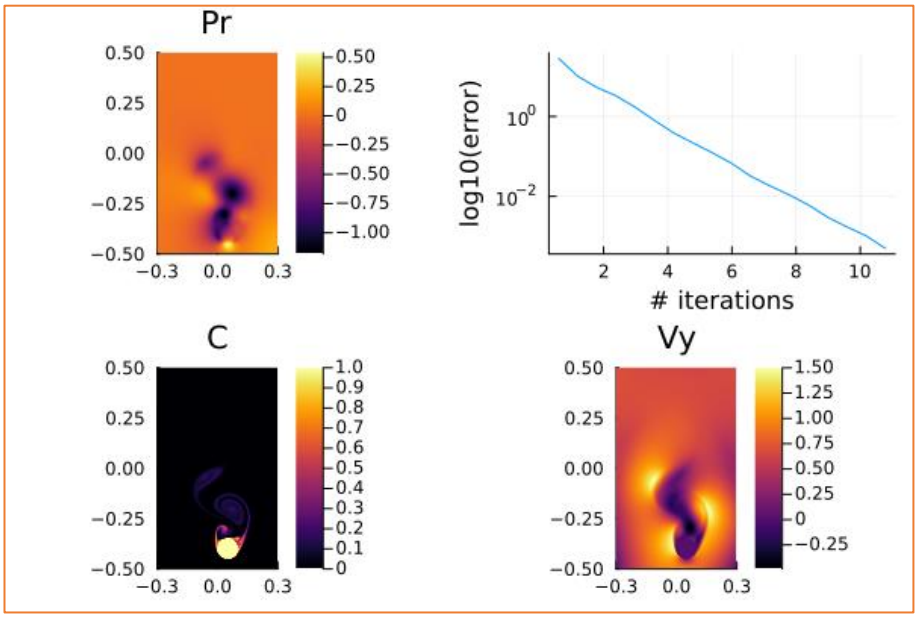
ny	510
nx	306
Re	1.00E+06



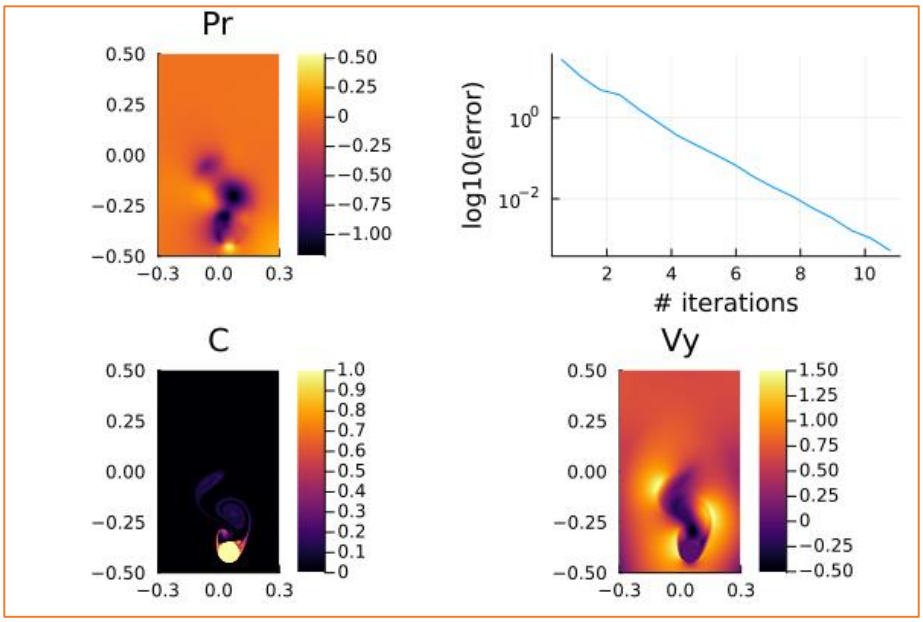
ny	1020
nx	612
Re	1.00E+02



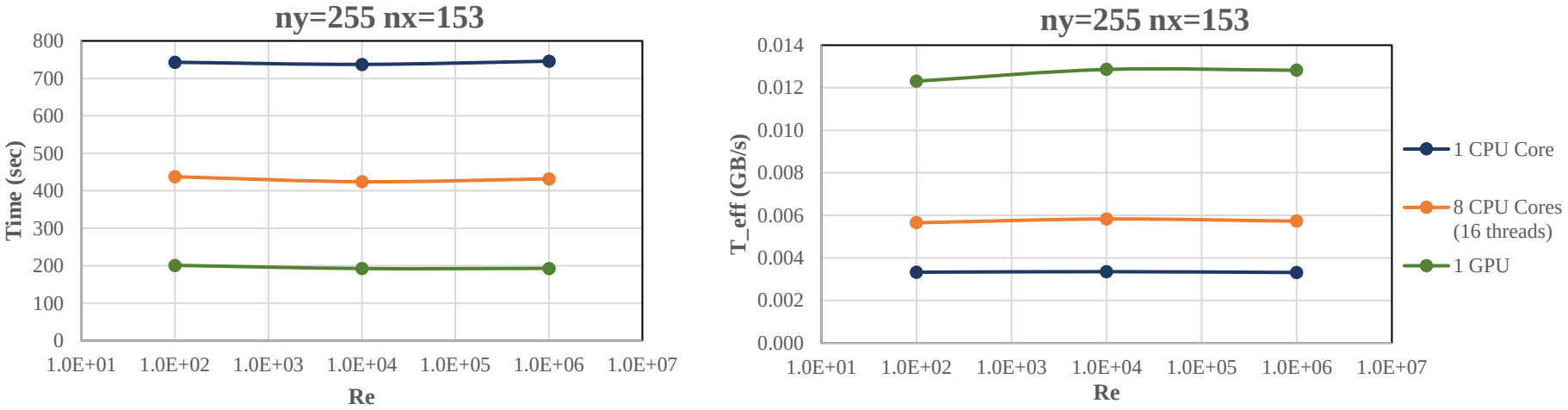
ny	1020
nx	612
Re	1.00E+04



ny	1020
nx	612
Re	1.00E+06

