ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΟΡΥΚΤΩΝ ΠΟΡΩΝ

Παραλληλοποιήσιμες Πολυπλεγματικές τεχνικές σε επιλυτές *Navier – Stokes* για την μοντελοποίηση ασυμπίεστων ροών

Διδακτορικη Διατριβη

Βασιλειος Γ. Μανδικας

Ιούνιος, 2017

Περίληψη

Οι Navier-Stokes ρευστοδυναμικές εξισώσεις για ασυμπίεστες ή συμπιεστές ροές είναι γνωστές τα τελευταία εκατό χρόνια. Ο μη-γραμμικός χαρακτήρας τους καθιστά αδύνατη την αναλυτική επίλυση των προβλημάτων μοντελοποίησης, τα οποία παρουσιάζονται για παράδειγμα στις ερευνητικές περιοχές της Αεροδιαστημικής, Βιοϊατρικής και Εμβιομηχανικής. Οι αναλυτικές λύσεις εξισώσεων κίνησης είναι γνωστές μόνο για ένα πολύ περιορισμένο αριθμό στοιχειωδών ροών και για απλές γεωμετρίες με συγκεκριμένες αρχικές/οριακές συνθήκες. Έτσι είναι αναγκαία η χρήση αριθμητικών μεθόδων επίλυσης για την προσέγγιση των λύσεων τους. Οι αριθμητικές αυτές μέθοδοι υλοποιούνται σε σύγχρονα υπολογιστικά περιβάλλοντα ιδιαίτερα για ρεαλιστικές εφαρμογές. Από τις αρχές του 1980 υπήρξε μια μεγάλη ανάπτυξη αριθμητικών μεθόδων επίλυσης προβλημάτων Ρευστοδυναμικής (Computational Fluid Dynamics-CFD). Παρόλαυτα, ακόμα και στις μέρες μας, η ανάπτυξη υψηλής ακρίβειας αριθμητικών σχημάτων επίλυσης των εξισώσεων Navier-Stokes για ασυμπίεστες ροές είναι αναγκαία για την προσομοίωση μεγάλου ενδιαφέροντος προβλημάτων. Για παράδειγμα, η μοντελοποίηση της ροής γύρω από υδροπτέρυγα, πτέρυγες ανεμογεννητριών και αεροσκαφών, κατά την διαδικασία απογείωσης και προσγείωσης, της ροής αίματος και ροής του αέρα γύρω από πουλιά και έντομα κατά το πέταγμα τους. Τέτοια προβλήματα ροών μπορούν να αντιμετωπιστούν μόνο με την χρήση υψηλής ακρίβειας αριθμητικών μεθόδων. Όμως, η υψηλή ακρίβεια που απαιτείται αυξάνει συνήθως εκθετικά τον υπολογιστικό χρόνο επίλυσης. Ο συνδυασμός υψηλής ποιότητας προσομοιώσεων ασυμπίεστων ροών σε εύλογο χρονικό διάστημα απεικόνισης των ταχυτήτων ροής, αποτέλεσε το κίνητρο για την συγγραφή της παρούσας διδακτορικής διατριβής. Αντικείμενο της αποτελεί η ανάπτυξη, ανάλυση και εφαρμογή ενός αποτελεσματικού επιλυτή ασυμπίεστων μεταβατικών τυρβωδών ροών, ο οποίος θα εκμεταλλεύεται τις σύγχρονες πολυεπεξεργαστικές υπολογιστικές αρχιτεκτονικές με επιταχυντές υπολογισμών. Για την αποδοτική αριθμητική επίλυση προβλημάτων ασυμπίεστων ροών έχουν επικρατήσει μέθοδοι διόρθωσης της πίεσης και της ψευδοσυμπιεστότητας, με την δεύτερη να μην ενδείκνυται για μη μόνιμες ροές. Η αριθμητική μέθοδος που θα αναπτυχθεί θα βασιστεί στην εισαγωγή της πίεσης στην εξίσωση συνέχειας (μέθοδος διόρθωσης πίεσης) ενσωματώνοντας την τεχνική πολλαπλών πλεγμάτων (multigrid technique) στον αριθμητικό επιλυτή διακριτοποίησης. Ειδικότερα, γίνεται χρήση υψηλής τάξης ακρίβειας συμπαγών αριθμητικών σχημάτων πεπερασμένων διαφορών για μετατοπισμένα πλέγματα διακριτοποίησης (staggered grids). Τεχνικές παραλληλοποίησης της επαναληπτικής διαδικασίας επίλυσης του παραγόμενου αραιού και γενικού γραμμικού συστήματος εφαρμόζονται για την αποδοτική υλοποίηση της μεθόδου. Τα συμπαγή σχήματα πεπερασμένων

διαφορών (compact or implicit schemes) έχουν το πλεονέκτημα, ότι επιτυγχάνουν σε μικρό εύρος πλέγματος μεγαλύτερη τάξη ακρίβειας, σε σχέση με τα κλασικά σχήματα πεπερασμένων διαφορών. Η ιδιότητα αυτή είναι χρήσιμη για την εφαρμογή του σχήματος κοντά στα όρια του υπολογιστικού χωρίου. Μία καινοτομία της προτεινόμενης μεθόδου είναι η επιβολή οποιοδήποτε είδους συνοριακών συνθηκών (Dirichlet, Neumann, Robin and mixed συνοριακές συνθήκες) στον αριθμητικό επιλυτή. Επίσης, σε αντίθεση με τα κλασικά σχήματα πεπερασμένων διαφορών, η χρήση συμπαγών σχημάτων ελαχιστοποιεί το ποσοστό της αριθμητικής διάχυσης, ελαχιστοποιώντας τη διαφορά μεταξύ πραγματικών και αριθμητικών κυματικών αριθμών σε όλες τις διαδιδόμενες στο πλέγμα συχνότητες. Η ελαχιστοποίηση της αριθμητικής διάχυσης είναι επιθυμητή ιδιότητα του αριθμητικού σχήματος, ιδιαίτερα στις περιπτώσεις προσομοίωσης αριθμητικής μετάδοσης κυμάτων (π.χ. αερο-ακουστική και ηλεκτρομαγνητισμός), αλλά και σε προβλήματα με ευρύ συχνοτικό φάσμα ή σε προβλήματα που πραγματοποιούνται ταυτόχρονα σε ευρύ φάρμα χωρικών κλιμάκων.

Στην αρχή της παρούσας διατριβής, περιγράφεται η κατασκευή ένος αριθμητικού επιλυτή για ελλειπτικού τύπου ΜΔΕ στις δύο χωρικές διαστάσεις, ο οποίος είναι ικανός να αντιμετωπίσει Προβλήματα Συνοριακών Τιμών με ανισοτροπίες. Το αριθμητικό σχήμα είναι κατάλληλο για γεωμετρικές διαμερίσεις με τους αγνώστους κατανεμημένους στα κέντρα των υπολογιστικών κελιών (cell-centered grids). Επιπλέον, για την εφαρμογή της τεχνικής Πολυπλέγματος κατασκευάζονται πρωτότυποι υψηλής ακρίβειας τελεστές μεταφοράς της πληροφορίας από το πυκνό στο αραιό πλέγμα και αντίστροφα. Ειδικότερα, οι τελεστές αυτοί εμπλέκουν λιγότερες αριθμητικές πράξεις σε σχέση με τους συνηθισμένους για cell-centered διακριτοποιήσεις. Στη συνέχεια, χρησιμοποιώντας ανάλυση Fourier σε τοπικό επίπεδο, δηλαδή σε επίπεδο υπολογιστικού κελίου, παράγονται θεωρητικά αποτελέσματα σύγκλισης της τεχνικής πολυπλέγματος για το προτεινόμενο αριθμητικό σχήμα. Η ανάλυση αυτή επικεντρώνεται τόσο στην επιλογή της μεθόδου χαλάρωσης, όσο και στα υπόλοιπα μέρη που συνθέτουν την τεχνική Πολυπλέγματος, με σκοπό τη βελτιστοποίηση των ρυθμών σύγκλισης του επιλυτή. Ο προτεινόμενος αριθμητικός επιλυτής μελετάται σε δύο προβλήματα δοκιμής, ώστε να επιβεβαιωθεί η τάξη ακρίβειας για κάθε επιλογή συνοριακών συνθηκών. Επίσης, υπολογίζονται οι ρυθμοί σύγκλισης της Πολυπλεγματικής μεθόδου για όλους τους γνωστούς τελεστές μεταφοράς, συμπεριλαμβανομένου και των νέων προτεινόμενων τελεστών. Οι θεωρητικοί ρυθμοί σύγκλισης βρίσκονται σε συμφωνία με τα αντίστοιχα αριθμητικά αποτελέσματα, παρέχοντας σχετική ασφάλεια για την επιλογή των βασικών συστατικών της τεχνικής Πολυπλέγματος. Επιπρόσθετα, οι παρατηρηθείσες ρυθμοί σύγκλισης είναι σημαντικά καλύτεροι σε σχέση με αυτούς των αριθμητικά συμπαγών σχημάτων δεύτερης τάξης ακρίβειας, ενώ ταυτόχρονα μπορούν να συγκριθούν με τους ρυθμούς σύγκλισης που μπορεί κάποιος να επιτύχει σε διακριτοποιήσεις, όταν οι άγνωστοι κατανέμονται στις κορυφές των υπολογιστικών κελιών (vertex-centered grids). Το νέο αυτό σχήμα της Πολυπλεγματικής μεθόδου εφαρμόζεται με επιτυχία σε cell-centered διακριτοποιήσεις για υψηλής ακρίβειας μεθόδους. Ταυτόχρονα με την προσπάθεια βελτιστοποίησης της τεχνικής Πολυπλέγματος του αριθμητικού σχήματος, μελετάται και η διαδικασία επίλυσης του παραγόμενου γραμμικού συστήματος με την συγκριτική εφαρμογή των πιο δημοφιλών επαναληπτικών μεθόδων. Τα συγκριτικά αποτελέσματα που παρουσιάζονται αναδεικνύουν την α-

ναγκαιότητα χρήσης της τεχνικής Πολυπλέγματος για προβλήματα του συγκεκριμένου είδους, αφού παρατηρήθηκε μία αξιόλογη μείωση στο χρόνο επίλυσης του γραμμικού συστήματος, σε σύγκριση με την αποδοτικότερη κλασική επαναληπτική μέθοδο επίλυσης.

Ο προτεινόμενος αριθμητικός επιλυτής των εξισώσεων Navier-Stokes μπορεί να αντιμετωπίσει ικανοποιητικά προβλήματα μη-σταθερών ασυμπίεστων ροών, τα οποία παρουσιάζουν διαφορετική κατανομή της φυσικής ποσότητας σε διαφορετικές διευθύνσεις (ανισοτροπίες). Για την διακριτοποίηση των χωρικών μεταβλητών χρησιμοποιούνται συμπαγή σχήματα πεπερασμένων διαφορών τέταρτης τάξης, ενώ για αυτή του χρόνου η κλασική αριθμητική μέθοδος Runge-Kutta τέταρτης τάξης. Η ασυμπιεστότητα του ρευστού εφαρμόζεται σε κάθε χρονικό βήμα του αριθμητικού χρονικού σχήματος μέσω της επίλυσης ενός ελλειπτικού τύπου προβλήματος με σκοπό τη διόρθωση της πίεσης και επακόλουθα του πεδίου ταχυτήτων. Η αποδοτικότητα και η ακρίβεια του αριθμητικού επιλυτή Navier-Stokes επαληθεύεται για προβλήματα δοκιμών σταθερών και μεταβαλλόμενων χρονικά ροών. Επιπρόσθετα, ο προτεινόμενος επιλυτής στα προβλήματα δοκιμής βελτίωσε σημαντικά τους χρόνους εκτέλεσης της επίλυσης των Navier-Stokes. Στις περισσότερες περιπτώσεις, η νέα μέθοδος με τη χρήση της τεχνικής Πολυπλέγματος επιτάχυνε τη διαδικασία επίλυσης πάνω από δέκα φορές.

Οι μετρήσεις απόδοσης του αλγορίθμου αναδεικνύουν ότι για προβλήματα μεγέθους διακριτοποίησης μέχρι 256×256 υπολογιστικών κελιών, η ενσωμάτωση της τεχνικής Πολυπλέγματος αντιμετωπίζει με επιτυχία τον αυξανόμενο χρόνο εκτέλεσης που απαιτείται. Ωστόσο, για μεγαλύτερες διακριτοποιήσεις του χωρίου, οι οποίες χρειάζονται μικρότερα χρονικά βήματα διακριτοποίησης, το υπολογιστικό κόστος αποτελεί έναν αποτρεπτικό παράγοντα για την επίλυση ρεαλιστικών προβλημάτων. Επίσης, υπολογίστηκε ότι η πιο χρονοβόρα διαδικασία του αλγορίθμου είναι αυτή της διόρθωσης της πίεσης. Αυτό υπήρξε το κίνητρο για τον επανασχεδιασμό του αλγορίθμου επίλυσης των Navier-Stokes και την ανάπτυξη ενός αποδοτικού παράλληλου αλγορίθμου για σύγχρονες υπολογιστικές αρχιτεκτονικές. Για την αύξηση της παραλληλοποίησης του αλγορίθμου χρησιμοποιήθηκαν τεχνικές όπως: 1) αναδιάταξη των κόμβων του πλέγματος σύμφωνα με το σχήμα ομαδοποίησης κόκκινου-μαύρου(horizontal red-black ordering) με σκοπό την ανεξαρτητοποίηση ομάδων αγνώστων και 2) η μέθοδος Block Cyclic Reduction για την επιτάχυνση της επίλυσης των ομαδοποιημένων γραμμικών υπο-συστημάτων. Οι τεχνικές αυτές εφαρμόστηκαν για να μην αυξηθούν τα βήματα εκτέλεσης του σειριακού αλγορίθμου. Η κατάλληλη οργάνωση των υπολογισμών έκανε εφικτή την εκτέλεση όλης της υπολογιστικής προσομοίωσης σε συσκευή επιτάχυνσης υπολογισμών, με αποτέλεσμα την μείωση στο ελάχιστο του κόστους επικοινωνίας μεταξύ της κύριας μνήμης του υπολογιστικού συστήματος και της μνήμης του επιταχυντή. Επιπλέον, η προτεινόμενη διαδικασία παραλληλοποίησης εκμεταλλεύεται την επαναλαμβανόμενη block δομή του πίνακα των συντελεστών, ελαχιστοποιώντας τις απαιτήσεις σε αποθηκευτικό χώρο. Η υλοποίηση του παράλληλου αλγορίθμου πραγματοποιήθηκε σε περιβάλλον ανάπτυξης του προτύπου OpenACC και δοκιμάστηκε σε τρεις υπολογιστικές αρχιτεκτονικές κοινής μνήμης με διαφορετικών τύπων επιταχυντές. Τα αποτελέσματα απόδοσης του προτεινόμενου αλγορίθμου είναι πολύ ενθαρρυντικά, αφού ο παράλληλος Navier-Stokes επιλυτής επιτυγχάνει μια επιτάχυνση μεγαλύτερη από 10 φορές σε σύγκριση με την σειριακή υλοποίηση του και 4 φορές σε σχέση με την

υλοποίηση σε κοινής μνήμης πολυεπεξεργαστικές αρχιτεκτονικές χωρίς όμως την χρήση επιταχυντών. Επίσης, η εφαρμογή του σε προβλήματα με μεγάλες ανισοτροπίες ανέδειξε την υπεροχή της Πολυπλεγματικής μεθόδου, που χρησιμοποιεί την τεχνική της ημι-αραίωσης του πλέγματος στην διεύθυνση που παρουσιάζεται η ανισοτροπία σε σχέση με την τεχνική της πλήρους-αραίωσης του και στις δύο διαστάσεις.

Μέρος των ερευνητικών αποτελεσμάτων της παρούσας διατριβής έχει δημοσιευθεί στα παρακάτω διεθνή περιοδικά και επιστημονικά συνέδρια:

Δημοσιεύσεις σε διεθνή περιοδικά

1) V. Mandikas, E. Mathioudakis, "*A parallel multigrid solver for incompressible flows on computing architectures with accelerators*", Journal of Supercomputing (2017), Springer, doi="10.1007/s11227-017-2066-y".

2) E. Mathioudakis, V. Mandikas, G. Kozyrakis, N. Kampanis, J. Ekaterinaris, "*Multigrid cell-centered techniques for high-order incompressible flow numerical solutions*", Aerospace Science and Technology - AESTE, vol. 64, pp. 85-101, Elsevier, 2017.

3) V. Mandikas, E. Mathioudakis, E. Papadopoulou, N. Kampanis, "*A High order accurate multigrid pressure correction algorithm for incompressible Navier-Stokes equations*", Lecture Notes in Engineering and Computer Science, vol. 2204, IAENG, pp. 74-79, 2013.

Δημοσιεύσεις σε πρακτικά Συνεδρίων

1) V. Mandikas, E. Mathioudakis, G. Kozyrakis, N. Kampanis and J. Ekaterinaris, "*A MultiGrid accelerated high-order pressure correction compact scheme for incompressible Navier-Stokes solvers*, NumAn 2014 – Conference in Numerical Analysis 2014 - Chania, Greece, http://lib.amcl.tuc.gr/handle/triton/69.

2) V. Mandikas, E. Mathioudakis, E. Papadopoulou, and N. Kampanis, " *A High Order Accurate MultiGrid Pressure Correction Algorithm for Incompressible Navier-Stokes Equations*, Proceedings of The World Congress on Engineering 2013 , pp74-79.

3) V. Mandikas, E. Mathioudakis, N. Kampanis and J. Ekaterinaris, "*High-order accurate numerical pressure correction based on Geometric MultiGrid schemes for the incompressible Navier-Stokes equations*, NumAn 2010 – Conference in Numerical Analysis 2010 - Chania, Greece, pp. 198-203.

Αν και η παρούσα ερευνητική διαδικασία επικεντρώνεται σε προβλήματα ορισμένα σε διδιάστατα καρτεσιανά πλέγματα, η επέκτασή της σε τρεις διαστάσεις ή σε καμπυλόγραμμες συντεταγμένες είναι άμεση. Αυτό συμβαίνει διότι είναι εύκολη η επέκταση όλων των διαδικασιών του παραγόμενου αλγορίθμου, καθώς και της επιτάχυνσης μέσω της τεχνικής πολυπλέγματος. Επίσης, η μέθοδος ενδείκνυται για μεταβατικές τυρβώδεις ασυμπίεστες ροές με την μέθοδο προσομοιώσεις τύρβης μεγάλων δινών (large eddy simulation LES), ενώ με τη χρήση έμμεσων αριθμητικών χρονικών μεθόδων και μοντέλων τύρβης μπορούν να δημιουργηθούν προσομοιώσεις τυρβωδών ροών υψηλής πιστότητας.

Abstract

The Navier-Stokes equations, that govern the motion of an incompressible or compressible fluid, were introduced more than a hundred years ago. Their nonlinearity leads to significant difficulties, sometimes insurmountable, when trying to find an analytical solution. In order to obtain an exact solution, specific geometries and initial/boundary conditions must be considered. In more complex situations, such as applications in fluid structure interaction, low speed aerodynamics, biomechanics etc., the application of numerical methods may provide reliable solutions for the Navier-Stokes equations. Furthermore, the necessity for a fast high-resolution numerical solver for the incompressible Navier-Stokes equations emerges from real-life simulations, such as flows over hydrofoils, wind-turbine blades, and aircraft wings during takeoff and landing. Thus, the objective of this thesis is to develop, explicate and demonstrate the performance of a highly efficient flow-solver, which exploits the computational power of architectures with computing accelerators.

In the first part of this thesis, and after the preliminaries, an elliptic PDE multigrid solver is developed and demonstrated. The solver is capable of handling highly anisotropic 2D Boundary Value Problems (BVPs) and is based on high-order cell-centered Finite Difference Compact schemes and Multigrid techniques. Compact schemes provide a representation of the shorter length scales, when applied to problems with a range of spatial scales, compared to traditional finite difference approximations. An improvement of the proposed method is the treatment of PDE boundary conditions. Boundary closure formulas for Dirichlet, Neumann, Robin or mixed-type boundary conditions applied to the physical boundary are derived and tested herein for several simulation problems. Moreover, novel multigrid components for cell-centered discretization are being constructed, which could also be generalized to three dimensional problems in a straightforward manner. In particular, the new intergrid operators involve less non-zero entries than common operators in cell-centered grids, preserving at the same time the accuracy of high-order operators. Next, some theoretical convergence results for the multigrid solver using Local Fourier Analysis (LFA) are given in order to improve its multigrid convergence factors. The analysis focuses

on both the relaxation method used within the multigrid, as well as in the remaining components of the multigrid method. The proposed multigrid elliptic solver is evaluated on two classical BVPs, so that the fourth-order accuracy of the solver, as well as the boundary treatments, could be validated. The multigrid convergence rates for every acknowledged transfer operator, along with every novel one, are evaluated as to determine the convergence behavior of the corresponding multigrid solver. These results are also compared to the theoretical convergence factors based on the LFA. The concordance of the analytical and numerical results is acceptable. A comparison between the calculated convergence rates to the corresponding values, obtained from the second-order compact scheme, indicates the superiority of the high-order compact scheme. In addition, these convergence rates are as good as the factors one obtains from the vertex-centered case. It is noted, that the cell-centered multigrid numerical analysis is presented for a high-order scheme, opposed to earlier studies. Along with the investigation on the improvement of multigrid techniques for the high-order scheme, iterative methods are also tested for the resulting sparse linear system and compared with the multigrid numerical solver. The comparison results indicate the necessity of a multigrid solver for this kind of problems, as it is proved to be hundreds of times faster than the optimal iterative method when fine discretizations are used.

The proposed spatial discretization solver is incorporated in an effective Navier-Stokes solver capable of handling highly anisotropic flow problems. The solver is based on the pressure-velocity coupling and uses fourth-order compact schemes for discretizing each spatial dimension, formulated on a staggered grid arrangement. The temporal discretization is carried out by a fourth-order Runge-Kutta (RK4) method. Incompressibility is enforced at each time step using a global pressure correction method solving a Poisson-type PDE. In this method, the multigrid solver is being used within each stage of the RK4 method to compute the pressure correction. The spatial and temporal fourth-order accuracy of this Navier-Stokes solver are validated for a set of steady and unsteady classical test problems. Further, the performance results indicate that multigrid accelerates the solution procedure more than 10 times, comparing with other solvers in the literature.

The numerical study of the sequential Navier-Stokes solver evince that, in cases of grid sizes up to 256×256, the incorporation of the multigrid scheme handles the increasing execution time moderately. However, in case of finer grid sizes, the computational cost becomes intolerable despite the high convergence rates of the Multigrid method. It is noted that the most time-consuming part of the solver is the pressure correction procedure. This time restriction gives motive to redesign and develop an efficient parallel multigrid based Navier-Stokes algorithm, to exploit the benefits of modern parallel computer architectures

with accelerators. In order to increase parallelism at each computing phase of the algorithm, the horizontal red-black coloring scheme for grid nodes is chosen. The Block Cyclic Reduction method is also applied for the solution of the arising linear sub-systems, without modifying the multigrid cycling nature in the algorithm. This enables the execution of the entire simulation in the acceleration device, minimizing the communication cost between memory units. In addition, the proposed parallelization exploits the block structure of the coefficient matrix, minimizing data storage and increasing again the parallelism. The solver is implemented and examined on three parallel machines with different type of accelerator devices. The realization is developed using the OpenACC and OpenMP APIs. The effect of several multigrid components on modern and legacy acceleration architectures is investigated

Application's performance investigation demonstrates that the proposed parallel multigrid solver accomplishes an acceleration of more than ten times over the sequential solver and more than four times over multi-core CPU-only realizations. In case of highly anisotropic problems, the parallel semi-coarsening multigrid solver is preferred to the full-coarsening one, as the division of labor by the accelerator device provides faster computational rates, in case of non-uniform discretizations. Several scientific parts of this thesis have been already published in

Journal publications

1) V. Mandikas, E. Mathioudakis, "*A parallel multigrid solver for incompressible flows on computing architectures with accelerators*", Journal of Supercomputing (2017), Springer, doi="10.1007/s11227-017-2066-y".

2) E. Mathioudakis, V. Mandikas, G. Kozyrakis, N. Kampanis, J. Ekaterinaris, "*Multigrid cell-centered techniques for high-order incompressible flow numerical solutions*", Aerospace Science and Technology - AESTE, vol. 64, pp. 85-101, Elsevier, 2017

3) V. Mandikas, E. Mathioudakis, E. Papadopoulou, N. Kampanis, "*A High order accurate multigrid pressure correction algorithm for incompressible Navier-Stokes equations*", Lecture Notes in Engineering and Computer Science, vol. 2204, IAENG, pp. 74-79, 2013.

Conference publications

1) V. Mandikas, E. Mathioudakis, G. Kozyrakis, N. Kampanis and J. Ekaterinaris, "*A MultiGrid accelerated high-order pressure correction compact scheme for incompressible Navier-Stokes solvers*, NumAn 2014 Conference in Numerical Analysis 2014 - Chania, Greece, http://lib.amcl.tuc.gr/handle/triton/69.

2) V. Mandikas, E. Mathioudakis, E. Papadopoulou, and N. Kampanis, " *A High Order Accurate MultiGrid Pressure Correction Algorithm for Incompressible Navier-Stokes Equations*, Proceedings of The World Congress on Engineering 2013 , pp74-79

3) V. Mandikas, E. Mathioudakis, N. Kampanis and J. Ekaterinaris, "*High-order accurate numerical pressure correction based on Geometric MultiGrid schemes for the incompressible Navier-Stokes equations*, NumAn 2010 - Conference in Numerical Analysis 2010 - Chania, Greece, pp. 198-203.

The proposed numerical algorithm can be easily extended for the case of three dimensional flow problems, on curvilinear coordinates, expanding the applicability of the current methodology. Furthermore, the proposed numerical methodology can be applied for solving comparable problems, e.g. Maxwells equations. The design of a parallel algorithm for the utilization of a Heterogeneous Multi-Accelerator architecture using the MPI, OpenMP and OpenACC APIs, is considered to be a promising improvement.

Acknowledgments

Originally, I would like to express my sincere acknowledgments and gratitude to my PhD advisor Assistant Professor Emmanuel Mathioudakis. If it wasn't for his unlimited help, inspiration, guidance and support, I would not have been able to successfully complete this scientific work.

I am very grateful to the advisory committee member Prof. Ioannis Saridakis, for his great help and support throughout my PhD journey.

I would also like to thank Dr. Nikolaos Kampanis, who assisted in the development of the incompressible Navier-Stokes solver. His wide knowledge on the Finite Difference Compact Schemes proved to be remarkably useful in several aspects of my research.

Furthermore, I would like to sincerely thank the members of my examination committee Prof. Elena Papadopoulou, Ass. Prof. Anargiros Delis, Prof. Antonis Vafidis and Prof. Georgios Gravvanis for the interest shown in my dissertation and the insightful comments they provided, which added greatly to the value of this work.

I would also like to thank all my colleagues in AMCL and IACM research groups for their help and warm encouragement.

I wish to gratefully acknowledge the Alexander S. Onassis Public Benefit Foundation for the financial support under Grand GZF030/2009-10.

Last but not least, I would like to thank my life's best companion, my lovely wife, Vasiliki Pantoula. The love, care and support she tirelessly offered me throughout my PhD were priceless. I am also grateful for my little angels, Giorgos and Elena for all the joy and happiness they bring to our daily life. Finally, I want to express my countless thanks and gratitude to my parents, George and Kleri.

Contents

List of Figures

List of Tables

Chapter 1

Preliminaries

The present chapter provides information about discretization approaches of second-order scalar 2D Partial Differential Equations (PDEs), introducing the reader to the mathematical concepts used in the following chapters. The case of the elliptic boundary value problem is accentuated, as this particular type of problem is being used in the development of multigrid techniques.

1.1 Boundary Value problems

Let us consider the linear differential equations $Lu = f$ in a domain $\Omega \subset \mathbb{R}^2$, where

$$Lu = a_{11}u_{xx} + a_{12}u_{xy} + a_{22}u_{xx} + a_1u_x + a_2u_y + a_0u \tag{1.1}$$

with the coefficients functions a_{ij}, a_i, a_0, for $i, j = 1, 2$ and the right-hand side function f that may depend on x, y. Defining the discriminant to be $a_{12}^2 - 4a_{11}a_{22}$, the properties and the behavior of the solution depend on the type, as classified below:

- If $a_{12}^2 - 4a_{11}a_{22} > 0$, the equation is called hyperbolic.
- If $a_{12}^2 - 4a_{11}a_{22} = 0$, the equation is called parabolic.

- If $a_{12}^2 - 4a_{11}a_{22} < 0$, the equation is called elliptic.

Scalar linear boundary value problems are defined as

$$\begin{aligned} L^{\Omega}u(\mathbf{x}) &= f^{\Omega}(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ L^{\Gamma}u(\mathbf{x}) &= f^{\Gamma}(\mathbf{x}), \quad \mathbf{x} \in \Gamma := \partial\Omega, \end{aligned} \tag{1.2}$$

where $\mathbf{x} = (x, y)$ and Ω is a given open bounded domain with a boundary Γ.

1.2 Discretization approaches

The differential equations to be solved in this thesis are assumed to be discretized on a Cartesian grid type. The Finite Difference Method (FDM) is often used in this type of grids, whereas the Finite Element Method (FEM) and the Finite Volume Method (FVM) are widely used in the context of unstructured grids. Herein, FDM is considered as the discretization method in favor of differential equations. The FDM method is based on approximation schemes for the partial derivative components of the BVP (1.2). These discretization schemes use a set of grid points of the discretized domain Ω.

The discretized equations are often defined at the same points as the unknowns within a grid. The arrangement of the unknowns may be applied in one of the following three ways: vertex-centered, cell-centered and by taking different points for the different types of unknowns (e.g. staggered grid). Examples of the vertex-centered fashion, the cell-centered and the staggered Cartesian grid are provided in Fig. 1-1. In this thesis, the focus is drawn in discretization schemes based on the last two grid types.

1.3 Discrete Boundary Value problems

The numerical solution of the BVP (1.2) on Cartesian grids using the FDM, seeks after an approximation $u_{\boldsymbol{h}}(x, y)$ that satisfies the discrete BVP (1.2),

$$\begin{aligned} L_{\boldsymbol{h}}^{\Omega} u_{\boldsymbol{h}}(x,y) &= f_{\boldsymbol{h}}^{\Omega}(x,y), \quad (x,y) \in \Omega_{\boldsymbol{h}} \\ L_{\boldsymbol{h}}^{\Gamma} u_{\boldsymbol{h}}(x,y) &= f_{\boldsymbol{h}}^{\Gamma}(x,y), \quad (x,y) \in \Gamma_{\boldsymbol{h}}. \end{aligned} \tag{1.3}$$

with a discretization step $\boldsymbol{h}$.

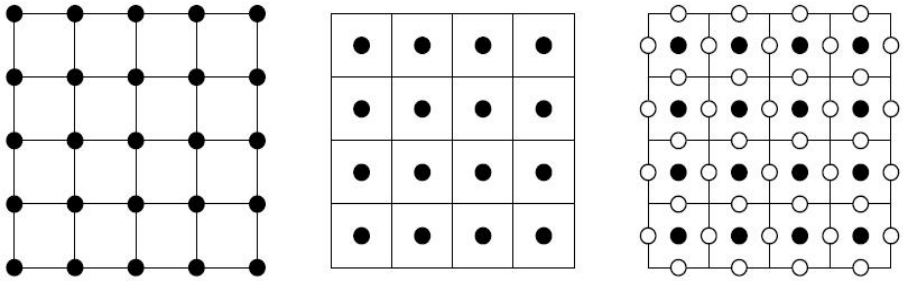

Figure 1-1: Discretization grid types: (a) a vertex-centered; (b) a cell-centered; (c) a staggered.

In particular, in a cell-centered uniform infinite grid type

$$\mathbf{G}_h := \{(x,y) : x = x_i = (i - \frac{1}{2})\Delta x,\ y = y_j = (j - \frac{1}{2})\Delta y;\ i,j \in \mathbb{Z}\}\ . \tag{1.4}$$

where $\boldsymbol{h} = (\Delta x, \Delta y)$ the vector of the fixed mesh sizes, $\Omega_h := \Omega \cap \mathbf{G}_h$ and $\Gamma_h := \partial\Omega \cap \mathbf{G}_h$. The BVP (1.3) may be consicely written

$$L_{\boldsymbol{h}} u_{\boldsymbol{h}} = f_{\boldsymbol{h}} \quad (\Omega_{\boldsymbol{h}})\ . \tag{1.5}$$

The terms $u_{\boldsymbol{h}}$ and $f_{\boldsymbol{h}}$ are grid functions of the continuous counterpart functions u and f on $\Omega_{\boldsymbol{h}}$. The linear space of all grid functions, that act on Ω_h is defined as $\mathcal{G}(\Omega_h)$. Thus L_h is a linear operator

$$L_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h). \tag{1.6}$$

It shall also be noted down that, Eq. (1.5) can be represented as a system of linear algebraic

equations or as a single grid equation.

The Euclidean inner product for functions u, v

$$\langle u, \mathrm{v}\rangle_2 := \frac{1}{\#\Omega_h} \sum_{\boldsymbol{x}\in\Omega_h} u_h(\boldsymbol{x}) v_h(\boldsymbol{x}) \tag{1.7}$$

where $\#\Omega_h$ is the number of the grid points of Ω_h, and the induced norm

$$\|u_h\|_2 = \sqrt{\langle u_h, u_h\rangle_2} \tag{1.8}$$

allows the comparison of grid functions in different grids. In case of the discrete operators L_h, the corresponding operator norm is the spectral norm given as

$$\|L_h\|_S = \sqrt{\rho(L_h L_h^*)} \quad , \tag{1.9}$$

where L_h^* is the adjoint of L_h and ρ is the spectral radius

$$\rho((L_h L_h^*) = \max\left(|\lambda|; \lambda \text{ is eigenvalue of } L_h L_h^*\right) \quad . \tag{1.10}$$

When in practice, the *infinity norm*

$$\|u_h\|_\infty := \max\{|u_h(\boldsymbol{x})| : \boldsymbol{x} \in \Omega_h\} \tag{1.11}$$

is frequently being used.

1.4 Stencil Notation

In case of a Cartesian grid, the use of the stencil notation is considered to be a quite convenient technique to define discrete operators, e.g L_h. The operator L_h can be defined on

$\mathcal{G}(\Omega_h)$ as

$$L_h u_h(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa}\in J} \ell_{\boldsymbol{\kappa}} u_h(\boldsymbol{x} + \boldsymbol{\kappa h}), \quad \boldsymbol{x} \in \Omega_h \tag{1.12}$$

with coefficients $\ell_{\boldsymbol{\kappa}} \in \mathbb{R}$ and a certain finite subset $J \subset \mathbb{Z}^2$. When gathering coefficients $\ell_{\boldsymbol{\kappa}}$ in a *stencil*, L_h can be written as

$$L_{\Delta x} \hat{=} [\ell_{\kappa_1}]_{\Delta x} = \begin{bmatrix} \dots & \ell_{-1} & \ell_0 & \ell_1 & \dots \end{bmatrix}_{\Delta x}, \qquad L_{\Delta y} \hat{=} [\ell_{\kappa_2}]_{\Delta y} = \begin{bmatrix} \vdots \\ \ell_1 \\ \ell_0 \\ \ell_{-1} \\ \vdots \end{bmatrix}_{\Delta y} \tag{1.13}$$

$$L_{\boldsymbol{h}} \hat{=} [\ell_{\boldsymbol{\kappa}}]_{\boldsymbol{h}} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \dots & \ell_{(-1,1)} & \ell_{(0,1)} & \ell_{(1,1)} & \dots \\ \dots & \ell_{(-1,0)} & \ell_{(0,0)} & \ell_{(1,0)} & \dots \\ \dots & \ell_{(-1,0)} & \ell_{(0,0)} & \ell_{(1,0)} & \dots \\ & \vdots & \vdots & \vdots & \end{bmatrix}_{\boldsymbol{h}}, \tag{1.14}$$

in one and two dimensions, respectively. Some of the stencils that will be presented herein using the FDM for the BVP (1.3) will be compact nine-point stencils

$$L_{\boldsymbol{h}} \hat{=} [\ell_{\boldsymbol{\kappa}}]_{\boldsymbol{h}} = \begin{bmatrix} \ell_{(-1,1)} & \ell_{(0,1)} & \ell_{(1,1)} \\ \ell_{(-1,0)} & \ell_{(0,0)} & \ell_{(1,0)} \\ \ell_{(-1,0)} & \ell_{(0,0)} & \ell_{(1,0)} \end{bmatrix}_{\boldsymbol{h}}. \tag{1.15}$$

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

15	7	16	8
5	13	6	14
11	3	12	4
1	9	2	10

13	14	15	16
5	6	7	8
9	10	11	12
1	2	3	4

Figure 1-2: Ordering strategies for grid points (a) lexicographic; (b) point red-black ; (c) zebra red-black

The points that are close to the boundary need their stencil notation to be modified. In cell-centered discretizations, case which in this thesis is the basis, the modification of L_h in grid equations near the boundary is not straightforward, see in Fig. 1-1 subfigure (b).

Furthermore, this concrete definition of discrete operators is also convenient in the context of the Local Fourier Analysis, which is essential in the analysis of the Multigrid Method (MG) in Chapter 3.

1.5 Matrix notation

Discrete operators L_h can be represented in a matrix form, i.e. each matrix row depicts the correlation of an unknown to its neighboring unknowns. Thus, the discretization approach produces a linear system of the following form

$$A\boldsymbol{u} = \boldsymbol{b}. \tag{1.16}$$

where $\boldsymbol{u}$ and $\boldsymbol{b}$ are vectors in the finite dimensional vector space $\mathbb{R}^{\#\Omega_h}$ and matrix $A \in \mathbb{R}^{\#\Omega_h \times \#\Omega_h}$. The structure of the matrix A depends on the numbering of the unknowns (or grid nodes). Standard ordering strategies for 2D applications are column- or row-wise numbering of grid points (*lexicographic ordering*) and red-black ordering, either using a checkerboard pattern (point-to-point) or in a zebra manner (line-to-line). Namely, in the case of the red-black ordering, the rule that is followed: *firstly, one should number the unknowns corresponding to the odd (red) grid points, and then the unknowns that correspond*

to the even (black) grid points. These three ordering types are illustrated in Fig. 1-2. The linear system (1.16) in the case of the red-black ordering is provided. The corresponding block matrix A contains the blocks D_{RR} and D_{BB}, representing the red (black) grid point correlation to the red (black) grid points. To this effect, blocks H_{RB} and H_{BR} represent the red (black) grid point correlation to the black (red) grid points.

$$\begin{bmatrix} D_{RR} & H_{RB} \\ H_{BR} & D_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{u}_R \\ \mathbf{u}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_R \\ \mathbf{b}_B \end{bmatrix} \quad . \tag{1.17}$$

1.6 Iterative methods

In a problem where the discretization is fine, the resulted linear system (1.16) is usually sparse and large. Thus, the use of an iterative method is considered vital in order to solve efficiently. The iterative process for the numerical solution of (1.16) can be described with the following iterative scheme

$$\boldsymbol{u}^{(i+1)} = \boldsymbol{u}^{(i)} + T^{(i)}(\boldsymbol{b} - A\boldsymbol{u}^{(i)}) \quad \text{for} \quad i = 0, 1, 2, \ldots \quad , \tag{1.18}$$

where $T^{(i)}$ is a sequence of matrices and $\boldsymbol{u}^{(i)}$ is a sequence of the solution approximations, whereas $\boldsymbol{u}^{(0)}$ is an arbitrary initial approximation. Different selections of the sequence of matrices $T^{(i)}$ lead to different iterative methods. There are two basic classes of iterative methods. The classical iterative methods are stationary processes (stationary methods), in which the matrices $T^{(i)}$ do not depend on the iteration count (i). If the sequence $T^{(i)}$ involves data that change in every iteration, the process is called non stationary (non-stationary methods).

1.6.1 Stationary methods

Stationary iterative methods are grounded in a coefficient matrix splitting, specifically $A = M - N$, where M is assumed to be non singular. Then, a linear fixed-point iteration is defined as

$$M\boldsymbol{u}^{(i+1)} = N\boldsymbol{u}^{(i)} + \boldsymbol{b} \quad \text{for } i = 0, 1, 2, \ldots \tag{1.19}$$

or

$$\boldsymbol{u}^{(i+1)} = M^{-1}N\boldsymbol{u}^{(i)} + M^{-1}\boldsymbol{b} \quad \text{for } i = 0, 1, 2, \ldots \tag{1.20}$$

The convergence of the iterative method depends on the properties of the matrix $G = M^{-1}N$, the so-called iteration matrix. The asymptotic convergence rate of the iterative method, depends on the spectral radius of its corresponding iteration matrix G.
Asymptotically (i.e for $i \to \infty$) we have

$$\|u - u^{(i+1)}\| \leqslant \rho(G)\|u - u^{(i)}\|. \tag{1.21}$$

Thus, an iterative method is called convergent if and only if $\rho(G) < 1$. It should ben noted that a smaller value of spectral radius leads to a faster convergence rate.

At this moment, five classical stationary iterative methods will be mentioned: Jacobi, Gauss-Seidel, weighted Jacobi, Successive Overrelaxation (SOR) and Symmetric Successive Overrelaxation (SSOR) methods. Considered the matrix splitting type

$$A = D - L - U \quad , \tag{1.22}$$

where matrix D is the diagonal part of A and $-L$ and $-U$ are the strictly lower and upper triangular parts of A, respectively. Classical iterative methods are defined as

- *Jacobi (JAC)* for $M = D_A$ and $N = L_A + U_A$

- *Gauss-Seidel* (GS) for $M = D_A - L_A$ and $N = U_A$
- *weighted Jacobi* $(\omega - JAC)$ for $M = D_A - \omega L_A$ and $N = \omega U_A + (1-\omega)D_A$
- *Successive Over Relaxation* (SOR) for $M = D_A - \omega L_A$ and $N = \omega U_A + (1-\omega)D_A$
- *Symmetric SOR* $(SSOR)$ for $G = (D-\omega L)^{-1}(\omega U+(1-\omega)D)(D-\omega U)^{-1}(\omega L+(1-\omega)D)$

The corresponding iteration matrices are G^{JAC}, G^{GS}, $G^{JAC}(\omega)$, $G^{GS}(\omega)$ (or G^{SOR}) and $G^{SGS}(\omega)$. It is clear that the undamped methods (a),(b) are recovered from (c),(d), respectively, in case of $\omega = 1$. Note that SOR is usually called the weighted Gauss-Seidel method.

A property of SOR attributed to Kahan [96] is given below:

Theorem 1.6.1 *A necessary convergence condition of the SOR method is that* $|\omega - 1| < 1$ *(for* $\omega \in \mathbb{R}$ *this condition becomes* $\omega \in (0, 2)$*).*

Furthermore, a few general results on the SOR and related methods are presented below, in case where A is a 2-cyclic consistently ordered matrix [79].

Definition 1.6.1 *A matrix* $A \in \mathbb{C}^{n,n}$ *is* $2-$*cyclic or possesses Young's "property A" if there exists a permutation matrix* P *such that*

$$PAP^T = \begin{bmatrix} D_1 & B \\ C & D_2 \end{bmatrix} , \tag{1.23}$$

where D_1, D_2 *are nonsingular diagonal matrices not necessarily of the same order.*

Definition 1.6.2 *Let* $A \in \mathbb{C}^{n,n}$ *be a* $2-$*cyclic matrix. Matrix* A *is considered to be consistently ordered based on the partition (1.22) if* $\sigma\left(D^{-1}\left(\alpha L + \frac{1}{\alpha}U\right)\right)$ *is independent of* $\alpha \in \mathbb{C} \setminus \{0\}$.

Remark 1.6.1 *Among others, matrices that do possess the "two-cyclic consistently ordered property" are tridiagonal matrices with nonzero diagonal elements, and, matrices that already have the form (1.23).*

For the 2−cyclic consistently ordered matrices A, Young [102] proved that the eigenvalues μ and λ of the Jacobi and SOR iteration matrices respectively, associate with A as:

Theorem 1.6.2 *Let $A \in \mathbb{C}^{n,n}$ be a 2−cyclic consistently ordered matrix, $\mu \in \sigma\left(D^{-1}(L+U)\right)$ and $\lambda \neq 0$ satisfied the following functional relationship*

$$(\lambda + \omega - 1)^2 = \omega^2 \mu^2 \lambda. \tag{1.24}$$

if and only if $\lambda \in \sigma\left(G^{SOR}\right) \setminus \{0\}$.

It may be handily noticed that $\rho(G^{GS}) = \rho^2(G^{JAC})$, hence, it is expected that the Gauss-Seidel method will be roughly two times faster compared to the Jacobi method, when the latter converges.

1.6.2 Non-stationary methods

Non-stationary methods, unlike stationary methods, do not have a constant iteration matrix. The most popular non-stationary iterative methods are listed below.

- Conjugate Gradient method(CG method)
- General minimal Residual method(GMRES method)
- Minimal Residual (MINRES method)
- Symmetric LQ method(SYMMLQ method)
- Biconjugate Gradient method (BiCG method)
- Biconjugate Gradient Stabilized (Bi-CGSTAB method)
- Conjugate Gradient Squared (CGS method)
- Quasi-Minimal Residual (QMR method)

- Conjugate Gradients on the Normal Equations (CGNE and CGNR methods)
- Chebyshev Iteration

This class of iterative methods construct the basis of a krylov subspace [101] where the linear solution can be found.

1.6.3 Preconditioners

The *condition* of a system (1.16) is described by its condition number

$$\kappa(A) = \|A\| \ \ \|A^{-1}\| \tag{1.25}$$

in some appropriate norm. The convergence rate of certain iterative methods, e.g. Krylov subspace methods, depends on the condition number of the coefficient matrix. Hence, the concept of the transformation of the linear system into an equivalent - in the sense that it accepts the same solution - but plain and with an improved condition number compared to the original one, is considered meaningful. This procedure is defined as *preconditioning*. There are a number of ways to apply this transformation. For instance, let M be a matrix that *approximates* of A, the equivalent linear system has the form

$$M^{-1}\, A\boldsymbol{u} \ = \ M^{-1}\, \boldsymbol{b} \tag{1.26}$$

where M is called a (left) preconditioner of A. Eventually, electing the iteration matrix of the aforementioned iterative methods as the preconditioner matrix for Krylov subspace methods results to the Jacobi, Gauss-Seidel, SOR and SSOR preconditioned methods respectively.

Block iterative methods (block relaxation schemes) are generalizations of the above *point* iterative methods. They update a set of unknowns corresponding to a block, e.g.

those associated to a line or a column in the discretization plane. There are certain cases, i.e. problems where anisotropies emerge (caused by strongly varying coefficients of the PDE) in which block relaxation schemes are more effective [50, 49].

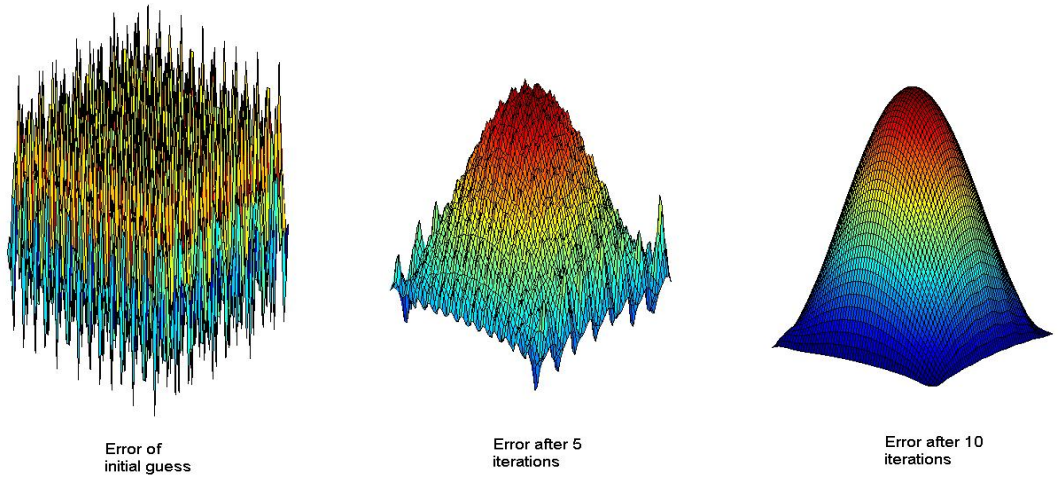

Figure 1-3: Influence of Horizontal Zebra Gauss-Seidel iteration based on the proposed discretization on the error (Model problem).

1.7 Elements of Multigrid

The above-mentioned classical point or block iterative methods (e.g. Jacobi, Gauss-Seidel) upgrade the initial approximation rapidly during the initial iterations, but convergence slows down and the entire scheme appears to stall as the iteration count increases (see Chapter 2 in [56]). In particular, although the global error decreases slowly with respect to the iteration count, the error smooths down promptly (see Figure 1-3). The fact can be explained by expressing the error in a Fourier series expansion, segregating the low-frequency (smooth) and the high-frequency (oscillatory) Fourier modes of error and exploring the way iteration behaves on these error components. Consequently, it may be noticed that the iteration eliminates the oscillatory modes of error while leaving aside the smooth modes. This so-called smoothing property is a critical limitation of the convectional relaxation methods.

However, it can be omited and multigrid is a way to remedy. The smoothing property of an iterative scheme is adequately analyzed using Local Fourier analysis (LFA), also called local mode analysis, introduced by Brandt [1]. It is an effective tool for the analysis of the Multigrid process, even though it is based on certain idealized assumptions and simplifications: Boundary conditions are neglected and the problem is considered on infinite grids G_h. Details and a thorough description of this kind of "smoothing analysis" are given in Chapter 3.

1.7.1 From Defect correction to Coarse-grid correction

For any approximation $u_h^{(i)}$ (after i iterations) of the solution u_h in the discrete problem

$$L_h u_h = f_h, \tag{1.27}$$

we denote the *error* by $e_h^{(i)} := u_h - u_h^{(i)}$ and the $defect$ (or *residual*) by $r_h^{(i)} := f_h - L_h u_h^{(i)}$. Using these definitions, it can be verified that the $defect\ equation$

$$L_h e_h^{(i)} = r_h^{(i)}, \tag{1.28}$$

is equivalent to the original equation since $u_h = u_h^{(i)} + e_h^{(i)}$. In order to introduce the two-grid idea (that leads to the multigrid method), the beginning is set from a general iteration for solving the defect equation. This process can be presented by the following procedural formulation:

$$\boxed{u_h^{(i)} \rightarrow r^{(i)} = f_h - L_h u_h^{(i)} \rightarrow L_h e_h^{(i)} = r_h^{(i)} \rightarrow u_h = u_h^{(i)} + e_h^{(i)}}$$

However, this procedure is not a meaningful numerical process. Consider now that L_h is approximated by the operator $\widehat{L}_h$, such as $\widehat{L}_h^{-1}$ exists. In that case, an approximation solution of the defect equation (1.28) can be obtained solving a "similar problem"

$$\widehat{L}_h e_h^{(i)} = r_h^{(i)}, \tag{1.29}$$

Using the above solution, a new approximation of u_h may be estimated by

$$\begin{aligned} u_h^{(i+1)} &= u_h^{(i)} + \hat{e}_h^{(i)} = u_h^{(i)} + \widehat{L}_h^{-1} r_h^{(i)} = u_h^{(i)} + \widehat{L}_h^{-1}\left(f_h - L_h u_h^{(i)}\right) \\ &= \left(I_h - \widehat{L}_h^{-1} L_h\right) u_h^{(i)} + \widehat{L}_h^{-1} f_h \end{aligned} \tag{1.30}$$

and, then, the procedural formulation turns up as

$$\boxed{u_h^{(i)} \rightarrow r^{(i)} = f_h - L_h u_h^{(i)} \rightarrow \widehat{L}_h \hat{e}_h^{(i)} = r_h^{(i)} \rightarrow u_h^{(i+1)} = u_h^{(i)} + \hat{e}_h^{(i)}}$$

Commencing with some initial approximation $u_h^{(0)}$, the successive application of this process leads to an iterative method. It can be seen from Eq. (1.30), that its iteration matrix is given by $M_h = I_h - \widehat{L}_h^{-1} L_h$, where I_h denotes the identity operator on $\mathcal{G}(\Omega_h)$. Since the solution of the defect equation has been used to "correct" the approximation, this procedure is called defect correction.

It is crucial to find a problem "similar" to the defect equation which comes cheaper in terms of operation counts, in order to benefit from the above rewriting of the original equation (1.28). An idea is the use of a coarser grid Ω_H ($H < h$) and an appropriate approximation L_H of L_h. Thus, the defect equation is replaced by

$$L_H \hat{e}_H^{(i)} = r_H^{(i)}, \qquad L_H : \mathcal{G}(\Omega_H) \rightarrow \mathcal{G}(\Omega_H) \tag{1.31}$$

which can be solved faster for fewer grid points in Ω_H.

Next, $\hat{e}_H^{(i)}$ and $r_H^{(i)}$ are grid functions on the Ω_H, allowing us to define two intergrid transfer operators

$$R_h^H : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_H), \qquad P_H^h : \mathcal{G}(\Omega_H) \to \mathcal{G}(\Omega_h). \tag{1.32}$$

The restriction operator R_h^H is applied to "restrict" $r_h^{(i)}$ to Ω_H:

$$\boxed{r_H^{(i)} := R_h^H r_h^{(i)}} \tag{1.33}$$

Then, the correction $e_H^{(i)}$ is transferred back to the fine grid Ω_h using an interpolation (prolongation) operator P_h^H:

$$\boxed{\hat{e}_h^{(i)} := P_H^h \hat{e}_H^{(i)}}. \tag{1.34}$$

This interpolated solution of the coarse-grid defect equation is an approximation of the fine-grid defect equation. It is being used to ammend the fine-grid approximation.

Altogether, one-coarse grid correction step (calculating $u_h^{(i+1)}$ from $u_h^{(i)}$) takes place as follows:

Coarse grid correction $u_h^{(i)} \to u_h^{(i+1)}$	
– Compute the defect (or the residual)	$r_h^{(i)} = f_h - L_h u_h^{(i)}$
– Restrict the defect (fine-to-coarse transfer)	$r_H^{(i)} = R_h^H r_h^{(i)}$
– Solve on Ω_H	$L_H \widehat{e}_H^{(i)} = r_H^{(i)}$
– Interpolate the correction (coarse-to-fine transfer)	$\hat{e}_h^{(i)} = P_H^h \widehat{e}_H^{(i)}$
– Compute a new approximation	$u_h^{(i+1)} = u_h^{(i)} + \hat{e}_h^{(i)}$

However, the coarse grid correction process does not converge when applied independently, as the spectral radius of the associated iteration operator $K_h = I_h - P_H^h L_H^{-1} R_h^H L_h$ is ≥ 1. This remark emerges as R_h^H maps into the lower dimensional space $\mathcal{G}(\Omega_H)$ and subsequently, $P_H^h L_H^{-1} R_h^H$ is not invertible, i.e. $P_H^h L_H^{-1} R_h^H L_h v_h = 0$ for certain $v_h \neq 0$. Thus, this approach has sense only when the coarse-grid quantities $e_H^{(i)}$ and $r_H^{(i)}$ are "reasonable" approximations of the corresponding fine-grid quantities. As stated earlier about the error, after a number of relaxation steps the high-frequency components smooth out and low-frequency components dominate. Such smooth quantities can be approximated quite well in coarser meshes.

1.7.2 Two-grid cycle

The development of multigrid methods for the numerical solution of an elliptic BVP (1.3) is motivated by two basic principles:

1. **Smoothing principle** Many classical iterative methods (Gauss-Seidel etc.) if appropriately applied to discrete elliptic problems have a strong smoothing effect on the error of any approximation.

The other basic principle is based on the following observation: a quantity that is smooth on a certain grid can, without any essential loss of information, be approximated on a coarser grid as well, for instance a grid with double the mesh size (see Figure 1-4 (a)) while an oscillatory one is not "visible" on the coarser grid (see Figure 1-4 (b)).

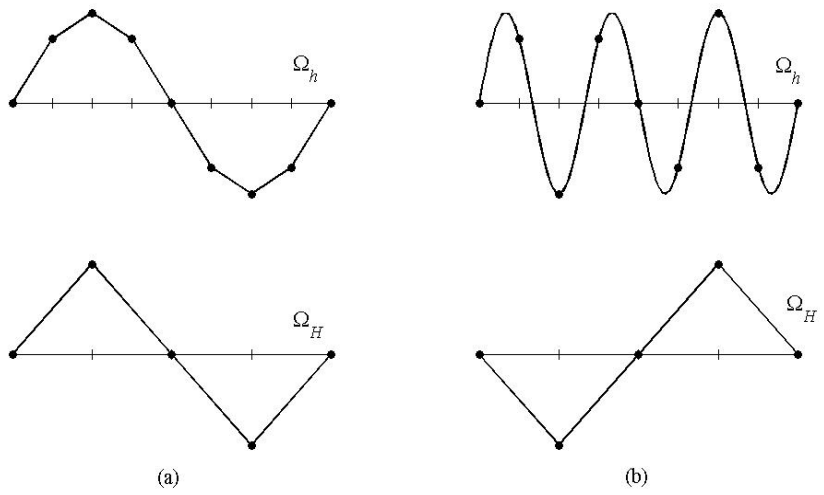

Figure 1-4: Smooth (a) and oscillatory (b) grid functions on Ω_h projected onto a coarser grid Ω_H ($H = 2h$).

