ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΥΠΟΛΟΓΙΣΤΙΚΗ ΜΗΧΑΝΙΚΗ

ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ ΜΕ ΚΑΡΤΕΣ ΓΡΑΦΙΚΩΝ ΣΤΗΝ ΠΡΟΣΟΜΟΙΩΣΗ ΦΑΙΝΟΜΕΝΩΝ ΔΙΑΒΡΟΧΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΡΗΓΟΡΙΟΥ Α. ΚΑΣΑΠΙΔΗ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΑΝΔΡΕΑΣ Γ. ΜΠΟΥΝΤΟΥΒΗΣ

ΑΘΗΝΑ, ΙΟΥΝΙΟΣ 2016

Περιεχόμενα

Περίληψη

Στόχος αυτής της εργασίας είναι η ανάπτυξη κώδικα για την προσομοίωση φαινομένων διαβροχής πάνω σε αυλακωτές επιφάνειες. Μελετώνται φαινόμενα διαβροχής όπου σταγόνες μεταπίπτουν ανάμεσα σε καταστάσεις ισορροπίας. Οι καταστάσεις ισορροπίας της σταγόνας μπορεί να είναι η υπερυδρόφοβη κατάσταση Cassie-Baxter (CB), και η υδρόφιλη κατάσταση Wenzel (W). Για τον προσδιορισμό των καταστάσεων διαβροχής χρησιμοποιείται η μέθοδος χορδής (String Method, SM) μέσω της οποίας είναι δυνατός ο προσδιορισμός του μονοπατιού ελάχιστης ενέργειας (Minimum Energy Path, MEP). Το MEP επιτρέπει τον υπολογισμό των ενεργειακών φραγμάτων και την αξιολόγηση της υπερυδρόφοβης συμπεριφοράς επιφανειών. Ο προσδιορισμός των ενδιάμεσων καταστάσεων κάθε μετάβασης υπολογίζεται με χρήση μεθόδου αναπαράστασης φάσεων (phase-field method). Χρησιμοποιείται μια τροποποιημένη μέθοδος που βασίζεται στη Cahn-Hilliard διατύπωση των εξισώσεων του πεδίου αναπαράστασης φάσεων ειδικά για φαινόμενα διαβροχής. Η μέθοδος χρησιμοποιεί μετασχηματισμούς Fourier, οι οποίοι μετατρέπουν τις διαφορικές εξισώσεις του προβλήματος σε απλές αλγεβρικές εξισώσεις που επιτρέπουν την ταχεία επίλυσή τους με χρήση καρτών γραφικών (Graphical Processing Units - GPUs). Αναπτύχθηκε κώδικας που εκτελείται σε κάρτες γραφικών χρησιμοποιώντας τη πλατφόρμα CUDA (Compute Unified Device Architecture). Ο εγγενής GPU κώδικας παρουσιάζει δεκαπλάσια επιτάχυνση σε σύγκριση με κώδικα MATLAB που επίσης χρησιμοποιεί κάρτες γραφικών. Παρουσιάζονται τεχνικές λεπτομέρειες της υλοποίησης που οδηγούν σε βελτιστοποίηση και επιτάχυνση του κώδικα. Στη συνέχεια παρουσιάζεται υλοποίηση που συνδυάζει το πρωτόκολλο MPI (Message Passing Interface) για την κλιμάκωση της εκτέλεσης σε πολλαπλές κάρτες γραφικών της υπολογιστικής συστοιχίας. Γίνεται επίλυση προβλημάτων διαβροχής που αφορούν μετάβαση σταγόνων από την CB στη W κατάσταση, αλλά και την απλή κίνηση της σταγόνας πάνω στην επιφάνεια. Ο συνδυασμός αυτών των αναλύσεων δίνει μια σαφή εικόνα για το ενεργειακό τοπίο

των μεταβάσεων και γιαυτό το λόγο εφαρμόζονται για διάφορες περιπτώσεις αυλακωτών επιφανειών. Τέλος γίνεται αξιολόγηση της υδροφοβικότητας των επιφανειών αυτών.

Λέξεις-Κλειδιά: υπερυδρόφοβες αυλακωτές επιφάνειες (superhydrophobic grooved surfaces), CUDA, MATLAB, MPI, επεξεργαστές γραφικών (GPUs), μέθοδος αναπαράστασης φάσεων (phase field method), μέθοδος χορδής (string method), μονοπάτια ελάχιστης ενέργειας (Minimum Energy Paths - MEPs)

Abstract

The purpose of this work is code development for the simulation of wetting phenomena on patterned surfaces. The wetting phenomena that are studied involve wetting transitions between equilibrium states of droplets on grooved surfaces. There are two significant equilibrium states for a droplet on a grooved surface: The Cassie-Baxter (CB) superhydrophobic state and the Wenzel (W) state. The String Method (SM) is used to calculate the Minimum Energy Path (MEP) of the transition (e.g. Cassie-Baxter to Wenzel), which in turn is used to calculate the energy barriers and evaluate the superhydrophobic behavior of surfaces. The calculation of the wetting states is accomplished using a modified phase-field method, that is based on the Cahn-Hilliard formulation. The computational method utilizes the Fast Fourier Transform (FFT) that converts the partial differential equations of the problem into simple algebraic equations, which are readily solved using GPUs (Graphical Processing Units). The developed code uses the CUDA framework for the execution on GPUs. The native CUDA GPU code shows tenfold speedup compared to the MATLAB code that also uses GPUs. A new implementation is presented which combines the MPI (Message Passing Interface) with the CUDA implementation that scales the execution on many GPUs in the available computational cluster. The studied problems involve the CB-W transition and also the lateral displacement of the droplet on the surface. Finally the superhydrophobic behavior of those patterned surfaces is evaluated and analysed.

Keywords: superhydrophobic grooved surfaces, CUDA, MATLAB, MPI, GPUs, Phase Field Method, String Method, Minimum Energy Path (MEP)

Ευχαριστίες

Ευχαριστώ θερμά τον επιβλέποντα της μεταπτυχιακής μου εργασίας, Καθηγητή Ανδρέα Γ. Μπουντουβή, για την εμπιστοσύνη και στήριξη που μου έδειξε. Θέλω επίσης να ευχαριστήσω τον Δρα Γιώργο Πάσχο για τις ενδιαφέρουσες συζητήσεις και την πολύτιμη καθοδήγησή του για την κατανόηση των θεμάτων που πραγματεύεται η εργασία. Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για τη διαρκή της υποστήριξη.

Γρηγόριος Α. Κασαπίδης

Εισαγωγή

Στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη κώδικα για την προσομοίωση φαινομένων διαβροχής, χρησιμοποιώντας επεξεργαστές γραφικών. Η μέθοδος που χρησιμοποιείται, υπολογίζει τα μονοπάτια ελάχιστης ενέργειας (Minimum Energy Paths - MEPs) χρησιμοποιώντας την Cahn-Hilliard διατύπωση μιας μεθόδου αναπαράστασης φάσεων συζευγμένη με την απλοποιημένη μέθοδο χορδής. Χρησιμοποιείται το προγραμματιστικό πλαίσιο CUDA, ενώ γίνεται σύγκριση με κώδικα MATLAB. Η CUDA υλοποίηση συνδυάζεται με το πρωτόκολλο επικοινωνίας MPI, για την παραλληλοποίηση της εκτέλεσης σε υπολογιστικούς κόμβους. Παρουσιάζονται αρκετές τεχνικές λεπτομέρειες των υλοποιήσεων και συγκρίνονται βάσει της απόδοσης και του χρόνου εκτέλεσής τους. Η προτεινόμενη μέθοδος εφαρμόζεται σε προβλήματα διαβροχής αυλακωτών επιφανειών και εξάγονται συμπεράσματα για την υδροφοβικότητα σε σχέση με τα γεωμετρικά τους χαρακτηριστικά.

Κεφάλαιο 1

Περιγραφή Λογισμικού

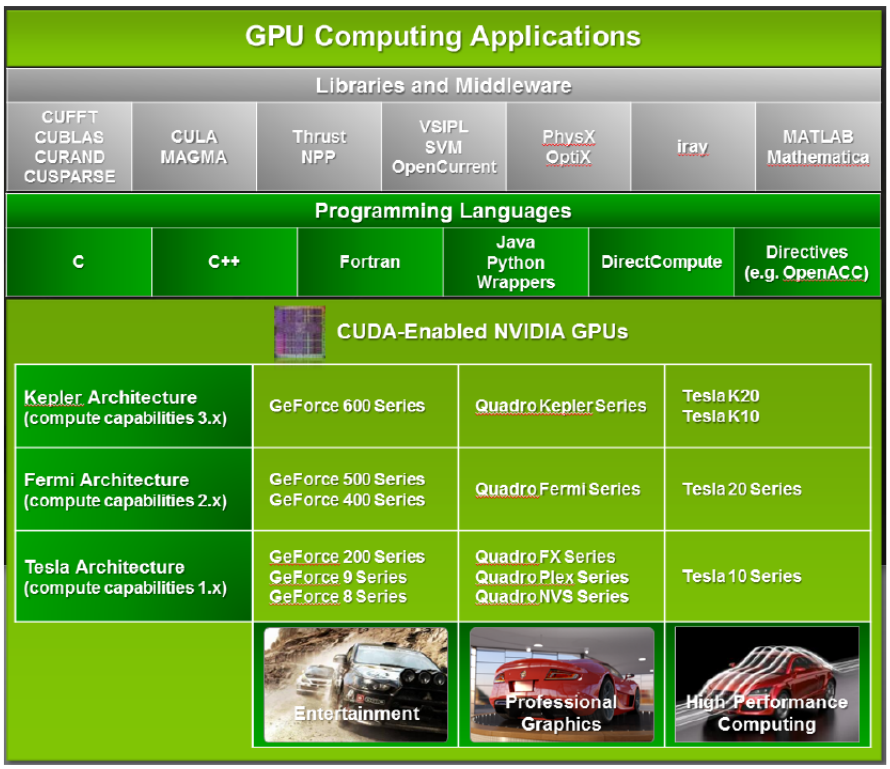
Στο κεφάλαιο αυτό θα περιγραφούν τα υπολογιστικά πακέτα και τα προγραμματιστικά πλαίσια που χρησιμοποιήθηκαν στην παρούσα εργασία. Το πλαίσιο CUDA της NVIDIA, χρησιμοποιείται για την ανάπτυξη κώδικα σε κάρτες γραφικών. Το πλαίσιο MPI, χρησιμοποιείται για την επικοινωνία-συνεργασία εφαρμογών σε κόμβους της υπολογιστικής συστοιχίας. Τέλος γίνεται μια σύντομη αναφορά στο υπολογιστικό πακέτο MATLAB και κυρίως στις δυνατότητές του για υπολογισμούς σε κάρτες γραφικών.

1.1 CUDA

1.1.1 Περιγραφή

Το αρκτικόλεξο CUDA [28] προέρχεται από την πλήρη ονομασία : *Compute Unified Device Architecture*. Είναι ουσιαστικά μια πλατφόρμα παράλληλης επεξεργασίας που δημιουργήθηκε και εφαρμόστηκε πάνω στις κάρτες γραφικών που δημιουργεί η NVIDIA [29]. Με τη βοήθεια της CUDA ο εκάστοτε προγραμματιστής έχει άμεση πρόσβαση σε σύνολο εντολών που αφορούν την κάρτα γραφικών αλλά και σε όλα τα στοιχεία μνήμης της κάρτας.

Η πρώτη παρουσίαση της CUDA έγινε το 2006, και σκοπός της ήταν να επιτρέψει στους προγραμματιστές να επιλύουν σύνθετα προβλήματα, μεγάλης κλίμακας πιο αποδοτικά σε σχέση με την επίλυση στη CPU. Η CUDA υποστηρίζει εξαρχής τη γλώσσα C, ωστόσο με την πάροδο τον χρόνων έχει δοθεί υποστήριξη σε μεγάλο πλήθος γλωσσών μέσω APIs ή μέσω directives. Παράλληλα έχουν δημιουργηθεί πολλά υπολογιστικά πακέτα που χρησιμοποιούν CUDA. Ενδεικτικός είναι ο πίνακας 1.1.



Σχήμα 1.1: CUDA Overview

1.1.2 Προγραμματιστικό μοντέλο CUDA

Threads-Blocks-Grids Το προγραμματιστικό μοντέλο της CUDA είναι βασισμένο στην παράλληλη εκτέλεση κώδικα με τη χρήση νημάτων (*threads*). Τα νήματα στην ουσία είναι τεχνητές, εικονικές (virtual), οντότητες, όπου η καθεμία από αυτές εκτελεί τον κώδικα μιας συγκεκριμένου τύπου συνάρτησης γνωστή ως *kernel*.

Κατά την εκτέλεση του κώδικα υπάρχει δυνατότητα οργάνωσης ενός πλέγματος (*grid*) από νήματα. Ο χρήστης μπορεί να ρυθμίσει ανάλογα με τις ανάγκες του τα πόσα μπλοκ (*blocks*) χρειάζεται εντός του πλέγματος, μάλιστα χρησιμοποιώντας μέχρι και 3 διαστάσεις (μονοδιάστατες, δισδιάστατες, τρισδιάστατες δομές από μπλοκ). Μέσα σε κάθε μπλοκ, ο χρήστης μπορεί να ρυθμίσει επίσης τα πόσα νήματα επιθυμεί να τρέχουν ανά μπλοκ. Σε άμεση αναλογία με το πλέγμα, και η διάσταση του μπλοκ μπορεί να ποικίλει ανάλογα με τις ανάγκες του χρήστη μέχρι και σε 3 διαστάσεις (μονοδιάστατες, δισδιάστατες, τρισδιάστατες δομές από νήματα).

Εδώ βέβαια θα πρέπει να αναφερθεί ότι, το κάθε μπλοκ περιλαμβάνει τον κώδικα ο οποίος θα εκτελεστεί σε έναν φυσικό πυρήνα της GPU. Επομένως υπάρχει φυσικός περιορισμός στον αριθμό των νημάτων τα οποία μπορούν να περιέχονται σε ένα μπλοκ, ο οποίος είναι 1024 για τους τρέχοντες πυρήνες CUDA (CUDA Cores).

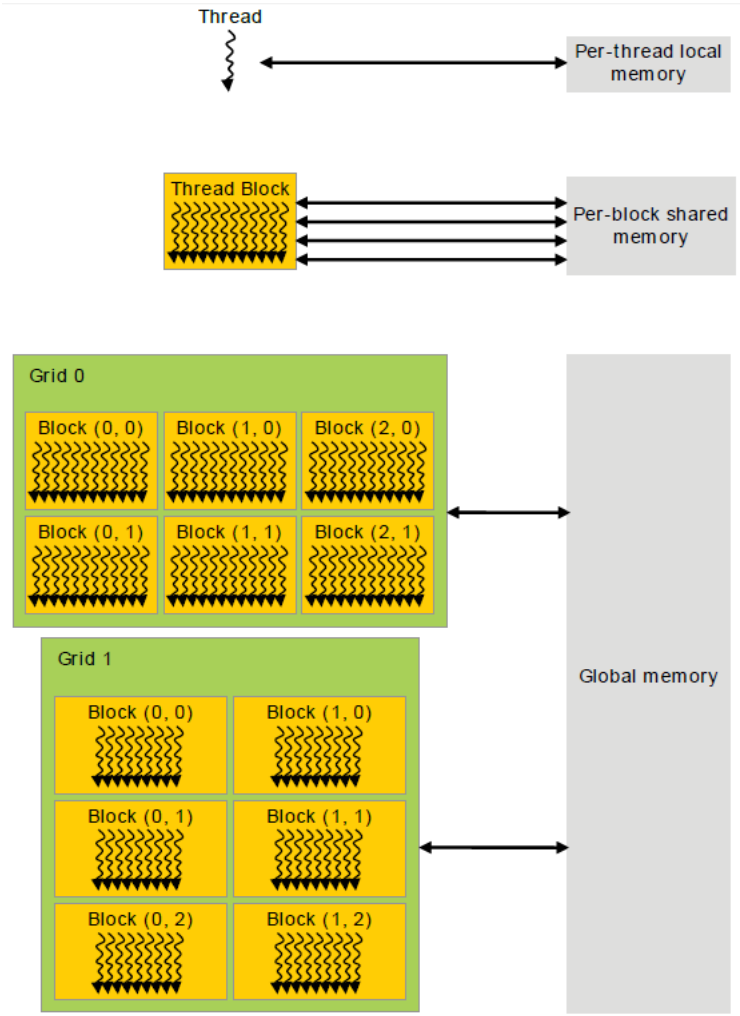
Kernels Οι kernels (συναρτήσεις-πυρήνες) είναι ειδικού τύπου συναρτήσεις τις οποίες περιέχει η CUDA C έναντι της απλής C. Λέγονται έτσι γιατί προορίζονται να εκτελεστούν τόσες φορές, όσα τα συνολικά νήματα που έχουν οριστεί από τον χρήστη (ο αριθμός των νημάτων θα προκύψει ανάλογα με την δομή του πλέγματος, τις διαστάσεις του, αλλά επίσης και των διαστάσεων και της δομής των περιεχομένων στο πλέγμα μπλοκ). Υπάρχουν ειδικά αναγνωριστικά για να ξεχωρίζουν τον ορισμό των συναρτήσεων *kernel*, σε σχέση με τις συμβατικές συναρτήσεις της C.

Κατά την δημιουργία όλου του πλέγματος (και επομένως κατά την δημιουργία όλων των μπλοκ και των νημάτων), δίνονται μοναδικές ταυτότητες (IDs), σε όλα τα *thread* και τα υπόλοιπα στοιχεία. Με αυτό τον τρόπο υπάρχει η δυνατότητα πρόσβασης (όχι μνήμης) σε οποιοδήποτε *thread*, εντός οποιασδήποτε διάστασης οποιουδήποτε μπλοκ, σε οποιαδήποτε διάσταση του πλέγματος.

Τα μπλοκ απαιτείται από το προγραμματιστικό μοντέλο να μπορούν να εκτελούνται ανεξάρτητα το ένα από το άλλο. Πρέπει να είναι δυνατόν να εκτελεστούν με οποιοδήποτε τρόπο σε οποιαδήποτε σειρά, σειριακά ή παράλληλα. Αυτή η απαραίτητη προϋπόθεση επιτρέπει στα μπλοκ να δρομολογούνται προς εκτέλεση σε οποιαδήποτε GPU (που το υποστηρίζει), με οποιοδήποτε αριθμό πυρήνων CUDA. Ο κώδικας προσαρμόζεται ανάλογα με τον αριθμό των πυρήνων (CUDA Cores) της εκάστοτε κάρτας.

Πρόσβαση μνήμης Υπάρχουν συγκεκριμένες κατηγορίες μνήμης στις οποίες έχει πρόσβαση ένα *thread*. Κάθε *thread* έχει την δική του αποκλειστική εσωτερική τοπική μνήμη, *Per-thread local memory*. Η μνήμη αυτή είναι ξεχωριστή για κάθε *thread* και κανένα άλλο *thread* ακόμα και εντός του ίδιου μπλοκ δεν έχει πρόσβαση σε αυτήν. Υπάρχει η κοινή ανά μπλοκ μνήμη *Per-block shared memory*. Σε αυτήν έχουν πρόσβαση όλα τα νήματα εντός του ίδιου μπλοκ. Κανένα εξωτερικό μπλοκ δεν έχει πρόσβαση σε αυτήν. Επίσης αυτή η μνήμη υπάρχει μόνο όσο το μπλοκ "τρέχει". Εφόσον η εκτέλεση ολοκληρωθεί, αυτό το κομμάτι μνήμης απελευθερώνεται έως ότου δεσμευθεί για κάποια άλλη λειτουργία. Τέλος υπάρχει η καθολική μνήμη *global memory* στην οποία έχουν πρόσβαση όλα τα νήματα σε όλα τα μπλοκ του πλέγματος.

Σχηματικά η πρόσβαση της μνήμης στα thread αποδίδεται στο σχήμα 1.2.



Σχήμα 1.2: Πρόσβαση Μνήμης στη CUDA

Συγχρονισμός Πολλές φορές είναι απαραίτητο τα νήματα εντός του ίδιου μπλοκ, που μοιράζονται το ίδιο κομμάτι μνήμης, να μπορούν να συγχρονίζονται έτσι ώστε να υπάρχει συντονισμένη πρόσβαση στη μνήμη. Αυτό έχει προβλεφθεί από το μοντέλο και αυτός ο συγχρονισμός μπορεί να γίνει μέσω εγγενούς συνάρτησης (__syncthreads()), η οποία λειτουργεί σαν φράγμα, το οποίο δεν μπορεί να ξεπεραστεί από κανένα thread, εάν όλα τα νήματα δεν φτάσουν στο σημείο του συγχρονισμού.

Ετερογενής Προγραμματισμός Ένα από τα κύρια χαρακτηριστικά του μοντέλου είναι η υποστήριξη του λεγόμενου *ετερογενούς προγραμματισμού-Heterogeneous Programming*. Αυτό σημαίνει ότι το κάλεσμα των όποιων συναρτήσεων "πυρήνων" (kernels) έχει δημιουργήσει ο χρήστης, συμπεριλαμβάνεται σε ένα γενικότερο σειριακό κώδικα. Με άλλα λόγια ο κύριος κώδικας του προγράμματος τρέχει στον κύριο επεξεργαστή του συστήματος (CPU) και όποτε κληθούν οι kernels αποστέλλονται στην συσκευή (GPU) για εκτέλεση, μαζί με τα όποια δεδομένα ενδέχεται να χρειαστούν. Η GPU στην ουσία δρα σαν συνεπεξεργαστής. Με

αυτό το σκεπτικό καθένας από τους επεξεργαστές έχει πρόσβαση σε διαφορετική μνήμη: Η CPU έχει πρόσβαση στην μνήμη του συστήματος *host memory*, ενώ η GPU στην μνήμη της συσκευής *device memory*. Η διαδικασία που λαμβάνει χώρα κάθε φορά που καλείται η εκτέλεση ενός προγράμματος, είναι πρώτον να μεταφέρονται τα απαραίτητα δεδομένα από τη μνήμη του συστήματος στην μνήμη της κάρτας γραφικών, δεύτερον με έναν δεδομένο διαχωρισμό του πλέγματος, να γίνεται η κατανομή της μνήμης ανά μπλοκ και thread, τρίτον εφόσον η επεξεργασία ολοκληρωθεί, αντιγράφονται τα δεδομένα από την μνήμη της κάρτας (device memory) πάλι πίσω στην (host memory) για περαιτέρω επεξεργασία. Το προγραμματιστικό μοντέλο προβλέπει επίσης τη χρήση ασύγχρονων κλήσεων είτε σε εγγενείς συναρτήσεις της CUDA, είτε σε πυρήνες. Αυτό επιτρέπει την παράλληλη χρήση και της CPU για υπολογισμούς σε συνδυασμό με τους επεξεργαστές της κάρτας γραφικών, προσφέροντας περισσότερες δυνατότητες για επιτάχυνση του κώδικα και γενικά για την μείωση του χρόνου εκτέλεσης.