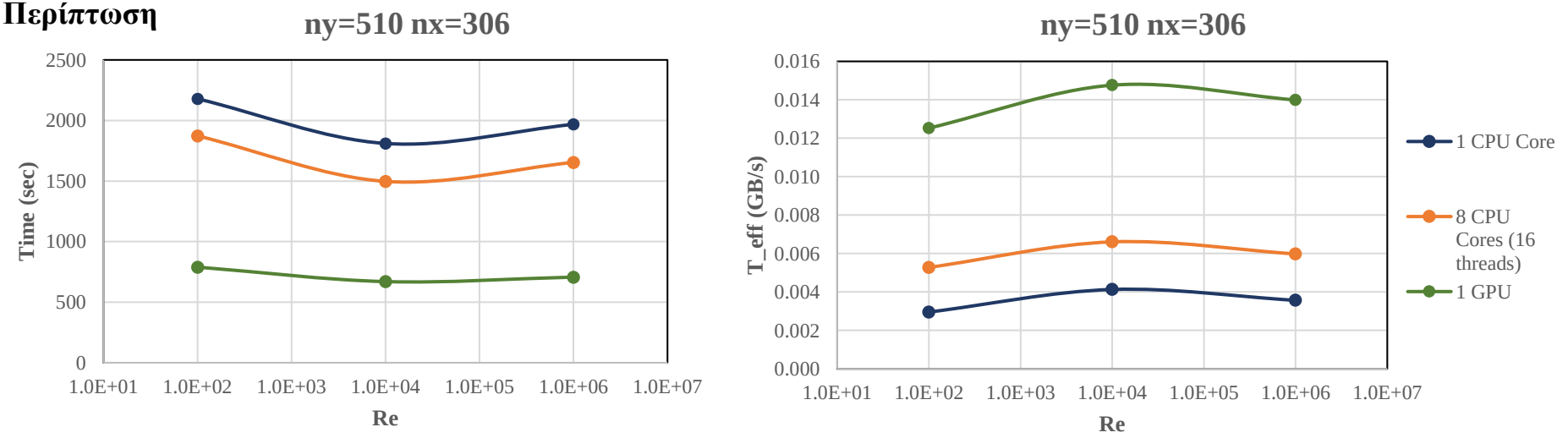


A' Περίπτωση



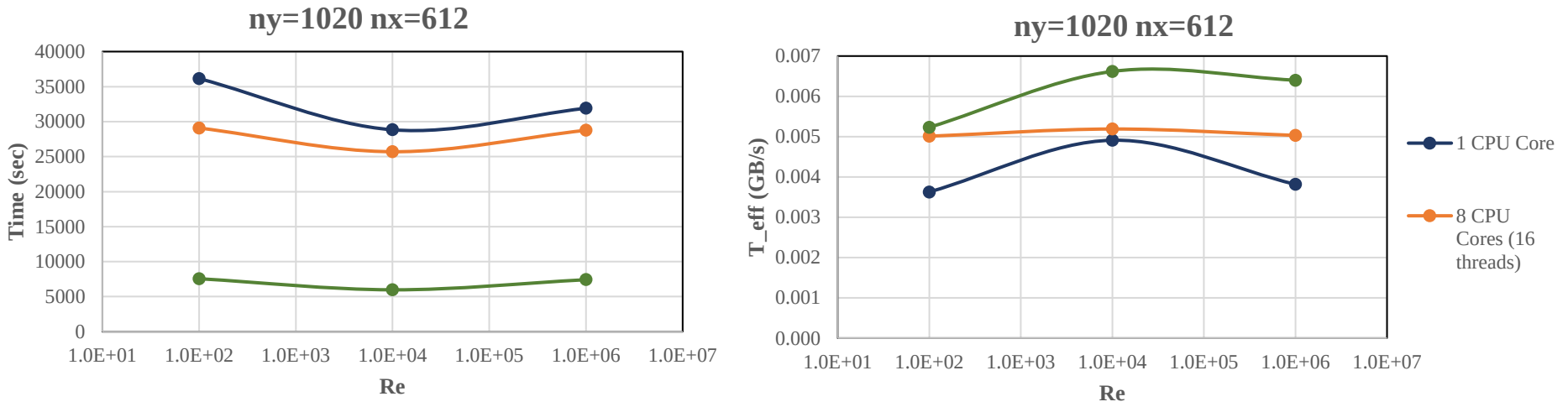
Διάγραμμα 1: Εκτέλεση κώδικα με αραιό πλέγμα για τους τρεις διαφορετικούς αριθμούς Re πάνω σε 1 CPU, 8 CPU και GPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

