National Technical University of Athens
Department of Mechanical Engineering
Fluids Section
Laboratory for Innovative Environmental Technologies

Data Assimilation in Fast Fluid Dynamics Methodologies

Diploma Thesis

Despoina Angeliki Spanou

Supervisor: Demetri Bouris, Professor NTUA

Athens, February 2024

National Technical University of Athens
Department of Mechanical Engineering
Fluids Section
Laboratory for Innovative Environmental Technologies

Περίληψη

Η μέθοδος Ταχείας Υπολογιστικής Ρευστομηχανικής είναι μια υπολογιστική τεχνική που έχει σχεδιαστεί για την αποτελεσματική επίλυση των εξισώσεων Navier-Stokes για ασυμπίεστα ρευστά. Σε αντίθεση με τις παραδοσιακές μεθόδους που δίνουν προτεραιότητα στην απόλυτη ακρίβεια, η Ταχεία Υπολογιστική Ρευστομηχανική δίνει προτεραιότητα στην υπολογιστική ταχύτητα, καθιστώντας την ιδιαίτερα ικανή σε σενάρια όπου οι γρήγορες προσομοιώσεις είναι απαραίτητες. Αν και μπορεί να θυσιάσει κάποια ακρίβεια, η δύναμή της έγκειται στην ικανότητά της να παρέχει οπτικά ελκυστική και δυναμική συμπεριφορά ρευστών σε εφαρμογές πραγματικού χρόνου. Η μέθοδος Ταχείας Υπολογιστικής Ρευστομηχανικής σχεδιάστηκε αρχικά ως εργαλείο για την ενίσχυση της οπτικής πιστότητας της δυναμικής ρευστών σε βιντεοπαιχνίδια, αλλά εξελίχθηκε ώστε να καλύπτει ένα ευρύτερο φάσμα εφαρμογών, συμπεριλαμβανομένων των προσομοιώσεων αστικού μικροκλίματος.

Από την άλλη πλευρά, η αφομοίωση δεδομένων είναι η επιστήμη του συνδυασμού διαφορετικών πηγών πληροφοριών για την πρόβλεψη πιθανών καταστάσεων ενός συστήματος καθώς αυτό εξελίσσεται με το χρόνο. Σε αυτό το πλαίσιο, το *nudging* είναι μια δυναμική μέθοδος που χρησιμοποιείται σε διάφορους τομείς, συμπεριλαμβανομένης της Υπολογιστικής Ρευστομηχανικής, για την προσαρμογή της κατάστασης ενός μοντέλου προς τα παρατηρούμενα δεδομένα μέσω της εισαγωγής ενός όρου ανατροφοδότησης.

Η παρούσα διπλωματική εργασία παρουσιάζει μια καινοτόμο και υπολογιστικά αποδοτική μέθοδο για την προσομοίωση δισδιάστατων ασυμπίεστων ροών με χρήση Ταχείας Υπολογιστικής Ρευστομηχανικής σε δομημένα πλέγματα. Για τον μετριασμό των πιθανών ανακριβειών που ενυπάρχουν στην Ταχεία Υπολογιστική Ρευστομηχανική, ενσωματώνεται η μέθοδος *nudging*, η οποία αφομοιώνει δεδομένα από εξωτερικές πηγές για να βελτιώσει τα αριθμητικά αποτελέσματα και να αντισταθμίσει τους εγγενείς περιορισμούς του επιλύτη στην ακριβή μοντελοποίηση φυσικών φαινομένων σε λεπτομερές επίπεδο.

Στα πλασία της διπλωματικής εργασίας αναπτύχθηκε ένας προσαρμοσμένος επιλύτης σε γλώσσα προγραμματισμού C++ για την υλοποίηση του αλγορίθμου Ταχείας Υπολογιστικής Ρευστομηχανικής και της τεχνικής *nudging*. Αρχικά, η αποδοτικότητα του επιλύτη αξιολογήθηκε σε μια ροή αναφοράς σε μια τετραγωνική κοιλότητα με κινούμενο άνω στερεό όριο, επιβεβαιώνοντας τα αποτελέσματα της βιβλιογραφίας. Ξεφεύγοντας από τις συμβατικές μεθόδους Ταχείας Υπολογιστικής Ρευστομηχανικής, ο επιλύτης ενσωμάτωσε ένα σχήμα υψηλής τάξης για την ημι-Λαγκρανζιανή μέθοδο στην επίλυση της εξίσωσης συναγωγής, με αποτέλεσμα να εκτελεί ακριβέστερες προσομοιώσεις. Επιπροσθέτως, η ενσωμάτωση δεδομένων υψηλής ανάλυσης από τη βιβλιογραφία μέσω *nudging* βελτίωσε σημαντικά την ακρίβεια του επιλύτη και την πιστότητα της ροής.