1.2 MPI

Περιγραφή Το MPI αποτελεί τα αρχικά του *Message Passing Interface*. Αποτελεί ένα σύστημα επικοινωνίας μεταξύ εφαρμογών που βασίζεται στην αποστολή/λήψη μηνυμάτων. Το πρότυπό του, ορίζει τη σύνταξη και τη σημασιολογία μιας πολύ βασικής βιβλιοθήκης από ρουτίνες, οι οποίες είναι διαθέσιμες στον χρήστη για τη κάλυψη οποιουδήποτε τρόπου επικοινωνίας που ενδέχεται να χρειαστεί. Υπάρχουν αρκετές υλοποιήσεις του MPI, πολλές από τις οποίες είναι και ανοιχτού κώδικα (π.χ. MPICH, OpenMPI).

Το MPI μέχρι και σήμερα αποτελεί τη πρώτη επιλογή όσον αφορά την επικοινωνία μεταξύ εφαρμογών, που αφορούν την εκτέλεση ενός παράλληλου προγράμματος σε συστήματα κατανεμημένης μνήμης (distributed memory systems). Μεγάλα υπολογιστικά συστήματα εκτελούν επί το πλείστον τέτοιου είδους εφαρμογές. Οι περισσότερες υλοποιήσεις του MPI, αφορούν τις γλώσσες C, C++, Fortran αλλά και κάθε άλλη γλώσσα η οποία μπορεί να χρησιμοποιήσει τις αντίστοιχες βιβλιοθήκες (C#, Java, Python). Το κύριο πλεονέκτημα του MPI, σε σχέση με όλα τα παλαιότερα συστήματα επικοινωνίας είναι κυρίως η φορητότητά (portability) του και η ταχύτητά του. Έχει σχεδιαστεί-προσαρμοστεί, για να λειτουργεί με όλες τις διαφορετικές αρχιτεκτονικές κατανεμημένης μνήμης. Έτσι εκτός από συστήματα κατανεμημένης μνήμης, το MPI μπορεί να λειτουργήσει αποδοτικά είτε σε συστήματα κοινής μνήμης (shared memory systems) είτε και σε υβριδικά συστήματα (hybrid systems), αξιοποιώντας τα ειδικά χαρακτηριστικά κάθε αρχιτεκτονικής.

Μέχρι σήμερα υπάρχουν διαθέσιμες αρκετές εκδόσεις του πρωτοκόλλου MPI. Η έκδοση *1.3* αποτελεί τη βασική έκδοση που δίνει έμφαση κυρίως στην ανταλλαγή μηνυμάτων και έχει ένα στατικό περιβάλλον εκτέλεσης. Η έκδοση *2.2* αποτελεί ένα υπερσύνολο της προηγούμενης, με κάποιες συναρτήσεις να έχουν αποσυρθεί, ενώ έχουν προστεθεί νέες δυνατότητες, όπως παράλληλο I/O, δυναμικός έλεγχος των εφαρμογών και απομακρυσμένες λειτουργίες στη μνήμη. Η πιο πρόσφατη έκδοση είναι η *3.1* η οποία παρέχει επεκτάσεις στις συλλεκτικές λειτουργίες αλλά και στις μονόπλευρες λειτουργίες.

Λειτουργία Η διεπιφάνεια του MPI, παρέχει δυνατότητες μιας απαραίτητης εικονικής τοπολογίας μεταξύ των εφαρμογών, και δυνατότητες επικοινωνίας μεταξύ αυτών. Στόχος είναι οι δυνατότητες αυτές να μην είναι εξαρτημένες από την εκάστοτε γλώσσα προγραμματισμού. Στη βιβλιοθήκη συναρτήσεων του MPI, παρέχονται ρουτίνες για πληθώρα λειτουργιών. Για παράδειγμα ρουτίνες που αφορούν τη δημιουργία εικονικής τοπολογίας είτε

καρτεσιανού τύπου, είτε γράφου, την αποστολή και λήψη μηνυμάτων μεταξύ ζευγών από διεργασίες, τη συλλογή ή το διαμοιρασμό δεδομένων από ή σε όλες τις διεργασίες (gather, reduce operations), το συγχρονισμό των διεργασιών καθώς και γενικού τύπου ρουτίνες που αφορούν τη συλλογή πληροφοριών του δικτύου ή του συστήματος που βρίσκεται κάποια διεργασία. Οι περισσότερες ρουτίνες που αφορούν την αποστολή/λήψη μηνυμάτων, παρέχονται σε δυο διαφορετικές εκδόσεις: σύγχρονες και ασύγχρονες. Οι εκδόσεις αυτές επιτρέπουν τη διαφοροποίηση του τρόπου επικοινωνίας σταματώντας ή όχι την εκτέλεση μέχρι την ολοκλήρωση της επικοινωνίας, παρέχοντας στον χρήστη μεγάλη ευχέρεια και ευκολία στον προγραμματισμό, ενώ ταυτόχρονα ενισχύουν τη παράλληλη εκτέλεση.

1.3 MATLAB

Το MATLAB είναι περιβάλλον αριθμητικής υπολογιστικής και μια γλώσσα προγραμματισμού τέταρτης γενιάς. Προσφέρει πληθώρα εργαλείων και ρουτινών που εξυπηρετούν εφαρμογές για πάρα πολλά διαφορετικά επιστημονικά πεδία π.χ μηχανική, στατιστική, εφαρμοσμένα μαθηματικά, ακουστική, ηλεκτρονική κλπ. Ταυτόχρονα, η δυνατότητα κατασκευής γραφημάτων κάθε τύπου το καθιστά ένα πάρα πολύ ισχυρό Post-processing εργαλείο. Επίσης παρέχεται η δυνατότητα πρόσβασης στα περιεχόμενα του πακέτου μέσω άλλων γλωσσών προγραμματισμού (C/C++, Java, .NET, Python, SQL) ή άλλων διαφορετικών εφαρμογών (Microsoft Excel).

Όλες οι διαθέσιμες ρουτίνες του πακέτου είναι κλειστού κώδικα. Ωστόσο το περιβάλλον του MATLAB δίνει τη δυνατότητα στο χρήστη να κατασκευάσει δικά του προγράμματα (scripts) χρησιμοποιώντας την εγγενή γλώσσα του περιβάλλοντος. Όπως προαναφέρθηκε η εγγενής γλώσσα του περιβάλλοντος είναι τέταρτης γενιάς. Πρακτικά αυτό σημαίνει ότι ο κώδικας εκτελείται επί τόπου, χωρίς πρόσθετες διαδικασίες μεταγλώττισης του κώδικα, ενώ δεν υπάρχει κάποιος πολύ αυστηρός τρόπος γραφής του κώδικα. Προγράμματα που γράφονται με αυτό το τρόπο, εξυπηρετούν την αυτοματοποιημένη εκτέλεση διαδικασιών που επιθυμεί ο χρήστης, χρησιμοποιώντας είτε υπάρχουσες είτε επιπρόσθετα ορισμένες διαδικασίες.

Με τις νεότερες εκδόσεις του, το MATLAB εκμεταλλεύεται όλο και περισσότερο την αρχιτεκτονική του εκάστοτε συστήματος. Αυτό μπορεί να σημαίνει, είτε ύπαρξη πολυάριθμων επεξεργαστικών πυρήνων είτε ύπαρξη κάποιου επιταχυντή π.χ. κάρτας γραφικών. Έτσι το περιβάλλον MATLAB εκτός από μια ισχυρότατη εργαλειοθήκη, μετατρέπεται με τη πάροδο των εκδόσεων σε ένα πολύτιμο εργαλείο παράλληλης επεξεργασίας, που έχει όλα τα εχέγγυα να συγκριθεί με προσαρμοσμένους παράλληλους κώδικες γραμμένων σε άλλες γλώσσες προγραμματισμού. Ειδικότερα για τους τρόπους με τους οποίους το MATLAB χρησιμοποιεί κάρτες γραφικών γίνεται ανάλυση στο υποκεφάλαιο 4.6.

Κεφάλαιο 2

Θεωρητικό Μέρος

Σε αυτό το κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο των υπολογισμών που πραγματοποιούνται στη παρούσα εργασία. Αναλύονται τα μοντέλα αναπαράστασης φάσεων (phase field models) και διατυπώνονται οι διαφορετικές καταστάσεις διαβροχής Cassie-Baxter (CB) και Wenzel (W). Γίνεται ενεργειακή ανάλυση της CB-W μετάβασης και περιγράφεται ο ρόλος των μονοπατιών ελάχιστης ενέργειας (Minimum Energy Paths - MEPs). Περιγράφεται η μέθοδος χορδής (String Method) και η απλοποιημένη εκδοχή της SSM (Simplified String Method).

2.1 Καταστάσεις διαβροχής

Η υπερυδρόφοβη συμπεριφορά στερεών επιφανειών μπορεί να επιτευχθεί μέσω κατάλληλου σχεδιασμού της επιφάνειας. Αυτές οι κατάλληλα κατασκευασμένες επιφάνειες μπορούν να χαρακτηριστούν σαν υπερυδρόφοβες, όταν οι σταγόνες που τοποθετούνται πάνω τους έχουν υψηλές γωνίες επαφής και έχουν τη τάση να γλιστρούν εύκολα (χαμηλή υστέρηση [27]). Η παρατηρούμενη υπερυδρόφοβη κατάσταση ονομάζεται κατάσταση διαβροχής Cassie-Baxter (CB). Ουσιαστικά πρόκειται για μια κατάσταση ισορροπίας τριών φάσεων (στερεή-υγρή-αέρια) με σχετικά μικρή διεπιφάνεια μεταξύ στερεής και υγρής φάσης, με τη σταγόνα να διαβρέχει μόνο τις προεξοχές της επιφανειακής μορφολογίας. Ωστόσο η κατάσταση CB είναι μετασταθής και αν διαταραχθεί θα μεταβεί πιθανά στην κατάσταση Wenzel (W). Η μετάβαση συνεπάγεται την εισχώρηση της σταγόνας στη μορφολογία της επιφάνειας, μεγιστοποιώντας τη διεπιφάνεια μεταξύ στερεής και υγρής φάσης. Η κατάσταση W, συνοδεύεται από την πάκτωση (droplet pinning [19]) της σταγόνας και επίσης από τη προφανή μείωση της γωνίας επαφής. Η μετάβαση στην κατάσταση W έχει ιδιαίτερα αρνητικές συνέπειες, ειδικά σε εφαρμογές που απαιτούν μια σταθερή υπερυδρόφοβη συμπεριφορά. Ο σχεδιασμός κατάλληλων επιφανειών που αποτρέπουν την μετάβαση από την CB στη W κατάσταση (CB-W) μπορούν να βελτιώσουν πολλές εφαρμογές όπως οι αυτοκαθαριζόμενες επιφάνειες [4, 5], ή βαλβίδες μικρορευστομηχανικών διατάξεων χωρίς μηχανικά μέρη [6] κλπ.

Η διαταραχή της CB κατάστασης μπορεί να οδηγήσει σε διαφορετικά αποτελέσματα ανάλογα με τη φύση της. Αν η διαταραχή είναι μικρή, τότε το σύστημα σταγόνας-επιφάνειας δεν θα οδηγηθεί σε μετάβαση σε διαφορετική κατάσταση διαβροχής και θα επιστρέψει στην αρχική του κατάσταση. Αν η διαταραχή είναι μεγάλη, αλλά όχι κατάλληλα "στοχευμένη", τότε το σύστημα μπορεί να καταλήξει σε κατάσταση ισορροπίας W, καταναλώνοντας όμως περίσσεια ενέργειας χωρίς να είναι απαραίτητο. Ανάλογα με τη φύση της διαταραχής, κάθε σύστημα αντιδρά διαφορετικά, ενώ επιδεικνύει και μεταβλητή αντίσταση στην CB-W μετάβαση. Έτσι αντί για την διερεύνηση των διαφορετικών ειδών διαταραχών, μπορεί να γίνει διερεύνηση για την ασθενέστερη δυνατή διαταραχή που απαιτείται για την εκκίνηση της μετάβασης. Αυτή η ασθενέστερη διαταραχή, προσθέτει στο σύστημα αρκετή ενέργεια έτσι ώστε να καλυφθεί το ενεργειακό φράγμα που διαχωρίζει τα δυο ελάχιστα των καταστάσεων ισορροπίας. Εφόσον ξεπεραστεί αυτό το ενεργειακό φράγμα, το σύστημα θα καταλήξει στο ενεργειακό ελάχιστο που αντιστοιχεί στην κατάσταση W. Το ενεργειακό φράγμα αποτελεί ένα μέτρο για την εκτίμηση της ανθεκτικότητας των υπερυδρόφοβων επιφανειών (μεγάλο

φράγμα - καλή ανθεκτικότητα).

2.2 Μονοπάτι Ελάχιστης Ενέργειας (Minimum Energy Path, MEP)

Το μονοπάτι που συνδέει δύο ενεργειακά ελάχιστα μέσω ενός ενεργειακού φράγματος σύμφωνα με τη βιβλιογραφία είναι γνωστό ως μονοπάτι ελάχιστης ενέργειας (MEP) [7–11]. Ένα MEP μπορεί να περιέχει πολλαπλά ενεργειακά φράγματα, που αντιστοιχούν σε επιπλέον ενεργειακά ελάχιστα μεταξύ των φραγμάτων. Τα ενδιάμεσα ενεργειακά ελάχιστα, αντιστοιχούν σε ενδιάμεσες καταστάσεις διαβροχής (impregnating states [14]), στις οποίες η σταγόνα έχει μερικώς εισχωρήσει στη μορφολογία της επιφάνειας. Εκτός αυτού, μπορεί να υπάρχουν περισσότερα του ενός MEP μεταξύ δύο διαφορετικών καταστάσεων ισορροπίας, υποδεικνύοντας έτσι διαφορετικές ενεργειακά προτιμητέες μεταβάσεις. Τα MEP αποτελούν ένα πολύ ενδιαφέρον κομμάτι στα φαινόμενα διαβροχής, με τον υπολογισμό των πιθανών μεταβάσεων αλλά και των ενεργειακών φραγμάτων.

Γενικά ένα μονοπάτι στο χώρο των φάσεων ορίζεται ως:

$$\gamma_c = \phi_\alpha : \alpha \in [0, 1], \tag{2.1}$$

όπου α είναι μια παραμετροποίηση του μονοπατιού και ϕ_α ένα χαρακτηριστικό μέγεθος του συστήματος στο σημείο α του μονοπατιού. Στη παρούσα εργασία το ϕ συμβολίζει τη συνάρτηση αναπαράστασης φάσεων (phase-field).

Η γενικευμένη θερμοδυναμική δύναμη που εφαρμόζεται σε ένα μονοπάτι πρέπει να είναι εφαπτομενική σε αυτό [3]. Στη παρούσα εργασία χρησιμοποιείται η λαπλασιανή του χημικού δυναμικού $\nabla^2 G$ ως η γενικευμένη θερμοδυναμική δύναμη. Έτσι ισχύει:

$$(\nabla^2 G)^\perp \gamma = 0 \tag{2.2}$$

Η εφαρμογή της $(\nabla^2 G)^\perp$ στο γ οδηγεί στον προσδιορισμό του MEP. Για τον αριθμητικό προσδιορισμό του MEP, έχουν αναπτυχθεί πολλές διαφορετικές μέθοδοι, όπως η elastic band method [12], η dimer method [13] και η string method [16]. Η τελευταία προσεγγίζει το MEP, σαν μια διακριτοποιημένη καμπύλη σε έναν πολυδιάστατο χώρο που συνδέει τα δυο ελάχιστα στο ενεργειακό τοπίο του συστήματος. Συγκεκριμένα στη παρούσα εργασία χρησιμοποιείται μια SSM (simplified string method) [17], η οποία παρουσιάζει καλύτερες ιδιότητες σύγκλισης αλλά και χαμηλότερο υπολογιστικό κόστος.

2.3 Μέθοδος χορδής (String Method, SM)

Ένα μονοπάτι στο χώρο των φάσεων γ, αναπαρίσταται από ένα διακριτό σύνολο καταστάσεων ϕ_i, που στα πλαίσια της μεθόδου SM ονομάζονται εικόνες (images).

$$\gamma = \phi_i : i = 0, 1, \cdots, N = \phi_i \tag{2.3}$$

Το μήκος του μονοπατιού ορίζεται ως

$$L(\gamma) = \sum_{i=0}^{N-1} |\phi_{i+1} - \phi_i| \tag{2.4}$$

Ο υπολογισμός αυτός όπως είναι προφανές, εμπεριέχει τον υπολογισμό της απόστασης μεταξύ δυο εικόνων και ορίζεται από την $L^2 - norm$.
Το εφαπτόμενο μοναδιαίο διάνυσμα $\hat{t}$ στη κατάσταση ϕ_i υπολογίζεται ως:

$$\hat{t}_i = \frac{\phi_{i+1} - \phi_i}{|\phi_{i+1} - \phi_i|} \tag{2.5}$$

Με αυτή τη λογική η εξέλιξη κάθε εικόνας γίνεται ως εξής

$$\phi_i^{(n+1)} = \Delta t \nabla^2 G + \phi_i^{(n)} \tag{2.6}$$

Η μετατόπιση των εικόνων έχει νόημα μόνο όταν γίνεται κάθετα στο γ, καθώς η εφαπτομενική συνιστώσα απλά συνεισφέρει στη μετατόπιση της εικόνας επί της καμπύλης. Έτσι οι τελικές εξισώσεις της SM θα είναι της μορφής

$$\gamma^{(n)} \rightarrow \gamma^{(n+1)} = \{\phi_i^{(n+1)}\}, \tag{2.7}$$

όπου $\phi_i^{(n+1)} = \Delta t (\nabla^2 G)^{\perp} + \phi_i^{(n)} + \lambda_i \hat{t}$

Οι πολλαπλασιαστές Lagrange λ_i υπολογίζονται ξεχωριστά για κάθε εικόνα, επιβάλλοντας μια ισοκατανομή των εικόνων κατά μήκος του μονοπατιού. Με Δt συμβολίζεται το χρονικό βήμα της επίλυσης, που ουσιαστικά καθορίζει τη ταχύτητα της εξέλιξης του μονοπατιού.

Για την υλοποίηση των παραπάνω υπολογισμών είναι απαραίτητος τόσο ο υπολογισμός των πολλαπλασιαστών Lagrange όσο και ο υπολογισμός της κάθετης στη καμπύλη συνιστώσας της θερμοδυναμικής δύναμης. Προφανώς οι υπολογισμοί αυτοί προσθέτουν υπολογιστικό κόστος, και γιαυτό χρησιμοποιείται μια διαφορετική προσέγγιση για την απλοποίηση των υπολογισμών.

Οι υπολογισμοί του SM χωρίζονται σε 2 βήματα. Αρχικά η χορδή εξελίσσεται σύμφωνα με την γενικευμένη θερμοδυναμική δύναμη και στη συνέχεια η καμπύλη επαναπαραμετροποιείται. Ουσιαστικά πρόκειται για την αντικατάσταση όλων των καταστάσεων που αποτελούν τη χορδή $\tilde{\phi}_i$, με άλλες ισαπέχουσες καταστάσεις ϕ_i, που είναι ισοκατανεμημένες κατά μήκος της χορδής. Αυτές οι νέες καταστάσεις προκύπτουν από τις αρχικές, με παρεμβολή. Με αυτό το μηχανισμό δεν είναι πλέον απαραίτητη η εύρεση της κάθετης στη χορδή συνιστώσας της θερμοδυναμικής δύναμης.

Η νέα αυτή μέθοδος στην ουσία αποτελεί μια απλουστευμένη εκδοχή της αρχικής μεθόδου και ονομάζεται SSM (Simplified String Method).

1. Βήμα εξέλιξης

$$\gamma^{(n)} \to \tilde{\gamma}^{(n+1)} = \{\tilde{\phi}_i^{(n+1)}\} \quad \text{με} \quad \tilde{\phi}_i^{(n+1)} = \Delta t(\nabla^2 G) + \phi_i^{(n)} \tag{2.8}$$

2. Βήμα επανεπαραμετροποίησης

$$\begin{aligned} \tilde{\gamma}^{(n+1)} \to \gamma^{(n+1)} = \phi_i^{(n+1)} \quad &\text{έτσι ώστε} \\ |\phi_{i+1}^{(n+1)} - \phi_i^{(n+1)}| = \frac{L(\tilde{\gamma}^{(n+1)})}{N-1} \quad &i = 0, 1, 2, \cdots, N-1 \end{aligned} \tag{2.9}$$

$\tilde{\gamma}^{(n+1)}$ και $\gamma^{(n+1)}$ αναπαριστούν το ίδιο μονοπάτι στο χώρο των φάσεων.

2.4 Μέθοδος αναπαράστασης φάσεων, Phase Field Method

Η μέθοδος αναπαράστασης φάσεων είναι ένα μαθηματικό μοντέλο για την επίλυση προβλημάτων με αλληλεπιδράσεις διαφορετικών φάσεων. Η κατάσταση της όλης δομής περιγράφεται από μια μεταβλητή γνωστή ως παράμετρο τάξης ϕ (order parameter). Π.χ. Με $\phi = -1$ αναπαρίσταται η αέρια φάση, με $\phi = 1$ η υγρή και με τις ενδιάμεσες τιμές $-1 < \phi < 1$ η διεπιφάνεια. Σύμφωνα με τα παραπάνω, η διεπιφάνεια μπορεί εύκολα να εντοπισθεί από την περιοχή στην οποία η μεταβλητή ϕ αλλάζει από την τιμή του ρευστού προς την τιμή του στερεού. Το εύρος στο χώρο στο οποίο μπορούν να εντοπιστούν τέτοιου είδους αλλαγές τιμών της ϕ καθορίζουν το πάχος της διεπιφάνειας. Το σύνολο των τιμών της παραμέτρου ϕ όλης της δομής αποτελούν το *πεδίο φάσεων*.

Στη παρούσα εργασία, η μέθοδος αναπαράστασης φάσεων είναι αυτή που χρησιμοποιείται για τον υπολογισμό των καταστάσεων διαβροχής και σε συνδυασμό με την μέθοδο SSM είναι εφικτός ο προσδιορισμός του MEP. Χρησιμοποιείται η Cahn-Hilliard (βλ. Κεφ 3) έναντι της Allen-Cahn διατύπωσης για το συγκεκριμένο μοντέλο αναπαράστασης φάσεων. Η Allen-Cahn διατύπωση, δεν διατηρεί τον όγκο της σταγόνας και έτσι απαιτείται η προσθήκη ενός ολοκληρωτικού περιορισμού. Ο επιπλέον υπολογισμός μειώνει την ταχύτητα εκτέλεσης και έτσι προτιμάται η Cahn-Hilliard διατύπωση, η οποία εγγενώς διατηρεί τον όγκο της σταγόνας.

Κεφάλαιο 3

Μαθηματική Προτυποποίηση

Στο κεφάλαιο αυτό καταστρώνεται μαθηματικά η υπολογιστική μέθοδος που εφαρμόζεται στις προγραμματιστικές υλοποιήσεις. Οι υπολογισμοί αφορούν τις καταστάσεις διαβροχής και στη συνέχεια τον υπολογισμό του MEP με χρήση της SSM. Οι καταστάσεις διαβροχής υπολογίζονται χρησιμοποιώντας την Cahn-Hilliard (CH) διατύπωση της μεθόδου αναπαράστασης φάσεων.