B' Περίπτωση



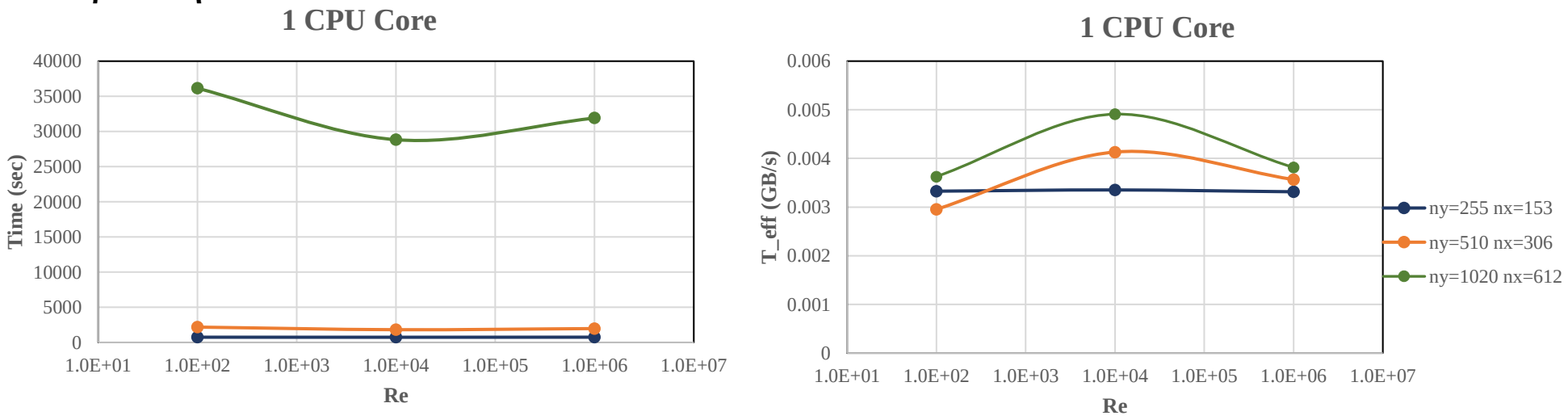
Διάγραμμα 2: Εκτέλεση κώδικα με πιο πυκνό πλέγμα για τους τρεις διαφορετικούς αριθμούς Re πάνω σε 1 CPU, 8 CPU και GPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

Γ' Περίπτωση



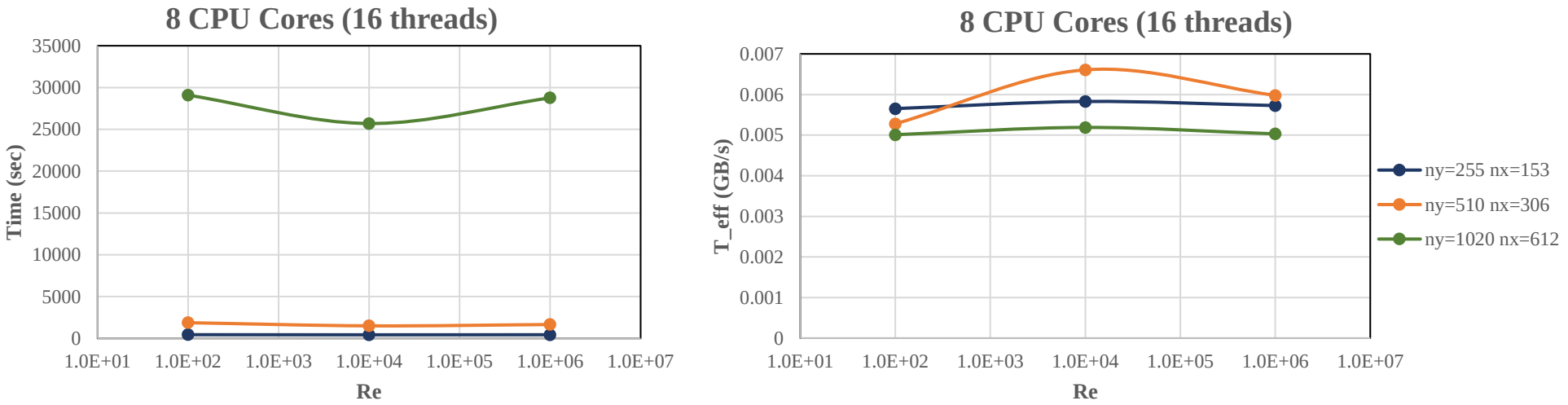
Διάγραμμα 3: Εκτέλεση κώδικα με πυκνότερο πλέγμα για τους τρεις διαφορετικούς αριθμούς Re πάνω σε 1 CPU, 8 CPU και GPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

Ι Περίπτωση



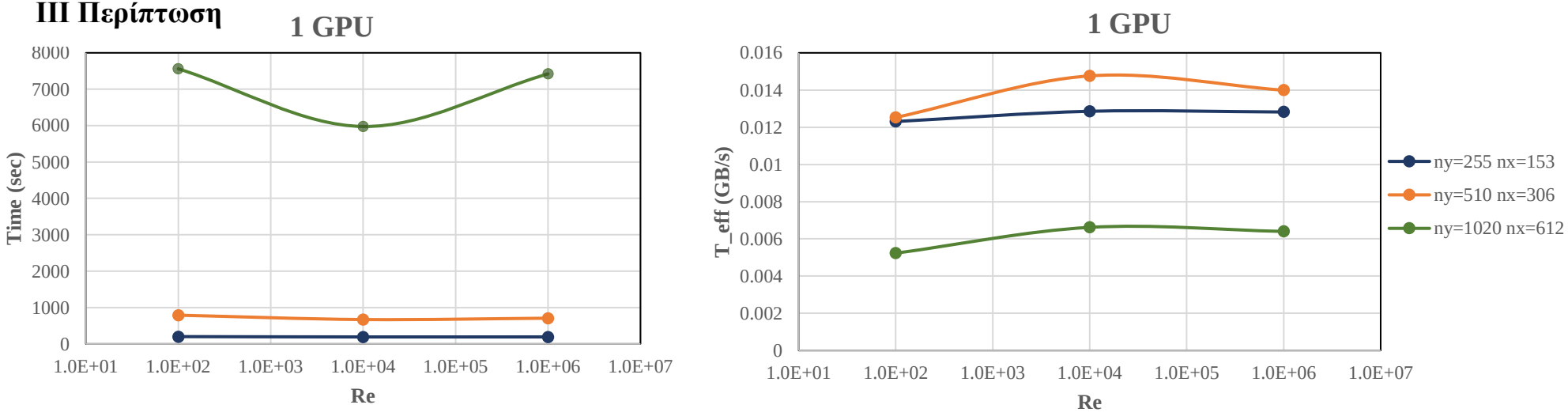
Διάγραμμα 4: Εκτέλεση κώδικα για τα τρία πλέγματα και για τους τρεις διαφορετικούς αριθμούς Re πάνω σε 1 CPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