2. **Coarse grid principle** A smooth error term can be well represented on a coarser grid Ω_H, where its approximation procedure is substantially less expensive.

Switching back to the Fourier error components, it is rather evident that, only the low frequencies are visible on Ω_H, since all the high frequencies coincide with a low frequency on Ω_H.

The above considerations imply that it is necessary to combine the two processes of the smoothing and coarse grid correction. In other words: if we are sure that the error of our approximation has become smooth (high-frequency components have been smoothed out) after some smoothing (relaxation) steps, we may approximate this error by a suitable procedure on a coarser grid. The baseline here is that smooth Fourier error components on a fine grid appear to be more oscillatory on a coarse grid (see Figure 1-4 (a)) and, thus, relaxation will be more effective.

To this end, one step of such an iterative two-grid method (calculating $u_h^{(i+1)}$ from $u_h^{(i)}$)

proceeds as follows:

Two-grid cycle $u_h^{(i)} \rightarrow u_h^{(i+1)}$

(1) Presmoothing

– Compute an approximation of u_h by performing v_1 steps of a given smoothing procedure ($S_h^{v_1}$) to $u_h^{(i)}$

(2) Coarse Grid Correction (CGC)

(3) Postsmoothing

– Compute $u_h^{(i+1)}$ by performing v_2 steps of a given smoothing procedure ($S_h^{v_2}$) to $u_h^{(i,\text{after CGC})}$

with the term smoothing procedure, an iterative method that possesses the smoothing property is signified. That is why it is called a smoother. Out of the above correction scheme, the iteration operator N_h^H of the two-grid cycle can be estimated, given by the formula

$$N_h^H = S_h^{v_1} K_h^H S_h^{v_2} \text{ with } K_h = I_h - P_H^h L_H^{-1} R_h^H L_h \quad , \tag{1.35}$$

where S_h is the iteration operator of the smoother.

1.7.3 Multigrid cycle

The *multigrid* idea is based on the concept that is not necessary the coarse-grid defect equation to be solved precisely; an approximate solution produced by a few two-grid cycles would do. Certainly, that can be put into practise recursively down to some coarsest grid, leading to a multigrid method which is being carried out as follows:

Multigrid cycle $u_k^{(i+1)} = MGCYC(k, r, u_k^{(i)}, L_k, f_k, \nu_1, \nu_2)$

(1) Presmoothing

– Compute an approximation of $u_k^{(i)}$ by performing ν_1 steps of smoothing procedure ($S_k^{\nu_1}$) to $u_k^{(i)}$

(2) Coarse grid correction (CGC)

– Compute the residual $r_k^{(i)} = f_k - L_k u_k^{(i)}$

– Restrict the residual $f_{k-1}^{(i)} = R_k^{k-1} r_k^{(i)}$

– Compute an approximation of $u_{k-1}^{(i)}$ of the residual equation on Ω_{k-1}

$$L_{k-1} u_{k-1}^{(i)} = r_{k-1}^{(i)} \quad (*)$$

as

if $k = 1$ (coarsest grid), use a direct or fast iterative solver for Eq. $(*)$

if $k > 1$, solve Eq. $(*)$ approximately by performing r $(\geq)$ $k-$grid cycles using the zero grid function as a first approximation

$$u_k^{(i)} = MGCYC(k-1, r, 0, L_{k-1}, r_{k-1}, \nu_1, \nu_2)$$

– Interpolate the correction $e_k^{(i)} = P_k^{k-1} u_{k-1}^{(i)}$

– Compute the corrected approximation on Ω_k $u_k^{(i,\text{after CGC})} = u_k^{(i)} + e_k^{(i)}$

(3) Postsmoothing

- Compute $u_k^{(i+1)}$ by performing ν_2 steps of smoothing procedure ($S_k^{\nu_2}$) to $u_k^{(i,\text{after CGC})}$

A multigrid cycle calculates a new approximation $u_k^{(i+1)}$ out of a given approximation $u_k^{(i)}$ to produce the solution u_k. The subscript k indicates the grid level with $k = 0$, corresponding to the coarsest grid.

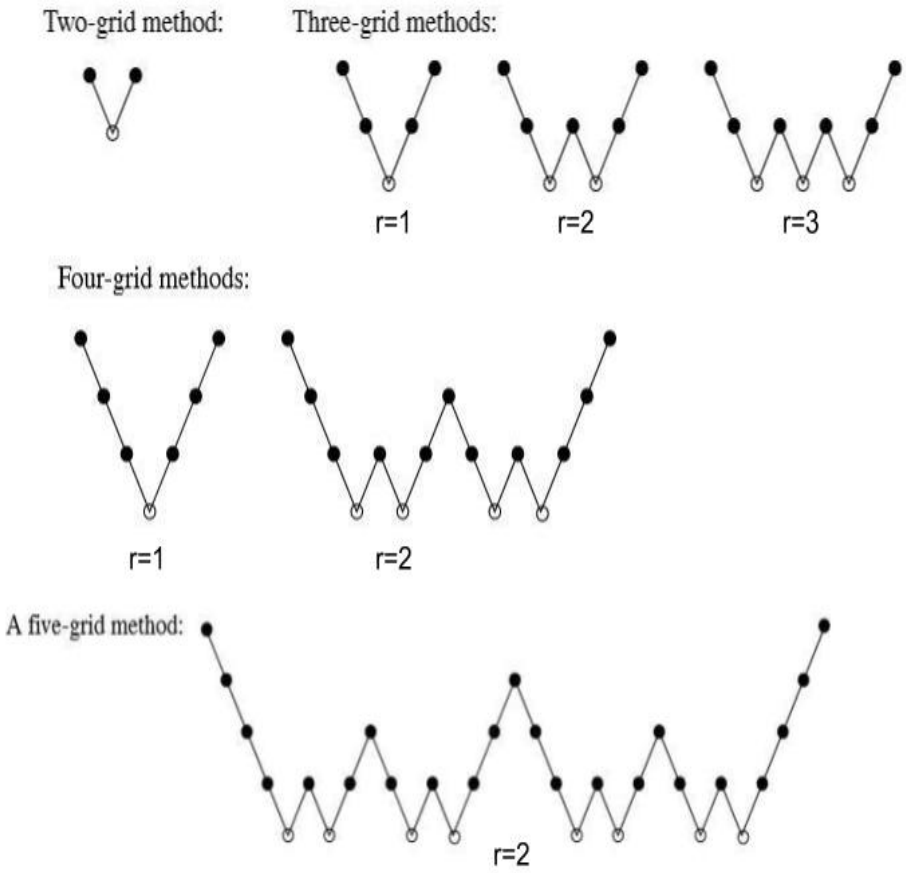

Figure 1-5: Structure of one multigrid cycle for different numbers of grids and different values of the cycle index r (•, smoothing; ∘, exact solution; fine-to-coarse; /, coarse-to-fine transfer).

The parameter r specifies the number of cycles to be carried out on the current coarse grid level. For $r = 1$ and $r = 2$, the multigrid-cycle refers to the V-cycle and W-cycle algorithm, respectively, see Fig. 1-5. A combination of $r = 1$ and $r = 2$ used in practice is called F-cycle [5] and is illustrated in Fig. 1-6. Furthermore, the algorithm that joins nested iteration with a cycling strategy is called the Full multigrid (FMG) technique [1]. The general idea of a nested iteration is to provide an initial approximation on a grid by the computation and interpolation of approximations on coarser grids. The structure of FMG with nested $V - cycle$ is illustrated in Fig. 1-7.

In order to define a unique multigrid method adequately, one should specify its components: the number of the grids involved, the smoother, the number of the smoothing steps v_1

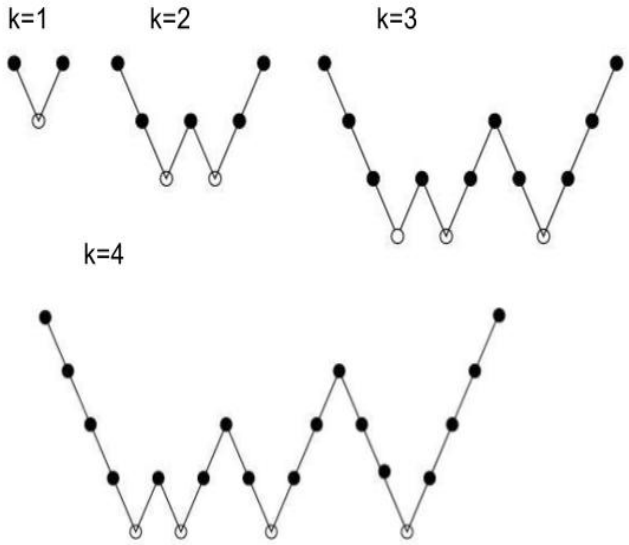

Figure 1-6: Structure of F-cycle.

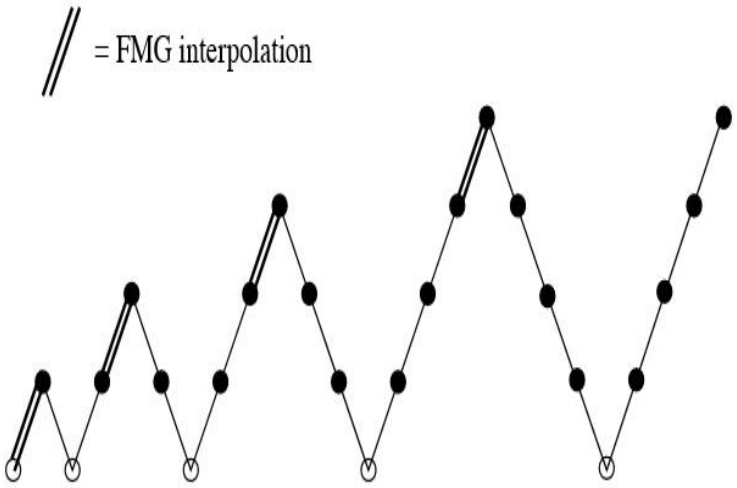

Figure 1-7: Structure of FMG with r=1 when using k=4(i.e. five grid levels).

and v_2, the coarsening strategy which determines the coarser grids, the transfer operators, the discretization operators in each grid. Since deciding on these components may have a strong influence on the efficiency of the resulting algorithm, a local Fourier analysis is

used to analyze the different behaviour as these multigrid components differ, targeting an optimal component set. Therefore, a programming code based on LFA has been developed to provide the theoretical framework necessary for the successful use of multigrid methods in our model problem (see Chapter 3).

Chapter 2

Cell-centered compact Finite Difference Multigrid techniques

Fourth-order compact vertex-centered finite difference schemes for 2D and 3D Boundary Value Problems (BVPs) have been extensively studied in [60, 97, 98]. However, the literature related to high-order finite difference schemes compatible on cell-centered grids is still considered insufficient. In this chapter, a high-order cell-centered finite difference compact scheme is presented for elliptic BVP in case of equal and unequal meshsize discretizations. The numerical scheme is coupled with multigrid techniques.

2.1 The Model Problem

The modified Helmholtz equation is used as the model problem over a rectangular bounded domain Ω, with proper boundary conditions defined on $\partial\Omega$. This BVP is often used as a benchmark problem for the comparison of various numerical elliptic solvers. It can be expressed as

$$u_{xx}(x,y) + u_{yy}(x,y) - \lambda u(x,y) \quad = \quad f(x,y) \quad , \quad (x,y) \in \Omega, \lambda \geq 0 \tag{2.1}$$

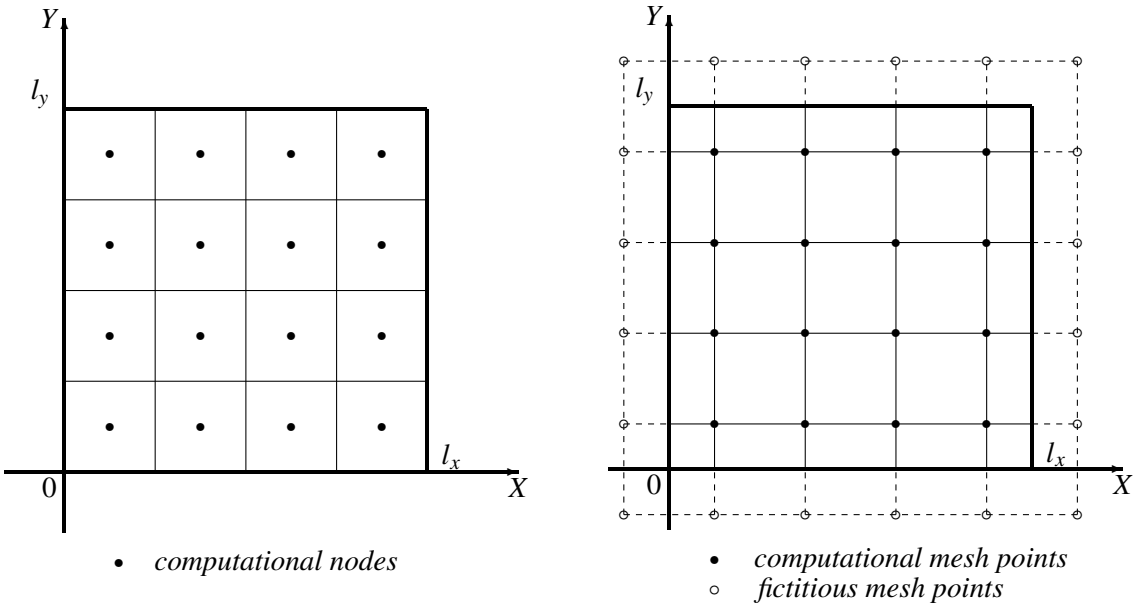

Figure 2-1: Dircretization for $N_x = N_y = 4$ subintervals (left) and the corresponding computational domain (right).

The solution function $u(x, y)$ and the right hand side function $f(x, y)$ are assumed to be sufficient smooth and have the required continuous partial derivatives (for example $u \in C_4(\Omega)$.

2.1.1 Domain discretization

This study employs the compact finite difference method as the discretization scheme for the elliptic linear operator. Compared to the standard finite difference approximations, compact schemes have improved resolution in wavespace [24, 51], i.e. compact schemes provide better representation of the shorter length scales, when applied to problems with a range of spatial scales, e.g. turbulent fluid flows.

A difference scheme is called compact, when it's restricted to the patch of cells immediately surrounding the given node and does not extend further. Alternately, high-order accuracy can be obtained by explicit finite difference formulas. These formulas lead to wider stencils, hence the resulting linear system has larger bandwidth. Additionally, the non-compact form of the difference scheme is less convenient, especially near the bound-

aries.

Taking a rectangular domain $\Omega \equiv [0, l_x]\times[0, l_y]$, the mesh sizes in a non-equal mesh discretization in each spatial direction will be Δx and Δy. Naming $N_x = l_x/\Delta x$ and $N_y = l_y/\Delta y$ the number of the uniform intervals along the x- and y- coordinate directions respectively, the mesh nodes are written as $(\bar{x}_i, \bar{y}_i)$ where $\bar{x}_i = i\Delta x$ and $\bar{y}_j = j\Delta y$, $0 \leq i \leq N_x$, $0 \leq j \leq N_y$. The cell-centered compact finite difference method seeks an approximation of the BVP solution in the center node (x_i, y_i) of every computational cell with $x_i = (i - \frac{1}{2})\Delta x$ and $y_j = (j - \frac{1}{2})\Delta y$ for $i = 0, 1, \ldots, N_x + 1$ and $j = 0, 1, \ldots, N_y + 1$. There are $N = N_x N_y$ centered mesh points (x_i, y_i) within Ω, which can be considered as grid points of a new computational grid. The mesh points denoted by the indices $i = 0, N_x + 1$ and $j = 0, N_y + 1$ are fictitious, lying outside the computational domain Ω.

Figure 2-1 presents an example of the primary mesh discretization (left) and the corresponding new cell-centered computational mesh discretization (right).

2.1.2 Low accurate approximations

The second derivatives in Eq. (2.1) can be approximated with a second-order of accuracy as

$$u_{xx}|_{i,j} = \delta_x^2 u_{i,j} + O(\Delta x^2), \quad u_{yy}|_{i,j} = \delta_y^2 u_{i,j} + O(\Delta y^2), \tag{2.2}$$

where $\delta_x^2 u_{i,j} = \dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$ and $\delta_y^2 u_{i,j} = \dfrac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$ define the second-order central difference operators at every grid point (x_i, y_j).

Using these finite difference operators, Eq. (2.1) can be discretized at a given point (x_i, y_j) as

$$\delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} - \lambda u_{i,j} = f_{i,j} + O(\|\boldsymbol{h}\|^2), \tag{2.3}$$

where $\boldsymbol{h} = (\Delta x, \Delta y)$, or, in a more compact form

$$L_h u_h = f_h + O(\|\boldsymbol{h}\|^2), \tag{2.4}$$

where L_h is the discrete partial differential operator L with

$$Lu = \Delta u - \lambda u \tag{2.5}$$

on the rectangular grid G_h. $O(\|\boldsymbol{h}\|^2)$ denotes the truncated terms in the order of $O(\Delta x^2 + \Delta y^2)$, i.e. consistency relations of the form can be derived

$$Lu - L_h = O(\|\boldsymbol{h}\|^2) \text{ for } \|\boldsymbol{h}\| \to 0 \quad .$$

In a stencil notation, and under the assumption of equal mesh-sizes in $x-$ and $y-$direction (i.e $\Delta x = \Delta y = h$), the discrete operator L_h can be written as

$$L_{\boldsymbol{h}} \hat{=} \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & 4-\lambda & 1 \\ & 1 & \end{bmatrix}_h \tag{2.6}$$

for $(x_i, y_j) \in \Omega_h$ not adjacent to the boundary. The points near the boundary are treated in a manner that will be discussed later.

It can be verified, that the second- order Central Difference Scheme (CDS) for the second-order linear elliptic PDEs is compact for the model problem.

2.1.3 High accurate approximations

High-order of accuracy discretization schemes for the model problem can be obtained following two different approaches. The first one suggests the discretization of the Eq. (2.1) with a fourth order central-difference scheme based on the fourth-order formula of the sec-

ond derivative

$$u_{xx}|_{i,j} = \frac{-u_{i+2,j} + 16u_{i+1,j} - 30u_{i,j} + 16u_{i-1,j} - u_{i-2,j}}{12\Delta x^2} + O(\Delta x^4). \tag{2.7}$$

It shall be noticed that the above formula involves more unknowns compared to the corresponding second-order, thus a wider, fourth-order accurate stencil emerges.
With the aim of a narrow stencil, an variant approach should be followed. Following the derivation procedure in [32, 49] a fourth order compact scheme will be constructed.

The one-dimensional case

At first, the analogous one-dimensional model problem

$$u_{xx} - \lambda u = f \ , \tag{2.8}$$

is considered, where the second derivative u_{xx} at a grid point x_i is approximated using the following central difference operator

$$u_{xx} = \delta_x^2 u - \frac{\Delta x^2}{12} u_{xxxx} + O(\Delta x^4), \tag{2.9}$$

where u_{xxxx} refers to the 4th derivative's evaluator at the grid point x_i. In order to achieve a fourth order truncation accuracy in (2.9), Eq. (2.8) is differentiated twice and the second-order central difference scheme on f_{xx} and u_{xx} is applied. A second-order approximation of u_{xxxx} has the form

$$\begin{aligned} u_{xxxx} &= f_{xx} + \lambda u_{xx} \\ &= \delta_x^2 f + \lambda \delta_x^2 u + O(\Delta x^2) \end{aligned} \ . \tag{2.10}$$

The substitution of (2.10) into (2.9) yields a fourth- order compact approximation for the second derivative of form

$$\begin{aligned} u_{xx} &= \delta_x^2 u - \frac{\Delta x^2}{12}(\delta_x^2 f + \lambda \delta_x^2 u + O(\Delta x^2)) + O(\Delta x^4) \\ &= \delta_x^2 u - \frac{\Delta x^2}{12}(\delta_x^2 f + \lambda \delta_x^2 u) + O(\Delta x^4) \\ &= \left(1 - \lambda \frac{\Delta x^2}{12}\right)\delta_x^2 u - \frac{\Delta x^2}{12}\delta_x^2 f + O(\Delta x^4). \end{aligned} \tag{2.11}$$

The fourth order compact approximation for the one dimensional problem (2.8) can be expressed as

$$\left(1 - \lambda \frac{\Delta x^2}{12}\right)\delta_x^2 u - \frac{\Delta x^2}{12}\delta_x^2 f - \lambda u = f + O(\Delta x^4), \tag{2.12}$$

which can be formulated as

$$\left[\left(1 - \lambda \frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right] u = \left(1 + \frac{\Delta x^2}{12}\delta_x^2\right) f + O(\Delta x^4), \tag{2.13}$$

or

$$\left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)^{-1} \left[\left(1 - \lambda \frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right] u = f + O(\Delta x^4). \tag{2.14}$$

The operator $\left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)^{-1}$ has a symbolic meaning only and the fourth-order compact difference discretization scheme is given by Eq. (2.12).

Therefore,

$$u_{xx} - \lambda u = \left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)^{-1} \left[\left(1 - \lambda \frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right] u + O(\Delta x^4). \tag{2.15}$$

Similarly, an analogous symbolic fourth-order compact approximation operator for the y direction can be produced

$$u_{yy} - \lambda u = \left(1 + \frac{\Delta y^2}{12}\delta_y^2\right)^{-1} \left[\left(1 - \lambda \frac{\Delta y^2}{12}\right)\delta_y^2 - \lambda\right] u + O(\Delta y^4). \tag{2.16}$$

The two-dimensional case

The model problem can take an appropriate form the following way

$$u_{xx} + u_{yy} - \lambda u = f \Leftrightarrow \overbrace{u_{xx} - \lambda u}^{Eq.\ (2.15)} + \overbrace{u_{yy} - \lambda u}^{Eq.\ (2.16)} = f - \lambda u. \tag{2.17}$$

The use of the symbolic fourth-order compact approximation operators for the quantities $u_{xx} - \lambda u$ and $u_{yy} - \lambda u$ in Eq. (2.1) leads to the following way of their expression

$$\begin{aligned}\left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)^{-1}\left[\left(1 - \lambda\frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right]u + \left(1 + \frac{\Delta y^2}{12}\delta_y^2\right)^{-1}\left[\left(1 - \lambda\frac{\Delta y^2}{12}\right)\delta_y^2 - \lambda\right]u \\ = f - \lambda u + O(\|\boldsymbol{h}\|^4),\end{aligned} \tag{2.18}$$

where $O(\|\boldsymbol{h}\|^4)$ denotes the truncated terms in the order of $O(\Delta x^4 + \Delta y^4)$. The application of the symbolic operators gives the following relation

$$\begin{aligned}\left(1 + \frac{\Delta y^2}{12}\delta_y^2\right)\left[\left(1 - \lambda\frac{\Delta x^2}{12}\right)\delta_x^2 - \lambda\right]u + \left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)\left[\left(1 - \lambda\frac{\Delta y^2}{12}\right)\delta_y^2 - \lambda\right]u \\ = \left(1 + \frac{\Delta x^2}{12}\delta_x^2\right)\left(1 + \frac{\Delta y^2}{12}\delta_y^2\right)(f - \lambda u) + O(\|\boldsymbol{h}\|^4),\end{aligned} \tag{2.19}$$

which is equivalent to the subsequent

$$\begin{aligned}&\left(1 + \frac{\Delta y^2}{12}\delta_y^2 - \lambda\frac{\Delta x^2}{12} - \lambda\frac{\Delta x^2\Delta y^2}{144}\right)\delta_x^2 u + \left(1 + \frac{\Delta y^2}{12}\delta_y^2 - \lambda\frac{\Delta x^2}{12} - \lambda\frac{\Delta x^2\Delta y^2}{144}\right)\delta_x^2 u \\ &- \lambda\left(2 + \frac{\Delta y^2}{12}\delta_y^2 + \frac{\Delta x^2}{12}\delta_x^2\right)u = \left(1 + \frac{\Delta y^2}{12}\delta_y^2 + \frac{\Delta x^2}{12}\delta_x^2 + \frac{\Delta x^2\Delta y^2}{144}\right)(f - \lambda u) + O(\|\boldsymbol{h}\|^4).\end{aligned} \tag{2.20}$$

A few steps after, the above equation can be expressed as

$$\left(1+\frac{\Delta y^2}{12}\delta_y^2-\lambda\frac{\Delta x^2}{12}\right)\delta_x^2 u+\left(1+\frac{\Delta y^2}{12}\delta_y^2-\lambda\frac{\Delta x^2}{12}\right)\delta_x^2 u-\lambda u \\ =\left(1+\frac{\Delta y^2}{12}\delta_y^2+\frac{\Delta x^2}{12}\delta_x^2\right)f+O(\|\boldsymbol{h}\|^4). \tag{2.21}$$

Here, $O(\Delta x^2\cdot\Delta y^2)$ is absorbed into the $O(\|\boldsymbol{h}\|^4)$ global term. The general fourth-order compact approximation scheme for the Helmholtz equation after dropping the $O(\|\boldsymbol{h}\|^4)$ term and rearranging, can be formulated as

$$(\delta_x^2+\delta_y^2)u+\frac{1}{12}(\Delta x^2+\Delta y^2)\delta_x^2\delta_y^2 u-\frac{\lambda}{12}(\Delta x^2\delta_x^2+\Delta y^2\delta_y^2)u=f+\frac{1}{12}(\Delta x^2\delta_x^2+\Delta y^2\delta_y^2)f. \tag{2.22}$$

Defining the mesh ratio as $\gamma=\Delta x/\Delta y$ and multiplying by $6\Delta x^2$, Eq. (2.22) leads to the High-Order Compact discretization scheme (HOC)

$$d(u_{i+1,j+1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i-1,j-1})+c(u_{i+1,j}+u_{i-1,j})+b(u_{i,j+1}+u_{i,j-1}) \\ -a_{i,j}u_{i,j}=\frac{\Delta x^2}{2}(8f_{i,j}+f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1}), \tag{2.23}$$

with coefficients

$$a=10(1+\gamma^2)+4\lambda\Delta x^2,\quad b=5-\gamma^2-\tfrac{\lambda\Delta x^2}{2},\quad c=5\gamma^2-1-\tfrac{\lambda\Delta x^2}{2}\quad d=(1+\gamma^2)/2,\ \text{for}$$

$i=1,\ldots,N_x$ and $j=1,\ldots,N_y$.

In the specific case of $\lambda=0$ (Poisson equation) with equal mesh sizes $\Delta x=\Delta y$, the above approximation formula can be written as

$$u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} + 4(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - 20u_{i,j} = \frac{\Delta x^2}{2}(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}), \tag{2.24}$$

This compact nine-point numerical scheme (2.24) is generally called "Mehrstellen", [57]. Several authors have made use of fourth-order compact schemes to solve elliptic PDEs ([32, 63, 61]). In this thesis, the upgrade presented points out an improved touch to impose realistic boundary conditions, described elaborately below.

2.1.4 HOC for boundary equations

The proposed cell-centered HOC difference equations to the boundary node with indexes $i = 1, N_x$ or $j = 1, N_y$ involve fictitious unknowns outside the domain Ω. These fictitious unknowns can be eliminated from the resulting linear system using appropriate boundary closure formulas, applying Dirichlet, Neumann and/or Robin boundary conditions (BCs), valid on the physical boundary.

Case of Dirichlet BCs

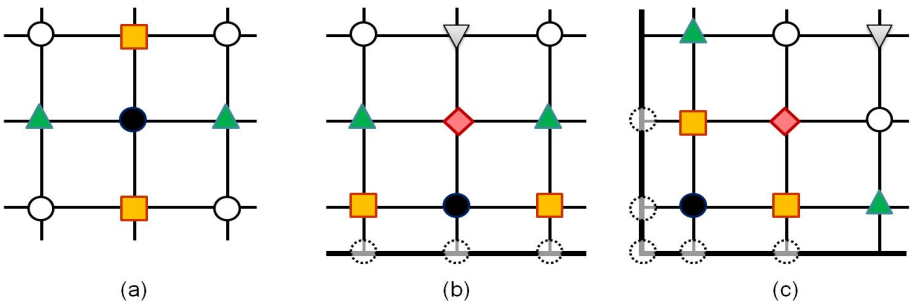

Figure 2-2: The finite difference stencil with the weights of each unknown for Dirichlet boundary conditions; (a) interior nodes, (b) nodes close to bottom boundary and (c) bottom-left node close to the boundary.

In the first instance, the study seeks out compact high-order formulas for Dirichlet conditions. Namely, Dirichlet boundary conditions are considered and applied on the left

($x = 0$) and bottom (y=0) vertices of the domain Ω, for convenience purposes. Similar formulas apply for all other vertices.

Next, by considering the grid function ϕ with stepsize h and the function values $\phi_i = \phi(x_i)$, the fourth-order one-dimensional approximation formulas are developed as

$$\phi_{i+\frac{1}{2}} + \alpha\phi_{i+\frac{3}{2}} = a\phi_i + b\phi_{i+1} + c\phi_{i+2} + d\phi_{i+3}, \tag{2.25}$$

where $\alpha = free$, $a = \frac{5}{16} - \frac{\alpha}{16}$, $b = \frac{15}{16} + \frac{9\alpha}{16}$, $c = -\frac{5}{16} + \frac{9\alpha}{16}$ and $d = \frac{1}{16} - \frac{\alpha}{16}$ [24]. Especially when $\alpha = 0$, Eq. (5.30) takes the form

$$\phi_{i+\frac{1}{2}} = \frac{5}{16}\phi_i + \frac{15}{16}\phi_{i+1} - \frac{5}{16}\phi_{i+2} + \frac{1}{16}\phi_{i+3}. \tag{2.26}$$

Solving for the variable ϕ_i,

$$\phi_i = \frac{16}{5}\phi_{i+\frac{1}{2}} - 3\phi_{i+1} + \phi_{i+2} - \frac{1}{5}\phi_{i+3} \tag{2.27}$$

the above formula can be rewritten for the variable $u_{i,j}$ as

$$u_{i,0} = \frac{16}{5}u_{i,\frac{1}{2}} - 3u_{i,1} + u_{i,2} - \frac{1}{5}u_{i,3} \qquad \text{for } i = 2, ..., N_x - 1, \tag{2.28}$$

$$u_{0,j} = \frac{16}{5}u_{\frac{1}{2},j} - 3u_{1,j} + u_{2,j} - \frac{1}{5}u_{3,j} \qquad \text{for } j = 2, ..., N_y - 1 \tag{2.29}$$

in the x- and y-direction, respectively. High-order difference formulas (2.28) and (2.29) involve known values on the boundary and unknowns inside the domain Ω. That way, the totality of fictitious unknowns can be eliminated outside the bottom or left boundary for any arbitrary HOC difference equation $(i, 1),\ i = 2, N_x - 1$ or $(1, j),\ j = 2, N_y - 1$.

The four fictitious unknowns adjacent to the edges of Ω are being treated separately. For instance, the following relation is being used to approximate the fictitious unknown at the bottom left corner

$$u_{0,0} = \frac{16}{5} u_{\frac{1}{2},\frac{1}{2}} - 3u_{1,1} + u_{2,2} - \frac{1}{5} u_{3,3} \quad . \tag{2.30}$$

Alternately, a more intricate formula that is based on the tensor product of the one-dimensional formula (2.27), with respect to $x-$ and $y-$directions, can be used. Namely, when writing (Eq. 5.31) in the following form

$$\phi(x_i) = S_h \phi(x) = \sum_{\kappa=1}^{6} s_\kappa \phi(x_i + \kappa \frac{h}{2}) \tag{2.31}$$

the stencil

$$S_h \hat{=} [s_\kappa]_h = [\ \tfrac{16}{5} \quad -3 \quad 0 \quad 1 \quad 0 \quad -\tfrac{1}{5}\ ] \tag{2.32}$$

suggests the application of the above-mentioned formula. Using the Kronecker product of the above vector, with respect to $x-$ and $y-$direction, a new approximation formula for the bottom left fictitious unknown can be obtained

$$\begin{aligned} u_{0,0} =& \frac{256}{25} u_{\frac{1}{2},\frac{1}{2}} - \frac{48}{5}(u_{1,\frac{1}{2}} + u_{\frac{1}{2},1}) + \frac{16}{5}(u_{2,\frac{1}{2}} + u_{\frac{1}{2},2}) \\ & - \frac{16}{25}(u_{3,\frac{1}{2}} + u_{\frac{1}{2},3}) + 9u_{1,1} + u_{2,2} + \frac{1}{25} u_{3,3} \\ & - 3(u_{1,2} + u_{2,1}) + \frac{3}{5}(u_{1,3} + u_{3,1}) - \frac{1}{5}(u_{2,3} + u_{3,2}) \end{aligned} \tag{2.33}$$

that interprets the stencil

$$[s_{\boldsymbol{\kappa}}]_{\boldsymbol{h}} = [s_{\kappa_1}]_{\Delta x} \otimes [s_{\kappa_2}]_{\Delta y} = \begin{bmatrix} -\frac{16}{25} & \frac{3}{5} & 0 & -\frac{1}{5} & 0 & \frac{1}{25} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{16}{5} & -3 & 0 & 1 & 0 & -\frac{1}{5} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-48}{5} & 9 & 0 & -3 & 0 & \frac{3}{5} \\ \frac{256}{25} & \frac{-48}{5} & 0 & \frac{16}{5} & 0 & -\frac{16}{25} \end{bmatrix} \tag{2.34}$$

with $\boldsymbol{\kappa} = (\kappa_1, \kappa_2) \in \mathbb{Z}^2$ and $\boldsymbol{h} = (\Delta x, \Delta y) \in \mathbb{R}$ the spatial discretization (grid-steps) of the computational domain. The other corner fictitious unknowns are eliminated in a similar way.

Fig. 2-2 presents the stencil form of the fourth-order finite compact scheme for the Dirichlet case based on the curtailed (restricted) closure formula. The dashed circles represent the values of function u on the boundary.

Case of Neumann BCs

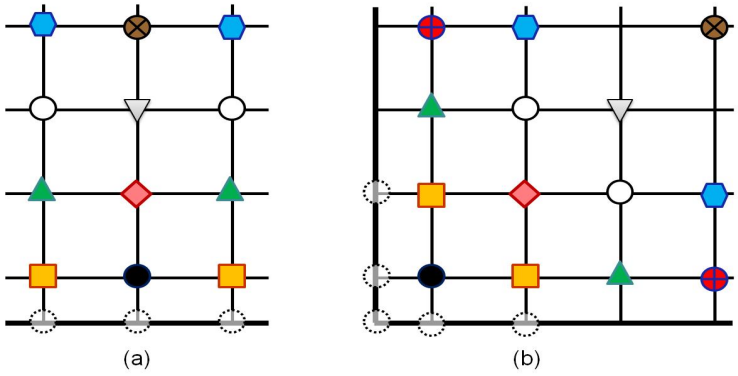

Figure 2-3: The finite difference stencil with the weights of each unknown for Neumann boundary conditions; (a) nodes close to bottom boundary and (b) bottom-left node close to the boundary.

A comparable elimination procedure can be followed in the case of Neumann conditions. Linear equations corresponding to the bottom left vertices of the domain are derived, while those on the rest boundary vertices can be similarly formulated.

For this end, the fourth-order one-dimensional approximation formula [24]

$$\phi'_{i+\frac{1}{2}} + \alpha\phi'_{i+\frac{3}{2}} = (a\phi_i + b\phi_{i+1} + c\phi_{i+2} + d\phi_{i+3} + e\phi_{i+4})/h, \tag{2.35}$$

where $\alpha = free$, $a = -\frac{22}{24} + \frac{\alpha}{24}$, $b = \frac{17}{24} - \frac{9\alpha}{8}$, $c = \frac{3}{8} + \frac{9\alpha}{8}$, $d = -\frac{5}{24} - \frac{\alpha}{24}$ and $e = \frac{1}{24}$ is applied, selecting $\alpha = 0$ and resulting to

$$\phi'_{i+\frac{1}{2}} = (-\frac{22}{24}\phi_i + \frac{17}{24}\phi_{i+1} + \frac{3}{8}\phi_{i+2} - \frac{5}{24}\phi_{i+3} + \frac{1}{24}\phi_{i+4})/\Delta x\ . \tag{2.36}$$

Solving for the nodal unknown ϕ_i variable, the corresponding $u_{i,j}$ unknowns can be expressed as

$$u_{i,0} = -\frac{12\Delta y}{11}\frac{\partial u_{i,\frac{1}{2}}}{\partial y} + \frac{17}{22}u_{i,1} + \frac{9}{22}u_{i,2} - \frac{5}{22}u_{i,3} + \frac{1}{22}u_{i,4}\ for\ i = 2, ..., N_x - 1, \tag{2.37}$$

$$u_{0,j} = -\frac{12\Delta x}{11}\frac{\partial u_{\frac{1}{2},j}}{\partial x} + \frac{17}{22}u_{1,j} + \frac{9}{22}u_{2,j} - \frac{5}{22}u_{3,j} + \frac{1}{22}u_{4,j}\ for\ j = 2, ..., N_y - 1 \tag{2.38}$$

in x- and y-direction respectively. It is should be cited, that the respective elimination formulae for Neumann conditions on the right and the top boundaries can be derived from (2.35) by reversing the signs in the variables from a to e but not that of α.

The nodal unknown $u_{0,0}$ at the bottom left boundary corner is substituted with

$$u_{0,0} = -\frac{12}{11}\sqrt{\Delta x^2 + \Delta y^2}\frac{\partial u_{\frac{1}{2},\frac{1}{2}}}{\partial \mathbf{k}} + \frac{17}{22}u_{1,1} + \frac{9}{22}u_{2,2} - \frac{5}{22}u_{3,3} + \frac{1}{22}u_{4,4}, \tag{2.39}$$

where $\mathbf{k} = \frac{\Delta x}{\sqrt{\Delta x^2+\Delta y^2}}\mathbf{i} + \frac{\Delta y}{\sqrt{\Delta x^2+\Delta y^2}}\mathbf{j}$ and vectors $\mathbf{i}$ and $\mathbf{j}$ are the unitary vectors for each spatial direction.

The relation (2.39) can be effortlessly written as

$$u_{0,0} = -\frac{12}{11}\Delta x\left(\frac{\partial u}{\partial x} + \frac{1}{\gamma}\frac{\partial u}{\partial y}\right)_{\frac{1}{2},\frac{1}{2}} + \frac{17}{22}u_{1,1} + \frac{9}{22}u_{2,2} - \frac{5}{22}u_{3,3} + \frac{1}{22}u_{4,4}, \tag{2.40}$$

where γ is the mesh ratio. Alternatively, the tensor product of the one dimensional formulas (2.37) and (2.38) generates a new approximation formula for the left-bottom fictitious corner. Using the stencil forms of these formulae and the Kronecker product, the extended formula can be derived

$$\begin{aligned} u_{0,0} =& \frac{144}{121}\frac{\partial^2 u_{\frac{1}{2},\frac{1}{2}}}{\partial x \partial y} + \frac{1}{121}[-102(\frac{\partial u_{1,\frac{1}{2}}}{\partial y} + \frac{\partial u_{\frac{1}{2},1}}{\partial x}) - 54(\frac{\partial u_{2,\frac{1}{2}}}{\partial y} + \frac{\partial u_{\frac{1}{2},2}}{\partial x}) + 30(\frac{\partial u_{3,\frac{1}{2}}}{\partial y} + \frac{\partial u_{\frac{1}{2},3}}{\partial x}) \\ & - 6(\frac{\partial u_{4,\frac{1}{2}}}{\partial y} + \frac{\partial u_{\frac{1}{2},4}}{\partial x})] + \frac{1}{484}[289u_{1,1} + 81u_{2,2} + 25u_{3,3} + u_{4,4} + 153(u_{1,2} + u_{2,1}) \\ & - 85(u_{1,3} + u_{3,1}) + 17(u_{1,4} + u_{4,1}) - 45(u_{2,3} + u_{3,2}) + 9(u_{2,4} + u_{4,2}) - 5(u_{3,4} + u_{4,3})] \end{aligned} \tag{2.41}$$

Contrary to Eq. (2.33), in the latter formula the mixed partial derivative is present at the origin, and as a result, its application is a rather complicated procedure. Eq. (2.41) can be applied right after the association of the term $\partial^2 u_{\frac{1}{2},\frac{1}{2}}/\partial x \partial y$ to known function values of u on the boundary.

Fig. 2-3 depicts the stencil form of the fourth-order finite compact scheme in the case of Neumann boundary conditions, based on the closure formula Eq. (2.39); the dashed circles represent the values of the derivatives of u on the $\partial\Omega$. The interior stencil is independent of the boundary conditions, hence it is omitted from the figure.

Case of Robin BCs

In case of Robin boundary conditions, the fictitious unknowns in the following manner. Considering the following boundary relation

$$\delta u + \epsilon \frac{\partial u}{\partial \mathbf{n}} = g \ , \tag{2.42}$$

for some non-zero constants δ and ϵ and a given function g defined on the boundary $\partial\Omega$. Multiplying formulas (2.26) and (2.36) by δ and ϵ respectively, adding them afterwards and solving for ϕ_i, the fictitious unknowns orientated on the outside of the left edge of $\partial\Omega$ can be approximated by

$$u_{0,j} = (\alpha u_{\frac{1}{2},j} + \beta \frac{\partial u}{\partial x}_{\frac{1}{2},j})/P(\Delta x) + Q(u_{1,j}, u_{2,j}, u_{3,j}, u_{4,j}; \Delta x), \quad \text{for } 1 \leq j \leq N_y \ , \tag{2.43}$$

where $P(\Delta x)$ is a function of Δx and $Q(u_{i+1,j}, u_{i+2,j}, u_{i+3,j}, u_{i+4,j}; \Delta x)$ is a quantity of components of u based on the grid step Δx. Working in a similar way, a complete set of elimination formulae can be produced, in order to manage all the fictitious unknowns in the Robin boundary conditions case.

Mixed BCs

When mixed boundary conditions in BVPs apply, the solution should satisfy the Dirichlet or the Neumann boundary condition in a mutually exclusive way on disjoint parts of the boundary. For example, if $\partial\Omega = \Gamma_1 \cup \Gamma_2$, where Γ_1 and Γ_2 two disjoint parts of the boundary, u verifies the following equations:

$$\begin{array}{l} u\big|_{\Gamma_1} = s \\ \dfrac{\partial u}{\partial n}\Big|_{\Gamma_2} = t \end{array} \ , \tag{2.44}$$

where s and t are given functions defined on those portions of the boundary. If it's assumed that Γ_1 applies on the left and right edges of $\partial\Omega$ and $\Gamma_2 = \partial\Omega - \Gamma_1$, the unknown on the outside of Γ_1 can be handled by Eq. (2.29), while using Eq. (2.37) the unknowns on the outside of Γ_2 can be eliminated. Moreover, as both boundary conditions (Dirichlet and

Neumann) are valid in the corners of Ω, the corresponding fictitious unknowns can be approximated by Eq. (2.30) or, alternately, using the tensor product of the operators based on Eq. (2.29) and (2.37). Then, the resulted formula can be written in a stencil form as

$$[s_\kappa]_h = \begin{bmatrix} \frac{8}{55} & \frac{-3}{22} & 0 & \frac{1}{22} & 0 & \frac{-1}{110} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-8}{11} & \frac{15}{22} & 0 & \frac{-5}{22} & 0 & \frac{1}{22} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{72}{55} & \frac{-27}{22} & 0 & \frac{9}{22} & 0 & \frac{-9}{110} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{136}{55} & \frac{-51}{22} & 0 & \frac{17}{22} & 0 & \frac{-17}{110} \\ \frac{-192\Delta y}{55} & \frac{36\Delta y}{11} & 0 & \frac{-12\Delta y}{11} & 0 & \frac{12\Delta y}{55} \end{bmatrix}. \tag{2.45}$$

It should be noted that the last row of the above stencil refers to the boundary values of $\partial u/\partial y$, which are known since the Neumann type of condition is considered on the bottom edge of the boundary.

2.1.5 Right hand side discretization

The evaluation methodology for handling fictitious values of the function $f(x, y)$ appears on the right hand side of Eq. (2.23), is presented regardless of the boundary conditions. For this, the discrete values of f are defined as $f(x_i, y_j) \equiv f_{i,j}$. The equations corresponding to the nodes in the vicinity of the boundary contain a single fictitious value, except those that correspond to the corner nodes, containing an additional one.

To set it straight, the first equation for $(i = 1, j = 1)$ involves values $f_{1,0}$ and $f_{0,1}$, while the equation for $(i = 2, j = 1)$ involves only value $f_{1,0}$. All these values can be eliminated upon the application of the fourth order approximation formula

$$\phi_i = 4\phi_{i+1} - 6\phi_{i+2} + 4\phi_{i+3} - \phi_{i+4}. \tag{2.46}$$

It is worth of mention, that the above equation does not involve values on the $\partial\Omega$. There are certain applications where function $f(x, y)$ can not be determined or even defined on the boundary, thus formula (2.46) fullfils this restriction. Fig. 2-4 represents the right hand side of the HOC scheme in a stencil form.

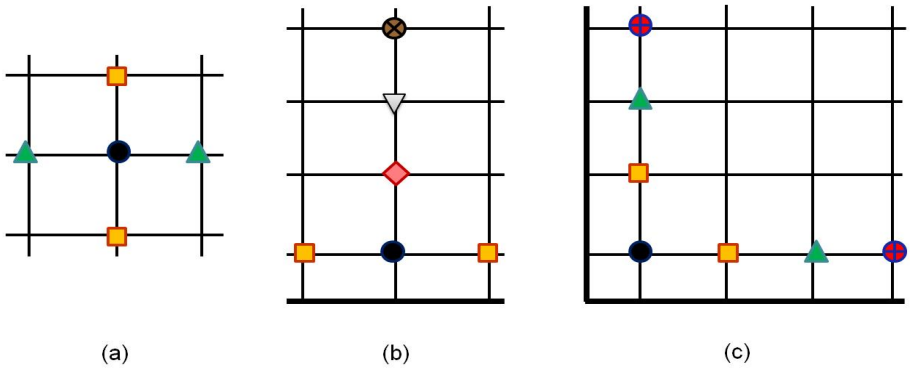

Figure 2-4: The finite difference stencil with the weights for each value of function f independent of the boundary conditions; (a) interior nodes, (b) nodes close to bottom boundary and (c) bottom-left node close to the boundary.

2.1.6 The Stencil form

Using the infinite grid type

$$\mathbf{G}_h := \{(x, y) : x = x_i = (i - \frac{1}{2})\Delta x,\ y = y_j = (j - \frac{1}{2})\Delta y;\ i, j \in \mathbb{Z}\}\ . \tag{2.47}$$

the fourth-order compact scheme for the problem, when any type of boundary conditions is applied, is simplified as

$$L_h u_h(\mathbf{x}) = K_h f_h(\mathbf{x}) \text{ at any } \mathbf{x} \in \Omega_h, \tag{2.48}$$

Here, u_h and f_h are grid functions of the function u and f on Ω_h, and L_h and K_h are linear operators

$$L_h, K_h : \mathcal{G}(\Omega_h) \to \mathcal{G}(\Omega_h) \,, \tag{2.49}$$

where $\mathcal{G}$ denotes the linear space of grid functions on Ω. Eq. (2.48) can be represented either as a system of linear algebraic equations or as a single grid equation.

For instance, Eq. (2.23) is equivalent to Eq. (2.48) for any interior equation. In this case, the linear operators have the stencil notations

$$L_h = \frac{1}{\Delta x^2} \begin{bmatrix} d & c & d \\ b & -a & b \\ d & c & d \end{bmatrix}_h \quad \text{and} \quad K_h = \frac{1}{2} \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix}_h . \tag{2.50}$$

For the boundary equations, wider stencils are obtained because the discretization scheme involves more unknowns on the left-hand side and the function f needs to be estimated more times on the right-hand side.

2.2 Linear system solution

The HOC discretization scheme produces a linear system of equations. It is hereby assumed that the grid points order comes in a row-wise lexicographical fashion. Then, the discrete Eq. (2.48) for all interior points results to a sparse linear system

$$A\boldsymbol{u} = \boldsymbol{b}. \tag{2.51}$$

with the block structure

$$\begin{bmatrix} A_1 & A_2 & A_3 & A_4 & O & \dots & O & O \\ A_5 & A_6 & A_5 & O & O & \dots & O & O \\ O & A_5 & A_6 & A_5 & O & \dots & O & O \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ O & O & \dots & O & A_5 & A_6 & A_5 & O \\ O & O & \dots & O & O & A_5 & A_6 & A_5 \\ O & O & \dots & O & \widehat{A_4} & \widehat{A_3} & \widehat{A_2} & \widehat{A_1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_{N_y-2} \\ \mathbf{u}_{N_y-1} \\ \mathbf{u}_{N_y} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \vdots \\ \mathbf{b}_{N_y-2} \\ \mathbf{b}_{N_y-1} \\ \mathbf{b}_{N_y} \end{bmatrix}, \tag{2.52}$$

where the coefficient matrix A is of $N \times N$ order, and $N = N_x N_y$. The vector $\mathbf{u}$ of the unknowns and the right-hand side vector (RHS) $\mathbf{b}$ are both of order N. The basic-matrices A_i for $i = 1, \dots, 6$ and $\widehat{A_j}$ for $j = 1, \dots, 4$ are structured as

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & 0 & \dots & 0 & 0 \\ a_5 & a_6 & a_5 & 0 & 0 & \dots & 0 & 0 \\ 0 & a_5 & a_6 & a_5 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_5 & a_6 & a_5 & 0 \\ 0 & 0 & \dots & 0 & 0 & a_5 & a_6 & a_5 \\ 0 & 0 & \dots & 0 & \widehat{a_4} & \widehat{a_3} & \widehat{a_2} & \widehat{a_1} \end{bmatrix} \in \mathbb{R}^{N_x \times N_x}. \tag{2.53}$$

It can be noticed that coefficients a_3, a_4 and $\widehat{a_3}$, $\widehat{a_4}$ in the first and last line of the sub-matrices A_i respectively have usually non-zero entries. Their presence results to the unability to apply the Thomas' algorithm [90] or the Cyclic Reduction algorithm [95] directly, when solving the linear system. However, the elimination of these entries is feasible by multiplying with a_4/a_5 the third equation and subtracting it from the first equation and so on.

2.2.1 Special cases for matrix A

It can be shown, that, in the case of the same boundary conditions apply in all sides of the domain Ω , the matrices $\widehat{A_j}$ are equal to A_j for $j = 1, \ldots, 4$. Their entries are listed in Table 2.1 when Dirichlet boundary conditions apply around the domain Ω

Table 2.1: Entries of the basic-matrices for Dirichlet boundary conditions.

	a_1	a_2	a_3	a_4	a_5	a_6
A_1	$\frac{e-47s}{2}$	$t^{'} - e$	$-\frac{u^{'}}{5} + \frac{e}{10}$	0	$\frac{v^{'}-e}{2}$	$w^{'} - e$
A_2	$\frac{t}{2} - 2e$	$2s$	$-\frac{s}{10}$	0	s	$2(u-e)$
A_3	$\frac{e-u}{5}$	$-\frac{s}{10}$	$-\frac{s}{10}$	0	$-\frac{s}{10}$	$\frac{e-u}{5}$
A_4	0	0	0	0	0	0
A_5	$\frac{v}{2} - e$	s	$-\frac{s}{10}$	0	$\frac{s}{2}$	$u - e$
A_6	$-w - \frac{5e}{2}$	$2u^{'} - e$	$-\frac{u^{'}}{5} + \frac{e}{10}$	0	$u^{'} - \frac{e}{2}$	$-2(5s + 2e)$

where $s = 1 + \gamma^2$, $t = 17\gamma^2 - 7$, $u = 5\gamma^2 - 1$, $v = 7\gamma^2 - 5$, $w = 7\gamma^2 + 25$, $t' = 17 - 7\gamma^2$, $u' = 5 - \gamma^2$ $v' = 7 - 5\gamma^2$, $w' = 7 + 25\gamma^2$, $e = \lambda\Delta x^2$.

In case of Neumann boundary conditions, the entries of these matrices are displayed in the following Table 2.2

Table 2.2: Entries of the basic-matrices for Neumann boundary conditions.

	a_1	a_2	a_3	a_4	a_5	a_6
A_1	$-\frac{287s+227e}{44}$	$\frac{t'-31e}{44}$	$\frac{5(e-2u')}{44}$	$-\frac{2u'+e}{44}$	$\frac{v'-22e}{44}$	$-\frac{105e+w'}{22}$
A_2	$\frac{t-62e}{44}$	$\frac{49s}{44}$	$-\frac{5s}{44}$	$\frac{s}{44}$	$\frac{31s}{44}$	$\frac{31(u-e)}{22}$
A_3	$\frac{5(e-u)}{22}$	$-\frac{5s}{44}$	$-\frac{5s}{44}$	0	$-\frac{5s}{44}$	$\frac{5(e-u)}{22}$
A_4	$\frac{u-e}{22}$	$\frac{s}{44}$	0	$\frac{s}{44}$	$\frac{s}{44}$	$\frac{u-e}{22}$
A_5	$\frac{v}{44}-e$	$\frac{9s}{22}$	$-\frac{5s}{44}$	$\frac{s}{44}$	$\frac{s}{2}$	$u-e$
A_6	$-\frac{2w+193e}{44}$	$\frac{9(2u'-e)}{22}$	$\frac{5(e-2u')}{44}$	$-\frac{2u'+e}{44}$	$\frac{2u'-e}{2}$	$-2(5s+2e)$

where $s = 1+\gamma^2$, $t = 327\gamma^2 - 45$, $u = 5\gamma^2 - 1$, $v = 237\gamma^2 - 27$, $w = 237\gamma^2 + 135$, $t' = 327 - 45\gamma^2$, $u' = 5 - \gamma^2$, $v' = 237 - 27\gamma^2$, $w' = 237 + 135\gamma^2$, $e = \lambda\Delta x^2$.

An interesting comment regarding matrix A_4 is that A_4 is the zero matrix, when Dirichlet BCs apply. Additionally, the structure of the basic-matrices depends on the applied conditions. For instance, if Dirichlet BCs are assumed, then each basic-matrix is pentadiagonal while in the Neumann BCs case every basic-matrix is eptadiagonal. These differences arise from the application of different interpolation operators in order to manipulate the boundary conditions. Wider interpolation schemes result to banded matrices with larger bandwidth. For example, involving an additional node into the interpolation formula, the bandwidth of the basic-matrices is increased by two.

2.2.2 Special cases of RHS

For the special case of a zero function on the boundary, the vector $\boldsymbol{b}$ can be simplified to

$$\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2^{(2)} \\ \mathbf{b}_2^{(3)} \\ \vdots \\ \mathbf{b}_2^{(N_y-2)} \\ \mathbf{b}_2^{(N_y-1)} \\ \widehat{\mathbf{b}_1} \end{bmatrix} , where \quad \mathbf{b}_1, \mathbf{b}_2^{(j)}, \widehat{\mathbf{b}_1} = \begin{bmatrix} b_1 \\ b_2^{(2)} \\ b_2^{(3)} \\ \vdots \\ b_2^{(N_x-2)} \\ b_2^{(N_x-1)} \\ \widehat{b_1} \end{bmatrix} \quad for\ j = 2, \ldots, N_x - 1\,. \tag{2.54}$$

Tables 2.3 and 2.4 list the entries of those sub-vectors. If non-zero boundary conditions

Table 2.3: Entries of the sub-vectors $\mathbf{b}_1$ and $\widehat{\mathbf{b}_1}$ for zero boundary conditions.

	$\mathbf{b}_1$	$\widehat{\mathbf{b}_1}$
b_1	$16f_{1,1} - 5(f_{2,1} + f_{1,2}) + 4(f_{3,1} + f_{1,3}) - (f_{4,1} + f_{1,4})$	$16f_{1,N_y} - 5(f_{2,N_y} + f_{1,N_y-1}) + 4(f_{3,N_y} + f_{1,N_y-2}) - (f_{4,N_y} + f_{1,N_y-3})$
$b_2^{(i)}$	$12f_{i,1} + f_{i-1,1} + f_{i+1,1} - 5f_{i,2} + 4f_{i,3} - f_{i,4}$	$12f_{i,N_y} + f_{i-1,N_y} + f_{i+1,N_y} - 5f_{i,N_y-1} + 4f_{i,N_y-2} - f_{i,N_y-3}$
$\widehat{b_1}$	$16f_{N_x,1} - 5(f_{N_x-1,1} + f_{N_x,2}) + 4(f_{N_x-2,1} + f_{N_x,3}) - (f_{N_x-3,1} + f_{N_x,4})$	$16f_{N_x,N_y} - 5(f_{N_x-1,N_y} + f_{N_x,N_y-1}) + 4(f_{N_x-2,N_y} + f_{N_x,N_y-2}) - (f_{N_x-3,N_y} + f_{N_x,N_y-3})$

Table 2.4: Entries of the sub-vectors $\mathbf{b}_2^{(j)}$ for zero boundary conditions.