3.1 Υπολογισμός καταστάσεων διαβροχής

Οι καταστάσεις διαβροχής υπολογίζονται μέσω της CH διατύπωσης μιας τροποποιημένης μεθόδου αναπαράστασης φάσεων, σύμφωνα με την οποία η ενέργεια ανάμιξης του συστήματος προσαυξημένη με τη συνεισφορά του στερεού υπολογίζεται:

$$f_{mix}(\phi, \nabla\phi) = \frac{1}{2}\lambda|\nabla\phi|^2 + \frac{(1+\alpha)}{2}\frac{\lambda}{3\epsilon^2}(\phi^2-1)^2 + \frac{(1-\alpha)}{2}\frac{\lambda}{2\epsilon^2}k_s(\phi-\phi_s)^2, \tag{3.1}$$

όπου ϕ συμβολίζει το πεδίο-φάσεων (η χωρική μεταβλητή που υποδεικνύει την διανομή των φάσεων, $\phi = 1$ υγρό, $\phi = -1$ αέριο, $\phi = \phi_s$ στερεό). Η παράμετρος λ είναι η πυκνότητα της ενέργειας ανάμιξης [22, 23], η οποία σχετίζεται με την επιφανειακή τάση μέσω της σχέσης $\sigma = 2\sqrt{2}\lambda/3\epsilon$ και το ϵ ρυθμίζει το πάχος της διεπιφάνειας [24, 25] . Η χωρική παράμετρος α σηματοδοτεί την στερεή φάση με τον ακόλουθο τρόπο: $\alpha = -1$ αν υπάρχει στερεό, $\alpha = 1$ σε οποιαδήποτε άλλη περίπτωση.

Ο πρώτος όρος του δεξιού μέλους της εξίσωσης 3.1 αντιστοιχεί στην διεπιφανειακή ενέργεια. Ο δεύτερος όρος επιβάλλει τη μη αναμιξιμότητα των ρευστών οδηγώντας την ϕ στην ρευστή (1) ή αέρια φάση (-1) και συνεισφέρει στην εξίσωση 3.1 μόνο σε περιοχές που καταλαμβάνονται από ρευστά ($\alpha = 1$). Παρομοίως ο τρίτος όρος που αντιστοιχεί στην ενέργεια του στερεού, συνεισφέρει μόνο όταν $\alpha = -1$. Η συνεισφορά του στερεού στην ενεργειακή πυκνότητα γίνεται μέσω ενός δυναμικού με ένα πηγάδι, με το ελάχιστο στο $\phi = \phi_s$, όπου ο όρος ϕ_s σχετίζεται άμεσα με τη διαβρεκτικότητα του στερεού (αναλύεται παρακάτω). Η 3.1 εμπεριέχει μια αυθαίρετη σταθερά k_s, η οποία λαμβάνει επαρκώς μεγάλες τιμές έτσι ώστε να προκύπτει $\phi \simeq \phi_s$ όπου υπάρχει παρουσία του στερεού.

Η παράμετρος ϕ_s είναι συνάρτηση της διαβρεκτικότητας του στερεού, δηλαδή της γωνίας επαφής Young:

$$\theta_Y = 90(1-\phi_s) \tag{3.2}$$

Η εξίσωση 3.2 προκύπτει πολύ απλά. Αν $\phi_s = -1$ τότε η παρουσία του στερεού είναι ισοδύναμη με αυτή του αερίου, οπότε $\theta_Y = 180$ και αντίστοιχα αν $\phi_s = 1$ τότε $\theta_Y = 0$. Αυτό εφαρμόζεται και σε όλες τις ενδιάμεσες καταστάσεις, ακολουθώντας προφανώς μια γραμμική σχέση.

Η ελεύθερη ενέργεια του συστήματος είναι το συναρτησιακό της ενέργειας ανάμιξης που ορίζεται στο υπολογιστικό χωρίο Ω.

$$E(\phi) = \int_\Omega \left(\frac{1}{2}\lambda|\nabla\phi|^2 + \frac{(1+\alpha)}{2}\frac{\lambda}{3\epsilon^2}(\phi^2-1)^2 + \frac{(1-a)}{2}\frac{\lambda}{2\epsilon^2}k_s(\phi-\phi_s)^2\right)dV \tag{3.3}$$

Η συναρτησιακή παράγωγος της εξίσωσης 3.3 δίνει το χημικό δυναμικό του συστήματος:

$$G = \frac{\lambda}{\epsilon^2}\left(-\epsilon^2\nabla^2\phi + \frac{(1+\alpha)}{2}(\phi^2-1)\phi + \frac{(1-\alpha)}{2}k_s(\phi-\phi_s)\right) \tag{3.4}$$

Για λόγους αριθμητικής ευστάθειας η εξίσωση 3.4 τροποποιείται έτσι ώστε να εξαλειφθεί ο όρος $\frac{1+\alpha}{2}$ [2].

Η τροποποιημένη εξίσωση 3.4 διατυπώνεται ως εξής:

$$G = \frac{\lambda}{\epsilon^2}\left(-\epsilon^2\nabla^2\phi + \phi^3 - \phi + \frac{(1-\alpha)}{2}k_s(\phi-\phi'_s)\right) \tag{3.5}$$

με

$$\phi'_s = \frac{(\phi^2{}_s - 1)\phi_s + k_s\phi_s}{k_s} \tag{3.6}$$

Σύμφωνα με την CH διατύπωση η εξέλιξη του ϕ είναι συντηρητική και έχει τη μορφή της εξίσωσης συνέχειας:

$$\frac{\partial\phi}{\partial t} = \nabla\cdot(\mu\nabla G), \tag{3.7}$$

όπου μ είναι το μέτρο της ευκινησίας και για τους σκοπούς αυτής της εργασίας, τίθεται ίσο με 1. Η επιλογή της τιμής για το μ είναι δικαιολογημένη γιατί η δυναμική της εξίσωσης 3.7 δεν επιδρά στον υπολογισμό του MEP. Η χρονική διακριτοποίηση της εξίσωσης 3.7, σε συνδυασμό με την εξίσωση 3.5, πραγματοποιείται με τη χρήση ενός ημι-πεπλεγμένου σχήματος:

$$\begin{aligned}\phi^{(n+1)} + \Delta\tau\nabla^2\left[(1-c_1)\phi^{(n+1)} + (1-c_2)\epsilon^2\nabla^2\phi^{(n+1)}\right] = \\ \phi^{(n)} + \Delta\tau\nabla^2\left[-c_1\phi^{(n)} - c_2\epsilon^2\nabla^2\phi^{(n)} + (\phi^{(n)})^3 + \frac{(1-\alpha)}{2}k_s(\phi^{(n)}-\phi'_s)\right]\end{aligned} \tag{3.8}$$

όπου n είναι ο μετρητής των στιγμιοτύπων και $\Delta\tau = \lambda\Delta t/\epsilon^2$ (Δt είναι το χρονικό βήμα). Η επιλογή των σταθερών c_1, c_2, k_s καθορίζει την σταθερότητα του σχήματος. Στη παρούσα εργασία έχουν επιλεγεί οι τιμές $c_1 = 4, c_2 = 0, k_s = 5$. Η επιλογή $k_s = 5$ είναι επαρκής για να επιβάλει $\phi = \phi_s$ χωρίς να επηρεάσει τη σταθερότητα του σχήματος, αρκεί η τιμή του ϕ_s να μη λαμβάνει οριακές τιμές (κοντά στο 1 ή το -1).

Τα υδρόφοβα υλικά παρουσιάζουν μέτριες τιμές για την γωνία θ_Y, το οποίο πρακτικά σημαίνει ότι οι τιμές του ϕ_s δεν είναι μικρότερες από -0.5. Έτσι για περιπτώσεις πρακτικού ενδιαφέροντος, η αριθμητική ευστάθεια του σχήματος 3.8 είναι εξασφαλισμένη.

3.2 Υπολογισμός μονοπατιού ελάχιστης ενέργειας (MEP)

Ένα MEP (Minimum Energy Path) συνδέει δύο σταθερές καταστάσεις ισορροπίας που αντιστοιχούν σε ενεργειακά ελάχιστα μέσω μιας καμπύλης γ, στο ενεργειακό τοπίο του συστήματος. Το MEP είναι ευθυγραμμισμένο με τις ενεργειακές κοιλάδες που οδηγούν σε σάγματα ενέργειας τα οποία αντιστοιχούν σε ασταθείς καταστάσεις ισορροπίας. Για τον υπολογισμό του MEP, η καμπύλη γ διακριτοποιείται σε μια γραμμική καμπύλη με M κόμβους, με κάθε κόμβο να αναπαριστά μια εικόνα της σταγόνας ($\phi_i, i = 1, 2, ..., M$). Κάθε εικόνα είναι μια διαφορετική χωρική κατανομή του ϕ, χωρίς απαραίτητα να αντιστοιχεί σε κατάσταση ισορροπίας, εκτός αν η συγκεκριμένη εικόνα είναι ενεργειακά τοποθετημένη στο ενεργειακό ελάχιστο ή σάγμα. Η καμπύλη γ, εξελίσσεται εφαρμόζοντας τη συνιστώσα της γενικευμένης θερμοδυναμικής δύναμης που είναι κάθετη στη γ, $(\nabla^2 G)^{\perp}(\gamma)$ σε κάθε μια από τις εικόνες. Με αυτό το τρόπο, η καμπύλη γ θα συγκλίνει στο MEP όπου εξορισμού ισχύει $(\nabla^2 G)^{\perp}(\gamma) = 0$.

Για τον υπολογισμό της κατάλληλης συνιστώσας του $\nabla^2 G$, πρέπει να γίνει η προβολή πάνω σε εφαπτομενικό διάνυσμα της γ, σε κάθε βήμα. Ωστόσο, η διαδικασία αυτή, προσθέτει αρκετά σημαντικό υπολογιστικό κόστος στη μέθοδο. Αν αντί αυτού, η καμπύλη γ εξελιχθεί χρησιμοποιώντας το $\nabla^2 G$, δε θα συγκλίνει στο MEP, αφού κάθε εικόνα θα συγκλίνει σε κάποια κατάσταση ισορροπίας. Σύμφωνα με την απλοποιημένη μέθοδο χορδής (SSM), η καμπύλη γ μπορεί να εξελιχθεί με το $\nabla^2 G$, αν υπάρχει ένα βήμα επαναπαραμετροποίησης της γ σε κάθε χρονικό βήμα, όπου όλες οι εικόνες ανακατανέμονται κατά μήκος της καμπύλης γ. Στη συγκεκριμένη εργασία, οι εικόνες κατανέμονται έτσι ώστε να ισαπέχουν. Η απόσταση μεταξύ 2 εικόνων ορίζεται ως $d_i = \|\phi_{i+1} - \phi_i\|_2, i = 1, 2, \ldots, M-1$ ενώ το μήκος τόξου μέχρι την j εικόνα υπολογίζεται : $s_j = \sum_{i=1}^{j-1} d_i$.

Οι εικόνες της μεθόδου χορδής εξελίσσονται σύμφωνα με την εξίσωση 3.7, προσαυξημένη με τον γενικευμένο όρο καμπυλότητας της γ που προσθέτει τάση στη χορδή.

$$\frac{\partial \phi_i}{\partial t} = \nabla^2 \left[G_i + D_s(2\phi_s - \phi_{i-1} - \phi_{i+1}) \right], i = 2, \ldots, M-1 \tag{3.9}$$

όπου D_s είναι η σταθερά καμπυλότητας. Αυτό γίνεται προκειμένου να αποφευχθούν "τυλίγματα" της γ όπου σε αυτή την περίπτωση το αριθμητικό σχήμα αποσταθεροποιείται.

Η χρονική διακριτοποίηση της εξίσωσης 3.9 γίνεται με τρόπο παρόμοιο με αυτόν της εξίσωσης 3.7:

$$\begin{aligned} \phi^{(n+1)} + \Delta\tau\nabla^2\left[-3\phi_i^{(n+1)} + \epsilon^2\nabla^2\phi_i^{(n+1)} - 2D_s\phi_i^{(n+1)}\right] \\ = \phi_i^{(n)} + \Delta\tau\nabla^2\left[-4\phi^{(n)} + (\phi^{(n)})^3 + \frac{(1-\alpha)}{2}5(\phi_i^{(n)} - \phi_s') - D_s(\phi_{i-1}^{(n)} + \phi_{i+1}^{(n)}\right] \end{aligned} \tag{3.10}$$

όπου οι σταθερές c_1, c_2, k_s αντικαθίστανται από τις τιμές $4, 0, 5$.

Η χωρική διακριτοποίηση της εξίσωσης 3.10 μπορεί να γίνει χρησιμοποιώντας οποιαδήποτε ήδη γνωστή μέθοδο π.χ. πεπερασμένων διαφορών, πεπερασμένων στοιχείων κλπ, οι οποίες όμως συμπεριλαμβάνουν την επίλυση ενός μεγάλου γραμμικού συστήματος. Εδώ αντίθετα, το υπολογιστικό χωρίο Ω - όπου ορίζεται το ϕ - ορίζεται τετραγωνικό με σκοπό να μετασχηματιστεί η 3.10 στο πεδίο συχνοτήτων χρησιμοποιώντας μετασχηματισμούς Fourier. Ο μετασχηματισμός μετατρέπει την 3.10 σε ένα σύνολο γραμμικών αλγεβρικών εξισώσεων που επιλύονται άμεσα.

Το χωρίο Ω κατανέμεται σε ένα τετραγωνικό πλέγμα N κόμβων. Έτσι κάθε εικόνα ϕ_i έχει N βαθμούς ελευθερίας ($\phi_i(x_j), i = 1, 2, \ldots, M; j = 1, 2, \ldots, N$), όπου x η χωρική μεταβλητή. Ο μετασχηματισμός του $\phi_i(x_j)$ είναι $\bar{\phi}_i(k_j)$, όπου k είναι ο κυματαριθμός του μετασχηματισμού Fourier, το ∇^2 μετασχηματίζεται σε $-k^2$ και αντίστοιχα $\nabla^4 \to k^4$. Έτσι η εξίσωση 3.10 γίνεται:

$$\begin{aligned} (1 + 3\Delta\tau k^2 + \Delta\tau k^4\epsilon^2 + 2\Delta\tau k^2 D_s)\overline{\phi_i^{(n+1)}} = \\ (1 + 4\Delta\tau k^2)\overline{\phi_i^{(n)}} - \Delta\tau k\overline{(\phi_i^{(n)})^3} - \\ \frac{5}{2}\Delta\tau k^2\overline{(1-\alpha)(\phi_i^{(n)} - \phi_s')} + \Delta\tau k^2 D_s(\overline{\phi_{i-1}^{(n)}} + \overline{\phi_{i+1}^{(n)}}) \end{aligned} \tag{3.11}$$

Η εξίσωση 3.11 είναι ο βασικός υπολογιστικός πυρήνας της μεθόδου, ο οποίος επιλύεται άμεσα ως προς το $\overline{\phi_i^{(n+1)}}$. Σε κάθε χρονικό βήμα γίνονται οι εξής διαδικασίες:

- Τρεις μετασχηματισμοί Fourier, $\phi_i^{(n)} \to \overline{\phi_i^{(n)}}$, $(\phi_i^{(n)})^3 \to \overline{(\phi_i^{(n)})^3}$, $(1-\alpha)(\phi_i^{(n)} - \phi_s') \to \overline{(1-\alpha)(\phi_i^{(n)} - \phi_s')}$
- Η επίλυση της εξίσωσης 3.11
- Ο αντίστροφος μετασχηματισμός $\overline{\phi_i^{(n+1)}} \to \phi_i^{(n+1)}$
- Ο υπολογισμός της απόστασης $d_i = \|\phi_{i+1} - \phi_i\|_2$ και η ισοκατανομή των $\phi_i^{(n+1)}$ στη καμπύλη $\gamma^{(n+1)}$

Κεφάλαιο 4

Προγραμματιστική Υλοποίηση

Στο κεφάλαιο αυτό αναλύονται οι προγραμματιστικές υλοποιήσεις που χρησιμοποιούνται για την επίλυση. Αρχικά συγκρίνονται δύο υλοποιήσεις που χρησιμοποιούν κάρτες γραφικών (CUDA, MATLAB). Η ριζική διαφορά βρίσκεται στο ότι η CUDA υλοποίηση, εκμεταλλεύεται καλύτερα τις δυνατότητες της κάρτας γραφικών καλώντας προσαρμοσμένους υπολογιστικούς πυρήνες και βελτιστοποιημένες διαδικασίες από βιβλιοθήκες ($cuFFT, cuBLAS$). Αντίθετα η MATLAB υλοποίηση, βασίζεται αποκλειστικά σε υπάρχουσες ρουτίνες-συναρτήσεις του MATLAB προκειμένου να γίνουν οι απαραίτητες διαδικασίες χρησιμοποιώντας την κάρτα γραφικών, ενώ είναι εξαιρετικά αμφίβολη η πλήρης εκμετάλλευση του υπάρχοντος υλικού. Παρουσιάζεται μια ακόμα πιο βελτιωμένη υλοποίηση, που χρησιμοποιεί το πρωτόκολλο MPI, για την δρομολόγηση της εκτέλεσης σε περισσότερες από μια κάρτες γραφικών. Ταυτόχρονα παρουσιάζεται και η υπολογιστική συστοιχία που χρησιμοποιείται για την εκτέλεση.

4.1 Ανδρομέδα

Για τους υπολογισμούς και τα δοκιμαστικά που έγιναν στην παρούσα εργασία χρησιμοποιήθηκε η συστοιχία Ανδρομέδα [30] (Andromeda computational cluster).

Η Ανδρομέδα είναι μια συστοιχία που φιλοξενείται στη Σχολή Χημικών Μηχανικών του Ε.Μ.Π και αποτελείται από 4 κόμβους (HP ProLiant SL390s G7). Κάθε ένας κόμβος (node) αποτελείται από 2 Intel Xeon CPU X5660 CPU's που τρέχουν στα 2.80GHz με 12 MB Cache. Οι επεξεργαστές αυτοί είναι εξαπύρηνοι, επομένως σε κάθε κόμβο υπάρχουν διαθέσιμοι 12 επεξεργαστές. Η συνολική μνήμη (Host Memory) σε κάθε κόμβο είναι 16 Gb RAM DDR3. Επιπροσθέτως σε κάθε κόμβο υπάρχουν εγκατεστημένες 2 GPU's nVIDIA Tesla M2050, η καθεμιά αποτελείται από 448 Cuda Cores ενώ έχει στη διάθεσή της 3 Gb DDR5 μνήμης.

Οι 4 κόμβοι διαθέτουν Gigabit Ethernet.

4.2 Παρουσίαση προβλήματος

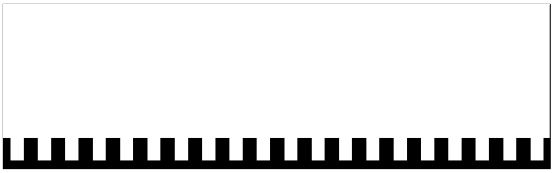
Για τη σύγκριση-ανάπτυξη των υλοποιήσεων επιλέγεται ένα συγκεκριμένο υπολογιστικό σενάριο που παρουσιάζεται παρακάτω (πρόκειται για το ίδιο σενάριο με αυτό που παρουσιάζεται στο [2]). Το υπολογιστικό πρόβλημα έχει διατυπωθεί ήδη στο κεφάλαιο 3. Συνοπτικά πρόκειται για τον υπολογισμό του μονοπατιού ελάχιστης ενέργειας (MEP) κατά την μετάβαση μιας σταγόνας από μια υπερυδρόφοβη κατάσταση Casie-Baxter σε κατάσταση πλήρους διαβροχής Wenzel. Η σταγόνα βρίσκεται πάνω σε μια αυλακωτή επιφάνεια, με αυλακώσεις ορθογωνίου σχήματος. Η σταγόνα έχει μεταφορική συμμετρία και έτσι το υπολογιστικό πρόβλημα είναι διδιάστατο.

Όλες οι παράμετροι του προβλήματος παρουσιάζονται στον πίνακα 4.1.

Παράμετρος	**Περιγραφή**	**Τιμή**
N^2	Σύνολο κόμβων	1024^2
θ_Y	Γωνία Young	112.5
ϵ	Πάχος διεπιφάνειας	0.0016
N_STRING	Πλήθος εικόνων	40

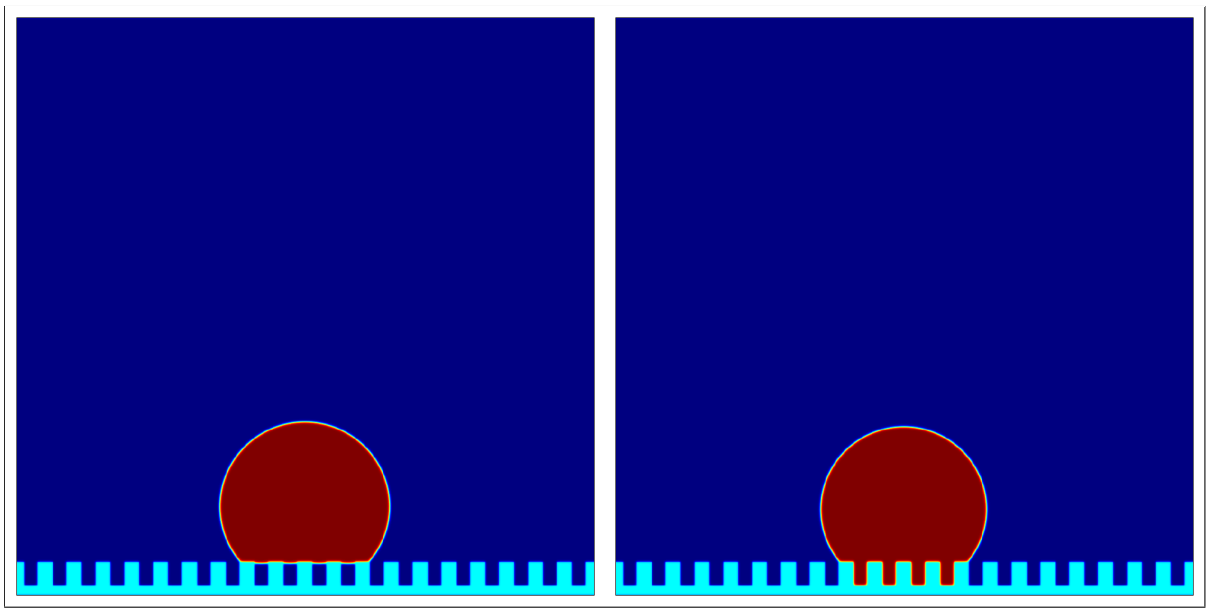
Πίνακας 4.1: Πίνακας παραμέτρων του προβλήματος

Παρακάτω παρουσιάζεται η μορφολογία της επιφάνειας που εξετάζεται.