II Περίπτωση



Διάγραμμα 5: Εκτέλεση κώδικα για τα τρία πλέγματα και για τους τρεις διαφορετικούς αριθμούς Re πάνω σε 8 CPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

III Περίπτωση



Διάγραμμα 6: Εκτέλεση κώδικα για τα τρία πλέγματα και για τους τρεις διαφορετικούς αριθμούς Re πάνω σε GPU. Χρόνος εκτέλεσης (αριστερά), Απόδοση (δεξιά).

7 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η ραγδαία ανάπτυξη της τεχνολογίας έχει προσφέρει αμέτρητες καινοτομίες σε όλους τους επιστημονικούς κλάδους. Μέσα από τα άπειρα δεδομένα και τους μεγάλους χρόνους αφιέρωσης εργασιών, τα υπολογιστικά εξαρτήματα (συνήθως pc hardware) έχουν αναπαύσει από κάθε έννοιας τους ερευνητές.

Η σωστή επιλογή των μεγεθών και παραμέτρων επιτρέπει στον χρήστη να έχει μια πιο ρεαλιστική αναπαράσταση του φυσικού φαινομένου. Ο παράλληλος προγραμματισμός διατυπώνει ακριβώς την χωροταξική λογική των υπολογιστικών στοιχείων. Στην προκειμένη περίπτωση βλέπουμε πως το μοντέλο παράλληλης επεξεργασίας, CUDA, υποστηρίζει τον παράλληλο προγραμματισμό σε κάρτες γραφικών GPU. Το μοντέλου CUDA γραμμένο στην γλώσσα προγραμματισμού Julia, υλοποιήθηκε στη κάρτα GPU και φαίνεται να έχει αξιοσημείωτα πλεονεκτήματα.

Όπως έχουμε ήδη αναφέρει, η επαναληπτική διαδικασία πραγματοποιείται για τη μείωση των σφαλμάτων και από τον ορισμό του μοντέλου επίλυσης των περασμένων διαφορών. Τα σφάλματα δεν μηδενίζονται αλλά ελαττώνονται γιατί έχουμε επίλυση σε πυκνότερο πλέγμα. Όσο πιο πυκνό πλέγμα τόσο πιο ακριβή τιμή αποτελέσματος και μεγαλύτερος υπολογιστικός χρόνος επίλυσης του μοντέλου. Στα διαγράμματα Σφάλματα - Επαναλήψεις βλέπουμε πως ελαττώνεται το σφάλμα ανάλογα με τις επαναλήψεις και την τιμή του σφάλματος σύγκλισης που ορίσαμε στον κώδικα. Από τον ορισμό του μοντέλου επίλυσης των περασμένων διαφορών τα σφάλματα ελαττώνονται σύμφωνα με τον κώδικα που θέτουμε. Όσο πιο πυκνό πλέγμα τόσο πιο ακριβή τιμή αποτελέσματος, μεγαλύτερη λεπτομέρεια εικόνας και εμφάνισης των εξεταζόμενων μεγεθών (Vy, C, Pr) κατάντη του σώματος και τόσο μεγαλύτερος υπολογιστικός χρόνος επίλυσης του μοντέλου. Όσες περισσότερες επαναλήψεις εκτελούμε τόσο λιγότερο σφάλμα και μεγαλύτερη σύγκλιση έχουμε. Στα διαγράμματα Σφάλματα - Επαναλήψεις βλέπουμε πως ελαττώνεται το σφάλμα ανάλογα με τις επαναλήψεις και την τιμή του σφάλματος σύγκλισης που ορίσαμε στον κώδικα.

