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

Αριθμητική επίλυση προβλημάτων αεροδυναμικής-αεροελαστικότητας σε επεξεργαστές καρτών γραφικών

Διδακτορική Διατριβή

Ξενοφών Σ. Τρομπούκης

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου
Καθηγητής ΕΜΠ

Αθήνα, 2012

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή Κ. Γιαννάκογλου, που μου ανέθεσε το πολύ ενδιαφέρον θέμα που πραγματεύεται η διατριβή και μου έδωσε την ευκαιρία να ασχοληθώ με τις κάρτες γραφικών. Η καθοδήγηση και οι συμβουλές του πάνω σε θέματα που σχετίζονται με τη διατριβή ήταν πολύ σημαντικές για την ολοκλήρωσή της.

Επίσης, ευχαριστώ τα υπόλοιπα δύο μέλη της συμβουλευτικής τριμελούς επιτροπής, τους Αν. Καθηγητή Σ. Βουτσινά και Καθηγητή Κ. Μαθιουδάκη, για τις υποδείξεις τους στην παρουσίαση και το κείμενο της διατριβής. Τον κ. Βουτσινά θα ήθελα να τον ευχαριστήσω ιδιαίτερα και για την υποστήριξη και τις συμβουλές του κατά τη συνεργασία μας στο ερευνητικό πρόγραμμα ACFA. Ευχαριστώ και τα υπόλοιπα μέλη της επταμελούς επιτροπής για την τιμή που μου έκαναν να συμμετέχουν στη διαδικασία της τελικής παρουσίασης της διατριβής.

Θα ήθελα να ευχαριστήσω θερμά και τα υπόλοιπα μέλη της ΜΠΥΡ&Β/ΕΘΣ, για τις γνώσεις που μοιράστηκαν μαζί μου, τη μεταξύ μας συνεργασία και το φιλικό κλίμα που διαμόρφωσαν στο χώρο εργασίας μας. Κατ΄ αρχήν τον Δρ. Θ. Ζερβογιάννη για την πολύτιμη βοήθεια που μου προσέφερε κατά τα πρώτα βήματα μου ως υπ. διδάκτορας. Τον Δρ. Ι. Καμπόλη, που διαδραμάτισε καταλυτικό ρόλο στην έναρξη της ενασχόλησης της ΜΠΥΡ&Β/ΕΘΣ με τις κάρτες γραφικών, για την πολύτιμη υποστήριξη. Την Δρ. Β. Ασούτη για τη μεταξύ μας συνεργασία στον προγραμματισμό ενός μεγάλου όγκου κωδίκων σε κάρτες γραφικών (και όχι μόνο) και γενικά την συμβολή της στο έργο της διατριβής. Επίσης, ευχαριστώ τους συνεργάτες-φίλους: Σ. Κυριάκου, Δρ. Ε. Κοντολέοντος, Δρ. Χ. Γεωργοπούλου, Δρ. Δ. Παπαδημητρίου, Ε. Παπουτσή-Κιαχαγιά, Κ. Τσιάκα, Ι. Καββαδία, Γ. Δημητρακόπουλο, Δρ. Π. Λιακόπουλο και Δρ. Α. Ζυμάρη για τη φιλία τους και τις απολαυστικές συζητήσεις που είχαμε πάνω σε θέματα επιστημονικά και μη.

Ακόμα, θέλω να ευχαριστήσω το Κοινωφελές Ίδρυμα Ωνάση, το οποίο υποστήριξε οικονομικά τη διατριβή αυτή με την υποτροφία που μου παρείχε.

Τέλος ένα μόνο ευχαριστώ δεν αρκεί στους γονείς και τον αδερφό μου Γιάννη για την υποστήριξη και την αγάπη τους.

Στην οικογένειά μου

Περίληψη

Σκοπός της παρούσας διδακτορικής διατριβής είναι η δημιουργία μεθόδων και λογισμικού για την επίλυση 2Δ/3Δ προβλημάτων αεροδυναμικής/αεροελαστικότητας. Βασικός στόχος είναι η σημαντική μείωση του υπολογιστικού κόστους και του χρόνου αναμονής του μηχανικού, κατά τη φάση ανάλυσης ή σχεδιασμού αεροδυναμικών μορφών, ιδίως σε προβλήματα μεγάλης κλίμακας, με τη βέλτιστη χρήση των επεξεργαστών καρτών γραφικών (GPUs).

Τα τελευταία 10 χρόνια οι GPUs, έχουν εξελιχθεί σε πολυπύρηνες παράλληλες μονάδες επεξεργασίας κοινής μνήμης, με υπολογιστική ισχύ υπερδεκαπλάσια οποιασδήποτε σύγχρονης κεντρικής μονάδας επεξεργασίας (CPU). Οι GPUs που χρησιμοποιήθηκαν στην παρούσα διατριβή στηρίζονται στην αρχιτεκτονική CUDA που έχει αναπτυχθεί από την NVIDIA. Συγκεκριμένα, χρησιμοποιήθηκαν τα μοντέλα καρτών γραφικών GeForce GTX 280, 285 και Tesla M2050, με τις τελευταίες να αποτελούν το πλέον σύγχρονο τύπο κάρτας γραφικών της παραπάνω εταιρίας τη στιγμή που γράφεται η διατριβή αυτή. Οι Tesla M2050 που χρησιμοποιήθηκαν, ανήκουν σε συστοιχία καρτών γραφικών της ΜΠΥΡ&Β/ΕΘΣ που αποτελείται από 4 διασυνδεδεμένους υπολογιστικούς κόμβους (blade servers). Κάθε υπολογιστικός κόμβος περιέχει 3 Tesla M2050. Συνολικά, η συστοιχία αποτελείται από 12 Tesla M2050, οι οποίες χρησιμοποιήθηκαν για την παράλληλη επίλυση μεγάλης κλίμακας αεροδυναμικών/αεροελαστικών προβλημάτων.

Στη διδακτορική διατριβή αναπτύχθηκε GPU-κώδικας υψηλής παράλληλης απόδοσης που επιλύει τις 2Δ χρονικά μόνιμες ή μη-μόνιμες εξισώσεις Navier–Stokes συμπιεστού ρευστού και τις 3Δ χρονικά μόνιμες ή μη-μόνιμες εξισώσεις Euler, σε δυναμικά μεταβαλλόμενα μη-δομημένα/υβριδικά πλέγματα. Η μοντελοποίηση της τύρβης, όπου χρειάστηκε, έγινε με το μοντέλο μίας εξίσωσης των Spalart-Allmaras. Η ολοκλήρωση των εξισώσεων της ροής γίνεται σύμφωνα με την τεχνική των πεπερασμένων όγκων με κεντροκομβική αποθήκευση των μεταβλητών της ροής. Ο κώδικας που αναπτύχθηκε αναπαράγει τα αριθμητικά αποτελέσματα προυπάρχοντος CPU-επιλύτη των εξισώσεων Navier–Stokes, επαρκώς πιστοποιημένου στο πλαίσιο παλαιότερων διατριβών στη ΜΠΥΡ&Β/ΕΘΣ. Συνεπώς, τουλάχιστον όσον αφορά την αεροδυναμική ανάλυση, έμφαση δόθηκε αποκλειστικά στη μείωση του υπολογιστικού χρόνου.

Βασικό μειονέκτημα των σύγχρονων GPUs σε σχέση με τις σημερινές CPUs είναι η περιορισμένη ενδιάμεση (cache) μνήμη τους. Οι cache μνήμες είναι μνήμες ταχείας προσπέλασης και χρησιμοποιούνται για την προσωρινή αποθήκευση δεδομένων. Όσο μεγαλύτερο είναι το μέγεθος της συνολικής cache μνήμης που έχει μία μονάδα επεξεργασίας τόσο ταχύτερη γίνεται η εκτέλεση ενός κώδικα σ΄ αυτή. Επομένως, η παράλληλη απόδοση οποιουδήποτε GPU-κώδικα είναι άρρηκτα συνδεδεμένη με θέματα διαχείρισης της μνήμης της GPU. Σημειώνεται ότι η επιλογή της κεντροκομβικής διατύπωσης σε μη-δομημένα πλέγματα αποτελεί τη δυσκολότερη περίπτωση σε θέματα διαχείρισης της μνήμης της GPU, συγκριτικά με την εναλλακτική χρήση δομημένων πλεγμάτων ή της κεντροκυψελικής διατύπωσης σε μη-δομημένα πλέγματα. Αυτό οφείλεται στη μη-δομημένη αρίθμηση των κόμβων του πλέγματος και στο μεταβλητό πλήθος γειτόνων ανά κόμβο.

Ο πρώτος GPU-επιλύτης που προγραμματίστηκε προέκυψε απλά ξαναγράφοντας τον υπάρχοντα CPU-επιλύτη στη γλώσσα του περιβάλλοντος προγραμματισμού της CUDA (C++ με την προσθήκη ορισμένων ειδικών συναρτήσεων για τη διαχείριση της GPU). Παρόλο που ο εν λόγω GPU-επιλύτης μείωσε το χρόνο επίλυσης προβλημάτων αεροδυναμικής (σε σχέση με τον αντίστοιχο CPU-κώδικα), η παράλληλη απόδοσή του ήταν αρκετά χαμηλή. Για την αύξησή της έγιναν οι ακόλουθες ενέργειες:

- Προγραμματίστηκαν και αξιολογήθηκαν, ως προς τον χρόνο εκτέλεσης, διαφορετικές τεχνικές υπολογισμού των αριθμητικών διανυσμάτων της ροής.
- Δεδομένου ότι η κεντρική μνήμη της GPU περιέχει 'ξεχωριστά' τμήματα μνήμης (constant, texture και local μνήμες) και ότι, διεργασίες που εκτελούνται στον ίδιο πολυεπεξεργαστή της κάρτας γραφικών μπορούν να ανταλλάσσουν δεδομένα μέσω της γρήγορα προσπελάσιμης shared μνήμης, η υψηλή απόδοση του GPU-επιλύτη επιτεύχθηκε με τη σωστή/προσεκτική διαχείριση των μνημών της GPU. Για παράδειγμα, ο προγραμματισμός στηρίχθηκε, κατά πολύ, στο ότι δεδομένα στα οποία έχουν πρόσβαση διεργασίες που εκτελούνται ταυτόχρονα πρέπει να καταλαμβάνουν κοντινές (ή ακόμα και διαδοχικές) θέσεις μνήμης.
- Το λογισμικό επίλυσης αναδομήθηκε πλήρως, με τρόπο ώστε οι σειριακές διεργασίες να εκτελούνται στη CPU ενώ, ταυτόχρονα, άλλες ανεξάρτητες διεργασίες να εκτελούνται παράλληλα στη GPU.
- Το πλέγμα χωρίστηκε σε υποχωρία και οι κόμβοι του κάθε υποχωρίου επαναριθμήθηκαν και έπειτα ταξινομήθηκαν με βάση το πλήθος των γειτονικών σε αυτούς κόμβους.
- Χρησιμοποιήθηκε αριθμητική μικτής ακρίβειας. Σύμφωνα με αυτή, τα μητρώα του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής αποθηκεύονται σε μεταβλητές απλής ακρίβειας, μειώνοντας τον αριθμό προσβάσεων στην κεντρική μνήμη της GPU. Αυτό συμβάλλει στην αύξηση της παράλληλης απόδοσης του GPU-κώδικα χωρίς να αλλοιώνεται η ακρίβεια των αποτελεσμάτων, η οποία καθορίζεται αποκλειστικά από τον τρόπο υπολογισμού των υπολοίπων των εξισώσεων της ροής. Τα τελευταία υπολογίζονται με την επιβαλλόμενη ακρίβεια δεύτερης τάξης. Η μείωση του αριθμού των προσβάσεων στην κεντρική μνήμη της GPU, λόγω της χρήσης της αριθμητικής μικτής ακρίβειας, οφείλεται στον 'ειδικό' τρόπο με τον οποίο η GPU χειρίζεται τις μνήμες της.

Οι ενέργειες που αναφέρθηκαν επέβαλαν, εκ των πραγμάτων, την πλήρη αναδόμηση του αρχικού GPU-κώδικα. Ο τελικός GPU-κώδικας έχει περίπου 20 φορές υψηλότερη παράλληλη απόδοση ως προς τον αρχικό. Για τον υπολογισμό της παράλληλης απόδοσης του τελικού GPU-κώδικα υπολογίστηκαν ροές γύρω από μεμονωμένες αεροτομές χρησιμοποιώντας πλέγματα διαφορετικού πλήθους κόμβων και διαφορετικές κάρτες γραφικών (GTX 280, 285 και Tesla M2050). Φάνηκε ότι η επιτάχυνση στην πρόλεξη της ροής που δίνει η χρήση μίας GPU αντί ενός πυρήνα μίας σημερινής CPU εξαρτάται από το μοντέλο της ροής (ατριβής, τυρβώδης), τη διάσταση του πλέγματος και, φυσικά, τον τύπο της κάρτας γραφικών. Συγκεκριμένα, οι εφαρμογές μεγάλης κλίμακας βοηθούνται περισσότερο από τη χρήση GPUs, καθώς οι προλέξεις στα μεγαλύτερα πλέγματα εμφανίζουν μεγαλύτερη παράλληλη επιτάχυνση. Για το λόγο αυτό οι εφαρμογές που παρουσιάζονται στη συνέχεια έχουν διαφορετικές παράλληλες επιταχύνσεις.

Η μεγαλύτερη επιτάχυνση που καταγράφηκε στην επίλυση 2Δ τυρβώδους συμπιεστής ροής είναι 60x, 90x και 110x εκτελώντας τον GPU-κώδικα σε μία Tesla M2050 και χρησιμοποιώντας αριθμητική διπλής, μικτής και απλής ακρίβειας, αντίστοιχα. Οι ίδιοι υπολογισμοί αν πραγματοποιούνταν σε μία GTX 285 θα έδιναν επιτάχυνση μικρότερη κατά περίπου 30%. Η μέγιστη επιτάχυνση που καταγράφηκε στην επίλυση των 3Δ εξισώσεων Euler είναι 40x, 55x και 90x χρησιμοποιώντας μία Tesla M2050 και αριθμητική διπλής, μικτής και απλής ακρίβειας αντίστοιχα. Οι αντίστοιχες επιταχύνσεις χρησιμοποιώντας μία GTX 285 είναι μικρότερες κατά περίπου 25%. Υπενθυμίζεται ότι η χρήση αριθμητικής μικτής ακρίβειας αναπαράγει τα αποτελέσματα της διπλής ακρίβειας συνιστώσας του GPU-κώδικα. Στον υπολογισμό των επιταχύνσεων που παρουσιάστηκαν προηγουμένως, ο CPU-κώδικας χρησιμοποιεί αριθμητική διπλής ακρίβειας και εκτελείται σε έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη.

Επικουρικά, και για λόγους πληρότητας, στη διατριβή πραγματοποιήθηκε και η ανάπτυξη GPU-επιλυτών των εξισώσεων ατριβούς συμπιεστής ροής σε δομημένα πλέγματα ή κάνοντας χρήση της κεντροκυψελικής διατύπωσης της τεχνικής των πεπερασμένων όγκων σε μη-δομημένα πλέγματα. Επιπλέον, η διατριβή συνεισέφερε στην αποδοτική μεταφορά κωδίκων επίλυσης των 2Δ εξισώσεων ασυμπίεστης ροής και του συζυγούς προβλήματος από τη CPU στη GPU.

Από την πλευρά της αεροδυναμικής ανάλυσης, ο GPU-επιλύτης χρησιμοποιήθηκε για τη μελέτη (α) της αλληλεπίδρασης του οριακού στρώματος με κύμα κρούσης (buffet) στην αεροτομή OAT15A και (β) της επίδρασης σύνθετης δέσμης (synthetic jet) στη μείωση της ζώνης αποκόλλησης κατάντι εμποδίου. Επιλύθηκε, επιπλέον, η ροή μη-συνεκτικού ρευστού γύρω από αεροσκάφος και σε πτερύγωση υπερηχητικού συμπιεστή. Η επιτάχυνση στην πρόλεξη της ροής από τη χρήση μίας GPU σε αυτές τις εφαρμογές είναι μεταξύ 40x και 45x. Στις εφαρμογές αυτές χρησιμοποιήθηκαν οι τότε διαθέσιμες GeForce GTX 285. Αξιοσημείωτο είναι το γεγονός ότι οι συγκεκριμένες κάρτες γραφικών έχουν τοποθετηθεί σε παλιούς-ξεπερασμένους υπολογιστές, αναβαθμίζοντας τους.

Έχοντας αναπτύξει έναν αξιόπιστο πολύ γρήγορο επιλύτη (τη μικτής ακρίβειας εκδοχή του GPU-κώδικα), επόμενο βήμα ήταν η χρήση αυτού ως λογισμικού αξιολόγησης σε εξελικτικό αλγόριθμο (EA, λογισμικό EASY), όπου απαιτείται μεγάλο πλήθος αξιολογήσεων. Έτσι, πραγματοποιήθηκε η βελτιστοποίηση των παραμέτρων ενεργού ελέγχου της ροής (μέσω συνεχούς αναρρόφησης) σε αγωγό με διαμόρφωση. Ο EA συντόνιζε περισσότερες από μία GPUs και ανέθετε την αξιολόγηση της εκάστοτε υποψήφιας λύσης σε μία κάρτα. Επομένως, το κέρδος από τη χρήση GPUs αντί CPUs υπερτίθεται σε εκείνο λόγω της ταυτόχρονης αξιολόγησης υποψήφιων λύσεων σε κάθε γενιά του EA.

Στοχεύοντας στην εκμετάλλευση της ταχύτητας εκτέλεσης της λιγότερο ακριβής και ακριβούς εκδοχής απλής ακρίβειας του GPU-κώδικα, προτάθηκε διεπίπεδο σχήμα βελτιστοποίησης καθοδηγούμενο από το λογισμικό EASY, όπου ο απλής ακρίβειας GPU-επιλύτης χρησιμοποιείται για μία γρήγορη πρώτη ανίχνευση του χώρου των υποψήφιων λύσεων στο χαμηλό επίπεδο. Ο αποδεκτής ακρίβειας και, ως εκ τούτου, πιο αργός (αλλά πολύ πιο γρήγορος σε σχέση με το αντίστοιχο CPU-λογισμικό) μικτής

ακρίβειας GPU-κώδικας χρησιμοποιείται για την αξιολόγηση των υποψήφιων λύσεων του υψηλού επιπέδου, από όπου προκύπτει το μέτωπο Pareto των μη-κυριαρχούμενων λύσεων (στην περίπτωση πολυκριτηριακής βελτιστοποίησης) ή η βέλτιστη λύση (στην περίπτωση μονοκριτηριακής βελτιστοποίησης). Το χαμηλό και το υψηλό επίπεδο ανταλλάσσουν μικρό πλήθος από τις καλύτερες ανά επίπεδο υποψήφιες λύσεις, οι οποίες επαναξιολογούνται από το λογισμικό αξιολόγησης του επιπέδου. Οι αξιολογήσεις και των δύο επιπέδων γίνονται ταυτόχρονα σε όλες τις διαθέσιμες GPUs. Ο ρυθμός επικοινωνίας των ΕΑ ανά επίπεδο καθορίζεται από το χρήστη. Ο διεπίπεδος ΕΑ χρησιμοποιήθηκε για το σχεδιασμό μεμονωμένης αεροτομής και αεροτομής συμπιεστή με στόχους τη μεγιστοποίηση του συντελεστή άνωσης/ελαχιστοποίηση του συντελεστή οπισθέλκουσας και την ελαχιστοποίηση του συντελεστή απωλειών, αντίστοιχα. Έτσι, το κέρδος από τη χρήση του διεπίπεδου αλγορίθμου, σε σχέση με τη χρήση ενός κλασικού ΕΑ (ενός επιπέδου), υπερτίθεται σε εκείνο από τη χρήση των GPUs. Με τις εφαρμογές αυτές αναδεικνύεται ένας καλός τρόπος χρήσης του απλής ακρίβειας GPU-επιλύτη, αν και ο τελευταίος, από μόνος του, δεν είναι σήμερα αποδεκτός για χρήση σε προβλήματα υπολογιστικής ρευστοδυναμικής. Σημειώνεται ότι, για τα πλέγματα που χρησιμοποιήθηκαν στις παραπάνω εφαρμογές, κάθε αξιολόγηση με το λογισμικό απλής ακρίβειας χρειάζεται περίπου 25% λιγότερο χρόνο σε σχέση με αντίστοιχη αξιολόγηση με το λογισμικό μικτής ακρίβειας.

Για την επίλυση αεροελαστικών προβλημάτων χρειάστηκε η επέκταση του GPU-κώδικα, ώστε να είναι δυνατή η χρήση δυναμικών πλεγμάτων, πλεγμάτων δηλαδή που ακολουθούν την κίνηση/παραμόρφωση της επιφάνειας του εξεταζόμενου αεροδυναμικού σώματος προσαρμοζόμενα στην εκάστοτε μορφή αυτής. Η χρήση της μεθόδου στρεπτικών ελατηρίων εξασφαλίζει τη διατήρηση της ποιότητας του πλέγματος μετά από την προσαρμογή του. Αρχικά, πιστοποιήθηκε ο GPU-επιλύτης (χρησιμοποιώντας δυναμικά πλέγματα) σε εφαρμογή εξωτερικής αεροδυναμικής όπου αεροτομή εκτελεί εξαναγκασμένη περιστροφική ταλάντωση γύρω από το 1/4 της χορδής της. Η σύγκριση με πειραματικά αποτελέσματα είναι ικανοποιητική. Στη συνέχεια, ο GPU-επιλύτης χρησιμοποιήθηκε για τη μελέτη της αεροελαστικής συμπεριφοράς αεροτομής, η κίνηση της οποίας περιορίζεται από ένα στρεπτικό και ένα γραμμικό ελατήριο. Υπολογίστηκε το όριο της περιοχής ευσταθούς λειτουργίας της αεροτομής (flutter boundary). Η σύγκριση με υπολογιστικά αποτελέσματα άλλων ερευνητικών ομάδων είναι πολύ καλή. Επιπλέον, πραγματοποιήθηκε η μελέτη της αεροελαστικής συμπεριφοράς αεροτομής με τρεις βαθμούς ελευθερίας. Η ακμή εκφυγής της αεροτομής θεωρήθηκε ως πτερύγιο ελέγχου, η κίνηση του οποίου περιορίζεται από ένα επιπλέον στρεπτικό ελατήριο. Επιπλέον, επιτεύχθηκε αύξηση της περιοχής ευσταθούς λειτουργίας με τον έλεγχο της γωνίας της ακμής εκφυγής (ως προς το κύριο σώμα της αεροτομής), μέσω ανάδρασης των μεταβλητών κατάστασης της αεροτομής. Επισημαίνεται ότι η εύρεση του ορίου της περιοχής ευσταθούς λειτουργίας μίας αεροτομής απαιτεί την επίλυση μεγάλου πλήθους μη-μόνιμων ροών, κάτι που θα ήταν εξαιρετικά δαπανηρό αν χρησιμοποιούνταν μόνο CPUs. Η επιτάχυνση που δίνει η χρήση των Tesla M2050 είναι 50x.

Για την επίλυση μεγάλης κλίμακας προβλημάτων αεροδυναμικής/αεροελαστικότητας, μία GPU δεν επαρκούσε εξαιτίας της περιορισμένης έκτασης της κεντρικής της μνήμης (1 GByte ή 3 GByte, αντίστοιχα, για τις GeForce GTX 285 και Tesla M2050).

Για το λόγο αυτό, ο GPU-κώδικας παραλληλοποιήθηκε σε πολλές GPUs του ίδιου, ή διαφορετικών υπολογιστικών κόμβων. Η διαχείριση των GPUs του ίδιου υπολογιστικού κόμβου ανατίθεται στο ίδιο CPU-thread. Για την επικοινωνία μεταξύ δύο ή περισσότερων GPUs διαφορετικών υπολογιστικών κόμβων χρησιμοποιήθηκε το πρωτόκολλο παράλληλης επικοινωνίας MPI. Ο παράλληλος σε πολλές κάρτες γραφικών GPU-επιλύτης χρησιμοποιήθηκε για την επιτυχή υλοποίηση των υπολογισμών ερευνητικού προγράμματος χρηματοδοτούμενου από την Ευρωπαϊκή Ένωση, σχετικού με το σχεδιασμό ενός νέου αεροσκάφους τύπου Blended-Wing-Body (BWB). Τα αεροσκάφη αυτού του τύπου χαρακτηρίζονται από πολλά πλεονεκτήματα από αεροδυναμικής πλευράς, όπως μεγαλύτερος λόγος άνωσης προς οπισθέλκουσα, μικρότερη κατανάλωση καυσίμου ανά επιβάτη και μίλι και μικρότερη εκπομπή θορύβου σε σχέση με συμβατικά αεροσκάφη ίδιων προδιαγραφών. Χρησιμοποιήθηκε ο GPU-επιλύτης της διατριβής για τη μελέτη της αεροδυναμικής συμπεριφοράς του αεροσκάφους όταν προεπιλεγμένη επιφάνεια ελέγχου ή ολόκληρη η κατασκευή εξαναγκάζεται σε περιοδική κίνηση/παραμόρφωση. Αν και η χρήση GPUs δεν ήταν επιβεβλημένη, επιλέχθηκε να χρησιμοποιηθεί ο GPU-κώδικας για τη μείωση του χρόνου επίλυσης των ζητούμενων αεροδυναμικών/αεροελαστικών προβλημάτων. Συγκεκριμένα, τα προβλήματα ροής που μελετήθηκαν επιλύονται 6 φορές πιο γρήγορα σε 3 Tesla M2050 του ίδιου υπολογιστικού κόμβου αντί σε 8 οκταπύρηνες CPUs (σύνολο 64 CPU-πυρήνες).

Συνοψίζοντας, στις εφαρμογές που αναφέρθηκαν, η επιτάχυνση από τη χρήση των GPUs κυμαίνεται από 40x έως 110x, ανάλογα με το χρησιμοποιούμενο μοντέλο ροής, τη διάσταση του πλέγματος, την κάρτα γραφικών και την ακρίβεια της αριθμητικής που χρησιμοποιείται (διπλής, μικτής ή απλής). Η επιτάχυνση αυτή είναι ιδιαίτερα σημαντική αφενός στην αεροδυναμική ή αεροελαστική ανάλυση σωμάτων αφετέρου δε (και κυρίως) στη βελτιστοποίηση-σχεδιασμό όπου ο απαιτούμενος αριθμός κλήσεων του λογισμικού αξιολόγησης αυξάνει σημαντικά. Με τον τρόπο αυτό η χρήση GPUs, αντί CPUs, διευρύνει το εύρος των βιομηχανικών εφαρμογών μεγάλης κλίμακας που μπορούν να γίνουν σε αποδεκτό χρόνο και μειώνει σημαντικά το χρόνο αναμονής του μηχανικού. Από τη χρήση των GPUs ευνοούνται περισσότερο προβλήματα μεγαλύτερης κλίμακας, καθώς καταγράφουν μεγαλύτερες επιταχύνσεις. Κλείνοντας, ας σημειωθεί ότι η γνώση που αποκτήθηκε μπορεί να χρησιμοποιηθεί στην επίλυση προβλημάτων άλλων επιστημονικών περιοχών που διέπονται από μ.δ.ε. διατυπωμένες σε μη-δομημένα πλέγματα.

Numerical Solution of Aerodynamic-Aeroelastic Problems on Graphics Processing Units

Xenofon S. Trompoukis

PhD Thesis

National Technical University of Athens,
School of Mechanical Engineering,
Laboratory of Thernal Turbomachines,
Parallel CFD & Optimization Unit.

Supervisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2012

Abstract

The main objective of this PhD thesis is the development of software for the solution of aerodynamic and aeroelastic problems, running on modern Graphics Processing Units (GPUs). On GPUs, the analysis and design of aerodynamic shapes can be carried out with substantial reduction in computational cost, as compared to the corresponding software when running on Central Processing Units (CPUs).

Over the last 10 years, GPUs have evolved into many-core shared memory parallel systems. Their computational capabilities exceed those of modern CPUs by more than one order of magnitude. The GPUs, used in this PhD thesis, are based on the CUDA architecture developed by NVIDIA. Regarding hardware, the GeForce 280, 285 and Tesla M2050 cards are used. The Tesla M2050 graphics cards are part of the PCopt/LTT GPU-cluster, which comprises four interconnected blade servers with three Tesla M2050 each. So, for the solution of large scale aerodynamic and aeroelastic problems, a cluster with twelve Tesla M2050 units was used.

In this PhD thesis, GPU-solvers for the 2D steady/unsteady Navier-Stokes equations for compressible fluids and the 3D steady/unsteady Euler equations for compressible fluids have been developed. Both GPU-solvers use dynamic unstructured/hybrid grids which are able to deform, following the deflection/deformation of the body surface. The grid quality is ensured by the spring analogy method. For turbulent flows, the one-equation turbulence model of Spalart-Allmaras is used to effect closure. The integration of the flow equations on the unstructured grids is carried out using the finite volume method, with vertex-centered storage. The GPU-enabled

software reproduces the results of the corresponding CPU solver, which has been thoroughly validated in previous PhD theses at PCopt/LTT. Therefore, emphasis was exclusively laid on the reduction of the computational cost.

In comparison to contemporary CPUs, the major drawback of programmable GPUs is their limited cache memory. Cache memories are low-latency memories, used for temporary data storage. The greater the size of the available cache memories, the faster a program runs. Thus, the parallel efficiency of any GPU-code is strongly related with memory handling issues. From this point of view, the vertex-centered approach of the finite volume technique on unstructured grids is the worst case, compared either to the use of structured grids or even the cell-centered finite volume approach for unstructured grids. This is due to the lack of structure in the numbering of the grid nodes and, compared to cell-centered schemes, the variable number of neighbors per grid node.

Programming efficient CFD codes on GPUs required rewriting the existing CPU-solver (programmed in FORTRAN90) in the language supported by CUDA (C++ with some extensions for the efficient control of the GPU in that period of time). Even if the firstly developed GPU-code could solve the flow equations faster than the corresponding CPU-software, its parallel speed-up was not as high as expected. In order to further increase the parallel efficiency of the GPU-solver, the following actions were taken:

- Different programming approaches for the computation of the flow fluxes were tried and assessed in terms of parallel speed-up.
- Since GPUs contain extra memories, other than the main (device or global) one, the programming technique was reconsidered so as to maximize their exploitation. These memories are the texture, constant, local and shared ones. The cached texture, cached constant and non-cached local memory spaces reside in device memory. On the other hand, the shared memory is distributed to the GPU multiprocessors and is used for interchanging data between the processes executed in parallel on the same multiprocessor. The efficient use of all GPU memories increased the parallel speed-up of the GPU-solver. Among other:
  - (a) It was ensured that concurrently running processes access nearby (or even consecutive) global memory spaces.
  - (b) CPU and GPU processes may run in parallel.
  - (c) The computational grid was divided into subdomains and the nodes of each subdomain were renumbered and sorted in ascending order, based on the number of their neighbours.
- A mixed precision arithmetic (MPA) solver was proposed. Since the discretized flow equations are solved for the flow variables' corrections, a less accurate numerical scheme for the computation of the l.h.s. coefficient matrices and an accurate one for the r.h.s. terms (i.e. the residuals) can be used. In the proposed MPA code variant, the l.h.s. coefficient matrices are stored into single precision (SPA) variables whereas the residuals are stored into double precision variables (DPA). This reduces the total number of global memory

accesses and, thus, increases the parallel efficiency of the GPU-code. Besides, the MPA does not harm the accuracy of the solution since the residuals are computed with double precision.

These actions called for the complete restructuring of the initial GPU-code. The final GPU-code solves the flow equations about 20 times faster compared to the initial one. In order to measure the parallel speed-up of the programmed GPU-enabled code(s), a number of inviscid and turbulent flow problems around an airfoil and a wing were solved. In these cases, unstructured grids (consisting of triangular or tetrahedral elements for 2D or 3D cases respectively) with different numbers of grid nodes were used on the GTX 280, 285 and Tesla M2050 graphics cards. It was shown that the flow model (inviscid of turbulent), the grid size and the compute capabilities of the GPUs affect the parallel speed-up of the GPU-solver. As a consequence, the examined problems give different speed-ups. Large scale problems take more advantage of the GPUs, as larger grids achieve higher speed-ups. For 2D turbulent flows of compressible fluids, the highest speed-up achieved is about 60x, 90x and 110x, using a Tesla M2050, compared to a single core of a modern CPU, using double, mixed or simple precision arithmetic, respectively. The same computations on a GTX 285 yield about 30% less speed-up. The highest recorded speed-ups for the solution of the 3D Euler equations is 40x, 55x and 90x, on Tesla M2050, with DPA, MPA and SPA, respectively. The corresponding speed-ups for GTX285 are about 25% less. For the measurement of the aforementioned speed-ups, the CPU-solver used DPA and was executed on a single core of an Intel Xeon CPU, 2.00GHz, 4096 MB cache memory.

For the purpose of comparison, an Euler solver using either structured or unstructured grids and the cell-centered approach was also programmed. Moreover, the present PhD thesis also contributed to the efficient porting of the adjoint equation solvers, from the CPU to the GPU, so as to reduce the cost of gradient-based optimization processes.

As far as the aerodynamic analysis is of concern, the GPU-solver was used for the study (a) of the dynamic interaction between the boundary layer and shock waves (buffet) on the OAT15A airfoil, and (b) the impact of a synthetic jet in the reduction of the separation region downstream of a bump geometry. Moreover, the GPU-solver is used for the prediction of inviscid flow around a civil aircraft and inside a hypersonic compressor cascade. For these problems, the speed-up ranges from 40x to 45x using a single GTX 285 graphic card vis-à-vis to a single core of a modern CPU. It is worth noting that these GPUs have been plugged in some old personal computers. Using the Tesla M2050 graphics cards, the speed-up increased by 20%, compared to GTX 285.

Having developed efficient GPU flow solvers, the next step was to use them in evolutionary algorithm (EA) based optimization (via the EASY software). The optimization of the steady suction control parameters, for the minimization of the length of the separation region downstream a bump placed inside a duct, was carried out on a number of GPUs (one GPU per evaluation). Thus, the gain from using GPUs, instead of CPUs, was superposed to the gain from the parallel evaluation of

the population members.

Based on the different speed-ups and prediction accuracy of the SPA and MPA GPU implementations of the Navier-Stokes equations solver, a hierarchical optimization method which is suitable for GPUs is proposed and demonstrated in turbulent 2D flow problems. The search for the optimal solution(s) splits into two EA-based levels, with different evaluation tools each. The low level EA uses the very fast SPA GPU implementation with relaxed convergence criteria for the inexpensive evaluation of candidate solutions. Promising solutions are regularly broadcast to the high level EA which uses the MPA GPU implementation of the same flow solver. Single- and two-objective aerodynamic shape optimization problems were solved using the developed software. Namely, the design of an isolated airfoil for maximum lift and minimum drag coefficient and the design of a compressor cascade airfoil for minimum pressure losses were carried out using the aforementioned two-level EA. The gain from using the hierarchical EA instead of a conventional (single-level) one was superposed to the gain from using GPUs instead of CPUs.

As far as aeroelastic problems are of concern, the GPU-solver was validated in a pitching airfoil case and used for the prediction of the flutter boundaries of an airfoil, with two or three degrees of freedom. In these cases, the motion of the airfoil is constrained by a linear (plunging) and a torsional (pitching) spring. In the three-degree of freedom airfoil case, the motion of the airfoil trailing edge (flap) is constrained by means of an additional torsional spring. The numerical results obtained by using the developed GPU-solver are in good agreement with published numerical results using different CFD tools.

Due to memory limitations, a single GPU cannot be used to solve large scale aerodynamic/aeroelastic problems. For large scale problems, GPU clusters must be used. In the present thesis, a single CPU-thread was used to control the available GPUs per computational node. The communication between the GPUs of the interconnected nodes is based on the MPI protocol. The parallel GPU-solver was used for the analysis of a Blended-Wing-Body (BWB) civil aircraft. Aircrafts of this type have many aerodynamic benefits (higher ratio lift to drag, lower fuel consumption per passenger and mile, reduced noise production, etc) compared to commercial aircrafts with equivalent specifications. The corresponding studies were carried out in the framework of a project (Active Control for Flexible 2020 Aircraft, ACFA 2020), funded by the European Union. In this computationally demanding application, the GPUs were used for the reduction of the wall clock time; the use of three Tesla M2050 on the same computational node was about six times faster than 64 CPU-cores. Steady and unsteady inviscid flows around the BWB aircraft are presented; in the unsteady cases, a control surface or the aircraft was forced to oscillate/deform periodically.

In conclusion, in this PhD thesis, GPUs were used for the solution of aerodynamic and aeroelastic problems using unstructured computational grids. Speed-ups from the use of GPUs ranging between 40x and 110x are reported. This substantial decrease in computational cost affects positively both the aerodynamic/aeroelastic analysis of aerodynamic bodies and the optimization-design via EAs, where a large

number of evaluations must be performed. Since, larger grids achieve higher speed-ups, large scale problems take more advantage of the GPUs. This fact extends the range of industrial applications that can be carried out using CFD analysis tools. Finally, it should be noted that the proposed programming techniques can easily be implemented in any other GPU-enabled software solving general purpose PDEs on unstructured grids.

Ακρωνύμια

ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
ΕΘΣ	Εργαστήριο Θερμικών Στροβιλομηχανών
ΜΠΥΡ&Β	Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης
ΥΡΔ	Υπολογιστική Ρευστοδυναμική

CFD	Computational Fluid Dynamics
NTUA	National Technical University of Athens
PCopt	Parallel CFD & Optimization unit

DPA	Αριθμητική διπλής ακρίβειας (Double Precision Arithmetic)
MPA	Αριθμητική μικτής ακρίβειας (Mixed Precision Arithmetic)
SPA	Αριθμητική απλής ακρίβειας (Single Precision Arithmetic)

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

Αριθμητική επίλυση προβλημάτων αεροδυναμικής-αεροελαστικότητας σε επεξεργαστές καρτών γραφικών

Διδακτορική Διατριβή

Ξενοφών Σ. Τρομπούκης

Τριμελής Συμβουλευτική Επιτροπή:

1. Γιαννάκογλου Κυριάκος,
   (επιβλέπων)
   Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
2. Μαθιουδάκης Κωνσταντίνος,
   Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
3. Βουτσινάς Σπυρίδων,
   Αν. Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών

Επταμελής Εξεταστική Επιτροπή:

1. Γιαννάκογλου Κυριάκος,
   (επιβλέπων)
   Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
2. Μαθιουδάκης Κωνσταντίνος,
   Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
3. Βουτσινάς Σπυρίδων,
   Αν. Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
4. Μπουντουβής Ανδρέας,
   Καθηγητής ΕΜΠ,
   Σχολή Χημικών Μηχανικών
5. Τζαμπίρας Γεώργιος,
   Καθηγητής ΕΜΠ,
   Σχολή Ναυπηγών Μηχανολόγων Μηχανικών
6. Τσαγγάρης Σωκράτης,
   Καθηγητής ΕΜΠ,
   Σχολή Μηχανολόγων Μηχανικών
7. Κοζύρης Νεκτάριος,
   Αν. Καθηγητής ΕΜΠ,
   Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Αθήνα, 2012

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η μετάβαση από την εποχή προγραμματισμού σε κεντρικές μονάδες επεξεργασίας (CPUs) στις πολλά υποσχόμενες μονάδες επεξεργασίας γραφικών (GPUs), με έμφαση στην επίλυση προβλημάτων υπολογιστικής ρευστοδυναμικής (ΥΡΔ), αποτέλεσε το βασικό στόχο της διατριβής. Συγκεκριμένα, επιλύθηκαν προβλημάτα αεροδυναμικής και αεροελαστικής ανάλυσης και σχεδιασμού. Πριν την παρουσίαση των σχετικών εφαρμογών, όμως, απαραίτητη είναι η παρουσίαση των υπολογιστικών δυνατοτήτων των σύγχρονων GPUs. Έτσι, το κεφάλαιο αυτό, αρχικά, εισάγει τον αναγνώστη στην τεχνολογία των καρτών γραφικών. Ακολουθεί μία σύντομη βιβλιογραφική επισκόπηση της χρήσης των GPUs στην επίλυση επιστημονικών προβλημάτων, με έμφαση στον τομέα της ΥΡΔ. Το κεφάλαιο ολοκληρώνεται με την παρουσίαση της δομής της διατριβής.

1.1 Οι GPUs ως σύγχρονες μονάδες παράλληλης επεξεργασίας

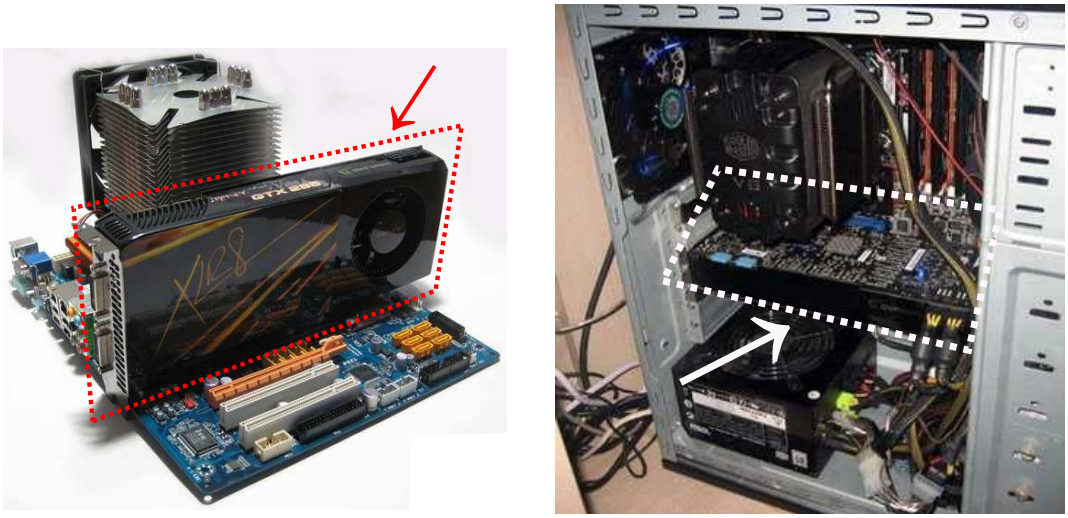
Οι σύγχρονες GPUs έχουν εξελιχθεί σε πανίσχυρες πολυπύρηνες μονάδες παράλληλης επεξεργασίας. Η τελευταία δεκαετία χαρακτηρίζεται από την ολοένα αυξανόμενη διάθεση για ενασχόληση της επιστημονικής κοινότητας με τη δημιουργία κωδίκων ή επεκτάσεων μαθηματικών πακέτων (όπως το MATLAB, ή το Mathematica), εμπορικών κωδίκων ή ελεύθερων λογισμικών (OpenFOAM) που εκτελούνται σε κάρτες γραφικών. Η χρήση των GPUs για την επίλυση επιστημονικών προβλημάτων φέρεται διεθνώς υπό το γενικό τίτλο General-Purpose computation on Graphics Processing Units, ή συντομογραφικά GPGPU, [1].

Οι κάρτες γραφικών διακρίνονται σε δύο κατηγορίες ανάλογα με την αυτονομία τους ή όχι σε μνήμη. Έτσι, διακρίνονται σε εκείνες που χρησιμοποιούν ένα τμήμα της μνήμης του υπολογιστή και σε εκείνες που περιέχουν τη δική τους αυτόνομη μνήμη. Οι πρώτες είναι ενσωματωμένες στη μητρική πλακέτα (motherboard) και κατασκευάζονται κυρίως από την Intel. Αντίθετα, οι κάρτες γραφικών με ενσωματωμένη μνήμη τοποθετούνται σε ειδικό δίαυλο (PCI express, PCIe) πάνω στη μητρική πλακέτα. Σήμερα, οι μεγαλύτερες εταιρίες παραγωγής τέτοιων καρτών γραφικών είναι η NVIDIA και η AMD

(πρώην ATI). Οι κάρτες γραφικών με ενσωματωμένη μνήμη παρουσιάζουν υψηλότερη υπολογιστική ισχύ ως προς εκείνες που χρησιμοποιούν αποκλειστικά την κεντρική μνήμη του υπολογιστή. Η διδακτορική αυτή διατριβή χρησιμοποιεί μόνο κάρτες γραφικών με ενσωματωμένη μνήμη. Επομένως, χάρη ευκολίας, στη συνέχεια της διατριβής, ο όρος GPU θα αναφέρεται αποκλειστικά σε κάρτες γραφικών αυτού του τύπου.

Μία GPU μπορεί να συνδεθεί σε οποιοδήποτε προσωπικό υπολογιστή ανεξαρτήτου ηλικίας, μετατρέποντας τον σε ένα σύγχρονο παράλληλο υπολογιστικό σύστημα. Όπως αναφέρθηκε προηγουμένως, ο μοναδικός περιορισμός είναι η μητρική του υπολογιστή να έχει ενσωματωμένο έναν ειδικό δίαυλο επικοινωνίας (PCIe). Τονίζεται ότι η απόδοση της κάρτας γραφικών εξαρτάται αποκλειστικά από την ίδια και είναι ανεξάρτητη από την ηλικία ή την τεχνολογία της CPU. Στα σχήματα 1.1 φαίνεται η κάρτα γραφικών της NVIDIA, GeForce GTX 285 (α′) τοποθετημένη σε μία μητρική πλακέτα και (β′) εσωτερικά ενός προσωπικού υπολογιστή της Μονάδας Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης του Εργαστηρίου Θερμικών Στροβιλομηχανών (ΜΠΥΡ&Β/ΕΘΣ) του ΕΜΠ.

Επιπλέον, το κόστος κτήσης μίας μέσης σημερινής GPU είναι σχετικά χαμηλό και, μάλιστα, δυσανάλογα χαμηλό σε σχέση με την υπολογιστική ισχύ που αυτή προσφέρει. Συνήθως, στο κόστος κτήσης αυτής πρέπει να προστεθεί η αγορά ενός νέου τροφοδοτικού ρεύματος του υπολογιστή, ιδίως όταν πρόκειται να 'αναβαθμισθεί' ένας σχετικά παλιός υπολογιστής. Ενδεικτικά αναφέρεται ότι μία GeForce GTX 285, GPU της NVIDIA τεχνολογίας του 2009, απαιτεί ως ελάχιστη ισχύ του συστήματος 500W.



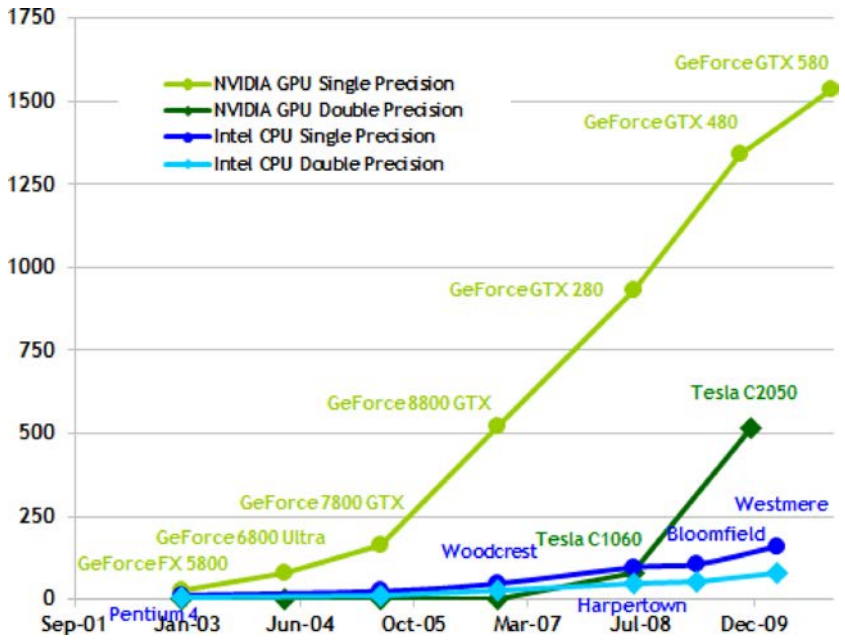
(α′) (β′)

Σχήμα 1.1: Η NVIDIA GeForce GTX 285 τοποθετημένη (α′) σε μία μητρική πλακέτα και (β′) εσωτερικά ενός προσωπικού υπολογιστή.

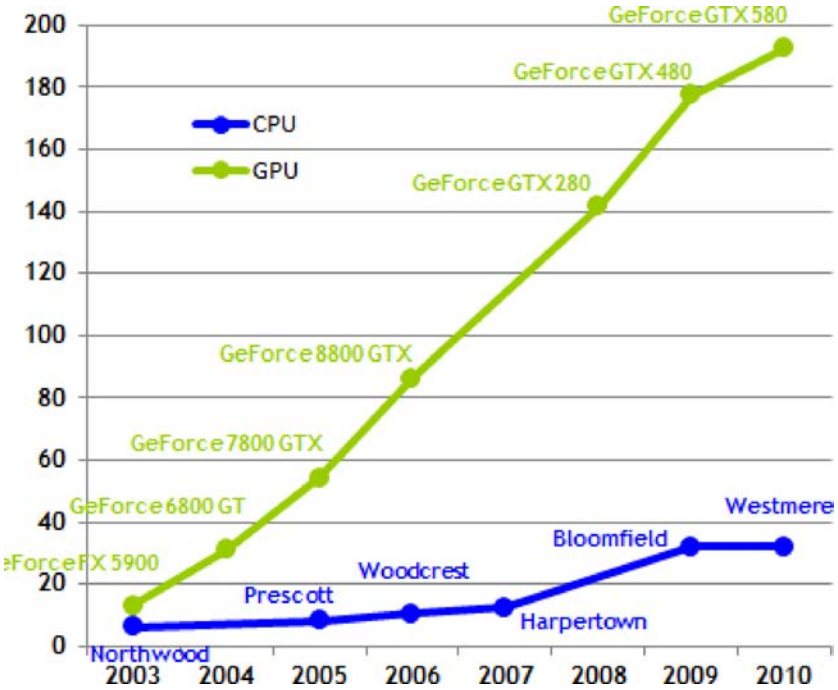
1.2 Εξέλιξη των GPUs

Η αλματώδης αύξηση της χρήσης των GPUs για την παράλληλη επεξεργασία δεδομένων, συγκριτικά με τις CPUs, οφείλεται κατά κύριο λόγο στην αλματώδη αύξηση της υπολογιστικής ισχύος και του ρυθμού προσπέλασης της κεντρικής μνήμης αυτών. Συγκεκριμένα, οι σημερινές κάρτες γραφικών εκτελούν αριθμητικές πράξεις κινητής υποδιαστολής άνω των δέκα φορών ταχύτερα σε σχέση με τις σημερινές CPUs. Το ίδιο ισχύει και για την ταχύτητα προσπέλασης της μνήμης. Χαρακτηριστικό παράδειγμα αποτελεί η Tesla M2090 που, κατά την περίοδο συγγραφής της διατριβής, αποτελούσε την πλέον σύγχρονη GPU της κατασκευάστριας εταιρίας NVIDIA. Η κάρτα αυτή περιέχει 512 πυρήνες και μπορεί να εκτελέσει έως και 665G αριθμητικές πράξεις διπλής ακρίβειας το δευτερόλεπτο (flops, floating point operations per second), ή 1331Gflops απλής ακρίβειας. Έχει μνήμη χωρητικότητας 6 GByte, με μέγιστη ταχύτητα προσπέλασης 177 GByte/sec.

Τα σχήματα 1.2, 1.3 παρουσιάζουν την εξέλιξη των GPUs, σε σχέση με αυτήν των CPUs, τη δεκαετία 2002-2011. Φαίνεται η υπεροχή των σημερινών GPUs, σε σχέση με τις σημερινές CPUs. Το σχήμα 1.2 δείχνει την αύξηση της μέγιστης ταχύτητας εκτέλεσης αριθμητικών πράξεων απλής και διπλής ακρίβειας, με την εξέλιξη της τεχνολογίας των καρτών γραφικών κατά την τελευταία δεκαετία. Όμοια, το σχήμα 1.3 παρουσιάζει την αύξηση του ρυθμού προσπέλασης της κεντρικής μνήμης των GPUs. Οι λόγοι της ξεκάθαρης αυτής υπεροχής αναπτύσσονται στις ενότητες 3.4 και 3.5, όπου περιγράφεται η αρχιτεκτονική των GPUs που χρησιμοποιήθηκαν στη διατριβή αυτή.



Σχήμα 1.2: Μέγιστη ταχύτητα εκτέλεσης αριθμητικών πράξεων κινητής υποδιαστολής απλής και διπλής ακρίβειας (Gflops) για τις GPUs και CPUs που αναπτύχθηκαν τη δεκαετία 2002-2011, [2].



Σχήμα 1.3: Μέγιστη ταχύτητα προσπέλασης (GByte/sec) της κεντρικής μνήμης ενός υπολογιστή και μίας GPU τεχνολογίας 2002-2011, [2].

Συνεπώς, η χρήση των GPUs επιταχύνει την επίλυση επιστημονικών προβλημάτων. Η επιτάχυνση αυτή συνήθως ξεπερνά κατά πολύ το 10x (10 φορές δηλαδή πιο γρήγορη επίλυση ενός προβλήματος χρησιμοποιώντας μία σύγχρονη GPU, αντί μίας σημερινής CPU) και εξαρτάται από το βαθμό παραλληλοποίησης του προς επίλυση προβλήματος, την υπολογιστική μέθοδο που ο προγραμματιστής έχει επιλέξει και, φυσικά, τις υπολογιστικές δυνατότητες της GPU που χρησιμοποιείται.

Η ενότητα 1.3 αναφέρεται στην ακρίβεια της αριθμητικής που χρησιμοποιούν οι προγραμματιζόμενες κάρτες γραφικών. Τονίζεται η μειωμένη ακρίβεια των πρώτων προγραμματιζόμενων GPUs, η οποία, αρχικά εμπόδισε την ευρεία αποδοχή τους ως συστημάτων παράλληλης επεξεργασίας δεδομένων από την επιστημονική κοινότητα. Η ενότητα καταλήγει σε πρόταση για χρήση αριθμητικής μικτής ακρίβειας που αποτελεί συνδυασμό αριθμητικής απλής και διπλής ακρίβειας. Στη συνέχεια, οι ενότητες 1.4, 1.5 και 1.6 παρουσιάζουν δημοσιευμένες εργασίες διαφορετικών επιστημονικών κλάδων, δίνοντας έμφαση στον κλάδο της υπολογιστικής ρευστοδυναμικής, όπου χρησιμοποιούνται κάρτες γραφικών ή συστοιχίες καρτών γραφικών για την επιτάχυνση της πρόλεξης των αριθμητικών αποτελεσμάτων. Σε καθεμία από τις ενότητες αυτές τονίζεται η συμβολή της διατριβής.

1.3 Ακρίβεια αριθμητικών πράξεων των προγραμματιζόμενων GPUs

Οι πρώτες προγραμματιζόμενες GPUs υποστήριζαν μόνο αριθμητική απλής ακρίβειας. Το γεγονός αυτό αποτέλεσε σημαντικό ανασταλτικό παράγοντα στην ανάπτυξη λογισμικού, για την παραγωγή υψηλής ακρίβειας αποτελεσμάτων, κατάλληλου προς εκτέλεση σε GPUs. Επιπλέον, οι πρώτες προγραμματιζόμενες GPUs που χειρίζονταν πραγματικούς αριθμούς απλής ακρίβειας δεν υποστήριζαν πλήρως το πρότυπο 754 της IEEE (Institute of Electrical and Electronic Engineers) για τη δυαδική αναπαράσταση κινητής υποδιαστολής. Αντίθετα, οι σημερινές GPUs, όπως φαίνεται και στο σχήμα 1.2, υποστηρίζουν την αριθμητική διπλής ακρίβειας και, πλήρως, το πρότυπο αποθήκευσης πραγματικών αριθμών IEEE754.

Το θέμα της μειωμένης τάξης ακρίβειας των πρώτων προγραμματιζόμενων GPUs θίγεται στην εργασία [3], όπου συγκρίνεται η ακρίβεια των υπολογισμών μίας GPU της NVIDIA αρχιτεκτονικής G80 χρησιμοποιώντας αριθμητική απλής ακρίβειας με μία σύγχρονη CPU που υποστηρίζει πλήρως το πρότυπο IEEE754 και χρησιμοποιεί επίσης αριθμητική απλής ακρίβειας. Για τη σύγκριση πραγματοποιήθηκε ο πολλαπλασιασμός δύο μητρώων. Ως αποτέλεσμα αναφοράς θεωρήθηκε το αποτέλεσμα του CPU-κώδικα όταν χρησιμοποιεί 16 Bytes για την αναπαράσταση των πραγματικών αριθμών (long double). Αποδείχθηκε ότι τα αποτελέσματα του GPU-κώδικα υστερούσαν σε ακρίβεια ακόμη και σε σχέση με εκείνα του CPU-κώδικα που χρησιμοποιούσε αριθμητική απλής ακρίβειας.

Στο πλαίσιο χρήσης απλής ακρίβειας GPUs, αλλά πρόλεξης υψηλής ακρίβειας αποτελεσμάτων, η εργασία [4] προτείνει τη συνεργασία μεταξύ CPU-GPU στη δημιουργία ενός υβριδικού κώδικα αριθμητικής μικτής ακρίβειας, όπου η GPU χρησιμοποιεί αριθμητική απλής και η CPU αριθμητική διπλής ακρίβειας. Η συνεργασία αυτή, βέβαια, επιβάλλει συχνή μεταφορά δεδομένων μεταξύ της μνήμης της GPU και εκείνης του υπολογιστή και αντίστροφα και μειώνει σημαντικά την παράλληλη απόδοση του υβριδικού κώδικα. Αποτέλεσε, όμως, μία χρήσιμη πρόταση κατά την περίοδο που οι προγραμματιζόμενες GPUs δεν υποστήριζαν αριθμητική διπλής ακρίβειας.

Η αριθμητική μικτής ακρίβειας βρήκε εφαρμογή και στις πρώτες προγραμματιζόμενες GPUs διπλής ακρίβειας. Συγκεκριμένα, οι εργασίες [5, 6] προτείνουν το διαχωρισμό του προς επίλυση προβλήματος σε υποπροβλήματα με βάση την επιθυμητή ακρίβεια στους αριθμητικούς υπολογισμούς. Έτσι, τμήμα του προβλήματος επιλύεται χρησιμοποιώντας αριθμητική απλής ακρίβειας και το υπόλοιπο διπλής ακρίβειας, στην ίδια GPU. Η ιδέα της αριθμητικής μικτής ακρίβειας στηρίχθηκε στο γεγονός ότι στις πρώτες προγραμματιζόμενες GPUs που υποστήριζαν αριθμητική διπλής ακρίβειας η ταχύτητα εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας ήταν χαμηλότερη του $1/8$ εκείνης μεταξύ πραγματικών αριθμών απλής ακρίβειας (παράγραφος 3.4). Οι εν λόγω εργασίες προτείνουν, επίσης, τη χρήση τεχνικών αριθμητικής ανάλυσης για τη βελτίωση της ακρίβειας των αποτελεσμάτων όταν χρησιμοποιείται αποκλειστικά αριθμητική απλής ακρίβειας, [7]. Οι τεχνικές αυτές αυξάνουν τον όγκο των αριθμητικών πράξεων, η μεγάλη διαφορά όμως ανάμεσα στο ρυθμό εκτέλεσης αριθμητικών πράξεων

μεταξύ πραγματικών αριθμών απλής και διπλής ακρίβειας στις GPUs της εποχής εκείνης επέτρεψε τη χρήση των τεχνικών αυτών χωρίς τη μείωση της παράλληλης απόδοσης του GPU-κώδικα.

Στην παρούσα διατριβή, αλλά και στη συναφή εργασία [8], προτείνεται μία διαφορετική προσέγγιση της χρήσης της αριθμητικής μικτής ακρίβειας, με στόχο την αύξηση της παράλληλης απόδοσης ενός GPU-κώδικα. Έτσι, η χρήση της αριθμητικής μικτής ακρίβειας αποσκοπεί στην ελάττωση του πλήθους των προσβάσεων στην κεντρική μνήμη της κάρτας γραφικών και όχι στην αντικατάσταση διπλής ακρίβειας αριθμητικών πράξεων με απλής, όπως προτείνεται στις εργασίες [5, 6]. Έτσι, η μικτής ακρίβειας εκδοχή ενός GPU-κώδικα αναπαράγει τις αριθμητικές προλέξεις της διπλής ακρίβειας εκδοχής του ίδιου GPU-κώδικα, με μικρότερο υπολογιστικό κόστος. Στην παρούσα διατριβή, η επίλυση των διακριτοποιημένων εξισώσεων της ροής γίνεται ως προς τη διόρθωση των μεταβλητών της ροής, συνεπώς επιτρέπεται η αποθήκευση των μητρώων του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής σε μεταβλητές απλής ακρίβειας. Αντίθετα, οι όροι του δεξιού μέλους (δηλαδή το υπόλοιπο) των διακριτοποιημένων εξισώσεων αποθηκεύονται σε μεταβλητές διπλής ακρίβειας.

1.4 Επίλυση επιστημονικών προβλημάτων σε GPUs

Την τελευταία δεκαετία οι προγραμματιζόμενες GPUs έχουν χρησιμοποιηθεί στην επίλυση μίας πληθώρας προβλημάτων διαφορετικών επιστημονικών κλάδων. Ενδεικτικά αναφέρονται ορισμένες από τις εργασίες που έχουν δημοσιευθεί.

Οι πρώτες εργασίες σε GPUs περιλαμβάνουν τον υπολογισμό του γινομένου δύο μητρώων, [9, 10, 11]. Σε συνέχεια αυτών, οι εργασίες [12, 13, 14, 15, 16] περιγράφουν την ανάλυση συμμετρικού και μη-συμμετρικού μητρώου σε άνω και κάτω τριγωνικά μητρώα (LU decomposition και μέθοδος Cholesky) σε GPUs. Στην εργασία [17] παρουσιάζεται η αντιστροφή άνω ή κάτω τριγωνικού μητρώου. Η επιτάχυνση από τη χρήση μίας GeForce GTX 280 είναι μεγαλύτερη του 30x, χρησιμοποιώντας αριθμητική διπλής ακρίβειας. Ο μετασχηματισμός Fourier σε GPUs δείχνεται στις εργασίες [18, 19]. Η χρήση GPUs δίνει επιτάχυνση περίπου 30x, με βάση το χρόνο εκτέλεσης αντίστοιχων συναρτήσεων σε μία CPU. Στην εργασία [20] επιλύεται γραμμικό σύστημα εξισώσεων με τη μέθοδο των συζυγών κλίσεων (conjugate gradient). Το μητρώο των συντελεστών του αριστερού μέλους είναι αραιό, σε αντίθεση με τις προηγούμενες εργασίες, που τα αντίστοιχα μητρώα ήταν πλήρη.

Η χρήση των GPUs δεν περιορίζεται όμως σε εφαρμογές γραμμικής άλγεβρας. Η εργασία [21] περιγράφει τον προγραμματισμό GPU-κώδικα διάσπασης ενός χώρου n-διαστάσεων σε υποχώρους με βάση την αποστάση από επιλεγμένα σημεία αναφοράς στον χώρο (διάγραμμα Voronoi). Οι εργασίες [22, 23] ασχολούνται με την αποκατάσταση αλλοιωμένων εικόνων μέσω της επίλυσης συνήθων διαφορικών εξισώσεων. Μάλιστα, η εργασία [23] χρησιμοποιεί την τεχνική πολυπλέγματος (multigrid) για την επιπλέον επιτάχυνση της επίλυσης των διαφορικών εξισώσεων.

Ενδεικτικές εργασίες στον τομέα της μοριακής δυναμικής (Molecular Dynamics) είναι οι [24, 25, 26, 27, 28]. Η επιτάχυνση από τη χρήση GPUs στις εργασίες αυτές

εξαρτάται από την χρησιμοποιούμενη GPU και τη γλώσσα προγραμματισμού. Η μεγαλύτερη επιτάχυνση που αναφέρεται σε αυτές τις εργασίες, είναι της τάξης του 80x σε μία GeForce 8800 GTX στην εργασία [28].

Στον τομέα της υπολογιστικής ηλεκτροδυναμικής (Computational Electrodynamics), οι εργασίες [29, 30, 31] επιλύουν τις εξισώσεις Maxwell με επιτάχυνση έως και περίπου 50x χρησιμοποιώντας μία GeForce 8800 GTX αντί μίας CPU. Εφαρμογές μαγνητοϋδροδυναμικής (magnetohydrodynamic) σε GPUs παρουσιάζονται στην εργασία [32]. Η ανάπτυξη GPU-κώδικα πεπερασμένων στοιχείων περιγράφεται στις εργασίες [33, 34], με την εργασία [33] να δίνει έμφαση στην κατασκευή του μητρώου των συντελεστών του αριστερού μέλους του συστήματος των εξισώσεων που προκύπτει από την ισορροπία των ασκούμενων, ανά κόμβο του υπολογιστικού πλέγματος, δυνάμεων και ροπών.

1.5 Υπολογιστική ρευστοδυναμική σε GPUs

Παρά την ήδη ευρεία χρήση των GPUs σε διάφορους επιστημονικούς τομείς, η χρήση καρτών γραφικών στην επίλυση προβλημάτων αεροδυναμικής είναι ακόμα σχετικά περιορισμένη. Ο βασικός λόγος είναι το γεγονός ότι οι πρώτες προγραμματιζόμενες GPU δεν υποστήριζαν αριθμητική διπλής ακρίβειας που είναι απαραίτητη, για παράδειγμα, στην ακριβή πρόλεξη του οριακού στρώματος. Για το λόγο αυτό, η έμφαση των πρώτων εργασιών στον τομέα της Υπολογιστικής Ρευστοδυναμικής (ΥΡΔ) δόθηκε ώστε οι αριθμητικές προλέξεις να είναι περισσότερο οπτικά πειστικές παρά φυσικά ακριβείς. Στο πλαίσιο αυτό, οι πρώτοι GPU-επιλύτες των 2Δ ασυμπίεστων εξισώσεων Navier–Stokes, χρησιμοποιώντας δομημένα πλέγματα, αναπτύχθηκαν στις εργασίες [35, 36, 20, 37]. Η πρόλεξη των αριθμητικών αποτελεσμάτων έγινε σε μία NVIDIA GeForce FX. Στην εργασία [38], χρησιμοποιήθηκαν GPUs για την επίλυση προβλημάτων μεταφοράς θερμότητας και την πρόλεξη χρονικά μη-μόνιμης ασυμπίεστης ροής γύρω από αεροτομή που μετακινείται και περιστρέφεται περιοδικά. Σε αντίθεση με τις προηγούμενες εργασίες, στην εν λόγω εργασία η διατύπωση των εξισώσεων της ροής έγινε μέσω της στροβιλότητας και της ροϊκής συνάρτησης. Για την επιπλέον επιτάχυνση της επίλυσης των προβλημάτων, οι εργασίες [20, 38] χρησιμοποιούν την τεχνική πολυπλέγματος (multigrid). Η επίλυση των εξισώσεων Boussinesq σε GPUs παρουσιάζεται στην εργασία [39], επιτυγχάνοντας παράλληλη επιτάχυνση περίπου 4x σε σχέση με λογισμικό που εκτελείται σε μία τετραπύρηνη CPU. Η εργασία [11] προτείνει ένα πλαίσιο εφαρμογής ρητών και πεπλεγμένων αριθμητικών σχημάτων σε GPUs (ATI 9800) και περιλαμβάνει επιλύσεις των 2Δ εξισώσεων μετάδοσης κύματος και των 2Δ ασυμπίεστων εξισώσεων Navier–Stokes. Συνέχεια της εργασίας [36] αποτελεί η [40], όπου ο GPU-κώδικας επεκτείνεται για την επίλυση 3Δ ροών. Η πρόλεξη της ροής μέσω της μεθόδου Lattice Boltzmann αποτελεί αντικείμενο των εργασιών [41, 42, 43, 44, 45]. Στην εργασία [44], ο χρόνος εκτέλεσης του GPU-κώδικα είναι περίπου δύο τάξεις μεγέθους μικρότερος του χρόνου εκτέλεσης του αντίστοιχου CPU-κώδικα.

Στην εργασία [46] παρουσιάζονται για πρώτη φορά εφαρμογές σε συμπιεστές ροές. Οι εργασίες [47, 48] επικεντρώνονται στη δημιουργία GPU-λογισμικού επίλυσης

των 2Δ και 3Δ εξισώσεων Euler, χρησιμοποιώντας δομημένα υπολογιστικά πλέγματα, διαφορετικές γλώσσες προγραμματισμού και δύο μοντέλα καρτών γραφικών. Συγκεκριμένα, χρησιμοποιήθηκε μία ATI 1950XT και μία NVIDIA GeForce 8800GTX. Σημειώνονται επιταχύνσεις της τάξης του 29x ή 16x στην πρόλεξη 2Δ ή 3Δ ροών αντίστοιχα, σε σχέση με τη χρήση ενός πυρήνα μίας CPU. Η επίλυση των εξισώσεων Navier–Stokes χρησιμοποιώντας σχήματα υψηλής τάξης σε δομημένα πλέγματα σε GPUs παρουσιάζεται στην εργασία [49]. Οι επιταχύνσεις, όμως, που σημειώνονται είναι μεταξύ 9x και 16.5x που είναι χαμηλές δεδομένου ότι χρησιμοποιήθηκε μία Tesla C1060 που είναι από τις τελευταίες κάρτες γραφικών της NVIDIA. Στην εργασία [50] περιγράφεται η επιτάχυνση παράλληλου σε συστοιχία διασυνδεδεμένων υπολογιστικών κόμβων, κώδικα επίλυσης των εξισώσεων Navier–Stokes, αντικαθιστώντας υπολογιστικά χρονοβόρα τμήματα του κώδικα με κλήσεις αντίστοιχων συναρτήσεων που εκτελούνται σε μία GPU. Η ολοκλήρωση των εξισώσεων Navier–Stokes, στην εργασία αυτή, γίνεται με τη μέθοδο των πεπερασμένων στοιχείων. Η ανάλυση ατριβούς ροής σε πιο σύνθετες γεωμετρίες, συμπεριλαμβανομένων υπερηχητικών ροών, αποτέλεσε αντικείμενο της εργασίας [51]. Σημειώθηκαν επιταχύνσεις μεταξύ 15x και 40x, αναλόγως του πλήθους των κόμβων του χρησιμοποιούμενου υπολογιστικού πλέγματος, με αριθμητική απλής ακρίβειας μιας και η χρησιμοποιούμενη κάρτα γραφικών (GeForce 8800 GTX) δεν υποστήριζε αριθμητική διπλής ακρίβειας.

Όλες οι εργασίες που παρουσιάστηκαν προηγουμένως ευνοούνται από τη χρήση δομημένων υπολογιστικών πλεγμάτων τα οποία εγγυώνται ότι διεργασίες που εκτελούνται ταυτόχρονα σε μία GPU έχουν πρόσβαση σε διαδοχικές θέσεις της μνήμης της κάρτας. Όπως θα εξηγηθεί στο κεφάλαιο 3, το γεγονός αυτό αυξάνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα. Η χρήση μη-δομημένων υπολογιστικών πλεγμάτων στην επίλυση των 3Δ εξισώσεων Euler παρουσιάζεται για πρώτη φορά στις εργασίες [52, 8]. Η εργασία [52] χρησιμοποιεί την κεντροκυψελική διατύπωση της τεχνικής των πεπερασμένων όγκων για την ολοκλήρωση των εξισώσεων της ροής. Αντίθετα, η εργασία [8] χρησιμοποιεί την κεντροκομβική διατύπωση, που απαιτεί πιο προσεκτικό χειρισμό της κάρτας γραφικών όπως αναλύεται στο κεφάλαιο 4. Οι κάρτες γραφικών που χρησιμοποιήθηκαν στις δύο εργασίες είναι οι NVIDIA Tesla και GeForce GTX 280, 285 αντίστοιχα. Επιπλέον, στην εργασία [8] επιλύονται οι χρονικά μόνιμες 2Δ εξισώσεις Navier–Stokes σε GPUs χρησιμοποιώντας το μοντέλο τύρβης των Spalart–Allmaras, [53]. Ελέγχεται η επίδραση της ακρίβειας της χρησιμοποιούμενης αριθμητικής στην παράλληλη επιτάχυνση του GPU-κώδικα και προτείνεται η χρήση αριθμητικής μικτής ακρίβειας όπως αναφέρθηκε στην ενότητα 1.3. Οι απλής και μικτής ακρίβειας εκδοχές του GPU-επιλύτη των 2Δ εξισώσεων Navier–Stokes, της εργασίας [8], χρησιμοποιούνται στο σχεδιασμό μεμονωμένης αεροτομής και αεροτομής συμπιεστή ως λογισμικά αξιολόγησης σε διεπίπεδο εξελικτικό αλγόριθμο. Ο εξελικτικός αλγόριθμος (ΕΑ) εκτελείται στη CPU και ελέγχει τέσσερις GPUs στις οποίες γίνονται οι αξιολογήσεις των υποψήφιων λύσεων. Η μεταφορά ΕΑ σε GPUs, όπου η αξιολόγηση των υποψήφιων λύσεων και η εκτέλεση των τελεστών εξέλιξης γίνονται στην ίδια κάρτα γραφικών, περιγράφεται στις εργασίες [54, 55, 56]. Η παράλληλη απόδοση του GPU-ΕΑ αυξάνεται με το πλήθος των ατόμων ανά γενιά. Οι τελευταίες εργασίες, όμως, περιορίζονται στην ελαχιστοποίηση μαθηματικών συναρτήσεων. Ο GPU-επιλύτης της [8] επεκτείνεται

στην εργασία [57] για την πρόλεξη και χρονικά μη-μόνιμων συνεκτικών ροών. Παρουσιάζεται η πρόλεξη της αλληλεπίδρασης οριακού στρώματος με κύμα κρούσης στην πλευρά υποπίεσης αεροτομής. Η εργασία [58] περιέχει μία αναλυτική περιγραφή της δομής ενός GPU-επιλύτη των εξισώσεων Navier–Stokes, χρησιμοποιώντας μη-δομημένα υπολογιστικά πλέγματα, και δίνει έμφαση στη δημιουργία GPU-λογισμικού υψηλής παράλληλης απόδοσης, τονίζοντας τη σημασία της σωστής διαχείρισης της μνήμης της κάρτας γραφικών. Παρουσιάζεται, επίσης, η βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης ρευστού, μέσω ΕΑ, με στόχο την ελαχιστοποίηση της περιοχής αποκόλλησης κατάντι διαμόρφωσης που τοποθετείται στο εσωτερικό ενός αγωγού.

Στην εργασία [59] περιγράφεται η αποδοτική μεταφορά κώδικα επίλυσης των εξισώσεων ασυμπίεστης, τυρβώδους ροής (συμπεριλαμβανομένης της εξίσωσης μεταφοράς θερμότητας) αλλά και των αντίστοιχων συζυγών εξισώσεων από τη CPU στη GPU. Η σύζευξη των δύο ανωτέρω επιλυτών, σύμφωνα με την τεχνική one-shot, χρησιμοποιήθηκε για το σχεδιασμό της μορφής των αγωγών εναλλάκτη με στόχο τη μέγιστη συναγωγή θερμότητας και ταυτόχρονη ελαχιστοποίηση των απωλειών ολικής πίεσης.

Οι εργασίες [8, 57, 58, 59] σχετίζονται με την παρούσα διατριβή. Ο GPU-επιλύτης των εξισώσεων Navier–Stokes/Euler, που αναπτύχθηκε στη διατριβή και χρησιμοποιήθηκε στις εργασίες αυτές, παρουσιάζεται στο κεφάλαιο 4. Η επιτάχυνση στην πρόλεξη της ροής εξαρτάται από το πλήθος των κόμβων του υπολογιστικού πλέγματος, το μοντέλο της ροής (συνεκτική, μη-συνεκτική), τη χρησιμοποιούμενη κάρτα γραφικών και την ακρίβεια της αριθμητικής που χρησιμοποιείται. Έτσι, η επιτάχυνση κατά την επίλυση των 2Δ εξισώσεων Navier–Stokes, σε μία Tesla M2050 σε σύγκριση με έναν πυρήνα μίας σημερινής CPU, είναι έως και 110x, 90x ή 60x χρησιμοποιώντας αριθμητική απλής, μικτής ή διπλής ακρίβειας αντίστοιχα. Όμοια, η πρόλεξη 3Δ ατριβούς ροής στην ίδια κάρτα γραφικών γίνεται έως και 90, 55 ή 40 φορές γρηγορότερα χρησιμοποιώντας αριθμητική απλής, μικτής ή διπλής ακρίβειας αντίστοιχα.

Ο GPU-επιλύτης χρησιμοποιήθηκε, πέρα των εφαρμογών που ήδη αναφέρθηκαν στην περιγραφή των εργασιών [8, 57, 58], για τη μελέτη της αεροελαστικής συμπεριφοράς αεροτομής δύο και τριών βαθμών ελευθερίας, όπου και υπολογίστηκε το όριο της περιοχής ευσταθούς λειτουργίας. Ο υπολογισμός του ορίου ευσταθούς λειτουργίας της αεροτομής απαιτεί μεγάλο όγκο αεροελαστικών προλέξεων που καθιστά ιδιαίτερα χρονοβόρα την επίλυση σε CPUs.

1.6 Επίλυση επιστημονικών προβλημάτων σε συστοιχίες από GPUs

Οι εργασίες που παρουσιάστηκαν στις δύο προηγούμενες ενότητες αφορούν την επίλυση επιστημονικών προβλημάτων σε μία GPU. Εφόσον η μνήμη των σημερινών καρτών γραφικών είναι περιορισμένη (η τελευταία κάρτα γραφικών της NVIDIA κατά την περίοδο συγγραφής της διατριβής, Tesla M2090, έχει 6 GByte μνήμη), η ανάπτυξη λογισμικού παράλληλου σε συστοιχίες από GPUs είναι επιβεβλημένη για την επίλυση επιστημονικών προβλημάτων μεγάλης κλίμακας.

Η επικοινωνία των συνεργαζόμενων GPUs γίνεται είτε μέσω της κοινής μνήμης

του υπολογιστικού κόμβου, όταν οι GPUs ανήκουν στον ίδιο υπολογιστικό κόμβο, είτε χρησιμοποιώντας ένα πρωτόκολλο επικοινωνίας διασυνδεδεμένων CPUs, όπως το PVM (Parallel Virtual Machine), [60], ή το MPI (Message Passing Interface), [61], όταν οι GPUs ανήκουν σε διαφορετικούς υπολογιστικούς κόμβους. Γενικά, οι δυνατοί τρόποι επικοινωνίας μεταξύ δύο ή περισσότερων GPUs αναλύονται στο κεφάλαιο 5. Στη συνέχεια, παρουσιάζονται ορισμένες ενδεικτικές εργασίες σε συστοιχίες από GPUs με έμφαση στην υπολογιστική ρευστοδυναμική.

Οι πιο πολλές εργασίες που έχουν δημοσιευτεί χρησιμοποιούν το πρωτόκολλο επικοινωνίας MPI. Έτσι, η διαχείριση των GPUs ανατίθεται σε διεργασίες που εκτελούνται σε μία ή πολλές διασυνδεδεμένες CPUs. Κάθε CPU-διεργασία είναι υπεύθυνη για τη διαχείριση μίας GPU. Ο συγχρονισμός και η επικοινωνία μεταξύ των CPU-διεργασιών γίνεται σύμφωνα με το πρωτόκολλο MPI. Σ′ αυτό το πλαίσιο, στην εργασία [62] αναπτύσσεται παράλληλος σε πολλές κάρτες γραφικών GPU-επιλύτης των συμπιεστών εξισώσεων Navier–Stokes. Ο παράλληλος GPU-κώδικας χρησιμοποιήθηκε για την πρόλεξη της ροής εσωτερικά πτερύγωσης στροβίλου. Χρησιμοποιήθηκε δομημένο υπολογιστικό πλέγμα με περίπου 4.5×10^6 κόμβους. Ο χρόνος επίλυσης σε τέσσερις κάρτες γραφικών της NVIDIA, αρχιτεκτονικής GT200, είναι μικρότερος από 10 λεπτά. Ανάλογα, η ανάπτυξη λογισμικού επίλυσης των 3Δ ασυμπίεστων εξισώσεων Navier–Stokes σε συστοιχία καρτών γραφικών περιγράφεται στην εργασία [63]. Η πρόλεξη της ροής μέσω της μεθόδου Lattice Boltzmann σε συστοιχίες από GPUs περιγράφεται στις εργασίες [43, 64, 65, 66]. Υψηλής τάξης σχήματα διακριτοποίησης σε δομημένα υπολογιστικά πλέγματα εφαρμόζονται σε συστοιχίες καρτών γραφικών στην εργασία [67]. Η ανάλυση ασυμπίεστης διφασικής ροής παρουσιάζεται στις εργασίες [68, 69].

Όπως αναφέρθηκε, λίγες είναι οι εργασίες που δεν χρησιμοποιούν το πρωτόκολλο επικοινωνίας MPI. Τέτοιες είναι οι [70, 71], όπου παρουσιάζεται η ανάπτυξη παράλληλου σε πολλές κάρτες γραφικών του ίδιου υπολογιστικού κόμβου GPU-επιλύτη των ασυμπίεστων εξισώσεων Navier–Stokes. Στις εργασίες αυτές η διαχείριση των GPUs ανατίθεται σε διαφορετικές διεργασίες που εκτελούνται στη CPU του υπολογιστικού κόμβου. Ο χειρισμός των CPU-διεργασιών γίνεται χρησιμοποιώντας Pthreads (POSIX threads), [72]. Η επικοινωνία των GPUs γίνεται μέσω της κοινής μνήμης του υπολογιστικού κόμβου. Η NVIDIA έχει κατασκευάσει υπολογιστικούς κόμβους που περιέχουν έως και τέσσερις κάρτες γραφικών. Οι κόμβοι αυτοί δεν περιέχουν κάποια CPU και πρέπει να συνδεθούν μέσω ειδικού διαύλου με άλλον υπολογιστικό κόμβο που περιέχει CPU, ώστε να είναι λειτουργικοί. Το 2010, η Hewlett Packard (HP) κατασκεύασε αυτόνομους υπολογιστικούς κόμβους που περιέχουν τρεις κάρτες γραφικών της NVIDIA.

Η εργασία [73] προτείνει τον υβριδισμό των παραπάνω μεθόδων επικοινωνίας. Έτσι η διαχείριση των GPUs γίνεται από διαφορετικές CPU-διεργασίες. Η διαχείριση των CPU-διεργασιών που εκτελούνται στην ίδια CPU γίνεται χρησιμοποιώντας Pthreads ή με OpenMP directives (Open Multi-Processing, [74, 75]). Η επικοινωνία διαφορετικών CPUs γίνεται σύμφωνα με το πρωτόκολλο MPI.

Οι εργασίες [76, 77] προτείνουν γρήγορους και αξιόπιστους αλγορίθμους συμπίεσης και αποσυμπίεσης των δεδομένων που μεταφέρονται μεταξύ των διασυνδεδεμένων υπολογιστικών κόμβων ενός παράλληλου υπολογιστικού συστήματος. Η εφαρμογή του

αλγορίθμου της εργασίας [77] μαζί με το πρωτόκολλο επικοινωνίας MPI, στην επίλυση επιστημονικών προβλημάτων σε συστοιχίες διασυνδεδεμένων CPUs, παρουσιάζεται στην εργασία [78]. Ο χρόνος συμπίεσης και αποσυμπίεσης των δεδομένων μεταφοράς δεν υπερκαλύπτει το κέρδος από τη μεταφορά δεδομένων μικρότερης διάστασης, με αποτέλεσμα την επιτάχυνση της παράλληλης επίλυσης. Οι προηγούμενες εργασίες σχετίζονται με CPUs και όχι GPUs. Δεν έχουν δημοσιευθεί ακόμα ανάλογες εργασίες σε κάρτες γραφικών. Όπως εξηγείται στο κεφάλαιο 5, ο λόγος του χρόνου επικοινωνίας προς επεξεργασίας δεδομένων είναι μεγαλύτερος σε εφαρμογές σε συστοιχίες καρτών γραφικών, σε σχέση με αντίστοιχες εφαρμογές σε συστοιχίες διασυνδεδεμένων CPUs. Χαρακτηριστικά αναφέρεται ότι στην εργασία [64], κατά την πρόλεξη της ασυμπίεστης ροής γύρω από σφαίρα, σε συστοιχία 100 καρτών γραφικών, ο χρόνος μεταφοράς δεδομένων ήταν ελαφρώς μεγαλύτερος σε σχέση με εκείνον της επεξεργασίας. Επομένως, το κέρδος από τη χρήση τεχνικών συμπίεσης-αποσυμπίεσης των μεταφερόμενων δεδομένων σε συστοιχίες καρτών γραφικών αναμένεται να είναι σημαντικότερο σε σχέση με το κέρδος σε συστοιχίες διασυνδεδεμένων CPUs, ιδιαίτερα στην περίπτωση που η GPU και όχι η CPU είναι υπεύθυνη για τη συμπίεση/αποσυμπίεση των δεδομένων.

Όλες οι εργασίες που προαναφέρθηκαν χρησιμοποιούν δομημένα υπολογιστικά πλέγματα. Στην παρούσα διατριβή, αναπτύχθηκε παράλληλος σε πολλές GPUs επιλύτης των συμπιεστών χρονικά μη-μόνιμων εξισώσεων Euler σε μη-δομημένα υπολογιστικά πλέγματα. Οι συνεργαζόμενες GPUs ανήκουν στον ίδιο ή διασυνδεδεμένους υπολογιστικούς κόμβους. Το λογισμικό αυτό χρησιμοποιήθηκε για την ανάλυση μοντέλου αεροσκάφους. Σημειώνεται ότι η χρήση τριών Tesla M2050 του ίδιου κόμβου επιτάχυνε την πρόλεξη χρονικά μη-μόνιμης ροής γύρω από το αεροσκάφος περίπου 6 φορές σε σχέση με τον αντίστοιχο CPU-κώδικα όταν εκείνος εκτελείται σε μία συστοιχία από 8 οκταπύρηνες διασυνδεδεμένες CPUs (συνολικά δηλαδή 64 CPU-πυρήνες). Αυτό πρακτικά σημαίνει ότι προβλήματα ροής, η επίλυση των οποίων απαιτεί περίπου τρεις ημέρες σε 64 CPU-πυρήνες, επιλύονται σε μόλις 12 ώρες χρησιμοποιώντας τρεις Tesla M2050 του ίδιου υπολογιστικού κόμβου και τη μικτής ακρίβειας εκδοχή του GPU-λογισμικού της διατριβής. Η υπολογιστική πλατφόρμα της ΜΠΥΡ&Β/ΕΘΣ, περιγράφεται στην ενότητα 1.9.

Πέρα του τομέα υπολογιστικής ρευστοδυναμικής, έχουν δημοσιευθεί αρκετές εργασίες σε συστοιχίες καρτών γραφικών και σε άλλους επιστημονικούς τομείς, όπως της μοριακής δυναμικής [79], της μετεωρολογίας [80], της σεισμολογίας [81, 82, 83, 84] ή γενικά στον τομέα της γεωφυσικής [85], κ.α. Η επιτάχυνση της ταξινόμησης εγγράφων (Document Clustering) χρησιμοποιώντας συστοιχίες από GPUs παρουσιάζεται στην εργασία [86].

1.7 Περιβάλλον και γλώσσες προγραμματισμού των GPUs

Καθοριστικό παράγοντα στην επέκταση της χρήσης των GPUs από τα ηλεκτρονικά παιχνίδια στην επίλυση επιστημονικών προβλημάτων, αποτέλεσε η ανάπτυξη κατάλληλων γλωσσών προγραμματισμού για την εκμετάλλευση της μεγάλης υπολογιστικής ισχύος των σύγχρονων GPUs. Την τελευταία δεκαετία αναπτύχθηκαν αρκετές γλώσσες προγραμματισμού χαμηλού (CTM, [87]) και υψηλού επιπέδου (Cg [88], GLSL [89], HLSL [90], CGiS [91], Brook stream [92, 93], Sequoia [94], Scout [95], Accelerator [96], [97]), το οποίο αποτελεί και ένδειξη αλλαγής στη στάση της επιστημονικής κοινότητας απέναντι στις GPUs.

Ανάμεσα στις γλώσσες υψηλού επιπέδου προγραμματισμού που αναπτύχθηκαν στην τελευταία δεκαετία ξεχωρίζουν:

(α') Η Cg (C for Graphics), [88], που έχει αναπτυχθεί από την NVIDIA σε συνεργασία με τη Microsoft και απευθύνεται μόνο στον προγραμματισμό GPUs της NVIDIA. Στην εργασία [98] περιλαμβάνεται δείγμα GPU-κώδικα σε Cg για τον υπολογισμό της λαπλασιανής βαθμωτού μεγέθους.

(β') Η HLSL (High Level Shading Language), [90] αναπτύχθηκε από τη Microsoft παράλληλα με τη Cg και υποστηρίζεται μόνο από το λειτουργικό σύστημα των Windows.

(γ') Η GLSL (OpenGL Shading Language), [89], που αναπτύχθηκε από τη OpenGL Architecture Review Board, ARB και υποστηρίζει GPUs της NVIDIA και της AMD.

(δ') Η Brook stream, [92, 93], που έχει αναπτυχθεί από το πανεπιστήμιο του Stanford και μπορεί να χρησιμοποιηθεί σε GPUs είτε της NVIDIA είτε της AMD.

Σημειώνεται ότι και η AMD (πρώην ATI) ανέπτυξε τη CTM (Close To Metal), [87], που είναι μία γλώσσα προγραμματισμού χαμηλού επιπέδου αποκλειστικά για τις GPUs της ίδιας εταιρίας. Η γλώσσα αυτή όμως εγκαταλείφθηκε σχετικά γρήγορα, κυρίως εξαιτίας της ύπαρξης γλωσσών προγραμματισμού υψηλού επιπέδου.

Παρά το γεγονός ότι οι γλώσσες προγραμματισμού που αναφέρθηκαν προηγουμένως αποσκοπούσαν στη χρήση των καρτών γραφικών για όχι γραφικούς λόγους, η ανάπτυξη ακόμα και ενός χαμηλής παράλληλης απόδοσης GPU-κώδικα απαιτούσε πολύ καλή γνώση του τρόπου λειτουργίας των καρτών γραφικών. Στην ουσία, ο προγραμματιστής προγραμματίζει τμήματα κώδικα (shaders) υπεύθυνα για το χρωματισμό των εικονοστοιχείων (PICTure ELements, ή pixels) της οθόνης του υπολογιστή. Το χρώμα κάθε pixel καθορίζεται από τέσσερις πραγματικούς αριθμούς απλής ακρίβειας στη βάση RBGA (Red Green Blue Alpha, η ποσότητα Alpha καθορίζει το ποσοστό της αδιαφάνειας). Δηλαδή, για την επίλυση ενός 2Δ προβλήματος αεροδυναμικής όπου χρησιμοποιείται ένα δομημένο πλέγμα με διάσταση ίση με τη μέγιστη ανάλυση της οθόνης του υπολογιστή (π.χ. 1920×1080 σε σημερινές υψηλής ανάλυσης οθόνες), κάθε pixel της οθόνης αντιστοιχεί σε έναν κόμβο του πλέγματος. Οι μεταβλητές της ροής σε κάθε κόμβο καθορίζουν το χρωματισμό του αντίστοιχου pixel στη βάση RGBA, όπου 'R' η πυκνότητα, 'G', 'B' οι καρτεσιανές συνιστώσες της ταχύτητας και 'A' η

πίεση στον κόμβο. Τα δεδομένα αποθηκεύονται σε τετράδες (στη λογική RGBA) σε textures που αποτελούν τμήματα της μνήμης της κάρτας γραφικών. Πρακτικά, δηλαδή, ένα texture αντιστοιχεί σε έναν πίνακα του αντίστοιχου CPU-κώδικα. Βασικό μειονέκτημα των γλωσσών αυτών προγραμματισμού, αποτέλεσε η έλλειψη δυνατότητας συγχρονισμού ή επικοινωνίας μεταξύ των διεργασιών που εκτελούνται ταυτόχρονα στη GPU (μία διεργασία ανά pixel). Επιπλέον, οι γλώσσες αυτές προγραμματισμού δεν επέτρεπαν την αποθήκευση στη μνήμη των καρτών γραφικών με τυχαίο τρόπο, αλλά οι διεργασίες που εκτελούνται παράλληλα σε μία GPU, έπρεπε να ακολουθούν ένα δομημένο πρότυπο πρόσβασης στη μνήμη.

Την τελευταία πενταετία αναπτύχθηκαν περιβάλλοντα προγραμματισμού με στόχο τον προγραμματισμό των GPUs χωρίς να απαιτείται η πολύ καλή γνώση του τρόπου λειτουργίας αυτών, τουλάχιστον για τη δημιουργία ενός χαμηλής παράλληλης απόδοσης GPU-κώδικα. Τα περιβάλλοντα αυτά προγραμματισμού, σε αντίθεση με τις γλώσσες προγραμματισμού που αναφέρθηκαν προηγουμένως, επιτρέπουν το συγχρονισμό και την επικοινωνία (μεταφορά δεδομένων) μεταξύ των διεργασιών που εκτελούνται ταυτόχρονα στους επεξεργαστές της κάρτας γραφικών. Επιπλέον, επιτρέπεται οι διεργασίες να έχουν πρόσβαση στη μνήμη της κάρτας με βάση ένα μη-δομημένο πρότυπο. Οι δυνατότητες αυτές αυξάνουν το εύρος των αριθμητικών μεθόδων που μπορούν να εφαρμοστούν σε GPUs. Συνεπώς, ο προγραμματιστής έχει μεγαλύτερη ευχέρεια στην επιλογή της αριθμητικής μεθόδου που θα εφαρμόσει, με στόχο την αύξηση της παράλληλης απόδοσης του GPU-κώδικα.

Τα κυριότερα περιβάλλοντα προγραμματισμού που σήμερα είναι διαθέσιμα είναι:

(α') Η αρχιτεκτονική παράλληλης επεξεργασίας CUDA (Compute Unified Device Architecture [99]) που έχει αναπτυχθεί από την NVIDIA και παρουσιάστηκε για πρώτη φορά τον Νοέμβριο του 2006. Το περιβάλλον προγραμματισμού της CUDA είναι μία επέκταση της C++, (αρχικά συμπεριλάμβανε περιορισμένες λειτουργίες της C++, οι τελευταίες εκδόσεις όμως υποστηρίζουν όλες τις λειτουργίες αυτής) με την προσθήκη ορισμένων νέων τύπων μεταβλητών που βοηθούν στον αποδοτικότερο προγραμματισμό της εκάστοτε NVIDIA GPU. Επίσης, περιλαμβάνονται κατάλληλες συναρτήσεις ικανές για το χειρισμό και την καθοδήγηση της GPU από τη CPU, όπως λ.χ. συναρτήσεις δέσμευσης/αποδέσμευσης θέσεων της κεντρικής μνήμης της GPU, μεταφοράς δεδομένων από την κεντρική μνήμη του υπολογιστή σε εκείνη της GPU και αντιστρόφως, ή ακόμα και μεταφοράς πληροφορίας από μία θέση μνήμης της κάρτας γραφικών σε μία άλλη της ίδιας κάρτας. Μόλις προσφάτως, προστέθηκε η επιλογή της FORTRAN ως γλώσσας προγραμματισμού.

(β') Το περιβάλλον προγραμματισμού της OpenCL (Open Computing Language [100, 101]) που, όπως και το αντίστοιχο της CUDA, επιτρέπει την ανάπτυξη λογισμικού ικανού να εκτελείται σε ετερογενείς πλατφόρμες (δηλαδή σε CPUs και GPUs). Αναπτύχθηκε από την Khronos και δόθηκε στην επιστημονική κοινότητα τον Ιούνιο του 2010. Αντίστοιχα με το περιβάλλον της CUDA αποτελεί επέκταση της C με επιπλέον λειτουργίες για την επιλογή και τη διαχείριση της πλατφόρμας όπου θα γίνει η επεξεργασία-διαχείριση των δεδομένων. Το εν λόγω

περιβάλλον υποστηρίζεται από τις GPUs της NVIDIA και της AMD.

Στη διατριβή αυτή προτιμήθηκε η χρήση του περιβάλλοντος προγραμματισμού της CUDA αντί μίας γλώσσας προγραμματισμού όπως η Cg ή η GLSL. Αξίζει να αναφερθεί ότι, στην εργασία [28], η απόδοση του ίδιου GPU-κώδικα προγραμματισμένου στο περιβάλλον προγραμματισμού της CUDA εκτελείται 2 φορές πιο γρήγορα σε μία GeForce 8800 GTX, αντί του ίδιου κώδικα προγραμματισμένου σε Cg. Ανάλογο είναι το συμπέρασμα της εργασίας [50], όπου παρατηρήθηκε περίπου 10% αύξηση της απόδοσης του GPU-κώδικα όταν εκείνος γράφτηκε στο περιβάλλον προγραμματισμού της CUDA αντί σε GLSL. Στην εργασία [48] προγραμματίστηκε GPU-επιλύτης των εξισώσεων Euler σε Brook stream. Παρά την ικανοποιητική απόδοση της 2Δ εκδοχής του GPU-επιλύτη σε μία ATI 1950XT (επιτάχυνση έως 29x), η απόδοση της 3Δ εκδοχής του ίδιου επιλύτη στην ίδια κάρτα γραφικών ήταν πολύ χαμηλή (επιτάχυνση μόλις 3x). Στη συνέχεια, ο GPU-επιλύτης προγραμματίστηκε στο προγραμματιστικά πιο 'ευέλικτο' περιβάλλον προγραμματισμού της CUDA, επιταχύνοντας την πρόλεξη της ροής έως και 16 φορές σε μία GeForce 8800 GTX κάρτα γραφικών, αντίστοιχης τεχνολογίας με την ATI 1950XT. Οι διαφορές αυτές στο χρόνο εκτέλεσης προστίθεται στα μειονεκτήματα των πρώτων γλωσσών προγραμματισμού καρτών γραφικών που αναφέρθηκαν προηγουμένως σχετικά με τη διαχείριση των παράλληλων διεργασιών που εκτελούνται στην κάρτα γραφικών και τους περιορισμούς στον τρόπο προσπέλασης της μνήμης.

Στην παρούσα διατριβή, η επιλογή του περιβάλλοντος προγραμματισμού της CUDA, περιόρισε σημαντικά τον τύπο των GPUs που μπορούν να χρησιμοποιηθούν αποκλειστικά σε εκείνες της NVIDIA. Αντίθετα, ο ίδιος κώδικας προγραμματισμένος στο περιβάλλον προγραμματισμού της OpenCL μπορεί να εκτελεστεί σε GPUs τόσο της NVIDIA όσο και της AMD, αλλά και σε CPUs στην περίπτωση που ο υπολογιστικός κόμβος δεν περιέχει GPU ή περιέχει μία όχι τελευταίας τεχνολογίας. Βέβαια, η εκτέλεση ενός κώδικα, που εφόσον είναι προγραμματισμένος να εκτελείται σε GPUs καλεί την εκτέλεση εκατοντάδων ή ακόμα και χιλιάδων παράλληλων διεργασιών, σε μία CPU (που υστερεί ως προς το πλήθος των διαθέσιμων πυρήνων σε σχέση με τις σύγχρονες GPUs), δεν χαρακτηρίζεται ως συνετή. Το περιβάλλον προγραμματισμού της OpenCL όμως, όπως αναφέρθηκε προηγουμένως, δόθηκε στην επιστημονική κοινότητα τον Ιούνιο του 2010. Την περίοδο εκείνη είχε ήδη προγραμματιστεί ο GPU-επιλύτης των εξισώσεων Navier–Stokes χρησιμοποιώντας το περιβάλλον προγραμματισμού της CUDA.

Επιπλέον, οι GPUs της AMD και της NVIDIA μπορεί να ακολουθούν τον ίδιο τρόπο λειτουργίας, στηρίζονται όμως σε διαφορετική αρχιτεκτονική. Αυτό σημαίνει ότι ο ίδιος κώδικας δεν μπορεί να έχει τη μέγιστη δυνατή παράλληλη απόδοση στις GPUs και των δύο εταιριών.

Στην εργασία [102] παρουσιάζεται μία πρώτη προσπάθεια μεταφοράς κώδικα από το περιβάλλον προγραμματισμού της CUDA σε εκείνο της OpenCL με τις ελάχιστες δυνατές επεμβάσεις στον αρχικό κώδικα. Η τελική OpenCL εκδοχή του GPU-κώδικα είχε αρκετά χαμηλότερη παράλληλη απόδοση σε σχέση με την αρχική CUDA εκδοχή του ίδιου κώδικα. Η μείωση της παράλληλης επιτάχυνσης είναι περίπου 50%. Μέρος της πολύ σημαντικής αυτής πτώσης στη παράλληλη απόδοση του GPU-κώδικα οφείλεται

στην απόδοση των μεταφραστών (compilers) της CUDA και της OpenCL. Συγκεκριμένα, σύμφωνα με την εργασία [102], ο compiler της CUDA δημιουργεί αποδοτικότερο εκτελέσιμο λογισμικό, με μικρότερο πλήθος εκτελέσιμων οδηγιών ανά διεργασία. Η μείωση του αριθμού εκτελέσιμων οδηγιών ανά διεργασία μειώνει το πλήθος των καταχωρητών (registers) που χρησιμοποιούνται ανά διεργασία και πολυεπεξεργαστή. Όπως αναλύεται στην παράγραφο 3.14.1, η χρήση λιγότερων registers ανά πολυεπεξεργαστή και διεργασία αυξάνει την παράλληλη απόδοση του GPU-κώδικα. Σημειώνεται ότι η εν λόγω εργασία δημοσιεύτηκε στις αρχές του 2011 και χρησιμοποιήθηκαν η NVIDIA Tesla c1060 και οι 3.1, 2.2 εκδόσεις του περιβάλλοντος προγραμματισμού της CUDA και της OpenCL, αντίστοιχα.

Σε ανάλογο συμπέρασμα καταλήγουν και οι εργασίες [103, 104]. Στη δεύτερη, μεταξύ άλλων, προγραμματίστηκε μία CUDA και μία OpenCL εκδοχή του ίδιου κώδικα. Ο κώδικας αυτός χρησιμοποιεί την τεχνική Monte Carlo για την προσομοίωση χημικών διεργασιών. Η σύγκριση της παράλληλης απόδοσης των CUDA και OpenCL εκδοχών έγινε σε μία NVIDIA GTX 285. Φάνηκε ότι η απόδοση της OpenCL εκδοχής υστερούσε σε σχέση με εκείνην της CUDA.

Σε συνέχεια των προηγούμενων, το περιβάλλον προγραμματισμού της OpenCL δεν υποστηρίζει ακόμα ορισμένες λειτουργίες της C++, σε αντίθεση με το περιβάλλον προγραμματισμού της CUDA που υποστηρίζει πλήρως τη C++ ως γλώσσα προγραμματισμού. Μάλιστα, από τα μέσα του 2010, το περιβάλλον της CUDA υποστηρίζει και τη FORTRAN ως γλώσσα προγραμματισμού.

Κλείνοντας την ενότητα, αξίζει να αναφερθεί ότι έχουν αναπτυχθεί, από διαφορετικές ομάδες, βιβλιοθήκες συμβατές με το περιβάλλον προγραμματισμού της CUDA, καθιστώντας περισσότερο προσιτό τον προγραμματισμό των GPUs. Ο αναγνώστης μπορεί να βρει κατάλογο τέτοιων βιβλιοθηκών στην ιστοσελίδα [105], αρκετές από τις οποίες διατίθενται ελεύθερα στην επιστημονική κοινότητα. Οι βιβλιοθήκες αυτές περιλαμβάνουν συναρτήσεις εκτέλεσης βασικών υπολογισμών γραμμικής άλγεβρας (λ.χ. πολλαπλασιασμό πινάκων κ.α.), μετασχηματισμού Fourier, κ.α. Η επιτάχυνση των συναρτήσεων αυτών εξαρτάται από το μέγεθος των δεδομένων που επεξεργάζονται παράλληλα και, συνήθως, είναι μεγαλύτερη του 10x. Στην εργασία [106] αξιολογούνται ως προς το ρυθμό εκτέλεσης αριθμητικών πράξεων κινητής υποδιαστολής ορισμένες συναρτήσεις της βιβλιοθήκης CUBLAS, που είναι η CUDA επέκταση της βιβλιοθήκης BLAS (Basic Linear Algebra Subroutines) για CPUs. Στην εργασία [16] χρησιμοποιούνται συναρτήσεις των βιβλιοθηκών CUBLAS και CUSPARSE για την ατελή (incomplete) παραγοντοποίηση συμμετρικού και μη-συμμετρικού μητρώου σε άνω και κάτω τριγωνικά μητρώα (LU decompsition και μέθοδος Cholesky).

1.8 Τεχνικές προγραμματισμού σε GPUs

Ο διαφορετικός τρόπος λειτουργίας των σύγχρονων GPUs, σε σχέση με τις σημερινές CPUs, επιβάλλει τη διαφορετική διαχείριση και προγραμματισμό αυτών με στόχο τη μέγιστη αξιοποίηση των υπολογιστικών δυνατοτήτων τους. Προγραμματιστικές τεχνικές υψηλής απόδοσης σε CPUs δεν έχουν την ανάλογη απόδοση, ως προς την αξιοποίηση των διαθέσιμων υπολογιστικών πόρων, όταν εφαρμόζονται σε GPUs. Στο συμπέρασμα αυτό καταλήγει και η εργασία [107]. Προγραμματιστικά θέματα γενικών πολυπύρηνων συστημάτων παράλληλης επεξεργασίας με κοινή μνήμη (τέτοια είναι οι σύγχρονες κάρτες γραφικών) θίγονται στις εργασίες [108, 109].

Γενικά, ο προγραμματισμός GPU-κώδικα υψηλής παράλληλης απόδοσης θέτει τους ακόλουθους στόχους:

(α') Ελαχιστοποίηση του πλήθους των προσβάσεων στη κεντρική μνήμη της κάρτας γραφικών.

(β') Ελάττωση των καταχωρητών (registers) που χρησιμοποιούνται ανά πολυεπεξεργαστή και διεργασία.

(γ') Μεγιστοποίηση της απόδοσης των ενδιάμεσων μνημών (cache μνήμες) της κάρτας γραφικών.

Ένα παράδειγμα προγραμματισμού GPU-κώδικα αυξημένης απόδοσης για τον πολλαπλασιασμό δύο μητρώων παρουσιάζεται στην εργασία [110]. Το παράδειγμα αυτό ικανοποιεί τα δύο πρώτα κριτήρια. Επιπλέον βελτίωση του GPU-κώδικα της εν λόγω εργασίας μπορεί να επιτευχθεί με την αύξηση της απόδοσης των ενδιάμεσων μνημών της κάρτας γραφικών. Η εργασία [111] παρουσιάζει ένα μοντέλο αξιολόγησης του βαθμού απόδοσης του πρότυπου προσπέλασης της μνήμης της κάρτας γραφικών που χρησιμοποιείται από ένα GPU-κώδικα.

Πρόκληση αποτελεί ο προγραμματισμός GPU-κώδικα που χρησιμοποιεί μη-δομημένα υπολογιστικά πλέγματα. Οι εργασίες [33, 34] προτείνουν την ομαδοποίηση των τριγωνικών/τετραεδρικών πλεγματικών στοιχείων του χρησιμοποιούμενου $2\Delta/3\Delta$ υπολογιστικού πλέγματος, έτσι ώστε διεργασίες που εκτελούνται ταυτόχρονα στη GPU να μην αποθηκεύουν δεδομένα σε ίδιες θέσεις μνήμης. Παρόλα αυτά, στην παρούσα διατριβή, όπως αναλύεται στο κεφάλαιο 4 και παρουσιάζεται επίσης στην εργασία [58], οι τεχνικές ομαδοποίησης των ακμών ενός μη-δομημένου υπολογιστικού πλέγματος που δοκιμάστηκαν δεν συνέβαλαν στη δημιουργία GPU-κώδικα υψηλής παράλληλης απόδοσης. Για το λόγο αυτό, προτείνονται εναλλακτικές προγραμματιστικές τεχνικές οι οποίες συνοδευόμενες από κατάλληλα πρότυπα προσπέλασης της μνήμης της κάρτας γραφικών οδηγούν στον προγραμματισμό GPU-κώδικα υψηλής παράλληλης απόδοσης. Τονίζεται ότι, παρόλο που η διατριβή επικεντρώνεται στην επίλυση προβλημάτων αεροδυναμικής και αεροελαστικότητας, οι προγραμματιστικές τεχνικές που αναλύονται, εφαρμόζονται σε οποιοδήποτε GPU-επιλύτη συστημάτων μερικών διαφορικών εξισώσεων.

Στην εργασία [52] επαναριθμούνται τα πλεγματικά στοιχεία του χρησιμοποιούμενου μη-δομημένου υπολογιστικού πλέγματος, σύμφωνα με την εργασία [112], έτσι ώστε

γειτονικά πλεγματικά στοιχεία να έχουν το δυνατόν πλησιέστερο αύξοντα αριθμό. Με την επαναρίθμηση των πλεγματικών στοιχείων, διεργασίες που εκτελούνται ταυτόχρονα έχουν πρόσβαση σε πλησιέστερες θέσεις μνήμης. Σημειώνεται, ότι στην εργασία αυτή, οι τιμές των μεταβλητών της ροής αποθηκεύονται στα πλεγματικά στοιχεία. Αντίθετα, στη διατριβή επαναριθμούνται οι κόμβοι του πλέγματος, εφόσον αποθηκεύονται εκεί οι ροϊκές μεταβλητές. Η επαναρίθμηση γίνεται σύμφωνα με τη μέθοδο των Cuthil McKee, [113].

1.9 Περιγραφή της υπολογιστικής πλατφόρμας της ΜΠΥΡ&Β/ΕΘΣ

Στη διδακτορική αυτή διατριβή χρησιμοποιήθηκαν GPUs της NVIDIA αρχιτεκτονικής CUDA. Συγκεκριμένα, χρησιμοποιήθηκαν οι GeForce GTX 280 και 285 (σχήμα 1.4(α′)) και Tesla M2050 (σχήμα 1.4(β′)).

(α′) (β′)

Σχήμα 1.4: Οι GPUs της NVIDIA (α′) GeForce GTX 280 και 285, και (β′) Tesla M2050, που χρησιμοποιήθηκαν στη διατριβή αυτή.

Η ταχύτητα προσπέλασης της κεντρικής μνήμης μίας GTX 280 είναι 141.7 GByte/sec. Η ταχύτητα εκτέλεσης αριθμητικών πράξεων κινητής υποδιαστολής της ίδιας κάρτας είναι 933 Gflops για πραγματικούς αριθμούς απλής ακρίβειας και 78 Gflops για πραγματικούς αριθμούς διπλής ακρίβειας, [114]. Οι αντίστοιχες αποδόσεις των GTX 285 είναι ελαφρώς αυξημένες, όντας ίσες με 159 GByte/sec και 1063 Gflops για πραγματικούς αριθμούς απλής ακρίβειας, ή 89 Gflops για πραγματικούς αριθμούς διπλής ακρίβειας, [115]. Η χωρητικότητα της κεντρικής μνήμης των δύο μοντέλων καρτών γραφικών είναι 1 GByte. Επιλέχθηκε οι κάρτες αυτές να τοποθετηθούν σε σχετικά παλιούς προσωπικούς υπολογιστές αναβαθμίζοντάς τους. Αποδείχθηκε, όπως αναμενόταν, ότι η απόδοση των GPUs δεν εξαρτάται από την ηλικία του υπολογιστή ή την τεχνολογία της CPU. Συγκεκριμένα, οι υπολογιστές στους οποίους εγκαταστάθηκαν οι GPUs έχουν διπύρηνους επεξεργαστές τεχνολογίας Intel Pentium D με συχνότητα 3.00 GHz και cache μνήμη 1024 KByte. Το μέγεθος της μνήμης RAM του κάθε υπολογιστή είναι 4 GByte. Η εικόνα των υπολογιστών αυτών με τοποθετημένες τις GPUs είναι αντίστοιχη του σχήματος 1.1(β′). Η πρώτη GTX 280 αποκτήθηκε στα τέλη του 2009 ενώ, το 2010, αποκτήθηκαν οι υπόλοιπες GTX 285 που χρησιμοποιήθηκαν στη διατριβή.

Στις αρχές του 2011, εγκαταστάθηκε στη ΜΠΥΡ&Β/ΕΘΣ μία τελευταίας τεχνολογίας συστοιχία (cluster) τεσσάρων διασυνδεδεμένων υπολογιστικών κόμβων (blade servers) της Hewlett Packard (HP). Κάθε υπολογιστικός κόμβος περιέχει τρεις Tesla M2050. Οι εν λόγω GPUs, μαζί με τις Tesla M2070, αποτελούσαν τα τελευταία μοντέλα της NVIDIA εκείνη την περίοδο. Η απόδοση των δύο μοντέλων είναι η ίδια. Δηλαδή, ο μέγιστος ρυθμός προσπέλασης της κεντρικής μνήμης είναι 148 GByte/sec και η θεωρητικά μέγιστη ταχύτητα εκτέλεσης αριθμητικών πράξεων κινητής υποδιαστολής είναι 515 Gflops για πραγματικούς αριθμούς διπλής ακρίβειας και 1030 Gflops για αριθμούς απλής ακρίβειας [116]. Η διαφορά των δύο μοντέλων έχει να κάνει με τη διάσταση των κεντρικών τους μνημών. Η M2050 έχει 3 GByte κεντρική μνήμη, ενώ η M2070 6 GByte. Την περίοδο όμως εκείνη, μόνο η M2050 ήταν διαθέσιμη στην Ελλάδα. Αργότερα, η NVIDIA κυκλοφόρησε τις Tesla M2090. Κάθε υπολογιστικός κόμβος έχει δύο τετραπύρηνους (quad-core) επεξεργαστές τεχνολογίας Intel Xeon (E5620) στα 2.40 GHz με 12288 KByte cache μνήμη. Η χωρητικότητα της μνήμης RAM ανά υπολογιστικό κόμβο είναι 16 GByte. Το σχήμα 1.5 δείχνει έναν υπολογιστικό κόμβο της HP. Η προμήθεια του εν λόγω υπολογιστικού συστήματος έγινε από το Αναπτυξιακό Πρόγραμμα Εργαστηριακών Υποδομών 2010 του ΕΜΠ.

Συνεπώς, εγκαταστάθηκε μία συστοιχία από $4 \times 3 = 12$ GPUs. Η συστοιχία αυτή χρησιμοποιείται για την επίλυση αεροδυναμικών και αεροελαστικών προβλημάτων μεγάλης κλίμακας. Η διάσταση των υπολογιστικών πλεγμάτων που χρησιμοποιήθηκαν ήταν αποτρεπτική για τη χρήση μόνο μίας GPU, καθώς χρειαζόταν μνήμη μεγαλύτερη των 3 GByte μίας Tesla M2050. Επομένως, ήταν επιβεβλημένη η παραλληλοποίηση του GPU-επιλύτη των 3Δ Navier–Stokes εξισώσεων, ώστε να μπορεί να εκτελείται σε περισσότερες από μία GPUs του ίδιου ή διαφορετικών υπολογιστικών κόμβων. Σχετικά θέματα παρουσιάζονται στο κεφάλαιο 5.

Σχήμα 1.5: Blade server της HP με τρεις ενσωματωμένες Tesla M2050. Στις αρχές του 2011, εγκαταστάθηκε στη ΜΠΥΡ&Β/ΕΘΣ συστοιχία τεσσάρων τέτοιων διασυνδεδεμένων υπολογιστικών κόμβων.

1.10 Δομή της διατριβής

Η ολοκλήρωση των χρονικά μη-μόνιμων εξισώσεων Navier–Stokes, εκφρασμένων σε μη-αδρανειακό σύστημα που μετακινείται και περιστρέφεται με μεταβαλλόμενο ρυθμό ως προς αδρανειακό σύστημα, σε όγκους ελέγχου με μεταβαλλόμενα όρια παρουσιάζεται στο κεφάλαιο 2. Οι όγκοι ελέγχου σχηματίζονται γύρω από τους κόμβους μη-δομημένου (υβριδικού) υπολογιστικού πλέγματος, σύμφωνα με την κεντροκομβική διατύπωση της τεχνικής των πεπερασμένων όγκων.

Το κεφάλαιο 3 περιγράφει τον τρόπο λειτουργίας των σύγχρονων προγραμματιζόμενων GPUs δίνοντας έμφαση σε κάρτες γραφικών αρχιτεκτονικής GT200 και Fermi, που είναι οι τελευταίες αρχιτεκτονικές καρτών γραφικών που αναπτύχθηκαν από την NVIDIA και στις οποίες στηρίζονται οι κάρτες γραφικών που χρησιμοποιήθηκαν στη διατριβή. Ακολουθεί η παρουσίαση του περιβάλλοντος προγραμματισμού της CUDA και δείχνεται πώς, μέσω αυτού, είναι δυνατή η αξιοποίηση των υπολογιστικών πόρων των GPUs.