Σχήμα 4.1: Τομή αυλακωτής επιφάνειας, με ορθογωνικές αυλακώσεις

Η αρχική κατάσταση CB και η τελική κατάσταση W, παρουσιάζονται στις παρακάτω εικόνες για λόγους πληρότητας:

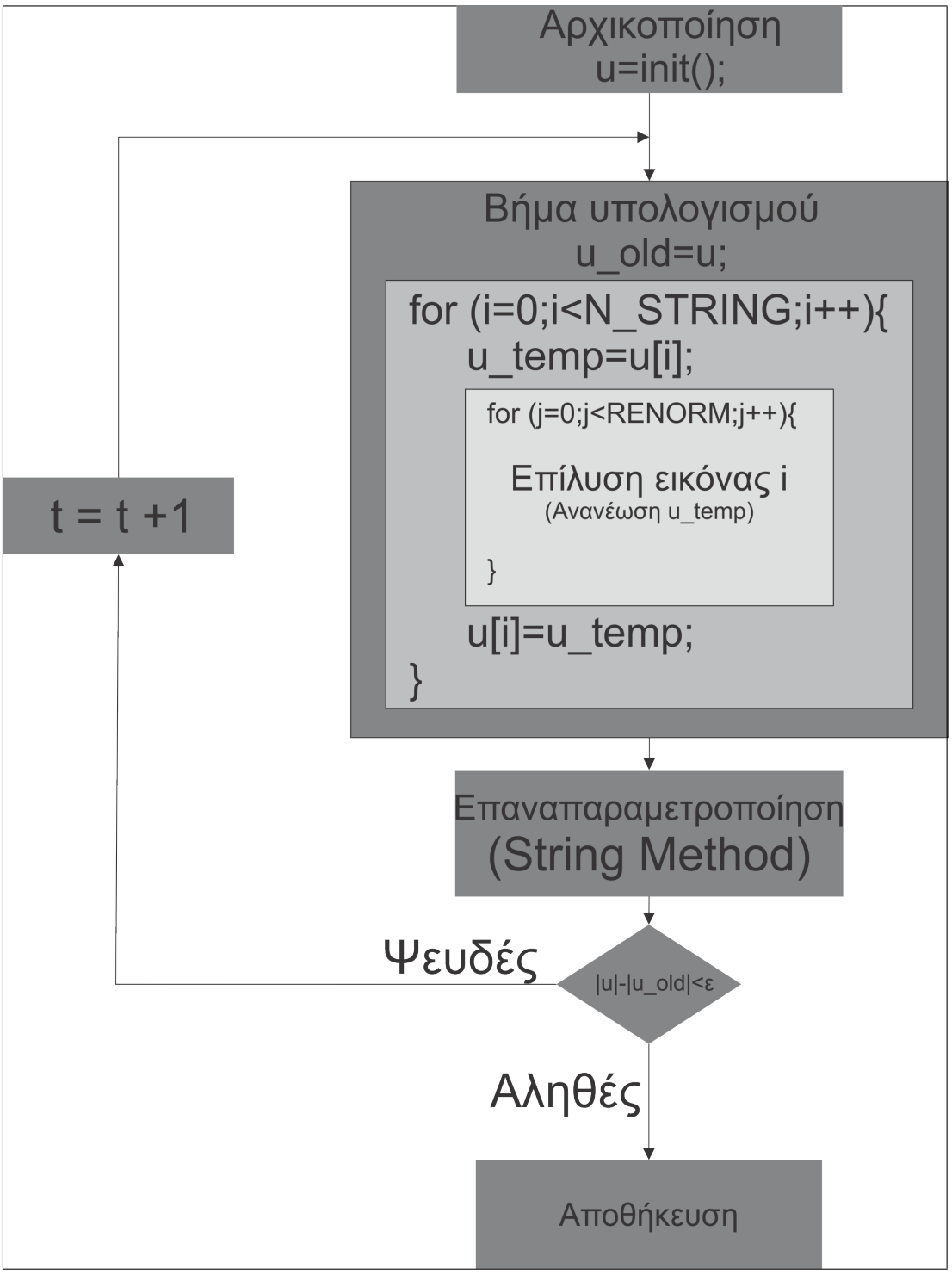


Σχήμα 4.2: Αρχικοποίηση

Οι καταστάσεις αυτές αποτελούν τις ακραίες εικόνες του MEP και υπολογίζονται με την εξίσωση 3.8. Οι υπόλοιπες εικόνες της χορδής υπολογίζονται σύμφωνα με τις εξισώσεις που έχουν ήδη παρουσιαστεί στο κεφάλαιο 3.

4.3 Διάγραμμα Ροής

Το διάγραμμα ροής του κώδικα παρουσιάζεται στο σχήμα 4.3:



Σχήμα 4.3: Διάγραμμα Ροής

Αρχικά πρέπει να γίνει μια πρώτη εκτίμηση της λύσης. Μετά τον υπολογισμό των οριακών συνθηκών (δηλαδή της πρώτης και της τελευταίας εικόνας της χορδής), γίνεται μια

εκτίμηση των ενδιάμεσων εικόνων με απλή γραμμική παρεμβολή. Στη συνέχεια εκκινεί ο χρονικός βρόχος με σκοπό τον υπολογισμό του MEP. Σε αυτό το βρόχο χρησιμοποιείται η τρέχουσα εκτίμηση της λύσης για τον υπολογισμό κάθε νέας. Ο υπολογισμός λαμβάνει χώρα ανά εικόνα της χορδής όπου εφαρμόζεται η εξίσωση 3.10. Οι υπολογισμοί πραγματοποιούνται στη κάρτα γραφικών, χρησιμοποιώντας συναρτήσεις-πυρήνες. Το βήμα αυτό περιλαμβάνει γραμμικές πράξεις μεταξύ των διανυσμάτων που πραγματοποιούνται πολύ γρήγορα σε κάρτες γραφικών, καθώς και μετασχηματισμούς Fourier. Οι μετασχηματισμοί Fourier επιταχύνονται επίσης από τους επεξεργαστές της κάρτας γραφικών, μέσω της βιβλιοθήκης cuFFT.

Αφού ολοκληρωθούν οι υπολογισμοί για όλες τις εικόνες της χορδής στη συνέχεια ακολουθεί το βήμα της επαναπαραμετροποίησης. Όπως έχει ήδη περιγραφεί στο κεφάλαιο 2.3, πρέπει γίνει ανακατανομή των εικόνων της χορδής έτσι ώστε να ισαπέχουν. Το κομμάτι της επαναπαραμετροποίησης, απαιτεί τον υπολογισμό αποστάσεων μεταξύ των εικόνων. Ο υπολογισμός αυτός γίνεται με χρήση $L^2 - norm$. Οι διαδικασίες αυτές αποτελούν διαδικασίες αναγωγής (reduction operations) επί των δεδομένων των εικόνων και κοστίζουν υπολογιστικά, παρότι ο υπολογισμός γίνεται παράλληλα από τους επεξεργαστές της κάρτας γραφικών. Ακόμη, είναι αναγκαίος ο υπολογισμός των αποστάσεων μεταξύ όλων των γειτονικών εικόνων, γεγονός που σημαίνει ότι όσο μεγαλύτερη είναι η διακριτοποίηση της χορδής, τόσο πιο χρονοβόρο είναι το βήμα της επαναπαραμετροποίησης. Για το λόγο αυτό, στο κώδικα υπάρχει πρόβλεψη για πιο "αραιή" εφαρμογή του βήματος αυτού. Στην πραγματικότητα αυτό που γίνεται είναι να συνεχίζεται η εξέλιξη των εικόνων χωρίς να έχουν πρώτα ισοκατανεμηθεί κατά μήκος της χορδής. Η συχνότητα εφαρμογής του βήματος αυτού ρυθμίζεται μέσω της παραμέτρου $RENORM$. Σε περιπτώσεις που το βήμα της επαναπαραμετροποίησης επιβαρύνει χρονικά την εκτέλεση σε βαθμό ίδιο με αυτό του υπολογιστικού τμήματος, επιλέγονται μεγάλες τιμές για την παράμετρο $RENORM$ (της τάξης του 50). Αντίθετα σε περιπτώσεις που το βήμα της επαναπαραμετροποίησης ολοκληρώνεται πολύ ταχύτερα από τους υπολογισμούς, η πιο συχνή εφαρμογή του οδηγεί σε συνολικά ταχύτερη σύγκλιση. Αφού ολοκληρωθεί το βήμα της επαναπαραμετροποίησης ακολουθεί έλεγχος σύγκλισης. Αν η επίλυση έχει συγκλίνει, το πρόγραμμα τερματίζει, διαφορετικά η επίλυση συνεχίζεται σε επόμενο χρονικό βήμα.

4.4 Ανάλυση απαιτήσεων Μνήμης

Η απαίτηση μνήμης του προβλήματος εξαρτάται κατά κύριο λόγο από 2 παράγοντες. Πρώτον από τη διάσταση του πλέγματος και δεύτερον από το πλήθος των εικόνων που αποτελούν τη χορδή (σταθερά N_STRING).

Ο συνολικός αριθμός αγνώστων-βαθμών ελευθερίας (DOFs) που πρέπει να προσδιοριστούν υπολογίζεται ως εξής:

$$\text{DOFs} = N\_STRING * N^2 \tag{4.1}$$

Αν επιλεγεί π.χ. $N\_STRING = 40$ και $N = 1024$, τότε ο συνολικός αριθμός αγνώστων που πρέπει να προσδιοριστούν, υπολογίζεται

$$\text{DOFs} = N\_STRING * N^2 = 40 * 1024^2 = 41943040 \tag{4.2}$$

Η συνολική απαίτηση σε μνήμη, δεδομένης της χρήσης δεκαδικών διπλής ακρίβειας είναι:

$$\text{Theoretical Total Mem} = 41943040 * 8\text{Bytes} = 335544320Bytes = 320MB \tag{4.3}$$

Ωστόσο, λόγω της χρήσης βοηθητικών διανυσμάτων η πραγματική απαίτηση σε μνήμη είναι περίπου το τριπλάσιο της παραπάνω τιμής:

$$\text{Total Mem} \simeq 3 * 320MB = 960MB = 0.94GB \tag{4.4}$$

Στη παρούσα εργασία όπως αναφέρεται στο Κεφάλαιο 4.1 η διαθέσιμη μνήμη στη κάρτα γραφικών Tesla M2050 είναι $\sim 3GB$. Ο μέγιστος θεωρητικά αριθμός εικόνων που θα μπορούσε να υποστηριχθεί από μια κάρτα του συστήματος για πρόβλημα διάστασης $N^2 = 1024 \cdot 1024$, θα ήταν $N\_STRING \simeq 360$. Λόγω των πρόσθετων απαιτήσεων όμως, κάτι τέτοιο είναι αδύνατο και το πραγματικό όριο για πρόβλημα παρόμοιας διάστασης είναι $N\_STRING \simeq 100$.

4.5 Μεταγλώττιση κώδικα (Code Compilation)

Αρχικά ο κώδικας που αναπτύχθηκε για τη παρούσα εργασία αποτελεί μια υλοποίηση της CUDA σε γλώσσα C. Αυτό πρακτικά σημαίνει ότι, προκειμένου να εκτελεστεί ο κώδικας, πρέπει εκ των προτέρων να μεταγλωττιστεί. Η μεταγλώττιση του κώδικα, έχει δυο βασικά προτερήματα:

1. Η μεταγλώττιση του κώδικα οδηγεί σε ένα ήδη υπάρχον σετ εντολών (Assembly) για τον κεντρικό επεξεργαστή (και για τον επεξεργαστή της κάρτας γραφικών στη συγκεκριμένη περίπτωση), το οποίο δεν χρειάζεται κάποια παραπάνω επεξεργασία κατά την εκτέλεση. Οι επεξεργαστές απλά "διαβάζουν" και εκτελούν άμεσα εντολές.

2. Με τη χρήση σύγχρονων μεταγλωττιστών, ο μεταγλωττιστής με κατάλληλες επιλογές (Optimization levels), μπορεί να βελτιώσει στις περισσότερες περιπτώσεις διαδικασίες τις οποίες αντιλαμβάνεται ότι δεν έχουν γραφεί με τον πιο αποδοτικό τρόπο. Έτσι με την "άγνοια" του προγραμματιστή, ένα πρόγραμμα, στις περισσότερες περιπτώσεις, εκτελείται πολύ πιο αποδοτικά απ'ότι αν χρησιμοποιηθούν επακριβώς οι εντολές που έχει ορίσει ο προγραμματιστής στον κώδικά του. Αξίζει να αναφερθεί ότι, ο CUDA μεταγλωττιστής (nvcc), αναλαμβάνει να κάνει βελτιστοποιήσεις και στον κώδικα που τρέχει στους επεξεργαστές γραφικών.

Από την άλλη πλευρά, ένα πρόγραμμα σε περιβάλλον MATLAB, ποτέ δεν μεταγλωττίζεται. Ουσιαστικά πρόκειται για προγραμματισμό μέσω "scripting", όπου προκειμένου να εκτελεστεί το πρόγραμμα, ο κώδικας "διαβάζεται" ανά γραμμή από έναν διερμηνέα (interpreter) με αποτέλεσμα, κάθε εντολή να εκτελείται επίσης ανά γραμμή. Αυτόματα χάνεται κάθε δυνατότητα περαιτέρω βελτίωσης της απόδοσης, ενώ η διαδικασία εκτέλεσης, είναι πολύ πιο αργή από την περίπτωση ενός ήδη μεταγλωττισμένου κώδικα.

4.6 Εκτέλεση στη GPU

Το μεγαλύτερο μέρος του κώδικα και στις δυο υλοποιήσεις καλείται να εκτελεστεί στις κάρτες γραφικών και όχι στους κύριους επεξεργαστές του συστήματος. Ο CPU κώδικας και στις δύο υλοποιήσεις αποτελείται από πολύ βασικές διαδικασίες (κατανομή μνήμης, κλήση ρουτινών, κλπ) που πρακτικά δεν επιβαρύνουν χρονικά την εκτέλεση. Η βασική διαφορά των υλοποιήσεων βρίσκεται στον τρόπο με τον οποίο εκτελούνται οι διαδικασίες στην κάρτα γραφικών.

Ένας κώδικας γραμμένος σε περιβάλλον MATLAB, όταν καλείται να εκτελέσει κάποια εντολή στη κάρτα γραφικών, πάντα ελέγχει αν τα δεδομένα εισόδου είναι διαθέσιμα στην κάρτα γραφικών. Αν δεν είναι διαθέσιμα αναλαμβάνει να κάνει την αντιγραφή πριν την εκτέλεση. Αυτή είναι μια απλή περίπτωση η οποία μπορεί να καθυστερήσει την εκτέλεση. Το MATLAB σαν προγραμματιστικό πλαίσιο παρέχει αρχικά έναν ειδικό τύπο δεδομένων προκειμένου να υπάρχει σαφής ορισμός των δεδομένων που θα βρίσκονται στην κάρτα γραφικών. Αυτός ο τύπος δεδομένων ονομάζεται $gpuarray$. Εκτός αυτού όμως παρέχει πολλούς τρόπους για ανάθεση της εκτέλεσης στην κάρτα γραφικών, αναφορικά αυτοί καταγράφονται στην ακόλουθη λίστα:

- Χρήση των εγγενών (built-in) συναρτήσεων που χρησιμοποιούν δεδομένα από δομές $gpuarray$.
- Εκτέλεση ανά στοιχείο (elementwise) συναρτήσεων χρησιμοποιώντας τη συνάρτηση $arrayfun$.
- Εκτέλεση GPU κώδικα γραμμένο ήδη σε C/C++, ο οποίος εισάγεται στη MATLAB μαζί με την PTX εκδοχή του κώδικα.

Η MATLAB υλοποίηση που χρησιμοποιείται σαν μέτρο σύγκρισης στην εργασία, χρησιμοποιεί τη πιο γρήγορη, εύκολη και απλή (από πλευράς υλοποίησης) από αυτές, που δεν είναι άλλη από την χρήση των εγγενών συναρτήσεων που παρέχει το πλαίσιο του MATLAB. Η επιλογή αυτή δημιουργεί πολύ μεγάλο overhead κατά την εκτέλεση κάθε εντολής. Κάθε φορά που πρέπει να γίνει εκτέλεση στην κάρτα γραφικών, γίνεται τόσο πληθώρα από άλλες "άχρηστες" διαδικασίες για την προετοιμασία των δεδομένων, όσο και καθόλου καλή προσέγγιση για την εκτέλεση της επιθυμητής εργασίας.

Ειδικά στη περίπτωση όπου διαδικασίες που πρέπει να εκτελεστούν στην GPU, βρίσκονται εντός επαναληπτικών βρόχων (όπως συμβαίνει στο υπό μελέτη πρόβλημα), η συνολική καθυστέρηση που παράγεται τελικά είναι τάξεις μεγέθους μεγαλύτερη και προβλέπεται ότι η απόδοση του συγκεκριμένου κώδικα θα είναι εξαιρετικά χαμηλή.

Μια προσέγγιση της εκτέλεσης χρησιμοποιώντας ξεχωριστές συναρτήσεις, και εκτελώντας τες ανά στοιχείο της εισόδου (με της χρήση της $arrayfun$), είναι σίγουρο ότι θα οδηγήσει σε καλύτερη απόδοση. Ενισχύεται ακόμα περισσότερο η αρχιτεκτονική SIMT των GPUs ενώ ταυτόχρονα, ανατίθενται περισσότερες εντολές στα ίδια δεδομένα, αυξάνοντας την απόδοση εκτέλεσης εντολών (instruction throughput).

Η επιλογή της ανάπτυξης CUDA κώδικα και η χρήση του μέσω MATLAB, είναι προφανώς η καλύτερη και η πιο υποσχόμενη επιλογή. Προφανώς υλοποιείται πιο δύσκολα καθώς ο προγραμματιστής μπαίνει στη διαδικασία της ανάπτυξης CUDA κώδικα, ενώ πλέον το πλαίσιο MATLAB χρησιμοποιείται μόνο για Pre-Post Processing. Ειδικότερα, υποστηρίζεται η εισαγωγή (και εκτέλεση) κώδικα συναρτήσεων για τις κάρτες γραφικών (*__device__* functions), ο οποίος μεταγλωττίζεται και στη συνέχεια μπορεί να χρησιμοποιηθεί σαν αντικείμενο (object) στο πρόγραμμα.

Από την άλλη πλευρά στην CUDA υλοποίηση προφανώς η ανάπτυξη εξειδικευμένου κώδικα για τη κάρτα γραφικών, είναι αναπόφευκτη. Εκτός των συναρτήσεων που προγραμματίζονται για να τρέχουν αποκλειστικά στην κάρτα γραφικών (kernels), απαιτείται επίσης ειδική μέριμνα για τους απαραίτητους τύπους δεδομένων των μεταβλητών του προβλήματος. Όλες οι μεταβλητές πρέπει να είναι σαφώς ορισμένες ενώ η αντιγραφή διανυσμάτων πρέπει να γίνει και αυτή "χειροκίνητα" καλώντας βασικές συναρτήσεις της C/CUDA. Ακόμα στην συγκεκριμένη υλοποίηση απαιτείται η χρήση-κλήση, βιβλιοθηκών που παρέχονται από το προγραμματιστικό πλαίσιο της CUDA. Αυτές είναι:

- Βιβλιοθήκη Fourier μετασχηματισμών (cufft)
- Βιβλιοθήκη γραμμικής άλγεβρας (cublas)

Στη περίπτωση της CUDA υλοποίησης, γίνεται άμεση κλήση βασικών συναρτήσεων που εμπεριέχονται στις δυο βιβλιοθήκες. Παρότι και η MATLAB υλοποίηση χρησιμοποιεί τις ίδιες βιβλιοθήκες, είναι πολύ πιθανό να υπάρχει overhead στη κλήση των ίδιων συναρτήσεων, λόγω του τρόπου λειτουργίας της.

4.7 Ειδικά χαρακτηριστικά CUDA υλοποίησης

4.7.1 Σταθερή μνήμη (Constant Memory)

Λόγω των αρκετών σταθερών που παρουσιάζονται στο πρόβλημα, η CUDA υλοποίηση χρησιμοποιεί τη σταθερή μνήμη (**Constant Memory**), που είναι διαθέσιμη στην κάρτα γραφικών. Συνολικά οι σταθερές που χρησιμοποιούνται είναι ελάχιστες (έξι στον αριθμό). Αυτό πρακτικά σημαίνει ότι με μία ανάγνωση ο κάθε επεξεργαστής της κάρτας γραφικών, μπορεί να φορτώσει στην constant cache του όλες τις σταθερές, και έτσι όλα τα νήματα εντός του επεξεργαστή μπορούν να έχουν άμεση πρόσβαση σε αυτές. Άμεσο αποτέλεσμα είναι η μεγαλύτερη ταχύτητα υπολογισμών, λόγω της ελαχιστοποίησης του ποσού της μνήμης που πρέπει να φορτωθεί από την κύρια μνήμη της κάρτας.

4.7.2 Συνενωμένη πρόσβαση στη μνήμη (Memory Coalescing)

Όλες οι συναρτήσεις-πυρήνες (kernels) κατασκευάζονται με τέτοιο τρόπο ώστε να γίνεται συνενωμένη πρόσβαση (coalesced access) στα δεδομένα της μνήμης της κάρτας γραφικών. Η συνενωμένη πρόσβαση στα δεδομένα της κάρτας γραφικών, επιτρέπει την όσο το δυνατόν πιο αποτελεσματική χρήση των διαύλων επικοινωνίας μεταξύ των επεξεργαστών και της κύριας μνήμης της κάρτας γραφικών. Παράλληλα οδηγεί στην πιο αποτελεσματική χρήση των L1 και L2 Cache των επεξεργαστών της κάρτας. Όταν το πρώτο νήμα ενός μπλοκ από νήματα ζητήσει δεδομένα από την μνήμη της κάρτας γραφικών, έως και 128 Bytes από δεδομένα θα φορτωθούν στις διαθέσιμες cache του επεξεργαστή. Αν το επόμενο στη σειρά νήμα του μπλοκ, χρειάζεται δεδομένα τα οποία βρίσκονται σε θέση διαδοχική με αυτή του προηγούμενου νήματος, τότε δεν υπάρχει ανάγκη να ζητήσει αυτά τα δεδομένα από την κύρια μνήμη. Το νήμα αυτό μπορεί να φορτώσει τα δεδομένα που χρειάζεται άμεσα από την L1 cache του επεξεργαστή. Στη πράξη αυτό σημαίνει ότι μία φόρτωση από την κύρια μνήμη της κάρτας γραφικών στην L1 cache του επεξεργαστή, αρκεί για να εξυπηρετήσει τις ανάγκες για δεδομένα πολλών νημάτων εντός του warp στο οποίο ανήκει, και (προφανώς) εντός του μπλοκ που ανήκει. Η συνενωμένη πρόσβαση σε δεδομένα, μειώνει το απαιτούμενο εύρος μνήμης (memory bandwidth), για την φόρτωση των απαραίτητων δεδομένων στις cache μνήμες του επεξεργαστή.