Στη συνέχεια, ο επιλύτης Ταχείας Υπολογιστικής Ρευστομηχανικής εφαρμόστηκε σε ένα μοντέλο οδικής χαράδρας, όπου εφαρμόστηκαν δύο “φθηνά” μοντέλα τύρβης. Παρά τις αρχικές προσδοκίες, τα μοντέλα αυτά παρουσίασαν κακή απόδοση, αποτυγχάνοντας να αποτυπώσουν με ακρίβεια τη δυναμική των ρευστών μέσα στο αστικό περιβάλλον. Για το λόγο αυτό, η μέθοδος *nudging* χρησιμοποιήθηκε και πάλι για να βελτιωθεί η ακρίβεια του μοντέλου. Σε αυτή την υλοποίηση, πειραματικά δεδομένα από την βιβλιογραφία αξιοποιήθηκαν για την ενημέρωση και την καθοδήγηση της διαδικασίας προσομοίωσης. Με την αφομοίωση αυτών των εμπειρικών δεδομένων μέσω της τεχνικής *nudging*, το μοντέλο προσαρμόστηκε αποτελεσματικά ώστε να αντικατοπτρίζει καλύτερα τις ιδιαιτερότητες των πραγματικών συνθηκών ροής που επικρατούν στις οδικές χαράδρες.

Συνδυάζοντας την αποδοτικότητα της Ταχείας Υπολογιστικής Ρευστομηχανικής με την αφομοίωση δεδομένων μέσω του *nudging*, αυτή η έρευνα καταδεικνύει μια πολλά υποσχόμενη προσέγγιση για πρακτικές προσομοιώσεις ροής ρευστών. Υπογραμμίζει την ανάγκη εξισορρόπησης της ακρίβειας και της υπολογιστικής ταχύτητας, αναγνωρίζοντας τους εγγενείς συμβιβασμούς.

Λέξεις Κλειδία:
Ταχεία Υπολογιστική Ρευστομηχανική, Ημι-Λαγκρανζιανό Σχήμα, Αφομοίωση Δεδομένων, Nudging, OpenFOAM

Abstract

Fast Fluid Dynamics (FFD) is a computational technique designed for efficiently solving the incompressible Navier-Stokes equations. Unlike traditional methods which prioritize absolute accuracy, FFD places a premium on computational speed, making it particularly adept at scenarios where rapid simulations are essential. While it may sacrifice some precision, its strength lies in its ability to provide visually appealing and dynamic fluid behavior in real-time applications.

On the other hand, data assimilation is the science of combining different sources of information to predict possible states of a system as it progresses with time. In this context, nudging is a dynamic method employed in various fields, including Computational Fluid Dynamics (CFD), to adjust a model's state toward observed data by introducing a feedback term.

This thesis presents an innovative and computationally efficient method for simulating 2D incompressible flows using FFD on structured grids. To mitigate potential inaccuracies inherent in FFD, a cost-effective nudging method is integrated, which assimilates data from external sources to enhance numerical results and compensate for the solver's limitations in accurately modeling physical phenomena at a detailed level.

The implementation of this approach involved developing a custom C++ solver to deploy the FFD algorithm and nudging technique. Initially, the solver's efficiency was rigorously evaluated on a benchmark lid-driven cavity flow, showcasing close agreement with established literature. Departing from conventional FFD methods, the solver incorporated a high-order scheme for the semi-Lagrangian method in solving the convection equation, resulting in more accurate simulations. To validate the results and assess the solver's performance, comparisons were made with similar simulations conducted using OpenFOAM. Additionally, integrating high-resolution data from literature through nudging played a crucial role in significantly improving the solver's accuracy and the fidelity of the flow.

Subsequently, the FFD solver was applied to a street canyon model, where two cost-effective turbulence models were implemented. Despite initial expectations, these models exhibited poor performance, failing to accurately capture real-world fluid dynamics within the urban environment. In response to these shortcomings, the nudging method was once again employed to enhance the model's accuracy. In this implementation, experimental data obtained from experiments were leveraged to inform and guide the simulation process. By assimilating this empirical data through the nudging technique, the model was effectively adjusted to better reflect the intricacies of the actual flow conditions experienced in street canyons.

By synergizing FFD's efficiency with data assimilation through nudging, this research demonstrates a promising approach for practical fluid flow simulations. It underscores the need to balance accuracy and computational speed, acknowledging the inherent trade-offs therein.

Keywords:
Fast Fluid Dynamics, Semi-Lagrangian Scheme, Data Assimilation, Nudging, OpenFOAM

[illegible]

Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation - Problem Statement

Fluid Dynamics is a branch of physics that studies the motion of fluids (liquids and gases) and the forces acting on them. In various fields such as engineering, environmental science, and biomechanics, understanding fluid flow is crucial for optimizing designs and predicting system behaviors. Traditional methods for simulating fluid dynamics, such as finite element (FEA) and finite volume (FVM) methods, have proven effective but can be computationally expensive, particularly for large-scale and time-dependent simulations. For this reason, alternative techniques like Fast Fluid Dynamics are being increasingly used.