Τα κεφάλαια 4 και 5 παρουσιάζουν τον παράλληλο, σε πολλές κάρτες γραφικών του ίδιου ή διασυνδεδεμένων υπολογιστικών κόμβων, GPU-κώδικα της διατριβής. Ο εν λόγω GPU-κώδικας επιλύει τις 2Δ χρονικά μη-μόνιμες εξισώσεις Navier–Stokes και τις 3Δ χρονικά μη-μόνιμες εξισώσεις Euler πολύ πιο γρήγορα σε σχέση με τον αντίστοιχο CPU-κώδικα. Το κεφάλαιο 4 αναλύει τις προγραμματιστικές τεχνικές που ακολουθήθηκαν για την επίτευξη υψηλής παράλληλης επιτάχυνσης σε μία GPU. Έμφαση δίνεται στην αποδοτική διαχείριση της μνήμης. Παρουσιάζεται, επίσης, μία τεχνική επαναρίθμησης και ταξινόμησης των κόμβων του πλέγματος με στόχο την αύξηση της παράλληλης επιτάχυνσης του GPU-κώδικα. Τονίζεται η χρήση αριθμητικής μικτής ακρίβειας για την επιπλέον αύξηση της παράλληλης απόδοσης του GPU-κώδικα χωρίς την αλλοίωση της ακρίβειας των αριθμητικών προλέξεων. Η απόδοση του GPU-κώδικα σε πολλές κάρτες γραφικών του ίδιου ή διασυνδεδεμένων κόμβων παρουσιάζεται στο κεφάλαιο 5. Στο ίδιο κεφάλαιο περιγράφονται οι δυνατοί τρόποι επικοινωνίας δύο ή περισσότερων GPUs.

Ο GPU-κώδικας της διατριβής χρησιμοποιείται για την αεροδυναμική ανάλυση αεροσκάφους και τη μελέτη της ροής σε πτερύγωση υπερηχητικού συμπιεστή. Οι αριθμητικές προλέξεις παρουσιάζονται στο κεφάλαιο 6. Άλλα προβλήματα αεροδυναμικής που επιλύονται χρησιμοποιώντας τον GPU-κώδικα της διατριβής και παρουσιάζονται στο ίδιο κεφάλαιο, είναι η μελέτη της αλληλεπίδρασης οριακού στρώματος και κύματος κρούσης στην πλευρά υποπίεσης αεροτομής και η μελέτη της επίδρασης σύνθετης δέσμης ρευστού στον έλεγχο της ροής γύρω από διαμόρφωση. Τονίζεται η αξιοποίηση 'τεχνολογικά ξεπερασμένων' προσωπικών υπολογιστών, που αναβαθμίστηκαν, με μικρό κόστος, σε σύγχρονες μονάδες παράλληλης επεξεργασίας δεδομένων, ενσωματώνοντας μία προγραμματιζόμενη GPU. Με την πραγματοποίηση των εφαρμογών αυτών χρησιμοποιώντας GPUs όχι τελευταίας τεχνολογίας αποδεικνύεται, επιπλέον, ότι, ακόμα και σήμερα, η χρήση τεχνολογικά ξεπερασμένων GPUs έχει νόημα καθώς επιταχύνουν σημαντικά την επίλυση επιστημονικών προβλημάτων έχοντας χαμηλό κόστος κτήσης.

Η μικτής ακρίβειας εκδοχή του GPU-κώδικα χρησιμοποιείται ως λογισμικό αξιο-

λόγησης σε ΕΑ για την εύρεση των βέλτιστων παραμέτρων δέσμης συνεχούς αναρρόφησης ρευστού με στόχο την ελαχιστοποίηση της περιοχής αποκόλλησης κατάντι μίας διαμόρφωσης στο εσωτερικό αγωγού. Τα αποτελέσματα της βελτιστοποίησης φαίνονται στο κεφάλαιο 7. Το ίδιο κεφάλαιο δείχνει το σχεδιασμό μεμονωμένης αεροτομής με στόχους τη μεγιστοποίηση του συντελεστή άνωσης και την ελαχιστοποίηση του συντελεστή οπισθέλκουσας αλλά και το σχεδιασμό αεροτομής συμπιεστή με στόχο την ελαχιστοποίηση των απωλειών πίεσης. Στις δύο τελευταίες εφαρμογές χρησιμοποιήθηκε διεπίπεδος ΕΑ, όπου οι απλής και μικτής ακρίβειας εκδοχές του GPU-κώδικα χρησιμοποιούνται ως λογισμικά αξιολόγησης στο χαμηλό και στο υψηλό επίπεδο, αντίστοιχα. Στο ίδιο κεφάλαιο περιλαμβάνεται η περιγραφή του διεπίπεδου ΕΑ. Γενικά, η χρήση GPU-λογισμικών αξιολόγησης μειώνει σημαντικά το χρόνο της βελτιστοποίησης. Στο κέρδος αυτό υπερτίθεται το κέρδος από την παράλληλη αξιολόγηση των υποψήφιων λύσεων σε πολλές κάρτες γραφικών (μία αξιολόγηση ανά GPU) και το κέρδος από τη χρήση του διεπίπεδου αντί ενός κλασικού ΕΑ. Από τις εφαρμογές αυτές αναδεικνύεται ένας τρόπος χρήσης της, πολύ γρήγορης αλλά λιγότερο ακριβούς, απλής ακρίβειας εκδοχής του GPU-κώδικα.

Στο κεφάλαιο 8 μελετάται η αεροελαστική συμπεριφορά αεροτομής δύο και τριών βαθμών ελευθερίας και υπολογίζεται το όριο ευσταθούς λειτουργίας αυτής. Χρησιμοποιείται ένας απλός νόμος ελέγχου της γωνίας εκτροπής της ακμής εκφυγής της αεροτομής, μέσω ανάδρασης των μεταβλητών κατάστασης της αεροτομής, με στόχο την αύξηση του ορίου ευσταθούς λειτουργίας της αεροτομής. Δοκιμάζονται δύο διαφορετικά κέρδη για το βρόχο ανάδρασης.

Στο κεφάλαιο 9 παρουσιάζεται η ανάλυση μοντέλου επιβατικού αεροσκάφους τύπου Blended Wing Body (BWB) 450 θέσεων. Πρόκειται για μη-συμβατικό αεροσκάφος με πολλά πλεονεκτήματα από αεροδυναμικής πλευράς, όπως μεγαλύτερος λόγος άνωσης προς οπισθέλκουσα, μικρότερη κατανάλωση καυσίμου ανά επιβάτη και μίλι και μικρότερη εκπομπή θορύβου σε σχέση με συμβατικά αεροσκάφη ίδιων προδιαγραφών. Τα προτερήματα αυτά οφείλονται στη γεωμετρικά ομαλή μετάβαση από την άτρακτο στις πτέρυγες και στο σχήμα αεροτομής που έχουν οι τομές της ατράκτου κατά το μήκος του αεροσκάφους. Έτσι, η άτρακτος συμμετέχει πιο ενεργά στη δημιουργία άνωσης. Η ανάλυση του αεροσκάφους έγινε χρησιμοποιώντας την παράλληλη σε πολλές GPUs του ίδιου υπολογιστικού κόμβου εκδοχή του GPU-κώδικα, στο πλαίσιο ερευνητικού προγράμματος χρηματοδοτούμενου από την Ευρωπαϊκή Ένωση. Αν και το πρόγραμμα δεν απαιτούσε τη χρήση καρτών γραφικών, η χρήση GPUs αντί CPUs μετέτρεψε υπολογισμούς διάρκειας λίγων ημερών (σε συστοιχία διασυνδεδεμένων CPUs) σε υπολογισμούς λίγων ωρών (σε υπολογιστικό κόμβο με τρεις GPUs τελευταίας αρχιτεκτονικής).

Κεφάλαιο 2

Οι εξισώσεις Navier–Stokes και η διακριτοποίησή τους

Ένας από τους στόχους της παρούσας διατριβής είναι η ανάπτυξη λογισμικού για την πρόλεξη της αεροελαστικής συμπεριφοράς αεροδυναμικών σωμάτων. Συγκεκριμένα, το κεφάλαιο 8 παρουσιάζει την αεροελαστική ανάλυση αεροτομής που δύναται να μετακινείται και να περιστρέφεται απαραμόρφωτη, σύμφωνα με τα ασκούμενα σε αυτήν αεροδυναμικά φορτία. Η κίνηση της αεροτομής περιορίζεται από ένα γραμμικό και ένα στρεπτικό ελατήριο (όπως αναλύεται στο κεφάλαιο 8). Η ροή γύρω από την αεροτομή μπορεί να περιγραφεί από τις εξισώσεις Navier–Stokes εκφρασμένες σε (σχετικό) σύστημα αξόνων προσδεδεμένο στην αεροτομή και, άρα, κινούμενο μαζί της. Το κεφάλαιο αυτό παρουσιάζει τις εξισώσεις Navier–Stokes εκφρασμένες σε (σχετικό) σύστημα το οποίο δύναται να περιστρέφεται και να μετακινείται ως προς το αδρανειακό με χρονικά μεταβαλλόμενη ταχύτητα περιστροφής και μεταφοράς αντίστοιχα. Πρόκειται, δηλαδή, για την πιο γενική περίπτωση που αντιμετωπίζεται στην παρούσα διατριβή (εφαρμογές κεφαλαίου 8). Κάθε άλλη 'απλούστερη' περίπτωση εμπεριέχεται στις εξισώσεις που παρατίθενται στη συνέχεια, αφού προφανώς απαλειφθούν οι μη-απαραίτητοι όροι.

Εναλλακτικά, ο υπολογισμός της ροής γύρω από ένα αεροδυναμικό σώμα που μετακινείται, περιστρέφεται ή παραμορφώνεται γίνεται από την ολοκλήρωση των εξισώσεων Navier–Stokes σε αδρανειακό σύστημα, σε δυναμικά μεταβαλλόμενα υπολογιστικά πλέγματα. Τα πλέγματα αυτού του τύπου προσαρμόζονται δυναμικά στην κίνηση/παραμόρφωση της βρεχόμενης επιφάνειας του εξεταζόμενου σώματος. Στην παρούσα διατριβή, η ολοκλήρωση των εξισώσεων Navier–Stokes γίνεται σε κεντροκομβικούς όγκους ελέγχου. Λόγω της δυναμικής προσαρμογής του χρησιμοποιούμενου υπολογιστικού πλέγματος στην κίνηση/παραμόρφωση του σώματος, μπορεί να μεταβάλλονται τα όρια των όγκων ελέγχου. Στο κεφάλαιο αυτό, μετά την παρουσίαση των εξισώσεων Navier–Stokes σε σχετικό σύστημα αξόνων, περιγράφεται η διακριτοποίηση αυτών σε μη-δομημένα, δυναμικά προσαρμοζόμενα, υπολογιστικά πλέγματα. Η επίλυση των διακριτοποιημένων εξισώσεων της ροής σε χρονικά μόνιμα προβλήματα παρουσιάζεται, μεταξύ άλλων, στο κεφάλαιο 4, όπου περιγράφεται και ο προγραμματισμός επιλύτη των

εξισώσεων Navier–Stokes, υψηλής παράλληλης απόδοσης, σε GPUs. Για την επίλυση χρονικά μη-μόνιμων αεροδυναμικών προβλημάτων χρησιμοποιείται η μέθοδος του διπλού χρονικού βήματος (dual time stepping). Η ενότητα 8.5 περιγράφει την ταυτόχρονη επίλυση των εξισώσεων της ροής με τις ελαστικές εξισώσεις παραμορφώσιμου αεροδυναμικού σώματος.

2.1 Σχετικό σύστημα αξόνων

Πριν την παρουσίαση των εξισώσεων Navier–Stokes εκφρασμένων σε σχετικό σύστημα που δύναται να μετακινείται και να περιστρέφεται ως προς αδρανειακό (απόλυτο) σύστημα αξόνων, προηγείται η σύντομη περιγραφή του τρόπου υπολογισμού του διανύσματος της ταχύτητας και της επιτάχυνσης σε μία θέση του σχετικού συστήματος από τα αντίστοιχα διανύσματα εκφρασμένα στο απόλυτο σύστημα. Για το σκοπό αυτό χρησιμοποιούνται τρία καρτεσιανά συστήματα αξόνων, τα (x_r, y_r, z_r), (x_e, y_e, z_e) και (x, y, z), όπου:

(α') (x, y, z) είναι το αδρανειακό (απόλυτο) σύστημα αξόνων,

(β') (x_e, y_e, z_e) είναι το σύστημα αξόνων που μεταφέρεται με χρονικά μεταβαλλόμενη (στη γενική περίπτωση) μεταφορική ταχύτητα ($\overrightarrow{u}_b$) ως προς το αδρανειακό (x, y, z) και

(γ') (x_r, y_r, z_r) είναι το σύστημα αξόνων που περιστρέφεται με χρονικά μεταβαλλόμενη γωνιακή ταχύτητα ($\overrightarrow{\omega}_b$) ως προς το (x_e, y_e, z_e). Τα συστήματα (x_r, y_r, z_r) και (x_e, y_e, z_e) έχουν κοινή αρχή.

Αρχικά περιγράφεται στην παράγραφο 2.1.1 ο υπολογισμός των διανυσμάτων της ταχύτητας και της επιτάχυνσης στο (x_r, y_r, z_r) με βάση τα αντίστοιχα διανύσματα στο σύστημα (x_e, y_e, z_e). Στη συνέχεια, στην παράγραφο 2.1.2 δείχνεται ο υπολογισμός των διανυσμάτων της ταχύτητας και της επιτάχυνσης στο (x_r, y_r, z_r) με βάση τα αντίστοιχα διανύσματα στο αδρανειακό σύστημα (x, y, z).

2.1.1 Σύστημα αξόνων που περιστρέφεται ως προς άλλο σύστημα με κοινή αρχή

Αν $\overrightarrow{c}$ είναι ένα τυχαίο διάνυσμα, η χρονική παράγωγος αυτού ως προς το σύστημα αξόνων (x_e, y_e, z_e) δίνεται από τη σχέση

$$\frac{d_e \overrightarrow{c}}{dt} = \frac{d_r \overrightarrow{c}}{dt} + \overrightarrow{\omega}_b \times \overrightarrow{c} \tag{2.1}$$

όπου $\frac{d_r \overrightarrow{c}}{dt}$ η ολική παράγωγος του διανύσματος $\overrightarrow{c}$ όπως την αντιλαμβάνεται παρατηρητής στο σύστημα (x_r, y_r, z_r).

Συνεπώς, το διάνυσμα της ταχύτητας σε μία τυχαία θέση του χώρου ως προς το σύστημα (x_e, y_e, z_e) δίνεται από τη σχέση

$$\overrightarrow{u}_e \doteq \frac{d_e \overrightarrow{r}}{dt} = \frac{d_r \overrightarrow{r}}{dt} + \overrightarrow{\omega}_b \times \overrightarrow{r}$$

ή

$$\overrightarrow{u}_e = \overrightarrow{u}_r + \overrightarrow{\omega}_b \times \overrightarrow{r} \tag{2.2}$$

όπου $\overrightarrow{u}_r \doteq \frac{d_r \overrightarrow{r}}{dt}$ η ταχύτητα στο ίδιο σημείο όπως την αντιλαμβάνεται παρατηρητής στο σύστημα (x_r, y_r, z_r), με $\overrightarrow{r}$ το διάνυσμα θέσης.

Αντίστοιχα, η επιτάχυνση ως προς το σύστημα (x_e, y_e, z_e) υπολογίζεται ως εξής

$$\begin{aligned}
\overrightarrow{a}_e \doteq \frac{d_e^2 \overrightarrow{r}}{dt^2} &= \frac{d_e}{dt}\left(\frac{d_r \overrightarrow{r}}{dt} + \overrightarrow{\omega}_b \times \overrightarrow{r}\right) = \\
&= \frac{d_e}{dt}\left(\frac{d_r \overrightarrow{r}}{dt}\right) + \frac{d_e}{dt}\left(\overrightarrow{\omega}_b \times \overrightarrow{r}\right) = \\
&= \frac{d_e}{dt}\left(\frac{d_r \overrightarrow{r}}{dt}\right) + \frac{d_e \overrightarrow{\omega}_b}{dt} \times \overrightarrow{r} + \overrightarrow{\omega}_b \times \frac{d_e \overrightarrow{r}}{dt} = \\
&= \frac{d_r^2 \overrightarrow{r}}{dt^2} + \overrightarrow{\omega}_b \times \frac{d_r \overrightarrow{r}}{dt} + \frac{d_e \overrightarrow{\omega}_b}{dt} \times \overrightarrow{r} + \overrightarrow{\omega}_b \times \frac{d_r \overrightarrow{r}}{dt} + \overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r})
\end{aligned}$$

ή

$$\overrightarrow{a}_e = \overrightarrow{a}_r + \frac{d_e \overrightarrow{\omega}_b}{dt} \times \overrightarrow{r} + 2\overrightarrow{\omega}_b \times \frac{d_r \overrightarrow{r}}{dt} + \overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r}) \tag{2.3}$$

όπου $\overrightarrow{a}_r \doteq \frac{d_r^2 \overrightarrow{r}}{dt^2}$ είναι η επιτάχυνση ως προς το σύστημα (x_r, y_r, z_r).

Οι σχέσεις 2.2, 2.3 αποτελούν διανυσματικές εκφράσεις και μπορούν να εκφραστούν στο (x_r, y_r, z_r) ή στο (x_e, y_e, z_e), αρκεί οι συνιστώσες όλων των διανυσματικών όρων που περιέχονται στις παραπάνω εκφράσεις να αναφέρονται, προφανώς, στο ίδιο σύστημα.

2.1.2 Σύστημα αξόνων που περιστρέφεται και μετακινείται ως προς αδρανειακό σύστημα

Το σύστημα (x_e, y_e, z_e) μεταφέρεται με χρονικά μεταβαλλόμενη ταχύτητα $\overrightarrow{u}_b$ ως προς απόλυτο (αδρανειακό) σύστημα (x, y, z). Οι εκφράσεις 2.2, 2.3 παίρνουν αντίστοιχα τις μορφές

$$\overrightarrow{u} = \overrightarrow{u}_r + \overrightarrow{u}_b + \overrightarrow{\omega}_b \times \overrightarrow{r} \tag{2.4}$$

$$\overrightarrow{a} = \overrightarrow{a}_r + \underbrace{\frac{d\overrightarrow{u}_b}{dt}}_{translational} + \underbrace{\frac{d\overrightarrow{\omega}_b}{dt} \times \overrightarrow{r}}_{angular} + \underbrace{2\overrightarrow{\omega}_b \times \frac{d_r \overrightarrow{r}}{dt}}_{Coriolis} + \underbrace{\overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r})}_{centripetal} \tag{2.5}$$

όπου $\overrightarrow{u} \doteq \frac{d\overrightarrow{r}}{dt}$, $\overrightarrow{a} \doteq \frac{d^2\overrightarrow{r}}{dt^2}$ είναι η ταχύτητα και η επιτάχυνση ως προς το απόλυτο σύστημα (x, y, z). Το διάνυσμα θέσης $\overrightarrow{r}$ ορίζεται ως προς τη θέση περιστροφής του (x_r, y_r, z_r) στο απόλυτο σύστημα.

Στο δεξιό μέλος της έκφρασης 2.5 ξεχωρίζουν κατά σειρά η μεταφορική (translational), η γωνιακή (angular), η Coriolis και η κεντρομόλος (centripetal) επιτάχυνση.

Θεωρώντας τα μοναδιαία διανύσματα $\overrightarrow{\hat{r}}_{\parallel}$, $\overrightarrow{\hat{r}}_{\perp}$, με κατεύθυνση παράλληλη και κάθετη στο διάνυσμα της γωνιακής ταχύτητας περιστροφής $\overrightarrow{\omega}_b$ αντίστοιχα, το διάνυσμα θέσης $\overrightarrow{r}$ και η γωνιακή ταχύτητα $\overrightarrow{\omega}_b$ αναλύονται σε συνιστώσες ως εξής

$$\begin{aligned} \overrightarrow{r} &= r_{\parallel} \overrightarrow{\hat{r}}_{\parallel} + r_{\perp} \overrightarrow{\hat{r}}_{\perp} \\ \overrightarrow{\omega}_b &= \omega_b \overrightarrow{\hat{r}}_{\parallel} \end{aligned}$$

με ω_b το μέτρο της γωνιακής ταχύτητας περιστροφής του συστήματος (x_r, y_r, z_r) και $r_{\parallel}$, $r_{\perp}$ τα μέτρα των συνιστωσών του διανύσματος θέσης $\overrightarrow{r}$ στην κατεύθυνση των μοναδιαίων $\overrightarrow{\hat{r}}_{\parallel}$, $\overrightarrow{\hat{r}}_{\perp}$ αντίστοιχα.

Σχηματίζοντας, με βάση τις παραπάνω εκφράσεις, τη σχέση υπολογισμού της κεντρομόλου επιτάχυνσης, προκύπτει

$$\overrightarrow{\omega}_b \times \overrightarrow{r} = \omega_b r_{\perp} \overrightarrow{\hat{r}}_{\parallel} \times \overrightarrow{\hat{r}}_{\perp}$$

ή

$$\overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r}) = -\omega_b^2 r_{\perp} \overrightarrow{\hat{r}}_{\perp}$$

ή

$$\overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r}) = -\omega_b^2 \overrightarrow{R} \tag{2.6}$$

όπου ως $\overrightarrow{R}$ ορίζεται η κάθετη, στο διάνυσμα $\overrightarrow{\omega}_b$, συνιστώσα του διανύσματος θέσης $\overrightarrow{r}$.

Τέλος, η σχέση 2.6 παίρνει τη μορφή

$$\overrightarrow{\omega}_b \times (\overrightarrow{\omega}_b \times \overrightarrow{r}) = \nabla \left(-\frac{1}{2} \omega_b^2 R^2 \right) \tag{2.7}$$

Η επόμενη ενότητα παρουσιάζει τις εξισώσεις Navier–Stokes στο σύστημα (x_r, y_r, z_r). Στη συνέχεια του κεφαλαίου για ευκολία, το σύστημα (x_r, y_r, z_r) θα αναφέρεται απλά ως σχετικό σύστημα αξόνων.

2.2 Οι εξισώσεις Navier–Stokes ως προς σχετικό σύστημα αξόνων

Η ροή συνεκτικού ρευστού μοντελοποιείται μέσω των εξισώσεων Navier–Stokes οι οποίες εκφράζουν σε διαφορική μορφή τη διατήρηση της συνέχειας, της ορμής και της ενέργειας. Το ρευστό θεωρείται συμπιεστό. Οι εξισώσεις Navier–Stokes, σε αδιάστατη μορφή, ως προς σύστημα συντεταγμένων που μετακινείται (με $\overrightarrow{u}_b$) και περιστρέφεται (με $\overrightarrow{\omega}_b$) ως προς αδρανειακό σύστημα, διατυπώνονται στη διανυσματική έκφραση

$$\frac{\partial \overrightarrow{W}}{\partial t} + \frac{\partial \overrightarrow{F}_i^{\,inv}}{\partial x_i} - \frac{\partial \overrightarrow{F}_i^{\,vis}}{\partial x_i} = \overrightarrow{S} \tag{2.8}$$

όπου $\overrightarrow{W}$ το διάνυσμα των συντηρητικών μεταβλητών

$$\overrightarrow{W} = \begin{bmatrix} \varrho \\ \varrho \overrightarrow{w} \\ \varrho E \end{bmatrix} \tag{2.9}$$

με ρ την πυκνότητα του ρευστού, $\overrightarrow{w}$ το διάνυσμα της ταχύτητας του ρευστού στο σχετικό σύστημα συντεταγμένων (σχετική ταχύτητα) και E την ολική ενέργεια ανά μονάδα μάζας που δίνεται από τη σχέση

$$\varrho E = \frac{p}{\gamma - 1} + \frac{1}{2}\varrho u_i u_i - \frac{1}{2}\varrho \omega_b^2 R^2 \tag{2.10}$$

όπου p είναι η πίεση, γ ο λόγος των ειδικών θερμοχωρητικοτήτων του ρευστού, ω_b το μέτρο της ταχύτητας περιστροφής του σχετικού ως προς το απόλυτο σύστημα και R το μέτρο της κάθετης συνιστώσας $\overrightarrow{R}$ του διανύσματος θέσης $\overrightarrow{r}$ στην κατεύθυνση του διανύσματος $\overrightarrow{\omega}_b$. Η σχετική ταχύτητα $\overrightarrow{w}$ συνδέεται με την απόλυτη ταχύτητα $\overrightarrow{u}$ μέσω της ταχύτητας μεταφοράς $\overrightarrow{u}_b$ και της ταχύτητας περιστροφής $\overrightarrow{\omega}_b$ του σχετικού συστήματος, σχέση 2.4, όπου η ταχύτητα ως προς το σχετικό σύστημα ($\overrightarrow{u}_r$) συμβολίζεται πλέον ως $\overrightarrow{w}$.

Το διάνυσμα $\overrightarrow{F}$ χωρίζεται σε ατριβές και συνεκτικό τμήμα. Ο εκθέτης inv αντιπροσωπεύει το ατριβές τμήμα (inviscid) ενώ ο εκθέτης vis το συνεκτικό (viscous). Μηδενίζοντας τους συνεκτικούς όρους προκύπτουν οι εξισώσεις Euler. Το ατριβές και συνεκτικό τμήμα του διανύσματος $\overrightarrow{F}$ δίνεται παρακάτω

$$\overrightarrow{F}_i^{inv} = \begin{bmatrix} \rho w_i \\ \rho w_1 w_i + p\delta_{1i} \\ \rho w_2 w_i + p\delta_{2i} \\ \rho w_3 w_i + p\delta_{3i} \\ w_i\left(\rho E + p\right) \end{bmatrix}, \quad \overrightarrow{F}_i^{vis} = \frac{1}{Re_0}\begin{bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ w_j \tau_{ji} + q_i \end{bmatrix} \tag{2.11}$$

με Re_0 ο αριθμός Reynolds που προκύπτει με βάση τις τιμές των μεγεθών αδιαστατο-

ποίησης όπως περιγράφεται στην επόμενη ενότητα. Επίσης, τ_{ij} είναι ο τανυστής των τάσεων. Υποθέτοντας ότι τα υπό μελέτη ρευστά είναι ισότροπα νευτώνεια και ότι ισχύει η υπόθεση του Stokes για τη συνεκτικότητα, [117], ο τανυστής των τάσεων στο σχετικό σύστημα δίνεται από τη σχέση

$$\tau_{ij} = \frac{\mu_{eff}}{Re_0}\left[\left(\frac{\partial w_i}{\partial x_j} + \frac{\partial w_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial w_k}{\partial x_k}\right] \tag{2.12}$$

όπου $\mu_{eff} = \mu + \mu_t$ είναι το άθροισμα της μοριακής και της τυρβώδους συνεκτικότητας και δ_{ij} το δέλτα του Kronecker. Η ροή θερμότητας $\overrightarrow{q}$ μοντελοποιείται σύμφωνα με το νόμο του Fourier,

$$q_i = \frac{c_p \mu_{eff}}{Re_0 Pr}\frac{\partial T}{\partial x_i} \tag{2.13}$$

όπου Pr είναι ο αριθμός Prandtl και c_p η ειδική θερμοχωρητικότητα υπό σταθερή πίεση του αερίου.

Το δεξιό μέλος $\left(\overrightarrow{S}\right)$ της έκφρασης 2.8 είναι

$$\overrightarrow{S} = \begin{bmatrix} 0 \\ \varrho\frac{d\overrightarrow{u}_b}{dt} + \varrho\frac{d\overrightarrow{\omega}_b}{dt}\times\overrightarrow{r} + 2\varrho\overrightarrow{\omega}_b\times\overrightarrow{w} + \varrho\overrightarrow{\omega}_b\times(\overrightarrow{\omega}_b\times\overrightarrow{r}) \\ \varrho\left(\frac{d\overrightarrow{u}_b}{dt} + \frac{d\overrightarrow{\omega}_b}{dt}\times\overrightarrow{r}\right)\cdot\overrightarrow{w} \end{bmatrix} \tag{2.14}$$

Παρατηρείται ότι στις εξισώσεις της ορμής προστίθενται οι όροι της μεταφορικής, γωνιακής, Coriolis και κεντρομόλου επιτάχυνσης. Το έργο λόγω της μεταφορικής και γωνιακής επιτάχυνσης του σχετικού συστήματος προστίθεται στην εξίσωση ενέργειας. Το έργο της δύναμης Coriolis ανά μονάδα μάζας είναι μηδενικό. Η κεντρομόλος δύναμη ανά μονάδα μάζας εκφράζεται στη μορφή 2.7 και υπεισέρχεται στο ατριβές διάνυσμα $\overrightarrow{F}^{inv}$. Εξάλλου στη σχέση 2.10 υπάρχει ο όρος $-\frac{1}{2}\omega_b^2 R^2$.

Το κλείσιμο του συστήματος των εξισώσεων πραγματοποιείται με την υπόθεση ότι το εργαζόμενο μέσο ικανοποιεί την καταστατική εξίσωση των τέλειων αερίων

$$p = \rho RT \tag{2.15}$$

όπου R η σταθερά του αερίου. Η τελευταία συνδέεται με τις ειδικές θερχωρητικότητες υπό σταθερή πίεση και όγκο (c_p, c_v αντίστοιχα) μέσω της σχέσης

$$R = c_p - c_v \tag{2.16}$$

Τέλος, $\gamma = \frac{c_p}{c_v}$ είναι ο λόγος των ειδικών θερμοχωρητικοτήτων.

2.3 Αδιαστατοποίηση των εξισώσεων Navier–Stokes

Για την αδιαστατοποίηση των εξισώσεων Navier–Stokes επιλέγεται ένα μήκος αναφοράς (l_{ref}), ένα μέτρο ταχύτητας αναφοράς (u_{ref}) και μία τιμή πυκνότητας αναφοράς (ϱ_{ref}). Με δεδομένες τις τρεις αυτές τιμές υπολογίζονται τα μεγέθη αναφοράς των υπόλοιπων μεταβλητών της ροής,

$$\begin{aligned} p_{ref} &= \varrho_{ref} u_{ref}^2 \\ T_{ref} &= \frac{u_{ref}^2}{c_v} \\ E_{ref} &= \varrho_{ref} u_{ref}^2 \end{aligned} \tag{2.17}$$

Επιλέγεται επίσης

$$(R_g)_{ref} = c_v \tag{2.18}$$

Επομένως, η αδιάστατη σταθερά των αερίων και οι αδιάστατες ειδικές θερμοχωρητικότητες είναι

$$\breve{R}_g = \gamma - 1, \quad \breve{c}_p = \gamma, \quad \breve{c}_v = 1 \tag{2.19}$$

Με βάση την ταχύτητα και το μήκος αδιαστατοποίησης προκύπτει και ο χρόνος αναφοράς

$$t_{ref} = \frac{l_{ref}}{u_{ref}} \tag{2.20}$$

Ο αδιάστατος αριθμός Reynolds, των εκφράσεων της προηγούμενης ενότητας, προκύπτει από τα μεγέθη αναφοράς

$$Re_0 = \frac{\varrho_{ref} u_{ref} l_{ref}}{\mu_{ref}} \tag{2.21}$$

όπου μ_{ref} είναι η συνεκτικότητα αναφοράς.

Τα μεγέθη l_{ref}, u_{ref} και ϱ_{ref}, με βάση τα οποία ορίζονται τα μεγέθη αναφοράς και των υπόλοιπων μεταβλητών της ροής, ορίζονται κατάλληλα ώστε ο αριθμός Reynolds που προκύπτει από τη σχέση 2.21 να συμπίπτει με τον αριθμό Reynolds της ροής που μελετάται. Για παράδειγμα, κατά τον υπολογισμό της ροής γύρω από αεροτομή, εφόσον ο αριθμός Reynolds που χαρακτηρίζει τη ροή υπολογίζεται ως προς το μήκος της χορδής της, το μέτρο της ταχύτητας, την πυκνότητα και τη μοριακή συνεκτικότητα του ρευστού της επ΄ άπειρο ροής, συμφέρει τα μεγέθη αυτά να χρησιμοποιηθούν ως μεγέθη αναφοράς (l_{ref}, u_{ref}, ϱ_{ref}, μ_{ref}) κατά την αδιαστατοποίηση.

2.4 Το μοντέλο τύρβης των Spalart-Allmaras

Για τη μοντελοποίηση της τύρβης επιλέχθηκε το μοντέλο μιας εξίσωσης των Spalart-Allmaras, [53], στην παραλλαγή του για συμπιεστές ροές, [118, 119]. Το μοντέλο των Spalart–Allmaras προσομοιώνει τη μεταφορά, παραγωγή και διάχυση της ποσότητας $\tilde{\mu}$, η οποία σχετίζεται άμεσα με το συντελεστή τυρβώδους συνεκτικότητας, μ_t, θεωρώντας ότι ισχύει η υπόθεση του Boussinesq, δηλαδή ότι οι τάσεις Reynolds (στο σχετικό σύστημα) μοντελοποιούνται κατά τρόπο αντίστοιχο με αυτόν των μοριακών τάσεων,

$$\tau_{ij}^{t} = -\rho\overline{w_i' w_j'} = \frac{\mu_t}{Re_0}\left(\frac{\partial w_i}{\partial x_j} + \frac{\partial w_j}{\partial x_i} - \frac{2}{3}\frac{\partial w_r}{\partial x_r}\delta_{ij}\right)$$

Σε αδιάστατη μορφή, η εξίσωση του μοντέλου για συμπιεστές ροές δίνεται ως

$$\begin{aligned}\frac{\partial(\rho\tilde{\mu})}{\partial t} + \underbrace{\frac{\partial(\rho w_i\tilde{\mu})}{\partial x_i}}_{Conv} = &\underbrace{c_{b1}\left(1-f_{t2}\right)\tilde{S}\rho\tilde{\mu} - \frac{1}{Re_0}\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left(\frac{\tilde{\mu}}{d}\right)^2}_{Source} + \\ &+ \underbrace{\frac{1}{Re_0}\frac{1+c_{b2}}{\sigma}\frac{\partial}{\partial x_i}\left[\left(\mu+\tilde{\mu}\right)\frac{\partial\tilde{\mu}}{\partial x_i}\right] - \frac{1}{Re_0}\frac{c_{b2}}{\sigma}\left(\mu+\tilde{\mu}\right)\frac{\partial}{\partial x_i}\left(\frac{\partial\tilde{\mu}}{\partial x_i}\right)}_{Diff} + \\ &+ \underbrace{Re_0\rho^2 f_{t1}\Delta\vec{w}^2}_{Tr}\end{aligned} \tag{2.22}$$

όπου ο όρος '$Conv$' εκφράζει τη μεταφορά της ποσότητας '$\rho\tilde{\mu}$', ο όρος πηγής '$Source$' την παραγωγή και καταστροφή αυτής, ο όρος '$Diff$' τη μοριακή και τυρβώδη διάχυση ενώ ο όρος 'Tr' ενεργοποιείται προαιρετικά και μόνο όταν επιθυμείται η προσομοίωση μετάβασης από στρωτή σε τυρβώδη ροή, σε θέση (trip) η οποία καθορίζεται από το χρήστη ή υπολογίζεται από χωριστό μοντέλο μετάβασης. Η παραπάνω εξίσωση επιλύεται ως προς το $\tilde{\mu}$, από το οποίο προκύπτει ο συντελεστής τυρβώδους συνεκτικότητας μ_t. Οι ποσότητες που εμφανίζονται στην παραπάνω εξίσωση δίνονται από τις σχέσεις

$$\begin{aligned}\chi &= \frac{\tilde{\mu}}{\mu} \\ \tilde{\nu} &= \frac{\tilde{\mu}}{\rho} \\ f_{v1} &= \frac{\chi^3}{\chi^3 + c_{v1}^3} \\ \tilde{f_{v2}} &= \frac{1}{\left(1+\frac{\chi}{c_{v2}}\right)^3}\end{aligned} \tag{2.23}$$

$$
\begin{aligned}
\tilde{f_{v3}} &= \frac{1}{\chi}\left(1+\chi f_{v1}\right)\left(1-\tilde{f_{v2}}\right) \\
\tilde{S} &= \tilde{f_{v3}}S + \frac{1}{Re_0}\frac{\tilde{\nu}}{\kappa^2 d^2}\tilde{f_{v2}} \\
S &= |\overrightarrow{\gamma}| \\
f_w &= g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{\frac{1}{6}} \\
g &= r + c_{w2}\left(r^6 - r\right) \\
r &= \frac{1}{Re_0}\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \\
f_{t1} &= c_{t1} g_t exp\left[-c_{t2}\frac{|\overrightarrow{\gamma}^t|}{\Delta w^2}\left(d^2 + g_t^2 d_t^2\right)\right] \\
g_t &= min\left(0.1, \frac{\Delta w}{\gamma^t \Delta x}\right) \\
f_{t2} &= c_{t3}\exp\left(-c_{t4}\chi^2\right) \\
\Delta w &= |\overrightarrow{w}| - |\overrightarrow{w}^t| \\
d_t &= |\overrightarrow{x} - \overrightarrow{x}^t|
\end{aligned}
\tag{2.24}
$$

όπου $\overrightarrow{x}^t$, $\overrightarrow{w}^t$ και $\overrightarrow{\gamma}^t$ είναι το διάνυσμα θέσης, η ταχύτητα και η στροβιλότητα αντίστοιχα στο σημείο μετάβασης, το οποίο συμβολίζεται με τον εκθέτη t, ενώ Δx είναι το μήκος που χαρακτηρίζει το μέγεθος των στοιχείων του πλέγματος στο σημείο αυτό και d, d_t είναι οι αποστάσεις από τον τοίχο και το σημείο μετάβασης, αντίστοιχα. Τέλος, η τυρβώδης συνεκτικότητα, αδιαστατοποιημένη ως προς τη μοριακή συνεκτικότητα του ρευστού, υπολογίζεται ως

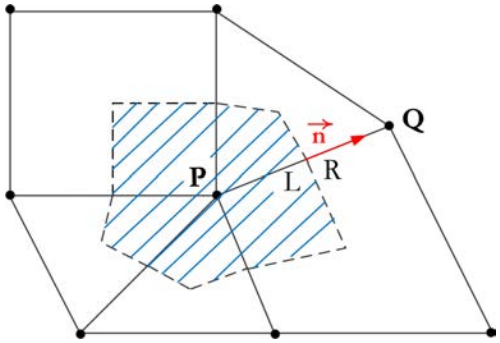
$$\mu_t = \chi f_{v1}$$

Οι σταθερές στην εξ. 2.22 και στις ποσότητες που εμπεριέχει λαμβάνουν τις εξής τιμές, [53]: $\sigma = \frac{2}{3}$, $\kappa = 0.41$, $c_{v1} = 0.71$, $c_{v2} = 5$, $c_{b1} = 0.1355$, $c_{b2} = 0.622$, $c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}$, $c_{w2} = 0.3$, $c_{w3} = 2$, $c_{t1} = 1$, $c_{t2} = 2$, $c_{t3} = 1.2$ και $c_{t4} = 0.5$.

2.5 Ολοκλήρωση σε κεντροκομβικούς όγκους ελέγχου με μεταβαλλόμενα όρια

Η ολοκλήρωση των εξισώσεων 2.8 γίνεται σε όγκους ελέγχου που σχηματίζονται γύρω από τους κόμβους του πλέγματος (κεντροκομβική διατύπωση της τεχνικής των πεπερασμένων όγκων). Ο όγκος ελέγχου (Ω_P) γύρω από έναν τυχαίο κόμβο P ενός υβριδικού υπολογιστικού πλέγματος, αποτελούμενο από τριγωνικά και τετραπλευρικά στοιχεία, φαίνεται στο σχήμα 2.1. Φαίνεται ότι ο όγκος αυτός σχηματίζεται ενώνοντας τα μέσα των ακμών που άγονται από τον κόμβο P με τα βαρύκεντρα των πλεγματικών

στοιχείων στα οποία ανήκουν οι ακμές αυτές. Ανάλογα, σχηματίζεται ένας όγκος ελέγχου σε 3Δ πλέγματα, με τα όρια του κάθε όγκου να ορίζονται από τα μέσα των ακμών, τα βαρύκεντρα των εδρών και τα βαρύκεντρα των πλεγματικών στοιχείων στα οποία ανήκει ο κόμβος P. Στο σχήμα 2.1, επίσης, ορίζεται το κάθετο διάνυσμα $\overrightarrow{n}$ στη διεπιφάνεια των όγκων ελέγχου που σχηματίζονται γύρω από τους γειτονικούς κόμβους P, Q. Το μέτρο του διανύσματος $\overrightarrow{n}$ είναι ίσο με το εμβαδόν της διεπιφάνειας. Ως L, R ορίζονται οι θέσεις εκατέρωθεν του μέσου της ακμής PQ.



Σχήμα 2.1: Όγκος ελέγχου σε υβριδικά πλέγματα.

Ολοκληρώνοντας τις εξισώσεις 2.8 στον όγκο ελέγχου που σχηματίζεται γύρω από τον κόμβο P του σχήματος 2.1 τη χρονική στιγμή t^{k+1} $\left(\Omega_P^{k+1} \doteq \Omega_P(t^{k+1})\right)$, προκύπτει

$$\iiint\limits_{\Omega_P^{k+1}} \left[\frac{\partial \overrightarrow{W}}{\partial t} + \frac{\partial \overrightarrow{F_i}^{inv}}{\partial x_i} - \frac{\partial \overrightarrow{F_i}^{vis}}{\partial x_i} \right] d\Omega = \iiint\limits_{\Omega_P^{k+1}} \overrightarrow{S}\, d\Omega$$

ή

$$\overrightarrow{\mathcal{R}}_P \doteq \underbrace{\iiint\limits_{\Omega_P^{k+1}} \overrightarrow{S}\, d\Omega}_{source} - \underbrace{\iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{W}}{\partial t} d\Omega}_{temporal} - \underbrace{\iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{F_i}^{inv}}{\partial x_i} d\Omega}_{inviscid} + \underbrace{\iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{F_i}^{vis}}{\partial x_i} d\Omega}_{viscous} = 0 \qquad (2.25)$$

όπου $\overrightarrow{\mathcal{R}}_P$ το υπόλοιπο των εξισώσεων της ροής στον κόμβο P. Υπενθυμίζεται ότι, στην παρούσα διατριβή, χρησιμοποιήθηκαν δυναμικά προσαρμοζόμενα πλέγματα. Συνεπώς, στη γενική περίπτωση, ο όγκος ελέγχου Ω_P μπορεί να μεταβάλλεται με το χρόνο.

Οι επόμενες παράγραφοι περιγράφουν, κατά σειρά, τον τρόπο υπολογισμού των ολοκληρωμάτων των ατριβών, συνεκτικών και χρονικών όρων της σχέσης 2.25 στον όγκο ελέγχου Ω_P^{k+1}.

2.5.1 Διακριτοποίηση των ατριβών όρων

Εφαρμόζοντας το θεώρημα των Green-Gauss προκύπτει

$$\mathcal{I}^{inv,k+1} \doteq \iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{F}_i}{\partial x_i}^{inv} d\Omega = \iint\limits_{\partial\Omega_P^{k+1}} \overrightarrow{F}_i^{inv} \cdot \hat{n}_i^{k+1} d(\partial\Omega)$$

όπου $\partial\Omega_P^{k+1}$ είναι το όριο του όγκου ολοκλήρωσης και $d(\partial\Omega)$ μία στοιχειώδης επιφάνεια του ορίου αυτού. $\overrightarrow{\hat{n}}^{k+1}$ είναι το μοναδιαίο κάθετο διάνυσμα στο όριο του όγκου ελέγχου (σχήμα 2.1) τη χρονική στιγμή t^{k+1}. Για ευκολία, στη συνέχεια της παραγράφου θεωρείται ότι οι μεταβλητές της ροής και όλες οι γεωμετρικές ποσότητες αναφέρονται στη χρονική στιγμή t^{k+1} και ο εκθέτης $k+1$ παραλείπεται.

Το επιφανειακό ολοκλήρωμα της παραπάνω έκφρασης προσεγγίζεται ως

$$\mathcal{I}^{inv} \simeq \sum_{Q \in nei(P)} \overrightarrow{\Phi}_{PQ}^{inv} \tag{2.26}$$

όπου Q είναι οι γειτονικοί στον P κόμβοι, το σύνολο των οποίων είναι $nei(P)$. $\overrightarrow{\Phi}_{PQ}^{inv}$ είναι το αριθμητικό διάνυσμα ατριβούς ροής που διαπερνά τη διεπιφάνεια των όγκων ελέγχου των γειτονικών κόμβων P και Q. Ο υπολογισμός των $\overrightarrow{\Phi}_{PQ}^{inv}$ γίνεται σύμφωνα με το σχήμα του Roe, [120],

$$\overrightarrow{\Phi}_{PQ}^{inv} = \frac{1}{2}\left[A_P \overrightarrow{W}_{PQ}^L + A_Q \overrightarrow{W}_{PQ}^R\right] - \frac{1}{2}\left|\tilde{A}_{PQ}\right|\left(\overrightarrow{W}_{PQ}^R - \overrightarrow{W}_{PQ}^L\right) \tag{2.27}$$

ή το σχήμα διάσπασης ροών (Flux Vector Splitting, [121]),

$$\overrightarrow{\Phi}_{PQ}^{inv} = A_{PQ}^- \overrightarrow{W}_{PQ}^R + A_{PQ}^+ \overrightarrow{W}_{PQ}^L \tag{2.28}$$

Αν $\overrightarrow{U}$ είναι το διάνυσμα των μη-συντηρητικών μεταβλητών

$$\overrightarrow{U} = \begin{bmatrix} \varrho \\ \overrightarrow{w} \\ p \end{bmatrix}, \tag{2.29}$$

οι οποίες χρησιμοποιούνται για τον υπολογισμό των $\overrightarrow{W}_{PQ}^L = \overrightarrow{W}_{PQ}^L\left(\overrightarrow{U}_{PQ}\right)$, $\overrightarrow{W}_{PQ}^R = \overrightarrow{W}_{PQ}^R\left(\overrightarrow{U}_{PQ}\right)$. Οι ροϊκές ποσότητες $\overrightarrow{U}_{PQ}^L$, $\overrightarrow{U}_{PQ}^R$ προκύπτουν από την προεκβολή των ροϊκών μεταβλητών στους κόμβους P, Q ($\overrightarrow{U}^P$, $\overrightarrow{U}^Q$ αντίστοιχα) στο μέσο της ακμής που ενώνει τους δύο κόμβους. Είναι, δηλαδή,

$$\begin{aligned}\overrightarrow{U}^{L}_{PQ} &= \overrightarrow{U}_P + \frac{1}{2}\,(\overrightarrow{PQ})\cdot\nabla\overrightarrow{U}_P \\ \overrightarrow{U}^{R}_{PQ} &= \overrightarrow{U}_Q - \frac{1}{2}\,(\overrightarrow{PQ})\cdot\nabla\overrightarrow{U}_Q\end{aligned} \tag{2.30}$$

Τα μητρώα A^+, A^-, $\tilde{A}$ αποτελούν τα ιακωβιανά μητρώα υπολογισμένα σύμφωνα με τις θετικές ή αρνητικές ιδιοτιμές, ή με βάση τις κατά Roe σταθμισμένες μέσες τιμές των ροϊκών μεταβλητών στους κόμβους P, Q, αντίστοιχα.

2.5.2 Διακριτοποίηση συνεκτικών όρων

Εφαρμόζοντας και πάλι το θεώρημα των Green-Gauss προκύπτει

$$\begin{aligned}\mathcal{I}^{vis,k+1} &\doteq \iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{F}^{\,vis}_i}{\partial x_i}\,d\Omega = \iint\limits_{\partial\Omega_P^{k+1}} \overrightarrow{F}^{vis}_i\cdot\hat{n}_i^{k+1}d(\partial\Omega) \Rightarrow \\ \Rightarrow\mathcal{I}^{vis,k+1} &\simeq \sum_{Q\in nei(P)} \overrightarrow{\Phi}^{vis,k+1}_{PQ}\end{aligned} \tag{2.31}$$

όπου $\overrightarrow{\Phi}^{vis}_{PQ}$ είναι τα αριθμητικά διανύσματα συνεκτικής ροής

$$\overrightarrow{\Phi}^{vis}_{PQ} = \begin{bmatrix} 0 \\ \tau^{PQ}_{1i}n_i \\ \tau^{PQ}_{2i}n_i \\ \tau^{PQ}_{3i}n_i \\ w_j\tau^{PQ}_{ji}n_i + q_in_i \end{bmatrix} \tag{2.32}$$

τ^{PQ}_{ij} είναι ο τανυστής των τάσεων στο μέσο της ακμής PQ και υπολογίζονται με βάση τις χωρικές παραγώγους των μη-συντηρητικών μεταβλητών της ροής $\left(\overrightarrow{U}\right)$ σύμφωνα με τη σχέση 2.12. Ο υπολογισμός των $\frac{\partial\overrightarrow{U}_{PQ}}{\partial x_i}$ γίνεται, [122], σύμφωνα με τη σχέση

$$\nabla\overrightarrow{U}_{PQ} = \frac{1}{2}[\nabla\overrightarrow{U}_P + \nabla\overrightarrow{U}_Q] - \left[\frac{1}{2}[\nabla\overrightarrow{U}_P + \nabla\overrightarrow{U}_Q]\,\overrightarrow{n} - \frac{\overrightarrow{U}_Q - \overrightarrow{U}_P}{(PQ)}\right]\overrightarrow{n}$$

Στις παραπάνω εκφράσεις οι μεταβλητές της ροής και όλες οι γεωμετρικές ποσότητες αναφέρονται στη χρονική στιγμή t^{k+1} και, για ευκολία, ο εκθέτης $k+1$ παραλείπεται. Η χωρική παράγωγος των μεταβλητών της ροής στους κόμβους του πλέγματος υπολογίζονται με βάση τις κλίσεις των μεταβλητών της ροής στα πλεγματικά στοιχεία

$$\nabla\overrightarrow{U}_T = \sum_{l=1}^{N_l} \overrightarrow{U}_l \frac{\partial L_l}{\partial x_l} \tag{2.33}$$

όπου το l αντιστοιχεί στις κορυφές του πλεγματικού στοιχείου και L_l είναι οι αντίστοιχες συναρτήσεις βάσης (shape functions). N_l είναι το πλήθος των κορυφών.

2.5.3 Διακριτοποίηση του χρονικού όρου

Χρησιμοποιώντας το θεώρημα του Leibniz, [123], η ολοκλήρωση της χρονικής παραγώγου των μεταβλητών της ροής υπολογίζεται ως

$$\iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{W}}{\partial t} d\Omega = \frac{\partial}{\partial t} \iiint\limits_{\Omega_P^{k+1}} \overrightarrow{W} dV - \iint\limits_{\partial\Omega_P^{k+1}} \overrightarrow{u}_{grid}^{k+1} \cdot \overrightarrow{\hat{n}}^{k+1} \overrightarrow{W} \, d(\partial\Omega) \tag{2.34}$$

όπου $\overrightarrow{u}_{grid}^{k+1}$ είναι η ταχύτητα παραμόρφωσης του ορίου του όγκου ολοκλήρωσης τη χρονική στιγμή t^{k+1}. Η τελευταία υπολογίζεται με τον περιορισμό (geometric conservation law), [123],

$$\iint\limits_{\partial\Omega_P} \overrightarrow{u}_{grid} \cdot \overrightarrow{\hat{n}} \, d(\partial\Omega) = \frac{\partial \Omega_p}{\partial t} \tag{2.35}$$

όπου $\frac{\partial \Omega_P}{\partial t}$ είναι ο ρυθμός μεταβολής του όγκου ελέγχου. Η σχέση 2.35 συνδέει το ρυθμό μεταβολής του Ω_P με την ταχύτητα παραμόρφωσης του ορίου του ίδιου όγκου ελέγχου.

Χρησιμοποιώντας ένα αριθμητικό σχήμα πίσω διαφόρισης στο χρόνο δεύτερης τάξης ακρίβειας, η σχέση 2.34 υπολογίζεται ως

$$\iiint\limits_{\Omega_P^{k+1}} \frac{\partial \overrightarrow{W}}{\partial t} d\Omega \simeq \frac{1}{2\Delta t} \left(3\overrightarrow{W}_P^{k+1} \Omega_P^{k+1} - 4\overrightarrow{W}_P^{k} \Omega_P^{k} + \overrightarrow{W}_P^{k-1} \Omega_P^{k-1} \right) - \sum_{Q \in nei(P)} \overrightarrow{\Phi}_{PQ}^{grid,k+1} \tag{2.36}$$

όπου $\overrightarrow{W}_P^{k+1}$ είναι οι τιμές των συντηρητικών μεταβλητών της ροής στον κόμβο P και Ω_P^{k+1} είναι ο όγκος ελέγχου γύρω από τον ίδιο κόμβο τη χρονική στιγμή t^{k+1} αντίστοιχα. Όμοια, οι ποσότητες $\overrightarrow{W}_P^{k}$, Ω_P^{k} αναφέρονται στη χρονική στιγμή t^k και οι ποσότητες $\overrightarrow{W}_P^{k-1}$, Ω_P^{k-1} στη χρονική στιγμή t^{k-1}. $\overrightarrow{\Phi}_{PQ}^{grid}$ είναι το αριθμητικό διάνυσμα της ροής λόγω της παραμόρφωσης του πλέγματος και υπολογίζεται ως

$$\overrightarrow{\Phi}_{PQ}^{grid,k+1} = \underbrace{\left[\int\limits_{\Omega_P \cap \Omega_Q} \overrightarrow{u}_{grid}^{k+1} \cdot \overrightarrow{\hat{n}}^{k+1} d(\partial\Omega) \right]}_{I_{grid,PQ}^{k+1}} \overrightarrow{W}_{PQ}^{k+1} = I_{grid,PQ}^{k+1} \overrightarrow{W}_{PQ}^{k+1} \tag{2.37}$$

όπου $\overrightarrow{W}_{PQ}$ είναι το διάνυσμα των συντηρητικών μεταβλητών της ροής στο μέσο της ακμής PQ και προσεγγίζεται ως το ημιάθροισμα των μεταβλητών της ροής στους κόμβους P, Q ($\overrightarrow{W}_P$, $\overrightarrow{W}_Q$ αντίστοιχα). Επιτυγχάνεται αύξηση της ακρίβειας του σχήματος χρησιμοποιώντας τις χωρικές παραγώγους των μεταβλητών της ροής στους κόμβους P, Q για την προεκβολή των $\overrightarrow{W}_P$, $\overrightarrow{W}_Q$ στο μέσο της ακμής PQ. Στη συνέχεια, οι μεταβλητές $\overrightarrow{W}_{PQ}$ υπολογίζονται ως το ημιάθροισμα των προεκβεβλημένων ποσοτήτων.

Θεωρώντας ότι τα στοιχεία του πλέγματος είναι τύπου P^1, η τιμή του ολοκληρώματος $I_{grid,PQ}$ ανά διεπιφάνεια ($\Omega_P \cap \Omega_Q$) δύο γειτονικών όγκων ελέγχου Ω_P, Ω_Q υπολογίζεται αναλυτικά και εκφράζεται ως προς τις τιμές της $\overrightarrow{u}_{grid}$ στους κόμβους του πλέγματος, οι οποίες υπολογίζονται αριθμητικά χρησιμοποιώντας ένα σχήμα πίσω διαφόρισης δεύτερης τάξης ακρίβειας. Οι όγκοι ελέγχου Ω_P^{k+1} στη σχέση 2.36 υπολογίζονται γεωμετρικά και όχι από την ολοκλήρωση της 2.35. Εξάλλου, με βάση τον τρόπο υπολογισμού των $I_{grid,PQ}$, ο περιορισμός 2.35 ικανοποιείται.

2.5.4 Επιβολή οριακών συνθηκών

Στερεά όρια

Ο τύπος της οριακής συνθήκης που επιβάλλεται στα στερεά όρια εξαρτάται από τη φυσική της ροής. Στην περίπτωση ατριβούς ροής, επιβάλλεται η συνθήκη μη-εισχώρησης,

$$(\overrightarrow{w} - \overrightarrow{u}_{grid}) \cdot \overrightarrow{n} = 0$$

Συνεπώς,

$$\overrightarrow{\Phi}_{wall}^{inv} - \overrightarrow{\Phi}_{wall}^{grid} = \begin{bmatrix} 0 \\ p\,\overrightarrow{n} \\ (\overrightarrow{u}_{grid} \cdot \overrightarrow{n})\, p \end{bmatrix} \tag{2.38}$$

Αντίθετα, στην περίπτωση συνεκτικής ροής, η ταχύτητα στον τοίχο είναι ίση με την ταχύτητα μετακίνησης/παραμόρφωσης του στερεού ορίου (συνθήκη μη-ολίσθησης).

$$\overrightarrow{w}_{wall} = \overrightarrow{u}_{grid}^{wall}$$

Είσοδος και έξοδος του πεδίου ροής

Για την είσοδο και την έξοδο (δείκτες I/O) του πεδίου ροής, με βάση το σχήμα διάσπασης ροών (Flux Vector Splitting), λαμβάνεται

$$\overrightarrow{\Phi}_{I/O}^{inv} = A_P^+ \overrightarrow{W}_P + A_P^- \overrightarrow{W}_{inf} \tag{2.39}$$

όπου οι δείκτες P και inv δηλώνουν έναν οριακό κόμβο και μία υποθετική κατάσταση

στο εξωτερικό του χωρίου αντίστοιχα ενώ A είναι το ιακωβιανό μητρώο των ατριβών ροών κατά την κάθετη στην είσοδο/έξοδο κατεύθυνση.

Περιοδικά όρια

Η εξασφάλιση της ισότητας των μεγεθών της ροής σε κάθε ζεύγος περιοδικών κόμβων (έστω P, P') επιτυγχάνεται συμπληρώνοντας τα διανύσματα ροής ενός κόμβου με αυτά του αντίστοιχου περιοδικού του, κατά τους σχετικούς ισολογισμούς.

2.6 Επίλυση των διακριτοποιημένων εξισώσεων

Αντικαθιστώντας τις εκφράσεις 2.26, 2.31, 2.36 στη σχέση 2.25 προκύπτει

$$\overrightarrow{\mathcal{R}}_P^{k+1} \doteq \mathcal{I}_{source}^{k+1} - \sum_{Q \in nei(P)} \overrightarrow{\Phi}_{PQ}^{k+1} - \frac{1}{2\Delta t}\left(3\overrightarrow{W}_P^{k+1}\Omega_P^{k+1} - 4\overrightarrow{W}_P^{k}\Omega_P^{k} + \overrightarrow{W}_P^{k-1}\Omega_P^{k-1}\right) = 0 \tag{2.40}$$

όπου $\mathcal{I}_{source}^{k+1}$ η ολοκλήρωση των όρων πηγής ($\overrightarrow{S}$) στον όγκο ελέγχου Ω_P^{k+1} και $\overrightarrow{\Phi}_{PQ}^{k+1} = \overrightarrow{\Phi}_{PQ}^{inv,k+1} - \overrightarrow{\Phi}_{PQ}^{vis,k+1} - \overrightarrow{\Phi}_{PQ}^{grid,k+1}$ είναι το αριθμητικό διάνυσμα της ροής που διαπερνά τη διεπιφάνεια των Ω_P^{k+1}, Ω_Q^{k+1}.

Το σύστημα των διακριτοποιημένων εξισώσεων 2.40 γραμμικοποιείται και επιλύεται επαναληπτικά για διαδοχικές χρονικές στιγμές σύμφωνα με την τεχνική διπλού χρονικού βήματος (dual time stepping) ως προς τη διόρθωση $\left(\Delta\overrightarrow{W}\right)$ των συντηρητικών μεταβλητών της ροής. Συγκεκριμένα, η επίλυση για κάθε πραγματική χρονική στιγμή θεωρείται ως ένα χρονικά μόνιμο πρόβλημα (με εμβόλιμους όρους πηγής για τη χρονική μεταβολή). Η επίλυση των γραμμικοποιημένων εξισώσεων, ανά πραγματικό χρονικό βήμα, γίνεται με τη μέθοδο Jacobi και περιγράφεται στην παράγραφο 4.4.1, όπου παρουσιάζεται και ο τρόπος υλοποίησης της μεθόδου σε GPUs. Η ενότητα 8.5 περιγράφει την ταυτόχρονη επίλυση των εξισώσεων της ροής με τις ελαστικές εξισώσεις παραμορφώσιμου αεροδυναμικού σώματος.

Κεφάλαιο 3

Η αρχιτεκτονική παράλληλης επεξεργασίας CUDA

Το κεφάλαιο αυτό παρουσιάζει την αρχιτεκτονική CUDA. Η αρχιτεκτονική αυτή αναπτύσσεται από την NVIDIA και αποτελεί την αρχή κατασκευής των προγραμματιζόμενων καρτών γραφικών της εταιρίας. Οι G80, GT200 και Fermi αποτελούν μερικές ενδεικτικές εκδόσεις της CUDA τη χρονική περίοδο 2006-2012.

Κάθε προγραμματιζόμενη GPU της NVIDIA φέρει έναν αριθμό που χαρακτηρίζει την υπολογιστική δυνατότητα της κάρτας. Για παράδειγμα, οι GPUs υπολογιστικής δυνατότητας 1.0, ή 1.1 δεν υποστηρίζουν την αριθμητική διπλής ακρίβειας. Οι GeForce GTX 280 και 285 που χρησιμοποιήθηκαν στη διατριβή, έχουν υπολογιστική δυνατότητα 1.3 και είναι δομημένες σύμφωνα με την αρχιτεκτονική GT200. Αντίθετα, οι GPUs υπολογιστικής δυνατότητας 2.x, όπως είναι οι Telsa M2050 (υπολογιστικής δυνατότητας 2.0), στηρίζονται στην τελευταία αναβάθμιση της αρχιτεκτονικής CUDA, με το όνομα Fermi. Στη συνέχεια του κειμένου, για ευκολία και συντόμευση, μπορεί να παραλείπεται ο όρος 'υπολογιστική δυνατότητα', δηλαδή ο όρος 'κάρτα γραφικών 2.x' σημαίνει 'κάρτα γραφικών υπολογιστικής δυνατότητας 2.x'.

Στο κεφάλαιο αυτό περιγράφεται η δομή των αρχιτεκτονικών GT200 και Fermi και παρουσιάζεται το περιβάλλον προγραμματισμού της CUDA. Το περιβάλλον αυτό χρησιμοποιείται για τον προγραμματισμό κώδικα που εκτελείται σε GPUs. Κατά την παρουσίαση του περιβάλλοντος προγραμματισμού της CUDA, τονίζεται η άμεση επίδραση του τρόπου προσπέλασης των μνημών μίας GPU, στην απόδοση ενός GPU-κώδικα. Για περισσότερες λεπτομέρειες σχετικά με το περιβάλλον προγραμματισμού της CUDA, ο αναγνώστης μπορεί να ανατρέξει στο αντίστοιχο εγχειρίδιο [2], ή στα βιβλία [124, 125].

Το κεφάλαιο αυτό αναφέρεται αποκλειστικά στη διαχείριση μίας GPU. Η διαχείριση πολλών GPUs του ίδιου ή διασυνδεδεμένων υπολογιστικών κόμβων αποτελεί αντικείμενο του κεφαλαίου 5.

3.1 Το Thread ως η βασική μονάδα επεξεργασίας

Το thread ορίζεται ως η βασική μονάδα επεξεργασίας. Κάθε CPU, ή GPU μπορεί να χειρίζεται τόσα threads όσα επιτρέπει η υπολογιστική τους δυνατότητα και η αρχιτεκτονική με την οποία αυτές είναι δομημένες. Δηλαδή, οι σημερινές CPUs και GPUs έχουν τη δυνατότητα δημιουργίας threads, στα οποία αναθέτουν την εκτέλεση ενός προγράμματος (kernel). Για ευκολία, στη συνέχεια του κειμένου της διατριβής τα threads που χειρίζεται μία CPU, θα αναφέρονται ως CPU threads. Αντίθετα, ο όρος threads θα αντιστοιχεί στα threads που χειρίζεται μία GPU, χωρίς να χρειάζεται ιδιαίτερος προσδιορισμός.

3.2 Οργάνωση των threads σε μία GPU

Οι σημερινές GPUs μπορούν να χειρίζονται ταυτόχρονα χιλιάδες threads. Για τη διαχείριση αυτών εφαρμόζουν τη λογική SIMT (Single Instruction Multiple Thread). Έτσι, το ίδιο τμήμα κώδικα (kernel) εκτελείται από μία ομάδα από threads, με κάθε thread να διαχειρίζεται διαφορετικό όγκο δεδομένων.

Τα threads οργανώνονται σε blocks και εκτελούνται στους διαθέσιμους πολυεπεξεργαστές της GPU. Κάθε πολυεπεξεργαστής μπορεί να αναλάβει την εκτέλεση έως και 8 blocks. Στην περίπτωση που δεν επαρκούν οι διαθέσιμοι πολυεπεξεργαστές της GPU, κάποια blocks μένουν ανενεργά, αναμένοντας την ολοκλήρωση της εκτέλεσης ορισμένων εκ των ενεργών. Το γεγονός αυτό αποτελεί ένα μεγάλο πλεονέκτημα της CUDA, καθώς ο ίδιος κώδικας εκτελείται σε κάρτες γραφικών διαφορετικού αριθμού πολυεπεξεργαστών, χωρίς να απαιτείται η τροποποίηση ή προσαρμογή αυτού. Πρακτικά, όσους περισσότερους πολυεπεξεργαστές έχει μία GPU, τόσα περισσότερα blocks από threads εκτελούνται ταυτόχρονα. Ένα block μπορεί να αποτελείται το πολύ από 512 ή 1024 threads, για GPUs 1.x ή 2.x αντίστοιχα.

Τα blocks που εκτελούν το ίδιο kernel, σχηματίζουν ένα πλέγμα από blocks. Στο υπόλοιπο του κειμένου της διατριβής για να αποφευχθεί η σύγχυση ανάμεσα στο πλέγμα των blocks και στο πλέγμα που χρησιμοποιείται για τη χωρική διακριτοποίηση του υπολογιστικού χωρίου, το πρώτο θα αποκαλείται 'grid' ενώ το δεύτερο 'πλέγμα'.

Η εκτέλεση των threads, όπως αναφέρθηκε προηγουμένως, γίνεται στους διαθέσιμους πολυεπεξεργαστές της GPU. Για την ακρίβεια, κάθε πολυεπεξεργαστής ομαδοποιεί τα threads σε ομάδες των 32, που ονομάζονται warps. Ο αριθμός των warps που μπορεί να χειρίζεται ταυτόχρονα ο πολυεπεξεργαστής εξαρτάται από την υπολογιστική δυνατότητα της GPU. Half-warp ορίζεται το πρώτο, ή το δεύτερο μισό ενός warp. Τονίζεται ότι τα threads του ίδιου warp εκτελούν την ίδια σειρά εντολών ταυτόχρονα. Επομένως στην περίπτωση που η δομή ενός kernel είναι αντίστοιχη του ψευδοκώδικα 1, ο προγραμματιστής πρέπει να εγγυηθεί ότι όλα τα threads του ίδιου warp θα ακολουθήσουν την ίδια σειρά εντολών (Α-Β-Δ), ή (Α-Γ-Δ). Στην περίπτωση που τα threads ενός warp διακρίνονται σε ομάδες Ο_B, Ο_Γ ανάλογα με ποια λογική έκφραση (ΛΕ_B ή ΛΕ_Γ) είναι αληθής, τότε όταν τα threads της Ο_B εκτελούν τη σειρά εντολών Β, εκείνα της Ο_Γ παραμένουν ανενεργά. Αντίστοιχα, όταν τα threads της Ο_Γ εκτελούν

τη 'δική τους' σειρά εντολών (Γ), τα threads της O_B παραμένουν ανενεργά. Γίνεται αντιληπτό ότι όσο περισσότερες ομάδες από threads σχηματίζονται στο ίδιο warp, τόσο περισσότερο ζημιώνεται η απόδοση του kernel. Threads διαφορετικών warps επιτρέπεται να ακολουθούν διαφορετική πορεία εκτέλεσης.

Ψευδοκώδικας 1

```
1: Σειρά εντολών Α
2: if Λογική έκφραση ΛΕ_B then
3:     Σειρά εντολών Β
4: else if Λογική έκφραση ΛΕ_Γ then
5:     Σειρά εντολών Γ
6: end if
7: Σειρά εντολών Δ
```

Κάθε thread χαρακτηρίζεται από έναν τοπικό αύξοντα αριθμό στο block στο οποίο ανήκει και, αντίστοιχα, κάθε block από έναν αύξοντα αριθμό στο grid. Έτσι, όπως θα δειχθεί στο παράδειγμα της παραγράφου 3.7, κάθε thread μπορεί να συσχετιστεί με συγκεκριμένες θέσεις μνήμης με βάση τον αύξοντα αριθμό του στο block και τον αύξοντα αριθμό του block (στο οποίο ανήκει) στο grid. Με τον τρόπο αυτό, κάθε thread επεξεργάζεται διαφορετικά δεδομένα.

Ο προγραμματιστής επιλέγει ανάμεσα σε 1Δ, 2Δ, ή 3Δ διάταξη των threads στα blocks και των blocks στο grid. Σημειώνεται ότι μόνο οι κάρτες γραφικών αρχιτεκτονικής Fermi υποστηρίζουν την 3Δ διάταξη των blocks στο grid. Η επιλογή της διαμέρισης του grid σε blocks και των blocks σε threads, εξαρτάται άμεσα με το προς επίλυση πρόβλημα. Σε επόμενη παράγραφο παρουσιάζεται ψευδοκώδικας υπολογισμού του γινομένου δύο πινάκων $\left([A]_{n_i \times n_k} \cdot [B]_{n_k \times n_j} = [C]_{n_i \times n_j}\right)$ σε μία GPU. Στο παράδειγμα αυτό, κάθε thread της GPU υπολογίζει ένα στοιχείο του πίνακα C. Γίνεται αντιληπτό ότι η συσχέτιση των στοιχειών του 2Δ πίνακα C με τα threads της GPU γίνεται πιο εύκολη αν ο προγραμματιστής επιλέξει το grid να έχει 2Δ δομή από blocks και τα blocks 2Δ δομή από threads. Ο εν λόγω ψευδοκώδικας παρουσιάζεται και αναλύεται στην παράγραφο 3.7.

Η διάταξη και η οργάνωση των threads σε blocks και των blocks σε grid φαίνεται στο σχήμα 3.1. Στο σχήμα αυτό φαίνονται επιπλέον οι μνήμες της GPU στις οποίες έχει πρόσβαση κάθε thread. Οι μνήμες αυτές παρουσιάζονται επιγραμματικά στην παράγραφο 3.3 και αναλυτικότερα στην παράγραφο 3.8.

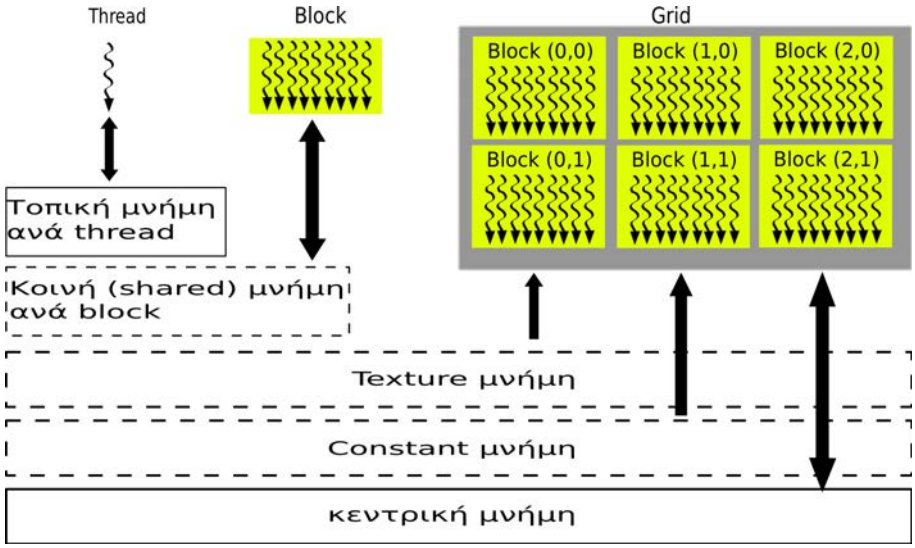
3.3 Μνήμες της GPU

Μία GPU έχει ενσωματωμένες τις ακόλουθες μνήμες:

1. Τοπική (Local) μνήμη ανά thread.
2. Κοινή μνήμη ανάμεσα στα threads του ίδιου block (shared μνήμη).
3. Τις texture και constant μνήμες που είναι προσβάσιμες από όλα τα threads.
4. Την κεντρική μνήμη της GPU που είναι προσβάσιμη από όλα τα threads.

Έτσι, κάθε thread έχει τη δική του τοπική μνήμη. Τα threads του ίδιου block έχουν τη δυνατότητα συγχρονισμού και μοιράζονται δεδομένα μέσω μίας κοινής, γρήγορα προσπελάσιμης μνήμης (shared μνήμη). Όλα τα threads του grid έχουν πρόσβαση στην κεντρική μνήμη της GPU και στις texture και constant, οι οποίες επιτρέπουν μόνο την ανάγνωση δεδομένων από αυτές.

Η οργάνωση των threads, όπως παρουσιάστηκε στην παράγραφο 3.2, φαίνεται στο σχήμα 3.1. Στο ίδιο σχήμα φαίνονται και οι διαθέσιμες μνήμες μίας GPU.



Σχήμα 3.1: Οργάνωση των threads σε blocks, και των blocks σε grid. Εδώ, ο προγραμματιστής έχει επιλέξει 1Δ κατανομή των threads στα blocks, και 2Δ κατανομή των blocks στο grid. Κάθε thread έχει τη δική του τοπική μνήμη. Τα threads του ίδιου block μοιράζονται δεδομένα μέσω της κοινής γρήγορα προσπελάσιμης shared μνήμης. Όλα τα threads του grid έχουν πρόσβαση στην κεντρική μνήμη της GPU, την texture και την constant. Οι τελευταίες επιτρέπουν μόνο την ανάγνωση δεδομένων από αυτές. Όλες οι μνήμες στις οποίες έχουν πρόσβαση τα threads μίας GPU αναλύονται εκτενέστερα στην ενότητα 3.8.

Για την καλύτερη κατανόηση του τρόπου λειτουργίας μίας GPU ακολουθεί η περιγραφή των αρχιτεκτονικών GT200 και Fermi. Στη συνέχεια, παρουσιάζεται ένα απλό

παράδειγμα παράλληλου προγραμματισμού σε μία GPU στο περιβάλλον της CUDA. Στο παράδειγμα αυτό χρησιμοποιείται σκόπιμα μόνο η κεντρική μνήμη της GPU. Ακολουθεί η ενότητα 3.8, όπου γίνεται η αναλυτικότερη παρουσίαση των μνημών μίας GPU. Περιγράφεται, επίσης, ο αποδοτικότερος τρόπος διαχείρισης αυτών. Στη συνέχεια, επεκτείνεται ο κώδικας του παραδείγματος με σκοπό την αύξηση της παράλληλης απόδοσής του.

3.4 Η αρχιτεκτονική καρτών γραφικών GT200

Οι κάρτες γραφικών που βασίζονται στην αρχιτεκτονική GT200 (σχήμα 3.2, [126]) αποτελούνται από 10 TPCs (Texture Processor Clusters).

Κάθε TPC περιέχει:

- 3 πολυεπεξεργαστές (Streaming Multiproccessors).
- 1 ενδιάμεση μνήμη (texture cache).

Η texture cache μοιράζεται μεταξύ των πολυεπεξεργαστών και χρησιμοποιείται για την επιτάχυνση της ανάγνωσης δεδομένων από την texture μνήμη. Η texture μνήμη βρίσκεται στον ίδιο χώρο με την κεντρική της GPU. Το μέγεθος της texture cache εξαρτάται από το μοντέλο της κάρτας γραφικών και κυμαίνεται ανάμεσα σε 6 και 8 KByte ανά πολυεπεξεργαστή.

Κάθε πολυεπεξεργαστής αποτελείται από:

- 8 μονάδες ικανές να πραγματοποιούν αριθμητικές πράξεις μεταξύ ακεραίων ή πραγματικών αριθμών απλής ακρίβειας (Streaming Processors).
- 1 μονάδα εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας.
- 2 ειδικές μονάδες εκτέλεσης μαθηματικών συναρτήσεων (Special Function Units, SFU).
- 1 warp scheduler.
- 1 γρήγορα προσπελάσιμη μνήμη (shared μνήμη).
- 1 ενδιάμεση μνήμη (constant cache).
- 16384 32-bit καταχωρητές (registers).

Δηλαδή, συνολικά, κάθε πολυεπεξεργαστής περιέχει $8+1+2=11$ μονάδες για την εκτέλεση αριθμητικών πράξεων κινητής υποδιαστολής ή γνωστών μαθηματικών συναρτήσεων (τριγωνομετρικοί αριθμοί, τετραγωνική ρίζα, κ.α.). Σημειώνεται ότι οι

SFUs πέρα από γνωστές μαθηματικές συναρτήσεις μπορούν να εκτελούν και αριθμητικές πράξεις μεταξύ αριθμών απλής ακρίβειας. Κάθε πολυεπεξεργαστής περιέχει $8\ SMs + 2\ SFUs = 10$ μονάδες για την εκτέλεση αριθμητικών πράξεων μεταξύ αριθμών απλής ακρίβειας και μόλις μία για διπλής ακρίβειας. Αυτό σημαίνει ότι η ταχύτητα εκτέλεσης διπλής ακρίβειας αριθμητικών πράξεων είναι πολύ χαμηλότερη εκείνης απλής ακρίβειας. Υπενθυμίζεται ότι η ταχύτητα εκτέλεσης αριθμητικών πράξεων κινητής υποδιαστολής μίας GTX 285 είναι 1063 Gflops για πραγματικούς αριθμούς απλής ακρίβειας και μόλις 89 Gflops για πραγματικούς αριθμούς διπλής ακρίβειας.

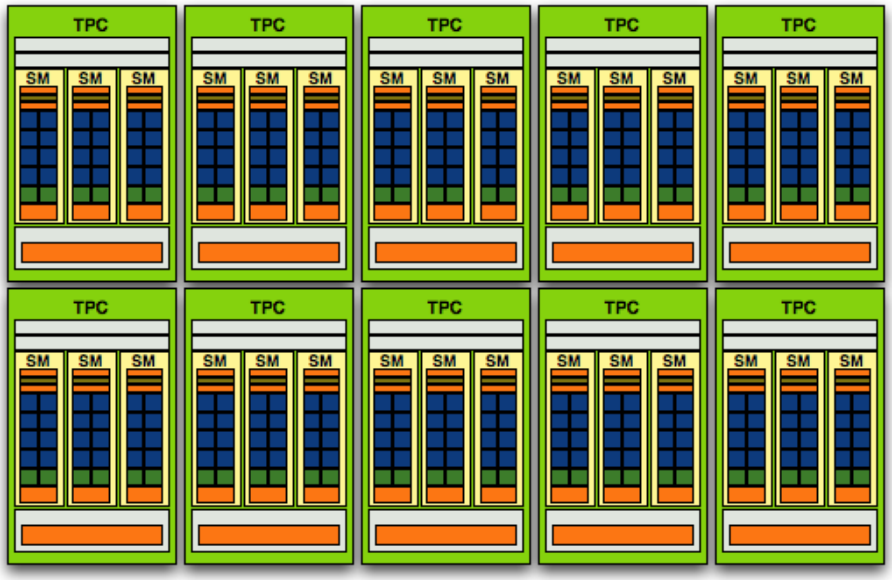
Ο warp scheduler είναι μία ειδική μονάδα που οργανώνει και διαχειρίζεται τα warps που εκτελούνται στον πολυεπεξεργαστή. Συγκεκριμένα, ομαδοποιεί τα threads των blocks που έχουν ανατεθεί στον πολυεπεξεργαστή σε warps και αναθέτει μία-προς-μία τις προς εκτέλεση εργασίες στα warps. Το σημαντικό είναι ότι από τη στιγμή που ο warp scheduler αναθέτει μία δουλειά σε ένα warp δεν περιμένει να ολοκληρωθεί η εκτέλεση αυτής, αλλά μπορεί να αναθέσει μία άλλη εργασία σε ένα άλλο warp. Δηλαδή, τη στιγμή που ένα warp προσπελαύνει θέσεις των μνημών της GPU, ένα άλλο warp στον ίδιο πολυεπεξεργαστή μπορεί να εκτελεί πράξεις κινητής υποδιαστολής, ενώ ένα τρίτο να χρησιμοποιεί τις SFUs.

Η constant cache έχει διάσταση 8 KByte και μεσολαβεί στην ανάγνωση δεδομένων από την constant μνήμη. Η τελευταία βρίσκεται στον ίδιο χώρο με την κεντρική της GPU και έχει διάσταση ίση με 64 KByte.

Η shared μνήμη έχει διάσταση 16 KByte, μοιράζεται από τα blocks που έχουν ανατεθεί στον πολυεπεξεργαστή, και μπορεί να χρησιμοποιηθεί για τη μεταβίβαση δεδομένων μεταξύ των threads του ίδιου block. Οι registers διανέμονται στα warps που χειρίζεται ο πολυεπεξεργαστής. Η διάσταση της shared μνήμης και το σύνολο των registers που διαθέτει ένας πολυεπεξεργαστής είναι αρκετά σημαντικά στοιχεία, καθώς καθορίζουν τον αριθμό των blocks, άρα και των warps, που ανατίθενται στους πολυεπεξεργαστές. Γενικά, ανεξάρτητα της διάστασης της shared μνήμης και του πλήθους των registers, ο μέγιστος αριθμός threads που μπορεί να χειριστεί ένας πολυεπεξεργαστής είναι 1024.

Τελικά, μία κάρτα γραφικών που βασίζεται στην αρχιτεκτονική GT200 περιέχει συνολικά $10 \times 3 = 30$ πολυεπεξεργαστές οι οποίοι μπορούν να χειριστούν έως και $30 \times 1024 = 30720$ threads.

Κάθε thread έχει πρόσβαση σε 16 KByte μνήμης (τοπική μνήμη). Ο χώρος που προορίζεται για τις ανά thread τοπικές μνήμες βρίσκεται στην κεντρική μνήμη της GPU. Η συνολική διάσταση της κεντρικής μνήμης της GPU είναι 1 GByte.



Σχήμα 3.2: Σχεδιάγραμμα της αρχιτεκτονικής GT200, [126]. Κάθε κάρτα γραφικών που βασίζεται στην αρχιτεκτονική GT200 αποτελείται από 10 TPCs. Κάθε TPC έχει τη δική του texture cache που μοιράζονται οι 3 πολυεπεξεργαστές που περιέχει. Συνεπώς, μία κάρτα γραφικών αρχιτεκτονικής GT200 έχει συνολικά 30 πολυεπεξεργαστές.

3.5 Η αρχιτεκτονική καρτών γραφικών Fermi

Οι πολυεπεξεργαστές των καρτών γραφικών που βασίζονται στην αρχιτεκτονική Fermi (σχήμα 3.3, [127]) συγκροτούνται σε GPCs (Graphics Processor Clusters). Αναλυτικά, κάθε GPC περιέχει:

- 4 πολυεπεξεργαστές.
- 1 ενδιάμεση μνήμη (L2 cache).

Κάθε πολυεπεξεργαστής αποτελείται από:

- 32 ή 48 πυρήνες.
- 4 ή 8 ειδικές μονάδες εκτέλεσης μαθηματικών συναρτήσεων (Special Function Units, SFU).
- 2 warp schedulers.
- 2 ενδιάμεσες μνήμες (constant cache, texture cache).
- 1 ενδιάμεση μνήμη (L1 cache).
- 1 γρήγορα προσπελάσιμη μνήμη (shared μνήμη).
- 32768 32-bit καταχωρητές (registers).

Ο αριθμός των πυρήνων ενός πολυεπεξεργαστή και εκείνος των SFUs εξαρτάται από την υπολογιστική δυνατότητα της GPU. Έτσι, κάρτες υπολογιστικής δυνατότητας 2.0, όπως οι Tesla M2050, έχουν 32 πυρήνες και 4 SFUs, ενώ κάρτες δυνατότητας 2.1 έχουν 48 πυρήνες και 8 SFUs.

Οι warp schedulers ορίζουν στα warps τις εργασίες που θα εκτελέσουν. Τα warps που εκτελούνται σε έναν πολυεπεξεργαστή χαρακτηρίζονται από έναν τοπικό αύξοντα αριθμό. Δηλαδή τα threads 0 έως 31 ανήκουν στο warp 0 κ.ο.κ. Έτσι, ο πρώτος από τους δύο warp schedulers χειρίζεται τα warps με περιττό αριθμό, ενώ ο δεύτερος με άρτιο. Επιπλέον, οι 32 (ή 48) πυρήνες του πολυεπεξεργαστή χωρίζονται σε δύο ομάδες των 16 (ή 24). Έτσι, ο πρώτος warp scheduler ορίζει στα warps που χειρίζεται εκείνος τη χρήση της πρώτης ομάδας πυρήνων για την εκτέλεση αριθμητικών πράξεων μεταξύ ακεραίων, ή πραγματικών αριθμών απλής ακρίβειας. Αντίστοιχα ενεργεί ο δεύτερος warp scheduler αξιοποιώντας τη δεύτερη ομάδα πυρήνων. Σε περίπτωση που απαιτείται η εκτέλεση αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας χρησιμοποιούνται από τους warp schedulers το σύνολο των πυρήνων ανά πολυεπεξεργαστή. Γίνεται αντιληπτό, ότι όταν ένα warp εκτελεί πράξεις διπλής ακρίβειας, κανένα άλλο warp του πολυεπεξεργαστή δεν μπορεί να χρησιμοποιήσει τους πυρήνες του πολυεπεξεργαστή. Αντίθετα, επιτρέπεται η χρήση των SFUs ή η προσπέλαση των μνημών της GPU από τα threads ενός άλλου warp. Στις κάρτες γραφικών 2.0 οι warp schedulers αναθέτουν 2 οδηγίες σε 2 warps. Αντίθετα, σε κάρτες γραφικών 2.1, οι warp schedulers αναθέτουν 2 οδηγίες ανά warp, δηλαδή συνολικά 4 οδηγίες σε 2 warps.

Το γεγονός ότι οι warp schedulers αναθέτουν την εκτέλεση των αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας στο σύνολο των πυρήνων του πολυεπεξεργαστή ενώ εκείνες μεταξύ πραγματικών αριθμών απλής ακρίβειας στους μισούς πυρήνες, κάνει την ταχύτητα εκτέλεσης αριθμητικών πράξεων διπλής ακρίβειας τη μισή εκείνων με απλή ακρίβεια. Για παράδειγμα, μια Tesla M2050 χαρακτηρίζεται από ταχύτητα εκτέλεσης αριθμητικών πράξεων απλής ακρίβειας ίση με 1030 Gflops και διπλής ακρίβειας ίση με 515 Gflops. Υπενθυμίζεται ότι ο λόγος των ταχυτήτων εκτέλεσης αριθμητικών πράξεων διπλής προς απλής ακρίβειας είναι περίπου ίσος με 1/12 στις κάρτες γραφικών αρχιτεκτονικής GT200.

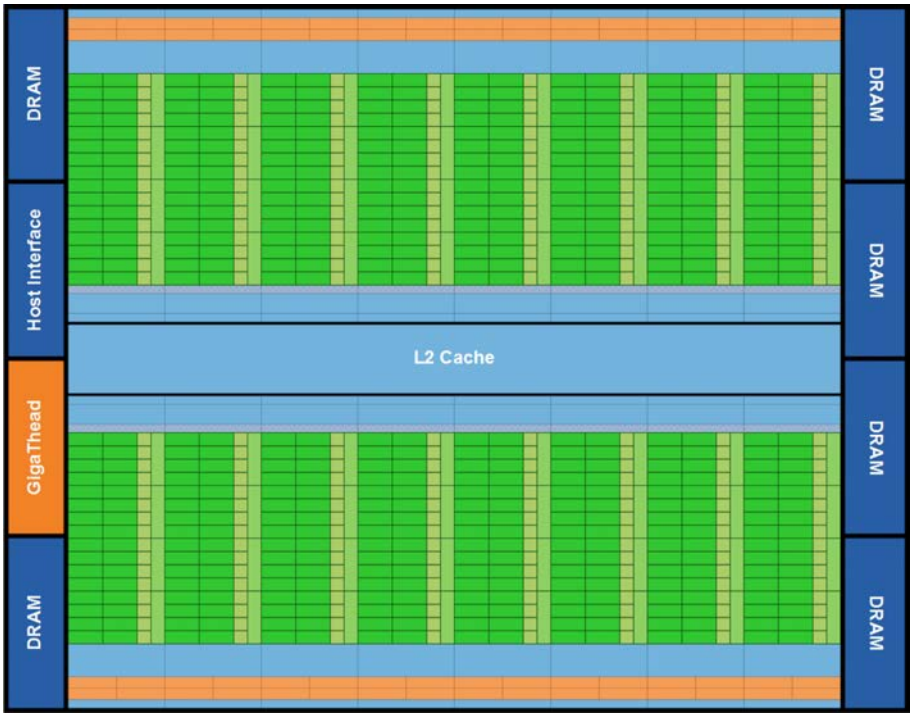
Οι ενδιάμεσες μνήμες constant, texture cache επιταχύνουν την ανάγνωση δεδομένων από την constant και την texture μνήμη, αντίστοιχα, που βρίσκονται στον ίδιο χώρο με την κεντρική μνήμη της GPU. Η διάσταση της texture cache εξαρτάται από τον τύπο της κάρτας γραφικών και είναι ανάμεσα σε 6 και 8 KByte ανά πολυεπεξεργαστή. Η constant cache έχει διάσταση 8 KByte και η constant μνήμη 64 KByte.

Οι ενδιάμεσες μνήμες L1, L2 cache επιταχύνουν την πρόσβαση στην κεντρική μνήμη της GPU και στις τοπικές ανά thread. Υπενθυμίζεται ότι σε κάθε thread αντιστοιχεί συγκεκριμένος χώρος της κεντρικής μνήμης. Ο χώρος αυτός ανά thread αποτελεί την τοπική ανά thread μνήμη. Στις κάρτες γραφικών αρχιτεκτονικής Fermi το μέγεθος της τοπικής ανά thread μνήμης είναι 512 KByte. Το συνολικό μέγεθος της L2 cache για όλους τους πολυεπεξεργαστές είναι 768 KByte. Η L1 βρίσκεται στον ίδιο χώρο με τη shared μνήμη και δίνεται η δυνατότητα στον προγραμματιστή να επιλέξει ανάμεσα σε 16 KByte L1 και 48 KByte shared (αποτελεί προεπιλογή) ή ανάποδα. Η επιλογή γίνεται σύμφωνα με τις απαιτήσεις του kernel σε shared μνήμη. Έτσι, στην περίπτω-

ση που ένα kernel χρησιμοποιεί λιγότερα των 16 KByte από την shared μνήμη ανά πολυεπεξεργαστή, συμφέρει ο προγραμματιστής να μοιράσει τα συνολικά 64 KByte ανά πολυεπεξεργαστή σε 16 KByte για την shared και 48 KByte για την L1 cache. Υπενθυμίζεται ότι η shared μνήμη επιτρέπει την επικοινωνία μεταξύ των threads του ίδιου block.

Κάθε πολυεπεξεργαστής περιέχει 32768 32-bit registers. Με βάση τις απαιτήσεις του kernel σε registers και KByte shared μνήμης ανά πολυεπεξεργαστή, καθορίζεται ο αριθμός των blocks που διανέμεται στους πολυεπεξεργαστές. Ο μέγιστος αριθμός threads που μπορεί γενικά να χειριστεί ένας πολυεπεξεργαστής είναι 1536.

Συνοψίζοντας, οι Tesla M2050 περιέχουν συνολικά 448 πυρήνες ή 14 πολυεπεξεργαστές. Δηλαδή, μπορούν να χειρίζονται έως και $14 \times 1536 = 21504$ threads.



Σχήμα 3.3: Σχεδιάγραμμα της αρχιτεκτονικής Fermi, [127]. Σε αυτήν, 16 πολυεπεξεργαστές μοιράζονται την ίδια L2 cache που επιταχύνει τη προσπέλαση της κεντρικής μνήμης της κάρτας γραφικών και της τοπικής ανά thread που βρίσκεται στον ίδιο χώρο. Ο δίαυλος επικοινωνίας (Host Interface) επιτρέπει τη μεταφορά δεδομένων από την κεντρική μνήμη της κάρτας γραφικών σε εκείνη του υπολογιστή και ανάποδα. Μία ειδική μονάδα (GigaThread) διανέμει τα blocks στους διαθέσιμους πολυεπεξεργαστές. Κάθε πολυεπεξεργαστής έχει 32 πυρήνες, 4 ειδικές μονάδες εκτέλεσης μαθηματικών συναρτήσεων, 2 μονάδες χειρισμού των warps, registers των 32-bit και 64 KB shared και L1 cache.

3.6 Σύγκριση των αρχιτεκτονικών Fermi και GT200

Συγκρίνοντας την αρχιτεκτονική των σημερινών καρτών γραφικών της NVIDIA, Fermi, με την προκάτοχό της GT200, φαίνεται ότι οι GPUs αρχιτεκτονικής Fermi χρησιμοποιούν λιγότερους, αλλά υψηλότερης υπολογιστικής ισχύος, πολυεπεξεργαστές σε σχέση με τις GPUs αρχιτεκτονικής GT200. Συγκεκριμένα, μία GPU αρχιτεκτονικής Fermi περιέχει έως και 16 πολυεπεξεργαστές ενώ μία GPU αρχιτεκτονικής GT200 έχει 30. Όμως, κάθε πολυεπεξεργαστής μίας κάρτας αρχιτεκτονικής Fermi έχει:

- υπερδιπλάσιο αριθμό μονάδων εκτέλεσης αριθμητικών πράξεων και μαθηματικών συναρτήσεων,
- τη δυνατότητα εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας πολύ πιο γρήγορα,
- δύο, αντί ενός, warp schedulers,
- τη δυνατότητα χρήση μεγαλύτερης έκτασης shared μνήμης και
- περισσότερους registers.

Τονίζεται η σημασία της μεγάλης αύξησης της ταχύτητας εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας, καθώς η χρήση αριθμητικής διπλής ακρίβειας είναι απαραίτητη σε πολλούς επιστημονικούς κλάδους, όπως λ.χ. στην υπολογιστική ρευστοδυναμική, όπου η ακριβής πρόλεξη των συνεκτικών φαινομένων επιβάλλει τη χρήση της. Επιπλέον, η δυνατότητα χρήσης περισσότερων registers και shared μνήμης επιτρέπει την ανάθεση περισσότερων blocks ανά πολυεπεξεργαστή.

Επίσης, η κεντρική μνήμη και η τοπική ανά thread μνήμη είναι cached[1], καθώς η πρόσβαση στις τελευταίες επιταχύνεται από τις L1, L2 cache. Επιπλέον, ορισμένες GPUs αρχιτεκτονικής Fermi επιτρέπουν την ταυτόχρονη εκτέλεση διαφορετικών kernels, με την προϋπόθεση φυσικά ότι επαρκούν οι διαθέσιμοι πολυεπεξεργαστές (παράγραφος 3.12).

[1]Μία κύρια μνήμη χαρακτηρίζεται ως cached, όταν η προσπέλαση αυτής γίνεται μέσω μίας ενδιάμεσης (cache) μνήμης. Η ενδιάμεση μνήμη έχει αισθητά μικρότερη χωρητικότητα σε σχέση με την κύρια, αλλά είναι εγκατεστημένη κοντά στους επεξεργαστές, οπότε η προσπέλασή της είναι αισθητά πιο γρήγορη. Χρησιμοποιείται για την προσωρινή αποθήκευση των θέσεων της κύριας μνήμης στις οποίες η πρόσβαση ζητήθηκε χρονικά τελευταία. Σημειώνεται ότι για τη σωστή λειτουργία της ενδιάμεσης μνήμης αποθηκεύονται και οι δείκτες που δείχνουν στις αντίστοιχες θέσεις της κύριας μνήμης. Συνεπώς όταν ζητείται η ανάγνωση μίας θέσης της κύριας μνήμης από ένα CPU-, ή GPU-thread, γίνεται ο έλεγχος αν η τιμή της θέσης αυτής βρίσκεται στην ενδιάμεση μνήμη. Στην περίπτωση που αυτή βρίσκεται στην ενδιάμεση μνήμη (cache hit), η ανάγνωση πραγματοποιείται από αυτή (γρήγορη ανάγνωση). Σε αντίθετη περίπτωση (cache miss), η ζητούμενη τιμή μεταφέρεται από την κύρια στην ενδιάμεση μνήμη αντικαθιστώντας μία άλλη θέση μνήμης και η ανάγνωση γίνεται πάλι από την ενδιάμεση μνήμη (αργή ανάγνωση). Στην περίπτωση που ζητείται η αποθήκευση μίας τιμής, τότε η αποθήκευση γίνεται στην ενδιάμεση μνήμη και το thread προχωρά στην εκτέλεση της επόμενης εντολής. Οι αντίστοιχες θέσεις της κύριας μνήμης ανανεώνονται είτε εκείνη τη στιγμή είτε στο τέλος της εκτέλεσης όλου του kernel. Τονίζεται ότι το thread αποθηκεύει πάντα στην ενδιάμεση μνήμη, δηλαδή σε αντίθεση με την ανάγνωση, η αποθήκευση ενός δεδομένου είναι γρήγορη και δεν καθυστερεί το thread.

3.7 Πολλαπλασιασμός δύο πινάκων στην CUDA

Με την ολοκλήρωση της παρουσίασης των αρχιτεκτονικών GT200 και Fermi, ο αναγνώστης έχει μία καλή εικόνα της εξέλιξης της αρχιτεκτονικής των σύγχρονων GPUs και του τρόπου παράλληλης επεξεργασίας ενός όγκου δεδομένων από αυτές. Η παράγραφος αυτή παρουσιάζει ένα απλό παράδειγμα παράλληλου υπολογισμού του γινομένου δύο πινάκων $\left([C]_{n_i \times n_j} = [A]_{n_i \times n_k} \cdot [B]_{n_k \times n_j}\right)$ στο περιβάλλον της CUDA. Υπενθυμίζεται ότι το περιβάλλον της CUDA δίνει τη δυνατότητα προγραμματισμού ετερογενών υπολογιστικών πλατφορμών, όπως είναι η CPU και η GPU. Συγκεκριμένα κάθε GPU-κώδικας ξεκινά στη CPU, η οποία χειρίζεται και καθοδηγεί τη GPU, με τη βοήθεια μίας σειράς συναρτήσεων δέσμευσης, αποδέσμευσης, αντιγραφής θέσεων μνήμης και την κλήση προγραμματισμένων συναρτήσεων που εκτελούνται στη GPU. Ο ψευδοκώδικας 3 παρουσιάζει την κύρια συνάρτηση (function) που εκτελείται στη CPU. Αντίστοιχα, ο ψευδοκώδικας 4 παρουσιάζει τη συνάρτηση υπολογισμού του γινομένου C, που εκτελείται σε μία GPU. Επιλέχθηκε σκόπιμα η συνάρτηση αυτή να χρησιμοποιεί μόνο την κεντρική μνήμη της GPU. Μετά την ενότητα 3.8, όπου τονίζεται η σημασία της σωστής διαχείρισης των μνημών της GPU και παρουσιάζεται ο αποδοτικότερος τρόπος πρόσβασης και διαχείρισης αυτών, θα επεκταθεί το παράδειγμα αυτό με σκοπό την αύξηση της παράλληλης απόδοσης του kernel υπολογισμού του γινομένου C.

Περιγραφή της κύριας συνάρτησης

Επισημαίνεται ότι η κύρια συνάρτηση του προγράμματος εκτελείται στην CPU. Αναλυτικότερα, όπως φαίνεται στον ψευδοκώδικα 3, η CPU αφού διαβάσει τη διάσταση των πινάκων A, B ($n_i \times n_k$, $n_k \times n_j$ αντίστοιχα) δεσμεύει το απαραίτητο πλήθος θέσεων της κεντρικής μνήμης του υπολογιστή και της GPU για την αποθήκευση των πινάκων A, B και C. Οι μεταβλητές A, B, C στον ψευδοκώδικα 3 αποτελούν τις θέσεις μνήμης του υπολογιστή στις οποίες αποθηκεύει η CPU τους αντίστοιχους πίνακες. Οι μεταβλητές $\_A$, $\_B$, $\_C$ αποτελούν τις αντίστοιχες θέσεις της κεντρικής μνήμης της GPU. Προφανώς, οι θέσεις μνήμης A, B, C είναι διαφορετικές από τις $\_A$, $\_B$, $\_C$. Έπειτα, η CPU διαβάζει από κατάλληλα διαμορφωμένο αρχείο τα στοιχεία των πινάκων A, B και τα αντιγράφει στις αντίστοιχες θέσεις μνήμης της GPU.

Στη συνέχεια, η CPU ορίζει τη διάσταση των blocks και του grid. Ο προγραμματιστής έχει πρωτίστως αποφασίσει ποιές εργασίες θα αναθέσει στα threads τις GPU. Στο παράδειγμα αυτό, ο προγραμματιστής έχει αναθέσει σε κάθε thread τον υπολογισμό ενός στοιχείου του πίνακα C. Συνεπώς, χρειάζονται συνολικά $n_i \times n_j$ threads. Επιλέγεται κάθε block να αποτελείται από 16×16 threads. Έτσι, αφού χρειάζονται $n_i \times n_j$ threads, ο προγραμματιστής ορίζει ένα grid διάστασης ${}^{n_i}/_{16} \times {}^{n_j}/_{16}$. Έστω ότι, για την απλούστευση του παραδείγματος, γίνεται η υπόθεση ότι οι διαστάσεις n_i, n_j, n_k είναι ακέραια πολλαπλάσια του 16. Σε αντίθετη περίπτωση, από την στιγμή που όλα τα blocks πρέπει να περιέχουν ίδιο αριθμό από threads, ο προγραμματιστής οφείλει να ζητήσει τη δημιουργία περισσότερων threads από $n_i \times n_j$. Τα πλεονάζοντα όμως threads μένουν ανενεργά. Στο σχήμα 3.4 φαίνονται οι θέσεις του πίνακα C που πρόκειται

να υπολογιστούν από τα threads ενός τυχαίου block.

Έπειτα η CPU καλεί την εκτέλεση της συνάρτησης *matmulGPU* που εκτελείται στη GPU και υπολογίζει τα στοιχεία του πίνακα C. Ξεχωρίζει η δήλωση των διαστάσεων του grid και των blocks στην κλήση της συνάρτησης. Ο τρόπος κλήσης από την CPU μίας συνάρτησης που εκτελείται στην GPU είναι

Ψευδοκώδικας 2 Κλήση από την CPU μίας συνάρτησης που εκτελείται στην GPU

```
1: ΟΝΟΜΑ_ΣΥΝΑΡΤΗΣΗΣ<<<GRID_SIZE,BLOCK_SIZE>>>(ορίσματα συνάρτησης)
```

Ανάλογα με την υπολογιστική δυνατότητα της GPU, οι μεταβλητές εισόδου που δέχεται μία συνάρτηση που εκτελείται στη GPU αντιγράφονται προσωρινά είτε στην shared μνήμη ανά πολυεπεξεργαστή (GPUs 1.x), είτε στην constant μνήμη (GPUs 2.x). Έτσι, αυτές είναι γρήγορα προσπελάσιμες από τα threads της GPU. Αυτό σημαίνει ότι υπάρχει ένα άνω όριο στο μέγεθος των μεταβλητών εισόδου μίας GPU-συνάρτησης. Συγκεκριμένα, το μέγιστο δυνατό συνολικό μέγεθος των μεταβλητών εισόδου είναι 256 Bytes, ή 4 KBytes σε κάρτες γραφικών υπολογιστικής δυνατότητας 1.x, ή 2.x αντίστοιχα. Το συνολικό μέγεθος των μεταβλητών εισόδου δεν συμφέρει να είναι μεγάλο, καθώς αυξάνει ο όγκος των δεδομένων που αντιγράφονται στη GPU κατά την κλήση μίας GPU-συνάρτησης. Επομένως, μόνο οι θέσεις μνήμης των πινάκων A, B, C δηλώνονται ως μεταβλητές εισόδου στην *matmulGPU* και όχι ολόκληροι οι πίνακες. Γι΄ αυτό, ορίσματα εισόδου είναι οι θέσεις $\_A$, $\_B$, $\_C$ της κεντρικής μνήμης της GPU.

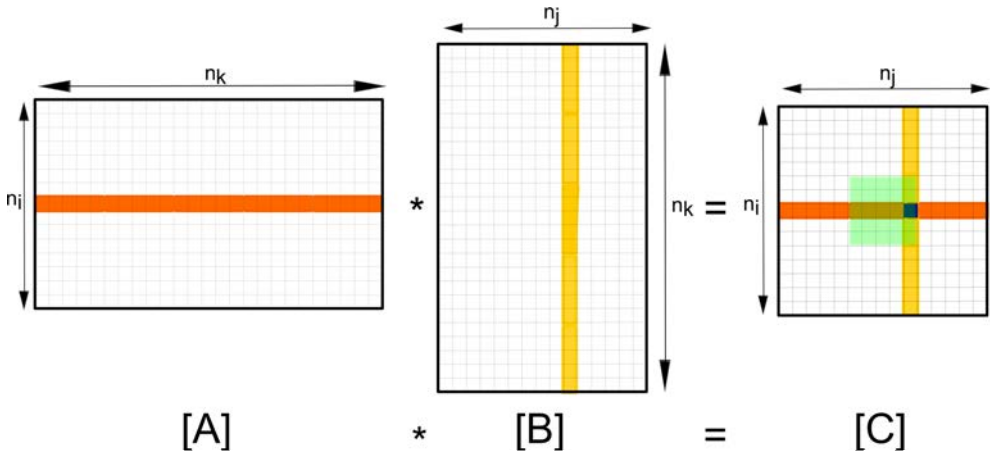
Με την ολοκλήρωση του υπολογισμού των στοιχείων του πίνακα C, εκείνα αντιγράφονται στη CPU, η οποία με την σειρά της τα αποθηκεύει σε κάποιο αρχείο. Σημειώνεται ότι η κλήση ενός kernel είναι ασύγχρονη. Δηλαδή, η CPU δεν περιμένει να ολοκληρώσουν τα threads της GPU την εκτέλεση της *matmulGPU* και προχωρά στην εκτέλεση της επόμενης εντολής του προγράμματος. Στην περίπτωση του παραδείγματος, όμως, αυτή η εντολή είναι η αντιγραφή δεδομένων από τη GPU στη CPU. Η GPU εκτελεί τις εντολές που της υπαγορεύει η CPU τη μία μετά την άλλη. Επομένως, η αντιγραφή του πίνακα C από τη GPU στη CPU θα γίνει μετά την ολοκλήρωση της *matmulGPU*.

Το πρόγραμμα τερματίζει με την αποδέσμευση των θέσεων μνήμης του υπολογιστή και της GPU. Η δέσμευση, αντιγραφή και αποδέσμευση θέσεων μνήμης της GPU γίνεται με τη χρήση ειδικών συναρτήσεων του περιβάλλοντος προγραμματισμού της CUDA.

Ψευδοκώδικας 3 Κεντρική συνάρτηση

```
1: #define nbx  16
2: #define nby  16
3:
4: function main
5:     // Ανάγνωση των διαστάσεων των πινάκων A, B
6:     read(ni, nj, nk)
7:
8:     // Δέσμευση θέσεων της κεντρικής μνήμης του υπολογιστή
9:     A ← allocateMem_on_CPU(ni, nk)
10:    B ← allocateMem_on_CPU(nk, nj)
11:    C ← allocateMem_on_CPU(ni, nj)
12:
13:    // Δέσμευση θέσεων της κεντρικής μνήμης της GPU
14:    _A ← allocateMem_on_GPU(ni, nk)
15:    _B ← allocateMem_on_GPU(nk, nj)
16:    _C ← allocateMem_on_GPU(ni, nj)
17:
18:    // Ανάγνωση των πινάκων A, B από αρχείο (από τη CPU)
19:    readMatrices(A, B)
20:
21:    // Αντιγραφή δεδομένων στη GPU
22:    sizeA ← ni * nk * sizeof(double)
23:    sizeB ← nk * nj * sizeof(double)
24:
25:    memory_copy(_A, A, sizeA, CPU_to_GPU)
26:    memory_copy(_B, B, sizeB, CPU_to_GPU)
27:
28:    // Πολλαπλασιασμός των πινάκων A, B της GPU
29:    blockSize ← (nbx     , nby     )
30:    gridSize  ← (ni/nbx, nj/nby)
31:    matmulGPU <<< gridSize, blockSize >>>(_A, _B, _C, nk)
32:
33:    // Αντιγραφή αποτελεσμάτων στη CPU
34:    sizeC ← ni * nj * sizeof(double)
35:    memory_copy(C, _C, sizeC, GPU_to_CPU)
36:
37:    // Αποθήκευση του πίνακα C σε αρχείου εξόδου
38:    save(C)
39:
```

```
40:     // Αποδέσμευση θέσεων της κεντρικής μνήμης του υπολογιστή
41:     deallocateMem_on_CPU( A)
42:     deallocateMem_on_CPU( B)
43:     deallocateMem_on_CPU( C)
44:
45:     // Αποδέσμευση θέσεων της κεντρικής μνήμης της GPU
46:     deallocateMem_on_GPU(_A)
47:     deallocateMem_on_GPU(_B)
48:     deallocateMem_on_GPU(_C)
49: end function
```



Σχήμα 3.4: Σχηματική αναπαράσταση του υπολογισμού του γινομένου δύο πινάκων $\left([C]_{n_i \times n_j} = [A]_{n_i \times n_k} \cdot [B]_{n_k \times n_j}\right)$ στη GPU. Κάθε thread υπολογίζει την τιμή ενός στοιχείου (i, j) του πίνακα C σαρώνοντας τη γραμμή i και τη στήλη j των πινάκων A, B αντίστοιχα. Στη γραφική απεικόνιση του πίνακα C φαίνεται με έντονο χρώμα το στοιχείο (i, j) που υπολογίζει ένα τυχαίο thread. Χρωματισμένα με λιγότερο έντονο χρώμα είναι τα στοιχεία υπολογίζονται από τα υπόλοιπα threads του block στο οποίο ανήκει το εν λόγω τυχαίο thread.

Περιγραφή της συνάρτησης υπολογισμού του γινομένου C σε μία GPU

Το kernel (συνάρτηση $matmulGPU$) υπολογισμού του πίνακα C περιγράφεται στον ψευδοκώδικα 4. Στην ουσία, ο προγραμματιστής δίνει τη σειρά εντολών που εκτελεί κάθε thread. Δηλαδή, η CPU καλεί μία φορά την εκτέλεση της $matmulGPU$, η οποία εκτελείται $n_i \times n_j$ φορές, μία φορά για κάθε ένα από τα $n_i \times n_j$ threads της GPU.

Ένα thread που εκτελεί την $matmulGPU$ αρχικά 'συνδέεται' με ένα στοιχείο (i, j) του πίνακα C. Αυτό γίνεται χρησιμοποιώντας τον τοπικό (στο block) αύξοντα αριθμό του thread $(threadIdx.x, threadIdx.y)$, τη διάσταση του block $(blockDim.x, blockDim.y)$ και τον αύξοντα αριθμό του block στο grid $(blockIdx.x, blockIdx.y)$. Στη συνέχεια, κάθε thread σαρώνει τα στοιχεία της γραμμής i του πίνακα A και τα πολλαπλασιάζει, ένα προς ένα, με εκείνα της στήλης j του πίνακα B για τον υπολογισμό του στοιχείου

(i, j) του πίνακα C. Κατά τη σάρωση της γραμμής i και της στήλης j των πινάκων A, B αντίστοιχα, ανανεώνεται η τοπική μεταβλητή cij και όχι η $C(i, j)$ που βρίσκεται στην κεντρική μνήμη της GPU, καθώς η πρόσβαση στην cij είναι κατά πολύ γρηγορότερη. Η σχηματική απεικόνιση της λειτουργίας ενός thread φαίνεται στο σχήμα 3.4.

Ψευδοκώδικας 4 kernel υπολογισμού του γινομένου των A, B

```
1: function matmulGPU(A, B, C, nk)
2:     // Συσχέτιση των threads με στοιχεία του πίνακα C
3:     i ← blockIdx.x * blockDim.x + threadIdx.x
4:     j ← blockIdx.y * blockDim.y + threadIdx.y
5:
6:     // Υπολογισμός της θέσης C(i, j)
7:     cij ← 0.
8:     for k ← 0, nk do
9:         cij ← cij + A(i, k) * B(k, j)
10:    end for
11:    C(i, j) ← cij
12: end function
```

3.8 Ανάλυση των μνημών μίας GPU

Το μέγεθος των ενδιάμεσων (cache) μνήμων των GPUs είναι πολύ μικρότερο σε σχέση με εκείνο των σύγχρονων CPUs. Ενδεικτικά αναφέρεται ότι κάθε blade server του cluster διασυνδεδεμένων GPUs της ΜΠΥΡ&Β/ΕΘΣ, έχει 2 quad core CPUs. Κάθε πυρήνας μίας CPU έχει 12288 KByte cache μνήμη. Το μέγεθος αυτό είναι πολύ μεγαλύτερο σε σχέση με το συνολικό μέγεθος των cache μνημών ανά πολυεπεξεργαστή μίας GPU. Οι cache μνήμες μίας GPU είναι η constant και η texture cache και οι L1, L2 cache (αν η GPU είναι υπολογιστικής δυνατότητας 2.x). Δεν θα ήταν λάθος να προστεθεί στις μνήμες που αναφέρθηκαν προηγουμένως και η γρήγορη shared μνήμη, καθώς η ταχύτητα πρόσβασης σε αυτή είναι ίδια με την ταχύτητα πρόσβασης σε cache μνήμες. Συνολικά, μία GPU αρχιτεκτονικής GT200 έχει 8 KByte constant cache, 8 KByte texture cache και 16 KByte shared μνήμη, ήτοι μόλις 32 KByte cache μνημών στο σύνολο. Το συνολικό μέγεθος των cache μνημών μίας κάρτας γραφικών αρχιτεκτονικής Fermi, παρόλο που είναι μεγαλύτερο αυτού μίας κάρτας αρχιτεκτονικής GT200, παραμένει πολύ μικρότερο σε σχέση με το αντίστοιχο μέγεθος των σύγχρονων CPUs. Συγκεκριμένα, μία κάρτα γραφικών αρχιτεκτονικής Fermi έχει 8 KByte constant cache, 8 KByte texture cache, 64 KByte για την shared και την L1 cache και 48 KByte L2 cache ανά πολυεπεξεργαστή. Συνολικά, δηλαδή, έχει 128 KByte cache μνημών.

Η μικρή έκταση των cache μνήμων της GPU τονίζει τη σημασία της επιλογής του αποδοτικότερου τρόπου προσπέλασης των μνημών της GPU και καθιστά άρρηκτα συνδεδεμένη την απόδοση ενός GPU-κώδικα με θέματα διαχείρισης των μνημών της

κάρτας γραφικών. Στις επόμενες παραγράφους περιγράφονται αναλυτικότερα οι μνήμες της GPU και παρουσιάζεται ο βέλτιστος τρόπος διαχείρισής τους.

3.8.1 Κεντρική μνήμη (global memory)

Η κεντρική μνήμη της GPU (global memory) αποτελεί τη μεγαλύτερη σε έκταση μνήμη της κάρτας γραφικών και είναι προσπελάσιμη από όλα τα threads. Χαρακτηρίζεται, όμως, από υψηλό χρόνο προσπέλασης. Συνεπώς, η χρήση της γρήγορα προσπελάσιμης shared μνήμης, ή των cached constant και texture μνημών αντί της κεντρικής, φυσικά όταν αυτό γίνεται, αυξάνει κατακόρυφα την απόδοση του GPU-κώδικα. Οι shared, constant και texture μνήμες παρουσιάζονται αναλυτικά στις επόμενες παραγράφους. Επιπλέον η χρήση της κεντρικής μνήμης πρέπει να περιορίζεται, κατά το δυνατό, σε ένα kernel. Για παράδειγμα, στο kernel υπολογισμού του γινομένου δύο πινάκων στην παράγραφο 3.7 χρησιμοποιείται η τοπική μεταβλητή cij και η αντίστοιχη θέση της κεντρικής μνήμης ανανεώνεται στο τέλος του kernel.

Οι GPUs αντιμετωπίζουν την κεντρική μνήμη ως μία αλληλουχία από τμήματα μήκους 32, 64, ή 128 Bytes. Αυτό πρακτικά σημαίνει ότι ο δίαυλος μεταφοράς δεδομένων από την κεντρική μνήμη στους registers των πολυεπεξεργαστών, μεταφέρει τμήματα της κεντρικής μνήμης μήκους 32, 64, ή 128 Bytes. Επομένως, όσο μικρότερο είναι το πλήθος των τμημάτων της κεντρικής μνήμης στα οποία ανήκουν οι θέσεις που χειρίζονται τα threads ενός warp, τόσο λιγότερες φορές θα μεταφέρει δεδομένα ο δίαυλος επικοινωνίας. Σημειώνεται ότι η διάσταση ενός ακεραίου είναι 2 Bytes, η διάσταση ενός πραγματικού αριθμού απλής ακρίβειας 4 Bytes και εκείνη ενός διπλής ακρίβειας 8 Bytes. Συνεπώς, η συνολική διάσταση των ακεραίων, πραγματικών αριθμών απλής, ή διπλής ακρίβειας που χειρίζονται και τα 16 threads ενός half–warp είναι 32, 64, ή 128 Bytes, αντίστοιχα. Στη συνέχεια, περιγράφονται αναλυτικότερα οι 'κανόνες' προσπέλασης της κεντρικής μνήμης ανάλογα με την υπολογιστική δυνατότητα της κάρτας γραφικών.