4.7.3 Πληρότητα (Occupancy)

Οι συναρτήσεις-πυρήνες κατασκευάζονται με στόχο την όσο το δυνατόν πληρέστερη χρήση των επεξεργαστών της κάρτας γραφικών. Αυτό συνήθως γίνεται μέσω της προσεγμένης επιλογής της διάστασης του μπλοκ των νημάτων. Ωστόσο αυτή η ρύθμιση δεν μπορεί αποκλειστικά να εξασφαλίσει πληρότητα στον επεξεργαστή. Αν η προς εκτέλεση συνάρτηση, απαιτεί παραπάνω πόρους (συνολικό μέγεθος καταχωρητών (registers) ή κοινής (shared) μνήμης) τότε δεν είναι σε θέση να δρομολογήσει προς εκτέλεση όλα τα νήματα. Σε μια τέτοια περίπτωση προφανώς αυξάνεται ο χρόνος εκτέλεσης της συνάρτησης και μειώνεται η απόδοση.
Στον πίνακα 4.2 παρουσιάζονται ονομαστικά όλοι οι πυρήνες και υπολογίζεται η πληρότητα (occupancy) όσον αφορά τη χρήση των καταχωρητών στο υπολογιστικό σύστημα το οποίο χρησιμοποιείται για την εκτέλεση.

Kernel	**Registers**	**Occupancy (Reg.)**
scheme_assembler	25	0.85
normalize_fft_result	19	1
scheme_precalc	14	1
scheme_assembler_main_loop	29	0.74

Πίνακας 4.2: Ενδεικτική μέτρηση πληρότητας που αφορά τη χρήση των καταχωρητών (registers) της κάρτας γραφικών από τους GPU πυρήνες (kernels).

Τιμές της πληρότητας που πλησιάζουν τη τιμή 1.00 υποδεικνύουν ότι κατά την εκτέλεση του πυρήνα θα υπάρχει πλήρης απασχόληση των επεξεργαστών της κάρτας γραφικών.

Οι συναρτήσεις-πυρήνες καλούνται με 192 νήματα (threads) ανά μπλοκ. Δεδομένου ότι η κάρτα γραφικών που χρησιμοποιείται για τους υπολογισμούς παρέχει δυνατότητα δρομολόγησης 8 μπλοκ ταυτόχρονα, η πληρότητα σύμφωνα με το μέγεθος του μπλοκ με το οποίο καλούνται οι πυρήνες, είναι επίσης 1.00 καθώς τα νήματα που καλούνται να εκτελεστούν παράλληλα κάθε στιγμή, είναι περισσότερα από αυτά που μπορεί να επεξεργαστεί ο κάθε επεξεργαστής της κάρτας γραφικών (Maximum Threads per SM $= 1536$). Θέτοντας στόχο την πλήρη απασχόληση όλων των επεξεργαστών της κάρτας γραφικών, είναι απαραίτητη η κλήση του κατάλληλου αριθμού νημάτων σύμφωνα με τις προδιαγραφές της κάρτας. Για την υλοποίηση όλων των συναρτήσεων πυρήνων στην κάρτα γραφικών, εξαιτίας και

της φύσης των υπολογισμών, επιλέγεται χρήση μονοδιάστατου πλέγματος από μπλοκ, με επίσης μονοδιάστατα μπλοκ από νήματα. Ο λόγος για τον οποίο επιλέγεται αυτή η προσέγγιση είναι το γεγονός ότι όλοι οι υπολογισμοί που πρέπει να γίνουν σε κάθε κόμβο του υπολογιστικού χωρίου, απαιτούν πληροφορίες μόνο του ίδιου κόμβου και όχι γειτονικών του. Αυτή η ιδιαιτερότητα του σχήματος επίλυσης, απλοποιεί πολύ την προγραμματιστική υλοποίηση και επιτρέπει την ανάπτυξη "μονολιθικών" συναρτήσεων-πυρήνων, όπου κάθε νήμα είναι υπεύθυνο για τους υπολογισμούς ενός κόμβου του χωρίου. Έτσι επιλέγεται ο αριθμός 192 νημάτων ανά μπλοκ. Είναι προφανές ότι ο αριθμός των 192 νημάτων, αποτελεί ακέραιο διαιρέτη του συνολικού διαθέσιμου αριθμού νημάτων ανά επεξεργαστή (Streaming Multiprocessor, SM) της κάρτας. Ταυτόχρονα, με 192 νήματα εντός του κάθε μπλοκ, αντιστοιχούν ακριβώς 8 μπλοκ ανά επεξεργαστή, αριθμός που είναι και ο μέγιστος που μπορεί να υποστηρίξει για ταυτόχρονη εκτέλεση η συγκεκριμένη κάρτα (Tesla M2050 βλ. Κεφ. 4.1).

Τυπικά η εκτέλεση στους επεξεργαστές της κάρτας γραφικών σε επίπεδο υλικού οργανώνεται σε οντότητες που ονομάζονται $warps$, οι οποίες ουσιαστικά αποτελούν ομάδες των 32 νημάτων (Σύμφωνα με την αρχιτεκτονική του μοντέλου εκτέλεσης, ο αριθμός αυτός αντιστοιχεί στον αριθμό των πυρήνων CUDA που υπάρχουν ανά επεξεργαστή της κάρτας γραφικών). Αν δεν υπάρξει σωστή επιλογή στο μέγεθος των μπλοκ με το οποίο καλούνται οι συναρτήσεις-πυρήνες, έτσι ώστε να είναι είτε ίσο είτε ακέραιο πολλαπλάσιο του 32, τότε είναι βέβαιο ότι θα υπάρξει μεγάλο κόστος στην απόδοση. Αυτό γίνεται διότι, σε οποιαδήποτε άλλη περίπτωση μεγέθους μπλοκ, θα κληθούν προς εκτέλεση περισσότερα warps από όσα είναι απαραίτητα, με αποτέλεσμα οι επεξεργαστές της κάρτας να υπολειτουργούν (πολλοί πυρήνες CUDA, μένουν αδρανείς), και επιπρόσθετα να αυξάνεται και ο χρόνος εκτέλεσης. Αυτό το είδος προβλήματος κατά την εκτέλεση ονομάζεται απόκλιση ελέγχου (control divergence). Η επιλογή των 192 νημάτων ανά μπλοκ αντιστοιχεί σε $192/32 = 6$ warps, αριθμός που μπορεί να δρομολογηθεί στους επεξεργαστές της κάρτας, απασχολώντας τους πλήρως.

Με βάση τα παραπάνω, και αν μελετηθεί ένα χωρίο με συνολικά $1024x1024$ βαθμούς ελευθερίας είναι προφανές ότι ο διαχωρισμός της διάστασης σε μπλοκ των 192 νημάτων, θα δημιουργήσουν πολλά παραπάνω μπλοκ νημάτων που πρέπει να ανατεθούν στους επεξεργαστές της κάρτας γραφικών. Συγκεκριμένα ο αριθμός των μπλοκ που ανατίθεται στους επεξεργαστές είναι:

$$\text{Total Blocks} = (int)(1024 * 1024 - 1)/192 + 1 = 5462 \tag{4.5}$$

Συνολικά καλούνται $5462 * 192 = 1,048,704$ νήματα ενώ απαιτούνται $1024 * 1024 = 1,048,576$. Υπάρχει μια περίσσεια 128 νημάτων τα οποία αντιστοιχούν σε ακριβώς $128/32 = 4$ warps, γεγονός που σημαίνει ότι δεν θα υπάρχει κάποια μείωση στην απόδοση λόγω απόκλισης ελέγχου. Η Tesla M2050 διαθέτει συνολικά 14 επεξεργαστές, επομένως μπορούν να δρομολογούνται ταυτόχρονα 112 μπλοκ νημάτων. Ο επιλεγόμενος τρόπος εκτέλεσης οδηγεί στην πλήρη απασχόληση όλων των επεξεργαστών της κάρτας γραφικών έως ότου δρομολογηθούν και τα 5462 συνολικά μπλοκ.

Σημείωση: Η τακτική του προγραμματισμού "μονολιθικών" συναρτήσεων πυρήνων, όπου κάθε νήμα αναλαμβάνει υπολογισμούς για κάθε κόμβο του υπολογιστικού χωρίου, εγκυμονεί κινδύνους. Σε προβλήματα μεγαλύτερης διάστασης, είναι πολύ πιθανόν να μην υπάρχει ο διαθέσιμος αριθμός νημάτων για να υπάρξει 1-1 ανάθεση της επεξεργασίας στα νήματα, χρησιμοποιώντας τη συγκεκριμένη ρύθμιση του μεγέθους μπλοκ. Στη συγκεκριμένη υλοποίηση, οι προδιαγραφές της κάρτας γραφικών παρέχουν αρκετούς πόρους έτσι ώστε να καλυφθεί η απαίτηση σε νήματα ακόμα και σε περίπτωση 12 φορές μεγαλύτερης διάστασης του υπολογιστικού χωρίου. Σε προβλήματα μεγαλύτερης διάστασης, είναι απαραίτητη είτε η επιλογή διαφορετικής προγραμματιστικής προσέγγισης των συναρτήσεων-πυρήνων είτε επιλογή διαφορετικού μεγέθους μπλοκ, που θα οδηγήσει σε μείωση της απόδοσης.

4.7.4 Απόδοση Υλοποίησης (Implementation Profiling)

Χρησιμοποιώντας, γνωστές εφαρμογές κατάλληλες για τον έλεγχο και την καταγραφή της επίδοσης των υλοποιήσεων CUDA, MATLAB, έγινε προσπάθεια σύγκρισης αυτών, σε κομμάτια κώδικα που αφορούν την εκτέλεση της ίδιας διαδικασίας. Οι εφαρμογές αυτές είναι: *Nvidia Visual Profiler*, για την καταγραφή της επίδοσης στην CUDA υλοποίηση, ενώ για το MATLAB κώδικα χρησιμοποιείται ο Profiler που παρέχεται μαζί με το πακέτο.

Αρχικά γίνεται μια σύγκριση αποκλειστικά για μια διαδικασία που αφορά καθαρά αριθμητικούς υπολογισμούς πάνω στα διανύσματα εισόδου. Η γενίκευση των αποτελεσμάτων θεωρείται ασφαλής, καθώς όπως έχει ήδη προαναφερθεί όλοι οι υπόλοιποι υπολογιστικοί πυρήνες είναι προγραμματιστικά παραπλήσιοι. Συγκεκριμένα ο MATLAB κώδικας που εξετάζεται είναι ο ακόλουθος:

Listing 4.1: MATLAB Sample Vector Kernel

```
1 X = (X.*(1+4*dt.*km_array)-Deff*dt*km_array.*Z-dt*km_array.*Y) ...
2 ./(1+3*dt*km_array+dt*epsilon2*km_array.*km_array);
```

Ο κώδικας αυτός υλοποιείται σε CUDA με τον ακόλουθο πυρήνα:

Listing 4.2: CUDA scheme_assembler Kernel

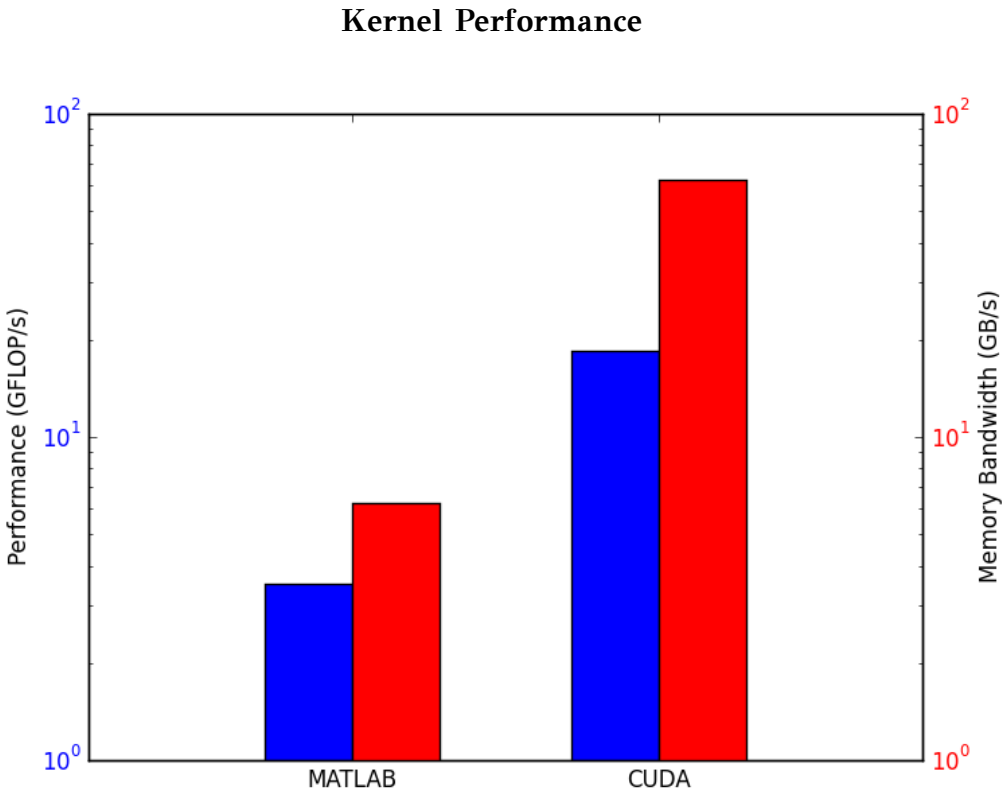
```
__global__ void scheme_assembler_old(cufftDoubleComplex *u, cufftDoubleComplex *calc1,
    cufftDoubleComplex *calc2, int n0, int n1)
{
int tidx = blockDim.x * blockIdx.x + threadIdx.x;
int tidy = blockDim.y * blockIdx.y + threadIdx.y;
int tid  = tidy * n1 + tidx;

if ((tidx < n1) & (tidy < n0)){
    //Calculate Fourier wavenumber
    double iiy, iix;
    double ix = (double) tidx;
    double iy = (double) tidy;

    if (tidx > 0.5*(NEX + 1))
        iiy = 2. * d_pi * (ix - (NEX + 1));
    else
        iiy = 2. * d_pi * ix;

    if (tidy > 0.5*(NEY + 1))
        iix = 2. * d_pi * (iy - (NEY + 1));
    else
        iix = 2. * d_pi * iy;

    double km = iiy * iiy + iix * iix;
    //Update values
    u[tid].x = (u[tid].x * (1.0 + 4.*d_dt * km) - d_Deff * d_dt * km * calc2[tid].x \
    - d_dt * km * calc1[tid].x) / (1. + 3.0 * d_dt * km + d_dt * d_epsilon2 * km * km);
    u[tid].y = (u[tid].y * (1.0 + 4.*d_dt * km) - d_Deff * d_dt * km * calc2[tid].y \
    - d_dt * km * calc1[tid].y) / (1. + 3.0 * d_dt * km + d_dt * d_epsilon2 * km * km);
    }
}
```

Είναι προφανής η διανυσματική (vectorized) προσέγγιση του MATLAB κώδικα. Γίνονται με τη σειρά πράξεις πάνω σε ολόκληρα διανύσματα προκειμένου να υπολογιστεί το τελικό διάνυσμα. Αντίθετα στον κώδικα CUDA, με βάση τα δεδομένα αρχικά διανύσματα της εισόδου, γίνονται όλοι οι απαραίτητοι υπολογισμοί στοιχείο ανά στοιχείο (element-wise approach) εκτελώντας στην ουσία όλους τους απαραίτητους υπολογισμούς κάθε φορά. Μάλιστα τα στοιχεία επεξεργάζονται παράλληλα από τους επεξεργαστές της κάρτας γραφικών βοηθώντας στην πολύ γρήγορη ολοκλήρωση των υπολογισμών. Αυτός ο τρόπος εκτέλεσης επίσης συνενώνει τις προσβάσεις στη μνήμη (αφού οι προσβάσεις γίνονται ανά στοιχείο) και οδηγεί σε πολύ καλύτερη επίδοση.

Οι χρόνοι εκτέλεσης που καταγράφονται για τους παραπάνω κώδικες, παρουσιάζονται στον πίνακα 4.3.

Run	CUDA (sec)	MATLAB (sec)
1	5.11	53.18
2	4.98	54.05
3	5.15	53.10
4	5.05	53.27
5	5.11	53.15
M.O	5.08	53.35

Πίνακας 4.3: Χρόνοι εκτέλεσης για τον υπολογιστικό πυρήνα που αφορά τους υπολογισμούς του σχήματος επίλυσης στον χώρο των συχνοτήτων (CUDA και MATLAB).



Σχήμα 4.4: Διάγραμμα απόδοσης/απόδοσης μνήμης του πυρήνα σε (Gflop/s)/(GB/s).

Η επιτάχυνση που λαμβάνεται στη συγκεκριμένη περίπτωση υπολογίζεται $\frac{\text{MATLAB TIME}}{\text{CUDA TIME}} = \frac{53.35}{5.08} = 10.50$. Αναμένεται ίδιας τάξης επιτάχυνση σε όλες τις υπόλοιπες διαδικασίες που

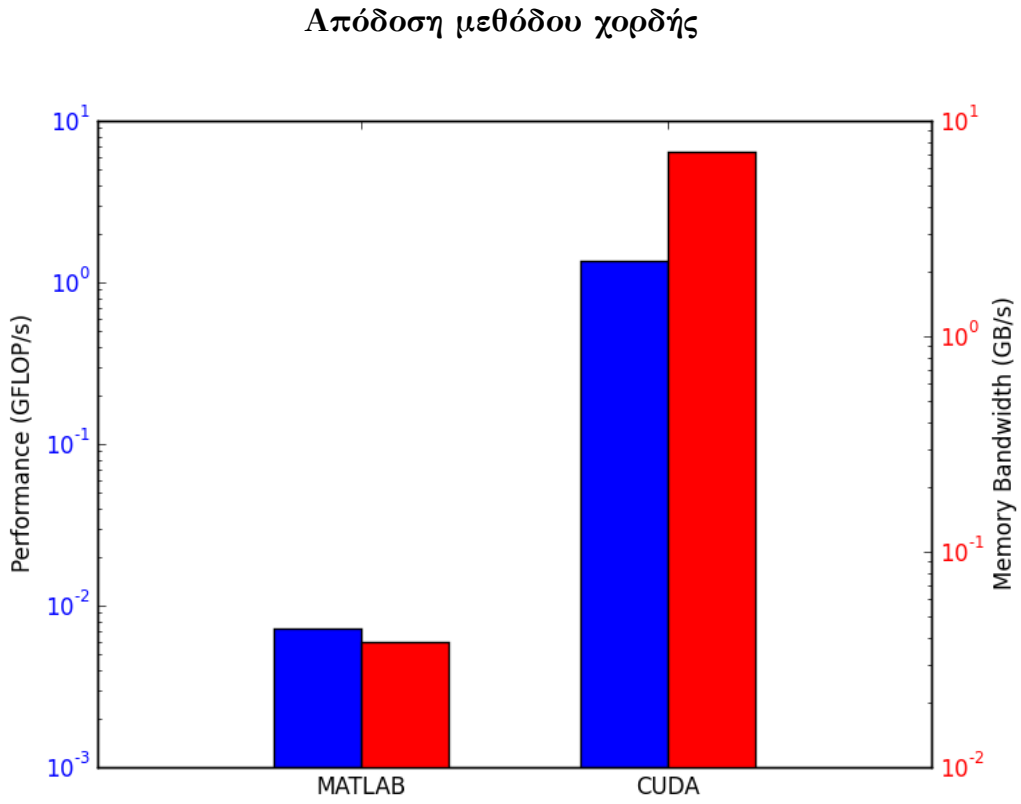
εμπλέκουν υπολογισμούς επί διανυσμάτων. Ο προσεκτικός προγραμματισμός του υπολογιστικού πυρήνα με σκοπό την πλήρη εκμετάλλευση των επεξεργαστών της κάρτας γραφικών καθώς και η διαφορετική υπολογιστική προσέγγιση είναι οι κύριοι λόγοι αυτής της πολύ μεγάλης διαφοράς.

Μια ακόμα διαδικασία που χρίζει σύγκρισης είναι αυτή που αφορά τους υπολογισμούς της μεθόδου χορδής. Η διαδικασία αυτή εμπλέκει υπολογισμούς L2-νορμών επί των δεδομένων της χορδής έτσι ώστε να υπολογιστούν οι μεταξύ τους αποστάσεις. Στη συνέχεια γίνεται ανακατανομή των εικόνων έτσι ώστε αυτές να ισαπέχουν. Η ανακατανομή γίνεται μέσω γραμμικών πράξεων στα δεδομένα των εικόνων, εφαρμόζοντας εξισώσεις παρεμβολής. Η θεωρητική διατύπωση της μεθόδου έχει γίνει στο κεφάλαιο 2.3. Στη συγκεκριμένη περίπτωση δεν υπάρχει ανάγκη χρήσης κάποιου προσαρμοσμένου υπολογιστικού πυρήνα. Στην CUDA υλοποίηση, όλες οι υπολογιστικές ανάγκες καλύπτονται με τη χρήση της βιβλιοθήκης cuBLAS.

Οι χρόνοι εκτέλεσης που καταγράφονται για την διαδικασία της επαναπαραμετροποίησης, παρουσιάζονται στον πίνακα 4.4.

Run	CUDA (sec)	MATLAB (sec)
1	0.18	33.74
2	0.19	34.01
3	0.17	33.90
4	0.18	33.68
5	0.18	33.85
M.O	0.18	33.836

Πίνακας 4.4: Χρόνοι εκτέλεσης για τη μέθοδο χορδής που καταγράφονται για την CUDA και MATLAB υλοποίηση αντίστοιχα.



Σχήμα 4.5: Διάγραμμα απόδοσης/απόδοσης της μεθόδου χορδής σε (Gflop/s)/(GB/s).

Η επιτάχυνση στη συγκεκριμένη διαδικασία υπολογίζεται $\frac{33.836}{0.18} = 188$. Αυτό το πολύ μεγάλο κέρδος στη διαδικασία της επαναπαραμετροποίησης επιτρέπει την πολύ πιο συχνή εφαρμογή της διαδικασίας, η οποία ρυθμίζεται καταλλήλως μέσω της μείωσης της σταθεράς $RENORM$ του κώδικα. Συνήθως για την ταχύτερη σύγκλιση της επίλυσης, Αποτέλεσμα αυτού είναι η πολύ πιο γρήγορη σύγκλιση στη τελική λύση, οπότε και προβλέπεται επίσης μεγάλη μείωση και στον συνολικό χρόνο επίλυσης.

4.8 Χρόνοι εκτέλεσης CUDA-MATLAB

Για τη σύγκριση των δυο υλοποιήσεων, λόγω της φύσης του κώδικα, επιλέγεται μόνο ένα μέγεθος προβλήματος το οποίο συνολικά εκτελέστηκε 5 φορές. Σε κάθε εκτέλεση καταγράφεται ο χρόνος που απαιτείται για την σύγκλιση του πλήρους προβλήματος (δηλαδή την εύρεση λύσης για όλες τις εικόνες της χορδής). Οι χρόνοι τόσο της αρχικοποίησης, όσο και της προετοιμασίας του εκάστοτε κώδικα, θεωρούνται αμελητέοι.

Τα αποτελέσματα παρουσιάζονται στον πίνακα 4.5.

Run	CUDA (min)	MATLAB (min)
1	48.20	500.47
2	48.50	500.52
3	47.90	500.45
4	47.70	500.49
5	48.10	500.47
M.O	48.08	500.47

Πίνακας 4.5: Χρόνοι εκτέλεσης για τις δυο υλοποιήσεις (CUDA και MATLAB). Το πρόβλημα που επιλύεται παρουσιάζεται στο κεφάλαιο 4.2.