Similarly, Data Assimilation methods have found extensive application in meteorology, oceanography, geophysics, and various other fields for addressing large-scale simulations. In recent years, increased attention has been directed towards integrating experimental fluid dynamics (EFD) and computational fluid dynamics (CFD) to address their respective limitations. Unlike current EFD approaches that heavily rely on advanced measurement devices and high-order post-processing schemes, a dynamical model can be seamlessly integrated into data assimilation. This integration offers a distinct advantage over traditional CFD methods as uncertainties are mitigated by directly incorporating measured data, bypassing the need to exhaustively model and define boundary conditions.

The scope of this thesis is to construct and validate a Fast Fluid Dynamics solver, written in C++ and improve its performance and accuracy with Data Assimilation methods.

1.2 Literature Review

This literature review focuses on the evolution, advancements, and applications of Fast Fluid Dynamics methods, which aim to provide efficient and accurate solutions to fluid flow problems. Additionally, the review explores the role of data assimilation techniques in improving the accuracy and reliability of CFD simulations, by integrating observational data.

1.2.1 Fast Fluid Dynamics

Fast Fluid Dynamics (FFD) were initially proposed by Jos Stam [1] as a simple and rapid simulation of fluid-like motions in three-dimensional graphics and computer animations. In comparison to traditional CFD methods, FFD stands out for its computational efficiency and unconditional stability, allowing for larger simulation steps and, consequently, faster simulations. The stability advantage lies in both the implicit and Lagrangian approach of the Navier-Stokes equations. The semi-Lagrangian scheme was extensively reviewed by Staniforth and Côté in 1991 [2], highlighting that the time step is not constrained by the Courant-Friedrichs-Lewy (CFL) condition. In Stam's research, he identified a drawback in the Semi-Lagrangian scheme, which caused the flow to dampen too quickly. In response, he developed the software so as to allow the animators to introduce external forces to the fluid and dynamically update the flow in real-time. To enhance the quality of animations, the team incorporated texture into the flow and advected it with density. This involved assigning texture coordinates at the beginning of the simulation, which would then follow the density changes within the flow over time [3]. As a result, the gas in the animations appears more detailed, even when working with low-resolution grids.

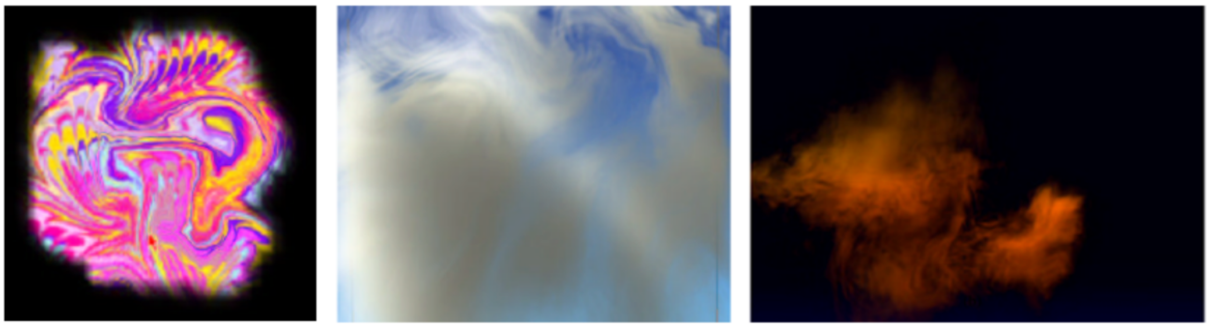


Figure 1.1: Snapshots from Stam's Interactive Solver [1]

However, the initially-developed FFD algorithm prioritized the shape and general behavior of the fluid, over the accuracy of physical quantities, and most benchmark tests focused on coarser grids and simpler turbulence models or laminar flows. In 2003 Fedkiw et al. [4] employed Stam's "Stable Fluids" algorithm for the simulation of smoke, a highly complex and turbulent phenomenon. The results showed poor performance of FFD, in terms of accuracy, mainly due to the numerical dissipation of the semi-Lagrangian scheme, which weakened the vortices in the flow. To alleviate this problem, they used a vorticity confinement method to inject the lost energy of the simulation. Essentially, this method identifies specific locations in a flow field where small-scale rotational and turbulent structures need to be generated. These features are then added in a physically realistic manner to create a passive-looking smoke on coarse grids. Additionally, the updated model efficiently handles solid boundaries within the computational domain and effectively manages the interaction of the flow with objects.