	$\mathbf{b}_2^{(j)}$
b_1	$12f_{j,1} + f_{1,j-1} + f_{1,j+1} - 5f_{2,j} + 4f_{3,j} - f_{4,j}$
$b_2^{(i)}$	$8f_{i,j} + f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1}$
$\widehat{b_1}$	$12f_{N_x,j} + f_{N_x,j-1} + f_{N_x,j+1} - 5f_{N_x-1,j} + 4f_{N_x-2,j} - f_{N_x-3,j}$

apply, then the values of $\boldsymbol{u}$ on the boundary appear to the RHS of the linear system (2.51),

since these values are known.

2.2.3 Horizontal zebra coloring scheme

For the resulted linear system by the HOC scheme, the horizontal zebra numbering of equations and unknowns (see Fig. 2-5) is considered. The idea of reordering equations and unknowns is equivalent to the application of a similarity transformation to the linear system (2.51) via a permutation matrix $P \in \mathbb{R}^{N\times N}$. The linear system

$$PAP^T(P\boldsymbol{u}) \;= P\,\boldsymbol{b}. \tag{2.55}$$

has, now, the block structure

$$\left[\begin{array}{ccccccc|ccccccc}
A_1 & A_3 & O & \dots & O & O & O & A_2 & A_4 & O & \dots & O & O & O \\
O & A_6 & O & \dots & O & O & O & A_5 & A_5 & O & \dots & O & O & O \\
O & O & A_6 & \dots & O & O & O & O & A_5 & A_5 & \dots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \dots & A_6 & O & O & O & O & O & \dots & A_5 & O & O \\
O & O & O & \dots & O & A_6 & O & O & O & O & \dots & A_5 & A_5 & O \\
O & O & O & \dots & O & O & A_6 & O & O & O & \dots & O & A_5 & A_5 \\
\hline
A_5 & A_5 & O & \dots & O & O & O & A_6 & O & O & \dots & O & O & O \\
O & A_5 & A_5 & \dots & O & O & O & O & A_6 & O & \dots & O & O & O \\
O & O & A_5 & \dots & O & O & O & O & O & A_6 & \dots & O & O & O \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & \dots & A_5 & A_5 & O & O & O & O & \dots & A_6 & O & O \\
O & O & O & \dots & O & A_5 & A_5 & O & O & O & \dots & O & A_6 & O \\
O & O & O & \dots & O & \widehat{A}_4 & \widehat{A}_2 & O & O & O & \dots & O & \widehat{A}_3 & \widehat{A}_1
\end{array}\right]
\left[\begin{array}{c}
\mathbf{u}_1 \\ \mathbf{u}_3 \\ \mathbf{u}_5 \\ \vdots \\ \mathbf{u}_{L-5} \\ \mathbf{u}_{L-3} \\ \mathbf{u}_{L-1} \\ \mathbf{u}_2 \\ \mathbf{u}_4 \\ \mathbf{u}_6 \\ \vdots \\ \mathbf{u}_{K-4} \\ \mathbf{u}_{K-2} \\ \mathbf{u}_K
\end{array}\right]
=
\left[\begin{array}{c}
\mathbf{b}_1 \\ \mathbf{b}_3 \\ \mathbf{b}_5 \\ \vdots \\ \mathbf{b}_{L-5} \\ \mathbf{b}_{L-3} \\ \mathbf{b}_{L-1} \\ \mathbf{b}_2 \\ \mathbf{b}_4 \\ \mathbf{b}_6 \\ \vdots \\ \mathbf{b}_{K-4} \\ \mathbf{b}_{K-2} \\ \mathbf{b}_K
\end{array}\right] \tag{2.56}$$

The pair of parameters (L, K) in the linear system (2.56) is equal to (N_y, N_y) for N_y even and equals to $(N_y + 1, N_y - 1)$ for an odd value of N_y.

The above linear system can be partitioned as

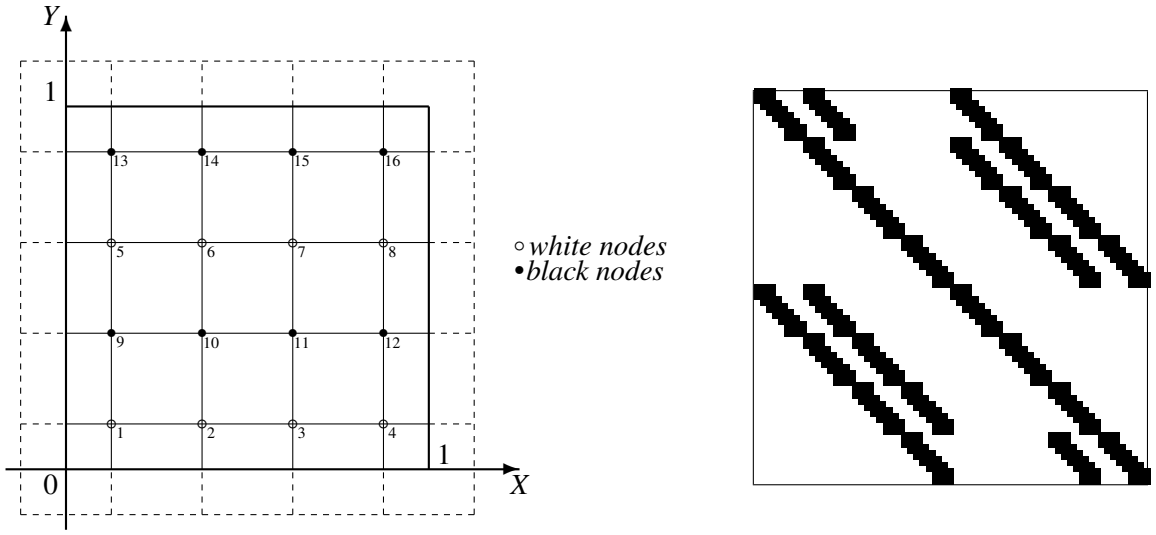

Figure 2-5: Zebra coloring scheme for unknowns and equations in case of $N_x = N_y = 4$ (left) and the structure of the relevant coefficient matrix A (right).

$$\left[\begin{array}{c|c} D_{RR} & H_{RB} \\ \hline H_{BR} & D_B \end{array}\right]\left[\begin{array}{c} \mathbf{u}_R \\ \hline \mathbf{u}_B \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_R \\ \hline \mathbf{b}_B \end{array}\right] \tag{2.57}$$

where

$$H_{BR} = \begin{bmatrix} A_5 & A_5 & O & \dots & O & O & O \\ O & A_5 & A_5 & \dots & O & O & O \\ O & O & A_5 & \dots & O & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ O & O & O & \dots & A_5 & A_5 & O \\ O & O & O & \dots & O & A_5 & A_5 \\ O & O & O & \dots & O & \widehat{A}_4 & \widehat{A}_2 \end{bmatrix}, \quad H_{RB} = \begin{bmatrix} A_2 & A_4 & O & \dots & O & O & O \\ A_5 & A_5 & O & \dots & O & O & O \\ O & A_5 & A_5 & \dots & O & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ O & O & O & \dots & A_5 & O & O \\ O & O & O & \dots & A_5 & A_5 & O \\ O & O & O & \dots & O & A_5 & A_5 \end{bmatrix}$$

and

$$D_R = \text{diag}[\tilde{A}_1 \underbrace{A_6 \cdots A_6}_{\frac{N_y}{2}-2}] \quad , \quad D_B = \text{diag}[\underbrace{A_6 \cdots A_6}_{\frac{N_y}{2}-2} \tilde{A}_2]$$

with

$$\tilde{A}_1 = \begin{bmatrix} A_1 & A_3 \\ O & A_6 \end{bmatrix} \quad \text{and} \quad \tilde{A}_2 = \begin{bmatrix} A_6 & O \\ \widehat{A}_3 & \widehat{A}_1 \end{bmatrix} .$$

It can be pointed out that the preconditioning coefficient matrix has a block 2-cyclic normal (or *red and black*) form with a highly parallelizable and of increased scalability structure [27, 101, 44]. In addition, PAP^T is block 2-cyclic consistently ordered [79].

2.2.4 Iterative methods

Dealing with realistic applications, the arising algebraic system (2.57) ends up to be very large, especially for fine discretizations problems. This fact suggests, that iterative methods [79, 44, 101] should be applied as an efficient solution method. For this reason, the following splitting of the coefficient matrix is considered

$$PAP^T = D_A - L_A - U_A \tag{2.58}$$

with

$$D_A = \begin{bmatrix} D_R & O \\ O & D_B \end{bmatrix} , \quad L_A = \begin{bmatrix} O & O \\ -H_{BR} & O \end{bmatrix} \text{ and } U_A = \begin{bmatrix} O & -H_{RB} \\ O & O \end{bmatrix} . \tag{2.59}$$

The above conformal partitioning allows the application of classical (stationary) and Krylov subspace (non-stationary) iterative numerical methods.

Another popular numerical method for the solution of two colored linear systems is based on the "Schur complement" form [101]. There, the original linear system is reordered into a 2×2 block system, as done in the linear system herein (2.57). Then, by eliminating the unknowns associated to the red grid points (Block-Gauss elimination procedure) the following form emerges

$$\begin{bmatrix} D_{RR} & H_{RB} \\ 0 & S \end{bmatrix} \begin{bmatrix} \mathbf{u}_R \\ \mathbf{u}_B \end{bmatrix} = \begin{bmatrix} \mathbf{b}_R \\ \widehat{\mathbf{b}}_B \end{bmatrix} \quad , \tag{2.60}$$

where S is the "Schur complement" matrix defined as

$$S = D_{BB} H_{BR} D_{RR}^{-1} H_{RB} \tag{2.61}$$

and $\widehat{\mathbf{b}}_B = b_B H_{BR} D_{RR}^{-1} \widehat{\mathbf{b}}_R$. Subsequently, the solution of the linear system (2.57) can be computed by first solving iteratively the reduced system (Schur complement system)

$$S\, u_B = \widehat{\mathbf{b}}_B \tag{2.62}$$

and in the sequel solving the sub-system

$$D_{RR} u_R = \mathbf{b}_R - H_{RB} \quad . \tag{2.63}$$

Altogether, the above procedure can be described by the following algorithm

Algorithm 1 Schur Complement algorithm for Red-Black linear systems

Step 1. *Solve* $D_W \hat{\boldsymbol{b}}_W = \boldsymbol{b}_W$

Step 2. *Evaluate* $\hat{\boldsymbol{b}}_B = \boldsymbol{b}_B - H_W \hat{\boldsymbol{b}}_W$

Step 3. *Solve iteratively* $S \boldsymbol{u}_B = \hat{\boldsymbol{b}}_B$ with $S = D_B - H_W D_W^{-1} H_B$

Step 4. *Evaluate* $\hat{\boldsymbol{u}}_B = H_B \boldsymbol{u}_B$

Step 5. *Solve* $D_W \hat{\boldsymbol{u}}_W = \hat{\boldsymbol{u}}_B$

Step 6. *Evaluate* $\boldsymbol{u}_W = \hat{\boldsymbol{b}}_W - \hat{\boldsymbol{u}}_W$

Algorithm's **Step 3** encapsulates an iterative method for the solution of the Schur complement system. This procedure is the most computationally intensive part of the algo-

rithm. The Schur complement method usually claims nearly half of the execution time to converge compared to other methods, and that's because in the iterative procedure the unknowns involved correlate to one color only. The Schur complement linear system solution is followed by the direct evaluation of the remaining unknowns. It is crucial to mention that, matrix S shall not be formed explicitly in order to solve the reduced system with an iterative method.

2.3 Applying MultiGrid techniques

The application of the fourth-order compact difference scheme results in a large and sparse linear system, ergo the use of an iterative solver in order to develop an efficient numerical solution is advisable. An incompressible Navier-Stokes solver based on a pressure correction method [71] needs at least one (but often up to ten) Poisson-type equations to be solved at each time step, so that the incompressibility condition is enforced. The computational cost when solving a single Poisson-type equation can be expressed in terms of the work unit (WU). Supposing that the approximation of the solution of an unsteady flow problem is needed, on a 2048×2048 grid, after T=100 seconds with a time step of $\Delta t = 10^{-4}$. In order to approximate the solution, at least 10^5 Poisson-type equations have to be solved. The above hypothesis has a computational cost of around 10^5 WUs.

The term FLOP is a prevalent term used for floating-point operation, for example, as a unit of counting floating-point operations (FLOPs) carried out by an algorithm or computer hardware. A processor's floating point operations per second (FLOAPS) can be estimated using the equation

$$\text{FLOPS} = \text{cores} \times \text{clock} \times \frac{\text{FLOPs}}{\text{cycle}}.$$

The majority of microprocessors up to this date, can perform 4 FLOPs per clock cycle; thus, the aforementioned processor has a theoretical performance of 40 billion FLOPS. A single sweep using the most uncomplicated iterative method, Jacobi, into a tridiagonal linear system requires about $4 * 2048^2 = 16.777.216$ operations and 0.42 seconds. Assuming that $1WU \equiv 0.42sec$, the overall computation time reaches a total of 4200 seconds.
It is apparent that the acceleration of the Poisson solver is a major factor in the numerical method's performance.

This limitation can be overcome by incorporating a geometric multigrid technique into the iterative procedure. Multigrid methods usually achieve high rates of convergence and are considered among the fastest methods used for solving large and sparse linear systems that result from the discretization of multi-dimensional boundary value problems. [1, 56,

75, 86, 92]. Furthermore, the convergence rate of the multigrid methods is independent of the grid size for specific elliptic BVP [1, 75].

A multigrid method solves the error correction sub-problem (coarse grid correction) on a coarse grid and then goes back to the fine grid, where it interpolates the error correction solution. Most of the time consuming computational operations are performed on the coarse grids, where the problem size is small, and as a result a considerable amount of computational time is conserved. The method consists of an iterative solver called smoother, which is a relaxation scheme for the error linear system, and two grid-transfer operators, the restriction for the mapping of the residual vectors from the fine Ω^h to the coarse Ω^H grid and the prolongation (interpolation) for the return of the amended error vectors back to the fine grid. The classical intergrid operations cannot be applied to cell-centered grids without proper modifications [15, 93, 75, 72] due to the fact that the coarse grid is not part of the finer grid after each grid-transferring procedure (see Figures 2-6 and 2-7). A variety of prolongation and restriction operators for cell-centered unequal discretizations are derived and presented herein.

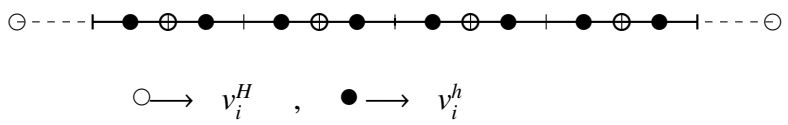

Figure 2-6: Fine and Coarse grid-function nodes for $N_x = 8$ and $H = 2h$; 1D domain.

As shown in subsection 1.7.2, the effect of one iteration of the two-grid method on the error can be expressed as the result of the application of two operators; the coarse grid correction operator K_h^H and the smoothing operator S_h,

$$N_h = S_h^{\nu_1} K_h^H S_h^{\nu_2} = S_h^{\nu_1}(I_h - P_H^h A_H^{-1} R_h^H A_h) S_h^{\nu_2} \tag{2.64}$$

to the error vector, where ν_1 and ν_2 indicate the number of pre- and pro-smoothing iterations; I_h is the identity operator of appropriate dimension; A_h and A_H are the discretization

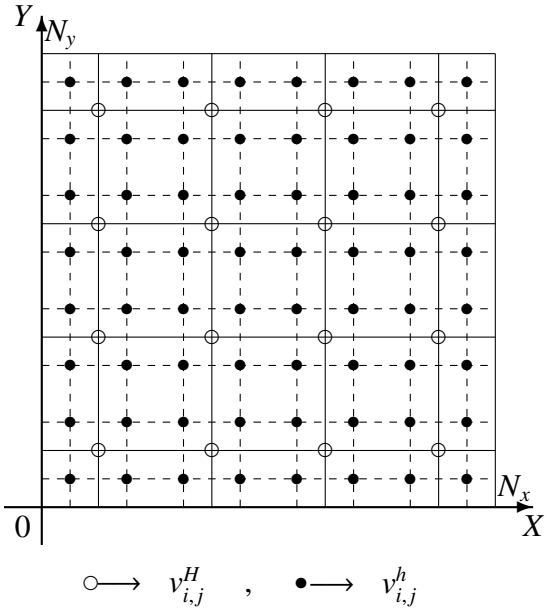

Figure 2-7: Fine and Coarse grid-function nodes for $N_x = N_y = 8$ and $H = 2h$; 2D domain

operators on Ω^h (fine grid operator) and Ω^H (coarse grid operator), respectively; and P^h_H and R^H_h are the transfer operators from the coarse to the fine grid and vice versa.

The fine grid operator A_h is the coefficient matrix A of the linear system for step-size h. The direct analog of matrix A_h on the coarse grid Ω_H is picked so as to determine the A_H in the two-grid method. Hence, the matrix A in the coarse grid is the discretization of the BVP on the corresponding coarse grid. The advantage of this approach lies on the fact that the structure of the A matrices remains unaltered whatever the grid, and as a result, the parallel properties of the algorithm are preserved.

2.3.1 Smoothing operator

The convergence behavior of a multigrid algorithm strongly depends on the error smoother having the smoothing property (see subsection 1.7.2). The decision on the smoothing method is determined by two parameters: the parallel properties and the efficiency of the smoother for both isotropic and anisotropic problems.

Instead of a lexicographic ordering, relaxation methods that use the red-black ordering schemes of grid nodes are well suited for parallel computations. In anisotropic problems,

line smoothers are often necessary, as they update all the unknowns in a line simultaneously when traversing the grid. Since anisotropy appears in the $x-$direction in our model problem for $\gamma < 1$, an $x-$line smoother is appropriate. In the following Chapter, smoothing analysis of a great number of methods using the Local Fourier Analysis (LFA), indicates that an acceptable compromise is reached when selecting the line zebra Gauss-Seidel as a smoother. In zebra sweeps the lines traversed in a red-black style, i.e first all odd, then all even lines. Due to the block structure of the matrix A the block version of the zebra Gauss-Seidel is preferred.

2.3.2 Coarsening strategies

The coarsening procedure defines the hierarchy of grids to be used within the multigrid method. Regular grids allow coarsening rules which are easy to use. Doubling the mesh size from G_h to G_H with $H = 2h$, is better known as standard coarsening. A basic non-uniform procedure doubles the mesh size only with respect to a subset of spatial directions. For instance, $x-$ and $y-$semicoarsening for two-dimensional applications yield $\boldsymbol{H} = (H1, H2) = (2h, h)$ and $\boldsymbol{H} = (h, 2h)$, respectively. Obviously, both restriction and prolongation shall to be selected in accordance with the coarsening strategy. Here, these multigrid components will be presented between two grids, Ω^h (fine) and Ω^H (coarse), under the consideration that the ratio of the coarse to the fine grid is two (standard coarsening).

2.3.3 Interpolation / Prolongation

In the ascent phase of a multigrid technique, e.g. V_{cycle} scheme, an interpolation formula is applied to correct the approximation vectors obtained on finer grids, with the estimated error vectors obtained on coarser grids. The interpolation operator will be expressed by P_H^h; using the coarse grid error vectors $\boldsymbol{v}_H$, it constructs the fine grid error vectors $\boldsymbol{v}_h$ based on the rule $\boldsymbol{v}_h = P_H^h \boldsymbol{v}_H$. As Fig. 2-6 depicts, the coarse grid points (v_i^H , $i = 1, \ldots, N_x/2$) do not coincide with the fine grid points (v_i^h , $i = 1, \ldots, N_x$). The points with indices $i = 0, N_x/2+1$

correspond to fictitious coarse values. To this effect, the coarse and fine grid nodes in 2D are defined, see Fig. 2-7.

In order to approximate the values of the function v on the fine grid, an interpolation formula based on the coarse grid values is being used.

Constant Prolongation

The simplest prolongation operator is derived by using the first-order formula based on Lagrange's interpolation polynomial. This operator is called constant prolongation (CP) because it interpolates precisely all the constant polynomials. The rule of thumb for its definition in 1D cases is that every fine grid value is equal to the nearest neighbor coarse value. This can be expressed in stencil notation as

$$\left] \begin{matrix} 1 & * & 1 \end{matrix} \right[, \qquad \left| \begin{matrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{matrix} \right|, \tag{2.65}$$

in one and two dimensions, respectively, where "*" marks the position of the coarse grid node. The notation indicates the weight whereby the value of the coarse grid function in the coarse grid node contributes to the fine grid function in its neighboring fine grid nodes. Although CP has a narrow stencil, i.e. a few operations are involved, it lacks in accuracy.

Bi-linear operator

Starting with the development of the 1D operator, the goal is to generalize to more dimensions.

1D operator The use of a more accurate second-order formula, based on Lagrange's interpolation polynomial, yields

$$\begin{cases} v^h_{2i} &= \frac{1}{4}(3v^H_i + v^H_{i+1}) \ ; \\ \\ v^h_{2i+1} &= \frac{1}{4}(v^H_i + 3v^H_{i+1}) \qquad 1 \le i \le N_x - 1. \end{cases} \tag{2.66}$$

which in stencil notation is simplified to

$$P^H_h = \frac{1}{4} \Big] \ 1 \quad 3 \quad * \quad 3 \quad 1 \ \Big[\quad . \tag{2.67}$$

The interpolation formula depends on the boundary conditions. Therefore, in case of zero Dirichlet boundary conditions $v^H_{-1/2} = v^H_{N_x/2+1/2} = 0$, inferring that the fictitious values are imposed numerically by the values $v^H_0 = -v^H_1$ and $v^H_{N_x/2+1} = -v^H_{N_x/2}$. In case of zero Neumann boundary conditions, $\frac{\partial v^H_{-1/2}}{\partial x} = \frac{\partial v^H_{N_x/2+1/2}}{\partial x} = 0$, inferring that the fictitious values are imposed numerically by the values $v^H_0 = v^H_1$ and $v^H_{N_x/2+1} = v^H_{N_x/2}$

The components of v^h, corresponding to points close to the boundary, are given by

$$v^h_1 = \tfrac{1}{2} v^H_1 \quad v^h_{N_x} = \tfrac{1}{2} v^H_{N_x/2} \quad , \tag{2.68}$$

$$v^h_1 = v^H_1 \quad v^h_{N_x} = v^H_{N_x/2} \quad , \tag{2.69}$$

in the case of Dirichlet and Neumann boundary conditions, respectively. Eventually, a general linear interpolation in the Dirichlet case is

$$P^H_h = \frac{1}{4} \Big] \ w \quad (2+w) \quad * \quad (2+e) \quad e \ \Big[, \tag{2.70}$$

whereas in the Neumann case

$$P_h^H = \frac{1}{4}\Big]\ w \quad (4-w) \quad * \quad (4-e) \quad e\ \Big[, \tag{2.71}$$

where w(west) and e(east) are equal to zero, when the corresponding coarse grid node lies outside the domain, otherwise both are equal to one. Due to this convenient form, the bi-linear prolongation can be directly implemented. Note that P_H^h is a linear operator from $\mathbb{R}^{N_x/2}$ to $\mathbb{R}^{N_x}$. In case of $N_x = 8$ and Dirichlet boundary conditions, this operator has the matrix form

$$P_h^H v_H \equiv P\boldsymbol{v}_H = \frac{1}{4}\begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_H = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{bmatrix}_h = \boldsymbol{v}_h \quad , \tag{2.72}$$

where P is a matrix of order $N_x/2 \times N_x$, describing the operator P_H^h. The above interpolation operator is $O(h^2)$ accurate because the interpolation is precise for all polynomials of first degree.

2D operator In two-dimensions, the bi-linear interpolation operator for cell-centered dis-

cetizations is given by

$$\begin{cases} v^h_{2i,2j} &= \frac{1}{16}(9v^H_{i,j} + 3v^H_{i+1,j} + 3v^{2h}_{i,j+1} + v^H_{i+1,j+1}) \ ; \\ \\ v^h_{2i+1,2j} &= \frac{1}{16}(3v^H_{i,j} + 9v^H_{i+1,j} + v^{2h}_{i,j+1} + 3v^H_{i+1,j+1}) \ ; \\ \\ v^h_{2i,2j+1} &= \frac{1}{16}(3v^H_{i,j} + v^H_{i+1,j} + 9v^{2h}_{i,j+1} + 3v^H_{i+1,j+1}) \ ; \\ \\ v^h_{2i+1,2j+1} &= \frac{1}{16}(v^H_{i,j} + 3v^H_{i+1,j} + 3v^{2h}_{i,j+1} + 9v^H_{i+1,j+1}), \quad 1 \le i, j \le \frac{n}{2} - 1. \end{cases} \tag{2.73}$$

The interpolation P^H_h is a 16-point prolongation and is expressed by the stencil

$$P^H_h = \frac{1}{16} \begin{bmatrix} 1 & 3 & & 3 & 1 \\ 3 & 9 & & 9 & 3 \\ & & * & & \\ 3 & 9 & & 9 & 3 \\ 1 & 3 & & 3 & 1 \end{bmatrix}, \tag{2.74}$$

In Dirichlet boundary conditions along the left boundary, $v^H_{-1/2,j} = 0$ is set for $j = 1, ..., Ny/2$, and it is inferred that the fictitious values $v_{0,j}$ are imposed numerically by the values $-v_{1,j}$. In Neumann boundary conditions along the left boundary, $\partial(v^H_{-1/2,j})/\partial x = 0$ is set, inferring that the fictitious values $v_{0,j}$ are imposed numerically by the values $v_{1,j}$.
Using this technique on the boundary, the components of v^h corresponding to points close to the corners are given by

$$v^h_{1,1} = \tfrac{1}{4}v^H_{1,1} \ ; \quad v^h_{1,N_y} = \tfrac{1}{4}v^H_{1,N_y/2} \ ; \quad v^h_{N_x,1} = \tfrac{1}{4}v^H_{N_x/2,1} \ ; \quad v^h_{N_x,N_y} = \tfrac{1}{4}v^H_{N_x/2,N_y/2} \ , \tag{2.75}$$

$$v^h_{1,1} = v^H_{1,1} \ ; \quad v^h_{1,N_y} = v^H_{1,N_y/2} \ ; \quad v^h_{N_x,1} = v^H_{N_x/2,1} \ ; \quad v^h_{N_x,N_y} = v^H_{N_x/2,N_y/2} \ , \tag{2.76}$$

in case of Dirichlet and Neumann boundary conditions, respectively. For the rest of the points close to boundary, e.g. close to the bottom boundary, it goes as

$$v^h_{2i,1} = \tfrac{1}{8}(3v^H_{i,1} + v^H_{i+1,1}) \; ; \quad v^h_{2i+1,1} = \tfrac{1}{8}(v^H_{i,1} + 3v^H_{i+1,1}) \quad for\, i = 1, \ldots, N_x/2 - 1 \quad , \tag{2.77}$$

$$v^h_{2i,1} = \tfrac{1}{4}(3v^H_{i,1} + v^H_{i+1,1}) \; ; \quad v^h_{2i+1,1} = \tfrac{1}{4}(v^H_{i,1} + 3v^H_{i+1,1}) \quad for\, i = 1, \ldots, N_x/2 - 1 \tag{2.78}$$

in Dirichlet and Neumann case, respectively. Following the above procedure, a generalization of the linear interpolation in (two or more) dimensions is rather intricate. A tractable technique that can be used to produce the linear interpolation in higher dimensions, including the stencil notation, is based on the tensor product.

The matrix describing the tensor product of two operators is the Kronecker product of the descriptive matrices. Therefore, the tensor product of the operators P^h_H with respect to the x- and y-direction constructs the same operator (bi-linear interpolation) in two dimensions. For instance, assuming $N_x = N_y = 4$ grid nodes in each direction, this operator has the matrix form

$$P_H^{h,x} \otimes P_H^{h,y} v_H = P^x \otimes P^y \boldsymbol{v}_H = \frac{1}{16} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 \\ 2 & 6 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 6 & 0 & 2 & 0 \\ 9 & 3 & 3 & 1 \\ 3 & 9 & 1 & 3 \\ 0 & 6 & 0 & 2 \\ 2 & 0 & 6 & 0 \\ 3 & 1 & 9 & 3 \\ 1 & 3 & 3 & 9 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 6 & 2 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_H = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \\ v_{11} \\ v_{12} \\ v_{13} \\ v_{14} \\ v_{15} \\ v_{16} \end{bmatrix}_h = \boldsymbol{v}_h \, . \tag{2.79}$$

Restating the stencil notations of (2.70) and (2.71) with respect to the y-direction

$$P_h^H = \frac{1}{4} \Big] \; n \quad (2+n) \quad (2+s) \quad s \; \Big[\, , \tag{2.80}$$

$$P_h^H = \frac{1}{4} \Big] \; w \quad (4-w) \quad (4-e) \quad e \; \Big[\, , \tag{2.81}$$

where n(north) and s(south) have the same role as w and e. Using again the tensor product in each direction and dropping the star $*$ for convenience purposes, the general bi-linear interpolation in the Dirichlet case emerges

$$P_H^{h,x} \otimes P_H^{h,y} = \frac{1}{16} \begin{bmatrix} nw & n(w+2) & n(e+2) & en \\ w(n+2) & (n+2)(w+2) & (e+2)(n+2) & e(n+2) \\ w(s+2) & (s+2)(w+2) & (e+2)(s+2) & e(s+2) \\ sw & s(w+2) & s(e+2) & es \end{bmatrix}, \tag{2.82}$$

and in the Neumann case

$$P_H^{h,x} \otimes P_H^{h,y} = \frac{1}{16} \begin{bmatrix} nw & -n(w-4) & -n(e-4) & en \\ -w(n-4) & (n-4)(w-4) & (e-4)(n-4) & -e(n-4) \\ -w(s-4) & (s-4)(w-4) & (e-4)(s-4) & -e(s-4) \\ sw & -s(w-4) & -s(e-4) & es \end{bmatrix}, \tag{2.83}$$

Other Prolongation operator types

The bi-linear interpolation (BP) is a well-known second order prolongation, having a compact 16-point stencil. Alternative second order prolongation operators employing sparser stencils are those of Wesseling/Khali (WP) and Kwak (KP) [62, 99].

WP: Based on the linear interpolation in triangles ABD and ACD (if A has indices (i, j) then a has $(2i, 2j)$)

$$\begin{aligned} v^h_{2i,2j} &= \tfrac{1}{4}(3v^H_{i,j} + v^H_{i+1,j+1}) \quad ; \quad v^h_{2i+1,2*j+1} = \tfrac{1}{4}(3v^H_{i,j} + v^H_{i+1,j+1}) \\ v^h_{2i+1,2j} &= \tfrac{1}{4}(2v^H_{i+1,j} + v^H_{i,j} + v^H_{i+1,j+1}) \quad ; \quad v^h_{2i,2*j+1} = \tfrac{1}{4}(v^H_{i,j} + 2v^H_{i,j+1} + v^H_{i+1,j+1}) \,, \end{aligned} \tag{2.84}$$

as illustrated in Fig. 2-9, the stencil by Wesseling and Khalil can be derived, see e.g [75], which in general (with the boundary closures) is given by

$$\begin{array}{llll} C & & & D \\ & c & d & \\ & a & b & \\ A & & & B \end{array}$$

Figure 2-8: Coarse cell centers A, B, C, D and fine cell centers a, b, c, d; 2D cell-centered grid point configuration.

$$\frac{1}{4}\begin{bmatrix} nw & nw & . & . \\ nw & 2+nw & 2 & . \\ . & 2 & 2+se & se \\ . & . & se & se \end{bmatrix}, \tag{2.85}$$

$$\frac{1}{4}\begin{bmatrix} nw & 2n-nw & . & . \\ 2w-nw & 6-2w-2n+nw & 2 & . \\ . & 2 & 6-2s-2e+se & 2e-se \\ . & . & 2s-se & se \end{bmatrix}, \tag{2.86}$$

for Dirichlet and Neumann boundary conditions, respectively.

KP: In the two-dimensional case, a linear interpolation for a fine grid node based on its nearest coarse grid neighbors leads to the Kwak's stencil [99]

$$\frac{1}{4}\begin{bmatrix} . & n & n & . \\ w & w+n & e+n & e \\ w & w+s & e+s & e \\ . & s & s & . \end{bmatrix}, \tag{2.87}$$

$$\frac{1}{4}\begin{bmatrix} . & n & n & . \\ w & 4-w-n & 4-e-n & e \\ w & 4-w-s & 4-e-s & e \\ . & s & s & . \end{bmatrix}, \tag{2.88}$$

in Dirichlet and Neumann boundary conditions respectively.

A new Prolongation operator

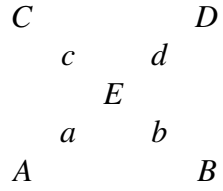

Figure 2-9: Coarse cell nodes A, B, C, D and fine cell nodes a, b, c, d. E are the central points of the squares ABDC and abcd; 2D cell-centered grid point configuration.

MP: Next, a new prolongation operator in 2D will be developed, based on WP. The $ABCD$ square is divided in four triangles ABE, ACE, CDE and BDE, and, then, using a linear interpolation in the direction of $\overrightarrow{k_1} = (1,1)$ and $\overrightarrow{k_2} = (1,-1)$ the following emerge

$$\frac{1}{4}\begin{bmatrix} 1 & . & . & 1 \\ . & 3 & 3 & . \\ & & * & \\ . & 3 & 3 & . \\ 1 & . & . & 1 \end{bmatrix}. \tag{2.89}$$

This operator is valid for interior domain discretization nodes and with its eight non-zero

entries competes against the 9-point-stencil of bi-linear interpolation in the vertex-centered case. It is also worthy of note, that the MP is of order two and, at the same time, preserves the narrow stencil. Discretization nodes near the boundary provide the following general stencil in the Dirichlet conditions case

$$\frac{1}{4}\begin{bmatrix} nw & nw-n & ne-n & ne \\ nw-w & 4-n-w+nw & 4-n-e+ne & ne-e \\ ws-w & 4-w-s+ws & 4-e-s+es & es-e \\ ws & ws-s & es-s & es \end{bmatrix}, \tag{2.90}$$

where w(west), e(east), s(south) and n(north) are equal to zero when the corresponding coarse grid node lies outside the domain, otherwise are equal to one. In the Neumann case the MP stencil pattern transpires as follows

$$\frac{1}{4}\begin{bmatrix} nw & n-nw & n-ne & ne \\ w-nw & 4-n-w+nw & 4-n-e+ne & e-ne \\ w-ws & 4-w-s+ws & 4-e-s+es & e-es \\ ws & s-ws & s-es & es \end{bmatrix}. \tag{2.91}$$

High-order interpolation

In the case of the Full Multigrid technique [1], a high-order interpolation operator is needed to interpolate the values of the problem's solution $\boldsymbol{u}$ to a finer grid. For this purpose, a bi-cubic interpolation operator is derived for cell-centered schemes.

1D operator Using the fourth-order formula based on the Lagrange's interpolation polynomial, the cubic interpolation for cell-centered grids is developed, at some interior points

$$\begin{cases} v_{2i}^h &= \frac{1}{128}(-7v_{i-1}^H + 105v_i^H + 35v_{i+1}^H - 5v_{i+2}^H) \\ v_{2i+1}^h &= \frac{1}{128}(-5v_{i-1}^H + 35v_i^H + 105v_{i+1}^H - 7v_{i+2}^H) \quad , \; 2 \le i \le N_x/2 - 2 \end{cases} \tag{2.92}$$

from which the following stencil comes up

$$P_H^h = \frac{1}{128} \Big] \; -5 \quad -7 \quad 35 \quad 105 \quad * \quad 105 \quad 35 \quad -7 \quad -5 \; \Big[\quad . \tag{2.93}$$

It should be noted that any cubic polynomial is interpolated by (2.92) exactly. Formula (2.92) cannot be applied if one (v_1^h and $v_{N_x}^h$) or two (v_2^h, v_3^h and $v_{N_x-1}^h$, $v_{N_x-2}^h$) neighboring values of v^H are not within the domain Ω (see Fig.2-6). In such case, it is more convenient to use the following quadratic formulae

$$\begin{aligned} v_1^h &= \tfrac{1}{24}(10 * v_L + 15v_1^H - v_2^H) \; ; \quad v_{N_x}^h = \tfrac{1}{24}(10 * v_R + 15v_{N_x/2}^H - v_{N_x/2-1}^H), \\ v_2^h &= \tfrac{1}{8}(-2 * v_L + 9v_1^H + v_2^H) \; ; \quad v_{N_x-1}^h = \tfrac{1}{8}(-2 * v_R + 9v_{N_x/2}^H + v_{N_x/2-1}^H) \; , \\ v_3^h &= \tfrac{1}{8}(-2 * v_L + 5v_1^H + 5v_2^H) \; ; \quad v_{N_x-2}^h = \tfrac{1}{8}(-2 * v_R + 5v_{N_x/2}^H + 5v_{N_x/2-1}^H), \end{aligned} \tag{2.94}$$

based on the Dirichlet conditions v_L (left boundary) and v_R (right boundary). The following equation sums up the cubic interpolation in a matrix form, in case of $N_x = 8$ and homogeneous Dirichlet conditions

$$P_H^h v^H = P_2 \boldsymbol{v}^H = \frac{1}{128} \begin{bmatrix} 80 & -16/3 & 0 & 0 \\ 144 & 16 & 0 & 0 \\ 80 & 80 & 0 & 0 \\ -7 & 105 & 35 & -5 \\ -5 & 35 & 105 & -7 \\ 0 & 0 & 80 & 80 \\ 0 & 0 & 16 & 144 \\ 0 & 0 & -16/3 & 80 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_H = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{bmatrix}_h = \boldsymbol{v}^h \,, \quad (2.95)$$

where P_2 is a matrix of order $N_x/2 \times N_x$, describing the operator P_H^h. For completeness' sake, the one-sided quadratic interpolation in the interior yields

$$\begin{cases} v_{2i}^h &= \frac{1}{32}(-3v_{i-1}^H + 30v_i^H + 5v_{i+1}^H) \\ v_{2i+1}^h &= \frac{1}{32}(5v_i^H + 30v_{i+1}^H - 3v_{i+2}^H) \quad , \; 2 \le i \le N_x/2 - 2 \end{cases} \quad (2.96)$$

or, alternatively,

$$\begin{cases} v_{2i}^h &= \frac{1}{32}(21v_i^H + 14v_{i+1}^H - 3v_{i+2}^H) \\ v_{2i+1}^h &= \frac{1}{32}(-3v_{i-1}^H + 14v_i^H + 21v_{i+1}^H) \quad , \; 2 \le i \le N_x/2 - 2 \end{cases} \quad (2.97)$$

which leads to a 6-point or an 8-point stencil, respectively.

The 2D operator

In order to generalize the cubic (or quadratic) interpolation in higher dimensions, the tensor product of the one-dimension operator in both x- and y-directions is used so that the bi-cubic (BCP) interpolation is generated. This operator in a matrix form entails the existence of a 8×8 grid, i.e. 64 fine grid values, which from the presentation point of view, is rather unattractive. Below, only the formula needed to manipulate some interior values ($2 \le i \le \frac{N_x}{2} - 2$ and $2 \le i \le \frac{N_x}{2} - 2$) is presented.

$$
\begin{aligned}
u^h_{2i,2j} \;=\; & \tfrac{1}{128^2}(49u^H_{i-1,j-1} - 735u^H_{i,j-1} - 235u^H_{i+1,j-1} + 35u^H_{i+2,j-1} - \\
& 735u^H_{i-1,j} + 11025u^H_{i,j} + 3675u^H_{i+1,j} - 525u^H_{i+2,j} - \\
& 245u^H_{i-1,j+1} + 3675u^H_{i,j+1} + 1225u^H_{i+1,j+1} - 175u^H_{i+2,j+1} + \\
& 35u^H_{i-1,j+2} - 525u^H_{i,j+2} - 175u^H_{i+1,j+2} + 25u^H_{i+2,j+2}
\end{aligned}
\qquad . \tag{2.98}
$$

Solution node values $u^h_{2i+1,2j}$, $u^h_{2i,2j+1}$, $u^h_{2i+1,2j+1}$ are estimated in a similar way and lead to the following stencil

$$
P^h_H = \frac{1}{128^2}\begin{bmatrix}
25 & 35 & -175 & -525 & -525 & -175 & 35 & 25 \\
35 & 49 & -245 & -735 & -735 & -245 & 49 & 35 \\
-175 & -245 & 1225 & 3675 & 3675 & 1225 & -245 & -175 \\
-525 & -735 & 3675 & 11025 & 11025 & 3675 & -735 & -525 \\
-525 & -735 & 3675 & 11025 & 11025 & 3675 & -735 & -525 \\
-175 & -245 & 1225 & 3675 & 3675 & 1225 & -245 & -175 \\
35 & 49 & -245 & -735 & -735 & -245 & 49 & 35 \\
25 & 35 & -175 & -525 & -525 & -175 & 35 & 25
\end{bmatrix} . \tag{2.99}
$$

Next, the two bi-quadratic (BQ6P and BQ8P) interpolation formulas for some interior nodes follow

$$\begin{aligned} u^h_{2i,2j} &= \tfrac{1}{32^2}(9u^H_{i-1,j-1} - 90u^H_{i,j-1} - 15u^H_{i+1,j-1} + \\ &\quad -90u^H_{i-1,j} + 900u^H_{i,j} + 150u^H_{i+1,j} - \\ &\quad 15u^H_{i-1,j+1} + 150u^H_{i,j+1} + 25u^H_{i+1,j+1} \end{aligned} \tag{2.100}$$

and

$$\begin{aligned} u^h_{2i,2j} &= \tfrac{1}{32^2}(441u^H_{i,j} + 294u^H_{i+1,j} - 63u^H_{i+2,j} + \\ &\quad 294u^H_{i,j+1} + 196u^H_{i+1,j+1} - 42u^H_{i+2,j+1} - \\ &\quad 63u^H_{i,j+2} - 42u^H_{i+1,j+2} + 9u^H_{i+2,j+2} \end{aligned} \tag{2.101}$$

corresponding to the interpolations (2.96) and (2.97), respectively.

The wide stencil of the aforementioned high-order operators is certainly a deterrent factor when applying them, not only in terms of how efficient the parallelism will be, but also regarding the computation involved. However, there is at least one good reason to argue on these high-order operators and that is the convergence rates that can be achieved when these operators join with the multigrid method, and, is the scope of next chapter.

2.3.4 Restriction

The restriction operator which is the reverse intergrid operator of prolongation, moves the residual vectors from a fine to a coarse grid and is defined as $R^H_h v^h = v^H$. Obviously, injection does not have a particular meaning in the case of a cell-centered discretization. The classical average operator (CR)

$$v_i^H = \frac{1}{2}(v_{2i-1}^h + v_{2i}^h) \tag{2.102}$$

$$v_{i,j}^H = \frac{1}{4}(v_{2i-1,2j-1}^h + v_{2i,2j-1}^h + v_{2i-1,2j}^h + v_{2i,2j}^h) \tag{2.103}$$

in 1D and 2D, respectively, can be applied in this grid type. The stencil notation is similar to CP

$$\frac{1}{2}\begin{bmatrix} 1 & * & 1 \end{bmatrix}, \qquad \frac{1}{4}\begin{bmatrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{bmatrix}. \tag{2.104}$$

It shall be noted that the combination of the CP and CR satisfies the *variational property*, i.e.

$$P_H^h = 2^d (R_h^H)^T \quad , \tag{2.105}$$

where d is the dimension space. Using the Eq. (2.105) a variety of restriction BR, WR, KR, MR, BQ6R, BQ8R and BCR operators for a cell-centered grid can be constructed, relevant to BP, WP, KP, MP, BQ6P, BQ8P and BCP prolongations, as described in the previous section. One can effortlessly obtain the restriction formulae from the (2.105), as well as the stencils of the corresponding prolongation operators.

The order of the operators has a significant role in the convergence analysis of the multigrid methods. It is rather known that transfer operators should fulfill the rule

$$m_p + m_r > M \quad , \tag{2.106}$$

where M denotes the higher order of the derivatives of the PDE, m_r and m_p the order of the

restriction and prolongation operators, respectively. The rule (2.106) is a necessary condition that multigrid should fulfill in order to prove that its convergence rates are independent of the mesh-size [41]. In our case, this rule is followed for every combination of prolongations and restrictions, apart from the pair (CP, CR). However, and at least in case of the cell-centered multigrid methods, this condition may be weakened as shown in [94, 43]

$$m_p^{high} + m_r^{high} \geq M \quad , \tag{2.107}$$

where m_p^{high} and m_r^{high} are the high-frequency order of the prolongation and restriction, respectively. Further details are provided in the next Chapter.

2.3.5 Non-equal mesh-size approach

In many CFD or practical applications, such as the temperature distribution in a thin rod, the Stokes oscillating plate and the Blasius flow over a flat plate, the distribution of the unknown physical quantity may vary in each direction. In fact, for zero pressure gradient boundary layer flows (Blasius flow), the solution along the x direction produces a self-similar velocity field. Therefore, there are certain cases in which the discretization of the rectangular domain Ω may have unequal mesh sizes in each coordinate direction. In such cases, the use of a semi-coarsening strategy, i.e. where the mesh coarsening procedure is only performed along the dominant spatial direction, is proved to be cost-effective. Nevertheless, grid stretching is a common practice in computational fluid dynamics, where high mesh resolution must be used near the wall or in regions with steep velocity gradients. The application of a multigrid technique with a partial semicoarsening strategy, see for example [49],[46], in flow problems is described in the following lines.

Under the assumption that the predominant direction is on the x-axis, only the dominant direction is coarsened till the mesh aspect ratio γ is equal to 1 (i.e. $\Delta x = \Delta y$). Starting with this grid size, a multigrid technique is conducted with full coarsening (mesh coarsening

performed in both directions). For the multigrid technique with partial semi-coarsening strategy, one way residual restriction ($\widetilde{R}_h^H$) and correction interpolation ($\widetilde{P}_H^h$) operators is applied, thus all the equivalent one dimensional prementioned operators can be applied. Additionally, once the grid is reduced to an equal meshsize in both directions, all the above-mentioned operators can be used.

2.3.6 Multigrid algorithm

The algorithm of an implemented multigrid cycle has a compact recursive definition (see Algorithm 2).

Algorithm 2 Multigrid V-cycle algorithm
$\boldsymbol{u}^{(i+1,k)} = MG\_Cycle(k, \boldsymbol{u}^{(i,k)}, A^{(k)}, \boldsymbol{b}^{(k)}, v_1, v_2)$

Presmoothing:
Step 1. $\boldsymbol{u}^{(i,k)} = Smooth^{v_1}(\boldsymbol{u}^{(i,k)}, A^{(k)}, \boldsymbol{b}^{(k)})$
Restriction:
if $\Delta x = \Delta y$ $(\gamma = 1)$ **then**
Step 2. $\boldsymbol{b}^{(k-1)} = R_h^H(\boldsymbol{b}^{(k)} - A^{(k)}\boldsymbol{u}^{(i,k)})$
elseif $\Delta x < \Delta y$ $(\gamma < 1)$ **then**
Step 3. $\boldsymbol{b}^{(k-1)} = \widetilde{R}_H^h(\boldsymbol{b}^{(k)} - A^k\boldsymbol{u}^{(i,k)})$
end
Recursion:
if $k = 1$ **then**
Step 4. use a fast iterative solver for $A^{(1)}\boldsymbol{u}^{(i,1)} = \boldsymbol{b}^{(1)}$
elseif $k > 1$ **then**
Step 5. performing $r(\geq 0)$ cycles $\boldsymbol{u}^{(i,k-1)} = MG\_Cycle(k-1, \boldsymbol{O}, A^{(k-1)}, \boldsymbol{b}^{(k-1)}, v_1, v_2)$
end
Interpolation:
if $\Delta x = \Delta y$ $(\gamma = 1)$ **then**
Step 6. $\boldsymbol{u}^{(i,k)} = \boldsymbol{u}^{(i,k)} + P_H^h\boldsymbol{u}^{(i,k-1)}$
elseif $\Delta x < \Delta y$ $(\gamma < 1)$ **then**
Step 7. $\boldsymbol{u}^{(i,k)} = \boldsymbol{u}^{(i,k)} + \widetilde{P}_H^h\boldsymbol{u}^{(k-1)}$
end
Postsmoothing:
Step 8. $\boldsymbol{u}^{(i+1,k)} = Smooth^{v_2}(\boldsymbol{u}^{(i,k)}, A^{(k)}, \boldsymbol{b}^{(k)})$

Algorithm 2 computes $\boldsymbol{u}^{(i+1,k)}$ in every new iteration, using a given approximation $\boldsymbol{u}^{(i,k)}$ until it reaches the solution $u^{(k)}$. The subscript k indicates the grid level, with $k = 1$ cor-

responding to the coarsest grid. In the description of the multigrid-cycle, performing v smoothing steps with an iterative method, e.g. Jacobi, Gauss-Seidel, applied to any discrete problem of the form $A^{(k)}\boldsymbol{u}^{(k)} = \boldsymbol{b}^{(k)}$ with an initial approximation $\boldsymbol{u}^{(k)}$, is denoted by the $Smooth^{v}$ procedure. The number of the pre- and post-smoothing iterations in the descent and ascent phase of the multigrid-cycle are denoted by v_1 and v_2, respectively. The parameter r specifies the number of cycles to be carried out on the current coarse grid level. In contrast to the interpolation in the multigrid-cycle, which is applied to corrections, the FMG uses a higher accuracy interpolation formula to transfer approximations of the solution to the fine grid. In each level k of the multigrid-cycle scheme, the matrices $A^{(k)}$ are defined by the discretization of the model problem, conditional to this grid size. The advantage of this approach is that this way, the parallel properties of the algorithm are preserved due to the the structure of the discretization matrices $A^{(k)}$.

Chapter 3

The Local Fourier Analysis

The Local Fourier convergence analysis is an interesting process that can be used for multigrid methods, in order efficient solvers for generalized problems to be developed [1]. This type of analysis is well-established in the case of vertex-centered case, e.g. in[1, 54, 75, 80], and for cell-centered discretizations, e.g. in [64].

This Chapter focuses on the smoothing procedure and the coarse-grid correction properties. Attention is drawn to the relaxation method used within the multigrid solver and, using the k-grid analysis (k=2,3), the intergrid transfer operators (P_h^H, R_H^h, etc.) are taken into account.

3.1 Basic principles

The discrete model problem is considered

$$L_h u_h(\boldsymbol{x}) = K_h f_h(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \Omega_h \quad . \tag{3.1}$$

The Local Fourier analysis (LFA) for the multigrid technique is based on the following simplifications. It is assumed that the coefficient operators L_h are constant and, at the same time, overlooks the boundary conditions, thus all occurring operators are extended to the

infinite grid G_h. Nevertheless, LFA is not restricted to such kind of problems only, since, under general assumptions, any nonlinear discrete operator with non-constant coefficients can be locally linearized and replaced (by freezing the coefficient) by an operator with constant coefficients.

The standard coarsening in the two dimensional case is considered, thus $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{h} = (h_1, h_2)$. The discrete operator L_h for a fixed point $\boldsymbol{x} \in \boldsymbol{G}_h$ can be expressed in a stencil notation as

$$L_h \hat{=} [\ell_{\boldsymbol{\kappa}}]_h \qquad (\boldsymbol{\kappa} = (\kappa_1, \kappa_2) \in \mathbb{Z}^2) \tag{3.2}$$

$$i.e. \quad L_h u_h(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in J} \ell_{\boldsymbol{\kappa}} u_h(\boldsymbol{x} + \boldsymbol{\kappa h}). \tag{3.3}$$

The Fourier nodes are considered to be given from

$$\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}_h) = e^{\mathrm{i}\boldsymbol{\theta x}/\boldsymbol{h}} = e^{\mathrm{i}\theta_1 x_1/h_1} e^{\mathrm{i}\theta_2 x_2/h_2} \tag{3.4}$$

where $\boldsymbol{\theta} \in \mathbb{R}^2$. As the above grid functions have a period range of 2π, it is sufficient to consider that $\boldsymbol{\theta} \in \Theta = (-\pi, \pi]^2$.

Lemma 3.1.1 *All grid functions $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}_h)$ are eigenfunctions of any constant coefficient infinite grid operator L_h.*

Proof: *Let the grid functions be $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}_h)$ and the discrete operator L_h, where $\boldsymbol{x} \in G_h$. Then*

$$\begin{aligned} L_h \varphi_h(\boldsymbol{x}) &= \sum_{\boldsymbol{\kappa} \in J} \ell_{\boldsymbol{\kappa}} \varphi_h(\boldsymbol{x} + \boldsymbol{\kappa} h) = \sum_{\boldsymbol{\kappa} \in J} \ell_{\boldsymbol{\kappa}} e^{\mathrm{i}\boldsymbol{\theta x} + \boldsymbol{\kappa h}/\boldsymbol{h}} = \sum_{\boldsymbol{\kappa} \in J} \ell_{\boldsymbol{\kappa}} e^{\mathrm{i}\boldsymbol{\theta\kappa}} e^{\mathrm{i}\boldsymbol{\theta x}/\boldsymbol{h}} \quad . \\ &= \tilde{L}_h(\boldsymbol{\theta}) \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}_h) \end{aligned}$$

The notation $\tilde{L}_h(\boldsymbol{\theta})$ refers to the eigenvalues or Fourier symbols of L_h. ■

3.1.1 Standard coarsening frequencies

If the standard coarsening is selected, i.e $\boldsymbol{H} = (2h_1, 2h_2)$, it's just the Fourier components

$$\varphi_h(\boldsymbol{\theta}, \cdot) \quad \text{with} \quad \boldsymbol{\theta} \in \Theta^{00} := (-\pi/2, \pi/2]^2$$

that are visible on the (infinite) coarse grid G_H. This leads to the following distinction between high- and low-frequency components on G_h, with respect to G_H. Each low-frequency $\boldsymbol{\theta} = \boldsymbol{\theta}^{00} \in \Theta^{00}$ is coupled with three high-frequencies marked with (•) in Fig. 3-1)

$$\boldsymbol{\theta}^{\alpha} := \theta^{00} - (\alpha_x sign(\theta_1), \alpha_y sign(\theta_2))\pi \quad \text{with} \quad \alpha = (\alpha_x, \alpha_y) \in \{(1,0),(0,1),(1,1)\}.$$

The corresponding spaces of $2h - harmonics$ are

$$\mathcal{E}_h^{\theta} := span\{\varphi_h(\boldsymbol{\theta}^{00}, \cdot), \varphi_h(\boldsymbol{\theta}^{11}, \cdot), \varphi_h(\boldsymbol{\theta}^{10}, \cdot), \varphi_h(\boldsymbol{\theta}^{01}, \cdot)\} \tag{3.5}$$

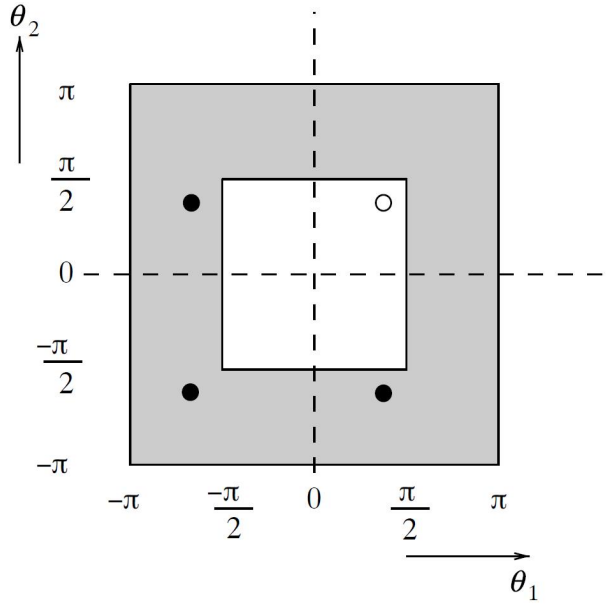

Figure 3-1: Low (white region) and high (shaded region) frequencies.

3.1.2 Smoothing factors

The smoothing behavior of the relaxation scheme can be analyzed using the Local Fourier Analysis.

Definition of smoothing factor

The assumption that a relaxation method of the discrete problem (3.1) on an infinite grid can be written as

$$L_h^+ u_h + L_h^0 u_h + L_h^- \bar{u}_h = K_h f_h \tag{3.6}$$

is taken, where u_h corresponds to the old and $\bar{u}_h$ to the new (after the relaxation step) approximation of the solution u. The L_h^0 refers to the grid points, where the unknowns are updated simultaneously, e.g in a line. The relaxation uses previous approximations at the grid points related to L_h^-, whereas at those related to L_h^+ all the updated values are already available.