Προσπέλαση της κεντρικής μνήμης μίας GPU 1.0 ή 1.1

Στις GPUs υπολογιστικής δυνατότητας 1.0 και 1.1, η προσπέλαση της κεντρικής μνήμης γίνεται με βάση τα half–warps. Έτσι, τα threads του δεύτερου μισού ενός warp πρέπει να περιμένουν να ολοκληρωθεί η πρόσβαση στην κεντρική μνήμη από τα threads του πρώτου half–warp. Το βέλτιστο πρότυπο πρόσβασης στην κεντρική μνήμη υπαγορεύει οι προς προσπέλαση από τα threads ενός half–warp θέσεις μνήμης να βρίσκονται στο ίδιο τμήμα της κεντρικής μνήμης. Το μήκος του τμήματος αυτού είναι 32 Bytes στην περίπτωση προσπέλασης ακεραίων αριθμών, 64 Bytes για πραγματικούς αριθμούς απλής ακρίβειας και 128 Bytes για πραγματικούς αριθμούς διπλής ακρίβειας. Απαιτείται επίσης, τα threads του ίδιου half–warp να χειρίζονται διαδοχικές θέσεις της κεντρικής μνήμης. Δηλαδή, το πρώτο thread του half–warp να διαβάζει/αποθηκεύει στην πρώτη θέση του τμήματος της κεντρικής μνήμης κ.ο.κ. Αν ικανοποιούνται οι παραπάνω απαιτήσεις πραγματοποιείται μόνο μία πρόσβαση σε ένα τμήμα της κεντρικής

μνήμης για όλα τα threads του half–warp. Σε αντίθετη περίπτωση, τα threads του half–warp πραγματοποιούν διαδοχικά πρόσβαση στο 32-Byte τμήμα της κεντρικής μνήμης όπου ανήκει η θέση που χειρίζεται το κάθε thread.

Προσπέλαση της κεντρικής μνήμης μίας GPU 1.2 ή 1.3

Όπως και στις κάρτες γραφικών υπολογιστικής δυνατότητας 1.0 και 1.1, έτσι και στις GPUs με δυνατότητα 1.2 και 1.3 η προσπέλαση της κεντρικής μνήμης γίνεται με βάση τα half–warps. Αντίθετα, δεν απαιτείται από τα threads του ίδιου half–warp να χειρίζονται διαδοχικές θέσεις της κεντρικής μνήμης.

Στην περίπτωση αποθήκευσης/ανάγνωσης πραγματικών αριθμών διπλής ακρίβειας από τα threads του ίδιου half–warp, η προσπέλαση της κεντρικής μνήμης γίνεται σύμφωνα με τα ακόλουθα βήματα:

1. Ορίζεται το 128-Byte τμήμα της κεντρικής μνήμης στο οποίο ανήκει η θέση που χειρίζεται το πρώτο ενεργό thread του half–warp.

2. Ορίζονται ποιά άλλα ενεργά threads του ίδιου half–warp χειρίζονται θέσεις του ίδιου τμήματος της κεντρικής μνήμης.

3. Στην περίπτωση που χρησιμοποιείται μόνο το πρώτο ή το δεύτερο μισό του 128-Byte τμήματος μνήμης ορίζεται εκείνο το 64-Byte τμήμα μνήμης ως το τμήμα μνήμης στο οποίο θα έχουν πρόσβαση τα ενεργά threads του half–warp.

4. Στην περίπτωση που το μήκος του τμήματος μνήμης που θα έχουν πρόσβαση τα ενεργά threads του half–warp μειώθηκε στο προηγούμενο βήμα σε 64 Bytes, γίνεται ο έλεγχος αν από το 64-Byte τμήμα χρησιμοποιείται μόνο το πρώτο ή το δεύτερο μισό. Αν χρησιμοποιείται μόνο ένα 32-Byte τμήμα μνήμης, τότε τα ενεργά threads του half–warp πραγματοποιούν πρόσβαση σε αυτό.

5. Γίνεται η προσπέλαση του τμήματος της κεντρικής μνήμης από τα ενεργά threads του half–warp.

6. Τα threads του half–warp που ολοκλήρωσαν την πρόσβαση στην κεντρική μνήμη χαρακτηρίζονται ως 'μη-ενεργά' και η διαδικασία επαναλαμβάνεται από το πρώτο βήμα μέχρι όλα τα threads του half–warp να ολοκληρώσουν την ανάγνωση/αποθήκευση στην κεντρική μνήμη.

Στην περίπτωση ανάγνωσης/αποθήκευσης πραγματικών αριθμών απλής ακρίβειας, ή ακεραίων, η προσπέλαση της κεντρικής μνήμης από τα threads του ίδιου half–warp ακολουθεί τα ίδια βήματα που παρουσιάστηκαν προηγουμένως, με τη διαφορά ότι το μήκος του τμήματος μνήμης που ορίζεται στο πρώτο βήμα είναι 64, ή 32 Bytes για πραγματικούς αριθμούς απλής ακρίβειας, ή ακεραίους, αντίστοιχα. Προφανώς, όταν χρησιμοποιούνται ακέραιοι τα βήματα 3, 4 δεν έχουν νόημα. Επίσης, όταν χρησιμοποιούνται πραγματικοί αριθμοί απλής ακρίβειας το βήμα 3 δεν έχει νόημα.

Προσπέλαση της κεντρικής μνήμης μίας GPU 2.x

Σε αντίθεση με ότι αναφέρθηκε προηγουμένως για τις GPUs με υπολογιστική δυνατότητα 1.x, οι κάρτες γραφικών 2.x αντιμετωπίζουν την κεντρική μνήμη αποκλειστικά ως ένα σύνολο από τμήματα μήκους 128 Bytes. Επιπλέον, η προσπέλαση της κεντρικής μνήμης γίνεται από το σύνολο των threads ενός warp και όχι του πρώτου και του δεύτερου half–warp διαδοχικά. Έτσι, οι θέσεις που χειρίζονται τα threads ενός warp ομαδοποιούνται σε τμήματα μνήμης των 128 Bytes, ανεξάρτητα αν χρησιμοποιούνται ακέραιοι ή πραγματικοί αριθμοί απλής, ή διπλής ακρίβειας. Επομένως, ο δίαυλος μεταφοράς δεδομένων χρησιμοποιείται τόσες φορές όσα είναι τα 128-Byte τμήματα της κεντρικής μνήμης που χρησιμοποιούν τα threads του warp.

Τονίζεται ότι η πρόσβαση στην κεντρική μνήμη των GPUs 2.x επιταχύνεται από τις L1, L2 cache.

Παραδείγματα προσπέλασης της κεντρική μνήμης

Το σχήμα 3.5 παρουσιάζει τρεις τρόπους προσπέλασης της κεντρικής μνήμης από τα threads ενός warp. Και στις τρεις περιπτώσεις χρησιμοποιούνται πραγματικοί αριθμοί διπλής ακρίβειας. Το σχήμα 3.5(α′) παρουσιάζει το βέλτιστο πρότυπο προσπέλασης της κεντρικής μνήμης από threads που εκτελούνται σε μία GPU 1.0, ή 1.1, καθώς τα threads του ίδιου half–warp έχουν πρόσβαση σε συνεχόμενες θέσεις του ίδιου 128-Byte τμήματος μνήμης. Προφανώς, threads που εκτελούνται σε GPUs υπολογιστικής δυνατότητας μεγαλύτερης ή ίσης του 1.2 ακολουθώντας το πρότυπο αυτό επιτυγχάνουν τη μέγιστη απόδοση. Συνεπώς, πραγματοποιείται μία πρόσβαση στην κεντρική ανά half–warp, ή δύο ανά warp.

Στο σχήμα 3.5(β′), threads που ανήκουν στο ίδιο half–warp έχουν τυχαία πρόσβαση σε θέσεις του ίδιου 128-Byte τμήματος της κεντρικής μνήμης. Η τυχαία πρόσβαση στην κεντρική μνήμη ζημιώνει την απόδοση των threads που εκτελούνται σε GPUs χαμηλής υπολογιστικής δυνατότητας (1.0, ή 1.1), καθώς εκείνα έχουν πρόσβαση, το ένα μετά το άλλο, σε 32-Byte τμήματα της κεντρικής μνήμης. Δηλαδή για να διαβάσει το δεύτερο thread του half–warp από την κεντρική μνήμη, πρέπει πρώτα να έχει διαβάσει το πρώτο thread του ίδιου half–warp κ.ο.κ. Αντίθετα, ο χρόνος πρόσβασης των threads που εκτελούνται σε GPUs υψηλότερης υπολογιστικής δυνατότητας, στην κεντρική μνήμη δεν επηρεάζεται σε σχέση με την περίπτωση (α′) αφού όλα τα threads του ίδιου half–warp έχουν πρόσβαση σε ένα 128 Byte τμήμα της κεντρικής μνήμης. Δηλαδή, συνολικά, πραγματοποιούνται μόλις δύο προσβάσεις στην κεντρική μνήμη ανά warp.

Το σχήμα 3.5(γ′) παρουσιάζει ένα πρότυπο προσπέλασης, όπου τα threads του ίδιου warp χειρίζονται συνεχόμενες θέσεις της κεντρικής μνήμης. Η πρώτη από τις θέσεις αυτές όμως, δεν αποτελεί την αρχή ενός 128-Byte τμήματος μνήμης. Συνεπώς, σε GPUs χαμηλής υπολογιστικής δυνατότητας (1.0, ή 1.1) πραγματοποιούνται 16 προσβάσεις από το πρώτο half–warp και άλλες 16 από το δεύτερο, συνολικά 32 προσβάσεις. Δηλαδή, τα threads έχουν σειριακά (το ένα μετά το άλλο) πρόσβαση σε 32-Byte τμήματα μνήμης. Υπενθυμίζεται ότι, στις κάρτες γραφικών 1.x, πραγματοποιείται πρώτα η πρόσβαση του πρώτου half–warp στην κεντρική μνήμη και έπειτα

του δεύτερου. Επομένως, σε κάρτες δυνατότητας 1.2, ή 1.3 πραγματοποιούνται δύο προσβάσεις ανά half–warp: μία σε ένα τμήμα μήκους 128 Bytes και μία σε ένα τμήμα μήκους 32 Bytes, ήτοι συνολικά, 4 προσβάσεις ανά warp. Σε κάρτες δυνατότητας 2.x, η πρόσβαση στην κεντρική μνήμη γίνεται από όλο το warp. Επιπλέον, αυτές οι GPUs αντιμετωπίζουν την κεντρική μνήμη ως ένα σύνολο από 128-Byte τμήματα. Συνεπώς, γίνονται 3 προσβάσεις, μία για κάθε 128-Byte τμήμα μνήμης, για όλο το warp.

Το πλήθος των προσβάσεων στην κεντρική μνήμη από ένα warp φαίνονται στους πίνακες 3.1, 3.2 και 3.3, για τις περιπτώσεις (α′), (β′) και (γ′) αντίστοιχα.

Υπολογιστική δυνατότητα της GPU	Πλήθος τμημάτων της κεντρικής μνήμης που έχουν πρόσβαση τα threads ενός warp
1.0, ή 1.1	1 x 128 Byte τμήμα ανά half–warp
1.2, ή 1.3	1 x 128 Byte τμήμα ανά half–warp
2.x	2 x 128 Byte τμήματα ανά warp

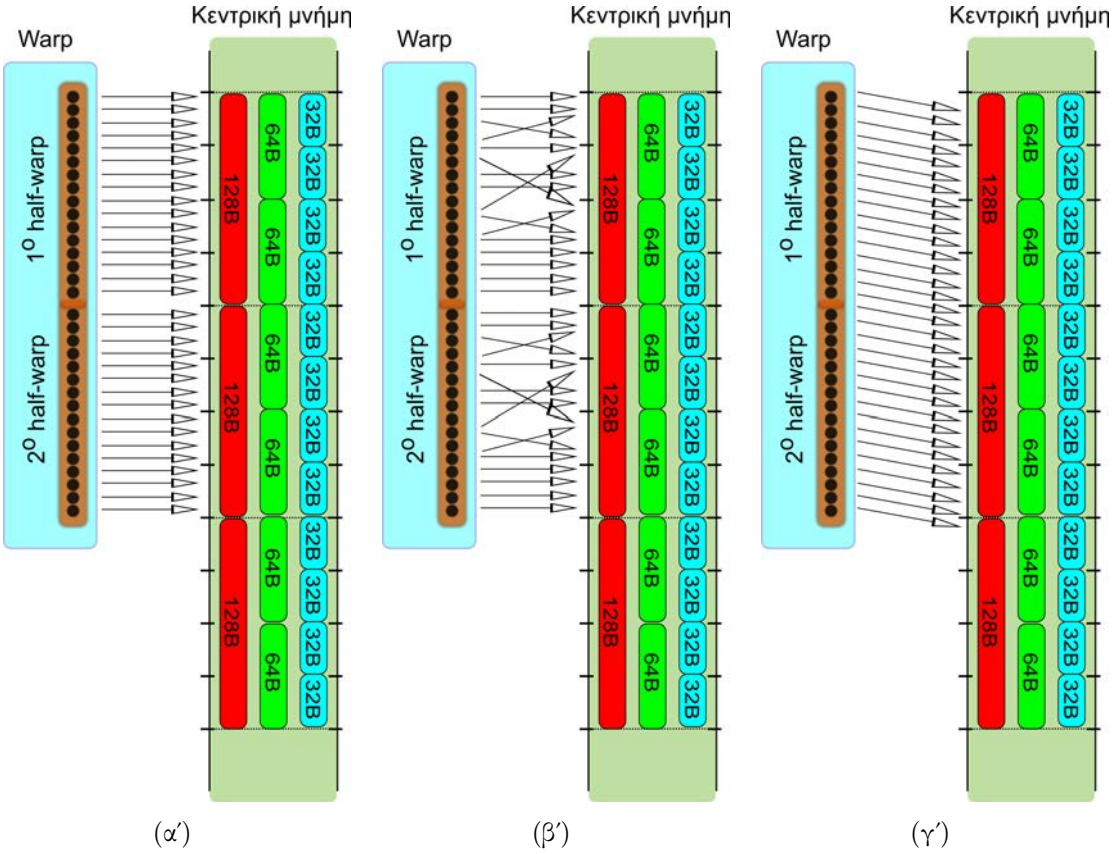
Πίνακας 3.1: Προσπέλαση της κεντρικής μνήμης σύμφωνα με το πρότυπο του σχήματος 3.5(α′) από GPUs διαφορετικής υπολογιστικής δυνατότητας. Χρησιμοποιούνται πραγματικοί αριθμοί διπλής ακρίβειας.

Υπολογιστική δυνατότητα της GPU	Πλήθος τμημάτων της κεντρικής μνήμης που έχουν πρόσβαση τα threads ενός warp
1.0, ή 1.1	16 x 32 Byte τμήματα ανά half–warp
1.2, ή 1.3	1 x 128 Byte τμήμα ανά half–warp
2.x	2 x 128 Byte τμήματα ανά warp

Πίνακας 3.2: Προσπέλαση της κεντρικής μνήμης σύμφωνα με το πρότυπο του σχήματος 3.5(β′) από GPUs διαφορετικής υπολογιστικής δυνατότητας. Χρησιμοποιούνται πραγματικοί αριθμοί διπλής ακρίβειας.

Υπολογιστική δυνατότητα της GPU	Πλήθος τμημάτων της κεντρικής μνήμης που έχουν πρόσβαση τα threads ενός warp
1.0, ή 1.1	16 x 32 Byte τμήματα ανά half–warp
1.2, ή 1.3	1 x 128 Byte και 1 x 32 Byte τμήμα ανά half–warp
2.x	3 x 128 Byte τμήματα ανά warp

Πίνακας 3.3: Προσπέλαση της κεντρικής μνήμης σύμφωνα με το πρότυπο του σχήματος 3.5(γ′) από GPUs διαφορετικής υπολογιστικής δυνατότητας. Χρησιμοποιούνται πραγματικοί αριθμοί διπλής ακρίβειας.



Σχήμα 3.5: Τρεις πιθανοί τρόποι προσπέλασης της κεντρικής μνήμης της GPU από τα threads ενός warp: (α′) Προσπέλαση συνεχόμενων θέσεων της κεντρικής μνήμης που ανήκουν στο ίδιο 128-Byte τμήμα αυτής από τα threads των half–warps. (β′) Τυχαία προσπέλαση των θέσεων ενός 128-Byte τμήματος μνήμης από τα threads ενός half–warps. (γ′) Προσπέλαση συνεχόμενων θέσεων της κεντρικής μνήμης, με την πρώτη από αυτές να μην ταυτίζεται με την αρχή ενός 128-Byte τμήματος μνήμης. Σημειώνεται ότι και στις τρεις περιπτώσεις κάθε thread διαβάζει/αποθηκεύει έναν πραγματικό αριθμό διπλής ακρίβειας (8 Bytes). Στο σχήμα, επίσης, φαίνεται η κεντρική μνήμη της GPU ως τμήματα μνήμης μήκους 32, 64, ή 128 Bytes.

3.8.2 Constant μνήμη

Η constant μνήμη βρίσκεται στον ίδιο χώρο με την κεντρική μνήμη της GPU. Αντίθετα, όμως, με την κεντρική μνήμη είναι cached, δηλαδή η πρόσβαση σε αυτήν επιταχύνεται από την constant cache, και επιτρέπει στα threads που εκτελούνται στη GPU μόνο να διαβάζουν από αυτή. Επομένως, ενδείκνυται η αποθήκευση σταθερών ποσοτήτων στη μνήμη αυτή. Η αποθήκευση των ποσοτήτων αυτών από τη στιγμή που τα threads της GPU δεν μπορούν να γράψουν σε αυτή, γίνεται από τη CPU, αντιγράφοντας δεδομένα από την κεντρική μνήμη του υπολογιστή στην constant μνήμη της GPU. Υπενθυμίζεται ότι η διάσταση της constant μνήμης είναι 64 KByte και εκείνη της constant cache 8 KByte ανεξάρτητα της υπολογιστικής δυνατότητας της GPU.

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

Για να επιτευχθεί η μέγιστη ταχύτητα προσπέλασης της constant μνήμης, πρέπει τα threads του ίδιου warp για GPUs 2.x ή του ίδιου half–warp για GPUs 1.x, να διαβάζουν την ίδια θέση της constant μνήμης. Σε αντίθετη περίπτωση, το σύνολο των αναγνώσεων εντός του warp (ή του halfwarp, ανάλογα την υπολογιστική δυνατότητα της GPU) πραγματοποιείται σειριακά σε ομάδες ανάγνωσης ίδιων θέσεων της constant μνήμης.

Συνοψίζοντας, ο χρόνος ανάγνωσης δεδομένων από την constant μνήμη είναι πολύ μικρός όταν τα ζητούμενα δεδομένα βρίσκονται στην constant cache. Σε διαφορετική περίπτωση, είναι μεγαλύτερος ή ίσος του χρόνου προσπέλασης της κεντρικής μνήμης ανάλογα με το πλήθος των διαφορετικών θέσεων της constant μνήμης που διαβάζουν τα threads του ίδιου warp, ή του half–warp, ανάλογα με την υπολογιστική δυνατότητα της GPU. Περισσότερες θέσεις μνήμης προς ανάγνωση σημαίνει μεγαλύτερο χρόνο ανάγνωσης. Σημειώνεται ότι, αν όλα τα warps (ή half–warps) που εκτελούνται στον ίδιο πολυεπεξεργαστή διαβάζουν την ίδια θέση της constant μνήμης, τότε μόνο το πρώτο warp (ή half–warp) θα διαβάσει την τιμή της ζητούμενης σταθεράς από την constant μνήμη (αργή ανάγνωση) και όλα τα υπόλοιπα από την constant cache (γρήγορη ανάγνωση).

3.8.3 Texture μνήμη

Η texture μνήμη δεν καταλαμβάνει συγκεκριμένο χώρο εσωτερικά της GPU. Αντίθετα, ο προγραμματιστής ορίζει δυναμικά το χώρο αυτής. Πρακτικά, ο προγραμματιστής επιλέγει τμήματα της κεντρικής μνήμης της GPU που έχουν ήδη δεσμευτεί για την αποθήκευση δεδομένων. Τα τμήματα αυτά της κεντρικής μνήμης ονομάζονται textures και ορίζουν την texture μνήμη. Για το λόγο αυτό, δεν δόθηκε συγκεκριμένη διάσταση για την texture μνήμη στις παραγράφους 3.4, 3.5, όπου αναλύονται οι τελευταίες αρχιτεκτονικές των GPUs της NVIDIA, GT200 και Fermi. Σημειώνεται ότι από τη στιγμή που έχει οριστεί ένα τμήμα της κεντρικής μνήμης ως texture, το τμήμα συνεχίζει να είναι προσπελάσιμο ως κομμάτι της κεντρικής μνήμης από τα threads της GPU, και ισχύουν όσα αναλύθηκαν στην παράγραφο 3.8.1.

Η προσπέλαση των textures γίνεται μέσω ορισμένων ειδικών συναρτήσεων, τις texture fetches. Η θέση ενός texture στην κεντρική μνήμη καθορίζεται από μεταβλητές τύπου texture, τις texture references, οι οποίες ορίζονται κατά την επιλογή των τμημάτων της κεντρικής μνήμης που θα αποτελούν την texture μνήμη. Περισσότερες από μία texture references μπορούν να δείχνουν στο ίδιο texture. Επίσης, επιτρέπεται ένα texture να περιέχει κοινές θέσεις μνήμης με ένα άλλο texture.

Τονίζεται ότι οι texture fetches επιτρέπουν μόνο την ανάγνωση των δεδομένων που είναι αποθηκευμένα στα textures. Όπως αναφέρθηκε προηγουμένως όμως, οι θέσεις μνήμης ενός texture μπορούν να προσπελαστούν ως θέσεις της κεντρικής μνήμης. Συνεπώς τα threads που εκτελούνται σε μία GPU μπορούν να ανανεώνουν θέσεις της κεντρικής μνήμης. Αν οι ίδιες θέσεις αποτελούν τμήμα ενός texture, τότε οι ανανεωμένες τιμές μπορούν να διαβαστούν μέσω των texture fetches.

Πλεονεκτήματα της χρήσης της texture μνήμης

Η ανάγνωση δεδομένων από την texture μνήμη επιταχύνεται από την texture cache. Σημειώνεται ότι, όπως συμβαίνει με όλες τις cached μνήμες έτσι και με την texture, όταν ζητείται η ανάγνωση ενός δεδομένου το οποίο δεν βρίσκεται στην cache (στην texture cache στην περίπτωση της texture μνήμης), τότε εκείνο μεταφέρεται από την κεντρική μνήμη στην cache και η ανάγνωση γίνεται από την cache μνήμη. Δηλαδή, η ανάγνωση μίας θέσης της texture μνήμης (δηλαδή μίας θέσης ενός texture) από ένα thread, γίνεται είτε μέσω της texture cache (γρήγορη ανάγνωση, cache hit), είτε από το texture (αργή ανάγνωση, cache miss). Η διάσταση της texture cache εξαρτάται από το μοντέλο της κάρτας γραφικών και είναι 6 με 8 KByte ανά πολυεπεξεργαστή.

Στην ουσία, ο προγραμματιστής ορίζει μία cache μνήμη, την texture cache, που θα μεσολαβεί στην ανάγνωση ενός τμήματος της κεντρικής μνήμης. Το ποιό θα είναι αυτό το τμήμα της κεντρικής μνήμης αποτελεί επιλογή του προγραμματιστή. Σημειώνεται ότι στις GPUs 2.x η κεντρική μνήμη είναι cached. Συνεπώς, η αυθαίρετη επιλογή των τμημάτων της κεντρικής μνήμης που θα σχηματίσουν τα textures μπορεί να προκαλέσει ελάττωση (αντί της επιδιωκόμενης αύξησης) της παράλληλης απόδοσης ενός GPU-κώδικα. Αντίθετα, η προσεκτική επιλογή, μπορεί να αυξήσει κατακόρυφα την παράλληλη απόδοση ενός GPU-κώδικα. Προτείνεται η χρήση της texture μνήμης για την ανάγνωση δεδομένων που δεν ανανεώνονται συχνά κατά τη διάρκεια της αριθμητικής επίλυσης ενός προβλήματος και που η προσπέλαση αυτών στην κεντρική ή την constant μνήμη δεν τηρεί τα αντίστοιχα βέλτιστα πρότυπα πρόσβασης.

Επιπλέον λειτουργίες της texture μνήμης

Όπως αναφέρθηκε προηγουμένως η προσπέλαση ενός texture γίνεται με τις texture fecthes. Οι texture fetches δέχονται ως ορίσματα ένα texture reference που καθορίζει ποιο texture θα προσπελαστεί και την τοπική θέση (texture coordinate) της προς ανάγνωση θέσης μνήμης. Τα δεδομένα που είναι αποθηκευμένα σε ένα texture ονομάζονται texture elements, ή απλά texels.

Το περιβάλλον της CUDA δίνει τη δυνατότητα χρήσης πραγματικών αριθμών για τον ορισμό της θέσης εσωτερικά ενός texture. Σε μία τέτοια περίπτωση η τιμή που επιστρέφει μία texture fetch προκύπτει από το γραμμικό συνδυασμό των texels των οποίων οι θέσεις μνήμης βρίσκονται πλησιέστερα στη ζητούμενη θέση.

Αν ένα texture περιέχει N texels, τότε οι texture coordinates κινούνται στο διάστημα $[0,\ N-1]$. Ο προγραμματιστής μπορεί να ορίσει κατάλληλα το texture, ώστε

το διάστημα των texture coordinates να είναι το [0, 1]. Προφανώς, αυτό δεν θα μπορούσε να συμβεί εάν το περιβάλλον της CUDA δεν υποστήριζε τη χρήση πραγματικών αριθμών για texture coordinates.

Οι texture fetches δέχονται, εξ ορισμού, texture coordinates εντός του διαστήματος $[0, N-1]$, ή [0, 1] αν χρησιμοποιείται η λειτουργία που περιγράφηκε προηγουμένως. Για αυτό, στην κλήση μίας texture fetch γίνεται πάντα ο έλεγχος αν η ζητούμενη συντεταγμένη είναι εντός του επιτρεπτού διαστήματος. Στην περίπτωση που δεν είναι, η texture fetch επιστρέφει την τιμή του πλησιέστερου οριακού texel. Δηλαδή εάν χρησιμοποιούνται texture coordinates στο διάστημα [0, 1] και ο προγραμματιστής ζητήσει την τιμή με συντεταγμένη -0.1, η texture fetch θα επιστρέψει την τιμή του πρώτου texel στο texture. Αντίστοιχα, αν ζητηθεί η τιμή με συντεταγμένη μεγαλύτερη του 1.0 θα επιστραφεί η τιμή του τελευταίου texel του texture. Αυτή η λειτουργία μπορεί, από τη μία, να εξασφαλίζει τη σωστή λειτουργία των texture fetches αλλά, από την άλλη, μπορεί και να κρύψει πιθανό προγραμματιστικό λάθος. Οπότε, ο προγραμματιστής πρέπει να είναι προσεκτικός όταν χρησιμοποιεί την texture μνήμη.

Σε συνέχεια όσων αναφέρθηκαν στην προηγούμενη παράγραφο και μόνο για την περίπτωση που οι texture fetches δέχονται κανονικοποιημένες texture coordinates, δηλαδή στο διάστημα [0, 1], ο προγραμματιστής μπορεί να ορίσει η ζητούμενη texture coordinate να προβάλλεται εντός του επιτρεπτού διαστήματος [0, 1]. Δηλαδή, αν ζητηθεί η τιμή με texture coordinate 1.25, που βρίσκεται εκτός του διαστήματος [0, 1], επιστρέφεται η τιμή με συντεταγμένη 0.25, αντί εκείνης του τελευταίου texel του texture, κ.ο.κ. Προτείνεται η χρήση αυτής της λειτουργίας, όταν τα δεδομένα που έχουν αποθηκευτεί σε ένα texture αναφέρονται σε μία περίοδο ενός περιοδικά μεταβαλλόμενου φαινόμενου, λ.χ. αποτελούν διαδοχικές μετρήσεις εντός της περιόδου ενός περιοδικού φαινομένου.

3.8.4 Shared μνήμη

Η shared μνήμη είναι μνήμη ταχείας προσπέλασης και επιτρέπει την επικοινωνία μεταξύ των threads του ίδιου block. Η διάσταση της shared μνήμης είναι 16 KByte ανά πολυεπεξεργαστή σε GPUs 1.x. Στις GPUs 2.x ο προγραμματιστής επιλέγει ανάμεσα σε 48 KByte shared μνήμης και 16 KByte L1 cache ανά πολυεπεξεργαστή, ή αντίστροφα.

Σε GPUs 1.x η προσπέλαση της shared μνήμης γίνεται από τα threads του ίδιου half–warp. Η μέγιστη ταχύτητα προσπέλασης της shared μνήμης επιτυγχάνεται είτε όταν τα threads του ίδιου half–warp χειρίζονται διαφορετικές θέσεις αυτής, είτε όταν όλα τα threads του half–warp διαβάζουν την ίδια θέση μνήμης. Σε περίπτωση που δύο ή περισσότερα threads (όχι όμως το σύνολο αυτών) διαβάζουν από την ίδια θέση της shared μνήμης τότε το σύνολο των αναγνώσεων σειριοποιείται σε ομάδες αναγνώσεων διαφορετικών θέσεων μνήμης. Στην περίπτωση που ικανοποιείται ο παραπάνω περιορισμός η ταχύτητα προσπέλασης της shared μνήμης είναι περίπου ίδια με εκείνη μίας cache μνήμης.

Αντίθετα, σε GPUs 2.x, η προσπέλαση της shared μνήμης γίνεται από όλο το warp. Επιπλέον, δύο ή περισσότερα threads μπορούν να διαβάζουν από την ίδια θέση

της shared μνήμης χωρίς να επηρεάζεται η ταχύτητα προσπέλασης της shared μνήμης.

3.8.5 Τοπική μνήμη (local memory)

Σε κάθε thread αντιστοιχεί ένα τμήμα της κεντρικής μνήμης, η λεγόμενη τοπική μνήμη (local memory). Ως τμήμα της κεντρικής μνήμης, η τοπική χαρακτηρίζεται από υψηλούς χρόνους προσπέλασης, ενώ χρησιμοποιείται κυρίως από την ίδια την κάρτα γραφικών όταν οι απαιτήσεις ενός kernel σε registers υπερβαίνουν τους διαθέσιμους ανά πολυεπεξεργαστή. Είναι δομημένη με τέτοιο τρόπο, ώστε σε περίπτωση χρήσης της από τα threads του ίδιου warp (ή halfwarp, για GPUs 1.x), εκείνα να διαβάζουν/αποθηκεύουν σε θέσεις του ίδιου τμήματος μνήμης, όπως δηλαδή υπαγορεύει το βέλτιστο πρότυπο προσπέλασης της κεντρικής μνήμης. Έτσι επιτυγχάνεται ο ελάχιστος δυνατός χρόνος προσπέλασης. Στις GPUs 2.x η πρόσβαση στην τοπική μνήμη επιταχύνεται από τις L1, L2 cache. Η διάσταση της τοπικής ανά thread μνήμης είναι 16 KBytes ή 512 KBytes για GPUs 1.x ή 2.x, αντίστοιχα.

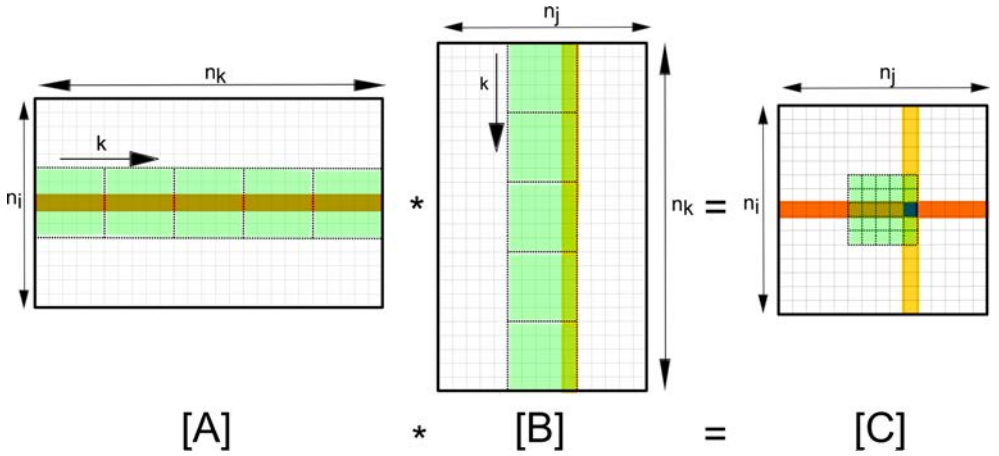
3.9 Αύξηση της παράλληλης απόδοσης του παραδείγματος της παραγράφου 3.7

Στην παράγραφο 3.7 παρουσιάστηκε ένα απλό παράδειγμα παράλληλου υπολογισμού του γινομένου δύο πινάκων $\left([C]_{n_i \times n_j} = [A]_{n_i \times n_k} \cdot [B]_{n_k \times n_j}\right)$ στο περιβάλλον της CUDA. Στην παράγραφο αυτή ξαναγράφεται το kernel υπολογισμού του πίνακα C με σκοπό την αύξηση της παράλληλης απόδοσης αυτού. Για να επιτευχθεί ο στόχος αυτός χρησιμοποιείται η γρήγορη shared μνήμη και μειώνεται το πλήθος των προσβάσεων των threads της GPU στην αργή κεντρική μνήμη. Χρησιμοποιείται επίσης η constant μνήμη. Η χρήση της shared μνήμης όμως αντί της κεντρικής είναι η βασική αιτία της μεγάλης αύξησης της απόδοσης του kernel.

Περιγραφή του kernel υπολογισμού του πίνακα C

Ο ψευδοκώδικας 6 παρουσιάζει το καινούργιο kernel υπολογισμού του πίνακα C. Η σχηματική απεικόνιση της λειτουργίας του kernel φαίνεται στο σχήμα 3.6. Σύμφωνα με το kernel αυτό κάθε thread υπολογίζει ένα στοιχείο του πίνακα C. Δηλαδή, συνολικά καλούνται $n_i \times n_j$ threads που ομαδοποιούνται σε blocks των 16×16 threads, τα οποία σχηματίζουν ένα ${}^{n_i}/_{16} \times {}^{n_j}/_{16}$ grid από blocks. Όπως δηλαδή συμβαίνει και στο παράδειγμα της παραγράφου 3.7. Υπενθυμίζεται ότι οι διαστάσεις των πινάκων n_i, n_k, n_j έχουν θεωρηθεί πολλαπλάσια του 16, ώστε να διευκολυνθεί η παρουσίαση του παραδείγματος.

Σύμφωνα με το kernel της παραγράφου 3.7 κάθε thread διαβάζει τα στοιχεία της γραμμής i του πίνακα A και εκείνα της στήλης j του B για τον υπολογισμό του στοιχείου (i, j) του C. Αυτό σημαίνει ότι τα threads που ανήκουν στο ίδιο block και υπολογίζουν αντίστοιχα τα στοιχεία (i, j) και $(i + 1, j)$ του πίνακα C διαβάζουν την ίδια στήλη του



Σχήμα 3.6: Σχηματική αναπαράσταση του υπολογισμού του γινομένου δύο πινάκων $\left([C]_{n_i \times n_j} = [A]_{n_i \times n_k} \cdot [B]_{n_k \times n_j}\right)$ στη GPU. Κάθε thread υπολογίζει την τιμή ενός στοιχείου του πίνακα C. Για το σκοπό αυτό εκτελεί $N_K = n_k/blockDim.y$ σαρώσεις. Σε μία τυχαία επανάληψη k τα threads του ίδιου block αντιγράφουν τα τονισμένα τμήματα των πινάκων A, B στη shared μνήμη, συγχρονίζονται, ώστε να αποφευχθεί ταυτόχρονη ανάγνωση και ανανέωση της ίδιας θέσης της shared μνήμης από threads του ίδιου block και κάθε thread πραγματοποιεί τον πολλαπλασιασμό της $threadIdx.x$ γραμμής του τμήματος του A με την $threadIdx.y$ στήλη του τμήματος του B. [$threadIdx.x$, $threadIdx.y$] είναι οι τοπικές συντεταγμένες του thread στο block. Στο τέλος των σαρώσεων, οι υπολογισμένες από κάθε thread τιμές των στοιχείων του C αποθηκεύονται στις αντίστοιχες θέσεις της κεντρικής μνήμης της GPU.

πίνακα B από την κεντρική μνήμη. Όμοια, τα threads που υπολογίζουν τα (i, j) και $(i, j+1)$ στοιχεία του C διαβάζουν την ίδια γραμμή του A. Το kernel που περιγράφεται στην παράγραφο αυτή χρησιμοποιεί τη shared μνήμη, ώστε τα στοιχεία των γραμμών i ή των στηλών j των πινάκων A, B, αντίστοιχα, να διαβάζονται μία φορά για όλα τα threads του ίδιου block.

Συγκεκριμένα, το kernel εκτελεί $n_k/16$ σαρώσεις φορτώνοντας σε κάθε σάρωση ένα τμήμα των πινάκων A, B στη shared μνήμη του block. Στο σχήμα 3.6 k είναι ο μετρητής αυτών των σαρώσεων, ενώ τα τονισμένα τμήματα των πινάκων A, B είναι εκείνα που αντιγράφονται στη shared μνήμη σε κάθε σάρωση k. Για να αποφευχθεί η ανάγνωση της τιμής μίας θέσης της shared μνήμης από κάποιο thread με την ταυτόχρονη ανανέωση εκείνης από ένα άλλο του ίδιου block, τα threads του ίδιου block συγχρονίζονται αμέσως μετά τη μεταφορά των δεδομένων στη shared μνήμη. Μετά το συγχρονισμό, τα threads του block πολλαπλασιάζουν τα στοιχεία της $threadIdx.x$ γραμμής του τμήματος του πίνακα A με εκείνα της $threadIdx.y$ στήλης του τμήματος του πίνακα B. [$threadIdx.x$, $threadIdx.y$] είναι οι τοπικές συντεταγμένες του κάθε thread στο block. Με το πέρας των σαρώσεων, κάθε thread ανανεώνει την αντίστοιχη θέση του πίνακα C. Η χρήση της shared μνήμης μειώνει τον αριθμό των προσβάσεων στην κεντρική μνήμη και αυξάνει περίπου 4 φορές την παράλληλη απόδοση του kernel.

Τονίζεται ότι, κατά τη διάρκεια των N_K σαρώσεων, ανανεώνεται η τιμή της τοπικής μεταβλητής cij. Μόνο στο τέλος των σαρώσεων κάθε thread αποθηκεύει την τιμή που

υπολόγισε στην αντίστοιχη θέση της κεντρικής μνήμης.

Η μεταβλητή $\_NK$ που εμφανίζεται στον ψευδοκώδικα 6 βρίσκεται στην constant μνήμη και ισούται με $^{n_k}/_{16}$. Η τιμή αυτή αντιγράφεται από τη CPU στην constant μνήμη της GPU. Η αντιγραφή αυτή είναι, στην ουσία, η μόνη προσθήκη στην κύρια συνάρτηση που παρουσιάστηκε στην παράγραφο 3.7. Ο ψευδοκώδικας 5 παρουσιάζει τη νέα κύρια συνάρτηση. Η αντιγραφή του πηλίκου $^{n_k}/_{16}$ στην constant μνήμη είναι τονισμένη με έντονα γράμματα.

Από τη στιγμή που μόνο η $\_NK$ είναι αποθηκευμένη στην constant μνήμη, τα threads του ίδιου warp δεν μπορούν παρά να έχουν πρόσβαση στην ίδια θέση της constant μνήμης. Επομένως, τα πρώτα warps που εκτελούνται στους πολυεπεξεργαστές διαβάζουν τη σταθερά $\_NK$ με τη μέγιστη δυνατή ταχύτητα ανάγνωσης από την constant μνήμη. Τα υπόλοιπα warps που εκτελούνται στον ίδιο πολυεπεξεργαστή διαβάζουν την τιμή της σταθεράς $\_NK$ από την constant cache.

Τέλος, threads που ανήκουν σε διαφορετικό block μπορεί να διαβάζουν την ίδια γραμμή του πίνακα A, ή στήλη του B. Όπως για παράδειγμα συμβαίνει για τα threads που υπολογίζουν τις θέσεις (i, j), $(i + 16, j)$, ή τις θέσεις (i, j), $(i, j + 16)$ του πίνακα C. Υπενθυμίζεται ότι η διάσταση του block ανά κατεύθυνση είναι ίση με 16. Για το λόγο αυτό, ιδίως σε GPUs 1.x όπου η προσπέλαση της κεντρικής μνήμης δεν είναι cached, προτείνεται η χρήση της texture μνήμης για την ανάγνωση των στοιχείων των πινάκων A, B. Στην περίπτωση αυτή, πρακτικά, η ανάγνωση ενός στοιχείου του πίνακα A, ή B από διαφορετικά threads θα κόστιζε όσο μία μόλις ανάγνωση από την κεντρική μνήμη, αφού οι υπόλοιπες (εκτός της πρώτης) θα γίνονταν από την texture cache. Στο κεφάλαιο 4 παρουσιάζονται μέθοδοι προγραμματισμού υψηλής απόδοσης κώδικα σε GPUs, δίνοντας έμφαση στην επίλυση αεροδυναμικών προβλημάτων.

Ψευδοκώδικας 5 Κύρια συνάρτηση

```
1: #define nbx  16
2: #define nby  16
3:
4: function main
5:     // Ανάγνωση των διαστάσεων των πινάκων A, B
6:     read(ni, nj, nk)
7:
8:     // Δέσμευση θέσεων της κεντρικής μνήμης του υπολογιστή
9:     A  ← allocateMem_on_CPU(ni, nk)
10:    B  ← allocateMem_on_CPU(nk, nj)
11:    C  ← allocateMem_on_CPU(ni, nj)
12:
13:    // Δέσμευση θέσεων της κεντρικής μνήμης της GPU
14:    _A ← allocateMem_on_GPU(ni, nk)
15:    _B ← allocateMem_on_GPU(nk, nj)
16:    _C ← allocateMem_on_GPU(ni, nj)
17:
18:    // Ανάγνωση (από τη CPU) των πινάκων A, B από αρχείο
19:    readMatrices(A, B)
20:
21:    // Αντιγραφή δεδομένων στη GPU
22:    sizeA ← ni * nk * sizeof(double)
23:    sizeB ← nk * nj * sizeof(double)
24:
25:    memory_copy(_A, A, sizeA, CPU_to_GPU)
26:    memory_copy(_B, B, sizeB, CPU_to_GPU)
27:
28:    // Δέσμευση θέσεων της constant μνήμης της GPU
29:      NK   ←   nk/nbx
30:      _NK  ←   allocateMem_on_ConstantMem
31:      copy_data_to_ConstantMemory(_NK,NK)
32:
33:    // Πολλαπλασιασμός των πινάκων A, B της GPU
34:    blockSize ← (nbx     , nby    )
35:    gridSize  ← (ni/nbx, nj/nby)
36:    matmulGPU <<< gridSize, blockSize >>>(_A, _B, _C)
37:
38:    // Αντιγραφή αποτελεσμάτων στη CPU
39:    sizeC ← ni * nj * sizeof(double)
40:    memory_copy(C, _C, sizeC, GPU_to_CPU)
41:
42:    // Αποθήκευση πίνακα C σε αρχείο εξόδου
43:    save(C)
```

```
44:     // Αποδέσμευση θέσεων της κεντρικής μνήμης του υπολογιστή
45:     deallocateMem_on_CPU( A)
46:     deallocateMem_on_CPU( B)
47:     deallocateMem_on_CPU( C)
48:
49:     // Αποδέσμευση θέσεων της κεντρικής μνήμης της GPU
50:     deallocateMem_on_GPU(_A)
51:     deallocateMem_on_GPU(_B)
52:     deallocateMem_on_GPU(_C)
53: end function
```

Ψευδοκώδικας 6 kernel υπολογισμού του γινομένου των A, B

```
1:  function matmulGPU(A, B, C)
2:      // Δέσμευση θέσεων της shared μνήμης
3:      Aloc ← allocateMem_on_SharedMem(blockDim.x, blockDim.y)
4:      Bloc ← allocateMem_on_SharedMem(blockDim.y, blockDim.x)
5:
6:      // Συσχέτιση των threads με στοιχεία του πίνακα C
7:      igl ← blockIdx.x * blockDim.x + threadIdx.x
8:      jgl ← blockIdx.y * blockDim.y + threadIdx.y
9:
10:     // Υπολογισμός της θέσης C(i, j)
11:     cij ← 0.
12:     for k ← 0, _NK do
13:         k1 ← threadIdx.x + k * _NK
14:         k2 ← threadIdx.y + k * _NK
15:
16:         Aloc(threadIdx.x, threadIdx.y) ← A(igl, k1 )
17:         Bloc(threadIdx.x, threadIdx.y) ← B(k2 , jgl)
18:         synchronize_threads_of_the_same_block
19:
20:         for m ← 0, blockDim.y do
21:             cij ← cij + Aloc(threadIdx.x, m) * Bloc(m, threadIdx.y)
22:         end for
23:     end for
24:
25:     C(i, j) ← cij
26: end function
```

3.10 Εξωτερικές ως προς τη GPU προσβάσιμες μνήμες από τα threads

Στην ενότητα 3.8 παρουσιάστηκαν οι μνήμες εσωτερικά μίας GPU. Τα threads μίας GPU όμως έχουν τη δυνατότητα της απευθείας πρόσβασης και στην κεντρική μνήμη του υπολογιστή και στην κεντρική μνήμη κάθε άλλης GPU του ίδιου υπολογιστικού κόμβου. Προφανώς, η πρόσβαση στις μνήμες εσωτερικά της GPU είναι πολύ γρηγορότερη σε σχέση με εκείνη σε μνήμες εξωτερικά της GPU. Για το λόγο αυτό, αποφεύχθηκε η απευθείας πρόσβαση από τα threads της GPU στις θέσεις A, B, C της κεντρικής μνήμης του υπολογιστή στα παραδείγματα 3.7 και 3.9. Συνήθως, ζητείται από τα threads μίας GPU η προσπέλαση εξωτερικών της GPU μνημών, όταν χρειάζεται η συνεργασία CPU-GPU (όπως αναλύεται στην παράγραφο 3.11) ή η επικοινωνία μεταξύ των threads διαφορετικών GPUs. Η χρήση συστοιχίας από GPUs αποτελεί αντικείμενο του κεφαλαίου 5. Οι παράγραφοι που ακολουθούν αναφέρονται στην πρόσβαση των threads μιας GPU σε μνήμες εξωτερικά αυτής.

3.10.1 Προσπέλαση της κεντρικής μνήμης του υπολογιστή

Το περιβάλλον προγραμματισμού της CUDA επιτρέπει τη δέσμευση θέσεων στην κεντρική μνήμη του υπολογιστή, ορίζοντας ταυτόχρονα τις θέσεις αυτές προσβάσιμες από τα threads μίας GPU (page-locked, ή pinned θέσεις μνήμης). Προφανώς, η πρόσβαση από τα threads σε pinned θέσεις μνήμης είναι αρκετά πιο αργή σε σχέση με την πρόσβαση θέσεων μνημών εσωτερικά της GPU. Η ταχύτητα προσπέλασης pinned θέσεων μνήμης εξαρτάται από την ταχύτητα μεταφοράς δεδομένων του διαύλου (PCIe) που είναι τοποθετημένη η GPU. Ενδεικτικά αναφέρεται ότι η μέγιστη ταχύτητα προσπέλασης της κεντρικής μνήμης μία Tesla M2050 είναι 148 GByte/sec. Συγκριτικά, η μέγιστη ταχύτητα μεταφοράς δεδομένων μέσω ενός διαύλου PCIe x16 Gen2 είναι μόλις 8 GByte/sec. Το βέλτιστο πρότυπο προσπέλασης pinned θέσεων μνήμης είναι ίδιο με εκείνο της κεντρικής μνήμης της GPU.

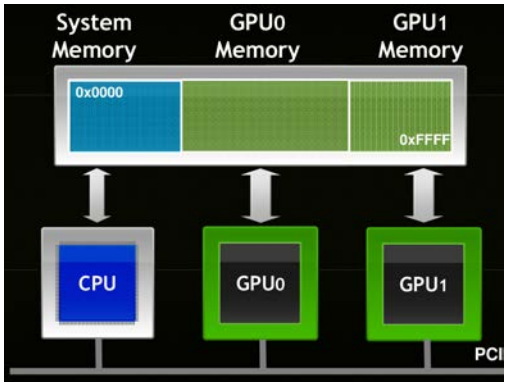
3.10.2 Προσπέλαση της κεντρικής μνήμης διαφορετικής GPU του ίδιου κόμβου

Η 4η έκδοση (τελευταία έκδοση τη στιγμή που γράφεται η διατριβή αυτή) του περιβάλλοντος προγραμματισμού της CUDA θεωρεί μία ενιαία εικονική μνήμη (Unified Virtual Addressing, UVA) στην οποία συνυπάρχουν η κεντρική μνήμη του υπολογιστή και οι κεντρικές μνήμες των GPUs που ανήκουν στον ίδιο υπολογιστικό κόμβο (σχήμα 5.1). Οι θέσεις στη μεγάλη αυτή εικονική μνήμη είναι συνεχόμενες, επομένως ο κώδικας

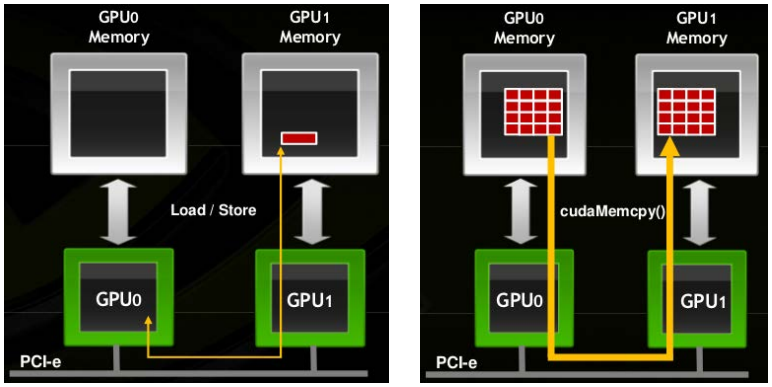
καταλαβαίνει από τη δεκαεξαδική τιμή της θέσης σε ποια μνήμη (ποια πλατφόρμα) αυτή πραγματικά ανήκει.

Επιπλέον, η 4η έκδοση του περιβάλλοντος προγραμματισμού της CUDA επιτρέπει στα threads που εκτελούνται σε μία GPU 2.x να έχουν πρόσβαση σε θέσεις της κεντρικής μνήμης μίας άλλης GPU, επίσης υπολογιστικής δυνατότητας 2.x, του ίδιου υπολογιστικού κόμβου (peer-to-peer memory access), χωρίς τη διαμεσολάβηση της CPU (σχήμα 3.8(α΄)). Σε προηγούμενες εκδόσεις της CUDA, η ανταλλαγή δεδομένων μεταξύ δύο GPUs του ίδιου υπολογιστικού κόμβου γινόταν υποχρεωτικά μέσω της κεντρικής μνήμης του υπολογιστή. Δηλαδή, αν οι GPU0 και GPU1 ανήκουν στον ίδιο υπολογιστικό κόμβο, τότε για να διαβάσουν τα threads της GPU0 δεδομένα της κεντρικής μνήμης της GPU1, θα έπρεπε προηγουμένως τα δεδομένα αυτά να έχουν αντιγραφεί σε pinned θέσεις μνήμης. Αυτό παρακάμπτεται με τη χρήση της 4ης έκδοσης της CUDA. Τονίζεται η σημασία της εικονικής κοινής μνήμης ανάμεσα σε CPU και GPUs του ίδιου υπολογιστικού κόμβου καθώς, για παράδειγμα, τα threads της GPU0 γνωρίζουν τη μνήμη (άρα και την πλατφόρμα) από την οποία θα διαβάσουν ή θα αποθηκεύσουν μία σειρά δεδομένων. Η L2 cache της GPU0 μεσολαβεί κατά την προσπέλαση της κεντρικής μνήμης της GPU1 από τα threads της GPU0. Προφανώς, η ταχύτητα προσπέλασης θέσεων της κεντρικής μνήμης μίας άλλης GPU του ίδιου υπολογιστικού κόμβου εξαρτάται από την ταχύτητα μεταφοράς δεδομένων των διαύλων (PCIe) που είναι τοποθετημένες οι δύο GPUs.

Ακόμα, επιτρέπεται η απευθείας αντιγραφή δεδομένων από την κεντρική μνήμη μίας GPU στην κεντρική μνήμη μίας άλλης GPU του ίδιου υπολογιστικού κόμβου (Peer-to-Peer memory copy, σχήμα 3.8(β΄)). Για περισσότερα σχετικά με την ταυτόχρονη χρήση πολλών GPUs ο αναγνώστης παραπέμπεται στο κεφάλαιο 5.



Σχήμα 3.7: Εικονική κοινή μνήμη (Unified Virtual Addressing (UVA)), [128]. Η 4η έκδοση του περιβάλλοντος προγραμματισμού της CUDA θεωρεί μία ενιαία εικονική μνήμη στην οποία συνυπάρχουν η κεντρική μνήμη του υπολογιστή και οι κεντρικές μνήμες των GPUs που ανήκουν στον ίδιο υπολογιστικό κόμβο. Οι θέσεις σε αυτήν τη μεγάλη εικονική μνήμη είναι συνεχόμενες, επομένως ο κώδικας καταλαβαίνει από τη δεκαεξαδική τιμή της θέσης σε ποια μνήμη αυτή πραγματικά ανήκει.



(α′) (β′)

Σχήμα 3.8: Η 4η έκδοση του περιβάλλοντος προγραμματισμού της CUDA επιτρέπει σε GPUs 2.x (α′) την απευθείας προσπέλαση της κεντρικής μνήμης της GPU1 από threads που εκτελούνται στη GPU0 (Peer-to-Peer memory access), και (β′) την απευθείας μεταφορά δεδομένων από την κεντρική μνήμη της GPU0 στην κεντρική μνήμη GPU1 (Peer-to-Peer memory copy), χωρίς την διαμεσολάβηση της CPU, [128]. Οι GPU0, GPU1 ανήκουν στον ίδιο υπολογιστικό κόμβο.

3.11 Συνεργασία CPU-GPU

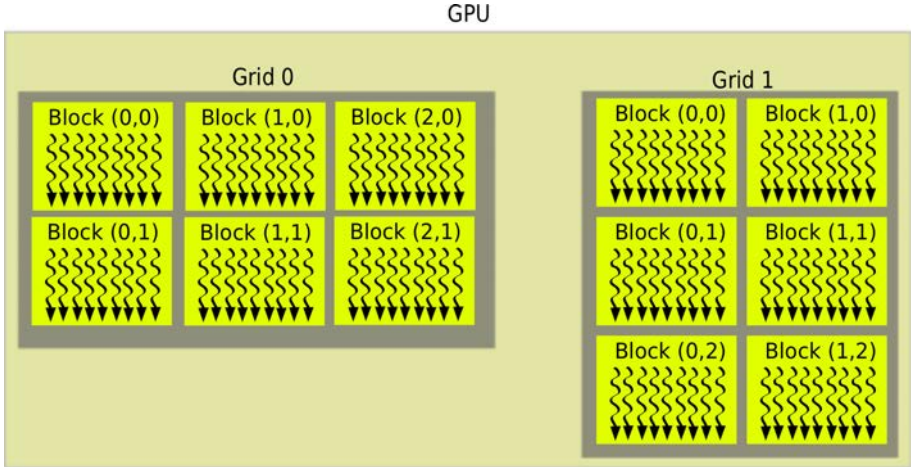
Έχει ήδη αναφερθεί ότι η κλήση ενός kernel από τη CPU είναι ασύγχρονη. Δηλαδή, ο έλεγχος επιστρέφει στη CPU πριν την ολοκλήρωση της εκτέλεσης του kernel. Αυτό πρακτικά σημαίνει ότι η CPU μπορεί να εκτελεί ένα τμήμα ενός κώδικα χαμηλού ή ακόμα και μηδενικού βαθμού παραλληλησιμότητας, επεξεργαζόμενη δεδομένα που βρίσκονται στην κεντρική μνήμη του υπολογιστή, ταυτόχρονα με την εκτέλεση ενός kernel στη GPU. Έτσι επιτυγχάνεται η αξιοποίηση όλων των διαθέσιμων υπολογιστικών πόρων και ως εκ τούτου, αυξάνεται η παράλληλη απόδοση του κώδικα.

Η παράγραφος 4.4.6 παρουσιάζει ένα παράδειγμα συνεργασίας της GPU με τη CPU. Ο κώδικας που περιγράφεται αποτελεί τμήμα του GPU-κώδικα επίλυσης των εξισώσεων Navier–Stokes που αναπτύχθηκε στο πλαίσιο της διατριβής και παρουσιάζεται στο κεφάλαιο 4. Το τμήμα αυτό του GPU-κώδικα υπολογίζει το υπόλοιπο των διακριτοποιημένων Navier–Stokes εξισώσεων σε ένα ψευδοχρονικό βήμα, με σκοπό τον έλεγχο της σύγκλισης της επαναληπτικής διαδικασίας της επίλυσης. Τα threads της GPU χειρίζονται από έναν κόμβο του πλέγματος[1] και η GPU χρησιμοποιώντας την shared μνήμη υπολογίζει τα ανά block (επιμέρους) αθροίσματα των ανά κόμβο του πλέγματος υπολοίπων μίας εξίσωσης. Η τελική άθροιση των επιμέρους αθροισμάτων γίνεται στη CPU. Τα παραπάνω δύο βήματα επαναλαμβάνονται για κάθε εξίσωση που επιλύεται. Ο υπολογισμός των επιμέρους αθροισμάτων από τη GPU για τη δεύτερη εξίσωση γίνεται ταυτόχρονα με την άθροιση αυτών της πρώτης εξίσωσης από τη CPU κ.ο.κ. Ο έλεγχος της σύγκλισης γίνεται από τη CPU.

[1] Ο όρος 'πλέγμα' αναφέρεται στο πλέγμα διακριτοποίησης του υπολογιστικού χωρίου και όχι στο grid.

3.12 Ασύγχρονη επεξεργασία δεδομένων σε μία GPU

Το περιβάλλον της CUDA επιτρέπει την ομαδοποίηση των εργασιών που εκτελεί μία GPU σε streams. Στην ουσία, κάθε stream αποτελείται από μία σειρά από εντολές προς τη GPU, οι οποίες εκτελούνται η μία μετά την άλλη. Εφόσον, εντολές που ανήκουν σε διαφορετικά streams εκτελούνται ασύγχρονα, είναι δυνατή η ταυτόχρονη αντιγραφή δεδομένων από την κεντρική μνήμη του υπολογιστή σε εκείνη της GPU (ή αντίστροφα), με την εκτέλεση ενός kernel στη GPU ή η εκτέλεση δύο ή περισσότερων kernels στην ίδια GPU ταυτόχρονα. Για να γίνει αυτό πρέπει οι προς εκτέλεση από τη GPU εντολές να ανήκουν σε διαφορετικά streams. Σημειώνεται ότι για την ανάθεση της αντιγραφής δεδομένων από την κεντρική μνήμη του υπολογιστή σε εκείνη της GPU, ή αντίστροφα, πρέπει οι θέσεις της κεντρικής μνήμης του υπολογιστή να είναι pinned (παράγραφος 3.10.1). Επιπλέον, μόνο ορισμένες GPUs 2.x επιτρέπουν την ταυτόχρονη εκτέλεση διαφορετικών kernels (επιτρέπεται η ταυτόχρονη εκτέλεση έως και 16 kernels). Στις περιπτώσεις αυτές, κάθε kernel εκτελείται σε διαφορετικό grid από threads. Στο σχήμα 3.9 φαίνεται ο σχηματισμός δύο grids (Grid0, Grid1) στα οποία εκτελούνται παράλληλα δύο ανεξάρτητα kernels στην ίδια κάρτα γραφικών. Ο αριθμός των threads ανά block και εκείνος των blocks στο grid μπορεί να διαφέρουν ανάμεσα στα grids. Προφανώς, απαραίτητη προϋπόθεση για την παράλληλη εκτέλεση περισσότερων του ενός kernels στην ίδια κάρτα γραφικών είναι να επαρκούν οι διαθέσιμοι πολυεπεξεργαστές της κάρτας.



Σχήμα 3.9: Ταυτόχρονη εκτέλεση δύο kernels σε μία GPU. Κάθε kernel έχει τη δική του κατανομή threads στα blocks και των blocks στο grid. Μόνο ορισμένες GPUs 2.x υποστηρίζουν την ταυτόχρονη εκτέλεση δύο ή περισσότερων kernels. Βασική προϋπόθεση είναι να επαρκούν οι πολυεπεξεργαστές της κάρτας γραφικών.

3.13 Atomic functions

Οι σημερινές GPUs αποτελούν μονάδες παράλληλης επεξεργασίας κοινής μνήμης. Η χρήση κοινής μνήμης από τα threads της GPU πρέπει να λαμβάνεται σοβαρά υπόψη από τον προγραμματιστή, ώστε εκείνος να αποτρέπει την ταυτόχρονη ανάθεση τιμών από διαφορετικά, ταυτόχρονα εκτελούμενα, threads στην ίδια θέση μνήμης. Στην περίπτωση που ο προγραμματιστής δεν μπορεί να εγγυηθεί το παραπάνω, το περιβάλλον προγραμματισμού της CUDA περιέχει μία σειρά από ειδικές συναρτήσεις (atomic functions) που εξασφαλίζουν την ανανέωση μίας θέσης της κεντρικής ή της shared μνήμης από ένα thread, αποτρέποντας την ταυτόχρονη ανανέωση της ίδιας θέσης μνήμης από ένα άλλο thread που πιθανώς εκτελείται την ίδια στιγμή. Οι συναρτήσεις αυτές μπορούν να χειρίζονται ακέραιους, πραγματικούς αριθμούς απλής ακρίβειας και μόλις στην 4η έκδοση της CUDA και πραγματικούς αριθμούς διπλής ακρίβειας. Φυσικά, η χρήση των συναρτήσεων αυτών πρέπει να γίνεται με σύνεση, καθώς συγχρονίζουν τα threads που εκτελούνται ταυτόχρονα, μειώνοντας, ως εκ τούτου, την απόδοση του GPU-κώδικα.

3.14 Ρυθμιστικοί παράμετροι της παράλληλης απόδοσης

Η ενότητα αυτή παρουσιάζει ορισμένες παραμέτρους, η ρύθμιση των οποίων μπορεί να αυξήσει την παράλληλη απόδοση ενός kernel χωρίς να απαιτείται οποιαδήποτε επέμβαση σε αυτό. Μία τέτοια παράμετρος είναι ο αριθμός των threads ανά block, η επιλογή του οποίου περιγράφεται στην παράγραφο 3.14.1. Επιπλέον, οι GPUs 2.x δίνουν τη δυνατότητα ρύθμισης ορισμένων επιπλέον παραμέτρων οι οποίες παρουσιάζονται στην 3.14.2.

3.14.1 Κατανομή των threads στα blocks

Ο αριθμός των threads ανά block είναι μία πολύ σημαντική παράμετρος, καθώς ορίζει τον αριθμό των warps που εκτελούνται σε έναν πολυεπεξεργαστή. Η επιλογή του αριθμού αυτού γίνεται σύμφωνα με τις απαιτήσεις του kernel σε shared μνήμη και registers ανά πολυεπεξεργαστή. Προφανώς, προτείνεται ο διαμερισμός των threads σε blocks να είναι πολλαπλάσιο του 32, ώστε να μην σχηματίζονται warps με λιγότερα από 32 threads.

Κάθε πολυεπεξεργαστής μπορεί να επεξεργαστεί έως και 8 blocks. Το πλήθος των warps που χειρίζεται ένας πολυεπεξεργαστής εξαρτάται από την υπολογιστική δυνατότητα της GPU. Συγκεκριμένα, κάθε πολυεπεξεργαστής μίας GPU 1.0 ή 1.1 μπορεί να χειρίζεται έως και 24 warps. Αντίθετα, GPUs 1.2 ή 1.3 μπορούν να χειρίζονται έως και 32 warps ανά πολυεπεξεργαστή. Οι πολυεπεξεργαστές καρτών γραφικών 2.x χειρίζονται έως και 48 warps. Κάθε block μπορεί να αποτελείται το πολύ από 512 ή 1024 threads σε κάρτες 1.x ή 2.x, αντίστοιχα.

Αν οι απαιτήσεις του block υπερβαίνουν τις δυνατότητες των πολυεπεξεργαστών της κάρτας γραφικών, τότε το kernel δεν μπορεί να εκτελεστεί. Σε αυτές τις περιπτώσεις, ο προγραμματιστής πρέπει είτε να μειώσει τον αριθμό των threads ανά block, ώστε να μειωθούν οι απαιτήσεις σε shared μνήμη ή να ξαναπρογραμματίσει το kernel με στόχο την ελάττωση των απαιτήσεων σε registers. Το τελευταίο επιτυγχάνεται με τη μείωση των μεταβλητών που χρησιμοποιούνται εσωτερικά του kernel.

Παράδειγμα υπολογισμού του αριθμού των threads ανά block

Έστω ένα kernel που χρησιμοποιεί 20 registers ανά thread και μικρό τμήμα της διαθέσιμης shared μνήμης ανά πολυεπεξεργαστή. Δηλαδή, ο μέγιστος αριθμός warps που μπορούν να εκτελεστούν σε έναν πολυεπεξεργαστή περιορίζεται από το διαθέσιμο ανά πολυεπεξεργαστή πλήθος από registers και όχι από την διάσταση της shared μνήμη.

Η εκτέλεση ενός warp χρειάζεται $32{\times}20{=}640$ registers. Έστω ότι χρησιμοποιείται μία GPU αρχιτεκτονικής Fermi, η οποία περιέχει 32768 registers ανά πολυεπεξεργαστή. Οι διαθέσιμοι registers ανά πολυεπεξεργαστή επαρκούν, ώστε να εκτελεστούν έως και $32768/640 \approxeq 51$ warps. Όμως, ένας πολυεπεξεργαστής μίας GPU 2.x (αρχιτεκτονικής Fermi) μπορεί να χειριστεί το πολύ 48 warps. Αυτό σημαίνει ότι για να αξιοποιηθούν οι δυνατότητες κάθε πολυεπεξεργαστή στο μέγιστο δυνατό βαθμό, πρέπει το σύνολο των threads που εκτελούνται ανά πολυεπεξεργαστή να είναι πολλαπλάσιο του 48.

Έτσι, αν ο προγραμματιστής επιλέξει κάθε block να αποτελείται από 256 threads, δηλαδή $256/32 = 8$ warps, τότε, κάθε πολυεπεξεργαστής εκτελεί 6 blocks, δηλαδή $6{\times}8{=}48$ warps, και χρησιμοποιεί τους $6{\times}256{\times}20{=}30720$ από τους 32768 διαθέσιμους registers, αξιοποιώντας στην ουσία το 94% των διαθέσιμων registers. Το πλήθος των threads που εκτελούνται ανά πολυεπεξεργαστή είναι $256 \times 6 = 1536$, που είναι πολλαπλάσιο του 48.

Αντίθετα, αν ο προγραμματιστής επιλέξει αυθαίρετα κάθε block να αποτελείται από 320 threads, δηλαδή $320/32 = 10$ warps, το μέγιστο πλήθος από blocks που εκτελούνται ανά πολυεπεξεργαστή είναι 4, επειδή 5 blocks ισοδυναμούν με $5 \times 10 = 50$ warps που είναι περισσότερα των 48 που είναι το μέγιστο επιτρεπόμενο πλήθος από warps. Συνεπώς, κάθε πολυεπεξεργαστής εκτελεί $4 \times 10 = 40$ warps και χρησιμοποιεί μόλις τους $4{\times}320{\times}20{=}25600$ από τους 32768 διαθέσιμους registers. Στην ουσία κάθε πολυεπεξεργαστής χρησιμοποιεί μόλις το 78% των διαθέσιμων registers.

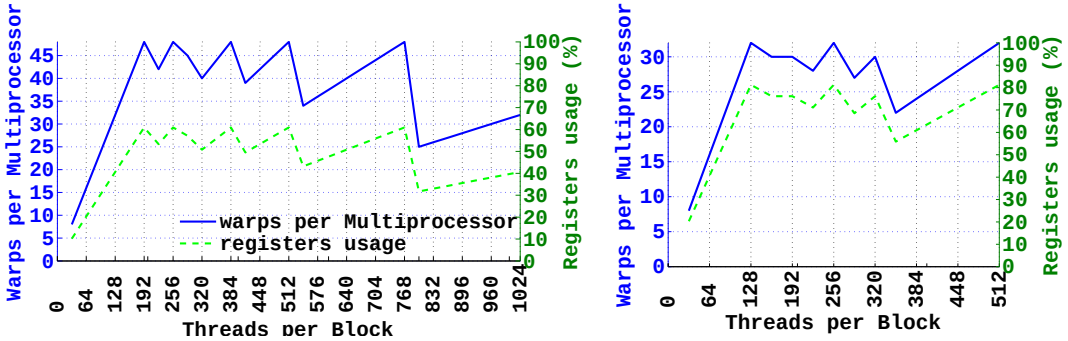
Από το παραπάνω παράδειγμα φαίνεται ότι η επιλογή του πλήθους των threads ανά block επηρεάζει σημαντικά το ποσοστό των registers που χρησιμοποιούνται ανά πολυεπεξεργαστή και στην ουσία καθορίζει την απόδοση των πολυεπεξεργαστών της κάρτας γραφικών. Γι΄ αυτό πρέπει, η επιλογή αυτού να γίνεται προσεκτικά ανά kernel.

Σημειώνεται ότι στην επιλογή των threads ανά block πρέπει να ληφθεί υπόψη και το μέγιστο επιτρεπόμενο πλήθος των threads ανά block και των blocks ανά πολυεπεξεργαστή. Στο παράδειγμα που παρουσιάστηκε προηγουμένως η επιλογή των 256 ή 320 threads ανά block σέβεται τον περιορισμό των 1024 threads ανά block για τις GPUs 2.x. Επίσης, τα 6 ή 4 blocks ανά πολυεπεξεργαστή είναι λιγότερα των 8 που αποτελεί το αντίστοιχο άνω όριο.

Τα σχήματα 3.10, 3.11, 3.12 δείχνουν το σύνολο των warps που χειρίζεται κάθε πο-

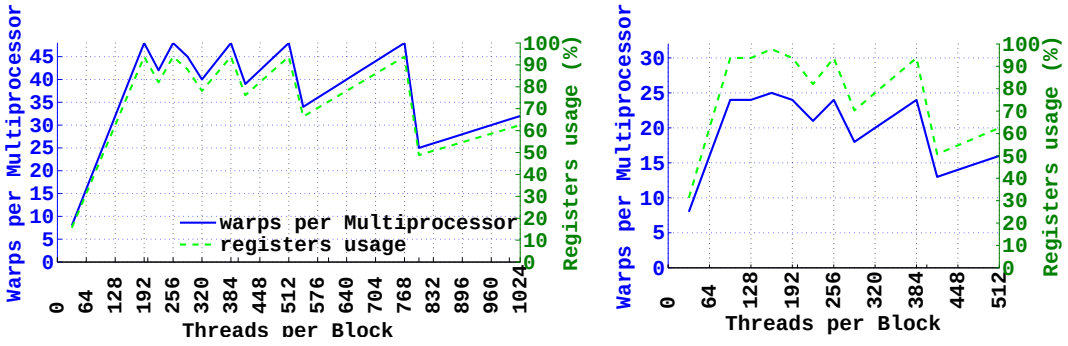
λυεπεξεργαστής μίας κάρτας γραφικών αρχιτεκτονικής GT200 ή Fermi συναρτήσει του αριθμού των threads ανά block. Στα ίδια σχήματα φαίνεται με διακεκομμένη γραμμή το ποσοστό των registers, ως προς τους διαθέσιμους, που χρησιμοποιούνται ανά πολυεπεξεργαστή. Τα σχήματα αυτά αναφέρονται σε kernels που έχουν μικρές απαιτήσεις σε shared μνήμη (όπως δηλαδή στο παράδειγμα που προηγήθηκε) και χρειάζονται διαφορετικό πλήθος από registers. Συγκεκριμένα, στα σχήματα 3.10, 3.11, 3.12 κάθε thread χρησιμοποιεί 13, 20 ή 36 registers αντίστοιχα.

Στην περίπτωση που κάθε thread χρειάζεται 13 registers για την εκτέλεση ενός kernel (σχήματα 3.10(α′), 3.10(β′)), αν επιλεχθεί προσεκτικά ο αριθμός των threads ανά block, οι πολυεπεξεργαστές της GPU εκτελούν το μέγιστο δυνατό πλήθος από warps χρησιμοποιώντας σαφώς λιγότερους από τους διαθέσιμους registers. Αντίθετα, στην περίπτωση που κάθε thread χρειάζεται 36 registers (σχήματα 3.12(α′), 3.12(β′)), κάθε πολυεπεξεργαστής χρησιμοποιεί σχεδόν το σύνολο των διαθέσιμων registers, το πλήθος όμως των warps που εκτελούνται ανά πολυεπεξεργαστή είναι σαφώς μικρότερο του ανώτερου δυνατού πλήθους (48 για GPUs αρχιτεκτονικής Fermi, ή 32 για GPUs αρχιτεκτονικής GT200). Στα σχήματα 3.12(α′), 3.12(β′) παρατηρείται, επίσης, ότι για πλήθος threads ανά block μεγαλύτερο του 896 ή 448 για GPUs αρχιτεκτονικής Fermi ή GT200 αντίστοιχα, οι διαθέσιμοι ανά πολυεπεξεργαστή registers δεν επαρκούν για την εκτέλεση ενός block. Συνεπώς, αν ο προγραμματιστής επιλέξει πλήθος threads ανά block μεγαλύτερο εκείνων, η εκτέλεση του kernel θα αποτύχει. Στο σχήμα 3.11(α′) φαίνεται ότι οι πολυεπεξεργαστές μίας GPU αρχιτεκτονικής Fermi χρησιμοποιούν το σύνολο σχεδόν των διαθέσιμων registers εκτελώντας ταυτόχρονα το μέγιστο δυνατό αριθμό warps. Αυτό όμως δεν συμβαίνει και στις GPUs αρχιτεκτονικής GT200 (σχήμα 3.11(β′)) όπου οι πολυεπεξεργαστές εκτελούν λιγότερα από το μέγιστο δυνατό αριθμό threads. Στα σχήματα 3.11(α′), 3.11(β′) κάθε thread χρειάζεται 20 registers.



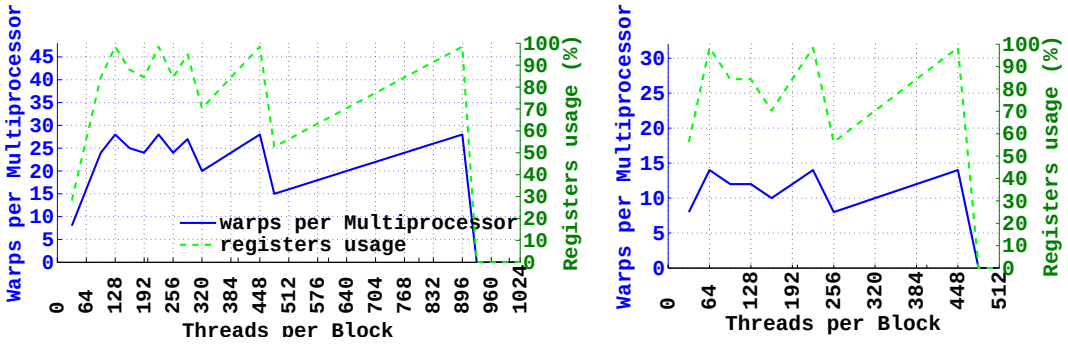
(α′) (β′)

Σχήμα 3.10: Σύνολο των warps που εκτελούνται ανά πολυεπεξεργαστή μίας GPU αρχιτεκτονικής (α′) Fermi ή (β′) GT200 συναρτήσει του αριθμού των threads ανά block, για ένα kernel η εκτέλεση του οποίου απαιτεί 13 registers ανά thread. Φαίνεται, επίσης, το ποσοστό των διαθέσιμων registers που χρησιμοποιούνται ανά πολυεπεξεργαστή.



(α′) (β′)

Σχήμα 3.11: Σύνολο των warps που εκτελούνται ανά πολυεπεξεργαστή μίας GPU αρχιτεκτονικής (α′) Fermi ή (β′) GT200 συναρτήσει του αριθμού των threads ανά block, για ένα kernel η εκτέλεση του οποίου απαιτεί 20 registers ανά thread. Φαίνεται, επίσης, το ποσοστό των διαθέσιμων registers που χρησιμοποιούνται ανά πολυεπεξεργαστή.



(α′) (β′)

Σχήμα 3.12: Σύνολο των warps που εκτελούνται ανά πολυεπεξεργαστή μίας GPU αρχιτεκτονικής (α′) Fermi ή (β′) GT200 συναρτήσει του αριθμού των threads ανά block, για ένα kernel η εκτέλεση του οποίου απαιτεί 36 registers ανά thread. Φαίνεται, επίσης, το ποσοστό των διαθέσιμων registers που χρησιμοποιούνται ανά πολυεπεξεργαστή.

3.14.2 Ρύθμιση των παραμέτρων μίας GPU 2.x

Οι GPUs υπολογιστικής δυνατότητας 2.x δίνουν τη δυνατότητα ρύθμισης ορισμένων επιπλέον παραμέτρων που επηρεάζουν την παράλληλη απόδοση ενός kernel. Οι παράμετροι αυτοί είναι:

- Το μέγεθος των shared, L1 cache μνημών ανά πολυεπεξεργαστή.

  Όπως έχει ήδη αναφερθεί στην παράγραφο 3.5, η L1 cache και η shared μνήμη ανά πολυεπεξεργαστή βρίσκονται στον ίδιο χώρο εσωτερικά της GPU. Αυτό επιτρέπει στον προγραμματιστή να επιλέξει τη διάσταση των δύο αυτών μνημών ανάμεσα σε 16 και 48 KBytes. Η προεπιλογή είναι 16 KBytes για την L1 cache και 48 KBytes για τη shared μνήμη. Η επιλογή αυτή μπορεί να είναι καθολική για όλα τα

kernels που εκτελούνται στη GPU ή για κάθε kernel ξεχωριστά. Στην περίπτωση που ο προγραμματιστής επιλέξει 16 KBytes shared μνήμη ανά πολυεπεξεργαστή, αλλά οι απαιτήσεις ενός block σε shared μνήμη είναι μεγαλύτερες των 16 και μικρότερες των 48 KBytes, η GPU αγνοεί την επιλογή του προγραμματιστή και μοιράζει τα συνολικά 64 KBytes ανά πολυεπεξεργαστή σε 16 KBytes L1 cache και 48 KBytes shared μνήμη.

Η απόδοση ενός kernel που έχει πρόσβαση σε θέσεις της κεντρικής μνήμης με βάση ένα μη-δομημένο πρότυπο ή που χρησιμοποιεί την τοπική ανά thread μνήμη αυξάνει σημαντικά με την αύξηση της χωρητικότητας της L1 αντί της shared ανά πολυεπεξεργαστή μνήμης.

- Χρήση των L1 και L2 cache ή μόνο της L2 cache κατά την προσπέλαση της κεντρικής και της τοπικής μνήμης.

  Ο προγραμματιστής μπορεί να επιλέξει αν θα χρησιμοποιούνται η L1 και η L2 cache (προεπιλογή) ή μόνο η L2 cache. Υπενθυμίζεται ότι κάθε πολυεπεξεργαστής έχει τη δική του L1 cache, ενώ η L2 cache είναι κοινή ανάμεσα σε αυτούς. Συνεπώς, μη-δομημένα μοτίβα πρόσβασης στη κεντρική μνήμη της GPU είναι πιθανό να ευνοηθούν από τη χρήση μόνο της L2 cache σε σχέση με το συνδυασμό L1 και L2 cache. Ο ισχυρισμός αυτός ενισχύεται από το γεγονός ότι ο συνδυασμός των L1 και L2 cache αντιλαμβάνεται την κεντρική μνήμη ως μία αλληλουχία από τμήματα των 128 Bytes. Αντίθετα, η χρήση μόνο της L2 cache χειρίζεται την κεντρική μνήμη ως τμήματα των 32 Bytes.

- Ορισμός του μέγιστου αριθμού registers που μπορεί να χρησιμοποιήσει ένα kernel.

  Ο προγραμματιστής μπορεί να ορίσει το μέγιστο αριθμό από registers που μπορεί να χρησιμοποιήσει ένα kernel. Στην περίπτωση που το kernel χρειάζεται περισσότερους registers από το άνω όριο που όρισε ο προγραμματιστής, η επιλογή αυτή έχει ένα πλεονέκτημα και ένα μειονέκτημα και χρειάζονται δοκιμές για να βρεθεί η βέλτιστη αναλογία.

  Το πλεονέκτημα είναι ότι μειώνοντας τον αριθμό των registers που χρησιμοποιεί ένα kernel (όπως παρουσιάστηκε στην παράγραφο 3.14.1), αυξάνεται ο αριθμός των warps που χειρίζεται ένας πολυεπεξεργαστής. Επομένως, αυξάνεται η παράλληλη απόδοση του kernel.

  Το μειονέκτημα είναι ότι, για να εκτελεστεί το kernel, χρησιμοποιεί συχνότερα την τοπική μνήμη, η οποία έχει υψηλό κόστος προσπέλασης και αυξάνει το χρόνο εκτέλεσης του κάθε thread.

Κεφάλαιο 4

Ανάλυση του GPU-κώδικα επίλυσης πεδίων ροής

Το κεφάλαιο αυτό παρουσιάζει μία πολύ αποδοτική μεταφορά ήδη υπάρχοντος CPU-κώδικα επίλυσης των χρονικά μόνιμων και μη-μόνιμων εξισώσεων Navier–Stokes/Euler σε μη-δομημένα υπολογιστικά πλέγματα σε GPUs χρησιμοποιώντας το περιβάλλον προγραμματισμού της CUDA. Ο CPU-κώδικας είναι γραμμένος σε FORTRAN90 και έχει αναπτυχθεί και πιστοποιηθεί στο πλαίσιο παλαιότερων διδακτορικών διατριβών που εκπονήθηκαν στη ΜΠΥΡ&Β/ΕΘΣ [118, 129, 130]. Συνεπώς, στο τμήμα της αεροδυναμικής, βασικό μέλημα της παρούσας διατριβής αποτέλεσε η μεγιστοποίηση της παράλληλης απόδοσης του GPU-κώδικα.

Η πρώτη απόπειρα προγραμματισμού σε GPUs, στην αρχική φάση της παρούσας διατριβής, έγινε συνεργατικά με τον δόκτορα Ι. Καμπόλη στα τέλη του 2008 (την περίοδο εκείνη ο Ι. Καμπόλης ολοκλήρωνε την διατριβή του, [131], στη ΜΠΥΡ&Β/ΕΘΣ). Την εποχή εκείνη, το περιβάλλον προγραμματισμού της CUDA υποστήριζε ως γλώσσα προγραμματισμού μόνο τη C++ (με την προσθήκη ορισμένων επιπλέον συναρτήσεων για τη διαχείριση των GPUs). Συνεπώς, αναγκαστικά, ο CPU-επιλύτης των 2Δ εξισώσεων Euler ξαναγράφτηκε από Fortran90 σε C++. Παρόλο που ο GPU-επιλύτης που προέκυψε μείωσε τον χρόνο επίλυσης προβλημάτων αεροδυναμικής (σε σχέση με τη χρήση του αντίστοιχου CPU-κώδικα), η παράλληλη απόδοση που είχε ήταν αρκετά χαμηλή. Ο τελικός GPU-κώδικας της διατριβής έχει 20 φορές υψηλότερη παράλληλη απόδοση. Στο κεφάλαιο αυτό παρουσιάζονται-αναλύονται οι ενέργειες που έγιναν και συνέβαλαν στην αύξηση αυτή.

Το βασικό μειονέκτημα των σημερινών GPUs είναι η περιορισμένη (σε σχέση με τις σημερινές CPUs) χωρητικότητα των cache μνημών που έχουν. Ενδεικτικά, όπως αναφέρεται στην παράγραφο 3.8, κάθε πυρήνας μίας CPU ενός υπολογιστικού κόμβου της συστοιχίας διασυνδεδεμένων GPUs της ΜΠΥΡ&Β/ΕΘΣ χρησιμοποιεί 12288 KByte cache μνήμης. Αντίθετα, όλα τα threads μίας GPU αρχιτεκτονικής GT200 ή Fermi που εκτελούνται στον ίδιο πολυεπεξεργαστή χρησιμοποιούν μόλις 32 ή 128 KByte cache μνήμης αντίστοιχα. Υπενθυμίζεται ότι όσο μεγαλύτερη cache μνήμη έχει μία μονάδα επεξεργασίας τόσο πιο γρήγορα εκτελείται ένας κώδικας σε αυτή. Η μικρή έκταση

των cache μνημών μίας GPUs καθιστά την παράλληλη απόδοση οποιουδήποτε GPU-κώδικα, άρρηκτα συνδεδεμένη με την απόδοση του πρότυπου προσπέλασης των μνημών της GPU. Έτσι, μεγάλο τμήμα του κεφαλαίου αυτού αναγκαστικά επικεντρώνεται σε θέματα διαχείρισης των μνημών μίας GPU.

Πέρα της προσεκτικής διαχείρισης των μνημών μίας GPU, σημαντική αύξηση της παράλληλης απόδοσης του GPU-κώδικα επιτεύχθηκε και με τη χρήση της αριθμητικής 'μικτής' ακρίβειας, [8]. Αυτή αποτελεί έναν συνδυασμό απλής και διπλής ακρίβειας, με τους συντελεστές του αριστερού μέλους των διακριτοποιημένων εξισώσεων να αποθηκεύονται σε μεταβλητές απλής ακρίβειας και οι όροι του δεξιού μέλους (δηλαδή τα υπόλοιπα των εξισώσεων) σε μεταβλητές διπλής ακρίβειας. Η χρήση αριθμητικής μικτής ακρίβειας αυξάνει την παράλληλη απόδοση του GPU-κώδικα χωρίς να αλλοιώνει την ακρίβεια επίλυσης των εξισώσεων Navier–Stokes. Η εφαρμογή της αριθμητικής αυτής παρουσιάζεται επίσης στο κεφάλαιο αυτό.

Σημαντική αύξηση της παράλληλης απόδοσης και, κατά συνέπεια, της επιτάχυνσης από τη χρήση GPUs στην επίλυση προβλημάτων αεροδυναμικής/αεροελαστικότητας, έδωσε η επαναρίθμηση των κόμβων του πλέγματος και η ταξινόμηση αυτών με βάση το πλήθος των γειτονικών κόμβων ανά κόμβο. Η τεχνική της επαναρίθμησης που ακολουθήθηκε αναλύεται στη συνέχεια. Η βελτιστοποίηση της συνεργασίας μεταξύ CPU και GPU βοήθησε στην περαιτέρω αύξηση της παράλληλης απόδοσης, όπως παρουσιάζεται επίσης.

Στο τέλος του κεφαλαίου, παρουσιάζεται η απόδοση του GPU-κώδικα σε όρους επιτάχυνσης της πρόλεξης πεδίων ροής. Οι επιταχύνσεις προκύπτουν με βάση το χρόνο που χρειάζεται ο αντίστοιχος CPU-κώδικας σε έναν πυρήνα μίας σύγχρονης CPU. Η εν λόγω επιτάχυνση αυξάνει με τον αριθμό των κόμβων του πλέγματος. Δηλαδή, προβλήματα μεγάλης κλίμακας επωφελούνται περισσότερο της χρήσης των GPUs. Ενδεικτικά αναφέρεται ότι, επιτυγχάνεται επιτάχυνση στην επίλυση των 2Δ εξισώσεων Navier–Stokes έως και 110x, 90x ή 60x χρησιμοποιώντας αριθμητική απλής, μικτής ή διπλής ακρίβειας, αντίστοιχα. Όμοια, η χρήση GPUs επιταχύνει την επίλυση 3Δ ατριβών ροών έως και 90, 55 ή 40 φορές κάνοντας χρήση αριθμητικής απλής, μικτής ή διπλής ακρίβειας, αντίστοιχα.

Τα θέματα που αναπτύσσονται στο κεφάλαιο αυτό δεν αφορούν αποκλειστικά GPU-επιλύτες των εξισώσεων Navier–Stokes σε μη-δομημένα πλέγματα καθώς τα συμπεράσματα που απορρέουν γενικεύονται εύκολα και μπορούν να χρησιμοποιηθούν στην επίλυση οποιουδήποτε συστήματος μερικών διαφορικών εξισώσεων σε πλέγματα αυτού του τύπου.

Το κεφάλαιο αυτό αναφέρεται αποκλειστικά στην αποδοτικότερη μεταφορά του CPU-επιλύτη σε μία GPU. Η παραλληλοποίηση του GPU-επιλύτη σε επίπεδο πολλών GPUs αποτελεί αντικείμενο του κεφαλαίου 5.

4.1 Ο GPU-επιλύτης

Ο GPU-κώδικας που αναπτύχθηκε στη διατριβή αυτή επιλύει τις 2Δ χρονικά μόνιμες και μη-μόνιμες εξισώσεις Navier–Stokes και τις 3Δ χρονικά μόνιμες και μη-μόνιμες

εξισώσεις ατριβούς ρευστού (αριθμητική επίλυση των εξισώσεων Euler). Η μοντελοποίηση της τύρβης, όπου χρειάζεται, γίνεται με το μοντέλο μίας εξίσωσης των Spalart-Allmaras. Η ολοκλήρωση των εξισώσεων της ροής γίνεται σε μη-δομημένα (υβριδικά, για την επίλυση 2Δ ροών αποτελούμενα από τριγωνικά και τετραπλευρικά πλεγματικά στοιχεία) χρησιμοποιώντας την κεντροκομβική διατύπωση της τεχνικής των πεπερασμένων όγκων. Ο GPU-κώδικας είναι δεύτερης τάξης ακρίβειας τόσο στο χώρο όσο και στο χρόνο και έχει τη δυνατότητα χρήσης δυναμικών πλεγμάτων. Τα πλέγματα αυτά ακολουθούν την κίνηση-παραμόρφωση της επιφάνειας του εξεταζόμενου σώματος, προσαρμοζόμενα σ΄ αυτή. Τα παραπάνω χαρακτηρίζουν την επίλυση τόσο αεροελαστικών όσο και αεροδυναμικών προβλημάτων σε χωρία με κινούμενα στερεά όρια. Η επέκταση του GPU-κώδικα στην επίλυση 3Δ τυρβωδών ροών, συμπεριλαμβανομένων των περιφερειακών πτερυγώσεων, γίνεται στο πλαίσιο άλλης διδακτορικής διατριβής που εκπονείται από τον Ι. Καββαδία στη ΜΠΥΡ&Β/ΕΘΣ. Ο GPU-κώδικας είναι παράλληλος σε πολλές κάρτες γραφικών του ίδιου ή διαφορετικών υπολογιστικών κόμβων.

4.2 Η δομή του GPU-επιλύτη

Η εκτέλεση του GPU-επιλύτη ξεκινά από τη CPU η οποία διαβάζει τα δεδομένα του πλέγματος, δηλαδή τις συντεταγμένες των κόμβων αυτού και το πώς αυτοί συνδέονται μεταξύ τους για να σχηματίσουν τα πλεγματικά στοιχεία. Διαβάζει επίσης τις συνθήκες της ροής, όπως και μία σειρά από παραμέτρους σχετικές με την αριθμητική ολοκλήρωση των εξισώσεων της ροής. Στη συνέχεια, υπολογίζει μία σειρά τοπολογικών και γεωμετρικών στοιχείων σχετικών με τον τρόπο επίλυσης στο συγκεκριμένο πλέγμα, τα οποία αντιγράφει στην κεντρική μνήμη της GPU, αφού έχει δεσμεύσει προηγουμένως τον κατάλληλο χώρο σ΄ αυτή. Μεταξύ των απαραίτητων τοπολογικών δεδομένων είναι το πλήθος και οι αύξοντες αριθμοί των κόμβων που ενώνονται άμεσα (δηλαδή μέσω ακμής) με κάθε κόμβο κ.α. Τα κάθετα διανύσματα στα όρια των όγκων ελέγχου και ο όγκος των τελευταίων είναι μερικά από τα γεωμετρικά στοιχεία που υπολογίζονται.

Στη συνέχεια, πραγματοποιείται η δέσμευση του απαραίτητου χώρου στην κεντρική μνήμη της GPU (επιπλέον εκείνου που χρειάζονται τα τοπολογικά και γεωμετρικά στοιχεία) και ξεκινά η επίλυση των εξισώσεων Navier–Stokes/Euler. Για την επίλυση μη-μόνιμων ροών, το πεδίο της ροής υπολογίζεται σε διαδοχικές χρονικές στιγμές. Η απόσταση μεταξύ δύο διαδοχικών χρονικών στιγμών είναι σταθερή και καθορίζεται από το χρήστη. Ο υπολογισμός του πεδίου της ροής σε κάθε χρονική στιγμή γίνεται επαναληπτικά και περιλαμβάνει σειρά από kernels τα οποία καλούνται από τη CPU και εκτελούνται παράλληλα στους διαθέσιμους πολυεπεξεργαστές της GPU. Τα kernels αυτά πραγματοποιούν τα ακόλουθα:

- Αρχικοποίηση του πεδίου της ροής. Στην περίπτωση επίλυσης χρονικά μη-μόνιμης ροής, η αρχικοποίηση του πεδίου της ροής την τρέχουσα χρονική στιγμή γίνεται με το συγκλιμένο πεδίο της προηγούμενης χρονικής στιγμής.
- Υπολογισμό των αριθμητικών διανυσμάτων της ροής σύμφωνα με το σχήμα του Roe ή της τεχνικής Flux Vector Splitting (FVS, κεφάλαιο 2) που εισέρχονται/εξέρχονται στους/από τους όγκους ελέγχου και ο ισολογισμός των ροών αυτών

για το σχηματισμό των υπολοίπων των εξισώσεων της ροής ανά κόμβο αλλά και των μητρώων/συντελεστών του αριστερού μέλους των διακριτοποιημένων εξισώσεων.

- Ανανέωση του πεδίου τιμών των μεταβλητών της ροής με τη μέθοδο Jacobi.
- Ανανέωση του πεδίου τιμών των μεταβλητών του μοντέλου τύρβης με τη μέθοδο Jacobi (στην περίπτωση επίλυσης τυρβωδών ροών).
- Υπολογισμό της νόρμας του υπολοίπου των διακριτοποιημένων εξισώσεων.

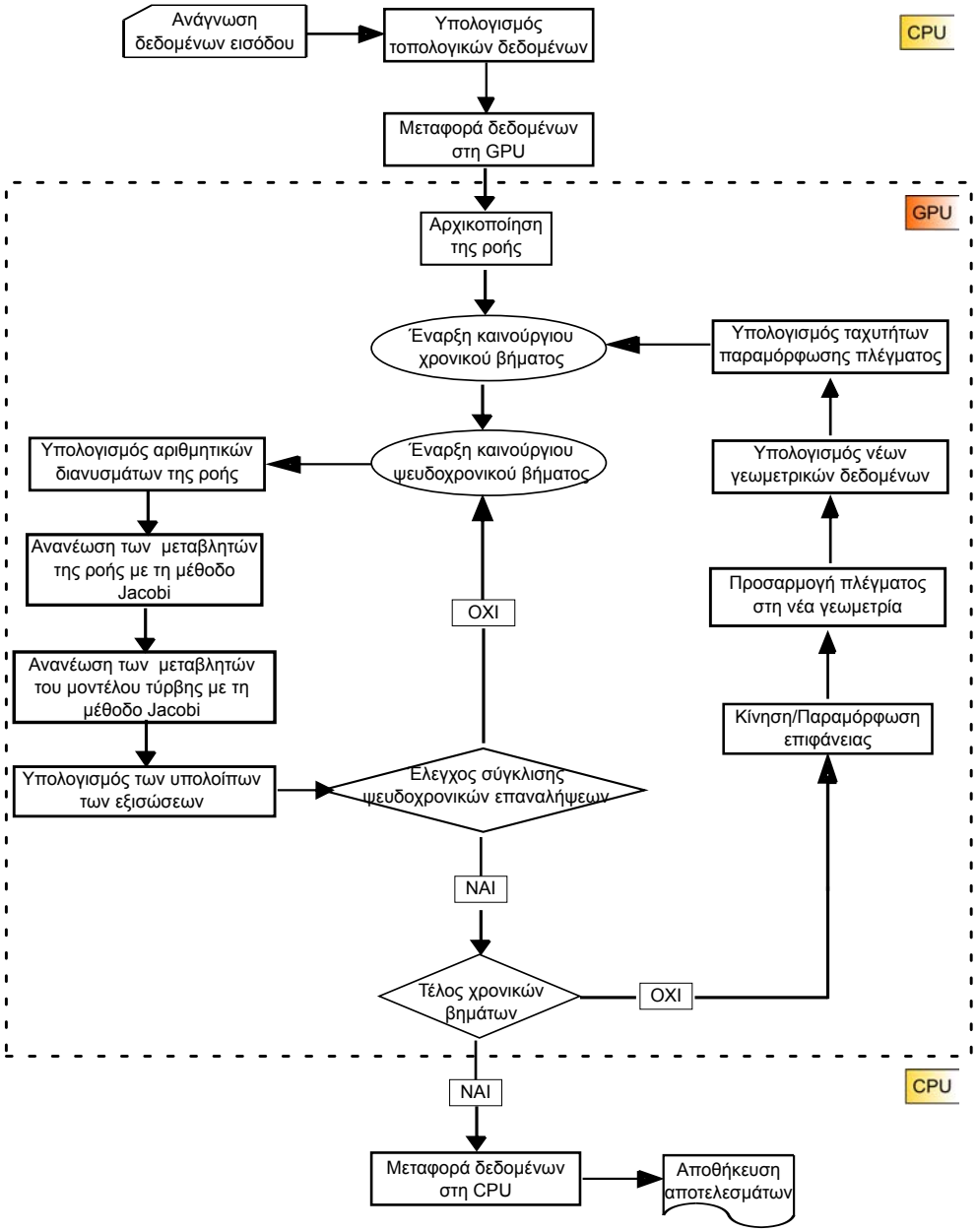
Στον υπολογισμό της νόρμας του υπολοίπου των διακριτοποιημένων εξισώσεων συμμετέχει και η CPU, καθώς η GPU υπολογίζει το μερικό άθροισμα των υπολοίπων ανά κόμβο και εξίσωση και η CPU υπολογίζει την τιμή της νόρμας ανά εξίσωση. Για περισσότερες λεπτομέρειες σχετικά με τη συνεργασία αυτή μεταξύ CPU-GPU, στον υπολογισμό της νόρμας του υπολοίπου των εξισώσεων, ο αναγνώστης παραπέμπεται στην παράγραφο 4.4.6. Έπειτα, η CPU ελέγχει αν η επίλυση του τρέχοντος χρονικού βήματος έχει συγκλίνει. Στην περίπτωση που δεν έχει επέλθει σύγκλιση επαναλαμβάνονται οι παραπάνω ενέργειες. Αντίθετα, σε περίπτωση σύγκλισης των επαναλήψεων στον ψευδοχρόνο σε μία χρονική στιγμή ενός χρονικά μη-μόνιμου προβλήματος, ο GPU-επιλύτης προχωρά στην επίλυση του επόμενου χρονικού βήματος. Αν το πρόβλημα που επιλύεται είναι χρονικά μόνιμο και οι επαναλήψεις στον ψευδο-χρόνο έχουν συγκλίνει, η CPU αντιγράφει το πεδίο της ροής από την κεντρική μνήμη της GPU σε εκείνην του υπολογιστή και αποθηκεύει τα αποτελέσματα. Πριν την ολοκλήρωση του τρεξίματος, γίνεται η αποδέσμευση των θέσεων μνήμης που είχαν δεσμευτεί κατά την επίλυση τόσο στη GPU όσο και στη CPU.

Στην περίπτωση επίλυσης ενός μη-μόνιμου προβλήματος όπου το εξεταζόμενο αεροδυναμικό σώμα μετακινείται (όπως συμβαίνει στην επίλυση προβλημάτων όπου το αεροδυναμικό σώμα εκτελεί μία εξαναγκασμένη ταλάντωση), πριν την επίλυση της ροής κατά την επόμενη πραγματική χρονική στιγμή, η GPU εκτελεί τις ακόλουθες ενέργειες:

- Μετακινεί το αεροδυναμικό σώμα.
- Προσαρμόζει το πλέγμα στη νέα θέση ή/και στο νέο σχήμα του σώματος.
- Ανανεώνει τα γεωμετρικά στοιχεία που είχε υπολογίσει προηγουμένως.
- Υπολογίζει την ταχύτητα μετακίνησης των κόμβων του πλέγματος.

Η διαδικασία προσαρμογής του πλέγματος στη νέα θέση-σχήμα του εξεταζόμενου σώματος αναλύεται-παρουσιάζεται στην παράγραφο 8.2.

Το σχήμα 4.1 δείχνει το διάγραμμα ροής του GPU-επιλύτη.



Σχήμα 4.1: Διάγραμμα ροής του GPU-κώδικα. Το τμήμα κώδικα που περιβάλλονται από τη διακεκομμένη γραμμή εκτελούνται στη GPU.

4.3 Δυσκολίες στη μεταφορά του CPU-επιλύτη στη GPU

Η ενότητα αυτή αναφέρεται στα βασικά προβλήματα που έπρεπε να ξεπεραστούν για την αποδοτικότερη μεταφορά του CPU-επιλύτη στη GPU. Αυτά απορρέουν από τη χρήση

(α') Κοινής μνήμης από τα threads της GPU.

(β') Μη-δομημένων υπολογιστικών πλεγμάτων.

(γ') Της κεντροκομβικής διατύπωσης της τεχνικής των πεπερασμένων όγκων σε μη-δομημένα πλέγματα.

Τα προβλήματα αυτά αναλύονται και σχολιάζονται στις επόμενες παραγράφους.

4.3.1 Χρήση κοινής μνήμης από τα threads της GPU

Οι σύγχρονες GPUs είναι μονάδες παράλληλης επεξεργασίας κοινής μνήμης. Δηλαδή, δύο threads που εκτελούνται ταυτόχρονα απαγορεύεται να ανανεώνουν την ίδια θέση μνήμης. Συνεπώς, ο προγραμματιστής ενός GPU-λογισμικού οφείλει να αποτρέπει την ταυτόχρονη αποθήκευση σε θέσεις μνήμης της GPU από δύο ή περισσότερα threads.

Μία δυνατή λύση είναι η χρήση των atomic functions, οι οποίες όμως επιβάλλουν επιπλέον συγχρονισμούς ανάμεσα στα threads, με αποτέλεσμα τη μείωση της παράλληλης απόδοσης του GPU-κώδικα. Επιπλέον, η επιλογή-ανάγκη χρήσης αριθμητικής διπλής ακρίβειας (π.χ. σε εφαρμογές ΥΡΔ) περιορίζει τη χρήση των atomic functions σε GPUs τελευταίας τεχνολογίας καθώς, μόνο στις κάρτες γραφικών τελευταίας αρχιτεκτονικής, οι atomic functions μπορούν και χειρίζονται διπλής ακρίβειας πραγματικούς αριθμούς.

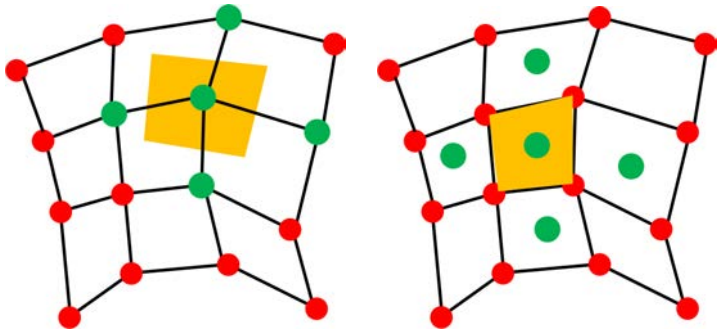
Για τη μεγιστοποίηση της παράλληλης απόδοσης του GPU-επιλύτη της διατριβής δοκιμάστηκαν και αξιολογήθηκαν, σύμφωνα με το χρόνο εκτέλεσης τους, διαφορετικές προγραμματιστικές τεχνικές για τον υπολογισμό των αριθμητικών διανυσμάτων της ροής [57, 58]. Οι τεχνικές αυτές αποτρέπουν την ταυτόχρονη ανανέωση θέσεων μνήμης της κάρτας γραφικών και παρουσιάζονται στην ενότητα 4.4.

4.3.2 Χρήση μη-δομημένων πλεγμάτων

Στα μη-δομημένα πλέγματα, η αρίθμηση των κόμβων, των ακμών και των πλεγματικών στοιχείων χαρακτηρίζεται από έλλειψη δομής-οργάνωσης, που συνεπάγεται μία ακατάστατη πρόσβαση στην κεντρική μνήμη της GPU από τα threads που εκτελούνται σ΄ αυτή. Το γεγονός αυτό μειώνει την παράλληλη απόδοση του GPU-κώδικα. Αντίθετα, οι GPU-κώδικες που χρησιμοποιούν δομημένα πλέγματα (σχήμα 4.2) επωφελούνται κατά πολύ από τη δομημένη αρίθμηση των κόμβων και επιτυγχάνουν μεγάλες, ή ακόμη μεγαλύτερες, επιταχύνσεις.

Στην παράγραφο 4.4.3 αναλύεται ο τρόπος προσπέλασης της κεντρικής μνήμης της GPU από τα threads, με σκοπό τη μεγιστοποίηση της παράλληλης απόδοσης του GPU-κώδικα. Υπενθυμίζεται ότι η κεντρική μνήμη είναι cached μόνο σε κάρτες γραφικών

αρχιτεκτονικής Fermi και, γενικά, χαρακτηρίζεται από υψηλούς χρόνους προσπέλασης. Επιπλέον, η παράγραφος 4.4.5 παρουσιάζει μία τεχνική επαναρίθμησης των κόμβων και των ακμών του πλέγματος, ώστε threads που εκτελούνται ταυτόχρονα να έχουν πρόσβαση σε πλησιέστερες θέσεις μνήμης της GPU.



(α′) (β′)

Σχήμα 4.2: Σχηματισμός (α) κεντροκομβικού και (β) κεντροκυψελικού όγκου ελέγχου σε δομημένα πλέγματα. Τα δομημένα πλέγματα επωφελούνται της δομημένης αρίθμησης των κόμβων/στοιχείων σε αντίθεση με ότι συμβαίνει στα μη-δομημένα πλέγματα.

4.3.3 Κεντροκομβική διατύπωση πεπερασμένων όγκων σε μη-δομημένα πλέγματα

Η ολοκλήρωση των εξισώσεων Navier–Stokes/Euler γίνεται σύμφωνα με την κεντροκομβική διατύπωση της τεχνικής των πεπερασμένων όγκων σε μη-δομημένα πλέγματα. Σύμφωνα με τη διατύπωση αυτή, οι εξισώσεις της ροής ολοκληρώνονται σε όγκους ελέγχου που σχηματίζονται γύρω από τους κόμβους του πλέγματος όπως φαίνεται στο σχήμα 4.3(α′). Κατά την επαναληπτική επίλυση των εξισώσεων της ροής, είναι απαραίτητη η ανταλλαγή πληροφορίας μεταξύ γειτονικών κόμβων του πλέγματος. Ο χρόνος εκτέλεσης ενός thread που σχετίζεται με έναν κόμβο του πλέγματος και χρησιμοποιεί πληροφορία από τους γειτονικούς κόμβους είναι ανάλογος του πλήθους των γειτονικών κόμβων του. Threads που συσχετίζονται με κόμβους και ζητούν πληροφορία από τους γειτονικούς είναι εκείνα που υπολογίζουν τα αριθμητικά διανύσματα της ροής που εισέρχονται σε κάθε όγκο ελέγχου και σχηματίζουν το ανά κόμβο υπόλοιπο των διακριτοποιημένων εξισώσεων, ανανεώνουν τις ροϊκές μεταβλητές στους κόμβους του πλέγματος μέσω της μεθόδου επίλυσης Jacobi, ή υπολογίζουν τις χωρικές παραγώγους των ροϊκών μεταβλητών στους κόμβους του πλέγματος.

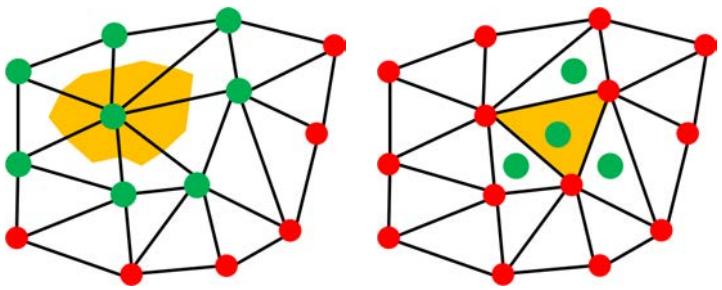
Το πλήθος των γειτονικών κόμβων ανά κόμβο, σε ένα μη-δομημένο πλέγμα, γενικά μπορεί να μεταβάλλεται από κόμβο σε κόμβο. Το γεγονός αυτό μειώνει την παράλληλη απόδοση του GPU-κώδικα. Για να γίνει καλύτερα αντιληπτό αυτό, θεωρούμε τα threads $thread_A$ και $thread_B$ που ανήκουν στο ίδιο warp και σχετίζονται με τους κόμβους A και B του πλέγματος, αντίστοιχα. Υποθέτουμε ότι ο κόμβος A έχει διπλάσιο πλήθος

γειτονικών κόμβων από τον B. Αν τα $thread_A$ και $thread_B$ εκτελούνταν ανεξάρτητα το ένα από το άλλο, τότε ο χρόνος εκτέλεσης του $thread_A$ αναμένεται να είναι περίπου διπλάσιος εκείνου του $thread_B$. Επειδή όμως ανήκουν στο ίδιο warp, το $thread_B$ περιμένει την ολοκλήρωση της εκτέλεσης του $thread_A$. Δηλαδή, ο χρόνος εκτέλεσης των $thread_A$ και $thread_B$ θα είναι τελικά ίδιος, παρά το ότι το πλήθος των γειτονικών κόμβων του κόμβου A είναι διπλάσιο εκείνου του κόμβου B. Γίνεται αντιληπτό ότι όσο μεγαλύτερη είναι η διασπορά του πλήθους των γειτονικών κόμβων ανάμεσα στους κόμβους με τους οποίους έχουν αντιστοιχηθεί τα threads ενός warp, τόσο χαμηλότερη είναι η παράλληλη απόδοση του GPU-κώδικα.

Αντίθετα, η χρήση της κεντροκυψελικής διατύπωσης συνεπάγεται σταθερό και εκ προοιμίου γνωστό πλήθος γειτονικών πλεγματικών στοιχείων ανά πλεγματικό στοιχείο (τουλάχιστον για τα εσωτερικά). Για παράδειγμα, σε ένα πλέγμα τριγωνικών στοιχείων, ένα μη-οριακό τριγωνικό στοιχείο περικλείεται αυστηρά από τρία άλλα (σχήμα 4.3(β′)). Συνεπώς, threads που σχετίζονται με τα στοιχεία του πλέγματος (τρίγωνα) και συσσωρεύουν πληροφορία από τα γειτονικά σε αυτά στοιχεία (τρίγωνα), έχουν τον ίδιο χρόνο εκτέλεσης.

Η μείωση της διασποράς του πλήθους των γειτόνων των κόμβων που σχετίζονται με τα threads του ίδιου warp επιτεύχθηκε με την ταξινόμηση των κόμβων του πλέγματος σύμφωνα με το πλήθος των γειτονικών σ′ αυτούς κόμβους.

Επιπλέον, το διαφορετικό πλήθος γειτονικών κόμβων ανά κόμβο επηρεάζει τον τρόπο προσπέλασης της κεντρικής μνήμης της κάρτας γραφικών. Η παράγραφος 4.4.3 περιγράφει τον τρόπο αποθήκευσης, [57, 58], των μη-διαγώνιων στοιχείων των μητρώων του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής, στην κεντρική μνήμη, με στόχο την ταχύτερη δυνατή προσπέλαση της μνήμης αυτής από τα threads.



Σχήμα 4.3: Σχηματισμός (α) κεντροκομβικού και (β) κεντροκυψελικού όγκου ελέγχου σε μη-δομημένα πλέγματα. Η κεντροκυψελική διατύπωση πλεονεκτεί της κεντροκομβικής λόγω του γνωστού και σταθερού πλήθους των γειτονικών όγκων ελέγχου ανά όγκο ελέγχου.

4.4 Μεταφορά του επιλύτη των εξισώσεων Navier–Stokes στη GPU

Η προηγούμενη ενότητα παρουσίασε τις βασικές δυσκολίες που προέκυψαν κατά τη μεταφορά του CPU-επιλύτη στη GPU. Η ενότητα αυτή αναλύει τους τρόπους με τους οποίους ξεπεράστηκαν οι δυσκολίες αυτές, με γνώμονα πάντα την παραγωγή GPU-λογισμικού υψηλής παράλληλης απόδοσης.

Αξίζει να αναφερθεί ότι οι δυσκολίες που αναφέρθηκαν προηγουμένως καθιστούν την περίπτωση της κεντροκομβικής διατύπωσης της τεχνικής των πεπερασμένων όγκων σε μη-δομημένα πλέγματα ως εκείνη με τις περισσότερες δυσκολίες, σε σύγκριση με τη χρήση δομημένων πλεγμάτων ή, έστω, την κεντροκυψελική διατύπωση σε μη-δομημένα πλέγματα. Αυτό αποτέλεσε πρόκληση. Εξάλλου, ο προϋπάρχων CPU-κώδικας, ο οποίος είναι επαρκώς πιστοποιημένος, στηρίζεται στην κεντροκομβική διατύπωση. Παρά τις δυσκολίες αυτές, ο GPU-επιλύτης της διατριβής επιτυγχάνει επιτάχυνση της πρόλεξης της ροής αντίστοιχη εκείνης GPU-λογισμικών άλλων ερευνητικών ομάδων που χρησιμοποιούν δομημένα πλέγματα ή την κεντροκυψελική διατύπωση.

Στην ενότητα αυτή, αρχικά περιγράφεται η ολοκλήρωση των εξισώσεων Navier–Stokes σε κεντροκομβικούς όγκους ελέγχου και επαναλαμβάνονται βασικές εκφράσεις που διατυπώθηκαν στο κεφάλαιο 2 για την πιο εύκολη αποπομπή σε αυτές όταν αυτό χρειάζεται. Στη συνέχεια, παρουσιάζονται και αξιολογούνται, ως προς τον χρόνο εκτέλεσης, οι προγραμματιστικές τεχνικές που δοκιμάστηκαν για τον υπολογισμό των αριθμητικών διανυσμάτων της ροής και παρουσιάζεται ο τρόπος πρόσβασης στην κεντρική μνήμη της GPU των αποδοτικότερων τεχνικών. Δοκιμάζεται η απόδοση μίας τεχνικής επαναρίθμησης των κόμβων του πλέγματος και αξιολογείται με βάση τη μείωση του χρόνου εκτέλεσης του GPU-επιλύτη. Προτείνεται η συνεργασία της CPU με τη GPU για τον υπολογισμό της νόρμας των υπολοίπων των εξισώσεων της ροής και παρουσιάζεται η χρήση της αριθμητικής μικτής ακρίβειας που αυξάνει σημαντικά την απόδοση του GPU-κώδικα χωρίς να αλλοιώνει την ακρίβεια των αποτελεσμάτων.

4.4.1 Ολοκλήρωση των εξισώσεων Navier–Stokes σε κεντροκομβικούς όγκους ελέγχου

Οι χρονικά μόνιμες εξισώσεις Navier–Stokes ολοκληρωμένες στους όγκους ελέγχου που σχηματίζονται γύρω από τους κόμβους του πλέγματος συμβολίζονται ως

$$\overrightarrow{\mathcal{R}}\left(\overrightarrow{W}\right) = 0 \tag{4.1}$$

όπου $\overrightarrow{W}$ οι συντηρητικές μεταβλητές της ροής.

Το σύστημα των διακριτοποιημένων εξισώσεων 4.1 γραμμικοποιείται και επιλύεται επαναληπτικά ως προς τη διόρθωση $\left(\Delta\overrightarrow{W}\right)$ των συντηρητικών μεταβλητών της ροής,

$$\frac{\partial \overrightarrow{\mathcal{R}}}{\partial \overrightarrow{W}} \Delta \overrightarrow{W} = -\overrightarrow{\mathcal{R}} \left(\overrightarrow{W} \right) \tag{4.2}$$

$$\overrightarrow{W}^{n+1} = \overrightarrow{W}^{n} + \Delta \overrightarrow{W} \tag{4.3}$$

όπου n, είναι ο μετρητής των βημάτων επίλυσης-επαναλήψεων στον ψευδοχρόνο.