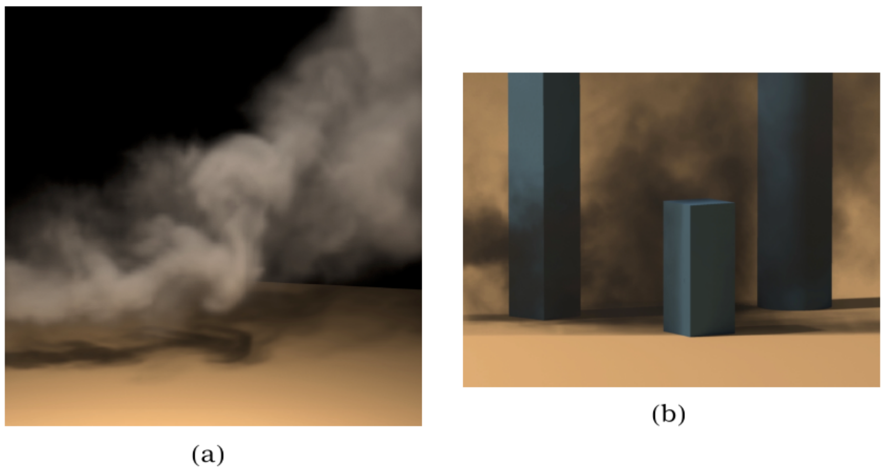


Figure 1.2: (a) Rising Smoke with Vortices Preserved with Vorticity Confinement Method ; (b) Smoke Interaction with Several Objects [4]

Harris el al. [5] drew upon the work of both Stam and Fedkiw to model the behavior of clouds, in terms of dynamics and radiometry. His research utilized GPUs to conduct these simulations and concluded that they were approximately 5× faster than their CPU counterparts. GPUs, or Graphics Processing Units, are specialized hardware components designed to accelerate the processing of graphics and were originally developed to handle the complex calculations involved in rendering graphics for video games and other graphical applications. Programmable GPUs are specifically designed for computations on pixels, and these pixels can effectively represent a grid of cells and CFD simulations can also take advantage of the GPU's ability to perform matrix operations in parallel.

In 2009 Zuo et al. [6] introduced an FFD method for indoor flows, in hope of developing a faster than real-time simulation model for smoke and contaminant management in fire emergencies. The goal was to find a compromise between traditional CFD, which was very expensive computationally, and nodal models, which failed to provide detailed information of the flow. Indeed, FFD was found to be 50× times faster [7] and was an adequate tool for preliminary investigations, especially for laminar flow simulations. However, when dealing with turbulent phenomena, investigation for cheap turbulence models, proved that FFD employed with zero equation models overestimated the turbulence viscosity and could not compete with a conventional CFD $k-\epsilon$ simulation [6].

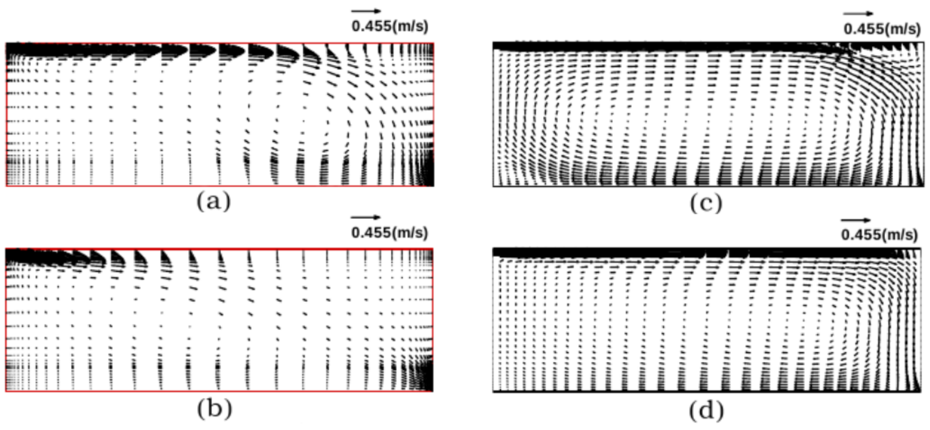


Figure 1.3: (a) Laminar Flow FFD ; (b) Zero Equation FFD ; (c) Laminar CFD ; (d) $k-\epsilon$ CFD [6]

After further research, Zuo et al. [8] proposed a few improvements for indoor modeling like a mass conservation correction function, as mass imbalance was an inherent disadvantage of the semi-Lagrangian scheme. They speculated that the mass conservation violation could be attributed to the pressure correction step in the FFD algorithm, which, contrary to the SIMPLE algorithm, was performed only once. Additionally, the performance of FFD on a GPU was again investigated and it was found, that, as a whole, FFD on GPU can be $500-1500\times$ times faster than on CPU [8]. This allows for real-time simulation and provides a very powerful tool for inverse design processes.