Στην Α', Β' και Γ' Περίπτωση: παρατηρούμε ότι όσο πυκνώνουμε το πλέγμα τόσο επιμηκύνεται ο χρόνος εκτέλεσης, ιδιαίτερα για τα στοιχεία CPU. Η GPU από την άλλη, βλέπουμε ότι χρειάζεται παραπάνω χρόνο ολοκλήρωσης, αλλά και στις τρεις περιπτώσεις αυτές, νικάει χρονικά με διαφορά τα υπόλοιπα στοιχεία. Ταυτόχρονα, η τιμή της απόδοσης είναι χαμηλότερη με την πύκνωση, παρ' όλο που η GPU είναι σε πλεονάζουσα θέση από τις CPU. Στην I, II και III Περίπτωση: φαίνεται πως το πυκνότερο πλέγμα εκτελεί με μεγάλη διαφορά περισσότερο χρόνο. Στα αντίστοιχα διαγράμματα απόδοσης, στην 1 CPUcore συγκεκριμένα βλέπουμε ότι αποδοτικότερη ήταν η περίπτωση με το πυκνότερο πλέγμα, ενώ στις 8 CPUcores και 1 GPU καταστάσεις αποδοτικότερη ήταν του πυκνού πλέγματος και όχι του αραιού όπως θα περιμέναμε. Αυτό πιθανόν γίνεται γιατί εκείνη τη στιγμή κάτι προέκυψε κατά την εκτέλεση του κώδικα και το εξάρτημα (hardware, CPU/GPU) δεν μπόρεσε επαρκώς να αποθηκεύσει τα αποτελέσματα του κώδικα και απλώς προσπαθούσε ολοκληρώσει τη διαδικασία, καθώς έχουμε χαμηλές τιμές απόδοσης και χρόνου.

Γενικά αποδείχθηκε το προαναφερόμενο φαινόμενο. Το εξάρτημα GPU δείχνει να απαιτεί λιγότερο χρόνο εκτέλεσης έχοντας καλύτερες αποδόσεις, ανεξαρτήτως από τις προδιαγραφές της συσκευής. Η παρούσα εργασία έχει προοπτικές αναβάθμισης και εξέλιξης. Η εφαρμογή της GPU μπορεί να τεθεί αναμφίβολα σε θέση λειτουργίας, αρκεί να γίνει προσθήκη ή και αναβάθμιση της κάρτας, για μια όλο και πιο ακριβή μοντελοποίηση και βελτιστοποίηση των φυσικών φαινομένων μέσα σε πάρα πολύ μικρούς χρόνους εκτέλεσης και μεγάλης απόδοσης.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

- Anastasis Vasileiadis (2020). Julia: Μια γλώσσα για Cybersecurity, Data analysis και Machine Learning. Από: Julia: Μια γλώσσα για Cybersecurity, Data analysis και Machine Learning – Cerebrux

- Bastian E. Rapp (2017). Microfluidics: Modeling, Mechanics and Mathematics. Micro and Nano Technologies. p. 243-263. Από: Fluids – ScienceDirect

- Bastian E. Rapp (2017). Microfluidics: Modeling, Mechanics and Mathematics. Micro and Nano Technologies. p. 655-678 . Από: Finite Element Method - ScienceDirect

- CPU vs GPU. HEAVY.AI. Από: CPU vs GPU | Definition and FAQs | HEAVY.AI

- High-performance GPU programming in a high-level language. Από: JuliaGPU

- How a CPU Works. Από: How a CPU Works - Hardware Secrets

- Jiyuan Tu, Guan-Heng Yeoh, Chaoqun Liu (2018). Computational Fluid Dynamics – A Practical Approach. Third Edition. Butterworth-Heinemann is an imprint of Elsevier.

- Joel H. Ferziger, Milovan Perić, Robert L. Street (2020). Computational Methods for Fluid Dynamics. Fourth Edition. Springer Nature Switzerland AG 2020.

- John D. Anderson, Jr. Fundamentals of Aerodynamics. Fifth Edition. p.795.

- Julia Programming (2020). Tutorialspoint. Από: Julia Programming Language (tutorialspoint.com)

- Myrto Zeneli, Aris Nikolopoulos, Sotirios Karellas, Nikolaos Nikolopoulos (2021). Ultra-High Temperature Thermal Energy Storage, Transfer and Conversion. Energy. p. 165-199. Από: Numerical methods for solid-liquid phase-change problems - ScienceDirect

- Suraj Pawar, Omer San (2019). CFD Julia: A Learning Module Structuring an Introductory Course on Computational Fluid Dynamics. School of Mechanical and Aerospace Engineering. Fluids. Από: Fluids | Free Full-Text | CFD Julia: A Learning Module Structuring an Introductory Course on Computational Fluid Dynamics | HTML (mdpi.com)

- Parallel Computing. Από: Parallel Computing · The Julia Language

- Parallel programming with Julia using MPI (2018). Marginalia. Από: Parallel programming with Julia using MPI | Marginalia (claudiobellei.com)

➢ Systems architecture. Common CPU components. Computer Science, Computer Systems. Common CPU components - Systems architecture - OCR - GCSE Computer Science Revision - OCR - BBC Bitesize

➢ Top 6 Reasons to Choose Structured Grids in CFD. Design Engineering. Από: Top 6 Reasons to Choose Structured Grids in CFD - Design Engineering (design-engineering.com)

➢ ΚΕΛΑΪΔΗ ΑΙΜΙΛΙΑ (2012). ΜΕΛΕΤΗ ΤΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ CUDA ΚΑΙ ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΡΤΩΝ GPU ΤΗΣ NVIDIA. Πτυχιακή Εργασία, Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Πληροφορικής. Θεσσαλονίκη.

➢ Κωνσταντίνος Κολοβός (2015). Προγραμματισμός μεθόδου συναρτήσεων ακτινικής βάσης (RBF) για τη μετατόπιση υπολογιστικών πλεγμάτων σε κάρτες γραφικών. ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ, ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΕΜΠ, ΤΟΜΕΑΣ ΡΕΥΣΤΩΝ ΜΟΝΑΔΑ ΠΑΡΑΛΛΗΛΗΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΡΕΥΣΤΟΔΥΝΑΜΙΚΗΣ & ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ. ΑΘΗΝΑ. Από: kolovos.pdf (ntua.gr)

➢ ΛΑΒΔΑΝΗΣ ΓΕΩΡΓΙΟΣ (2011). ΣΥΣΤΗΜΑ ΟΛΟΚΛΗΡΩΜΕΝΗΣ ΔΙΑΧΕΙΡΙΣΗΣ GRID ΒΑΣΙΣΜΕΝΟ ΣΕ BOINC (FRONT END). ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Καβάλας, Σχολή Τεχνικών Εφαρμογών, Τμήμα Βιομηχανικής Πληροφορικής. ΚΑΒΑΛΑ.