Διακρίνοντας το αριστερό μέλος της 4.2 σε διαγώνιους (D) και μη-διαγώνιους όρους (Z), η 4.2 σε έναν κόμβο P του πλέγματος παίρνει τη μορφή

$$D_P^n \Delta \overrightarrow{W}_P^{n+1} + \sum_{Q \in nei(P)} Z_Q^n \Delta \overrightarrow{W}_Q^{n+1} = -\overrightarrow{\mathcal{R}}_P^n \tag{4.4}$$

όπου Q είναι οι γειτονικοί στον P κόμβοι και $nei(P)$ το σύνολο των Q. $\overrightarrow{\mathcal{R}}_P^n$ είναι το υπόλοιπό των διακριτοποιημένων εξισώσεων της ροής στον κόμβο P κατά το ψευδοχρονικό βήμα n.

Για κάθε n, οι διακριτοποιημένες εξισώσεις της ροής 4.4 επιλύονται επαναληπτικά με τη μέθοδο Jacobi,

$$\Delta \overrightarrow{W}_P^{n+1,j+1} = -\left[D_P^n\right]^{-1} \cdot \left[\overrightarrow{\mathcal{R}}_P^n + \sum_{Q \in nei(P)} Z_Q^n \Delta \overrightarrow{W}_Q^{n+1,j} \right] \tag{4.5}$$

όπου j, είναι ο μετρητής των Jacobi επαναλήψεων. Η μέθοδος αυτή είναι μία ρητή μέθοδος υψηλής παράλληλης απόδοσης όταν εφαρμόζεται σε GPUs.

Ο υπολογισμός των υπολοίπων των εξισώσεων της ροής ανά κόμβο P του πλέγματος σε κάθε επανάληψη n $\left(\overrightarrow{\mathcal{R}}_P^n \right)$ γίνεται από τον ισολογισμό των αριθμητικών διανυσμάτων της ροής που εισέρχονται/εξέρχονται στον/από τον όγκο ελέγχου του P,

$$\overrightarrow{\mathcal{R}}_P^n = \sum_{Q \in nei(P)} \overrightarrow{\Phi}_{PQ}^n \tag{4.6}$$

όπου $\overrightarrow{\Phi}_{PQ}$ είναι το αριθμητικό διάνυσμα της ροής που διαπερνά την κοινή επιφάνεια των γειτονικών όγκων ελέγχου που σχηματίζονται γύρω από τους κόμβους P, Q.

Αντίστοιχα, τα μητρώα του αριστερού μέλους της 4.4 (D, Z) υπολογίζονται από τις σχέσεις

$$D_P^n = \sum_{Q \in nei(P)} \frac{\partial \overrightarrow{\Phi}_{PQ}^n}{\partial \overrightarrow{W}_P^n} \tag{4.7}$$

$$Z_Q^n = \frac{\partial \overrightarrow{\Phi}_{PQ}^n}{\partial \overrightarrow{W}_Q^n} \tag{4.8}$$

Στις σχέσεις 4.7, 4.8, P είναι πάντα ο 'κεντρικός' κόμβος (ο κόμβος δηλαδή γύρω από τον οποίο σχηματίζεται ο όγκος ελέγχου όπου γίνεται ο ισολογισμός των ροών $\overrightarrow{\Phi}_{PQ}$) και Q οι γείτονες του P.

Ο τρόπος υπολογισμού των ροών $\overrightarrow{\Phi}_{PQ}$ περιγράφεται αναλυτικά στο κεφάλαιο 2. Ο υπολογισμός των ροών αυτών γίνεται με βάση τα ροϊκά μεγέθη (μη-συντηρητικές μεταβλητές $\overrightarrow{U}$) και τις χωρικές παραγώγους αυτών (για την επίτευξη δεύτερης τάξης ακρίβειας στον χώρο) στους κόμβους P, Q και το κάθετο διάνυσμα ($\overrightarrow{n}_{PQ}$) στη διεπιφάνεια των όγκων ελέγχου των κόμβων P, Q. Δηλαδή,

$$\overrightarrow{\mathcal{R}}_P = \overrightarrow{\mathcal{R}}\left(\overrightarrow{n}_{PQ}, \overrightarrow{U}_P, \overrightarrow{U}_Q, \nabla\overrightarrow{U}_P, \nabla\overrightarrow{U}_Q\right) \tag{4.9}$$

$$D_P = D\left(\overrightarrow{n}_{PQ}, \overrightarrow{U}_P, \overrightarrow{U}_Q, \nabla\overrightarrow{U}_P, \nabla\overrightarrow{U}_Q\right) \tag{4.10}$$

$$Z_Q = Z\left(\overrightarrow{n}_{PQ}, \overrightarrow{U}_P, \overrightarrow{U}_Q, \nabla\overrightarrow{U}_P, \nabla\overrightarrow{U}_Q\right) \tag{4.11}$$

4.4.2 Προγραμματιστικές τεχνικές υπολογισμού των αριθμητικών διανυσμάτων ροής

Η ενότητα αυτή περιγράφει και συγκρίνει τις προγραμματιστικές τεχνικές που δοκιμάστηκαν για τον υπολογισμό των ροών $\overrightarrow{\Phi}$ με σκοπό το σχηματισμό του υπολοίπου $\overrightarrow{\mathcal{R}}$ και των μητρώων D, Z.

Η τεχνική που ακολουθείται στο CPU-κώδικα περιλαμβάνει τη σάρωση των ακμών του πλέγματος, τον υπολογισμό των αριθμητικών διανυσμάτων ανά ακμή και το 'μοίρασμα της πληροφορίας' (ισολογισμός ροών) στους δύο κόμβους κάθε ακμής. Η άμεση εφαρμογή της τεχνικής αυτής στη GPU συσχετίζει κάθε thread με μία ακμή του πλέγματος. Threads που εκτελούνται ταυτόχρονα μπορούν να υπολογίσουν, χωρίς κίνδυνο λάθους λόγω των ταυτόχρονων υπολογισμών, τα αριθμητικά διανύσματα της ροής που αντιστοιχούν στις ακμές με τις οποίες έχουν συσχετιστεί. Αντίθετα, η τεχνική αυτή δεν αποτρέπει τα threads να ανανεώνουν ταυτόχρονα-εσφαλμένα την τιμή ίδιων θέσεων μνήμης όταν εκείνα μοιράζουν πληροφορία στους κόμβους των ακμών που τους αντιστοιχούν. Συνεπώς, αυτή η 'CPU-τεχνική' δεν μπορεί να εφαρμοστεί άμεσα στη GPU.

Οι σχετικές 'GPU-τεχνικές' που δοκιμάστηκαν είναι οι:

- Απευθείας εφαρμογή της 'CPU-τεχνικής' με τη χρήση των atomic functions.
- Ομαδοποίηση των ακμών του πλέγματος και ταυτόχρονη επεξεργασία των ακμών μόνο μίας ομάδας από αυτές, με σκοπό την αποφυγή ταυτόχρονης αποθήκευσης δεδομένων στις μνήμες της GPU από περισσότερα του ενός threads.

- Συσχέτιση των κόμβων του πλέγματος με threads που υπολογίζουν και συγκεντρώνουν τις ροές $\overrightarrow{\Phi}$ που εισέρχονται στον όγκο ελέγχου του εκάστοτε κόμβου, για το σχηματισμό των $\overrightarrow{\mathcal{R}}$, D, Z. Απαιτείται ο διπλός υπολογισμός των ροών $\overrightarrow{\Phi}$, μία φορά για κάθε κόμβο της αντίστοιχης ακμής.
- Υπολογισμός και άθροιση στους όγκους ελέγχου των ροών $\overrightarrow{\Phi}$ από δύο διαφορετικά kernels τα οποία εκτελούνται στη σειρά. Απαιτείται η χρήση επιπλέον μνήμης.

Χρήση των atomic functions

Η τεχνική αυτή δοκιμάστηκε χρονικά τελευταία. Αποτελεί όμως την πιο άμεση εφαρμογή της 'CPU-τεχνικής', υπό την έννοια ότι κάθε ακμή του πλέγματος αντιστοιχεί σε ένα thread που υπολογίζει το αντίστοιχο διάνυσμα $\overrightarrow{\Phi}$ και μοιράζει πληροφορία στους κόμβους της ακμής και για αυτό παρουσιάζεται πρώτη. Ο λόγος που η τεχνική αυτή δοκιμάστηκε τελευταία είναι ότι οι atomic functions μπορούν να χειριστούν μεταβλητές διπλής ακρίβειας μόνο σε GPUs τελευταίας αρχιτεκτονικής. Όπως έχει ήδη όμως αναφερθεί, οι atomic functions επιβάλλουν επιπλέον συγχρονισμούς ανάμεσα στα threads, με σκοπό δύο ή περισσότερα threads να μην αποθηκεύουν πληροφορία ταυτόχρονα σε ίδιες θέσης μνήμης. Έτσι, η χρήση των atomic functions μειώνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα και, για το λόγο αυτό, η τεχνική αυτή εγκαταλείφθηκε.

Επεξεργασία ακμών κατά ομάδες

Οι ακμές του πλέγματος ομαδοποιήθηκαν έτσι ώστε διεργασίες που εκτελούνται ταυτόχρονα σε μία GPU να μην αποθηκεύουν σε ίδιες θέσεις μνήμης.

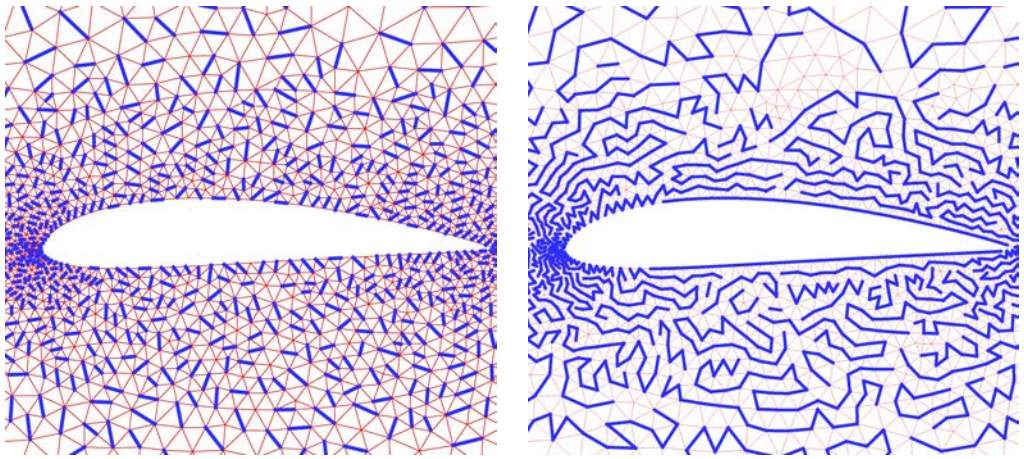
Μία απλή τεχνική ομαδοποίησης των ακμών ('edge1') υπαγορεύει οι ακμές της ίδιας ομάδας να μην έχουν κοινούς κόμβους (σχήμα 4.4(α′)). Έτσι, threads που συσχετίζονται με ακμές της ίδιας ομάδας, μπορούν να μοιράζουν πληροφορία ταυτόχρονα στους ακραίους κόμβους των ακμών με τις οποίες συσχετίζονται. Κάθε thread αντιστοιχεί σε μία ακμή μίας ομάδας, υπολογίζει το διάνυσμα $\overrightarrow{\Phi}$ που σχετίζεται με την ακμή αυτή και ανανεώνει το υπόλοιπο $\overrightarrow{\mathcal{R}}$ των κόμβων της. Επιπλέον, κάθε thread ανανεώνει το μητρώο D των κόμβων της προς επεξεργασία ακμής και αποθηκεύει το Z. Το σχήμα 4.5 παρουσιάζει σχηματικά το εν λόγω kernel. Το kernel που μόλις περιγράφηκε εκτελείται για κάθε ομάδα ακμών. Δηλαδή, ο αριθμός των κλήσεων του kernel είναι ίσος με τον αριθμό των ομάδων των ακμών. Οι διαδοχικές εκτελέσεις του kernel ανατίθενται στο ίδιο stream. Με τον τρόπο αυτό, δεν θα επεξεργαστούν οι ακμές της ομάδας 2 αν δεν έχει ολοκληρωθεί η επεξεργασία των ακμών της ομάδας 1 κ.ο.κ. Λόγω του τρόπου υλοποίησης της ομαδοποίησης και της χρήσης μη-δομημένων υπολογιστικών πλεγμάτων, το πλήθος των ακμών ανά ομάδα μειώνεται από την πρώτη στην τελευταία ομάδα.

Η παραπάνω τεχνική οδηγεί στο σχηματισμό τουλάχιστον τόσων ομάδων όσος

ο μέγιστος αριθμός ακμών που άγονται από έναν κόμβο. Όσο μεγαλύτερο είναι το πλήθος των ομάδων, τόσο μειώνεται η παράλληλη απόδοση του GPU-κώδικα, καθώς μειώνεται το ποσοστό των πολυεπεξεργαστών της κάρτας γραφικών που χρησιμοποιούνται ταυτόχρονα. Για παράδειγμα, ενδέχεται το πλήθος των ακμών των τελευταίων ομάδων να είναι μικρότερο του συνόλου των threads που μπορεί να χειρίζεται η κάρτα γραφικών ταυτόχρονα σε όλους τους πολυεπεξεργαστές αυτής.

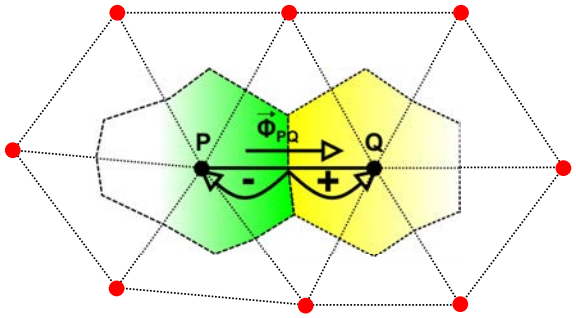
Μία παραλλαγή ('edge2') της παραπάνω τεχνικής η οποία δοκιμάστηκε με σκοπό τη μείωση των ομάδων (άρα και του αριθμού κλήσεων του παραπάνω kernel) είναι ο σχηματισμός ομάδων από μη-αναδιπλωμένες 'λωρίδες' ακμών. Κάθε 'λωρίδα' αποτελείται το πολύ από 32 συνεχόμενες ακμές. Κάθε κόμβος του πλέγματος ανήκει το πολύ σε μία λωρίδα ανά ομάδα λωρίδων. Στο σχήμα 4.4(β′) φαίνονται οι ακμές μίας τυχαίας ομάδας λωρίδων. Η ιδέα της τεχνικής αυτής στηρίζεται στο γεγονός ότι τα threads εκτελούνται στη GPU σε ομάδες των 32 (warps). Αυτό σημαίνει ότι τα threads του ίδιου warp (άρα και της ίδιας λωρίδας ακμών) δίνουν πληροφορία ταυτόχρονα στον 'αριστερό' κόμβο της ακμής που επεξεργάζονται και στη συνέχεια στον 'δεξιό', αποτρέποντας με τον τρόπο αυτό την ταυτόχρονη αποθήκευση στην ίδια θέση μνήμης. Δυστυχώς, όπως αναμένονταν, εξαιτίας της χρήσης μη-δομημένων πλεγμάτων προέκυψαν και λωρίδες με λιγότερες από 32 ακμές.

Το σχήμα 4.5 περιγράφει σχηματικά τον τρόπο δράσης των δύο τεχνικών ομαδοποίησης των ακμών του πλέγματος που παρουσιάστηκαν προηγουμένως.



(α′) edge1 (β′) edge2

Σχήμα 4.4: Ομαδοποίηση των ακμών του πλέγματος. Με έντονο χρώμα φαίνονται οι ακμές μίας ομάδας ενώ με μη-έντονο το συνολικό υπολογιστικό πλέγμα. Το σχήμα (α′) παρουσιάζει μία ομάδα μη-συνδεδεμένων ακμών (τεχνική ομαδοποίησης ακμών 'edge1'), ενώ το σχήμα (β′) μία ομάδα από μη-αναδιπλωμένες 'λωρίδες' ακμών που δεν συνδέονται μεταξύ τους (τεχνική ομαδοποίησης ακμών 'edge2'). Μία 'λωρίδα' περιέχει έως 32 ακμές. Εξαιτίας της χρήσης μη-δομημένων υπολογιστικών πλεγμάτων, αναπόφευκτα προκύπτουν και λωρίδες με μη-συνεχόμενες ακμές.



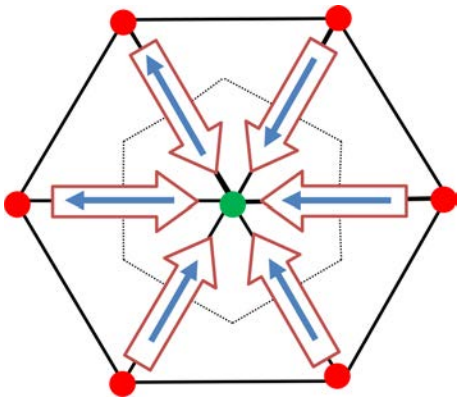
Σχήμα 4.5: Υπολογισμός του διανύσματος $\overrightarrow{\Phi}_{PQ}$ της ροής που διαπερνά τη διεπιφάνεια των όγκων ελέγχου που σχηματίζονται γύρω από τους κόμβους P, Q μίας ακμής του πλέγματος. Κάθε thread χειρίζεται μία ακμή μίας ομάδας ακμών του πλέγματος, υπολογίζει το διάνυσμα της ροής $\overrightarrow{\Phi}_{PQ}$ και μοιράζει την πληροφορία στους κόμβους P και Q της ακμής. Επειδή τα threads που εκτελούνται ταυτόχρονα χειρίζονται ακμές αποκλειστικά της ίδιας ομάδας, αποτρέπεται η ταυτόχρονη αποθήκευση σε ίδιες θέσεις μνήμης της GPU.

Επιπλέον αριθμητικοί τρόποι ισολογισμού των ροών $\overrightarrow{\Phi}$

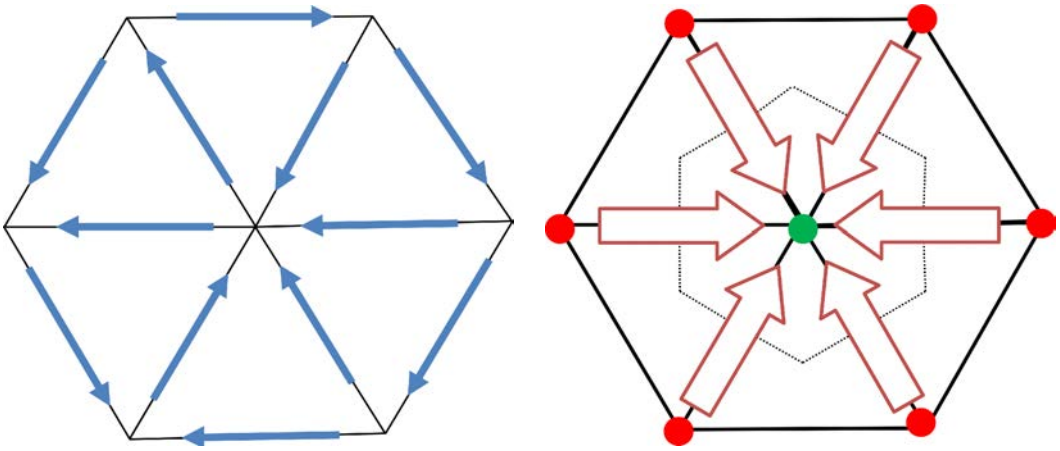
Πέρα των υπολογιστικών τεχνικών που αναφέρθηκαν, δοκιμάστηκαν δύο επιπλέον τεχνικές η δομή των οποίων διαφοροποιείται σημαντικά σε σχέση με τη CPU-τεχνική τόσο σε επίπεδο σάρωσης των στοιχείων του πλέγματος (κόμβους, ακμές κ.α.), όσο και στον τρόπο πρόσβασης στην κεντρική μνήμη.

Η πρώτη τεχνική (σχήμα 4.6) χρησιμοποιεί ένα kernel. Για το λόγο αυτό, στη συνέχεια του κειμένου θα αναφέρεται ως η τεχνική ενός kernel (One-kernel scheme). Το kernel αυτό αντιστοιχεί τους κόμβους του πλέγματος στα threads που εκτελούνται στη GPU. Κάθε thread σαρώνει τις ακμές που άγονται από τον κόμβο που χειρίζεται και υπολογίζει τις ροές $\overrightarrow{\Phi}$ που εισέρχονται στον όγκο ελέγχου του κόμβου. Το ίδιο kernel κάνει τον ισολογισμό των ροών (που υπολόγισε) για το σχηματισμό του υπολοίπου $\overrightarrow{\mathcal{R}}$ και κατασκευάζει τα μητρώα D, Z ανά κόμβο.

Αντίθετα, η δεύτερη τεχνική χρησιμοποιεί δύο kernels ('τεχνική των δύο kernels', Two-kernel scheme). Το πρώτο kernel μοιράζει τις ακμές του πλέγματος σε threads που υπολογίζουν και αποθηκεύουν στην κεντρική μνήμη της GPU τις ροές $\overrightarrow{\Phi}$ και τις παραγώγους αυτών ως προς τις μεταβλητές της ροής (ανά ακμή). Το δεύτερο kernel κάνει τον ισολογισμό των ροών (που υπολόγισε το πρώτο) για τον σχηματισμό του υπολοίπου $\overrightarrow{\mathcal{R}}$ και κατασκευάζει-αποθηκεύει τα μητρώα D, Z στους κόμβους του πλέγματος. Σχηματική αναπαράσταση της τεχνικής των δύο kernels δίνεται στο σχήμα 4.7.



Σχήμα 4.6: Υπολογισμός των μητρώων D, Z και του υπολοίπου $\overrightarrow{\mathcal{R}}$ με τη χρήση ενός kernel (τεχνική ενός kernel, One-kernel scheme). Το kernel αντιστοιχεί τους κόμβους του πλέγματος σε threads τα οποία σαρώνουν τις ακμές που άγονται από τους κόμβους που χειρίζονται, υπολογίζουν τις ροές $\overrightarrow{\Phi}$ και κατασκευάζουν τα μητρώα D, Z και το υπόλοιπο $\overrightarrow{\mathcal{R}}$ ανά κόμβο. Τα λεπτά βέλη υποδηλώνουν υπολογισμό αριθμητικών ροών ενώ τα παχιά συγκέντρωση αυτής.



(α′) 1^o kernel (β′) 2^o kernel

Σχήμα 4.7: Υπολογισμός των μητρώων D, Z και του υπολοίπου $\overrightarrow{\mathcal{R}}$ με τη χρήση δύο kernels (τεχνική με δύο kernels, Two-kernel scheme). Το πρώτο kernel (αριστερό σχήμα) σαρώνει τις ακμές του πλέγματος, υπολογίζει τα αριθμητικά διανύσματα $\overrightarrow{\Phi}$ και αποθηκεύει πληροφορία, που θα χρησιμοποιηθεί στη συνέχεια, στην κεντρική μνήμη της GPU με βάση τον αύξοντα αριθμό της ακμής που χειρίζεται το αντίστοιχο thread. Το δεύτερο kernel (δεξιό σχήμα) διαβάζει-συγκεντρώνει ότι υπολόγισε το πρώτο και σχηματίζει τα μητρώα D, Z και το υπόλοιπο $\overrightarrow{\mathcal{R}}$ ανά κόμβο του πλέγματος. Το πρώτο kernel αντιστοιχεί τις ακμές, ενώ το δεύτερο του κόμβους του πλέγματος στα threads που εκτελούνται στη GPU. Τα λεπτά (αριστερά) βέλη υποδηλώνουν υπολογισμό αριθμητικών ροών, ενώ τα παχιά (δεξιά) συγκέντρωση/άθροιση αυτών.

Αξιολόγηση των τεχνικών ισολογισμού των ροών $\overrightarrow{\Phi}$

Οι τεχνικές που παρουσιάστηκαν στις προηγούμενες παραγράφους δοκιμάστηκαν σε μη-δομημένα υπολογιστικά πλέγματα, με διαφορετικό πλήθος κόμβων, γύρω από μεμονωμένη αεροτομή και αξιολογήθηκαν με βάση το χρόνο υπολογισμού του υπολοίπου $\overrightarrow{\mathcal{R}}$ των διακριτοποιημένων εξισώσεων της ροής και της κατασκευής των μητρώων D, Z σε ένα ψευδοχρονικό βήμα. Τα σχήματα 4.8(α΄), 4.8(β΄), 4.8(γ΄) παρουσιάζουν τους σχετικούς χρόνους εκτέλεσης στα τρία μοντέλα καρτών γραφικών που χρησιμοποιήθηκαν στη διατριβή. Οι χρόνοι εκτέλεσης που παρουσιάζονται είναι ανηγμένοι ως προς το μεγαλύτερο χρόνο εκτέλεσης που καταγράφηκε. Έτσι, ο χρόνος υπολογισμού του $\overrightarrow{\mathcal{R}}$ και των D, Z ενός ψευδοχρονικού βήματος χρησιμοποιώντας την πρώτη τεχνική ομαδοποίησης των ακμών (edge1) σε μία GTX 280 (GPU χαμηλότερων υπολογιστικών δυνατοτήτων σε σχέση με τις υπόλοιπες δύο που χρησιμοποιήθηκαν) και το πυκνότερο πλέγμα θεωρείται ίσος με 1. Η απόδοση της χρήσης των atomic functions δεν έχει τυπωθεί, καθώς υστερούσε σημαντικά σε σχέση με εκείνη των υπολοίπων τεχνικών. Εξάλλου, μπορεί να εφαρμοστεί μόνο σε GPUs τελευταίας αρχιτεκτονικής.Τα πλέγματα που χρησιμοποιήθηκαν κατασκευάστηκαν με την τεχνική του προελαύνοντος μετώπου (advancing front). Όλοι οι υπολογισμοί έγιναν χρησιμοποιώντας αριθμητική διπλής ακρίβειας.

Όπως φαίνεται από τα σχήματα 4.8(α΄), 4.8(β΄), 4.8(γ΄) οι τεχνικές ενός και δύο kernel έχουν σαφώς υψηλότερη απόδοση (μικρότερος χρόνος εκτέλεσης) σε σχέση με τις τεχνικές ομαδοποίησης των ακμών.

Συγκεκριμένα, στις GPUs αρχιτεκτονικής GT200 (GTX 280 και 285, σχήματα 4.8(α΄), 4.8(β΄) αντίστοιχα), η απόδοση των δύο τεχνικών ομαδοποίησης των ακμών του πλέγματος είναι σχεδόν η ίδια, με την τεχνική ομαδοποίησης συνεχόμενων ακμών (edge2) να έχει ελαφρώς αυξημένη απόδοση. Όμοια, η τεχνική των δύο kernels έχει ελαφρώς υψηλότερη απόδοση σε σχέση με την τεχνική του ενός kernel. Οι δύο τεχνικές του ενός ή των δύο kernels δίνουν χρόνους εκτέλεσης περίπου 40% (GTX 280) ή 35% (GTX 285) μικρότερους σε σχέση με εκείνους των τεχνικών ομαδοποίησης των ακμών.

Αντίθετα, στις GPUs τελευταίας αρχιτεκτονικής (σχήμα 4.8(γ΄)), η τεχνική των δύο kernels έχει σαφώς την υψηλότερη απόδοση.

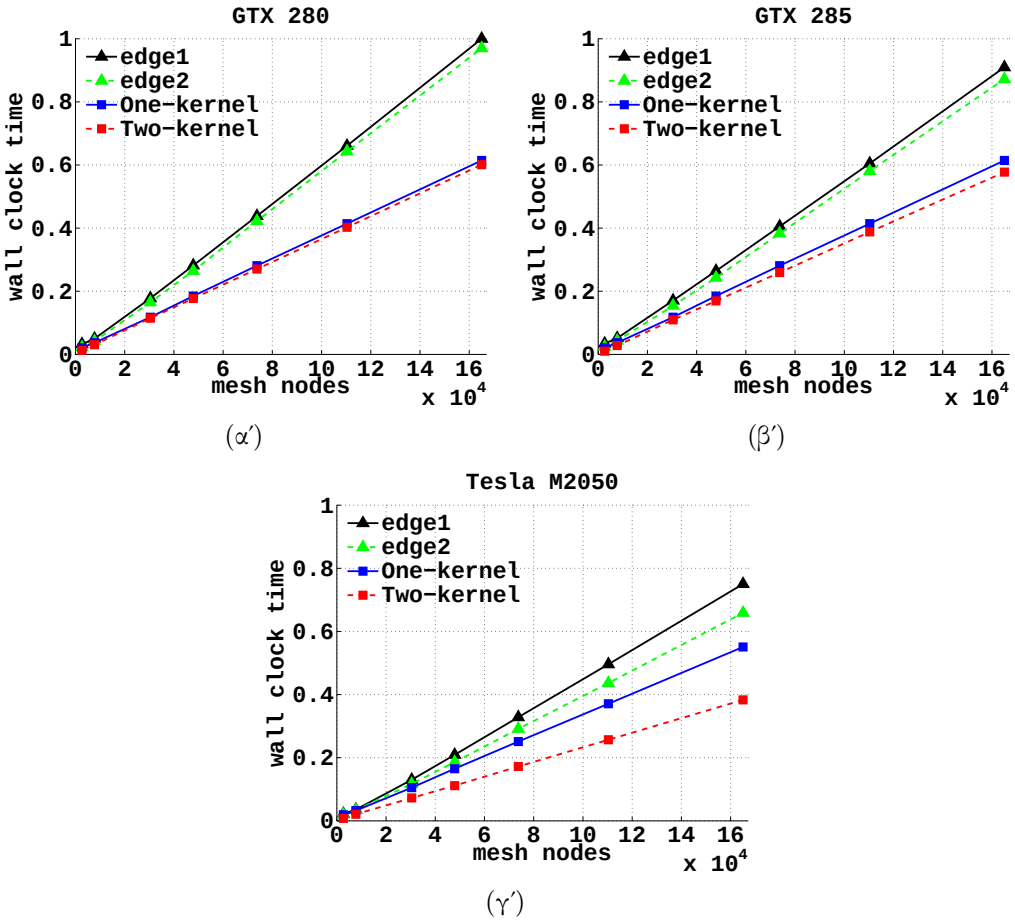
Όπως αναμενόταν, οι χρόνοι εκτέλεσης σε μία GPU τελευταίας αρχιτεκτονικής είναι αισθητά χαμηλότεροι σε σχέση με τους αντίστοιχους σε κάρτες της προηγούμενης αρχιτεκτονικής. Συγκεκριμένα, χρησιμοποιώντας την τεχνική των δύο kernels σε μία GPU τελευταίας αρχιτεκτονικής, ο υπολογισμών των $\overrightarrow{\mathcal{R}}$, D, Z γίνεται περίπου στο 65% του χρόνου υπολογισμού σε μία GPU της προηγούμενης αρχιτεκτονικής.

Ορισμένα από τα βασικά πλεονεκτήματα των GPUs τελευταίας αρχιτεκτονικής σε σχέση με GPUs της προηγούμενης αρχιτεκτονικής είναι:

(*α΄*) Η υψηλότερη ταχύτητα εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών διπλής ακρίβειας.

(*β΄*) Η προσπέλαση της κεντρικής μνήμης είναι cached.

(*γ΄*) Η οργάνωση των προσβάσεων στην κεντρική μνήμη από τους warp schedulers γίνεται σε επίπεδο warps και όχι half-warps (παράγραφος 3.8.1).



Σχήμα 4.8: Αδιάστατος χρόνος υπολογισμού του υπολοίπου των διακριτοποιημένων εξισώσεων της ροής $\overrightarrow{\mathcal{R}}$ και των μητρώων D, Z σε μη-δομημένα πλέγματα με διαφορετικό πλήθος κόμβων για ένα ψευδοχρονικό βήμα χρησιμοποιώντας τις τεχνικές ομαδοποίησης των ακμών και εκείνες ενός και δύο kernel και διαφορετικές GPUs. Οι υπολογισμοί χρησιμοποιούν αριθμητική διπλής ακρίβειας.

Το (α) ευνοεί κυρίως kernels με υψηλό λόγο συνόλου αριθμητικών πράξεων προς σύνολο προσβάσεων στην κεντρική μνήμη. Τέτοιο είναι το kernel υπολογισμού των ροών $\overrightarrow{\Phi}$ (πρώτο) της τεχνικής με τα δύο kernels. Από τα (β),(γ) ευνοούνται κυρίως kernels που χαρακτηρίζονται από τη μη-δομημένη προσπέλαση της κεντρικής μνήμης της GPU. Τέτοιου τύπου kernels είναι το kernel ισολογισμού των ροών $\overrightarrow{\Phi}$, δηλαδή το δεύτερο kernel της τεχνικής των δύο kernels, όπως επίσης και το kernel των τεχνικών ομαδοποίησης των ακμών του πλέγματος. Για το λόγο αυτό, η απόδοση των τεχνικών αυτών αυξάνει σημαντικά κατά τη μετάβαση από GPUs της προηγούμενης σε αυτές της τελευταίας αρχιτεκτονικής.

Αντίθετα, η απόδοση του kernel υπολογισμού και ισολογισμού των ροών $\overrightarrow{\Phi}$ (δηλαδή εκείνο της τεχνικής ενός kernel) εξαρτάται κατά κύριο λόγο από τη μέγιστη ταχύτητα προσπέλασης της κεντρικής μνήμης της κάρτας γραφικών, η οποία είναι περίπου ίδια

σε GPUs της προηγούμενης και τελευταίας αρχιτεκτονικής. Για το λόγο αυτό, παρατηρείται ελάχιστη αύξηση (πολύ μικρή συγκριτικά με εκείνη των υπόλοιπων τεχνικών) στην απόδοση της τεχνικής ενός kernel κατά τη μετάβαση σε GPUs τελευταίας αρχιτεκτονικής. Το ότι η ταχύτητα εκτέλεσης του kernel αυτού εξαρτάται κατά κύριο λόγο από τη μέγιστη ταχύτητα προσπέλασης της μνήμης αποτελεί μία επιπλέον απόδειξη ότι ο τρόπος διαχείρισης της κεντρικής μνήμης της GPU από το εν λόγω kernel είναι ο βέλτιστος. Έτσι, ο τρόπος αυτός χρησιμοποιείται και από το kernel που ανανεώνει τις μεταβλητές της ροής με τη μέθοδο Jacobi και καλείται μετά το σχηματισμό των $\overrightarrow{\mathcal{R}}$, D, Z. Αυτός ο τρόπος διαχείρισης της μνήμης (αποθήκευσης δηλαδή των $\overrightarrow{\mathcal{R}}$, D, Z) περιγράφεται στην παράγραφο 4.4.3.

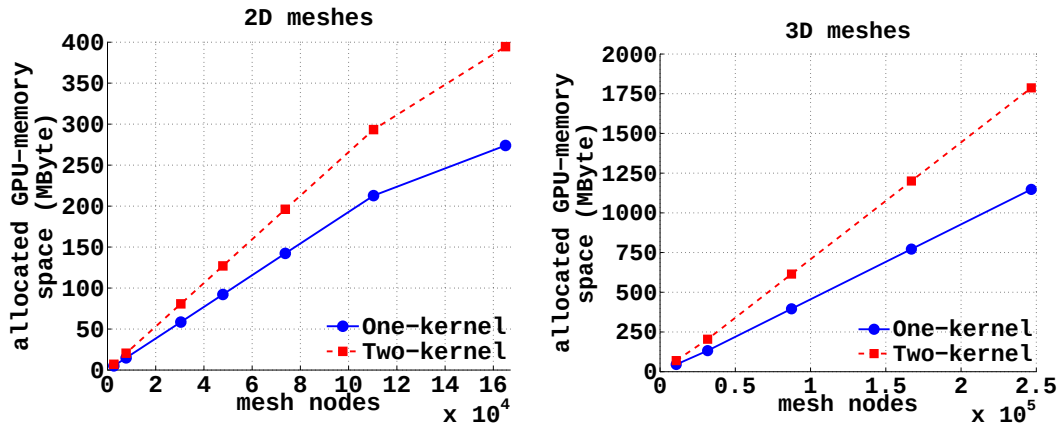
Συγκρίνοντας τις τεχνικές ενός και δύο kernels παρατηρείται ότι η πρώτη τεχνική (ενός kernel) εκτελεί διπλό υπολογισμό των ροών $\overrightarrow{\Phi}$ ανά ακμή του πλέγματος (μία φορά για κάθε κόμβο της ακμής). Αυτό αποφεύγεται στην τεχνική των δύο kernels με το κόστος δέσμευσης και χρήσης περισσότερης μνήμης, αφού οι ροές $\overrightarrow{\Phi}$ αρχικά υπολογίζονται από το πρώτο kernel και αθροίζονται, στη συνέχεια, από το δεύτερο. Αυξάνεται, έτσι ο αριθμός προσβάσεων στην αργή κεντρική μνήμη. Έμμεσα δηλαδή τα σχήματα 4.8 δίνουν την απάντηση στην ερώτηση: τι είναι αποδοτικότερο; Ο διπλός υπολογισμός ορισμένων δεδομένων ή η προσωρινή αποθήκευση αυτών που συνεπάγεται και περισσότερες προσβάσεις στην αργή κεντρική μνήμη. Φαίνεται ότι η απόδοση των δύο τεχνικών είναι περίπου ίδια σε GPUs της προηγούμενης αρχιτεκτονικής, ενώ υπερέχει η τεχνική χρήσης επιπλέον μνήμης σε GPUs τελευταίας αρχιτεκτονικής, όπου η χωρητικότητα των cache μνημών είναι μεγαλύτερη και η οργάνωση των προσβάσεων στην κεντρική μνήμη από τους warp schedulers είναι αποδοτικότερη.

Το σχήμα 4.9 δείχνει την έκταση της κεντρικής μνήμης που δεσμεύει ο GPU-επιλύτης στην επίλυση 2Δ και 3Δ ατριβών ροών σε συνάρτηση με το πλήθος των κόμβων του υπολογιστικού πλέγματος. Φαίνεται ότι ο GPU-κώδικας με την τεχνική των δύο kernels χρησιμοποιεί περίπου 45% περισσότερη μνήμη σε σχέση με τη συνιστώσα του GPU-κώδικα που στηρίζεται στην τεχνική του ενός kernel.

Επειδή οι τεχνικές ενός και δύο kernels έχουν υψηλότερη απόδοση σε σχέση με εκείνες ομαδοποίησης των ακμών, η ανάπτυξη του GPU-επιλύτη στηρίχθηκε στις τεχνικές αυτές. Για το λόγο αυτό, η συνέχεια του κειμένου αναφέρεται αποκλειστικά στις τεχνικές ενός και δύο kernels.

4.4.3 Προσπέλαση της κεντρικής μνήμης

Έχει αναφερθεί πολλές φορές ότι η απόδοση οποιουδήποτε GPU-κώδικα είναι άρρηκτα συνδεδεμένη με την απόδοση του τρόπου προσπέλασης της κεντρικής μνήμης της GPU εξαιτίας της μικρής χωρητικότητας των cache μνημών αυτής. Για παράδειγμα, περίπου το 75% του χρόνου υπολογισμού των $\overrightarrow{\mathcal{R}}$, D, Z σε μία Tesla M2050 είναι ο χρόνος ισολογισμού των ήδη υπολογισμένων ροών $\overrightarrow{\Phi}$ και μόλις 25% ο χρόνος υπολογισμού αυτών, χρησιμοποιώντας την τεχνική των δύο kernels στην πρόλεξη ατριβούς ροής. Στην περίπτωση επίλυσης συνεκτικών ροών, επειδή αυξάνει ο όγκος των αριθμη-



(α′) (β′)

Σχήμα 4.9: Διάσταση σε MByte της κεντρικής μνήμης που δεσμεύεται από το GPU-κώδικα όταν χρησιμοποιείται η τεχνική ενός ή δύο kernels σε συνάρτηση με το πλήθος των κόμβων του πλέγματος που χρησιμοποιείται στην επίλυση 2Δ και 3Δ ατριβών ροών.

τικών πράξεων για τον υπολογισμό των ροών $\overrightarrow{\Phi}$, τα αντίστοιχα ποσοστά είναι 45% για τον υπολογισμό και 55% για τον ισολογισμό των ροών $\overrightarrow{\Phi}$. Ακόμα και στην περίπτωση αυτή, όμως, ο χρόνος του ισολογισμού των ροών ως προς εκείνο του υπολογισμού τους είναι υψηλός. Αυτό δείχνει τη σημασία της σωστής διαχείρισης της κεντρικής μνήμης της GPU.

Στη συνέχεια, παρουσιάζεται ο τρόπος αποθήκευσης των μητρώων D, Z. Ταυτόχρονα γίνεται η σύγκριση του 'CPU' (σειριακού) τρόπου αποθήκευσης αυτών με τον προτεινόμενο-βέλτιστο 'GPU' (παράλληλο) τρόπο αποθήκευσης σε όρους χρόνου προσπέλασης της κεντρικής μνήμης της GPU. Συγχρόνως, αποδεικνύεται γιατί η απευθείας μεταφορά ενός CPU-κώδικα στη GPU, απλά ξαναγράφοντας τον πρώτο στη γλώσσα που υποστηρίζει το εκάστοτε περιβάλλον προγραμματισμού, οδηγεί σε GPU-λογισμικό αρκετά χαμηλής παράλληλης απόδοσης. Για το λόγο αυτό, χρειάζεται η πλήρης αναδόμηση του 'αρχικού' CPU-κώδικα σύμφωνα με την παράλληλη λογική στην οποία βασίζεται και λειτουργεί μία σημερινή GPU. Σημειώνεται ότι τα μητρώα D, Z, δηλαδή τα μητρώα συντελεστών του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής, καταναλώνουν περίπου το 70% του δεσμευμένου χώρου μνήμης της GPU. Συνεπώς, η απόδοση του GPU-κώδικα της διατριβής εξαρτάται κατά μεγάλο ποσοστό από το χρόνο προσπέλασης των εν λόγω μητρώων.

Αποθήκευση/ανάγνωση των διαγώνιων στοιχείων των συντελεστών του αριστερού μέλους (D)

Σε κάθε κόμβο αντιστοιχεί ένα μητρώο D το οποίο υπολογίζεται σύμφωνα με την έκφραση 4.7. Κατά την εκτέλεση του CPU-επιλύτη η αποθήκευση του μητρώου D γίνεται ανά κόμβο του πλέγματος, όπως φαίνεται στο σχήμα 4.10. Ο τρόπος αυτός αποθήκευσης αντιτίθεται στην παράλληλη λογική της GPU και, έτσι, οδηγεί σε GPU-

κώδικα χαμηλής παράλληλης απόδοσης.

Όπως έχει ήδη αναφερθεί, κατά την εκτέλεση ενός GPU-κώδικα, πολλές ομάδες των 32 threads (warps) εκτελούνται ασύγχρονα στους διαθέσιμους πολυεπεξεργαστές της κάρτας γραφικών. Τα 32 threads που απαρτίζουν ένα warp εκτελούνται ταυτόχρονα, δηλαδή τα 32 threads έχουν ταυτόχρονη πρόσβαση στην κεντρική μνήμη της GPU. Ο δίαυλος μεταφοράς δεδομένων από την κεντρική μνήμη στους πολυεπεξεργαστές μπορεί να μεταφέρει έως και 128 Bytes[1] συνεχούς τμήματος της κεντρικής μνήμης της GPU. Συνεπώς, η ελαχιστοποίηση του χρόνου προσπέλασης της κεντρικής μνήμης επιτυγχάνεται όταν οι θέσεις μνήμης που έχουν πρόσβαση τα 32 threads του warp βρίσκονται στο ελάχιστο πλήθος από 128Byte τμήματα μνήμης (για περισσότερα σχετικά με την προσπέλαση της κεντρικής μνήμη καρτών γραφικών της προηγούμενης και τελευταίας αρχιτεκτονικής ο αναγνώστης παραπέμπεται στην παράγραφο 3.8.1). Για το λόγο αυτό, ο βέλτιστος τρόπος αποθήκευσης των στοιχείων του μητρώου D είναι ανά στοιχείο πίνακα και όχι ανά κόμβο, όπως γίνεται στον CPU-κώδικα. Το σχήμα 4.11 παρουσιάζει τον τρόπο αποθήκευσης των μητρώων D των πρώτων 32 κόμβων του πλέγματος (ο αύξων αριθμός του πρώτου κόμβου του πλέγματος είναι το 0). Έτσι αποθηκεύονται στις πρώτες 32 θέσεις μνήμης τα πρώτα στοιχεία (0,0) των μητρώων D των πρώτων 32 κόμβων. Ακολουθούν τα δεύτερα στοιχεία (0,1) κ.ο.κ. Τα μητρώα D των υπόλοιπων κόμβων αποθηκεύονται, στη συνέχεια, με τον ίδιο ακριβώς τρόπο. Στη γενική περίπτωση που το πλήθος των κόμβων του πλέγματος δεν είναι πολλαπλάσιο του 32, χρησιμοποιούνται τεχνητά επιπλέον θέσεις μνήμης όσες θα αναλογούσαν στους επιπλέον κόμβους έως ότου το πλήθος των κόμβων του πλέγματος γίνει πολλαπλάσιο του 32. Η χρήση επιπλέον θέσεων μνήμης εξασφαλίζει ότι και το τελευταίο warp ακολουθεί τον ίδιο τρόπο αποθήκευσης/ανάγνωσης των μητρών D των κόμβων που χειρίζεται. Τονίζεται ότι δεν προστίθενται επιπλέον κόμβοι (τόσοι ώστε το σύνολο των κόμβων να γίνει πολλαπλάσιο του 32) στο πλέγμα, δεσμεύεται όμως ελαφρώς περισσότερη μνήμη.

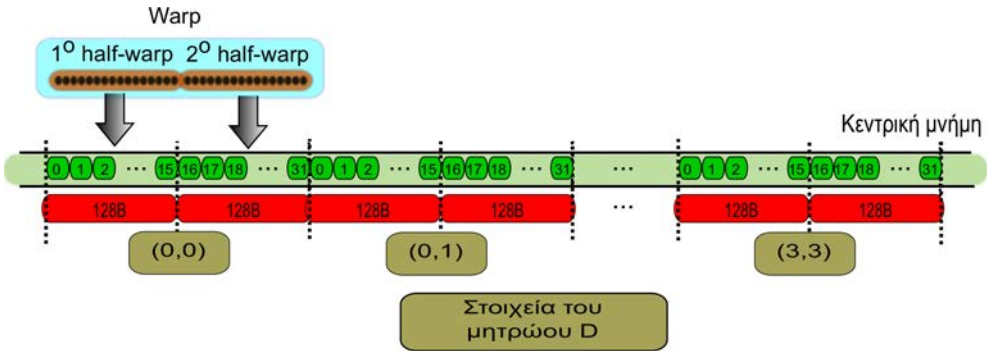
Η ταχύτητα μεταφοράς δεδομένων από την κεντρική μνήμη στους πολυεπεξεργαστές και ανάποδα, όταν χρησιμοποιείται ο προτεινόμενος τρόπος προσπέλασης των στοιχείων των μητρώων D, σε μία GPU αρχιτεκτονικής GT200, είναι περίπου 120 GByte/sec. Η ταχύτητα αυτή είναι πολύ κοντά στη μέγιστη ταχύτητα μεταφοράς δεδομένων της ίδιας κάρτας, που είναι 123 GByte/sec, όπως μετρήθηκε χρησιμοποιώντας ως εργαλείο έτοιμο λογισμικό από την ιστοσελίδα της NVIDIA [132]. Αντίθετα, αν χρησιμοποιηθεί ο CPU-τρόπος αποθήκευσης, η ταχύτητα μεταφοράς δεδομένων είναι περίπου μόλις 15 GByte/sec. Το σχήμα 4.12 παρουσιάζει την προσπέλαση της κεντρικής μνήμης από τα threads του πρώτου warp αν χρησιμοποιούνταν ο CPU-τρόπος αποθήκευσης. Φαίνεται στο σχήμα αυτό, ότι οι προς πρόσβαση θέσεις μνήμης είναι αρκετά 'απομακρυσμένες' και μάλιστα ανήκουν σε διαφορετικά τμήματα των 128 Bytes. Αυτό αυξάνει το πλήθος των προσβάσεων στην κεντρική μνήμη, και όπως μετρήθηκε μειώνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα. Τονίζεται, ότι οι μετρήσεις της ταχύτητας μεταφοράς δεδομένων που παρουσιάστηκαν νωρίτερα έγινε κατά την επίλυση 2Δ προβλημάτων όπου η διάσταση των μητρώων D είναι 4×4. Κα-

[1] Ένα τμήμα μνήμης από 128 Bytes περιέχει 16 αριθμούς διπλής ακρίβειας.

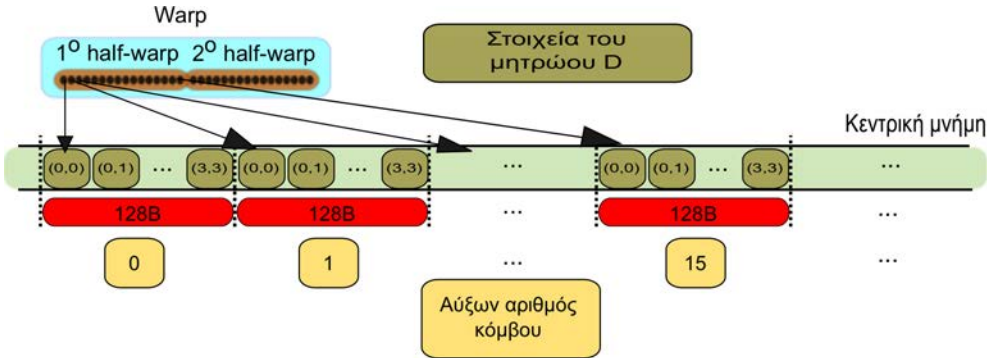
τά την επίλυση 3Δ προβλημάτων, η ταχύτητα μεταφοράς δεδομένων χρησιμοποιώντας τον προτεινόμενο τρόπο αποθήκευσης παραμένει περίπου ίση με 120 GByte/sec ενώ χρησιμοποιώντας τον CPU-τρόπο μειώνεται κάτω των 10 GByte/sec. Αυτό συμβαίνει επειδή η διάσταση των μητρώων D, για το συμπιεστό ρευστό, είναι 5×5. Συνεπώς, αν χρησιμοποιούνταν ο CPU-τρόπος, η 'απόσταση' των προς πρόσβαση θέσεων μνήμης γίνεται ακόμα μεγαλύτερη.

Στις δημοσιεύσεις [57, 58] που σχετίζονται με την παρούσα διατριβή, είχε προταθεί η αποθήκευση των στοιχείων των μητρώων D σύμφωνα με το πλήθος των threads ανά block και όχι των threads ανά warp, όπως περιγράφηκε προηγουμένως. Η απόδοση των δύο τεχνικών είναι ταυτόσημη. Η αποθήκευση σύμφωνα με το πλήθος των threads στο warp δίνει μεγαλύτερη ευελιξία στον προγραμματιστή και, για αυτό, ο τελικός GPU-κώδικας χρησιμοποιεί την τεχνική που περιγράφηκε προηγουμένως. Επιπλέον, η αποθήκευση σύμφωνα με το πλήθος των threads στο warp εξοικονομεί λίγη μνήμη σε σχέση με την αποθήκευση σύμφωνα με το πλήθος των threads στο block.

Σχήμα 4.10: Τρόπος αποθήκευσης/ανάγνωσης των στοιχείων των μητρώων D των κόμβων του πλέγματος στον CPU-επιλύτη. Η αποθήκευση/ανάγνωση γίνεται ανά κόμβο. Το παράδειγμα του σχήματος αναφέρεται στην επίλυση ενός 2Δ προβλήματος συμπιεστού ρευστού, για αυτό και η διάσταση των μητρώων D είναι 4×4. Ο ίδιος τρόπος αποθήκευσης/ανάγνωσης γενικεύεται εύκολα και χρησιμοποιείται στην επίλυση και 3Δ προβλημάτων, όπου η διάσταση των μητρώων D είναι 5×5.



Σχήμα 4.11: Τρόπος αποθήκευσης/ανάγνωσης των στοιχείων των μητρώων D των κόμβων του πλέγματος στον GPU-επιλύτη. Η αποθήκευση/ανάγνωση γίνεται ανά στοιχείο. Το σχήμα παρουσιάζει τον τρόπο αποθήκευσης των στοιχείων των μητρώων D των πρώτων 32 κόμβων του πλέγματος. Ο αύξων αριθμός του πρώτου κόμβου θεωρείται το 0. Το παράδειγμα του σχήματος χρησιμοποιεί αριθμητική διπλής ακρίβειας. Δηλαδή, τα 32 threads του warp έχουν ταυτόχρονη πρόσβαση σε 32 συνεχόμενες θέσεις μνήμης για πραγματικούς αριθμούς διπλής ακρίβειας συνολικού μήκους 32×8Bytes/πραγματικό αριθμό, ή 2 τμήματα των 128 Bytes της κεντρικής μνήμης. Επίσης, το παράδειγμα του σχήματος αναφέρεται στην επίλυση ενός 2Δ προβλήματος συμπιεστού ρευστού, για αυτό και η διάσταση των μητρώων D είναι 4×4. Ο ίδιος τρόπος αποθήκευσης/ανάγνωσης γενικεύεται εύκολα και χρησιμοποιείται στην επίλυση και 3Δ προβλημάτων, όπου η διάσταση των μητρώων D είναι 5×5.



Σχήμα 4.12: Προσπέλαση της κεντρικής μνήμης της GPU αν χρησιμοποιούνταν ο CPU-τρόπος αποθήκευσης των μητρώων D. Φαίνεται ότι οι προς πρόσβαση θέσεις μνήμης είναι αρκετά απομακρυσμένες και, μάλιστα, ανήκουν σε διαφορετικά τμήματα των 128 Bytes. Αυτό αυξάνει το πλήθος των προσβάσεων στην κεντρική μνήμη και, κατά συνέπεια, μειώνει δραματικά την παράλληλη απόδοση του GPU-κώδικα. Το παράδειγμα του σχήματος αναφέρεται στην πρόλεξη 2Δ συμπιεστών ροών. Στην επίλυση 3Δ προβλημάτων η χρήση του CPU-τρόπου αποθήκευσης μειώνει ακόμα περισσότερο την απόδοση του GPU-κώδικα καθώς αυξάνονται οι αποστάσεις μεταξύ των προς πρόσβαση θέσεων μνήμης.

Αποθήκευση/ανάγνωση των μη-διαγώνιων στοιχείων των συντελεστών του αριστερού μέλους (Z)

Το πλήθος των κόμβων Q που ενώνονται μέσω ακμής του πλέγματος με τον υπόψη κόμβο P ορίζει το πλήθος των μητρώων Z_Q που αντιστοιχούν στον κόμβο P. Ο υπολογισμός των μητρώων αυτών γίνεται από τη σχέση 4.8. Η αποθήκευση των μητρώων Z στον CPU-κώδικα γίνεται με βάση την αρίθμηση των ακμών του πλέγματος. Συνεπώς, ο τρόπος αποθήκευσης των μητρώων Z στον CPU-κώδικα είναι όμοιος με εκείνον των μητρώων D με τη διαφορά ότι γίνεται ανά ακμή και όχι ανά κόμβο.

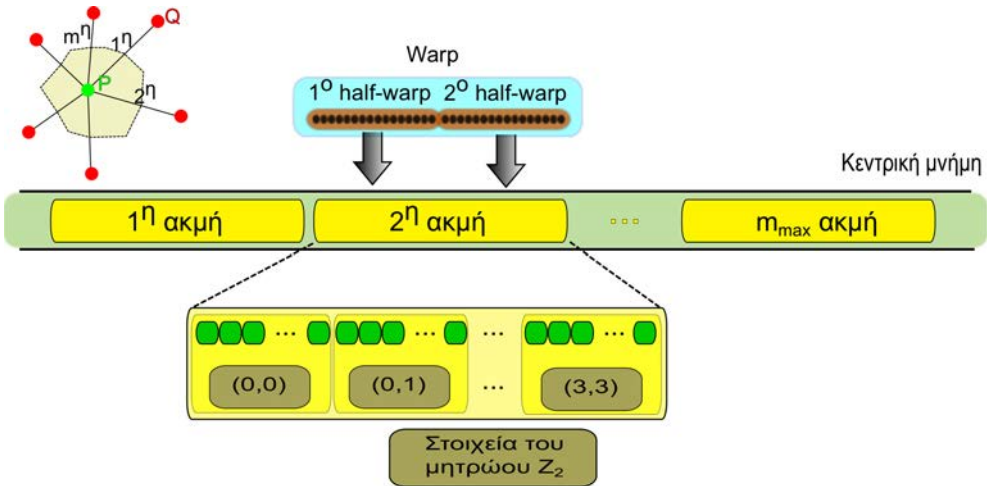
Αντίθετα, η αποθήκευση των μητρώων Z στην κεντρική μνήμη της GPU ακολουθεί ένα τελείως διαφορετικό πρότυπο. Για πληρότητα, επαναλαμβάνεται λίγο πιο περιγραφικά ο τρόπος λειτουργίας των τεχνικών ενός και δύο kernel.

Στο μοναδικό kernel της πρώτης τεχνικής, κάθε thread χειρίζεται έναν κόμβο (P) και σαρώνοντας τους γειτονικούς (Q) σε αυτόν κόμβους υπολογίζει τις ροές $\overrightarrow{\Phi}_{PQ}$ και τα ιακωβιανά μητρώα $\frac{\partial \overrightarrow{\Phi}_{PQ}}{\partial \overrightarrow{W}_Q}$, $\frac{\partial \overrightarrow{\Phi}_{PQ}}{\partial \overrightarrow{W}_P}$. Υπενθυμίζεται ότι το πρώτο από τα δύο ιακωβιανά μητρώα είναι το μητρώο Z_Q το οποίο και αποθηκεύεται στην κεντρική μνήμη της GPU. Κατά τη σάρωση, γίνεται ο ισολογισμός των ροών και των ιακωβιανών μητρώων για το σχηματισμό του υπολοίπου $\overrightarrow{R}_P$ και του μητρώου D_P τα οποία αποθηκεύονται στην κεντρική μνήμη στο τέλος των σαρώσεων.

Στην τεχνική των δύο kernels, το πρώτο αντιστοιχεί τα threads σε ακμές του πλέγματος και υπολογίζει τις ροές $\overrightarrow{\Phi}_{PQ}$ και τα ιακωβιανά μητρώα $\frac{\partial \overrightarrow{\Phi}_{PQ}}{\partial \overrightarrow{W}_Q}$, $\frac{\partial \overrightarrow{\Phi}_{PQ}}{\partial \overrightarrow{W}_P}$ ανά ακμή PQ. Το δεύτερο kernel αντιστοιχεί τα threads σε κόμβους (P) και σαρώνει τις ακμές PQ που άγονται από τον P. Όταν η σάρωση φτάσει λ.χ. στην ακμή PQ, διαβάζει από την κεντρική μνήμη την αντίστοιχη ροή, τα ήδη υπολογισμένα ιακωβιανά μητρώα και αποθηκεύει το μητρώο Z_Q. Κατά τη σάρωση, γίνεται ο ισολογισμός των ροών και των ιακωβιανών μητρώων για το σχηματισμό του υπολοίπου $\overrightarrow{\mathcal{R}}_P$ και του μητρώου D_P τα οποία αποθηκεύονται στην κεντρική μνήμη της κάρτας γραφικών, στο τέλος των σαρώσεων.

Ο τρόπος αποθήκευσης των μητρώων Z στις δύο τεχνικές είναι ο ίδιος και γίνεται όπως περιγράφηκε προηγουμένως κατά τη σάρωση των γειτόνων. Το πρότυπο αποθήκευσης φαίνεται στο σχήμα 4.13. Αν m είναι το πλήθος των ακμών που άγονται από έναν κόμβο P, τότε συμβολίζουμε $Z_1,\ Z_2 \ldots Z_m$ τα μητρώα Z που σχετίζονται με την πρώτη, τη δεύτερη ... τη m-ιοστή ακμή που άγεται από τον P. Σύμφωνα με το σχήμα 4.13, τα threads του ίδιου warp αποθηκεύουν πρώτα τα Z_1 μητρώα των 32 κόμβων που χειρίζεται το warp. Στη συνέχεια, αποθηκεύονται τα Z_2 κ.ο.κ. Για να ακολουθούν όλα τα $Z_i,\ i=1,m$ το ίδιο πρότυπο αποθήκευσης θεωρείται ότι, με κόστος τη δέσμευση επιπλέον θέσεων μνήμης, οι 32 κόμβοι που χειρίζεται το warp έχουν το ίδιο πλήθος γειτόνων (ίσο με m_{max}, όπου m_{max} είναι το μέγιστο πλήθος γειτόνων ανάμεσα στους 32 κόμβους που χειρίζεται το warp). Τα μητρώα Z_i αποθηκεύονται ανά στοιχείο, όπως γίνεται με τα μητρώα D. Στη συνέχεια, αποθηκεύονται τα μητρώα Z_i των υπόλοιπων κόμβων του πλέγματος. Ακολουθώντας αυτόν τον τρόπο αποθήκευσης, τα threads του ίδιου half-warp έχουν πρόσβαση σε θέσεις μνήμης του ίδιου 128

Byte τμήματος της κεντρικής μνήμης. Έτσι, η ταχύτητα μεταφοράς δεδομένων είναι περίπου ίση με 120 GByte/sec, πολύ κοντά στη μέγιστη τιμή των 123 GByte/sec σε μία GPU αρχιτεκτονικής GT200.



Σχήμα 4.13: Αποθήκευση των μητρώων Z στην κεντρική μνήμη της GPU. Τα μητρώα Z αποθηκεύονται ανά warp και με βάση τον τοπικό αύξοντα αριθμό των ακμών που άγονται από τους κόμβους που χειρίζεται το warp.

Η σύγκριση της απόδοσης του CPU-τρόπου αποθήκευσης με τον προτεινόμενο τρόπο σε GPUs δεν έχει νόημα, καθώς ο CPU-τρόπος αποθηκεύει τα μητρώα Z ανά ακμή του πλέγματος ενώ ο προτεινόμενος GPU-τρόπος ανά κόμβους και, συγκεκριμένα, ανά 32 (πλήθος threads ανά warp) κόμβους.

Αναφέρθηκε ότι η χρήση του προτεινόμενου τρόπου αποθήκευσης δεσμεύει επιπλέον θέσεις μνήμης. Όσο μεγαλύτερη είναι η διασπορά του πλήθους των γειτόνων ανάμεσα στους κόμβους που χειρίζονται τα warps, τόσο περισσότερη επιπλέον μνήμη χρειάζεται να δεσμευτεί. Η ελαχιστοποίηση των επιπλέον απαιτούμενων θέσεων μνήμης γίνεται με την ταξινόμηση των κόμβων του πλέγματος με βάση το πλήθος των γειτόνων. Για παράδειγμα, σε ένα 2Δ μη-δομημένο πλέγμα για τον υπολογισμό ατριβούς ροής γύρω από αεροτομή, το οποίο έχει κατασκευαστεί με τη μέθοδο του προελαύνοντος μετώπου, η ταξινόμηση των κόμβων μειώνει την έκταση της απαιτούμενης μνήμης κατά 30% όταν χρησιμοποιείται η τεχνική του ενός kernel. Εντοπίζεται, δηλαδή, μία ακόμη θετική επίδραση της ταξινόμησης των κόμβων πέρα εκείνης που αναλύθηκε στην παράγραφο 4.3.3.

4.4.4 Χρήση των υπολοίπων μνημών της GPU

Η προσεκτική διαχείριση της κεντρικής μνήμης της GPU, όπως παρουσιάστηκε στην ενότητα 4.4.3, συνεισέφερε κατά μεγάλο ποσοστό στον προγραμματισμό GPU-κώδικα υψηλής παράλληλης απόδοσης. Μικρότερο αλλά αρκετό σημαντικό ποσοστό της επιτυχίας αυτής αναλογεί στη σωστή διαχείριση και των υπόλοιπων μνημών της GPU.

Μπορεί με τη χρήση των τεχνικών αποθήκευσης που παρουσιάστηκαν στην ενότητα 4.4.3, η πρόσβαση στο μεγαλύτερο ποσοστό της δεσμευμένης μνήμης να είναι βέλτιστη, υπάρχουν, όμως, και τμήματα μνήμης η πρόσβαση στα οποία εξαιτίας της χρήσης μη-δομημένων πλεγμάτων είναι ακατάστατη. Αυτή είναι η περίπτωση ανάγνωσης των τιμών των μεταβλητών της ροής ή των χωρικών παραγώγων κατά τη σάρωση των γειτόνων. Η πρόσβαση στη πληροφορία αυτή γίνεται μέσω της cached texture μνήμης.

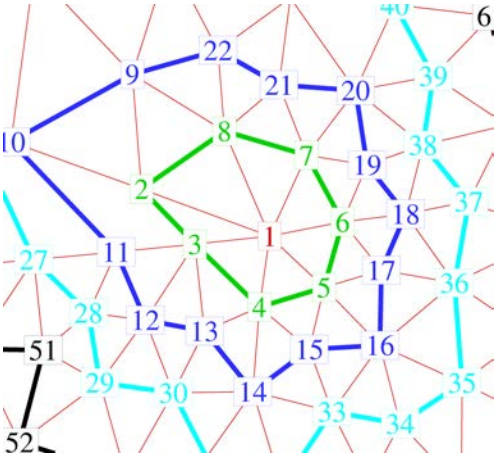
Επιπλέον, σταθερές ποσότητες που δεν μεταβάλλονται κατά τη διάρκεια της επίλυσης, όπως ο λόγος των θερμοχωρητικοτήτων, αποθηκεύονται και διαβάζονται από την cached constant μνήμη.

Η shared μνήμη, όπως αναφέρεται στην ενότητα 4.4.6, χρησιμοποιήθηκε για τη γρήγορη ανταλλαγή πληροφορίας μεταξύ των threads του ίδιου block για τον υπολογισμό του υπολοίπου των διακριτοποιημένων εξισώσεων της ροής.

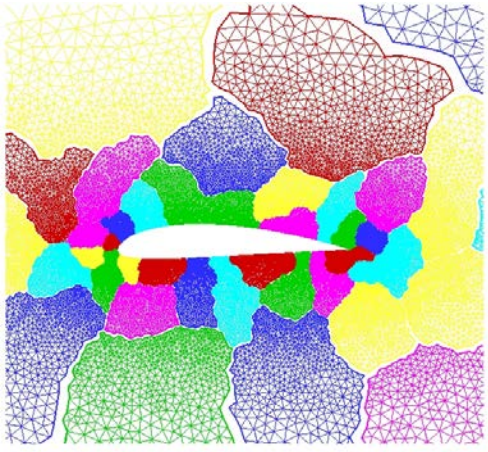
4.4.5 Επαναρίθμηση των κόμβων μη-δομημένου πλέγματος

Με στόχο δύο γειτονικοί κόμβοι να έχουν το δυνατόν πλησιέστερο αύξοντα αριθμό, δοκιμάστηκε μία μέθοδος επαναρίθμησης των κόμβων του πλέγματος. Η μέθοδος αυτή φαίνεται στο σχήμα 4.14. Αρχικά σημειώνεται/καταγράφεται ένας κόμβος ως κόμβος εκκίνησης της επαναρίθμησης. Στο παράδειγμα του σχήματος 4.14, ως τέτοιος κόμβος έχει επιλεχθεί ο κόμβος στην ακμή εκφυγής της αεροτομής. Στη συνέχεια, επαναλαμβάνονται τα ακόλουθα βήματα μέχρι να ολοκληρωθεί η επαναρίθμηση όλων των κόμβων του πλέγματος:

1. Επαναριθμούνται οι πρώτοι γείτονες των ήδη καταγεγραμμένων κόμβων.
2. Διαγράφονται οι καταγεγραμμένοι κόμβοι και καταγράφονται οι κόμβοι που επαναριθμήθηκαν στο προηγούμενο βήμα.



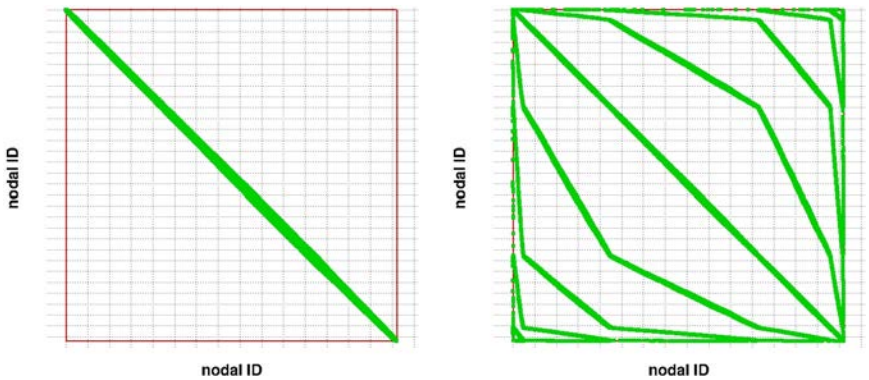
Σχήμα 4.14: Επαναρίθμηση των κόμβων ενός μη-δομημένου πλέγματος.



Σχήμα 4.15: Διαχωρισμός πλέγματος σε υποχωρία.

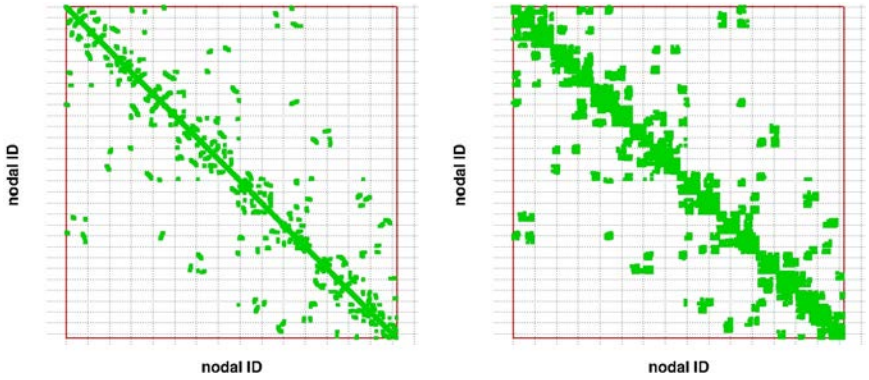
Το σχήμα 4.16(α΄) δείχνει ότι εφαρμόζοντας τη μέθοδο επαναρίθμησης στους κόμβους του μη-δομημένου πλέγματος του σχήματος 4.15 οι αύξοντες αριθμοί γειτονικών κόμβων είναι αρκετά πλησιέστεροι. Αυτό έχει ως αποτέλεσμα, threads του ίδιου warp να έχουν πρόσβαση σε πλησιέστερες θέσεις της κεντρικής μνήμης, μειώνοντας τον αριθμό των προσβάσεων σε αυτή (μειώνοντας το πλήθος των προς πρόσβαση 128Byte τμημάτων της κεντρικής μνήμης). Επομένως, αυξάνεται η παράλληλη απόδοση του GPU-κώδικα. Η επαναρίθμηση των κόμβων αυξάνει επίσης την απόδοση της texture cache μνήμης της GPU, καθώς αυξάνεται η πιθανότητα τα threads να διαβάζουν από την cache μνήμη αντί της αργής κεντρικής μνήμης. Υπενθυμίζεται ότι η texture μνήμη αποτελεί τμήμα της κεντρικής μνήμης. Συνεπώς, η ταχύτητα προσπέλασής της είναι ίδια με εκείνη της κεντρικής μνήμης, αν τα δεδομένα στα οποία ζητείται η πρόσβαση δεν βρίσκονται στην texture cache μνήμη.

Με βάση όσα έχουν αναφερθεί σε προηγούμενες παραγράφους, η ταξινόμηση των κόμβων σύμφωνα με το πλήθος των γειτονικών σ΄ αυτούς κόμβους είναι υποχρεωτική. Η ταξινόμηση αυτή καταστρέφει τη διάταξη των κόμβων όπως αυτή προέκυψε από την επαναρίθμηση, όπως φαίνεται στο σχήμα 4.16(β΄). Ο βέλτιστος συνδυασμός της τεχνικής της επαναρίθμησης με την ταξινόμηση των κόμβων του πλέγματος επιτυγχάνεται χωρίζοντας το πλέγμα σε υποχωρία (σχήμα 4.15). Οι κόμβοι του κάθε υποχωρίου επαναριθμούνται και ταξινομούνται με βάση το πλήθος των γειτόνων. Τα σχήματα 4.17(α΄), 4.17(β΄) δείχνουν τη διάταξη των επαναριθμημένων κόμβων του πλέγματος με και χωρίς ταξινόμηση αυτών, αντίστοιχα. Η επαναρίθμηση σε συνδυασμό με την ταξινόμηση των κόμβων του πλέγματος αυξάνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα. Η θετική αυτή επίδραση παρουσιάζεται στην ενότητα 4.5.



(α΄) (β΄)

Σχήμα 4.16: Διάταξη των επαναριθμημένων κόμβων του πλέγματος (α΄) με και (β΄) χωρίς ταξινόμηση. Σχεδιάζονται οι αύξοντες αριθμοί (nodal ID) των γειτονικών κόμβων ανά κόμβο.



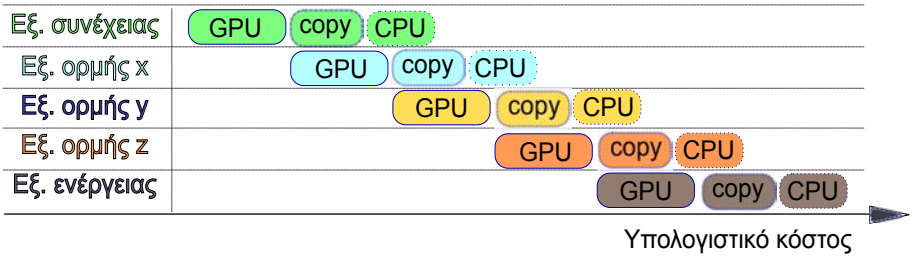
(α′) (β′)

Σχήμα 4.17: Διάταξη των κόμβων του πλέγματος. Το πλέγμα αρχικά χωρίστηκε σε υποχωρία. Οι κόμβοι του κάθε υποχωρίου (α′) επαναριθμήθηκαν ή (β′) επαναριθμήθηκαν και ταξινομήθηκαν. Σχεδιάζονται οι αύξοντες αριθμοί (nodal ID) των γειτονικών κόμβων ανά κόμβο.

4.4.6 Ταυτόχρονη επεξεργασία δεδομένων από τη CPU και τη GPU

Τμήματα κώδικα, η εκτέλεση των οποίων δεν μπορεί να γίνει παράλληλα ή έχουν χαμηλό βαθμό παραλληλοποίησης, εκτελούνται στη CPU ενώ, ταυτόχρονα, ανεξάρτητες διεργασίες εκτελούνται παράλληλα στη GPU. Έτσι αξιοποιείται το 100% της διαθέσιμης υπολογιστικής ισχύος.

Ένα παράδειγμα συνεργασίας και παράλληλης επεξεργασίας δεδομένων από τη CPU και τη GPU είναι ο υπολογισμός της νόρμας των υπολοίπων των διακριτοποιημένων εξισώσεων της ροής σε κάθε ψευδοχρονικό βήμα, με σκοπό τον έλεγχο της σύγκλισης της επαναληπτικής διαδικασίας της επίλυσης. Ο υπολογισμός αυτός είναι διαδικασία χαμηλού βαθμού παραλληλοποίησης καθώς επιβάλλει την άθροιση των υπολοίπων $\overrightarrow{\mathcal{R}}$ ανά κόμβο. Η GPU υπολογίζει τις ανά block (επιμέρους) νόρμες των υπόλοιπων μίας εξίσωσης χρησιμοποιώντας τη shared μνήμη. Η τελική άθροιση των επιμέρους νορμών γίνεται στη CPU. Ενώ η CPU αθροίζει τα ανά block υπόλοιπα (για τον υπολογισμό της νόρμας του υπολοίπου) μίας εξίσωσης (π.χ. της εξίσωσης συνέχειας), η GPU υπολογίζει τα ανά block αθροίσματα της επόμενης εξίσωσης (π.χ. της εξίσωσης ορμής κατά x), κ.ο.κ. Τα παραπάνω φαίνονται στο σχήμα 4.18.



Σχήμα 4.18: Συνεργασία της CPU με τη GPU στον υπολογισμό των υπολοίπων των διακριτοποιημένων εξισώσεων της ροής. Η GPU υπολογίζει τα αθροίσματα των υπολοίπων $\vec{\mathcal{R}}$ των κόμβων που χειρίζονται τα blocks. Η CPU πραγματοποιεί την τελική άθροιση για τον υπολογισμό της νόρμας των υπολοίπων των εξισώσεων της ροής. Όταν λ.χ. η CPU υπολογίζει το υπόλοιπο της εξίσωσης της συνέχειας, η GPU υπολογίζει τα επιμέρους αθροίσματα των ανά κόμβο υπολοίπων της εξίσωσης της κατά x ορμής, κ.ο.κ.

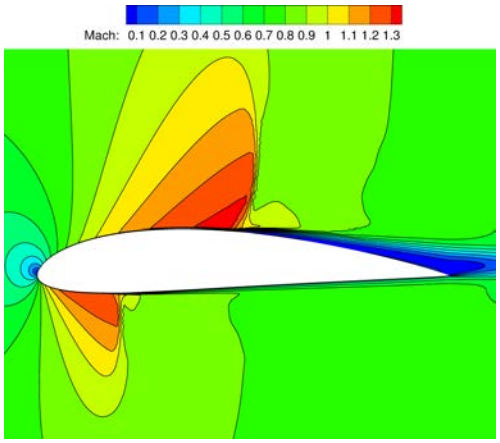
4.4.7 Αριθμητική μικτής ακρίβειας

Οι διακριτοποιημένες εξισώσεις της ροής επιλύονται ως προς τη διόρθωση των μεταβλητών της ροής $\left(\Delta\vec{W}\right)$, έκφραση 4.2. Στη γραφή αυτή το δεξιό μέλος είναι το υπόλοιπο των διακριτοποιημένων εξισώσεων της ροής και αποτελεί τη 'φυσική' του προβλήματος. Συνεπώς, μπορεί να χρησιμοποιηθεί ένα ακριβές σχήμα υπολογισμού των υπολοίπων $\vec{\mathcal{R}}$ και ένα λιγότερο ακριβές σχήμα στον υπολογισμό των μητρώων του αριστερού μέλους $\left(\frac{\partial\vec{\mathcal{R}}}{\partial\vec{W}}\right)$. Μία συνήθης τακτική είναι ο υπολογισμός του $\frac{\partial\vec{\mathcal{R}}}{\partial\vec{W}}$ με σχήμα πρώτης τάξης ακρίβειας και ο υπολογισμός του $\vec{\mathcal{R}}$ με δεύτερης τάξης. Στη διατριβή, τα στοιχεία των μητρώων του αριστερού μέλους D, Z (υπενθυμίζεται ότι αυτά αποτελούν τους διαγώνιους και μη-διαγώνιους όρους του $\frac{\partial\vec{\mathcal{R}}}{\partial\vec{W}}$) αποθηκεύονται σε μεταβλητές απλής ακρίβειας ενώ το δεξιό μέλος αποθηκεύεται σε μεταβλητές διπλής ακρίβειας. Τονίζεται ότι δεν χρησιμοποιείται αριθμητική απλής ακρίβειας κατά τον υπολογισμό των μητρώων του αριστερού μέλους, αλλά οι υπολογισμοί γίνονται με μεταβλητές διπλής ακρίβειας, το αποτέλεσμα όμως αποθηκεύεται σε μεταβλητές απλής (αποκόπτοντας ένα τμήμα σημαντικών ψηφίων). Το σχήμα αυτό ονομάστηκε αριθμητική 'μικτής' ακρίβειας (Mixed Precision Arithmetic, MPA). Στη συνέχεια, για ευκολία, η αριθμητικής απλής ή διπλής ακρίβειας θα αναφέρονται ως SPA (Single Precision Arithmetic) ή DPA (Double Precision Arithmetic), αντίστοιχα

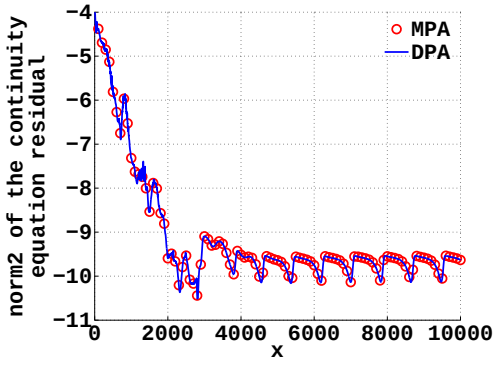
Εφόσον χρησιμοποιούνται μεταβλητές διπλής ακρίβειας για την αποθήκευση των υπολοίπων $\vec{\mathcal{R}}$, η ακρίβεια των αριθμητικών προλέξεων δεν αλλοιώνεται. Το σχήμα 4.20 δείχνει τη σύγκλιση της εξίσωσης συνέχειας κατά την επίλυση 2Δ τυρβώδους ροής γύρω από μεμονωμένη αεροτομή (σχήμα 4.19). Η επίλυση έγινε χρησιμοποιώντας MPA ή DPA. Οι δύο καμπύλες σχεδόν ταυτίζονται, αποδεικνύοντας ότι η χρήση αριθμητικής μικτής ακρίβειας δεν αλλοιώνει την ακρίβεια των προλέξεων της ροής. Ταυτόσημη πορεία σύγκλισης με τη DPA εκδοχή του GPU-κώδικα (δεν απεικονίζεται στο σχήμα) ακολουθεί και ο CPU-κώδικας. Εξάλλου, μεταξύ άλλων, σκοπός της διατριβής ήταν η

μεταφορά του CPU-επιλύτη στη GPU, η παραγωγή δηλαδή ενός νέου επιλύτη αρκετά υψηλής παράλληλης απόδοσης, οι αριθμητικές προλέξεις του οποίου να είναι ταυτόσημες με εκείνες του CPU-κώδικα.

Τα σχήματα 4.21(α΄), 4.21(β΄) δείχνουν την κατανομή των συντελεστών πίεσης και τριβής στην επιφάνεια της αεροτομής που προέκυψε από την επίλυση του προβλήματος του σχήματος 4.19, χρησιμοποιώντας DPA και MPA. Οι δύο κατανομές συμπίπτουν.

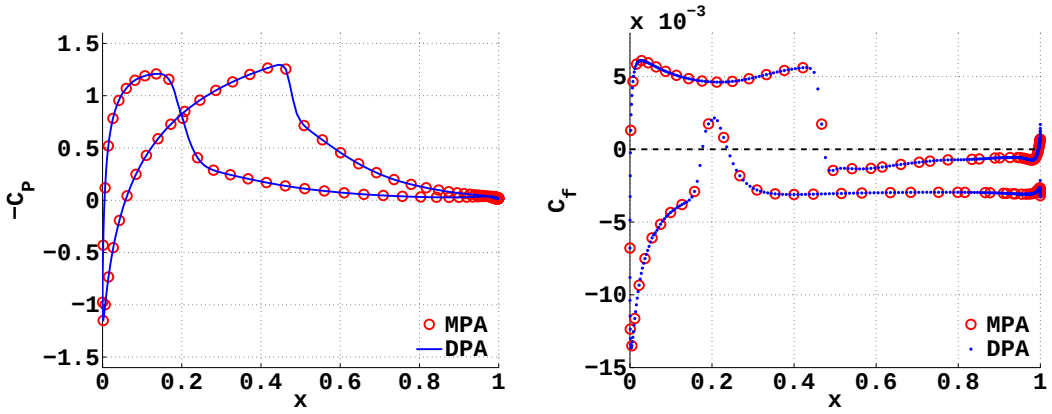


Σχήμα 4.19: Χρονικά μόνιμη τυρβώδης ροή γύρω από μεμονωμένη αεροτομή NACA4415: πεδίο του αριθμού Mach. Οι τιμές του αριθμού Mach και της γωνίας της επ΄άπειρο ροής είναι 0.75 και 0^o αντίστοιχα. Ο αριθμός Reynolds ως προς το μήκος της χορδής είναι ίσος με 10^6.



Σχήμα 4.20: Χρονικά μόνιμη τυρβώδης ροή γύρω από μεμονωμένη αεροτομή NACA4415: σύγκλιση της εξίσωσης συνέχειας σε λογαριθμική κλίμακα. Ο GPU-επιλύτης χρησιμοποιεί αριθμητική μικτής (MPA) ή διπλής (DPA) ακρίβειας. Οι δύο καμπύλες σχεδόν ταυτίζονται.

Η χρήση μεταβλητών απλής ακρίβειας για την αποθήκευση των μητρώων του αριστερού μέλους (υπενθυμίζεται ότι τα D, Z αποτελούν περίπου το 70% του συνολικού δεσμευμένου χώρου κατά την επίλυση ατριβών ροών) μειώνει σημαντικά τον αριθμό προσβάσεων στην κεντρική μνήμη και αυξάνει σημαντικά, όπως θα φανεί στην παράγραφο 4.5, την παράλληλη απόδοση του GPU-κώδικα. Τονίζεται ότι η μείωση του



(α′) (β′)

Σχήμα 4.21: Χρονικά μόνιμη τυρβώδης ροή γύρω από μεμονωμένη αεροτομή NACA4415: κατανομή των συντελεστών πίεσης και τριβής. Ο GPU-κώδικας χρησιμοποιεί MPA ή DPA. Τα αποτελέσματα των δύο τρεξιμάτων ταυτίζονται, αποδεικνύοντας ότι η χρήση της αριθμητικής μικτής ακρίβειας δεν αλλοιώνει την ακρίβεια της πρόλεξης της ροής.

αριθμού των προσβάσεων στην κεντρική μνήμη οφείλεται στον τρόπο διαχείρισης αυτής από τη GPU. Συνεπώς, η χρήση της αριθμητικής μικτής ακρίβειας έχει σαφώς σημαντικότερα οφέλη όταν χρησιμοποιείται σε GPUs, αντί CPUs, ως προς την επιτάχυνση της πρόλεξης των αποτελεσμάτων.

Το περιβάλλον προγραμματισμού της CUDA προσφέρει τη δυνατότητα χρήσης επιπλέον τύπων μεταβλητών, όπως float2 ή float4[2]. Μία μεταβλητή τύπου float2, όπως φαίνεται και στον ψευδοκώδικα 7, αποτελείται από δύο πραγματικούς αριθμούς απλής ακρίβειας (float) x, y και έχει μήκος όσο ένας πραγματικός αριθμός διπλής ακρίβειας (double). Αντίθετα, μία μεταβλητή τύπου float4 (ψευδοκώδικας 8) αποτελείται από 4 συνιστώσες x, y, z και w.

Ψευδοκώδικας 7 Ορισμός τύπου μεταβλητής float2

```
1: struct __align__(8) float2
2: {
3:         float x;
4:         float y;
5: };
```

Το πρόθεμα __align__, γενικά, καθορίζει το βήμα με το οποίο μία μεταβλητή αποθηκεύεται/διαβάζεται στη/από τη μνήμη της GPU. Σημειώνεται ότι, τα threads μίας GPU, έχουν τη δυνατότητα πρόσβασης σε τμήματα της κεντρικής μνήμης μήκους 4, 8 ή 16 Bytes. Το πρόθεμα __align__(16) ορίζει βήμα μήκους 16 Bytes. Συνεπώς, η ανάγνωση

[2]Για το σύνολο των επιπλέον τύπων μεταβλητών που προσφέρει το περιβάλλον προγραμματισμού της CUDA ο αναγνώστης παραπέμπεται στο [2].

Ψευδοκώδικας 8 Ορισμός τύπου μεταβλητής float4

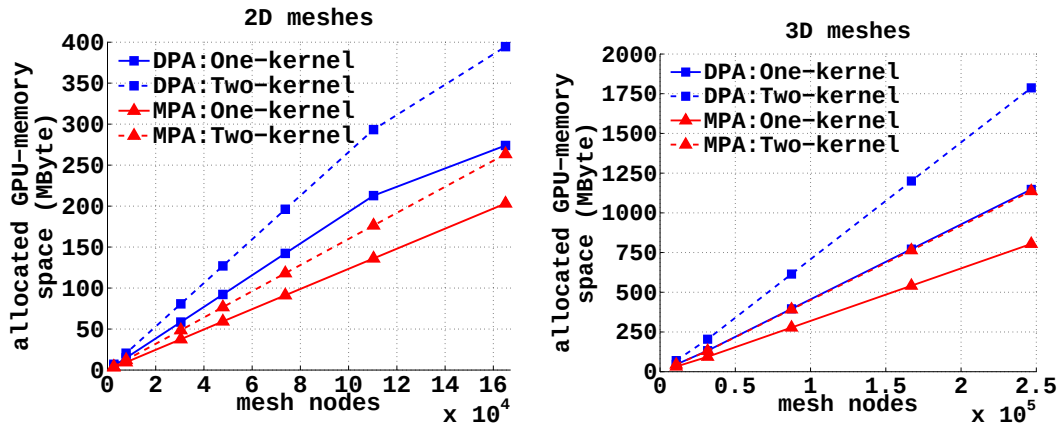
```
1: struct __align__(16) float4
2: {
3:     float x;
4:     float y;
5:     float z;
6:     float w;
7: };
```

ή η αποθήκευση μίας float4 μεταβλητής στην κεντρική μνήμη μπορεί να γίνει 'με τη μία' και όχι ανά συνιστώσα. Δηλαδή, το thread πραγματοποιεί πρόσβαση στην κεντρική μνήμη μία, αντί για τέσσερις φορές. Αντίστοιχα συμβαίνει για τις μεταβλητές τύπου float2. Επομένως, για τη μείωση του αριθμού των προσβάσεων στην κεντρική μνήμη, χρησιμοποιήθηκαν όχι απλά μεταβλητές απλής ακρίβειας (float), αλλά float2 και float4, για την αποθήκευση των D, Z. Ο ορισμός των παραπάνω τύπων μεταβλητών δεν απαιτείται από τον προγραμματιστή καθώς οι εν λόγω τύποι μεταβλητών υποστηρίζονται από το περιβάλλον προγραμματισμού της CUDA. Για λόγους πληρότητας όμως, οι ορισμοί αυτοί συμπεριλήφθηκαν στο κείμενο.

Ταυτόχρονα, η μείωση του απαιτούμενου χώρου στην κεντρική μνήμη της κάρτας γραφικών αυξάνει το εύρος των αεροδυναμικών και αεροελαστικών εφαρμογών σε GPUs. Τα σχήματα 4.22(α′), 4.22(β′) δείχνουν την έκταση της μνήμης σε MBytes που απαιτείται για την επίλυση 2Δ και 3Δ ατριβών ροών χρησιμοποιώντας DPA και MPA και τις τεχνικές ενός ή δύο kernels για πλέγματα διαφορετικού πλήθους κόμβων. Η χρήση μικτής ακρίβειας μειώνει περίπου 25% την απαιτούμενη μνήμη στην επίλυση 2Δ ή 3Δ προβλημάτων, χρησιμοποιώντας την αποδοτικότερη τεχνική των δύο kernels. Υπενθυμίζεται ότι η συνολική έκταση της κεντρικής μνήμης μίας κάρτας γραφικών αρχιτεκτονικής GT200 ή Fermi είναι 1 GByte ή 3 GByte αντίστοιχα. Έτσι, για παράδειγμα, το πυκνότερο 3Δ πλέγμα του σχήματος 4.22(β′) (περίπου 250000 κόμβων) που αρχικά δεν μπορούσε να χρησιμοποιηθεί σε μία GPU αρχιτεκτονικής GT200, εξαιτίας των υψηλών για τις προδιαγραφές της κάρτας απαιτήσεων σε μνήμη (μεγαλύτερη του 1 GByte) χρησιμοποιώντας αριθμητική διπλής ακρίβειας, μπορεί πλέον να χρησιμοποιηθεί με την αριθμητική μικτής ακρίβειας και την τεχνική του ενός kernel.

4.5 Απόδοση του GPU-κώδικα

Για τη μέτρηση της παράλληλης απόδοσης του GPU-κώδικα της διατριβής επιλύθηκε η ροή γύρω από αεροτομή και πτέρυγα χρησιμοποιώντας τον GPU και τον CPU-επιλύτη και πλέγματα με διαφορετικό πλήθος κόμβων. Στους υπολογισμούς αυτούς, ο CPU-κώδικας χρησιμοποιεί αριθμητική διπλής ακρίβειας και εκτελείται σε έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη. Αντίθετα ο GPU-επιλύτης χρησιμοποιεί αριθμητική διπλής, μικτής ή απλής ακρίβειας και την τεχνική ενός ή δύο kernels. Υπενθυμίζεται ότι η DPA και MPA εκδοχές του GPU-επιλύτη



(α′) (β′)

Σχήμα 4.22: Διάσταση σε MBytes της κεντρικής μνήμης που δεσμεύεται από τον GPU-κώδικα όταν χρησιμοποιείται η αριθμητική διπλής ή μικτής ακρίβειας με την τεχνική ενός ή δύο kernels για την επίλυση (α′) 2Δ ή (β′) 3Δ ατριβών ροών, χρησιμοποιώντας πλέγματα διαφορετικού πλήθους κόμβων.

παράγουν τα ίδια αποτελέσματα με τον CPU-κώδικα και έχουν σχεδόν ίδια πορεία σύγκλισης. Τα αποτελέσματα των αεροδυναμικών αυτών εφαρμογών δεν παρουσιάζονται εδώ, εφόσον πραγματοποιήθηκαν με σκοπό τη μέτρηση της παράλληλης απόδοσης του GPU-κώδικα. Οι αριστερές στήλες των σχημάτων 4.23, 4.24, 4.25 παρουσιάζουν τις επιταχύνσεις, δηλαδή τους χρόνους εκτέλεσης του CPU-κώδικα προς τους αντίστοιχους του GPU-κώδικα στον ίδιο αριθμό ψευδοχρονικών βημάτων, που σημειώθηκαν χρησιμοποιώντας πλέγματα με διαφορετικό πλήθος κόμβων στην επίλυση ατριβούς ροής γύρω από αεροτομή, τυρβώδους ροής γύρω από αεροτομή και ατριβούς ροής γύρω από πτέρυγα, αντίστοιχα. Τα πλέγματα που χρησιμοποιήθηκαν, δημιουργήθηκαν με τη μέθοδο του προελαύνοντος μετώπου. Επαναλήφθηκαν οι ίδιοι υπολογισμοί επαναριθμώντας και ταξινομώντας τους κόμβους των πλεγμάτων. Οι σχετικές επιταχύνσεις δείχνονται στις δεξιές στήλες των ίδιων σχημάτων.

Οι μετρήσεις δείχνουν ότι η επιτάχυνση στην πρόλεξη της ροής με τη χρήση μίας GPU αντί ενός πυρήνα μίας σημερινής CPU εξαρτάται από τη φύση της ροής (ατριβής, τυρβώδης), τη διάσταση του πλέγματος και, φυσικά, το μοντέλο της κάρτας γραφικών. Συγκεκριμένα, η μεγαλύτερη επιτάχυνση που καταγράφηκε στην επίλυση 2Δ τυρβώδους συμπιεστής ροής (σχήμα 4.24(ϛ′)) είναι 60x, 90x και 110x εκτελώντας τον GPU-κώδικα σε μία Tesla M2050 και χρησιμοποιώντας DPA, MPA και SPA, αντίστοιχα. Οι ίδιοι υπολογισμοί αν πραγματοποιούνταν σε μία GTX 280 ή 285 θα έδιναν επιτάχυνση μικρότερη κατά περίπου 30% (σχήματα 4.24(β′), 4.24(δ′)). Η μέγιστη επιτάχυνση που καταγράφηκε στην επίλυση των 3Δ εξισώσεων Euler (σχήμα 4.25(ϛ′)) είναι 40x, 55x και 90x χρησιμοποιώντας μία Tesla M2050 και DPA, MPA και SPA, αντίστοιχα. Οι αντίστοιχες επιταχύνσεις χρησιμοποιώντας μία GTX 280 ή 285 είναι μικρότερες κατά περίπου 25% (σχήματα 4.25(β′), 4.25(δ′)). Οι επιταχύνσεις που σημειώθηκαν είναι ιδιαίτερα υψηλές παρά τη χρήση της κεντροκομβικής διατύπωσης των πεπερασμένων όγκων σε μη-δομημένα πλέγματα. Πρακτικά αυτό σημαίνει ότι αεροδυ-

ναμικά ή αεροελαστικά προβλήματα, η επίλυση των οποίων θα απαιτούσε ώρες σε μία σημερινή CPU, μπορούν να επιλυθούν μέσα σε λίγα λεπτά χρησιμοποιώντας μία GPU.

Αξιοσημείωτο είναι το γεγονός ότι οι μέγιστες επιταχύνσεις σημειώθηκαν στα πλέγματα με τους περισσότερους κόμβους. Δηλαδή, εφαρμογές μεγάλης κλίμακας με υψηλούς χρόνους επίλυσης ευνοούνται περισσότερο από τη χρήση των GPUs.

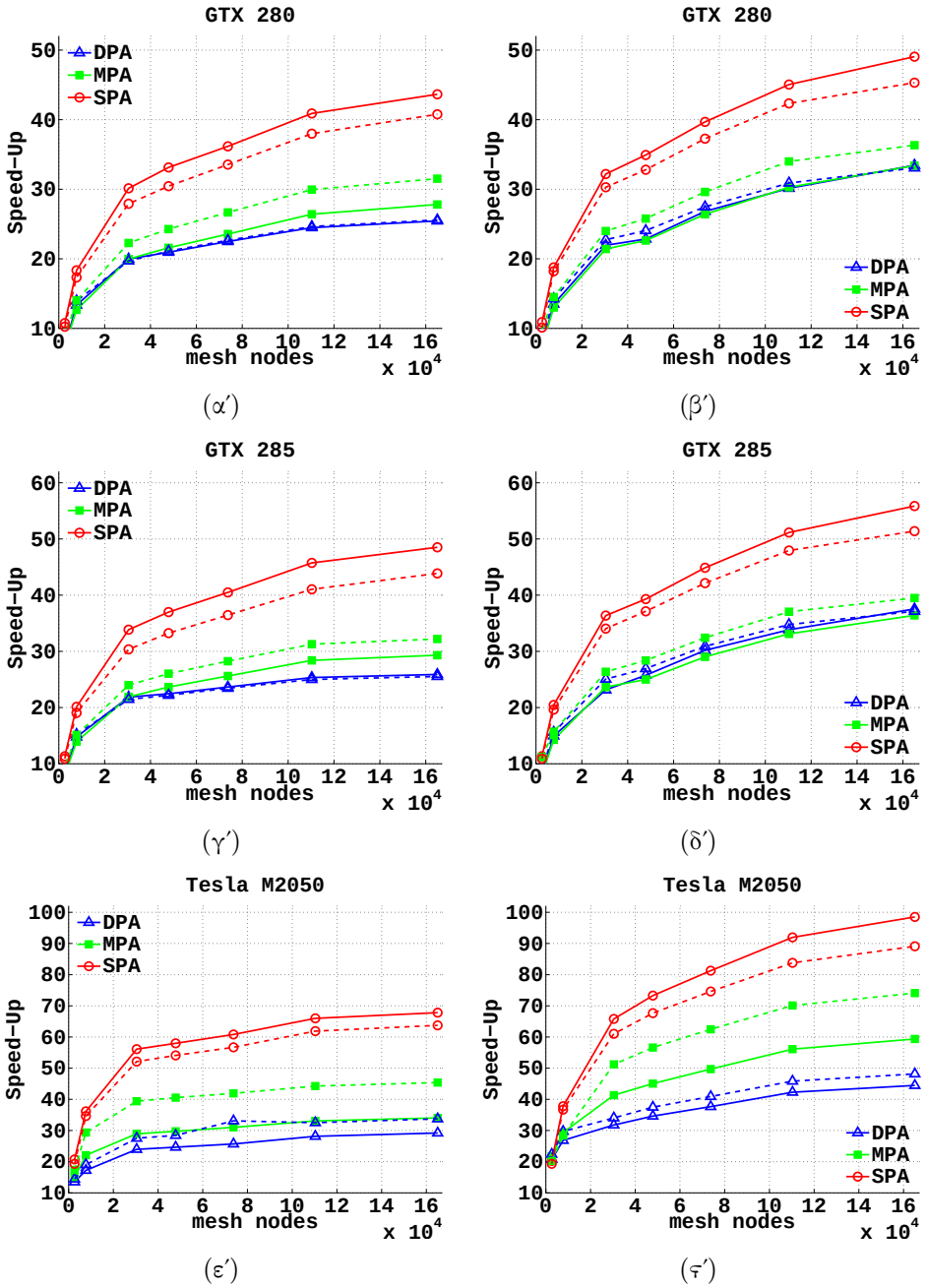
Η χρήση MPA αντί DPA, όπως φαίνεται στα σχήματα 4.23, 4.24, 4.25, αυξάνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα, και κατά συνέπεια την επιτάχυνση στην πρόλεξη της ροής. Σημαντικότερη είναι η επίδραση στη συνιστώσα του GPU-κώδικα που χρησιμοποιεί την τεχνική των δύο kernels. Συγκεκριμένα, η MPA εκδοχή του GPU-επιλύτη δίνει περίπου 40% μεγαλύτερες επιταχύνσεις στην πρόλεξη της ροής σε σχέση με τη DPA εκδοχή, χρησιμοποιώντας την τεχνική των δύο kernels. Αυτό αναμένονταν επειδή η τεχνική των δύο kernels χρησιμοποιεί περισσότερη μνήμη σε σχέση με εκείνη ενός kernel.

Ιδιαίτερα σημαντική είναι η αύξηση της παράλληλης απόδοσης του GPU-κώδικα με την επαναρίθμηση και ταξινόμηση (ως προς το πλήθος των γειτόνων) των κόμβων του πλέγματος. Ενδεικτικά αναφέρεται ότι η μέγιστη επιτάχυνση που επιτεύχθηκε στην πρόλεξη 3Δ ατριβούς συμπιεστής ροής είναι περίπου 30x, 40x ή 70x χρησιμοποιώντας DPA, MPA και SPA, αντίστοιχα. Με την επαναρίθμηση και ταξινόμηση των κόμβων, οι αντίστοιχες επιταχύνσεις γίνονται 40x, 50x και 90x, αντίστοιχα.

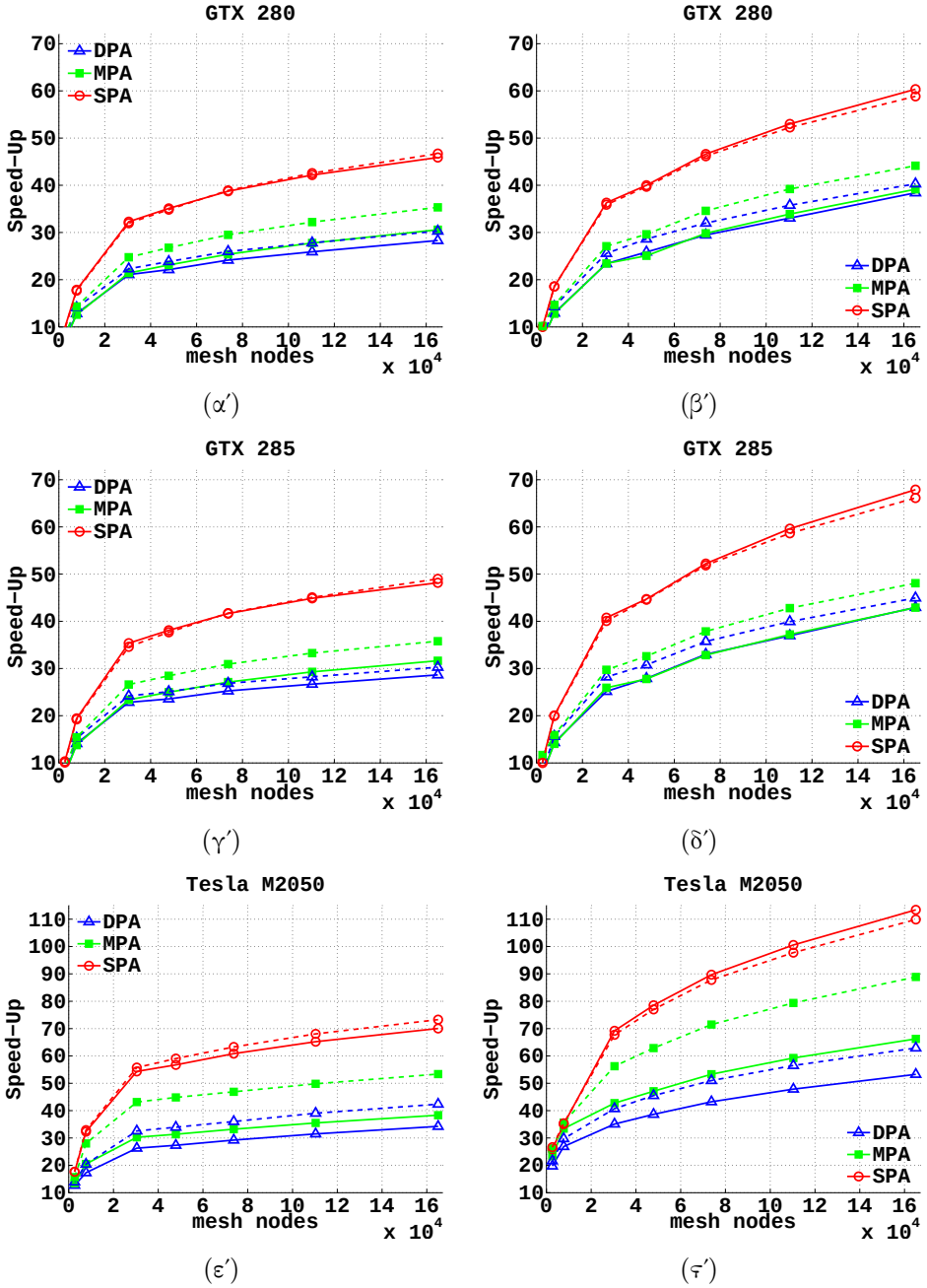
Παρατηρείται ότι, επαναριθμώντας και ταξινομώντας τους κόμβους του πλέγματος, η SPA εκδοχή του GPU-κώδικα που χρησιμοποιεί την τεχνική ενός kernel είναι πιο γρήγορη σε σχέση με την ίδια εκδοχή του GPU-επιλύτη που χρησιμοποιεί την τεχνική των δύο kernels, αντίθετα με ότι συμβαίνει στις εκδοχές DPA και MPA. Πρακτικά αυτό σημαίνει ότι, χρησιμοποιώντας αποκλειστικά SPA, είναι αποδοτικότερος ο διπλός υπολογισμός των ροών $\overrightarrow{\Phi}$ αντί του υπολογισμού και της προσωρινής αποθήκευσης αυτών με κόστος τη δέσμευση επιπλέον θέσεων μνήμης. Υπενθυμίζεται ότι η ταχύτητα εκτέλεσης αριθμητικών πράξεων μεταξύ πραγματικών αριθμών απλής ακρίβειας σε GPUs αρχιτεκτονικής GT200 ή Fermi είναι περίπου 12 ή 2 φορές μεγαλύτερη σε σχέση με εκείνη μεταξύ πραγματικών αριθμών διπλής ακρίβειας.

Στα σχήματα 4.25(α΄)-(ϝ΄) φαίνεται ότι, στις GPUs αρχιτεκτονικής GT200 δοκιμάστηκαν πλέγματα μέχρι ενός πλήθους κόμβων εξαιτίας της περιορισμένης μνήμης αυτών των καρτών γραφικών (1 GByte). Το κεφάλαιο 5 παρουσιάζει την παραλληλοποίηση του GPU-επιλύτη σε GPUs του ίδιου ή διαφορετικών υπολογιστικών κόμβων για την επίλυση μεγάλης κλίμακας αεροδυναμικών και αερελαστικών εφαρμογών.

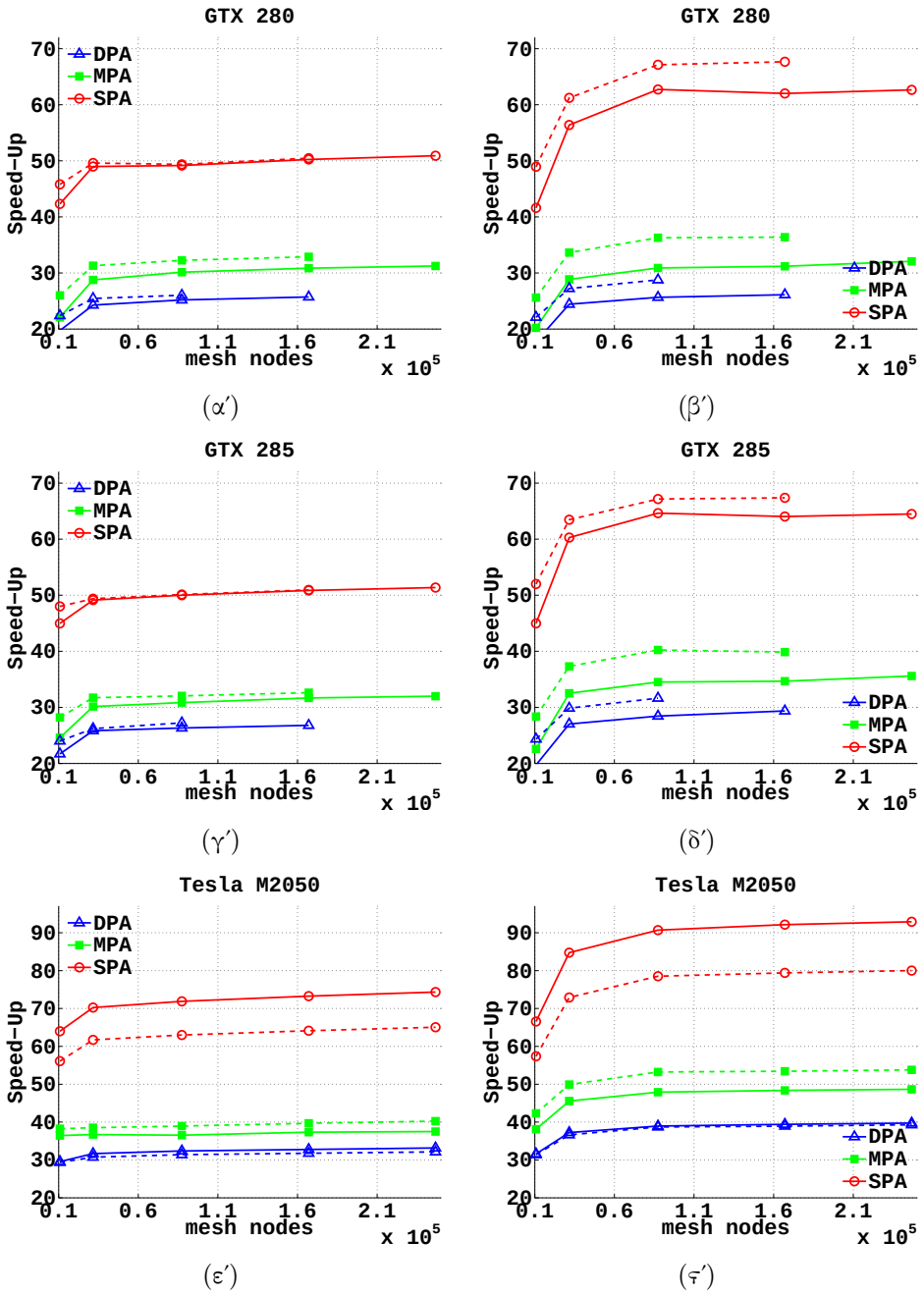
Η απλής ακρίβειας εκδοχή του GPU-επιλύτη, όπως ήταν λογικό, έχει την υψηλότερη παράλληλη απόδοση. Υστερεί όμως ως προς την ακρίβεια των αριθμητικών αποτελεσμάτων, σε σχέση με τη μικτής ακρίβειας εκδοχή. Στο κεφάλαιο 7 παρουσιάζεται το πώς μπορεί να αξιοποιηθεί αυτό το πολύ γρήγορο αλλά όχι τόσο ακριβές λογισμικό στο πλαίσιο ενός διεπίπεδου εξελικτικού αλγορίθμου, για το σχεδιασμό μεμονωμένης αεροτομής ή αεροτομής συμπιεστή.



Σχήμα 4.23: Επιτάχυνση στην πρόλεξη 2Δ ατριβούς ροής συμπιεστού ρευστού χρησιμοποιώντας πλέγματα διαφορετικού πλήθους κόμβων, διαφορετικές GPUs, SPA, MPA ή DPA και την τεχνική ενός ή δύο kernels. Η απόδοση του GPU-κώδικα με την τεχνική ενός ή δύο kernels φαίνεται με τη συνεχή και τη διακεκομμένη καμπύλη, αντίστοιχα. Τα πλέγματα των σχημάτων της δεξιάς στήλης είναι ίδια με εκείνα των αντίστοιχων σχημάτων της αριστερής στήλης με επαναριθμημένους και κατάλληλα ταξινομημένους κόμβους.



Σχήμα 4.24: Επιτάχυνση στην πρόλεξη 2Δ τυρβώδους ροής συμπιεστού ρευστού χρησιμοποιώντας πλέγματα διαφορετικού πλήθους κόμβων, διαφορετικές GPUs, SPA, MPA ή DPA και την τεχνική ενός ή δύο kernels. Η απόδοση του GPU-κώδικα με την τεχνικής ενός ή δύο kernels φαίνεται με τη συνεχή και τη διακεκομμένη καμπύλη, αντίστοιχα. Τα πλέγματα των σχημάτων της δεξιάς στήλης είναι ίδια με εκείνα των αντίστοιχων σχημάτων της αριστερής στήλης με επαναριθμημένους και κατάλληλα ταξινομημένους κόμβους.



Σχήμα 4.25: Επιτάχυνση στην πρόλεξη 3Δ ατριβούς ροής συμπιεστού ρευστού χρησιμοποιώντας πλέγματα διαφορετικού πλήθους κόμβων, διαφορετικές GPUs, SPA, MPA ή DPA και την τεχνική ενός ή δύο kernels. Η απόδοση του GPU-κώδικα με την τεχνικής ενός ή δύο kernels φαίνεται με τη συνεχή και τη διακεκομμένη καμπύλη, αντίστοιχα. Τα πλέγματα των σχημάτων της δεξιάς στήλης είναι ίδια με εκείνα των αντίστοιχων σχημάτων της αριστερής στήλης με επαναριθμημένους και κατάλληλα ταξινομημένους κόμβους.

Κεφάλαιο 5

Παραλληλοποίηση του GPU-κώδικα σε πολλές κάρτες γραφικών

Στο προηγούμενο κεφάλαιο αναλύθηκε ο GPU-κώδικας της διατριβής και παρουσιάστηκαν αποδοτικές τεχνικές και τρόποι προγραμματισμού μίας κάρτας γραφικών. Η διαθέσιμη μνήμη, όμως, των σημερινών GPUs (1 ή 3 GByte για τις κάρτες γραφικών αρχιτεκτονικής GT200 ή Fermi αντίστοιχα) δεν αρκεί για την επίλυση μεγάλης κλίμακας προβλημάτων αεροδυναμικής και αεροελαστικότητας. Το κεφάλαιο αυτό ασχολείται με τον προγραμματισμό πολλών GPUs του ίδιου ή διαφορετικών διασυνδεδεμένων υπολογιστικών κόμβων και αναλύει την επέκταση του GPU-κώδικα της διατριβής σε πολλές κάρτες γραφικών. Η μικτής ακρίβειας (MPA) εκδοχή του παράλληλου (σε πολλές κάρτες γραφικών του ίδιου υπολογιστικού κόμβου) GPU-κώδικα χρησιμοποιήθηκε στην ανάλυση μοντέλου επιβατικού αεροσκάφους τύπου Blended Wing Body (BWB), όπως παρουσιάζεται στο κεφάλαιο 9.

5.1 Συστοιχίες καρτών γραφικών

Ως συστοιχία καρτών γραφικών ορίζεται εκείνη η συστοιχία που αποτελείται από διασυνδεδεμένους υπολογιστικούς κόμβους, κάθε ένας από τους οποίους έχει τουλάχιστον μία ενσωματωμένη ή συνδεδεμένη (εξωτερικά) σε αυτόν, κάρτα γραφικών. Τις GPUs του κάθε υπολογιστικού κόμβου μίας τέτοιας συστοιχίας χειρίζονται διεργασίες (CPU-threads) που εκτελούνται στις CPUs των ίδιων υπολογιστικών κόμβων. Οι επόμενες παράγραφοι περιγράφουν γενικά το τρόπο επικοινωνίας μεταξύ δύο ή περισσοτέρων GPUs του ίδιου ή διαφορετικών υπολογιστικών κόμβων. Παρουσιάζονται, δηλαδή, οι δυνατοί τρόποι παράλληλης διαχείρισης των CPU-threads που χειρίζονται τις GPUs της συστοιχίας. Επίσης, επισημαίνονται οι τρόποι επικοινωνίας που επιλέχθηκαν στη διατριβή.