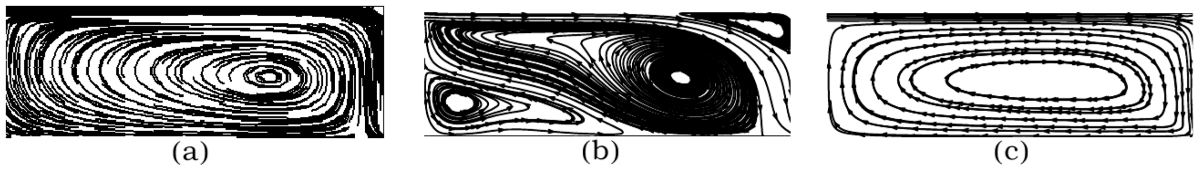
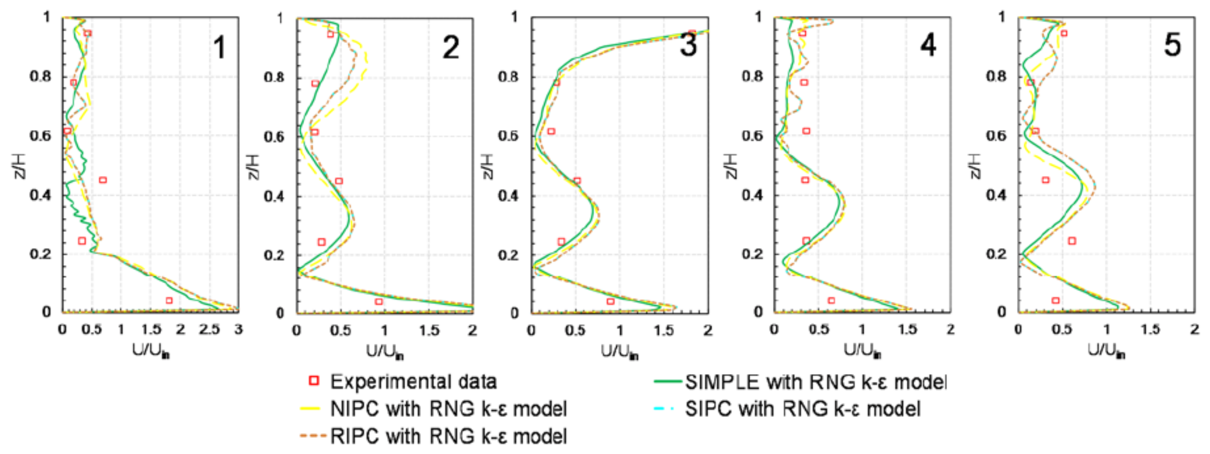
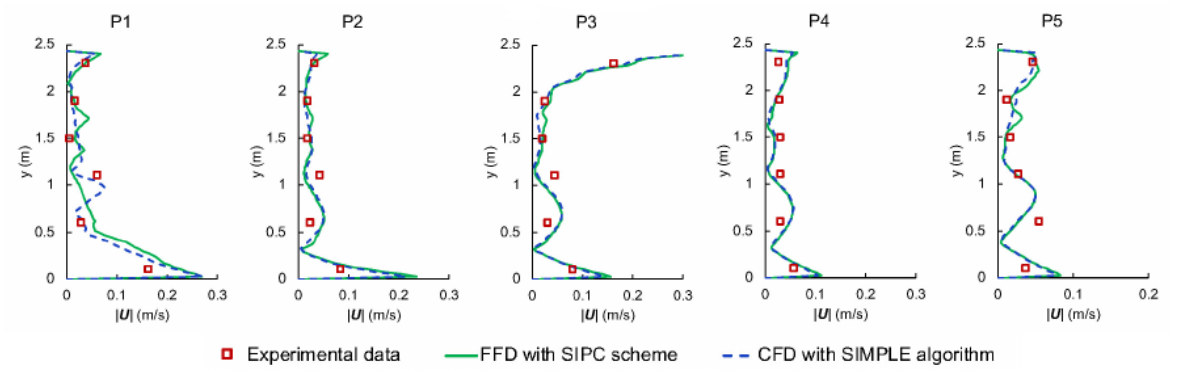


Figure 1.4: (a) $k-\epsilon$ CFD ; (b) FFD without Mass Correction ; (c) FFD with Mass Correction [8]

Inverse design processes involve working backward from desired specifications or performance requirements to generate a design that meets those criteria. This often involves the use of optimization algorithms, such as genetic algorithms or gradient-based methods like adjoint optimization. Inverse design requires tens of simulations, either for examining a potential solution in genetic algorithms or for solving the Navier-Stokes and the adjoint equations sequentially in each design cycle. Liu et al. [9] combined an adjoint optimization technique with FFD for indoor environments which proved to be $4 - 10\times$ times faster than CFD-adjoint. They explored various turbulence models and pressure correction techniques to compensate for the numerical diffusion in the Semi Lagrangian scheme and found good performance of the algorithm for both the steady-state and transient flow. The least favorable results were noted when employing a non-incremental pressure-correction scheme with SL scheme. Adjoint optimization is effective in identifying the local optima but relies heavily on the initial values for the design variable, making genetic algorithms a preferable alternative. However, genetic algorithms are computationally slower than the CFD-adjoint method. To address this, Xue et al. [10] aimed to incorporate FFD into GA optimization. In their study on indoor ventilation optimization, they reported a substantial 75% reduction in computational costs, when using a FFD based genetic algorithm.