Consequently, a damped relaxation method for the error is defined by

$$(L_h^0 + \omega L_h^+)e_h = ((1-\omega)L_h^0 - \omega L_h^-)\bar{e}_h$$

or

$$e_h = S_h \bar{e}_h,$$

where S_h is the resulting smoothing operator. Applying S_h to the Fourier modes $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$ leads to

$$S_h \varphi_h(\boldsymbol{\theta}, \boldsymbol{x}) = \tilde{S}_h(\boldsymbol{\theta}, \omega)\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}), \tag{3.7}$$

where the *amplification factor*

$$\tilde{S}_h(\boldsymbol{\theta}, \omega) := \frac{(1-\omega)L_h^0 - \omega L_h^-}{(L_h^0 + \omega L_h^+)}. \tag{3.8}$$

Moreover, in the case where $\omega = 1$,

$$\tilde{S}_h(\boldsymbol{\theta}) = -\frac{L_h^-}{(L_h^0 + L_h^+)}. \tag{3.9}$$

Then, the *smoothing factor* is defined as

$$\rho_1(\omega) := sup\{|\tilde{S}_h(\boldsymbol{\theta}, \boldsymbol{\omega})| : \boldsymbol{\theta} \in \Theta_{high}\}, \tag{3.10}$$

and represents the worst factor according to which the high frequencies of error reduce per relaxation step. The abovementioned factor could also be named as *one − grid factor* since it only takes into account fine-grid operators.

Now, the smoothing procedures of the Fourier components that are no longer eigenfunctions of the relaxation operators are considered [54] and, it is assumed that the relaxation method has the *invariance property*

$$S_h \ : \ \mathcal{E}_h^\theta \to \mathcal{E}_h^\theta \ \ \forall \boldsymbol{\theta} \in \Theta^{00}. \tag{3.11}$$

Then, using an *ideal* coarse grid operator (instead of the real coarse grid operator)

$$Q_h^H \varphi_h(\boldsymbol{\theta}, \cdot) = \begin{cases} 0 & \text{if } \ \boldsymbol{\theta} \in \Theta_{Low} \\ \\ \varphi_h(\boldsymbol{\theta}, \cdot) & \text{if } \ \boldsymbol{\theta} \in \Theta_{High} \end{cases}, \tag{3.12}$$

which annihilates the low-frequency error components and leaves the high-frequency components unchanged, the smoothing factor of S_h is defined as

$$\rho_1 := sup\{\sqrt[\nu]{\rho(\tilde{Q}_h^H \tilde{S}_h^\nu(\boldsymbol{\theta}))} : \boldsymbol{\theta} \in \Theta_{Low}\} \quad , \tag{3.13}$$

where $\tilde{Q}_h^H = \text{diag}(0, 1, 1, 1)$ (in standard coarsening) is the Fourier representation of the re-

lated ideal coarse-grid correction operator and $\nu = \nu_1 + \nu_2$ is the sum of pre- and postsmoothing steps. Pattern smoothers, like "red-black", "multicolor" and "zebra"-type relaxations, fulfill the invariance property [54, 92, 18, 42]. Thus, the smoothing properties of such coloring smoothers can be evaluated, using the definition (3.13).

3.1.3 H-ellipticity

In order to validate the existence of an efficient pointwise smoother for a certain operator L_h, one can use the "measure of h-ellipticity" defined by

$$E_h(L_h) := \frac{min\{|L_h(\boldsymbol{\theta})| \ : \ \boldsymbol{\theta} \in \Theta_{high}\}}{max\{|L_h(\boldsymbol{\theta})| \ : \ \boldsymbol{\theta} \in \Theta\}} = \frac{m}{M}. \tag{3.14}$$

with $0 \leq E_h \leq 1$. Note that a discretization operator is $h - elliptic$ if E_h is sufficiently far from zero.

Lemma 3.1.2 *If E_h is close to zero then any damped relaxation based on splitting $L_h = L_h^+ + L_h^0 + L_h^-$ has $\rho_1 \geq 1$.*

Proof: *Assuming that $E_h(L_h) = 0$, there is a high frequency $\boldsymbol{\theta}^*$ with $\widetilde{L}_h^-(\boldsymbol{\theta}^*) = -\widetilde{L}_h^+(\boldsymbol{\theta}^*) - \widetilde{L}_h^0(\boldsymbol{\theta}^*)$. Considering $\widetilde{L}_h^0(\boldsymbol{\theta}^*) + \omega\widetilde{L}_h^+(\boldsymbol{\theta}^*) \neq 0$, this yields that*

$$\rho_1(\omega) = \sup_{\boldsymbol{\theta}\in\Theta_{high}} \left| \frac{(1-\omega)\tilde{L}_h^0(\boldsymbol{\theta}) - \omega\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^0(\boldsymbol{\theta}) + \omega\tilde{L}_h^+(\boldsymbol{\theta})} \right| \geq \left| \frac{(1-\omega)\tilde{L}_h^0(\boldsymbol{\theta}^*) - \omega\tilde{L}_h^-(\boldsymbol{\theta}^*)}{\tilde{L}_h^0(\boldsymbol{\theta}^*) + \omega\tilde{L}_h^+(\boldsymbol{\theta}^*)} \right|$$

$$= \left| \frac{(1-\omega)\tilde{L}_h^0(\boldsymbol{\theta}^*) + \omega(\tilde{L}_h^+(\boldsymbol{\theta}^*) + \tilde{L}_h^0(\boldsymbol{\theta}^*))}{\tilde{L}_h^0(\boldsymbol{\theta}^*) + \omega\tilde{L}_h^+(\boldsymbol{\theta}^*)} \right| = 1$$

This result leads to the following theorem of existence of a point Smoother.

Theorem 3.1.1 (Existence of a point Smoother)

1. *If* $E_h(L_h) \geq c > 0$ *(for* $h \to 0$*) by some constant* $c > 0$*, there exists a pointwise smoothing procedure* S_h*, with* $\rho_1 < 1$*, that is bounded away from* 1 *by some constant that only depends on c.*

2. *Furthermore, if* L_h *is described by a symmetric stencil*

$$\ell_{i,j} = \ell_{-i,-j} \quad (i, j = 0, 1, 2, ..)$$

with $\ell_{0,0} > 0$ *(or* $\ell_{0,0} < 0$*) and if* $\tilde{L}_h(\boldsymbol{\theta}) > 0$ *(or* $\tilde{L}_h(\boldsymbol{\theta}) < 0$*) for* $\boldsymbol{\theta} \neq 0$*, then an optimal* $\omega - JAC$ *point smoother can be constructed with*

$$\omega_{opt} = \frac{2|\ell_{(0,0)}|}{m + M} \quad and \quad \rho_1(\omega_{opt}) = \frac{M - m}{M + m} = \frac{1 - E_h(l_h)}{1 + E_h(l_h)}. \tag{3.15}$$

For more details, the reader may see [2, 92].

3.2 The model problem

As shown in the previous Chapter, the discretization of the modified Helholmtz problem using fourth-order compact schemes leads to the discrete operator

$$L_h = \frac{1}{\Delta x^2}\begin{bmatrix} 1/2 & -1 & 1/2 \\ 5 & -10 & 5 \\ 1/2 & -1 & 1/2 \end{bmatrix} + \frac{\gamma^2}{\Delta x^2}\begin{bmatrix} 1/2 & 5 & 1/2 \\ -1 & -10 & -1 \\ 1/2 & 5 & 1/2 \end{bmatrix} - \lambda\begin{bmatrix} & 1/2 & \\ 1/2 & 4 & 1/2 \\ & 1/2 & \end{bmatrix}. \tag{3.16}$$

3.2.1 Fourier symbol of the model problem

The corresponding Fourier symbol is given by

$$\begin{aligned}\tilde{L}_h(\boldsymbol{\theta}) =& \frac{1}{\Delta x^2}[(10\cos(\theta_1) - 2\cos(\theta_2) + 2\cos(\theta_1)\cos(\theta_2) - 10) + \\ & \gamma^2(10\cos(\theta_2) - 2\cos(\theta_1) + 2\cos(\theta_1)\cos(\theta_2) - 10) - \lambda\Delta x^2(\cos(\theta_1) + \cos(\theta_2) - 4)]\end{aligned} \tag{3.17}$$

which is multiplied by Δx^2 and simplifies to the relation

$$\begin{aligned}\tilde{L}_h(\boldsymbol{\theta}) =& 2(\cos(\theta_2) + 5)(\cos(\theta_1) - 1) \\ & + \gamma^2 2(\cos(\theta_1) + 5)(\cos(\theta_2) - 1) \\ & - \lambda\Delta x^2(4 + \cos(\theta_1) + \cos(\theta_2))\end{aligned} \quad . \tag{3.18}$$

In the following investigation, it is assumed, without any loss of generality, that parameter's $\gamma = \dfrac{\Delta x}{\Delta y}$ upper bound is one.

3.2.2 Measurements of the h-ellipticity

As described in the previous Section, a discretization operator is $h - elliptic$ if E_h is sufficiently far from zero. In our case, the value of E_h depends on the following two parameters; (a) the anisotropy parameter γ and (b) the combination of the Helmholtz parameter λ and the discretization step size Δx.

Theorem 3.2.1 *The maximum value of the denominator of E_h is the maximum absolute value of the set*

$$\begin{aligned}\mathcal{A} =& \{\tilde{L}_h(0,\pi)\ ,\ \tilde{L}_h(\pi,0)\ ,\ \tilde{L}_h(\pi,\pi)\ ,\ \tilde{L}_h(0,0)\} \\ =& \{-4\lambda\Delta x^2 - 24\gamma^2, -4\lambda\Delta x^2 - 24, -2\lambda\Delta x^2 - 16\gamma^2 - 16, -6\Delta x^2\lambda\}\end{aligned}\ , \tag{3.19}$$

while the minimum value of the numerator is the minimum absolute value of the set

$$\begin{aligned}\mathcal{B} =&\{\tilde{L}_h(0,\pi)\ ,\ \tilde{L}_h(\pi,0)\ ,\ \tilde{L}_h(\pi,\pi)\ ,\ \tilde{L}_h(0,\pi/2)\ ,\ \tilde{L}_h(\pi/2,\pi/2)\}\\ &= \{-4\lambda\Delta x^2 - 24\gamma^2, -4\lambda\Delta x^2 - 24, -2\lambda\Delta x^2 - 16\gamma^2 - 16, -5\lambda\Delta x^2 - 12\gamma^2\\ &, -10 - 10\gamma^2 - 4\lambda\Delta x^2\}\end{aligned} \tag{3.20}$$

for the discrete operator (3.16).

Proof: *The proof in the Poisson case, where $\lambda = 0$, is provided.*

Due to symmetry, it stands that $\tilde{L}_h(-\theta_1,\theta_2) = \tilde{L}_h(\theta_1,-\theta_2) = \tilde{L}_h(-\theta_1,-\theta_2) = \tilde{L}_h(\theta_1,\theta_2)$, therefore, only the cases where $\theta_1 \geq 0$ and $\theta_2 \geq 0$ have to be considered. Since the function is continuous in Θ, the possible relative extrema can turn up on the boundary or at the critical points of the function $\tilde{L}_h(\boldsymbol{\theta})$ based on (3.18). It is considered known that the list of critical points may be found when solving the following system

$$\begin{cases}\dfrac{\partial\tilde{L}_h(\boldsymbol{\theta})}{\partial\theta_1} = \sin(\theta_1)(-10 + 2\gamma^2 - 2(1+\gamma^2)\cos(\theta_2) = 0\\ \dfrac{\partial\tilde{L}_h(\boldsymbol{\theta})}{\partial\theta_2} = \sin(\theta_2)(-10\gamma^2 + 2 - 2(1+\gamma^2)\cos(\theta_1) = 0\end{cases} \Leftrightarrow \begin{cases}\theta_1 = 0 \text{ or } \theta_1 = \pi\\ \\ \theta_2 = 0 \text{ or } \theta_2 = \pi\end{cases} . \tag{3.21}$$

Consequently, there are four critical points $(0,0),(0,\pi),(\pi,0),(0,\pi)$. The evaluation of the function at the critical points reveals that all the values in the extrema are non-negative. Hence, there are non-negative numbers only in the range of $\tilde{L}_h(\boldsymbol{\theta})$, indicating that the denominator of E_h is the absolute value of the minimum extrema of $\tilde{L}_h(\boldsymbol{\theta})$. In order to identify the numerator of E_h, the points in specific segments on the boundary of Θ_{high} are inspected, described by $\theta_1 = \pi/2,\ 0 \leq \theta_2 \leq \pi/2$ and $\theta_2 = \pi/2,\ 0 \leq \theta_1 \leq \pi/2$.

Case 1: $\theta_1 = \frac{\pi}{2}$

The function $\tilde{L}_h(\boldsymbol{\theta})$ simplifies to a function of a single real variable $\tilde{L}_h(\boldsymbol{\theta_2})$=$-2(\cos(\theta_2 + 5) + 10\gamma^2(\cos(\theta_2 - 1)$. Following the differentiation of the function $\tilde{L}_h$, the function is strictly increasing, hence the maximum (which is the minimum of the $|\tilde{L}_h|$) turns up at $\theta_2 = \pi/2$,

and thus $(\pi/2, \pi/2)$.

Case 2: $\theta_2 = \frac{\pi}{2}$

Similarly, $\tilde{L}_h$ *is a strictly decreasing function leading to a maximum extrema in* $\theta_1 = 0$ *,and thus* $(0, \pi/2)$. ■

Likewise, the sets $\mathcal{A}$ and $\mathcal{B}$ can be obtained for an arbitrary parameter λ.

Situation 1: Isotropy($\gamma = 1$)

If γ equals to 1, then the sets $\mathcal{A}$ and $\mathcal{B}$ consist of three and four individual elements respectively, instead of four and five as happens in the general case, due to the equal meshsizes in x- and y- direction (isotropic case). Hence,

$$\mathcal{A} = \{-4\varepsilon - 24, -2\varepsilon - 32, -6\varepsilon\} \tag{3.22}$$

and

$$\mathcal{B} = \{-4\varepsilon - 24, -2\varepsilon - 32, -5\varepsilon - 12, -4\varepsilon - 20\} \tag{3.23}$$

where $\varepsilon = \lambda \Delta x^2$. It is not possible to provide a global expression for m and M, in the whole range of ε. However, one can obtain that

$$m = \begin{cases} 5\varepsilon + 12 & \text{, if } \varepsilon \in [0, \frac{20}{3}] \\ 2\varepsilon + 32 & \text{, if } \varepsilon \in [\frac{20}{3}, \infty) \end{cases} \tag{3.24}$$

and

$$M = \begin{cases} 2\varepsilon + 32 & , \text{if } \varepsilon \in [0, 4] \\ 4\varepsilon + 24 & , \text{if } \varepsilon \in (4, 12] \\ 6\varepsilon & , \text{if } \varepsilon \in (12, \infty) \end{cases} . \tag{3.25}$$

As $E_h = \dfrac{5\varepsilon + 12}{2\varepsilon + 32}$ when $\varepsilon \to 0$, $E_h = \dfrac{3}{8}$ when $\varepsilon = 0$. On the other hand, if $\varepsilon \to \infty$ then $E_h \to \dfrac{1}{3}$. The maximum value $E_h = 17/19$ emerges when $\varepsilon = 20/3$. Hence, the following range of values for $h - ellipticity$ is obtained

$$E_h(L_h) \in \left(\frac{1}{3}, \frac{17}{19}\right]. \tag{3.26}$$

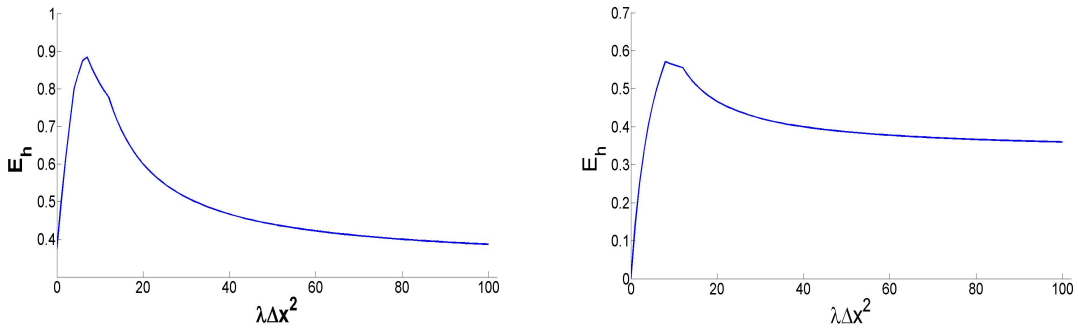

Figure 3-2: The measure of "h-ellipticity" for the isotropic case (at left) and for the fully anisotropic case (at right).

Situation 2: Anisotropy ($\gamma \in [0, 1)$)

Following the same path as in the isotropic case and assuming, for simplicity sake, that $\gamma \to 0$, values m and M are given by

$$m = \begin{cases} 4\varepsilon & , \text{if } \varepsilon \in [0, 8] \\ 2\varepsilon + 16 & , \text{if } \varepsilon \in [8, \infty) \end{cases} \tag{3.27}$$

and

$$M = \begin{cases} 4\varepsilon + 24 & , \text{if } \varepsilon \in [0, 12] \\ 6\varepsilon & , \text{if } \varepsilon \in (12, \infty) \end{cases}, \tag{3.28}$$

and result to

$$E_h(L_h) \in \left[0, \frac{4}{7}\right). \tag{3.29}$$

The maximum value $E_h = 4/7$ is obtained for $\varepsilon = 8$. Figure 3-2 depicts the behavior of E_h with respect to ε in both isotropic (left) and anisotropic (right) cases.

A flawed value of E_h is obtained when $\varepsilon = 0$, indicating the failure of pointwise smoothers. As defined in the first Statement of Theorem 3.1.1, it is unattainable to construct efficient pointwise smoothers for highly anisotropic problems. In such case, linewise smoothers instead of pointwise could be considered.

Next, the performance of smoothing factors for a variety of classical and pattern relaxation methods is investigated.

3.2.3 Jacobi relaxation scheme

The splitting of the model's problem operator L_h in the case of the Jacobi relaxation (JAC), is given by

$$L_h^+ = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad L_h^0 = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -a & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad L_h^- = \frac{1}{\Delta x^2}\begin{bmatrix} d & c & d \\ b & 0 & b \\ d & c & d \end{bmatrix}. \tag{3.30}$$

According to (3.9), the Fourier symbol $\tilde{S}_h(\boldsymbol{\theta}, \omega)$ of the weighted Jacobi ($\omega - JAC$) is

$$\tilde{S}_h(\boldsymbol{\theta}, \omega) = 1 - \omega\left(1 + \frac{\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^0(\boldsymbol{\theta})}\right) = 1 - \omega\frac{\tilde{L}_h(\boldsymbol{\theta})}{\tilde{L}_h^0(\boldsymbol{\theta})}, \tag{3.31}$$

since $\tilde{L}_h^+(\boldsymbol{\theta})$ equals zero. The smoothing factor for the undamped JAC point relaxation, applied to the isotropic Poisson equation is given by

$$\rho_1 = sup\left\{\left|\frac{\cos(\theta_1)\cos(\theta_2) + 2\cos(\theta_1) + 2\cos(\theta_2)}{5}\right| : \boldsymbol{\theta} \in \Theta_{high}\right\}. \tag{3.32}$$

The supremum of $\tilde{S}_h(\boldsymbol{\theta}, 1)$ is attained at $\theta = (\pm\pi, \pm\pi)$ yielding $\rho_1 = \frac{3}{5}$. It is noteworthy, that the second-order accurate scheme (2.3) in the Poisson problem leads to the same smoothing factor as in the case of the optimal $\omega - JAC$ ($\omega = 4/5$) [92]. Furthermore, for a low-order scheme the undamped JAC has no smoothing properties ($\rho = 1$), making the high-order scheme more enticing.

In most cases, an optimal point $\omega - JAC$ (see Fig. 3-9) can be traced as the discrete operator L_h fulfills the conditions of the Theorem 3.1.1. Theorem's first Statement is satisfied, unless both γ and ε equal *zero*, case that leads to $E_h = 0$. Furthermore, $l_{0,0} = -20(1+\gamma^2) - 4\lambda\Delta x^2$ is less equal than -20 and $l_{i,j} = l_{-i,-j}\ for\ i, j = 0, 1$, thus the second Statement is valid.

The smoothing factors for the optimal parameters of $\omega - JAC$ are presented in Fig. 3-9, with respect to ε in both isotropic (left) and anisotropic (right) cases. In the special case of $\lambda = 0$, a well-known optimal value of $\omega_{opt} = 10/11$ [1] can be obtained by

$$\omega_{opt} = \frac{2|l_{(0,0)}|}{m + M}. \tag{3.33}$$

for $l_{(0,0)} = -20$, $m = 12$ and $M = 32$, resulting that $\rho_1(\omega_{opt}) = 5/11$.

3.2.4 Lexicographic Gauss-Seidel relaxation schemes

Considering the splitting of L_h in the operator

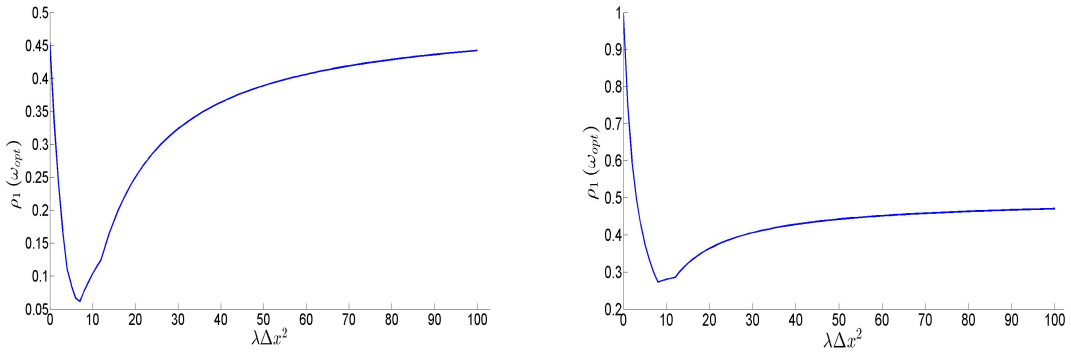

Figure 3-3: Smoothing factors corresponding to optimal parameters; isotropic case (at left) and fully anisotropic case (at right).

13	14	15	16	4	4	4	4
9	10	11	12	3	3	3	3
5	6	7	8	2	2	2	2
1	2	3	4	1	1	1	1
	(a)				(b)		

Figure 3-4: Grid point stencil (a) Lexicographic; (b) $x-$line ordering

$$L_h^+ = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ b & 0 & 0 \\ d & c & d \end{bmatrix} \quad L_h^0 = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -a & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad L_h^- = \frac{1}{\Delta x^2}\begin{bmatrix} d & c & d \\ 0 & 0 & b \\ 0 & 0 & 0 \end{bmatrix} \tag{3.34}$$

and the row-wise ordering scheme of the grid points, starting with left lower corner points (see Figure 3-4(a)) the Fourier symbol of the Gauss-Seidel ($GS - LEX$) is given by

$$\tilde{S}_h(\boldsymbol{\theta}) = \frac{2d\cos(\theta_1)e^{\mathrm{i}\theta_2} + be^{\mathrm{i}\theta_1} + ce^{\mathrm{i}\theta_2}}{\alpha - (2d\cos(\theta_1)e^{-\mathrm{i}\theta_2} + be^{-\mathrm{i}\theta_1} + ce^{-\mathrm{i}\theta_2})}. \tag{3.35}$$

For $\varepsilon = 0$ and $\gamma = 1$ eventuates that

$$\tilde{S}_h(\boldsymbol{\theta}) = \frac{2\cos(\theta_1)e^{i\theta_2} + 4e^{i\theta_1} + 4e^{i\theta_2}}{20 - (2\cos(\theta_1)e^{-i\theta_2} + 4e^{-i\theta_1} + 4e^{-i\theta_2})} = \frac{A}{20 - \bar{A}}, \tag{3.36}$$

where $A = 2\cos(\theta_1)e^{i\theta_2} + be^{i\theta_1} + ce^{i\theta_2}$ and $\bar{A}$ is the conjugate of A. Obviously,

$$A = 2\cos(\theta_1)\cos(\theta_2)+4\cos(\theta_1)+4\cos(\theta_2)+i(2\cos(\theta_1)\sin(\theta_2)+4\sin(\theta_1)+4\sin(\theta_2)). \tag{3.37}$$

Fig. 3-5 depicts the behavior of the smoothing factors in the case of $GS - LEX$ with the symbol (3.36), with respect to ε in both isotropic (left) and anisotropic (right) cases. Fig. 3-5 untangles that the smoothing factor (ρ_1) is bounded by $\frac{1}{2}$ for isotropic problems.

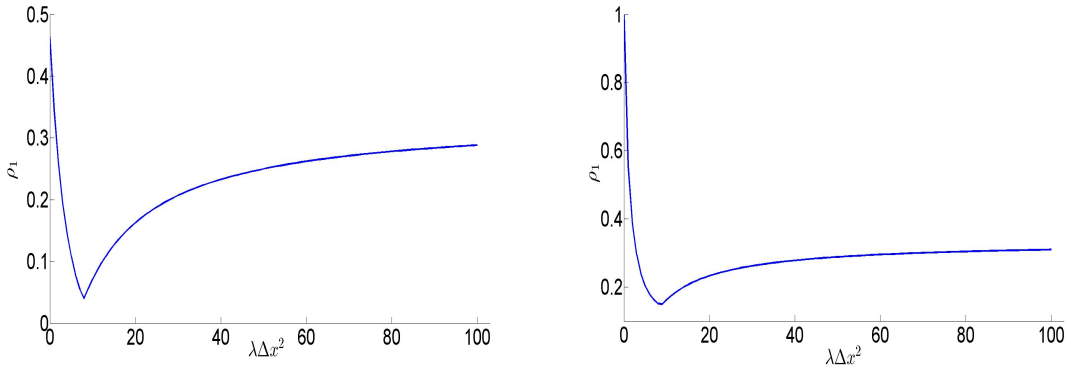

Figure 3-5: Smoothing factors corresponding to optimal parameters; isotropic case (at left) and fully anisotropic case (at right).

Next, $GS - LEX$ with a relaxation parameter ω (SOR method) is applied and the $\rho_1(\omega)$ with respect to ω is tracked. Using $\omega \neq 0$, SOR hardly improves its smoothing property for the isotropic Poisson case problem (see Figure 3-6). The optimal value of ω is excessively close to 1. Additionally, it is clear that this outcome is true regardless of the value of parameter ε.

Remark 3.2.1 *The influence of the parameter ω on the smoothing properties of $\omega - GS - LEX$ is similar to that of those factors obtained from a five-point stencil for Poisson equation (see Figure 4.2 in [92]).*

Moreover, point Gauss-Seidel is not functional in case of very small γ values as ex-

pected, therefore $GS - LEX$ is not considered robust for anisotropic problems.

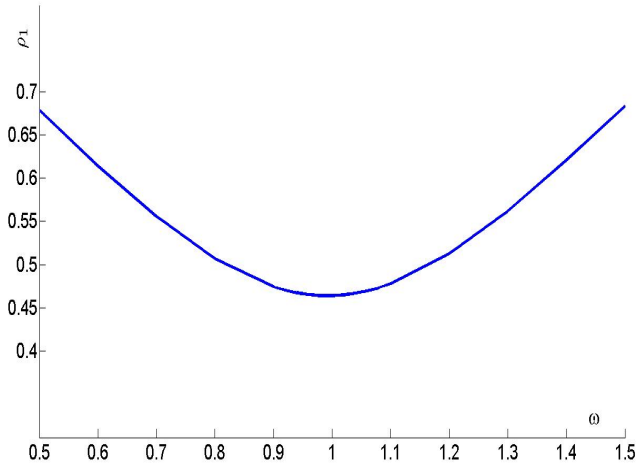

Figure 3-6: Smoothing factors (ρ_1) as a function of ω for the Poisson problem; isotropic case

3.2.5 Line Gauss-Seidel relaxation schemes

Horizontal line Gauss-Seidel iteration method (based on the $x-line$ ordering scheme of grid points), see Figure 3-4(b), applied to the model problem discretization grid, corresponds to the splitting

$$L_h^+ = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ d & c & d \end{bmatrix}, \; L_h^0 = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ b & -a & b \\ 0 & 0 & 0 \end{bmatrix}, \; L_h^- = \frac{1}{\Delta x^2}\begin{bmatrix} d & c & d \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.38}$$

The amplification factor is estimated as

$$\tilde{S}_h(\boldsymbol{\theta}) = \frac{ce^{\mathrm{i}\theta_2} + 2d\cos(\theta_1)e^{\mathrm{i}\theta_2}}{a - (ce^{-\mathrm{i}\theta_2} + 2d\cos(\theta_1)e^{-\mathrm{i}\theta_2})}. \tag{3.39}$$

Fig. 3-7 shows the smoothing factors of the line $GS - LEX$ in the isotropic case, with respect to ε (left). As the anisotropy is increased, the right graph in Fig. 3-7) depicts the

behavior of the line-wise smoother. It is important to mention that the smoothing factors are bounded regardless the values of parameters γ and ε. It can be also found that these values are bounded by $1/sqrt(5) \approx 0.4473$ and, especially in the fully-anisotropic case, ρ_1 is approximately equal to 0.11. In these conditions, the line $GS - LEX$ is considered a robust relaxation scheme for our model problem, leading to the general rule: *points that are strongly coupled*, i.e. points with large coefficients in L_h, must be updated simultaneously.

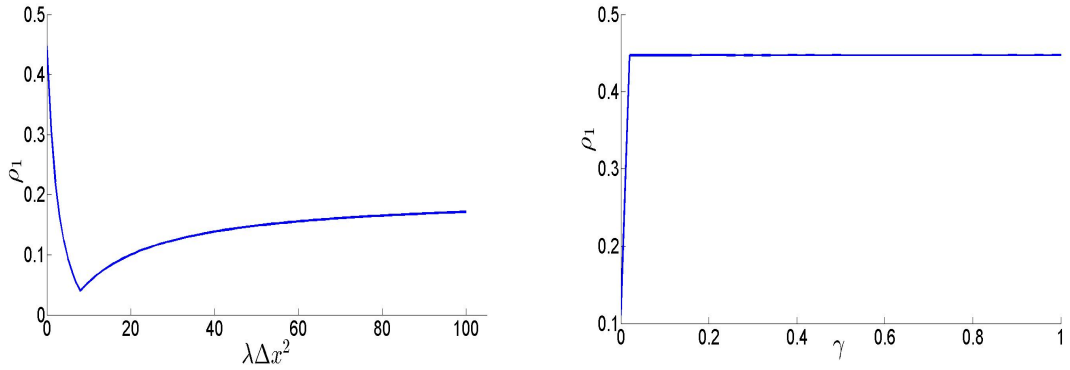

Figure 3-7: Smoothing factors for line $GS - LEX$; isotropic case (at left) and increasing anisotropy (at right).

Remark 3.2.2 ($GS - LEX$ versus line $GS - LEX$ relaxation) *An advantage when using line $GS - LEX$ is that its smoothing factors are similar to those of $GS - LEX$ in the isotropic case while, at the same time, it can effectively handle anisotropic problems.*

3.2.6 Relaxations patterns

The Fourier analysis of point and line red-black relaxations, as illustrated in Figure 3-8), requires special treatments because pattern smoothers consist of two or more partial relaxation steps. The partial relaxation operator is defined as

$$S_h^{partial}\varphi_h(\boldsymbol{\theta}, \boldsymbol{x}_h) = \begin{cases} S_h\varphi(\boldsymbol{\theta}, \boldsymbol{x}_h) & \text{for } \boldsymbol{x_h} \in \tilde{\boldsymbol{G}}_h \\ \varphi(\boldsymbol{\theta}, \boldsymbol{x}) & \text{for } \boldsymbol{x} \in \boldsymbol{G}_h \, \tilde{\boldsymbol{G}}_h \end{cases}, \tag{3.40}$$

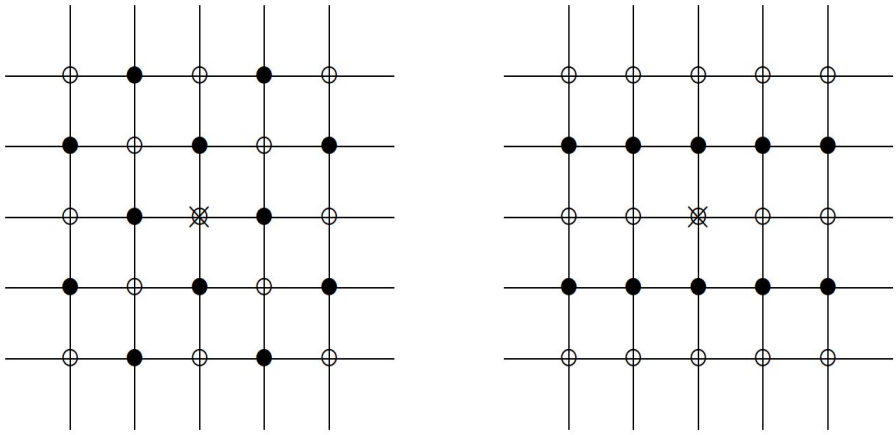

Figure 3-8: Red-black point coloring (left) and red-black horizontal line coloring (right); Red (∘), black (•) points and × denotes the origin of G_h.

where $\tilde{\boldsymbol{G}}_h$ is a subset of $\boldsymbol{G}_h$, defined by a pattern that describes the smoothing procedure. It is rather clear that only the grid points of $\tilde{\boldsymbol{G}}_h$ are processed in the above partial relaxation step, while the remaining points ($\boldsymbol{G}_h \backslash \tilde{\boldsymbol{G}}_h$) remain intact.

Point red-black Jacobi/Gauss-Seidel relaxation

Point red-black Jacobi($JAC-RB$) or Gauss-Seidel($GS-RB$) consist of two partial smoothing steps. In the first step, $\tilde{\mathbf{G}}_h$ handles the "red" points, while the second only involves the "black" points, i.e.

$$\mathbf{G}_h^R = \{\mathbf{x} = (k_1\Delta x, k_2\Delta y) \in \mathbf{G}_h | k_1 + k_2 \;\; even\}, \quad \mathbf{G}_h^B = \{\mathbf{x} = (k_1\Delta x, k_2\Delta y) \in \mathbf{G}_h | k_1 + k_2 \;\; odd\}. \tag{3.41}$$

The complete $JAC - RB$ or $GS - RB$ iteration operator is given by

$$S_h^{RB} = S_h^{BLACK} S_h^{RED} \tag{3.42}$$

with

$$S_h^R\varphi_h(\boldsymbol{\theta},\boldsymbol{x}_h)=\begin{cases}A^\alpha\varphi(\boldsymbol{\theta},\boldsymbol{x}_h) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^R\\ \varphi(\boldsymbol{\theta},\boldsymbol{x}) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^B\end{cases},\ S_h^B\varphi_h(\boldsymbol{\theta},\boldsymbol{x}_h)=\begin{cases}A^\alpha\varphi(\boldsymbol{\theta},\boldsymbol{x}_h) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^B\\ \varphi(\boldsymbol{\theta},\boldsymbol{x}) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^R\end{cases}, \tag{3.43}$$

where

$$A^\alpha=\frac{2b\cos(\theta_1)+2c\cos(\theta_1)+4d\cos(\theta_1)\cos(\theta_2)}{a} \tag{3.44}$$

$$A^\alpha=\frac{2b\cos(\theta_1)+2c\cos(\theta_1)+2d\cos(\theta_1)e^{\mathrm{i}\theta_2}}{a-2d\cos(\theta_1)e^{-\mathrm{i}\theta_2}} \tag{3.45}$$

and $A^\alpha := A(\boldsymbol{\theta}^\alpha)$ for the $JAC-RB$ and $GS-RB$, respectively. The above partial smoothing operators can be easily written as a linear combination of $\varphi_h(\boldsymbol{\theta}^\alpha,\boldsymbol{x}_h)$ and $\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\boldsymbol{x}_h)$, where $\tilde{\boldsymbol{\alpha}}=(1,1)-\boldsymbol{a}$. Hence

$$\begin{aligned}S_h^R\varphi_h(\boldsymbol{\theta}^\alpha,\boldsymbol{x}_h)&=\tfrac{1}{2}(1+A^\alpha)\varphi_h(\boldsymbol{\theta}^\alpha,\boldsymbol{x}_h)+\tfrac{1}{2}(A^\alpha-1)\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\boldsymbol{x}_h)\\ S_h^B\varphi_h(\boldsymbol{\theta}^{\boldsymbol{a}},\boldsymbol{x}_h)&=\tfrac{1}{2}(1+A^\alpha)\varphi_h(\boldsymbol{\theta}^\alpha,\boldsymbol{x}_h)+\tfrac{1}{2}(-A^\alpha+1)\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\boldsymbol{x}_h)\end{aligned} \tag{3.46}$$

leading to the following (4x4)-matrix representations for S_h^R and S_h^B

$$\tilde{S}_h^R=\frac{1}{2}\begin{bmatrix}A^{(0,0)}+1 & A^{(1,1)}-1 & 0 & 0\\ A^{(0,0)}-1 & A^{(1,1)}+1 & 0 & 0\\ 0 & 0 & A^{(1,0)}+1 & A^{(0,1)}-1\\ 0 & 0 & A^{(1,0)}-1 & A^{(0,1)}+1\end{bmatrix}_h \quad \textit{and} \tag{3.47}$$

$$\tilde{S}_h^B = \frac{1}{2}\begin{bmatrix} A^{(0,0)}+1 & -A^{(1,1)}-1 & 0 & 0 \\ -A^{(0,0)}-1 & A^{(1,1)}+1 & 0 & 0 \\ 0 & 0 & A^{(1,0)}+1 & -A^{(0,1)}-1 \\ 0 & 0 & -A^{(1,0)}-1 & A^{(0,1)}+1 \end{bmatrix}_h . \tag{3.48}$$

The combination of the two steps provide the Fourier representation of the global smoothing operator.

Horizontal Zebra Gauss-Seidel relaxation

In the two dimensional case discretizations, Zebra Gauss-Seidel ($GS - Zebra$) coincides with the Zebra Jacobi relaxation ($JAC - Zebra$) for 9−point compact schemes. However, this relation is no longer valid for discrete operators based on schemes with wider "stencils". Horizontal $GS - Zebra$ relaxation is described by

$$\mathbf{G}_h^R = \{\mathbf{x} = (k_1\Delta x, k_2\Delta y) \in \mathbf{G}_h |\ \ k_2 \ \ even\}, \quad \mathbf{G}_h^B = \{\mathbf{x} = (k_1\Delta x, k_2\Delta y) \in \mathbf{G}_h |\ \ k_2 \ \ odd\}, \tag{3.49}$$

and the following splitting of the L_h within each half-sweep:

$$L_h^+ = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ b & 0 & b \\ 0 & 0 & 0 \end{bmatrix} L_h^0 = \frac{1}{\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -a & 0 \\ 0 & 0 & 0 \end{bmatrix} L_h^- = \frac{1}{\Delta x^2}\begin{bmatrix} d & c & d \\ 0 & 0 & 0 \\ d & c & d \end{bmatrix}. \tag{3.50}$$

Again, the complete $GS - Zebra$ iteration operator is given by

$$S_h^{RB} = S_h^{BLACK} S_h^{RED} \tag{3.51}$$

with

$$S_h^R\varphi_h(\boldsymbol{\theta},\boldsymbol{x}_h)=\begin{cases}A^{\alpha}\varphi(\boldsymbol{\theta},\boldsymbol{x}_h) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^R\\ \varphi(\boldsymbol{\theta},\boldsymbol{x}) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^B\end{cases},\qquad S_h^B\varphi_h(\boldsymbol{\theta},\boldsymbol{x}_h)=\begin{cases}A_{\alpha}\varphi(\boldsymbol{\theta},\boldsymbol{x}_h) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^B\\ \varphi(\boldsymbol{\theta},\boldsymbol{x}) & \text{for } \boldsymbol{x}_h\in\boldsymbol{G}_h^R\end{cases} \tag{3.52}$$

where

$$A^{\alpha}=\frac{2c\cos(\theta_1)+4d\cos(\theta_1)\cos(\theta_2)}{a-2b\cos(\theta_1)}. \tag{3.53}$$

Now, the partial smoothing operators can be expressed as a line combination of $\varphi_h(\boldsymbol{\theta}^{\alpha},\boldsymbol{x}_h)$ $\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\boldsymbol{x}_h)$, where $\tilde{\boldsymbol{\alpha}}=(a_1,1-a_2)$ with $\boldsymbol{\alpha}=(a_1,a_2)$

$$\begin{aligned}S_h^R\varphi_h(\boldsymbol{\theta}^{\alpha},\boldsymbol{x}_h)&=\tfrac{1}{2}(1+A^{\alpha})\varphi_h(\boldsymbol{\theta}^{\alpha},\boldsymbol{x}_h)+\tfrac{1}{2}(A^{\alpha}-1)\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\mathbf{x}_h)\\ S_h^B\varphi_h(\boldsymbol{\theta}^{\boldsymbol{\alpha}},\boldsymbol{x}_h)&=\tfrac{1}{2}(1+A^{\alpha})\varphi_h(\boldsymbol{\theta}^{\alpha},\boldsymbol{x}_h)+\tfrac{1}{2}(-A^{\alpha}+1)\varphi_h(\boldsymbol{\theta}^{\tilde{\alpha}},\boldsymbol{x}_h)\end{aligned} \tag{3.54}$$

leading to the following (4x4)-matrix representations

$$\tilde{S}_h^R=\frac{1}{2}\begin{bmatrix}A^{(0,0)}+1 & 0 & 0 & A^{(0,1)}-1\\ 0 & A^{(1,1)}+1 & A^{(1,0)}-1 & 0\\ 0 & A^{(1,1)}-1 & A^{(1,0)}+1 & 0\\ A^{(0,0)}-1 & 0 & 0 & A^{(0,1)}+1\end{bmatrix}_h \quad \textit{and} \tag{3.55}$$

$$\tilde{S}_h^R=\frac{1}{2}\begin{bmatrix}A^{(0,0)}+1 & 0 & 0 & -A^{(0,1)}+1\\ 0 & A^{(1,1)}+1 & A^{(1,0)}-1 & 0\\ 0 & -A^{(1,1)}+1 & A^{(1,0)}+1 & 0\\ -A^{(0,0)}+1 & 0 & 0 & A^{(0,1)}+1\end{bmatrix}_h. \tag{3.56}$$

Figure 3-9 presents values of the smoothing factors for the described pattern relaxation

methods for two instances; the isotropic (left) and fully anisotropic (right). The graph lines of $JAC - RB$ and $GS - RB$ are particularly close for both test case problems. Despite the fact that the point red-black relaxations have satisfactory smoothing factors for the isotropic problems, they are not robust if the anisotropy is increased, as illustrated in Figure 3-9 (right), unless the parameter ε is increased simultaneously. On the other hand, Zebra Gauss-Seidel has better smoothing factors than the point red-black relaxations and, at the same time, can effectively handle anisotropies regardless the parameter value.

Remark 3.2.3 *Clearly, Zebra Gauss-Seidel smoother seems to be an excellent smoother for our model problem.*

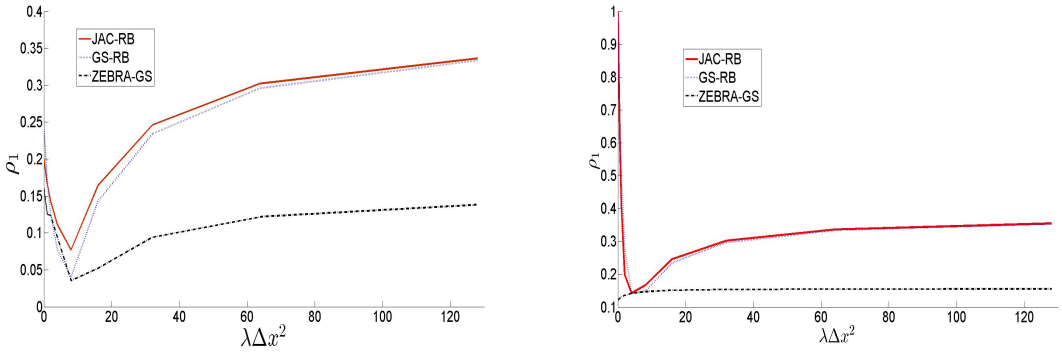

Figure 3-9: Smoothing factors for a variety of pattern relaxations; isotropic case (at left) and fully anisotropic case (at right).

Comparison between different coarsening strategies and relaxations

Since the combination of pointwise relaxation and standard coarsening strategy is inefficient for highly anisotropic problems, there are at least two different strategies that indicate promising smoothing factors as $\gamma \to 0$. As mentioned above, the first is to switch the relaxation procedure from pointwise to linewise and retain the standard coarsening strategy. An alternate multigrid approach in such problem cases is to reatain the pointwise relaxation and replace the standard coarsening strategy by the semi-coarsening one. In the case of semicoarsening , e.g. x-semicoarsing ($H = (2h, h)$), the range of high frequencies is configured to the set

$$\Theta_{high} = \left\{(\theta_1, \theta_2) : \theta_2 \in [-\pi, \pi] \setminus [-\frac{\pi}{2}, \frac{\pi}{2}]\right\}$$

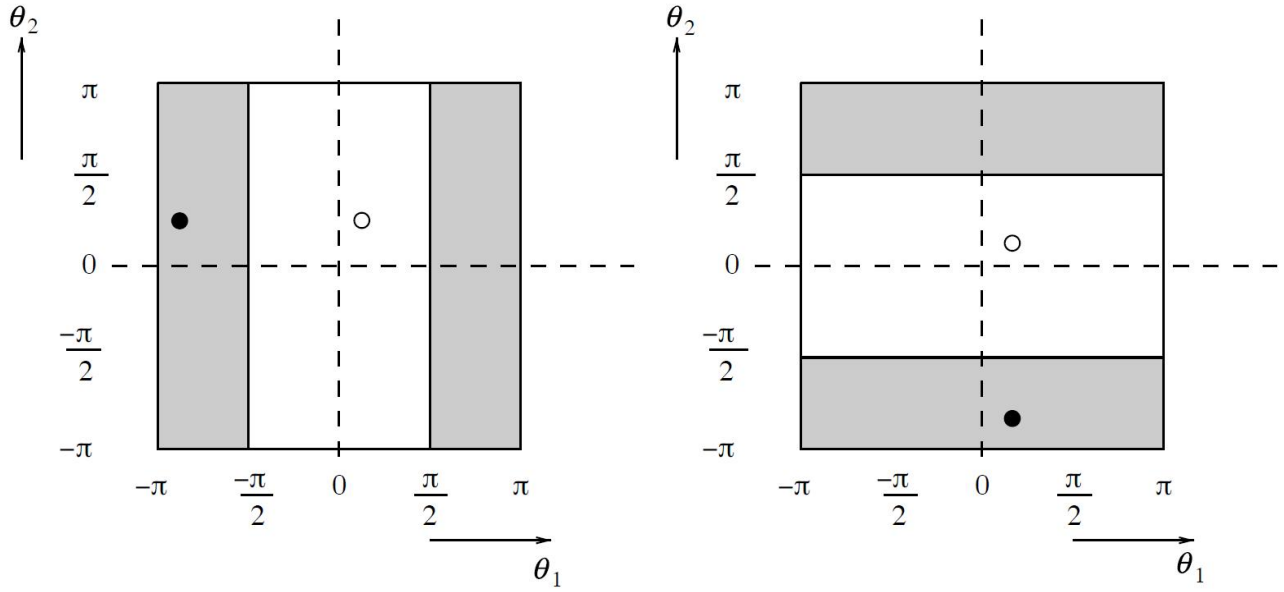

Figure 3-10: Low (white region) and high (shaded region) frequencies for $x-$semicoarsening (left) and $y-$semicoarsening (right); the high frequency (•) correspond to a given low frequency (∘).

Figure 3-10 illustrates the distinction of low and high frequencies in case of $x-$ or $y-$ semicoarsening. Table 3.1 presents the comparison between the smoothing factors of several relaxations using different coarsening strategies in a wide range of values of the anisotropic parameter γ. The linewise relaxations are paired with standard relaxation in contrast to pointwise smoothers that use $x-$semicoarsening.

Data in Table 3.1 indicates that zebra line Gauss-Seidel should be preferred over linewise relaxations. Comparing the results of the pointwise relaxations, a slight superiority of the $RB - GS$ smoother can be discerned. As for the parallel properties, the degree of parallelism is higher using the $RB - JAC$, but its smoothing factors turn to be worse than those of zebra line Gauss-Seidel for every possible γ value.

The smoothing factor values in Table 3.2 indicate that if a pointwise smoother is paired with semicoarsening for isotropic problems, the outcome is a worse convergence factor when compared to that of the standard coarsening strategy. An explanation is that the

Table 3.1: Smoothing factors ρ_1 in the anisotropic Poisson problem case using relaxations with different coarsening strategies for several values of γ.

γ	1	0.9	0.5	0.1	0.01	0.001	0.0001
line GS-LEX	0.447	0.447	0.447	0.447	0.447	0.447	0.447
zebra GS-LEX	0.160	0.125	0.125	0.125	0.125	0.125	0.125
GS-LEX	0.400	0.391	0.393	0.519	0.528	0.528	0.528
RB-JAC	0.280	0.242	0.238	0.178	0.176	0.176	0.176
RB-GS	0.280	0.225	0.180	0.151	0.150	0.150	0.150

Table 3.2: Smoothing factors ρ_1 in the Poisson problem case using pointwise relaxations with different coarsening strategies.

Relaxation	Coarsening strategy	
	standard	semicoarsening
GS-LEX	0.400	0.467
RB-JAC	0.200	0.280
RB-GS	0.246	0.280

semicoarsening process causes a discrete anisotropy by excessively stretching the computational cells.

3.3 Two- and three- grid analysis

First, the LFA to the two-grid transfer operator is applied. The error of a two-grid cycle can be represented as $e_h = N_h^H \bar{e}_h$, where the two-grid operator N_h^H is given by

$$N_h^H = S_h^{\nu_1} K_h^H S_h^{\nu_2} = S_h^{\nu_1} (I_h - P_H^h L_H^{-1} R_h^H L_h) S_h^{\nu_2}. \tag{3.57}$$

S_h is a smoothing operator on G_h, while v_1 and v_2 indicate the number of pre- and pro-smoothing iterations. Furthermore, K_H^h is the coarse grid correction operator, I_h the G_h-identity, L_H and L_h the discretization operators on G_H (coarse) and G_h (fine) grids. Lastly, P_H^h and R_h^H are the transfer operators from the coarse to the fine grid and vice versa. Subsequently, the impact of the above intergrid operators on the Fourier components will be explicated and their Fourier representations with respect to the $2h$-harmonics will be calculated.

3.3.1 Fourier representation of the discretization operators

The fine-grid operator L_h is represented on E_h^θ by the (4x4)-matrix

$$\hat{L}_h^H(\boldsymbol{\theta}) = \begin{pmatrix} \tilde{L}_h(\boldsymbol{\theta}^{00}) & 0 & 0 & 0 \\ 0 & \tilde{L}_h(\boldsymbol{\theta}^{11}) & 0 & 0 \\ 0 & 0 & \tilde{L}_h(\boldsymbol{\theta}^{10}) & 0 \\ 0 & 0 & 0 & \tilde{L}_h(\boldsymbol{\theta}^{01}) \end{pmatrix} \tag{3.58}$$

for each $\theta \in \boldsymbol{\Theta}_{Low}$. The Fourier representation of the coarse-grid operator L_H needs further treatment. It can be shown that for any low frequency $\boldsymbol{\theta}^{00}$ and $\boldsymbol{x} \in G_H$ applies that

$$\varphi_h(\boldsymbol{\theta}^{00}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{11}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{10}, \boldsymbol{x}) = \varphi_h(\boldsymbol{\theta}^{01}, \boldsymbol{x}). \tag{3.59}$$

and, moreover, it can be proved that

$$\varphi_h(\boldsymbol{\theta}^{\alpha}, \boldsymbol{x}) = \varphi_H(2\boldsymbol{\theta}^{00}, \boldsymbol{x}). \tag{3.60}$$

Hence,

$$L_H\varphi_h(\boldsymbol{\theta}^{\alpha}, \boldsymbol{x}) = L_H\varphi_H(2\boldsymbol{\theta}^{00}, \boldsymbol{x}) = \tilde{L}_H(2\boldsymbol{\theta}^{00})\varphi_H(2\boldsymbol{\theta}^{00}, \boldsymbol{x}) \tag{3.61}$$

that leads to

$$\hat{L}_H(\boldsymbol{\theta}) = \tilde{L}_H(2\boldsymbol{\theta}^{00}). \tag{3.62}$$

3.3.2 Fourier representation of the transfer operators

Focus is only drawn on the restriction operator as the following relation stands if prolongation and restriction are adjoined to each other

$$\tilde{P}_h^H(\theta_\alpha) = 4\tilde{R}_H^h(\theta_\alpha) \quad . \tag{3.63}$$

Assuming that the restriction operator is described by a stencil of type $R_h^H \hat{=} [r_\kappa]_h^H$

$$\begin{aligned} &R_h^H \hat{=} [r_\kappa]_h^H \\ &i.e. \quad R_h^H u_h(\mathbf{x}) = \sum_{\kappa\in J} r_\kappa u_h(\mathbf{x} + \kappa h), \qquad (\mathbf{x} \in G_H) \end{aligned} \quad . \tag{3.64}$$

Then, the following is obtained

$$\begin{aligned} R_h^H\varphi_h(\theta^{\alpha}, \mathbf{x}) &= \sum_{\kappa\in J} r_\kappa\varphi_h(\theta^{\alpha}, \mathbf{x} + \kappa h) = \sum_{\kappa\in J} r_\kappa e^{\mathrm{i}\theta^{\alpha}(\mathbf{x}+\kappa\mathbf{h})/\mathbf{h}} \\ &= \sum_{\kappa\in J} r_\kappa e^{\mathrm{i}\theta^{\alpha}\kappa} e^{\mathrm{i}\theta^{\alpha}\mathbf{x}/\mathbf{h}} = \sum_{\kappa\in J} r_\kappa e^{\mathrm{i}\theta\kappa}\varphi_h(\theta^{\alpha}, \mathbf{x}) \\ &= \tilde{R}_h^H(\theta^{\alpha})\varphi_h(2\theta^{00}, \mathbf{x}), \qquad (\mathbf{x} \in G_H) \end{aligned} \tag{3.65}$$

with the Fourier symbol $\tilde{R}_h^H(\theta^{\alpha})$, leading to

$$\hat{R}_h^H(\theta) = \left[\begin{array}{cccc} \tilde{R}_h^H(\theta^{00}) & \tilde{R}_h^H(\theta^{11}) & \tilde{R}_h^H(\theta^{10}) & \tilde{R}_h^H(\theta^{01}) \end{array} \right] \tag{3.66}$$

Next, the Fourier symbols of the developed restriction operators are derived and their

polynomial order is calculated along with the low- and high-orders. In order to illustrate the formulation that may be involved in the determination of the Fourier symbols $\hat{R}_h^H$ and the corresponding orders, the detailed derivation of the **CR** operator is provided.

Fourier symbols for restriction operators

The two dimensional **CR** operator can be written in the stencil notation

$$R_h^H \hat{=} \begin{bmatrix} 1 & & 1 \\ & * & \\ 1 & & 1 \end{bmatrix}. \tag{3.67}$$

Hence, the Fourier symbol is given by

$$\begin{aligned} \tilde{R}_h^H(\boldsymbol{\theta}^\alpha) = \sum_{\boldsymbol{\kappa}\in J} r_{\boldsymbol{\kappa}} e^{\mathrm{i}\boldsymbol{\theta}^\alpha \boldsymbol{\kappa}} &= \tfrac{1}{4}(e^{-\mathrm{i}\frac{\theta_1}{2}-\mathrm{i}\frac{\theta_2}{2}} + e^{\mathrm{i}\frac{\theta_1}{2}-\mathrm{i}\frac{\theta_2}{2}} + e^{-\mathrm{i}\frac{\theta_1}{2}+\mathrm{i}\frac{\theta_2}{2}} + e^{+\mathrm{i}\frac{\theta_1}{2}+\mathrm{i}\frac{\theta_2}{2}}) \\ &= \cos(\tfrac{\theta_1}{2})\cos(\tfrac{\theta_2}{2}). \end{aligned} \tag{3.68}$$

The Fourier symbols for the rest restriction operators are listed in Table 3.3 .