5.1.1 Επικοινωνία μεταξύ πολλών GPUs του ίδιου υπολογιστικού κόμβου

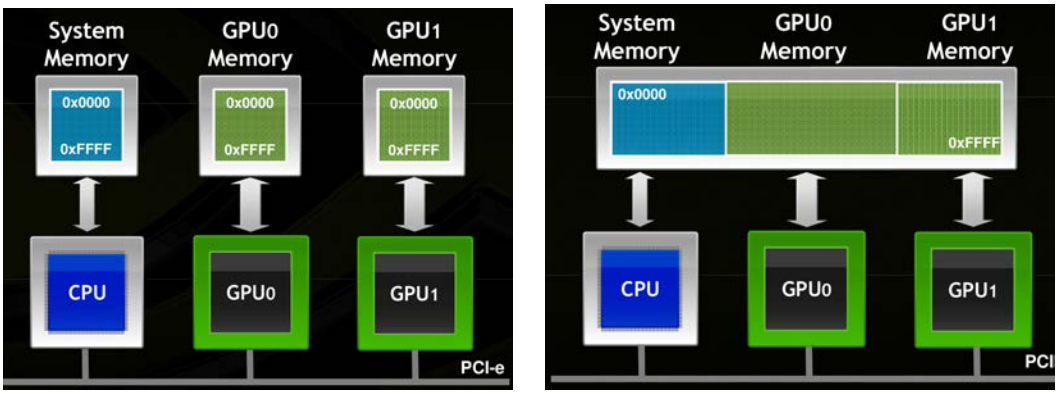
Σε προηγούμενες εκδόσεις του περιβάλλοντος προγραμματισμού της CUDA (3.x ή παλαιότερες εκδόσεις), ένα CPU-thread μπορούσε να διαχειριστεί μόνο μία GPU. Συνεπώς, χρησιμοποιούνταν είτε Pthreads (POSIX threads, [72]), είτε directives OpenMP (Open Multi-Processing, [74, 75]) που επέτρεπαν τη δημιουργία πολλαπλών CPU-threads, κάθε ένα από τα οποία συντόνιζε μία από τις διαθέσιμες στον υπολογιστικό κόμβο GPUs. Η σωστή χρήση και διαχείριση των Pthreads δίνει παράλληλο κώδικα σε πολλές GPUs του ίδιου υπολογιστικού κόμβου υψηλότερης παράλληλης απόδοσης. Ο προγραμματισμός όμως χρησιμοποιώντας directives της βιβλιοθήκης OpenMP είναι σαφώς πιο ευθύς, με λιγότερες αναγκαίες επεμβάσεις στον υπάρχοντα GPU-κώδικα που εκτελείται σε μία κάρτα γραφικών.

Η τέταρτη έκδοση του περιβάλλοντος προγραμματισμού της CUDA επιτρέπει σε ένα CPU-thread τη διαχείριση πολλών GPUs του ίδιου υπολογιστικού κόμβου. Έτσι, αποφεύγεται η δυσχέρεια στον προγραμματισμό μέσω των Pthreads, χωρίς να μειώνεται η παράλληλη απόδοση του παραγόμενου, παράλληλου σε πολλές κάρτες γραφικών του ίδιου υπολογιστικού κόμβου GPU-κώδικα.

Στη διατριβή αυτή χρησιμοποιήθηκαν διαφορετικές εκδόσεις του περιβάλλοντος προγραμματισμού της CUDA. Η ανάπτυξη του GPU-κώδικα ακολούθησε την εξέλιξη του περιβάλλοντος προγραμματισμού σε κάρτες γραφικών. Έτσι, ο τελικός GPU-κώδικας είναι προγραμματισμένος στο τελευταίο περιβάλλον προγραμματισμού της CUDA και χρησιμοποιεί ένα CPU-thread για τη διαχείριση όλων των διαθέσιμων GPUs ανά υπολογιστικό κόμβο.

Η χρήση του τέταρτου περιβάλλοντος προγραμματισμού της CUDA, απλουστεύει ακόμα περισσότερο τον προγραμματισμό παράλληλου σε πολλές κάρτες γραφικών GPU-κώδικα καθώς υποστηρίζει την τεχνολογία Unified Virtual Adressing (UVA), παράγραφος 3.10.1. Έτσι, όλες οι μονάδες επεξεργασίας (CPUs και GPUs), στον ίδιο υπολογιστικό κόμβο, έχουν πρόσβαση σε μία εικονική κοινή μνήμη. Ως αποτέλεσμα, κάθε μονάδα επεξεργασίας γνωρίζει σε ποια μνήμη είναι αποθηκευμένες οι μεταβλητές που χειρίζεται και ακολουθεί την ανάλογη διαδρομή για την ανάγνωση/αποθήκευση αυτών. Εναλλακτικά, η επικοινωνία δύο ή περισσότερων GPUs του ίδιου υπολογιστικού κόμβου γίνεται με τη χρήση ενδιάμεσων pinned θέσεων της μνήμης του υπολογιστή. Παρά την απλούστευση της επικοινωνίας μέσω της χρήσης μίας κοινής εικονικής μνήμης, η χρήση ενδιάμεσων pinned θέσεων μνήμης έχει αυξημένη παράλληλη απόδοση, και χρησιμοποιήθηκε στην παραλληλοποίηση του GPU-κώδικα σε πολλές κάρτες γραφικών του ίδιου υπολογιστικού κόμβου. Το σχήμα 5.1 δείχνει μία CPU και δύο GPUs του ίδιου υπολογιστικού κόμβου, που χειρίζονται (α) διακριτές μνήμες και (β) μία εικονική κοινή μνήμη. Τονίζεται, ότι η επικοινωνία μεταξύ δύο ή περισσότερων GPUs του ίδιου υπολογιστικού κόμβου, είτε γίνεται μέσω ενδιάμεσων pinned θέσεων μνήμης είτε απευθείας (UVA), γίνεται μέσω του διαύλου PCIe. Συνεπώς, η ταχύτητα μεταφοράς δεδομένων εξαρτάται από την ταχύτητα του διαύλου.

Το CPU-thread καλεί μία σειρά από kernels, τα οποία ορίζουν σε ποια GPU του υ-



(α′) (β′)

Σχήμα 5.1: Επικοινωνία μεταξύ GPUs του ίδιου υπολογιστικού κόμβου. Οι μονάδες επεξεργασίας (CPUs και GPUs) του ίδιου υπολογιστικού κόμβου χειρίζονται (α′) διακριτές μνήμες ή (β′) μία κοινή εικονική μνήμη.

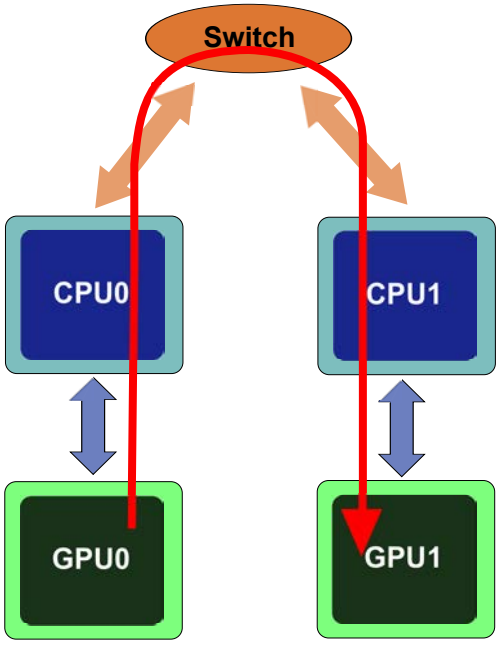
πολογιστικού κόμβου θα εκτελεστούν. Για παράδειγμα, στην περίπτωση χρήσης τριών GPUs, το kernel υπολογισμού των αριθμητικών διανυσμάτων της ροής καλείται διαδοχικά τρεις φορές, μία για κάθε GPU, και όχι μία φορά για όλες τις διαθέσιμες GPUs. Τονίζεται ότι η εκτέλεση οποιουδήποτε kernel γίνεται ασύγχρονα ως προς την εκτέλεση του CPU-thread. Άρα, κατά τη σάρωση των διαθέσιμων GPUs του υπολογιστικού κόμβου, το CPU-thread δεν περιμένει την ολοκλήρωση της εκτέλεσης του kernel που μόλις κάλεσε, αλλά συνεχίζει με την κλήση του ίδιου kernel στην επόμενη GPU. Αυτή είναι μία πολύ σημαντική λειτουργία χωρίς την οποία δεν θα είχε νόημα η δυνατότητα διαχείρισης πολλών GPUs από το ίδιο CPU-thread. Ανάλογη διαχείριση πρέπει να επιβάλλει ο προγραμματιστής και στην αντιγραφή δεδομένων από την κεντρική μνήμη του υπολογιστή στην κεντρική μνήμη των καρτών γραφικών (και αντίστροφα), χρησιμοποιώντας τις ασύγχρονες συναρτήσεις μεταφοράς δεδομένων και όχι εκείνες που επιβάλλουν το συγχρονισμό του CPU-thread (host) με την GPU (device) που (από την οποία) μεταφέρονται τα δεδομένα.

5.1.2 Επικοινωνία μεταξύ πολλών GPUs διασυνδεδεμένων υπολογιστικών κόμβων

Για την επικοινωνία δύο ή περισσότερων GPUs διαφορετικών υπολογιστικών κόμβων είναι απαραίτητη η χρήση ενός πρωτοκόλλου παράλληλης επικοινωνίας μεταξύ διασυνδεδεμένων υπολογιστών, όπως είναι το MPI (Message Passing Interface, [61]), ή το PVM (Parallel Virtual Machine, [60]). Στη διατριβή προτιμήθηκε η χρήση του πρωτοκόλλου MPI. Έτσι, πλέον υπάρχει η ελευθερία χρήσης πολλών GPUs (όσες είναι διαθέσιμες), ανεξάρτητα πού είναι αυτές τοποθετημένες. Δηλαδή, ο παράλληλος σε πολλές κάρτες γραφικών, GPU-κώδικας, δεν περιορίζεται στη χρήση μόνο των GPUs του υπολογιστικού κόμβου. Σημειώνεται ότι μπορεί να χρησιμοποιηθεί το πρωτόκολλο MPI (ή το

PVM) και για την επικοινωνία μεταξύ GPUs του ίδιου υπολογιστικού κόμβου.

Το MPI, (ή το PVM) είναι πρωτόκολλο επικοινωνίας μεταξύ διασυνδεδεμένων CPUs. Η ροή της πληροφορίας κατά την επικοινωνία δύο GPUs διαφορετικών υπολογιστικών κόμβων απεικονίζεται στο σχήμα 5.2. Φαίνεται ότι η GPU0 αρχικά αποθηκεύει (μέσω του διαύλου PCIe) τα ζητούμενα από τη GPU1 δεδομένα στη μνήμη του υπολογιστή 0. Στη συνέχεια, σύμφωνα με το πρωτόκολλο επικοινωνίας MPI, τα ζητούμενα δεδομένα μεταφέρονται στη μνήμη του υπολογιστή 1, από όπου διαβάζονται (μέσω του διαύλου PCIe) από τη GPU1. Προφανώς, η ταχύτητα μεταφοράς των δεδομένων από τον υπολογιστή 0 στον υπολογιστή 1, εξαρτάται από το υλικό (hardware) σύνδεσης των δύο υπολογιστών.



Σχήμα 5.2: Ροή της πληροφορίας κατά την επικοινωνία δύο GPUs διασυνδεδεμένων υπολογιστικών κόμβων.

5.2 Ανάλυση του παράλληλου σε πολλές κάρτες γραφικών GPU-κώδικα

Στην ενότητα αυτή αναλύεται ο παράλληλος σε πολλές κάρτες γραφικών ίδιου ή διαφορετικών υπολογιστικών κόμβων, GPU-κώδικας της διατριβής. Στη συνέχεια της ενότητας, για χάρη ευκολίας, ο παράλληλος σε πολλές κάρτες γραφικών GPU-κώδικας, θα αναφέρεται απλά ως *παράλληλος GPU-κώδικας*. Αν και όπως έχει ήδη αποσαφηνιστεί, οποιαδήποτε εργασία που εκτελείται σε μία μόνο κάρτα γραφικών, εκτελείται παράλληλα από τα threads της κάρτας.

Η επόμενη παράγραφος περιγράφει τις προκλήσεις-δυσκολίες που αντιμετωπίστηκαν κατά την ανάπτυξη του παράλληλου GPU-κώδικα. Στη συνέχεια, παρουσιάζεται ο

τρόπος διάσπασης του υπολογιστικού πλέγματος σε υποχωρία ίδιου πλήθους κόμβων. Η ενότητα κλείνει με την ανάπτυξη προγραμματιστικών τεχνικών με στόχο την αύξηση της παράλληλης απόδοσης του παράλληλου GPU-κώδικα.

Η παράλληλη επιτάχυνση του GPU-κώδικα σε πολλές κάρτες γραφικών του ίδιου ή διασυνδεδεμένων υπολογιστικών κόμβων παρουσιάζεται στην ενότητα 5.3.

5.2.1 Προκλήσεις ανάπτυξης παράλληλου GPU-κώδικα

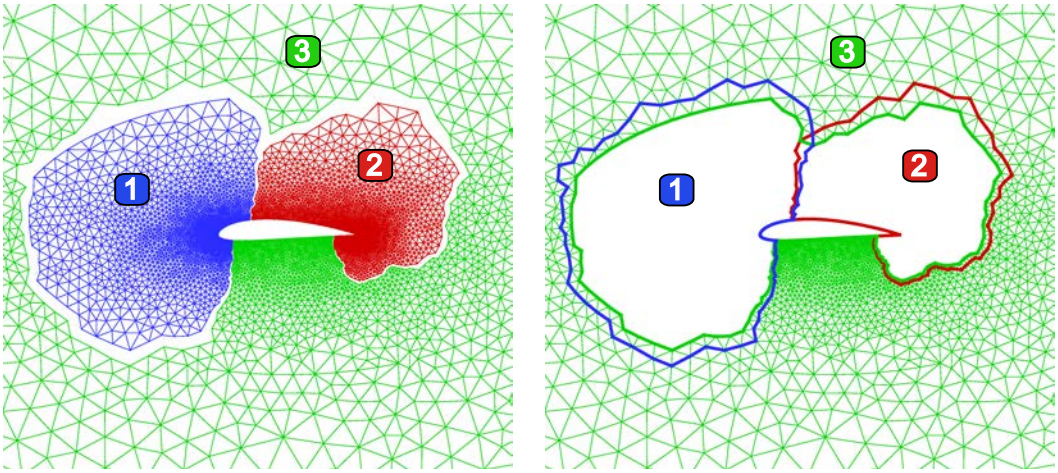
Κατά την επικοινωνία δύο ή περισσότερων GPUs διασυνδεδεμένων υπολογιστικών κόμβων, η διαδρομή που ακολουθεί η πληροφορία (σχήμα 5.2), όπως αναλύθηκε στην ενότητα 5.1.2, αποτελείται από δύο τμήματα: (α) τη μεταφορά δεδομένων μεταξύ της μνήμης του υπολογιστή και της κεντρικής μνήμης της αντίστοιχης κάρτας γραφικών (μέσω του διαύλου PCIe), και (β) την επικοινωνία μεταξύ των διασυνδεδεμένων υπολογιστών σύμφωνα με το χρησιμοποιούμενο πρωτόκολλο επικοινωνίας. Το δεύτερο τμήμα αφορά στις επικοινωνίες κατά την εκτέλεση ενός παράλληλου CPU-κώδικα. Συνεπώς, το κόστος επικοινωνίας ενός παράλληλου GPU-κώδικα αναμένεται να είναι υψηλότερο, εξαιτίας της επιπλέον χρήσης του διαύλου PCIe, σε σχέση με το κόστος επικοινωνίας αντίστοιχου παράλληλου CPU-κώδικα που χρησιμοποιεί το ίδιο πρωτόκολλο επικοινωνίας.

Επιπλέον, από τη στιγμή που ο χρόνος επίλυσης ενός προβλήματος σε μία GPU, σε σχέση με μία σημερινή CPU, είναι πολύ χαμηλότερος (υπενθυμίζεται ότι η μέγιστη επιτάχυνση στη πρόλεξη 2Δ τυρβώδους ροής που καταγράφηκε στην ενότητα 4.5 είναι 110x), αναμένεται το σχετικό κόστος επικοινωνίας, δηλαδή ο χρόνος μεταφοράς δεδομένων μεταξύ των συνεργαζόμενων καρτών γραφικών προς το συνολικό χρόνο εκτέλεσης του παράλληλου GPU-κώδικα, να είναι μεγαλύτερο σε σχέση με τον αντίστοιχο λόγο σε έναν παράλληλο CPU-κώδικα.

5.2.2 Τρόπος διαμερισμού υπολογιστικού πλέγματος

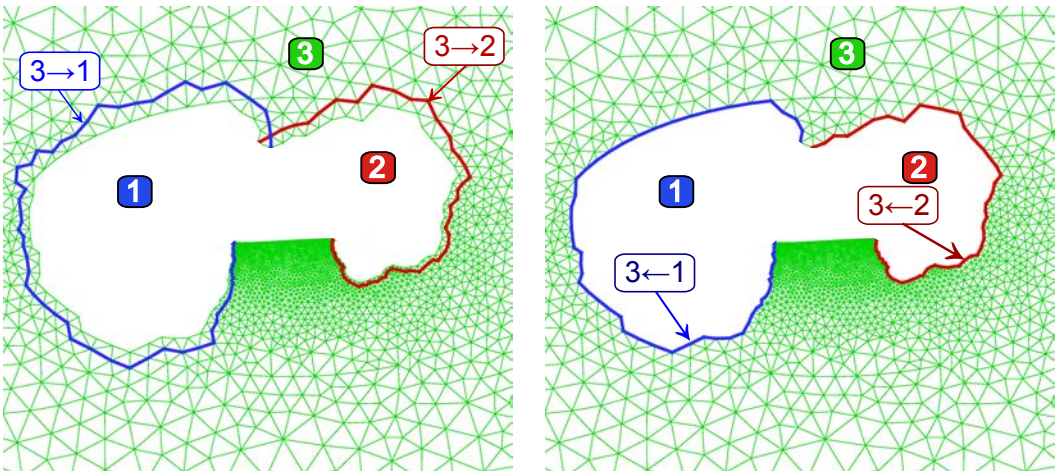
Για τη μείωση του πλήθους των επικοινωνιών επιλέχθηκε η διάσπαση του υπολογιστικού πλέγματος σε επικαλυπτόμενα υποχωρία (overlapped subdomains). Το σχήμα 5.3(α′) δείχνει τη διάσπαση υπολογιστικού πλέγματος γύρω από μεμονωμένη αεροτομή σε τρία επικαλυπτόμενα υποχωρία. Οι κενές ζώνες μεταξύ των υποχωρίων είναι κοινές και ανήκουν και στα δύο γειτονικά υποχωρία. Το σχήμα 5.3(β′) δείχνει το υπολογιστικό πλέγμα του τρίτου (με βάση την αρίθμηση των υποχωρίων του σχήματος 5.3(α′)) υποχωρίου, όπου είναι πιο ευδιάκριτες οι κοινές περιοχές ανάμεσα στα υποχωρία (δηλαδή οι κενές ζώνες του σχήματος 5.3(α′)). Κάθε υποχωρίο συσχετίζεται με μία GPU, η οποία είναι υπεύθυνη για την ολοκλήρωση των εξισώσεων της ροής στο υποχωρίο αυτό.

Οι οριακοί κόμβοι των υποχωρίων, που ανήκουν στις κοινές ζώνες, ορίζονται ως κόμβοι επικοινωνίας, καθώς οι ροϊκές ποσότητες που αποθηκεύονται στους κόμβους αυτούς ανταλλάσσονται μεταξύ των συνεργαζόμενων GPUs. Οι κόμβοι επικοινωνίας



(α′) (β′)

Σχήμα 5.3: (α′) Διάσπαση υπολογιστικού πλέγματος γύρω από μεμονωμένη αεροτομή σε τρία επικαλυπτόμενα υποχωρία. Οι κενές ζώνες μεταξύ των υποχωρίων είναι κοινές για τα γειτονικά υποχωρία. (β′) Υπολογιστικό πλέγμα τρίτου υποχωρίου. Διακρίνονται οι επικαλυπτόμενες περιοχές από τα τρία υποχωρία.



(α′) πληροφορία προς τα υπόλοιπα υποχωρία (β′) πληροφορία από τα υπόλοιπα υποχωρία

Σχήμα 5.4: Κόμβοι επικοινωνίας του τρίτου υποχωρίου. Τα βέλη δείχνουν τη ροή της πληροφορίας μεταξύ των συνεργαζόμενων GPUs.

του τρίτου υποχωρίου φαίνονται με έντονη γραμμή στα σχήματα 5.4(α′), 5.4(β′). Οι κόμβοι επικοινωνίας του σχήματος 5.4(α′) ορίζονται ως κόμβοι 'αποστολής' πληροφορίας στα υπόλοιπα υποχωρία. Αντίθετα, οι κόμβοι επικοινωνίας του σχήματος 5.4(β′) ορίζονται ως κόμβοι 'υποδοχής' πληροφορίας από τα υπόλοιπα υποχωρία. Οι οριακοί κόμβοι του υποχωρίου, που ανήκουν στις κοινές περιοχές, αποτελούν τους κόμβους υποδοχής. Αντίθετα, οι κόμβοι αποστολής είναι οι πρώτοι γείτονες των κόμβων υποδοχής. Οι κόμβοι αποστολής ενός υποχωρίου ταυτίζονται με τους κόμβους υποδοχής

των γειτονικών υποχωρίων και αντίστροφα.

Οι κόμβοι υποδοχής (σχήμα 5.4(β′)) χρησιμοποιούνται, πέρα της επικοινωνίας μεταξύ των συνεργαζόμενων GPUs, μόνο για την αποθήκευση των ροϊκών μεταβλητών. Δηλαδή, η ολοκλήρωση των εξισώσεων της ροής γίνεται στους όγκους ελέγχου που σχηματίζονται γύρω από τους υπόλοιπους κόμβους των υποχωρίων. Συνεπώς, δεν αυξάνεται ο χρόνος επίλυσης ανά υποχωρίο, αφού δεν γίνονται διπλοί υπολογισμοί. Επιπλέον, από τη στιγμή που οι κόμβοι υποδοχής είναι 'ανενεργοί' κατά τον υπολογισμό και ισολογισμό των αριθμητικών διανυσμάτων της ροής στους όγκους ελέγχου, δεν απαιτείται ο υπολογισμός και η αποθήκευση των ιακωβιανών μητρώων D και Z σε αυτούς. Άρα, ως προς τη μνήμη, το επιπλέον κόστος χρήσης επικαλυπτόμενων ζωνών είναι μικρό.

Για το διαμερισμό του υπολογιστικού πλέγματος χρησιμοποιήθηκε το ελεύθερο λογισμικό metis, [133], το οποίο για μικρό αριθμό υποχωρίων (μικρότερο του 8) εγγυάται τη διάσπαση του υπολογιστικού πλέγματος σε υποχωρία ίδιου πλήθους κόμβων, σε λογικό χρόνο. Για παράδειγμα, στο σχήμα 5.3, ένα υποχωρίο αποτελείται από 2577 κόμβους και τα υπόλοιπα δύο από 2578. Ταυτόχρονα, ο αριθμός των ακμών εντός των 'κενών' ζωνών είναι ο ελάχιστος δυνατός. Επιλέχθηκε η διάσπαση του υπολογιστικού πλέγματος σε υποχωρία ίδιου πλήθους κόμβων από τη στιγμή που, όπως φάνηκε στο κεφάλαιο 4, τα πιο χρονοβόρα kernels (ισολογισμός των αριθμητικών διανυσμάτων της ροής στους όγκους ελέγχου, ή ανανέωση των ροϊκών μεταβλητών με τη μέθοδο Jacobi) συσχετίζουν threads με κόμβους του πλέγματος.

5.2.3 Τεχνικές ελάττωσης του κόστους μεταφοράς δεδομένων

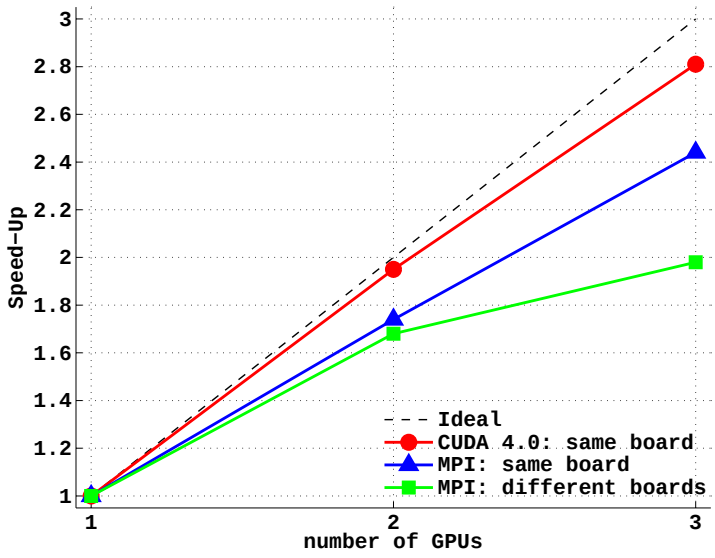
Με στόχο την αύξηση της απόδοσης του παράλληλου GPU-κώδικα, ο χειρισμός των κόμβων του κάθε υποχωρίου χωρίζεται σε δύο τμήματα. Κάθε τμήμα ανατίθεται σε διαφορετικό stream. Έτσι, ένα stream χειρίζεται τους κόμβους επικοινωνίας (υποδοχής και αποστολής) και ένα άλλο τους υπόλοιπους κόμβους του υποχωρίου. Αυτό σημαίνει ότι η επικοινωνία μεταξύ των συνεργαζόμενων GPUs, μέσω ενδιάμεσων pinned θέσεων μνήμης (και το πρωτόκολλο MPI αν χρειάζεται), γίνεται ταυτόχρονα με την επεξεργασία των ροϊκών μεταβλητών των υπολοίπων κόμβων των υποχωρίων από τις GPUs. Έτσι, οι πολυεπεξεργαστές των καρτών γραφικών παραμένουν ενεργοί κατά τη μεταφορά δεδομένων μεταξύ των συνεργαζόμενων GPUs. Αυτό έχει ως αποτέλεσμα την ελάττωση (ιδιαίτερα σημαντική όταν χρησιμοποιούνται διαφορετικοί υπολογιστικοί κόμβοι) του χρόνου εκτέλεσης του παράλληλου GPU-κώδικα.

Η απόδοση οποιουδήποτε παράλληλου λογισμικού εξαρτάται από την απόδοση του εγκατεστημένου υλικού (hardware) μεταφοράς δεδομένων. Για παράδειγμα, ο συνδυασμός της τεχνολογίας GPUDirect της NVIDIA, που υποστηρίζουν οι κάρτες γραφικών αρχιτεκτονικής Fermi, με κάρτες μεταφοράς δεδομένων τελευταίας τεχνολογίας, παρακάμπτει τις CPUs κατά την επικοινωνία μεταξύ των GPUs διασυνδεδεμένων υπολογιστικών κόμβων. Συνεπώς, μειώνεται ο χρόνος μεταφοράς δεδομένων. Σύμφωνα

με την [134], η μείωση στο χρόνο μεταφοράς δεδομένων είναι της τάξης του 30%. Στη διατριβή αυτή δεν χρησιμοποιήθηκαν κάρτες μεταφοράς δεδομένων τελευταίας τεχνολογίας.

5.3 Παράλληλη επιτάχυνση του παράλληλου GPU-επιλύτη

Για τη μέτρηση της παράλληλης επιτάχυνσης του GPU-κώδικα, επιλύθηκε η ατριβής, χρονικά μόνιμη ροή γύρω από πτέρυγα χρησιμοποιώντας δύο με τρεις GPUs ίδιου ή διασυνδεδεμένων υπολογιστικών κόμβων. Οι αριθμητικές προλέξεις δεν συμπεριλαμβάνονται στο κείμενο, καθώς έγιναν με σκοπό τη μέτρηση της παράλληλης απόδοσης του GPU-κώδικα. Υπενθυμίζεται ότι, όταν χρησιμοποιούνται GPUs διαφορετικών υπολογιστικών κόμβων, είναι απαραίτητη η χρήση ενός πρωτοκόλλου επικοινωνίας. Ως τέτοιο, στη διατριβή χρησιμοποιήθηκε το MPI. Αντίθετα, όταν χρησιμοποιούνται GPUs του ίδιου υπολογιστικού κόμβου, ο χρήστης έχει τη δυνατότητα επιλογής ανάμεσα στη χρήση του πρωτοκόλλου MPI, για την επικοινωνία των συνεργαζόμενων καρτών γραφικών, ή στην ανάθεση της διαχείρισης όλων των καρτών γραφικών στο ίδιο CPU-thread. Έχοντας ως βάση το χρόνο επίλυσης του προβλήματος αεροδυναμικής σε μία GPU, το σχήμα 5.5 δείχνει την παράλληλη επιτάχυνση του GPU-κώδικα όταν χρησιμοποιούνται δύο με τρεις κάρτες γραφικών. Στη λεζάντα του σχήματος, η ένδειξη 'CUDA 4.0' αναφέρεται στην ανάθεση της διαχείρισης όλων των συνεργαζόμενων GPUs (του ίδιου υπολογιστικού κόμβου) στο ίδιο CPU-thread. Όπως αναμένονταν, ο παράλληλος GPU-κώδικας έχει υψηλότερη απόδοση όταν χρησιμοποιούνται κάρτες γραφικών του ίδιου υπολογιστικού κόμβου (συντομότερη 'διαδρομή' πληροφορίας). Η επιλογή της ανάθεσης της διαχείρισης των GPUs στο ίδιο CPU-thread δίνει την υψηλότερη παράλληλη επιτάχυνση, η οποία μάλιστα είναι μεγαλύτερη του 2.8x (όταν χρησιμοποιούνται τρεις GPUs του ίδιου υπολογιστικού κόμβου), που είναι πολύ κοντά στο ιδανικό 3.0x.



Σχήμα 5.5: Παράλληλη επιτάχυνση του παράλληλου GPU-κώδικα όταν εκτελείται σε δύο ή τρεις κάρτες γραφικών του ίδιου ή διασυνδεδεμένων υπολογιστικών κόμβων. Όταν χρησιμοποιούνται GPUs διαφορετικών υπολογιστικών κόμβων χρησιμοποιείται το MPI ως πρωτόκολλο επικοινωνίας. Αντίθετα, όταν χρησιμοποιούνται GPUs του ίδιου υπολογιστικού κόμβου, ο χρήστης έχει τη δυνατότητα επιλογής ανάμεσα στη χρήση του MPI ή στην ανάθεση της διαχείρισης όλων των GPUs στο ίδιο CPU-thread (CUDA 4.0).

Κεφάλαιο 6

Επίλυση αεροδυναμικών προβλημάτων σε GPUs

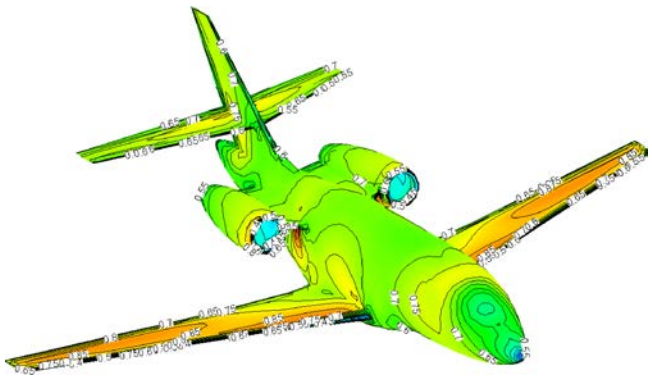
Ο GPU-επιλύτης της διατριβής χρησιμοποιήθηκε για την επίλυση χρονικά μόνιμων και μη-μόνιμων αεροδυναμικών προβλημάτων, τα οποία παρουσιάζονται στο κεφάλαιο αυτό. Όλες οι εφαρμογές του κεφαλαίου αυτού έγιναν αποκλειστικά σε μία GPU. Η επίλυση μεγάλης κλίμακας αεροδυναμικών προβλημάτων σε πολλές GPUs είναι αντικείμενο του κεφαλαίου 9.

Στο κεφάλαιο αυτό, μεταξύ άλλων, τονίζεται το κέρδος από τη χρήση προγραμματιζόμενων GPUs αντί σημερινών CPUs, ως προς τη μείωση του χρόνου επίλυσης αεροδυναμικών προβλημάτων. Για το λόγο αυτό, επιλέχθηκε να επιλυθούν αεροδυναμικά προβλήματα που έχουν μελετηθεί ήδη σε προηγούμενες διατριβές στη ΜΠΥΡ&Β/ΕΘΣ ώστε να τονιστεί η μείωση, από τη χρήση GPUs, στο χρόνο επίλυσης των προβλημάτων αυτών. Επισημαίνεται, ότι η μικτής ακρίβειας (MPA) εκδοχή του GPU-επιλύτη, που χρησιμοποιήθηκε στις εφαρμογές του κεφαλαίου, αναπαράγει τις αριθμητικές προλέξεις του CPU-κώδικα, ο οποίος έχει πιστοποιηθεί επαρκώς στις διατριβές [135, 129, 130, 136]. Συνεπώς, όπως αναμένονταν, τα αριθμητικά αποτελέσματα του GPU-κώδικα είναι πολύ κοντά στα αντίστοιχα πειραματικά δεδομένα, τα οποία, όπου είναι διαθέσιμα, δείχνονται επίσης.

Σε όλες τις εφαρμογές του κεφαλαίου χρησιμοποιήθηκαν GPUs μοντέλου GTX 285, τεχνολογίας του 2008. Για τον υπολογισμό της επιτάχυνσης στην επίλυση της ροής από τη χρήση καρτών γραφικών, χρησιμοποιήθηκε ως βάση, ο χρόνος επίλυσης του ίδιου προβλήματος σε έναν πυρήνα μιας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη. Δηλαδή, όλες οι επιταχύνσεις που αναφέρονται στο κεφάλαιο αυτό έχουν ως αναφορά την απόδοση του CPU-επιλύτη στην παραπάνω CPU.

6.1 Ανάλυση ατριβούς ροής γύρω από αεροσκάφος

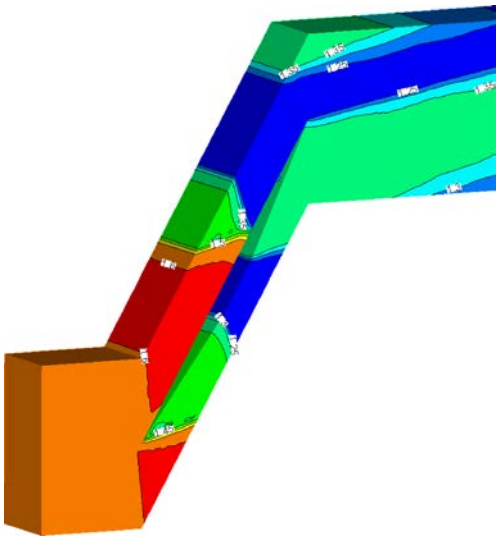
Ο GPU-κώδικας χρησιμοποιήθηκε στην ανάλυση ατριβούς ροής γύρω από μοντέλο επιβατικού αεροσκάφους. Το σχήμα 6.1 δείχνει την κατανομή του αριθμού Mach στην επιφάνεια αυτού, όταν ο αριθμός Mach της πτήσης (M_∞) είναι 0.7 και η γωνία της επ΄ άπειρο ροής (α_∞) $0.^o$. Χρησιμοποιήθηκε μη-δομημένο πλέγμα, αποτελούμενο από τετραεδρικά πλεγματικά στοιχεία, με περίπου 45000 κόμβους. Η ροή γύρω από αυτό επιλύθηκε 37 φορές πιο γρήγορα. Ο χρόνος που απαιτήθηκε για τη σειριακή επίλυση τους παρόντος προβλήματος σε μία σημερινή CPU είναι περίπου 40 λεπτά. Αντίθετα, τα ίδια αποτελέσματα προλέχθηκαν σε λιγότερο από 2 λεπτά χρησιμοποιώντας μία GPU τεχνολογίας του 2008.



Σχήμα 6.1: Πρόλεξη ατριβούς ροής γύρω από αεροσκάφος ($M_\infty = 0.7$, $\alpha_\infty = 0^o$): υπολογισθέν πεδίο αριθμού Mach.

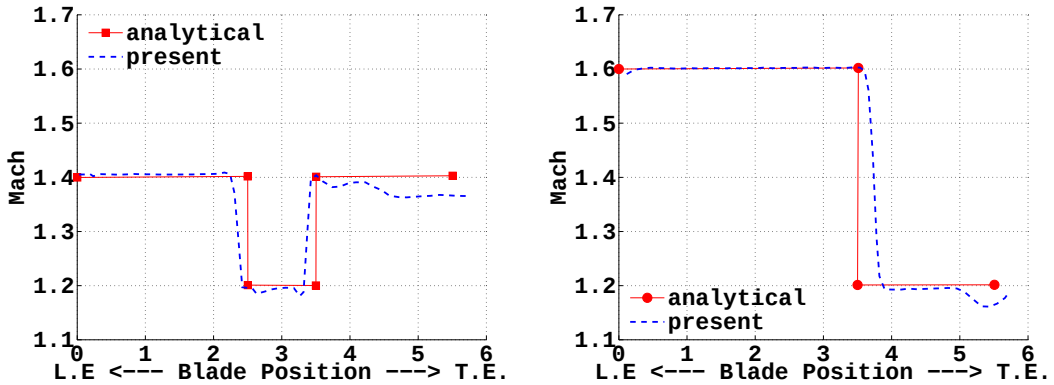
6.2 Ανάλυση ατριβούς ροής σε γραμμική πτερύγωση υπερηχητικού συμπιεστή

Μελετήθηκε η συμπεριφορά ατριβούς ροής σε γραμμική πτερύγωση υπερηχητικού συμπιεστή. Η ροή στην είσοδο του συμπιεστή έχει αριθμό Mach (M_1) ίσο με 1.6 και κατεύθυνση παράλληλη με την πλευρά υποπίεσης που είναι επίπεδη. Συγκεκριμένα, η γωνία εισόδου της ροής (α_1) είναι 60^o. Η ακμή πρόσπτωσης του πτερυγίου έχει σχήμα σφήνας που οδηγεί στο σχηματισμό πλάγιου κυμάτος κρούσης. Το κύμα αυτό ανακλάται στην πλευρά υποπίεσης και, στη συνέχεια, στην πλευρά υπερπίεσης όπου και εξαφανίζεται εκεί που η πλευρά υπερπίεσης γίνεται παράλληλη της πλευράς υποπίεσης. Περίπου στο 65% της χορδής, η πλευρά υπερπίεσης αλλάζει πάλι κατεύθυνση και κινείται προς την πλευρά υποπίεσης για το σχηματισμό της ακμής εκφυγής. Η αλλαγή αυτή στη γεωμετρία συνοδεύεται από ένα σύστημα κυμάτων αποτόνωσης. Τα παραπάνω φαίνονται και στο σχήμα 6.2, όπου φαίνεται η κατανομή του αριθμού Mach στο εσωτερικό της πτερύγωσης. Για την αναλυτική περιγραφή της γεωμετρίας των πτερυγίων ο αναγνώστης παραπέμπεται στο [137].



Σχήμα 6.2: Πρόλεξη ατριβούς ροής σε γραμμική πτερύγωση υπερηχητικού συμπιεστή, [137], ($M_1 = 1.6$ και $\alpha_1 = 60^o$): υπολογισθέν πεδίο αριθμού Mach.

Το συγκεκριμένο πρόβλημα έχει αναλυτική λύση, [137], που προκύπτει από τη θεωρία χαρακτηριστικών και τις σχέσεις πλάγιων κυμάτων κρούσης. Η σύγκριση της κατανομής του υπολογισμένου αριθμού Mach κατά μήκος του πτερυγίου με την αναλυτική λύση φαίνεται στο σχήμα 6.3. Παρατηρείται ικανοποιητική πρόλεξη με εξαίρεση το κύμα αποτόνωσης στην πλευρά υπερπίεσης. Όπως αναμένονταν, τα ίδια αριθμητικά αποτελέσματα παρουσιάζονται και στη διατριβή του δόκτορα Ν. Λαμπρόπουλου, [136], που εκπονήθηκε στη ΜΠΥΡ&Β/ΕΜΠ και χρησιμοποίησε τον CPU-κώδικα.



(α′) Πλευρά υπερπίεσης (β′) Πλευρά υποπίεσης

Σχήμα 6.3: Πρόλεξη ατριβούς ροής σε γραμμική πτερύγωση υπερηχητικού συμπιεστή ($M_1 = 1.6$ και $\alpha_1 = 60.^o$): σύγκριση της υπολογισμένης κατανομής του αριθμού Mach κατά μήκος της αεροτομής του πτερυγίου με την αναλυτική λύση, [137].

Για την πρόλεξη υψηλής ακρίβειας αριθμητικών αποτελεσμάτων το μη-δομημένο πλέγμα που χρησιμοποιήθηκε είναι πιο πυκνό στις περιοχές των κρουστικών κυμάτων, [138], και αποτελείται περίπου από 45000 κόμβους.

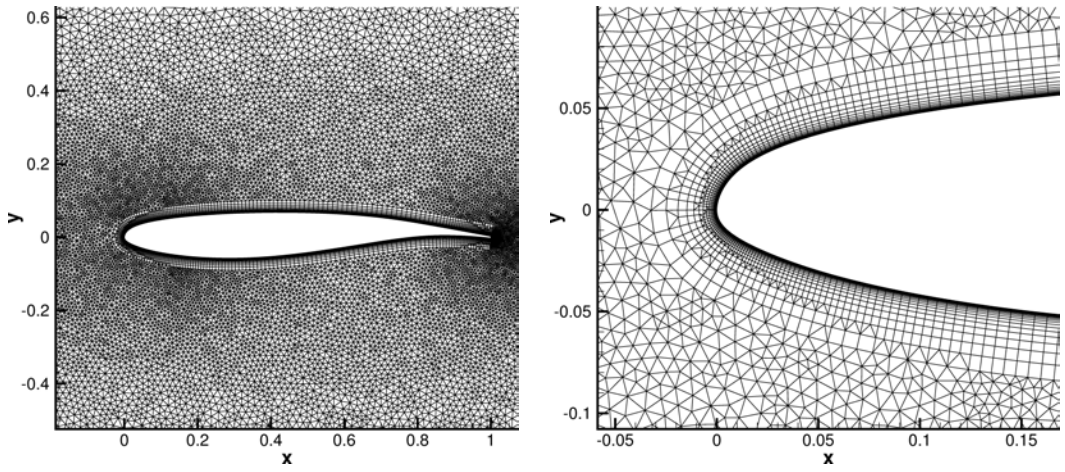
Η ολοκλήρωση των εξισώσεων ατριβούς ροής συμπιεστού ρευστού έγινε 37 φορές πιο γρήγορα χρησιμοποιώντας μία GTX 285.

6.3 Αλληλεπίδραση κύματος κρούσης με οριακό στρώμα

Η επόμενη εφαρμογή σχετίζεται με τη μελέτη του φαινομένου δυναμικής αλληλεπίδρασης οριακού στρώματος με κύμα κρούσης (buffet) σε μεμονωμένη αεροτομή, [129]. Το φαινόμενο buffet είναι αυτοδιατηρούμενο και προκαλεί την ταλάντωση της θέσης του κύματος κρούσης κατά μήκος της χορδής της αεροτομής. Το πλάτος και η συχνότητα της ταλάντωσης αυτής εξαρτώνται από τη γεωμετρία της αεροτομής και τις συνθήκες της ροής. Η ταλάντωση του κύματος κρούσης προκαλεί τη δυναμική φόρτιση της αεροτομής, η οποία μπορεί να προκαλέσει ακόμα και αστοχία της αεροτομής, μειώνοντας την περιοχή ευσταθούς λειτουργίας αυτής. Διεύρυνση της περιοχής ευσταθούς λειτουργίας μπορεί να γίνει με παθητικό ή ενεργητικό έλεγχο της ροής με τη χρήση γεννητριών δινών ή συστήματος ελέγχου της κίνησης της ακμής εκφυγης της αεροτομής (flap actuator), αντίστοιχα, [139, 140, 141].

Ο GPU-επιλύτης χρησιμοποιήθηκε για τη μελέτη του φαινομένου buffet στην αεροτομή OAT15A. Η ροή γύρω από την συγκεκριμένη αεροτομή έχει μελετηθεί υπολογιστικά, χρησιμοποιώντας CPUs, και από άλλες ερευνητικές ομάδες ([139, 142, 143, 140, 141]), και στη διατριβή [129] που εκπονήθηκε στη ΜΠΥΡ&Β/ΕΘΣ. Στις εργασίες [139, 142, 143, 140, 141] μελετήθηκε η επίδραση του μοντέλου τύρβης στη σωστή πρόλεξη του φαινομένου. Στη διατριβή χρησιμοποιείται το μοντέλο τύρβης μίας εξίσωσης των Spalart-Allmaras, [53]. Η ροή γύρω από την αεροτομή έχει μελετηθεί και πειραματικά στη διηχητική αεροδυναμική σήραγγα S3 της ONERA [144]. Οι μετρήσεις, όπως και λεπτομέρειες σχετικά με την πειραματική διάταξη βρίσκονται στην τεχνική έκθεση [145]. Στις υπολογιστικές και πειραματικές μετρήσεις επιβλήθηκε σταθερό σημείο μετάβασης στο 7% της χορδής της αεροτομής και στις δύο πλευρές της. Οι πειραματικές μετρήσεις έγιναν για αριθμό Mach της επ΄ άπειρο ροής $M_\infty = 0.73$, με αριθμό Reynolds ως προς τη χορδή της αεροτομής $Re_c = 3 \times 10^6$ και γωνίες της επ΄ άπειρο ροής $\alpha_\infty = 1.36^o$–3.9^o. Σύμφωνα με τα πειράματα το φαινόμενο buffet εμφανίζεται για γωνίες της απ΄ άπειρο ροής μεγαλύτερες των 3.25^o. Παρόλα αυτά, οι περισσότερες πειραματικές μετρήσεις έχουν γίνει για γωνία $\alpha_\infty = 3.5^o$. Σ΄ αυτήν τη γωνία, το πλάτος της ταλάντωσης της θέσης του κύματος κρούσης κατά μήκος της χορδής της αεροτομής είναι περίπου ίσο με το 20% της χορδής και έχει συχνότητα 69 Hz.

Στη συνέχεια, παρουσιάζονται οι περιπτώσεις χρονικά μη-μόνιμης ροής που επιλύθηκαν χρησιμοποιώντας την MPA εκδοχή του GPU-κώδικα. Οι ίδιες περιπτώσεις ροής έχουν μελετηθεί και από την δόκτορα Β. Ασούτη στη διατριβή της, [129], που εκπονήθηκε στη ΜΠΥΡ&Β/ΕΘΣ όπου χρησιμοποιήθηκε ο υπάρχων CPU-επιλύτης. Εφόσον η MPA εκδοχή του GPU-κώδικα αναπαράγει τις αριθμητικές προλέξεις του CPU-επιλύτη της ροής, τα αριθμητικά αποτελέσματα που παρουσιάζονται στο κεφάλαιο ταυτίζονται με εκείνα που παρουσιάζονται στη διατριβή [129]. Στη συνέχεια δείχνεται, επίσης, η σύγκριση με πειραματικά δεδομένα.



(α′) (β′)

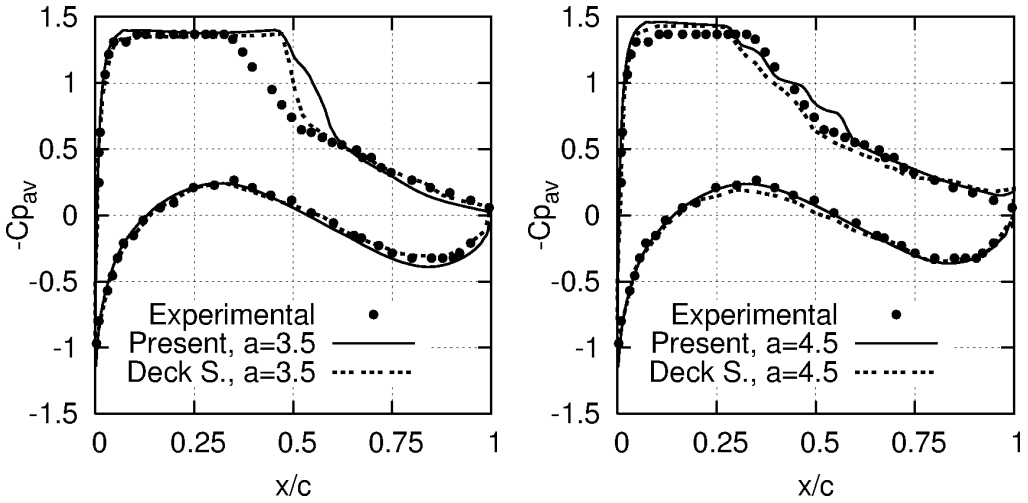
Σχήμα 6.4: Μελέτη του φαινομένου buffet στην αεροτομή OAT15A: το υβριδικό πλέγμα που χρησιμοποιήθηκε. Το σχήμα (β′) δείχνει λεπτομέρεια του πλέγματος στην περιοχή της ακμής πρόσπτωσης.

Αρχικά, ο GPU-κώδικας χρησιμοποιήθηκε για τον υπολογισμό της ροής όταν $\alpha_\infty = 3.5^o$. Χρησιμοποιήθηκε υβριδικό υπολογιστικό πλέγμα (σχήμα 6.4(α′)) αποτελούμενο από τετραπλευρικά στοιχεία κοντά στην αεροτομή, για την ακριβέστερη πρόλεξη του οριακού στρώματος, και τριγωνικά στο υπόλοιπο του χωρίου της ροής. Μία μεγέθυνση του πλέγματος κοντά στην ακμή πρόσπτωσης φαίνεται στο σχήμα 6.4(β′). Η συχνότητα ταλάντωσης της θέσης του κύματος κρούσης με βάση τη παρούσα πρόλεξη είναι 65.6 Hz. Η τιμή αυτή είναι ικανοποιητική συγκρινόμενη με την πειραματικά υπολογισμένη τιμή των 69 Hz. Σημειώνεται ότι, σε αριθμητικούς υπολογισμούς άλλων ερευνητικών ομάδων [139, 141], η τιμή της συχνότητας ταλάντωσης υπερεκτιμάται (73-74 Hz). Η κατανομή της χρονικά μέσης τιμής του συντελεστή πίεσης κατά μήκος της χορδής της αεροτομής και το rms πεδίο της οριζόντιας ταχύτητας της ροής γύρω από την αεροτομή φαίνονται στα σχήματα 6.5(α′), 6.6(α′), αντίστοιχα. Το σχήμα 6.5(α′) παρουσιάζει επίσης τη σύγκριση των αριθμητικών αποτελεσμάτων της παρούσας πρόλεξης με αριθμητικά αποτελέσματα άλλης ερευνητικής ομάδας [139] και πειραματικά [145].

Προκειμένου να ελεγχθεί ο ισχυρισμός του Deck, [139] ότι τα αριθμητικά αποτελέσματα για $\alpha_\infty = 4.5^o$ είναι πλησιέστερα στα πειραματικά για $\alpha_\infty = 3.5^o$, επιλύθηκε η ροή γύρω από την αεροτομή με $\alpha_\infty = 4.5^o$. Η κατανομή της χρονικά μέσης τιμής του συντελεστή πίεσης κατά μήκος της χορδής της αεροτομής και το rms πεδίο της οριζόντιας ταχύτητας της ροής φαίνονται στα σχήματα 6.5(β′), 6.6(β′), αντίστοιχα. Φαίνεται ότι πράγματι, τα υπολογιστικά αποτελέσματα για $\alpha_\infty = 4.5^o$ είναι αρκετά πλησιέστερα στα πειραματικά για $\alpha_\infty = 3.5^o$. Η αύξηση της γωνίας της επ΄ άπειρο ροής αυξάνει το πλάτος της ταλάντωσης της θέσης του οριακού στρώματος κατά μήκος της χορδής της αεροτομής. Για το λόγο αυτό, μειώνεται η κλίση της κατανομής της χρονικά μέσης τιμής του συντελεστή πίεσης στη περιοχή που εμφανίζεται το κύμα κρούσης. Η αύξηση του πλάτους της ταλάντωσης φαίνεται και στα σχήματα 6.7 και 6.8, τα οποία δείχνουν το πεδίο του αριθμού Mach γύρω από την αεροτομή σε διαφορετικές χρονικές

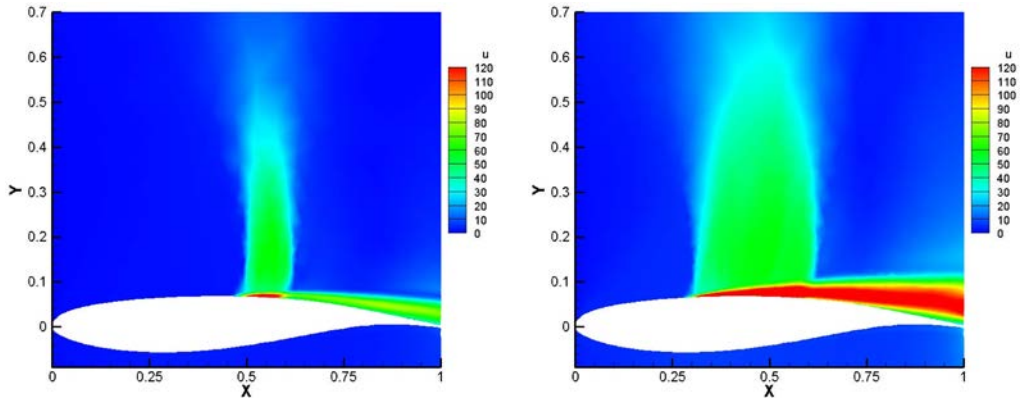
στιγμές, αφού επέλθει το περιοδικό φαινόμενο, για τις περιπτώσεις με $\alpha_\infty = 3.5^o$ και 4.5^o αντίστοιχα.

Το υβριδικό πλέγμα που χρησιμοποιήθηκε (σχήμα 6.4(α′)) αποτελείται από περίπου 130000 κόμβους, 230000 τριγωνικά και 12000 τετραπλευρικά στοιχεία. Η επιτάχυνση στην πρόλεξη των αριθμητικών αποτελεσμάτων από τη χρήση μίας GTX 285 είναι περίπου 45x. Τα πλεονεκτήματα από τη χρήση GPUs έναντι CPUs αναδεικνύονται καλύτερα στην παρούσα εφαρμογή σε σχέση με τις εφαρμογές που προηγήθηκαν εξαιτίας του χρονικά μη-μόνιμου χαρακτήρα της ροής. Η πρόλεξη της ροής σε μία GTX 285 χρησιμοποιώντας τη μικτής ακρίβειας εκδοχή του GPU-κώδικα απαιτεί περίπου 1.5 ώρα. Αν αντί του GPU-κώδικα, χρησιμοποιούνταν ο σειριακός CPU-επιλύτης, ο απαιτούμενος χρόνος θα ήταν περίπου 3 ημέρες. Η επιτάχυνση που συνεπάγεται η χρήση των GPUs παρέχει τη δυνατότητα στο μηχανικό να χρησιμοποιεί πυκνότερα πλέγματα ή μικρότερα χρονικά βήματα, για την ακριβέστερη πρόλεξη της ροής.



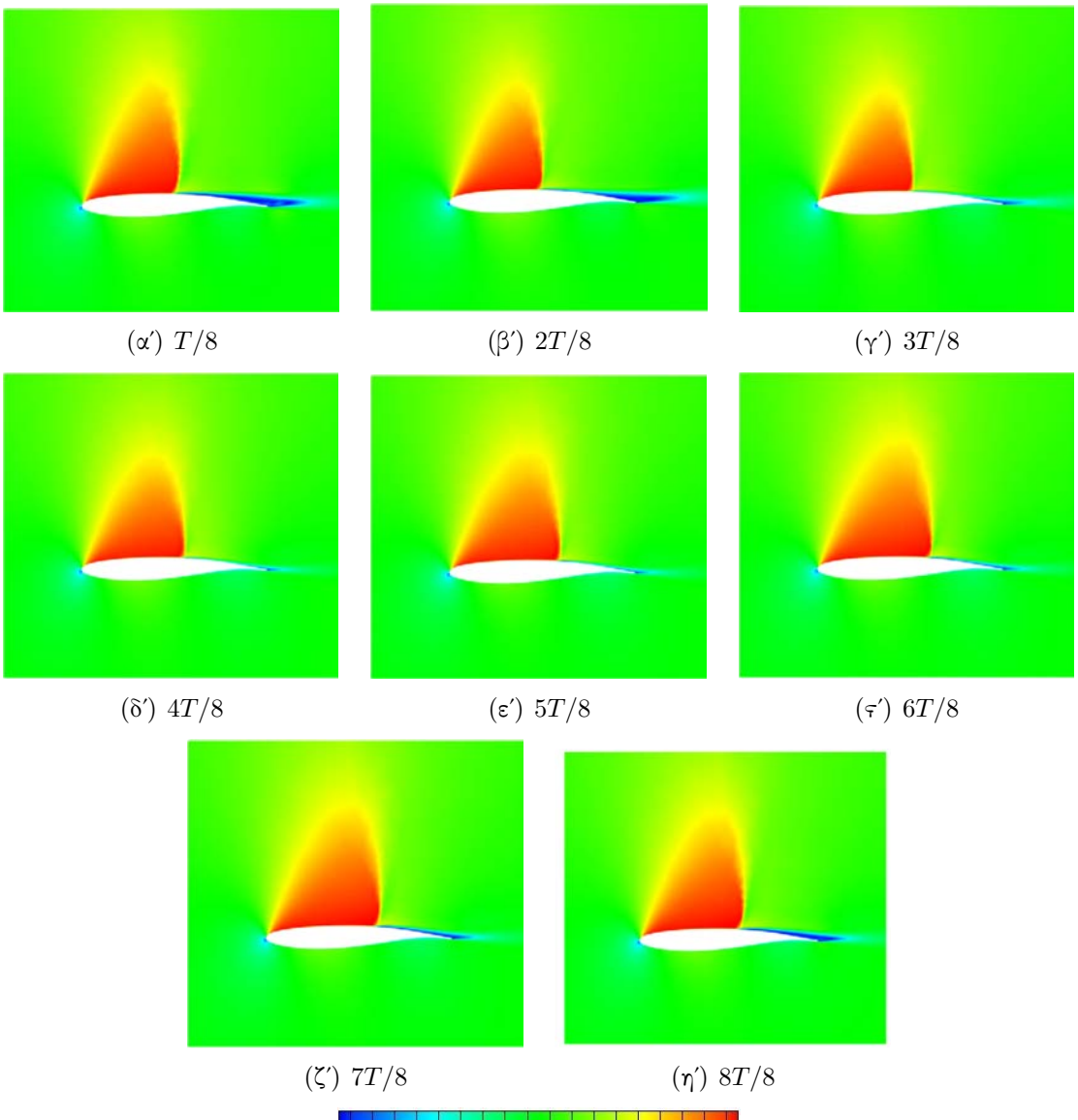
(α′) (β′)

Σχήμα 6.5: Μελέτη του φαινομένου buffet στην αεροτομή OAT15A: κατανομή της χρονικά μέσης τιμής του συντελεστή πίεσης κατά μήκος της χορδής της αεροτομής με $M_\infty = 0.73$, $Re_c = 3\times10^6$ και (α′) $\alpha_\infty = 3.5^o$ ή (β′) $\alpha_\infty = 4.5^o$. Σύγκριση υπολογισθέντων αποτελεσμάτων με υπολογιστικά, [139], και πειραματικά, [145], αποτελέσματα. Οι αριθμητικές προλέξεις του GPU-επιλύτη ταυτίζονται με τα αποτελέσματα του CPU-κώδικα που παρουσιάζονται στη διατριβή [129].



(α′) (β′)

Σχήμα 6.6: Μελέτη του φαινομένου buffet στην αεροτομή OAT15A: rms πεδίο της οριζόντιας ταχύτητας της ροής για $M_\infty = 0.73$, $Re_c = 3\times10^6$ και (α′) $\alpha_\infty = 3.5^o$ ή (β′) $\alpha_\infty = 4.5^o$. Οι αριθμητικές προλέξεις του GPU-επιλύτη ταυτίζονται με τα αποτελέσματα του CPU-κώδικα που παρουσιάζονται στη διατριβή [129].



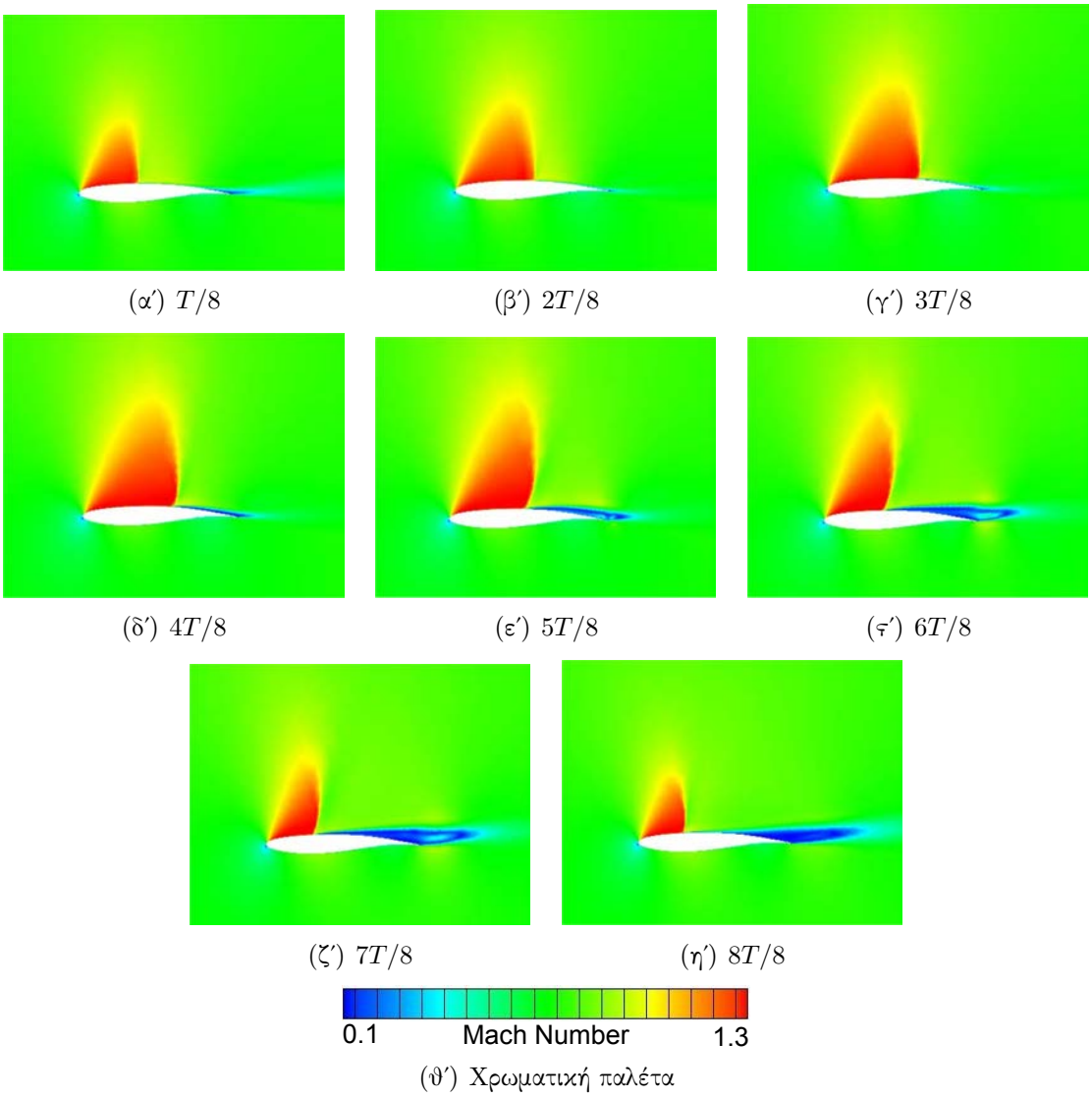
(α′) $T/8$ (β′) $2T/8$ (γ′) $3T/8$

(δ′) $4T/8$ (ε′) $5T/8$ (ϛ′) $6T/8$

(ζ′) $7T/8$ (η′) $8T/8$

(ϑ′) Χρωματική παλέτα

Σχήμα 6.7: Μελέτη του φαινομένου buffet στην αεροτομή OAT15A: υπολογισθέντα πεδία του αριθμού Mach γύρω από την αεροτομή σε διαδοχικές χρονικές στιγμές ανά $T/8$, όπου T η περίοδος του φαινομένου ($M_\infty = 0.73$, $Re_c = 3 \times 10^6$, $\alpha_\infty = 3.5^o$). Η χρωματική παλέτα (ϑ′) είναι κοινή για όλα τα πεδία.



(α′) $T/8$ (β′) $2T/8$ (γ′) $3T/8$

(δ′) $4T/8$ (ε′) $5T/8$ (ϛ′) $6T/8$

(ζ′) $7T/8$ (η′) $8T/8$

(ϑ′) Χρωματική παλέτα

Σχήμα 6.8: Μελέτη του φαινομένου buffet στην αεροτομή OAT15A: υπολογισθέντα πεδία του αριθμού Mach γύρω από την αεροτομή σε διαδοχικές χρονικές στιγμές ανά $T/8$, όπου T η περίοδος του φαινομένου ($M_\infty = 0.73$, $Re_c = 3 \times 10^6$, $\alpha_\infty = 4.5^o$). Η χρωματική παλέτα (ϑ′) είναι κοινή για όλα τα πεδία.

6.4 Έλεγχος της ροής

Οι τεχνικές ελέγχου της ροής χρησιμοποιούνται για τη βελτίωση της αεροδυναμικής συμπεριφοράς σωμάτων και διακρίνονται σε παθητικές και ενεργητικές. Ακολουθεί μία σύντομη περιγραφή των κυριότερων τεχνικών ελέγχου της ροής που συναντώνται στη βιβλιογραφία και στη συνέχεια παρουσιάζεται μία εφαρμογή με έλεγχο της ζώνης αποκόλλησης κατάντι εμποδίου στο εσωτερικό αγωγού, χρησιμοποιώντας GPUs.

6.4.1 Παθητικές τεχνικές ελέγχου της ροής

Κλασική τεχνική ελέγχου της ροής, παθητικού τύπου, με ευρεία χρήση στη βιομηχανία αεροσκαφών είναι οι γεννήτριες δινών (vortex generators). Αυτές είναι ελάσματα, συνήθως με σχήμα σφήνας, που τοποθετούνται στην επιφάνεια της πτέρυγας. Η ροή γύρω από επίπεδη πλάκα ή αεροτομή με εγκατεστημένη σειρά γεννητριών δινών, έχει μελετηθεί πειραματικά στις εργασίες [146, 147, 148]. Σημαντικοί παράμετροι στην επίδραση των γεννητριών δινών στη μείωση της ζώνης αποκόλλησης αποτελούν η θέση εγκατάστασης, η μορφή και το ύψος αυτών. Το ύψος είναι ανάλογο του πάχους του οριακού στρώματος στη θέση εγκατάστασης. Πειράματα σε πραγματική πτέρυγα στη σήραγγα χαμηλών διαταραχών της ροής στην είσοδο σε κέντρο ερευνών της NASA (NASA Langley Research Center - LaRC), [149], έδειξαν ότι η χρήση των vortex generators μειώνει σημαντικά τη ζώνη αποκόλλησης. Η μείωση αυτή αυξάνει το συντελεστή άνωσης και μειώνει το συντελεστή οποσθέλκουσας της πτέρυγας. Επίσης αυξάνει την ελεγξιμότητα του αεροσκάφους καθώς η ροή γύρω από τις επιφάνειες ελέγχου της πορείας αυτού είναι λιγότερο ή καθόλου αποκολλημένη. Επιπλέον, αυξάνεται η γωνία απώλειας στήριξης, όπως και η γωνία επ΄ άπειρο ροής όπου εμφανίζεται το φαινόμενο buffet, όπως συζητήθηκε στην ενότητα 6.3. Τονίζεται, ότι η μείωση του συντελεστή οπισθέλκουσας, μειώνει τις απαιτήσεις του αεροσκάφους σε ώση, και οδηγεί σε μικρότερη κατανάλωση καυσίμων. Η επίδραση των γεννητριών δινών στην αεροδυναμική συμπεριφορά αεροτομής έχει μελετηθεί και υπολογιστικά στην εργασία [150]. Οι γεννήτριες δινών βρίσκουν εφαρμογή και στην αυτοκινητοβιομηχανία, [151].

Εναλλακτική τεχνική παθητικού τύπου είναι η επικάλυψη τμήματος της επιφάνειας του εξεταζόμενου σώματος με πορώδες υλικό. Η μείωση του εύρους της περιοχής αποκόλλησης της ροής κατάντι απότομης διαμόρφωσης (backstep flow), με τη χρήση στρώματος πορώδους υλικού έχει μελετηθεί πειραματικά στην εργασία [152]. Η πειραματική μελέτη της επίδρασης της μεθόδου στη μείωση του συντελεστή οπισθέλκουσας και αύξηση του συντελεστή άνωσης αεροτομής είναι αντικείμενο της εργασίας [153]. Οι εργασίες [154, 155], χρησιμοποιώντας υπολογιστικά μέσα, επιτυγχάνουν τη μείωση του συντελεστή απισθέλκουσας σε μοντέλο αυτοκινήτου (Ahmed body) με τη χρήση στρώματος πορώδους υλικού.

6.4.2 Ενεργητικές τεχνικές ελέγχου της ροής

Κύριοι εκπρόσωποι της κατηγορίας των ενεργητικών τεχνικών ελέγχου της ροής είναι οι τεχνικές χρήσης δεσμών συνεχούς αναρρόφησης (steady suction) ή έγχυσης (steady blowing) ρευστού ή σύνθετων δεσμών (synthetic jets). Στις τεχνικές ελέγχου της ροής μέσω δέσμης συνεχούς αναρρόφησης ή έγχυσης, ρευστό αναρροφάται ή εγχύεται μέσω οπής στην επιφάνεια του εξεταζόμενου αεροδυναμικού σώματος, σχηματίζοντας μία δέσμη συνεχούς αναρρόφησης ή έγχυσης, αντίστοιχα. Αντίθετα, σε μία σύνθετη δέσμη οι φάσεις αναρρόφησης/έγχυσης εναλλάσσονται, με την παροχή του ρευστού που διέρχεται από την οπή να δίνεται ανά χρονική στιγμή από μία περιοδική συνάρτηση. Πρακτικά, ο σχηματισμός της σύνθετης δέσμης μπορεί να οφείλεται στην ταλάντωση μεμβράνης που έχει εγκατασταθεί εσωτερικά της οπής. Η συχνότητα και το πλάτος ταλάντωσης της μεμβράνης καθορίζονται από κατάλληλο σύστημα ελέγχου. Η δυνατότητα χρήσης συστήματος ελέγχου που προσαρμόζει τη δέσμη στις συνθήκες της ροής διαφοροποιεί τις τεχνικές ελέγχου της ροής μέσω δέσμης ρευστού και, ειδικότερα, τις ενεργητικές τεχνικές ελέγχου της ροής από τις παθητικές, όπου δεν υπάρχει η δυνατότητα δυναμικής προσαρμογής του συστήματος ελέγχου στις συνθήκες της ροής.

Η επίδραση της τεχνικής χρονικά μόνιμης ή μεταβαλλόμενης έγχυσης ρευστού, στη μείωση της ζώνης αποκόλλησης κατάντι διαμόρφωσης έχει μελετηθεί πειραματικά στις εργασίες [156, 157, 158, 159]. Επίσης έχει μελετηθεί πειραματικά η επίδραση δέσμης ροής στη ροή ρευστού πάνω σε επίπεδη πλάκα, [160, 161]. Μάλιστα, στην εργασία [160] χρησιμοποιούνται ενεργητικές τεχνικές ελέγχου της ροής για την αποφυγή του φαινομένου buffet. Αντίθετα, στις εργασίες [162, 152, 163, 164] μελετάται υπολογιστικά η επίδραση χρήσης σύνθετης δέσμης στη μείωση της ζώνης αποκόλλησης κυρίως στην πλευρά υποπίεσης αεροτομής.

Η παραμετρική μελέτη της επίδρασης δέσμης συνεχούς αναρρόφησης ή έγχυσης στη μείωση της ζώνης αποκόλλησης κατάντι εμποδίου εντός αγωγού ή στην πλευρά υποπίεσης αεροτομής γίνεται με υπολογιστικά μέσα στις εργασίες [165, 166]. Οι εργασίες [167, 168] προχωρούν ένα βήμα παρακάτω στην εύρεση των βέλτιστων παραμέτρων (για συγκεκριμένη γωνία της επ΄ άπειρο ροής) σύνθετης δέσμης, με στόχο τη μεγιστοποίηση του συντελεστή άνωσης αεροτομής. Οι εν λόγω εργασίες χρησιμοποιούν αιτιοκρατική μέθοδο προσδιορισμού της βέλτιστης λύσης. Η εργασία [169] υπολογίζει το χάρτη παραγώγων ευαισθησίας της συνάρτησης κόστους (απώλειες ολικής πίεσης αγωγού) ως προς το μέτρο της υποθετικής ταχύτητας αναρρόφησης/έγχυσης με τη συνεχή συζυγή μέθοδο. Στη διδακτορική διατριβή [170], χρησιμοποιείται γενετικός αλγόριθμος για τον προσδιορισμό των βέλτιστων παραμέτρων δύο δεσμών (μία δέσμη συνεχούς αναρρόφησης και μία συνεχούς έγχυσης) στην πλευρά υποπίεσης της αεροτομής, με στόχους την ελαχιστοποίηση του συντελεστή οπισθέλκουσας και τη μεγιστοποίηση του συντελεστή άνωσης.

Ο συνδυασμός παθητικών (επίστρωση της επιφάνειας του εξεταζόμενου σώματος με πορώδες υλικό ή εγκατάσταση κατάλληλης διαμόρφωσης σ΄ αυτή) και ενεργητικών (μέσω δέσμης συνεχούς έγχυσης ή αναρρόφησης σταθερής ή χρονικά μεταβαλλόμενης παροχής ρευστού) μεθόδων στην ελάττωση του συντελεστή οπισθέλκουσας μοντέλου αυτοκινήτου (Ahmed body) ή αεροτομής παρουσιάζεται στις εργασίες [155, 171] αντί-

στοιχα.

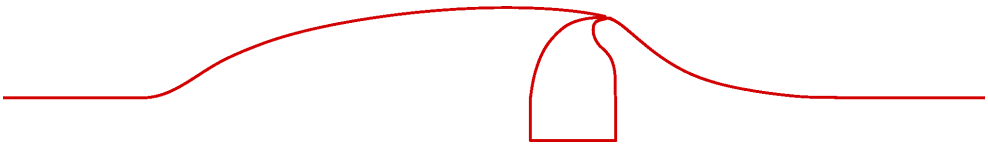
Οι εργασίες [172, 173, 174] χρησιμοποιούν σύνθετες δέσμες για τον έλεγχο της αεροελαστικής συμπεριφοράς αεροτομής δύο βαθμών ελευθερίας. Στην εργασία [175] χρησιμοποιούνται ενεργητικές τεχνικές ελέγχου της ροής μέσω δέσμης συνεχούς αναρρόφησης ή έγχυσης χρονικά μεταβαλλόμενης ή αμετάβλητης παροχής ρευστού ή σύνθετης δέσμης, με σκοπό τη μείωση των κραδασμών αεροτομής όταν εκείνη εκτίθεται σε δυναμικές φορτίσεις λόγω διαταραχών της ροής (Gust Load Alleviation).

Επιπλέον ενεργητικές τεχνικές είναι η χρήση συστήματος ελέγχου της κίνησης του ουραίου τμήματος αεροτομής ή επιφάνειας ελέγχου της πορείας του αεροσκάφους ή, ακόμα, του ύψους μίας σειράς γεννητριών δινών [150].

Όλες οι εφαρμογές που παρουσιάστηκαν προηγουμένως έγιναν σε CPUs. Στην παράγραφο που ακολουθεί παρουσιάζεται μία εφαρμογή ελέγχου της ζώνης αποκόλλησης κατάντι διαμόρφωσης στο εσωτερικό αγωγού με χρήση σύνθετης δέσμης. Οι υπολογισμοί έγιναν σε GPUs. Η βελτιστοποίηση, με το λογισμικό *EASY* (Evolutionary Argotithms SYstem), [176], των παραμέτρων δέσμης συνεχούς αναρρόφησης σε στόχο την ελαχιστοποίηση της περιοχής αποκόλλησης κατάντι αντίστοιχης διαμόρφωσης στο εσωτερικό αγωγού δείχνεται στη παράγραφο 7.3.

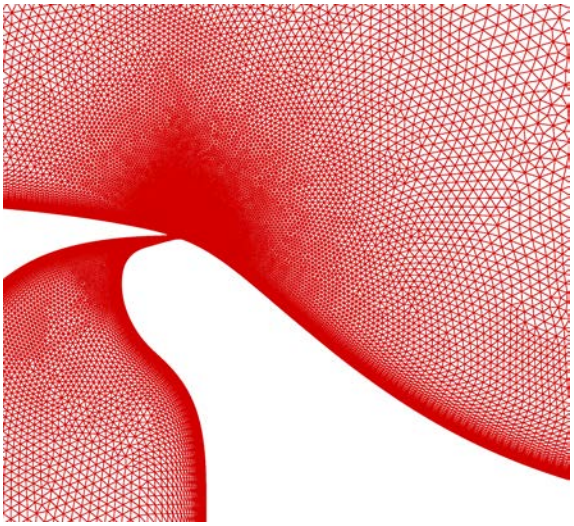
6.4.3 Έλεγχος της ροής πάνω από διαμόρφωση με χρήση σύνθετης δέσμης

Με στόχο να αναλυθούν οι διαδικασίες και οι υπολογιστικές μέθοδοι για την εφαρμογή σύνθετων δεσμών, το κέντρο ερευνών NASA LaRC διοργάνωσε το CFD Validation of Synthetic Jets and Turbulent Separation Control (CFDVal2004) Workshop, [177], στο οποίο, μεταξύ άλλων, μελετήθηκε η ροή πάνω από ένα σώμα τύπου Glauert–Goldschmied, [178] (σχήμα 6.9).



Σχήμα 6.9: Έλεγχος της ροής με σύνθετη δέσμη ρευστού: η γεωμετρία της εξεταζόμενης διαμόρφωσης, [178], στο CFD Validation of Synthetic Jets and Turbulent Separation Control (CFDVal2004) Workshop που διοργάνωσε το κέντρο ερευνών NASA LaRC.

Ο GPU-κώδικας της διατριβής χρησιμοποιήθηκε για την πρόλεξη της ελεγχόμενης μέσω σύνθετης δέσμης ροής πάνω από τη διαμόρφωση του σχήματος 6.9. Τα αριθμητικά αποτελέσματα συγκρίθηκαν με πειραματικά και υπολογιστικά (χρησιμοποιώντας CPUs) αποτελέσματα άλλων ερευνητικών ομάδων που συμμετείχαν στο Workshop. Το υπολογιστικό πλέγμα που χρησιμοποιήθηκε είναι κοινό ανάμεσα στις ερευνητικές ομάδες και λαμβάνεται από την ιστοσελίδα του Workshop, [177]. Αποτελείται από περίπου 116000 κόμβους και 230000 τριγωνικά στοιχεία, σχήμα 6.10.



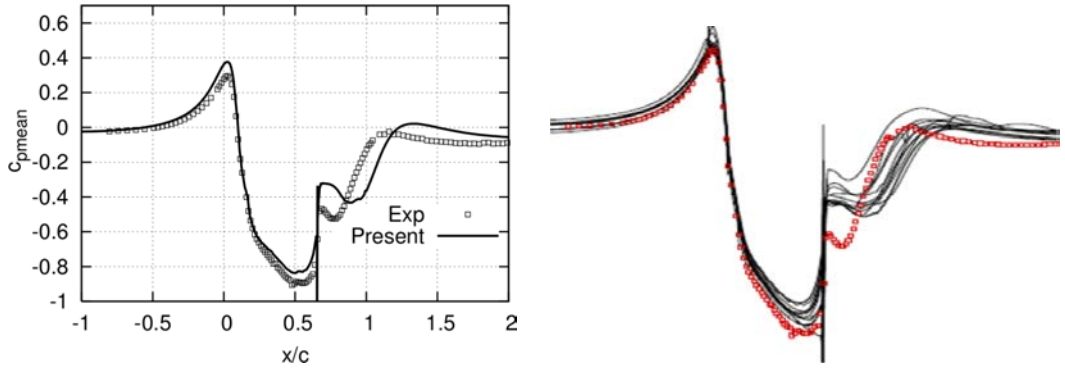
Σχήμα 6.10: Έλεγχος της ροής μέσω σύνθετης δέσμης ρευστού: λεπτομέρεια του μη-δομημένου πλέγματος που χρησιμοποιήθηκε στην περιοχή της οπής ελέγχου της ροής. Το ίδιο πλέγμα χρησιμοποιήθηκε από όλες τις ερευνητικές ομάδες που συμμετείχαν στο Workshop και λαμβάνεται από την ιστοσελίδα [177].

Η επιβολή των οριακών συνθηκών δεν γίνεται απευθείας στη θέση της οπής. Αντ΄ αυτού, για υπολογιστικούς λόγους, επιβάλλονται οριακές συνθήκες στο κάτω μέρος (βάση) της κοιλότητας που προσαρμόζεται στην οπή. Η εν λόγω κοιλότητα φαίνεται στα σχήματα 6.9 και 6.10.

Ο αριθμός Mach και ο αριθμός Reynolds ως προς τη χορδή της διαμόρφωσης της αδιατάρακτης ροής είναι $M_\infty = 0.1$, $Re_c = 9.36{\times}10^5$ αντίστοιχα. Οι συνθήκες της ροής είναι $p_\infty = 101.325 kg/ms^2$, $T_\infty = 298K$. Η συχνότητα της σύνθετης δέσμης είναι $f = 135.5Hz$ και ο συντελεστής παροχής μάζας (mass flux coefficient) $C_m \triangleq \frac{\rho h u^2_{jet,rms}}{\frac{1}{2} c \rho_\infty u^2_\infty} = 0.111\%$, όπου h το πλάτος της οπής της δέσμης.

Το σχήμα 6.11(α΄) δείχνει την κατανομή της χρονικά μέσης τιμής του συντελεστή πίεσης κατά μήκος της διαμόρφωσης, όπως υπολογίστηκε από τον GPU-επιλύτη και μετρήθηκε στο κέντρο ερευνών NASA LaRC. Το σχήμα 6.11(β΄) δείχνει την κατανομή της ίδιας ποσότητας όπως υπολογίστηκε από τις υπόλοιπες ερευνητικές ομάδες που συμμετείχαν στο Workshop σε σύγκριση με τις πειραματικές μετρήσεις. Σημειώνεται ότι κάθε ερευνητική ομάδα χρησιμοποιεί διαφορετικό μοντέλο τύρβης. Εδώ, ο GPU-επιλύτης της διατριβής χρησιμοποιεί το μοντέλο τύρβης των Spalart-Allmaras. Τα υπολογιστικά αποτελέσματα όλων των ερευνητικών ομάδων παρουσιάζουν μία μικρή απόκλιση σε σχέση με τα αντίστοιχα πειραματικά, μεταξύ τους όμως είναι όμοια. Στο σχήμα 6.12 φαίνεται το πεδίο του αριθμού Mach σε διαφορετικές χρονικές στιγμές.

Το ίδιο πρόβλημα επιλύθηκε χρησιμοποιώντας το CPU-κώδικα από την δόκτορα Β. Ασούτη στο πλαίσιο της διατριβής της, [129]. Οι αριθμητικές προλέξεις που παρουσιάζονται στη διατριβή αυτή ταυτίζονται με τα αριθμητικά αποτελέσματα που προέκυψαν από τη χρήση του GPU-κώδικα και παρουσιάζονται στο κεφάλαιο αυτό. Όμως, η χρήση μίας GTX 285 επιταχύνει την πρόλεξη των αριθμητικών αποτελεσμάτων 43 φο-



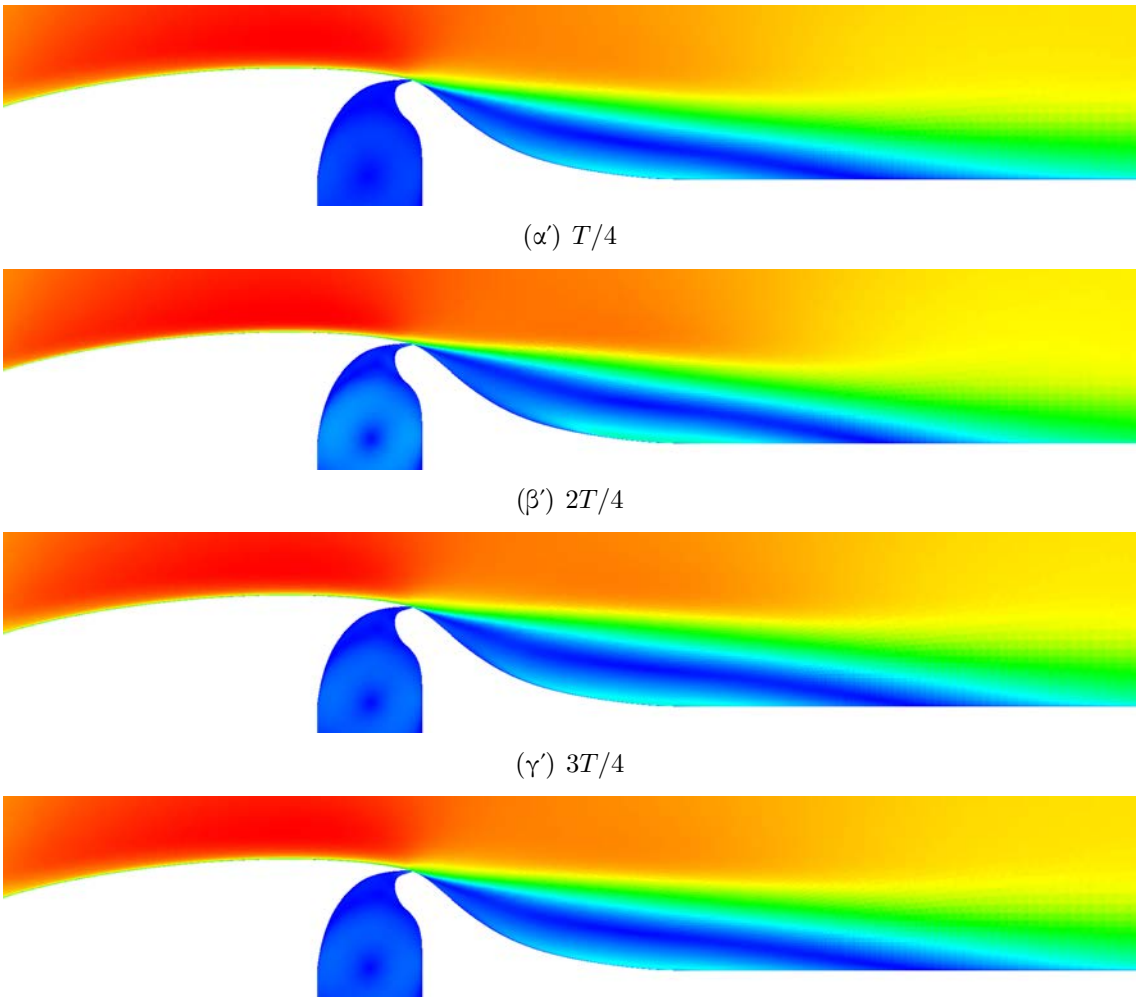
(α′) (β′)

Σχήμα 6.11: Έλεγχος της ροής μέσω σύνθετης δέσμης ρευστού: κατανομή της χρονικά μέσης τιμής του συντελεστή της πίεσης κατά μήκος της διαμόρφωσης του σχήματος 6.9. Σύγκριση των αριθμητικών αποτελεσμάτων του GPU-κώδικα της διατριβής (αριστερά) και των CPU-κωδίκων άλλων ερευνητικών ομάδων (δεξιά) με τις πειραματικές μετρήσεις (αριστερά).

ρές σε σχέση με έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη. Ο χρόνος εκτέλεσης του GPU-κώδικα, χρησιμοποιώντας το πλέγμα του σχήματος 6.10 με 116000 κόμβους, είναι περίπου 1.5 ώρα. Δηλαδή, η πρόλεξη της ίδιας ροής από τον αντίστοιχο CPU-κώδικα θα χρειαζόταν κάτι λιγότερο από 3 ημέρες σε μία σημερινή CPU.

6.5 Σύνοψη: Κέρδος από τη χρήση μίας GPU

Στο κεφάλαιο αυτό επιδείχθηκε η σημαντική μείωση του χρόνου επίλυσης αεροδυναμικών προβλημάτων που επιφέρει η χρήση μίας GPU. Έτσι προβλήματα, η επίλυση των οποίων απαιτεί ημέρες σε μία σημερινή CPU, επιλύονται σε μερικές ώρες χρησιμοποιώντας μία μόνο κάρτα γραφικών. Σημειώνεται ότι, στις εφαρμογές που παρουσιάστηκαν χρησιμοποιήθηκε η μικτής ακρίβειας εκδοχή του GPU-επιλύτη. Επιπλέον, χρησιμοποιήθηκαν GPUs τεχνολογίας του 2008 (GTX 285). Η χρήση GPUs τελευταίας τεχνολογίας θα αύξανε την επιτάχυνση στην πρόλεξη των αριθμητικών αποτελεσμάτων περίπου κατά 30%, μειώνοντας ακόμα περισσότερο το χρόνο επίλυσης. Αξιοσημείωτο, επίσης, είναι το γεγονός ότι οι GPUs που χρησιμοποιήθηκαν, αποκτήθηκαν στα τέλη του 2008 έναντι μικρού σχετικά κόστους (περίπου 500 ευρώ έκαστη) και τοποθετήθηκαν σε τότε τεχνολογικά ξεπερασμένους υπολογιστές (οι οποίοι μάλιστα ήταν ανενεργοί το διάστημα αυτό) μετατρέποντας τους με μικρό κόστος σε σύγχρονες μονάδες παράλληλης επεξεργασίας που, όπως φάνηκε, επιλύουν τη ροή γύρω από μοντέλο αεροσκάφος περίπου 37 φορές πιο γρήγορα σε σχέση με έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη.



(α′) $T/4$

(β′) $2T/4$

(γ′) $3T/4$

(δ′) $4T/4$

Σχήμα 6.12: Έλεγχος της ροής μέσω σύνθετης δέσμης ρευστού: υπολογισθέν πεδίο του αριθμού Mach σε διαφορετικές χρονικές στιγμές. T είναι η περίοδος της δέσμης και $M_\infty = 0.1$, $Re_c = 9.36 \times 10^5$, $f = 135.5Hz$, $C_m = 0.111\%$.

Κεφάλαιο 7

Χρήση των GPUs στο σχεδιασμό αεροδυναμικών μορφών μέσω εξελικτικών αλγορίθμων

Στο κεφάλαιο αυτό παρουσιάζεται μία ακόμη χρήση του κώδικα υπολογιστικής ρευστοδυναμικής που εκτελείται σε GPUs και τονίζεται το κέρδος που αυτή επιφέρει κατά το σχεδιασμό αεροδυναμικών μορφών μέσω εξελικτικών αλγορίθμων (EAs). Είναι γνωστό ότι βασικό μειονέκτημα των EAs είναι ο μεγάλος αριθμός αξιολογήσεων που απαιτούνται. Η 'έξυπνη' χρήση των GPUs, λ.χ. στη μορφή ενός διεπίπεδου-ιεραρχικού EA που χρησιμοποιεί την απλής (SPA) και τη μικτής (MPA) ακρίβειας εκδοχή του GPU-επιλύτη, μειώνει σημαντικά το χρόνο που απαιτείται για την ολοκλήρωση της βελτιστοποίησης. Από μία απλούστερη οπτική, θα αρκούσε η αντικατάσταση του CPU-επιλύτη με τον GPU-επιλύτη ώστε ο χρόνος αναμονής ενός προβλήματος βελτιστοποίησης-σχεδιασμού να μειωθεί σημαντικά.

7.1 Σχεδιασμός αεροδυναμικών μορφών – Γενικά

Οι μέθοδοι σχεδιασμού αεροδυναμικών μορφών διακρίνονται σε δύο μεγάλες κατηγορίες, ανάλογα με το αν χρησιμοποιούν ή όχι την παράγωγο της συνάρτησης κόστους ως προς τις μεταβλητές σχεδιασμού.

Στην αεροδυναμική, οι μέθοδοι που στηρίζονται στην παράγωγο της συνάρτησης κόστους (gradient–based methods) χρησιμοποιούν τη συζυγή τεχνική, [179, 180, 181, 182, 183, 184] για τον υπολογισμό των πρώτων (ή των δεύτερων [185, 186]) παραγώγων της συνάρτησης στόχου ως προς τις μεταβλητές σχεδιασμού. Βασικό πλεονέκτημα των μεθόδων αυτών, είναι ότι ο υπολογισμός των παραγώγων δεν εξαρτάται από το πλήθος των μεταβλητών σχεδιασμού. Ο χρόνος επίλυσης του συζυγούς προβλήματος, είναι ανάλογος εκείνου του ευθέος προβλήματος. Το μοναδικό μειονέκτημα των μεθόδων αυτών είναι ότι μπορεί να παγιδευτούν σε τοπικά ελάχιστα.

Οι EAs είναι μέθοδοι σχεδιασμού που δεν χρησιμοποιούν τις παραγώγους της συνάρτησης στόχου (gradient-free methods) και χειρίζονται οποιοδήποτε λογισμικό αξιο-

λόγησης, χωρίς να χρειάζεται να επέμβουν σε αυτό ή να έχουν πρόσβαση στον πηγαίο κώδικά του. Το βασικό πλεονέκτημα των EAs είναι ότι βρίσκουν πάντα το ολικό ελάχιστο και δεν παγιδεύονται σε τοπικά ελάχιστα. Ο χρόνος της βελτιστοποίησης εξαρτάται, όμως, σημαντικά από το πλήθος των μεταβλητών σχεδιασμού. Επιπλέον μειονέκτημα, αποτελεί ο μεγάλος αριθμός αξιολογήσεων που απαιτούνται. Πολλές εργασίες έχουν γίνει με σκοπό τη μείωση του αριθμού των αξιολογήσεων χρησιμοποιώντας μεταπρότυπα (surrogate evaluation models), [187, 188, 189, 190, 191, 192, 193, 194], ιεραρχικά ή πολυεπίπεδα σχήματα, [195, 196, 197, 198], και/ή υβριδισμό με μεθόδους σχεδιασμού που στηρίζονται στην παράγωγο της συνάρτησης κόστους, [196, 199, 200, 201].

Στην παρούσα διατριβή χρησιμοποιήθηκαν EAs βοηθούμενοι από μεταπρότυπα (Metamodel Assisted Evolutionary Algorithms, MAEAs) ενός ή δύο επιπέδων, για το σχεδιασμό αεροδυναμικών μορφών. Η ενότητα 7.2 περιγράφει το διεπίπεδο MAEA που χρησιμοποιήθηκε.

7.2 Διεπίπεδος εξελικτικός αλγόριθμος

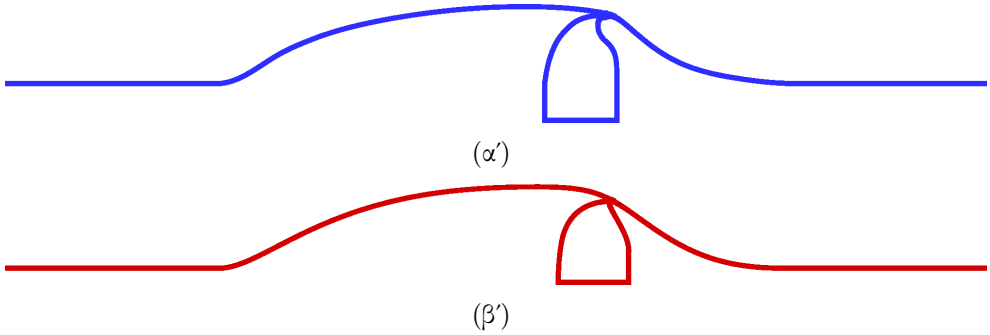
Ο διεπίπεδος ΕΑ, τουλάχιστον στην εκδοχή που ενδιαφέρει την παρούσα διατριβή, αποτελείται από δύο ημιαυτόνομους EAs που χρησιμοποιούν διαφορετικό λογισμικό αξιολόγησης. Στο πρώτο, ή χαμηλό, επίπεδο χρησιμοποιείται ένα πολύ γρήγορο αλλά λιγότερο ακριβές λογισμικό αξιολόγησης για τη γρήγορη σάρωση του χώρου των υποψήφιων λύσεων. Η βέλτιστη λύση (για μονοκριτηριακή βελτιστοποίηση) ή το μέτωπο των μη-κυριαρχούμενων λύσεων (μέτωπο Pareto, για πολυκριτηριακή βελτιστοποίηση) προκύπτει από το δεύτερο ή υψηλό επίπεδο, που χρησιμοποιεί ένα ακριβές αλλά κατά, ως εκ τούτου, πιο χρονοβόρο, λογισμικό αξιολόγησης. Τα δύο επίπεδα επικοινωνούν μεταξύ τους ανά συγκεκριμένο αριθμό γενεών και ανταλλάσσουν τις καλύτερες λύσεις που έχουν υπολογίσει μέχρι τότε, οι οποίες επαναξιολογούνται από το λογισμικό του επιπέδου άφιξης. Ο ρυθμός επικοινωνίας και το πλήθος των λύσεων που ανταλλάσσονται ορίζονται από το χρήστη. Όταν ένα επίπεδο 'φτάσει' πρώτο στη γενιά επικοινωνίας περιμένει το άλλο επίπεδο να 'φτάσει' και εκείνο στη δική του γενιά επικοινωνίας.

Οι EAs των δύο επιπέδων είναι δυνατό να επιταχύνονται με τη χρήση μεταπροτύπων (MAEAs). Ως μεταπρότυπα, στις εφαρμογές του κεφαλαίου χρησιμοποιούνται δίκτυα ακτινικών συναρτήσεων βάσης (Radial Basis Function networks ή RBF networks, [202]), τα οποία εκπαιδεύονται κατά τη διάρκεια της βελτιστοποίησης. Στην αρχή, ο MAEA λειτουργεί ως κλασικός ΕΑ. Οι πρώτες αξιολογήσεις χρησιμοποιούνται για την εκπαίδευση του μεταπροτύπου, με το πλήθος αυτών (ανά επίπεδο) να ορίζεται από το χρήστη. Με την ολοκλήρωση των πρώτων αξιολογήσεων ενεργοποιείται η τεχνική της προσεγγιστικής προ-αξιολόγησης (Inexact Pre-Evalution, IPE, [203]) και όλες οι αξιολογήσεις γίνονται από το μεταπρότυπο. Μόνο ορισμένες από τις καλύτερες λύσεις (το πλήθος αυτών ορίζεται από το χρήστη) επαναξιολογούνται από το λογισμικό αξιολόγησης του επιπέδου.

Για περισσότερες λεπτομέρειες, γενικά, σχετικά με τη λειτουργία ενός διεπίπεδου MAEA ο αναγνώστης παραπέμπεται στις εργασίες [196, 197] όπου περιγράφονται γενικότερα πολυεπίπεδα σχήματα βελτιστοποίησης με ΕΑ.

7.3 Βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης

Η ενότητα 6.4 αναφέρθηκε σε ενεργητικές και παθητικές τεχνικές ελέγχου της ροής, και παρουσιάστηκε η πρόλεξη ελεγχόμενης μέσω σύνθετης δέσμης, ροής ρευστού πάνω από διαμόρφωση, σχήμα 7.1(α΄). Στην ενότητα αυτή παρουσιάζεται η βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης ρευστού, με στόχο την ελαχιστοποίηση του μήκους της περιοχής αποκόλλησης κατάντι διαμόρφωσης, όμοιας με εκείνη του σχήματος 7.1(α΄), εσωτερικά αγωγού. Η διαμόρφωση που χρησιμοποιήθηκε για τη βελτιστοποίηση φαίνεται στο σχήμα 7.1(β΄).



Σχήμα 7.1: Γεωμετρία της διαμόρφωσης (α΄) που μελετήθηκε στο CFDVal2004 Workshop, [177], και (β΄) που χρησιμοποιήθηκε για τη βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης ρευστού με στόχο την ελαχιστοποίηση της ζώνης αποκόλλησης.

Στην αρχή της διατριβής, πριν την ενασχόληση με την πρόλεξη ροών σε GPUs, έγινε η παραμετρική διερεύνηση του παραπάνω προβλήματος χρησιμοποιώντας CPUs, [165]. Συγκεκριμένα, μελετήθηκε αρχικά η επίδραση της θέσης της οπής και της παροχής αναρρόφησης στη μείωση της ζώνης αποκόλλησης. Στη συνέχεια, για τις καλύτερες τιμές των παραπάνω παραμέτρων μελετήθηκε η επίδραση της κλίσης της δέσμης αναρρόφησης. Τελευταία μελετήθηκε η επίδραση του ανοίγματος της οπής. Η τοποθέτηση της δέσμης κοντά στο σημείο αποκόλλησης (θέση εκκίνησης της ζώνης αποκόλλησης) μείωσε περισσότερο το μήκος της ζώνης αποκόλλησης. Επιπλέον, όσο μεγαλύτερη είναι η παροχή αναρρόφησης ρευστού, τόσο περισσότερο μειώνεται η έκταση της ζώνης αποκόλλησης.

Η τότε χρήση των CPUs για την πρόλεξη της ροής πάνω από τη διαμόρφωση, με ή χωρίς έλεγχο, απέτρεψε τη βελτιστοποίηση των παραμέτρων της δέσμης μέσω EA, καθαρά για λόγους υψηλού υπολογιστικού κόστους. Η χρήση όμως των GPUs μειώνει σημαντικά τον χρόνο κάθε αξιολόγησης και καθιστά εφικτή μία τέτοια εφαρμογή.

Οι παράμετροι της δέσμης συνεχούς αναρρόφησης ρευστού, δηλαδή οι μεταβλητές σχεδιασμού είναι:

1. Η θέση της οπής πάνω στο τοίχωμα της διαμόρφωσης.
2. Η κλίση της δέσμης ως προς την κάθετη στο στερεό τοίχωμα, στη θέση της οπής.

3. Το άνοιγμα της οπής.

Ο ορισμός της κλίσης της σύνθετης δέσμης ως προς το τοπικό κάθετο στη διαμόρφωση φαίνεται στο σχήμα 7.3. Η παροχή αναρρόφησης δεν συμπεριλαμβάνεται στις μεταβλητές σχεδιασμού καθώς όσο υψηλότερη είναι η τιμή της παροχής αναρρόφησης, τόσο μεγαλύτερη είναι η ελάττωση της έκτασης της περιοχής αποκόλλησης, όπως συμπεραίνεται και από την παραμετρική μελέτη που προηγήθηκε.

Ο αριθμός Mach της ροής μακριά από τη διαμόρφωση είναι 0.3, ο αριθμός Reynolds, ως προς το μήκος της χορδής της διαμόρφωσης, είναι 6.35×10^6, και ο λόγος της παροχής αναρρόφησης προς την παροχή του αέρα εισόδου είναι 5.48×10^{-4}.

Για την πραγματοποίηση της βελτιστοποίησης χρησιμοποιήθηκε αυτοματοποιημένο λογισμικό προσαρμογής του πλέγματος στις εκάστοτε παραμέτρους της δέσμης. Έτσι αποφεύχθηκε η χρονοβόρα διαδικασία επανακατασκευής του πλέγματος. Το λογισμικό αυτό προγραμματίστηκε στο πλαίσιο της διπλωματικής εργασίας [204]. Ονομάζεται ως 'αρχικό' το υπολογιστικό πλέγμα εσωτερικά του αγωγού χωρίς έλεγχο. Η προσαρμογή του αρχικού πλέγματος ακολουθεί τα επόμενα βήματα:

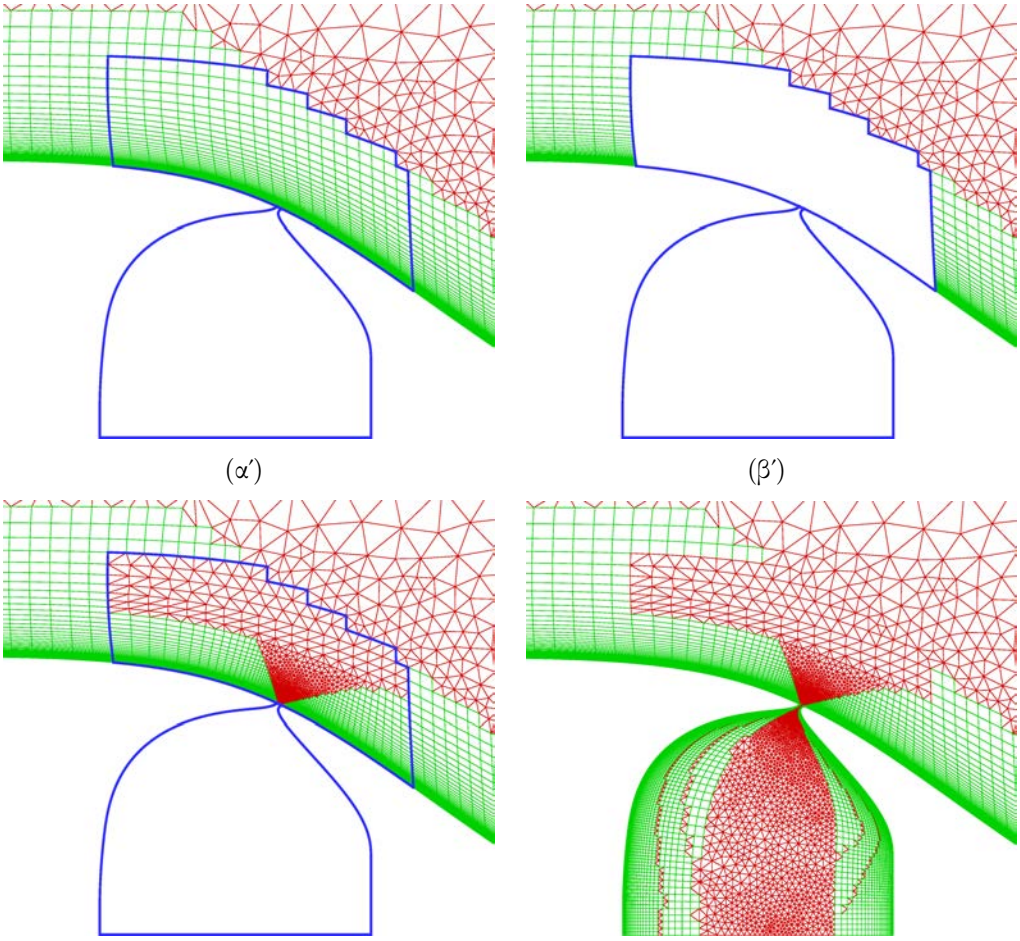
1. Ορισμός και απομόνωση της περιοχής του αρχικού πλέγματος στη 'γειτονιά' της οπής.
2. Απομάκρυνση του τμήματος του αρχικού πλέγματος που σημειώθηκε στο προηγούμενο βήμα.
3. Κατασκευή μη-δομημένου, υβριδικού, υπολογιστικού πλέγματος πάνω από την οπή.
4. Κατασκευή μη-δομημένου, υβριδικού, υπολογιστικού πλέγματος, εσωτερικά της κοιλότητας που προσαρμόστηκε στην οπή.

Τα παραπάνω βήματα φαίνονται και στα σχήματα 7.2(α′)-7.2(δ′). Η μεγέθυνση του προσαρμοσμένου (τελικού) πλέγματος στην περιοχή της οπής φαίνεται στο σχήμα 7.3. Κοντά στα τοιχώματα έχουν χρησιμοποιηθεί τετραπλευρικά πλεγματικά στοιχεία για την ακριβέστερη πρόλεξη του οριακού στρώματος. Αντίθετα, μακριά από τα τοιχώματα έχουν χρησιμοποιηθεί τριγωνικά στοιχεία. Το πλήθος των κόμβων του προσαρμοσμένου πλέγματος εξαρτάται από τις τιμές των παραμέτρων της δέσμης. Κατά μέσο όρο, τα υπολογιστικά πλέγματα που χρησιμοποιήθηκαν έχουν περίπου 75000 κόμβους, 34000 τριγωνικά και 57000 τετραπλευρικά στοιχεία. Η επιτάχυνση στην πρόλεξη της ροής από τη χρήση μίας GTX 285 και της MPA εκδοχής του GPU-επιλύτη, είναι 30x.

Για τη βελτιστοποίηση χρησιμοποιήθηκε ένας κλασικός (10,15) ΕΑ, δηλαδή ένας ΕΑ με $\mu = 10$ γονείς και $\lambda = 15$ απογόνους ανά γενιά (λογισμικό *EASY*, [176]). Ο *EASY* εκτελείται, ουσιαστικά με αμελητέο κόστος σε μία CPU ενώ οι αξιολογήσεις γίνονται στις διαθέσιμες GPUs.

Αναφέρθηκε ότι η χρήση μίας GPU επιταχύνει την αξιολόγηση μίας υποψήφιας λύσης περίπου 30 φορές. Για την επιπλέον μείωση του πραγματικού χρόνου της βελτιστοποίησης χρησιμοποιήθηκαν πολλές κάρτες GPUs, σχήμα 7.4. Κάθε στιγμή, μία GPU είναι υπεύθυνη για την αξιολόγηση μίας υποψήφιας λύσης.

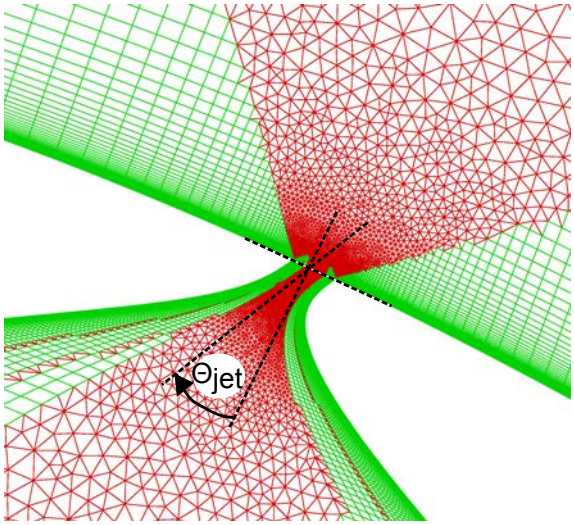
Κατά τη βελτιστοποίηση, η θέση της οπής μπορεί να μεταβάλλεται από το 60% έως το 70% της χορδής της διαμόρφωσης. Σημειώνεται ότι η θέση αποκόλλησης του οριακού στρώματος είναι περίπου στο 65% της χορδής, όπως προέκυψε από την ανάλυση



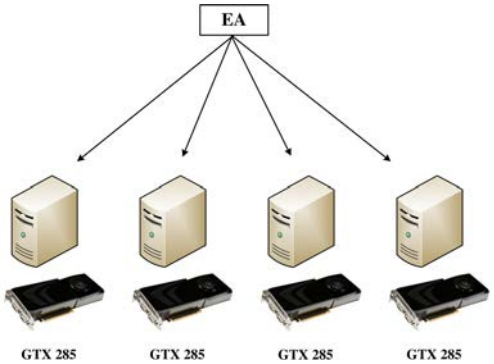
Σχήμα 7.2: Μονοκριτηριακή βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης για την ελαχιστοποίηση του μήκους της ζώνης αποκόλλησης: προσαρμογή του πλέγματος στην εκάστοτε θέση, κλίση της δέσμης αναρρόφησης ή/και άνοιγμα της οπής.

της ροής πάνω από την διαμόρφωση, χωρίς έλεγχο, χρησιμοποιώντας την MPA εκδοχή του GPU-επιλύτη και μία GTX 285. Η κλίση της δέσμης παίρνει τιμές από -70^o έως 70^o, ως προς την κάθετη στη διαμόρφωση στη θέση της οπής, και το άνοιγμα της οπής μεταβάλλεται μεταξύ 0.15% και 0.25% της χορδής της διαμόρφωσης.

Το σχήμα 7.5 παρουσιάζει τη σύγκλιση του EA. Ο οριζόντιος άξονας δείχνει το συνολικό αριθμό κλήσεων του GPU-επιλύτη και ο κατακόρυφος το μήκος της ζώνης αποκόλλησης ως προς το μήκος της χορδής της διαμόρφωσης. Φαίνεται ότι η βέλτιστη (πρακτικά, η καλύτερη λύση τη στιγμή που τερματίστηκε ο EA) επιτυγχάνει περίπου 34% μείωση του μήκους της ζώνης αποκόλλησης. Το πεδίο της ροής και η κατανομή του αριθμού Mach πάνω από τη διαμόρφωση φαίνονται στα σχήματα 7.6(α′), 7.6(β′) χωρίς έλεγχο και με έλεγχο σύμφωνα με τις βέλτιστες τιμές των παραμέτρων που υπολογίστηκαν από τον EA, αντίστοιχα. Η ελάττωση του μήκους της ζώνης αποκόλλησης είναι εμφανής. Στο σχήμα 7.7 φαίνεται η κατανομή του συντελεστή τριβής κατά μήκος

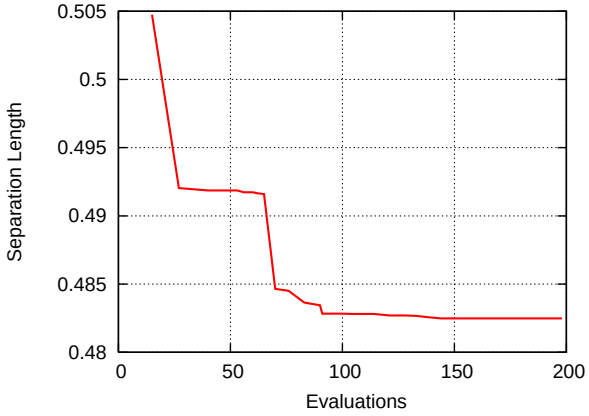


Σχήμα 7.3: Μονοκριτηριακή βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης για την ελαχιστοποίηση του μήκους της ζώνης αποκόλλησης: λεπτομέρεια του προσαρμοσμένου πλέγματος στην περιοχή της οπής και ορισμός της κατεύθυνσης (θ_{jet}) της σύνθετης δέσμης ως προς το τοπικό κάθετο στη διαμόρφωση.



Σχήμα 7.4: Οι αξιολογήσεις των υποψήφιων λύσεων γίνονται παράλληλα σε πολλές κάρτες γραφικών (μία αξιολόγηση ανά GPU).

της διαμόρφωσης με και χωρίς έλεγχο. Η μείωση της ζώνης αποκόλλησης (περιοχή όπου ο συντελεστής τριβής έχει αρνητικές τιμές) είναι εμφανής.

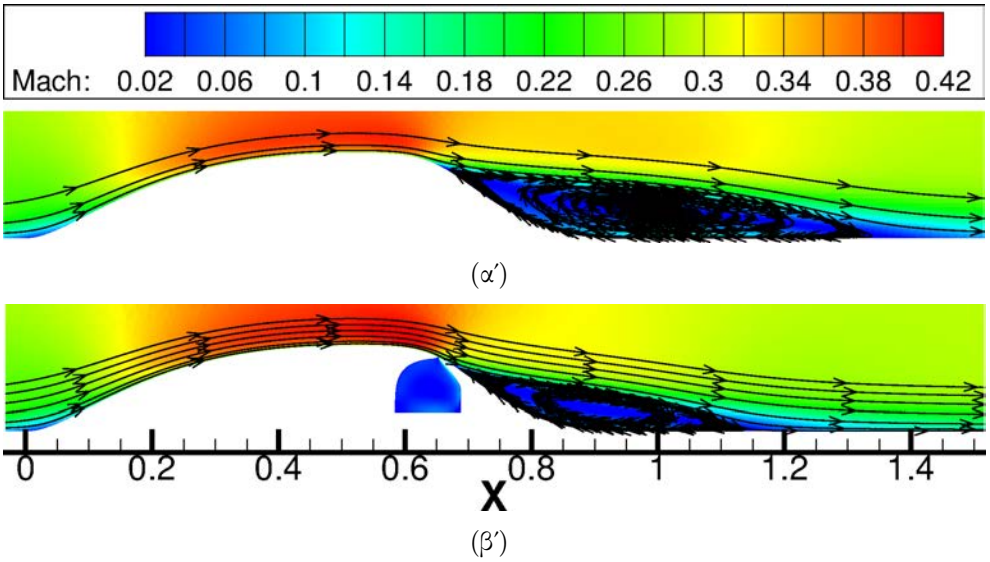


Σχήμα 7.5: Μονοκριτηριακή βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης για την ελαχιστοποίηση του μήκους της ζώνης αποκόλλησης: σύγκλιση του ΕΑ.