(a)



(b)

Figure 1.5: Indoor Ventilation Results (a) Liu et al. [9] ; (b) Xue et al. [10]

In 2019, Tial et al [11] conducted a comparative analysis of FFD and CFD simulations in data center applications. With identical simulation parameters, such as grid resolution, turbulence method, and discretization schemes, the study emphasizes that both methods capture the re-circulation of airflow and show minimal differences in the pressure field.

It is important to note that, unlike prior studies that utilized a Semi-Lagrangian scheme, they employed a first-order upwind scheme for advection, highlighting the SL method drawbacks. Lastly, efforts were made to employ FFD for outdoor simulations. Ting et al. [12] reviewed various turbulence models and reported that the Smagorinsky and dynamic Smagorinsky model was the best fit as it was as fast and accurate as the RNG $k - \epsilon$ model. The RNG $k - \epsilon$ tubulence model solves two differential equations to calculate the turbulent kinetic energy and its dissipation, while the Smagorinsky model adds an artificial viscosity to account for the smaller scale turbulent eddies.

1.2.2 Data Assimilation in Computational Fluid Mechanics

Data Assimilation (DA) is a statistical method used to enhance the accuracy of numerical simulations by incorporating real-time measurement data into the simulation process. In this approach, the error term of the numerical simulation is characterized by a probability distribution, treating the simulation as a "system model". When experimental values representing true dynamics are introduced, the probability distribution of error is adjusted to reduce uncertainty in the simulation [13].

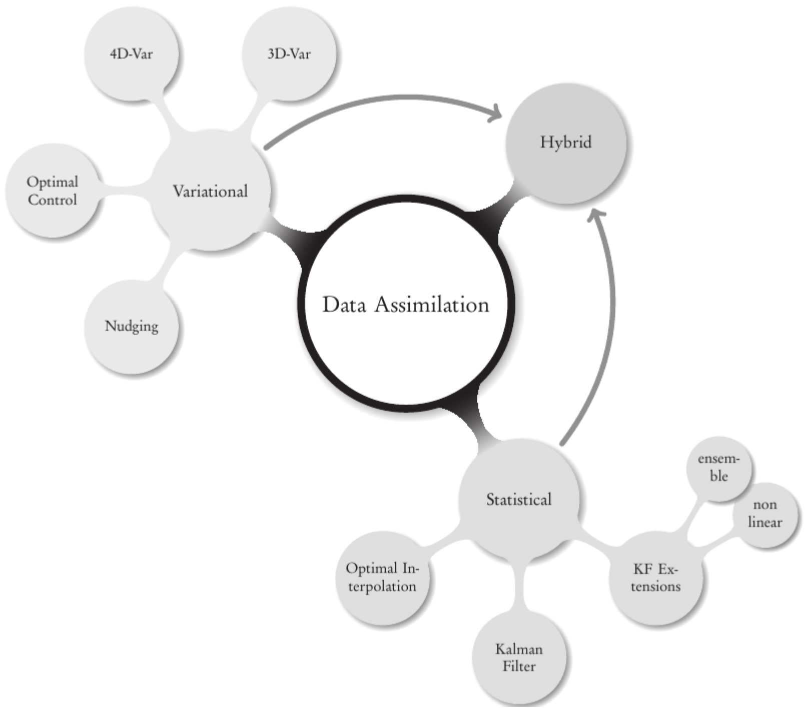


Figure 1.6: Schematic of Data Assimilation Methods [14]

There are two fundamental approaches to data assimilation: sequential or statistical assimilation, based on statistical estimation theory and Bayes's law, and non-sequential or variational assimilation, rooted in optimal control theory [14]. Sequential data assimilation involves updating simulation estimates continuously at each observation time, while variational data assimilation methods aim to find the optimal solution for a numerical model by fitting it with a set of observations over a specific time span. In statistical assimilation, the focus is on minimizing variance, while in variational assimilation, the emphasis is on minimizing a suitable cost or error function. Despite the statistical approach often being more intricate and time-consuming, it has the potential to provide a richer information structure. Ideally, a unified or hybrid approach should be pursued, combining both sequential and non-sequential methods. By doing so, one can take advantage of the quick and robust nature of the variational approach, while simultaneously obtaining a more information-rich solution through the statistical/probabilistic approach.