Με βάση τους χρόνους εκτέλεσης εξάγεται η επιτάχυνση (speedup) που επιτυγχάνεται με την νέα υλοποίηση:

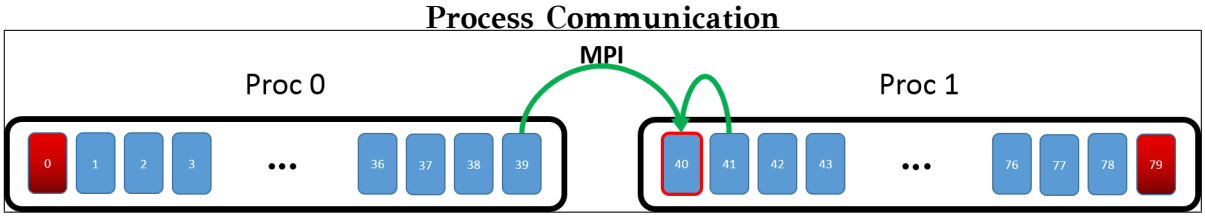
$$SpeedUp = \frac{\text{MATLAB TIME}}{\text{CUDA TIME}} = \frac{500.47}{48.08} = 10.41 \tag{4.6}$$

Η CUDA υλοποίηση αποδεικνύεται **10** φορές πιο γρήγορη από την MATLAB υλοποίηση. Η μεγάλη επιτάχυνση όλων των διαδικασιών καθιστούν τη CUDA υλοποίηση πολύ ισχυρότερη, και εξαιρετικά αποδοτικότερη σε σχέση με τη MATLAB υλοποίηση.

4.9 CUDA-MPI Υλοποίηση

Η υπολογιστική συστοιχία που χρησιμοποιείται για τους υπολογισμούς, περιέχει συνολικά 4 κόμβους με 2 κάρτες γραφικών στο καθένα (Κεφάλαιο 4.1). Σκοπός αυτής της νέας υλοποίησης είναι η εκμετάλλευση όλων των καρτών γραφικών του συστήματος και για το λόγο αυτό, επιλέγεται η χρήση του πρωτοκόλλου MPI, για την δρομολόγηση της εκτέλεσης σε όλους του κόμβους του συστήματος. Αυτή η υλοποίηση επιτρέπει την αύξηση του μεγέθους του προς επίλυση προβλήματος σε μεγέθη που σε καμία περίπτωση δε θα μπορούσαν να εξυπηρετηθούν από μία κάρτα γραφικών.

Ο κώδικας παραμένει στο μεγαλύτερο μέρος ίδιος με αυτόν της σειριακής CUDA υλοποίησης, απλά γίνεται η κατάλληλη τροποποίηση των διαδικασιών που απαιτούν πλέον την επικοινωνία μεταξύ των διεργασιών. Επικοινωνία απαιτείται για την ανταλλαγή πληροφοριών που αφορά γειτονικές εικόνες της χορδής στο βήμα των υπολογισμών, για την ολοκλήρωση του βήματος της επαναπαραμετροποίησης και του ελέγχου σύγκλισης.



Σχήμα 4.6: Παράδειγμα επικοινωνίας 2 διεργασιών. Ένα πρόβλημα συνολικά 40 εικόνων χωρίζεται ισόποσα σε 2 διεργασίες. Ο υπολογισμός των συνοριακών εικόνων απαιτεί την επικοινωνία μεταξύ των διεργασιών.

Πρακτικά, εκτελείται μόνο μια διεργασία για κάθε κάρτα γραφικών. Ο κώδικας είναι κατασκευασμένος έτσι ώστε όλοι οι υπολογισμοί να λαμβάνουν χώρα στη κάρτα γραφικών. Οι CPUs μένουν επί το πλείστον αδρανείς. Για το λόγο αυτό και δεδομένου του συγκεκριμένου υπολογιστικού συστήματος, δεν υπάρχει νόημα να κληθούν πάνω από 8 MPI διεργασίες (8 GPUs)

Ο συνολικός αριθμός εικόνων της χορδής που πρέπει να υπολογιστούν, χωρίζεται ισόποσα και γίνεται κατανομή στις διαθέσιμες κάρτες. Στη συνέχεια, γίνεται η απαραίτητη επικοινωνία για την ανταλλαγή πληροφοριών από γειτονικές διεργασίες και έπειτα κάθε διεργασία εκτελεί αυτόνομα τους απαραίτητους υπολογισμούς στα τοπικά της πλέον δεδομένα. Μετά το πέρας των υπολογισμών της κάθε διεργασίας, απαιτείται επικοινωνία

μεταξύ των διεργασιών έτσι ώστε να ολοκληρωθεί το βήμα της επαναπαραμετροποίησης.

Όπως έχει ήδη προαναφερθεί προκειμένου να μην υπάρχει πρόσθετη καθυστέρηση στους υπολογισμούς, όλα τα απαραίτητα δεδομένα βρίσκονται στη κάρτα γραφικών. Με αυτό το τρόπο πρακτικά εκμηδενίζεται η επικοινωνία CPU-GPU στο υπολογιστικό κομμάτι. Κάθε φορά όμως που υπάρχει ανάγκη επικοινωνίας μεταξύ δυο διεργασιών, τα δεδομένα πρέπει να μετακινηθούν μεταξύ της μνήμης της διεργασίας (host memory - RAM) και της μνήμης της κάρτας γραφικών (device memory - VRAM), καθώς αυτός είναι ο τρόπος που υποστηρίζεται από την χρησιμοποιούμενη MPI υλοποίηση (OpenMPI 1.4.3) . Έτσι στη περίπτωση αποστολής δεδομένων, πρέπει πρώτα να αντιγραφούν τα δεδομένα από τη μνήμη της κάρτας γραφικών, στη κύρια μνήμη της διεργασίας και σε δεύτερο βήμα να αποσταλούν. Αντίστοιχα στη περίπτωση λήψης δεδομένων, πρέπει αυτά πρώτα να παραληφθούν στη κύρια μνήμη της διεργασίας, και σε δεύτερο βήμα να αντιγραφούν στη μνήμη της αντίστοιχης κάρτας γραφικών.

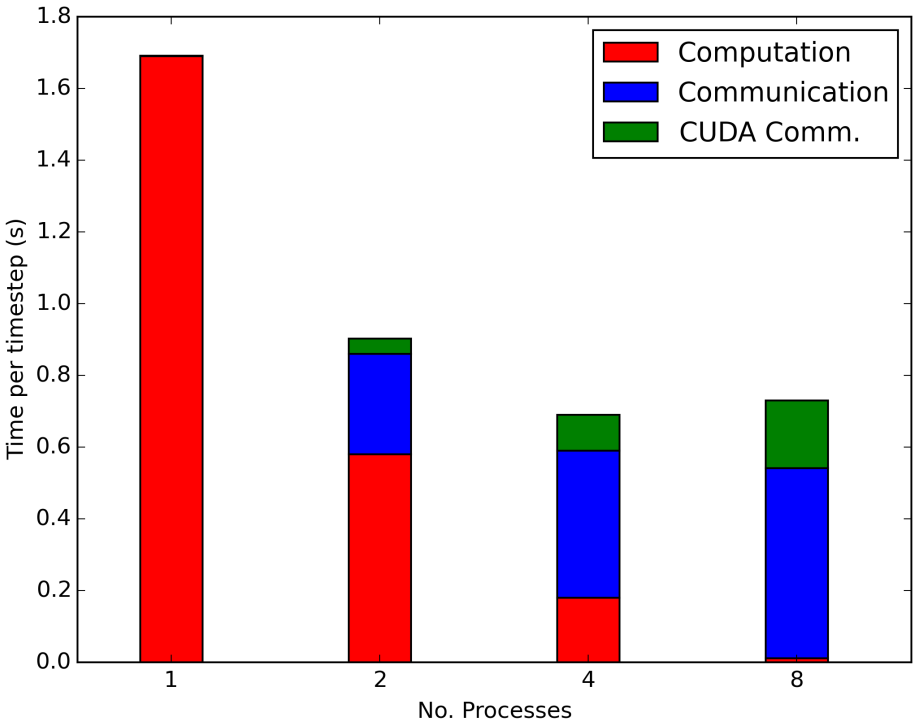
Σε αυτή τη παράλληλη υλοποίηση, η υπολογιστική μέθοδος που χρησιμοποιείται, έχει μερικώς τροποποιηθεί. Πλέον η επίλυση των εικόνων παύει να γίνεται σειριακά, και έτσι αυτό που πρακτικά συμβαίνει είναι οι διεργασίες να μην επεξεργάζονται πλήρως ενημερωμένα δεδομένα όσον αφορά αυτά που προέρχονται από γειτονικές διεργασίες. Το γεγονός αυτό, οδηγεί στην ολοκλήρωση της επίλυσης σε περισσότερα χρονικά βήματα από αυτά που θα απαιτούσε η αντίστοιχη σειριακή επίλυση. Ωστόσο το κέρδος από την παραλληλοποίηση της διαδικασίας εξακολουθεί να είναι πολύ σημαντικό.

Εκτός από τη παραλληλοποίηση του υπολογιστικού μέρους (που επιτυγχάνεται μέσω του διαμοιρασμού των εικόνων σε πολλές διεργασίες), έχουν παραλληλοποιηθεί οι διαδικασίες που αφορούν τόσο την επαναπαραμετροποίηση, όσο και τον έλεγχο σύγκλισης. Οι διαδικασίες αυτές εμπεριέχουν τον υπολογισμό νορμών 2ης τάξης στα δεδομένα των εικόνων της χορδής. Σε αυτή την υλοποίηση, κάθε διεργασία αναλαμβάνει τον υπολογισμό των νορμών των τοπικών της δεδομένων και εν τέλει αποστέλλονται μόνο τα αποτελέσματα, παραλληλίζοντας τη διαδικασία ενώ ταυτόχρονα μειώνεται σε πολύ μεγάλο βαθμό ο όγκος των δεδομένων που λαμβάνονται/αποστέλλονται. Ακριβώς το ίδιο συμβαίνει με τον έλεγχο σύγκλισης που απαιτεί επίσης υπολογισμό νορμών, ο οποίος πραγματοποιείται παράλληλα.

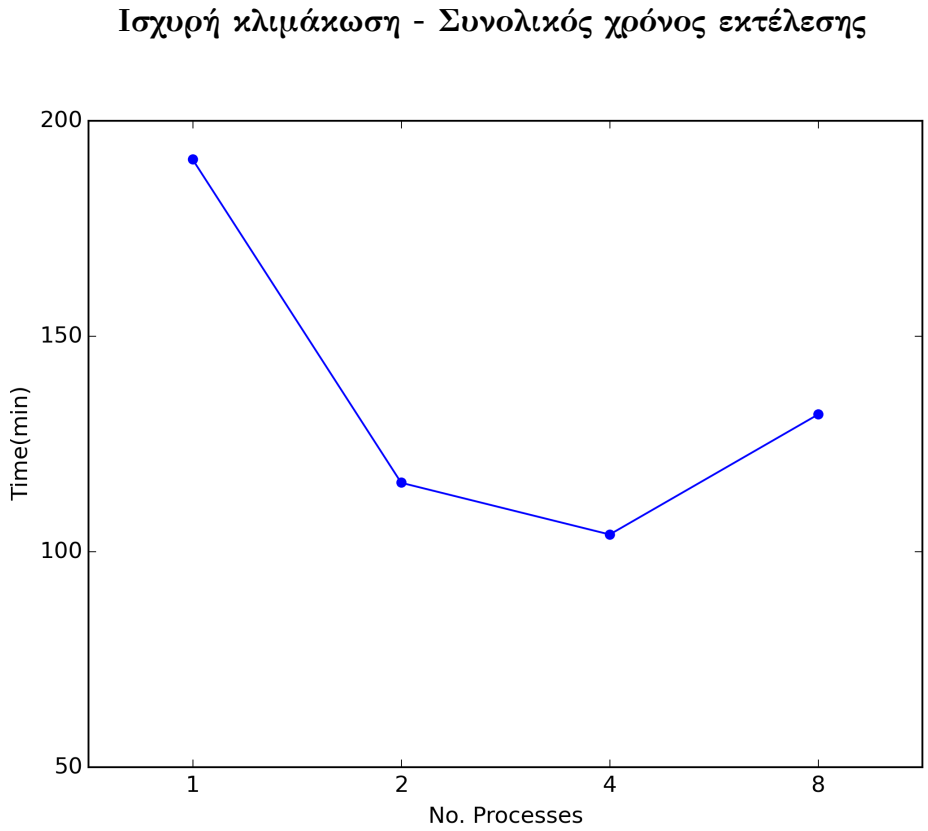
Στο διάγραμμα 4.7, παρουσιάζεται η ισχυρή κλιμάκωση των χρόνων υπολογισμών και επικοινωνίας συναρτήσει του αριθμού καρτών/διεργασιών για ένα συγκεκριμένο μέγεθος

προβλήματος (80 εικόνες με συνολικά $80 \cdot 10^6$ βαθμοί ελευθερίας). Εύκολα παρατηρεί κανείς την πολύ καλή κλιμάκωση του κώδικα όσον αφορά τον χρόνο υπολογισμών ο οποίος μειώνεται σχεδόν υπεργραμμικά με την αύξηση του αριθμού των διεργασιών. Από πλευράς επικοινωνίας παρατηρείται αντίστοιχα μια γραμμική αύξηση του χρόνου, με αρκετά μικρότερη κλίση. Έτσι συνολικά ο απαιτούμενος χρόνος ανά χρονικό βήμα μειώνεται και για το λόγο αυτό θα ανέμενε κανείς την μείωση και του συνολικού χρόνου εκτέλεσης. Ωστόσο στη πραγματικότητα ο συνολικός χρόνος από ένα σημείο και μετά παύει να μειώνεται, και αυτό γιατί όπως προαναφέρθηκε απαιτούνται όλο και περισσότερα χρονικά βήματα για την επίτευξη σύγκλισης (διάγραμμα 4.8). Η συνεχής κλιμάκωση μπορεί να οδηγήσει σε αύξηση των χρονικών βημάτων σε σημείο που η χρήση λιγότερων διεργασιών - καρτών γραφικών να οδηγεί σε συνολικά καλύτερη απόδοση. Η συμπεριφορά αυτή είναι εμφανής σε μικρά σχετικά προβλήματα (40-100 εικόνες συνολικά στη χορδή), ενώ αρχίζει να εκλείπει σε προβλήματα που ξεπερνούν τους $200 \cdot 10^6$ βαθμούς ελευθερίας (περισσότερες από 200 εικόνες στη χορδή) πιθανότατα εξαιτίας της έλλειψης περισσότερων πόρων για το περαιτέρω διαμερισμό του προβλήματος.

Ισχυρή Κλιμάκωση - Χρόνος/χρονικό βήμα

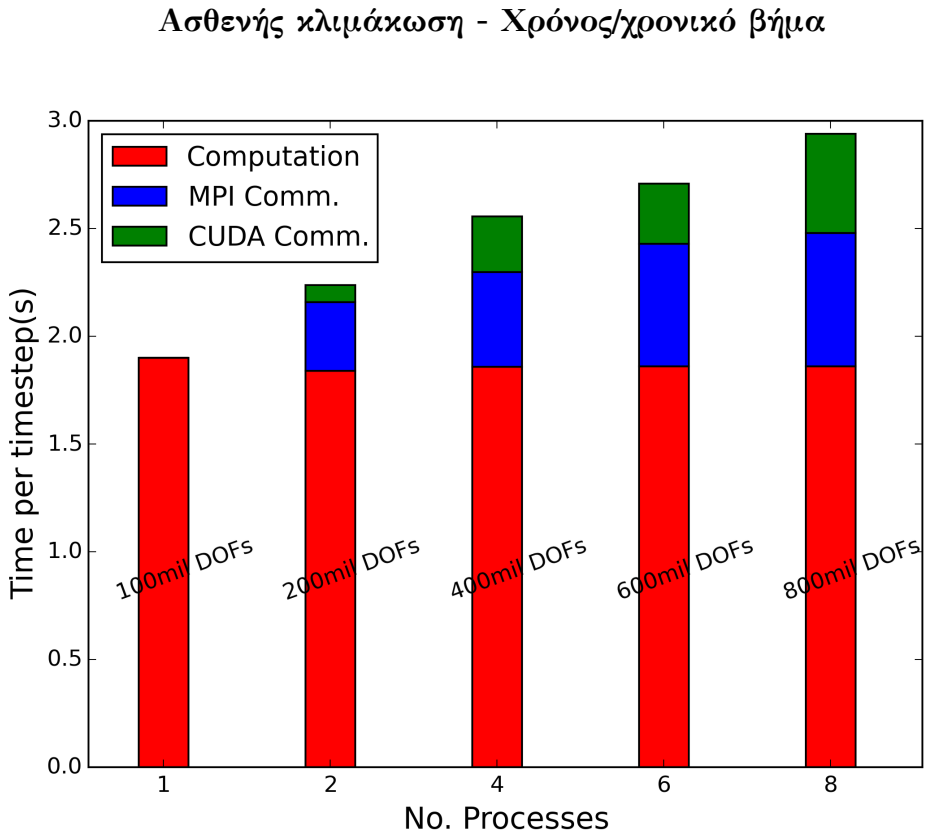


Σχήμα 4.7: Ισχυρή Κλιμάκωση χρόνου υπολογισμού-επικοινωνίας ανά χρονικό βήμα συναρτήσει του αριθμού των διεργασιών - GPUs. Επιλύεται πρόβλημα συνολικά $80 \cdot 10^6$ βαθμών ελευθερίας. Ο χρόνος επικοινωνίας περιλαμβάνει και το βήμα της επαναπαραμετροποίησης.

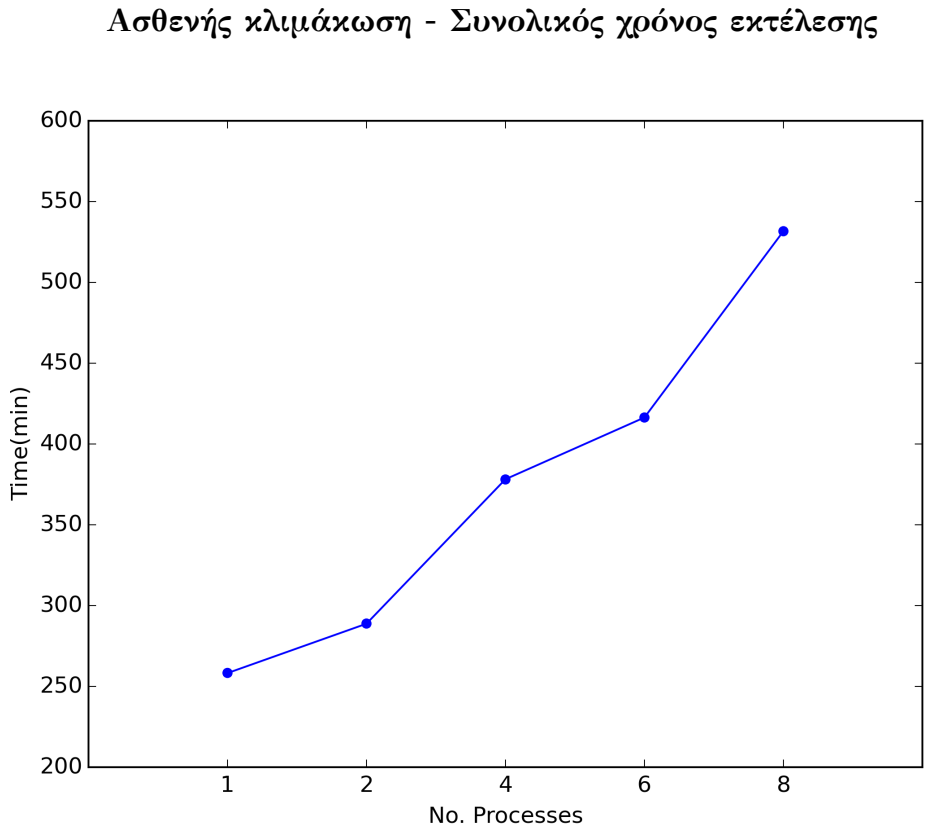


Σχήμα 4.8: Ισχυρή Κλιμάκωση συνολικού χρόνου εκτέλεσης συναρτήσει του αριθμού των διεργασιών - GPUs. Επιλύεται πρόβλημα συνολικά $80 \cdot 10^6$ *βαθμών ελευθερίας.*

Στο διάγραμμα 4.9, παρουσιάζεται η ασθενής κλιμάκωση των χρόνων υπολογισμών και επικοινωνίας συναρτήσει του αριθμού καρτών/διεργασιών για διάφορα μεγέθη προβλημάτων. Σε αυτή τη περίπτωση ο αριθμός των βαθμών ελευθερίας ανά διεργασία-κάρτα γραφικών διατηρείται σταθερός στους $100 \cdot 10^6$ (100 εικόνες ανά διεργασία). Σε αυτή τη περίπτωση είναι ξεκάθαρη η μηδενική επιβάρυνση του κόστους των υπολογισμών λόγω χρήσης περισσότερων διεργασιών, αφού κάθε διεργασία επεξεργάζεται τον ίδιο όγκο δεδομένων. Ωστόσο είναι εμφανής μια γραμμική αύξηση του χρόνου επικοινωνίας, ο οποίος ουσιαστικά οδηγεί στην επίσης γραμμική αύξηση του χρόνου εκτέλεσης ανά χρονικό βήμα. Στο διάγραμμα 4.10 παρατηρείται η επίσης αναμενόμενη μεγάλη αύξηση του συνολικού χρόνου για τη συγκεκριμένη περίπτωση. Τόσο το παραπάνω κόστος της επικοινωνίας όσο και η αύξηση του μεγέθους του προς επίλυση προβλήματος, οδηγούν σε μεγάλη αύξηση του συνολικού χρόνου εκτέλεσης.



Σχήμα 4.9: Ασθενής κλιμάκωση χρόνου υπολογισμού-επικοινωνίας ανά χρονικό βήμα συναρτήσει του αριθμού των διεργασιών - GPUs. Επιλύονται $100 \cdot 10^6$ *βαθμοί ελευθερίας ανά διεργασία. Ο χρόνος επικοινωνίας περιλαμβάνει και το βήμα της επαναπαραμετροποίησης.*



Σχήμα 4.10: Ασθενής κλιμάκωση συνολικού χρόνου εκτέλεσης συναρτήσει του αριθμού των διεργασιών - GPUs. Επιλύονται $100 \cdot 10^6$ *βαθμοί ελευθερίας ανά διεργασία.*

Η επικοινωνία μεταξύ των διεργασιών εξαιτίας της υποδομής της υπολογιστικής συστοιχίας (10 Gigabit Ethernet), μπορεί να γίνει με δύο τρόπους. Αν δύο διεργασίες τυγχάνει να βρίσκονται εντός του ίδιου υπολογιστικού κόμβου, οι διεργασίες επικοινωνούν εσωτερικά. Και οι δύο έχουν άμεση και πολύ γρήγορη πρόσβαση στην κεντρική μνήμη του συστήματος, επομένως τα μηνύματα αποστέλλονται και λαμβάνονται από τις διεργασίες με πολύ μικρή καθυστέρηση. Αν όμως οι διεργασίες βρίσκονται σε διαφορετικούς κόμβους, τότε τα μηνύματα πρέπει να περάσουν μέσω του δικτύου προκειμένου να φτάσουν στον προορισμό τους. Η διέλευση των μηνυμάτων μέσω των καρτών δικτύου (network adapters), στους διακόπτες δικτύου (switches) και στην (ethernet) καλωδίωση του δικτύου γενικότερα, προκαλεί μεγάλη καθυστέρηση στην αποστολή/λήψη μηνυμάτων εκτός των κόμβων. Εκτός αυτού η συγκεκριμένη υποδομή δικτύου δεν αφιερώνεται αποκλειστικά στις ανάγκες επικοινωνίας μεταξύ των κόμβων αλλά αποτελεί και υποδομή για το σύστημα αρχείων (filesystem), γεγονός το οποίο μπορεί να οδηγήσει σε ακόμα μεγαλύτερη καθυστέρηση στην αποστολή/λήψη μηνυμάτων.