Οι βέλτιστες τιμές των μεταβλητών σχεδιασμού που υπολογίσθηκαν είναι:

- Θέση οπής ως προς το μήκος της χορδής της διαμόρφωσης = 0.6521
- Κλίση της δέσμης ως προς την κάθετη στη διαμόρφωση =-19.22^o
- Άνοιγμα οπής ως προς το μήκος της χορδής της διαμόρφωσης = 0.00234

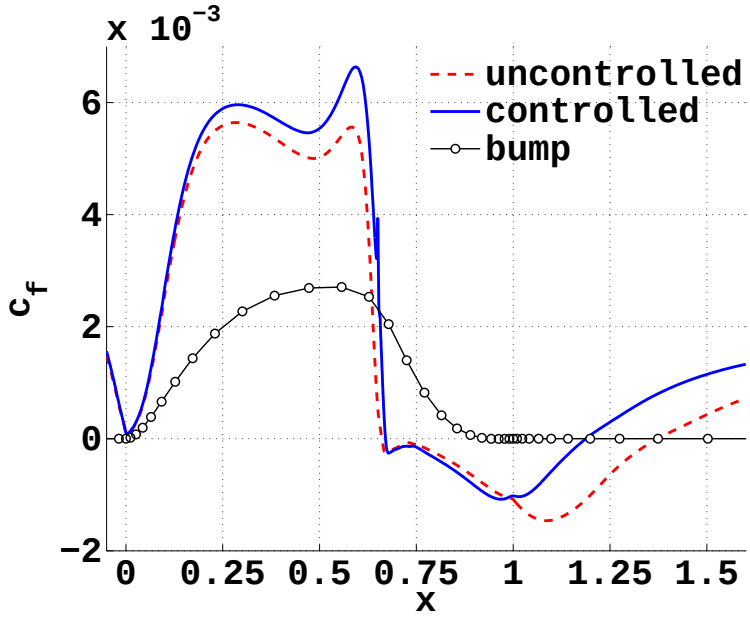
Επιβεβαιώνεται ότι η βέλτιστη θέση της δέσμης είναι πολύ κοντά στο σημείο αποκόλλησης του οριακού στρώματος.



(α′)

(β′)

Σχήμα 7.6: Μονοκριτηριακή βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης για την ελαχιστοποίηση του μήκους της ζώνης αποκόλλησης: γραμμές ροής και το πεδίο του αριθμού Mach πάνω από τη διαμόρφωση, (α′) χωρίς έλεγχο, και (β′) με έλεγχο σύμφωνα με τις βέλτιστες τιμές των παραμέτρων που υπολογίστηκαν από τον ΕΑ.



Σχήμα 7.7: Μονοκριτηριακή βελτιστοποίηση των παραμέτρων δέσμης συνεχούς αναρρόφησης για την ελαχιστοποίηση του μήκους της ζώνης αποκόλλησης: κατανομή του συντελεστή τριβής χωρίς (διακεκομμένη γραμμή) και με (συνεχόμενη γραμμή) έλεγχο, με βάση τις βέλτιστες τιμές των παραμέτρων σχεδιασμού, κατά μήκος της διαμόρφωσης. Φαίνεται, επίσης, η γεωμετρία της διαμόρφωσης (bump) χωρίς σχετική κλίμακα.

Η βελτιστοποίηση που παρουσιάστηκε, ολοκληρώθηκε σε περίπου τρεις ημέρες, χρησιμοποιώντας 4 GPUs αρχιτεκτονικής GT200 και την MPA εκδοχή του GPU-κώδικα. Η βελτιστοποίηση αυτή θα χρειαζόταν περίπου έξι ημέρες σε 64 CPU-πυρήνες της μονάδας παράλληλης επεξεργασίας της ΜΠΥΡ&Β/ΕΘΣ. Αυτός ήταν και ο λόγος που δεν πραγματοποιήθηκε η εν λόγω βελτιστοποίηση κατά την πρώτη περίοδο της διατριβής, δηλαδή πριν τον προγραμματισμό του GPU-επιλύτη. Το κέρδος είναι διπλό αν συνυπολογιστεί το κόστος κτήσης των τεσσάρων GPUs προηγούμενης αρχιτεκτονικής (περίπου 500 ευρώ η κάθε μία το 2008) σε σχέση με το κόστος κτήσης μίας μονάδας παράλληλης επεξεργασίας από 64 CPU-πυρήνες (ένας blade server της Dell με δύο τετραπύρηνες CPUs σήμερα κοστίζει περίπου 3000 ευρώ).

7.4 Σχεδιασμός μεμονωμένης αεροτομής

Η επόμενη εφαρμογή του GPU-επιλύτη αφορά το σχεδιασμό μεμονωμένης αεροτομής. Για το σκοπό αυτό χρησιμοποιήθηκε ο διεπίπεδος ΕΑ που περιγράφηκε στην παράγραφο 7.2 (λογισμικό *EASY*, [176]). Ως λογισμικό αξιολόγησης στο χαμηλό επίπεδο χρησιμοποιήθηκε η πολύ γρήγορη, αλλά λιγότερο ακριβής, SPA εκδοχή του GPU-επιλύτη. Αντίθετα, η ακριβής, αλλά σχετικά πιο χρονοβόρα, εκδοχή MPA του GPU-επιλύτη χρησιμοποιήθηκε στο υψηλό επίπεδο, από όπου προκύπτει η βέλτιστη λύση (στην περίπτωση μονοκριτηριακής βελτιστοποίησης) ή το μέτωπο των μη-κυριαρχούμενων λύσε-

ων (στην περίπτωση πολυκριτηριακής βελτιστοποίησης). Αυτή είναι μία ιδανική χρήση της απλής ακρίβειας εκδοχής του GPU-επιλύτη, που εξαιτίας της χρήσης αριθμητικής απλής ακρίβειας είναι μεν πολύ γρήγορη (επιταχύνει την πρόλεξη τυρβωδών ροών έως και 110 φορές όπως φαίνεται στην παράγραφο 4.5), αλλά δεν μπορεί να σταθεί από μόνη της ως λογισμικό επίλυσης. Επιπλέον, ένας διεπίπεδος ΕΑ συνήθως χρησιμοποιεί δύο διαφορετικά λογισμικά αξιολόγησης, για παράδειγμα έναν επιλύτη που χρησιμοποιεί μία πολύ γρήγορη αλλά λιγότερη ακριβή ολοκληρωματική μέθοδο στο χαμηλό επίπεδο και έναν επιλύτη των εξισώσεων Navier–Stokes στο υψηλό. Αντίθετα, στην περίπτωση της εφαρμογής της ενότητας αυτής, χρησιμοποιούνται δύο εκδοχές του ίδιου επιλύτη, που απλά χρησιμοποιούν αριθμητική διαφορετικής ακρίβειας. Δεν χρειάζεται δηλαδή ο χρήστης να διαθέτει δύο κώδικες.

Στο σχεδιασμό μεμονωμένης αεροτομής που πραγματοποιήθηκε, οι στόχοι της βελτιστοποίησης ήταν η μεγιστοποίηση του συντελεστή άνωσης και η ελαχιστοποίηση του συντελεστή οπισθέλκουσας της αεροτομής. Ο αριθμός Mach και η γωνία της επ΄ άπειρο ροής είναι αντίστοιχα $M_\infty = 0.3$, $\alpha_\infty = 4^o$ και ο αριθμός Reynolds, ως προς το μήκος της χορδής της αεροτομής, είναι $Re_c = 5{\times}10^6$.

Η γεωμετρία των δύο πλευρών της αεροτομής παραμετροποιήθηκε χρησιμοποιώντας δύο καμπύλες Bézier. Κάθε καμπύλη Bézier ελέγχεται από 9 σημεία ελέγχου (control points), τα οποία είναι ελεύθερα να κινηθούν κατά μήκος της χορδής της αεροτομής ή κάθετα σ΄ αυτή. Τα σημεία ελέγχου στις ακμές πρόσπτωσης και εκφυγής παραμένουν ακίνητα, ενώ τα αμέσως διπλανά τους σημεία ελέγχου είναι ελεύθερα να κινούνται με τον περιορισμό όμως να εξασφαλίζουν συνέχεια τύπου C^1 στις ακμές πρόσπτωσης και εκφυγής αντίστοιχα. Συνολικά, χρησιμοποιήθηκαν 24 μεταβλητές σχεδιασμού.

Περιορισμοί επιβλήθηκαν στο ελάχιστο πάχος (t) της αεροτομής ως εξής:

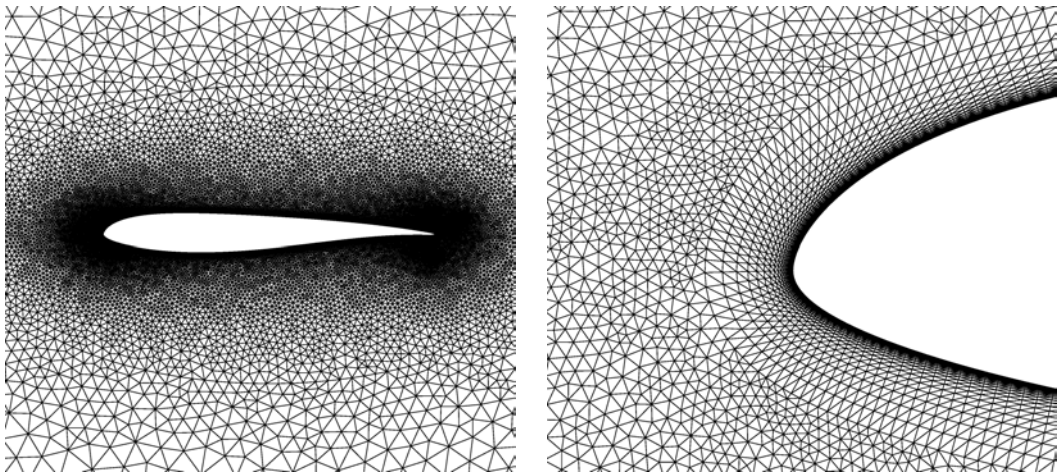
$$t(0.25c) \geqslant 0.07c,\ t(0.60c) \geqslant 0.05c,\ t(0.85c) \geqslant 0.02c$$

όπου c, είναι το μήκος της χορδής.

Κατά μέσο όρο, τα μη-δομημένα υπολογιστικά πλέγματα που χρησιμοποιήθηκαν αποτελούνταν από 35000 κόμβους. Το σχήμα 7.8(α΄) δείχνει ενδεικτικά ένα από τα υπολογιστικά πλέγματα που χρησιμοποιήθηκαν κατά τη βελτιστοποίηση. Λεπτομέρεια του ίδιου πλέγματος στην περιοχή της ακμής πρόσπτωσης της αεροτομής φαίνεται στο σχήμα 7.8(β΄). Γύρω από το τοίχωμα της αεροτομής χρησιμοποιήθηκαν τετραπλευρικά στοιχεία, τα οποία στη συνέχεια διασπάστηκαν σε τριγωνικά. Το υπόλοιπο χωρίο καλύφθηκε από τριγωνικά στοιχεία με τη μέθοδο του προελαύνοντος μετώπου.

Η επιτάχυνση στην πρόλεξη τυρβώδους ροής χρησιμοποιώντας την MPA εκδοχή του GPU-επιλύτη σε μία GTX 285 και τα υπολογιστικά πλέγματα της εφαρμογής αυτής, είναι 25x. Αντίθετα, η χρήση της εκδοχής απλής ακρίβειας επιταχύνει την πρόλεξη των αριθμητικών αποτελεσμάτων 40 φορές. Αυτή η σημαντική διαφορά στην επιτάχυνση που δίνει η χρήση των δύο εκδοχών αποτέλεσε το έναυσμα για τη χρήση του διεπίπεδου ΕΑ.

Στο χαμηλό επίπεδο του διεπίπεδου ΕΑ υλοποιείται ένας (20,50) ΕΑ, με $\mu = 20$ γονείς και $\lambda = 50$ απογόνους ανά γενιά. Στο υψηλό επίπεδο, εξελίσσεται ένας (10,20) ΕΑ. Οι ΕΑ και των δύο επιπέδων επιταχύνονται από τη χρήση RBF, δηλαδή μεταπροτύπων,



(α′) (β′)

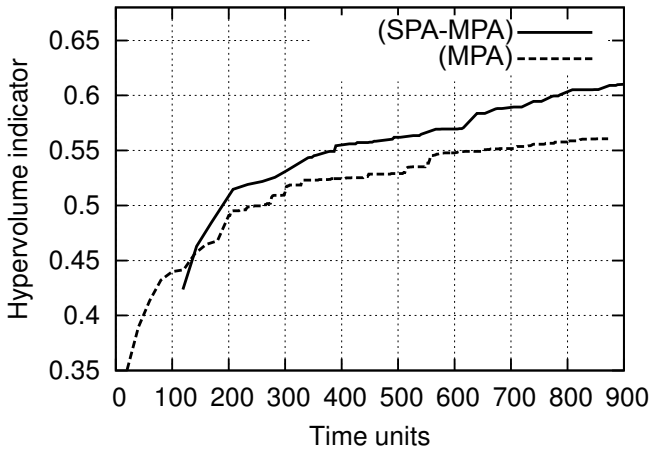
Σχήμα 7.8: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: (α′) Μη-δομημένο υπολογιστικό πλέγμα γύρω από αεροτομή που προέκυψε κατά το σχεδιασμό. (β′) Λεπτομέρεια του υπολογιστικού πλέγματος στην ακμή πρόσπτωσης της ίδιας αεροτομής.

για την προσεγγιστική προ-αξιολόγηση (IPE) των υποψήφιων λύσεων. Στην αρχή, οι MAEA των δύο επιπέδων δρουν ως κλασικοί ΕΑ μέχρι την ολοκλήρωση των 200 πρώτων αξιολογήσεων από το λογισμικό αξιολόγησης του επιπέδου, καθώς το πλήθος των 200 εγγραφών έχει οριστεί ως το ελάχιστο επιτρεπτόμενο πλήθος εγγραφών στη βάση δεδομένων, πριν αυτή χρησιμοποιηθεί από το μεταπρότυπο. Στη συνέχεια, μόνο 2 ή 5 υποψήφιες λύσεις επαναξιολογούνται από το λογισμικό αξιολόγησης στο υψηλό ή στο χαμηλό επίπεδο, αντίστοιχα. Ο ρυθμός επικοινωνίας των δύο επιπέδων επιλέχθηκε να είναι ανά 10 ή 5 γενιές του χαμηλού ή του υψηλού επιπέδου, αντίστοιχα. Οι 15 καλύτερες λύσεις στο χαμηλό επίπεδο μεταναστεύουν στο υψηλό, όπου και επαναξιολογούνται με βάση την MPA εκδοχή του GPU-επιλύτη. Αντίστοιχα, οι 6 καλύτερες λύσεις του υψηλού επιπέδου μεταναστεύουν στο χαμηλό.

Η πορεία του δείκτη υπερόγκου, [205], με την εξέλιξη του διεπίπεδου MAEA φαίνεται στο σχήμα 7.9 (συνεχόμενη γραμμή). Ο δείκτης υπερόγκου ποσοτικοποιεί την ποιότητα του μετώπου των μη-κυριαρχούμενων λύσεων, δηλαδή, όσο μεγαλύτερη τιμή έχει (δηλαδή πλησιέστερα στη μονάδα), τόσο 'καλύτερο' είναι το μέτωπο. Στον οριζόντιο άξονα του ίδιου σχήματος καταγράφεται η διάρκεια της βελτιστοποίησης σε μονάδες χρόνου. Ως μονάδα χρόνου θεωρείται ο χρόνος μίας αξιολόγησης με βάση το λογισμικό αξιολόγησης του υψηλού επιπέδου, δηλαδή την MPA εκδοχή του GPU-κώδικα. Επομένως, μία αξιολόγηση με την SPA εκδοχή θεωρείται ότι, κατά μέσο όρο, κοστίζει $\frac{25}{40} = 0.625$ μονάδες χρόνου.

Η ίδια εφαρμογή επαναλήφθηκε χρησιμοποιώντας ένα μονοεπίπεδο MAEA, με λογισμικό αξιολόγησης την MPA εκδοχή του GPU-επιλύτη. Η σύγκλιση του δείκτη υπερόγκου παρουσιάζεται με τη διακεκομμένη καμπύλη στο σχήμα 7.9. Ο διεπίπεδος MAEA παράγει ίδιας ποιότητας (ίδιας τιμής του δείκτη υπερόγκου) μέτωπο μη-κυριαρχούμενων λύσεων σε μικρότερο αριθμό αξιολογήσεων. Συγκεκριμένα ένα μέτω-

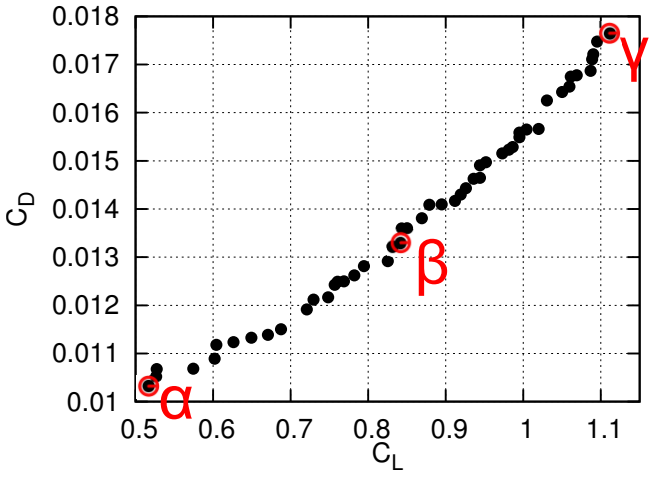
πο μη-κυριαρχούμενων λύσεων ίδιας ποιότητας με εκείνο που προκύπτει έπειτα από 900 χρονικές μονάδες (έπειτα δηλαδή από 900 κλήσεις του λογισμικού μικτής ακρίβειας) από τον μονοεπίπεδο MAEA, προκύπτει σε περίπου 500 χρονικές μονάδες από το διεπίπεδο αλγόριθμο. Η χρήση δηλαδή του διεπίπεδου, αντί του μονοεπίπεδου, MAEA μείωσε κατά περίπου 45% το χρόνο της βελτιστοποίησης. Το κέρδος αυτό υπερτίθεται σε εκείνο από τη χρήση GPUs. Επίσης, όπως και στην προηγούμενη εφαρμογή, χρησιμοποιήθηκαν όλες οι διαθέσιμες GTX 285 για την αξιολόγηση των υποψήφιων λύσεων (μία αξιολόγηση ανά GPU).



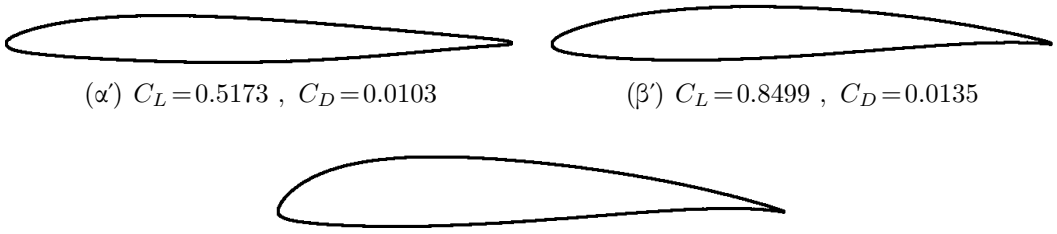
Σχήμα 7.9: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: σύγκλιση του δείκτη υπερόγκου για την περίπτωση ενός μονοεπίπεδου (διακεκομμένη καμπύλη) και του διεπίπεδου (συνεχής καμπύλη) MAEA.

Σημειώνεται ότι οι MAEA των δύο επιπέδων δεν ξεκινούν ταυτόχρονα με εκείνον του υψηλού επιπέδου να ξεκινά μετά την ολοκλήρωση των πρώτων 10 γενιών του χαμηλού επιπέδου. Για την αρχικοποίηση του πληθυσμού του υψηλού επιπέδου χρησιμοποιούνται τα μέλη του μετώπου των μη-κυριαρχούμενων λύσεων που έχει παραχθεί στο χαμηλό επίπεδο ως εκείνη τη στιγμή. Για το λόγο αυτό, στην περίπτωση του διεπίπεδου MAEA, η σύγκλιση του δείκτη υπερόγκου δεν ξεκινά από το μηδέν στο σχήμα 7.9. Υπενθυμίζεται ότι το μέτωπο των μη-κυριαρχούμενων λύσεων, το οποίο ποσοτικοποιεί ο δείκτης υπερόγκου στο σχήμα 7.9, προκύπτει από το υψηλό επίπεδο.

Το μέτωπο των μη-κυριαρχούμενων λύσεων που προέκυψε από το διεπίπεδο MAEA φαίνεται στο σχήμα 7.10. Τα σχήματα 7.11(α′)-7.11(γ′) δείχνουν τη γεωμετρία της αεροτομής και τις αντίστοιχες τιμές των συντελεστών άνωσης και οπισθέλκουσας τριών ενδεικτικών μελών του μετώπου των μη-κυριαρχούμενων λύσεων. Το πεδίο του αριθμού Mach γύρω από την αεροτομή και η κατανομή του συντελεστή πίεσης πάνω στην αεροτομή για τα ίδια μέλη του μετώπου φαίνονται στα σχήματα 7.12 και 7.13 αντίστοιχα.



Σχήμα 7.10: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: το μέτωπο των μη-κυριαρχούμενων λύσεων που προέκυψε από το διεπίπεδο MAEA.



(α′) $C_L = 0.5173$, $C_D = 0.0103$

(β′) $C_L = 0.8499$, $C_D = 0.0135$

(γ′) $C_L = 1.0770$, $C_D = 0.0172$

Σχήμα 7.11: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: γεωμετρία της αεροτομής και οι αντίστοιχες τιμές των συντελεστών άνωσης και οπισθέλκουσας τριών ενδεικτικών μελών του μετώπου των μη-κυριαρχούμενων λύσεων του σχήματος 7.10.

7.5 Σχεδιασμός αεροτομής πτερύγωσης συμπιεστή

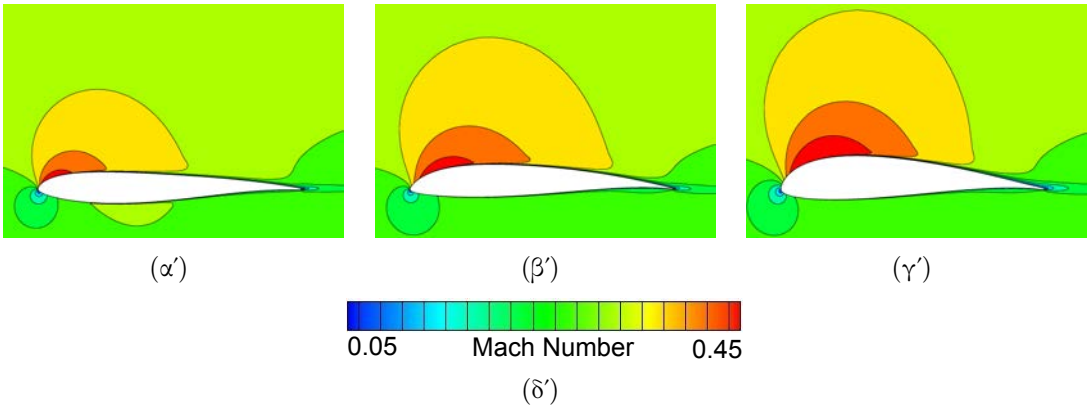
Ο ίδιος διεπίπεδος MAEA χρησιμοποιήθηκε και στο σχεδιασμό αεροτομής πτερύγωσης συμπιεστή με στόχο την ελαχιστοποίηση του συντελεστή απωλειών.

$$\omega = \frac{p_{t1} - p_{t2}}{p_{t1} - p_2} \tag{7.1}$$

όπου p_t, p η ολική και στατική πίεση αντίστοιχα. Οι δείκτες 1, 2 αναφέρονται στην είσοδο και την έξοδο του συμπιεστή.

Οι συνθήκες της ροής είναι: $M_{2,is} = 0.3$, $\alpha_1 = 44^o$ και $Re_c = 9 \times 10^5$ και η σταθερή γωνία κλίσης είναι 25^o.

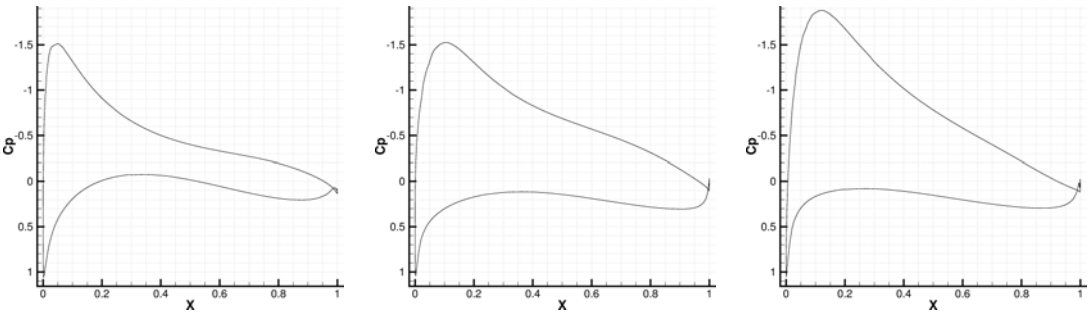
Οι δύο πλευρές της αεροτομής παραμετροποιήθηκαν με πολυώνυμα Bézier, τα σημεία ελέγχου των οποίων είναι ελεύθερα να κινηθούν προς την κατεύθυνση της χορδής της αεροτομής και κάθετα σ′ αυτήν. Τα σημεία ελέγχου στις ακμές πρόσπτωσης και



(α′) (β′) (γ′)

(δ′)

Σχήμα 7.12: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: πεδίο του αριθμού Mach γύρω από τις αεροτομές του σχήματος 7.11. Η χρωματική παλέτα (ε′) είναι κοινή για όλα τα πεδία.



(α′) (β′) (γ′)

Σχήμα 7.13: Δικριτηριακή βελτιστοποίηση της μορφής αεροτομής: κατανομή του συντελεστή πίεσης πάνω στις αεροτομές του σχήματος 7.11.

εκφυγής παραμένουν ακίνητα και τα αμέσως πλησιέστερα σε αυτά επιτρέπεται να κινούνται αρκεί να εξασφαλίζουν C^1 συνέχεια στις ακμές πρόσπτωσης και εκφυγής. Το σύνολο των μεταβλητών σχεδιασμού είναι 24.

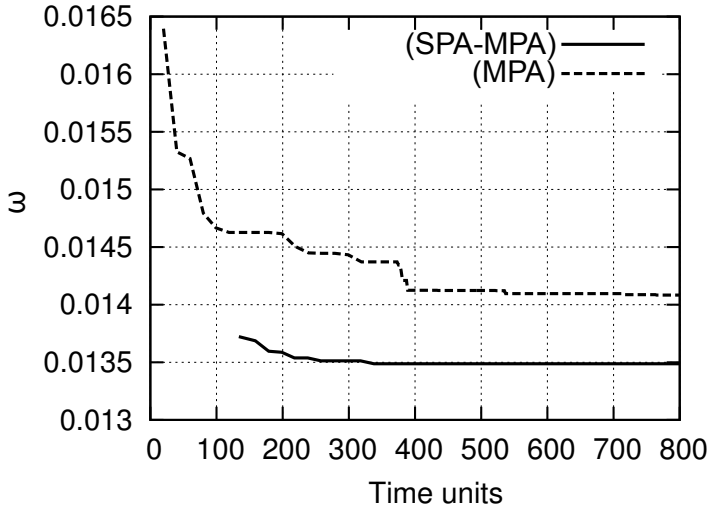
Για την αποφυγή σχηματισμού μη-αποδεκτών λεπτών αεροτομών, επιβλήθηκαν οι παρακάτω περιορισμοί πάχους:

$$t(0.30c) \geqslant 0.075c,\ t(0.60c) \geqslant 0.065c,\ t(0.90c) \geqslant 0.018c$$

όπου c, είναι το μήκος της χορδής. Επιβλήθηκε, επίσης, ελάχιστη τιμή στη στροφή της ροής ($\alpha_1 - \alpha_2 \geqslant 25^o$).

Στην εφαρμογή αυτή, στο υψηλό επίπεδο εξελίσσεται ένας (10,20) MAEA που χρησιμοποιεί την MPA εκδοχή του GPU-κώδικα ως λογισμικό αξιολόγησης. Αντίθετα, στο χαμηλό επίπεδο εξελίσσεται ένας (15,60) MAEA με την SPA εκδοχή του GPU-επιλύτη ως λογισμικό αξιολόγησης. Η τεχνική της προσεγγιστικής προ-αξιολόγησης (IPE) ενεργοποιείται έπειτα από 300 ή 200 κλήσεις του λογισμικού αξιολόγησης στο υψηλό και στο χαμηλό επίπεδο, αντίστοιχα. Μετά την ενεργοποίηση των μεταπρο-

τύπων, 2 και 5 από τα καλύτερα άτομα του πληθυσμού της γενιάς επαναξιολογούνται από το λογισμικό ΥΡΔ του κάθε επιπέδου. Όπως επιλέχθηκε και στην προηγούμενη εφαρμογή, ο MAEA στο υψηλό επίπεδο ξεκινά μετά την ολοκλήρωση των 10 πρώτων γενιών του MAEA στο χαμηλό επίπεδο. Οι επικρατέστερες λύσεις που έχουν προκύψει μέχρι εκείνη τη στιγμή χρησιμοποιούνται για την αρχικοποίηση του πληθυσμού στο υψηλό επίπεδο. Τα δύο επίπεδα ανταλλάσσουν τις καλύτερες λύσεις τους ανά 10 για το χαμηλό και 5 για το υψηλό επίπεδο γενιές.

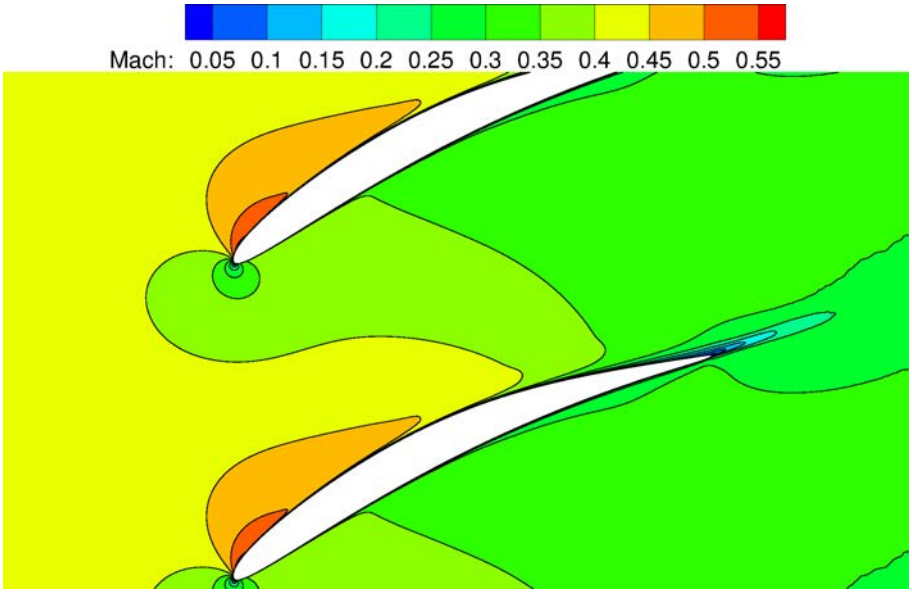


Σχήμα 7.14: Σχεδιασμός αεροτομής πτερύγωσης συμπιεστή με στόχο την ελαχιστοποίηση του συντελεστή απωλειών ολικής πίεσης: σύγκλιση του συντελεστή απωλειών με την εξέλιξη του διεπίπεδου (συνεχής γραμμή) ή ενός μονοεπίπεδου (διακεκομμένη γραμμή) MAEA.

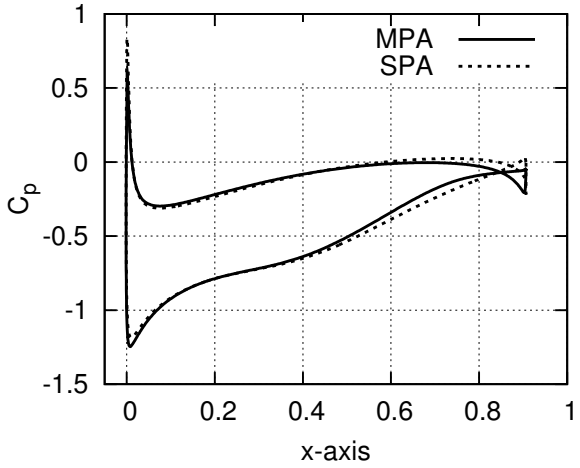
Στην περίπτωση του μονοεπίπεδου MAEA, επιλέχθηκε ο τερματισμός του σχεδιασμού να γίνει στις 800 χρονικές μονάδες ή ισοδύναμα μετά από 800 κλήσεις της MPA εκδοχής του GPU-κώδικα. Η σύγκλιση, δηλαδή η εξέλιξη του συντελεστή απωλειών με το χρόνο, του διεπίπεδου και ενός μονοεπίπεδου MAEA φαίνονται στο σχήμα 7.14. Η βέλτιστη λύση ($\omega = 0.0135$) που προκύπτει από το διεπίπεδο MAEA είναι καλύτερη σε σχέση με εκείνη ($\omega = 0.0141$) του μονοεπίπεδου στο ίδιο υπολογιστικό κόστος. Αξιοσημείωτο είναι ότι όλες οι λύσεις που δόθηκαν από το διεπίπεδο αλγόριθμο είναι καλύτερες από εκείνες του μονοεπίπεδου, για ίδιας διάρκειας βελτιστοποίηση. Στο διεπίπεδο αλγόριθμο, ο MAEA του υψηλού επιπέδου ξεκινά με την ολοκλήρωση των πρώτων 10 γενεών στο χαμηλό επίπεδο. Οπότε, αφού η σύγκλιση του διεπίπεδου αλγορίθμου είναι εκείνη του MAEA στο υψηλό επίπεδο, η σύγκλιση στο σχήμα 7.14 δεν ξεκινά από το μηδέν.

Το σχήμα 7.15 δείχνει το πεδίο του αριθμού Mach στην πτερύγωση με τη βέλτιστη αεροτομή που προέκυψε από το διεπίπεδο MAEA. Η κατανομή του συντελεστή πίεσης στη βέλτιστη αεροτομή όπως υπολογίζεται από τις απλής και μικτής ακρίβειας εκδοχές του GPU-επιλύτη φαίνεται στο σχήμα 7.16.

Κατά μέσο όρο, τα υπολογιστικά πλέγματα που χρησιμοποιήθηκαν στην εφαρμογή αποτελούνται από 61000 κόμβους. Για αυτό το μέγεθος υπολογιστικών πλεγμάτων, ο



Σχήμα 7.15: Σχεδιασμός αεροτομής πτερύγωσης συμπιεστή με στόχο την ελαχιστοποίηση του συντελεστή απωλειών ολικής πίεσης: πεδίο του αριθμού Mach στην πτερύγωση με τη βέλτιστη αεροτομή που υπολογίστηκε.



Σχήμα 7.16: Σχεδιασμός αεροτομής πτερύγωσης συμπιεστή με στόχο την ελαχιστοποίηση του συντελεστή απωλειών ολικής πίεσης: Κατανομή του συντελεστή πίεσης στη βέλτιστη αεροτομή όπως προκύπτει από τις εκδοχές μικτής (συνεχόμενη γραμμή) και απλής (διακεκομμένη γραμμή) ακρίβειας του GPU-επιλύτη.

χρόνος αξιολόγησης με βάση την SPA εκδοχή προς εκείνον με βάση την MPA εκδοχή είναι $\frac{30}{50}$. Για την κατασκευή των πλεγμάτων χρησιμοποιήθηκαν τετραπλευρικά στοιχεία κοντά στο τοίχωμα της αεροτομής και τριγωνικά μακριά. Η τιμή του y^+ στους πρώτους εσωτερικούς κόμβους κατά μήκος της αεροτομής είναι μικρότερη της μονάδας.

Όπως και στις προηγούμενες εφαρμογές του κεφαλαίου, χρησιμοποιήθηκαν όλες οι διαθέσιμες GTX 285 για τη μείωση του χρόνου αναμονής. Σε κάθε χρονική στιγμή,

μία GPU ήταν υπεύθυνη για μία αξιολόγηση.

7.6 Σύνοψη: Κέρδος από τη χρήση GPUs στο σχεδιασμό αεροδυναμικών μορφών

Στο κεφάλαιο αυτό χρησιμοποιήθηκε ο GPU-κώδικας ως λογισμικό αξιολόγησης των υποψήφιων λύσεων ενός ΕΑ. Από τη χρήση και μόνο GPUs, αντί CPUs, μειώνεται σημαντικά ο χρόνος της βελτιστοποίησης. Στις εφαρμογές που προηγήθηκαν, η επιτάχυνση στην πρόλεξη των αριθμητικών αποτελεσμάτων, χρησιμοποιώντας μία GTX 285 αντί ενός πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη, κυμαίνεται από 30x έως 45x. Η επιτάχυνση εξαρτάται από τη φυσική της ροής (ατριβής ή τυρβώδης) και το πλήθος των κόμβων του υπολογιστικού πλέγματος. Η χρήση τελευταίας τεχνολογίας GPUs θα μείωνε ακόμα περισσότερο το χρόνο της βελτιστοποίησης.

Στο κέρδος από τη χρήση GPUs υπερτίθεται το κέρδος από την παράλληλη αξιολόγηση των υποψήφιων λύσεων (μία αξιολόγηση ανά GPU) και το κέρδος από τη χρήση διεπίπεδου σχήματος βελτιστοποίησης, όπου ο ΕΑ στο χαμηλό επίπεδο χρησιμοποιεί την πολύ γρήγορη αλλά λιγότερο ακριβή SPA εκδοχή του GPU-κώδικα. Αντίθετα, η ακριβής MPA εκδοχή του GPU-κώδικα χρησιμοποιείται ως λογισμικό αξιολόγησης του υψηλού επιπέδου, από όπου προκύπτει(-ουν) η(οι) βέλτιστη(-ες) λύση(-εις). Ένας τέτοιος διεπίπεδος ΕΑ εκμεταλλεύεται την υψηλή ταχύτητα εκτέλεσης της απλής ακρίβειας εκδοχής του GPU-κώδικα και μπορεί να χρησιμοποιηθεί σε βιομηχανικές εφαρμογές.

Κεφάλαιο 8

Επίλυση αεροελαστικών προβλημάτων σε GPUs

Το κεφάλαιο αυτό παρουσιάζει την επέκταση και τη χρήση του GPU-κώδικα για την αεροελαστική ανάλυση αεροτομής που δύναται να μετακινείται, χωρίς να παραμορφώνεται, σύμφωνα με τις ασκούμενες σ΄ αυτήν αεροδυναμικές δυνάμεις και ροπές. Η κίνηση της αεροτομής περιορίζεται από γραμμικά και στρεπτικά ελατήρια. Επιτεύχθηκε η διεύρυνση της περιοχής ευσταθούς λειτουργίας της εξεταζόμενης αεροτομής χρησιμοποιώντας έναν απλό νόμο ελέγχου της εκτροπής της ακμής εκφυγής της αεροτομής, μέσω ανάδρασης των μεταβλητών κατάστασης της αεροτομής. Η εφαρμογή αυτή αποτελεί απλή προσομοίωση του αυτόματου πιλότου που ελέγχει την κίνηση των επιφανειών ελέγχου της πορείας ενός αεροσκάφους. Δοκιμάστηκαν διαφορετικά κέρδη για το βρόχο ανάδρασης, με στόχο τη μεγαλύτερη αύξηση της περιοχής ευσταθούς λειτουργίας της αεροτομής.

Γενικά, κατά την επίλυση αεροελαστικών προβλημάτων, η εξεταζόμενη αεροδυναμική κατασκευή παραμορφώνεται από τα ασκούμενα σ΄ αυτήν αεροδυναμικά φορτία. Για το λόγο αυτό, είναι απαραίτητη η προσαρμογή του υπολογιστικού πλέγματος στη χρονικά μεταβαλλόμενη γεωμετρία της βρεχόμενης επιφάνειας του εξεταζόμενου σώματος. Η περιγραφή της μεθόδου προσαρμογής του υπολογιστικού πλέγματος προηγείται της περιγραφής της σύζευξης του αεροδυναμικού GPU-επιλύτη με τον CPU-επιλύτη των ελαστικών εξισώσεων ισορροπίας της αεροτομής και της παρουσίασης των αεροελαστικών εφαρμογών.

8.1 Αεροελαστικότητα

Αεροελαστικότητα είναι ο κλάδος της επιστήμης που μελετά την αμοιβαία αλληλεπίδραση ελαστικών, αδρανειακών και αεροδυναμικών δυνάμεων που ασκούνται σε ένα σώμα που εκτίθεται σε ρεύμα ρευστού, [206, 207].

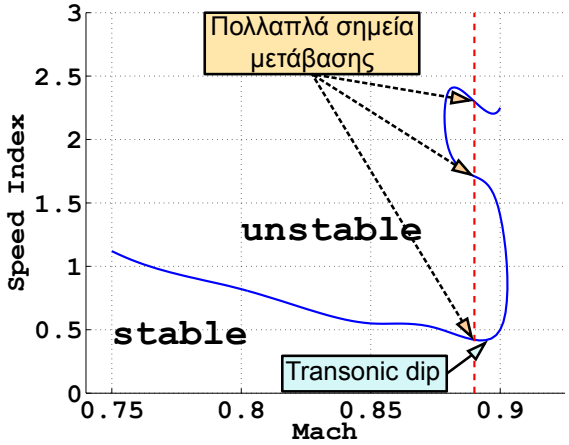
Η πρόλεξη αεροελαστικών φαινομένων αποτελεί ένα από τα αντικείμενα της διατριβής. Τα φαινόμενα αυτά κατηγοριοποιούνται σε ευσταθή, κρίσιμα ευσταθή ή ασταθή. Χαρακτηριστικό παράδειγμα ασταθούς αεροελαστικού φαινομένου σε πτέρυγες αεροσκαφών είναι ο πτερυγισμός (flutter), όπου η πτέρυγα ταλαντώνεται με συνεχώς αυξανόμενο πλάτος ταλάντωσης έως ότου καταρρεύσει. Γενικά, τα ασκούμενα αεροδυναμικά φορτία παραμορφώνουν την πτέρυγα. Εξαιτίας της μεταβολής της γεωμετρίας της πτέρυγας, τα χαρακτηριστικά της ροής γύρω από αυτή μεταβάλλονται και τα αεροδυναμικά φορτία αλλάζουν. Η μεταβολή των αεροδυναμικών φορτίων παραμορφώνει διαφορετικά την πτέρυγα κ.ο.κ. Το αν η πτέρυγα τελικά σταθεροποιηθεί σε μία μορφή (φθίνουσα ταλάντωση) ή συνεχίσει να ταλαντώνεται με σταθερό ή αυξανόμενο πλάτος εξαρτάται τόσο από τα χαρακτηριστικά της ροής όσο και από τα δομικά χαρακτηριστικά της κατασκευής.

Ένας τυπικός στόχος της αεροελαστικής ανάλυσης μίας πτέρυγας είναι ο υπολογισμός του ορίου της περιοχής ευσταθούς λειτουργίας αυτής. Το όριο αυτό αποτελεί την κρίσιμη καμπύλη της πτέρυγας και δίνεται σε διάγραμμα της ανηγμένης ταχύτητας της επ΄ άπειρο ροής ως συνάρτηση του αριθμού Mach της ροής. Το σχήμα 8.1 δείχνει μία τυπική κρίσιμη καμπύλη που πρέκυψε από την αεροελαστική ανάλυση πτέρυγας. Στον οριζόντιο άξονα μεταβάλλεται ο αριθμός Mach της ροής και στον κατακόρυφο ο δείκτης ταχύτητας (Speed Index) ή, αλλιώς, η ανηγμένη ταχύτητα της ροής που ορίζεται ως

$$U^* \triangleq \frac{U_\infty}{b\omega_\alpha\sqrt{\mu}} \tag{8.1}$$

όπου U_∞ η ταχύτητα της επ΄ άπειρο ροής, b ένα μήκος αναφοράς (λ.χ. το μισό του μήκους της μέσης χορδής της πτέρυγας), ω_α μία φυσική συχνότητα της εξεταζόμενης κατασκευής και μ ο λόγος των πυκνοτήτων του υλικού κατασκευής της πτέρυγας προς την πυκνότητα του ρευστού μακριά από την πτέρυγα (ϱ_∞). Παρατηρείται ότι η αναγωγή της ταχύτητας της ροής γίνεται με βάση δομικά χαρακτηριστικά της κατασκευής. Η κρίσιμη καμπύλη χωρίζει την περιοχή λειτουργίας της πτέρυγας σε ευσταθή και ασταθή περιοχή. Έτσι, η περιοχή που βρίσκεται κάτω από την κρίσιμη καμπύλη αποτελεί την περιοχή ευσταθούς λειτουργίας. Αντίθετα, πάνω από την κρίσιμη καμπύλη, εντοπίζεται η περιοχή ασταθούς λειτουργίας. Για ένα συνδυασμό ανηγμένης ταχύτητας ροής-αριθμού Mach επί της κρίσιμης καμπύλης, η πτέρυγα εκτελεί ταλάντωση σταθερού πλάτους (οριακά ευσταθής απόκριση). Η κρίσιμη ανηγμένη ταχύτητα παίρνει την ελάχιστη τιμή της στη διηχητική περιοχή (transonic dip). Συγκεκριμένα, έχει παρατηρηθεί ότι η κρίσιμη ανηγμένη ταχύτητα της επ΄ άπειρο ροής μειώνεται με την αύξηση του αριθμού Mach έως ότου φτάσει ένα κάτω όριο στη διηχητική περιοχή και, στη συνέχεια, αυξάνει καθώς η ροή γίνεται υπερηχητική.

Να σημειωθεί ότι, στη διηχητική περιοχή, μία πτέρυγα μπορεί να εμφανίζει πολλαπλά



Σχήμα 8.1: Τυπική κρίσιμη καμπύλη ελαστικής πτέρυγας.

σημεία μετάβασης από ευσταθή σε ασταθή κατάσταση λειτουργίας και αντίστροφα, [208], όπως φαίνεται και στο σχήμα 8.1. Αυτό εξαρτάται από τα δομικά χαρακτηριστικά της πτέρυγας και τις ασυνέχειες (κύματα κρούσης) του πεδίου της ροής.

Επιπλέον, στη διηχητική περιοχή, για ανηγμένες ταχύτητες ροής μεγαλύτερες της κρίσιμης, υπάρχει περίπτωση η ταλάντωση της κατασκευής να ενισχύεται στην αρχή και, στη συνέχεια, να σταθεροποιεί το πλάτος της, [209, 210]. Το φαινόμενο αυτό ονομάζεται Limit Cycle Oscillation (LCO). Το τελικό πλάτος της ταλάντωσης, όμως, είναι αρκετά μεγαλύτερο σε σχέση με εκείνο της οριακά ευσταθούς λειτουργίας. Επίσης, το χρονικό διάστημα που χρειάζεται ώστε να σταθεροποιηθεί το πλάτος της ταλάντωσης μπορεί να είναι πολύ μεγάλο σε σχέση με την περίοδο της ταλάντωσης. Πάντως, το γεγονός ότι το πλάτος της ταλάντωσης της κατασκευής μπορεί να έχει ένα άνω όριο δεν είναι κατ΄ ανάγκη θετικό καθώς η κατασκευή μπορεί να καταρρεύσει πριν αυτό σταθεροποιηθεί. Ακόμα και στην περίπτωση που η κατασκευή αντέξει τη δυναμική φόρτιση μέχρις ότου σταθεροποιηθεί το πλάτος, μία ταλάντωση τέτοιου πλάτους επιφέρει υψηλή καταπόνηση της κατασκευής και ελάττωση του χρόνου ζωής της. Ο λόγος που η κατασκευή μπορεί να οδηγηθεί σε LCO, αντί να αυξάνει συνεχώς το πλάτος της ταλάντωσης, είναι η ύπαρξη ασυνεχειών (κυμάτων κρούσης) στο πεδίο της ροής ή δομικών ασυνεχειών. Συνοψίζοντας, η αεροελαστική απόκριση μίας πτέρυγας (αλλά και γενικά μίας κατασκευής) μπορεί να είναι ευσταθής, οριακά ευσταθής, ασταθής ή να οδηγεί σε LCO.

Εξαιτίας του υψηλού υπολογιστικού κόστους της πρόλεξης της αεροελαστικής συμπεριφοράς μίας πτέρυγας, στη βιβλιογραφία χρησιμοποιούνται ευρέως απλά μοντέλα υπολογισμού των αεροδυναμικών φορτίων (π.χ. θεωρία λεπτών αεροτομών), [173, 174]. Τα μοντέλα αυτά, όμως, αδυνατούν να υπολογίσουν με ικανοποιητική ακρίβεια τα αεροδυναμικά φορτία στη διηχητική περιοχή, λόγω της ύπαρξης ασυνεχειών (κυμάτων κρούσης). Εξάλλου, η ελάχιστη τιμή της κρίσιμης ανηγμένης ταχύτητας εντοπίζεται στη διηχητική περιοχή όπου, επιπλέον, η απόκριση της κατασκευής μπορεί να οδηγήσει σε LCO. Απαιτείται, δηλαδή, η χρήση υψηλότερης ακρίβειας (άρα και υπολογιστικά πιο δαπανηρών) μοντέλων για την επίλυση της ροής γύρω από την κατασκευή. Η χρήση

GPUs μπορεί να μειώσει σημαντικά το υπολογιστικό κόστος μίας τέτοιας μελέτης.

Η επίλυση των εξισώσεων Euler ή Navier–Stokes για την χάραξη της κρίσιμης καμπύλης λειτουργίας μίας πτέρυγας και τον ακριβή υπολογισμό του κατώτερου ορίου της κρίσιμης ανηγμένης ταχύτητας της επ΄ άπειρο ροής δεν συναντάνται για πρώτη φορά στη βιβλιογραφία. Χαρακτηριστικά, στις δημοσιεύσεις [211, 212, 213] περιγράφεται η ταυτόχρονη ολοκλήρωση των εξισώσεων Euler με τις ελαστικές εξισώσεις μίας αεροτομής με το σχήμα Runge-Kutta. Στην εργασία [214] μελετάται η αεροελαστική συμπεριφορά αεροτομής για ατριβή ροή. Αντίθετα, στις εργασίες [215, 216] για τη μελέτη της αεροελαστικής συμπεριφοράς αεροτομής χρησιμοποιείται το μοντέλο τύρβης των Baldwin-Lomax, [217], ή εκείνο των Spalart-Allmaras, [53], αντίστοιχα. Οι εργασίες [218, 219] εξετάζουν την επίδραση της κίνησης της ακμής εκφυγής αεροτομής στην αεροελαστική συμπεριφορά αυτής. Στις εργασίες [172, 208, 220] μελετάται η διεύρυνση της περιοχής ευσταθούς λειτουργίας μίας αεροτομής χρησιμοποιώντας τεχνικές ενεργητικού ελέγχου. Συγκεκριμένα, στην εργασία [172], χρησιμοποιείται η τεχνική της παλλόμενης δέσμης ρευστού (synthetic jet). Αντίθετα, στις εργασίες [220, 208] ελέγχεται η κίνηση της ακμής εκφυγής της αεροτομής. Οι εργασίες [221, 220] μελετούν την επίδραση δομικών ασυνεχειών στην αεροελαστική συμπεριφορά μίας αεροτομής. Στην εργασία [222] κατασκευάζεται μοντέλο μειωμένης τάξης (Reduced Order Model, ROM) για την πρόλεξη της αεροελαστικής συμπεριφοράς αεροτομής και γίνεται η πιστοποίηση αυτού μέσω κώδικα επίλυσης των εξισώσεων Euler. Στις εργασίες [223, 224] γίνεται ο σχεδιασμός του συστήματος ελέγχου της κίνησης της ακμής εκφυγής αεροτομής με στόχο την αύξηση της περιοχής ευσταθούς λειτουργίας. Η αεροελαστική απόκριση της αεροτομής υπολογίζεται μέσω ενός ROM. Στην εργασία [225] περιγράφεται η σύζευξη κώδικα επίλυσης των εξισώσεων Navier–Stokes με κώδικα δομικής ανάλυσης για την πρόλεξη της αεροελαστικής συμπεριφοράς ανεμογεννήτριας.

Στο σύνολο των προηγούμενων εργασιών, η ολοκλήρωση των εξισώσεων της ροής γίνεται σε δομημένα υπολογιστικά πλέγματα. Αντίθετα, στην παρούσα διατριβή χρησιμοποιήθηκαν μη-δομημένα, δυναμικά προσαρμοζόμενα, υπολογιστικά πλέγματα. Η επιτάχυνση της πρόλεξης της αεροελαστικής συμπεριφοράς της εξεταζόμενης αεροτομής έγινε με τη χρήση GPUs τελευταίας αρχιτεκτονικής.

8.2 Προσαρμογή πλέγματος

Η μεταβολή της γεωμετρίας της επιφάνειας του σώματος κατά την επίλυση ενός αεροελαστικού προβλήματος δημιουργεί την ανάγκη προσαρμογής ή ανακατασκευής του υπολογιστικού πλέγματος. Στην παρούσα διδακτορική διατριβή, προτιμήθηκε η προσαρμογή του πλέγματος στις νέες οριακές συνθήκες (νέα γεωμετρία της βρεχόμενης επιφάνειας) καθώς η ανακατασκευή του θα αύξανε σημαντικά το υπολογιστικό κόστος της επίλυσης. Προτιμήθηκε να προηγηθεί η παρουσίαση του τρόπου προσαρμογής του πλέγματος, της παρουσίασης των ελαστικών εξισώσεων, της αριθμητικής ολοκλήρωσης αυτών και της παρουσίασης των αποτελεσμάτων διαφορετικών αεροελαστικών προβλημάτων που ακολουθούν στις επόμενες ενότητες. Στην ενότητα αυτή, αρχικά, παρουσιάζονται οι τρόποι προσαρμογής του πλέγματος που συναντώνται στη βιβλιογραφία και,

στη συνέχεια, περιγράφονται οι μέθοδοι που ακολουθήθηκαν στην παρούσα διατριβή.

Οι μέθοδοι παραμόρφωσης ενός υπολογιστικού πλέγματος διακρίνονται σε:

(α') Μεθόδους όπου ο όγκος που καταλαμβάνει το υπολογιστικό πλέγμα αντιμετωπίζεται ως ένα ελαστικό στερεό σώμα, [226, 227, 228]. Οι ελαστικές εξισώσεις εκφράζουν την ισορροπία των ασκούμενων δυνάμεων και ροπών σε κάθε στοιχειώδες τμήμα του όγκου αυτού. Οι ίδιες εξισώσεις προσδιορίζουν το σχήμα του φανταστικού παραμορφωμένου στερεού σώματος, δηλαδή το 'σχήμα' του προσαρμοζόμενου στην κίνηση/παραμόρφωση της βρεχόμενης επιφάνειας του σώματος, υπολογιστικού πλέγματος. Η επίλυση των ελαστικών εξισώσεων γίνεται αριθμητικά και χρησιμοποιείται το ίδιο πλέγμα υπολογισμού της ροής για τη διακριτοποίηση των ελαστικών εξισώσεων.

(β') Μεθόδους όπου το πλέγμα αντιμετωπίζεται ως ένα σύστημα ελατηρίων, [229, 230, 231]. Η παραμόρφωση των στοιχείων της φανταστικής αυτής κατασκευής (δηλαδή των ακμών του υπολογιστικού πλέγματος) περιορίζεται από γραμμικά ή στρεπτικά ελατήρια, τοποθετημένα επί των ακμών (σχήμα 8.2(α')) ή των κόμβων του υπολογιστικού πλέγματος (σχήμα 8.2(β')). Η μετατόπιση των οριακών κόμβων του πλέγματος επηρεάζει τη θέση των εσωτερικών. Η μετατόπιση των τελευταίων, δηλαδή η παραμόρφωση των στοιχείων της φανταστικής κατασκευής, υπολογίζεται επιλύοντας τις εξισώσεις ισορροπίας των δυνάμεων και των ροπών στους κόμβους του πλέγματος.

(γ') Μεθόδους όπου η μετατόπιση των κόμβων του πλέγματος υπολογίζεται με βάση μία απλή αλγεβρική συνάρτηση που συνδέει τη μετατόπιση των κόμβων του πλέγματος με τους αντίστοιχους πλησιέστερους οριακούς, [232]. Η μορφή μίας τέτοιας συνάρτησης είναι

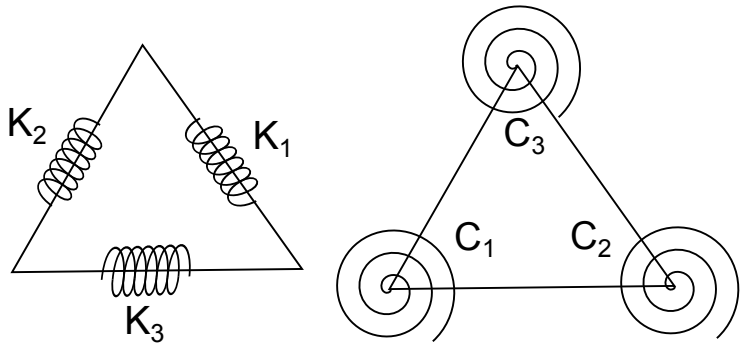
$$\overrightarrow{u}_i = f_i \overrightarrow{u}^i_{wall} \tag{8.2}$$

όπου $\overrightarrow{u}_i = [\delta x_i \; \delta y_i \; \delta z_i]^T$ είναι η μετατόπιση του κόμβου i του πλέγματος και υπολογίζεται ως το ποσοστό f_i της μετατόπισης $\overrightarrow{u}^i_{wall}$ του πλησιέστερου στον i κόμβου της στερεής επιφάνειας. Το ποσοστό f_i μεταβάλλεται από κόμβο σε κόμβο του πλέγματος, παίρνει τιμές στο διάστημα [0, 1] και εξαρτάται από την απόσταση του κόμβου από την κινούμενη επιφάνεια. Όσο πλησιέστερα στην κινούμενη επιφάνεια είναι ο κόμβος i, τόσο η τιμή του f_i πλησιάζει τη μονάδα.

(δ') Μεθόδους όπου η μετατόπιση των κόμβων της βρεχόμενης επιφάνειας προεκβάλλεται στους υπόλοιπους κόμβους του πλέγματος χρησιμοποιώντας συναρτήσεις ακτινικής βάσης (Radial Basis Function networks, RBF), [233, 234].

(ϵ') Μεθόδους στις οποίες ορίζεται ως συνάρτηση κόστους μία έκφραση που ποσοτικοποιεί την ποιότητα του υπολογιστικού πλέγματος και επιλύεται ένα πρόβλημα ελαχιστοποίησης. Το αποτέλεσμα της ελαχιστοποίησης είναι οι θέσεις των κόμβων του προσαρμοσμένου πλέγματος, [235, 236].

Οι μέθοδοι της τρίτης κατηγορίας εξασφαλίζουν τη γρηγορότερη, αλλά χαμηλότερης ποιότητας, προσαρμογή του υπολογιστικού πλέγματος σε σχέση με τις μεθόδους



Σχήμα 8.2: Παραμόρφωση του υπολογιστικού πλέγματος χρησιμοποιώντας (α′) γραμμικά ελατήρια προσδεδεμένα σε κάθε ακμή, ή (β′) στρεπτικά προσδεδεμένα στους κόμβους.

των υπόλοιπων κατηγοριών. Σε εφαρμογές με μικρές παραμορφώσεις ή μετακινήσεις του εξεταζόμενου σώματος, η χρήση μίας συνάρτησης της μορφής 8.2 εξασφαλίζει επαρκώς τη διατήρηση της ποιότητας του πλέγματος. Τέτοιες είναι οι περιπτώσεις των 2Δ αεροελαστικών εφαρμογών που παρουσιάζονται στο κεφάλαιο αυτό.

Σε εφαρμογές, όμως, με μεγάλες ή σύνθετες παραμορφώσεις/μετακινήσεις του εξεταζόμενου σώματος η χρήση μίας συνάρτησης της μορφής 8.2 δεν αρκεί και επιβάλλεται η χρήση μίας μεθόδου άλλης κατηγορίας. Στην παρούσα διατριβή, το χρησιμοποιούμενο υπολογιστικό πλέγμα θεωρήθηκε ως ένα σύστημα ελατηρίων. Η μέθοδος αυτή απαιτεί την επαναληπτική επίλυση των εξισώσεων ισορροπίας των δυνάμεων και ροπών στους κόμβους του πλέγματος για την εύρεση της τελικής θέσης των κόμβων. Για την αρχικοποίηση της επαναληπτικής επίλυσης χρησιμοποιήθηκε μία συνάρτηση της μορφής 8.2 (η ίδια με αυτή που χρησιμοποιήθηκε στην επίλυση των 2Δ αεροελαστικών εφαρμογών του κεφαλαίου). Τέτοιες είναι οι περιπτώσεις των 3Δ αεροδυναμικών εφαρμογών που παρουσιάζονται στο κεφάλαιο 9.

Η παράγραφος 8.2.1 παρουσιάζει τη συνάρτηση της μορφής 8.2 που χρησιμοποιήθηκε στις 2Δ αεροελαστικές εφαρμογές του κεφαλαίου αυτού. Στη συνέχεια, η παραγραφος 8.2.2 περιγράφει τον τρόπο προσαρμογής του πλέγματος στις 3Δ αεροδυναμικές εφαρμογές του κεφαλαίου 9.

8.2.1 Προσαρμογή πλέγματος με χρήση αλγεβρικής συνάρτησης

Στην επίλυση των 2Δ αεροελαστικών προβλημάτων που παρουσιάζονται στο κεφάλαιο αυτό ο υπολογισμός των μετακινήσεων των κόμβων του πλέγματος έγινε μέσω της σχέσης, [232],

$$\overrightarrow{u}_i = f\left(\hat{d}_i\right) \overrightarrow{u}^i_{wall} \tag{8.3}$$

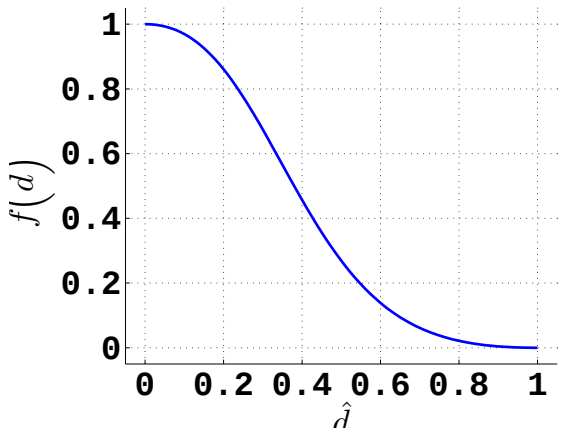
όπου η συνάρτηση $f\left(\hat{d}\right)$ δίνεται από την έκφραση

$$f\left(\hat{d}\right) = \frac{f_2^2}{f_1^2 + f_2^2} \tag{8.4}$$

όπου

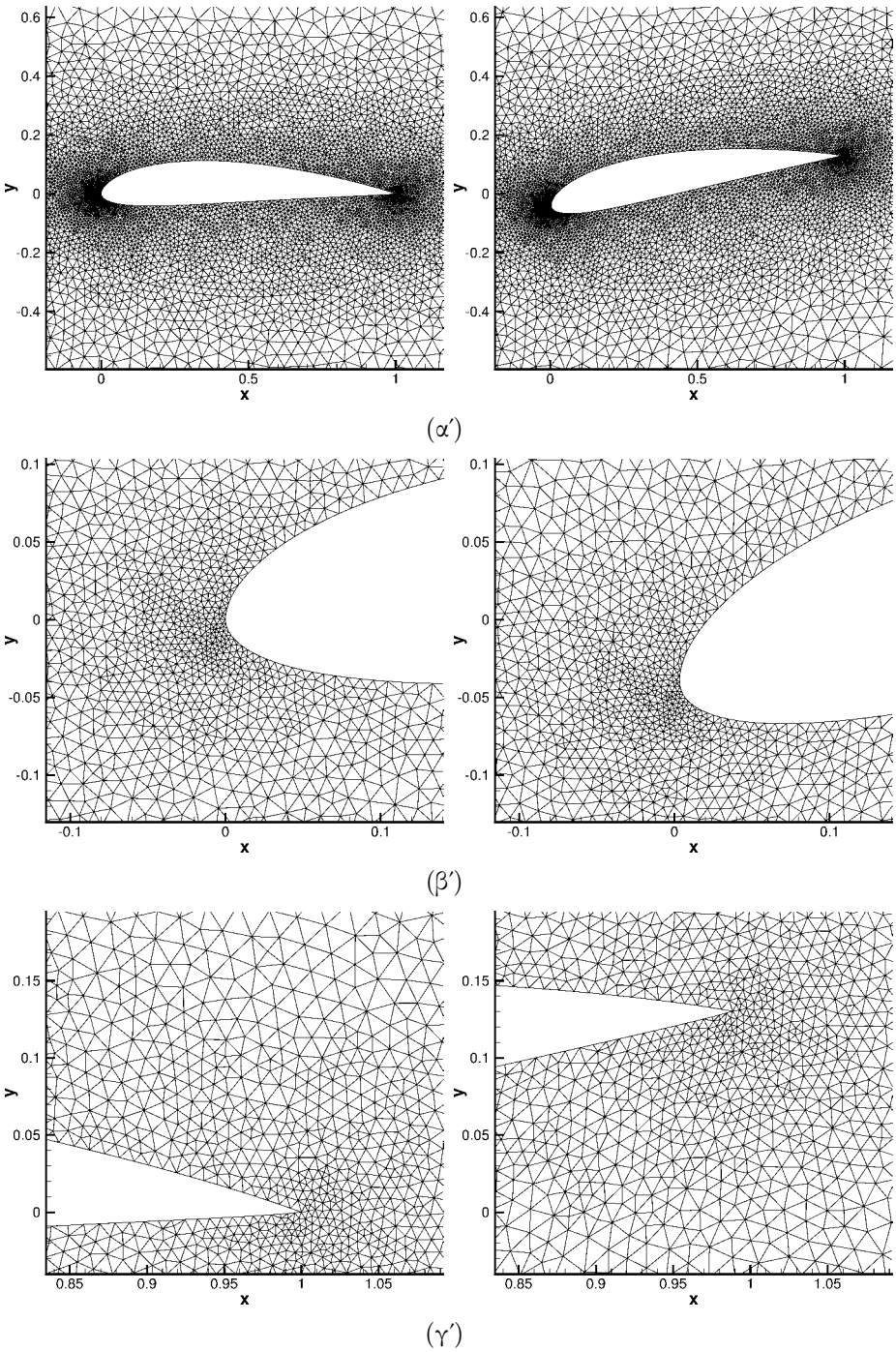
$$f_1\left(\hat{d}\right) = \frac{e\left(1 - e^{-\hat{d}}\right)}{-1 + e} \qquad f_2\left(\hat{d}\right) = \frac{\left(1 - e^{1-\hat{d}}\right)}{1 - e} \tag{8.5}$$

Στις παραπάνω εκφράσεις, e είναι ο νεπέριος πραγματικός αριθμός και $\hat{d}_i \doteq \frac{d_i}{d_{max}}$ είναι η απόσταση d_i του κόμβου i του πλέγματος από τον πλησιέστερο σε αυτόν κόμβο της επιφάνειας του σώματος, κανονικοποιημένη ως προς την απόσταση d_{max} που είναι η μέγιστη τιμή όλων των d_i. Η συνάρτηση 8.4 παίρνει τιμές στο διάστημα [0 1], με την απόσταση $\hat{d}$ να τείνει στο μηδέν η $f\left(\hat{d}\right)$ τείνει στη μονάδα ενώ, όταν η απόσταση $\hat{d}$ τείνει στη μονάδα, η $f\left(\hat{d}\right)$ τείνει στο μηδέν, όπως φαίνεται και στο σχήμα 8.3. Δηλαδή όσο πλησιέστεροι είναι οι κόμβοι του πλέγματος στην επιφάνεια του αεροδυναμικού σώματος, τόσο μεγαλύτερη είναι και η μετακίνησή τους. Για την παραγωγή προσαρμοσμένου πλέγματος καλύτερης ποιότητας, στην περίπτωση που κάποιος κόμβος βρίσκεται πλησιέστερα σε μέσο οριακής ακμής (ή σε βαρύκεντρο οριακής πλευράς σε 3Δ πλέγμα), η μετατόπισή του σχετίζεται με το ημιάθροισμα των μετατοπίσεων των κόμβων της πλησιέστερης οριακής ακμής (ή της μετατόπισης του βαρύκεντρου της οριακής πλευράς).



Σχήμα 8.3: Γραφική παράσταση της $f\left(\hat{d}\right)$ με την έκφραση 8.4.

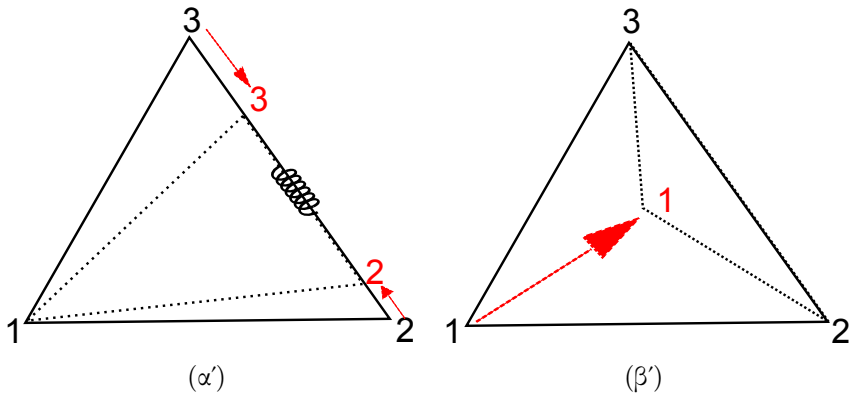
Τα σχήματα 8.4 δείχνουν ένα μη-δομημένο υπολογιστικό πλέγμα γύρω από μία αεροτομή πριν και μετά την προσαρμογή αυτού σε περιστροφή της αεροτομής κατά 10^o γύρω από το $1/4$ της χορδής της. Η ποιότητα του προσαρμοσμένου πλέγματος, όπως φαίνεται και στα σχήματα 8.4(β′), 8.4(γ′) είναι αντίστοιχη του πλέγματος 'βάσης' (αριστερή στήλη εικόνων 8.4).



Σχήμα 8.4: Περιστροφή αεροτομής γύρω από το $1/4$ της χορδής της κατά 10^o. Η αριστερή στήλη δείχνει το πλέγμα πριν την περιστροφή της αεροτομής ενώ η δεξιά μετά την προσαρμογή του πλέγματος στη νέα θέση αυτής μέσω της σχέσης 8.3. Δείχνεται η περιοχή γύρω (α′) από την αεροτομή, (β′) την ακμή πρόσπτωσης και (γ′) την ακμή εκφυγής. Όπως φαίνεται, η ποιότητα του πλέγματος δεν αλλοιώνεται μετά την προσαρμογή.

8.2.2 Θεώρηση του υπολογιστικού πλέγματος ως συστήματος ελατηρίων

Στις 3Δ αεροδυναμικές εφαρμογές του κεφαλαίου 9 θεωρήθηκε το χρησιμοποιούμενο υπολογιστικό πλέγμα ως ένα σύστημα ελατηρίων. Η χρήση γραμμικών ελατηρίων (linear spring analogy method), σχήμα 8.2(α΄), αποτρέπει να συμπέσουν δύο κόμβοι, αφού όσο πλησιάζει ένας κόμβος κάποιο γειτονικό του, τόσο αυξάνει η δύναμη που τους απωθεί αμοιβαία (σχήμα 8.5(α΄)). Στην περίπτωση όμως του σχήματος 8.5(β΄), όπου ο κόμβος 1 πλησιάζει την ακμή 23, η μέθοδος των γραμμικών ελατηρίων δεν αποτρέπει τον εκφυλισμό του τριγώνου 123. Το τελευταίο αποτρέπεται με τη χρήση στρεπτικών ελατηρίων (torsional spring analogy method, σχήμα 8.2(β΄)). Η αύξηση της γωνίας του τριγώνου στην κορυφή 1 (θ_1), συνεπάγεται αύξηση της ασκούμενης ροπής στους κόμβους 2, 3 που τείνει να επαναφέρει τη θ_1 στην προηγούμενη τιμή της. Στη μέθοδο των στρεπτικών ελατηρίων, η μέτρηση των γωνιών των τριγώνων γίνεται σύμφωνα με τον προσανατολισμό των κόμβων πριν την παραμόρφωση του πλέγματος. Συνεπώς, στην περίπτωση που ο κόμβος 1 διαπεράσει την ακμή 23 η τιμή της της θ_1 θα είναι μεγαλύτερη των 180^o. Έτσι, η μέθοδος εγγυάται το σωστό προσανατολισμό των κόμβων του προσαρμοσμένου πλέγματος. Για τον παραπάνω λόγο επιλέχθηκε η χρήση μόνο στρεπτικών αντί γραμμικών ελατηρίων.



Σχήμα 8.5: Αποφυγή (α΄) ταύτισης δύο κόμβων, ή (β΄) εκφυλισμού τριγώνου κατά την προσαρμογή του υπολογιστικού πλέγματος με τη μέθοδο (α΄) γραμμικών ή (β΄) στρεπτικών ελατηρίων.

Παρόλο που η μέθοδος αυτή προσαρμογής του πλέγματος, χρησιμοποιήθηκε μόνο σε 3Δ εφαρμογές, είναι υποχρεωτική, αρχικά, η παρουσίαση της 2Δ εκδοχής της. Στη συνέχεια περιγράφεται ο υπολογισμός του μητρώου δυσκαμψίας, αρχικά σε 2Δ μη-δομημένα πλέγματα αποτελούμενα από τριγωνικά στοιχεία και, στη συνέχεια, σε 3Δ μη-δομημένα πλέγματα που αποτελούνται από τετραεδρικά στοιχεία.

Εφαρμογή της μεθόδου σε 2Δ μη-δομημένα πλέγματα: Μητρώο δυσκαμψίας τριγωνικού στοιχείου

Για τον υπολογισμό των σταθερών δυσκαμψίας των στρεπτικών ελατηρίων, χρησιμοποιήθηκαν οι εκφράσεις της εργασίας [231], οι οποίες για το τρίγωνο 123 του σχήματος 8.5(β′) διατυπώνονται ως

$$C_1 = \frac{l_{12}^2 l_{13}^2}{4A^2}, \qquad C_2 = \frac{l_{21}^2 l_{23}^2}{4A^2}, \qquad C_3 = \frac{l_{31}^2 l_{32}^2}{4A^2} \tag{8.6}$$

όπου l_{12}, l_{13}, l_{23} είναι τα μήκη των πλευρών του τριγώνου σύμφωνα με τους δείκτες-αριθμούς των αντίστοιχων κορυφών και A είναι το εμβαδόν αυτού. Το εμβαδόν του τριγώνου εμφανίζεται στον παρονομαστή των παραπάνω εκφράσεων. Συνεπώς, η μείωση του εμβαδού ενός τριγωνικού στοιχείου του πλέγματος αυξάνει τις σταθερές δυσκαμψίας των στρεπτικών ελατηρίων στους κόμβους του τριγώνου άρα και την τιμή των ροπών που τείνουν να αυξήσουν την έκταση του τριγωνικού στοιχείου. Έτσι εξασφαλίζεται ότι τα τριγωνικά στοιχεία του πλέγματος δεν θα εκφυλιστούν κατά την προσαρμογή του.

Οι ισοδύναμες δυνάμεις που ασκούνται στους κόμβους του τριγώνου λόγω των αντίστοιχων μετατοπίσεων υπολογίζονται ως

$$\underline{f} = \underline{K}\ \underline{u} \tag{8.7}$$

όπου $\underline{u}$ είναι το διάνυσμα των καρτεσιανών μετατοπίσεων των κόμβων του τριγώνου και $\underline{f}$ είναι το διάνυσμα των ισοδύναμων ασκούμενων δυνάμεων. Δηλαδή

$$\underline{f} = \begin{bmatrix} \overrightarrow{f}_1 \\ \overrightarrow{f}_2 \\ \overrightarrow{f}_3 \end{bmatrix}, \qquad \underline{u} = \begin{bmatrix} \overrightarrow{u}_1 \\ \overrightarrow{u}_2 \\ \overrightarrow{u}_3 \end{bmatrix} = \begin{bmatrix} [\delta x_1\ \delta y_1]^T \\ [\delta x_2\ \delta y_2]^T \\ [\delta x_3\ \delta y_3]^T \end{bmatrix} \tag{8.8}$$

και $\underline{K}$ είναι το (τοπικό) μητρώο δυσκαμψίας του τριγωνικού στοιχείου. Ο υπολογισμός του συμμετρικού αυτού μητρώου γίνεται από την έκφραση

$$\underline{K} = \underline{R}^T\ \underline{C}\ \underline{R} \tag{8.9}$$

όπου

$$\underline{C} = \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & C_3 \end{bmatrix} \tag{8.10}$$

και

$$\underline{R} = \begin{bmatrix} \hat{y}_{13} - \hat{y}_{12} & -\hat{x}_{13} + \hat{x}_{12} & \hat{y}_{12} & -\hat{x}_{12} & -\hat{y}_{13} & \hat{x}_{13} \\ -\hat{y}_{21} & \hat{x}_{21} & \hat{y}_{21} - \hat{y}_{23} & -\hat{x}_{21} + \hat{x}_{23} & \hat{y}_{23} & -\hat{x}_{23} \\ \hat{y}_{31} & -\hat{x}_{31} & -\hat{y}_{32} & \hat{x}_{32} & \hat{y}_{32} - \hat{y}_{31} & -\hat{x}_{32} + \hat{x}_{31} \end{bmatrix} \tag{8.11}$$

με

$$\hat{x}_{ij} = \frac{x_j - x_i}{l_{ij}^2}, \qquad \hat{y}_{ij} = \frac{y_j - y_i}{l_{ij}^2} \tag{8.12}$$

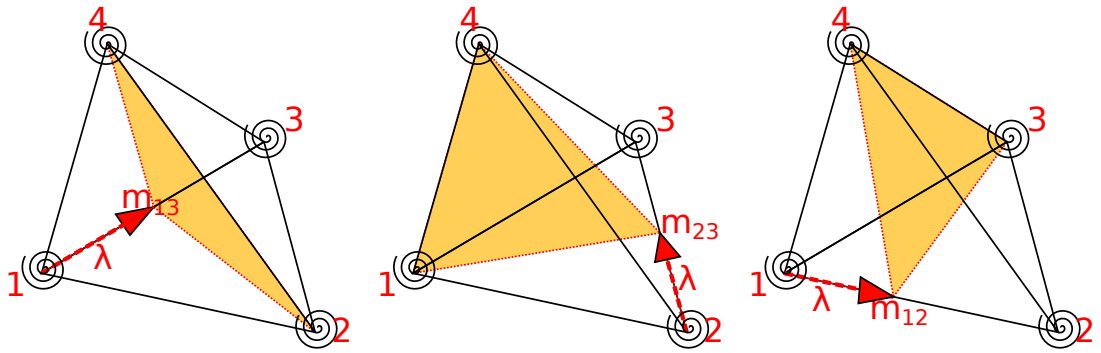
Τελικά, το μητρώο δυσκαμψίας ενός τριγωνικού στοιχείου είναι ένα συμμετρικό μητρώο 6×6 που μπορεί να γραφεί στη μορφή

$$\underline{K} = \begin{bmatrix} K^{11} & K^{12} & K^{13} \\ K^{21} & K^{22} & K^{23} \\ K^{31} & K^{32} & K^{33} \end{bmatrix} \tag{8.13}$$

όπου τα μητρώα K^{ij} έχουν διάσταση 2×2 και ισχύει $[K^{ij}] = [K^{ji}]^T$.

Εφαρμογή της μεθόδου σε 3Δ μη-δομημένα πλέγματα: Μητρώο δυσκαμψίας τετραεδρικού στοιχείου

Για τον υπολογισμό του μητρώου δυσκαμψίας ενός τετραέδρου, σχηματίζονται 12 τριγωνικά στοιχεία εσωτερικά του τετραέδρου (3 ανά κορυφή), όπως περιγράφεται στην εργασία [231]. Η σχέση 8.9 εφαρμόζεται στο τοπικό σύστημα αξόνων του κάθε τριγωνικού στοιχείου για τον υπολογισμό του αντίστοιχου μητρώου δυσκαμψίας $\underline{K}^{tri}$. Από τη σύνθεση των 12 $\underline{K}^{tri}$ προκύπτει το μητρώο δυσκαμψίας του τετραέδρου $\underline{K}^{tetra}$. Ακολουθεί η αναλυτικότερη περιγραφή του υπολογισμού του $\underline{K}^{tetra}$.



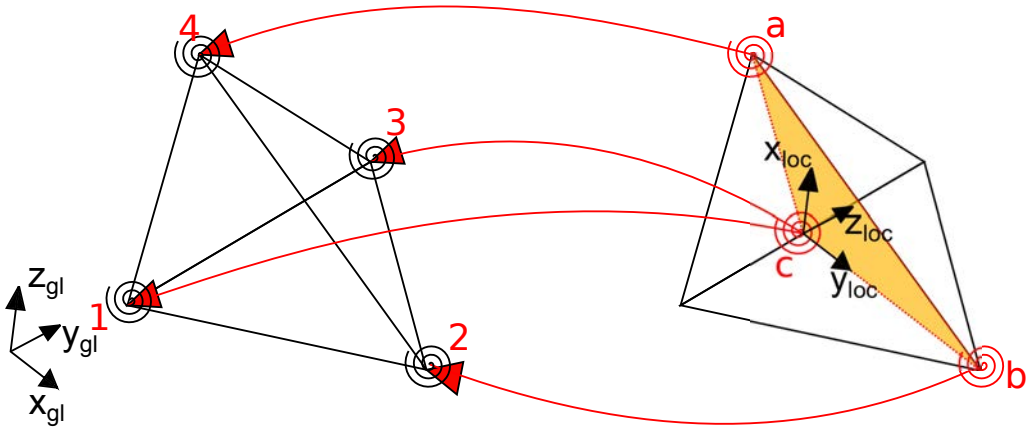
Σχήμα 8.6: Σχηματισμός τριών τριγώνων εσωτερικά του τετραέδρου 1234 που ανήκουν στην κορυφή 4. Το επίπεδο που ορίζει κάθε ένα από αυτά τα τριγωνικά στοιχεία είναι κάθετο στο επίπεδο που ορίζουν οι κορυφές του τριγώνου 123 που είναι η πλευρά του τετραέδρου 'απέναντι' στην κορυφή 4. Ανάλογα ορίζονται $3\times3=9$ τριγωνικά στοιχεία που ανήκουν στις υπόλοιπες κορυφές του τετραέδρου. Συνολικά, σχηματίζονται 12 τριγωνικά στοιχεία για τις 4 κορυφές του κάθε τετραέδρου.

Το σχήμα 8.6 παρουσιάζει το σχηματισμό των τριών τριγωνικών στοιχείων για την τέταρτη κορυφή ενός τετραέδρου. Όμοια ορίζονται τα τρία τριγωνικά στοιχεία για τις υπόλοιπες κορυφές του τετραέδρου. Τα τριγωνικά στοιχεία ανά κορυφή του τετραέδρου ορίζονται από την εξεταζόμενη κορυφή (στο σχήμα 8.6 την κορυφή 4), μία άλλη

κορυφή του τετραέδρου και ένα σημείο επί της ακμής που ορίζουν οι δύο υπόλοιπες κορυφές του τετραέδρου (σημεία m_{13}, m_{23}, m_{12} στα σχήματα 8.6(α′), 8.6(β′), 8.6(γ′) αντίστοιχα), τέτοιο ώστε το επίπεδο που ορίζει το τριγωνικό στοιχείο να είναι κάθετο στο επίπεδο της πλευράς του τετραέδρου που είναι απέναντι από τον εξεταζόμενη κορυφή. Η τρίτη κορυφή του τριγώνου (σημεία m_{13}, m_{23}, m_{12}) επιβάλλεται να βρίσκεται εντός ακμής του τετραέδρου και, γενικά, δεν είναι το μέσο της. Στην περίπτωση που δεν υπάρχει σημείο επί ακμής τέτοιο ώστε το επίπεδο που ορίζει το τριγωνικό στοιχείο να είναι κάθετο στη βάση του τετραέδρου, χρησιμοποιείται μία τρίτη κορυφή του τετραέδρου. Στο παράδειγμα του σχήματος 8.6, που αναφέρεται στην κορυφή 4, η έδρα 123 αντιμετωπίζεται ως βάση του τετραέδρου. Προφανώς, η βάση του τετραέδρου αλλάζει με την εξεταζόμενη κορυφή.

Στο σχήμα 8.7 ορίζονται το τοπικό σύστημα αξόνων $(x_{loc}, y_{loc}, z_{loc})$. Ο άξονας z_{loc} είναι κάθετος στο επίπεδο του τριγωνικού στοιχείου του σχήματος 8.6(α′). Για την αποφυγή σύγχυσης όλες οι μεταβλητές που χρησιμοποιούνται στη συνέχεια συνοδεύονται από τον εκθέτη tri ή $tetra$ αν αναφέρονται σε τριγωνικό ή τετραεδρικό στοιχείο, αντίστοιχα. Επιπλέον, οι δείκτες gl, loc υποδηλώνουν αν το αντίστοιχο διάνυσμα ή μητρώο αναφέρεται στο ολικό (global) ή στο τοπικό (local) ανά τριγωνικό στοιχείο καρτεσιανό σύστημα αξόνων.

Στη συνέχεια, παρουσιάζεται ο υπολογισμός του μητρώου δυσκαμψίας του τριγώνου $42m_{13}$ του σχήματος 8.6(α′). Όμοια προκύπτουν τα μητρώα δυσκαμψίας των υπόλοιπων 11 τριγωνικών στοιχείων. Για ευκολία οι κορυφές του τριγώνου $42m_{13}$ στο εξής θα αναφέρονται ως a, b και c, όπως φαίνεται στο σχήμα 8.7.



Σχήμα 8.7: Συνεισφορά του τριγώνου abc στο μητρώο δυσκαμψίας του τετραέδρου.

Αφού κάθε τριγωνικό στοιχείο μπορεί να δέχεται μετατοπίσεις μόνο στο επίπεδο που αυτό ορίζει, η έκφραση 8.13 γράφεται

$$\underline{K}_{loc}^{tri} = \begin{bmatrix} K^{aa} & K^{ab} & K^{ac} \\ K^{ba} & K^{bb} & K^{bc} \\ K^{ca} & K^{cb} & K^{cc} \end{bmatrix} \tag{8.14}$$

με K^{ij} μητρώα διάστασης 3×3 με μηδενικά τα στοιχεία της τρίτης γραμμής και στήλης, αντίστοιχα. Δηλαδή

$$K^{ij} = \begin{bmatrix} K_{11}^{ij} & K_{12}^{ij} & 0 \\ K_{21}^{ij} & K_{22}^{ij} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{8.15}$$

Συνεπώς, το μητρώο δυσκαμψίας του κάθε τριγωνικού στοιχείου $\underline{K}^{tri}$ υπολογίζεται από τη σχέση 8.9 με την προσθήκη επιπλέον μηδενικών στοιχείων όπως φαίνεται παραπάνω.

Αν T είναι το μητρώο στροφής (διάστασης 3×3) από το ολικό σύστημα συντεταγμένων (x_{gl}, y_{gl}, z_{gl}) στο τοπικό σύστημα $(x_{loc}, y_{loc}, z_{loc})$, τότε ισχύουν οι μετατροπές

$$\underline{u}_{loc}^{tri} = \underline{T}\ \underline{u}_{gl}^{tri} \tag{8.16}$$

$$\underline{f}_{loc}^{tri} = \underline{T}\ \underline{f}_{gl}^{tri} \tag{8.17}$$

όπου

$$\underline{f}^{tri} = \begin{bmatrix} \overrightarrow{f}_a \\ \overrightarrow{f}_b \\ \overrightarrow{f}_c \end{bmatrix}, \qquad \underline{u}^{tri} = \begin{bmatrix} \overrightarrow{u}_a \\ \overrightarrow{u}_b \\ \overrightarrow{u}_c \end{bmatrix} = \begin{bmatrix} [\delta x_a\ \delta y_a\ \delta z_a]^T \\ [\delta x_b\ \delta y_b\ \delta z_b]^T \\ [\delta x_c\ \delta y_c\ \delta z_c]^T \end{bmatrix}$$

και

$$\underline{T} = \begin{bmatrix} T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix}$$

Η μετατόπιση της κορυφής c συνδέεται γραμμικά με τις μετατοπίσεις των κορυφών 1 και 3 του τετραέδρου,

$$\overrightarrow{u}_c = \lambda \overrightarrow{u}_1 + (1-\lambda)\overrightarrow{u}_3 \tag{8.18}$$

όπου λ αδιάστατη ποσότητα που ορίζεται στο σχήμα 8.6(α′) με βάση τον κανόνα του μοχλού.

Συνεπώς, αν

$$\underline{f}^{tetra} = \begin{bmatrix} \overrightarrow{f}_1 \\ \overrightarrow{f}_2 \\ \overrightarrow{f}_3 \\ \overrightarrow{f}_4 \end{bmatrix}, \qquad \underline{u}^{tetra} = \begin{bmatrix} \overrightarrow{u}_1 \\ \overrightarrow{u}_2 \\ \overrightarrow{u}_3 \\ \overrightarrow{u}_4 \end{bmatrix} = \begin{bmatrix} [\delta x_1\ \delta y_1\ \delta z_1]^T \\ [\delta x_2\ \delta y_2\ \delta z_2]^T \\ [\delta x_3\ \delta y_3\ \delta z_3]^T \\ [\delta x_4\ \delta y_4\ \delta z_4]^T \end{bmatrix}$$

ισχύει η σχέση

$$\underline{u}_{gl}^{tri} = \underline{S}\ \underline{u}_{gl}^{tetra} \tag{8.19}$$

όπου

$$\underline{S} = \begin{bmatrix} 0 & 0 & 0 & I_{3\times3} \\ 0 & I_{3\times3} & 0 & 0 \\ \lambda I_{3\times3} & 0 & (1-\lambda)I_{3\times3} & 0 \end{bmatrix} \tag{8.20}$$

με $I_{3\times3}$ το μοναδιαίο πίνακα διάστασης 3×3.

Για να είναι ενεργειακά ισοδύναμες οι δυνάμεις που ασκούνται στις κορυφές του τετραέδρου με εκείνες που ασκούνται στις κορυφές του τριγώνου, πρέπει

$$\left(\underline{f}_{gl}^{tetra}\right)^T \underline{u}_{gl}^{tetra} = \left(\underline{f}_{gl}^{tri}\right)^T \underline{u}_{gl}^{tri} \tag{8.21}$$

Χρησιμοποιώντας τη σχέση 8.19 η 8.21 γίνεται

$$\left(\underline{f}_{gl}^{tetra}\right)^T \underline{u}_{gl}^{tetra} = \left(\underline{f}_{gl}^{tri}\right)^T \underline{S}\ \underline{u}_{gl}^{tetra}$$

Δηλαδή,

$$\underline{f}_{gl}^{tetra} = \underline{S}^T\ \underline{f}_{gl}^{tri}$$

ή, χρησιμοποιώντας τη σχέση 8.17,

$$\underline{f}_{gl}^{tetra} = \underline{S}^T\ \underline{T}^T\ \underline{f}_{loc}^{tri} \tag{8.22}$$

Η σχέση 8.22, χρησιμοποιώντας κατά σειρά τις εκφράσεις 8.7, 8.16, 8.19, γίνεται

$$\underline{f}_{gl}^{tetra} = \left(\underline{S}^T\ \underline{T}^T\ \underline{K}_{loc}^{tri}\ \underline{T}\ \underline{S}\right) \underline{u}_{gl}^{tetra}$$

ή

$$\underline{f}_{gl}^{tetra} = \underline{K}_{gl}^{tri}\ \underline{u}_{gl}^{tetra} \tag{8.23}$$

με $\underline{K}_{gl}^{tri}$ τη συνεισφορά του μητρώου δυσκαμψίας του τριγωνικού στοιχείου στο μητρώο δυσκαμψίας $\left(\underline{K}^{tetra}\right)$ του τετραέδρου, το οποίο προκύπτει από τη συγκέντρωση των 12 $\underline{K}_{gl}^{tri}$. Δηλαδή,

$$\underline{K}^{tetra} = \sum_{tri=1,12} \underline{K}_{gl}^{tri} \tag{8.24}$$

Υπολογισμός ολικού μητρώου δυσκαμψίας

Το ολικό μητρώο δυσκαμψίας $\underline{K}$ προκύπτει από τη σύνθεση των μητρώων δυσκαμψίας κάθε τετραέδρου $\left(\underline{K}^{tetra}\right)$. Τελικά, από την επίλυση του συστήματος αλγεβρικών εξισώσεων

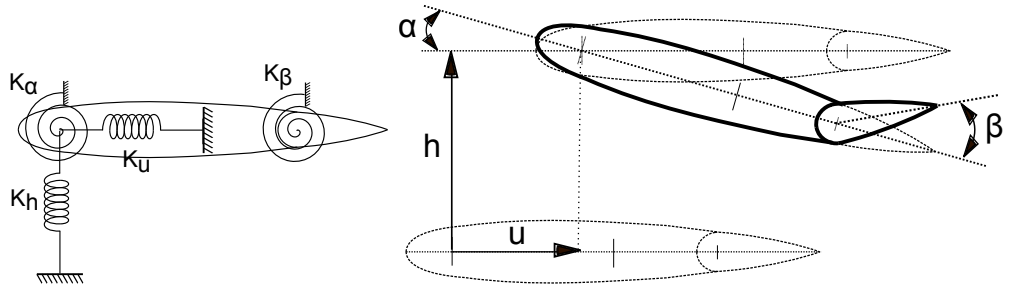
$$\underline{K}\ \underline{u} = 0 \tag{8.25}$$

που εκφράζει την ισορροπία των δυνάμεων στους κόμβους του πλέγματος, προκύπτει η θέση των κόμβων στο προσαρμοσμένο πλέγμα. Το σύστημα των εξισώσεων 8.25 δεν περιλαμβάνει τους οριακούς κόμβους η μετατόπιση των οποίων είναι γνωστή. Η επίλυση του γίνεται επαναληπτικά. Για την αρχικοποίηση της επαναληπτικής διαδικασίας χρησιμοποιείται η σχέση 8.3.

8.3 Ελαστικές εξισώσεις ισορροπίας απαραμόρφωτης αεροτομής

Η ενότητα αυτή περιγράφει τις ελαστικές εξισώσεις ισορροπίας μίας αεροτομής. Ακολουθεί, στις επόμενες ενότητες, η αδιαστατοποίηση αυτών και η περιγραφή των μεθόδων αριθμητικής ολοκλήρωσης που χρησιμοποιήθηκαν.

Θεωρείται απαραμόρφωτη αεροτομή ικανή να περιστρέφεται γύρω από τον ελαστικό της άξονα[1] και να μετακινείται κατά την κατακόρυφη και την οριζόντια κατεύθυνση. Επίσης, το τμήμα εκφυγής δύναται να στρέφεται γύρω από συγκεκριμένο άξονα περιστροφής (hinge axis). Οι τέσσερις αυτοί βαθμοί ελευθερίας της αεροτομής περιορίζονται από (γραμμικά ή στρεπτικά) ελατήρια όπως φαίνεται στο σχήμα 8.8.



Σχήμα 8.8: Οι τέσσερις βαθμοί ελευθερίας της αεροτομής, τμήμα της οποίας αποτελεί το πτερύγιο ελέγχου.

[1]Γενικά, στην περίπτωση δυναμικής ελαστικής παραμόρφωσης μίας πτέρυγας, ως ελαστικός άξονας ορίζεται η γραμμή που ενώνει τα κέντρα διάτμησης των διατομών της. Πρακτικά, ο άξονας αυτός έχει νόημα μόνο στις περιπτώσεις που η πτέρυγα δύναται να παραμορφώνεται με βάση μόνο μία καθαρή καμπτική και μία καθαρή περιστροφική ιδιομορφή αυτής. Σε πιο σύνθετες περιπτώσεις, όπου η παραμόρφωση της πτέρυγας αναλύεται σε πιο πολλές ιδιομορφές, ο ελαστικός άξονας δεν μπορεί να οριστεί. Οι ιδιομορφές μίας κατασκευής ορίζουν μία βάση δυνατών μορφών τις οποίες μπορεί να πάρει η ελαστική κατασκευή. Το σχήμα της παραμορφωμένης ελαστικής κατασκευής δίνεται ως γραμμικός συνδυασμός των ιδιομορφών αυτής. Στην περίπτωση απαραμόρφωτης αεροτομής (τομή πτέρυγας που δύναται να παραμορφώνεται με βάση μόνο μια καθαρή καμπτική και μία καθαρή περιστροφική ιδιομορφή), ο ελαστικός άξονας ή το ελαστικό κέντρο της αεροτομής πρακτικά είναι το σημείο γύρω από το οποίο δύναται αυτή να περιστρέφεται.

Οι ελαστικές εξισώσεις μπορούν να γραφούν στη μορφή

$$M\,\ddot{\overrightarrow{q}} + K\,\overrightarrow{q} = \overrightarrow{Q} \tag{8.26}$$

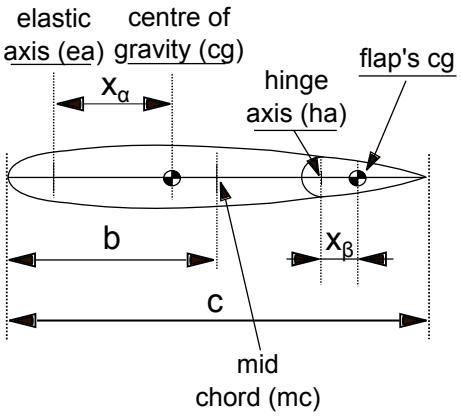
όπου $\overrightarrow{q} = \begin{bmatrix} h & u & \alpha & \beta \end{bmatrix}^T$ είναι το διάνυσμα των γενικευμένων μετατοπίσεων, με h, u και α να αποτελούν τη κατακόρυφη, την οριζόντια και την περιστροφική γύρω από τον ελαστικό άξονα μετατόπιση της αεροτομής και β τη στροφή του πτερυγίου ελέγχου (τμήμα εκφυγής). Αντίστοιχα, $\overrightarrow{Q} = \begin{bmatrix} F_y & F_x & M_{ea} & M_{ha} \end{bmatrix}^T$ είναι το διάνυσμα των γενικευμένων δυνάμεων που ασκούνται στην αεροτομή, με F_y, F_x και M_{ea} να αποτελούν τις αεροδυναμικές δυνάμεις και ροπές γύρω από τον ελαστικό άξονα που ασκούνται στην αεροτομή και M_{ha} τη ροπή που ασκείται στο πτερύγιο ελέγχου. Τα μητρώα μάζας και δυσκαμψίας (M, K αντίστοιχα) υπολογίζονται από τις εκφράσεις

$$M = \begin{bmatrix} m & 0 & S_{\alpha,x} & S_{\beta,x} \\ 0 & m & -S_{\alpha,y} & -S_{\beta,y} \\ S_{\alpha,x} & -S_{\alpha,y} & I_\alpha & I_{\alpha\beta} \\ S_{\beta,x} & -S_{\beta,y} & I_{\alpha\beta} & I_\beta \end{bmatrix} \qquad K = \begin{bmatrix} K_h & 0 & 0 & 0 \\ 0 & K_u & 0 & 0 \\ 0 & 0 & K_\alpha & 0 \\ 0 & 0 & 0 & K_\beta \end{bmatrix} \tag{8.27}$$

όπου m είναι η μάζα της αεροτομής (μαζί με το πτερύγιο ελέγχου) ανά μονάδα βάθους, $I_\alpha = mr_\alpha^2$ και $I_\beta = mr_\beta^2$ οι ροπές αδράνειας της αεροτομής και του πτερυγίου ελέγχου γύρω από τους αντίστοιχους άξονες περιστροφής και r_α, r_β οι αντίστοιχες ακτίνες περιστροφής (radii of gyration). $S_{\alpha,x} = mX_\alpha$, $S_{\alpha,y} = mY_\alpha$ και $S_{\beta,x} = mX_\beta$, $S_{\beta,y} = mY_\beta$ είναι οι στατικές ροπές (static moments) της αεροτομής και του πτερυγίου ελέγχου γύρω από τους αντίστοιχους άξονες. $\begin{bmatrix} X_\alpha & Y_\alpha \end{bmatrix}^T$, $\begin{bmatrix} X_\beta & Y_\beta \end{bmatrix}^T$ είναι οι σχετικές θέσεις των κέντρων μάζας της αεροτομής και του πτερυγίου ελέγχου ως προς τον ελαστικό άξονα και τον άξονα περιστροφής του τελευταίου. Τα κέντρα βάρους των αεροτομών που χρησιμοποιήθηκαν βρίσκονταν επί της χορδής αυτών, δηλαδή οι συντεταγμένες Y_α, Y_β είναι μηδενικές. Η ροπή αδράνειας $I_{\alpha\beta}$ υπολογίζεται συναρτήσει της σχετικής απόστασης του κέντρου περιστροφής του πτερυγίου ελέγχου από τον ελαστικό άξονα, της ροπής αδράνειας του πτερυγίου ελέγχου I_β και τις αντίστοιχες στατικές ροπές $S_{\beta,x}$, $S_{\beta,y}$, ως

$$I_{\alpha\beta} = I_\beta + S_{\beta,x}\,(x_{ha} - x_{ea}) + S_{\beta,y}\,(y_{ha} - y_{ea}) \tag{8.28}$$

όπου x_{ea}, y_{ea} είναι οι συντεταγμένες του ελαστικού άξονα και x_{ha}, y_{ha} εκείνες του κέντρου περιστροφής του πτερυγίου ελέγχου. Επιπλέον, c είναι το μήκος της χορδής της αεροτομής και $b = c/2$. Τα γεωμετρικά μεγέθη που αναφέρθηκαν προηγουμένως φαίνονται στο σχήμα 8.9.



Σχήμα 8.9: Βασικές γεωμετρικές αποστάσεις. Η θέση του ελαστικού άξονα, των κέντρων βάρους και του κέντρου περιστροφής του πτερυγίου ελέγχου είναι γνωστά.

8.4 Αδιαστατοποίηση των ελαστικών εξισώσεων

Από την αδιαστατοποίηση των εξισώσεων της ροής, οι κλίμακες αδιαστατοποίησης του χρόνου, της δύναμης και της ροπής αντίστοιχα είναι

$$t_{ref} = \frac{l_{ref}}{u_{ref}} \qquad F_{ref} = \rho_{ref} l_{ref} u_{ref}^2 \qquad M_{ref} = \rho_{ref} l_{ref}^2 U_{ref}^2$$

Στις περιπτώσεις αεροελαστικής ανάλυσης αεροτομών που παρουσιάζονται στις επόμενες ενότητες χρησιμοποιήθηκε ως μήκος αδιαστατοποίησης το μήκος της χορδής της αεροτομής ($l_{ref} = c$), ως ταχύτητα αναφοράς το μέτρο της επ΄ άπειρο ταχύτητας της ροής ($u_{ref} = U_\infty$) και ως πυκνότητα αναφοράς η πυκνότητα του ρευστού μακριά από την αεροτομή ($\varrho_{ref} = \varrho_\infty$). Δηλαδή

$$t_{ref} = \frac{c}{U_\infty} \qquad F_{ref} = \rho_\infty c U_\infty^2 \qquad M_{ref} = \rho_\infty c^2 U_\infty^2$$

Στη συνέχεια της παραγράφου, οι ποσότητες με δείκτη ref αναφέρονται στα μεγέθη αναφοράς των αντίστοιχων ελαστικών ή αεροδυναμικών ποσοτήτων. Επίσης, οι αδιάστατες ποσότητες ορίζονται με 'περισπωμένη'.

Από την εξίσωση ισορροπίας κατά την κατακόρυφη κατεύθυνση (πρώτη συνιστώσα του συστήματος 8.26) προκύπτει ότι

$$m\ddot{h} + S_{\alpha,x}\ddot{\alpha} + S_{\beta,x}\ddot{\beta} + K_h h = F_y \Rightarrow$$
$$\Rightarrow \left(\frac{m_{ref} l_{ref}}{F_{ref} t_{ref}^2}\right) \tilde{m}\ddot{\tilde{h}} + \left(\frac{S_{ref}}{F_{ref} t_{ref}^2}\right) \tilde{S}_{\alpha,x}\ddot{\tilde{\alpha}} + \left(\frac{S_{ref}}{F_{ref} t_{ref}^2}\right) \tilde{S}_{\beta,x}\ddot{\tilde{\beta}} + \left(\frac{K_{h,ref} l_{ref}}{F_{ref}}\right) \tilde{K}\tilde{h} = \tilde{F}_y$$

Για να έχουν η αδιάστατη και η διαστατή έκφραση της εξίσωσης ισορροπίας κατά τη κατακόρυφη κατεύθυνση την ίδια μορφή επιβάλλεται να είναι οι όροι εντός των παρενθέσεων ίσοι με τη μονάδα. Επομένως,

$$m_{ref} = \frac{F_{ref} t_{ref}^2}{l_{ref}} = \rho_\infty c^2$$
$$S_{ref} = F_{ref} t_{ref}^2 = \rho_\infty c^3$$
$$K_{h,ref} = \frac{F_{ref}}{l_{ref}} = \rho_\infty U_\infty^2$$

Όμοια, από την εξίσωση ισορροπίας κατά την οριζόντια κατεύθυνση, προκύπτει ότι

$$K_{u,ref} = \frac{F_{ref}}{l_{ref}} = \rho_\infty U_\infty^2 = K_{h,ref}$$

Αντίστοιχα, αντικαθιστώντας τα διαστατά με αδιάστατα μεγέθη στην εξίσωση ισορροπίας κατά την περιστροφική κατεύθυνση γύρω από τον ελαστικό άξονα (τρίτη εξίσωση του συστήματος 8.26) προκύπτει ότι

$$I_\alpha \ddot{\alpha} + S_{\alpha,x} \ddot{h} - S_{\alpha,y} \ddot{u} + I_{\alpha\beta} \ddot{\beta} + K_\alpha \alpha = M_{ea} \Rightarrow$$
$$\Rightarrow \left(\frac{I_{ref}}{M_{ref} t_{ref}^2} \right) \tilde{I}_\alpha \ddot{\tilde{\alpha}} + \left(\frac{S_{ref} l_{ref}}{M_{ref} t_{ref}^2} \right) \tilde{S}_{\alpha,x} \ddot{\tilde{h}} - \left(\frac{S_{ref} l_{ref}}{M_{ref} t_{ref}^2} \right) \tilde{S}_{\alpha,y} \ddot{\tilde{u}} + \left(\frac{I_{ref}}{M_{ref} t_{ref}^2} \right) \tilde{I}_{\alpha\beta} \ddot{\tilde{\beta}}$$
$$+ \left(\frac{K_{\alpha,ref}}{M_{ref}} \right) \tilde{K}_\alpha \tilde{\alpha} = \tilde{M}_{ea}$$

και, επιβάλλοντας οι όροι εντός των παρενθέσεων να είναι ίσοι με τη μονάδα, προκύπτουν

$$I_{ref} = M_{ref} t_{ref}^2 = \rho_\infty c^4$$
$$K_{\alpha,ref} = M_{ref} = \rho_\infty c^2 U_\infty^2$$

Όμοια, από την ισορροπία των ροπών γύρω από το κέντρο περιστροφής του πτερυγίου ελέγχου, προκύπτει

$$K_{\beta,ref} = \rho_\infty c^2 U_\infty^2 = K_{\alpha,ref}$$

Παρατηρείται ότι οι ελαστικές ποσότητες της αεροτομής αδιαστατοποιούνται με ποσότητες της ροής. Στη συνέχεια του κειμένου, ο συμβολισμός των αδιάστατων μεγεθών παραλείπεται, όμως, κάθε μέγεθος είναι αδιάστατο.

8.5 Αριθμητική ολοκλήρωση των ελαστικών εξισώσεων

Εισάγοντας το διάνυσμα $\overrightarrow{\xi} = \begin{bmatrix} q & \dot{q} \end{bmatrix}^T$ στις εξισώσεις 8.26, εκείνες παίρνουν τη μορφή συστήματος συνήθων διαφορικών εξισώσεων πρώτης τάξης

$$A \dot{\overrightarrow{\xi}} + B \overrightarrow{\xi} = \overrightarrow{C} \tag{8.29}$$

όπου

$$A = \begin{bmatrix} I_{4\times4} & 0 \\ 0 & M \end{bmatrix} \quad B = \begin{bmatrix} 0 & -I_{4\times4} \\ K & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 \\ \overrightarrow{Q} \end{bmatrix} \tag{8.30}$$

με $I_{4\times4}$ το μοναδιαίο πίνακα.

Στη διατριβή, η αριθμητική ολοκλήρωση της 8.29 πραγματοποιήθηκε τόσο με τη μέθοδο Runge-Kutta, όσο και χρησιμοποιώντας πεπερασμένες διαφορές για την προσέγγιση της χρονικής μεταβολής των μεταβλητών επίλυσης $\overrightarrow{\xi}$. Οι δύο τεχνικές που χρησιμοποιήθηκαν αναλύονται στις ακόλουθες παραγράφους.

Χρήση της μεθόδου Runge-Kutta

Η 8.29 ξαναγράφεται στη μορφή

$$\dot{\overrightarrow{\xi}} = A^{-1}\left[\overrightarrow{C} - B\overrightarrow{\xi}\right]$$

ή

$$\dot{\overrightarrow{\xi}} = \begin{bmatrix} 0 & I_{4\times4} \\ -M^{-1}K & 0 \end{bmatrix} \overrightarrow{\xi} + \begin{bmatrix} 0 \\ M^{-1}\overrightarrow{Q} \end{bmatrix} = \overrightarrow{\mathcal{G}}\left(t, \overrightarrow{Q}, \overrightarrow{\xi}\right) \tag{8.31}$$

Η επίλυση της 8.31 απαιτεί τον ταυτόχρονο υπολογισμό των ασκούμενων αεροδυναμικών φορτίων $\left(\overrightarrow{Q}\right)$ σε κάθε χρονική στιγμή. Στη συνέχεια, παρατίθεται ο αλγόριθμος επίλυσης των ελαστικών εξισώσεων με τη μέθοδο Runge-Kutta, ταυτόχρονα με εκείνη των εξισώσεων Navier–Stokes. Συγκεκριμένα, περιγράφονται οι ενέργειες που εκτελούνται για την υλοποίηση ενός χρονικού βήματος ($i \to i+1$). Κάθε βήμα της Runge-Kutta απαιτεί την επίλυση των εξισώσεων Navier–Stokes σε αντίστοιχα χρονικά βήματα με διαφορετικές οριακές συνθήκες (ταχύτητα μετατόπισης της αεροτομής). Για την επίλυση αυτή πραγματοποιείται κάθε φορά το αναγκαίο πλήθος ψευδοχρονικών βημάτων. Στην περίπτωση που οι σταθερές των ελατηρίων εξαρτώνται από την επιμήκυνση/βράχυνση αυτών (δηλαδή $K = K\left(\overrightarrow{\xi}\right)$), στον παρακάτω αλγόριθμο πρέπει η εξάρτηση αυτή να προστεθεί σε κάθε βήμα της Runge-Kutta. Δηλαδή, πρέπει να υπολογίζονται οι σταθερές των ελατηρίων μετά τον καθορισμό της θέσης της αεροτομής ($\overrightarrow{q}$).

1^o **Runge-Kutta βήμα**

- Χρονική στιγμή: $t \leftarrow t_i$
- Θέση αεροτομής: $\overrightarrow{q} \leftarrow \overrightarrow{q}_i$
- Οριακές συνθήκες: $\dot{\overrightarrow{q}} \leftarrow \dot{\overrightarrow{q}}_i$
- Μετατόπιση της αεροτομής στη νέα θέση
- Προσαρμογή του πλέγματος στη νέα θέση της αεροτομής
- Ανανέωση των γεωμετρικών στοιχείων του προβλήματος
- Εύρεση του πεδίου ροής και των αεροδυναμικών φορτίων $\overrightarrow{Q}$ τη χρονική στιγμή

t γύρω από την αεροτομή επιλύοντας τις μη-μόνιμες εξισώσεις Navier–Stokes για το χρονικό βήμα $t_{i-1} \to t_i$

- Υπολογισμός συντελεστών $\left[\overrightarrow{K}_1 \;\; \overrightarrow{L}_1\right]^T \leftarrow \Delta t \cdot \overrightarrow{\mathcal{G}}\left(t, \overrightarrow{Q}, \left[\overrightarrow{q} \;\; \overrightarrow{\dot{q}}\right]^T\right)$

2^o **Runge-Kutta βήμα**

- Χρονική στιγμή: $t \leftarrow t_{i+1/2} = t_i + \frac{1}{2}\Delta t$
- Θέση αεροτομής: $\overrightarrow{q} \leftarrow \overrightarrow{q}_i + \frac{1}{2}\overrightarrow{K}_1$
- Οριακές συνθήκες: $\overrightarrow{\dot{q}} \leftarrow \overrightarrow{\dot{q}}_i + \frac{1}{2}\overrightarrow{L}_1$
- Μετατόπιση της αεροτομής στη νέα θέση
- Προσαρμογή του πλέγματος στη νέα θέση της αεροτομής
- Ανανέωση των γεωμετρικών στοιχείων του προβλήματος
- Εύρεση του πεδίου ροής και των αεροδυναμικών φορτίων $\overrightarrow{Q}$ τη χρονική στιγμή t γύρω από την αεροτομή επιλύοντας τις μη-μόνιμες εξισώσεις Navier–Stokes για το χρονικό βήμα $t_i \to t_{i+1/2}$
- Υπολογισμός συντελεστών $\left[\overrightarrow{K}_2 \;\; \overrightarrow{L}_2\right]^T \leftarrow \Delta t \cdot \overrightarrow{\mathcal{G}}\left(t, \overrightarrow{Q}, \left[\overrightarrow{q} \;\; \overrightarrow{\dot{q}}\right]^T\right)$

3^o **Runge-Kutta βήμα**

- Χρονική στιγμή: $t \leftarrow t_{i+1/2} = t_i + \frac{1}{2}\Delta t$
- Θέση αεροτομής: $\overrightarrow{q} \leftarrow \overrightarrow{q}_i + \frac{1}{2}\overrightarrow{K}_2$
- Οριακές συνθήκες: $\overrightarrow{\dot{q}} \leftarrow \overrightarrow{\dot{q}}_i + \frac{1}{2}\overrightarrow{L}_2$
- Μετατόπιση της αεροτομής στη νέα θέση
- Προσαρμογή του πλέγματος στη νέα θέση της αεροτομής
- Ανανέωση των γεωμετρικών στοιχείων του προβλήματος
- Εύρεση του πεδίου ροής και των αεροδυναμικών φορτίων $\overrightarrow{Q}$ τη χρονική στιγμή t γύρω από την αεροτομή επιλύοντας τις μη-μόνιμες εξισώσεις Navier–Stokes για το χρονικό βήμα $t_i \to t_{i+1/2}$
- Υπολογισμός συντελεστών $\left[\overrightarrow{K}_3 \;\; \overrightarrow{L}_3\right]^T \leftarrow \Delta t \cdot \overrightarrow{\mathcal{G}}\left(t, \overrightarrow{Q}, \left[\overrightarrow{q} \;\; \overrightarrow{\dot{q}}\right]^T\right)$

4^o **Runge-Kutta βήμα**

- Χρονική στιγμή: $t \leftarrow t_{i+1} = t_i + \Delta t$
- Θέση αεροτομής: $\overrightarrow{q} \leftarrow \overrightarrow{q}_i + \overrightarrow{K}_3$
- Οριακές συνθήκες: $\overrightarrow{\dot{q}} \leftarrow \overrightarrow{\dot{q}}_i + \overrightarrow{L}_3$
- Μετατόπιση της αεροτομής στη νέα θέση
- Προσαρμογή του πλέγματος στη νέα θέση της αεροτομής
- Ανανέωση των γεωμετρικών στοιχείων του προβλήματος
- Εύρεση του πεδίου ροής και των αεροδυναμικών φορτίων $\overrightarrow{Q}$ τη χρονική στιγμή t γύρω από την αεροτομή επιλύοντας τις μη-μόνιμες εξισώσεις Navier–Stokes για το χρονικό βήμα $t_i \to t_{i+1}$
- Υπολογισμός συντελεστών $\left[\overrightarrow{K}_4 \;\; \overrightarrow{L}_4\right]^T \leftarrow \Delta t \cdot \overrightarrow{\mathcal{G}}\left(t, \overrightarrow{Q}, \left[\overrightarrow{q} \;\; \overrightarrow{\dot{q}}\right]^T\right)$

Προετοιμασία επόμενου χρονικού βήματος

- $t_{i+1} = t_i + \Delta t$
- $\overrightarrow{q}_{i+1} = \overrightarrow{q}_i + \frac{1}{6}\left(\overrightarrow{K}_1 + 2\overrightarrow{K}_2 + 2\overrightarrow{K}_3 + \overrightarrow{K}_4\right)$
- $\dot{\overrightarrow{q}}_{i+1} = \dot{\overrightarrow{q}}_i + \frac{1}{6}\left(\overrightarrow{L}_1 + 2\overrightarrow{L}_2 + 2\overrightarrow{L}_3 + \overrightarrow{L}_4\right)$

Χρήση πεπερασμένων διαφορών

Εναλλακτικά της χρησιμοποίησης της μεθόδους αριθμητικής ολοκλήρωσης Runge-Kutta, η χρήση πεπερασμένων διαφορών για την προσέγγιση της χρονικής παραγώγου $\dot{\overrightarrow{\xi}}$ μετατρέπει το διαφορικό σύστημα 8.29 στην αλγεβρική εξίσωση

$$\frac{1}{2\Delta t} A \left(3\overrightarrow{\xi}^{n+1} - 4\overrightarrow{\xi}^{n} + \overrightarrow{\xi}^{n-1}\right) + B\overrightarrow{\xi}^{n+1} = \overrightarrow{C}\left(\overrightarrow{\xi}^{n+1}\right) \tag{8.32}$$

Τη χρονική στιγμή t_{n+1}, το αλγεβρικό σύστημα 8.32 μπορεί να επιλυθεί επαναληπτικά μέσω της αναδρομικής σχέσης

$$\underbrace{\left[\frac{3}{2\Delta t}A + B\right]}_{lhs\left(\overrightarrow{\xi}_k^{n+1}\right)} \overrightarrow{\xi}_{k+1}^{n+1} = \underbrace{\overrightarrow{C}\left(\overrightarrow{\xi}_k^{n+1}\right) + \frac{1}{2\Delta t}A\left(4\overrightarrow{\xi}^{n} - \overrightarrow{y}^{n-1}\right)}_{rhs\left(\overrightarrow{\xi}_k^{n+1}\right)} \tag{8.33}$$

όπου k είναι ο μετρητής των βημάτων στο ψευδοχρόνο.