➢ Σαβιολάκης Ιωάννης (2015). Αλγόριθμοι Ομαδοποίησης σε γενετικές δομές με OpenMP. Διπλωματική Εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Πατρών. Πάτρα.

➢ ΡΟΗ ΑΕΡΑ ΓΥΡΩ ΑΠΟ ΚΥΛΙΝΔΡΟ - PDF Free Download (docplayer.gr)

ΠΑΡΑΡΤΗΜΑ Α΄

```
const USE_GPU = false # Use GPU? If this is set false, then no GPU needs to be available
using ParallelStencil
using ParallelStencil.FiniteDifferences2D
@static if USE_GPU
@init_parallel_stencil(CUDA, Float64, 2)
else
@init_parallel_stencil(Threads, Float64, 2)
end
using LinearAlgebra, Printf, Statistics
using MAT, Plots
using WriteVTK

@views function runme(; do_vis=true, do_save=true)
# physics
## dimensionally independent
ly = 1.0 # [m]
ρ = 1.0 # [kg/m^3]
vin = 0.01 # [m/s]
## scales
psc = ρ*vin^2
## nondimensional parameters
Re = 1e2 # rho*vsc*ly/μ
Fr = Inf # vsc/sqrt(g*ly)
lx_ly = 0.6 # lx/ly
a_ly = 0.05 # rad/ly
b_ly = 0.05 # rad/ly
ox_ly = 0.05
oy_ly = -0.4
β = 0*π/6
## dimensionally dependent
lx = lx_ly*ly
ox = ox_ly*ly
oy = oy_ly*ly
μ = 1/Re*ρ*vin*ly
g = 1/Fr^2*vin^2/ly
a2 = (a_ly*ly)^2
b2 = (b_ly*ly)^2
sinβ,cosβ = sincos(β)
# numerics
ny = 255
nx = ceil(Int,ny*lx_ly)
εit = 1e-3
niter = 50*nx
nchk = 1*(nx-1)
nvis = 10
nt = 1000
nsave = 10
CFLτ = 0.9/sqrt(2)
CFL_visc = 1/4.1
CFL_adv = 1.0
# preprocessing
dx,dy = lx/nx,ly/ny
dt = min(CFL_visc*dy^2*ρ/μ,CFL_adv*dy/vin)
damp = 2/ny
dτ = CFLτ*dy
xc,yc = LinRange(-(lx-dx)/2,(lx-dx)/2,nx ),LinRange(-(ly-dy)/2,(ly-dy)/2,ny )
xv,yv = LinRange(-lx/2 ,lx/2 ,nx+1),LinRange(-ly/2 ,ly/2 ,ny+1)
# allocation
Pr = @zeros(nx ,ny )
```

```
dPrdτ = @zeros(nx-2,ny-2)
C = @zeros(nx ,ny )
C_o = @zeros(nx ,ny )
τxx = @zeros(nx ,ny )
τyy = @zeros(nx ,ny )
τxy = @zeros(nx-1,ny-1)
Vx = @zeros(nx+1,ny )
Vy = @zeros(nx ,ny+1)
Vx_o = @zeros(nx+1,ny )
Vy_o = @zeros(nx ,ny+1)
∇V = @zeros(nx ,ny )
Rp = @zeros(nx-2,ny-2)
# init
Vprof = Data.Array([4*vin*x/lx*(1.0-x/lx) for x=LinRange(0.5dx,lx-0.5dx,nx,)])
Vy[:,1] .= Vprof
Pr .= .-(yc'.-ly/2).*ρ.*g

# if do_save !ispath("./out_vis") && mkdir("./out_vis");
matwrite("out_vis/step_0.mat",Dict("Pr"=>Array(Pr),"Vx"=>Array(Vx),"Vy"=>Array(Vy
),"C"
=>Array(C),"dx"=>dx,"dy"=>dy)) end

# Preparation of visualisation
ENV["GKSwstype"]="nul"; if isdir("viz2D_out")==false mkdir("viz2D_out") end;
loadpath = "./viz2D_out/"; anim = Animation(loadpath,String[])
println("Animation directory: $(anim.dir)")
X, Y, Yv = 0:dx:lx, 0:dy:ly, (-dy/2):dy:(ly+dy/2)


if do_save !ispath("./out_vis") && mkdir("./out_vis");
vtmfile = vtk_multiblock("out_vis/step_0")
vtkfile = vtk_grid(vtmfile, xc, yc)
vtkfile["Pr"] = Array(Pr)
vtkfile["C"] = Array(C)
vtkfile = vtk_grid(vtmfile, xc, yv)
vtkfile["Vy"] = Array(Vy)
vtkfile = vtk_grid(vtmfile, xv, yc)
vtkfile["Vx"] = Array(Vx)
outfiles = vtk_save(vtmfile)
end
#global wtime0 = Base.time()
# action
for it = 1:nt
if (it==11) global wtime0 = Base.time() end