Orders of the transfer operators

The orders of the intergrid transfer operator strongly depend on the highest order of the PDE derivatives. The polynomial order of the prolongation (m_p) is equal to $k + 1$, if all polynomials of degree k are precisely interpolated. Value m_r indicates the order of the restriction operator and is defined as the order of its adjoint prolongation. The condition that the orders of the transfer operators should meet differ from the vertex-centered to cell-centered case, as mentioned in the previous Chapter. In the cell-centered case a slightly restrictive condition

$$m_p^{high} + m_r^{high} \geq M \quad , \tag{3.69}$$

Table 3.3: Fourier symbols for a variety of restriction operators.

restriction	Fourier symbols $\tilde{R}_h^H(\theta^\alpha)$
CR	$\cos(\frac{\theta_1}{2})\cos(\frac{\theta_2}{2})$
BR	$\cos^3(\frac{\theta_1}{2})\cos^3(\frac{\theta_2}{2})$
WR	$\cos(\frac{\theta_1}{2})\cos(\frac{\theta_2}{2})\cos(\frac{\theta_1-\theta_2}{2})$
KR	$\frac{1}{2}\cos(\frac{\theta_1}{2})\cos(\frac{\theta_2}{2})(\cos(\theta_1)+\cos(\theta_2))$
MR	$\frac{1}{4}(3\cos(\frac{\theta_1}{2})\cos(\frac{\theta_2}{2})+\cos(\frac{3\theta_1}{2})\cos(\frac{3\theta_2}{2}))$
BQ6R	$\frac{1}{1024}(3\cos(\frac{5\theta_1}{2})-5\cos(\frac{3\theta_1}{2})+30\cos(\frac{\theta_1}{2}))$
	$(3\cos(\frac{5\theta_2}{2})-5\cos(\frac{3\theta_2}{2})+30\cos(\frac{\theta_2}{2}))$
BQ8R	$\frac{1}{1024}(3\cos(\frac{7\theta_1}{2})-14\cos(\frac{3\theta_1}{2})-21\cos(\frac{\theta_1}{2}))$
	$(3\cos(\frac{5\theta_2}{2})-14\cos(\frac{3\theta_2}{2})-21\cos(\frac{\theta_2}{2}))$
BCR	$\frac{1}{16384}(5\cos(\frac{7\theta_1}{2})+7\cos(\frac{5\theta_1}{2})-35\cos(\frac{3\theta_1}{2})-105\cos(\frac{\theta_1}{2}))$
	$(5\cos(\frac{7\theta_2}{2})+7\cos(\frac{5\theta_2}{2})-35\cos(\frac{3\theta_2}{2})-105\cos(\frac{\theta_2}{2}))$

is needed to apply. The definitions of the low- (m_r^{low}) and high-frequency orders (m_r^{high}) of the restriction operator are: m_r^{low} and m_r^{high}, which are the largest numbers satisfying

$$\begin{aligned} \tilde{R}_h^H(\boldsymbol{\theta}^{(00)}) &= 1 + O(|\boldsymbol{\theta}^{(00)}|^{m_r^{low}} \quad \tilde{R}_h^H(\boldsymbol{\theta}^{(11)}) = O(|\boldsymbol{\theta}^{(00)}|^{m_r^1}) \\ \tilde{R}_h^H(\boldsymbol{\theta}^{(10)}) &= O(|\boldsymbol{\theta}^{(00)}|^{m_r^2}) \qquad \tilde{R}_h^H(\boldsymbol{\theta}^{(01)}) = O(|\boldsymbol{\theta}^{(00)}|^{m_r^3}) \end{aligned} \tag{3.70}$$

with $m_r^{high} = min\{m_r^1, m_r^2, m_r^3\}$. The vaules of m_p^{high} and $m_p^{low}$are similarly defined. Apart from (3.69), the low- and high-frequency orders are preferred to be positive in both the restriction and the prolongation. These orders of several transfer operators are shown in Table 3.4.

For instance, using the Taylor expansion of the corresponding Fourier symbol of *CR*, it can be found that

Table 3.4: Orders of several restriction operators.

		CR	BR	WR	KR	MR	BQ6R	BQ8R	BCR
low-frequency order	m_r^{low}	2	2	2	2	2	4	4	4
high-frequency order	m_r^{high}	1	3	2	2	2	3	3	5
polynomial order	m_r	1	2	2	2	2	3	3	4

$$\begin{aligned}
\tilde{R}_h^H(\theta^{(00)}) &= 1 - \tfrac{1}{8}(\theta_1^2 + \theta_2^2) + O(|\theta^{(00)}|^4) \\
\tilde{R}_h^H(\theta^{(11)}) &= \tfrac{1}{4}\theta_1\theta_2 + O(|\theta^{(00)}|^4) \\
\tilde{R}_h^H(\theta^{(10)}) &= sign(\theta_1)(\tfrac{1}{2}\theta_1 - \tfrac{1}{48}\theta_1^3 - \tfrac{1}{16}\theta_1\theta_2^2) + O(|\theta^{(00)}|^5) \\
\tilde{R}_h^H(\theta^{(01)}) &= sign(\theta_2)(\tfrac{1}{2}\theta_2 - \tfrac{1}{48}\theta_2^3 - \tfrac{1}{16}\theta_2\theta_1^2) + O(|\theta^{(00)}|^5)
\end{aligned} \quad , \tag{3.71}$$

hence the low-frequency order equals two, while the high-frequency equals 1. The polynomial order of CR is 1 since CP is a constant interpolation. It is evident that m_p, m_p^{low} and m_p^{high} are directly connected to the relation $m_p \leq m_p^{low}$, m_p^{high}.

To sum up, the Fourier representation of the two-grid operator for each $\theta \in \Theta^{low}$ by a (4×4)-matrix can be estimated using

$$\hat{N}_h^H(\theta) = \hat{S}_h^{v_1}(\theta)(\hat{I}_h - \hat{P}_H^h(\theta)\hat{L}_H^{-1}(2\theta)\hat{R}_h^H(\theta)\hat{L}_h(\theta))\hat{S}_h^{v_2}(\theta). \tag{3.72}$$

Based on the (4×4)-matrix representation of $N_h^{2h}$the asymptotic two-grid convergence factor can be calculated by

$$\rho_2(N_h^{2h}) := sup\{\rho(\hat{N}_h^{2h}(\boldsymbol{\theta})) \ : \ \boldsymbol{\theta} \in \Theta_{low}\} \quad , \tag{3.73}$$

where $\rho(\hat{N}_h^{2h}(\boldsymbol{\theta}))$ is the spectral radius of the $\hat{N}_h^{2h}(\boldsymbol{\theta})$.

Fourier three-grid analysis is based on a recursive application of the two-grid analysis. As a consequence, the three-grid operator is given by

$$N_h^{4h} = S_h^{\nu_2} K_h^{4h} S_h^{\nu_1} = S_h^{\nu_2} (I_h - P_{2h}^h (I_{2h} - (N_{2h}^{4h})^\gamma) L_{2h}^{-1} R_h^{2h} L_h) S_h^{\nu_1} \quad , \tag{3.74}$$

where $N_{2h}^4 h$ is defined from Eq. 3.57 [82].

In the next chapter, the multigrid technique is applied to the discretization scheme of the model problem and numerical results are compared with the theoretical one-, two- and three-grid convergence factor values.

Chapter 4

Multigrid Discretization Schemes Performance

In this chapter, a performance investigation of the numerical solution of the linear system, which arises from the proposed high-order discretization method, takes place. Moreover, theoretical convergence factors are compared with the numerical measurements of multigrid techniques. Furthermore, a review on every iterative solver's, described in Section 2.2, impact for solving the sparse linear system follows. The numerical validation of the fourth-order of accuracy of the discretization scheme, coupled with the multigrid technique including the boundary treatments is also performed. Consequently, an extended investigation of the implemented multigrid algorithm for the discretization method is presented and convergence rates for every transfer operator combination are documented. The following transfer operator combinations are being examined: BP, WP, KP, MP, BQ6P, BQ8P and BCP prolongation, with their corresponding adjacent restrictions. Horizontal Zebra Gauss-Seidel solver [79, 56, 86, 92] is chosen as a pre- and post-smoother within V-cycle, W-cycle , F-cycle and FMG cycling strategy. The coarsest grid of the multigrid cycle consists of 4×4 cells for all cases.

The numerical scheme coupled with the multigrid technique is compared against the op-

timal iterative solver. To this end, the termination criterion is set as the point where the Euclidean norm of the residual gets to a value of less than 10^{-10}. All the numerical tests were performed on a SunFire X2200M2 machine with 4GB of memory, and two dual core Opteron@3.0GHz processors. The application was developed in double precision Fortran code. All the basic linear algebra computations were performed using subroutines from the Lapack [45] scientific library.

4.1 Test problem 1

The first Helmholtz test problem accepts the exact solution

$$u(x,y) = 10\,\phi(x)\,\phi(y) \quad , \quad (x,y) \in \Omega \equiv [0,1]\times[0,1] \quad , \quad \text{with} \quad \phi(x) = e^{-100(x-0.1)^2}\,(x^2 - x).$$

Parameter λ is set to 1, unless considered otherwise, with Dirichlet conditions. As stated in Section 2.2, due to the structure of the emerged linear system, the corresponding coefficient matrix is 2-cyclic consistently ordered. Table 4.1 provides the numerical approximations of the values of parameter ω_{opt} of the SOR method, in several grid mesh cases.

Table 4.1: Numerical approximations of SOR's ω_{opt} and the corresponding convergence factors.

$N_x = N_y$	16	32	64	128	256	512	1024	2048
$\omega_{opt}^{numeric}$	1.5030	1.7460	1.8630	1.9300	1.9475	1.9705	1.9760	1.9910
$\rho_{opt}^{numeric}$	0.5021	0.7383	0.863	0.9294	0.9428	0.9678	0.9790	0.9870

Attention shall now be turned to the implementation of the iterative methods for the test problem 1. Figure 4-1 and Table 4.2 present the convergence behavior and execution time, respectively, of the stationary methods Jacobi, Gauss-Seidel and optimal SOR for the case of problems with 64, 128, 256 and 512 computing cells in each direction. As Fig. 4-1 illustrates, the optimal SOR method always converges faster.

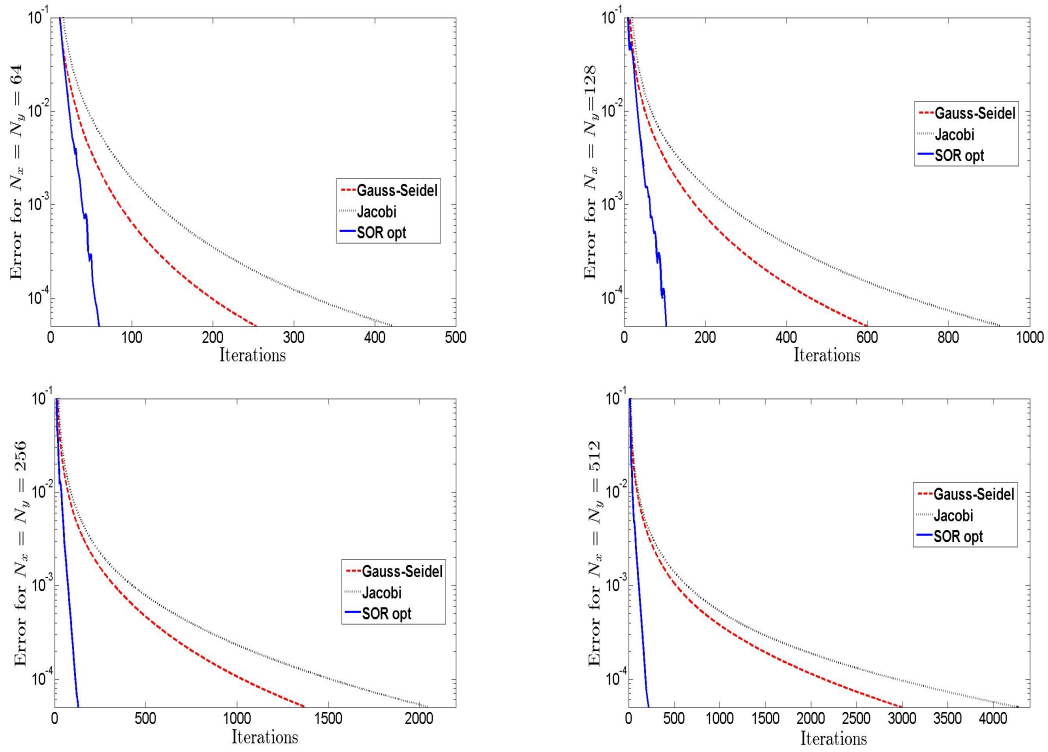

Figure 4-1: Residual norms for JAC, GS and SOR opt methods for selected discretizations.

The numerical results also indicate that Gauss-Seidel method converges almost twice as fast as Jacobi. That of course, shall be considered typical, as the eigenvalues of those two iterative methods are associated by the relation $\lambda_{GS} = \lambda_{JAC}^2$. As suggested from the hitherto measurement results, the SOR method shows promising convergence rates. However, for fine discretization problems the optimal parameter of SOR approaches two (see Table 4.1), leading to very slow convergence rates.

Table 4.2: CPU time in seconds for Jacobi, GS and optimal SOR methods.

	Jacobi	GS	SOR_{opt}
$N_x = N_y$	CPU time		
64	1.952	1.005	0.080
128	26.69	14.56	0.598
256	393	216	8.827
512	3780	2050	86.88

The most popular Krylov subspace methods are also considered for the solution of the linear system (2.55) . Preconditioning techniques are considered in order to improve the performance of the iterative methods. The convergence properties of the Generalized Minimal Residual method (GMRES) and Biconjugate Gradient Stabilized method (BICGSTAB) are inspected, in terms of the field of the Ritz values of the corresponding preconditioned matrix. It is well-known that, these values approximate the eigenvalues of the preconditioned matrix of the linear system.

Fig. 4-2 presents the first fifty Ritz values of the preconditioned matrix on the complex plane, using the Arnoldi iteration [101, 45]. A significant number of preconditioners is reviewed, such as Jacobi (top right), Gauss-Seidel (bottom left) and SGS (bottom right). Plot results indicate that Bi-CGSTAB will converge without precondition, since Ritz values are real and lie in the half complex plane (see top left panel of Figure 4-2). Moreover, the convex hull containing them does not enclose the origin [101, 45]. In addition, the Ritz values demonstrate the superiority of the Jacobi preconditioned Bi-CGSTAB against all other preconditioners and GMRES class methods, since these eigenvalues are real and clustered around unity.

Figure 4-3 demonstrates the convergence behavior of the unpreconditioned GMRES(50) and Bi-CGSTAB, along with the selected preconditioners. Gauss-Seidel and SGS preconditioned Bi-CGSTAB methods do not converge in the fine discretization cases, while in case of coarser grids their behaviour is rather irregular. The SGS preconditioned GMRES(50) requires less iterations than GMRES method. In accordance with the investigation of the Ritz values, the Jacobi preconditioned Bi-CGSTAB achieves a better rate of convergence than the Bi-CGSTAB family methods.

Table 4.3 presents the execution times for all Krylov subspace methods, in case of 256 and 512 computing cells in each direction, measured at the time convergence was accomplished. It can be noticed that the GMRES(50) family method requires significantly more time to reach an acceptable approximation of the solution, even more than the unprecondi-

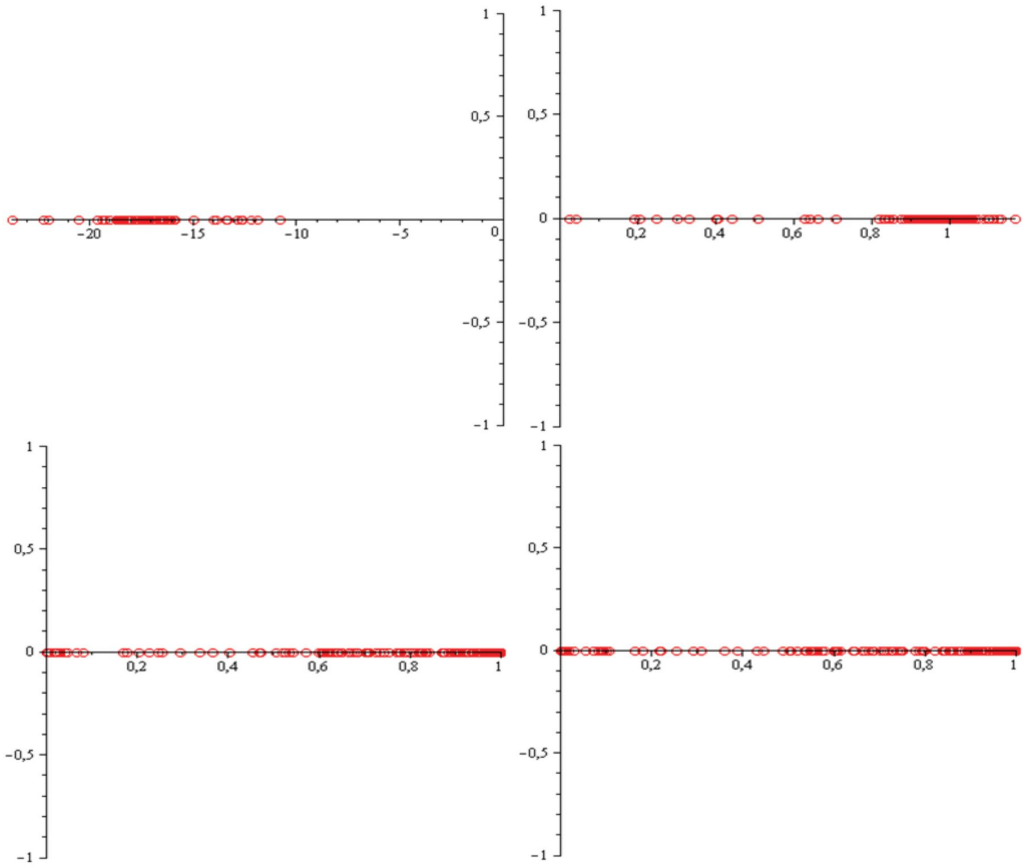

Figure 4-2: Ritz values for unpreconditioned and JAC, GS and SGS preconditioned Krylov subspace methods.

tioned Bi-CGSTAB. Again, the Bi-CGSTAB method coupled with Jacobi preconditioning is found to be the faster solver. Although the execution time was reduced with the use of

Table 4.3: CPU time in seconds for GMRES(50) and Bi-CGSTAB methods.

$N_x = N_y$	Krylov Method	CPU time	$N_x = N_y$	Krylov Method	CPU time
256	GMRES	51.17	512	GMRES	405
	JC. prec. GMRES	39.94		JC. prec. GMRES	447
	GS. prec. GMRES	14.58		GS. prec. GMRES	194
	SGS. prec. GMRES	31.81		SGS. prec. GMRES	362
	BiCGSTAB	11.98		BiCGSTAB	98.12
	JC. prec. GMRES	8.401		JC. prec. BiCGSTAB	73.85

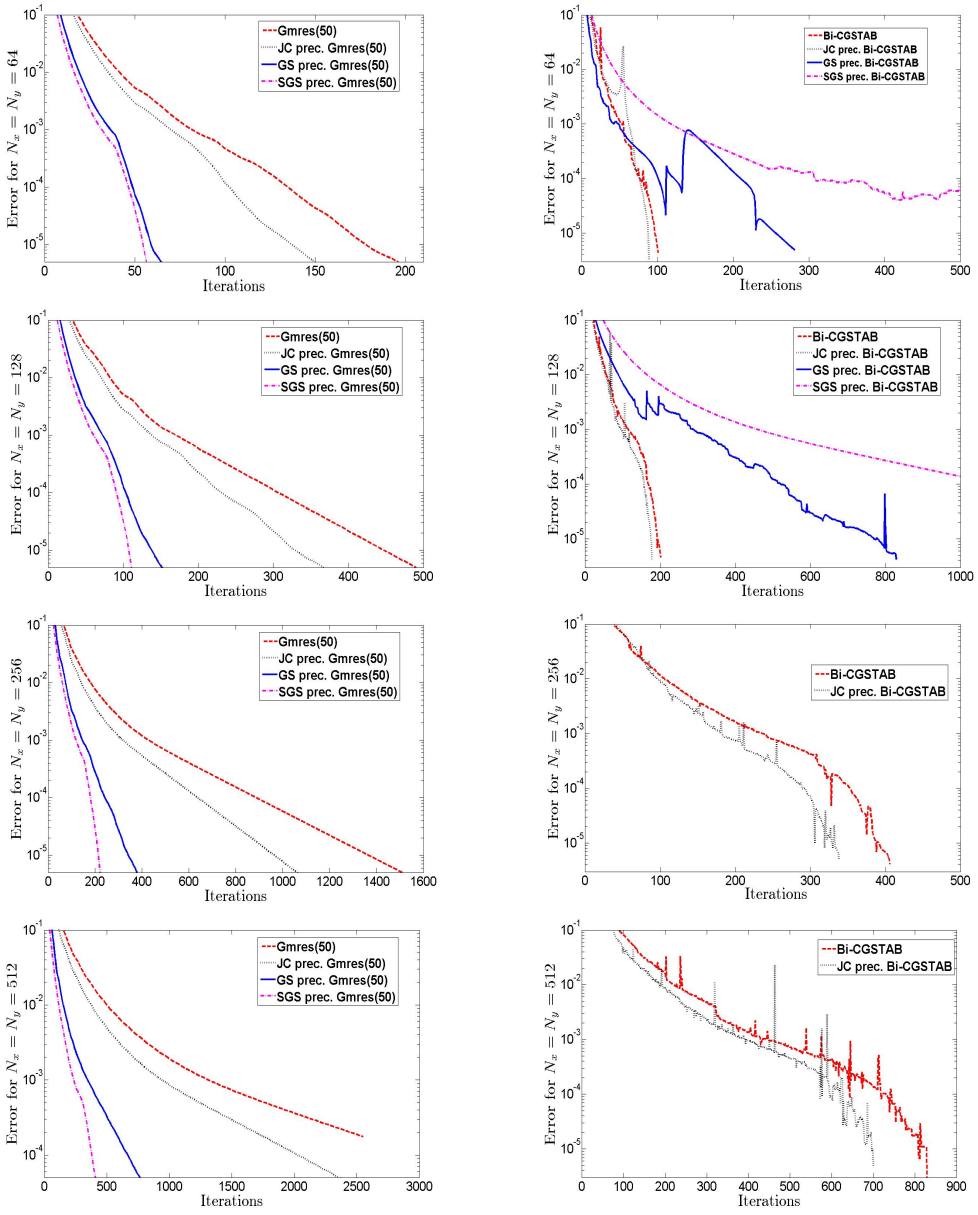

Figure 4-3: Convergence behavior for GMRES and Bi-CGSTAB methods.

SOR_{opt} and the Jacobi preconditioned Bi-CGSTAB solver, it still remains a challenging issue in case of finer discretizations. The convergence instabilities were noted using the SOR_{opt} and effort will be put in to decrease the execution time using multigrid acceleration techniques.

4.1.1 Boundary conditions

The high order finite difference compact solver based on the $V-$cycle multigrid technique with the Gauss-Seidel error relaxation scheme and $BR - BP$ intergrid transfer operators is used to demonstrate the accuracy using several boundary condition cases as presented below

$$\textbf{(a)} \quad u = \delta \qquad on \quad \partial\Omega \quad , \qquad \textbf{(b)} \quad \partial u/\partial \mathbf{n} = \delta \quad on \quad \partial\Omega$$

$$\textbf{(c)} \quad u + \partial u/\partial \mathbf{n} = \delta \quad on \quad \partial\Omega \quad , \qquad \textbf{(d)} \quad \begin{array}{l} u = \delta \qquad on \quad x = 0, x = 1, y = 1 \\ \partial u/\partial \mathbf{n} = \epsilon \quad on \quad y = 0 \end{array}$$

Functions δ and ϵ are the appropriate boundary functions for the test problem 1. Table 4.4 presents the infinite error norms between the computed ($\hat{u}$) and the exact solution (u), in the cases of 128 up to 2048 mesh size. The numerical results indicate, that the fourth-order of accuracy is verified for every BC choice. In addition, it is evident that the multigrid approach with the boundary condition treatments, as presented in Chapter 2, is stable.

Table 4.4: Order of accuracy of the method.

$N_x = N_y$	$\|u - \hat{u}\|_\infty$	order	$\|u - \hat{u}\|_\infty$	order	$\|u - \hat{u}\|_\infty$	order	$\|u - \hat{u}\|_\infty$	order
	(a)		**(b)**		**(c)**		**(d)**	
128	1.32e-6	-	1.95e-5	-	4.46e-5	-	2.54e-6	-
256	7.24e-7	4.19	1.71e-6	3.51	3.94e-6	3.50	2.58e-7	3.30
512	4.24e-9	4.09	1.15e-7	3.89	2.66e-7	3.89	1.80e-8	3.84
1024	2.57e-10	4.04	7.28e-9	3.98	1.70e-8	3.97	1.16e-9	3.96
2048	1.58e-11	4.02	4.62e-10	3.98	1.05e-9	4.02	7.36e-11	3.98

4.1.2 Local Fourier Analysis predictions versus asymptotic convergence rates

Now the test problem 1 with Dirichlet boundary conditions will be looked into. A $V(\nu_1, \nu_2)$ cycling strategy, with the GS smoother, is further assumed. The index pair (ν_1, ν_2) indicates the number of pre- and post-smoothing iterations within $V-$cycle and, in fact, pair $(1, 1)$ is considered. The predictions ρ_{3g} of the LFA with asymptotic convergence rates $\tilde{\rho}_{3g}$ for the three-level case are being compared, with a finest grid Ω_h of $h = 1/128$. The convergence factors $\tilde{\rho}_{3g}$ are experientially estimated using the observed residual reduction through

$$\tilde{\rho}_{3g} = \left(\frac{\|r_h^m\|_2}{\|r_h^0\|_2} \right)^{1/(m)} \quad , \tag{4.1}$$

where $\|\cdot\|_2$ indicates the discrete Euclidean norm and r_h^m is the residual after m-cycles. Table 4.5 summarizes the analytical and numerical smoothing factors. All the approximated values cited were noted after 10 cycles.

The agreement between the analytical and numerical results is acceptable. However, there are some discrepancies, as in the majority of instances, multigrid scheme involves either the prolongations CP and WP or the restrictions CR and WR. Especially in the case of (CP,CR), the LFA is promising, but inconsistent with the observed numerical convergence factor. This event might be explained by the rule (2.106). Although in the cell-centered case this rule is less restricted (2.107), it seems that while performing numerical simulations both of them should be taken into account.

The measured convergence rates compared to the corresponding values from the second-order compact scheme [64], evinces the superiority of the HOC scheme. In addition, the data in Table (4.5) is as decent as the factors obtained when in the vertex-centered case.

The observed convergence rates suggest an intergrid pair of a high- and a second-order operator. For instance, the $\tilde{\rho}_{3g}$ of (MP,BCR) or (BCP,MP) is around 0.046. However, a high-order transfer operator is rather intricate, when applied to nodes near the boundary,

Table 4.5: Prediction ρ_{3g} of LFA versus asymptotic convergence rates for the test problem 1 (V(1,1), h=1/128).

	CP	BP	WP	KP	MP	BQ6P	BCP	
CR	0.117	0.262	0.321	0.262	0.262	0.352	0.390	
BR	0.226	0.071	0.139	0.071	0.086	0.026	0.033	
WR	0.233	0.102	0.170	0.097	0.118	0.092	0.096	Local Fourier
KR	0.226	0.076	0.146	0.071	0.095	0.026	0.032	Analysis
MR	0.226	0.071	0.120	0.071	0.071	0.067	0.061	
BQ6R	0.166	0.050	0.137	0.050	0.050	0.144	0.160	
BCR	0.186	0.026	0.100	0.026	0.028	0.120	0.122	
CR	0.478	0.203	0.203	0.203	0.204	0.249	0.221	
BR	0.229	0.101	0.101	0.101	0.101	0.048	0.047	
WR	0.274	0.190	0.194	0.191	0.193	0.220	0.220	Numerical
KR	0.228	0.101	0.109	0.101	0.101	0.048	0.047	Results
MR	0.229	0.101	0.109	0.101	0.101	0.048	0.046	
BQ6R	0.312	0.076	0.091	0.076	0.077	0.105	0.082	
BCR	0.221	0.047	0.220	0.046	0.046	0.082	0.094	

because of its wide stencil. This restriction can be overcome by using lower-order transfer operators (see SubSection 2.3.3) for those nodes, without reducting the corresponding intergrid operator's convergence factors.

Proposed operators MR and MP behave similarly, as far as convergence is concerned, with the rest second-order operators and, at the same time, involve fewer nodes.

4.1.3 Comparing the multigrid schemes

In this section, an investigation on the efficiency of several multigrid type algorithms (V, W, F) takes place. Hierarchy grids are being used, starting from $h = 1/16$ and two transfer operator combinations (CP,CR) and (BP,BR). The results in Table 4.6 indicate that W(1,1)

and F(1,1) schemes produce smoothing factors which seem to be nearly independent of the grid size. In contrast, V(1,1) scheme along with the pair (CR,CP), exhibits poor convergence rates when used in increased grid size problems. Further, the convergence factors are stable using the pair (BP,BR), regardless the implemented cycling strategy.

Table 4.6: Convergence rates for the test problem 1.

Cycle	h=1/16	h=1/32	h=1/64	h=1/128	h=1/256	h=1/512	
V(1,1)	0.376	0.506	0.590	0.659	0.714	0.752	
W(1,1)	0.268	0.291	0.300	0.305	0.308	0.310	(CP,CR)
F(1,1)	0.268	0.292	0.303	0.307	0.309	0.310	
V(1,1)	0.100	0.110	0.120	0.122	0.129	0.131	
W(1,1)	0.070	0.074	0.074	0.074	0.074	0.074	(BP,BR)
F(1,1)	0.070	0.073	0.075	0.075	0.075	0.075	

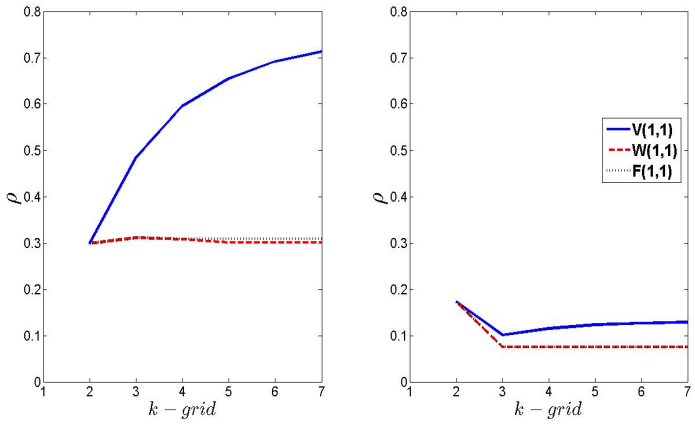

Figure 4-4: Numerical smoothing factors for increased grid level depth using cycling schemes (V, W, F) and two intergrid combinations (CP,CR) (left) and (BP,BR) (right); h=1/256.

A similar behavior is recorded as the grid level depth increases. Firstly, the low-order pair (CP,CR) is selected. The left graph of Fig. 4-4 shows the inefficiency of the V(1,1) scheme to stabilize the convergence rates in higher level grids. Consquently, the accuracy of the transfer operator switches to second-order (see right plot of Fig. 4-4). The smoothing factors now seem to be invariant for every cycling strategy.

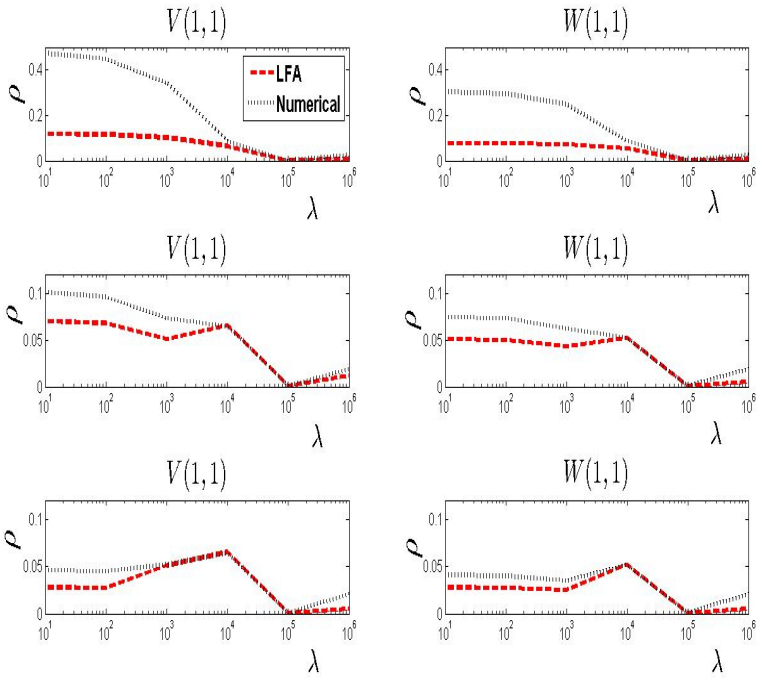

Figure 4-5: Smoothing factors (three-grid LFA and numerical) as function of the parameter λ for cycling schemes (V,W) and three intergrid operators combinations; (CP,CR) at top, (BP,BR) at middle and (MP,BCR) at bottom; h=1/128.

Figure 4-5 illustrates a comparison of different multigrid schemes with respect to parameter λ. Again, two transfer operator combinations are being considered. As the λ values increase, improved smoothing factors are being produced. In the case of (CP,CR) and the first cycling strategy V(1,1), a considerable decrease of the convergence factors is observed. Thus, for $\lambda > 10^4$ the V(1,1) cycle is comparable to the W(1,1) cycle. Additionally, all tested transfer combinations have identical convergence rates for an λ value greater than 10^4.

It is apparent that the W(1,1) and F(1,1) cycles do not provide better convergence rates than the V(1,1), but they validate the efficiency of the multigrid method.

In order to demonstrate the influence of v_1 and v_2 values on the V-cycle, Tables 4.7 and 4.8 present the convergence rates along with the execution time of those cycling strategies. Increasing the number of cycles for each V algorithm, leads to better convergence factors,

although the execution time increases as well. That can be explained by the fact that as the number of the pre- and post- smoothing sweeps increase, the arithmetic operations in every V-cycle also increase. As the λ values rise, the performance of the low-order intergrid combinations is improved. To sum up, the cycling strategy V(2,1) (or V(1,2)) is preferred.

Table 4.7: Convergence rates for several λ choices; h=1/256

Cycle	V(1,1)	V(2,1)	V(1,2)	V(2,2)	
(CP,CR)	0.841	0.612	0.657	0.524	
(MP,MR)	0.117	0.081	0.077	0.057	$\lambda = 1$
(BCP,MR)	0.069	0.039	0.029	0.019	
(CP,CR)	0.739	0.543	0.583	0.483	
(MP,MR)	0.116	0.079	0.074	0.057	$\lambda = 10^2$
(BCP,MR)	0.141	0.105	0.106	0.089	
(CP,CR)	0.355	0.240	0.252	0.199	
(MP,MR)	0.086	0.058	0.058	0.035	$\lambda = 10^4$
(BCP,MR)	0.090	0.053	0.053	0.037	

Table 4.8: CPU time for several λ choices; $h = 1/256$).

Cycle	V(1,1)	V(2,1)	V(1,2)	V(2,2)	
(CP,CR)	15.35	6.265	6.842	5.497	
(MP,MR)	1.222	1.251	1.073	1.301	$\lambda = 1$
(BCP,MR)	0.972	0.973	0.944	0.887	
(CP,CR)	8.865	5.292	5.521	4.552	
(MP,MR)	1.241	1.254	1.110	1.305	$\lambda = 10^2$
(BCP,MR)	1.273	1.592	1.331	1.452	
(CP,CR)	2.761	2.368	2.412	2.311	
(MP,MR)	1.238	1.341	1.294	1.311	$\lambda = 10^4$
(BCP,MR)	1.290	1.310	1.332	1.345	

4.1.4 Multigrid performance

Now, an investigation of the performance of V, W, F cycling schemes, coupled with the Full Multigrid (FMG) algorithm, will be presented. In every cycling strategy, the previ-

ously discussed intergrid transfer operators along with the GS smoother were incorporated. Dirichlet conditions are considered for the model problem in the boundary while the values for the pair (v_1, v_2) in the smoothing process are set to $(2, 1)$.

In Table 4.9, the execution time measurements and the required cycles in several mesh-size cases are presented. It should be noticed that the V-cycle scheme is sensitive when applying the intergrid operators. Nonetheless, the rest multigrid schemes seem to be less affected by the selection of the multigrid operators. The intergrid compination (BCP,MR) resulted to fastest cpu rates for all cycling strategies. Both W- and F-cycles require the same number of cycles to converge while, at the same time, F-cycle converges faster. Data in Table 4.9 also demonstrates an improved performance by the application of the FMG scheme.

Table 4.9: CPU time measurements and multigrid iterations for intergrid operators.

grid size	strategy	(CP,CR)		(MP,MR)		(BCP,MR)	
		Time	Cycles	Time	Cycles	Time	Cycles
128 x 128	$V-cycle$	0.259	28	0.065	7	0.058	6
	$W-cycle$	0.202	13	0.094	6	0.080	5
	$F-cycle$	0.170	13	0.081	6	0.069	5
	FMG	0.136	10	0.072	5	0.062	4
256x256	$V-cycle$	1.309	32	0.307	7	0.215	5
	$W-cycle$	0.748	12	0.371	7	0.215	5
	$F-cycle$	0.645	12	0.309	6	0.223	4
	FMG	0.434	8	0.230	4	0.186	3
512 x 512	$V-cycle$	6.261	35	1.321	7	0.957	5
	$W-cycle$	3.128	12	1.334	5	1.086	4
	$F-cycle$	2.781	12	1.170	5	0.956	4
	FMG	1.421	6	0.826	3	0.794	3
1024 x 1024	$V-cycle$	27.73	37	4.677	6	3.811	5
	$W-cycle$	11.94	11	5.529	5	4.613	4
	$F-cycle$	10.70	11	4.801	5	4.050	4
	FMG	4.915	7	3.080	3	3.223	3
2048 x 2048	$V-cycle$	122	38	18.58	6	12.72	4
	$W-cycle$	49.42	11	18.02	4	14.16	3
	$F-cycle$	44.47	11	16.18	4	12.74	3
	FMG	16.51	4	8.936	2	9.221	2

The analysis in the previous section aims to highlight the superiority of Multigrid techniques. This superiority is numerically verified by comparing the multigrid methods to every method - stationary, non-stationary, Schur complement - in terms of execution time. Table 4.10 presents the execution times for every method. The optimal SOR is the most cost inefficient method, while the Jacobi preconditioned Bi-CGSTAB method is slightly faster. This method coupled with the Schur complement method reduces execution time even more. As expected, the Schur complement method requires nearly half the time compared to the Jacobi preconditioned Bi-CGSTAB, as the iterative process only involves half of the unknowns (the black ones) . However, the FMG multigrid method can accelerate the solution procedure hundreds of times more, even in the coarse discretization of a 256x256 mesh choice.

With respect to parallel implementations, which are the objective of this study, the step from V- to FMG-cycles gives a substantial increase of parallel complexity: from $\log(\#\Omega_h)$ to $\log^2(\#\Omega_h)$. The parallel complexity increases the more the coarse grids are being processed in each cycle. Thus, the V-cycle along with the itergrid pair (BCP,MR), should be preferred from a parallel point of view.

Table 4.10: CPU time for several solver choices.

$N_x = N_y$	Iterative Method	CPU time	$N_x = N_y$	Iterative Method	CPU time
256	SOR_{opt}	8.827	512	SOR_{opt}	86.88
	JC. prec. BiCGSTAB	8.401		JC. prec. BiCGSTAB	73.85
	Schur complement	4.312		Schur complement	42.30
	FMG	0.186		FMG	0.794
1024	SOR_{opt}		2048	SOR_{opt}	9932
	JC. prec. BiCGSTAB	700		JC. prec. BiCGSTAB	7603
	Schur complement	390		Schur complement	3832
	FMG	3.080		FMG	8.936

4.2 Test problem 2

The second test problem aims to assess the efficiency of the proposed method, when the multigrid technique with partial semi-coarsening strategy is applied, compared to the full-coarsening (standard coarsening) approach. The problem accepts the exact solution

$$u(x, y) = \sin(10\pi x)cos(2\pi y) \quad ,$$

when solved in the unit square with Dirichlet boundary conditions. The numerical algorithms are being examined for several values of parameter $\lambda \in \{1, 10^2, 10^4\}$. It is apparent, that the exact solution alters faster in the x-direction than in the y-direction. Hence, a discretization with mesh ratio $\gamma \leq 1$ is favoured. Every multigrid method is based on the $V-cycle$ algorithm, as it is the most popular multigrid algorithm for such problems [49]. The results are summarized in Tables 4.11-4.13.

Table 4.11: Iteration counts for multigrid method.

Parameter	N_x	$N_y = 128$	$N_y = 256$	$N_y = 512$	$N_y = 1024$
$\lambda = 1$	128	8			
$\lambda = 10^2$		8			
$\lambda = 10^4$		8			
$\lambda = 1$	256	7	9		
$\lambda = 10^2$		7	9		
$\lambda = 10^4$		4	9		
$\lambda = 1$	512	6	7	9	
$\lambda = 10^2$		6	7	9	
$\lambda = 10^4$		3	6	9	
$\lambda = 1$	1024	6	7	8	9
$\lambda = 10^2$		5	6	8	9
$\lambda = 10^4$		2	5	7	9

Table 4.11 presents the iteration counts of the V-cycle algorithm. The experiments indicate that an increased λ value corresponds to fewer V-cycles, proving multigrid with the partial semicoarsening to be a superb strategy for the solution of anisotropic Helmholtz problems. Multigrid number of iterations with full-coarsening strategy is not affected by

the raise of the Helholmz parameter value.

Table 4.12: CPU time in seconds for multigrid method.

Parameter	N_x	$N_y = 128$	$N_y = 256$	$N_y = 512$	$N_y = 1024$
$\lambda = 1$	128	0.075			
$\lambda = 10^2$		0.075			
$\lambda = 10^4$		0.073			
$\lambda = 1$	256	0.165	0.374		
$\lambda = 10^2$		0.166	0.366		
$\lambda = 10^4$		0.097	0.364		
$\lambda = 1$	512	0.332	0.784	1.625	
$\lambda = 10^2$		0.332	0.779	1.654	
$\lambda = 10^4$		0.166	0.677	1.659	
$\lambda = 1$	1024	0.798	1.764	3.751	6.832
$\lambda = 10^2$		0.649	1.528	3.862	6.821
$\lambda = 10^4$		0.267	1.263	3.275	6.852

The reading of the results in Table 4.11 untangles that as the grid size becomes finer in the y-direction only, more V-cycles are required to converge. The CPU time almost doubles as the N_y value doubles (Table 4.12). Further, the CPU measurements indicate the advantage of using unequal meshsize discretizations in case of increased anisotropy ($N_x \gg N_y$) or/and the parameter λ. Finally, Table 4.13 contains the maximum error norms obtained using the HOC method under different multigrid strategies. It can also be noticed, that increasing N_y does not always end up to a more accurate solution, e.g. the least error obtained in the case of $N_x = 512$ reduces substantially, when $N_y = 256$ for every λ value (Table 4.13).

4.3 Concluding Remarks

The performance investigation demonstrates that multigrid discretization schemes are capable of solving efficiently the sparse linear system that emerges from the application of the HOC discretization method compared to classical iterative solvers. The selection of intergrid operators within the V-cycle algorithm affects the efficiency of the solution process.

Table 4.13: Maximum error norms of the solution.

Parameter	N_x	$N_y = 128$	$N_y = 256$	$N_y = 512$	$N_y = 1024$
$\lambda = 1$	128	8.24e-6			
$\lambda = 10^2$		6.68e-6			
$\lambda = 10^4$		3.37e-6			
$\lambda = 1$	256	3.18e-7	3.92e-7		
$\lambda = 10^2$		2.72e-7	3.43e-7		
$\lambda = 10^4$		7.81e-8	7.89e-8		
$\lambda = 1$	512	3.26e-8	1.75e-8	2.37e-8	
$\lambda = 10^2$		3.07e-8	1.58e-8	2.10e-8	
$\lambda = 10^4$		2.17e-8	2.68e-9	2.97e-9	
$\lambda = 1$	1024	3.15e-8	2.08e-9	2.93e-9	1.43e-9
$\lambda = 10^2$		3.31e-8	3.94e-9	9.91e-10	1.31e-9
$\lambda = 10^4$		2.21e-8	1.08e-9	1.91e-10	1.60e-10

Multigrid technique following a partial semi-coarsening strategy is preferable to that of a full coarsening strategy in highly anisotropic problems. Moreover, the proposed boundary treatment applied for Neumann, Dirichlet, Robin and mixed condition cases does not influence the discretization method's accuracy.

In the next Chapter, the HOC method is applied in order to solve the Navier-Stokes equations over equal and unequal discretizations.

Chapter 5

Solving the Navier-Stokes equations

The Navier-Stokes equations model the motion of viscous fluid substances. From a mathematical point of view, unsteady incompressible Navier-Stokes equations are parabolic equations in time and elliptic boundary problems in space. Hence, in the process of their solution (a) a set of initial conditions and (b) boundary conditions at each space boundary point for a fine value $t > 0$, are required. In practice, steady viscous flows are usually converted to unsteady problems, and are being solved using a time marching scheme. The desired solution of the steady viscous flow problem comes from the solution that supervenes as significant amount of time t has passed.

In incompressible flows, the density is assumed to be constant, to wit $\frac{\partial \rho}{\partial t} = 0$, which is equivalent to a divergence free velocity field. A general assumption in fluid flows at low speeds ($Ma < 0.2 - 0.3$) is that they can be treated as incompressible flows. Mach number (Ma) is a dimensionless quantity representing the ratio of flow velocity past a boundary to the local speed of sound and is mathematically written as

$$Ma = \frac{u}{c} \quad ,$$

where u is the local flow velocity with respect to the boundaries and c is the speed of sound in the medium. In a simple explanation, a speed of Mach 1 equals the speed of sound.

Therefore, Mach 0.75 is about 75% of the speed of sound (subsonic), and Mach 1.35 is about 35% faster than the speed of sound (supersonic).

The incompressible Navier-Stokes equations in the absence of body force, and in two spatial dimensions, can be written as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{5.1}$$

(conservation form)

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = -\nabla p + \frac{1}{Re}\Big(\frac{\partial \mathbf{F_v}}{\partial x} + \frac{\partial \mathbf{G_v}}{\partial y}\Big), \tag{5.2}$$

where $\mathbf{u} = [u, v,]^T$ and p are the velocity vector and pressure respectively, and $Re = \frac{\rho UL}{\mu} = \frac{UL}{\nu}$ is the non-dimensional Reynolds number based on the characteristic velocity U and the characteristic length-scale L. The kinematic viscosity (ν) is the ratio of the dynamic viscosity μ to the density (ρ) of the fluid. Vectors $\mathbf{F}$ and $\mathbf{G}$ are both inviscid flux vectors, while $\mathbf{F_v}$ and $\mathbf{G_v}$ are the viscous flux terms given by:

$$\mathbf{F} = [u^2, uv]^T, \ \mathbf{G} = [vu, v^2]^T$$

$$\mathbf{F_v} = \Big[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\Big]^T, \quad \mathbf{G_v} = \Big[\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\Big]^T.$$

The first equation (5.2) is called the continuity equation and describes the conservation of mass. The second equation (5.1) is called the momentum equation and describes the conservation of flow momentum. The first term on the right hand side of the momentum equation refers to the forces that occur when pressure changes.

The development of high-order accurate finite difference numerical methods for the solution of the incompressible Navier-Stokes equations is a subject of continuous interest with applications in several fields; fluid structure interaction, low speed aerodynamics,

biomechanics, direct and large eddy simulations of turbulence with immersed boundary methods [78, 31] to name a few. In this study, pressure-velocity coupling is employed and the solution technique is based on a high-order accurate compact discretization [51], combined with a multigrid acceleration method [1, 4, 56, 92, 75, 86]. Traditional numerical methods to reach a solution of an incompressible flow equation include pressure correction [40, 21] or fractional time-step methods [11, 25, 55, 65]. Fractional time-step methods are often combined with a Poisson-type equation in the pressure correction phase. Operator splitting and predictor-corrector methods, that disassociate the computation of the velocity and the pressure [81], are also frequently used. A way to circumvent the difficulties associated with the enforcement of the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ is to use the artificial compressibility or pseudocompressibility method, originally introduced in [10] for the solution of steady-state incompressible flows. In [16] an extension to the time-dependent incompressible Navier-Stokes equations was cited. The main disadvantage of this technique however, is that the numerical diffusion required to stabilize the solution is rather high, to a degree that this approach is not suitable for high resolution time dependent flow simulations, even combined with high order accuracy techniques.

The continuity equation of the incompressible flow equations has a non-evolutionary character, because the pressure is included in a non-time-dependent form. An entirely different approach would be a way to rewrite the Navier Stokes equations in vorticity–stream–function formulation as in [17], [12] and [77]. This formulation however, faces additional difficulties, when applied to three dimensional flow simulations.

The incompressibility constraint can be enforced iteratively to each Runge-Kutta stage, using a local pressure correction or a global pressure correction method. A Poisson equation utilizes the global pressure correction obtained from the momentum equations, and yields the pressure field distribution in terms of the velocities. Pressure correction with a Poisson equation is accomplished by advancing the velocity field in time, solving numerically the momentum equations and, subsequently, solving the correction equation for

pressure at the current time step, so that continuity can be enforced in the next time step [22]. Hence, in this method velocity and pressure are indirectly coupled.

In this solver the Global Pressure correction method is employed based on the Poisson equation [72]. The continuity equation is combined with the momentum and the divergence-free constraint becomes an elliptic (in nature [25, 65]) equation for the pressure.

In order to ensure energy conservation in the discrete level [71], the velocities are advanced in time by numerically solving the discretized momentum equations on the cell-face centers. Next, the Poisson equation for the pressure is being solved, and discretization on the cell centers at the current time step takes place, so that continuity at the next time step is fulfilled. This approach has been employed in several approximation methods used to numerically solve the incompressible Navier-Stokes equations, [11], [81], [71]. For example, the SIMPLE method [88] is using a Poisson-type equation to enforce incompressibility. A simplified and physically appealing approach that involves the cell-by-cell pressure correction [21] at each time step has been proposed, and still is a popular technique, despite the increased computational cost for higher Reynolds number flows. That is imputed to the fact that, as the Reynolds number increases, the number of iterations required in the cell-by-cell pressure correction also considerably increases and the associated computational cost becomes intolerable. It is generally observed, that pressure correction methods based on the solution of a Poisson-type equation, require fewer iterations for high Reynolds numbers in order to converge [71], compared to the cell-by-cell method. Therefore, they can efficiently be extended to the three dimensional case.

It has long been recognized that improved aerodynamic design requires detailed information of the near wall flowfield. For example, although the high drag caused by turbulent flow has significant impact on the operational cost of vehicles, mechanisms of receptivity and nonlinear growth of instabilities, which under various noise environments lead complex flows to transition and turbulence, are poorly understood. Even less understood is the im-

pact of recently developed conventional and unconventional flow control techniques on performance and operational cost. Typical second-order accurate in space methods employed in computational fluid dynamics (CFD) algorithms for incompressible flows, e.g. [21], [85], require a large number of grid points to adequately resolve the steep flow gradients. On the other hand, high-order upwind methods often introduce artificial diffusivity, partly contaminating the numerical solution. Recent studies have focused on the development of methods based on high-order discretizations of the numerical solution [71] with high-order finite difference compact schemes, applied on a staggered-grid for the convective and viscous fluxes in the momentum equations. Additionally, this requires the development of high resolution techniques in order to incorporate the incompressibility condition, i.e. the development of high resolution techniques for solving elliptic Boundary Value Problems (BVPs). In [71], the Poisson-type equation is solved using a fourth-order accurate finite difference scheme, applied to enforce the incompressibility condition to the Navier-Stokes approximated solution universally.

The conservative or divergence form of the convective terms is employed in order to ensure energy conservation (analyzed in detail by Morinishi et al. in [100] for high order explicit schemes and in [24], [71], [73] for compact schemes). Energy conservation in the discrete level is important on the overall stability and accuracy of the computational method and of critical importance for large eddy simulation (LES). The conservative staggered grid approach was extended to the finite volume method, with compact high order schemes, by Mahesh et al. in [52] and Nagarajan et al. in [73].

In this work, the Poisson-type equation is solved using a fourth-order accurate finite difference scheme as presented in Chapter 2, applied to enforce the incompressibility condition to the Navier-Stokes solution universally. The pressure is computed in the center of each computational cell, discretized under fourth-order accurate compact schemes. The resulting discrete Poisson equation has a nine-point stencil. Fourth order compact finite difference schemes have been derived for vertex-centered grids and for 2D and 3D Pois-

son equations [60, 97, 98, 70]. The present scheme is compatible with cell-centered grids. An improvement of the proposed method is the treatment of realistic boundary conditions. Boundary closure formulas in case of Dirichlet, Neumann, Robin or mixed-type boundary conditions applied on the physical boundary, are derived.

The numerical solution of a Poisson-type equation is a highly demanding task both in terms of computational cost and in core memory, even with low order accurate discretization schemes. The arising linear system is large and sparse, necessitating the use of an iterative solver in order to minimize the computational time. Significant convergence acceleration however, may be achieved with the employment of a multigrid technique paired to the iterative solver. Based on the previous analysis (see Chapter 3) and investigation (see Chapter 4) the numerical solver can incorporate a cell-centered multigrid technique [74, 99, 43, 62], using a partial semi-coarsening strategy [49] and zebra grid-line Gauss-Seidel relaxation for solving unsteady incompressible Navier-Stokes equations. However, it is worth mentioning that applications of cell-centered multigrid method are usually based on second-order discretizations [64, 62, 99]. There is a poor literature concerning investigations on the effects of various multigrid components of the pressure correction procedure. A comparison of different smoothers, including some pressure correction methods on a staggered grid, can be found in [19] while in [35] are a few available on non-staggered grids. These comparisons pertain to nonlinear multigrid methods based on Full Approximation Scheme (FAS) [1]. Gjesdal *et. al.* [35] reported that changing the restriction and prolongation operator had no significant effect on the convergence of the method.

The outline of this chapter is the following: firstly, the discretization of the momentum equations is presented. Then, the pressure correction scheme and the discretization of the resulting Poisson-type equation along with the boundary treatments follow, and finally, the validation and the performance results of the implemented solver over a set of steady and unsteady problems and high resolution simulations sign this section off.

5.1 The Numerical Scheme

In this section, a description of the numerical method is presented. The numerical method uses fourth order accurate compact schemes, formulated on a staggered grid arrangement. The corresponding spatial discretization of the first- and second-order derivatives in (5.1) and (5.2) are described in a more detailed way in [71]. Incompressibility is enforced using a globally defined pressure correction scheme, computed from the solution of a Poisson-type problem, coupling (5.1) and (5.2) through the associated perturbed velocities. In order to ensure high-order spatial accuracy, the pressure correction equation is also discretized through a fourth-order finite compact scheme. In the implementation of compact schemes, the boundary conditions need to be discretized using a single layer of fictitious cells along the boundary, unlike the explicit high order finite difference methods, where a wider fictitious layer is required. It is known that high accurate approximations of pressure correction techniques based on the global Poisson-type equation are effective in flows with a high Reynolds number. A fourth order accurate method [71], which confines the need for very fine grids, does not guarantee the reduction of the computational needs that emerge from the numerical solution of the Poisson equation for large scale problems. A cell-centred geometric multigrid technique [74] for the acceleration of the numerical solution of the Poisson-type equation is preferred in order to reduce the computational cost. Multigrid accelerating techniques are of interest because of their ability to maintain problem size-independent convergence rates in certain elliptic problems [1, 4, 56]. The temporal discretization is carried out by the classical explicit fourth-order Runge-Kutta method [47].