The most common techniques for sequential data assimilation include the Kalman filter, which dynamically adjusts the state estimate and its uncertainty. It involves a prediction step, where the model anticipates the system's behavior, and an update step, where actual observations are used to correct and refine the predicted state. Meldi et al. [15] incorporated a Kalman filter-based sequential estimator into the PISO algorithm of a segregated solver designed for handling incompressible flows. This integration enables the generation of an augmented flow state that considers both the model's confidence and the provided observations. The initial investigation focused on a two-dimensional flow around a cylinder with a Reynolds number of $Re = 100$ and reported a reduction in computational cost of $10 - 15\%$, combined with efficient predictions. Additionally, an improvement to the Kalman filter is the Ensemble Kalman Filter (EnKF), which uses a Monte Carlo approach by representing the state estimate as an ensemble of samples. It maintains and updates a set of ensemble members through the prediction and update steps and is particularly useful for high-dimensional systems. In the study conducted by Kato et al. [16], the ensemble Kalman filter was utilized in the context of a wind tunnel experiment featuring a square cylinder. The experiment involved the generation of a von Kármán vortex street, representing a quasi-unsteady flow. Pressure measurements on the front surface and lateral faces of the square cylinder were obtained through pressure ports and additional measurements with Laser Doppler Velocimetry (LDV) were conducted for the velocity data.

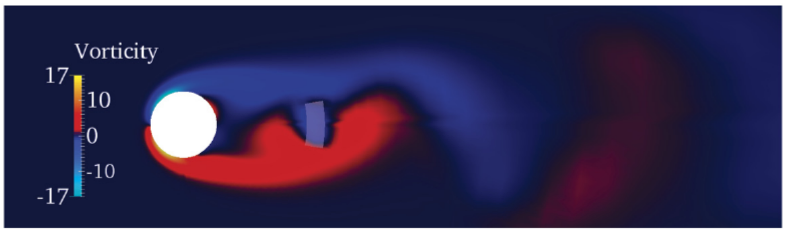


Figure 1.7: Instantaneous Vorticity Isocontours during the First Phases of Data Assimilation via Estimator. The Model Acts as a Forcing while Synchronizing the Numerical Model to the Observation [15]

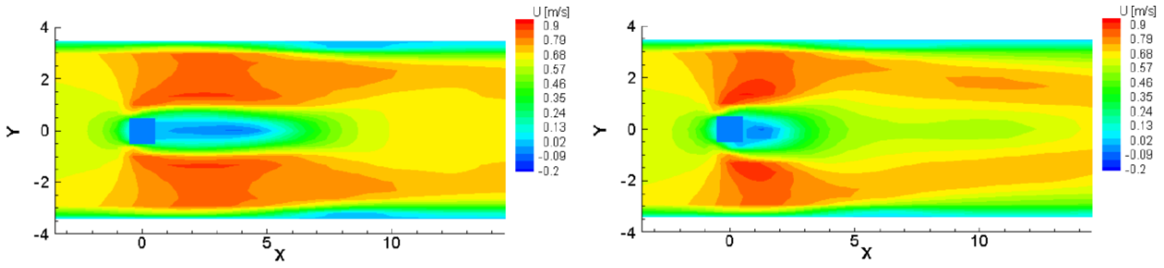


Figure 1.8: Mean Velocity Comparison of Numerical Simulation (left) and EnKF Implementation (right) [16]

Regarding variational data assimilation methods, they can be classified into 3D-Var and 4D-Var, depending on the spatial dimensions of the simulation and the inclusion, or not, of a temporal window for the system's dynamical evolution. Both methods demand substantial computational power, limiting their application to simple 2D flows in fluid mechanics. Chandramouli et al. [17] utilized a 4D-Var approach to reconstruct time-resolved, incompressible turbulent flows from measurements on two orthogonal 2D planes. The algorithm proved successful in a 3D turbulent wake flow, and validation was conducted on a synthetic 3D dataset acquired with experimental Particle Image Velocimetry (PIV) observations. Variational methods are often coupled with adjoint optimization for inverse design, where model parameters such as initial conditions are optimized to match measurements. In Symon et al.'s [18] paper, variational methods combined with adjoint optimization were employed to minimize the differences in mean velocity fields between a Direct Numerical Simulation (DNS) and an incompressible Reynolds-Averaged Navier-Stokes (RANS) simulation for a 2D flow around an idealized airfoil at high Reynolds numbers. The implemented direct-adjoint optimization procedure reported good agreement with the PIV experimental results.