err_evo = Float64[]; iter_evo = Float64[]
@parallel update_τ!(τxx,τyy,τxy,Vx,Vy,μ,dx,dy)
@parallel predict_V!(Vx,Vy,τxx,τyy,τxy,ρ,g,dt,dx,dy)
@parallel set_sphere!(C,Vx,Vy,a2,b2,ox,oy,sinβ,cosβ,lx,ly,dx,dy)
@parallel update_∇V!(∇V,Vx,Vy,dx,dy)
println("#it = $it")
for iter = 1:niter
@parallel update_dPrdτ!(Pr,dPrdτ,∇V,ρ,dt,dτ,damp,dx,dy)
@parallel update_Pr!(Pr,dPrdτ,dτ)
set_bc_Pr!(Pr, 0.0)
if iter % nchk == 0
@parallel compute_res!(Rp,Pr,∇V,ρ,dt,dx,dy)
err = maximum(abs.(Rp))*ly^2/psc
push!(err_evo, err); push!(iter_evo,iter/ny)
@printf(" #iter = %d, err = %1.3e\n", iter, err)
if err < εit || !isfinite(err) break end
end
end
@parallel correct_V!(Vx,Vy,Pr,dt,ρ,dx,dy)
```

```
@parallel set_sphere!(C,Vx,Vy,a2,b2,ox,oy,sinβ,cosβ,lx,ly,dx,dy)
set_bc_Vel!(Vx, Vy, Vprof)
Vx_o .= Vx; Vy_o .= Vy; C_o .= C
@parallel advect!(Vx,Vx_o,Vy,Vy_o,C,C_o,dt,dx,dy)
if do_vis && it % nvis == 0

p1=heatmap(xc,yc,Array(Pr)';aspect_ratio=1, xlims=(-lx/2,lx/2),
ylims=(-ly/2,ly/2), x_ticks=-lx/2:0.3:lx/2, title="Pr")
p2=plot(iter_evo,err_evo, legend=false, xlabel="# iterations",
ylabel="log10(error)"; yscale=:log10)
p3=heatmap(xc,yc,Array(C)';aspect_ratio=1, xlims=(-lx/2,lx/2),
ylims=(-ly/2,ly/2), x_ticks=-lx/2:0.3:lx/2, title="C")
p4=heatmap(xc,yv,Array(Vy)';aspect_ratio=1, xlims=(-lx/2,lx/2),
ylims=(-ly/2,ly/2), x_ticks=-lx/2:0.3:lx/2, title="Vy")

#display(plot(p1,p2,p3,p4))
plot(p1, p2, p3, p4); frame(anim);
gif(anim, "Stokes2D.gif", fps = 15)
end
if do_save && it % nsave == 0

#matwrite("out_vis/step_$it.mat",Dict("Pr"=>Array(Pr),"Vx"=>Array(Vx),"Vy"
=>Array(Vy),"C"=>Array(C),"dx"=>dx,"dy"=>dy))
vtmfile = vtk_multiblock("out_vis/step_$it")
vtkfile = vtk_grid(vtmfile, xc, yc)
vtkfile["Pr"] = Array(Pr)
vtkfile["C"] = Array(C)
vtkfile = vtk_grid(vtmfile, xc, yv)
vtkfile["Vy"] = Array(Vy)
vtkfile = vtk_grid(vtmfile, xv, yc)
vtkfile["Vx"] = Array(Vx)
outfiles = vtk_save(vtmfile)
end
end
# Performance
wtime = Base.time() - wtime0
A_eff = (4*2)/1e9*nx*ny*sizeof(Data.Number) # Effective main memory access
per iteration [GB] (Lower bound of required memory access: Te has to be read and
written: 2 whole-array memaccess; Ci has to be read: : 1 whole-array memaccess)
wtime_it = wtime/(nt-10) # Execution time per iteration [s]
T_eff = A_eff/wtime_it
@printf("Total steps = %d, time = %1.3e sec (@ T_eff = %1.9f GB/s) \n", nt,
wtime, round(T_eff, sigdigits=9))
return
end



macro ∇V() esc(:( @d_xa(Vx)/dx + @d_ya(Vy)/dy )) end
@parallel function update_τ!(τxx,τyy,τxy,Vx,Vy,μ,dx,dy)
@all(τxx) = 2μ*(@d_xa(Vx)/dx - @∇V()/3.0)
@all(τyy) = 2μ*(@d_ya(Vy)/dy - @∇V()/3.0)
@all(τxy) = μ*(@d_yi(Vx)/dy + @d_xi(Vy)/dx)
return
end

@parallel function predict_V!(Vx,Vy,τxx,τyy,τxy,ρ,g,dt,dx,dy)
@inn(Vx) = @inn(Vx) + dt/ρ*(@d_xi(τxx)/dx + @d_ya(τxy)/dy )
@inn(Vy) = @inn(Vy) + dt/ρ*(@d_yi(τyy)/dy + @d_xa(τxy)/dx - ρ*g)
return
end