Αντί της επίλυσης του συστήματος 8.32 μέσω της αναδρομικής σχέσης 8.33, οι όροι του δεξιού μέλους του συστήματος 8.32 γραμμικοποιούνται και το σύστημα που προκύπτει επιλύεται επαναληπτικά ως προς τη διόρθωση $\delta\overrightarrow{\xi}$. Δηλαδή

$$\underbrace{\left[\frac{3}{2\Delta t}A + B - \frac{\partial\overrightarrow{C}}{\partial\overrightarrow{\xi}}\right]}_{\frac{\partial\overrightarrow{\mathcal{R}}}{\partial\overrightarrow{\xi}}\left(\overrightarrow{\xi}_k^{n+1}\right)} \delta\overrightarrow{\xi} = \underbrace{\overrightarrow{C}\left(\overrightarrow{\xi}_k^{n+1}\right) - \left[\frac{1}{2\Delta t}A\left(3\overrightarrow{\xi}_k^{n+1} - 4\overrightarrow{\xi}^{n} + \overrightarrow{\xi}^{n-1}\right) + B\overrightarrow{\xi}_k^{n+1}\right]}_{-\overrightarrow{\mathcal{R}}\left(\overrightarrow{\xi}_k^{n+1}\right)}$$

ή

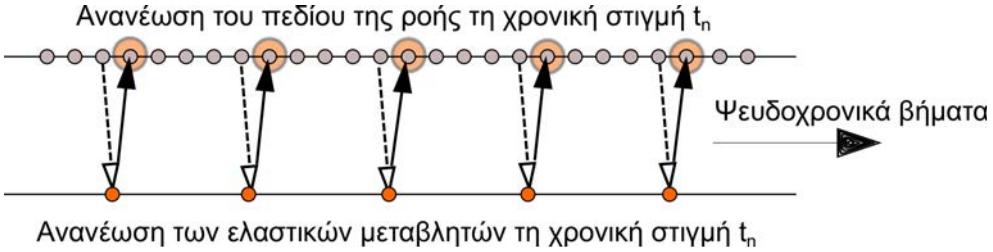
$$\frac{\partial\overrightarrow{\mathcal{R}}}{\partial\overrightarrow{\xi}}\left(\overrightarrow{\xi}_k^{n+1}\right)\ \delta\overrightarrow{\xi} = -\overrightarrow{\mathcal{R}}\left(\overrightarrow{\xi}_k^{n+1}\right) \tag{8.34}$$

με $\overrightarrow{\xi}_{k+1}^{n+1} = \overrightarrow{\xi}_k^{n+1} + \delta\overrightarrow{\xi}$.

Στη σχέση 8.34, οι παράγωγοι των αεροδυναμικών δυνάμεων και ροπών ως προς τις μεταβλητές επίλυσης $\left(\frac{\partial\overrightarrow{C}}{\partial\overrightarrow{\xi}}\right)$ υπολογίζονται προσεγγιστικά με βάση τη θεωρία λεπτών αεροτομών, δηλαδή

$$\begin{array}{llll} \frac{\partial F_y}{\partial\alpha} = \pi\rho_\infty U_\infty^2 c & \frac{\partial F_y}{\partial\dot{\alpha}} = \frac{\pi\rho_\infty U_\infty c^2}{4} & \frac{\partial F_y}{\partial h} = 0 & \frac{\partial F_y}{\partial\dot{h}} = \pi\rho_\infty U_\infty c \\ \frac{\partial M}{\partial\alpha} = \frac{\pi\rho_\infty U_\infty^2 c^2}{4} & \frac{\partial M}{\partial\dot{\alpha}} = 0 & \frac{\partial M}{\partial h} = 0 & \frac{\partial M}{\partial\dot{h}} = \frac{\pi\rho_\infty U_\infty c^2}{4} \end{array} \tag{8.35}$$

Γενικά, η επίλυση της 8.33 ή της 8.34 χρειάζεται πολύ μικρότερο αριθμό βημάτων στον ψευδοχρόνο σε σχέση με εκείνη των εξισώσεων Navier–Stokes, για κάθε χρονική στιγμή. Επομένως, προτείνεται η επικοινωνία μεταξύ των δύο επιλυτών να μην γίνεται στο τέλος κάθε ψευδοχρονικού βήματος αλλά μετά το πέρας μίας σειράς (λ.χ. 100) ψευδοχρονικών βημάτων της επίλυσης των εξισώσεων Navier–Stokes. Κατά την επικοινωνία των δύο επιλυτών, ο επιλύτης των ελαστικών εξισώσεων χρησιμοποιεί τις τιμές των αεροδυναμικών φορτίων που έχει υπολογίσει ο επιλύτης της ροής τη στιγμή εκείνη, για να ανανεώσει τις ελαστικές μεταβλητές της αεροτομής (μετατοπίσεις και χρονικές παραγώγους αυτών). Στη συνέχεια, το υπολογιστικό πλέγμα προσαρμόζεται στη νέα θέση της αεροτομής, ανανεώνονται τα γεωμετρικά στοιχεία του τελευταίου και συνεχίζεται η επίλυση της ροής με τις νέες οριακές συνθήκες (όπως προέκυψαν από την ανανέωση των ελαστικών μεταβλητών) για μία σειρά ψευδοχρονικών βημάτων μέχρι την επόμενη επικοινωνία του ελαστικού με τον επιλύτη της ροής. Η διαδικασία συνεχίζεται μέχρι τη σύγκλιση τόσο των ελαστικών όσο και των εξισώσεων της ροής. Τα παραπάνω επαναλαμβάνονται για κάθε πραγματικό χρονικό βήμα, μέχρι την ολοκλήρωση του επιθυμητού αριθμού αυτών. Το σχήμα 8.10 παρουσιάζει σχηματικά την επίλυση των ελαστικών και αεροδυναμικών εξισώσεων για μία χρονική στιγμή.



Σχήμα 8.10: Ανταλλαγή πληροφορίας μεταξύ του επιλύτη των ελαστικών εξισώσεων και των εξισώσεων της ροής. Ο πρώτος χρησιμοποιεί τα αεροδυναμικά φορτία που έχει υπολογίσει τη στιγμή εκείνη ο αεροδυναμικός επιλύτης (διακεκομμένο βέλος) για να ανανεώσει τις ελαστικές μεταβλητές. Ο δεύτερος προσαρμόζει το πλέγμα στη νέα θέση της αεροτομής, ανανεώνει τα σχετικά με το πλέγμα γεωμετρικά στοιχεία (διπλός κύκλος) και χρησιμοποιεί τις ανανεωμένες ελαστικές μεταβλητές (συνεχές βέλος) ως οριακές συνθήκες για την ανανέωση του πεδίου της ροής. Η επικοινωνία μεταξύ των δύο επιλυτών γίνεται ανά μία σειρά ψευδοχρονικών βημάτων του επιλύτη της ροής.

8.6 Εξαναγκασμένη περιοδική κίνηση αεροτομής - Πιστοποίηση του GPU-κώδικα

Ο GPU-κώδικας, πριν χρησιμοποιηθεί για την επίλυση αεροελαστικών προβλημάτων, πιστοποιήθηκε σε εφαρμογή εξαναγκασμένης περιοδικής ταλάντωσης αεροτομής (pitching airfoil) ως προς το $1/4$ της χορδής. Τα αριθμητικά αποτελέσματα συγκρίθηκαν με αντίστοιχα πειραματικά και η συμφωνία, όπως θα δειχθεί είναι ικανοποιητική.

Επιλέχθηκε η συμμετρική αεροτομή NACA0012. Η αεροτομή εκτελεί ημιτονοειδή περιστροφική κίνηση πλάτους 2.51^o, με την επ΄ άπειρο ροή να σχηματίζει γωνία 0.016^o

με την οριζόντια κατεύθυνση. Η ανηγμένη συχνότητα $k \triangleq \frac{\omega c}{2U_\infty}$ της ταλάντωσης είναι ίση με 0.0814 και ο αριθμός Mach της επ΄ άπειρο ροής είναι 0.755. Στον ορισμό της ανηγμένης συχνότητας, ω είναι η γωνιακή ταχύτητα περιστροφής της αεροτομής. Τα χαρακτηριστικά της κίνησης της αεροτομής και οι συνθήκες της επ΄ άπειρο ροής συνοψίζονται στον πίνακα 8.1. Στη συνέχεια της ενότητας, ως γωνία πρόσπτωσης, ορίζεται η σχετική γωνία της επ΄ άπειρο ροής $\alpha(t)$ ως προς τη στιγμιαία θέση της αεροτομής.

Τύπος αεροτομής	NACA0012
Σχετική γωνία της επ΄ άπειρο ροής ως προς τη στιγμιαία θέση της αεροτομής	$\alpha(t) = \alpha_m + \alpha_0 sin(\omega t)$
Γωνία επ΄ άπειρο ροής	$\alpha_m = 0.016^o$
Πλάτος ταλάντωσης	$\alpha_0 = 2.51^o$
Ανηγμένη συχνότητα	$k \triangleq \frac{\omega c}{2U_\infty} = 0.0814$
Άξονας περιστροφής	$0.25c$
Αριθμός $Mach$ της ροής	$M_\infty = 0.755c$

Πίνακας 8.1: Χαρακτηριστικά μεγέθη της περιοδικής κίνησης της αεροτομής.

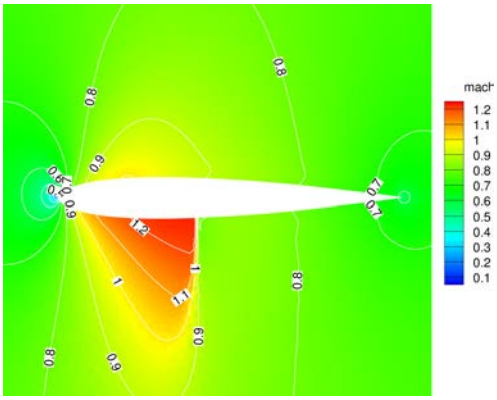
Η ίδια περίπτωση έχει μελετηθεί και πειραματικά. Οι σχετικές μετρήσεις δείχνουν ότι, κατά την ταλάντωση της αεροτομής, η ροή παραμένει προσκολλημένη και εμφανίζεται κύμα κρούσης τόσο στην πλευρά υποπίεσης όσο και στην πλευρά υπερπίεσης, σε διαφορετικές φάσεις της ταλάντωσης.

Στους υπολογισμούς που έγιναν, χρησιμοποιήθηκε η MPA εκδοχή του GPU-κώδικα για την επίλυση των εξισώσεων Euler σε μη-δομημένο πλέγμα. Υπενθυμίζεται ότι η επιτάχυνση της πρόλεξης των αριθμητικών αποτελεσμάτων εξαρτάται από το μοντέλο της ροής (εδώ ατριβής), το πλήθος των κόμβων του χρησιμοποιούμενου πλέγματος και την τεχνολογία της χρησιμοποιούμενης GPU. Το πλέγμα που χρησιμοποιήθηκε κατασκευάστηκε με την τεχνική του προελαύνοντος μετώπου και αποτελείται από περίπου 24000 κόμβους και 48000 τριγωνικά στοιχεία. Η επιτάχυνση στην πρόλεξη της ροής που δίνει η χρήση μίας Tesla M2050, αντί ενός πυρήνα μίας σημερινής CPU, είναι περίπου 40x. Η ανανέωση των γεωμετρικών στοιχείων του πλέγματος, μετά την προσαρμογή αυτού, γίνεται στη GPU.

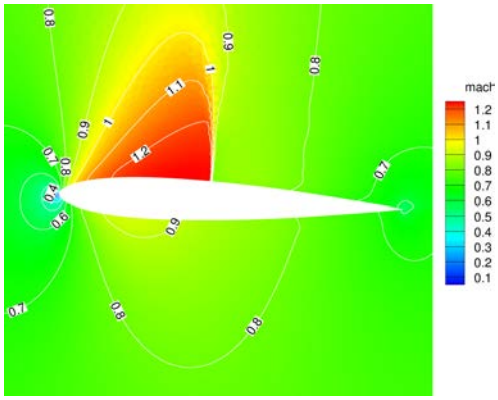
Τα σχήματα 8.11 δείχνουν τα υπολογισθέντα πεδία του αριθμού Mach γύρω από την αεροτομή όταν η αεροτομή περνά από (α) τη 'μέση' θέση με αυξανόμενη γωνία πρόσπτωσης (σχήμα 8.11(α΄)), (β) τη θέση της μέγιστης θετικής γωνίας πρόσπτωσης (σχήμα 8.11(β΄)), (γ) τη 'μέση' θέση με μειούμενη γωνία πρόσπτωσης (σχήμα 8.11(γ΄)) και (δ) τη θέση της μέγιστης αρνητικής γωνίας πρόσπτωσης (σχήμα 8.11(δ΄)).

Τα σχήματα 8.12, 8.13 δείχνουν την κατανομή του συντελεστή πίεσης στην αεροτομή για διαφορετικές φάσεις ($\phi = \omega t$) της κίνησης με θετικές (σχήμα 8.12) ή αρνητικές (σχήμα 8.13) τιμές της γωνίας πρόσπτωσης $\alpha(t)$. Η σύγκριση με τα πειραματικά αποτελέσματα, [237], που παρουσιάζονται επίσης στα σχήματα 8.12 και 8.13 είναι πολύ καλή.

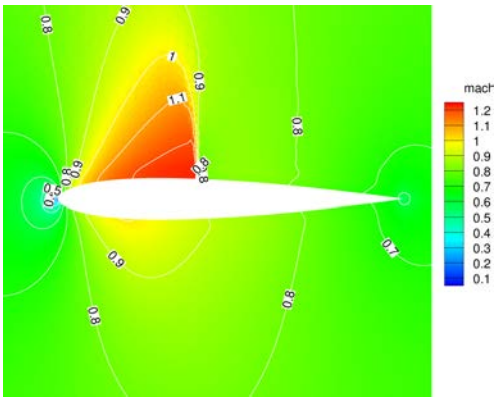
Η ανάλυση κατά Fourier των συντελεστών πίεσης ανά κόμβο της αεροτομής που υ-



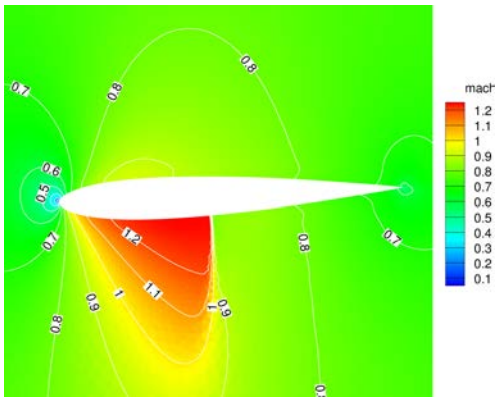
(α′) Μέση γωνία πρόσπτωσης ($\alpha = 0.016^o$), στη φάση που η γωνία πρόσπτωσης αυξάνει.



(β′) Μέγιστη θετική γωνία πρόσπτωσης ($\alpha = 0.016^o + 2.51^o$).



(γ′) Μέση γωνία πρόσπτωσης ($\alpha = 0.016^o$), στη φάση που η γωνία πρόσπτωσης μειώνεται.

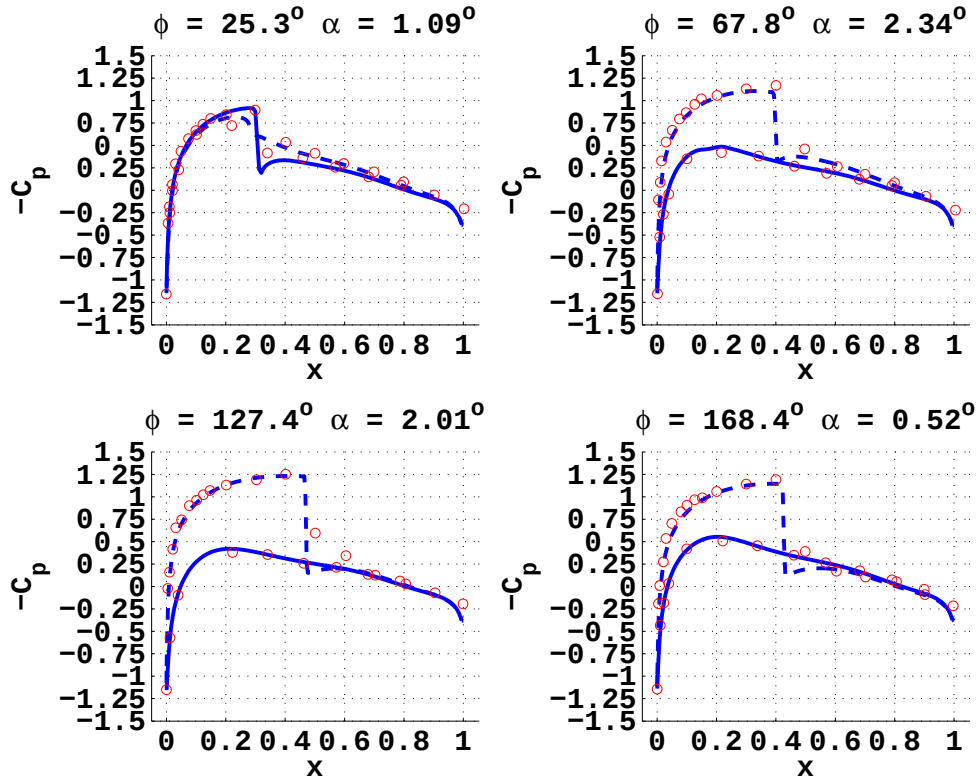


(δ′) Μέγιστη αρνητική γωνία πρόσπτωσης ($\alpha = 0.016^o - 2.51^o$).

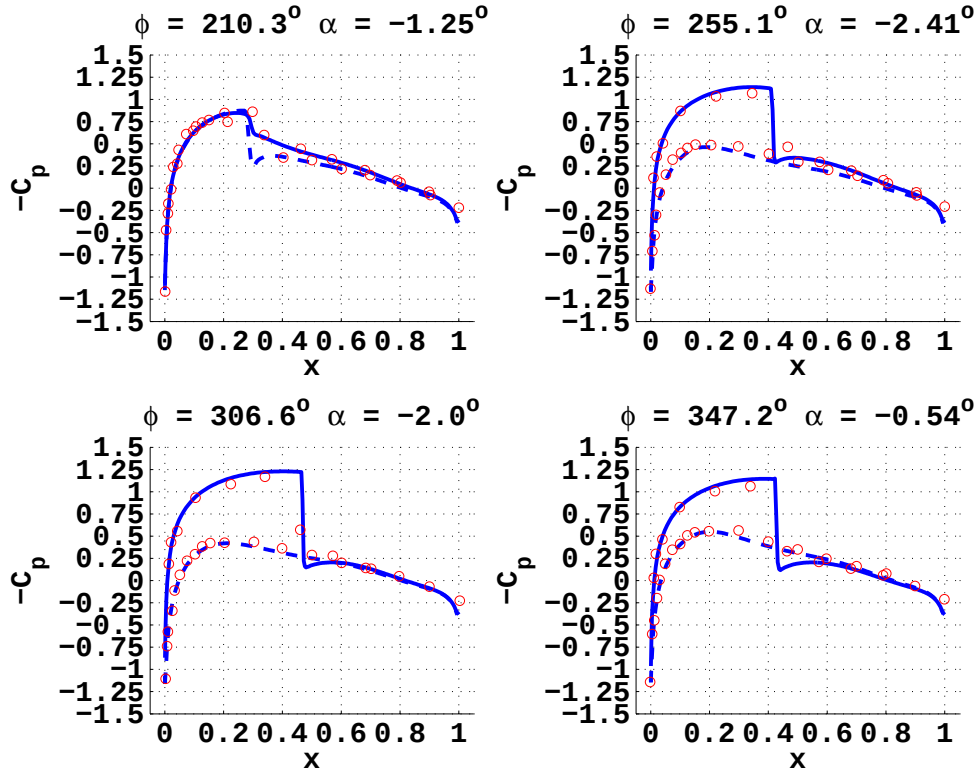
Σχήμα 8.11: Περιοδική κίνηση πρόνευσης αεροτομής NACA0012 ως προς το $1/4$ της χορδής με $k = 0.0814$ και $\alpha_m = 0.016^o$, $\alpha_0 = 2.51^o$, $M_\infty = 0.755$. Υπολογισθέντα πεδία του αριθμού Mach γύρω από την αεροτομή σε τέσσερις διαφορετικές φάσεις της κίνησης.

πολογίστηκαν αριθμητικά φαίνεται στο σχήμα 8.14. Η συμφωνία με τους αντίστοιχους συντελεστές που προκύπτουν από την ανάλυση των πειραματικών δεδομένων είναι, επίσης, πολύ καλή. Συγκεκριμένα, στο σχήμα 8.14 παρουσιάζεται το πραγματικό (αριστερή στήλη) και το φανταστικό (δεξιά στήλη) μέρος των τριών πρώτων συντελεστών που προκύπτουν από την ανάλυση.

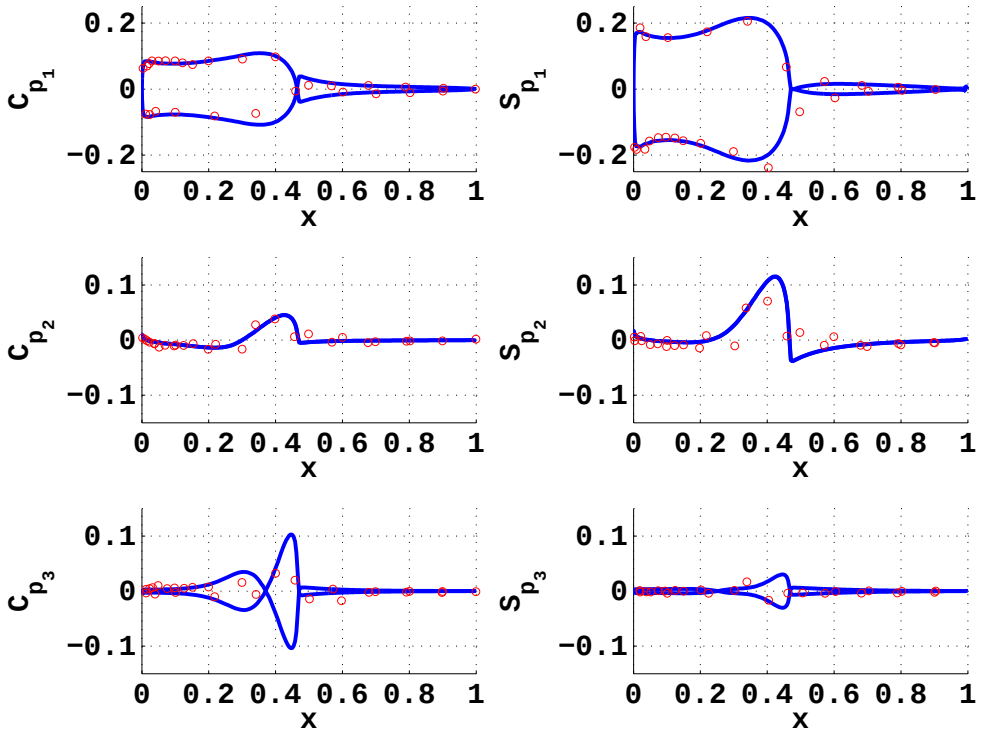
Τα σχήματα 8.15 δείχνουν τη μεταβολή του συντελεστών άνωσης και ροπής ως προς το $1/4$ της χορδής της αεροτομής με τη γωνία πρόσπτωσης. Η σύγκριση με πειραματικές μετρήσεις είναι ικανοποιητική. Παρατηρείται ότι, αν και η αεροτομή είναι συμμετρική, η μέση τιμή του συντελεστή άνωσης με βάση τα πειραματικά δεδομένα είναι διάφορη του μηδενός. Το γεγονός αυτό πιθανώς οφείλεται στην επίδραση των τοιχωμάτων της αεροδυναμικής σήραγγας όπου έγιναν οι συγκεκριμένες μετρήσεις και δικαιολογεί τη μικρή απόκλιση των υπολογιστικών από τα πειραματικά αποτελέσματα.



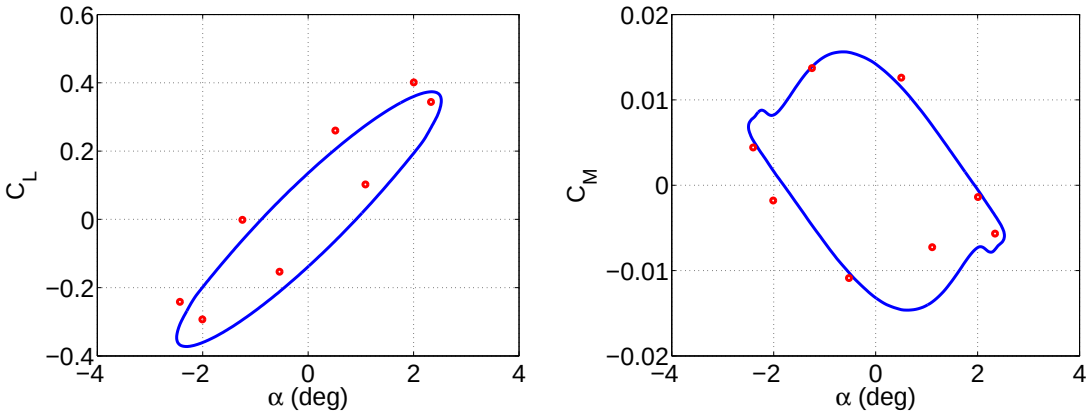
Σχήμα 8.12: Περιοδική κίνηση πρόνευσης αεροτομής NACA0012 ως προς το 1/4 της χορδής με $k = 0.0814$ και $\alpha_m = 0.016^o$, $\alpha_0 = 2.51^o$, $M_\infty = 0.755$. Υπολογισθείσες κατανομές του συντελεστή πίεσης επί της πλευράς υπερπίεσης (συνεχής γραμμή) και υποπίεσης (διακεκομμένη γραμμή) της αεροτομής, σε διαφορετικές φάσεις της ταλάντωσης με θετική στιγμιαία γωνία πρόσπτωσης. Σύγκριση με πειραματικά αποτελέσματα (κύκλοι), [237].



Σχήμα 8.13: Περιοδική κίνηση πρόνευσης αεροτομής NACA0012 ως προς το 1/4 της χορδής με $k=0.0814$ και $\alpha_m=0.016^o$, $\alpha_0=2.51^o$, $M_\infty=0.755$. Υπολογισθείσες κατανομές του συντελεστή πίεσης επί της πλευράς υπερπίεσης (συνεχής γραμμή) και υποπίεσης (διακεκομμένη γραμμή) της αεροτομής, σε διαφορετικές φάσεις της ταλάντωσης με στιγμιαία αρνητική γωνία πρόσπτωσης. Σύγκριση με πειραματικά αποτελέσματα (κύκλοι), [237].



Σχήμα 8.14: Περιοδική κίνηση πρόνευσης αεροτομής NACA0012 ως προς το 1/4 της χορδής με $k = 0.0814$ και $\alpha_m = 0.016^o$, $\alpha_0 = 2.51^o$, $M_\infty = 0.755$. Ανάλυση κατά Fourier του συντελεστή της πίεσης. Φαίνεται το πραγματικό (αριστερή στήλη) και το φανταστικό (δεξιά στήλη) μέρος των τριών πρώτων συντελεστών της ανάλυσης Fourier τόσο των υπολογιστικών αποτελεσμάτων, όσο και των πειραματικών δεδομένων.



(α′) (β′)

Σχήμα 8.15: Περιοδική κίνηση πρόνευσης αεροτομής NACA0012 ως προς το 1/4 της χορδής με $k = 0.0814$ και $\alpha_m = 0.016^o$, $\alpha_0 = 2.51^o$, $M_\infty = 0.755$. Μεταβολή (α′) του συντελεστή άνωσης και (β′) του συντελεστή ροπής συναρτήσει της γωνίας πρόσπτωσης, όπως προέκυψαν από την αριθμητική ολοκλήρωση των εξισώσεων Euler και πειραματικές μετρήσεις, [237].

Ολοκληρώνοντας την ενότητα, αξίζει να σημειωθεί ότι, στους υπολογισμούς που παρουσιάστηκαν, οι εξισώσεις της ροής εκφράστηκαν ως προς τους απόλυτους (αδρανειακούς) άξονες συντεταγμένων με την αεροτομή να περιστρέφεται ημιτονοειδώς γύρω από το $^1/_4$ της χορδής της, παραμορφώνοντας το κομμάτι του πλέγματος που την περιβάλλει (όπως υποδεικνύουν τα σχήματα 8.4). Εναλλακτικά, για τη συγκεκριμένη εφαρμογή, θα μπορούσε το σύνολο του πλέγματος να περιστρέφεται μαζί με την αεροτομή, είτε εκείνο να παραμένει ακίνητο και οι εξισώσεις της ροής να εκφραστούν ως προς τους σχετικούς άξονες της αεροτομής. Οι τελευταίοι ακολουθούν την κίνηση της αεροτομής και έχουν αρχή το $^1/_4$ της χορδής. Όταν περιστρέφεται το συνολικό πλέγμα, ο υπολογισμός των ταχυτήτων μετατόπισης των κόμβων του πλέγματος δίνεται από μία απλή αναλυτική σχέση. Αντίθετα, εκφράζοντας τις εξισώσεις της ροής ως προς τους σχετικούς άξονες της αεροτομής, αποφεύγεται ο υπολογισμός των ταχυτήτων των κόμβων του πλέγματος, μιας και η αεροτομή στο σχετικό σύστημα είναι ακίνητη. Πρέπει να προστεθεί, όμως, η επίδραση της κεντρομόλου δύναμης, της δύναμης Coriolis και της χρονικής μεταβολής της γωνιακής ταχύτητας περιστροφής του σχετικού συστήματος ως προς το απόλυτο στις εξισώσεις ενέργειας και ισορροπίας των ροπών (κεφάλαιο 2). Οι δύο εναλλακτικές μέθοδοι προγραμματίστηκαν και αξιολογήθηκαν τόσο ως προς την ταύτιση των αποτελεσμάτων τους όσο και ως προς την επιτάχυνση της πρόλεξης της ροής. Η απόδοση του GPU-κώδικα παρέμεινε αντίστοιχα υψηλή και με τους δύο τρόπους. Επιπλέον, τα αποτελέσματα που προκύπτουν είναι ταυτόσημα με εκείνα που ήδη έχουν παρουσιαστεί και για αυτό και αποφεύχθηκε η επανάληψη αυτών. Οι εφαρμογές που ακολουθούν επιλύουν τις εξισώσεις της ροής στο απόλυτο σύστημα, παραμορφώνοντας ένα τμήμα του πλέγματος γύρω από το αεροδυναμικό σώμα ώστε αυτό να ακολουθεί την κίνηση του τελευταίου.

8.7 Αεροελαστική ανάλυση αεροτομής δύο βαθμών ελευθερίας

Στην ενότητα αυτή μελετάται η αεροελαστική συμπεριφορά της τομής του ακροπτερυγίου μίας οπισθοκλινούς πτέρυγας, [238]. Πρόκειται για αεροτομή τύπου NACA64A010. Η κίνηση της περιορίζεται από ένα γραμμικό και ένα στρεπτικό ελατήριο. Το γραμμικό ελατήριο είναι κατά την κατακόρυφη κατεύθυνση ενώ το στρεπτικό είναι τοποθετημένο στον ελαστικό άξονα της αεροτομής και περιορίζει την κίνηση αυτής κατά την περιστροφική κατεύθυνση. Πρόκειται δηλαδή για ένα ελαστικό μοντέλο δύο βαθμών ελευθερίας.

Τα χαρακτηριστικά του μοντέλου φαίνονται στον πίνακα 8.2. Ο ελαστικός άξονας βρίσκεται εκτός της πτέρυγας στην περιοχή του ακροπτερυγίου αυτής (ανάντι της ακμής πρόσπτωσης). ω_α, ω_h είναι οι φυσικές συχνότητες της αεροτομής στους δύο βαθμούς ελευθερίας α (γωνιακή μετατόπιση ως προς τον ελαστικό άξονα), h (κατακόρυφη μετατόπιση) αντίστοιχα. m είναι η μάζα της αεροτομής.

Για την αεροελαστική ανάλυση της αεροτομής, αρχικά αυτή τίθεται σε ημιτονοειδή κίνηση κατά την περιστροφική κατεύθυνση με πλάτος ίσο με 1^o και συχνότητα ταλάντωσης ίση με τη φυσική συχνότητα περιστροφής της αεροτομής (ω_α). Το χρονικό

Τύπος αεροτομής	NACA64A010
Θέση ελαστικού άξονα	$x_{ea} = -0.5$
Κέντρο μάζας της αεροτομής	$X_\alpha = 0.9$
Λόγος φυσικών συχνοτήτων	$\frac{\omega_h}{\omega_\alpha} = 1$
Ακτίνα περιστροφής	$r_\alpha^2 = 3.48$
Λόγος πυκνοτήτων	$\mu \triangleq \frac{m}{\pi\rho_\infty b^2} = 60$

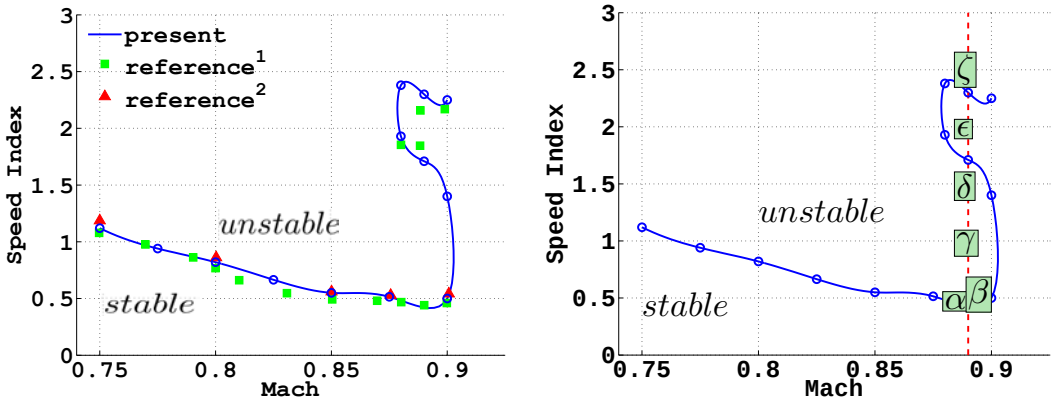
Πίνακας 8.2: Χαρακτηριστικά του αεροελαστικού μοντέλου.

διάστημα της εξαναγκασμένης ταλάντωσης έχει διάρκεια ίση με την αντίστοιχη περίοδο $\left(\frac{2.\pi}{\omega_\alpha}\right)$. Στη συνέχεια, η αεροτομή αφήνεται να κινηθεί σύμφωνα με το γραμμικό και στρεπτικό ελατήριο που είναι προσδεδεμένα σ΄ αυτή. Ανάλογα με την τιμή του αριθμού Mach και της ανηγμένης ταχύτητας της ροής $\left(U^* \triangleq \frac{U_\infty}{b\omega_\alpha\sqrt{\mu}}\right)$, η αεροτομή συγκλίνει στη θέση ισορροπίας (ευσταθής απόκριση), συνεχίζει να ταλαντώνεται με σταθερό πλάτος (οριακά ευσταθής απόκριση) ή συνεχίζει να ταλαντώνεται με αυξανόμενο πλάτος (ασταθής απόκριση). Όπως αναφέρθηκε στην ενότητα 8.1, στη διηχητική περιοχή για μεγάλες τιμές της ανηγμένης ταχύτητας της ροής, η απόκριση της αεροτομής μπορεί να συγκλίνει σε μία ταλάντωση σταθερού πλάτους αρκετά μεγαλύτερου σε σχέση με το πλάτος ταλάντωσης της πρώτης περιόδου (Limit Cycle Oscillation, LCO).

Στόχος της αεροελαστικής ανάλυσης που πραγματοποιήθηκε ήταν η χάραξη της κρίσιμης καμπύλης της αεροτομής. Για τη χάραξη αυτής πραγματοποιήθηκε σειρά υπολογισμών με τις τιμές του αριθμού Mach της ροής να μεταβάλλονται από 0.75 έως 0.91 και εκείνες της ανηγμένης ταχύτητας της ροής να μεταβάλλονται από 0. έως 3. Η γωνία της επ΄ άπειρον ροής είναι 0^o. Ο υπολογισμός των αεροδυναμικών φορτίων έγινε με την επίλυση των εξισώσεων Euler.

Το σχήμα 8.16(α΄) δείχνει την υπολογισμένη κρίσιμη καμπύλη της αεροτομής. Η σύγκριση με αριθμητικά αποτελέσματα άλλων ερευνητικών ομάδων είναι ικανοποιητική. Στο σχήμα αυτό, η κρίσιμη καμπύλη χωρίζει το χώρο σε ευσταθή (περιοχή κάτω από την κρίσιμη καμπύλη) και ασταθή (περιοχή πάνω από την κρίσιμη καμπύλη) περιοχή. Ξεκινώντας από ένα σημείο εντός της περιοχής ευσταθούς λειτουργίας και αυξάνοντας την ανηγμένη ταχύτητα της ροής, διατηρώντας σταθερή την τιμή του αριθμού Mach, η αεροτομή εισέρχεται στην περιοχή ασταθούς λειτουργίας. Πρακτικά, αύξηση της ανηγμένης ταχύτητας της ροής με σταθερό αριθμό Mach σημαίνει ελάττωση της ποσότητας αδιαστατοποίησης της ταχύτητας της επ΄ άπειρο ροής ($b\omega_\alpha\sqrt{\mu}$). Επειδή, στην εφαρμογή, ο λόγος πυκνοτήτων ρευστού προς υλικού κατασκευής της αεροτομής είναι σταθερός ($\mu = 60$), ελάττωση του όρου $b\omega_\alpha\sqrt{\mu}$ σημαίνει μείωση της φυσικής συχνότητας ω_α της αεροτομής. Η σταθερά ελατηρίου (K_α) συνδέεται με τη φυσική συχνότητα ω_α μέσω της σχέσης: $K_\alpha = I_\alpha\omega_\alpha^2$. Συνεπώς, η μείωση του ω_α συνεπάγεται τη μείωση του K_α αλλά και του K_h, μιας και οι δύο φυσικές συχνότητες συνδέονται, στην παρούσα εφαρμογή $\frac{\omega_h}{\omega_\alpha} = 1$, και ισχύει $K_h = m\omega_h^2$. Τελικά, η αύξηση της ανηγμένης ταχύτητας της ροής με σταθερό αριθμό Mach, συνεπάγεται τη μείωση των σταθερών δυσκαμψίας της αεροτομής, που έχει ως λογικό επακόλουθο η κατασκευή της αεροτομής να γίνεται

πιο 'εύκαμπτη' και η αεροτομή να πλησιάζει την περιοχή ασταθούς λειτουργίας. Πρακτικά, σε κάθε σημείο στο χώρο του σχήματος 8.16(α΄) ο οριζόντιος άξονας (αριθμός Mach) ορίζει τις συνθήκες της ροής ενώ ο κατακόρυφος άξονας (ανηγμένη ταχύτητα της επ΄ άπειρο ροής) ορίζει τα δομικά χαρακτηριστικά της αεροτομής.



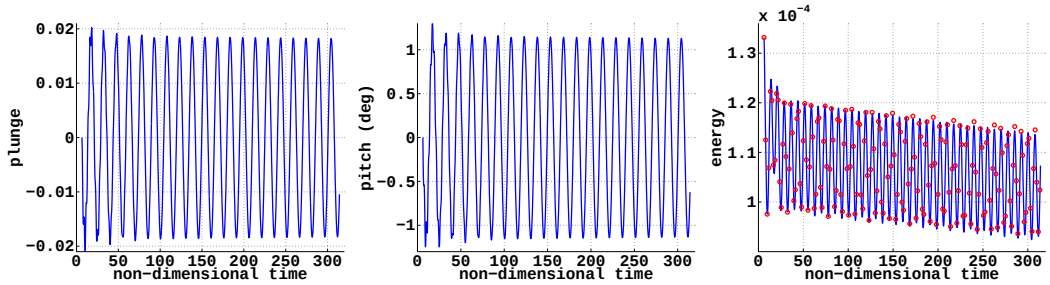
(α΄) (β΄)

Σχήμα 8.16: Αεροελαστική ανάλυση αεροτομής NACA64A010 δύο βαθμών ελευθερίας. Υπολογισθείσα κρίσιμη καμπύλη αεροτομής. (α΄) Σύγκριση με αριθμητικά αποτελέσματα άλλων ερευνητικών ομάδων, [208, 211]. (β΄) Ορισμός με γράμματα διαφορετικών περιπτώσεων σε διηχητικές συνθήκες λειτουργίας. Η απόκριση των περιπτώσεων αυτών φαίνεται στα σχήματα 8.17.

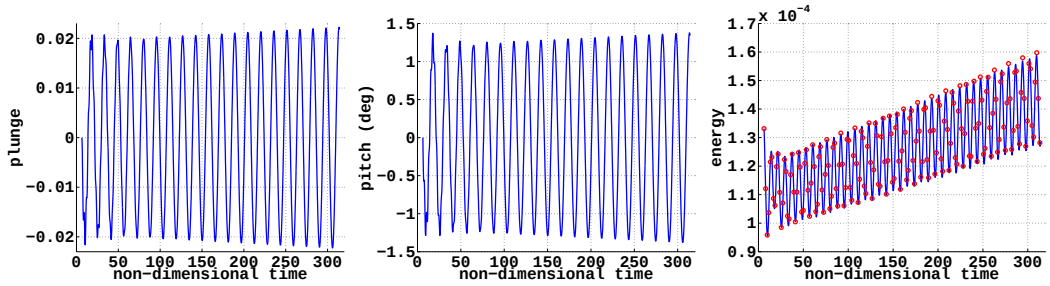
Στο σχήμα 8.16(α΄), παρατηρείται επίσης η πτώση της κρίσιμης ανηγμένης ταχύτητας της ροής με την αύξηση της τιμής του αριθμού Mach της ροής έως τη διηχητική περιοχή (transonic dip). Στη συνέχεια, η κρίσιμη ανηγμένη ταχύτητα της ροής αυξάνει. Αρκετά ενδιαφέρον είναι ότι, τουλάχιστον για τις τιμές ανηγμένης ταχύτητας που δοκιμάστηκαν, η απόκριση της αεροτομής για αριθμούς Mach μεγαλύτερους περίπου του 0.9 είναι ευσταθής.

Στη διηχητική περιοχή παρατηρείται ότι υπάρχουν πολλαπλά σημεία μετάβασης από την ευσταθή περιοχή στην ασταθή και αντίστοιχα. Τα σχήματα 8.17 δείχνουν την απόκριση των περιπτώσεων που έχουν σημειωθεί στο σχήμα 8.16(β΄). Οι δύο πρώτες στήλες των σχημάτων 8.17 δείχνουν την απόκριση της αεροτομής κατά την κατακόρυφη και την περιστροφική κατεύθυνση αντίστοιχα. Η τρίτη στήλη δείχνει το άθροισμα της στιγμιαίας αδιάστατης κινητικής και δυναμικής ενέργειας της αεροτομής. Το άθροισμα αυτό, σε κάθε χρονική στιγμή, ταυτίζεται με τη συνολική ενέργεια που δίνεται από το ρευστό στην αεροτομή, δηλαδή το έργο των ασκούμενων αεροδυναμικών φορτίων. Η επιτυχής αυτή σύγκριση πιστοποιεί την ορθότητα της σύζευξης του αεροδυναμικού GPU-κώδικα με τον CPU-επιλύτη των ελαστικών εξισώσεων. Οι περιπτώσεις α, β είναι ελαφρώς κάτω και πάνω από την κρίσιμη καμπύλη αντίστοιχα. Αντίθετα, οι περιπτώσεις γ,δ έχουν ασταθή απόκριση. Η περίπτωση ε, που έχει υψηλότερη σε σχέση με τις προηγούμενες περιπτώσεις τιμή ανηγμένης ταχύτητας επ΄ άπειρο ροής, είναι ξεκάθαρα ευσταθής. Τέλος, η περίπτωση ζ οδηγεί σε ταλάντωση σταθερού πλάτους (LCO) εμφανώς πολύ μεγαλύτερου σε σχέση με το πλάτος της αρχικής (εξαναγκασμένης)

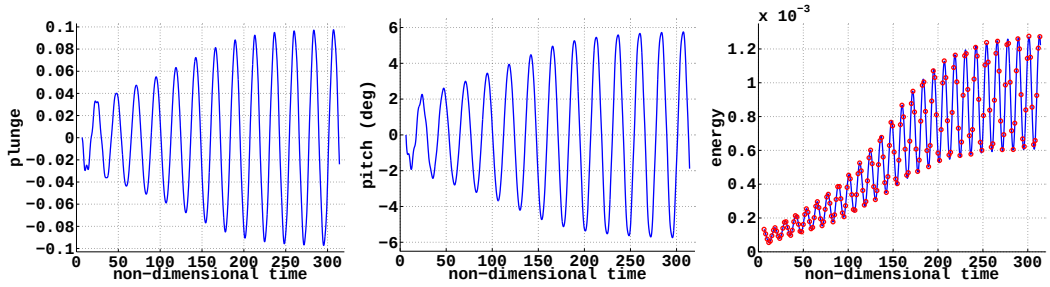
ταλάντωσης.



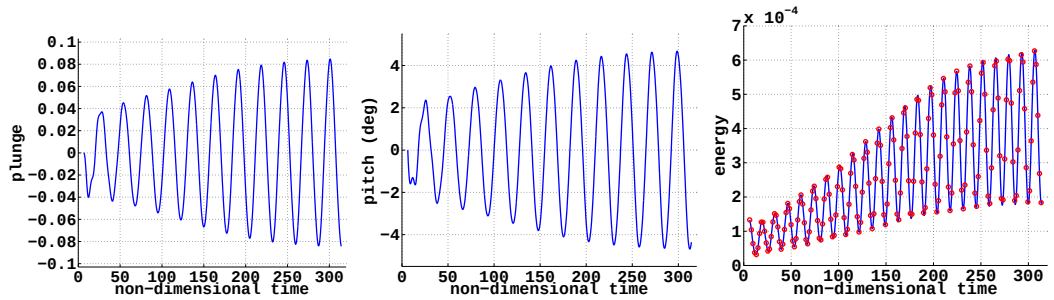
(α′) $U^* = 0.49,\ Mach = 0.89$



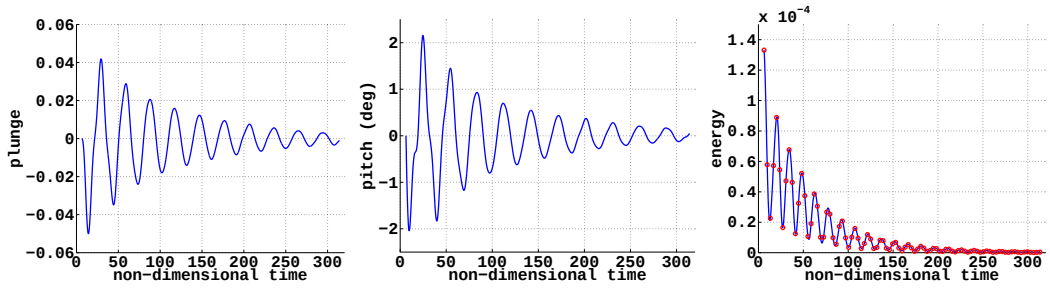
(β′) $U^* = 0.51,\ Mach = 0.89$



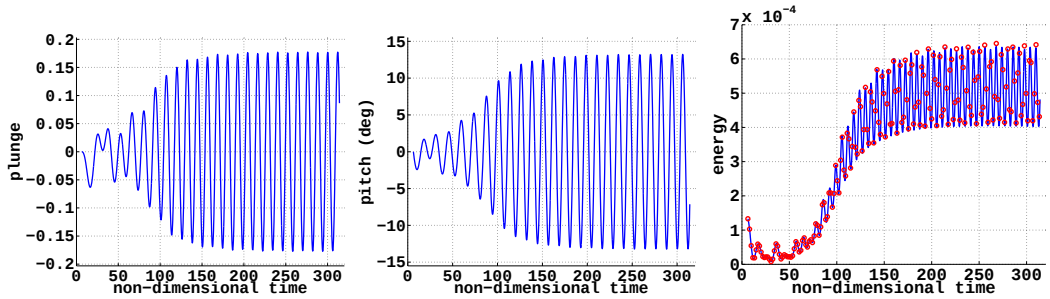
(γ′) $U^* = 1.00,\ Mach = 0.89$



(δ′) $U^* = 1.50,\ Mach = 0.89$



(ε′) $U^* = 2.00,\ Mach = 0.89$



(ϛ′) $U^* = 2.50,\ Mach = 0.89$

Σχήμα 8.17: Αεροελαστική ανάλυση αεροτομής NACA64A010 δύο βαθμών ελευθερίας. Φαίνεται η απόκριση 6 διαφορετικών περιπτώσεων με ίδιο αριθμό Mach και διαφορετική ανηγμένη ταχύτητα της επ΄ άπειρο ροής. Οι περιπτώσεις αυτές φαίνονται στο χώρο λειτουργίας της αεροτομής στο σχήμα 8.16(β′). Οι αρχικές συνθήκες των υπολογισμών είναι: $\dot{\alpha}=1,\ \dot{h}=0$.

8.8 Αεροελαστική ανάλυση αεροτομής τριών βαθμών ελευθερίας

Στην ενότητα αυτή γίνεται η αεροελαστική ανάλυση τομής πτέρυγας αεροσκάφους τριών βαθμών ελευθερίας. Η κίνηση της αεροτομής περιορίζεται από ένα γραμμικό, κατά την κατακόρυφη κατεύθυνση, και ένα στρεπτικό ελατήριο κατά την περιστροφική ως προς τον ελαστικό άξονα κατεύθυνση. Το τμήμα εκφυγής της αεροτομής αντιμετωπίζεται ως πτερύγιο ελέγχου που δύναται να στρέφεται ως προς συγκεκριμένο άξονα (hinge axis). Η κίνηση του πτερυγίου ελέγχου περιορίζεται από αντίστοιχο στρεπτικό ελατήριο. Πρόκειται δηλαδή για ένα ελαστικό μοντέλο τριών βαθμών ελευθερίας. Το μήκος του πτερυγίου ελέγχου είναι ίσο με $0.25c$.

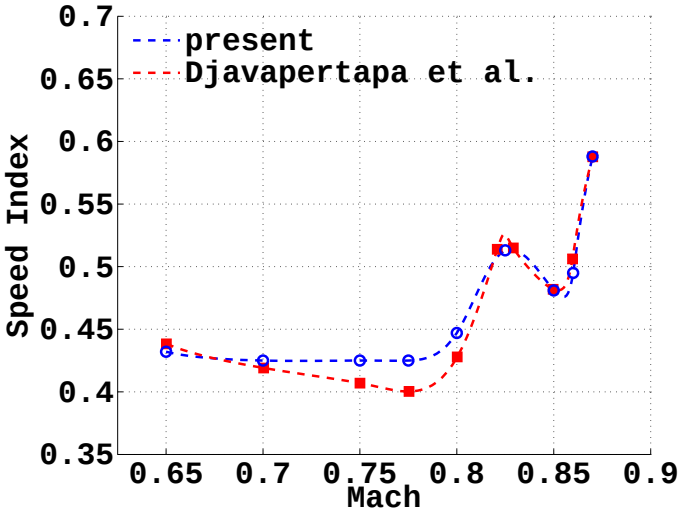
Τα αδιάστατα δομικά χαρακτηριστικά της αεροτομής και ο λόγος των πυκνοτήτων του ρευστού προς του υλικού κατασκευής συνοψίζονται στον πίνακα 8.3. Η αεροτομή που μελετήθηκε είναι η NACA64A010, όπως και στην προηγούμενη ενότητα.

Όπως και στην προηγούμενη ενότητα, η αεροτομή αρχικά εκτελεί εξαναγκασμένη ταλάντωση με πλάτος ίσο με 1^o γύρω από τον ελαστικό άξονά της και διάρκεια ίση με $\frac{2\pi}{\omega_\alpha}$ και, στη συνέχεια, αφήνεται ελεύθερη να κινηθεί σύμφωνα με τα προσδεδεμένα σ΄ αυτή ελατήρια. Η απόκριση της αεροτομής εξαρτάται από τις τιμές του αριθμού Mach και της

Τύπος αεροτομής	NACA64A010
Θέση ελαστικού άξονα	$x_{ea} = 0.4$
Κέντρο μάζας της αεροτομής	$X_\alpha = 0.1$
Λόγος φυσικών συχνοτήτων	$\frac{\omega_h}{\omega_\alpha} = 0.3$
Ακτίνα περιστροφής (radius of gyration) της αεροτομής	$r_\alpha = 0.5$
Λόγος πυκνοτήτων	$\mu \triangleq \frac{m}{\pi\rho_\infty b^2} = 23.48$
Θέση άξονα πτερυγίου ελέγχου (hinge axis)	$x_{ha} = 0.75$
Κέντρο μάζας του πτερυγίου	$X_\beta = 0.004$
Λόγος φυσικών συχνοτήτων	$\frac{\omega_\beta}{\omega_\alpha} = 1.5$
Ακτίνα περιστροφής (radius of gyration) του πτερυγίου ελέγχου	$r_\beta = 0.06$

Πίνακας 8.3: Χαρακτηριστικά του αεροελαστικού μοντέλου.

ανηγμένης ταχύτητας της επ΄ άπειρο ροής. Στη μελέτη που πραγματοποιήθηκε ο αριθμό Mach της ροής μεταβάλλεται από 0.65 έως 0.87 και η τιμή της ανηγμένης ταχύτητας από 0.35 έως 0.65. Η υπολογισθείσα κρίσιμη καμπύλη φαίνεται στο σχήμα 8.18. Η σύγκριση με αριθμητικά αποτελέσματα άλλης ερευνητικής ομάδας, [208], που παρουσιάζονται στο ίδιο σχήμα, είναι ικανοποιητική. Διαφορές υπάρχουν στην περιοχή από $M_\infty = 0.75$ έως $M_\infty = 0.80$, όπου καταγράφεται και η ελάχιστη τιμή της κρίσιμης ανηγμένης ταχύτητας της ροής. Η ταύτιση στις υπόλοιπες περιοχές είναι εξαιρετική. Η έλλειψη πειραματικών αποτελεσμάτων δεν επιτρέπει περαιτέρω κρίσεις για τα αποτελέσματα των δύο συγκρινόμενων υπολογισμών.



Σχήμα 8.18: Αεροελαστική ανάλυση αεροτομής NACA64A010 τριών βαθμών ελευθερίας. Υπολογισθείσα κρίσιμη καμπύλη και σύγκριση με αριθμητικά αποτελέσματα άλλης ερευνητικής ομάδας, [208]. Τα αδιάστατα δομικά χαρακτηριστικά της αεροτομής και ο λόγος των πυκνοτήτων του ρευστού προς του υλικού κατασκευής φαίνονται στον πίνακα 8.3.

8.9 Διεύρυνση της περιοχής ευσταθούς λειτουργίας αεροτομής

Εδώ μελετάται η διεύρυνση της περιοχής ευσταθούς λειτουργίας της αεροτομής της προηγούμενης ενότητας μέσω αυτόματου ελέγχου της κίνησης του πτερυγίου ελέγχου.

Προτιμήθηκε ένας απλός νόμος ελέγχου, που δίνεται από τη σχέση

$$\beta_c = G_{\dot{\alpha}}\dot{\alpha} + G_{\dot{h}}\dot{h} \tag{8.36}$$

όπου β_c είναι η διόρθωση στη γωνία του πτερυγίου ελέγχου που επιβάλλει ο έλεγχος και $G_{\dot{\alpha}}$, $G_{\dot{h}}$ είναι τα κέρδη ανάδρασης της γωνιακής ως προς τον ελαστικό άξονα και της κατακόρυφης ταχύτητας της αεροτομής.

Συνεπώς, αν β είναι η γωνία του πτερυγίου τη χρονική στιγμή t, ο έλεγχος τείνει να στρέψει το πτερύγιο κατά γωνία β_c ώστε η τελική γωνία του πτερυγίου να είναι ίση με $\beta+\beta_c$. Επειδή η στροφή του πτερυγίου στην πράξη δεν γίνεται ακαριαία αλλά απαιτεί ένα χρονικό διάστημα, εισάγεται στις ελαστικές εξισώσεις της αεροτομής η ισοδύναμη ροπή που χρειάζεται για να γίνει η στροφή του πτερυγίου κατά γωνία β_c. Δηλαδή, οι ελαστικές εξισώσεις της αεροτομής παίρνουν τη μορφή

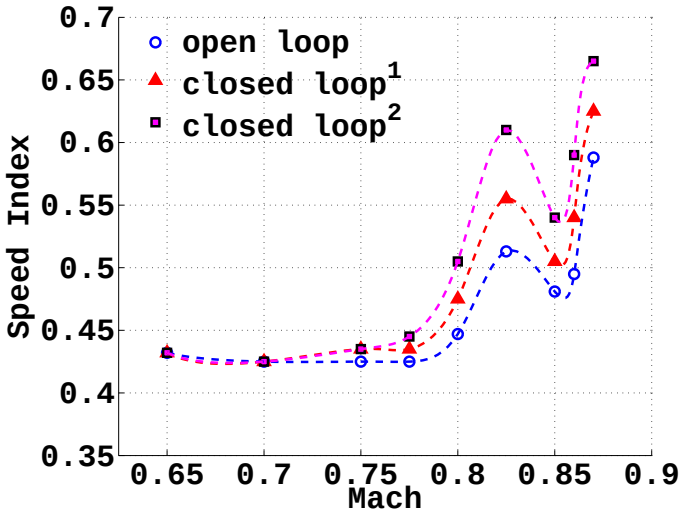
$$M\,\overrightarrow{\ddot{q}} + K\,\overrightarrow{q} = \overrightarrow{Q} + \overrightarrow{Q}_c \tag{8.37}$$

όπου

$$\overrightarrow{Q}_c = \begin{bmatrix} 0 & 0 & 0 & M_c \end{bmatrix}^T \tag{8.38}$$

και $M_c = K_\beta \beta_c$ είναι η ισοδύναμη ροπή που χρειάζεται για να στραφεί το πτερύγιο κατά γωνία β_c. Με τον τρόπο αυτό μοντελοποιείται η δράση του αυτόματου ελέγχου. Δηλαδή ο αυτόματος έλεγχος δεν ρυθμίζει αυτόματα τη γωνία στροφής του πτερυγίου ελέγχου, αλλά έμμεσα επιβάλλοντας μία ισοδύναμη ροπή, η οποία προστίθεται στο δεξιό μέλος των ελαστικών εξισώσεων της αεροτομής.

Στο σχήμα 8.19 φαίνονται οι κρίσιμες καμπύλες της αεροτομής οι οποίες υπολογίσθηκαν με και χωρίς έλεγχο. Δοκιμάστηκαν δύο διαφορετικά κέρδη ανάδρασης $G_{\dot{\alpha}} = G_{\dot{h}} = 1.$ και $G_{\dot{\alpha}} = G_{\dot{h}} = 2.$ Η θετική επίδραση του αυτόματου ελέγχου είναι εμφανής. Φαίνεται ότι όσο μεγαλύτερο είναι το κέρδος ανάδρασης τόσο περισσότερο διευρύνεται η περιοχή ευσταθούς λειτουργίας.

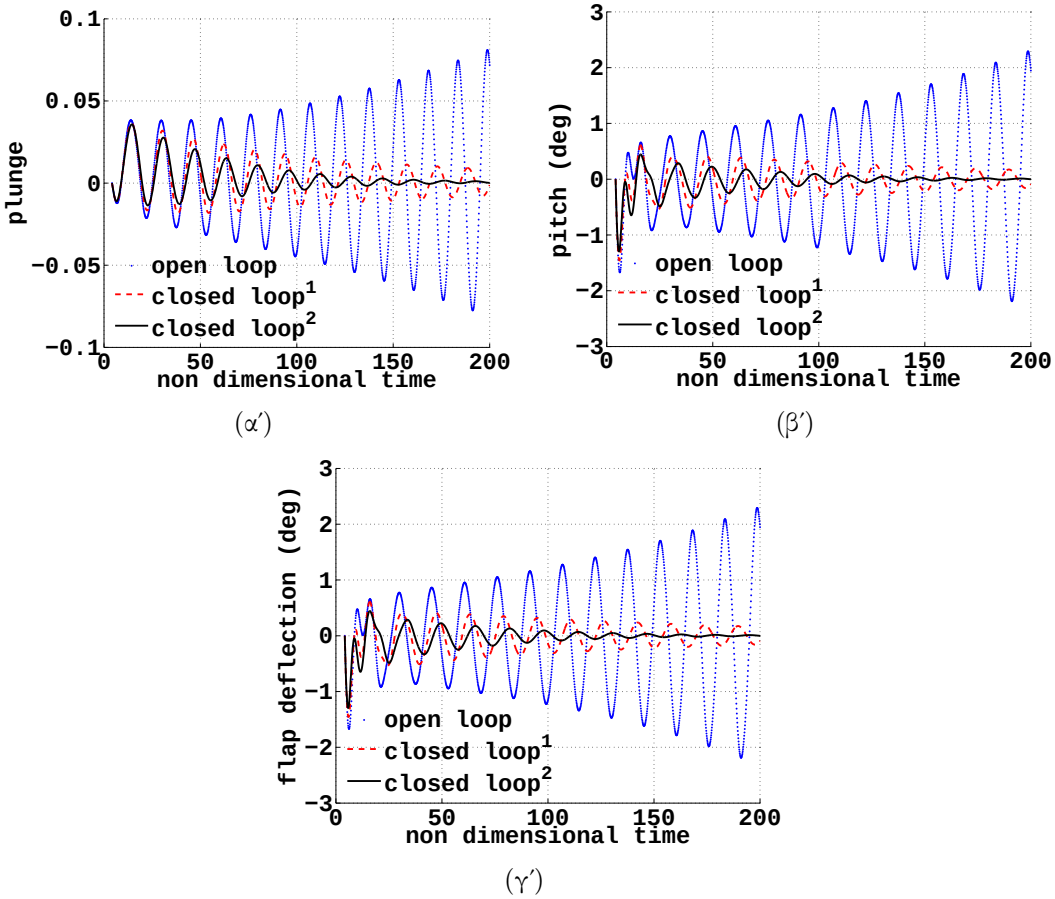


Σχήμα 8.19: Αύξηση της περιοχής ευσταθούς λειτουργίας αεροτομής NACA64A010 τριών βαθμών ελευθερίας με ανάδραση των μεταβλητών κατάστασης της αεροτομής. Κρίσιμη καμπύλη (α) χωρίς έλεγχο (open loop), (β) με έλεγχο και $G_{\dot{\alpha}}=G_{\dot{h}}=1.$ (closed loop^1), (γ) με έλεγχο και $G_{\dot{\alpha}}=G_{\dot{h}}=2.$ (closed loop^2).

Τα σχήματα 8.20 δείχνουν την επίδραση του αυτόματου ελέγχου σε μία περίπτωση ασταθούς λειτουργίας.

8.10 Κέρδος από τη χρήση GPUs

Για την αεροελαστική ανάλυση των αεροτομών δύο ή τριών βαθμών ελευθερίας που παρουσιάστηκαν στις προηγούμενες ενότητες χρησιμοποιήθηκαν GPUs τελευταίας αρχιτεκτονικής. Η επιτάχυνση από τη χρήση μίας GPU αντί ενός πυρήνα μίας σημερινής CPU είναι περίπου 40x. Η χάραξη των κρίσιμων καμπυλών που παρουσιάστηκαν νωρίτερα, απαιτεί περισσότερους από 100 χρονικά μη-μόνιμους υπολογισμούς. Ο χρόνος επίλυσης ενός τέτοιου προβλήματος ροής σε μία Tesla M2050 είναι περίπου 45 λεπτά. Λαμβάνοντας υπόψη το μεγάλο πλήθος υπολογισμών που απαιτήθηκαν και την επιτάχυνση της πρόλεξης των αριθμητικών αποτελεσμάτων από τη χρήση GPUs, γίνε-



Σχήμα 8.20: Αύξηση της περιοχής ευσταθούς λειτουργίας αεροτομής NACA64A010 τριών βαθμών ελευθερίας με ανάδραση των μεταβλητών κατάστασης της αεροτομής. Απόκριση αεροτομής (α) χωρίς έλεγχο (open loop), (β) με έλεγχο και $G_{\dot{\alpha}} = G_{\dot{h}} = 1$. (closed loop^1), (γ) με έλεγχο και $G_{\dot{\alpha}} = G_{\dot{h}} = 2$. (closed loop^2). $M_\infty = 0.80$, $U^* = 0.46$

ται εύκολα αντιληπτό το αρκετά υψηλό υπολογιστικό κόστος που θα απαιτούσε η ίδια εφαρμογή στην περίπτωση που χρησιμοποιούνταν μόνο CPUs.

Κεφάλαιο 9

Ανάλυση μοντέλου αεροσκάφους σε GPUs

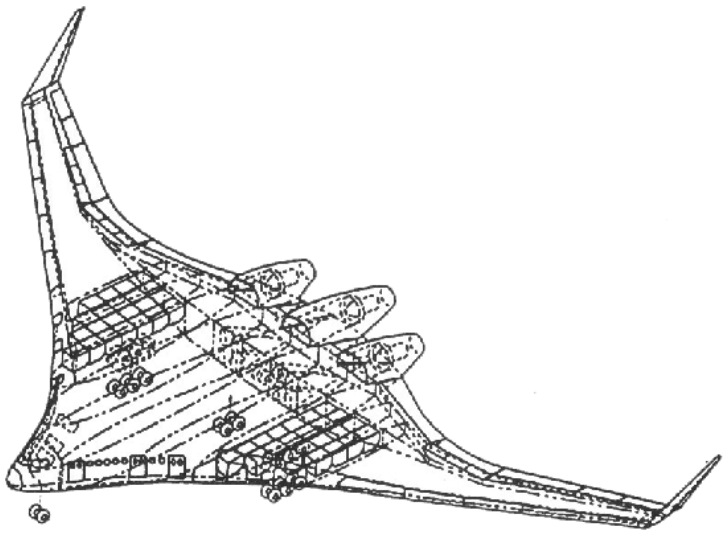
Το κεφάλαιο αυτό ασχολείται με την αεροδυναμική ανάλυση επιβατικού αεροσκάφους τύπου Blended Wing Body (BWB), 450 θέσεων. Για την ανάλυση χρησιμοποιήθηκε η παράλληλη εκδοχή του, για πολλές κάρτες γραφικών του ίδιου υπολογιστικού κόμβου, GPU-επιλύτη που χρησιμοποιεί αριθμητική μικτής ακρίβειας.

Τα αεροσκάφη τύπου BWB είναι μη-συμβατικά μοντέλα αεροσκαφών όπου η άτρακτος και οι πτέρυγες αποτελούν, κατασκευαστικά, ένα μοναδικό συνεχόμενο τμήμα παρά ένωση διακριτών επιμέρους τμημάτων. Κατά συνέπεια, η μετάβαση από την άτρακτο στις πτέρυγες γίνεται γεωμετρικά πιο ομαλά σε σχέση με τα συμβατικά αεροσκάφη. Τα αεροσκάφη αυτού του τύπου εμφανίζουν πολλά αεροδυναμικά προτερήματα, όπως υψηλότερο λόγο άνωσης προς οπισθέλκουσα, χαμηλότερη κατανάλωση καυσίμου ανά επιβάτη και μίλι, χαμηλότερη εκπομπή θορύβου και χαμηλότερο βάρος απογείωσης, σε σχέση με συμβατικά αεροσκάφη αντίστοιχων προδιαγραφών. Τα αεροδυναμικά προτερήματα των αεροσκαφών τύπου BWB έναντι συμβατικών αεροσκαφών αντίστοιχων προδιαγραφών παρουσιάζονται στην αρχή του κεφαλαίου στη μορφή σύντομης βιβλιογραφικής επισκόπησης. Το κεφάλαιο κλείνει με την παρουσίαση της αεροδυναμικής ανάλυσης του αεροσκάφους που πραγματοποιήθηκε χρησιμοποιώντας τον GPU-επιλύτη της διατριβής. Οι περιπτώσεις ροής που μελετήθηκαν περιλαμβάνουν χρονικά μόνιμες και μη-μόνιμες ροές γύρω από το αεροσκάφος. Στις χρονικά μη-μόνιμες περιπτώσεις, μία προεπιλεγμένη επιφάνεια ελέγχου ή ολόκληρο το αεροσκάφος εξαναγκάζεται σε περιοδική κίνηση/παραμόρφωση.

9.1 Μοντέλο αεροσκάφους τύπου Blended Wing Body

Τα επιβατικά αεροσκάφη τύπου Blended Wing Body (BWB), [239, 240, 241, 242, 243], είναι μη-συμβατικά, αεροδυναμικά πολλά υποσχόμενα μοντέλα, που μπορεί να αποτελέσουν το μέλλον της αεροπορικής βιομηχανίας. Δεν έχουν ουραίο τμήμα και η μετάβαση από την κύρια άτρακτο στις πτέρυγες είναι γεωμετρικά πιο ομαλή σε σχέση με τα συμβατικά αεροσκάφη, σχήμα 9.1. Επιπλέον, η διαμήκης τομή της ατράκτου έχει

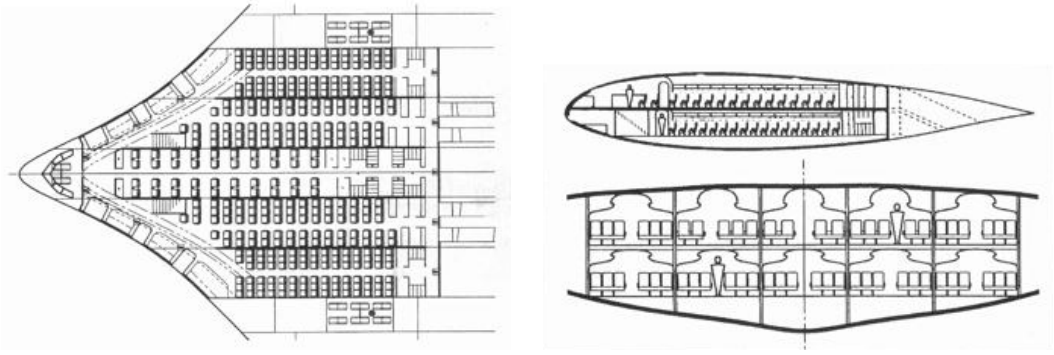
σχήμα αεροτομής. Το σχήμα της διαμήκους τομής της ατράκτου, σε συνδυασμό με την πλάτυνση αυτής, κάνει πιο ενεργό το ρόλο της ατράκτου στη δημιουργία της άνωσης, χωρίς όμως να αυξάνεται η συνολική βρεχόμενη επιφάνεια σε σχέση με συμβατικά αεροσκάφη ίδιου όγκου.



Σχήμα 9.1: Αεροσκάφος τύπου Blended Wing Body (BWB), [244].

Μία πρώτη μελέτη που έγινε στην εργασία [239] έδειξε ότι ένα αεροσκάφος τύπου BWB, 800 θέσεων με Mach πτήσης 0.85 και εμβέλεια 7000 ναυτικών μιλίων, έχει μεγαλύτερο λόγο άνωσης προς οπισθέλκουσα κατά περίπου 40% και περίπου 25% χαμηλότερη κατανάλωση καυσίμου ανά επιβάτη και ναυτικό μίλι σε σχέση με συμβατικό αεροσκάφος αντίστοιχων προδιαγραφών. Σε συνέχεια της προηγούμενης εργασίας, η εργασία [240] υπολόγισε 15% χαμηλότερο βάρος απογείωσης για ένα αεροσκάφος τύπου BWB σε σχέση με ένα συμβατικό αεροσκάφος ίδιας εμβέλειας χρησιμοποιώντας κινητήρες ίδιας ισχύος. Επιπλέον, σύμφωνα με την εργασία [245], αεροσκάφη τύπου BWB παράγουν λιγότερο θόρυβο κατά τη διάρκεια της πτήσης. Στην εργασία [246] επιτεύχθηκε η αύξηση της ελεγξιμότητας και η επιπλέον μείωση του παραγόμενου θορύβου χρησιμοποιώντας διάταξη πολλαπλών συστημάτων πρόωσης μικρής κλίμακας με υψηλό λόγο παράκαμψης (by-pass ratio). Στις εργασίες [247, 248] πραγματοποιήθηκε ο σχεδιασμός των δομικών χαρακτηριστικών ενός αεροσκάφους τύπου BWB με στόχο την ελαχιστοποίηση του βάρους απογείωσης. Αντικείμενο της εργασίας [249] αποτέλεσε η εύρεση της βέλτιστης θέσης του κέντρου βάρους αεροσκάφους τύπου BWB, κατανέμοντας κατάλληλα τα καύσιμά του, με στόχο την αύξηση της απόδοσης του αεροσκάφους κατά τη διάρκεια ευθείας πτήσης.

Η NASA, σε συνεργασία με τη Boeing, έχει ήδη κατασκευάσει πρότυπο αεροσκάφους τύπου BWB σε μικρή κλίμακα. Το πρότυπο αυτό είχε, αρχικά, δοκιμαστεί σε αεροδυναμική σήραγγα ερευνητικού κέντρου της NASA και, πλέον, δοκιμάζεται και σε πραγματικές πτήσεις, [244]. Προσχέδιο του χώρου επιβατών φαίνεται στα σχήματα 9.2(α′), 9.2(β′).



(α′) (β′)

Σχήμα 9.2: Καμπίνα αεροσκάφους τύπου BWB σε μοντέλο που σχεδίασε η NASA σε συνεργασία με τη Boeing, [244].

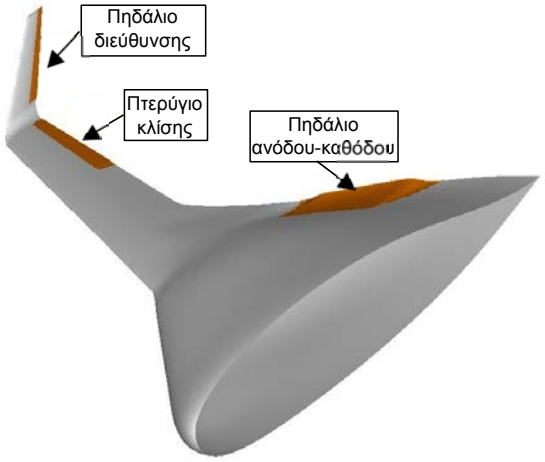
9.2 Αεροδυναμική ανάλυση του αεροσκάφους

Στις επόμενες ενότητες παρουσιάζεται η μελέτη της δυναμικής επίδρασης της εξαναγκασμένης κίνησης-ταλάντωσης των επιφανειών ελέγχου του αεροσκάφους και της εξαναγκασμένης παραμόρφωσης όλης της κατασκευής, στη ροή γύρω από το αεροσκάφος. Αρχικά μελετήθηκε η ροή γύρω από το αεροσκάφος για διαφορετικές συνθήκες ευθείας πτήσης (διαφορετικοί αριθμοί Mach και γωνίες της επ΄ άπειρο ροής) με ή χωρίς εκτροπή των επιφανειών ελέγχου. Στη συνέχεια, πραγματοποιήθηκε μία σειρά αεροδυναμικών υπολογισμών, για διαφορετικές συνθήκες ροής και συχνότητες ή πλάτη ταλάντωσης, είτε συγκεκριμένων επιφανειών ελέγχου είτε όλης της κατασκευής. Η ροή γύρω από το αεροσκάφος θεωρείται ατριβής.

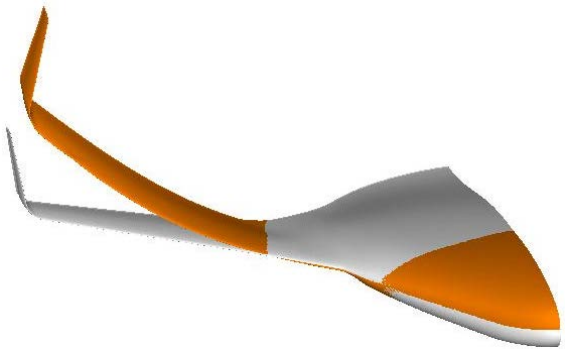
Οι επιφάνειες ελέγχου (στο χείλος εκφυγής) που μελετήθηκαν ως προς τη δυναμική επίδραση της περιοδικής κίνησής τους στη ροή γύρω από το αεροσκάφος, φαίνονται στο σχήμα 9.3. Πρόκειται για το πηδάλιο ανόδου-καθόδου (elevator), το πτερύγιο κλίσης (aileron) και το πηδάλιο διεύθυνσης (rudder). Ως θετική ορίζεται η προς τα κάτω εκτροπή των 'οριζόντιων' επιφανειών ελέγχου (πηδάλιο ανόδου-καθόδου και πτερύγιο κλίσης) και η προς την άτρακτο εκτροπή της κατακόρυφης επιφάνειας ελέγχου (πηδαλίου διεύθυνσης).

Στις περιπτώσεις περιοδικής παραμόρφωσης ολόκληρου του αεροσκάφους, η παραμόρφωση αυτού ακολουθούσε την πρώτη καμπτική ιδιομορφή του αεροσκάφους. Για τον υπολογισμό των ελαστικών ιδιομορφών της κατασκευής χρησιμοποιήθηκε το εμπορικό πρόγραμμα NASTRAN. Το σχήμα 9.4 δείχνει τη γεωμετρία της πρώτης καμπτικής ιδιομορφής. Για να γίνει καλύτερα αντιληπτό το αποτέλεσμα, η παραμόρφωση έχει μεγεθυνθεί 100 φορές.

Οι αριθμοί Mach και οι γωνίες της επ΄ άπειρο ροής των αεροδυναμικών υπολογισμών που πραγματοποιήθηκαν είναι: $M_\infty = 0.75$ ή 0.85 και $\alpha_\infty = 1.7^o$ ή 3.0^o. Η ανηγμένη συχνότητα ταλάντωσης ($\kappa \triangleq \frac{\pi f \cdot MAC}{U_\infty}$, όπου $MAC = 26.468m$ η μέση αεροδυναμική χορδή του αεροσκάφους, U_∞ η ταχύτητα πτήσης και f η συχνότητα ταλάντωσης) των επιφανειών ελέγχου ή ολόκληρης της κατασκευής είναι μεταξύ 0.01 και 5.0. Το πλάτος



Σχήμα 9.3: Ορισμός των επιφανειών ελέγχου της πορείας του αεροσκάφους που μελετήθηκαν χρησιμοποιώντας το GPU-επιλύτη της διατριβής σε συστοιχία καρτών γραφικών.



Σχήμα 9.4: Πρώτη καμπτική ιδιομορφή του αεροσκάφους.

της ταλάντωσης των επιφανειών ελέγχου είναι από 3.0^o έως 5.0^o.

Στους χρονικά μη-μόνιμους αεροδυναμικούς υπολογισμούς, το υπολογιστικό πλέγμα κάθε χρονικής στιγμής προκύπτει από την προσαρμογή του υπολογιστικού πλέγματος της προηγούμενης χρονικής στιγμής στη νέα/στιγμιαία γεωμετρία του αεροσκάφους. Η γεωμετρία του αεροσκάφους, όπως έχει αναφερθεί, ορίζεται από την εκτροπή των επιφανειών ελέγχου ή ολόκληρης της κατασκευής με βάση προεπιλεγμένη ιδιομορφή. Η προσαρμογή του υπολογιστικού πλέγματος έγινε χρησιμοποιώντας μία μέθοδο γραμμικών στρεπτικών ελατηρίων, σύμφωνα με την οποία το υπολογιστικό πλέγμα θεωρείται ως ένα δικτύωμα με τις ακμές του πλέγματος να αποτελούν τα δομικά στοιχεία της θεωρητικής αυτής κατασκευής. Σε κάθε κόμβο του πλέγματος αντιστοιχεί ένα γραμμικό στρεπτικό ελατήριο, η σταθερά δυσκαμψίας του οποίου εξαρτάται από γεωμετρικά χαρακτηριστικά. Η μετατόπιση/παραμόρφωση της επιφάνειας του εξεταζόμενου αεροδυναμικού σώματος εκτρέπει τη θεωρητική αυτή κατασκευή από τη θέση ελαστικής ισορροπίας. Επιλύοντας τις ανά κόμβο εξισώσεις ισορροπίας προκύπτει το τελικό προσαρμοσμένο πλέγμα. Η επίλυση των εξισώσεων ισορροπίας γίνεται

επαναληπτικά. Για την επαναληπτική αυτή διαδικασία, η αρχική παραμόρφωση δίνεται από μία αλγεβρική συνάρτηση που σχετίζει τη μετατόπιση των κόμβων του πλέγματος με τη μετατόπιση των πλησιέστερων κόμβων της επιφάνειας του αεροσκάφους, όπως αναλύεται στην παράγραφο 8.2. Κατά την εκτροπή τους, οι επιφάνειες ελέγχου παραμένουν συνδεδεμένες στο κύριο σώμα του αεροσκάφους, χωρίς σχηματισμό διακένων. Έτσι δεν χρειάζεται η προσθήκη επιπλέον κόμβων στο διάκενο που θα σχηματίζονταν ανάμεσα στην επιφάνεια ελέγχου και το κύριο σώμα του αεροσκάφους.

Συνολικά πραγματοποιήθηκαν πάνω από 40 χρονικά μη-μόνιμες αεροδυναμικές προσομοιώσεις. Το υπολογιστικό πλέγμα που χρησιμοποιήθηκε αποτελείται από περίπου 1400000 κόμβους. Η μείωση του χρόνου αναμονής, από τη χρήση του GPU-κώδικα συγκριτικά με τη χρήση του αντίστοιχου CPU-κώδικα σε συστοιχία διασυνδεδεμένων υπολογιστών, είναι αρκετά σημαντική ιδίως στην επίλυση των χρονικά μη-μόνιμων προβλημάτων. Συγκεκριμένα, ο χρόνος πρόλεξης των χρονικά μόνιμων και μη-μόνιμων ροών που μελετήθηκαν, μειώθηκε περίπου 6 φορές χρησιμοποιώντας τρεις GPUs τελευταίας αρχιτεκτονικής του ίδιου υπολογιστικού κόμβου, σε σχέση με 64 CPU-πυρήνες διασυνδεδεμένων υπολογιστών. Η επιτάχυνση αυτή μείωσε το χρόνο επίλυσης των χρονικά μόνιμων προβλημάτων από περίπου 40 λεπτά ανά περίπτωση ροής (σε 64 CPU-πυρήνες διασυνδεδεμένων υπολογιστών) σε περίπου 6.5 λεπτά (σε τρεις GPUs του ίδιου υπολογιστικού κόμβου). Ο χρόνος επίλυσης των χρονικά μη-μόνιμων προβλημάτων εξαρτάται από τη συχνότητα και το πλάτος ταλάντωσης της επιφάνειας ελέγχου ή της κατασκευής. Περίπου χρειάστηκαν τρεις ημέρες για την επίλυση ενός τέτοιου προβλήματος σε 64 CPU-πυρήνες διασυνδεδεμένων υπολογιστών. Είναι σημαντικό ότι, ο χρόνος επίλυσης μειώνεται σε μόλις 12 ώρες χρησιμοποιώντας τρεις κάρτες γραφικών του ίδιου υπολογιστικού κόμβου και τη μικτής ακρίβειας εκδοχή του GPU-κώδικα.

Η αεροδυναμική ανάλυση πραγματοποιήθηκε στο πλαίσιο του ερευνητικού προγράμματος με τίτλο Active Control for Flexible 2020 Aircraft[1] (ACFA 2020), [250], που χρηματοδοτήθηκε από την Ευρωπαϊκή Ένωση. Επιλέχθηκε να χρησιμοποιηθεί ο παράλληλος σε συστοιχίες καρτών γραφικών GPU-κώδικας της διατριβής, αντί του παράλληλου σε συστοιχίες διασυνδεδεμένων υπολογιστών CPU-κώδικα, λόγω της σημαντικής μείωσης του χρόνου επίλυσης από τη χρήση GPUs.

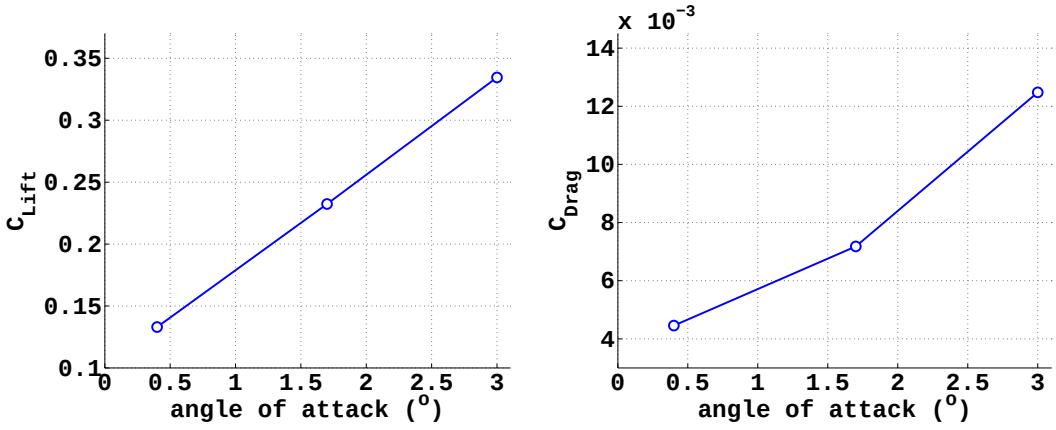
Οι ενότητες 9.3, 9.4, 9.5 παρουσιάζουν ορισμένες ενδεικτικές χρονικά μόνιμες και μη-μόνιμες αριθμητικές προλέξεις.

9.3 Πρόλεξη χρονικά μόνιμων ροών

Τα σχήματα 9.5, 9.6 δείχνουν τις πολικές καμπύλες του αεροσκάφους για αριθμούς Mach $M_\infty = 0.75$ και $M_\infty = 0.85$ αντίστοιχα. Η σύγκριση με αριθμητικά αποτελέσματα άλλων ερευνητικών ομάδων που συμμετείχαν στο ίδιο έργο, φαίνεται στα σχήματα 9.6(α′), 9.6(β′).

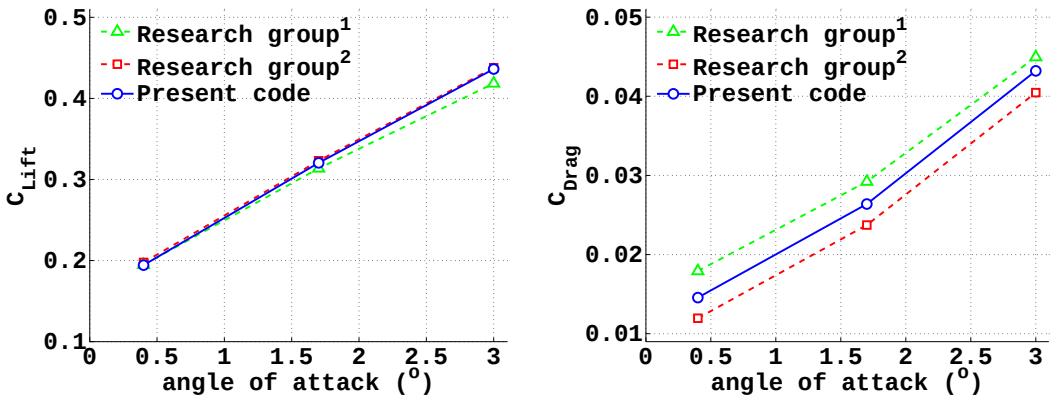
Η κατανομή του αριθμού Mach πάνω στην επιφάνεια του αεροσκάφους, για διαφορετικές συνθήκες ευθείας πτήσης, φαίνεται στο σχήμα 9.7. Το κύμα κρούσης που εμφανίζεται στις περιπτώσεις με αριθμό Mach $M_\infty = 0.85$ είναι αρκετά ισχυρό. Τα

[1] Αριθμός συμβολαίου: ACP7-GA-2008-213321



(α′) (β′)

Σχήμα 9.5: Υπολογισθείσα πολική καμπύλη του αεροσκάφους για $M_\infty = 0.75$.

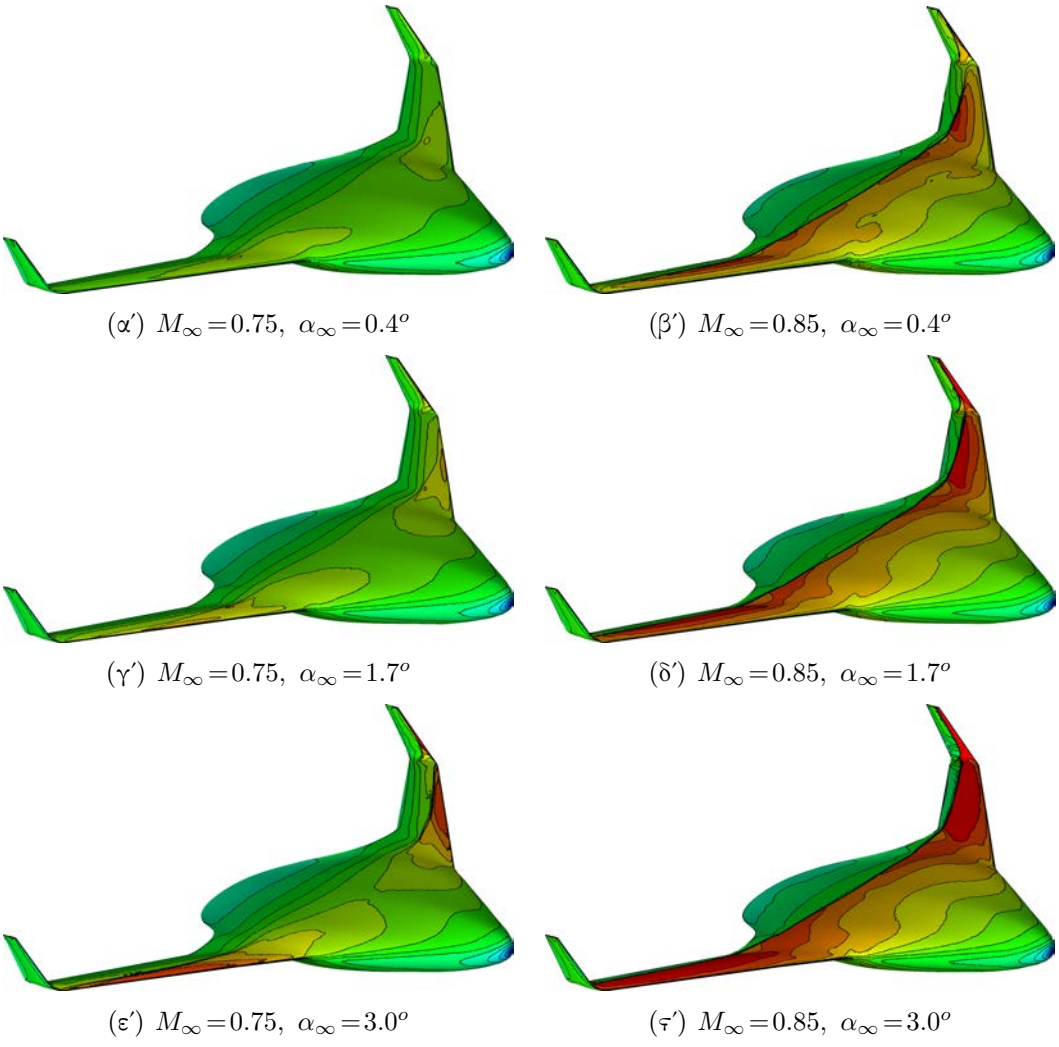


(α′) (β′)

Σχήμα 9.6: Υπολογισθείσα πολική καμπύλη του αεροσκάφους για $M_\infty = 0.85$. Σύγκριση με αριθμητικά αποτελέσματα άλλων ερευνητικών ομάδων που συμμετείχαν στο ίδιο έργο.

σχήματα 9.8(α′), 9.8(β′) δείχνουν την έκταση της περιοχής υπερηχητικής ροής γύρω από το αεροσκάφος για γωνία επ΄ άπειρο ροής $\alpha_\infty = 1.7^o$ και αριθμό Mach $M_\infty = 0.75$ ή $M_\infty = 0.85$. Παρατηρείται μεγάλη αύξηση της υπερηχητικής περιοχής για $M_\infty = 0.85$, συγκριτικά με τους υπολογισμούς για $M_\infty = 0.75$. Μάλιστα, στην περίπτωση που $M_\infty = 0.85$, εμφανίζεται υπερηχητική ροή και στην πλευρά υπερπίεσης.

Η κατανομή του συντελεστή πίεσης κατά το μήκος των εγκάρσιων τομών του σχήματος 9.9, φαίνονται στα σχήματα 9.10, 9.11 για $\alpha_\infty = 1.7^o$ και $M_\infty = 0.75$ ή $M_\infty = 0.85$, αντίστοιχα. Στο σχήμα 9.11 συγκρίνονται οι αριθμητικές προλέξεις του GPU-κώδικα της διατριβής με αριθμητικές προλέξεις του CPU-κώδικα άλλης ερευνητικής ομάδας. Παρατηρείται πολύ καλή σύμπτωση των καμπυλών, με μία πολύ μικρή μετατόπιση της θέσης αλλά και της έντασης του κρουστικού κύματος.



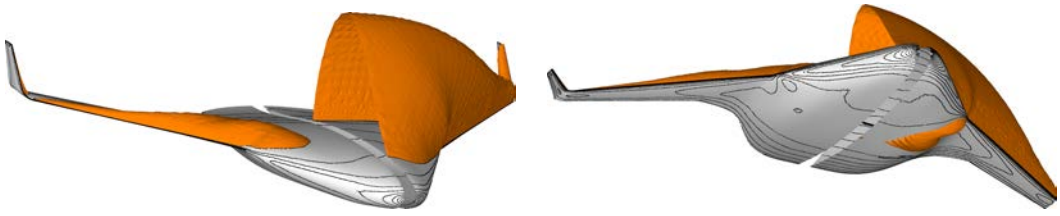
(α′) $M_\infty = 0.75,\ \alpha_\infty = 0.4^o$ (β′) $M_\infty = 0.85,\ \alpha_\infty = 0.4^o$

(γ′) $M_\infty = 0.75,\ \alpha_\infty = 1.7^o$ (δ′) $M_\infty = 0.85,\ \alpha_\infty = 1.7^o$

(ε′) $M_\infty = 0.75,\ \alpha_\infty = 3.0^o$ (ϛ′) $M_\infty = 0.85,\ \alpha_\infty = 3.0^o$

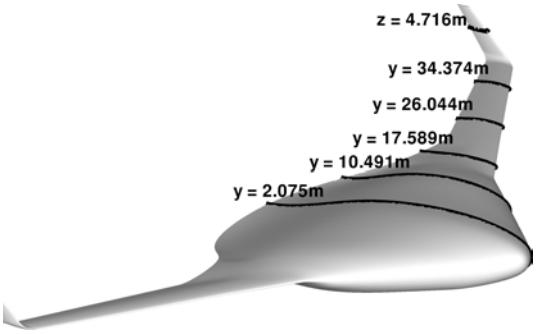
(ζ′) Χρωματική παλέτα

Σχήμα 9.7: Χρονικά μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους για διαφορετικές συνθήκες ροής. Η χρωματική παλέτα (ζ′) είναι κοινή σε όλα τα πεδίου ροής.

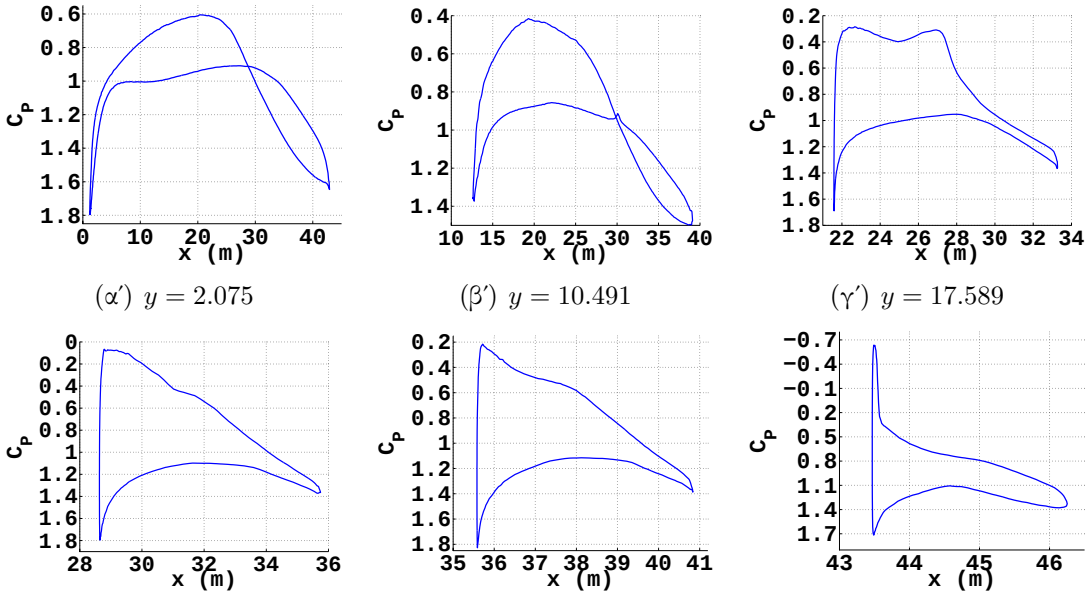


(α′) Πλευρά υποπίεσης (β′) Πλευρά υπερπίεσης

Σχήμα 9.8: Χρονικά μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Περιοχή υπερηχητικής ροής γύρω από το αεροσκάφος για $\alpha_\infty = 1.7^o$ και $M_\infty = 0.75$ (στο δεξιό ως προς τον πιλότο τμήμα του αεροσκάφους) ή $M_\infty = 0.85$ (στο αριστερό ως προς τον πιλότο τμήμα του αεροσκάφους).



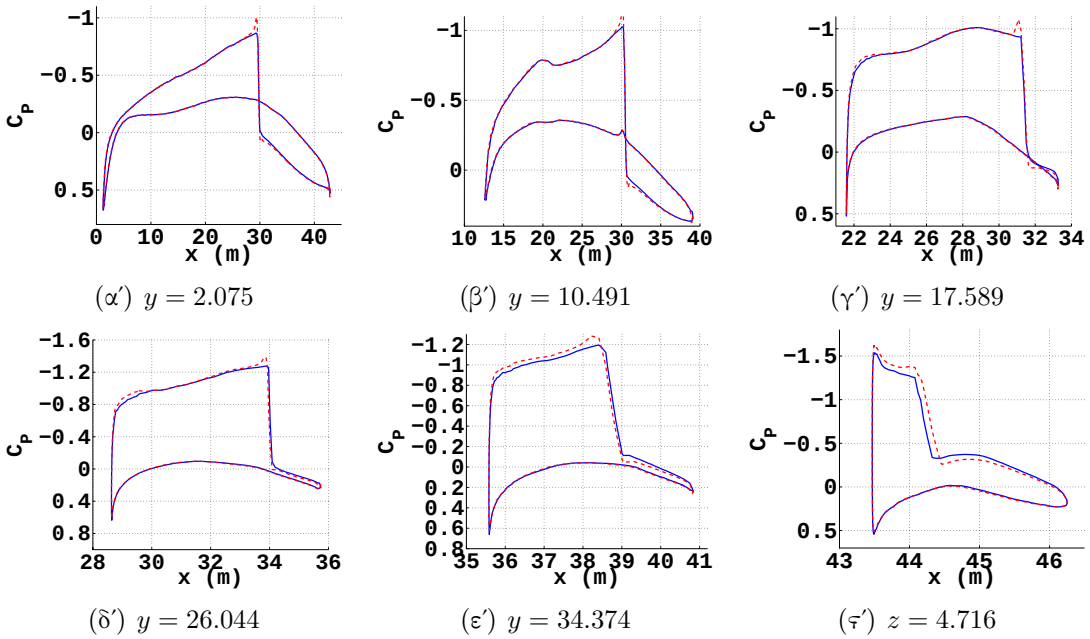
Σχήμα 9.9: Θέσεις υπολογισμού της κατανομής του συντελεστή πίεσης.



(α′) $y = 2.075$ (β′) $y = 10.491$ (γ′) $y = 17.589$

(δ′) $y = 26.044$ (ε′) $y = 34.374$ (ϛ′) $z = 4.716$

Σχήμα 9.10: Χρονικά μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Κατανομή του συντελεστή πίεσης στις τομές της πτέρυγας του αεροσκάφους του σχήματος 9.9 ($M_\infty = 0.75,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$).



(α′) $y = 2.075$ (β′) $y = 10.491$ (γ′) $y = 17.589$

(δ′) $y = 26.044$ (ε′) $y = 34.374$ (ϛ′) $z = 4.716$

Σχήμα 9.11: Χρονικά μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Κατανομή του συντελεστή πίεσης στις τομές της πτέρυγας του αεροσκάφους του σχήματος 9.9 ($M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$). Σύγκριση με αριθμητικές προλέξεις συνεργαζόμενης ερευνητικής ομάδας (διακεκομμένη γραμμή).

9.4 Υπολογισμός της δυναμικής επίδρασης των επιφανειών ελέγχου στη ροή

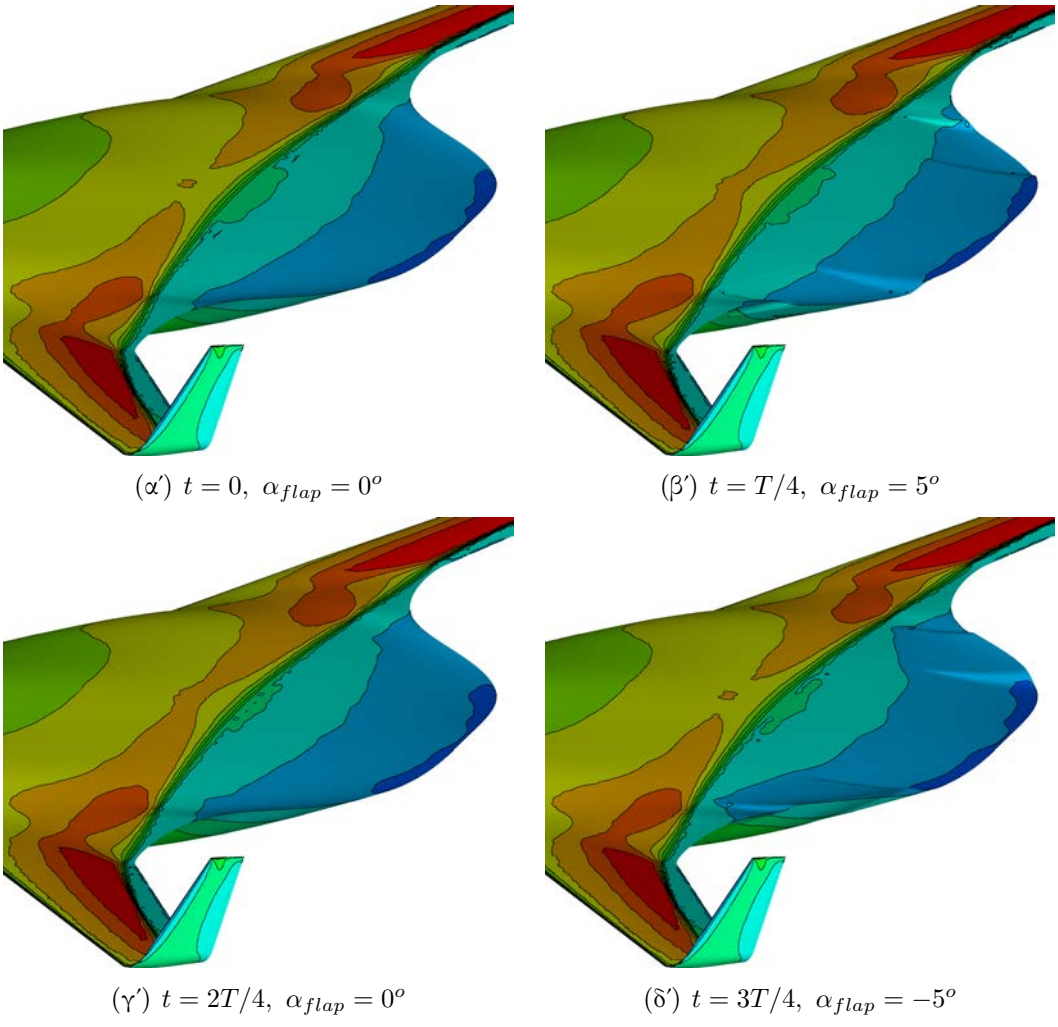
9.4.1 Επίδραση του πηδαλίου ανόδου-καθόδου

Η κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους στην περιοχή της πηδαλίου ανόδου-καθόδου (σχήμα 9.3), για μία εφαρμογή εξαναγκασμένης ταλάντωσης της εν λόγω επιφάνειας ελέγχου, φαίνεται στο σχήμα 9.12, σε διαφορετικές χρονικές στιγμές. Ο αριθμός Mach της επ΄ άπειρο ροής και οι γωνίες προσβολής (incidence) και πλαγιολίσθησης (sideslip) είναι $M_\infty = 0.85$, $\alpha_\infty = 1.7^o$, $\beta_\infty = 0^o$. Το πλάτος και η ανηγμένη συχνότητα της ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 5^o$, $\kappa = 0.1$. Στο ίδιο σχήμα διακρίνεται ο τρόπος εκτροπής της επιφάνειας ελέγχου. Κατά την κίνησή της, η επιφάνεια ελέγχου παραμένει ενωμένη στο κύριο σώμα του αεροσκάφους.

Η επίδραση της κίνησης του πηδαλίου ανόδου-καθόδου στο συντελεστή άνωσης του αεροσκάφους φαίνεται στο σχήμα 9.13. Στο ίδιο σχήμα έχει σημειωθεί ο στιγμιαίος συντελεστής άνωσης του αεροσκάφους στις τέσσερις χρονικές στιγμές του σχήματος 9.12.

Στο σχήμα 9.14(α΄) φαίνεται η τομή του αεροσκάφους στο μέσο της επιφάνειας ελέγχου ($y = 10.491m$) τις χρονικές στιγμές μέγιστης (κατά απόλυτη τιμή) εκτροπής της. Η εν λόγω επιφάνεια ελέγχου βρίσκεται πίσω από την καμπίνα των επιβατών, δηλαδή στο τέλος της ατράκτου του αεροσκάφους. Φαίνεται ότι η άτρακτος έχει σχήμα παρόμοιο με τομή πτέρυγας συμβατικού αεροσκάφους. Αυτός είναι ο βασικός λόγος που η συνεισφορά της ατράκτου στη δημιουργία της άνωσης είναι πολύ αυξημένη σε σχέση με τα συμβατικά αεροσκάφη και ένας από τους λόγους που τα αεροσκάφη τύπου BWB εμφανίζουν καλύτερα αεροδυναμικά χαρακτηριστικά (παράγραφος 9.1) σε σχέση με συμβατικά αεροσκάφη αντίστοιχων προδιαγραφών.

Η κατανομή του αριθμού Mach κατά το μήκος της ίδιας τομής φαίνεται στο σχήμα 9.14(β΄), όπου παρατηρείται η μετατόπιση της θέσης του κύματος κρούσης κατά το μήκος της ατράκτου του αεροσκάφους, με την εκτροπή του πηδαλίου ανόδου-καθόδου.

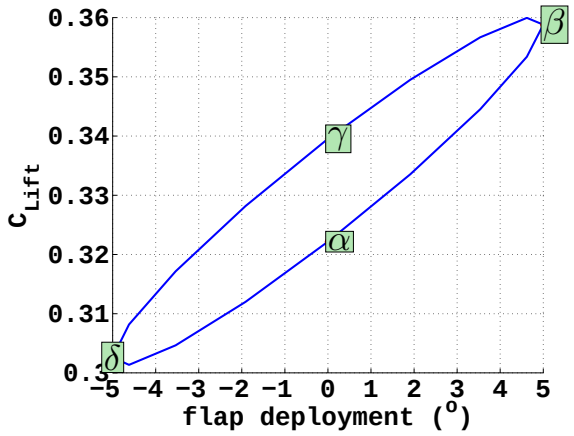


(α′) $t = 0,\ \alpha_{flap} = 0^o$ (β′) $t = T/4,\ \alpha_{flap} = 5^o$

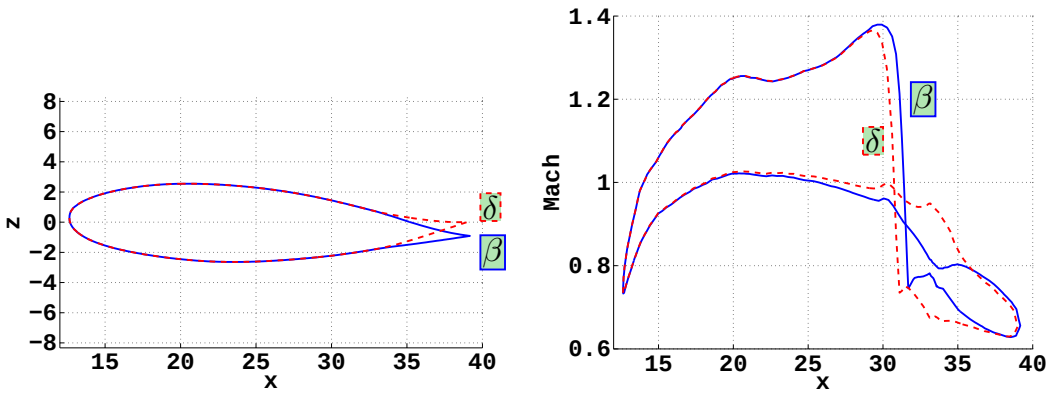
(γ′) $t = 2T/4,\ \alpha_{flap} = 0^o$ (δ′) $t = 3T/4,\ \alpha_{flap} = -5^o$

(ε′) Χρωματική παλέτα

Σχήμα 9.12: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πηδάλιο ανόδου-καθόδου κινείται περιοδικά. Κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους, στην περιοχή του πηδαλίου ανόδου-καθόδου, σε διαφορετικές χρονικές στιγμές. T είναι η περίοδος ταλάντωσης και α_{flap} η στιγμιαία γωνία εκτροπής της επιφάνειας ελέγχου. Η χρωματική παλέτα (ε′) είναι κοινή σε όλα τα παρουσιαζόμενα πεδία ροής. Οι συνθήκες της ροής είναι $M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$ και το πλάτος και η ανηγμένη συχνότητα της ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 5^o$, $\kappa = 0.1$ αντίστοιχα.



Σχήμα 9.13: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πηδάλιο ανόδου-καθόδου κινείται περιοδικά. Μεταβολή του συντελεστή άνωσης με την περιοδική εκτροπή του πηδαλίου ανόδου-καθόδου για $M_\infty = 0.85$, $\alpha_\infty = 1.7^o$, $\beta_\infty = 0^o$. Η ανηγμένη συχνότητα ταλάντωσης της επιφάνειας ελέγχου είναι $\kappa = 0.1$. Στο σχήμα έχουν σημειωθεί, με γράμματα, οι τέσσερις θέσεις του σχήματος 9.12.



(α′) (β′)

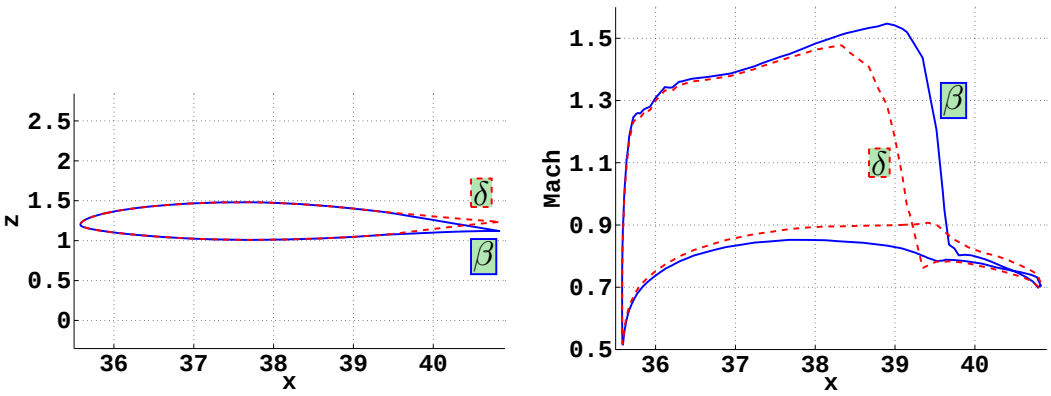
Σχήμα 9.14: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πηδάλιο ανόδου-καθόδου κινείται περιοδικά. (α′) Τομή του αεροσκάφους στο μέσο του πηδαλίου ανόδου-καθόδου ($y = 10.491m$) τις χρονικές στιγμές όπου η στιγμιαία γωνία εκτροπής είναι $\alpha_{flap} = 5^o$ (συνεχόμενη γραμμή, θέση β) και $\alpha_{flap} = -5^o$ (διακεκομμένη γραμμή, θέση δ). (β′) Κατανομή του αριθμού Mach κατά μήκος της τομής τις ίδιες χρονικές στιγμές. Παρατηρείται μετατόπιση της θέσης του κύματος κρούσης κατά το μήκος της ατράκτου του αεροσκάφους λόγω της κίνησης της επιφάνειας ελέγχου. Οι συνθήκες της ροής, το πλάτος και η ανηγμένη συχνότητα ταλάντωσης της επιφάνειας ελέγχου είναι $M_\infty = 0.85$, $\alpha_\infty = 1.7^o$, $\beta_\infty = 0^o$, A $= 5^o$ και $\kappa = 0.1$ αντίστοιχα.

9.4.2 Επίδραση του πτερυγίου κλίσης

Στην παράγραφο αυτή παρουσιάζονται τα αριθμητικά αποτελέσματα από την επίλυση ενός προβλήματος χρονικά μη-μόνιμης ροής γύρω από το αεροσκάφος με εξαναγκασμένη περιοδική κίνηση του πτερυγίου κλίσης (σχήμα 9.3). Στην εφαρμογή αυτή, οι σύνθήκες της ροής είναι $M_\infty = 0.85$, $\alpha_\infty = 1.7^o$, $\beta_\infty = 0^o$. Το πλάτος και η ανηγμένη συχνότητα ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 3^o$, $\kappa = 0.1$ αντίστοιχα.