Για λόγους πληρότητας στον πίνακα 4.6 καταγράφονται οι χρόνοι επικοινωνίας για την αποστολή/λήψη μηνυμάτων για αυτές τις δυο διαφορετικές περιπτώσεις επικοινωνίας. Οι μετρήσεις αφορούν μηνύματα μεγέθους 1 MB. Μηνύματα τέτοιου μεγέθους απαιτούνται για την αποστολή/λήψη πληροφορίας μεταξύ των γειτονικών κόμβων στη περίπτωση του προβλήματος που επιλύεται. Είναι προφανές ότι η εξωτερική επικοινωνία μεταξύ 2 διεργασιών που βρίσκονται σε διαφορετικούς κόμβους της συστοιχίας, κοστίζει 35 φορές περισσότερο. Σαν απόλυτοι χρόνοι επικοινωνίας και οι δύο χρόνοι δεν μπορούν να θεωρηθούν μεγάλοι. Ωστόσο το μεγάλο χρονικό κόστος προκύπτει εξαιτίας του γεγονότος ότι αποστέλλονται δεκάδες χιλιάδες μηνύματα τέτοιου είδους κατά τη διάρκεια της εκτέλεσης. Αυτά τα μηνύματα αφορούν και εσωτερική και εξωτερική επικοινωνία (ο συνολικός αριθμός διεργασιών πρέπει να είναι μεγαλύτερος από δύο για να υπάρξουν και οι δύο τρόποι επικοινωνίας). Τα μηνύματα που αποστέλλονται εσωτερικά, παραλαμβάνονται πολύ γρήγορα, κάτι που δεν συμβαίνει με τα μηνύματα που αποστέλλονται εξωτερικά μέσω του δικτύου. Αυτά λόγω της μεγαλύτερης καθυστέρησης θα παραληφθούν πιο αργά, και αυτά είναι που καθορίζουν την συνολική χρονική επιβάρυνση στον χρόνο επικοινωνίας.

Τρόπος επικοινωνίας	Καθυστέρηση
Εσωτερική (εντός κόμβου)	1 ms
Εξωτερική (μεταξύ κόμβων)	35 ms

Πίνακας 4.6: Χρόνοι καθυστέρησης (latency) κατά την αποστολή μηνυμάτων μεγέθους 1 MB με χρήση MPI.

Παρά την κακή κλιμάκωση του κώδικα, και δεδομένης της πολύ απαιτητικής σε υπολογισμούς φύσης του προβλήματος, οι χρόνοι εκτέλεσης μπορούν να θεωρηθούν αρκετά θετικοί. Όπως έχει ήδη παρουσιαστεί, ο χρόνος συνολικής εκτέλεσης του MATLAB κώδικα για πρόβλημα $40 \cdot 10^6$ βαθμών ελευθερίας είναι 500min. Στο διάγραμμα 4.10 παρατηρεί κανείς ότι έγινε εκτέλεση προβλήματος διπλάσιας διάστασης ($80 \cdot 10^6$ βαθμών ελευθερίας) σε περίπου 250min. Γίνεται εύκολα αντιληπτό ότι στο μισό χρόνο, είναι δυνατή πλέον η επίλυση διπλάσιου προβλήματος προβλήματος. Ειδικότερα, η μεγαλύτερη διάσταση προβλήματος που μπορεί να διαχειριστεί η CUDA-MPI υλοποίηση, δεδομένων των περιορισμών μνήμης των καρτών γραφικών, είναι της τάξης των $800 \cdot 10^6$ βαθμών ελευθερίας ($100 \cdot 10^6$ ανά κάρτα γραφικών), και χρειάζεται χρόνο της τάξης των 530min για την επίλυσή του. Ουσιαστικά με την CUDA-MPI υλοποίηση επιλύεται πρόβλημα **20** φορές μεγαλύτερης διάστασης στον ίδιο πρακτικά χρόνο με αυτόν της αρχικής MATLAB υλοποίησης.

Κεφάλαιο 5

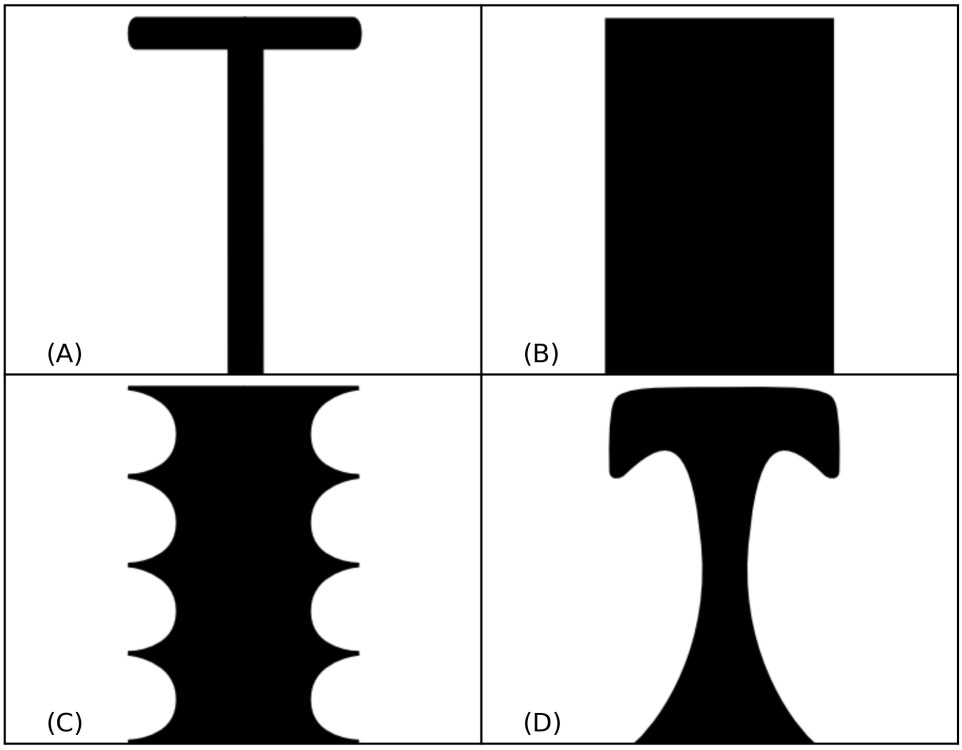
Αποτελέσματα

Εξετάζεται η υπερυδρόφοβη συμπεριφορά επιφανειών με διαφορετικά είδη μικρο-κολώνων. Η ανάλυση βασίζεται στον προσδιορισμό του MEP της CB-W μετάβασης και της οριζόντιας μετατόπισης της σταγόνας. Μέσω του υπολογισμού των ενεργειών των καταστάσεων του MEP υπολογίζονται τα ενεργειακά φράγματα που χρησιμεύουν στην αξιολόγηση της υπερυδρόφοβης συμπεριφοράς. Τέλος εξάγονται συμπεράσματα για την υπερυδρόφοβη συμπεριφορά των εξεταζόμενων επιφανειών, καθώς και για τις δυνατότητες των προγραμματιστικών υλοποιήσεων.

5.1 Μετάβαση CB-W

Τα είδη των εξεταζόμενων μικρο-κολώνων παρουσιάζονται στο διάγραμμα 5.1. Μέσω του υπολογισμού της ελεύθερης ενέργειας του MEP είναι δυνατός ο υπολογισμών ενεργειακών φραγμάτων που αφορούν μεταβάσεις της σταγόνας μεταξύ καταστάσεων ισορροπίας.

Το ενεργειακό φράγμα της μετάβασης CB-W ποσοτικοποιεί την "αντίσταση" που προβάλλει το σύστημα κατά τη μετάβαση και επομένως υψηλά CB-W φράγματα (μεγάλη αντίσταση) υποδηλώνουν εύρωστη υπερυδρόφοβη συμπεριφορά. Στην περίπτωση που το CB-W ενεργειακό φράγμα είναι χαμηλό, ευνοείται η κατάσταση ισορροπίας W και συνεπώς η υπερυδρόφοβη συμπεριφορά είναι επιρρεπής σε διαταραχές.



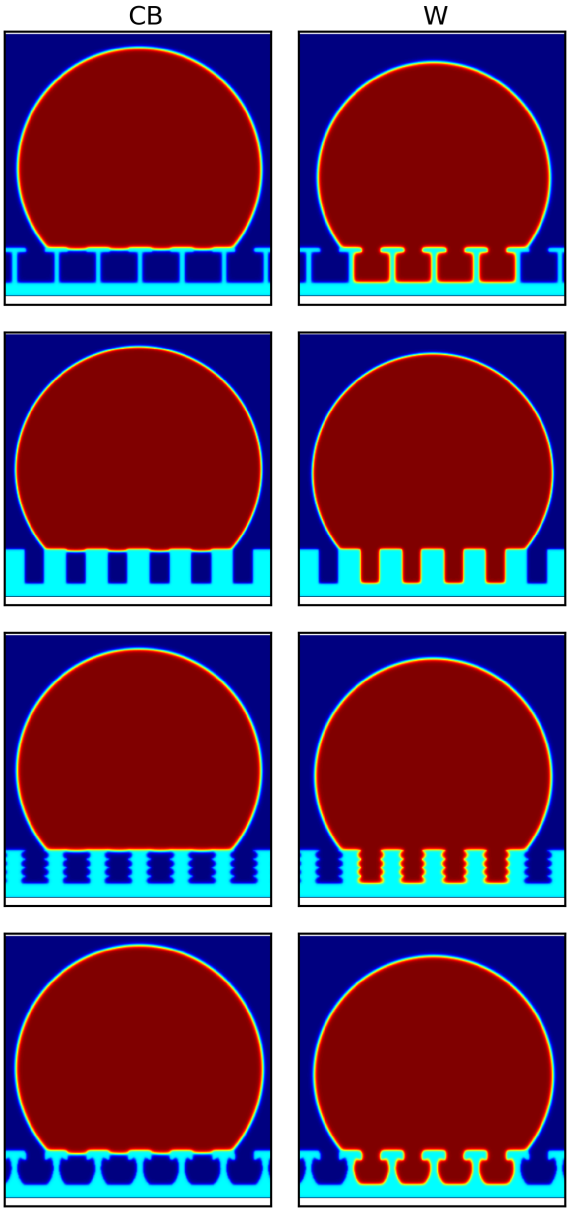
Σχήμα 5.1: Εξεταζόμενες μικρο-κολώνες. Οι μικρο-κολώνες είναι τοποθετημένες σε ίσες αποστάσεις κατά μήκος όλης της επιφάνειας. Όλες οι μικρο-κολώνες έχουν το ίδιο εμβαδό στην πάνω επιφάνειά τους.

Για πιο αναλυτικά αποτελέσματα, επιλέγεται ανάλυση του MEP σε συνολικά 80 εικόνες, ενώ η διακριτοποίηση του υπολογιστικού χωρίου παραμένει ίδια με αυτή του προβλήματος

στο κεφάλαιο 4.2. Οι παράμετροι επίλυσης συγκεντρώνονται στο πίνακα 5.1.

Παράμετρος	**Περιγραφή**	**Τιμή**
N^2	Σύνολο κόμβων	1024^2
θ_Y	Γωνία Young	112.5
ϵ	Πάχος διεπιφάνειας	0.002
N_STRING	Πλήθος εικόνων	80

Πίνακας 5.1: Πίνακας Παραμέτρων



Σχήμα 5.2: Καταστάσεις ισορροπίας CB και W για τις υπό μελέτη επιφάνειες.

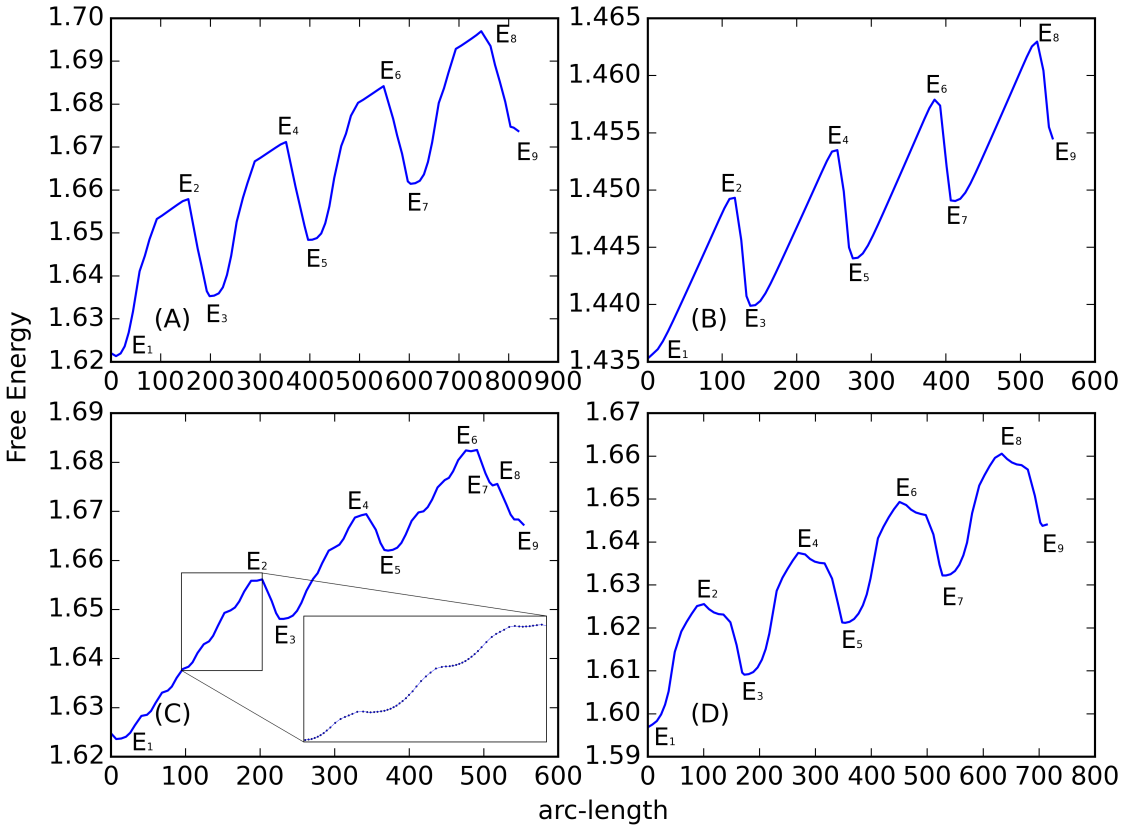
Η υπολογιζόμενη ενέργεια αδιαστατοποιείται χρησιμοποιώντας την ενέργεια αναφοράς $\sigma\Omega^2$, όπου σ είναι η επιφανειακή τάση του υγρού και Ω^2 είναι το εμβαδό του υπολογιστικού χωρίου. Η ενέργεια αποτυπώνεται συναρτήσει του μήκους τόξου του MEP. Το μήκος τόξου αποτελεί έναν πολύ καλό δείκτη της εξέλιξης της μετάβασης και έτσι το τελικό διάγραμμα αποτυπώνει την ενεργειακή κατάσταση της σταγόνας σε όλες τις ενδιάμεσες καταστάσεις του κατά μήκος του MEP. Τα αποτελέσματα παρουσιάζονται στο σχήμα 5.3.

Οι κορυφές (σάγματα) που εμφανίζονται στα ενεργειακά διαγράμματα αντιστοιχούν σε ασταθείς καταστάσεις διαβροχής ενώ τα ελάχιστα αντιστοιχούν σε ενδιάμεσες ευσταθείς καταστάσεις διαβροχής, όπου το υγρό έχει εισχωρήσει μερικώς στα αυλάκια της επιφάνειας. Στο σχήμα 5.3, για όλες τις υπό μελέτη επιφάνειες υπάρχουν 5 ευσταθείς καταστάσεις, που αντιστοιχούν σε ενεργειακά ελάχιστα E_1, E_3, E_5, E_7, E_9. Αντίστοιχα υπάρχουν 4 ασταθείς καταστάσεις,οι οποίες αντιστοιχούν στα ενεργειακά σάγματα E_2, E_4, E_6, E_8 και παρεμβάλλονται ανάμεσα στα ελάχιστα.

Το ενεργειακό φράγμα υπολογίζεται ως η διαφορά της ενέργειας της ευσταθούς κατάστασης διαβροχής από την ενέργεια της κατάντη (στην κατεύθυνση της μετάβασης) ασταθούς κατάστασης. Εάν παρεμβάλλονται ενδιάμεσες ευσταθείς καταστάσεις διαβροχής τότε το ενεργειακό φράγμα υπολογίζεται ως το άθροισμα όλων των επιμέρους ενεργειακών φραγμάτων. Έτσι, για τη περίπτωση του συστήματος Α, τα επιμέρους ενεργειακά φράγματα που παρουσιάζονται (βλ. Σχήμα 5.3) είναι τα $E_2-E_1, E_4-E_3, E_6-E_5, E_8-E_7$. Το ενεργειακό φράγμα της CB-W μετάβασης υπολογίζεται ως : $E_{CB-W} = (E_2-E_1)+(E_4-E_3)+(E_6-E_5)+(E_8-E_7)$. Αυτή η μεθοδολογία, εφαρμόζεται και στα συστήματα Β και D. Το σύστημα C αποτελεί μια ειδικότερη περίπτωση, καθώς σε αντίθεση με τα υπόλοιπα συστήματα, υπάρχει εισχώρηση της σταγόνας στις μικρο-αυλακώσεις των πλαϊνών τοιχωμάτων, έχοντας σαν αποτέλεσμα την ύπαρξη σημαντικά περισσότερων ενδιάμεσων καταστάσεων ισορροπίας.

Ο συνυπολογισμός όλων των επιμέρους ενεργειακών φραγμάτων του MEP, αποτελεί το ελάχιστο ποσό ενέργειας για μια οιονεί-στατική μετάβαση της σταγόνας από την CB στη W κατάσταση. Μια τέτοιου είδους μετάβαση υπονοεί την κατανάλωση ενέργειας όταν το σύστημα προσπελάσει ενεργειακά ευνοϊκά (κατηφορικά) κομμάτια του MEP. Εν αντιθέσει, αν το σύστημα διατηρεί ενέργεια, τότε κατά τη προσπέλαση των κατηφορικών κομματιών του MEP, αυξάνεται η ορμή της σταγόνας και κάνουν την εμφάνισή τους φαινόμενα αδράνειας. Σε αυτή τη περίπτωση η ενέργεια που συσσωρεύεται θα είναι επαρκής προκειμένου να ξεπεραστούν τα επερχόμενα επιμέρους ενεργειακά φράγματα και έτσι η απαραίτητη

ενέργεια της CB-W μετάβασης είναι το μέγιστο επιμέρους ενεργειακό φράγμα. Με παρουσία φαινομένων αδράνειας (π.χ σε λεπτόρρευστες σταγόνες) το ενεργειακό φράγμα του συστήματος A υπολογίζεται $E^*_{CB-W} = \max(E_2 - E_1, E_4 - E_3, E_6 - E_5, E_8 - E_7)$



Σχήμα 5.3: Διαγράμματα ελεύθερης ενέργειας συναρτήσει του μήκους τόξου του MEP. Τα διαγράμματα αφορούν τις γεωμετρίες που παρουσιάστηκαν στο διάγραμμα 5.1.

Τα υπολογιζόμενα ενεργειακά φράγματα παρουσία και ελλείψει φαινομένων αδράνειας, παρουσιάζονται στον πίνακα 5.2.

Σύστημα	**Energy Barrier (quasi-static)**	**Energy Barrier (inertial effects)**
A	0.14347	0.03677
B	0.05617	0.01462
C	0.07184	0.01796
D	0.11375	0.02888

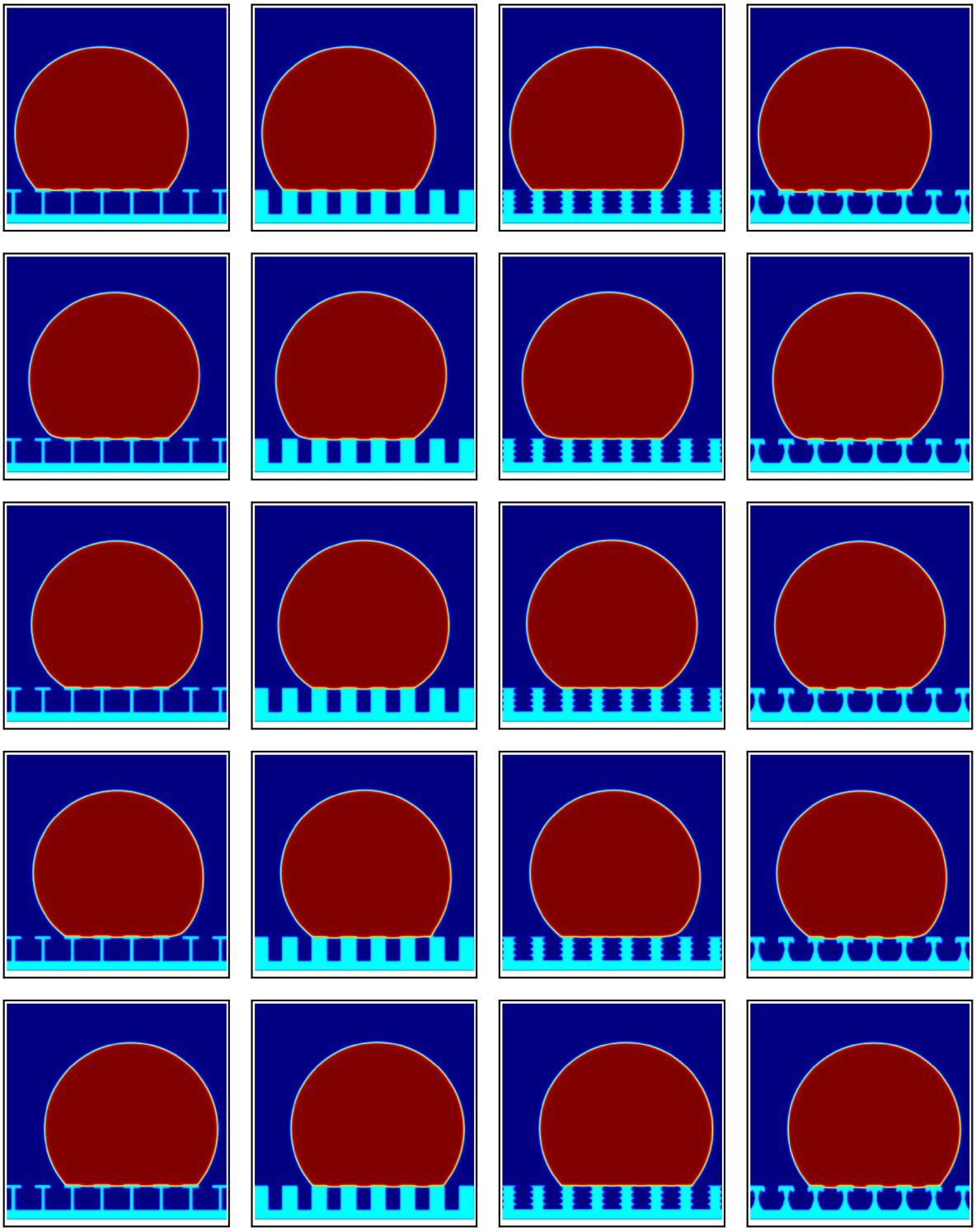
Πίνακας 5.2: Ενεργειακά φράγματα για την CB-W μετάβαση για τις υπό μελέτη επιφάνειες (Σχήμα 5.1).