```

```
173 @parallel function update_∇V!(∇V,Vx,Vy,dx,dy)
174 @all(∇V) = @∇V()
175 return
176 end
177
178 @parallel function update_dPrdτ!(Pr,dPrdτ,∇V,ρ,dt,dτ,damp,dx,dy)
179 @all(dPrdτ) = @all(dPrdτ)*(1.0-damp) + dτ*(@d2_xi(Pr)/dx/dx +
@d2_yi(Pr)/dy/dy -
ρ/dt*@inn(∇V))
180 return
181 end
182
183 @parallel function update_Pr!(Pr,dPrdτ,dτ)
184 @inn(Pr) = @inn(Pr) + dτ*@all(dPrdτ)
185 return
186 end
187
188 @parallel function compute_res!(Rp,Pr,∇V,ρ,dt,dx,dy)
189 @all(Rp) = @d2_xi(Pr)/dx/dx + @d2_yi(Pr)/dy/dy - ρ/dt*@inn(∇V)
190 return
191 end
192
193 @parallel function correct_V!(Vx,Vy,Pr,dt,ρ,dx,dy)
194 @inn(Vx) = @inn(Vx) - dt/ρ*@d_xi(Pr)/dx
195 @inn(Vy) = @inn(Vy) - dt/ρ*@d_yi(Pr)/dy
196 return
197 end
198
199 @parallel_indices (iy) function bc_x!(A)
200 A[1 , iy] = A[2 , iy]
201 A[end, iy] = A[end-1, iy]
202 return
203 end
204
205 @parallel_indices (ix) function bc_y!(A)
206 A[ix, 1 ] = A[ix, 2 ]
207 A[ix, end] = A[ix, end-1]
208 return
209 end
210
211 @parallel_indices (ix) function bc_yV!(A, V)
212 A[ix, 1 ] = V[ix]
213 A[ix, end] = A[ix, end-1]
214 return
215 end
216
217 @parallel_indices (ix) function bc_yval!(A, val)
218 A[ix, 1 ] = A[ix, 2]
219 A[ix, end] = val
220 return
221 end
222
223 function set_bc_Vel!(Vx, Vy, Vprof)
224 @parallel bc_y!(Vx)
225 @parallel bc_x!(Vy)
226 @parallel bc_yV!(Vy, Vprof)
227 return
228 end
229
230 function set_bc_Pr!(Pr, val)
231 @parallel bc_x!(Pr)
232 @parallel bc_yval!(Pr, val)
233 return
```

```
234 end
235
236 @inline function backtrack!(A,A_o,vxc,vyc,dt,dx,dy,ix,iy)
237 δx,δy = dt*vxc/dx, dt*vyc/dy
238 ix1 = clamp(floor(Int,ix-δx),1,size(A,1))
239 iy1 = clamp(floor(Int,iy-δy),1,size(A,2))
240 ix2,iy2 = clamp(ix1+1,1,size(A,1)),clamp(iy1+1,1,size(A,2))
241 δx = (δx>0) - (δx%1); δy = (δy>0) - (δy%1)
242 fx1 = lerp(A_o[ix1,iy1],A_o[ix2,iy1],δx)
243 fx2 = lerp(A_o[ix1,iy2],A_o[ix2,iy2],δx)
244 A[ix,iy] = lerp(fx1,fx2,δy)
245 return
246 end
247
248 @inline lerp(a,b,t) = b*t + a*(1-t)
249
250 @parallel_indices (ix,iy) function advect!(Vx,Vx_o,Vy,Vy_o,C,C_o,dt,dx,dy)
251 if ix > 1 && ix < size(Vx,1) && iy <= size(Vx,2)
252 vxc = Vx_o[ix,iy]
253 vyc = 0.25*(Vy_o[ix-1,iy]+Vy_o[ix-1,iy+1]+Vy_o[ix,iy]+Vy_o[ix,iy+1])
254 backtrack!(Vx,Vx_o,vxc,vyc,dt,dx,dy,ix,iy)
255 end
256 if iy > 1 && iy < size(Vy,2) && ix <= size(Vy,1)
257 vxc = 0.25*(Vx_o[ix,iy-1]+Vx_o[ix+1,iy-1]+Vx_o[ix,iy]+Vx_o[ix+1,iy])
258 vyc = Vy_o[ix,iy]
259 backtrack!(Vy,Vy_o,vxc,vyc,dt,dx,dy,ix,iy)
260 end
261 if checkbounds(Bool,C,ix,iy)
262 vxc = 0.5*(Vx_o[ix,iy]+Vx_o[ix+1,iy])
263 vyc = 0.5*(Vy_o[ix,iy]+Vy_o[ix,iy+1])
264 backtrack!(C,C_o,vxc,vyc,dt,dx,dy,ix,iy)
265 end
266 return
267 end
268
269 @parallel_indices (ix,iy) function
set_sphere!(C,Vx,Vy,a2,b2,ox,oy,sinβ,cosβ,lx,ly,dx,dy)
270 xv,yv = (ix-1)*dx - lx/2, (iy-1)*dy - ly/2
271 xc,yc = xv+dx/2, yv+dx/2
272 if checkbounds(Bool,C,ix,iy)
273 xr = (xc-ox)*cosβ - (yc-oy)*sinβ
274 yr = (xc-ox)*sinβ + (yc-oy)*cosβ
275 if xr*xr/a2 + yr*yr/b2 < 1.05
276 C[ix,iy] = 1.0
277 end
278 end
279 if checkbounds(Bool,Vx,ix,iy)
280 xr = (xv-ox)*cosβ - (yc-oy)*sinβ
281 yr = (xv-ox)*sinβ + (yc-oy)*cosβ
282 if xr*xr/a2 + yr*yr/b2 < 1.0
283 Vx[ix,iy] = 0.0
284 end
285 end
286 if checkbounds(Bool,Vy,ix,iy)
287 xr = (xc-ox)*cosβ - (yv-oy)*sinβ
288 yr = (xc-ox)*sinβ + (yv-oy)*cosβ
289 if xr*xr/a2 + yr*yr/b2 < 1.0
290 Vy[ix,iy] = 0.0
291 end
292 end
293 return
294 end
295
296 runme()
```