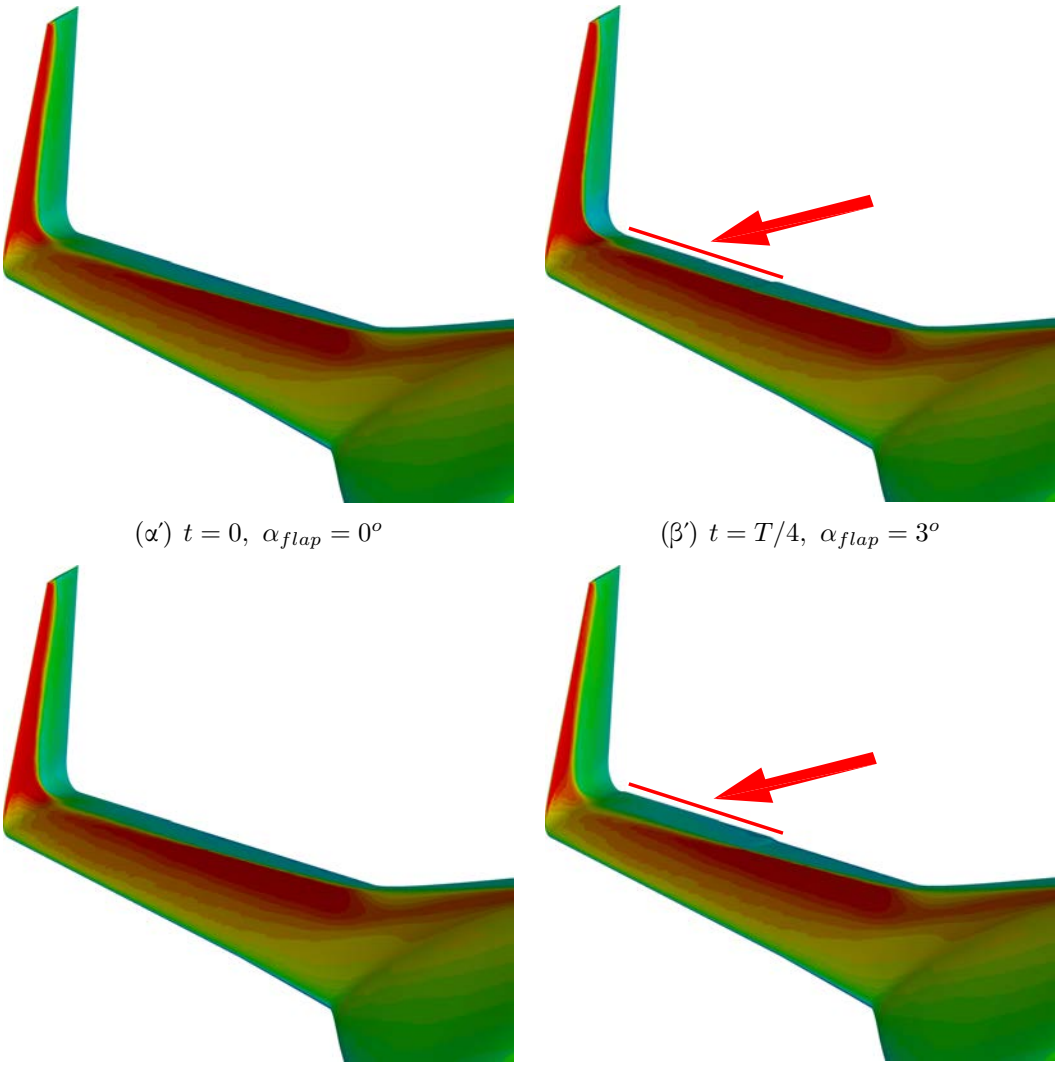
Το σχήμα 9.15(α′) δείχνει τη τομή της πτέρυγας του αεροσκάφους στο μέσο της εν λόγω επιφάνειας ελέγχου ($y = 34.374m$) με την επιφάνεια ελέγχου μετατοπισμένη κατά $\mp 3^o$. Η κατανομή του αριθμού Mach κατά μήκος της τομής, για τις ίδιες θέσεις της επιφάνειας ελέγχου, φαίνεται στο σχήμα 9.15(β′). Ιδιαίτερα εμφανής είναι η μετατόπιση της θέσης του κύματος κρούσης πάνω στην πτέρυγα του αεροσκάφους, με την κίνηση του πτερυγίου κλίσης.

Το σχήμα 9.16 δείχνει την κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους, στην περιοχή της εξεταζόμενης επιφάνειας ελέγχου.



(α′) (β′)

Σχήμα 9.15: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πτερύγιο κλίσης κινείται περιοδικά. (α′) Τομή της πτέρυγας του αεροσκάφους στο μέσο του πτερυγίου κλίσης ($y = 34.374m$) τις χρονικές στιγμές όπου η στιγμιαία γωνία εκτροπής της επιφάνειας ελέγχου είναι $\alpha_{flap} = 3^0$ (συνεχόμενη γραμμή, θέση β) και $\alpha_{flap} = -3^0$ (διακεκομμένη γραμμή, θέση δ). (β′) Κατανομή του αριθμού Mach κατά μήκος της τομής τις ίδιες χρονικές στιγμές. Παρατηρείται η μετατόπιση της θέσης του κύματος κρούσης πάνω στην πτέρυγα του αεροσκάφους με την κίνηση της επιφάνειας ελέγχου. Οι συνθήκες της ροής, το πλάτος και η ανηγμένη συχνότητα ταλάντωσης της επιφάνειας ελέγχου είναι $M_\infty = 0.85$, $\alpha_\infty = 1.7^o$, $\beta_\infty = 0^o$, $\mathrm{A} = 3^o$ και $\kappa = 0.1$ αντίστοιχα.



(α′) $t = 0,\ \alpha_{flap} = 0^o$ (β′) $t = T/4,\ \alpha_{flap} = 3^o$

(γ′) $t = 2T/4,\ \alpha_{flap} = 0^o$ (δ′) $t = 3T/4,\ \alpha_{flap} = -3^o$

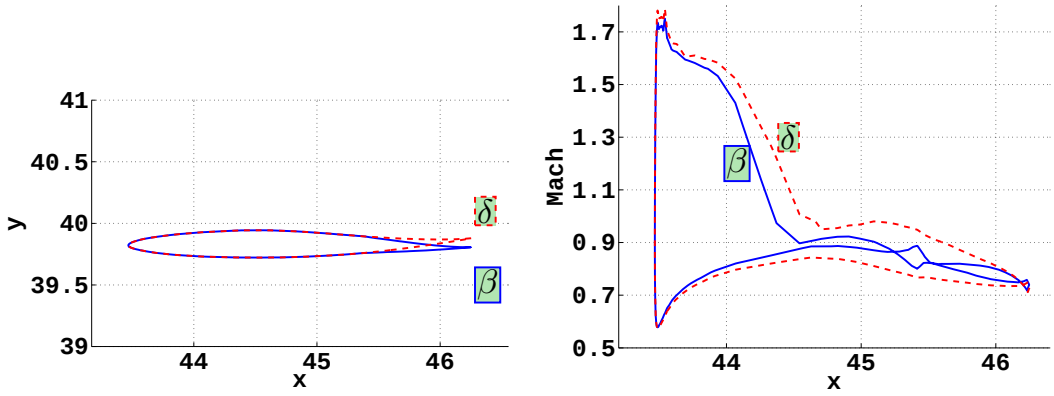
(ε′) Χρωματική παλέτα

Σχήμα 9.16: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πτερύγιο κλίσης κινείται περιοδικά. Κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους στη περιοχή του πτερυγίου κλίσης σε διαφορετικές χρονικές στιγμές. T είναι η περίοδος ταλάντωσης και α_{flap} η στιγμιαία γωνία εκτροπής της επιφάνειας ελέγχου. Η χρωματική παλέτα (ε′) είναι κοινή σε όλα τα παρουσιαζόμενα πεδία ροής. Οι συνθήκες της ροής είναι $M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$, το πλάτος και η ανηγμένη συχνότητα της ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 3^o$ και $\kappa = 0.1$ αντίστοιχα.

9.4.3 Επίδραση του πηδαλίου διεύθυνσης

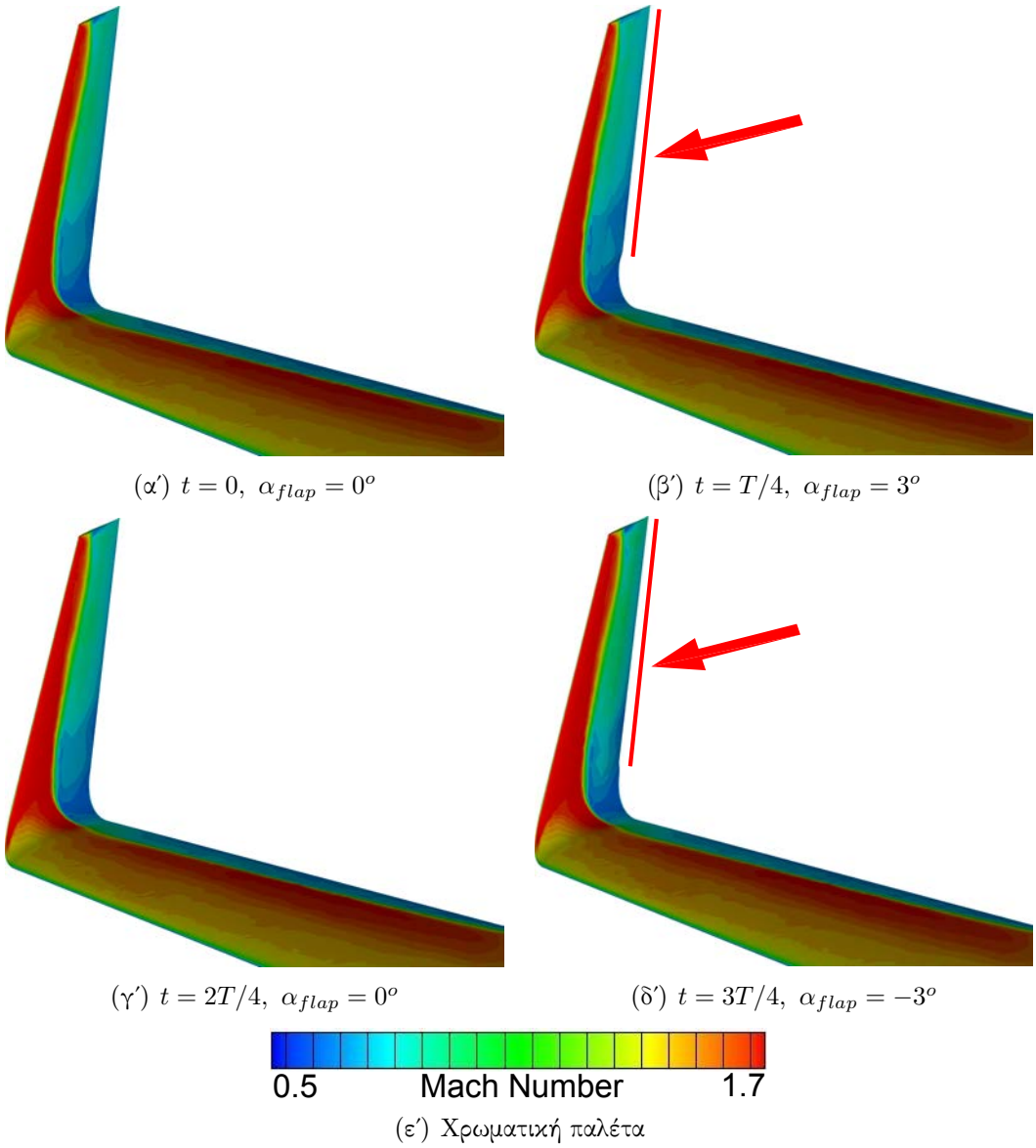
Η παράγραφος αυτή παρουσιάζει τα αριθμητικά αποτελέσματα ενός από τα προβλήματα χρονικά μη-μόνιμης ροής που επιλύθηκαν με εξαναγκασμένη περιοδική κίνηση του πηδαλίου διεύθυνσης (σχήμα 9.3). Οι συνθήκες της ροής είναι $M_\infty = 0.85$, $\alpha_\infty = 3.0^o$, $\beta_\infty = 0^o$. Το πλάτος και η ανηγμένη συχνότητα της ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 3^o$, $\kappa = 0.1$ αντίστοιχα.

Το σχήμα 9.17(α′) δείχνει τη τομή της πτέρυγας του αεροσκάφους στο μέσο του πηδαλίου διεύθυνσης ($z = 4.716m$) με την εν λόγω επιφάνεια ελέγχου μετατοπισμένη κατά $\mp 3^o$. Η κατανομή του αριθμού Mach κατά μήκος της τομής και στην επιφάνεια του αεροσκάφους (στην περιοχή της εξεταζόμενης επιφάνειας ελέγχου) φαίνεται στα σχήματα 9.17(β′) και 9.18, αντίστοιχα.



(α′) (β′)

Σχήμα 9.17: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πηδάλιο διεύθυνσης κινείται περιοδικά. (α′) Τομή της πτέρυγας στο μέσο του πηδαλίου διεύθυνσης ($z = 4.716m$) τις χρονικές στιγμές όπου η γωνία εκτροπής της επιφάνειας ελέγχου είναι $\alpha_{flap} = 3^0$ (συνεχόμενη γραμμή, θέση β) και $\alpha_{flap} = -3^0$ (διακεκομμένη γραμμή, θέση δ). (β′) Κατανομή του αριθμού Mach κατά μήκος της τομής τις ίδιες χρονικές στιγμές. Οι συνθήκες της ροής, το πλάτος και η ανηγμένη συχνότητα ταλάντωσης της επιφάνειας ελέγχου είναι $M_\infty = 0.85$, $\alpha_\infty = 3.0^o$, $\beta_\infty = 0^o$, $\mathrm{A} = 3^o$ και $\kappa = 0.1$ αντίστοιχα.



(α′) $t = 0,\ \alpha_{flap} = 0^o$ (β′) $t = T/4,\ \alpha_{flap} = 3^o$

(γ′) $t = 2T/4,\ \alpha_{flap} = 0^o$ (δ′) $t = 3T/4,\ \alpha_{flap} = -3^o$

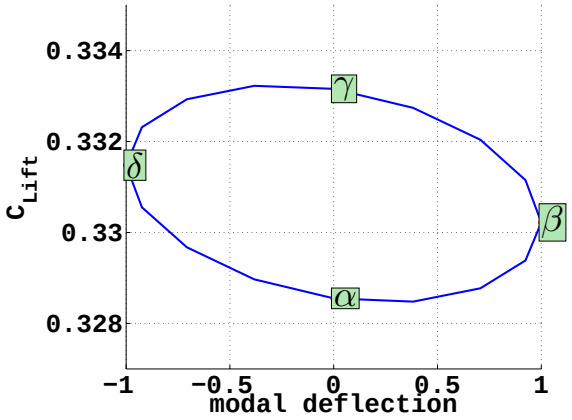
(ε′) Χρωματική παλέτα

Σχήμα 9.18: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το πηδάλιο διεύθυνσης κινείται περιοδικά. Κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους στην περιοχή του πηδαλίου διεύθυνσης σε διαφορετικές χρονικές στιγμές. T είναι η περίοδος ταλάντωσης και α_{flap} η στιγμιαία γωνία εκτροπής της επιφάνειας ελέγχου. Η χρωματική παλέτα (ε′) είναι κοινή σε όλα τα παρουσιαζόμενα πεδία ροής. Οι συνθήκες της ροής είναι $M_\infty = 0.85,\ \alpha_\infty = 3.0^o,\ \beta_\infty = 0^o$ και το πλάτος και η ανηγμένη συχνότητα της ταλάντωσης της επιφάνειας ελέγχου είναι $\mathrm{A} = 3^o$, $\kappa = 0.1$ αντίστοιχα.

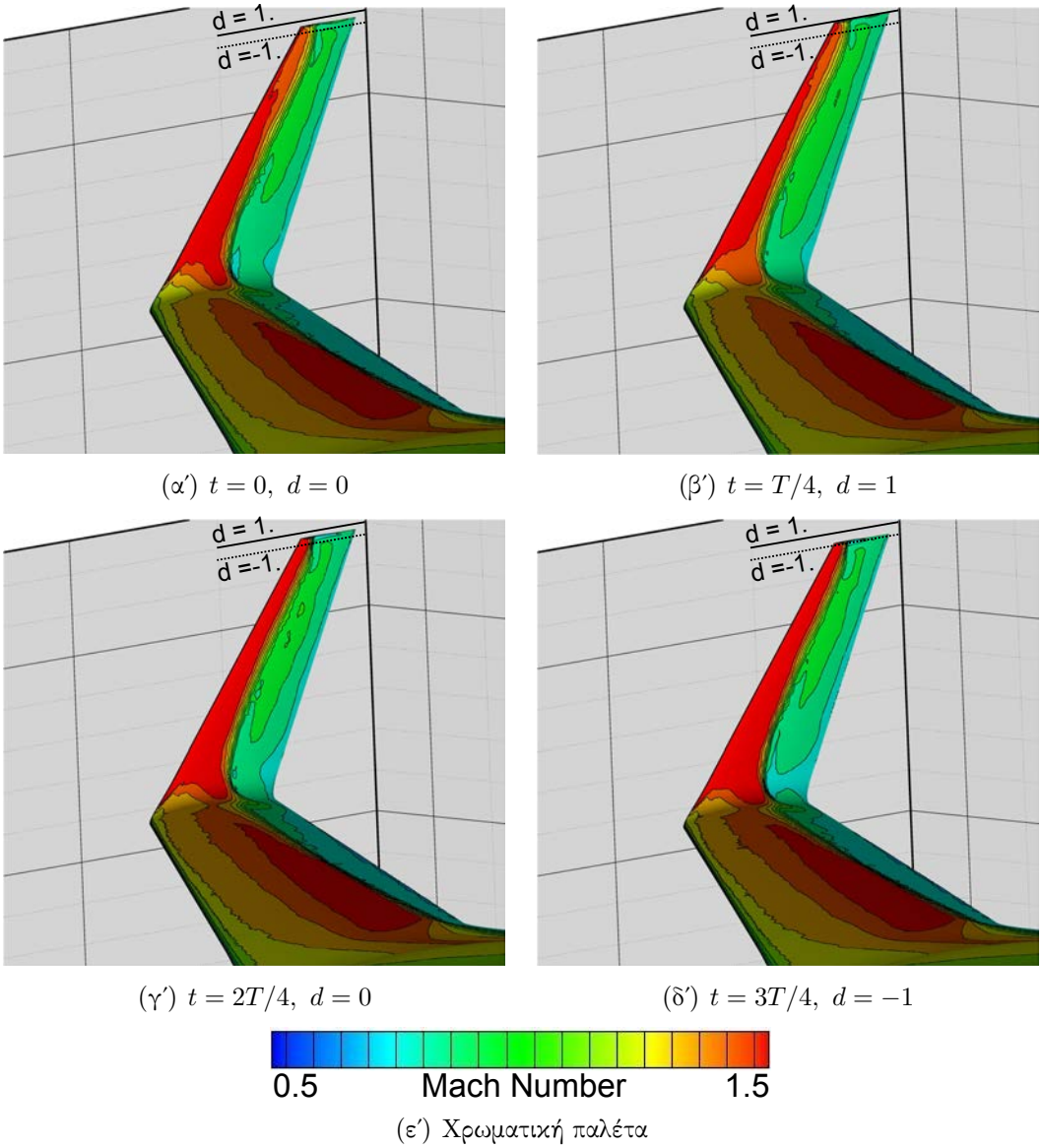
9.5 Επίδραση της παραμόρφωσης του αεροσκάφους

Επιπλέον των περιπτώσεων εξαναγκασμένης κίνησης κάποιας προεπιλεγμένης επιφάνειας ελέγχου, μελετήθηκε η επίδραση της παραμόρφωσης όλης της κατασκευής, σύμφωνα με την 1η καμπτική ιδιομορφή (σχήμα 9.4), στην αεροδυναμική συμπεριφορά του αεροσκάφους. Έτσι, επιλύθηκαν περιπτώσεις εξαναγκασμένης περιοδικής παραμόρφωσης όλης της κατασκευής, σύμφωνα με την 1η καμπτική ιδιομορφή, για διαφορετικές ανηγμένες συχνότητες ταλάντωσης και για αριθμό Mach και γωνίες πρόσπτωσης της ροής $M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$ αντίστοιχα. Η ελαστική ιδιομορφή πολλαπλασιάστηκε με κατάλληλο συντελεστή, ώστε η μέγιστη παραμόρφωση να είναι ίση με το 1/500 της μέσης χορδής του αεροσκάφους, δηλαδή περίπου 0.05 m.

Το σχήμα 9.19 δείχνει τη μεταβολή του συντελεστή άνωσης με την περιοδική παραμόρφωση όλης της κατασκευής. Η ανηγμένη συχνότητα για τη συγκεκριμένη εφαρμογή είναι $\kappa = 5.0$. Στο ίδιο σχήμα έχουν σημειωθεί οι θέσεις μηδενικής (α, γ), μέγιστης θετικής (β) και μέγιστης αρνητικής (δ) παραμόρφωσης. Η κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους στις αντίστοιχες χρονικές στιγμές (μηδενικής, μέγιστης θετικής και μέγιστης αρνητικής παραμόρφωσης) παρουσιάζεται στο σχήμα 9.20.



Σχήμα 9.19: Χρονικά μη-μόνιμη ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το αεροσκάφος εξαναγκάζεται σε περιοδική παραμόρφωση σταθερού πλάτους σύμφωνα με την πρώτη καμπτική ιδιομορφή για $M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$. Μεταβολή του συντελεστή άνωσης του αεροσκάφους. Η ανηγμένη συχνότητα ταλάντωσης του αεροσκάφους είναι $\kappa = 5.0$. Στο σχήμα έχουν σημειωθεί οι θέσεις μηδενικής (α, γ), μέγιστης θετικής (β) και μέγιστης αρνητικής (δ) παραμόρφωσης. Οι ίδιες θέσεις εμφανίζονται και στο σχήμα 9.20.



(α′) $t = 0,\ d = 0$ (β′) $t = T/4,\ d = 1$

(γ′) $t = 2T/4,\ d = 0$ (δ′) $t = 3T/4,\ d = -1$

(ε′) Χρωματική παλέτα

Σχήμα 9.20: Χρονικά μη-μόνιμη, ατριβής ροή γύρω από αεροσκάφος τύπου BWB. Το αεροσκάφος εξαναγκάζεται σε περιοδική παραμόρφωση σταθερού πλάτους σύμφωνα με την πρώτη καμπτική ιδιομορφή για $M_\infty = 0.85,\ \alpha_\infty = 1.7^o,\ \beta_\infty = 0^o$. Κατανομή του αριθμού Mach στην επιφάνεια του αεροσκάφους σε διαφορετικές χρονικές στιγμές. T είναι η περίοδος ταλάντωσης και d η στιγμιαία εκτροπή της κατασκευής. Η μέγιστη παραμόρφωση είναι περίπου $0.05m$. Η χρωματική παλέτα (ε′) είναι κοινή σε όλα τα παρουσιαζόμενα πεδία ροής.

Κεφάλαιο 10

Ανακεφαλαίωση-Συμπεράσματα

Η διατριβή αυτή επιχειρεί, κατά κάποιο τρόπο, τη μετάβαση από την εποχή ανάπτυξης λογισμικού πρόλεξης ροών σε CPUs στους πολλά υποσχόμενους επεξεργαστές καρτών γραφικών, τουλάχιστον όσον αφορά στις ερευνητικές δραστηριότητες της ΜΠΥΡ&Β/ΕΘΣ του ΕΜΠ. Στη διατριβή δομήθηκε η βάση για τη χρήση των GPUs ως μονάδων παράλληλης επεξεργασίας για την ανάλυση και το σχεδιασμό αεροδυναμικών μορφών. Παρακάτω συνοψίζεται το έργο και η συνεισφορά της διατριβής. Η τελευταία ενότητα παρουσιάζει εργασίες που βρίσκονται σε εξέλιξη ως τμήματα άλλων διδακτορικών διατριβών που, αυτή τη στιγμή, εξελίσσονται στη ΜΠΥΡ&Β/ΕΘΣ.

10.1 Ανάπτυξη GPU-κώδικα υψηλής παράλληλης απόδοσης

Στη διατριβή αναπτύχθηκε GPU-επιλύτης των 2Δ χρονικά μη-μόνιμων εξισώσεων Navier–Stokes και των 3Δ χρονικά μη-μόνιμων εξισώσεων Euler. Η μοντελοποίηση της τύρβης (όπου χρειάστηκε) έγινε μέσω του μοντέλου των Spalart-Allmaras. Ο προγραμματισμός του GPU-κώδικα επιλέχθηκε να γίνει στο περιβάλλον προγραμματισμού της CUDA, καθώς την εποχή εκείνη αυτό αποτελούσε την καλύτερη επιλογή σε σχέση με προγενέστερες γλώσσες προγραμματισμού (Cg, GLCL, HLSL, Brook Stream, κ.α.), σε όρους διαχείρισης της GPU ως σύστημα παράλληλης επεξεργασίας και όχι ως μονάδα προβολής υψηλής ανάλυσης εικόνων. Πάντως, ακόμα και σήμερα, φαίνεται ότι GPU-κώδικες προγραμματισμένοι στο περιβάλλον προγραμματισμού της CUDA εκτελούνται πιο γρήγορα σε κάρτες γραφικών της NVIDIA σε σχέση με αντίστοιχους GPU-κώδικες που έχουν προγραμματιστεί στο περιβάλλον προγραμματισμού της OpenCL.

Στοχεύοντας σε GPU-κώδικα υψηλής απόδοσης, απαιτήθηκε εκ των πραγμάτων η πλήρης αναδόμηση του CPU-κώδικα που αποτέλεσε τη βάση ανάπτυξης. Δηλαδή, δεν αρκεί απλά η μεταφορά κώδικα από FORTRAN90 (γλώσσα προγραμματισμού στην οποία είναι γραμμένος ο προϋπάρχων CPU-κώδικας της ΜΠΥΡ&Β/ΕΘΣ) σε C++ (γλώσσα προγραμματισμού που υποστηρίζει το περιβάλλον προγραμματισμού της CU-

DA). Συγκεκριμένα, παρόλο που ο πρώτος GPU-επιλύτης που προγραμματίστηκε, απλά ξαναγράφοντας τον CPU-κώδικα στο περιβάλλον προγραμματισμού της CUDA, έλυνε τις εξισώσεις της ροής πιο γρήγορα σε σχέση με τον CPU-κώδικα που εκτελείται σε έναν πυρήνα μίας σημερινής CPU, είχε μικρή παράλληλη απόδοση. Ο τελικός GPU-κώδικας είναι περίπου 20 φορές ταχύτερος, όταν εκτελείται σε μία κάρτα γραφικών σε σχέση με τον αρχικό GPU-κώδικα. Οι ενέργειες που έγιναν για την αύξηση της παράλληλης απόδοσης συνοψίζονται παρακάτω:

- Δοκιμάστηκαν και αξιολογήθηκαν, ως προς το χρόνο εκτέλεσης, διαφορετικές προγραμματιστικές τεχνικές υπολογισμού των αριθμητικών διανυσμάτων της ροής και σχηματισμού των μητρώων του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής και των υπολοίπων. Επικρατέστερη αναδείχτηκε η τεχνική των δύο kernels (κεφάλαιο 4) στις GPUs αρχιτεκτονικής Fermi ενώ οι τεχνικές ενός και δύο kernels έχουν περίπου την ίδια απόδοση σε GPUs αρχιτεκτονικής GT200. Η υπεροχή της τεχνικής των δύο kernels στις κάρτες γραφικών τελευταίας αρχιτεκτονικής οφείλεται κατά κύριο λόγο στη διαφορετική οργάνωση-διαχείριση της κεντρικής μνήμης, η οποία πλέον είναι cached. Αντίθετα, η μέγιστη ταχύτητα προσπέλασης της κεντρικής μνήμης έμεινε σχεδόν η ίδια, για αυτό και η απόδοση της τεχνικής ενός kernel δεν αυξήθηκε σημαντικά κατά τη μετάβαση από κάρτες γραφικών αρχιτεκτονικής GT200 σε κάρτες αρχιτεκτονικής Fermi.

- Έμφαση δόθηκε στην αποδοτικότερη προσπέλαση και χρήση των διαθέσιμων μνημών των καρτών γραφικών με στόχο την ελαχιστοποίηση του αριθμού των προσβάσεων στην κεντρική μνήμη και τη βέλτιστη εκμετάλλευση των μικρής χωρητικότητας cache μνημών των προγραμματιζόμενων GPUs.

- Χρησιμοποιήθηκε επικουρικά η CPU. Αυτόνομες διεργασίες εκτελούνται παράλληλα σε CPU και GPUs του ίδιου υπολογιστικού κόμβου.

- Το χρησιμοποιούμενο υπολογιστικό πλέγμα διαμερίστηκε σε επιμέρους υποχωρία. Οι κόμβοι του κάθε υποχωρίου επαναριθμήθηκαν και ταξινομήθηκαν με βάση το πλήθος των γειτονικών τους κόμβων.

- Χρησιμοποιήθηκε αριθμητική μικτής ακρίβειας. Οι διακριτοποιημένες εξισώσεις της ροής επιλύονται ως προς τη διόρθωση των μεταβλητών της ροής (δέλτα διατύπωση) και όχι απευθείας ως προς τις μεταβλητές της ροής. Επομένως, επιτρέπεται η χρήση ενός λιγότερο ακριβούς σχήματος υπολογισμού των μητρώων του αριστερού μέλους των διακριτοποιημένων εξισώσεων της ροής. Σύμφωνα με την προτεινόμενη αριθμητική μικτής ακρίβειας, τα μητρώα του αριστερού μέλους αποθηκεύονται σε μεταβλητές απλής ακρίβειας. Η χρήση μεταβλητών απλής ακρίβειας μειώνει το συνολικό πλήθος των προσβάσεων στην αργή κεντρική μνήμη της κάρτας γραφικών και, κατά συνέπεια, αυξάνει την παράλληλη επιτάχυνση του GPU-κώδικα. Αντίθετα, οι όροι του δεξιού μέλους, δηλαδή τα υπόλοιπα των εξισώσεων της ροής, αποθηκεύονται σε μεταβλητές διπλής ακρίβειας. Επομένως, οι αριθμητικές προλέξεις της μικτής ακρίβειας εκδοχής του GPU-κώδικα είναι ίδιες με εκείνες του CPU-κώδικα που χρησιμοποιεί αριθμητική διπλής ακρίβειας.

Η πρόλεξη 2Δ τυρβώδους ροής σε μία Tesla M2050 χρησιμοποιώντας τον GPU-

κώδικα της διατριβής γίνεται έως και 60, 90 ή 110 φορές πιο γρήγορα, χρησιμοποιώντας αριθμητική διπλής, μικτής ή απλής ακρίβειας αντίστοιχα, σε σχέση με τη χρήση αντίστοιχου CPU-κώδικα που εκτελείται σε έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη. Η επιτάχυνση που σημειώθηκε στην πρόλεξη 3Δ ατριβούς ροής, σε μία Tesla M2050, είναι έως και 40x, 55x ή 90x χρησιμοποιώντας αριθμητική διπλής, μικτής ή απλής ακρίβειας. Οι επιταχύνσεις αυτές θεωρούνται σημαντικά υψηλές επειδή χρησιμοποιήθηκαν μη-δομημένα υπολογιστικά πλέγματα και η κεντροκομβική διατύπωση της τεχνικής των πεπερασμένων όγκων. Ο συνδυασμός των δύο τελευταίων αποτελεί τη δυσκολότερη περίπτωση σε θέματα διαχείρισης των μνημών της κάρτας γραφικών, σε σχέση με τη χρήση δομημένων υπολογιστικών πλεγμάτων ή ακόμη και μη-δομημένων πλεγμάτων με την κεντροκυψελική διατύπωση της τεχνικής των πεπερασμένων όγκων.

Η σημαντική επιτάχυνση στην πρόλεξη της ροής από τη χρήση καρτών γραφικών διευρύνει τη κλίμακα εφαρμογών υπολογιστικής ρευστοδυναμικής. Αξιοσημείωτο είναι το γεγονός ότι προβλήματα μεγάλης κλίμακας ευνοούνται περισσότερο από τη χρήση GPUs, καθώς σε μεγαλύτερα υπολογιστικά πλέγματα σημειώνονται μεγαλύτερες επιταχύνσεις.

10.2 Ανάλυση-σχεδιασμός αεροδυναμικών μορφών σε GPUs

Ο GPU-κώδικας της διατριβής χρησιμοποιήθηκε για την αεροδυναμική ανάλυση αεροδυναμικών μορφών συμπεριλαμβανομένης της χρήσης τεχνικών ενεργητικού ελέγχου της ροής (τεχνική παλλόμενης δέσμης, synthetic jet).

Η προσθήκη της δυνατότητας χρήση δυναμικά παραμορφώσιμων υπολογιστικών πλεγμάτων και η σύζευξη του αεροδυναμικού GPU-επιλύτη με CPU-επιλύτη των ελαστικών εξισώσεων αεροτομής που δύναται να μετακινείται και να περιστρέφεται χωρίς να παραμορφώνεται έδωσε τη δυνατότητα της αεροελαστικής ανάλυσης αεροτομής δύο και τριών βαθμών ελευθερίας. Τα αριθμητικά αποτελέσματα ταιριάζουν πολύ σε δημοσιευμένα αριθμητικά αποτελέσματα άλλων ερευνητικών ομάδων. Η διεύρυνση της περιοχής ευσταθούς λειτουργίας της αεροτομής τριών βαθμών ελευθερίας επιτεύχθηκε με τον αυτόματο έλεγχο της γωνίας εκτροπής της ακμής εκφυγής της αεροτομής, μέσω ανάδρασης των μεταβλητών κατάστασης της αεροτομής. Δοκιμάστηκαν διαφορετικά κέρδη ανάδρασης και φάνηκε ότι τουλάχιστον, για τα κέρδη που χρησιμοποιήθηκαν, θετικότερη επίδραση είχε το μεγαλύτερο κέρδος ανάδρασης.

Προτάθηκε τρόπος χρησιμοποίησης του πολύ γρήγορου, αλλά όχι αποδεκτής ακρίβειας, απλής ακρίβειας GPU-λογισμικού, εντάσσοντάς το ως συνιστώσα της πολυεπίπεδης βελτιστοποίησης μέσω ΕΑ. Το κέρδος από τη χρήση του ιεραρχικού ΕΑ υπερτίθεται στο κέρδος από τη χρήση GPUs αντί σημερινών CPUs. Επιπλέον κέρδος προκύπτει από την παράλληλη αξιολόγηση των υποψήφιων λύσεων. Ο ιεραρχικός ΕΑ χρησιμοποιήθηκε για το σχεδιασμό μεμονωμένης αεροτομής με στόχους την ελαχιστοποίηση του συντελεστή οπισθέλκουσας και τη μεγιστοποίηση του συντελεστή άνωσης και αεροτομής πτερύγωσης συμπιεστή με στόχο την ελαχιστοποίηση του συντελεστή

απωλειών.

10.3 Χρήση συστοιχίας GPUs

Για την επίλυση μεγάλης κλίμακας προβλημάτων ροής χρησιμοποιήθηκε συστοιχία καρτών γραφικών. Γενικά, μία τέτοια συστοιχία αποτελείται από υπολογιστικούς κόμβους και κάθε υπολογιστικός κόμβος συνδέεται (εξωτερικά) ή έχει ενσωματωμένες (περίπτωση συστοιχίας στη ΜΠΥ&Β/ΕΘΣ) μία ή περισσότερες GPUs. Στη διατριβή, η διαχείριση των GPUs του ίδιου υπολογιστικού κόμβου ανατέθηκε στο ίδιο CPU-thread. Η επικοινωνία μεταξύ των GPUs διασυνδεδεμένων υπολογιστικών κόμβων έγινε μέσω του πρωτοκόλλου MPI. Για την αύξηση της παράλληλης επιτάχυνσης του GPU-κώδικα, η μεταφορά δεδομένων κατά την επικοινωνία των συνεργαζόμενων καρτών γραφικών γίνεται ταυτόχρονα με την επεξεργασία άλλων δεδομένων από τους πολυεπεξεργαστές των συνεργαζόμενων καρτών γραφικών.

Ο παράλληλος σε πολλές κάρτες γραφικών GPU-κώδικας χρησιμοποιήθηκε για την ανάλυση επιβατικού αεροσκάφους τύπου Blended-Wing-Body (BWB). Οι υπολογισμοί που πραγματοποιήθηκαν περιλαμβάνουν την επίλυση χρονικά μόνιμων και μη-μόνιμων προβλημάτων ροής. Όσον αφορά τα χρονικά μη-μόνιμα προβλήματα, μελετήθηκε η δυναμική επίδραση της εξαναγκασμένης ταλάντωσης επιλεγμένων επιφανειών ελέγχου ή ολόκληρης της κατασκευής, σύμφωνα με επιλεγείσα ιδιομορφή αυτής, στη ροή γύρω από το αεροσκάφος. Η χρήση τριών Tesla M2050 στον ίδιο υπολογιστικό κόμβο μείωσε το χρόνο επίλυσης των προβλημάτων αυτών περίπου 6 φορές σε σχέση με τη χρήση του παράλληλου CPU-κώδικα σε 64 CPU-πυρήνες. Τα αεροσκάφη τύπου BWB είναι μη-συμβατικά αεροσκάφη με πολλά αεροδυναμικά προτερήματα σε σχέση με συμβατικά αεροσκάφη ίδιων προδιαγραφών και θεωρούνται, ενδεχομένως, το μέλλον της αεροπορικής βιομηχανίας.

10.4 Τρέχουσες και μελλοντικές εργασίες

Επικουρικά, και για λόγους επιπλέον συγκρίσεων και πληρότητας, στη διατριβή πραγματοποιήθηκε και η ανάπτυξη GPU-επιλυτών των 3Δ εξισώσεων ατριβούς συμπιεστής ροής σε δομημένα πλέγματα, ή κάνοντας χρήση της κεντροκυψελικής διατύπωσης της τεχνικής των πεπερασμένων όγκων σε μη-δομημένα πλέγματα. Λεπτομέρειες αναφέρονται στο παράρτημα Α΄. Επιπλέον, η διατριβή συνεισέφερε στην αποδοτική μεταφορά κωδίκων επίλυσης των 2Δ εξισώσεων ασυμπίεστης ροής και του συζυγούς προβλήματος από την CPU, στη GPU (παράρτημα Β΄). Η επέκταση του παράλληλου σε πολλές κάρτες γραφικών GPU-κώδικα της διατριβής στην επίλυση των 3Δ χρονικά μη-μόνιμων εξισώσεων Navier–Stokes συμπιεστού και ασυμπίεστου ρευστού, σε μη-δομημένα υβριδικά υπολογιστικά πλέγματα αποτελούμενα από τετράεδρα, πυραμίδες, πρίσματα και/ή εξάεδρα, βρίσκεται σε εξέλιξη στο πλαίσιο άλλων διδακτορικών διατριβών που εκπονούνται από τους Ι. Καββαδία και Κ. Τσιάκα στη ΜΠΥΡ&Β/ΕΘΣ, [251, 252].

10.5 Δημοσιεύσεις

Αποτελέσματα της έρευνας της παρούσας διδακτορικής διατριβής έχουν, μέχρι τώρα, δημοσιευθεί σε τεύχος επιστημονικού βιβλίου, σε διεθνή επιστημονικά περιοδικά ή στα πρακτικά διεθνών επιστημονικών συνεδρίων. Ακολουθεί σχετικός κατάλογος.

1. X.S. Trompoukis, V.G. Asouti, I.C. Kampolis, K.C. Giannakoglou, "CUDA Implementation of Vertex-Centered, Finite Volume CFD methods on Unstructured Grids with Flow Control Applications", GPU Gems, NVIDIA (2011).

2. I.C. Kampolis, X.S Trompoukis, V.G. Asouti and K.C. Giannakoglou. "CFD-based analysis and two-level aerodynamic optimization on Graphics Processing Units", Computer Methods in Applied Mechanics and Engineering 2010; 199(9-12):712-722.

3. V.G. Asouti, X.S Trompoukis, I.C. Kampolis and K.C. Giannakoglou. "Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units", International Journal for Numerical Methods in Fluids 2011; 67(2):232-246.

4. X.S. Trompoukis, V.G. Asouti, T.D. Zervogiannis and K.C. Giannakoglou, "CFD Analysis and Parametric Study-Optimization of Suction-Blowing Flow Control Techniques", 6th GRACM International Congress on Computational Mechanics, Thessaloniki, 19-21 June, 2008.

5. V.G. Asouti, E.A. Kontoleontos, X. S. Trompoukis and K.C. Giannakoglou, "Shape Optimization Using The One-Shot Adjoint Technique on Graphics Processing Units", 7th GRACM International Congress on Computational Mechanics, Athens, 30 June – 2 July, 2011.

Παράρτημα Α΄

GPU-επιλύτης σε δομημένα πλέγματα

Στη διατριβή, σε συνεργασία με τον υπ. διδάκτορα Ι. Καββαδία στο πλαίσιο της διπλωματικής του, τότε, εργασίας στη ΜΠΥΡ&Β/ΕΘΣ, πραγματοποιήθηκε και η ανάπτυξη GPU-επιλύτη των εξισώσεων ατριβούς ροής συμπιεστού ρευστού σε 3Δ δομημένα πλέγματα με κεντροκομβική αποθήκευση των μεταβλητών της ροής, [253]. Ο GPU-επιλύτης που αναπτύχθηκε χρησιμοποιεί το σχήμα του Roe, κεφάλαιο 2, για τον υπολογισμό των αριθμητικών διανυσμάτων ατριβούς ροής. Η επίλυση των διακριτοποιημένων εξισώσεων Euler γίνεται με τη μέθοδο Jacobi. Στη συνέχεια, παρατίθενται συνοπτικά τα βασικά συμπεράσματα, ως προς τη διαχείριση της GPU, που προέκυψαν από τη χρήση δομημένων αντί μη-δομημένων πλεγμάτων[1].

Α΄.1 Επίλυση των εξισώσεων Euler σε δομημένα πλέγματα

Στα δομημένα πλέγματα, η αρίθμηση των κόμβων ακολουθεί ένα δομημένο πρότυπο το οποίο βοηθά στη βέλτιστη προσπέλαση της κεντρικής μνήμης της κάρτας γραφικών.

Στη διπλωματική εργασία [253] επιλύονται οι εξισώσεις Euler σε 3Δ δομημένα πλέγματα. Ο υπολογισμός των ιακωβιανών μητρώων D, Z και των υπολοίπων $\overrightarrow{R}$ γίνεται σύμφωνα με την τεχνική ενός kernel. Υπενθυμίζεται ότι στην τεχνική αυτή κάθε

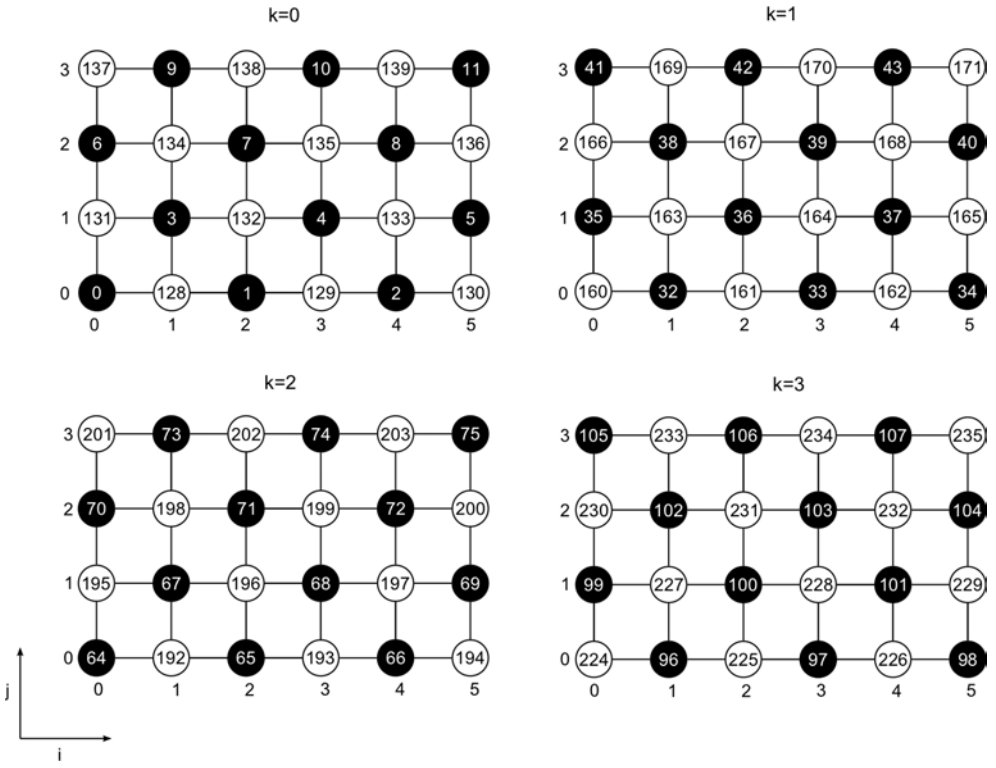
[1]Επιπλέον, στο πλαίσιο της παρούσας διατριβής και σε συνεργασία με εκπονηθείσα διπλωματική εργασία [254] αναπτύθηκε GPU-κώδικας που επιλύει τις εξισώσεις Euler σε 2Δ μη-δομημένα πλέγματα μέσω της κεντροκυψελικής διατύπωσης της τεχνικής των πεπερασμένων όγκων. Ο υπολογισμός των αριθμητικών διανυσμάτων ατριβούς ροής υπολογίζεται σύμφωνα με την τεχνική ενός kernel. Χρησιμοποιώντας την κεντροκυψελική διατύπωση, αντί της κεντροκομβικής, ο αριθμός των γειτόνων ανά τριγωνικό στοιχείο είναι σταθερός, λ.χ. ίσος με 3 για τα εσωτερικά τρίγωνα. Το γεγονός αυτό αυξάνει την παράλληλη απόδοση του kernel υπολογισμού των $\overrightarrow{R}$, D και Z. Εφόσον, κάθε thread σαρώνει πάντα τρεις (ή δύο για τα οριακά τρίγωνα) γείτονες, ο χρόνος εκτέλεσης αυτών είναι περίπου ίδιος. Συνεπώς, τα threads του ίδιου warp ολοκληρώνουν την εκτέλεση του kernel ταυτόχρονα. Τα τριγωνικά στοιχεία αριθμήθηκαν με τέτοιο τρόπο, ώστε τα threads του ίδιου warp να σχετίζονται μόνο με εσωτερικά ή μόνο με οριακά στοιχεία. Η μη-διαθεσιμότητα αντίστοιχου CPU-κώδικα απέτρεψε την απευθείας σύγκριση ως προς την παράλληλη επιτάχυνση.

thread σχετίζεται με έναν κόμβο του πλέγματος και σαρώνει τις ακμές που άγονται από τον κόμβο αυτόν υπολογίζοντας τις αντίστοιχες ροές $\overrightarrow{\Phi}^{inv}$ και τα ιακωβιανά μητρώα $\frac{\overrightarrow{\Phi}^{inv}}{\partial \overrightarrow{W}}$ τα οποία χρησιμοποιεί για το σχηματισμό των $\overrightarrow{R}$, D και Z που σχετίζονται με τον εξεταζόμενο κόμβο. Έχει τονιστεί, ότι βασικό μειονέκτημα της τεχνικής αυτής είναι ο διπλός υπολογισμός των ροών $\overrightarrow{\Phi}^{inv}_{PQ}$ και των ιακωβιανών μητρώων $\frac{\overrightarrow{\Phi}^{inv}_{PQ}}{\partial \overrightarrow{W}}$ ανά ακμή PQ του πλέγματος.

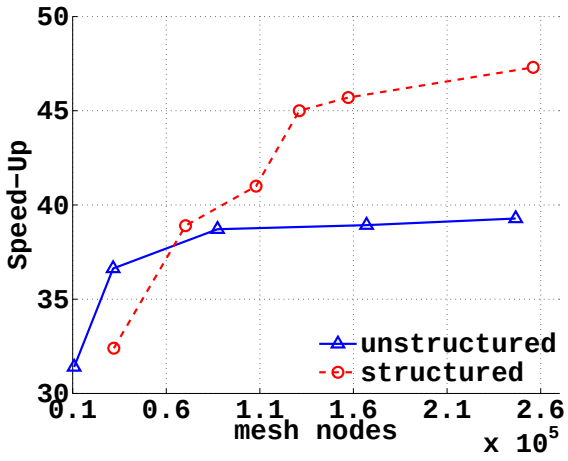
Σε δομημένα πλέγματα προτείνεται ο διαμερισμός των κόμβων του πλέγματος σε δύο ομάδες έτσι ώστε δύο κόμβοι της ίδιας ομάδας να μην ενώνονται μέσω ακμής, σχήμα Α΄.1. Στην περίπτωση αυτή, το kernel της παραπάνω τεχνικής διαμορφώνεται έτσι ώστε κάθε thread να μην υπολογίζει μόνο τα $\overrightarrow{R}$, D, Z του κόμβου με τον οποίο σχετίζεται, αλλά να συνεισφέρει και στον υπολογισμό των $\overrightarrow{R}$, D, Z των γειτονικών κόμβων. Το προσαρμοσμένο σε δομημένα πλέγματα kernel εκτελείται μόνο στους κόμβους της πρώτης ομάδας. Έτσι, αποτρέπεται ο διπλός υπολογισμός των $\overrightarrow{\Phi}^{inv}_{PQ}$, $\frac{\overrightarrow{\Phi}^{inv}_{PQ}}{\partial \overrightarrow{W}}$ και μειώνεται σημαντικά ο χρόνος εκτέλεσης του kernel. Η χρήση του προτεινόμενου kernel σε μη-δομημένα πλέγματα είναι απαγορευτική λόγω της ακατάστατης αρίθμησης των κόμβων του πλέγματος. Για παράδειγμα, σε μη-δομημένα πλέγματα, το kernel δεν εξασφαλίζει ότι threads που εκτελούνται ταυτόχρονα δεν αποθηκεύουν σε ίδιες θέσεις μνήμης παρά μόνο με τη χρήση των atomic functions. Αυτές, όπως έχει τονιστεί, μειώνουν την παράλληλη απόδοση του GPU-επιλύτη. Επιπλέον, λόγω της ακατάστατης αρίθμησης των κόμβων, η πρόσβαση σε θέσεις μνήμης που σχετίζονται με δεδομένα των γειτονικών κόμβων (για την ανανέωση των $\overrightarrow{R}$, D, Z αυτών) οδηγεί σε ακατάστατη προσπέλαση της κεντρικής μνήμης της κάρτας γραφικών και μειώνει σημαντικά την παράλληλη απόδοση του GPU-κώδικα. Αντίθετα, στα δομημένα πλέγματα, η ανανέωση των $\overrightarrow{R}$, D, Z των γειτονικών κόμβων γίνεται με τη βέλτιστη προσπέλαση της κεντρικής μνήμης της κάρτας γραφικών, λόγω της δομημένης αρίθμησης των κόμβων. Επιπλέον, ο συγχρονισμός των threads, όταν ανανεώνουν τα $\overrightarrow{R}$, D, Z των γειτονικών κόμβων γίνεται έμμεσα, χωρίς τη χρήση των atomic functions, στο επίπεδο των warps. Υπενθυμίζεται, ότι τα threads του ίδιου warp εκτελούν ταυτόχρονα τις γραμμές εντολών του εκάστοτε kernel. Δηλαδή, όλα τα threads του ίδιου warp ανανεώνουν ταυτόχρονα τα $\overrightarrow{R}$, D και Z των 'αριστερών' γειτόνων, στη συνέχεια των 'δεξιών' κ.ο.κ.

Οι κόμβοι του πλέγματος επαναριθμούνται με βάση την ομάδα στην οποία ανήκουν. Έτσι, πρώτα αριθμούνται οι κόμβοι της πρώτης ομάδας και, μετά, οι κόμβοι της δεύτερης, όπως φαίνεται στο σχήμα Α΄.1. Προτείνεται, με κόστος δέσμευσης επιπλέον θέσεων μνήμης, η αρίθμηση των κόμβων της δεύτερης ομάδας να μην είναι συνεχόμενη με την αρίθμηση των κόμβων της πρώτης ομάδας, αλλά να ξεκινά από ένα πολλαπλάσιο του 32 (διάσταση warp), όπως φαίνεται στο σχήμα Α΄.1.

Κατά την επίλυση των εξισώσεων Euler σε 3Δ δομημένα πλέγματα, με διαφορετικό πλήθος κόμβων, σε μία Tesla M2050, σημειώθηκαν επιταχύνσεις έως και περίπου 50x, σχήμα Α΄.2. Πρόκειται για περίπου 25% μεγαλύτερη επιτάχυνση σε σχέση με αυτήν της πρόλεξης ατριβών ροών του GPU-κώδικα της διατριβής σε μη-δομημένα πλέγματα με ανάλογο πλήθος κόμβων. Η αύξηση αυτή είναι αποτέλεσμα της δομημένης αρίθμησης



Σχήμα Α΄.1: Ομαδοποίηση κόμβων 3Δ δομημένου πλέγματος διάστασης $6 \times 4 \times 4$ σε δύο ομάδες, [253]. Οι κόμβοι της μίας ομάδας δεν ενώνονται μέσω ακμής με κόμβους της άλλης ομάδας. Στο σχήμα φαίνεται, επίσης, η αρίθμηση των κόμβων κατά τις i, j κατευθύνσεις και οι αύξοντες αριθμοί αυτών.



Σχήμα Α΄.2: Επιτάχυνση της επίλυσης ατριβούς ροής γύρω από πτέρυγα ή εσωτερικά αγωγού χρησιμοποιώντας μη-δομημένα και δομημένα πλέγματα, [253], αντίστοιχα. Οι δύο GPU-κώδικες εκτελούνται σε μία Tesla M2050 και χρησιμοποιούν αριθμητική διπλής ακρίβειας.

των κόμβων του πλέγματος και της σημαντικής μείωσης του υπολογιστικού κόστους

του kernel υπολογισμού των $\overrightarrow{R}$, D, Z. Οι δύο GPU-κώδικες χρησιμοποιούν αριθμητική διπλής ακρίβειας.

Παράρτημα Β′

Σχεδιασμός μορφής αγωγού με στροφή 90^o

Η διατριβή συνεισέφερε, επιπλέον, στην αποδοτική μεταφορά κωδίκων επίλυσης των 2Δ εξισώσεων Navier–Stokes ασυμπίεστου ρευστού και των συζυγών εξισώσεων όπως εκείνες προκύπτουν από τη συνεχή συζυγή μέθοδο (continuous adjoint), [255, 256, 257, 183, 184, 185, 186], από την CPU στη GPU. Επιτεύχθηκε η σύζευξη του GPU-επιλύτη του εξισώσεων Navier–Stokes (ευθύ πρόβλημα) με τον GPU-επιλύτη των συζυγών εξισώσεων μέσω της τεχνικής one-shot, [258, 259, 260, 261, 262, 263], και έγινε ο σχεδιασμός αγωγού με στροφή της ροής κατά 90^o με στόχο την ελαχιστοποίηση των απωλειών ολικής πίεσης χρησιμοποιώντας τόσο την 'κλασική' προσέγγιση όσο και την τεχνική one-shot. Η ίδια εφαρμογή παρουσιάζεται και στη διατριβή της Ε. Κοντολέοντος, [264]. Εξάλλου, στην παρούσα διατριβή έμφαση δόθηκε στην αποδοτική μεταφορά των CPU-κωδίκων στη GPU.

Η 'κλασική' προσέγγιση περιλαμβάνει, σε κάθε κύκλο βελτιστοποίησης, πρώτα την επίλυση του ευθέος προβλήματος, μετά αυτήν των συζυγών εξισώσεων και, στο τέλος, την ανανέωση των μεταβλητών σχεδιασμού βάσει των υπολογισθεισών παραγώγων ευαισθησίας. Αντίθετα, σύμφωνα με την τεχνική one-shot, μόνο για τον πρώτο κύκλο βελτιστοποίησης, επιλύεται αρχικά το ευθύ πρόβλημα και στη συνέχεια το συζυγές μέχρι σύγκλισης των αντίστοιχων επιλυτών στον ψευδοχρόνο. Στους υπόλοιπους κύκλους βελτιστοποίησης, η επίλυση του ευθέος και του συζυγούς προβλήματος και η ανανέωση των μεταβλητών σχεδιασμού με βάση τις υπολογισθείσες παραγώγους ευαισθησίας γίνεται ταυτόχρονα. Δηλαδή, για την ολοκλήρωση ενός κύκλου βελτιστοποίησης (εκτός του πρώτου), ο επιλύτης του ευθέος και του συζυγούς προβλήματος πραγματοποιούν ταυτόχρονα ένα βήμα στον ψευδοχρόνο, στη συνέχεια υπολογίζονται οι παράγωγοι ευαισθησίας και ανανεώνονται οι μεταβλητές σχεδιασμού. Η χρήση της τεχνικής one-shot, αντί της 'κλασικής' προσέγγισης, μειώνει το χρόνο της βελτιστοποίησης. Το κέρδος αυτό υπερτίθεται στο κέρδος από τη χρήση GPUs.

Από την πλευρά του προγραμματισμού σε GPUs, η χρήση της τεχνικής one-shot επιβάλλει τη διαχείριση πινάκων και διανυσμάτων μεγαλύτερης διάστασης (αφού επιλύονται ταυτόχρονα οι εξισώσεις Navier–Stokes με τις συζυγείς εξισώσεις). Γι′ αυτό,

η προσεκτική διαχείριση της μνήμης της κάρτας γραφικών αποκτά ακόμη μεγαλύτερη σημασία. Η εφαρμογή των τεχνικών που αναλύονται στο κεφάλαιο 4 εξασφαλίζουν την αποδοτική προσπέλαση και χρήση των μνημών της κάρτας γραφικών και οδηγούν σε GPU-κώδικα αντίστοιχης παράλληλης επιτάχυνσης με το GPU-επιλύτη των εξισώσεων Navier–Stokes συμπιεστού ρευστού.

Πραγματοποιήθηκε ο σχεδιασμός αγωγού με στροφή 90^o τόσο με την 'κλασική' προσέγγιση όσο και με τη τεχνική one-shot, με στόχο την ελαχιστοποίηση του συντελεστή απωλειών ολικής πίεσης σταθμισμένου με την παροχή

$$F = -\int_{S_{I,O}} \frac{1}{\varrho}\left(p + \frac{1}{2}\varrho u^2\right) u_i n_i dS \tag{B′.1}$$

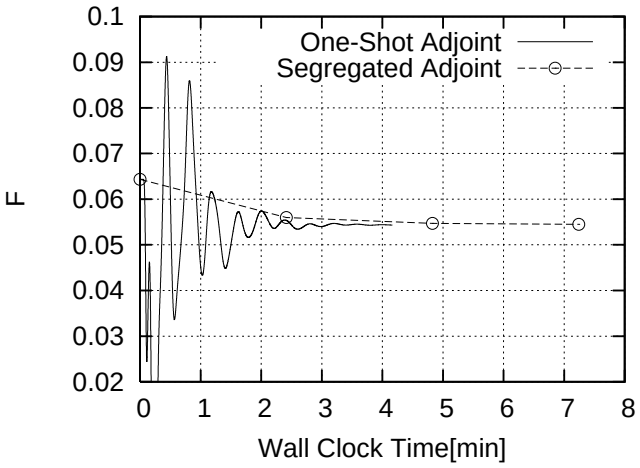
όπου S_I, S_O είναι οι διατομές εισόδου (Inlet) και εξόδου (Outlet), αντίστοιχα, του συμπιεστή και dS στοιχειώδες τμήμα αυτών. Στην εφαρμογή αυτή, ο αριθμός Reynolds ως προς το ύψος του αγωγού στην είσοδο ισούται με $Re = 65000$. Το τμήμα της στροφής του αγωγού παραμετροποιείται χρησιμοποιώντας πολυώνυμα Bézier–Bernstein με 7 σημεία ελέγχου σε κάθε πλευρά του. Δεδομένου ότι, το πρώτο και το τελευταίο σημείο ελέγχου κάθε πλευράς διατηρούνται σταθερά και οι τετμημένες των υπολοίπων σημείων ελέγχου δεν επιτρέπεται να μετακινούνται, χρησιμοποιήθηκαν συνολικά 10 μεταβλητές σχεδιασμού.

Το σχήμα B′.1 δείχνει τη σύγκλιση των αλγορίθμων βελτιστοποίησης. Στον οριζόντιο άξονα σχεδιάζεται ο χρόνος της βελτιστοποίησης ενώ στον κατακόρυφο ο συντελεστής απωλειών ολικής πίεσης. Η χρήση της τεχνικής one-shot μειώνει το χρόνο σχεδιασμού κατά περίπου 40% (1.7x). Το κέρδος αυτό υπερτίθεται σε αυτό από τη χρήση GPUs αντί CPUs.

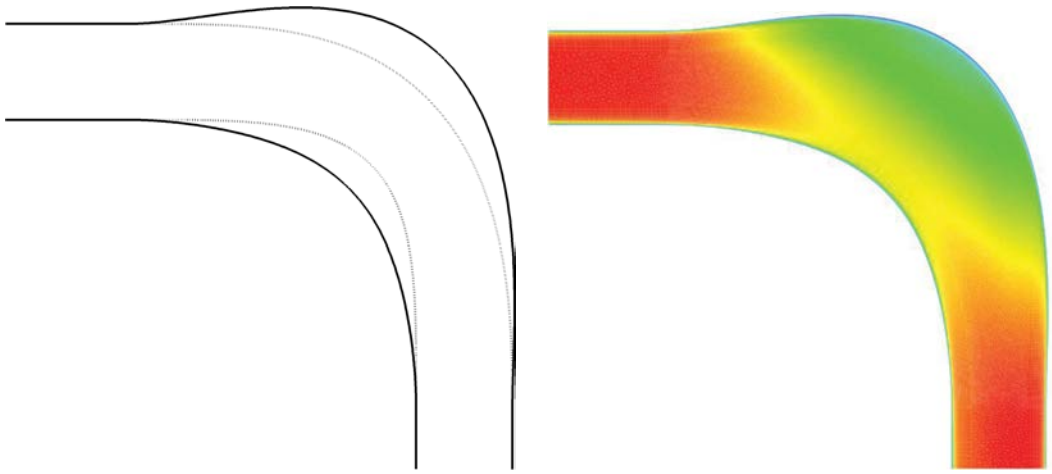
Τα σχήματα B′.2(α′), B′.2(β′) παρουσιάζουν το βέλτιστο σχήμα του αγωγού σε σχέση με το αρχικό και το πεδίο του μέτρου της ταχύτητας του ρευστού στο εσωτερικό του βέλτιστου αγωγού, αντίστοιχα.

Το πλέγμα που χρησιμοποιήθηκε αποτελείται από τριγωνικά στοιχεία τα οποία προέκυψαν από τη διάσπαση τετραπλευρικών κοντά στο τοίχωμα και με βάση την τεχνική του προελαύνοντος μετώπου στο υπόλοιπο του χωρίου της ροής. Συγκεκριμένα, το πλέγμα αποτελείται περίπου από 40000 κόμβους και 85000 τριγωνικά στοιχεία. Η πραγματοποίηση ενός βήματος στον ψευδοχρόνο για την επίλυση των εξισώσεων Navier–Stokes και των συζυγών τους εξισώσεων γίνεται 37 φορές πιο γρήγορα σε μία Tesla M2050 ως προς έναν πυρήνα μίας Intel Xeon CPU στα 2.00 GHz, με 4096 MByte cache μνήμη. Υπερθέτοντας το κέρδος από τη χρήση της τεχνικής one-shot (περίπου 1.7x) σε εκείνο από τη χρήση μίας GPU, η εύρεση της βέλτιστης λύσης πραγματοποιείται περίπου 60 φορές πιο γρήγορα από το να χρησιμοποιούσε κανείς την 'κλασική' προσέγγιση σε έναν πυρήνα μίας CPU.

Η προσαρμογή του πλέγματος στην ανανεωμένη γεωμετρία γίνεται όπως περιγράφεται στο κεφάλαιο 8 για 2Δ εφαρμογές. Στην αρχή κάθε κύκλου βελτιστοποίησης, ο υπολογισμός των γεωμετρικών στοιχείων του πλέγματος γίνεται στη GPU.



Σχήμα Β′.1: Σχεδιασμός αγωγού με στροφή 90^o, [264]. Η πορεία της σύγκλισης με την 'κλασική' προσέγγιση (Segregated Adjoint) και την τεχνική one-shot (One-Shot Adjoint).



(α′) (β′)

Σχήμα Β′.2: Σχεδιασμός αγωγού με στροφή 90^o, [264]. (α′) Αρχικό (διακεκομμένη γραμμή) και βέλτιστο (συνεχής γραμμή) σχήμα αγωγού. (β′) Πεδίο του μέτρου της ταχύτητας στο εσωτερικό του αγωγού με το βέλτιστο σχήμα.

Βιβλιογραφία

[1] *General–purpose computation on graphics processing units.* `http://gpgpu.org/`.

[2] NVIDIA CUDA^{TM}: *NVIDIA CUDA C Programming Guide*, 4.0 edition, March 2011.

[3] Meredith, J., Alvarez, G., Maier, T., Schulthess, T., and Vetter, J.: *Accuracy and performance of graphics processors: A Quantum Monte Carlo application case study.* Parallel Computing, 35(3):151–163, 2009.

[4] Göddeke, Dominik, Strzodka, Robert, and Turek, Stefan: *Accelerating double precision FEM simulations with GPUs.* In 18th Symposium on Simulation Technique, ASIM, Erlangen, September 12-15 2005.

[5] Göddeke, D., Strzodka, R., Mohd-Yusof, J., McCormick, P., Wobker, H., Becker, C., and Turek, S.: *Using GPUs to improve multigrid solver performance on a cluster.* International Journal of Computational Science & Engineering, 4(1):36–55, 2008.

[6] Ford, E.: *Parallel algorithm for solving Kepler's equation on Graphics Processing Units: Application to analysis of Doppler exoplanet searches.* New Astronomy, 14(4):406–412, 2009.

[7] Kahan, W.: *Further remarks on reducing truncation errors.* Communications of the ACM, 8(1):40–48, 1965.

[8] Kampolis, I., Trompoukis, X., Asouti, V., and Giannakoglou, K.: *CFD–based analysis and two–level aerodynamic optimization on Graphics Processing Units.* Computer Methods in Applied Mechanics and Engineering, 199(9–12):712–722, 2010.

[9] Larsen, E. and McAllister, D.: *Fast matrix multiplies using graphics hardware.* In High Performance Computing, Networking, Storage and Analysis Conference, Denver, Colorado, November 10–16 2001.

[10] Thompson, C., Hahn, S., and Oskin, M.: *Using modern graphics architectures for general-purpose computing: A framework and analysis.* In 35th International Symposium on Microarchitecture Conference, Istanbul, Turkey, November 18-22 2002.

[11] Krügen, J. and Westermann, R.: *Linear algebra operators for GPU implentation of numerical algorithms.* ACM Transactions on Graphics, 22(3):908–916, 2003.

[12] Galoppo, N., Govindaraju, N., Henson, M., and Manocha, D.: *LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware.* In High Performance Computing, Networking, Storage and Analysis Conference, Seattle, Washington, November 12–18 2005.

[13] Volkov, V. and Demmel, J.: *LU, QR and Cholesky factorizations using vector capabilities of GPUs.* Technical Report UCB/EECS-2008-49, 2008.

[14] Volkov, V. and Demmel, J.: *Benchmarking GPUs to tune dense linear algebra.* In High Performance Computing, Networking, Storage and Analysis Conference, Austin, Texas, November 15-21 2008.

[15] Barrachina, S., Castillo, M., Igual, F., Mayo, R., and Quintana-Ortí, E.: *Solving dense linear systems on graphics processors.* In 14th international Euro-Par Conference on Parallel Processing, Las Palmas de Gran Canaria, Spain, August 26-29 2008.

[16] Naumov, M.: *Incomplete-LU and Cholesky preconditioned iterative methods using CUSPARSE and CUBLAS.* NVIDIA white paper, 2011.

[17] Ries, F., De Marco, T., Zivieri, M., and Guerrieri, R.: *Triangular matrix inversion on Graphics Processing Unit.* In High Performance Computing, Networking, Storage and Analysis Conference, Portland, Oregon, November 14–20 2009.

[18] Bauke, H. and Keitel, C.: *Accelerating the Fourier split operator method via Graphics Processing Units.* Computer Physics Communications, 182(12):2454–2463, 2011.

[19] Govindaraju, N., Lloyd, B., Dotsenko, Y., Smith, B., and Manferdelli, J.: *High-performance discrete Fourier transforms on graphics processors.* In High Performance Computing, Networking, Storage and Analysis Conference, Austin, Texas, November 15-21 2008.

[20] Bolz, J., Farmer, I., Grinspun, E., and Schröder, P.: *Sparse matrix solvers on the GPU: Conjugate gradients and multigrid.* ACM Transactions on Graphics, 22(3):917–924, 2003.

[21] Hoff, K., Keyser, J., Lin, M.C., Manocha, D., and Culver, T.: *Fast computation of generalized Voronoi diagrams using graphics hardware.* In SIGGRAPH 1999, Los Angeles, California, August 8-13 1999.

[22] Rumpf, M. and Strzodka, R.: *Nonlinear diffusion in graphics hardware.* In EG/IEEE TCVG Symposium on Visualization, 2001.

[23] Grossauer, H. and Thoman, P.: *GPU-based multigrid: Real-time performance in high resolution nonlinear image processing*. Lecture Notes in Computer Science, 5008:141–150, 2008.

[24] Liu, W., Schmidt, B., Voss, G., and Müller-Wittig, W.: *Accelarting molecular dynamics simulations using Graphics Processing Units with CUDA*. Computer Physics Communications, 179(9):634–641, 2008.

[25] Stone, J., Phillips, J., Feddolino, P., Hardy, D., Trabuco, L., and Schulten, K.: *Accelerating molecular modelling applications with graphics processors*. Journal of Computational Chemistry, 28(16):2618–2640, 2007.

[26] Anderson, J., lorenz, C., and Travesset, A.: *General purpose molecular dynamics simulations fully implented on Graphics Processing Units*. Journal of Computational Physics, 227(10):5342–5359, 2008.

[27] Yang, J., Wang, Y., and Chen, Y.: *GPU accelerated molecular dynamics simulation of thermal conductivities*. Journal of Computational Physics, 221(2):799–804, 2007.

[28] Meel, J. van, Arnold, A., Frenkel, D., Portegies Zwart, S., and Belleman, R.: *Harvesing graphics power for MD simulations*. Molecular Simulation, 34(3):259–266, 2008.

[29] Balevic, A., Rockstroh, L., Tausendfreund, A., Patzelt, S., Goch, G., and Simon, S.: *Accelerating simulations of light scattering based on finite-difference time-domain method with general purpose GPUs*. In 11th IEEE International Conference on Computational Science and Engineering, Sao Paulo, Brazil, July 16-18 2008.

[30] Dziekonski, A., Sypek, P., Kulas, L., and Mrozowski, M.: *Implementation of matrix-type FDTD algorithm on a graphics accelerator*. In 17th International Conference on Microwaves, Radar and Wireless Communications, Wroclaw, Poland, May 19-21 2008.

[31] Klöckner, A., Warburton, T., Bridge, J., and Hesthaven, J. S.: *Nodal discontinuous Galerkin methods on graphics processors*. Journal of Computational Physics, 228(21):7863–7882, 2009.

[32] Wong, H., Wong, U., Feng, X., and Tang, Z.: *Efficient magnetohydrodynamic simulations on Graphics Processing Units with CUDA*. Computer Physics Communications, 182(10):2132–2160, 2011.

[33] Cecka, C., Lew, A., and Darve, E.: *Assembly of finite element methods on graphics processors*. International Journal for Numerical Methods in Engineering, 85(5):640–669, 2011.

[34] Komatitsch, D., Michéa, D., and Erlebacher, G.: *Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA*. Journal of Parallel & Distributed Computing, 69(5):451–460, 2009.

[35] Harris, M., Coombe, G., Scheuermann, T., and Lastra, A.: *Physically–based visual simulation on graphics hardware*. In SIGGRAPH 2002, San Antonio, Texas, July 21–26 2002.

[36] Harris, M.: *Fast fluid dynamics simulation on the GPU*. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, chapter 38, pages 637–665. Addison-Wesley Professional, 2004.

[37] Liu, Y., Liu, X., and Wu, E.: *Real-time 3D fluid simulation on GPU with complex obstacles*. In Computer Graphics and Applications, 12th Pacific Conference, Seoul, Korea, October 6–8 2004.

[38] Goodnight, N., Woolley, C., Lewin, G., Luebke, D., and Humphreys, G.: *A multigrid solver for boundary value problems using programmable graphics hardware*. In SIGGRAPH 2003, San Diego, California, July 27–31 2003.

[39] Cohen, J. and Molemaker, M.: *A fast double precision CFD code using CUDA*. In Parallel Computational Fluid Dynamics Conference, California, May 18–22 2009.

[40] Crane, K., Llamas, I., and Tariq, S.: *Real-time simulation and rendering of 3D fluids*. In *GPU Gems 3*, chapter 30, pages 633–676. Addison-Wesley Professional, 2007.

[41] Li, W., Wei, X., and Kaufman, A.: *Implementing Lattice Boltzmann computation on graphics hardware*. The Visual Computer, 19(7):444–456, 2003.

[42] Z. Fan, W. Li amd, Wei, X., and Kaufman, A.: *GPU-based flow simulation with complex boundaries*. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (GPU Gems)*, chapter 47, pages 747–764. Addison-Wesley Professional, 2005.

[43] Fan, Z., Qiu, F., Kaufman, A., and Yoakum-Stover, S.: *GPU cluster for high performance computing*. In ACM/IEEE conference on Supercomputing, Pittsburgh, Pennsylvania, November 6-12 2004.

[44] Tolke, J. and Krafczyk, M.: *TeraFLOP computing on a desktop PC with GPUs for 3D CFD*. International Journal of Computational Fluid Dynamics, 22(7):443–456, 2008.

[45] Kuznik, F., Obrecht, C., Rusaouen, G., and Roux, J.: *LBM based flow simulation using GPU computing processor*. Computers & Mathematics with Applications, 59(7):2380–2392, 2010.

[46] Hagen, T.R., Lie, K.A., and Natvig, J.R.: *Solving the Euler equations on Graphics Processing Units*. Computational Science, 3994(2):220–227, 2006.

[47] Brandvik, T. and Pullan, G.: *Acceleration of a two-dimensional Euler flow solver using commodity graphics hardware*. Journal of Mechanical Engineering Science, 221(12):1745–1748, 2007.

[48] Brandvik, T. and Pullan, G.: *Acceleration of a 3D Euler solver using commodity graphics hardware*. AIAA Paper 2008-607, 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 7–10 2008.

[49] Tutkun, B. and Edis, F.: *A GPU application for high-order compact finite difference scheme*. In 22nd International Conference on Parallel CFD 2010, Kaohsiung, Taiwan, May 17–21 2010.

[50] Göddeke, D., Buijssen, S., Wobker, H., and Turek, S.: *GPU acceleration of an unmodified parallel finite element Navier-Stokes solver*. In High Performance Computing Simulation Conference, San Diego, California, June 21–24 2009.

[51] Elsen, E., LeGresley, P., and Darve, E.: *Large calculation of the flow over a hypersonic vehicle using a GPU*. Journal of Computational Physics, 227(24):10148–10161, 2008.

[52] Corrigan, A., Camelli, F., Löhner, R., and Wallin, J.: *Running unstructured grid based CFD solvers on modern graphics hardware*. AIAA Paper 2009-4001, 19th AIAA Computational Fluid Dynamics, San Antonio, Texas, USA, June 22-25 2009.

[53] Spalart, P. and Allmaras, S.: *A one-equation turbulence model for aerodynamic flows*. AIAA Paper 1992-439, 30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 6–9 1992.

[54] Wong, M., Wong, T., and Fok, K.: *Parallel evolutionary algorithms on Graphics Processing Unit*. In IEEE Congress on Evolutionary Computation Conference, Edinburgh, UK, September 2–5 2005.

[55] Fok, K., Wong, T., and Wong, M.: *Evolutionary computing on consumer graphics hardware*. Intelligent Systems, IEEE, 22(2):69–78, 2007.

[56] Yu, Q., Chen, C., and Pan, Z.: *Parallel genetic algorithms on Programmable Graphics Hardware*. In 1st international Conference on Advances in Natural Computation, Changsha, China, August 27-29 2005.

[57] Asouti, V., Trompoukis, X., Kampolis, I., and Giannakoglou, K.: *Unsteady CFD computations using vertex–centered finite volumes for unstructured grids on Graphics Processing Units*. International Journal for Numerical Methods in Fluids, 67(2):232–246, May 2011.

[58] Trompoukis, X., Asouti, V., Kampolis, I., and Giannakoglou, K.: *CUDA implementation of vertex–centered, finite volume CFD methods on unstructured grids with flow control applications.* In *GPU Computing Gems*, chapter 17, pages 207–224. Addison-Wesley Professional, 2011.

[59] Asouti, V., Kontoleontos, E., Trompoukis, X., and Giannakoglou, K.: *Shape optimization using the one–shot adjoint technique on Graphics Processing Units.* In 7th GRACM International Congress on Computational Mechanics Conference, Athens, Greece, 30 June-2 July 2011.

[60] Dongarra, J., Geist, A., Beguelin, A., Jiang, W., Manchek, R., and Sunderam, V.: *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Network Parallel Computing (Scientific and Engineering Computation).* The MIT Press, 1994.

[61] Gropp, W., Lusk, E., and Thakur, R.: *Using MPI-2: Advanced Features of the Message Passing Interface (Scientific and Engineering Computation).* The MIT Press, 1999.

[62] Brandvik, T. and Pullan, G.: *An accelerated 3D Navier–Stokes solver for flows in turbomachines.* Journal of Turbomachinery, 133(2):619–629, 2011.

[63] Jacobsen, D., Thibault, J., and Senocak, I.: *An MPI-CUDA implementation for massively parallel incompressible flow computations on multi-GPU clusters.* AIAA Paper 2010-522, 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, Orlando, Florida, USA, January 4-7 2010.

[64] Xian, W. and Takayuki, A.: *Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster.* Parallel Computing, 37(9):521–535, 2011.

[65] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.: *Multi-GPU implementation of the lattice Boltzmann method.* International Journal of High Performance Computing Applications, 25(3):295–303, 2011.

[66] Obrecht, C., Kuznik, F., Tourancheau, B., and Roux, J.: *Multi-GPU implementation of a hybrid thermal lattice Boltzmann solver using the TheLMA framework.* Computers & Fluids, 2012, in press.

[67] Appleyard, J. and Drikakis, D.: *Higher-order CFD and interface tracking methods on highly-parallel MPI and GPU systems.* Computers & Fluids, 46(1):101–105, 2011.

[68] Griebel, M. and Zaspel, P.: *A multi-GPU accelerated solver for the three-dimensional two-phase incompressible Navier-Stokes equations.* Computer Science - Research and Development, 25(1-2):65–73, 2010.

[69] Zaspel, P. and Griebel, M.: *Solving incompressible two-phase flows on multi-GPU clusters.* Computers & Fluids, 2012, in press.

[70] Thibault, J. and Senocak, I.: *CUDA implementation of a Navier–Stokes solver on multi–GPU desktop platforms for incompressible flows.* AIAA Paper 2009-758, 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando, Florida, USA, January 5-8 2009.

[71] Thibault, J. and Senocak, I.: *Accelerating incompressible flow computations with a Pthreads-CUDA implementation on small-footprint multi-GPU platforms.* Journal of Supercomputing, 59(2):693–719, 2012.

[72] *POSIX threads programming.* `https://computing.llnl.gov/tutorials/pthreads/`.

[73] Lu, F., Song, J., Yin, F., and Zhu, X.: *Performance evaluation of hybrid programming patterns for large CPU/GPU heterogeneous clusters.* Computer Physics Communications, 183(6):1172–1181, 2012.

[74] *The OpenMP API specification for parallel programming.* `http://openmp.org/wp/`.

[75] Chapman, B., van der Jost, G., Pas, R., and Kuck, D.: *Using OpenMP: Portable shared memory programming.* The MIT Press, 2007.

[76] Engelson, V., Fritzson, D., and Fritzson, P.: *Lossless compression of high-volume numerical data from simulations.* In Data Compression Conference, Snowbird, Utah, March 28–30 2000.

[77] Ratanaworabhan, P., Ke, J., and Burtscher, M.: *Fast lossless compression of scientific floating-point data.* In Data Compression Conference, Snowbird, Utah, March 28–30 2006.

[78] Ke, J., Burtscher, M., and Speight, E.: *Runtime compression of MPI messanes to improve the performance and scalability of parallel applications.* In ACM/IEEE Conference on Supercomputing, Pittsburgh, Pennsylvania, November 6–12 2004.

[79] Phillips, J.C., Stone, J.E., and Schulten, K.: *Adapting a message-driven parallel application to GPU-accelerated clusters.* In High Performance Computing, Networking, Storage and Analysis International Conference, Austin, Texas, November 15–21 2008.

[80] Shimokawabe, T., Aoki, T., Muroi, C., Ishida, J., Kawano, K., Endo, T., Nukada, A., Maruyama, N., and Matsuoka, S.: *An 80-fold speedup, 15.0 TFlops, full GPU acceleration of non-hydrostatic weather model ASUCA production code.* In High Performance Computing, Networking, Storage and Analysis International Conference, New Orleans, Los Angeles, November 13–19 2010.

[81] Micikevicius, P.: *3D finite difference computation on GPUs using CUDA*. In 2nd Workshop on General Purpose Processing on Graphics Processing Units, Washington, D.C., March 8 2009.

[82] Abdelkhalek, R., Calandra, H., Coulaud, O., Roman, J., and Latu, G.: *Fast seismic modeling and reverse time migration on a GPU cluster*. In High Performance Computing, Networking, Storage and Analysis International Conference, Portland, Oregon, June 21-24 2009.

[83] Komatitsch, D., Erlebacher, G., Göddeke, D., and Michéa, D.: *High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster*. Journal of Computational Physics, 229(20):7692–7714, 2010.

[84] Komatitsch, D.: *Fluid–solid coupling on a cluster of GPU graphics cards for seismic wave propagation*. Comptes Rendus Mécanique, 339(2–3):125–135, 2011.

[85] Lu, F., Song, J., Cao, X., and Zhu, X.: *CPU/GPU computing for long-wave radiation physics on large GPU clusters*. Computers & Geosciences, 41(3):47–55, 2012.

[86] Zhang, Y., Mueller, F., Cui, X., and Potok, T.: *Data-intensive document clustering on Graphics Processing Unit (GPU) clusters*. Journal of Parallel and Distributed Computing, 71(2):211–224, 2011.

[87] *AMD ATI close to the metal (CTM) guide*. `http://ati.amd.com/companyinfo/researcher/documents/ATI_CTM_Guide.pdf`.

[88] Mark, W., Steven, R., Kurt, G., Mark, A., and Kilgard, J.: *Cg: A system for programming graphics hardware in a C-like language*. ACM Transactions on Graphics, 22(3):896–907, 2003.

[89] *Opengl shading language*. `http://www.opengl.org/documentation/glsl/`.

[90] *High Level Shading Language*. `http://msdn.microsoft.com/en-us/library/windows/desktop/bb509561(v=vs.85).aspx`.

[91] Fritz, N., Lucas, P., and Slusallek, P.: *CGiS, a new language for data-parallel GPU programming*. In 9th International Workshop for Vision, Modeling, and Visualization, Stanford, California, November 16–18 2004.

[92] *BrookGPU homepage*. `http://graphics.stanford.edu/projects/brookgpu`.

[93] Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., and Hanrahan, P.: *Brook for GPUs: stream computing on Graphics Hardware*. Transactions on Graphics, 23(3):777–786, 2004.

[94] Fatahalian, K., Horn, D.R., Knight, T.J., Leem, L., Houston, M., Park, J.Y., Erez, M., Ren, M., Aiken, A., Dally, W.J., and Hanrahan, P.: *Sequoia: Programming the memory hierarchy.* In High Performance Computing, Networking, Storage and Analysis International Conference, Tampa, Florida, November 11-17 2006.

[95] McCormick, P., Inman, J., Ahrnes, J., Mohd-Yusof, J., Roth, G., and Cummins, S.: *Scout: a data-parallel programming language for graphics processors.* Parallel Computing, 33(10–11):648–662, 2007.

[96] Tarditi, D., Puri, S., and Oglesby, J.: *Accelerator: Using data parallelism to program GPUs for general purpose uses.* Computer & Information Science, 34(5):325–335, 2006.

[97] Proudfoot, K., Mark, W., Tzvetkov, S., and Hanrahan, P.: *A real time procedural shading system for programmable graphics hardware*, August 12–17 2001.

[98] Davidson, A.: *Using a Graphics Processor Unit (GPU) for feature extraction from turbulent flow datasets.* In 20th National Conference on Undergraduate Research, California, April 6–8 2006.

[99] *NVIDIA CUDA homepage.* `http://developer.nvidia.com/what-cuda`.

[100] *OpenCL homepage.* `http://www.khronos.org/opencl`.

[101] Stone, J.E., Gohara, D., and Shi, Guochun: *OpenCL: a parallel programming standard for heterogeneous computing systems.* Computing in Science Engineering, 12(3):66–73, 2010.

[102] M. Harvey, G. De Fabritiis: *Swan: A tool for porting CUDA programs to OpenCL.* Computer Physics Communications, 182(4):1093–1099, 2011.

[103] Du, P., Weber, R., Luszczek, P., Tomov, S., Peterson, G., and Dongarra, J.: *From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming.* Parallel Computing, 2011.

[104] Weber, R., Gothandaraman, A., Hinde, R.J., and Peterson, G.D.: *Comparing hardware accelerators in scientific applications: A case study.* Parallel & Distributed Systems, IEEE Transactions, 22(1):58–68, 2011.

[105] *GPU-Accelerated Libraries, webpage.* `http://developer.nvidia.com/gpu-accelerated-libraries`.

[106] Barrachina, S., Castillo, M., Igual, F.D., Mayo, R., and Quintana-Orti, E.S.: *Evaluation and tuning of the level 3 CUBLAS for graphics processors.* In International Symposium on Parallel and Distributed Processing with Applications, Sydney, Australia, December 10–12 2008.

[107] Fatahalian, K., Sugerman, J., and Hanrahan, P.: *Understanding the efficiency of GPU algorithms for matrix-matrix multiplication.* In ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics hardware, Grenoble, France, August 29-30 2004.

[108] Natarajan, R.: *Finite element applications on a shared-memory multiprocessor: Algorithms and experimental results.* Journal of Computational Physics, 94(2):352 – 381, 1991.

[109] Farhat, C. and Crivelli, L.: *A general approach to nonlinear FE computations on shared-memory multiprocessors.* Computer Methods in Applied Mechanics and Engineering, 72(2):153 – 171, 1989.

[110] Ryoo, S., Rodrigues, Ch. I., Baghsorkhi, S., Stone, S., Kirk, D., and Hwu, W.: *Optimization principles and application performance evaluation of a multithreaded GPU using CUDA.* In 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming Conference, Salt Lake City, Utah, February 20-23 2008.

[111] Govindaraju, N., Larsen, S., Gray, J., and Manocha, D.: *A memory model for scientific algorithms on graphics processors.* In ACM/IEEE Conference on Supercomputing, Tampa, Florida, November 11–17 2006.

[112] Löhner, R.: *Applied CFD Techniques: An Introduction Based on Finite Element Methods.* Wiley, 2008.

[113] Cuthill, E. and McKee, J.: *Reducing the bandwidth of sparse symmetric matrices.* In 24th ACM national conference, 1969.

[114] *GeForce GTX 280, webpage.* `http://www.nvidia.com/object/product_geforce_gtx_280_us.html`.

[115] *GeForce GTX 285, webpage.* `http://www.nvidia.com/object/product_geforce_gtx_285_us.html`.

[116] *Tesla, webpage.* `http://www.nvidia.com/object/preconfigured-clusters.html`.

[117] White, F.: *Viscous fluid flow.* McGraw-Hill International Editions, 1991.

[118] Κουμπογιάννης, Δ.: *Αριθμητική επίλυση των εξισώσεων Navier–Stokes με χρήση μη–δομημένων πλεγμάτων σε περιβάλλον παράλληλης επεξεργασίας.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 1998.

[119] Koubogiannis, D., Athanasiadis, A., and Giannakoglou, K.: *One– and two–equation turbulence models for the prediction of complex cascade flows using unstructured grids.* Computers & Fluids, 32(3):403–430, 2003.

[120] Roe, P.: *Approximate Riemann solvers, parameter vectors, and difference schemes*. Journal of Computational Physics, 43(2):357–372, 1981.

[121] van Leer, B.: *Flux vector splitting for the Euler equations*. In 8th International Conference on Numerical Methods in Fluid Dynamics, Aachen, Germany, June 28-July 2 1982.

[122] Barth, T. and Jespersen, D.: *The design and application of upwind schemes on unstructured meshes*. AIAA Paper 1989-366, 27th Aerospace Sciences Meeting, Reno, Nevada, USA, January 9-12 1989.

[123] Πάππου, Θ.: *Ανάπτυξη αριθμητικής μεθοδολογίας τεχνητής συμπιεστότητας για τον υπολογισμό μη μόνιμων ροών σε κινούμενα όρια*. Διδακτορική διατριβή, Εργαστήριο Αεροδυναμικής, Ε.Μ.Π., Αθήνα, 1998.

[124] Sanders, J. and Kandrot, E.: *CUDA by example: An introduction to General–Purpose GPU Programming*. Addison–Wesley, 2010.

[125] Farber, R.: *CUDA application design and development*. Morgan Kaufmann, 2012.

[126] *GT200 architecture, webpage*. `http://www.anandtech.com/show/2549/2`.

[127] *Fermi architecture, webpage*. `http://www.nvidia.com/object/fermi_architecture.html`.

[128] Schroeder, T.: *Peer–to–Peer and Unified Virtual Addressing*. CUDA Webinar, August 2011.

[129] Ασούτη, Β.: *Μέθοδοι αεροδυναμικής ανάλυσης και σχεδιασμού για ροές υψηλών και χαμηλών ταχυτήτων, σε πολυεπεξεργαστικό περιβάλλον*. Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2009.

[130] Ζερβογιάννης, Θ.: *Μέθοδοι βελτιστοποίησης στην αεροδυναμική και τις στροβιλομηχανές με χρήση συζυγών τεχνικών, υβριδικών πλεγμάτων και του ακριβούς εσσιανού μητρώου*. Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2011.

[131] Καμπόλης, Ι.: *Πολυεπίπεδοι, πολυεπεξεργαστικοί αλγόριθμοι αεροδυναμικής βελτιστοποίησης στις στροβιλομηχανές*. Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2009.

[132] *CUDA C SDK code samples*. `http://developer.nvidia.com/cuda-cc-sdk-code-samples`.

[133] *Family of graph and hypergraph partitioning software*. `http://glaros.dtc.umn.edu/gkhome/views/metis`.

[134] *Mellanox OFED GPUDirect.* `http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=116&menu_section=34`.

[135] Γιώτης, Α.: *Χρήση εξελικτικών τεχνικών, υπολογιστικής ευφυίας και μεθόδων υπολογιστικής ρευστομηχανικής στη βελτιστοποίηση–αντίστροφη σχεδίαση πτερυγώσεων στροβιλομηχανών, μέσω παράλληλης επεξεργασίας.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2003.

[136] Λαμπρόπουλος, Ν.: *Τεχνικές πολυπλέγματος σε μη–δομημένα πλέγματα για την αριθμητική επίλυση πεδίων ροής στις στροβιλομηχανές, σε πολυεπεξεργαστικό περιβάλλον.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2007.

[137] Denton, J., Hirch, G., and Meauzè, Ch.: *Analytical test cases for cascades.* AGARD-AR-275, 1990.

[138] Λιακόπουλος, Π.: *Γένεση μη-δομημένων πλεγμάτων και διαχείρισή τους σε μεθόδους ανάλυσης και βελτιστοποίησης συνιστωσών στροβιλομηχανών και εφαρμογές, αξιοποιώντας τεχνολογίες πλέγματος (Grid computing).* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2008.

[139] Deck, S.: *Numerical simulation of transonic buffet over a supercritical airfoil.* AIAA Journal, 43(7):1556–1566, 2005.

[140] Thierry, M. and Coustols, E.: *URANS computations of shock–induced oscillations over 2D rigid airfoil: Influence of test section geometry.* Flow, Turbulence and Combustion, 74(4):331–354, 2005.

[141] Thierry, M. and Coustols, E.: *Numerical prediction of shock induced oscillations over a 2D airfoil: influence of tutbulence modeling and test section walls.* International Journal of Heat & Fluid Flow, 27(4):661–670, 2006.

[142] Brunet, V.: *Computational study of buffet phenomenon.* AIAA Paper 2003-3679, 21st AIAA Applied Aerodynamics Conference, Orlando, Florida, USA, June 23-26 2003.

[143] Kourta, A., Petit, G., Courty, J., and Rosenblum, J.: *Buffeting in transonic flow prediction using time–dependent turbulence model.* International Journal for Numerical Methods in Fluids, 49(2):171–182, 2005.

[144] *S3 transonic wind–tunnel.* `http://www.onera.fr/dafe-en/transonic-wind-tunnel-s3/`.

[145] Jacquin, L., Molton, P., Deck, S., Maury, B., and Soulevant, D.: *An experimental study of shock oscillation over a transonic supercritical profile.* AIAA Paper 2005-4902, 35th AIAA Fluid Dynamics Conference and Exhibit, Toronto, Ontario, USA, June 6–9 2005.

[146] Rao, D. and Karyia, T.: *Boundary–layer submerged vortex generators for separation control - an exploratory study.* AIAA Paper 1988-3546, 1st AIAA, ASME, SIAM, and APS, National Fluid Dynamics Congress, Cincinnati, Ohio, USA, July 25-28 1988.

[147] Lin, J.: *Control of turbulent boundary layer separation using micro–vortex generators.* AIAA Paper 1999-3404, 30th AIAA Fluid Dynamics Conference, Norfolk, Virginia, USA, June 28-July 1 1999.

[148] Godart, G. and Stanislas, M.: *Control of decelerating boundary layer. part1: Optimization of passive vortex generators.* Aerospace Science & Technology, 10(3):181–191, 2006.

[149] *Wind tunnels at NASA Langley Research Center.* `http://www.nasa.gov/centers/langley/news/factsheets/windtunnels.html`.

[150] Shan, H., Jiang, L., Liu, C., Love, M., and Maines, B.: *Numerical study of passive and active flow separation control over a NACA0012 aifoil.* Computers & Fluids, 37(8):975–992, 2008.

[151] Koike, M., Nagayoshi, T., and Hamamoto, N.: *Research on aerodynmic drag reduction by vortex generators.* Technical report, 2004.

[152] Kim, S. and Kim, C.: *Separation control on NACA23012 using synthetic jet.* Aerospace Science & Technology, 13(4–5):172–182, 2009.

[153] Raghunathan, S.: *Passive control of shoch–boundary layer interaction.* Progress in Aerospace Sciences, 25(3):271–296, 1988.

[154] Bruneau, C. and Mortazavi, I.: *Numerical modelling and passive flow control using porous media.* Cumputers & Fluids, 37(5):488–498, 2008.

[155] Bruneau, C., Creuse, E., Depeyras, D., Gillieron, P., and Mortazavi, I.: *Coupling active and passive techniques to control the flow past the squer back Ahmed body.* Computers & Fluids, 39(10):1875–1892, 2010.

[156] Kynigalakis, M. and Mathioulakis, D.: *Flow control through pulsating jets.* In 7th National Congress on Mechanics, Hania, Greece, June 24-26 2004.

[157] Hantziarasa, V., Mathioulakis, D., and Kaltsas, G.: *Flow control using time dependent jets.* In 2nd International Conference from Scientific Computing to Computational Engineering, Athens, Greece, July 5–8 2006.

[158] Godart, G., Foucaut, J., and Stanislas, M.: *Control of decelerating boundary layer. part1: Optimization of slotted jets vortex generators.* Aerospace Science & Technology, 10(3):394–400, 2006.

[159] Godart, G. and Stanislas, M.: *Control of decelerating boundary layer. part1: Optimization of round jets vortex generators.* Aerospace Science & Technology, 10(3):455–464, 2006.

[160] Delery, J.: *Shock wave/turbulent boundary layer interaction and its control.* Progress in Aerospace Sciences, 22(1–2):209–280, 1985.

[161] Lee, C., Hong, G., Ha, Q., and Mallinson, S.: *A piezoelectrically actuated micro synthetic jet for active flow control.* Sensors & Actuators, 108(1–3):168–174, 2003.

[162] Ekaterinaris, J.: *Prediction of active flow control performance on airfoils and wings.* Aerospace Science & Technology, 8(5):401–410, 2004.

[163] Skamnakis, D. and Papailiou, K.: *Flow stability analysis and excitation using pulsating jets.* Comptes Rendus Méchanique, 333(8):628–635, 2005.

[164] You, D. and Moin, P.: *Active control of flow separation over an airfoil using synthetic jets.* Journal of Fluids & Structures, 24(8):1349–1357, 2008.

[165] Trompoukis, X., Asouti, V., Zervogiannis, T., and Giannakoglou, K.: *CFD analysis and parametric study–optimization of suction–blowing flow control techniques.* In 6th GRACM International Congress on Computational Mechanics Conference, Thessaloniki, Greece, June 19-21 2008.

[166] Huang, L., Huang, P., and LeBeau, R.: *Numerical study of blowing and suction control mechanism on NACA0012 airfoil.* Journal of aircraft, 41(5):1005–1013, 2004.

[167] Duvigneau, R. and Visolnneau, M.: *Optimization of a synthetic jet actuator for aerodynamic stall control.* Cmputers & Fluids, 35(6):624–638, 2006.

[168] Duvigneau, R., Hey, A., and Visolnneau, M.: *Optimal lacation of a synthetic jet on an airfoil for stall control.* Journal of Fluids Engineering, 129(7):825–833, 2007.

[169] Zymaris, A., Papadimitriou, D., Giannakoglou, K., and Othmer, C.: *Optimal location of suction or blowing jets using the continuous adjoint approach.* In 5th ECCOMAS European Conference on Computational Fluid Dynamics, Lisbon, Portugal, June 14-17 2010.

[170] Huang, L.: *Optimization of blowing and suction control on NACA0012 airfoil using genetic algorithm with diversity control.* PhD thesis, University of Kentucky, Kentucky, 2004.

[171] Yagiz, B., Kandil, O., and Pehlivanoglu, Y.: *Drag minization using active and passive flow control techniques.* Aerospace Science & Technology, 17(1):21–31, 2012.

[172] Palaniappan, K., Sahu, P., Alonso, J., and Jameson, A.: *Design of adjoint based laws for wing flutter control.* AIAA Paper 2009-148, 47th Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Orlando, Florida, USA, January 5-8 2009.

[173] De Breuker, R., Abdalla, M., Milanese, A., and Marzocca, P.: *Optimal control of aeroelastic systems using synthetic jet actuators.* AIAA Paper 2008-1726, 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 16th AIAA/ASME/AHS Adaptive Structures Conference 10t, Schaumburg, Illinois, USA, April 7-10 2008.

[174] Donnel, K., Schober, S., De Breuker, R., and Abdalla, M.: *Active aeroelastic control aspects of an aircraft wing by using synthetic jet actuators: Modeling, simulations, experiments.* In Modeling, Signal Processing, and Control for Smart Structures Conference, San Diego, California, March 19–21 2007.

[175] Xiaoping, X., Xiaoping, Z., Zhou, Z., and Ruijun, F.: *Application of active flow control technique for gust load alleviation.* Chinese Journal of Aeronautics, 24(4):410–416, 2011.

[176] *The EASY (Evolutionary Algorithms SYstem) software.* `http://velos0.ltt.mech.ntua.gr/EASY`.

[177] *Langley Research Center Workshop, CFD validation of synthetic jets and turbulent separation control.* `http://cfdval2004.larc.nasa.gov/`.

[178] Seifert, A. and Pack, L.: *Active flow separation control on wall–mounted hump at high Reynolds numbers.* AIAA Journal, 8(1):10–32, 2002.

[179] Anderson, W. and Venkatakrishnan, V.: *Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation.* Computers & Fluids, 28(4-5):443–480, 1999.

[180] Giles, M. and Pierce, N.: *Improved lift and drag estimates using adjoint Euler equations.* AIAA Paper 1999-3293, 14th Computational Fluid Dynamics Conference, Norfolk, Virginia, USA, June 28–July 1 1999.

[181] Jameson, A.: *Aerodynamic design via control theory.* Journal of Scientific Computation, 3(3):33–260, 1988.

[182] Jameson, A.: *Optimum aerodynamic design using CFD and control theory.* AIAA Paper 1995-1729, 12th AIAA Computational Fluid Dynamics Conference and Open Forum, San Diego, California, USA, June 1995.

[183] Papadimitriou, D. and Giannakoglou, K.: *A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows.* Computers & Fluids, 36(2):325–341, 2007.

[184] Papadimitriou, D. and Giannakoglou, K.: *Total pressure loss minimization in turbomachinery cascades using a new continuous adjoint formulation.* Institution of Mechanical Engineers: Part A Journal of Power and Energy, 221(6):865–872, 2007.

[185] Papadimitriou, D. and Giannakoglou, K.: *Direct, adjoint and mixed approaches for the computation of Hessian in airfoil design problems.* International Journal for Numerical Methods in Fluids, 56(10):1929–1943, 2008.

[186] Papadimitriou, D. and Giannakoglou, K.: *Computation of the Hessian matrix in aerodynamic inverse design using continuous adjoint formulations.* Computers & Fluids, 37(8):1029–1039, 2008.

[187] Farina, M.: *A neural network based generalized response surface multiobjective evolutionary algorithm.* In Congress on Evolutionary Computation, Honolulu, Hawaii, May 12-17 2002.

[188] Giannakoglou, K.: *Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence.* Progress in Aerospace Sciences, 38(1):43–76, 2002.

[189] Emmerich, M., Giannakoglou, K., and Naujoks, B.: *Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels.* IEEE Transactions on Evolutionary Computation, 10(4):421–439, 2006.

[190] Jin, Y., Olhofer, M., and Sendhoff, B.: *A framework for evolutionary optimization with approximate fitness functions.* IEEE Transactions on Evolutionary Computation, 6(5):481–494, 2002.

[191] Kampolis, I., Karangelos, E., and Giannakoglou, K.: *Gradient–assisted radial basis function networks: theory and applications.* Applied Mathematical Modelling, 28(13):197–209, 2004.

[192] Karakasis, M. and Giannakoglou, K.: *On the use of metamodel–assisted, multi–objective evolutionary algorithms.* Engineering Optimization, 38(8):941–957, 2006.

[193] Ong, Y., Lum, K., Nair, P., Shi, D., and Zhang, Z.: *Global convergence of unconstrained and bound constrained surrogate–assisted evolutionary search in aerodynamic shape design.* In Congress on Evolutionary Computation, Canberra, Australia, December 8-12 2003.

[194] Ratle, A.: *Optimal sampling strategies for learning a fitness model.* In Congress on Evolutionary Computation, Washington, D.C., July 6-9 1999.

[195] Kampolis, I. and Giannakoglou, K.: *Distributed evolutionary algorithms with hierarchical evaluation.* Engineering Optimization, 41(11):1037–1049, 2009.

[196] Kampolis, I. and Giannakoglou, K.: *A multilevel approach to single- and multi-objective aerodynamic optimization.* Computer Methods in Applied Mechanics & Engineering, 197(33-40):2963–2975, 2008.

[197] Kampolis, I., Zymaris, A., Asouti, V., and Giannakoglou, K.: *Multilevel optimization strategies based on metamodel-assisted evolutionary algorithms, for computationally expensive problems.* In Congress on Evolutionary Computation, Singapore, September 25-28 2007.

[198] Karakasis, M., Koubogiannis, D., and Giannakoglou, K.: *Hierarchical distributed evolutionary algorithms in shape optimization.* International Journal for Numerical Methods in Fluids, 53(3):455–469, 2007.

[199] Kampolis, I., Papadimitriou, D., and Giannakoglou, K.: *Evolutionary optimization using a new radial basis function network and the adjoint formulation.* Inverse Problems in Science and Engineering, 14(4):397–410, 2006.

[200] Knowles, J. and Corne, D.: *M-PAES: A memetic algorithm for multiobjective optimization.* In Congress on Evolutionary Computation, La Jolla, California, July 16-19 2000.

[201] Muyl, F., Dumas, L., and Herbert, V.: *Hybrid method for aerodynamic shape optimization in automotive industry.* Computers & Fluids, 33(5–6):849–858, 2004.

[202] Haykin, S.: *Neural networks: A comprehensive foundation.* Prentice Hall, New Jersey, USA, 1999.

[203] Giannakoglou, K., Giotis, A., and Karakasis, M.: *Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters.* Inverse Problems in Engineering, 9(4):389–412, 2001.

[204] Τρομπούκης, Ξ.: *Υπολογιστική ανάλυση και παραμετρική διερεύνηση της τεχνικής συνεχούς αναρρόφησης για τον έλεγχο οριακών στρωμάτων.* Διπλωματική Εργασία. Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., 2002.

[205] Zitzler, E., Brockhoff, D., and Thiele, L.: *The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration.* In 4th International Conference on Evolutionary Multi-Criterion Optimization, Matsushima-Sendai, Japan, March 5-8 2007.

[206] Βουτσινάς, Σ.: *Μη-μόνιμη αεροδυναμική, Αεροελαστικότητα, Αεροακουστική.* Εκδόσεις Ε.Μ.Π., 2007.

[207] Bisplinghoff, R., Ashley, H., and Halfman, R.: *Aeroelasticity.* Addison–Wesley, 1995.

[208] Djayapertapa, L. and Allen, C.B.: *Simulation of transonic flutter and active shockwave control.* International Journal of Numerical Methods for Heat & Fluid Flow, 14(4):413–443, 2002.

[209] Dowell, E., Thomas, J., and Hall, K.: *Transonic limit cycle oscillation analysis using reduced order aerodynamic models.* Journal of Fluids & Structures, 19(1):17–27, 2004.

[210] Kholodar, D., Dowell, E., Thomas, J., and Hall, K.: *Limit cycle oscillations of typical airfoil in transonic flow.* Journal of Aircraft, 41(5):1067–1072, 2004.

[211] Alonso, J. and Jameson, A.: *Fully-implicit time–marching aeroelastic solutions.* AIAA Paper 1994-56, 32nd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 10-13 1994.

[212] Kirshman, D.J. and Liu, F.: *Flutter prediction by an Euler method on non–moving cartesian grids with gridless boundary conditions.* Computers & Fluids, 35(6):571–586, 2006.

[213] Bendiksen, O. and Kousen, K.: *Transonic flutter analysis using the Euler equations.* AIAA Paper 1987-911, Dynamics Specialists Conference, Monterey, California, USA, April 9–10 1987.

[214] Gao, C., Liu, F., Luo, S., Yang, S., and Schuster, D.: *Calculation of airfoil flutter by an Euler method with approximate boundary conditions.* AIAA Journal, 43(2):295–305, 2005.

[215] Chen, X., Ge-Cheng, Z., and Hu, Z.: *Flutter prediction based on fully coupled fluid–structural interactions.* AIAA Paper 2004-2240, 9th National Turbine High Cycle Fatigue Conference, Pinehurst, North Carolina, USA, March 16–19 2004.

[216] Bohbot, J., Garnier, J., Toumit, S., and Darracq, D.: *Computation of the flutter boundary of an airfoil with a parallel Navier-Stokes solver.* AIAA Paper 2001-572, 39th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 8-11 2001.

[217] Baldwin, B. and Lomax, H.: *Thin-layer approximation and algebraic model for separated turbulent flows.* AIAA Paper 1978-257, 16th American Institute of Aeronautics and Astronautics, Aerospace Sciences Meeting, Huntsville, Alabama, USA, January 16-18 1978.

[218] Yang, S. and Lee, I.: *Aeroelastic analysis for flap of airfoil in transonic flow.* Computers & Structures, 61(3):421–430, 1996.

[219] Zhang, W. and Ye, Z.: *Effect of control surface on airfoil flutter in transonic flow.* Acta Astronautica, 66(7–8):999–1007, 2010.

[220] Djayapertapa, L., Allen, C.B., and Fiddes, S.P.: *Two–dimensional transonic aeroservoelastic computattions in the time domain.* International Journal of Numerical Methods in Engineering, 52(12):1355–1377, 2001.

[221] Lee, B., Liu, L., and Chung, K.: *Airfoil motion in subsonic flow with strong cubic nonlienar restoring forces.* Journal of Sound & Vibration, 281(3–5):699–717, 2005.

[222] Allen, C., Taylor, N., Fenwick, C., Gaitonde, A., and Jones, D.: *A comparison of full non–linear and reduced order aerodynamic models in control law design using a two–dimensional aerofoil model.* International Journal for Numerical Methods in Engineering, 64(12):1628–1648, 2005.

[223] Librescu, L., Na, S., Marzocca, P., Chung, C., and Kwak, M.: *Active aeroelastic control of a 2D wing–flap systems operating in an incompressible flowfiel and impacted by a blast pulse.* Journal of Sound & Vibration, 283(3–5):685–706, 2005.

[224] Librescu, L., Marzocca, P., and Silva, W.: *Aeroelasticity of 2D lifting surfaces with time–delayed feedback control.* Journal of Fluids & Sructures, 20(2):197–215, 2005.

[225] Baxevanou, C., Chaviaropoulos, P., Voutsinas, S., and Vlachos, N.: *Evaluation study of a Navier-Stokes CFD aeroelastic model of wing turbine airfoils in classical flutter.* Journal of Wind Engineering & Industrial Aerodynamics, 96(8–9):1425–1443, 2008.

[226] Lynch, D.: *Unified approach to simulation on deforming elements with application to phase change problems.* Journal of Computational Physics, 47(3):387–411, 1982.

[227] Lynch, D. and ONeil, K.: *Elastic grid deformation for moving boundary problems in two space dimensions.* In 3rd International Conference, Mississippi University, Oxford, May 19-23 1980.

[228] Stein, K., Tezduyar, T., and Benney, R.: *Mesh moving techniques for fluid-structure interactions with large displacements.* Journal of Applied Mechanics, 70(1):58–63, 2003.

[229] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M.: *Torsional springs for two-dimensional dynamic unstructured fluid meshes.* Computer Methods in Applied Mechanics & Engineering, 163(1–4):231–245, 1998.

[230] Batina, J.: *Unsteady Euler algorithms with unstructured dynamic mesh for conplex-aircraft aerodynamic analysis.* AIAA Journal, 29(3):327–333, 200.

[231] Degand, C. and Farhat, C.: *A three-dimensional torsional spring analogy method for unstructured dynamic meshes.* Computers & Structures, 80(3–4):305–316, 2002.

[232] Zhao, Y. and Forhad, A.: *A general method for simulation of fluid flows with moving and compliant boundaries on unstructured grids.* Computer Methods in Applied Mechanics & Engineering, 192(39–40):4439–4466, 2003.

[233] De Boer, A., Van der Schoot, M., and Bijl, H.: *Mesh deformation based on radial basis function interpolation.* Computers & Structures, 85(11-14):784–795, 2007.

[234] Rendall, T. and Allen, C.: *Unified fluid-structure interpolation and mesh motion using radial basis functions.* International Journal for Numerical Methods in Engineering, 74(10):1519–1559, 2008.

[235] Braess, H. and Wriggers, P.: *Arbitrary Lagrangian Eulerian finite element analysis of free surface flow.* Computer Methods in Applied Mechanics & Engineering, 190(1–2):95–109, 2000.

[236] Diachin, L., Knupp, P., Munson, T., and Shontz, S.: *A comparison of two optimization methods for mesh quality improvement.* Engineering with Computers, 22(2):61–74, 2006.

[237] Landon, R.: *NACA0012 oscillating and transient pitching, compendium of unsteady aerodynamic measurements, data set 3.* AGARD-R-702, 1982.

[238] Isogai, K.: *On the transonic dip mechanism of flutter of a sweptback wing.* AIAA Journal, 17(7):793–795, 1979.

[239] Callaghan, J. and Liebeck, R.: *Some thoughts on the design of subsonic transport aircraft for the 21th century.* SAE Technical Paper 901987, 1990.

[240] Liebeck, R.: *Design of the Blended Wing Body subsonic transport.* Journal od Aircraft, 41(1):10–25, 2004.

[241] Qin, N., Vavalle, A., Moigne, A., Laban, M., Hackett, K., and Weinerfelt, P: *Aerodynamic studies for Blended Wing Body aicraft.* AIAA Paper 2002-5448, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, USA, September 4–6 2002.

[242] Qin, N., Vavalle, A., Moigne, A., Laban, M., Hackett, K., and Weinerfelt, P: *Aerodynamic considerations of Blended Wing Body aircraft.* Progress in Aerospace Sciences, 40(6):321–343, August 2004.

[243] Jianghao, W., Chenfang, C., and Yanlai, Z.: *The changes in structural and flight safety due to flap design of Blended Wing Body civil aircraft.* In 2nd International Symposium on Aircraft Airworthiness, Beijing, China, October 26-28 2011.

[244] *NASA's X-48B/Blended Wing Body page.* `http://www.nasa.gov/topics/aeronautics/features/bwb_main.html`.

[245] Hosder, S., Schetz j., Grossman, B., and Mason, W.: *Airframe noise modelling appropriate for multidisciplinary design and optimization.* AIAA Paper 2004-698, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 5–8 2004.

[246] Leifsson, L., Ko, A., Mason, W., Schetz, J., Grossman, B., and Haftka, R.: *Multidisciplinary design optimization of Blended Wing Body transport aircraft with distributed propulsion.* Aerospace Science and Technology, 2011 in press.

[247] Engels, H., Becker, W., and Morris, A.: *Implementation of a mulit–level optimisation methodology within the e–design of a Blended Wing Body.* Aerospace Science & Technology, 8(2):145–153, 2004.

[248] Mukhopadhyay, V.: *Blended Wing Body (BWB) fuselage structural design for weight reduction.* AIAA Paper 2005-2349, 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Austin, Texas, USA, April 18-21 2005.

[249] Wildschek, A., Stroscher, F., and Hanis, T.: *Fuel management system for cruise performance optimization on a large Blended Wing Body airliner.* In 4th European Conference for Aerospace Sciences (EUCASS), St Petersburg, Russia, July 4-8 2011.

[250] *ACFA 2020 - Active Control for Flexible 2020 Aircraft.* `http://www.acfa2020.eu/`.

[251] Καββαδίας, Ι.: *Αεροδυναμική βελτιστοποίηση για τυρβώδεις ροές με χρήση συζυγών μεθόδων και επεξεργαστών καρτών γραφικών.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, σε εξέλιξη.

[252] Τσιάκας, Κ.: *Ανάπτυξη μεθόδων βελτιστοποίησης με χρήση επεξεργαστών καρτών γραφικών και εφαρμογή στις στροβιλομηχανές.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, σε εξέλιξη.

[253] Καββαδίας, Ι.: *Προγραμματισμός επιλύτη 3Δ εξισώσεων ροής ατριβούς ρευστού σε δομημένα πλέγματα, σε κάρτες γραφικών.* Διπλωματική Εργασία. Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., 2011.

[254] Ντανάκας, Γ.: *Επίλυση διδιάστατης ροής σε μη-δομημένα πλέγματα με τη κεντροκυψελική διατύπωση σε επεξεργαστές καρτών γραφικών.* Διπλωματική Εργασία. Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., 2012.

[255] Ζυμάρης, Α.: *Συζυγείς μέθοδοι για το σχεδιασμό μορφών με βέλτιστη αεροδυναμική συμπεριφορά σε στρωτές και τυρβώδεις ροές.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2010.

[256] Παπαδημητρίου, Δ.: *Συζυγείς (Adjoint) διατυπώσεις για την ανάλυση–σχεδίαση πτερυγώσεων στροβιλομηχανών και τη βέλτιστη προσαρμογή πλέγματος με a posteriori εκτίμηση σφάλματος.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2007.

[257] Παπουτσής-Κιαχαγιάς, Ε.: *Συζυγείς μέθοδοι χαμηλού υπολογιστικού κόστους για τυρβώδεις ροές, στην αεροδυναμική βελτιστοποίηση.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, σε εξέλιξη.

[258] Taasan, S., Kuruvila, G., and Salas, M.: *Aerodynamic design and optimization in one-shot.* AIAA Paper 1992-25, 30th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 6–9 1992.

[259] Kuruvila, G., Taasan, S., and Salas, M.: *Airfoil design and optimization by the one-shot method.* AIAA Paper 1995-478, 33rd Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 9–12 1995.

[260] Hazra, S.: *An efficient method for aerodynamic shape optimization.* AIAA Paper 2004-4628, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, USA, August 30-September 1 2004.

[261] Hazra, S., Schulz, V., Brezillon, J., and Gauger, N.: *Aerodynamic shape optimization using simultaneous pseudo-time stepping.* Journal of Computational Physics, 204(1):46–64, 2005.

[262] Hazra, S. and Schulz, V.: *Simultaneous pseudo-time stepping for aerodynamic shape optimization problems with state constraints.* SIAM Journal on Scientific Computing, 28(3):1078–1099, 2006.

[263] Hazra, S.: *Multigrid one-shot method for state constrained aerodynamic shape optimization.* SIAM Journal on Scientific Computing, 30(6):3220–3248, 2008.

[264] Κοντολέοντος, Ε.: *Ανάλυση και βελτιστοποίηση σύνθετων ρευστομηχανικών διατάξεων με χρήση υπολογιστικής ρευστοδυναμικής και εξελικτικών αλγορίθμων.* Διδακτορική διατριβή, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα, 2012.