Στο διάγραμμα 5.3 παρατηρείται ότι η επιφάνεια *A* παρουσιάζει το μέγιστο ενεργειακό φράγμα όσον αφορά τη CB-W μετάβαση. Αυτή η γεωμετρία που προσεγγίζει το σχήμα μιας καρφίτσας, παρέχει τόσο την απαραίτητη οριζόντια επιφάνεια για τη στήριξη της σταγόνας, αλλά ταυτόχρονα, λόγω της πολύ λεπτής της βάσης, δημιουργούνται αυλακώσεις με αρκετά μεγαλύτερο όγκο απ'ότι οι άλλες επιφάνειες. Το γεγονός αυτό, οδηγεί σε ενδιάμεσες ευσταθείς καταστάσεις με τη μικρότερη ενέργεια (σταθερότερες), με αποτέλεσμα τα μερικά ενεργειακά φράγματα να είναι υψηλότερα. Η συμπεριφορά αυτή επιβεβαιώνεται και από τις υπόλοιπες επιφάνειες. Κατά σειρά ελαττωμένου όγκου οι επιφάνειες κατατάσσονται με την ακόλουθη σειρά: A, D, C, B. Αυτή είναι και η σειρά με την οποία μειώνονται τα ενεργειακά φράγματα, επιβεβαιώνοντας το συλλογισμό.

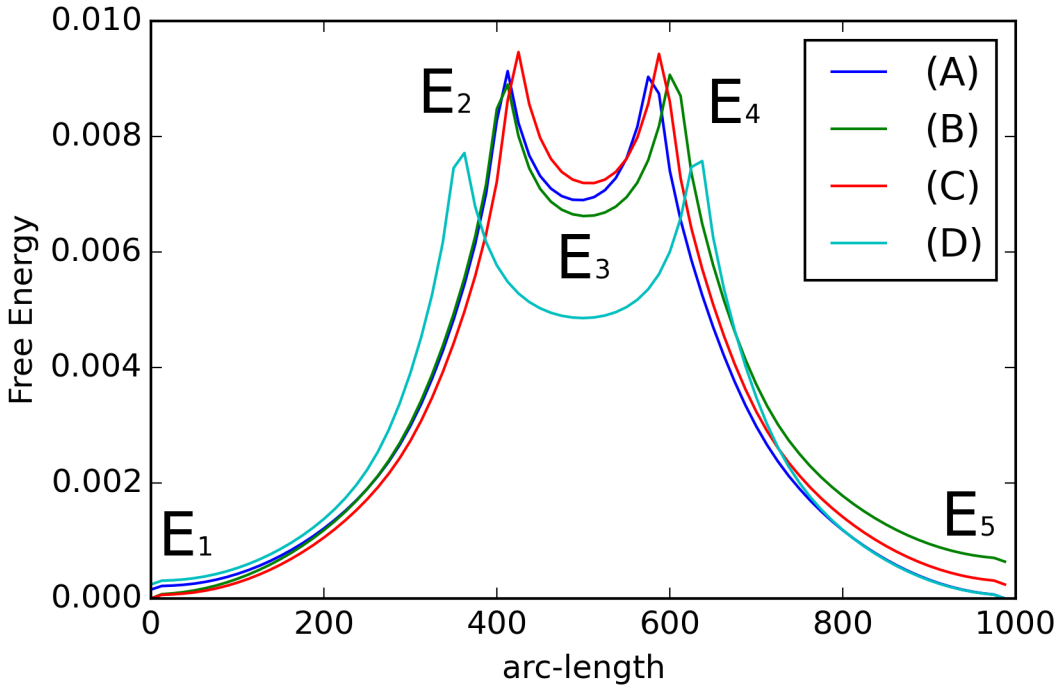
5.2 Οριζόντια μετατόπιση

Η CB-W μετάβαση και η ενεργειακή της ανάλυση, αποκαλύπτουν την υπερυδρόφοβη συμπεριφορά της επιφάνειας. Ωστόσο η ύπαρξη μεγάλων ενεργειακών φραγμάτων ενδέχεται να έχει αρνητικό αντίκτυπο στη κινητικότητα της σταγόνας. Για το λόγο αυτό θεωρείται σημαντική η μελέτη της περίπτωσης οριζόντιας κίνησης της σταγόνας. Η κίνηση αφορά την απλή οριζόντια μετακίνηση κατά μια αυλάκωση προς τα δεξιά.

Τα αποτελέσματα των επιλύσεων για όλες τις γεωμετρίες παρουσιάζονται στο διάγραμμα 5.4. Για όλες τις υπό μελέτη προεξοχές, παρατηρείται η ίδια συμπεριφορά κίνησης. Αρχικά η σταγόνα στηρίζεται σε πέντε προεξοχές (E_1). Κινούμενη προς τα δεξιά, αρχικά αποκολλάται από την αριστερότερη προεξοχή (ανάντη), και στηρίζεται πλέον σε τέσσερις προεξοχές (E_2). Ακολουθεί η προσκόλλησή της στην κατάντη προεξοχή (E_4) και καταλήγει σε νέα ευσταθή κατάσταση ισορροπίας (E_5).



Σχήμα 5.4: Εξέλιξη κίνησης κατά μήκος της επιφάνειας για όλες τις γεωμετρίες. Από πάνω προς τα κάτω παρουσιάζεται η σταδιακή μετακίνηση της σταγόνας προς τα δεξιά. Για όλες τις γεωμετρίες παρατηρείται η ίδια συμπεριφορά, με τη σταγόνα αρχικά να αποκολλάται από την ανάντη προεξοχή και έπειτα να προσκολλάται στην κατάντη προεξοχή.



Σχήμα 5.5: Διαγράμματα ελεύθερης ενέργειας συναρτήσει του μήκους τόξου του MEP. Τα διαγράμματα αφορούν τις γεωμετρίες που παρουσιάστηκαν στο διάγραμμα 5.1. Έχει γίνει κατάλληλη μετακίνηση έτσι ώστε να είναι πιο προφανής η σύγκριση των ενεργειακών φραγμάτων.

Τα ενεργειακά φράγματα αυτής της κίνησης υπολογίζονται πάλι σαν τα μερικά αθροίσματα των επιμέρους ενεργειακών φραγμάτων. Σύμφωνα με το σχήμα 5.4 αυτά θα είναι τα $E_2 - E_1, E_4 - E_3$. Έτσι το ενεργειακό φράγμα ολόκληρης της μετάβασης, ή διαφορετικά της μετάβασης από τη κατάσταση 1 στη κατάσταση 5 υπολογίζεται $E_{1\to 2} = (E_2 - E_1) + (E_4 - E_3)$. Ο υπολογισμός αυτός σε αναλογία με τη CB-W μετάβαση αφορά περιπτώσεις κίνησης χωρίς φαινόμενα αδράνειας. Αν υπάρχουν αδρανειακά φαινόμενα και συνεπώς διατήρηση ενέργειας κατά τη κίνηση της σταγόνας, τότε το απαιτούμενο ενεργειακό φράγμα για την ολοκλήρωση της μετάβασης, θα είναι το μέγιστο επιμέρους ενεργειακό φράγμα. Στη συγκεκριμένη περίπτωση, η τιμή του ενεργειακού φράγματος θα είναι ουσιαστικά ίση με την τιμή του πρώτου επιμέρους ενεργειακού φράγματος αφού αυτό είναι το μεγαλύτερο επιμέρους φράγμα, δηλαδή: $E^*_{1\to 2} = E_2 - E_1$. Οι υπολογιζόμενες τιμές των ενεργειακών φραγμάτων κατά την οριζόντια μετατόπιση παρουσιάζονται στον πίνακα 5.3.

Σύστημα	**Energy Barrier (quasi-static)**	**Energy Barrier (inertial effects)**
A	0.0113	0,0091
B	0.0113	0.0089
C	0.0117	0.0094
D	0.0104	0.0076

Πίνακας 5.3: Ενεργειακά φράγματα για την οριζόντια κίνηση της σταγόνας για τις υπό μελέτη επιφάνειες (Σχήμα 5.1).

Σύμφωνα με το σχήμα 5.5 οι γεωμετρίες A,B,C έχουν περίπου τα ίδια ενεργειακά φράγματα όσον αφορά την οριζόντια κίνηση της σταγόνας. Στη περίπτωση αυτή παρατηρείται ότι το σχετικά υψηλό ενεργειακό φράγμα της CB-W μετάβασης, πράγματι προκαλεί επίσης μεγάλη επιβάρυνση στη κινητικότητα της σταγόνας. Όπως έχει ήδη αναφερθεί, είναι επιθυμητή η ύπαρξη μικρών ενεργειακών φραγμάτων στη περίπτωση της οριζόντιας κίνησης, έτσι ώστε η κινητικότητα της σταγόνας να είναι όσο το δυνατόν μεγαλύτερη. Από τα εξεταζόμενα συστήματα, η γεωμετρία τύπου D είναι αυτή που παρουσιάζει ενεργειακό φράγμα αρκετά χαμηλότερο σε σχέση με τις υπόλοιπες ενώ ταυτόχρονα παρουσιάζει σχετικά υψηλό ενεργειακό φράγμα κατά την CB-W μετάβαση. Επομένως αυτός ο τύπος επιφάνειας ενδείκνυται για την χρήση του στο σχεδιασμό επιφανειών όπου είναι επιθυμητή η ταχεία και εύκολη κίνηση της σταγόνας πάνω στην επιφάνεια.

5.3 Συμπεράσματα

Αναπτύχθηκε κώδικας για την εύρεση του μονοπατιού ελάχιστης ενέργειας (MEP) για μεταβάσεις σταγόνων πάνω σε υπερυδρόφοβες επιφάνειες. Ο συνδυασμός της μεθόδου χορδής και της αναπαράστασης πεδίου φάσεων επιτρέπει τον υπολογισμό του MEP που δίνει τα ενεργειακά φράγματα των μεταβάσεων σε καταστάσεις διαβροχής. Οι FFTs (Fast Fourier Transforms) που εφαρμόζονται από τη μέθοδο μετατρέπουν τις μερικές διαφορικές εξισώσεις του προβλήματος σε απλές αλγεβρικές εξισώσεις οι οποίες επιλύονται πολύ γρήγορα με χρήση καρτών γραφικών. Η πλατφόρμα CUDA δίνει τη δυνατότητα για αξιοποίηση των μέγιστων δυνατοτήτων των καρτών γραφικών με αποτέλεσμα την πολύ καλή απόδοση του κώδικα. Ο κώδικας ενισχύεται με τη χρήση του πρωτοκόλλου MPI, που δίνει δυνατότητες κλιμάκωσης της εκτέλεσης σε πολλούς υπολογιστικούς κόμβους καθώς και επίλυση προβλημάτων πολύ μεγαλύτερου μεγέθους που δεν θα μπορούσαν να αντιμετωπιστούν διαφορετικά. Η MPI-CUDA υλοποίηση εμφανίζει πολύ καλά αποτελέσματα επιτάχυνσης σε σύγκριση με προηγούμενο κώδικα. Όσον αφορά την ισχυρή και ασθενή κλιμάκωση, η επιτυγχανόμενη επιτάχυνση εξαρτάται από το μέγεθος του προβλήματος. Για μικρά προβλήματα δεν υπάρχει λόγος για χρήση πολλών καρτών γραφικών, καθώς το κόστος επικοινωνίας είναι μεγάλο. Με βάση το υπάρχον υλικό, το μεγαλύτερο μέγεθος προβλήματος που επιλύθηκε αποτελείται από 800 εκατομμύρια βαθμούς ελευθερίας και επιλύεται σε σχεδόν 9 ώρες. Ο χρόνος εκτέλεσης θεωρείται αρκετά μικρός αναλογικά με το μέγεθος των προβλημάτων που επιλύονται και σε σχέση με τους διαθέσιμους υπολογιστικούς πόρους.

Η MPI-CUDA υλοποίηση εφαρμόζεται για τον υπολογισμό των MEPs τόσο σε μεταβάσεις από υπερυδρόφοβες καταστάσεις (Cassie-Baxter, CB) σε υδρόφιλες καταστάσεις (Wenzel, W), όσο και σε οριζόντιες μετατοπίσεις σταγόνων πάνω σε επιφάνειες. Η μεγάλη ταχύτητα που προσφέρει η συγκεκριμένη υλοποίηση επιτρέπει την επίλυση πολλών τέτοιου είδους προβλημάτων σε μικρό χρόνο. Οι υπολογισμοί γίνονται για δομημένες επιφάνειες με συγκεκριμένου τύπου μικρο-κολώνες. Οι μικροκολώνες σχήματος καρφίτσας (μικροκολώνα A στο σχήμα 5.1) προσδίδουν υψηλά ενεργειακά φράγματα για όλες τις περιπτώσεις μεταβάσεων, (μετάβαση CB-W και οριζόντια μετακίνηση), ενώ ειδικότερα η γεωμετρία τύπου μανιταριού (μικροκολώνα D στο σχήμα 5.1) προσδίδει αρκετά υψηλά ενεργειακά φράγματα όσον αφορά τη μετάβαση CB-W αλλά και πολύ χαμηλότερα ενεργειακά φράγματα όσον αφορά την οριζόντια μετατόπιση, καθιστώντας την ιδανική για εφαρμογές που απαιτούν υψηλή κινητικότητα της σταγόνας.

5.4 Προτάσεις για μελλοντική εργασία

Αρχικά η προγραμματιστική υλοποίηση και ειδικότερα η MPI-CUDA υλοποίηση επιδέχεται πολλές βελτιώσεις. Όπως έχει προαναφερθεί, ο κύριος περιορισμός της υλοποίησης είναι η διαθέσιμη μνήμη στις κάρτες γραφικών της συστοιχίας. Κύριο αποτέλεσμα αυτού του περιορισμού είναι η αδυναμία της υλοποίησης να αντιμετωπίζει προβλήματα πολύ μεγαλύτερης τάξης μεγέθους. Ο μοναδικός τρόπος για να ξεπεραστεί αυτός ο περιορισμός είναι η εκμετάλλευση της διαθέσιμης μνήμης των κόμβων της συστοιχίας. Προς το παρόν, χρησιμοποιώντας τις 8 κάρτες γραφικών της συστοιχίας, με $3GB$ μνήμης στη καθεμιά, υπάρχουν συνολικά $24GB$ διαθέσιμα για την επίλυση προβλημάτων. Η απαίτηση αυτή αντιστοιχεί στο μέγιστο μέγεθος προβλήματος που επιλύθηκε στα πλαίσια αυτής της εργασίας (διακριτοποίηση 800 εικόνων στη χορδή σε υπολογιστικό χωρίο με συνολικό αριθμό κόμβων $N = 1024x1024$). Αν υπάρξει εκμετάλλευση της κύριας μνήμης (RAM) των κόμβων της συστοιχίας, η συνολικά διαθέσιμη μνήμη ανέρχεται στα $64GB$ (Κεφάλαιο 4.1) και μπορεί να διατεθεί για την επίλυση προβλημάτων με 2500 εικόνες στη χορδή. Χονδρικά υπάρχει δυνατότητα για επίλυση προβλήματος με συνολικούς βαθμούς ελευθερίας της τάξης των 2.5 δισεκατομμυρίων. Το επιπλέον κόστος αυτής της βελτιστοποίησης αφορά την αντιγραφή δεδομένων που πρέπει να γίνεται μεταξύ της μνήμης των καρτών γραφικών και της κύριας μνήμης του συστήματος. Η μεταφορά δεδομένων από και προς την κάρτα γραφικών, γίνεται σχετικά γρήγορα, ωστόσο απαιτείται προσεκτικός προγραμματισμός έτσι ώστε να μην υπάρξει μεγάλο αντίκτυπο στην απόδοση της υλοποίησης.

Σκοπός των προγραμματιστικών υλοποιήσεων είναι τελικά η αξιολόγηση υπερυδρόφοβων επιφανειών. Προς το παρόν, η γεωμετρία της επιφάνειας αποτελεί εξωτερικό δεδομένο για την υλοποίηση. Αυτό σημαίνει ότι σε περίπτωση που μελετάται ένα συγκεκριμένο είδος επιφάνειας και αναζητείται η βέλτιστη μορφολογία της, αυτή πρέπει να γίνει διαισθητικά από τον χρήστη. Η υλοποίηση ενός κύκλου βελτιστοποίησης εντός του κώδικα σε συνδυασμό με την επίλυση του προβλήματος, μπορεί να επιτρέψει τον προσδιορισμό βέλτιστης μορφολογίας αυλακώσεων για τις αυλακωτές επιφάνειες, που θα οδηγήσουν σίγουρα σε καλύτερη υπερυδρόφοβη συμπεριφορά. Μια τέτοιου είδους μελέτη περιλαμβάνει τόσο την υλοποίηση κώδικα για την αυτόματη κατασκευή (π.χ. χρησιμοποιώντας splines) επιφανειών με βάση συγκεκριμένες παραμέτρους, όσο και την μαθηματική επεξεργασία του τελικού MEP, έτσι ώστε να εξαχθούν συμπεράσματα και αλληλοσυσχετίσεις μεταξύ της υπερυδρόφοβης συμπεριφοράς και των γεωμετρικών παραμέτρων της επιφάνειας.

Βιβλιογραφία

[1] G. Pashos, G. Kokkoris, and A.G. Boudouvis. A modified phase-field method for the investigation of wetting transitions of droplets on patterned surfaces. *Journal of Computational Physics*, 283:258–270, 2015.

[2] G. Pashos, G. Kokkoris, and A.G. Boudouvis. Minimum energy paths for wetting transitions on grooved surfaces. *Langmuir*, 31:3059–3068, 2015.

[3] R. Backofen and A. Voigt. A phase-field-crystal approach to critical nuclei. *Journal of Physics: Condensed Matter*, 22, 2010.

[4] T. Verho, J.T. Korhonen, L. Sainiemi, V. Jokinen, C. Bower, K. Franze, S. Franssila, P. Andrew, O. Ikalla, and R. H. A. Ras. Reversible switching between superhydrophobic states on hierachically structured surface. *Proceedings of the National Academy of Sciences U.S.A*, 109:10210–10213, 2012.

[5] K. Ellinas, A. Tserepi, and E. Gogolides. Superhydrophobic, passive microvalves with controllable opening threshold: Exploiting plasma nanotextured microfluids for a programmable flow switchboard. *Microfluid*, *Nanofluid*, pages 1–10, 2014.

[6] K. Tsougeni, D. Papageorgiou, A. Tserepi, and E. Gogolides. "Smart" polymeric microfluids fabricated by plasma processing: Controlled wetting, capillary filling and hydrophobic valving. *Lab Chip*, 10:462–469, 2010.

[7] G. Henkelman and H. Jonsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *The Journal of Chemical Physics*, 113:16080–16090, 2007.

[8] D. Sheppard, R. Terrell, and G. Henkelman. Optimization methods for finding minimum energy paths. *The Journal of Chemical Physics*, 128, 2008.

[9] R. Granot and R. Baer. A spline for your saddle. *The Journal of Chemical Physics*, 128, 2008.

[10] W. Quapp. Chemical reaction paths and calculus of variations. *Theoretical Chemistry Accounts*, 121:227–237, 2008.

[11] Y. Li and W. Ren. Numerical study of vapor condensation on patterned hydrophobic surfaces using the string method. *Langmuir*, 30:9567–9576, 2014.

[12] G. Henkelman, B. P. Uberuaga, and H. Jonsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *The Journal of Chemical Physics*, 113:9901–9904, 2000.

[13] G. Henkelman and H. Jonsson. A dimer method for finding saddle points on higher dimensional potential surfaces using only first derivatives. *The Journal of Chemical Physics*, 111:7010–7022, 1999.

[14] M. E. Kavousanakis, C. E. Colosqui, and A. G. Papathanasiou. Engineering the geometry of stripe-patterned surfaces toward efficient wettability switching. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 436:309–317, 2013.

[15] P. Yue, C. Zhou, J.J. Feng, C.F. Olivier-Gooch, and H. Hu. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219:47–67, 2006.

[16] W. Ren, E. Venden-Eijnden, and E. Weinan. String method for the study of rare events. *Physical Review B*, 66, 2002.

[17] W. Ren, E. Venden-Eijnden, and E. Weinan. Simplified and improved string method for computing the minimum energy paths in barrier-crossing events. *The Journal of Chemical Physics*, 126, 2007.

[18] W. Ren. Wetting transition on patterned surfaces and energy barriers. *Langmuir*, 30:2879–2885, 2014.

[19] P. Papadopoulos, L. Mammen, X. Deng, D. Vollmer, and H.-J. Butt. How superhydrophobicity breaks down. *Proceedings of the National Academy of Sciences U.S.A*, 110:3254–3258, 2013.

[20] P. Yue, C. Zhou, J.J. Feng, C. F. Olivier-Gooch, and H. H. Hu. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219:47–67, 2006.

[21] X. Yang, J. J. Feng, C. Liu, and J. Shen. Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method. *Journal of Computational Physics*, 218:417–428, 2006.

[22] P. T. Yue, J. J. Feng, C. Liu, and J. Shen. A diffuse-interface method for simulating two-phase flows of complex fluids. *Journal of Fluid Mechanics*, 515:293–317, 2004.

[23] V. V. Khatavkar, P. D. Anderson, and H. E. H. Meijer. Capillary spreading of a droplet in the partially wetting regime using a diffuse-interface model. *Journal of Fluid Mechanics*, 572:367–387, 2007.

[24] P. Yue, C. Zhou, and J. J. Feng. Spontaneous shrinkage of drops and mass conservation in phase-field simulations. *Journal of Computational Physics*, 223:1–9, 2007.

[25] P. Yue and Y. Renardy. Spontaneous penetration of a non-wetting drop into an exposed pore. *Physics of Fluids*, 25, 2013.

[26] C. Yang, C. Huang, and C. Lin. Hybrid cuda, openmp and mpi parallel programming on multicore gpu clusters. *Computer Physics Communications*, 182:266–269, 2011.

[27] Y. Yuan and T.R. Lee. Contact angle and wetting properties. *Surface Science Techniques*, 51:3–34, 2013.

[28] Cuda. `http://www.nvidia.com/object/cuda_home_new.html`.

[29] Nvidia. `http://www.nvidia.com`.

[30] Andromeda. `http://febui.chemeng.ntua.gr/andromeda.htm`.