5.1.1 Staggered grid arrangement

The physical domain $\Omega \equiv [0, L_x] \times [0, L_y]$ is equally subdivided into cells of width Δx and height Δy, where $N_x = L_x/\Delta x$ and $N_y = L_y/\Delta y$ is the number of computational cells in each direction. The vertices's of each cell $C_{i,j}$ are (x_i, y_j), with $x_i = i\Delta x$ and $y_j = j\Delta y$ for

$0 \leq i \leq N_x$ and $0 \leq j \leq N_y$. The midpoints on the vertical and horizontal edges, and centers of the cell are defined by the associated midpoints on each spatial direction denoted by $x_{i-1/2} = (i - 1/2)\Delta x$ and $y_{j-1/2} = (j - 1/2)\Delta y$ for $0 < i \leq N_x$ and $0 < j \leq N_y$. An exterior fictitious layer of a single-cell-width adjacent on each side of the physical domain is also added. The fictitious points (see Fig. 5-1) are numbered using the indices $i = -1, N_x + 1$ and $j = -1, N_y + 1$ in each axis. Fig. 5-1 right illustrates the dependent variables locations on each computing cell $C_{i,j}$ as follows: the pressure p, denoted by $p_{i,j}$, in the cell centers

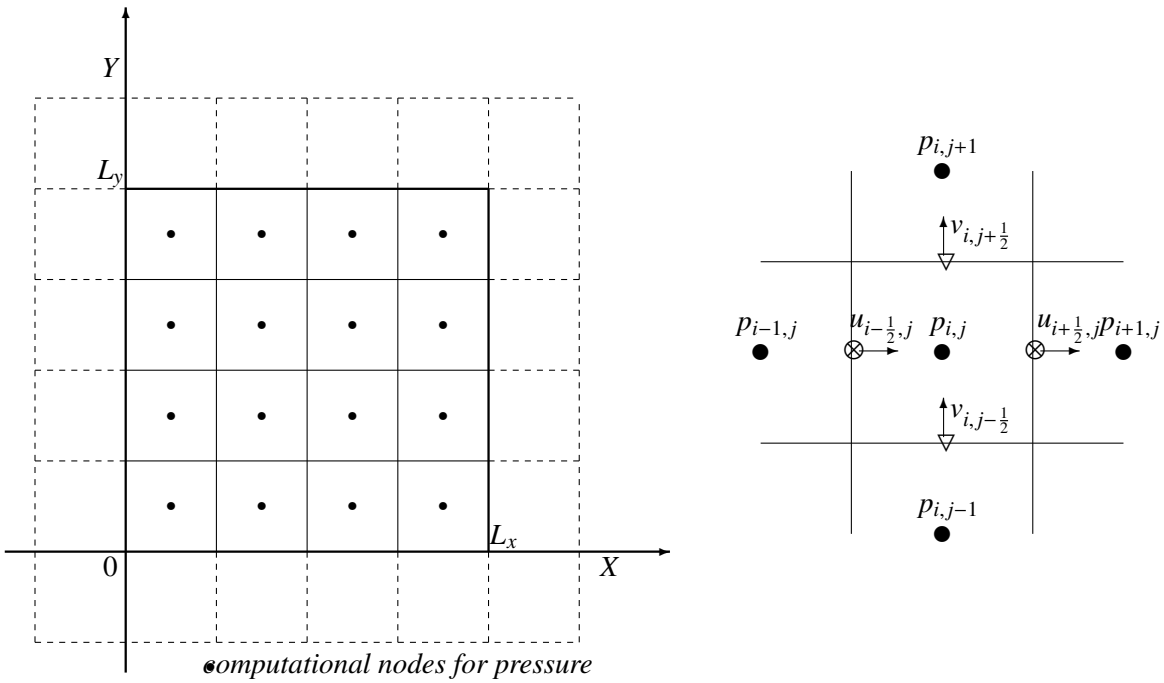

Figure 5-1: Staggered grid discretization $\left(N_x = N_y = 4\right)$ subintervals (left) and the schematic details for the computational cell C_{ij} (right)

(Fig. 5-1 right) with coordinates $(x_{i-1/2}, y_{j-1/2})$; the u velocity component is denoted by $u_{i+1/2,j}$ on the midpoint of the vertical edges with coordinates $(x_i, y_{j-1/2})$; and the v velocity component is denoted by $v_{i,j+1/2}$ on the midpoints of the horizontal edges with coordinates $(x_{i-1/2}, y_j)$.

5.1.2 Spatial Discretization

The Eq. (5.1) can be extensively expressed as

$$\frac{\partial u}{\partial t} = F(u,v,p;t) = -\frac{\partial p}{\partial x} - \frac{\partial}{\partial x}(u^2) - \frac{\partial}{\partial y}(uv) + \frac{1}{Re}\left(\frac{\partial}{\partial x}(\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(\frac{\partial u}{\partial y})\right) \tag{5.3}$$

$$\frac{\partial v}{\partial t} = G(u,v,p;t) = -\frac{\partial p}{\partial y} - \frac{\partial}{\partial y}(v^2) - \frac{\partial}{\partial x}(uv) + \frac{1}{Re}\left(\frac{\partial}{\partial x}(\frac{\partial v}{\partial x}) + \frac{\partial}{\partial y}(\frac{\partial v}{\partial y})\right) \tag{5.4}$$

The staggered grid discretization requires approximations of both the u- momentum and the v- momentum equations (5.3), (5.4) at the midpoints of the vertical and horizontal cell-edges respectively, while the continuity equation (5.1) is discretized at the center of each cell. Compact, Padé-type finite difference schemes [24, 83] are being used to reach a fourth-order of accuracy in the derivatives of the convective fluxes. Compact schemes evaluate the derivatives in a coupled fashion performing tridiagonal matrix inversions.
For the discretization of the second-order derivatives in the viscous fluxes $\mathbf{F}_v$ and $\mathbf{G}_v$ on staggered grids, the first derivative approximation is applied twice. A fourth order compact scheme based on Pad é approximations for the first derivatives in the left-hand side of the equations (5.3 and 5.4) will eventuate. For the implementation, the boundary conditions is necessary to be incorporated, such that we obtain closed systems. If h represents the step size, the first-order derivative at an integer point (x_i) is computed in a coupled fashion using functional values from integer points, solving the following linear system

$$(\mathcal{P}_1)\phi')_i = (\mathcal{Q}_1\phi)_i \quad , \; i = 1, ..., N \quad , \tag{5.5}$$

where the compact finite-difference operators $\mathcal{P}$ and $\mathcal{Q}$, are

$$(\mathcal{P}_1\phi^{'})_i = \begin{cases} \phi_1^{'} + 3\phi_2^{'} , & i = 1 \\ \phi_{i-1}^{'} + 4\phi_i^{'} + \phi_{i+1}^{'} , & i = 2, ..., N-1 \\ 3\phi_{N-1}^{'} + \phi_N^{'} , & i = N \end{cases} ,$$

$$(\mathcal{Q}_1\phi)_i = \begin{cases} \frac{1}{6h}(-17\phi_1 + 9(\phi_2 + \phi_3) - \phi_4) , & i = 1 \\ \frac{3}{h}(\phi_{i+1} - \phi_{i-1}) , & i = 2, ..., N-1 \\ \frac{1}{6h}(\phi_{N-3} - 9(\phi_{N-2} + \phi_{N-1}) + 17\phi_N , & i = N \end{cases} .$$

If known values at the midpoints are involved, the first-order derivatives at the integer points are computed using functional values from the midpoints, solving the following linear system

$$(\mathcal{P}_2)\phi')_i = (\mathcal{Q}_2\phi)_i \quad , \; i = 1, ..., N \tag{5.6}$$

where the compact finite-difference operators $\mathcal{P}$ and $\mathcal{Q}$ are

$$(\mathcal{P}_2\phi^{'})_i = \begin{cases} \phi_1^{'} + 22\phi_2^{'} , & i = 1 \\ \phi_{i-1}^{'} + 22\phi_i^{'} + \phi_{i+1}^{'} , & i = 2, ..., N-1 \\ 22\phi_{N-1}^{'} + \phi_N^{'} , & i = N \end{cases} ,$$

$$(\mathcal{Q}_2\phi)_i = \begin{cases} \frac{1}{24h}(-577\phi_{\frac{3}{2}} + 603\phi_{\frac{5}{2}} - 27\phi_{\frac{7}{2}} + \phi_{\frac{9}{2}}) , & i = 1 \\ \frac{24}{h}(\phi_{i+1/2} - \phi_{i-1/2}) , & i = 2, ..., N-1 \\ \frac{1}{24h}(-\phi_{N-\frac{7}{2}} + 27\phi_{N-\frac{5}{2}} - 603\phi_{N-\frac{3}{2}} + 577\phi_{N-\frac{1}{2}}) , & i = N \end{cases} .$$

The same formulas are functional for the half-points $x_{i-\frac{1}{2}}$ with a few simple modifications, denoted with operators $\tilde{\mathcal{P}}_2$ and $\tilde{\mathcal{Q}}_2$.

Moreover, interpolations are required; between the collocated points (x_i, y_j) and the staggered velocity midpoints of the vertical edges with coordinates $(x_i, y_{j-1/2})$ and the mid-

points of the horizontal edges with coordinates $(x_{i-1/2}, y_j)$. The necessary functional values at integer valued points emerge by a compact interpolation of the functional values at the midpoints, solving the following linear systems

$$(\mathcal{P}_0)\phi_i^* = (\mathcal{Q}_0\phi)_i \quad , \; i = 1, ..., N \; , \tag{5.7}$$

where the compact finite-difference operators $\mathcal{P}_0$ and $\mathcal{Q}_0$ are

$$(\mathcal{P}_0\phi)_i = \begin{cases} \phi_1^* + \phi_2^* \; , & i = 1 \\ \phi_{i-1}^* + 6\phi_i^* + \phi_{i+1}^* \; , & i = 2, ..., N-1 \\ \phi_{N-1}^* + \phi_N^* \; , & i = N \end{cases} ,$$

$$(\mathcal{Q}_0\phi)_i = \begin{cases} \frac{1}{4}(\phi_{\frac{1}{2}} + 6\phi_{\frac{3}{2}} + \phi_{\frac{5}{2}}) \; , & i = 1 \\ 4(\phi_{i-\frac{1}{2}} + \phi_{i+\frac{1}{2}}) \; , & i = 2, ..., N-1 \\ \frac{1}{4}(\phi_{N-\frac{3}{2}} + 6\phi_{N-\frac{1}{2}} + \phi_{N+\frac{1}{2}} \; , & i = N \end{cases} .$$

The computation of the second-order derivatives requires the successive application of (5.6) twice. The derivatives $\frac{\partial P}{\partial x}$ and $\frac{\partial P}{\partial y}$ in the pressure gradient ∇P, are fourth-order accurate and their evaluation is provided once the linear system (5.6) has been solved. These compact interpolations used herein are available in [71]. The numerical experiments in [71] confirmed the stability of this approach.

The space discretizations are earlier defined in one space dimension. The subscripts x and y are introduced to indicate the coordinates in directions and so, the terms in the momentum equations (5.3) and (5.4) have the following approximations

$$
\begin{aligned}
p_x &\approx \mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x}p \quad \text{at all points} \quad (x_i, y_{j-\frac{1}{2}}) \\
p_y &\approx \mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}p \quad \text{at all points} \quad (x_{i-\frac{1}{2}}, y_j) \\
(u^2)_x &\approx \mathcal{P}_{1x}^{-1}\mathcal{Q}_{1x}u^2 \quad \text{at all points} \quad (x_i, y_{j-\frac{1}{2}}) \\
(v^2)_y &\approx \mathcal{P}_{1y}^{-1}\mathcal{Q}_{1y}v^2 \quad \text{at all points} \quad (x_{i-\frac{1}{2}}, y_j) \\
(uv)_y &\approx E_y(u)E_x(v)\mathcal{P}_{1x}^{-1}\mathcal{Q}_{1x}uv \quad \text{at all points} \quad (x_i, y_{j-\frac{1}{2}}) \\
(uv)_x &\approx E_y(u)E_x(v)\mathcal{P}_{1y}^{-1}\mathcal{Q}_{1y}uv \quad \text{at all points} \quad (x_{i-\frac{1}{2}}, y_j) \\
u_{xx} &\approx \tilde{\mathcal{P}}_{2x}^{-1}\tilde{\mathcal{Q}}_{2x}\mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x}u \quad \text{at all points} \quad (x_i, y_{j-\frac{1}{2}}) \\
u_{yy} &\approx \tilde{\mathcal{P}}_{2y}^{-1}\tilde{\mathcal{Q}}_{2y}\mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}u \quad \text{at all points} \quad (x_i, y_{j-\frac{1}{2}}) \\
v_{xx} &\approx \tilde{\mathcal{P}}_{2x}^{-1}\tilde{\mathcal{Q}}_{2x}\mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x}v \quad \text{at all points} \quad (x_{i-\frac{1}{2}}, y_j) \\
v_{yy} &\approx \tilde{\mathcal{P}}_{2y}^{-1}\tilde{\mathcal{Q}}_{2y}\mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}v \quad \text{at all points} \quad (x_{i-\frac{1}{2}}, y_j)
\end{aligned}
\tag{5.8}
$$

where $E_y(u)$ and $E_x(v)$ are interpolations applied when solving the linear system (5.7). The argument indicates which velocity component is involved, while the subscript x or y denotes the coordinate direction.

To reach a conclusion, the discretized momentum equations are (subscripts ommitted)

$$
\begin{aligned}
R_1(u, v, p; t) = &- \mathcal{P}_{1x}^{-1}\mathcal{Q}_{1x}u^2 - E_y(u)E_x(v)\mathcal{P}_{1y}^{-1}\mathcal{Q}_{1y}uv - \mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x}p \\
&+ \frac{1}{Re}\left(\tilde{\mathcal{P}}_{2x}^{-1}\tilde{\mathcal{Q}}_{2x}\mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x} + \tilde{\mathcal{P}}_{2y}^{-1}\tilde{\mathcal{Q}}_{2y}\mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}\right)u
\end{aligned}
\tag{5.9}
$$

and

$$
\begin{aligned}
R_1(u, v, p; t) = &- \mathcal{P}_{1y}^{-1}\mathcal{Q}_{1y}v^2 - E_y(u)E_x(v)\mathcal{P}_{1x}^{-1}\mathcal{Q}_{1x}uv - \mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}p \\
&+ \frac{1}{Re}\left(\tilde{\mathcal{P}}_{2x}^{-1}\tilde{\mathcal{Q}}_{2x}\mathcal{P}_{2x}^{-1}\mathcal{Q}_{2x} + \tilde{\mathcal{P}}_{2y}^{-1}\tilde{\mathcal{Q}}_{2y}\mathcal{P}_{2y}^{-1}\mathcal{Q}_{2y}\right)u \quad .
\end{aligned}
\tag{5.10}
$$

5.1.3 Temporal Discretization

The momentum equation (5.2) can be written in the following compact form:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{R}(\mathbf{u}, p; t) \quad \text{, where} \quad \mathbf{R}(\mathbf{u}, p; t) = -\nabla p + A(\mathbf{u}; t). \tag{5.11}$$

The term $A = (A_u, A_v)$, with $A_u(u, v; t)$ and $A_v(u, v; t)$, denotes the remaining terms in (5.3) and (5.4) respectively. An explicit, fourth-order accurate Runge-Kutta scheme is used in the temporal discretization of equation (5.11),

$$\mathbf{u}^{n,1} = \mathbf{u}^n, \qquad p^{n,1} = p^n \tag{5.12}$$

$$\mathbf{u}^{n,2} = \mathbf{u}^n + \frac{\Delta t}{2}\mathbf{R}^{n,1}, \tag{5.13}$$

$$\mathbf{u}^{n,3} = \mathbf{u}^n + \frac{\Delta t}{2}\mathbf{R}^{n,2}, \tag{5.14}$$

$$\mathbf{u}^{n,4} = \mathbf{u}^n + \Delta t\mathbf{R}^{n,3}, \tag{5.15}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{\Delta t}{6}(\mathbf{R}^{n,1} + 2\mathbf{R}^{n,2} + 2\mathbf{R}^{n,3} + \mathbf{R}^{n,4}). \tag{5.16}$$

with $t^n = n\Delta t$, $t^{n,1} = t^n$, $t^{n,2} = t^{n,3} = t^n + \Delta t/2$, $t^{n,4} = t^n + \Delta t$, and $\mathbf{R}^{n,\ell} = \mathbf{R}(\mathbf{u}^{n,\ell}, p^{n,\ell}; t^{n,\ell})$, for $\ell = 2, 3, 4$.

The quantities $p^{n,\ell}$, $\ell = 2, 3, 4$ and p^{n+1}, appearing in equations (5.12) to (5.16), are determined by the enforcement of incompressibility on the velocity vectors $\mathbf{u}^{n,\ell}$, $\ell = 2, 3, 4$ and $\mathbf{u}^{n+1}$. Incompressibility is enforced on the intermediate velocity vectors $\mathbf{u}^{n,\ell}$, $\ell = 2, 3, 4$ and on the final solution $\mathbf{u}^{n+1}$ at time level $n + 1$, making use of the pressure correction methodology described in the following subsection.

Stability restrictions are imposed in the proposed numerical scheme, as implied by the standard type CFL condition in (5.17) and the diffusive stability constraint (5.18), proposed and critically examined by [76] and displayed below:

$$\mathrm{CFL} = \|\mathbf{u}\|\frac{\Delta t}{h} \leq 1, \tag{5.17}$$

and

$$\nu \frac{\Delta t}{h^2} \leq (1/2)^d, \tag{5.18}$$

where h is the smallest spatial grid resolution, d is the spatial dimension of the Navier-Stokes equations and ν is the fluid's kinematic viscosity.

5.1.4 Pressure correction method

The procedure followed to enforce the incompressibility condition iteratively by solving a globally defined Poisson-type equation for the computation of pressure updates is described hereupon. Considering the semidiscrete (discrete only in time) form of momentum equations (5.11) and using the divergence free condition at the pressure points of the staggered grid, an equation for every pressure point becomes available. Using the continuity equation, this equation now transforms to a Poisson-type equation for the pressure correction. This Poisson-type equation is discretized deploying a fourth-order accurate solver.

More specifically, by retrieving the equations (5.12) to (5.16) and letting $\mathbf{u}_{old}^{n,\ell}$ be the discrete in time (but continuous in space) solution of (5.2) obtained from the ℓ^{th} stage of the Runge-Kutta method as well as setting $p_{old}^{n,\ell} = p^{n,\ell-1}$ (where $p^{n,\ell-1}$ is the discrete in time pressure from the previous intermediate stage) the corrected pressure at the ℓ^{th} stage comes as

$$p_{new}^{n,\ell} = p_{old}^{n,\ell} + \Delta p. \tag{5.19}$$

An updated value of the velocity $\mathbf{u}_{new}^{n,\ell}$ is defined by

$$\mathbf{u}_{new}^{n,\ell} = \mathbf{u}^n + a_{\ell,\ell-1}\Delta t(-\nabla p_{new}^{n,\ell} + \mathbf{A}(\mathbf{u}_{old}^{n,\ell}; t)), \tag{5.20}$$

Then, the substitution of the relation (5.19) into (5.20) renders

$$\begin{aligned}\mathbf{u}_{new}^{n,\ell} &= \mathbf{u}^n + a_{\ell,\ell-1}\Delta t(-\nabla p_{old}^{n,\ell} - \Delta p + \mathbf{A}(\mathbf{u}_{old}^{n,\ell}; t)) \\ &= \mathbf{u}^n + a_{\ell,\ell-1}\Delta t(-\nabla p_{old}^{n,\ell} + \mathbf{A}(\mathbf{u}_{old}^{n,\ell}; t)) - a_{\ell,\ell-1}\Delta t\nabla\Delta p,\end{aligned} \tag{5.21}$$

which is equivalent to

$$\mathbf{u}_{new}^{n,\ell} = \mathbf{u}_{old}^{n,\ell} - a_{\ell,\ell-1}\Delta t\nabla(\Delta p) \quad . \tag{5.22}$$

Application of the continuity equation (5.1) to $\mathbf{u}_{new}^{n,\ell}$ implies that at the centers of the cells $C_{i,j}$, $i = 1,\ldots,I$, $j = 1,\ldots,J$,

$$\nabla \cdot \mathbf{u}_{new}^{n,\ell} = 0. \tag{5.23}$$

The substitution of (5.22) in (5.23) provides the following Poisson-type equation which stands at the cell centers, for $i = 1,\ldots,N_x$, $j = 1,\ldots,N_y$,

$$\nabla \cdot \nabla(\Delta p)_{i,j} = \frac{1}{a_{\ell,\ell-1}\Delta t}\left(\nabla \cdot \mathbf{u}_{old}^{n,\ell}\right)_{i,j}. \tag{5.24}$$

or expressed in an expanded form

$$\left(\frac{\partial^2(\Delta p)}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2(\Delta p)}{\partial y^2}\right)_{i,j} = \frac{1}{a_{\ell,\ell-1}\Delta t}\left(\nabla \cdot \mathbf{u}_{old}^{n,\ell}\right)_{i,j}. \tag{5.25}$$

The discretization procedure of the discrete Poisson equation (5.25) is thoroughly explained in the following subsection within the multigrid framework.

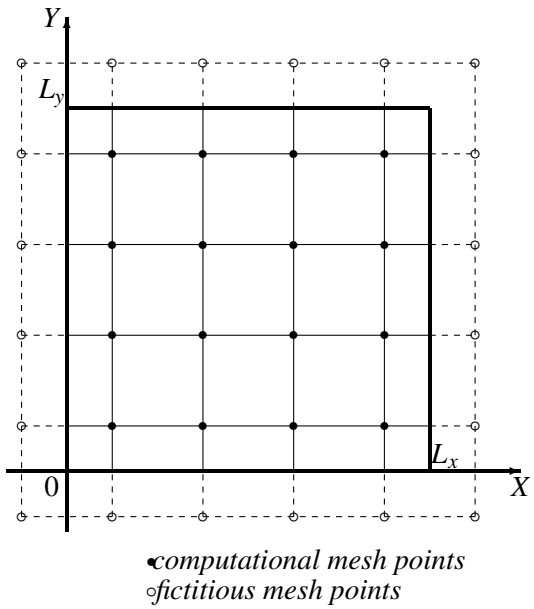

Figure 5-2: Pressure correction computational mesh.

5.2 Compact Finite Difference discretization for the pressure correction

As previously mentioned, the global pressure update process, which is applied for all cells simultaneously, needs a pressure equation solution for each stage, Eqs. (5.12) - (5.16), of the RK4 method.

The above numerical analysis (proposed in [71]) updates the value of the pressure by solving a pressure correction equation (5.26) at each time step, with the pressure correction term Δp defined globally on Ω, and valid at the cell centers M_{ij} ($i = 1, ..., N_x$, $j = 1, ...N_y$) of the prime computational mesh

$$\left(\frac{\partial^2(\Delta p)}{\partial x^2}\right)_{ij} + \left(\frac{\partial^2(\Delta p)}{\partial y^2}\right)_{ij} = f_{ij}, \tag{5.26}$$

where $f_{ij} = f(M_{ij}) = \frac{1}{a_{\ell,\ell-1}\Delta t}(\nabla \cdot \mathbf{u}^{n,\ell}_{old})_{i,j}$.

The pressure p is actually a Lagrange multiplier which confines the velocity field to a divergence free condition and enforces incompressibility. When applying the incompress-

ibility condition, it is also necessary to specify the appropriate boundary conditions for p and Δp functions. The Neumann type BC can be applied using the normal projection for the momentum equations (5.2) on the wall, as in [38], where a unique solution for $t \geq 0$ is provided. In most cases, mixed types of boundary conditions, will effectively yield a well converged and accurate solution [84]. In case of Dirichlet boundary conditions, p is constant on the walls of the domain and therefore $\Delta p = 0$. Under Neumann conditions, $\partial p/\partial n = 0$, which leads to $\partial(\Delta p)/\partial n = 0$, where n is the outward normal vector on the boundary.

The solution $\Delta p(x, y)$ and the right-hand side function $f(x, y)$ of equation (5.26) are assumed to be sufficiently smooth and have continuous partial derivatives; therefore, an approximation of the solution using an appropriate discretization scheme can be reached. In a staggered prime computational grid the solution of equation (5.26) is sought in every cell-center of Ω. The centered grid points $(x_{i-1/2}, y_{i-1/2})$, $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$ inside Ω are considered to be nodal points of a dual computational mesh (x_i, y_i), $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$, while fictitious points denoted by $i = 0, N_x + 1$ and $j = 0, N_y + 1$ correspond to fictitious pressure corrections outside the domain Ω. Fig. 5-2 presents these mesh nodes in the case where $N_x = N_y = 4$.

5.2.1 High order Compact Finite Discretization

This section presents an alternative methodology for the discretization of the pressure correction equation utilizing high order compact finite difference schemes.

The equivalent one-dimensional pressure correction problem of (5.26)

$$\Delta p_i'' = f_i \ , \ i = 1, ..., N \ . \tag{5.27}$$

is considered.

As proposed in [51], the second derivative's value ϕ_i'' at the mesh point x_i of the one dimensional function ϕ can be approximated using the following formula

$$\begin{aligned}\beta\phi''_{i-2} + \alpha\phi''_{i-1} + \phi''_i + \alpha\phi''_{i+1} + \beta\phi''_{i+2} &= c\frac{\phi_{i+3} - 2\phi_i + \phi_{i-3}}{9h^2} + b\frac{\phi_{i+2} - 2\phi_i + \phi_{i-2}}{4h^2} \\ &\quad + a\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2},\end{aligned} \tag{5.28}$$

which involves its neighboring node values. The fourth order of accuracy constraints for the coefficients satisfy the equations $a+b+c = 1+2\alpha+2\beta$ and $a+2^2b+3^2c = \frac{4!}{2!}(\alpha+2^2\beta)$. The above formula simplifies to

$$\phi''_{i-1} + 10\phi''_i + \phi''_{i+1} = 12\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} \quad , \tag{5.29}$$

when $c = b = \beta = 0$ are selected. In order to eliminate the fictitious unknown values ϕ_0, ϕ_{N+1} for $i = 1, N$ the following fourth order interpolation formula

$$\phi_{i-\frac{1}{2}} + \alpha\phi_{i+\frac{1}{2}} = a\phi_{i-1} + b\phi_i + c\phi_{i+1} + d\phi_{i+2}, \tag{5.30}$$

can be used, where $\alpha = free$, $a = \frac{5}{16} - \frac{\alpha}{16}$, $b = \frac{15}{16} + \frac{9\alpha}{16}$, $c = -\frac{5}{16} + \frac{9\alpha}{16}$ and $d = \frac{1}{16} - \frac{\alpha}{16}$.

If $\alpha = 0$ the values ϕ_0 and ϕ_{N+1} can be calculated using the following equations

$$\phi_0 = \frac{16}{5}\phi_{\frac{1}{2}} - 3\phi_1 + \phi_2 - \frac{1}{5}\phi_3 \quad ; \quad \phi_{N+1} = \tfrac{16}{5}\phi_{N+\frac{1}{2}} - 3\phi_N + \phi_{N-1} - \tfrac{1}{5}\phi_{N-2}. \tag{5.31}$$

This formula fits the case where Dirichlet boundary conditions apply, as the values $\phi_{\frac{1}{2}}$ and $\phi_{N+\frac{1}{2}}$ are computed on the boundary. The second derivative values ϕ_0'' and ϕ_{N+1}'' in relation (5.29) can be eliminated using the equation (5.30).

Now, discretized equation (5.29) can be written in the following operator's form

$$(\mathcal{P}\phi'')_i = (\mathcal{Q}\phi)_i \quad , \; i = 1, ..., N \; . \tag{5.32}$$

using the compact finite-difference operators $\mathcal{P}$ and $\mathcal{Q}$. These discrete operators are

$$(\mathcal{P}\phi^{''})_i = \begin{cases} 16\phi^{''}_{1/2} + 35\phi^{''}_1 + 10\phi^{''}_2 - \phi^{''}_3 \,, & i = 1 \\ \phi^{''}_{i-1} + 10\phi^{''}_i + \phi^{''}_{i+1} \,, & i = 2, ..., N-1 \\ -\phi^{''}_{N-2} + 10\phi^{''}_{N-1} + 35\phi^{''}_{N-1} + 16\phi^{''}_N \,, & i = N \end{cases},$$

$$(\mathcal{Q}\phi)_i = \begin{cases} \frac{12}{h^2}(16\phi_{1/2} - 25\phi_1 + 10\phi_2 - \phi_3) \,, & i = 1 \\ \frac{12}{h^2}(\phi_{i-1} - 2\phi_i + \phi_{i+1}) \,, & i = 2, ..., N-1 \\ \frac{12}{h^2}(-\phi_{N-2} + 10\phi_{N-1} - 25\phi_N + 16\phi_{N+1/2}) \,, & i = N \end{cases}.$$

The application of these operators in the one-dimensional pressure correction problem (5.27) results to the following discretization equation

$$(\mathcal{Q}\Delta p)_i = (\mathcal{P}f)_i + O(\Delta^4) \,, \tag{5.33}$$

out of which a fourth order of accuracy compact scheme is obtained.

In the case of Neumann boundary conditions an equivalent interpolation formula (5.30) has the form

$$h(\phi'_{i-\frac{1}{2}} + \alpha\phi'_{i+\frac{1}{2}}) = a\phi_{i-1} + b\phi_i + c\phi_{i+1} + d\phi_{i+2} + 3\phi_{i+3}, \tag{5.34}$$

where $\alpha = free$, $a = \frac{-22+\alpha}{24}$, $b = \frac{17}{24} - \frac{9\alpha}{8}$, $c = \frac{3+9\alpha}{8}$ and $d = \frac{-5-\alpha}{24}$ and $e = \frac{1}{24}$. In order to evaluate the fictitious values ϕ_0 and ϕ_{N+1}, the above formula for $\alpha = 0$ is applied, resulting to

$$\phi_0 = -\frac{12}{11}h\phi'_{\frac{1}{2}} + \frac{17}{22}\phi_1 + \frac{9}{22}\phi_2 - \frac{5}{22}\phi_3 + \frac{1}{22}\phi_4 \tag{5.35}$$

. (5.36)

$$\phi_{N+1} = \frac{12}{11}h\phi'_{N+\frac{1}{2}} + \frac{17}{22}\phi_N + \frac{9}{22}\phi_{N-1} - \frac{5}{22}\phi_{N-2} + \frac{1}{22}\phi_{N-3} \tag{5.37}$$

As in the case of Dirichlet boundary conditions, so in Neumann, a similar fourth order of accuracy compact discretization scheme emerges.

Robin or mixed type boundary conditions can be treated similarly by eliminating all fictitious node values using a combination of the approximation formulas (5.31),(5.35) and (5.37).

In the development of a two dimensional fourth order finite difference compact discretization scheme for the pressure correction equation (5.26), the compact finite difference operators $\mathcal{P}$ and $\mathcal{Q}$ are applied as

$$\mathcal{P}^y\mathcal{Q}^x\Delta p_{ij} + \mathcal{P}^x\mathcal{Q}^y\Delta p_{ij} = \mathcal{P}^y\mathcal{P}^x f_{ij} + O(\Delta^4)\ , \tag{5.38}$$

where $\mathcal{P}^x$, $\mathcal{Q}^x$, $\mathcal{P}^y$ and $\mathcal{Q}^y$ are the corresponding operators for each partial derivative's direction. The above relation is valid at every grid point (x_i, y_j) and, in addition, $O(\Delta^4)$ denotes the truncated terms of the order of $O(\Delta x^4 + \Delta y^4)$. After dropping the $O(\Delta^4)$ term, the fourth order compact discretization scheme of the pressure correction equation can be expressed as

$$\mathcal{T}\Delta p_{ij} = \mathcal{S} f_{ij} \tag{5.39}$$

The expanded form of operators $\mathcal{T}$ and $\mathcal{S}$ is

$$\begin{aligned} &a(\Delta p_{i+1,j+1} + \Delta p_{i+1,j-1} + \Delta p_{i-1,j+1} + \Delta p_{i-1,j-1}) + 2b(\Delta p_{i,j+1} + \Delta p_{i,j-1}) + \\ &2c(\Delta p_{i+1,j} + \Delta p_{i-1,j}) - 20a\Delta p_{i,j} \ = \ \Delta x^2(8f_{i,j} + f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}) \end{aligned} \tag{5.40}$$

with $a = 1 + \gamma^2$, $b = 5 - \gamma^2$ and $c = 5\gamma^2 - 1$ and $\gamma = \Delta x/\Delta y$ the mesh ratio, for all interior nodes $i = 2, ..., N_x - 1$, $j = 2, ..., N_y - 1$ (see Fig. 5-3(a)). Equation (5.40) formulates the fourth order accurate discretization of the interior scheme,[49]. Closure schemes have to be derived at the four edges of the domain. Equations corresponding to pressure nodes in

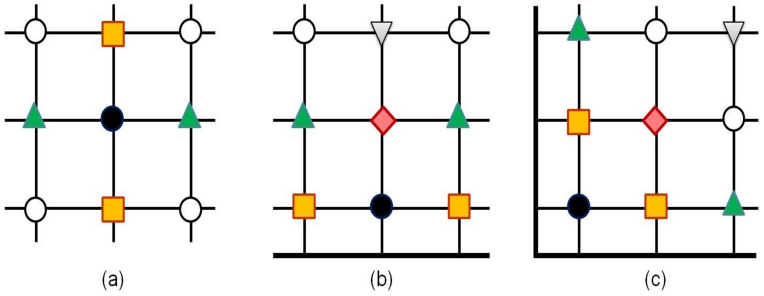

Figure 5-3: The compact finite difference stencil ; (a) interior weight unknown nodes, (b) boundary weight unknown nodes and (c) boundary corner weight unknown nodes .

the vicinity of the bottom boundary are formed as

$$\begin{aligned} &5d(\Delta p_{i-1,1} + \Delta p_{i+1,1}) + 10a(\Delta p_{i-1,2} + \Delta p_{i+1,2}) - a(\Delta p_{i-1,3} + \Delta p_{i+1,3}) - 10g\Delta p_{i,1} + \\ &2c(10\Delta p_{i,2} - \Delta p_{i,3}) = \frac{\Delta x^2}{12}(350 f_{i,1} + 100 f_{i,2} - 10 f_{i,3} + 35(f_{i-1,1} + f_{i+1,1}) + \\ &10(f_{i-1,2} + f_{i+1,2}) - (f_{i-1,3} + f_{i+1,3})) + A(\Delta p, f; h), \end{aligned} \tag{5.41}$$

with $d = 7 - 5\gamma^2$ and $g = 7 - 25\gamma^2$ for $i = 2, ..., N_x - 1$ based on Dirichlet boundary conditions (see Fig. 5-3(b)). The grid function $A(\Delta p, f; h)$ involves known values on the boundary of Δp and the right hand side function f. A similar formulation stands for the equations near the upper, left and right boundaries. Additional closures are required for the corners of the domain. The equation corresponding to the bottom-left boundary corner is

$$\begin{aligned} &a(-875\Delta p_{1,1} + 100\Delta p_{2,2} + \Delta p_{3,3}) + 5d(10\Delta p_{2,1} - \Delta p_{3,1}) + 5e(10\Delta p_{1,2} - \Delta p_{1,3}) - \\ &10a(\Delta p_{3,2} + \Delta p_{2,3}) = \frac{\Delta x^2}{12}(1225 f_{1,1} + 100 f_{2,2} + f_{3,3} + 350(f_{2,1} + f_{1,2}) - \\ &35(f_{3,1} + f_{1,3}) - 10(f_{3,2} + f_{2,3})) + B(\Delta p, f; h) \end{aligned} \tag{5.42}$$

with $e = 7\gamma^2 - 5$ and $B(\Delta p, f; h)$ again, a function of known values on the boundary, (see Fig. 5-3(c)). The other three corner equations are similarly derived.

Remark 5.2.1 *The discretization scheme (5.39) is equivalent to the fourth-order finite difference compact scheme (2.48), which is presented in Chapter 2.*

5.2.2 Discretization of the right hand side

On the staggered grid arrangement, the right-hand side of equation (5.39) is evaluated for every grid node equation (5.40) using the following one-dimensional compact fourth order approximation formula, available in [24],

$$h(\alpha\phi'_{i-1} + \phi'_i + \alpha\phi'_{i+1}) = a(\phi_{i+\frac{1}{2}} - \phi_{i-\frac{1}{2}}) + \frac{b}{3}(\phi_{i+\frac{3}{2}} - \phi_{i-\frac{3}{2}}), \tag{5.43}$$

where $a = (9 - 6\alpha)/8$ and $b = (-1 + 22\alpha)/8$. This relation involves derivative values ϕ' computed at the cell centers for staggered grids and function values ϕ computed at the corresponding cell edges. Gathering the terms at the right-hand side $b_{i,j}$ of (5.40) as

$$\begin{aligned}
b_{i,j} &= \Delta x^2(8f_{i,j} + f_{i+1,j} + f_{i,j+1} + f_{i-1,j} + f_{i,j-1}) \\
&= \Delta x^2[8(\nabla\cdot\mathbf{u}^{n,\ell}_{old})_{i,j} + (\nabla\cdot\mathbf{u}^{n,\ell}_{old})_{i-1,j} + (\nabla\cdot\mathbf{u}^{n,\ell}_{old})_{i+1,j} + (\nabla\cdot\mathbf{u}^{n,\ell}_{old})_{i,j-1} + (\nabla\cdot\mathbf{u}^{n,\ell}_{old})_{i,j+1}] \\
&= \frac{\Delta x^2}{a_{l,l-1}\Delta t}(8\partial_x u_{i,j} + 8\partial_y v_{i,j} + \partial_x u_{i+1,j} + \partial_y v_{i+1,j} \\
&+ \partial_x u_{i,j+1} + \partial_y v_{i,j+1} + \partial_x u_{i-1,j} + \partial_y v_{i-1,j} + \partial_x u_{i,j-1} + \partial_y v_{i,j-1}) \\
&= \frac{\Delta x^2}{a_{l,l-1}\Delta t}(\overbrace{\partial_x u_{i-1,j} + 8\partial_x u_{i,j} + \partial_x u_{i+1,j}} + \overbrace{\partial_y v_{i,j+1} + 8\partial_y v_{i,j} + \partial_y v_{i,j+1}} \\
&+ \partial_x u_{i,j-1} + \partial_x u_{i,j+1} + \partial_y v_{i,j-1} + \partial_y v_{i,j+1})
\end{aligned} \tag{5.44}$$

and applying the relation (5.43) for $\alpha = 0$ and $\alpha = \frac{1}{8}$, yields:

$$b_{i,j} = \frac{\Delta x}{a_{l,l-1}\Delta t 12}\left(2\begin{bmatrix}-7\\-99\\99\\7\end{bmatrix}^\top \boldsymbol{s} + \begin{bmatrix}1\\-27\\27\\-1\end{bmatrix}^\top \boldsymbol{t}\right) \quad , \tag{5.45}$$

where

$$\boldsymbol{s} = \begin{bmatrix} u_{i-3/2,j} + \gamma v_{i,j-3/2} \\ u_{i-1/2,j} + \gamma v_{i,j-1/2} \\ u_{i+1/2,j} + \gamma v_{i,j+1/2} \\ u_{i+3/2,j} + \gamma v_{i,j+3/2} \end{bmatrix} \quad , \quad \boldsymbol{t} = \begin{bmatrix} u_{i-3/2,j-1} + u_{i+3/2,j+1} + \gamma(v_{i-1,j-3/2} + v_{i+1,j-3/2}) \\ u_{i-1/2,j-1} + u_{i+1/2,j-1} + \gamma(v_{i-1,j-1/2} + v_{i+1,j-1/2}) \\ u_{i+1/2,j-1} + u_{i+1/2,j-1} + \gamma(v_{i-1,j+1/2} + v_{i+1,j+1/2}) \\ u_{i+3/2,j-1} + u_{i+3/2,j-1} + \gamma(v_{i-1,j+3/2} + v_{i+1,j+3/2}) \end{bmatrix} \quad . \quad (5.46)$$

The values in the previous relation correspond to the values of the vector $\mathbf{u}_{old}^{n,\ell}$ on the staggered grid and are already available from the previous time step.

Right-hand side values close to the boundary are evaluated in a similar manner with an appropriate modification of the above procedure, though taking into account the following fourth order approximation formula

$$h(\alpha\phi'_{i-1} + \phi'_i + \alpha\phi'_{i+1}) = a\phi_{i-\frac{1}{2}} + b\phi_{i+\frac{1}{2}} + c\phi_{i+\frac{3}{2}} + d\phi_{i+\frac{5}{2}} + e\phi_{i+\frac{7}{2}} \ , \quad (5.47)$$

where $a = (-11 - 46\alpha)/12$, $b = (17 + 202\alpha)/24$, $c = (3 - 66\alpha)/8$, $d = 5(-1 + 22\alpha)/24$ and $(1 - 22\alpha)/24$, eliminating fictitious mesh node values in each spatial direction.

The discretization of the pressure correction equation with the fourth order compact difference scheme provides a sparse linear system. In case of realistic applications where fine discretizations are required, it is also large. In such a system, the theoretical analysis, along with the numerical experiments in previous chapters, suggest the employment of an iterative method coupled with a multigrid method for an efficient solution process.

5.3 Numerical Results

The numerical experiments presented in this section were conducted in order to demonstrate the high-order of accuracy of the solver. The incompressible Navier-Stokes equations are being solved over equal and unequal mesh-size discretizations using multigrid techniques. The effectiveness of the proposed solver is under review for a set of steady and unsteady flow problems. Comparison results between the numerical method and [71] without the multigrid technique are obtained for the driven cavity flow. Additionally, in order to validate that the multigrid technique using partial-coarsening is preferable to that of full-coarsening in certain cases, the execution time and the L_2 error norm estimates of these strategies are documented in the Stokes boundary layer test problem. The $V-cycle$ multigrid acceleration with HOC discretization is implemented for the pressure correction equation in every case. The iterative process recurs until a coarse grid of 4×4 computing cells is reached. The Horizontal Zebra Gauss-Seidel solver is preferred as a smoother for the resulted pressure-correction linear system in every intermediate levels, including the coarsest grid. Two presmoothing and one postsmoothing steps are applied in each level. Multigrid integrid operations use the BCP prolongation and MR restriction operators for an optimal multigrid technique.

In order to retain the high-order of accuracy in time and avoid nonlinear instabilities in the numerical experiments, the computations use CFL numbers under the stability limit of the Runge-Kutta method. All single core numerical tests were performed on an OracleFire X2200M2 machine with 4GB RAM and two dual core Opteron@ 3.0GHz processors. The operation system installed is Oracle's Linux 6.3 and the implementations were developed in double precision Fortran code. All basic linear algebra operations were performed using subroutines fromthe Lapack [45] scientific library. It was detected that the divergence free condition on the discrete level is enforced to machine accuracy (less than 10^{-16}). The termination criteria in the solution of the pressure correction equation is dependent upon the residual of the finest grid in L_2 norm; in fact, when the norm takes a value less than 10^{-8} in

every simulations.

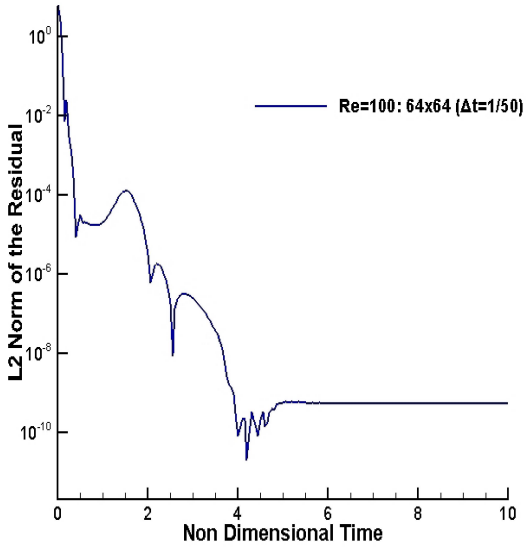

Figure 5-4: Residuals **R** in the L_2 norm of the new method; kovasznay flow (Re=100).

5.3.1 Kovasznay flow

Initially, a problem introduced by Kovasznay (see [34]) is considered, modeling laminar flow behind a two dimensional grid. Although this test case does not exhaust the resolving capabilities of the current methodology, it is suitable as a typical example to demonstrate the spatial accuracy of the method. The analytical solution of the velocity vector $\mathbf{u} = (u, v)$ and pressure p is given by

$$\begin{aligned} u &= 1 - e^{\lambda x}\cos(2\pi y) \\ v &= \frac{\lambda}{2\pi} e^{\lambda x}\sin(2\pi y) \\ p &= p_0 - \tfrac{1}{2}e^{2\lambda x}, \end{aligned} \tag{5.48}$$

where p_0 is an arbitrary constant and the parameter λ is given in terms of the Reynolds Re number by

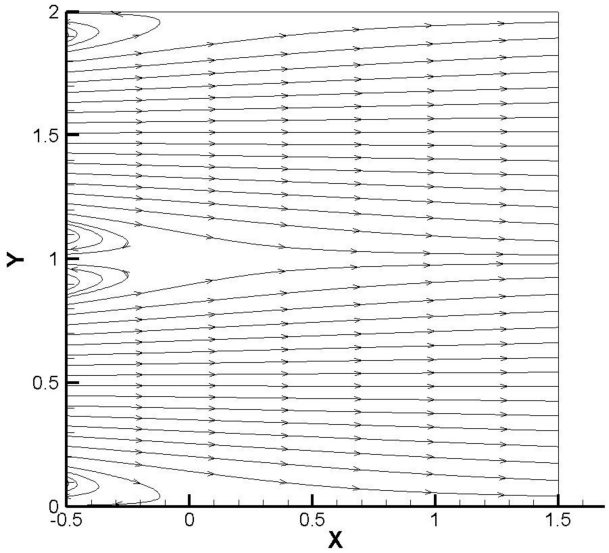

Figure 5-5: Streamlines of the approximated velocity on a 64x64 grid; Kovasznay flow (Re=100).

$$\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2} \quad < \quad 0 \tag{5.49}$$

This flow problem is solved in the domain $\Omega = (-0.5, 1.5) \times (0, 2)$ and when the Reynolds number equals 100. Dirichlet boundary conditions based on the exact solution are considered. In order to confirm the accuracy of the method, the L^2-norm error estimates and the associated order of convergence are presented in Table 5.1. The time step Δt was determined using $CFL = 0.75$. In every level, the desired order of accuracy is reached for both velocity and pressure. The numerical approximation solutions for the velocity $\mathbf{u} = (u, v)$ and pressure p are obtained using the multigrid Navier-Stokes solver. These approximations are denoted by $\mathbf{u}^*$ and p^*, respectively. It is observed that the combination of the multigrid technique with the fourth order compact finite difference scheme, retains the fourth order of the spatial accuracy. Fig. 5-5 depicts the streamlines of the approximated velocity on a 64×64 grid. Additionally, the time table of the value of residuals $\mathbf{R}$ with $\Delta t = 2 \times 10^{-2}$ (Fig. 5-4) indicates that steady-state solution was reached after $T = 5$ units

of time.

Table 5.1: Convergence estimates for the Kovasznay flow

grid size	$\|\mathbf{u}-\mathbf{u}_*\|_{L_2}$		$\|p-p_*\|_{L_2}$	
	Error	Order	Error	Order
8	4.51e-2	-	5.13e-2	-
16	3.79e-3	3.57	6.40e-3	3.01
32	2.23e-4	4.09	5.04e-4	3.67
64	1.41e-5	3.98	3.50e-5	3.85
128	8.87e-7	3.99	2.46e-6	3.92

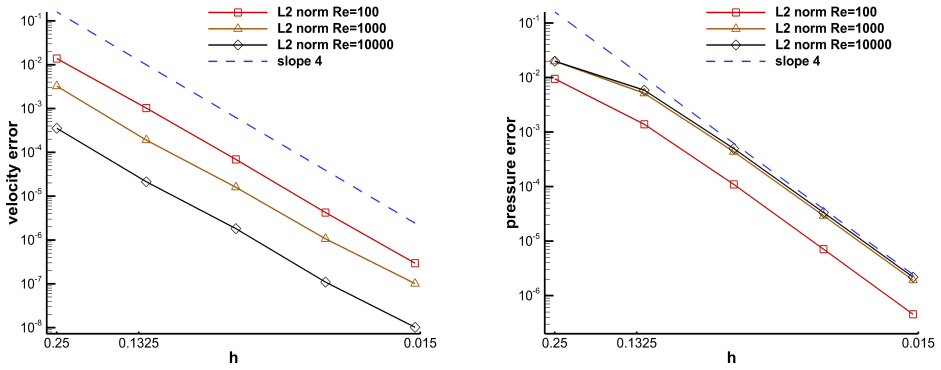

Figure 5-6: Convergence rates of velocity and pressure error L_2-norm versus the grid-size (h) at $T = 5$; Taylor vortex.

5.3.2 Taylor vortex

The Taylor vortex problem [53] is a well-known example of the unsteady incompressible Navier-Stokes equations, which has the following exact solution

$$\begin{aligned} u &= -\cos(\pi x)\sin(\pi y)e^{\frac{-2\pi^2 t}{Re}} \\ v &= \sin(\pi x)\cos(\pi y)e^{\frac{-2\pi^2 t}{Re}} \\ p &= \frac{1}{4}(cos(2\pi x)+\cos(2\pi y))e^{\frac{-2\pi^2 t}{Re}}, \end{aligned} \tag{5.50}$$

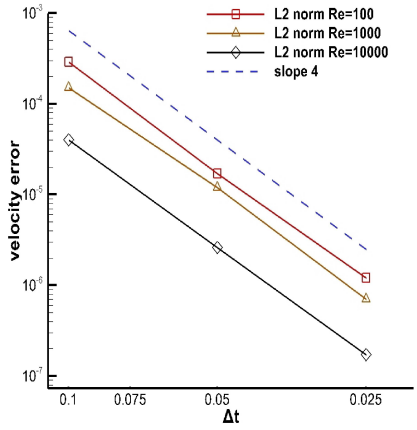

Figure 5-7: Temporal accuracy in L_2-norm of the velocity at $T = 5$; Taylor vortex.

where Re is the Reynolds number. In order to confirm the spatial and temporal accuracy of the numerical method for a wide range of Reynolds numbers, the Taylor vortex problem is solved in the square domain $[-1, 1]^2$. Periodic boundary conditions and initial condition are imposed based on the exact solution. The convergence rates of the computed solutions for different mesh sizes (4 up to 64 cells in each direction) and for Reynolds numbers 10^2, 10^3, and 10^4 are shown in Fig. 5-6. For all simulations the time step was determined for $CFL = 0.75$. The convergence rates for velocity and pressure error in L_2-norm are evaluated at the final time $T = 5$. As it is observed these rates are of order 4 for all test cases. As frequently detected in this test problem, the velocity magnitude error tends to drop more abruptly when increasing the Re numbers compared to the pressure error.

5.3.3 Oseen vortex decay

The temporal and spatial accuracy of the numerical scheme is evaluated next in an unsteady flow problem. The decay of an ideal vortex (Oseen vortex) with uniform pressure and initial velocity distribution

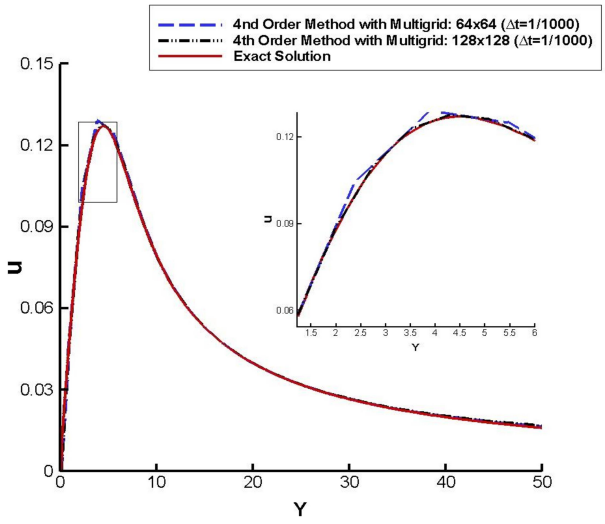

Figure 5-8: Comparison of the exact and computed u-velocity for 64x64 and 128x128 point grids at $T = 4$; Oseen vortex decay.

$$v_\theta(r, t = 0) = \frac{\Gamma}{2\pi r}, \tag{5.51}$$

where Γ defines the strength of the vortex and r the distance from the origin. This vortex decays under the action of viscous dissipation and the velocity distribution at time t is given by the following exact solution [58]

$$v_\theta(r, t) = \frac{\Gamma}{2\pi r}(1 - e^{-\frac{r^2 Re}{4t}}). \tag{5.52}$$

The time-dependent flow with the initial condition given by (5.52) was computed for $\Gamma = 5$. Fig. 5-8 presents the numerical solution of (5.52) obtained from equally spaced, Cartesian grids with 64x64 and 128x128 nodes, until final time $T = 4$.

It can be seen (see Fig. 5-8) that the computed velocities approximate the exact solution better as the resolution increases. Furthermore (see Fig. 5-9), fourth order spatial and temporal accuracy is accomplished.

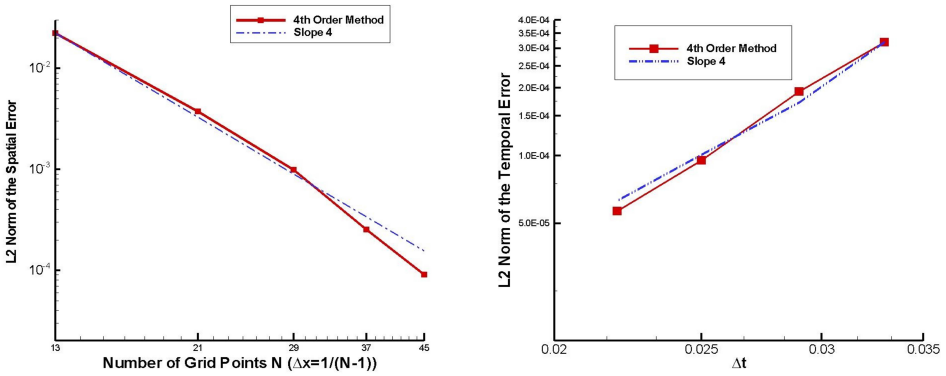

Figure 5-9: Comparison of the L_2 norm of spatial (left) and temporal (right) errors at $T = 1$; Oseen vortex decay.

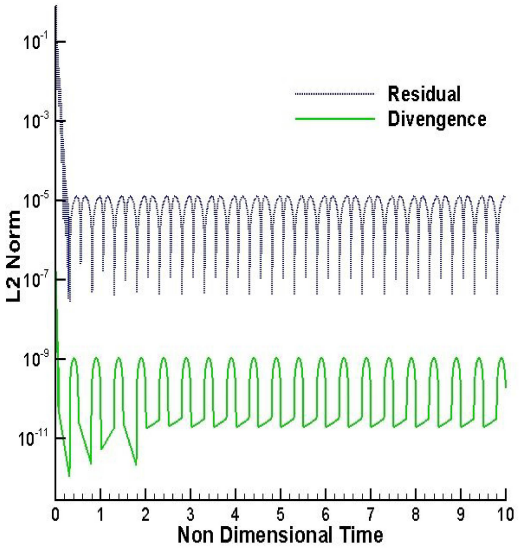

Figure 5-10: Divergence error and Residual **R** in the L^2 norm of the new method for 64x64, Δt=0.002; Stokes oscillatory plate (Re=1).

5.3.4 Stokes oscillating plate

This flow problem assesses the efficiency of the proposed method when the partial semi-coarsening multigrid strategy is applied, compared to the full coarsening approach.

Although uniform finite difference discretizaton using equal meshsizes in both x and y directions can be effortlessly implemented, there are cases where the use of unequal mesh-

sizes in different coordinate directions proves to be more cost effective. In such cases, the modeled physical quantity is unevenly distributed in different directions. The flow over an infinite oscillating plate (Stokes problem) is an unsteady incompressible flow problem with an exact solution that demands finer resolution than normally to the wall direction. The flow over the plate begins after the plate starts an oscillatory motion (on the plate $y = 0$) with a speed $u(x, 0, t) = u_0 \cos \Omega t$.

The exact solution [58] of the time-dependent velocity is:

$$u(x, y, t) = e^{-y\sqrt{\frac{\Omega Re}{2}}} \cos(\Omega t - y\sqrt{\frac{\Omega Re}{2}}). \tag{5.53}$$

The parameter values $u_0 = 1$ and $\Omega = 2\pi$ are used. Periodic boundary conditions are imposed in the streamwise direction and on the plate $v = \partial p/\partial n = 0$ is considered. The discretization uses a mesh ratio $\gamma \geq 1$, because the exact solution alters in the $y - direction$ only. Comparison results are summarized in Tables 5.2-5.3 for the implementation of the two multigrid techniques with partial semi-coarsening and full-coarsening strategies. When $Re = 1$, the diffusive time constraint must be satisfied. Increasing the anisotropy $\Delta x \gg \Delta y$, allows the temporal interval to increase. In this case, it is interesting to note that the behaviour of the problem is similar to the one-dimensional and that the time-step restriction is dictated by the equation (5.18) for $d = 1$. A fixed time interval was used ($\Delta t = 2 \times 10^{-3}$) in every spatial discretization. In Fig. 5-10, the solution converges to a steady repetitional pattern with reasonable small values in the time table of the residuals $\mathbf{R}$ and the velocity divergence, with $\Delta t = 2 \times 10^{-3}$.

Table 5.2: Total CPU time in seconds for Stokes oscillating plate.

N_y	$N_x = 8$	$N_x = 16$	$N_x = 32$	$N_x = 64$
8	3.5	-	-	-
16	7.20	11.49	-	-
32	13.97	22.61	35.22	-
64	23.25	44.23	81.10	140

Table 5.2 includes the total execution time required to converge to the solution at time $T = 1/4$ after the time periodic solution has been reached, on several equal and unequal meshsize grids. Execution time measurements indicate that the semi-coarsening multigrid technique is increasingly preferable in terms of CPU cost than the full-coarsening one for the increasing anisotropy ($\Delta y \ll \Delta x$) cases. Table 5.3 presents the L_2 error norm of the solution together with the convergence rates computed by the proposed method. Decreasing Δy does not always lead to a reasonable increase of the solution's accuracy. The identical L_2 norm error estimates are obtained by keeping the value of Δy fixed.

Table 5.3: L_2 error and convergence rate estimates for Stokes oscillating plate.

N_y	$N_x = 8$	$N_x = 16$	$N_x = 32$	$N_x = 64$	Order
8	3.17e-3	-	-	-	-
16	4.61e-4	4.61e-4	-	-	2.781
32	3.79e-5	3.79e-5	3.79e-5	-	3.604
64	2.61e-6	2.61e-6	2.61e-6	2.61e-6	3.860

5.3.5 Driven cavity flow

The steady-state driven-cavity flow problem [91], [30], [7] has been widely used as a validation example in numerical methods solving the incompressible Navier-Stokes equations. The flow is contained within a unit square cavity and no-slip boundary conditions are imposed on every wall, except the upper wall, where $u = 1$ and $v = 0$. The boundary condition for the pressure in every wall is obtained when considering zero normal gradient ($\partial p/\partial n = 0$).

Since this case has been solved quite a few times, there is a great deal of data to compare with. A good set of data for comparisons is that of [91] as it includes tabular results for various Reynolds numbers. Numerical solutions for the driven-cavity flow were obtained using different Reynolds numbers and compared to the calculations in [91] as presented in Fig. 5-11 and 5-12. The results herein are in acceptable agreement to those by Ghia et

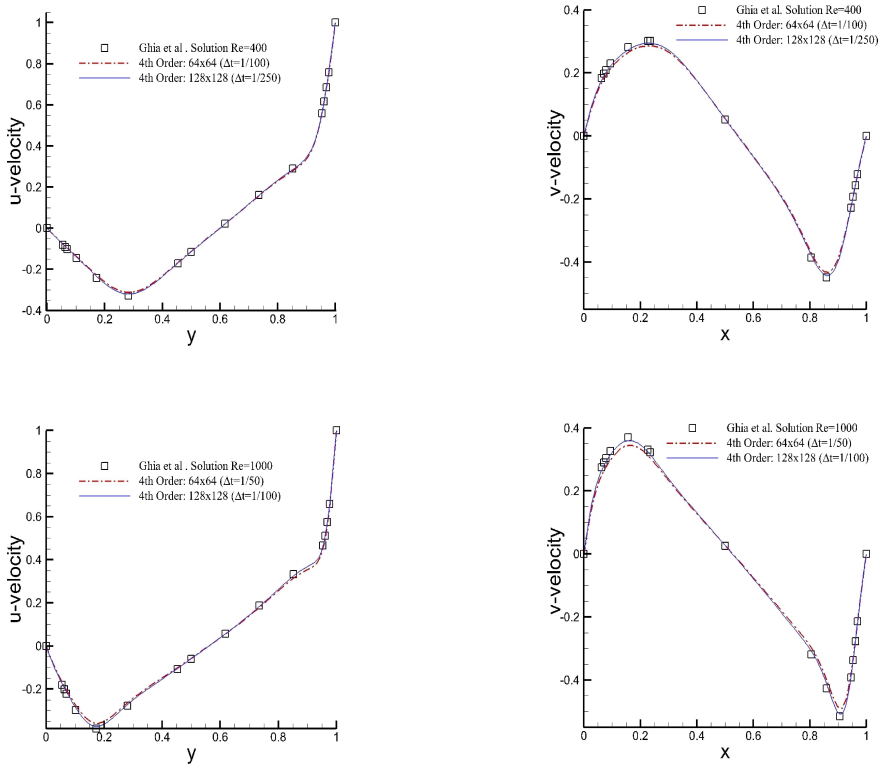

Figure 5-11: Comparison of velocity approximations with reference solutions by Ghia et al. [91] for Re = 400 and 1000.

al. [91]. It is noted that, while Reynolds numbers remain under 1000, a 64x64 grid size is acceptable for a sufficiently accurate solution. When increasing the Reynolds number, a finer grid is required (see Fig.5-12). The velocity magnitude distribution and vectors of uniform lengthscale for Re={400, 1000, 3200, 5000} are presented in Fig. 5-13. The flow pattern is also in agreement with those in [91], with the recirculation in the middle, driving two left-spinning vortices in the lower right and left corners of the domain for lower Reynolds numbers and an extra left-spinning vortex on the upper left corner for higher Reynolds numbers. As expected - and already well documented in the literature - the corner vortices increase in size as the Reynolds number increases, with a repetitive formation of additional vortices in the three corners (upper left, lower right, lower left). In higher Reynolds numbers, the resulting flow pattern implies an added flow instability

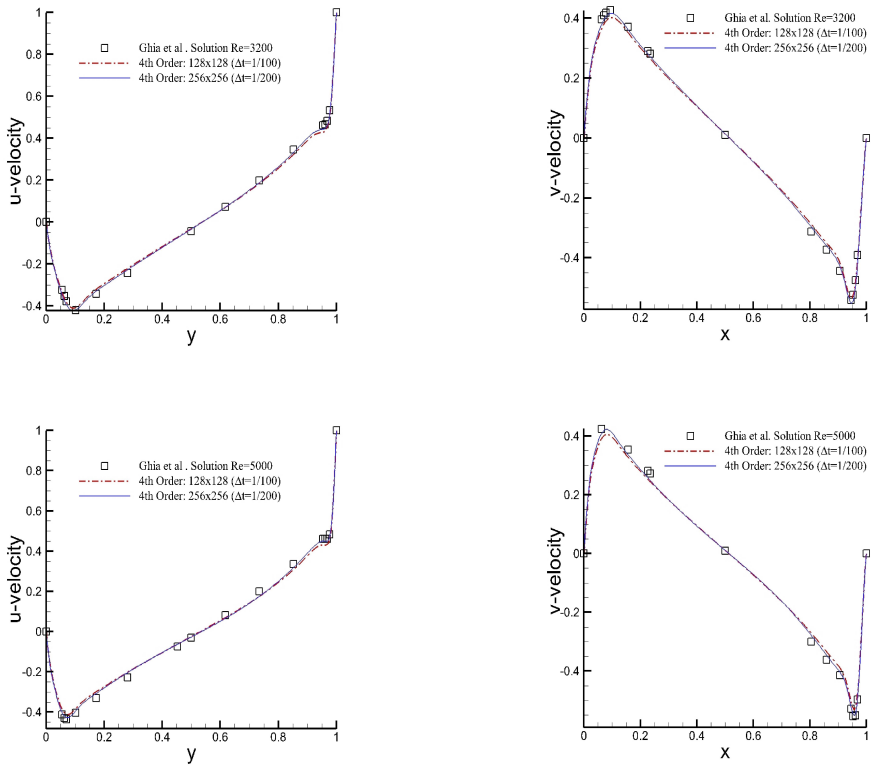

Figure 5-12: Comparison of velocity approximations with reference solutions by Ghia et al. [91] for Re = 3200 and 5000.

due to Reynolds number increase. To ensure that a steady state solution has been reached, the results in figures Fig. 5-11–5-13 were obtained for non-dimensional times $T = 35$ for $Re = 400$ and 1000, $T = 120$ for $Re = 3200$ and 5000. For time values greater than $T = 20$ for $Re = 400$, $T = 30$ for $Re = 1000$, T=85 for $Re = 3200$ and $T = 110$ for $Re = 5000$, no apparent differences where noticed by further propagating the solution. This was also established by examining graph in Figure 5-14, that shows the residuals **R** history with respect to non-dimensional time different temporal discretizations. The L_2-norm minimizes to values less that 5×10^{-10}, which indicates a solution reaching a steady state status.

Tables 5.4-5.7 display the total CPU time needed to converge to a steady state solution using the multigrid solver. For good practice comparison, the same values are also pre-

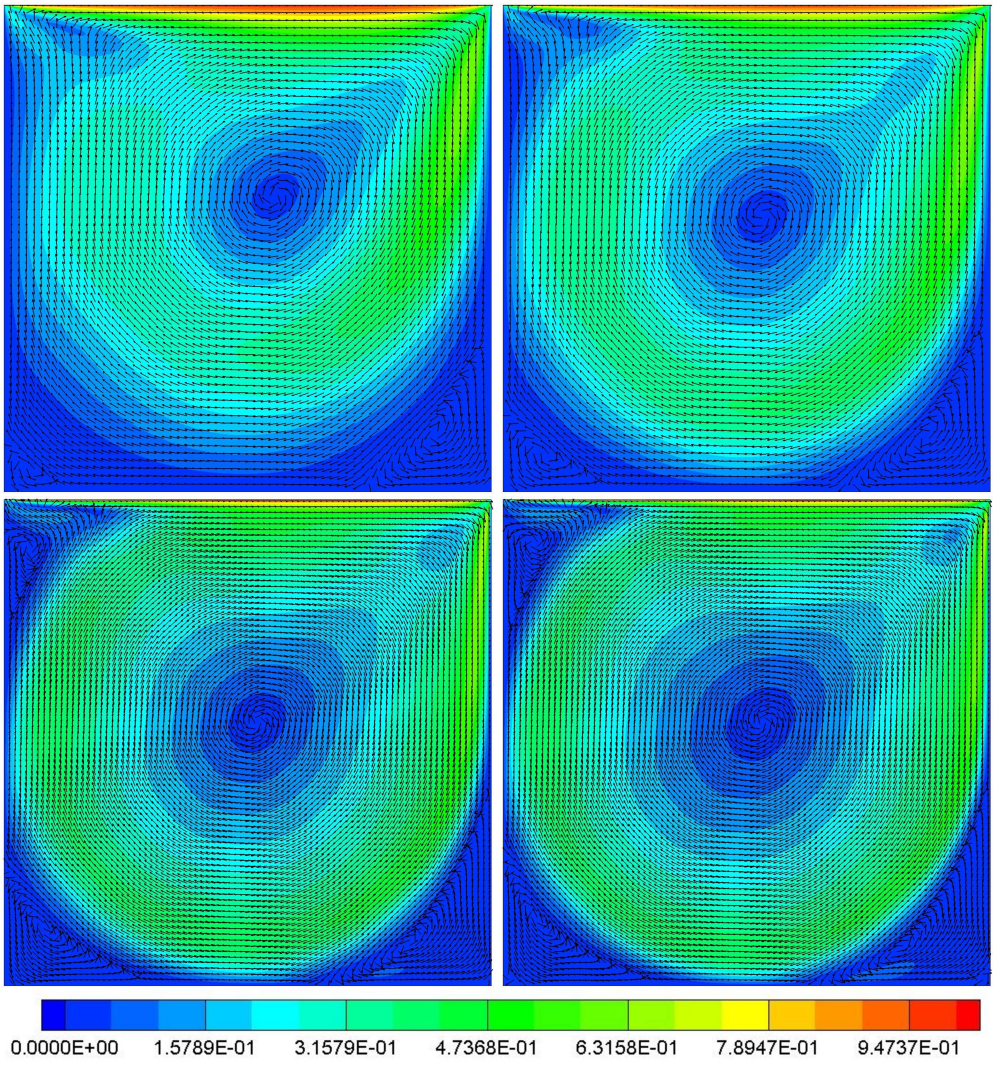

Figure 5-13: Velocity magnitude distribution and vectors of uniform lengthscale for Re={400, 1000, 3200, 5000}. The flow patterns agree well with those in [91], with the recirculation in the middle driving two left-spinning vorticies in the lower right and left corners of the domain for lower Reynolds numbers and an extra left-spinning vortex on the upper left corner for higher Reynolds numbers.

Table 5.4: Total CPU time and time per Stage in seconds for Re=400 and T=25.

Grid size	Δt	Multigrid solver		GMRES solver [71]	
		Total time	Time/Stage	Total Time	Time/Stage
64	1/100	74.82	0.007	182	0.018
128	1/250	714	0.029	3942	0.158
256	1/1000	11328	0.113	142300	1.423

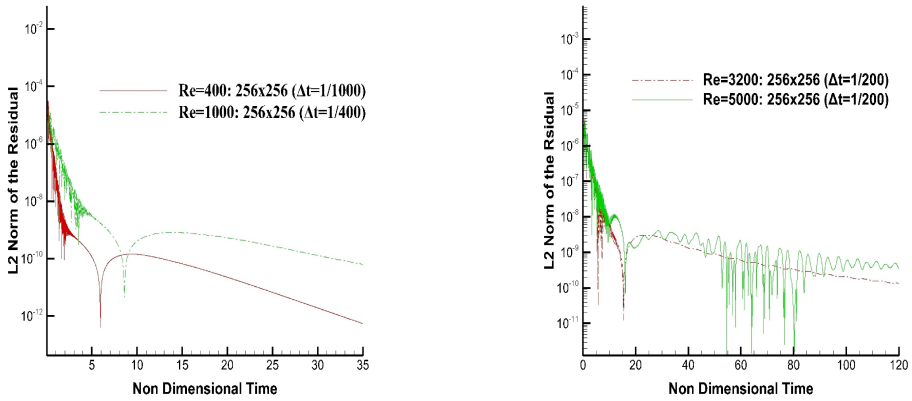

Figure 5-14: Residuals **R** in the L_2 norm of the new method for Reynolds 400 up to 5000; Driven cavity flow.

Table 5.5: Total CPU time and time per Stage in seconds for Re=1000 and T=35.

Grid size	Δt	Multigrid solver		GMRES solver [71]	
		Total time	Time/Stage	Total Time	Time/Stage
64	1/50	60.19	0.009	128	0.018
128	1/100	461	0.0329	2466	0.176
256	1/400	6993	0.129	85176	1.521

Table 5.6: Total CPU time and time per Stage in seconds for Re=3200 and T=100.

Grid size	Δt	Multigrid solver		GMRES solver [71]	
		Total time	Time/Stage	Total Time	Time/Stage
64	1/50	144	0.007	358	0.018
128	1/100	1088	0.027	6415	0.160
256	1/200	8667	0.108	116320	1.454

Table 5.7: Total CPU time and time per Stage in seconds for Re=5000 and T=120.

Grid size	Δt	Multigrid solver		GMRES solver [71]	
		Total time	Time/Stage	Total Time	Time/Stage
64	1/50	187	0.008	429	0.018
128	1/100	1445	0.030	7719	0.161
256	1/200	12895	0.134	140165	1.460

sented for the numerical method proposed in [71], where the ILUT preconditioned Krylov subspace iterative solver GMRES was employed in the numerical solution of the linear system. In addition, the average CPU time needed to perform a single stage of the Runge-Kutta method is included. Each Table refers to a specific Reynolds number.

This performance investigation unveiled the superiority of the proposed numerical scheme in terms of execution time. The ratio of the computational time between the two methods increases along the order of the linear systems.

Table 5.8 presents the number of $V-cycles$ required for the pressure correction subproblem solution per time-step. It is interesting to notice that, when the overall solution converges (approaching the steady state) the $V-cycles$ reduce to minimum (1), thus accelerating significantly the pressure correction procedure. It was observed that the execution time per Stage remains stable in case of the non-multigrid solver.

At the beginning of the overall solution process, with increasing Reynolds numbers, the multigrid solver converges faster (see Table 5.8), i.e. four $V-cycles$ for $Re = 400$ and 1000 and three cycles for greater Reynolds numbers are needed. That can be explained by the fact that large Reynolds numbers lead to more oscillatory solutions and thus to more oscillatory errors. The multigrid technique is highly capable of solving such kind of problems, because the multigrid smoothing procedures can quickly eliminate the oscillatory error components, while smooth modes are slowly damped.

5.3.6 Double Shear Layer Flow

This test case evaluates the ability of the computational algorithm to resolve unsteady flow features that require high resolution. It has been well documented in the literature [69], [23] that under-resolved numerical simulations of the evolution of a 2D vortex street for an incompressible fluid with double-periodic boundary conditions can produce artifacts characterized as 'spurious eddies'. In particular, the time-dependent flow with initial condition

Table 5.8: V-cycles required for convergence against time duration.

Time (sec)	Reynolds Number			
	400	1000	3200	5000
[0 , 0.306)	4	4	3	3
[0.306 , 0.365)	3	4	3	3
[0.365 , 1.09)	3	3	3	3
[1.09 , 1.945)	2	3	3	3
[1.945 , 3.10)	2	2	3	3
[3.10 , 3.89)	2	2	3	2
[3.89 , 11.73)	2	2	2	2
[11.73 , 23.53)	1	2	2	2
[23.53 , 39.65)	1	1	2	2
[39.65 , 106)	1	1	1	2
[106 , 120]	1	1	1	1

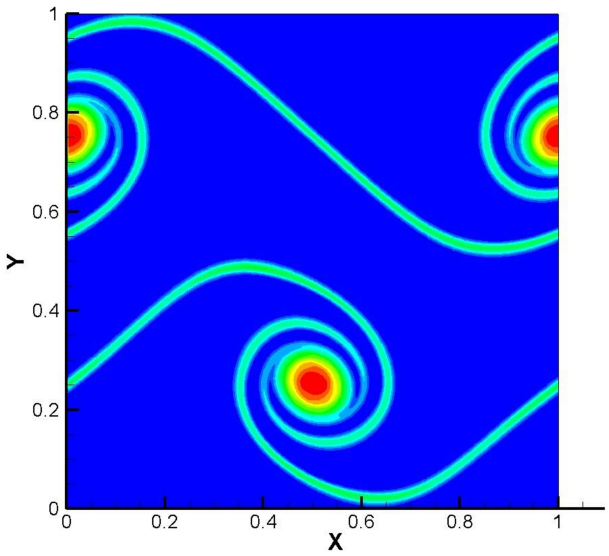

Figure 5-15: The computed vorticity field on a 256×256 grid at T=1; Shear Layer.

$$u(x, y, t = 0) = \begin{cases} \tanh(\delta(y - 0.25)) & \text{if } y \leq 0.5 \\ \tanh(\delta(0.75 - y)) & \text{if } y > 0.5 \end{cases},$$

that represents a shear layer (with δ determining its thickness), which evolves into a periodic eddy pattern, when it is perturbed in the y direction. Using a sinusoidal perturbation of the form $v(x, t = 0) = v' sin(2\pi x)$ as in [23], where v' is the perturbation amplitude, the ini-

tial shear layer and the perturbation converges to a solution that takes the form of a regular periodic vortex street. A unit-square domain [0,1]x[0,1] was preferred for the simulation over a 256×256 grid. Periodic boundary conditions were applied in every direction and a Reynolds number of $Re = 10000$, with a thickness parameter $\delta = 100$ and a perturbation amplitude $v' = 0.05$. Second and higher order schemes have been tested in this problem in [28], using the artificial compressibility (AC) method by Chorin [10] which was later extended for time-dependent incompressible flows by Merkle & Athavale [16] and others [85]. The major drawback of the AC methodology is that the numerical diffusivity required to stabilize the solution is considerably high in this kind of problems. It is expected that, the deficiency of low order methods to produce 'spurious eddies' for specific discretizations is lifted with the application of higher order schemes. As Fig. 5-15 shows, the fourth order accurate compact method produces a regular periodic eddy pattern, with excellent agreement with those presented in [23] and [28].

5.4 Concluding Remarks

An efficient high-order accurate, compact finite-difference solver for the solution of the incompressible Navier-Stokes equations is developed. The global pressure correction method is applied to enforce incompressibility. An effective fourth-order accurate compact scheme has been used as a discretizer for the approximation of the spatial and temporal terms. New approximation compact formulas were proposed for the boundary closures to ensure the global accuracy of the solver. An optimized multigrid technique is incorporated for the pressure correction algebraic system, as presented. It has been observed that, when increasing the Reynolds number in fine grid discretizations, the numerical solution of the pressure correction procedure is significantly accelerated, due to the application of the multigrid technique. The zebra coloring scheme numbering unknowns and equations, has increased the degree of parallelism, allowing efficient implementations of realistic problems on paral-

lel architectures. In the next section an efficient parallel algorithm of the solver is designed and implemented modern computing architectures.

Chapter 6

The Parallel Navier-Stokes Solver

A great number of large scale flow simulations are performed on parallel computing architectures. Today supercomputers' hardware comprises of several multicore processors on computing nodes and have a number of accelerators attached. These computing devices have become an substantial component of every supercomputer. Originally, the most common accelerator type, the GPU (Graphics Processing Unit) was designed with an aim to improve the efficiency of the graphics processing pipeline, but it was soon discerned that it would prove to be useful in enhancing the performance in scientific computing applications. Recent efforts accelerating CFD simulations using GPUs can be found in [29] and implementations employed GMG in [87]. In this chapter, the design of a parallel algorithm for the Navier-Stokes solver for architectures with accelerators is presented[66]. In the first section, the advancement of parallel multigrid algorithms is briefly described and, next, each part of the parallel Navier-Stokes algorithm follows separately. Finally, a presentation of the implementation and the performance investigation on shared memory architectures with accelerators end the chapter. The parallel algorithms of Pressure correction, Block Cyclic Reduction (BCR) and the Multigrid techniques are also presented. The parallel performance analysis of the Pressure Correction algorithm for high resolution simulations is the scope of Section 4.

6.1 Parallel Multigrid techniques

Although multigrid components may have a highly parallel nature, the overall structure of the standard multigrid algorithm is inherently not parallel. This sequential nature of multigrid is due to the fact that computation in every grid level of the multigrid algorithm is sequentially executed. Moreover, the computation count changes between different grid levels.

Historically, efforts on parallel multigrid algorithms started in early 80's [3]. In early 90's, McBryan et al. [68] present a study of parallel algorithms suited for multigrid. Parallel results, implementing numerous supercomputing architectures, exhibited very poor scaling in the most implementations. In [26], Chow et al. provide a more recent study of parallel algorithms for multigrid techniques.

In general, parallel algorithms for multigrid techniques can be classified in two classes. The first class is based on the divide and conquer method, subdividing the computational grid into sub-grids with or without overlaps, where each sub-grid is handled by one process. The communication between the sub-problems is usually minimized by minimizing the surface area of each sub-grid and, at the same time, keeping overlapped zones at the sub-grid boundaries. Particularly relevant to this technique is the domain decomposition multigrid methods. The second approach is to solve the global problem using a fast multigrid solver based on a grid partition for a parallel solver. Several other interesting parallelization approaches for specific situations have been explored in the literature: additive variants of multigrid [39, 6], multiple coarse grids [8] and parallel multilevel block LU factorization [59, 26].

6.1.1 Computing Accelerators

The world's first computing accelerator was Nvidia's GPU-GeForce 256 device, that came out in the market in 1999 [1]. As the name suggests, GPUs were initially designed for graphics/video processing and were programmed using specialized graphics languages. In applications like those, where many thousands to millions of pixels need to be displayed on screen at the same time, throughput is more important than latency. In contrast, CPUs execute a single instruction (or few instructions, in the case of multiple cores) rapidly. This led to the current shared memory design of highly parallel architecture of modern graphics processors. In the last decade, the employment of GPUs has entered a new era, as the devices are not only used for calculations related to 3D computer graphics, but for other general purpose computations also, leading to the term GPGPU (General Purpose Graphics Processing Unit). A researcher can identify a computing problem with sufficient data parallelism, exploit the GPU's power to compute the necessary quantities faster than before and, therefore, GPUs are becoming popular in many scientific computing areas.

Today, the top manufacturers of computing accelerators are Nvidia, AMD and Intel. The first two produce CPU based accelerators with thousands of cores, while Intel has accelerator devices with tens or hundreds, but more powerful cores. Top-of-the-line CPUs used in supercomputing clusters contain four to sixteen CPU cores. GPUs on the other hand, consist of hundreds or thousands of processing cores, and fall in the category of shared memory multi-core computing devices.

6.1.2 Multigrid on architectures with accelerators

In the last few years, various multigrid Navier-Stokes solvers have been successfully implemented on architectures with accelerators [33, 37]. Bolz et al. [14] and N. Goodnight et al. [36] first implemented a parallel version of multigrid on GPUs. Both studies use the OpenGL API to interact with the GPU. In [14], an optimized parallel multigrid algorithm

[1] http://www.nvidia.com/page/geforce256.html

was designed to layer the four quadrants of a scalar problem into a single four channel texture format to enable vectorization. They reported a roughly 2x speed-up over the CPU version of the algorithm. Alternatively, all multigrid data, i.e. current solution, residuals, intergrid operators could be stored as textures on the GPU. This practice allowed the authors in [36] to achieve speedups of around 5x on large grids over CPU. Both studies indicated, that memory bandwidth is a major bottleneck in the implementation of multigrid. The aforementioned efforts were put before the introduction of the Compute Unified Device Architecture (CUDA) toolkit. CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA in 2006. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing, an approach known as GPGPU. Unlike previous API solutions, e.g. Direct3D and OpenGL, the CUDA platform is designed to work with programming languages such as C, C++ and Fortran, making it easier for specialists in parallel programming to fully utilize the GPU resources. In this programming environment, several variants of GMG have been implemented on accelerators the last decade. Shinn and Vanka [9] implemented a multigrid Full Approximation Scheme (FAS) method using the semi-implicit method for the SIMPLE algorithm to solve the incompressible Navier-Stokes equations. Thibault and Senocak [89] developed a second-order CUDA-based GPU solver for the incompressible Navier-Stokes equations. The double precision simulation algorithm on a single GPU for the lid-driven cavity problem for a domain of 8388608 computational nodes achieves a roughly 13x speedup compared with an equivalent version running on one CPU core. Cohen and Molemaker [48] later presented the code for a second-order finite volume method solving the incompressible flow equations with the Boussinesq approximation. Using double precision, the GPU-based solver was approximately eight times faster over an eight-core CPU. However, it is should be mentioned that those implementations are based on low-order discretizations. The literature concerning investigations about the GPU performance of high-order compact schemes for incompressible flows is rather poor.

Furthermore, most studies focused on the investigation of the GPU performance of multigrid using full-coarsening. Emphasis in semicoarsening was given only in[67]. There, a massively parallel version of a semicoarsening multigrid 3d solver is developed coupled with a black-box multigrid solver for solving plane systems. Their multi-GPU implementation is found to be faster than the multi-core implementation running on 12 Intel® Xeon® E5-2620 cores, for models with planes large enough (i.e. sizes of at least 1 million cells).

In this chapter, the design of a parallel high-order multigrid Navier-stokes solver based on the sequential solver presented earlier is presented along with the implementation on computing systems equipped with three different GPU types of acceleration devices: a) Tesla M2070, b) Tesla K40 and c) Tesla K80.

6.1.3 The OpenACC API

OpenACC is a relatively new Application Program Interface (API) used for the acceleration of applications on shared memory architectures with accelerators developed by NVIDIA, Cray, the Portland Group and CAPS. The OpenACC API describes a collection of compiler directives to specify loops and regions of code in standard C, C++ or Fortran to be offloaded from a host CPU to an attached accelerator device. OpenACC is designed for portability across operating systems, host CPUs and a wide range of accelerators including APUs, GPUs and multi-core coprocessors like Intel's Phi device [2]. In constrast to CUDA, OpenACC is a high level, descriptive approach, that enables developers to express the parallelism in the code to the compiler using directives (e.g. C$ACC or !$acc in Fortran code), and then lets the compiler target and optimize this parallelism to any parallel architecture from a single source code [3]. In fact, the way to add the OpenACC directives in a legacy code is quite easy and similar to the OpenMP API. The example shown in Table 6.1.3 demonstrates the parallelization of a loop over grid cells for adding the pressure correc-

[2] http://www.intel.com
[3] http://www.openacc.org

tions ($\Delta p(1:N_x, 1:Ny)$) to the stored pressure vector $p(1:N_x, 1:Ny)$, where N_x and N_y is the number of nodes in $x-$ and $y-$direction . Both the OpenMP and OpenACC parallel construct directives can be applied to a readily vectorizable loop (see Table 6.1.3) without changing the original code.

Table 6.1: A Fortran code example with OpenMP / OpenACC parallel construct pressure update implementation

```
! ...OpenACC version
! ...loop over the cells

!$acc kernels
!$acc loop
do   i = 1, N_x
    do   j = 1, N_y
        !... pressure update to the cell
        p(i, j) = p(i, j) + Δp(i, j)
    end do
end do
!$acc end kernels
```

```
! ...OpenMP version
! ...loop over the cells

!$omp parallel
!$omp do
do   i = 1, N_x
    do   j = 1, N_y
        !... pressure update to the cell
        p(i, j) = p(i, j) + Δp(i, j)
    end do
end do
!$omp end parallel
```

In [20], the developed multigrid method is based on GMG HOC approach for the Poisson anisotropic problem over vertex-centered grid. They achieved a roughly 2.5x speedup over a very fine grid (16384x2048) comparing the computing time of the OpenACC program on an NVIDIA Tesla M2070 GPU. The smoothing procedure leads to a series of block tridiagonal linear systems. The key to explain the poor efficiency of the GPU algorithm [20] is the solution of the tridiagonal linear systems using a sequential solver (Thomas algorithm [90]).

Here, these type of linear systems are effectively solved by using the cyclic reduction algorithm [95], which is appropriate for vector programming. Our algorithm uses of backward substitution only, since the coefficient linear systems matrices are factorized at the beginning of the program.

In this work, the target is to accelerate the Navier-Stokes solver's performance on shared memory architectures with accelerator devices. Attention is focused on spotting the differ-

ences between grids with equal and unequal spacing.

6.2 The Block Cyclic Reduction algorithm

Considering the linear system $A\mathbf{u} = \mathbf{b}$, where the coefficient matrix

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & 0 & \dots & 0 & 0 \\ a_5 & a_6 & a_5 & 0 & 0 & \dots & 0 & 0 \\ 0 & a_5 & a_6 & a_5 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_5 & a_6 & a_5 & 0 \\ 0 & 0 & \dots & 0 & 0 & a_5 & a_6 & a_5 \\ 0 & 0 & \dots & 0 & \widehat{a_4} & \widehat{a_3} & \widehat{a_2} & \widehat{a_1} \end{bmatrix} \in \mathbb{R}^{N_x \times N_x} \quad . \tag{6.1}$$

and vectors $\mathbf{u}$ and $\mathbf{b}$ are of size N_x.

Assuming, that, $N_x = 2^q$, the above linear system can be expressed using the following block tridiagonal form

$$\left[\begin{array}{cc|cc|cc|cc} a_1 & a_2 & a_3 & a_4 & & & & 0 \\ a_5 & a_6 & a_5 & 0 & 0 & & & \\ \hline 0 & a_5 & a_6 & a_5 & 0 & 0 & & \\ 0 & 0 & a_5 & a_6 & a_5 & 0 & & \\ \hline & & \ddots & & \ddots & & \ddots & \\ \hline & & & & 0 & a_5 & a_6 & a_5 & 0 & 0 \\ & & & & 0 & 0 & a_5 & a_6 & a_5 & 0 \\ \hline & & & & & & 0 & a_5 & a_6 & a_5 \\ 0 & & & & & & \widehat{a_4} & \widehat{a_3} & \widehat{a_2} & \widehat{a_1} \end{array}\right] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{2^q-2} \\ u_{2^q-1} \\ u_{2^q} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{2^q-2} \\ b_{2^q-1} \\ b_{2^q} \end{bmatrix} \quad , \tag{6.2}$$

which is abbreviates to

$$\begin{bmatrix} \tilde{D} & \tilde{C} & & & \\ B & D & C & & \\ & \ddots & \ddots & \ddots & \\ & & \widehat{B} & \widehat{D} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{2^{q-1}-1} \\ \mathbf{u}_{2^{q-1}} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{2^{q-1}-1} \\ \mathbf{b}_{2^{q-1}} \end{bmatrix}, \tag{6.3}$$

where

$$\begin{aligned} \tilde{D} &= \begin{bmatrix} a_1 & a_2 \\ a_5 & a_6 \end{bmatrix}, \qquad \tilde{C} = \begin{bmatrix} a_3 & a_4 \\ a_5 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & a_5 \\ 0 & 0 \end{bmatrix} \\ D &= \begin{bmatrix} a_6 & a_5 \\ a_5 & a_6 \end{bmatrix}, \qquad \widehat{D} = \begin{bmatrix} a_6 & a_5 \\ \widehat{a_2} & \widehat{a_1} \end{bmatrix}, \qquad \widehat{B} = \begin{bmatrix} 0 & a_5 \\ \widehat{a_4} & \widehat{a_3} \end{bmatrix}, \qquad C = B^{\mathsf{T}} \end{aligned} \tag{6.4}$$

The method of Block Cyclic Reduction in the context of (6.3) will be illustrated. The algorithm proceeds in two stages: reduction and backward substitution. During each step of the reduction stage, half of the unknowns are eliminated. Based on the above matrix partition, after $q-1$ reductions, a 2×2 linear system is solved and the solution is evaluated. All previously eliminated unknowns are computed by the backward substitution procedure. Three consecutive equations, arranged in the following tableau, are considered:

$Eq\#$	$\mathbf{u}_{2j-2}$	$\mathbf{u}_{2j-1}$	$\mathbf{u}_{2j}$	$\mathbf{u}_{2j+1}$	$\mathbf{u}_{2j+2}$	$\mathbf{b}$	
$2j-1$	B	D	C			$\mathbf{b}_{2j-1}$	
$2j$		B	D	C		$\mathbf{b}_{2j}$	for $j = 2, \cdots, 2^{q-2}-1$
$2j+1$			B	D	C	$\mathbf{b}_{2j+1}$	

In order to eliminate $\mathbf{u}_{2j-1}$ and $\mathbf{u}_{2j+1}$, equation $\#(2j-1)$ is multiplied by $-BD^{-1}$, equation $\#(2j+1)$ by $-CD^{-1}$ and both added to equation $\#(2j)$. The result is a new equation

$Eq\#$	$\mathbf{u}_{2j-2}$	$\mathbf{u}_{2j-1}$	$\mathbf{u}_{2j}$	$\mathbf{u}_{2j+1}$	$\mathbf{u}_{2j+2}$	$\mathbf{b}$
j	B'	0	D'	0	C'	$\mathbf{b}'_j$

where

$$\begin{aligned} B' &= -BD^{-1}B \\ D' &= D - BD^{-1}C - CD^{-1}B = \left(D'\right)^{\mathsf{T}} \\ C' &= -CD^{-1}C = \left(B'\right)^{\mathsf{T}} \\ \mathbf{b}'_j &= \mathbf{b}_{2j} - BD^{-1}\mathbf{b}_{2j-1} - CD^{-1}\mathbf{b}_{2j+1} \end{aligned} \quad . \tag{6.5}$$

This way the odd indexed block unknowns are eliminated and a block tridiagonal reduced system for the even indexed block unknowns emerges

$$\begin{bmatrix} \tilde{D}' & \tilde{C}' & & \\ B' & D' & C' & \\ & \ddots & \ddots & \ddots \\ & & \widehat{B}' & \widehat{D}' \end{bmatrix} \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_4 \\ \vdots \\ \mathbf{u}_{2^{q-1}-2} \\ \mathbf{u}_{2^{q-1}} \end{bmatrix} = \begin{bmatrix} \mathbf{b}'_1 \\ \mathbf{b}'_2 \\ \vdots \\ \mathbf{b}'_{2^{q-2}-1} \\ \mathbf{b}'_{2^{q-2}} \end{bmatrix} , \tag{6.6}$$

where

$$\begin{aligned} \tilde{C}' &= C' \\ \tilde{D}' &= \tilde{D} - B\tilde{D}^{-1}\tilde{C} - C\tilde{D}^{-1}B \\ \widehat{D}' &= \widehat{D} - \widehat{B}D^{-1}C \\ \widehat{B}' &= -\widehat{B}D^{-1}B \\ \mathbf{b}'_1 &= \mathbf{b}_2 - B\tilde{D}^{-1}\mathbf{b}_1 - C\tilde{D}^{-1}\mathbf{b}_3 \\ \mathbf{b}'_{2^{q-2}} &= \mathbf{b}_{2^{q-1}} - \widehat{B}D^{-1}\mathbf{b}_{2^{q-1}-1} \end{aligned} \quad , \tag{6.7}$$

which has the same structure as the original system, and thus this process can be cyclically

repeated. Once the even numbered block unknowns have been computed, the odd indexed block unknowns can be obtained from the original system by solving the following block linear systems

$$\begin{aligned} &\tilde{D}\mathbf{u}_1 = b_1 - \tilde{C}\mathbf{u}_2 \\ &D\mathbf{u}_{2j-1} = b_{2j-1} - B\mathbf{u}_{2j-2} - C\mathbf{u}_{2j+2} \qquad \text{for} \quad j = 2, \cdots, 2^{q-2} - 1 \end{aligned} . \tag{6.8}$$

The reduction process generates the sequence of systems:

$$\begin{bmatrix} \tilde{D}^{(k)} & \tilde{C}^{(k)} & & \\ B^{(k)} & D^{(k)} & C^{(k)} & \\ & \ddots & \ddots & \ddots \\ & & \widehat{B}^{(k)} & \widehat{D}^{(k)} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{1\cdot 2^k} \\ \mathbf{u}_{2\cdot 2^k} \\ \vdots \\ \mathbf{u}_{(2^{q-1-k}-1)\cdot 2^k} \\ \mathbf{u}_{(2^{q-1-k})\cdot 2^k} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{(k)} \\ \mathbf{b}_2^{(k)} \\ \vdots \\ \mathbf{b}_{(2^{q-1-k}-1)}^{(k)} \\ \mathbf{b}_{(2^{q-1-k})}^{(k)} \end{bmatrix} \tag{6.9}$$

for $k = 0, 1, \ldots, q-1$. The matrices and vectors are defined by the means of the following recursions

$$\begin{aligned} B^{(k+1)} &= -B^{(k)}(D^{(k)})^{-1}B^{(k)} \\ D^{(k+1)} &= D^{(k)} - B^{(k)}(D^{(k)})^{-1}C^{(k)} - C^{(k)}(D^{(k)})^{-1}B^{(k)} \\ C^{(k+1)} &= \tilde{C}^{(k+1)} = -C^{(k)}(D^{(k)})^{-1}C^{(k)} \\ \tilde{D}^{(k+1)} &= \tilde{D}^{(k)} - B^{(k)}(\tilde{D}^{(k)})^{-1}\tilde{C}^{(k)} - C^{(k)}(\tilde{D}^{(k)})^{-1}B^{(k)} \\ \widehat{D}^{(k+1)} &= \widehat{D}^{(k)} - \widehat{B}^{(k)}(D^{(k)})^{-1}C^{(k)} \\ \widehat{B}^{(k+1)} &= -\widehat{B}^{(k)}(D^{(k)})^{-1}B^{(k)} \\ \mathbf{b}_1^{(k+1)} &= \mathbf{b}_2^{(k)} - B^{(k)}(\tilde{D}^{(k)})^{-1}\mathbf{b}_1^{(k)} - C^{(k)}(\tilde{D}^{(k)})^{-1}\mathbf{b}_3^{(k)} \\ \mathbf{b}_j^{(k+1)} &= \mathbf{b}_{2j}^{(k)} - B^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2j-1}^{(k)} - C^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2j+1}^{(k)} \quad j = 2, \cdots, 2^{q-1-k} - 1 \\ \mathbf{b}_{2^{q-1-k}}^{(k+1)} &= \mathbf{b}_{2^{q-k}}^{(k)} - \widehat{B}^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2^{q-k}-1}^{(k)} \end{aligned} \tag{6.10}$$

for $k = 0, 1, \ldots, q-2$. At the end of this process, the 2×2 system $D^{(q-1)}\mathbf{u}_{2^{q-1}} = \mathbf{b}_1^{q-1}$ is solved.

Taking into account the matrices' structure, one may easily verify that

$$\begin{aligned}
\tilde{D}^{(k)} &= \begin{bmatrix} a_1^{(k)} & a_2^{(k)} \\ a_5 & a_6^{(k)} \end{bmatrix}, \quad B^{(k)} = \begin{bmatrix} 0 & a_3^{(k)} \\ 0 & 0 \end{bmatrix}, \quad D^{(k)} = \begin{bmatrix} a_6^{(k)} & a_5 \\ a_5 & a_6^{(k)} \end{bmatrix}, \\
\widehat{D}^{(k)} &= \begin{bmatrix} a_6^{(k)} & a_5 \\ \widehat{a_2}^{(k)} & \widehat{a_1} \end{bmatrix}, \qquad \widehat{B}^{(k)} = \begin{bmatrix} 0 & a_3^{(k)} \\ 0 & \widehat{a_3}^{(k)} \end{bmatrix}, \qquad C^{(k)} = B^{(k)\mathsf{T}},
\end{aligned} \tag{6.11}$$

for $k = 1, \cdots, q-2$, and it can be seen that the structure for every matrix is retained at every cyclic reduction level, minimizing the storage requirements. Backward substitution is performed by evaluating the solution of the following linear systems

$$\begin{aligned}
\tilde{D}^{(k)}\mathbf{u}_{2^k} &= b_1^{(k)} - \tilde{C}^{(k)}\mathbf{u}_{2^{k+1}} \\
D^{(k)}\mathbf{u}_{(2j-1)2^k} &= b_{2j-1}^{(k)} - B^{(k)}\mathbf{u}_{(2j-2)2^k} - C^{(k)}\mathbf{u}_{(2j)2^k}
\end{aligned} \tag{6.12}$$

for $j = 2, \cdots, 2^{q-k-2}$ and $k = q-2, \ldots, 0$ starting with vector $\mathbf{u}_{2^{q-1}}$.

Algorithm 3 Multigrid Navier-Stokes solver

Step 1. *Guess* an initial pressure field
  for all time steps **do**
    for all RK4 stages **do**
      while (not converge) **do**
Step 2. - *Solve* pressure correction equation
Step 3. - *Correct* pressure and velocity fields
      end
    endfor
Step 4. *Update* velocities and pressure
Step 5. *Set* current pressure field as initial pressure
  endfor

6.3 The Parallel solver for incompressible Navier-Stokes equations

As descripted earlier, the Navier-Stokes solver's algorithm can be outlined with Algorithm 3. In addition, each time step of the discretization scheme for both the momentum and pressure correction equations can be algorithmically descripted as

Stage 1 :

$$u^{(n,1)}_{\tilde{i}+1/2,j} = u^{(n)}_{\tilde{i}+1/2,j} \tag{6.13}$$

$$v^{(n,1)}_{i,\tilde{j}+1/2} = v^{(n)}_{i,\tilde{j}+1/2} \tag{6.14}$$

$$p^{(n,1)}_{i,j} = p^{(n)}_{i,j} \tag{6.15}$$

Stage ℓ :

$$u^{(n,\ell)}_{\tilde{i}+1/2,j} = u^{(n)}_{\tilde{i}+1/2,j} + a_{\ell,\ell-1}\Delta t\ F^{(n,\ell-1)}_{\tilde{i}+1/2,j} \tag{6.16}$$

$$v^{(n,\ell)}_{i,\tilde{j}+1/2} = v^{(n)}_{i,\tilde{j}+1/2} + a_{\ell,\ell-1}\Delta t\ G^{(n,\ell-1)}_{i,\tilde{j}+1/2} \tag{6.17}$$

Pressure Correction Equation :

$$\mathcal{T}\Delta p_{ij} = \mathcal{S}(\nabla \cdot \mathbf{u}^{n,\ell})_{i,j} \tag{6.18}$$

Correct :

$$p^{(n,\ell)}_{i,j} = p^{(n,\ell-1)}_{i,j} + \Delta p_{i,j} \tag{6.19}$$

$$u^{(n,\ell)}_{\tilde{i}+1/2,j} = u^{(n,\ell)}_{\tilde{i}+1/2,j} - a_{\ell,\ell-1}\Delta t\ \mathcal{P}^{-1}_{2x}\mathcal{Q}_{2x}p_{\tilde{i},j} \tag{6.20}$$

$$v^{(n,\ell)}_{i,\tilde{j}+1/2} = v^{(n,l)}_{i,\tilde{j}+1/2} - a_{\ell,\ell-1}\Delta t\ \mathcal{P}^{-1}_{2y}\mathcal{Q}_{2y}p_{\tilde{i},j}, \qquad \textbf{for } \ell = 2,3,4 \tag{6.21}$$

Update :

$$u^{(n+1)}_{\tilde{i}+1/2,j} = u^{(n)}_{\tilde{i}+1/2,j} + \frac{\Delta t}{6}\left(F^{(n,1)}_{\tilde{i}+1/2,j} + 2F^{(n,2)}_{\tilde{i}+1/2,j} + 2F^{(n,3)}_{\tilde{i}+1/2,j} + F^{(n,4)}_{\tilde{i}+1/2,j}\right) \tag{6.22}$$

$$v^{(n+1)}_{i,\tilde{j}+1/2} = v^{(n)}_{i,\tilde{j}+1/2} + \frac{\Delta t}{6}\left(G^{(n,1)}_{i,\tilde{j}+1/2} + 2G^{(n,2)}_{i,\tilde{j}+1/2} + 2G^{(n,3)}_{i,\tilde{j}+1/2} + G^{(n,4)}_{i,\tilde{j}+1/2}\right) \tag{6.23}$$

$$p^{(n+1)}_{i,j} = p^{(n,4)}_{i,j} \tag{6.24}$$

for $i = 1, ..., N_x, \quad j = 1, ..., N_y,$ and $\tilde{i} = 1, ..., N_x - 1, \quad \tilde{j} = 1, ..., N_y - 1$. The first argument n in index pair (n, ℓ) indicates the time step (i.e corresponding to time $t^n = n\Delta t$). The parameter ℓ corresponds to a particular stage of RK4 method, with $t^{n,1} = t^n$, $t^{n,2} = t^{n,3} = t^n + \Delta t/2$, $t^{n,4} = t^n + \Delta t$, where Δt is the time-step size, and term $(F^{(n,\ell)}, G^{(n,\ell)})$ denotes the discrete vector of $\mathbf{R}(\mathbf{u}^{(n,\ell)}, p^{(n,\ell)}; t^{n,\ell})$ in space, for stages $\ell = 2, 3, 4$. As stated earlier, the discrete pressure correction equation (6.18) needs to be solved at each stage of the RK4 method, which is the most time consuming part of the solver. Thus, the effort to develop an efficient parallel Navier-Stokes solver focuses on this procedure.

As has been previously highlighted, the corresponding pressure correction algebraic system is sparse and large in realistic applications, where fine discretizations are necessary. Its solution process has been earlier accelerated using an iterative method coupled with multigrid techniques. For a efficient parallelization of the multigrid pressure correction phase, parallel computations for every component of the multigrid procedure have to be performed without changing the overall algorithmic steps. This approach is preferred in order to ensure the favorable convergence behavior of the solver, avoiding any change to the multigrid cycling steps. Among the multigrid components, the smoothing procedure is the most computationally intensive procedure. Besides, the grid-transfer operators application and the computation of the residual are fully parallelizable.

It is well-known that, smoothing schemes that use red-black ordering of grid nodes are well suited for parallel computations. Block Red-Black Gauss-Seidel smoother calculations can be carried out in parallel, since groups of red and black unknowns are decoupled. Another advantage of using zebra relaxation is its smoothing factors, since multigrid convergence factors are better than those of the lexicographic line Gauss-Seidel relaxations (see Chapter 3). As mentioned in Chapter 2, the resulting pressure correction linear system's form (2.57) in the horizontal zebra coloring scheme, is scalable and has interesting parallel properties [27, 101, 45], as the groups of unknowns can be processed in parallel. More specifically, each block of unknowns corresponds to an horizontal grid line. Their

evaluation is possible with a linear system solution using the basic-matrices A_1, A_6 or $\hat{A}_1$ as the coefficient matrix. Thus, their parallel solution can further increase the degree of parallelism of the numerical method proposed herein. It shall be also noted that the coefficient matrices A_1, A_6 or $\hat{A}_1$ of the basic linear systems have the non-zero entries a_3, a_4 and $\widehat{a_3}$, $\widehat{a_4}$ in their first and last lines. Their presence does not allow one to apply the Cyclic Reduction algorithm directly when solving a linear system with these coefficient matrices. Thus, the Block Cyclic Reduction procedure presented above, is preferred in solving these arising linear sub-systems.

As stated earlier, the selection of the multigrid cycling strategy (i.e. V, W or F-cycling) affects the convergence rate and the parallel performance of the pressure correction numerical scheme. Coarsest and middle grids, corresponding to small size problems, are frequent layovers in each F- and W-cycle, making these strategies less efficient on parallel environments. Moreover, in each cycle greater computation effort is put compared to the V-cycle algorithm. Another important issue is that the implemented intergrids transfer operators influence the convergence rates of the V-cycle algorithm. However, the proposed intergrid pair (BCP, MR) ensures good convergence rates of the V-cycle.

6.3.1 Parallel procedures of the solver

In the previous section, a multigrid based iterative solver has been presented for the solution of the pressure correction equation, which is the most computationally intensive part of the proposed Navier-Stokes numerical scheme. This iterative solver consists of several numerical procedures with high degree of parallelism. They together compose a parallel algorithm (Algorithm 4) of an efficient solution of the incompressible Navier-Stokes equations in shared memory multicore computing architectures. The solver's algorithm is ported to the OpenACC API support.

It is crucial to mention that in case of architectures with accelerators, the total amount of computations is performed on the accelerator device. Only the initialization procedures are

Algorithm 4 Parallel Multigrid Navier Stokes solver

Step 1. CPU: Initialization of u, v and p vectors
Step 2. CPU: Construction of $M^{(k)}$ matrices for all k multigrid levels
Step 3. CPU: Factorization of basic-matrices $A_1^{(k)}$, $A_6^{(k)}$, $\widehat{A}_1^{(k)}$ for all k levels (stored in $fA_1^{(k)}$, $fA_6^{(k)}$, $f\widehat{A}_1^{(k)}$) based on the block cyclic reduction process
Step 4. CPU to Accelerator Device Memory transfer: Copy vectors u, v, p and $M^{(k)}$, $fA_1^{(k)}$, $fA_6^{(k)}$, $f\widehat{A}_1^{(k)}$ for all k levels
while ($t^n < t_{final}$) **do**
Step 5. `acc_kernel`: Evaluate the RHS vectors $\boldsymbol{F}(u, v, p; t^n)$, $\boldsymbol{G}(u, v, p; t^n)$
for $\ell = 2$ **to** $\ell = 4$ **do**
while ($\nabla \cdot \mathbf{u}^{(\ell)} > tol$) **do**
Step 6. `acc_kernel`: Evaluate vectors $u^{(\ell)}$, $v^{(\ell)}$ according to (6.16),(6.17)
Step 7. `acc_kernel`: Evaluate the RHS vector for the Pressure Correction Equation (6.18)
while ($\| \boldsymbol{b} - M\Delta p \| > tol$) **do**
Step 8. `acc_kernel`: $\Delta p = MG\_V(k, \Delta p, M^{(k)}, fA_1^{(k)}, fA_6^{(k)}, f\widehat{A}_1^{(k)}, \boldsymbol{b})$
end
Step 9. `acc_kernel`: Correct $p^{(\ell)}$, $u^{(\ell)}$, $v^{(\ell)}$ according to (6.19),(6.20),(6.21)
endfor
Step 10. `acc_kernel`: Update u, v, p at $t^n + \Delta t$ according to (6.22),(6.23),(6.24)
end
Step 11. Accelerator Device to CPU Memory transfer: Copy vectors u, v, p

performed on the CPU (Steps 1-3) and the data is being sent to the accelerator device which performs the iterative solution procedure. When an acceptable approximation of pressure and velocities has been reached, their values are transferred back to the host's main memory. Steps 1-3 include the initialization of velocity and pressure vectors and the construction and cyclic reduction factorization of the basic matrices arising from the discretization of the pressure equation. These matrices are constructed for every k level of the multigrid cycle. Matrix-free storage method is preferred due to the block structure of all the matrices (Eq. 6.1). Their computation involves 10 value entries only for each basic-matrix regardless the multigrid level, which designates this algorithm as a very efficient one in terms of storage measurements. All the basic linear algebra operations involving these matrices are properly modified with a consideration of this matrix-free storage type. Initialization in Step 2 comes from the implementation of the iterative method and specifically from the direct solution of the relevant sub-systems. One should recall that in each time step four

pressure Poisson equations are being solved. To this end, the three basic-matrices found on the diagonal of the matrix M, are factorized, based on the Cyclic Reduction. As elaborated before, BCR consists of two phases, the forward reduction and the backward substitution. The matrices's factorization is equivalent to the forward reduction phase of the CR. Thus, this approach minimizes the overall computation cost.

Algorithm 5 Parallel Multigrid V-cycle algorithm

`acc_kernel` : $\Delta p^{(r)} = MG\_V(r, \Delta p^{(r)}, M^{(r)}, fA_1^{(r)}, fA_6^{(r)}, f\widehat{A}_1^{(r)}, \boldsymbol{b}^{(r)}$

Presmoothing:

Step 1. $\Delta p^{(r)} = ZebraGS\,(\Delta p^{(m)}, M^{(r)}, \boldsymbol{b}^{(r)}, fA_1^{(r)}, fA_6^{(r)}, f\widehat{A}_1^{(r)}, v_1, tol)$

Restriction:

if $\gamma = 1$ **then**

Step 2. $\boldsymbol{b}^{(r-1)} = R_h^H(\boldsymbol{b}^{(r)} - M^{(r)}\Delta p^{(r)})$

elseif $\gamma < 1$ **then**

Step 3. $\boldsymbol{b}^{(r-1)} = \widetilde{R}_H^h(\boldsymbol{b}^{(r)} - M^r\Delta p^{(r)})$

end

Recursion:

if $r = 1$ **then**

Step 4. $\Delta p^{(1)} = ZebraGS\,(\Delta p^{(1)}, M^{(1)}, \boldsymbol{b}^{(1)}, fA_1^{(1)}, fA_6^{(1)}, f\widehat{A}_1^{(1)}, maxstep, tol)$

elseif $k > 1$ **then**

Step 5. $\Delta p^{(r-1)} = MG\_V(r-1, \boldsymbol{O}, M^{(r-1)}, fA_1^{(r-1)}, fA_6^{(r-1)}, f\widehat{A}_1^{(r-1)}, \boldsymbol{b}^{(r-1)})$

end

Interpolation:

if $\gamma = 1$ **then**

Step 6. $\Delta p^{(r)} = \Delta p^{(r)} + P_H^h\Delta p^{(r-1)}$

elseif $\gamma < 1$ **then**

Step 7. $\Delta p^{(r)} = \Delta p^{(r)} + \widetilde{P}_H^h\Delta p^{(r-1)}$

end

Postsmoothing:

Step 8. $\Delta p^{(r)} = ZebraGS\,(\Delta p^{(r)}, M^{(r)}, \boldsymbol{b}^{(r)}, fA_1^{(r)}, fA_6^{(r)}, f\widehat{A}_1^{(r)}, v_2, tol)$

Every iterative computation can be performed on the accelerator device, including the RK4 time steps and the multigrid pressure correction technique. Steps 5-10 describe each stage (Steps 5-9) of RK4 computations with the solution of the pressure correction problem. Step 10 describes calculations for pressure and velocities updates at each time step.

The parallel V-cycle multigrid technique for the pressure correction procedure is described with a recursive algorithm (see Algorithm 5). Grid nodes are repeatedly reduced in

half, performing restriction, smoothing, prolongation and error correction phases calculating the new pressure approximation $\Delta p^{(r)}$.

Index k corresponds to the grid level, with $r = 1$ implying the coarsest grid. Values v_1 and v_2 correspond to the number of smoothing steps applied in every discrete problem of the form $M^{(r)}\Delta p^{(r)} = b^{(r)}$ during the pre-smoothing and post-smoothing procedure respectively. The fully parallelizable restriction and interpolation multigrid operators are applied in Steps 2, 3, 6 and 7. These transfer matrices are not being constructed explicity, instead an implementation of a matrix-free scheme using matrix-vector multiplication takes place. The recursive application of the multigrid technique at the coarser grid is applied in Step 5. The multigrid algorithm is concluded with the application of the smoother. This parallel procedure is applied twice (Steps 1 and 8), before the restriction and after the interpolation procedure for every multigrid grid level r. At the coarsest grid level $r = 1$ the smoother is also applied (Step 4) for the solution of the error pressure linear system.

Algorithm 6 presents the zebra $x-$line Gauss-Seidel smoothing procedure for the multigrid technique applied to the pressure correction linear system. In this parallel procedure a new approximation of the pressure error Δp is calculated using an initial value Δp^{old}. The algorithm consists of two main computing phases in the evaluation of the new pressure error approximation, based on the red-black Gauss-Seidel smoothing process. In the red phase (Steps 3a-4c) approximations that correspond to red pressure grid nodes are being calculated, involving the matrix-vector multiplication of Steps 3a-d and the direct block linear system solution of Steps 4a-c. The matrix-vector multiplication that involves matrix H_B is a fully block parallelizable procedure. The computation involves matrix-vector multiplications with the basic matrices A_2, A_4 and A_5 and some vector additions. All these basic linear computations are independent and can be performed efficiently on parallel architectures. In Steps 4a-c, a parallel block cyclic reduction is applied for the direct block linear system solution. The second part of the algorithm has a similar structure that involves the calculated red approximations, matrix-vector multiplication with matrix H_R and again a

Algorithm 6 Horizontal Zebra Gauss-Seidel smoother
`acc_kernel` : $\Delta p = ZebraGS(\Delta p^{old}, M, \boldsymbol{b}, fA_1, fA_6, \widehat{fA_1}, maxstep, tol)$

Step 1. $\Delta p = \Delta p^{old}$; $istep = 0$
do {
Step 2. $\Delta p^{old} = \Delta p$; $istep = istep + 1$
Red Phase
Parallel Matrix-Vector Multiplication: $\Delta p_R = H_B * \Delta p_R^{old}$
Step 3a. **for** $i = 1$ **to** $i = N_y/2 - 1$ **do in parallel** $s_i = A_5 * \Delta p_i^{old}$
Step 3b. **for** $i = 1$ **to** $i = N_y/2 - 1$ **do in parallel** $\Delta p_{i+1} = s_i + s_{i+1}$
Step 3c. $\Delta p_1 = A_2 * \Delta p_1^{old} + A_4 * \Delta p_2^{old}$
Step 3d. **for** $i = 1$ **to** $i = N_y/2$ **do in parallel** $\Delta p_i = -\Delta p_i + \boldsymbol{b}_i$
Parallel Solution of $D_R \Delta p_R = \Delta p_R$
for $i = 1$ **to** $i = N_y/2 - 1$ **do in parallel**
Step 4a. $\Delta p_{i+1} = BCR(\Delta p_{i+1}, A_6, fA_6)$
endfor
Step 4b. $\Delta p_1 = -A_3 * \Delta p_2 + \Delta p_1$
Step 4c. $\Delta p_1 = BCR(\Delta p_1, A_1, fA_1)$
Black phase
Step 5. **for** $i = N_y + 1$ **to** $i = N_y$ **do in parallel** $\Delta p_i = \boldsymbol{b}_i$
Matrix-Vector Operation: $\Delta p_B = -H_R * \Delta p_R + b_B$
Step 6a. **for** $i = 1$ to $i = N_y/2$ **do in parallel** $s_i = A_5 * \Delta p_i$
for $i = 1$ to $i = N_y/2 - 1$ **do in parallel**
Step 6b. $\Delta p_{N_y/2+i} = \boldsymbol{b}_{N_y/2+i} - s_i - s_{i+1}$
endfor
Step 6c. $\Delta p_{N_y} = \boldsymbol{b}_{N_y} - \widehat{A_4} * \Delta p_{N_y/2-1} - \widehat{A_2} * \Delta p_{N_y/2}$
Parallel Solution of $D_B \Delta p_B = \Delta p_B$
for $i = N_y/2 + 1$ **to** $i = N_y$ **do in parallel**
Step 7a. $\Delta p_i = BCR(\Delta p_i, A_6, fA_6)$
endfor
Step 7b. $\Delta p^{N_y} = -\widehat{A_3} * \Delta p_{N_y-1} + \Delta p_{N_y}$
Step 7c. $\Delta p_{N_y} = BCR(\Delta p_{N_y}, \widehat{A_1}, \widehat{fA_6})$
}**while** ($||\Delta p - \Delta p^{old}|| > tol$ **or** $istep < maxstep$)

block cyclic reduction linear system solution. The total amount of these computations is parallelizable on block form. Moreover, matrix-vector multiplications where the basic matrices A_5, $\widehat{A_2}$ and $\widehat{A_4}$ are involved and vector additions include a second level of parallelism. This additional level is also present on the cyclic reduction solving procedure.

The block cyclic reduction direct solver's algorithm is presented in Algorithm 7. It comprises of two parallel phases, the forward reduction of the right hand side vector (Steps 2a-c) and the backward substitution (Steps 3a-c). Parallel processes appear in each reduc-

Algorithm 7 Parallel Block Cyclic Reduction
`acc_kernel :` $\mathbf{u} = BCR(\boldsymbol{b}, A, fA)$

Step 1. $q = log_2(N_x)$
Parallel Factorization of right hand side vector $\boldsymbol{b}$
for $k = 0$ **to** $k = q - 2$ **do**
Step 2a. $\mathbf{b}_1^{(k+1)} = \mathbf{b}_2^{(k)} - B^{(k)}(\tilde{D}^{(k)})^{-1}\mathbf{b}_1^{(k)} - C^{(k)}(\tilde{D}^{(k)})^{-1}\mathbf{b}_3^{(k)}$
for $j = 2$ **to** $j = 2^{q-1-k} - 1$ **do in parallel**
Step 2b. $\mathbf{b}_j^{(k+1)} = \mathbf{b}_{2j}^{(k)} - B^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2j-1}^{(k)} - C^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2j+1}^{(k)}$
endfor
Step 2c. $\mathbf{b}_{2^{q-1-k}}^{(k+1)} = \mathbf{b}_{2^{q-k}}^{(k)} - \widehat{B}^{(k)}(D^{(k)})^{-1}\mathbf{b}_{2^{q-k}-1}^{(k)}$
endfor
Parallel Backward-Substitution
Step 3a. Solve $D^{(q-1)}\mathbf{u}_{2^{q-1}} = \boldsymbol{b}_1^{(q-1)}$
for $k = q - 2$ **to** $k = 0$ **do**
Step 3b. Solve $\tilde{D}\mathbf{u}_{2^k} = \boldsymbol{b}_1^{(k)} - \tilde{C}^{(k)}\mathbf{u}_{2^{k+1}}$
for $j = 2$ **to** $j = 2^{q-k-2}$ **do in parallel**
Step 3c. Solve $D\mathbf{u}_{(2j-1)2^k} = \boldsymbol{b}_{2j-1}^{(k)} - B^{(k)}\mathbf{u}_{(2j-2)2^k} - C^{(k)}\mathbf{u}_{(2j)2^k}$
endfor
endfor

tion level k (Steps 2b,3c). These processes involve 2×2 block matrix-vector multiplications, vector additions and linear system solution for specific matrix forms (see Eq. 6.11). This computation can be easily performed with closed formulas, optimizing the parallel computation in block level.

The computation in the above parallel Algorithms 4-7 includes two levels of parallelization. The first one exploits the block structure of the data while the second takes advantage of the basic linear algebra operations within each block. For instance, the parallel computations of Step 6a in Algorithm 6 can be performed as illustrated in Figure 6-1. The first level of parallelization lies on the block level for the evaluation of vector $s_i = A_5 * \Delta p_i$. This parallel matrix-vector multiplication is partitioned in $\frac{N_y}{2}$ blocks of N_x size. Each block operation is assigned to a N_x size of group cores (Block Thread). Taking into account the structure of A_5 matrix, each core of every Block Thread evaluates a single element of vector s_i using matrix values a_1, a_2, a_3, a_4, a_5 and a_6 (assuming Neumann boundary conditions). For this particular matrix-vector operation $\frac{N}{2} = \frac{N_y N_x}{2}$ core threads are necessary. The work assigned to each core thread is represented in the oblong core boxes in Figure

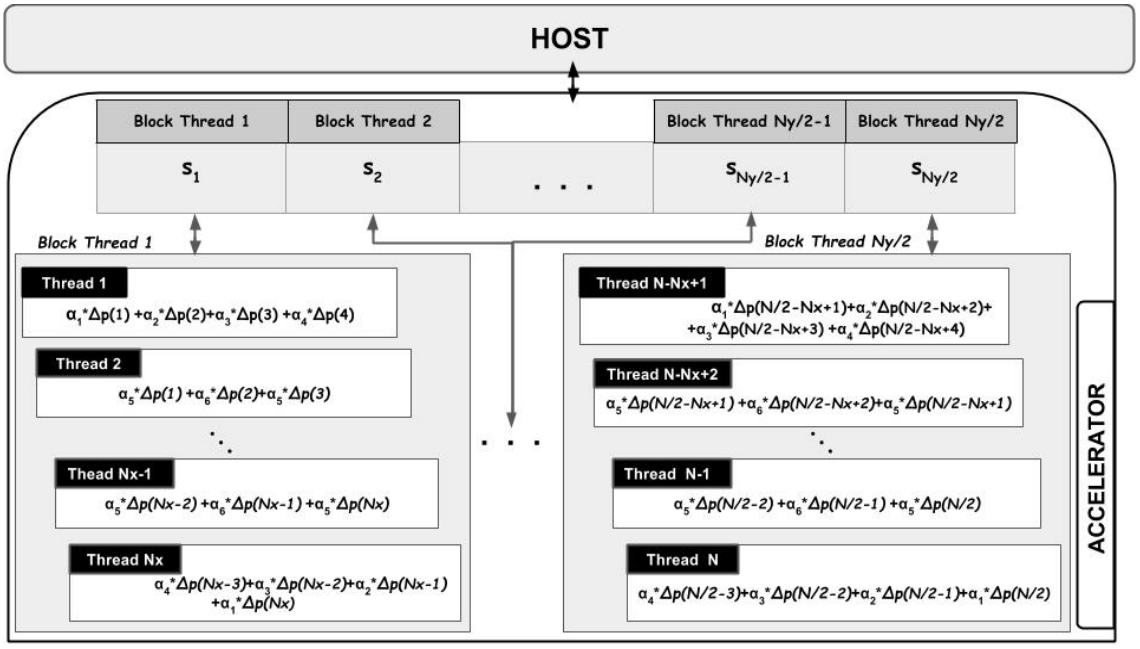

Figure 6-1: Parallelization approach for executing the Step 6a of the Algorithm 6 on GPU architecture. Each oblong *box* is handled solely by GPU core thread.

6-1. All the other parallel block computations are being assigned in a similar way to the acceleration cores.

The incompressible Navier-Stokes parallel solver's algorithm designed in this section, has been implemented on shared memory multicore architectures and its performance inverstigation is the scope of the following section.

6.4 Parallel implementation

This section includes the performance investigation of the Navier-Stokes solver realization for equal and unequal mesh-size discretization problems. The application is developed in double precision Fortran code with the OpenMP and OpenACC APIs support for PGI-CDK 16.7 compiler's suite. In order to demonstrate the computational efficiency of the solver, realizations of the sequential single CPU implementations are compared to OpenMP multicore CPU only and OpenACC device accelerator implementations. Compiler's option `-fast` is used, enabling compiler's optimization features, including the vectorization option.

The basic linear algebra operations were performed using the BLAS scientific library for CPU implementations [13]. Three types of parallel machines with different accelerator device architectural design were selected using CUDA's version 7.5.

- The first machine is an HP SL390s server that features two 6-core Xeon X5660@2.8GHz processors, with 24GB of memory and a Tesla M2070 GPU. The Nvidia's Fermi type M2070 GPU accelerator has 6GB GDDR5 of memory and 448 cores organized in 14 multiprocessors and is attached to the host via an PCI-E Gen2 slot connection. The operating system is Oracle Linux 6.1.

- The second machine is a Dell R730 server featuring two 8-core Xeon E5-2630@2.4GHz processors, with 16GB of memory and a Tesla K40 GPU. The Nvidia's Kepler type K40 GPU accelerator has 12GB GDDR5 of memory and 2880 cores organized in 15 multiprocessors and is attached to the host via an PCI-E Gen3 slot connection. The operating system is Ubuntu Linux 16.10.

- The last machine is a Dell R730 server with two 8-core Xeon E5-2695@2.3GHz processors, with 64GB of memory and a Tesla K80 GPU. The Nvidia's Kepler type K80 GPU accelerator has 24GB GDDR5 of memory and 4992 cores organized on 26 multiprocessors, and it is attached to the host via an PCI-E Gen3 slot connection. The operating system is Ubuntu Linux 16.10.

Two classical incompressible flow test problems are being solved. All the results documented are the average values of 35 numerical solutions, using the machines in standalone mode. The first is the Driven Cavity flow steady state simulation problem and the other is the unsteady Stokes Oscillatory Plate. In all the realizations the time-step size is determined by $CFL = 0.75$ and the size of the coarsest mesh ($\lambda = 0$) of the GMG pressure correction process consists of 4×4 computing cells. The number of pre- and post-smoothing iterations in the descent and ascent phase of every multigrid-cycle is $\nu_1 = 2$ and $\nu_2 = 1$, respectively.

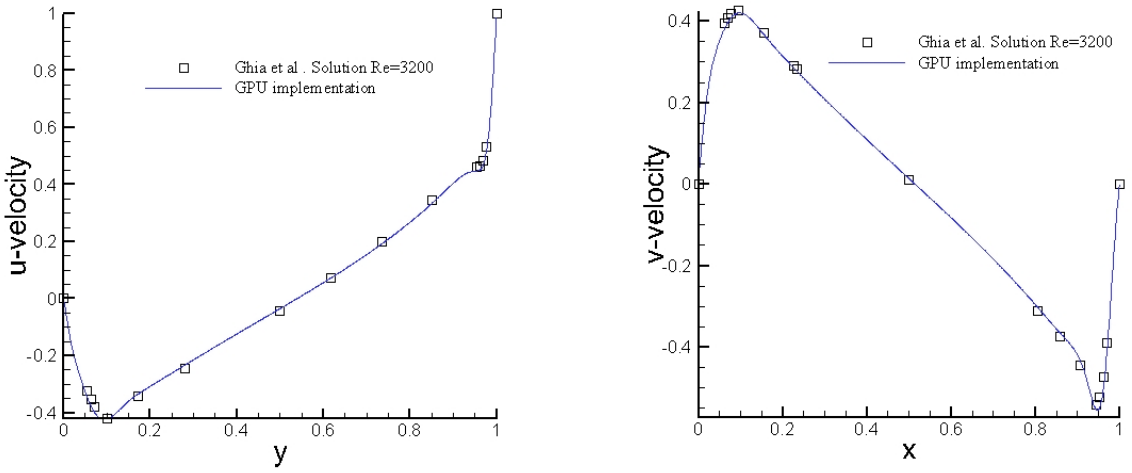

Figure 6-2: Comparison of velocity approximations with reference solutions by Ghia et al. [91] for Re = 3200 over a 1024x1024 mesh at $T = 60$; $\Delta t = 1/1500$

6.4.1 Driven cavity flow problem

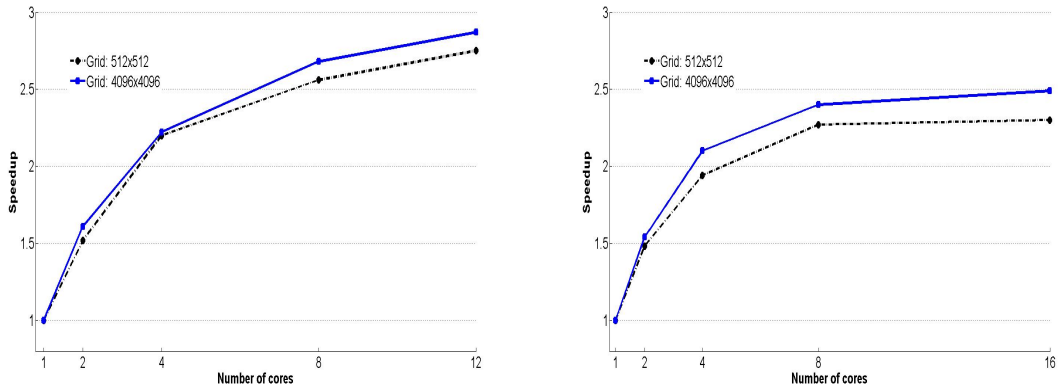

Figure 6-3: Speedup measurements for multi-core CPU only realizations. Top: For the first computer choice. Bottom: For the second computer choice; Driven cavity flow (Re=3200).

The results obtained from the implementation are compared to the numerical data from Ghia et al. [91], in Fig. 6-2. The agreement with the benchmark data for $Re = 3200$ is acceptable. It is noteworthy that the fourth-order accuracy of the parallel solver is verified.

The performance of the Navier-Stokes solver on multi-core CPU only architectures yields no significant acceleration (see Fig. 6-3) for an increased number of CPU cores, as

adding more than eight CPU cores leads to negligible performance improvement (see Fig. 6-3).

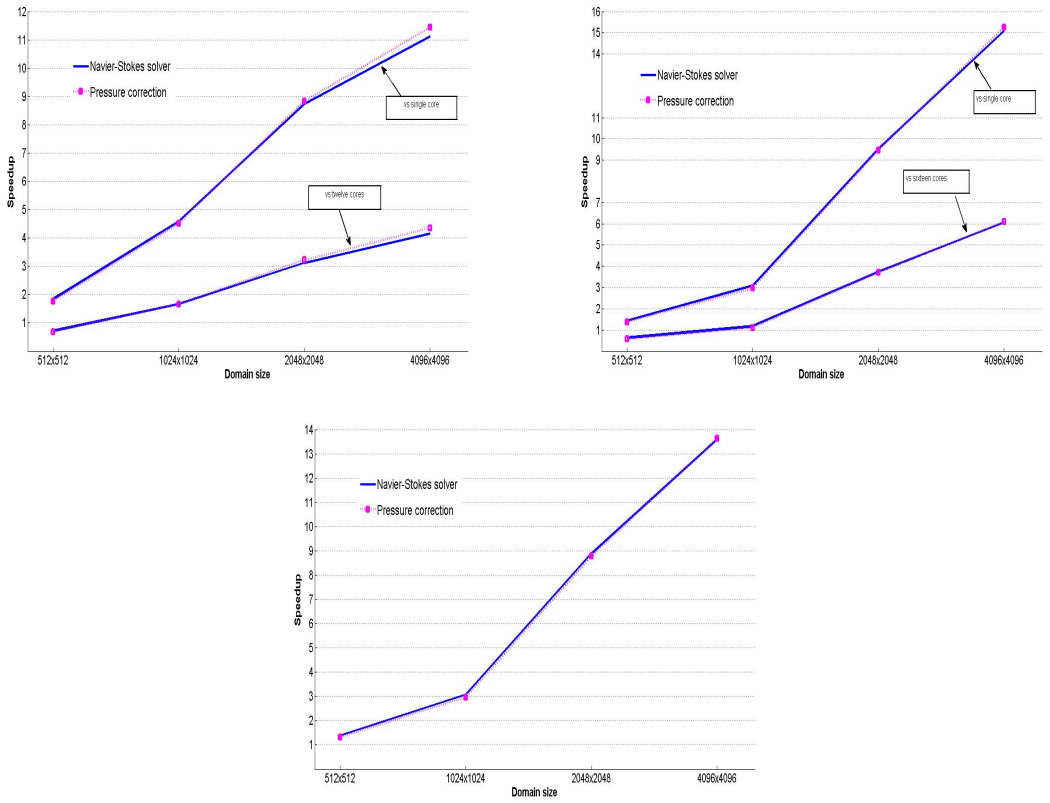

Figure 6-4: Pressure correction and entire Navier-Stokes solver speedups for GPU over single and multi-core CPU only realizations. *Top left* GPU type is M2070. *Top right* GPU type is Tesla K40. *Bottom* GPU type is Tesla K80; driven cavity flow (Re=3200).

With respect to the GPU implementations, speedup plot measurements in Fig. 6-4, for the three types of computer realizations, indicate that acceleration is accomplished for the pressure correction problem and for the entire Navier Stokes solver. The acceleration factor in GPU implementations over the multi-core CPU only implementations is also measured for the first two computer choices. As pointed out earlier, the CPU-GPU communication time is negligible as the data transferring is unvarying, independent of the total time steps.

The best GPU acceleration of the Navier-Stokes solver observed is about 11x using the Tesla M2070 GPU, 15x using the Tesla K40 GPU and 14x using the Tesla K80 GPU types versus a single-core CPU. It is noticeable that speedup measurements increase as the

discretization of a problem gets finer, since the computation also increases. Accelerators' technical specifications differences (PCI bus connection type, number of cores and memory size) accounts for the speedup differences among architecture implementations.

Speedup performance in case of the GPU over multi-core CPU only implementations is observed to be less efficient than those over the single CPU core implementations as expected. However, parallel implementations on the M2070 and K40 GPUs are found to be almost always faster than the OpenMP implementations running on 12 and 16 cores respectively, achieving as high as 4× and 6× speed-ups for a 4096×4096 grid size.

It can also noted that the pressure correction procedure has similar performance to the entire Navier-Stokes solver for every grid option. This outcome can be explained by estimating the time measurements (see Fig. 6-5). Execution time for solving the momentum equation is relatively small compared to the performance of the pressure correction procedure. This time difference becomes smaller for larger problem sizes. However, the pressure correction computation is still the dominant portion of the total run-time.

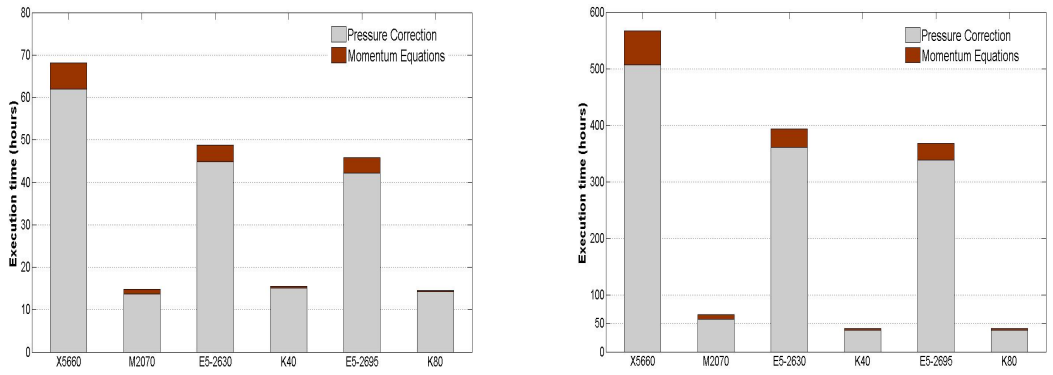

Figure 6-5: Execution time distribution for 1024x1024 (right figure) and 2048x2048 (left figure) discretization problems for single CPU and CPUGPU implementations; Driven cavity flow (Re=3200).

The energy consumption measurements of the algorithm is also investigated in the case of the driven cavity flow test problem. The energy monitoring tool likwid is used to measure the processor and memory energy consumption for CPU-only implementations. For the

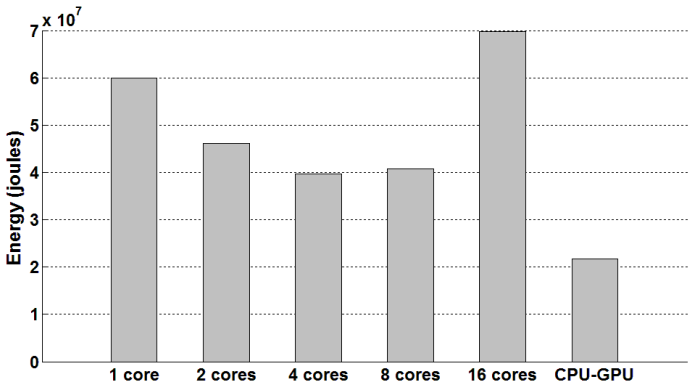

Figure 6-6: Energy measurements in Joules for the second machine choice; Driven cavity flow 2048×2048 discretization (Re=3200).

CPUGPU realizations the Nvidia-smi software is used. Figure 6-6 presents the energy cost in the second computer case, solving the 2048×2048 problem size with all the available computing resources. It seems that enabling all 8 cores of the first processor, is the most energy efficient choice for the CPU-only implementation. Adding the second's processor cores the energy consumption is significantly increased. CPUGPU computation choice is the most energy efficient one among all available algorithm realizations.

Table 6.2: $V-cycle$'s subroutines time percentage (%) for the $TeslaM2070$ multigrid realization.

Mesh size	Residual (%)	Smoother (%)	Restriction (%)	Prolongation (%) and Correction	Norms (%)
512×512	3.72	79.18	1.75	1.94	13.41
1024×1024	5.96	77.53	2.36	2.19	11.96
2048×2048	9.42	74.05	3.3	2.51	10.72
4096×4096	12.02	72.07	3.92	2.74	9.25

Next, the performance of the pressure correction procedure is thoroughly investigated. Tables 6.2-6.4 present the execution time measurements percent for every parallel computation component. The proportion data indicates that the most computationally intensive part of the algorithm is the smoother process. The smoothing procedure takes about 70%

Table 6.3: $V-cycle$'s subroutines time percentage (%) for the $TeslaK40$ multigrid realization.

Mesh size	Residual (%)	Smoother (%)	Restriction (%)	Prolongation (%) and Correction	Norms (%)
512×512	3.02	79.58	1.89	2.09	13.42
1024×1024	4.33	78.80	2.36	2.13	12.38
2048×2048	6.93	76.94	3.11	2.14	10.88
4096×4096	9.79	74.41	3.91	2.15	9.74

Table 6.4: $V-cycle$'s subroutines time percentage for the $TeslaK80$ multigrid realization.

Mesh size	Residual (%)	Smoother (%)	Restriction (%)	Prolongation (%) and Correction	Norms (%)
512×512	2.96	78.55	2.00	2.11	14.38
1024×1024	4.17	78.95	2.37	2.06	12.45
2048×2048	6.64	76.17	3.3	2.10	11.79
4096×4096	9.35	74.79	4.01	2.07	9.78

of the computation every time. On the contrary, the intergrid operators consume $< 7\%$ of the total kernel time. Among the V-cycle's subroutines, the Residual function varies widely as the problem size increases due to the involved matrix vector multiplication.

6.4.2 Stokes oscillating plate

In the previous Chapter, this test case for $Re = 1$ was investigated and numerical results were presented for the Multigrid Navier-Stokes serial algorithm. Now, results of the parallel version implemented on architectures with accelerators are obtained in order to assess the performance of the parallel solver over extremely anisotropic problems. The same simulation parameters are considered, except the fact that the Re number now equals 5000.

As Fig. 6-7 depicts, the speedup measurements for CPU only multi-core realizations

are between 2.6x and 2.3x. Further, it seems that the partial semi-coarsening strategy (solid graph lines in Fig. 6-7) has similar parallel performance and scalability as the full-coarsening (dash-dit graph lines in Fig. 6-7) for CPU only multi-core architectures.

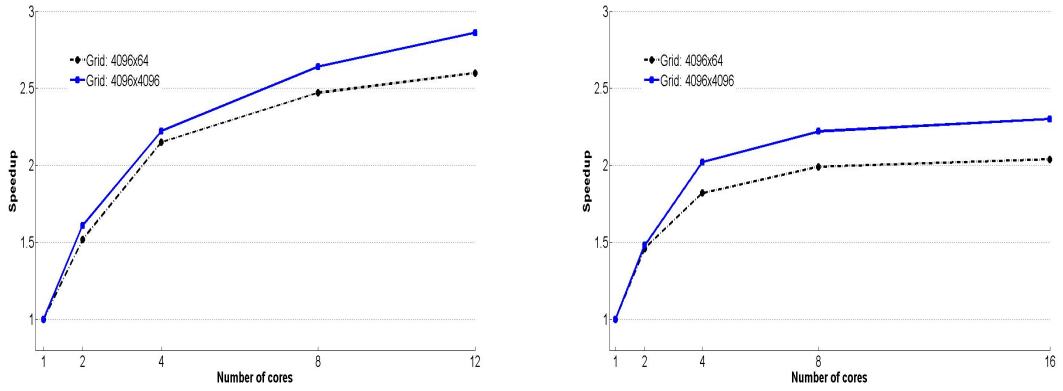

Figure 6-7: Speedup measurements for multi-core CPU only realizations. Left: For the first computer choice. Right: For the second computer choice; Stokes oscillatory plate (Re=5000).

The parallel performance of the CPUGPU partial semi-coarsening and full-coarsening multigrid Navier-Stokes solver is shown in Figure 6-8. The speedup measurements of the three different implementations are demonstrated against single CPU only and multi-core CPU only architectures. These measurements include both the pressure correction problem and the entire Navier Stokes solver for the available parallel computer architectures. In particular, the overall speedup for the finest grid size (4096x4096) is about 10x in Tesla M2070 GPU, 14.5x in Tesla K40 GPU and 12.2x in Tesla K80 GPU type over the single-core CPU solver realization. At first glance, it can be seen that the GPU performance implementation of the anisotropic algorithm is similar to the isotropic one for the cavity flow test problem. However, it is noticeable that in the first test case, the CPUGPU solver achieves a speedup slightly less than two in case of a 512×512 discretization, in the implementation where the first computer machine is involved, and close to one in both the second and third computers. In the anisotropic problem, in case of 4096×64 discretization size, a speedup a little less than 3 using the first computer and near two for the other

computer types is observed. These two different dimension problem cases have the same number of nodes (262144) but the division of labour by the GPU in the anisotropic case leads to improved computation efficiency. In this anisotropic case less but larger basic matrices are involved in the computation, compared to the isotropic case.

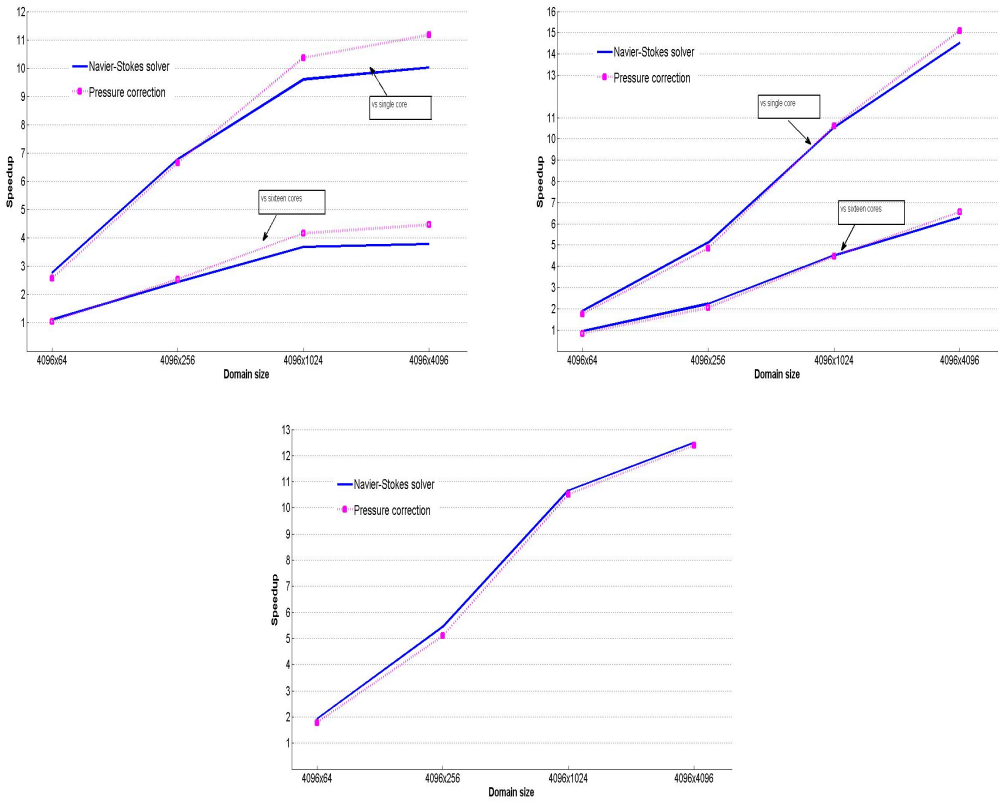

Figure 6-8: Pressure correction and entire Navier-Stokes solver speedups for GPU over single and multi-core CPU only realizations. *Top left* GPU type is M2070. *Top right* GPU type is Tesla K40. *Bottom* GPU type is Tesla K80; Stokes oscillatory plate (Re=5000).

It is also noted that the gap between speedup lines of the pressure correction and the entire Navier-Stokes solver is larger in fine grid problems. To identify this discrepancy, the distribution of the execution time between Pressure correction and Momentum subroutines is presented in Figure 6-9. The simulation execution time is measured in hours for two size problems, single CPU only and CPUGPU implementations for every computer option. It is obvious that in the anisotropic case, the momentum procedure possesses an increased percentage of time compared to the one in the isotropic case. This is due to the reduced

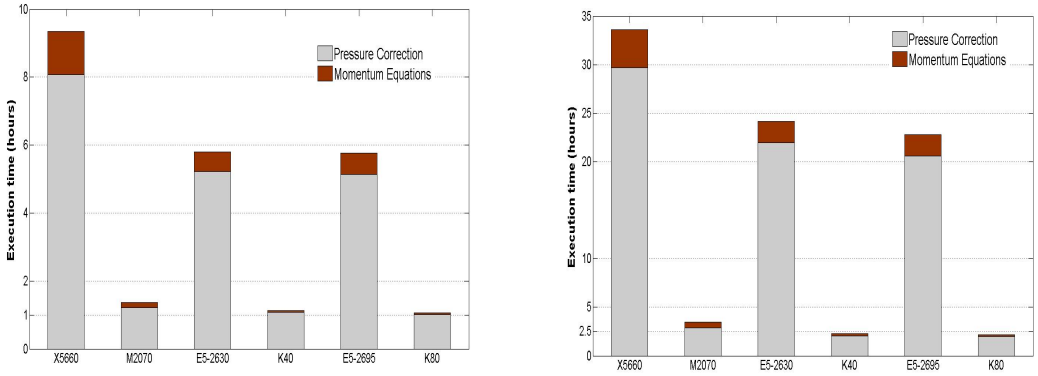

Figure 6-9: Execution time distribution of the CPUGPU Navier-Stokes solver for the Stokes oscillatory plate over 4096×64 and 4096×1024 mesh sizes at $T = 2.25$

number of multigrid cycles needed for the pressure correction sub-problem. For example, one V-cycle is required for every stage of the Runge-Kutta process in the anisotropic case, while in the isotropic problem at least three cycles are needed.

6.5 Concluding Remarks

An efficient parallel algorithm for the solution of the Navier-Stokes equations for incompressible flows was designed and realized. The solver is fourth order accurate and comprises a multigrid technique with compact finite difference discretizations. In order to increase parallelism, the zebra numbering scheme is preferred together with the Block Cyclic Reduction method for the solution of the linear sub-systems. Taking into account the involved basic linear algebra operations and the structure of the finite difference matrix, an extra level of parallelization appears within the algorithm. All the sub-matrices of the coefficient matrix are manipulated with their appropriate entries, avoiding thus the storage of the matrix in total in the computations. That minimizes the need for data storage and communication between the host and accelerator memories. Since the accelerator devices have lower memory limitations, the proposed computation handling allows the consummation of all the computations in the acceleration device. The performance investigation of

the parallel implementation established that multi-CPU only realizations are less efficient than those performed on computing architectures equipped with acceleration devices. The performance of the parallel solver has been investigated on three different types of GPU acceleration architectures, including a recent Kepler type and the legacy Fermi. An acceleration of more than 10x was observed when solving fine discretization problems in all the test cases and computing architectures.

Chapter 7

Conclusions

This chapter summarizes the scientific conclusions of this research work and presents directions for the future.

7.1 Summary of Present Work

The main purpose of this research study is to develop an efficient high resolution numerical solver for simulations of the incompressible flow problems modeled by the Navier-Stokes Equations. It is based on the Finite Difference Compact Scheme discretization in a two dimensional domain. Conclusion results are summarized as follows:

With respect to the elliptic PDEs numerical solver

- It is possible to apply the high-order finite compact elliptic solver directly on cell-centered grid discretizations or on staggered grids. New high-order approximation formulae are derived for the boundary closures, including Dirichlet, Neumann, Robin or mixed-type boundary conditions, to ensure the global accuracy of the solver.

- The arising linear system is large, sparse and block structured, therefore, the use of an efficient iterative method is necessary in order to minimize the computational cost.

- The fourth order of accuracy of the solver is verified, for all boundary condition type choices.

- The performance investigation for the efficient iterative solver of the linear system indicates the necessity of multigrid acceleration

- An analysis of cell-centered multigrid methods for the HOC scheme with LFA, provides us an efficient and robust multigrid solver for both isotropic and anisotropic problems.

With respect to the multigrid Navier-Stokes numerical solver

- An optimized multigrid technique is incorporated to the pressure correction algebraic system in order to accelerate the solution procedure of the Navier-Stokes solver.

- A number of experiments were conducted, including steady and unsteady flow problems, to validate the fourth-order accuracy in space and time of the Navier-Stokes solver.

- The most demanding part of the solver, in terms of computational resources, is the pressure correction solution procedure and is significantly accelerated, when applying the multigrid technique.

- In terms of the CPU workload, the semi-coarsening multigrid technique is prefered to the full-coarsening one for the anisotropy flow problems.

- Moreover, the pressure correction procedure turns up to be the most computationally intensive part of the Navier-Stokes numerical scheme. In particular, for problem sizes up to 256×256, the incorporation of the multigrid scheme produces a containment of the increasing execution time. However, in finer discretization problem sizes, and despite the high convergence rates of GMG on sequential computing architectures, it ultimately yields to unbearable computational cost.

With respect to the parallel multigrid Navier-Stokes numerical solver

- An algorithm that exploits the computational power of multicore computing architectures with accelerators was developed, in order to overcome the CPU time restrictions from sequential implementations of the Navier-Stokes solver, without modifying its primary algorithmic steps.

- In order to increase the solver's parallel properties the zebra coloring scheme is preferred, along with the Block Cyclic Reduction method for solving the linear subsystems.

- The block structure of the finite difference matrix indicates the use of a matrix-free storage method, in order to reduce the data stored and cut down the communication between the main and the accelerator memories. All the sub-matrices of the finite difference matrix are being manipulated with their appropriate block entries. A similar approach takes place in the intergrid transfer operators. The corresponding transfer matrices need not be explicitly computed, since the symbolic formulae for those operators are being used.

- The way that the computations could be conducted suggested herein, enables the total amount of workload to be performed on the acceleration device, in order to overcome the memory physical limitations in the accelerator devices.

- The computations incorporate two levels of parallelism. The first one exploits the block structure of the data, while the second one takes advantage of the basic linear algebra operations within each matrix block.

- The application of the parallel solver is being implemented on three shared-memory computing architectures with different accelerator devices, including a modern Kepler-class GPU and a legacy Fermi-class GPU.

- In multi-CPU only realizations, the parallel performance of the solver was considered to be poor as the speed-up factors went up to hardly 3x and low scalability was noted. On the other hand, in a CPU-GPU environment, the implementations were faster compared to the multi-CPU only, as the speed-up observed was 4x in every accelerator device tested.

- With respect to CPU-GPU implementations, an acceleration of more than 10x is obtained for fine discretization problems in every test case and computing architecture, compared to sequential implementations.

7.2 Future Work Suggestions

The proposed solver can be extended in several ways. A few recommendations follow:

With respect to the elliptic PDEs numerical solver

- The proposed fourth-order compact discretization methodology for the modified Helmholtz problem may be generalized for the complete elliptic operator and/or in three dimensions in a straightforward manner.

- Similar generalizations of specialized multigrid methods to 3D are not straightforward; at least their implementations are nontrivial. Zebra line GaussSeidel relaxation in 2D will be generalized to zebra plane relaxation in 3D, with each planewise solution being obtained by using zebra line relaxations. Partial semicoarsening is also complicated, since there are two directions with different mesh sizes.

- The different behavior between the multigrid method applied to cell-centered and vertex-centered grids can be featured on a great scale by using the Local Fourier Analysis for the HOC scheme.

With respect to the multigrid Navier-Stokes numerical solver

- The numerical Navier-Stokes algorithm can be extended for complex geometries on three dimensional curvilinear coordinates, thus expanding the applicability of the current methodology.

- The method is suitable for LES of moderate Reynolds number incompressible flows, while addition of implicit time marching and turbulence models would make it suitable for high fidelity simulations of high Reynolds number turbulent flows.

With respect to the parallel multigrid Navier-Stokes numerical solver

- The employment of a single accelerator device already yields to high speed-up factors. A potential improvement requires the design of a parallel algorithm that employs multiple devices from an Heterogeneous Multi-Accelerator Cluster, using OpenACC, OpenMP and MPI APIs.

Bibliography

[1] Brandt A. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation*, 31:333–390, 1977.

[2] Brandt A. Multi-grid solvers for non-elliptic and singular-perturbation steady-state problems. Technical report, Weizmann Institute of Science, 1981.

[3] Brandt A. Multi-grid solvers on parallel computers. In M. Schultz, editor, *Elliptic Problem Solvers*, pages 39–84. Academic Press, 1981.

[4] Brandt A. A guide to multigrid development in multigrid methods. pages 220–312. Springer Verlag, Berlin, 1982.

[5] Brandt A. Multigrid techniques: 1984 guide, with applications to fluid dynamics. Gmd studien nr. 85, GMD, GMD-AIW, Postfach 1240, D-5205, St. Augustin 1, Germany, 1984.

[6] Greenbaum A. A multigrid method for multiprocessors. *Applied Mathematics and Computation*, 19(1):75 – 88, 1986.

[7] Jordan S. A. An iterative scheme for numerical solution of steady incompressible viscous flows. *Comput. Fluids*, 21:503–517, 1992.

[8] McBryan Oliver A. and Frederickson Paul O. Parallel superconvergent multigrid. In S. McCormick, editor, *Multigrid Methods: Theory, Applications and Supercomputing*, pages 195–210. Marcel Dekker, New York, 1988.

[9] Shinn A.F and Vanka S.P. Implementation of a semi-implicit pressure-based multigrid fluid flow algorithm on a graphics processing unit. In *ASME Conf Proc 2009(43864):125?133*. 2009.

[10] Chorin A.J. A numerical method for solving incompressible, viscous flow problems. *J. Comp. Physics*, 2(1):12–26, 1967.

[11] Chorin A.J. Numerical solution of the Navier-Stokes equations. *Math. Comput.*, 22:745–762, 1968.

[12] Christopher R. Anderson and Marc B. Reider. A high order explicit method for the computation of flow about a circular cylinder. *Journal of Computational Physics*, 125(1):207–224, 1996.

[13] BLAS, http://www.netlib.org/blas/, 2017.

[14] Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröoder. Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. *ACM Trans. Graph.*, 22:917–924, 2003.

[15] Christara C. and Smitht B. Multigrid and multilevel methods for quadratic spline collocation. *BIT*, 37(4):781–803, 1997.

[16] Merkle C. and Athavale M. Time-accurate unsteady incompressible flow algorithms based on artificial compressibility. *AIAA Paper*, pages 87–1137, 1987.

[17] Wu J. C. Theory for aerodynamic force and moment in viscous flows. *AIAA Journal*, 19(4):432–441, 1981.

[18] Kuo C.-C.Jay and Bernard C. Levy. Two-color fourier analysis of the multigrid method with red-black gauss-seidel smoothing. *Applied Mathematics and Computation*, 29(1):69 – 87, 1989.

[19] Arakawa Ch., Demuren A. O., Rodi W., and Schonung B. In *in M. Deville (ed.) Proc. 7th GAMM Conf. on Numerical Methods in FluidMechanics*, volume 20, pages 1–8. 1988.

[20] Niki E. Charalampaki and Emmanuel N. Mathioudakis. CPU-GPU computations for multigrid techniques coupled with fourth-order compact discretizations for isotropic and anisotropic poisson problems. In *Proceedings of the 6th International Conference on Numerical Analysis (NumAn2014), Greece*.

[21] Hirt C.W., Nichols B.D., and Romero N.C. SOLA: a numerical solution algorithm for transient fluid flows. In *Los Alamos Report LA-5852*. 1975.

[22] Anderson J. D. *Computational Fluid Dynamics*. McGraw Hill, New York, 1995.

[23] Drikakis D. and Smolarkiewicz PK. On spurious vortical structures. *J. Comput. Physics*, 172:309–325, 2001.

[24] Gaitonde D. and Visbal M. High order schemes for navier-stokes equations : Algorithm and implementation into fdl3di. In *NASA, AFRL-VA-WP-TR-1998-3060*. 1998.

[25] Kwak D., Kiris C., and Kim S. Computational challenges of viscous incompressible flows. *Comp. and Fluids*, 34:283–299, 2005.

[26] Chow E., Falgout R., Hu J., Tuminaro R., and Yang U. *A Survey of Parallelization Techniques for Multigrid Solvers*. SIAM, Philadelphia, 2006.

[27] Mathioudakis E., Papadopoulou E., and Saridakis Y. Iterative solution of elliptic collocation systems on a cognitive parallel computer. *Computers and Maths with Appl.*, 48:951–970, 2004.

[28] John A. Ekaterinaris. High-order accurate numerical solutions of incompressible flows with the artificial compressibility method. *Int. J. Numer. Meth. Fluids*, 45:1187–1207, 2004.

[29] Erich Elsen, Patrick LeGresley, and Eric Darve. Large calculation of the flow over a hypersonic vehicle using a {GPU}. *Journal of Computational Physics*, 227(24):10148 – 10161, 2008.

[30] Auteri F., Parolini N., and Quartapelle L. Numerical investigation on the stabillity of singular driven cavity flow. *J. Comput. Physics*, 183:1–25, 2002.

[31] Sotiropoulos F. and Yang X. Immersed boundary methods for simulating fluid-structure interaction. *Progress in Aerospace Sciences*, 65:1–21, 2014.

[32] Spotz W. F. and Carey G. F. High-order compact scheme for the steady stream-function vorticity equations. *Internat. J. Numer. Methods Engrg.*, 38:3497–3512, 1995.

[33] Christos K. Filelis-Papadopoulos and George A. Gravvanis. Parallel multigrid algorithms based on generic approximate sparse inverses: an SMP approach. *The Journal of Supercomputing*, 67(2):384–407, 2014.

[34] Kovasznay L. I. G. Laminar flow behind a two-dimensional grid. *Proc. Camb. Phil. Soc.*, 44:58–62, 1948.

[35] Thor Gjesdal and Magni El??n Hope Lossius. Comparison of pressure correction smoothers for multigrid solution of incompressible flow. *International Journal for Numerical Methods in Fluids*, 25(4), 1997.

[36] Nolan Goodnight, Cliff Woolley, Gregory Lewin, David Luebke, and Greg Humphreys. A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware. In M. Doggett, W. Heidrich, W. Mark, and A. Schilling, editors, *Graphics Hardware*, pages 1–11. The Eurographics Association, 2003.

[37] George A. Gravvanis and Christos K. Filelis-Papadopoulos. On the multigrid cycle strategy with approximate inverse smoothing. *Engineering Computations*, 31(1):110–122, 2014.

[38] Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):"1111–1145", 1987.

[39] Bramble J. H., Pasciak J. E., and Xu J. Parallel multilevel preconditioners. *Mathematics of Computations*, 55:1 – 22, 1990.

[40] Harlow F. H. and Welch J. E. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids*, 8, 1965.

[41] W. Hackbusch. *Multigrid Methods and Applications*. Springer?Verlag, Berlin, 1985.

[42] Yavneh I. Multigrid Smoothing Factors for Red-Black Gauss-Seidel Relaxation Applied to a Class of Elliptic Operators, journal = SIAM Journal on Numerical Analysis, volume = 32, number = 4, pages = 1126-1138, year = 1995, doi = 10.1137/0732051, url = http://dx.doi.org/10.1137/0732051, eprint = http://dx.doi.org/10.1137/0732051.

[43] Bramble J., Ewing R., Pasciak J., and Shen J. The analysis of multigrid algorithms for cell centered finite difference methods. *Advance in Computational Mathematics*, 5:15?29, 1996.

[44] Dongarra J., Duff I., Sorensen D., and van der Vorst H. *Numerical Linear Algebra for high-performance computers*. SIAM, Philadelphia, 1998.

[45] Dongarra J., Duff I., Sorensen D., and van der Vorst H. *Numerical Linear Algebra for high-performance computers*. SIAM, Philadelphia, 1998.

[46] Larsson J., Lien F.S., and Yee E. Conditional semicoarsening multigrid algorithm for the Poisson equation on anisotropic grids. *J. Comp. Physics*, 208:368–383, 2005.

[47] Butcher J.C. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta Methods and General Linear Methods*. J. Wiley and Sons, Chichester, 1987.

[48] Cohen J.M and Molemaker M.J. A fast double precision cfd code using cuda. In *Parallel CFD 2009*. 2009.

[49] Zhang Jun. Multigrid method and fourth-order compact scheme for 2d poisson equation with unequal mesh-size discretization. *J. Comp. Physics*, 179:170–179, 2002.

[50] Zhang Jun and Zhao Jennifer J. Unconditionally stable finite difference scheme and iterative solution of 2d microscale heat transport equation. *Journal of Computational Physics*, 170(1):261 – 275, 2001.

[51] Lele S. K. Compact finite difference schemes with spectral-like resolution. *J. Comput. Physics*, 103:16–42, 1992.

[52] Mahesh K., Constantinescu G., and Moin P. A numerical method for large-eddy simulation in complex geometries. *Journal of Computational Physics*, 197(1):215–240, 2004.

[53] Shahbazi K., Fischer P. F., and Ethier C. R. A high-order discontinuous Galerkin method for the unsteady incompressible Navier-Stokes equations. *J. Comput. Physics*, 222:391–407, 2007.

[54] Stüben K. and Trottenberg U. Multigrid methods: fundamental algorithms, model problem analysis and applications, in: W. hackbusch, u. trottenberg (eds.), multigrid methods. volume 960 of *Lecture Notes in Mathematics*, pages 1–176. Springer, Berlin, 1982.

[55] George Em Karniadakis, Moshe Israeli, and Steven A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 97(2):414–443, 1991.

[56] Briggs W. L., Henson V. E., and McCormick S.F. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000.

[57] Collatz L. *The Numerical Treatment of Differential Equations*. Springer-Verlag, Berlin, 1960.

[58] Panton R. L. *Incompressible Flow*. J. Wiley and Sons, Chichester, 1984.

[59] Zhongze Li, Yousef Saad, and Masha Sosonkina. parms: a parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*, 10(5-6):485–509, 2003.

[60] Gupta M. M., Kouatchou J., and Zhang J. Comparison of second- and fourth-order discretizations for multigrid poisson solvers. *J. Comp. Physics*, 132:226–232, 1997.

[61] Gupta M. M., Manohar R. P., and Stephenson J. W. A single cell high order scheme for convection-diffusion equation with variable coefficients. *Int. J. Numer. Methods Fluids*, 4:641–651, 1984.

[62] Khalil M. and Wesseling P. Vertex-centered and cell-centered multigrid for interface problems. *J. Sci. Comput*, 98:1–20, 1992.

[63] Li M., Tang T., and Fornberg B. A compact fouth order finite difference scheme for the steady incompressible navier-stokes equations. *Int. J. Numer. Methods Fluids*, 20:1135–1151, 1995.

[64] Mohr M. and Wienands R. Cell-centred multigrid revisited. *Computing and Visualization in Science*, 7(3):129–140, 2004.

[65] Rosenfeld M. and Kwak D. Time-dependent solutions of viscous incompressible flows in moving co-ordinates. *Int. J. Num. Methods in Fluids*, 13:1311–1328, 1991.

[66] Vassilios G. Mandikas and Emmanuel N. Mathioudakis. A parallel multigrid solver for incompressible flows on computing architectures with accelerators. *The Journal of Supercomputing*, http://dx.doi.org/10.1007/s11227-017-2066-y, 2017.

[67] Abdulrahman M. Manea. *Parallel Multigrid and Multiscale flow solvers for high-performance-computing architectures*. PhD thesis, Department of Energy Resources Engineering, Stanford University, 2015.

[68] Oliver A. McBryan, Paul O. Frederickson, Johannes Lindenand, Anton Schüller, Karl Solchenbach, Klaus Stüben, Clemens-August Thole, and Ulrich Trottenberg. Multigrid methods on parallel computersa survey of recent developments. *IMPACT of Computing in Science and Engineering*, 3(1):1 – 75, 1991.

[69] ML. Minion and DL. Brown. Performance of under-resolved two-dimensional incopressible simulations. *J. Comput. Physics*, 122:734–765, 1997.

[70] Hamid Moghaderi and Mehdi Dehghan. A multigrid compact finite difference method for solving the one-dimensional nonlinear sine-gordon equation. *Mathematical Methods in the Applied Sciences*, 38(17):3901–3922, 2015.

[71] Kampanis N. and Ekaterinaris J. A staggered grid, high-order accurate method for the incompressible navier-stokes equations. *J. Comp. Physics*, 215:589–613, 2006.

[72] Mathioudakis Emmanuel N., Mandikas Vassilios G., Kozyrakis Georgios V., Kampanis Nikolaos A., and Ekaterinaris John A. Multigrid cell-centered techniques for high-order incompressible flow numerical solutions. *Aerospace Science and Technology*, 64:85–101, 2017.

[73] Santhanam Nagarajan, Sanjiva K. Lele, and Joel H. Ferziger. A robust high-order compact method for large eddy simulation. *Journal of Computational Physics*, 191(2):392–419, 2003.

[74] Wesseling P. Cell-centered multigrid for interface problems. *J. Comput. Phys.*, 79:85–91, 1988.

[75] Wesseling P. *An Itroduction to Multigrid Methods*. J. Wiley and Sons, Chichester, U.K., 1992.

[76] Wesseling P. Accurate, stable and efficient navier-stokes solvers based on explicit treatment of the pressure term. *J. Comput. Phys.*, 199:221–259, 2004.

[77] Loc Ta Phuoc and Bouard R. Numerical solution of the early stage of the unsteady viscous flow around a circular cylinder: a comparison with experimental visualization and measurements. *Journal of Fluid Mechanics*, 160:93–117, 1985.

[78] Zhipeng Qin, Keegan Delaney, Amir Riaz, and Elias Balaras. Topology preserving advection of implicit interfaces on cartesian grids. *J. Comput. Phys.*, 290:219–238, 2015.

[79] Varga R. *Matrix Iterative Analysis*. Springer Verlag, New York, 2000.

[80] Wienands R. and Oosterlee C. On three-grid fourier analysis for multigrid. *J. on Scientific Computing*, 23:651?671, 2006.

[81] Issa R.I. Solution of the implicitly discretised fluid flow equations by operator-splitting. *J. Comp. Physics*, 65:40–65, 1985.

[82] Wienands Roman and Oosterlee Cornelis W. On three-grid fourier analysis for multigrid. *SIAM J. Scientific Computing*, 23(2):651–671, 2001.

[83] Hirsh R. S. Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique. *J. Comput. Physics*, 19:90–109, 1975.

[84] Sani R. L., Shen J., Pironneau O., and Gresho P. M. Pressure boundary condition for the time-dependent incompressible Navier?Stokes equations. *International Journal for Numerical Methods in Fluids*, 50(6):"673–682", 2006.

[85] Rogers S.E. Numerical solution of the incompressible Navier-Stokes equations. In *NASA TM 102199, Ames Research Center, Moffett Field, CA*. 1990.

[86] McCormick S.F. *Multigrid Methods*. SIAM, Philadelphia, 1987.

[87] Vanka S.P, Shinn A.F, and Sahu K.C. Computational fluid dynamics using graphics processing units: challenges and opportunities. In *ASME Conf Proc 2011(54921):429?437*. 2011.

[88] Patankar S.V. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Co., Washington, DC, 1980.

[89] Julien Thibault and Inanc Senocak. CUDA Implementation of a Navier-Stokes Solver on Multi-GPU Desktop Platforms for Incompressible Flows. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, Aerospace Sciences Meetings, 2009, Orlando, Florida*.

[90] L. H. Thomas. Elliptic Problems in Linear Differential Equations over a Network. Technical report, Columbia University, 1949.

[91] Ghia U., Ghia K. N., and Shin C. T. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Physics*, 48:387–411, 1982.

[92] Trottenberg U., Oosterlee C., and Schüller A. *Multigrid*. Elsevier Academic Press, New York, 2001.

[93] Mandikas V., Mathioudakis E., Papadopoulou E., and Kampanis N. In *Proc. of the World Congress on Engineering 2013 (WCE2013, Imperial College - London, U.K., July 3-2, 2013)*, volume 1, pages 74–79. Award Certificate of Merit for The 2013 International Conference of Applied and Engineering Mathematics.

[94] Hemker P. W. On the order of prolongations and restrictions in multigrid procedures. *Computational and Applied Mathematics*, 32:423–429, 1990.

[95] Hockney R. W. A fast direct solution of poisson's equation using fourier analysis. *J. ACM*, 12(1):95–113, January 1965.

[96] Kahan W.M. *Gauss-Seidel Methods of solving large systems of linear equations*. 1958.

[97] Ge Y. Multigrid method and fourth-order compact difference discretization scheme with unequal meshsizes for 3d poisson equation. *J. Comp. Physics*, 229:6381–6391, 2010.

[98] Ge Y., Cao F., and Zhang Jun. A transformation-free hoc scheme and multigrid method for solving the 3d poisson equation on nonuniform grids. *J. Comp. Physics*, 234:199–216, 2013.

[99] Kwak D. Y. V-cycle multigrid convergence for cell-centered finite differences. *SIAM J.Sci. Comput.*, 21, 1999.

[100] Morinishi Y., Lund T. S., Vasilyev O. V., and Moin P. Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow. *Journal of Computational Physics*, 143(1):90–124, 1998.

[101] Saad Y. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003.

[102] David Young. Iterative methods for solving partial difference equations of elliptic type. *Transactions of the American Mathematical Society*, 76(1):92–111, 1954.