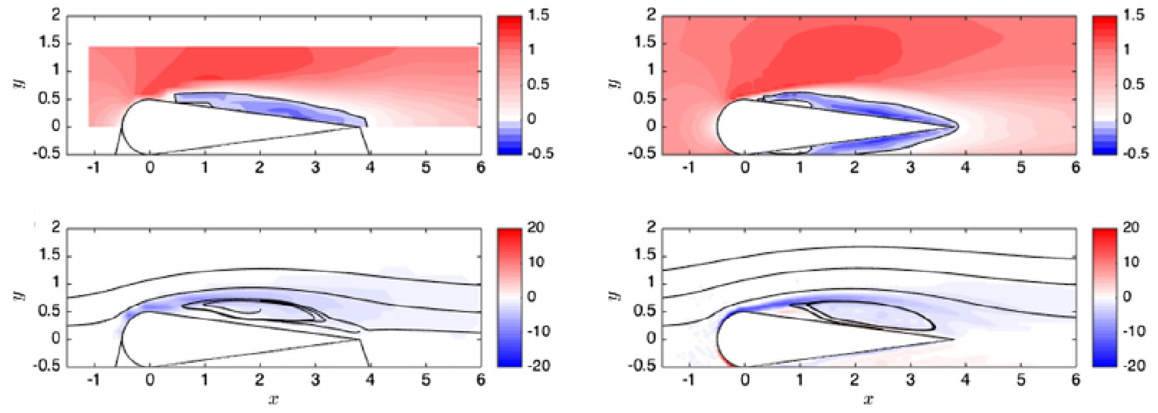


Figure 1.9: Comparison of Streamwise Velocity (top) and Vorticity (bottom) between the Experimental Mean Flow (left) Obtained from PIV and the Data-Assimilated Flow (right) around an Idealized Airfoil [18]

Finally, nudging is a variational method that shares some similarities with Kalman filter and often referred to as a state observer technique in control theory. It has a negligible computational cost compared to a standard simulation and is quite versatile, allowing users to fine-tune the nudging weight. In the context of Computational Fluid Dynamics (CFD), the utilization of nudging techniques dates back to 1997 [19], when a state observer provided feedback to the boundary conditions in a SIMPLER simulation of a fully developed turbulent flow through a square cross-section duct. The judicious selection of an appropriate gain led to an acceleration of convergence by a factor of 0.012. In 2010, Imagawa et al. [20] improved the application of nudging in the same flow problem, and were able to minimize uncertainties in initial conditions or numerical errors arising from the use of a coarser grid. Building upon these advancements, Zauner et al. [21] implemented nudging in an URANS solver for unsteady flow around a cylinder at $Re = 22000$, based on sparse velocity observations from DNS. Their findings revealed that the nudged URANS solver outperformed traditional URANS simulations without nudging, particularly in accurately capturing small-scale Kelvin-Helmholtz phenomena and vortex shedding, as shown in Figure 1.10.

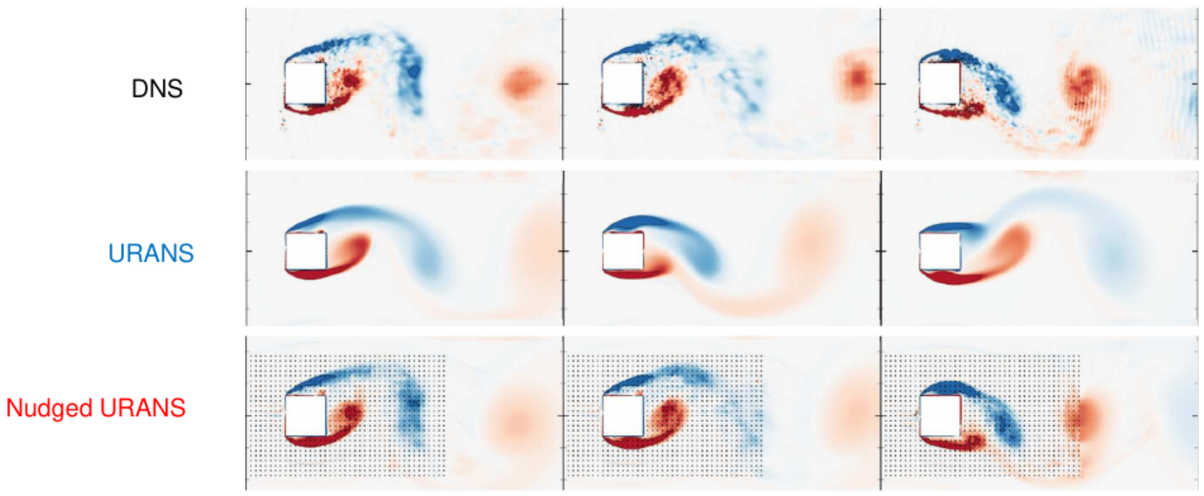


Figure 1.10: Comparative Results of Nudged URANS around a Square Cylinder for $Re = 22000$ [21]

1.3 Thesis Outline

After the opening chapter, the structure of the thesis is organized in the following manner:

- **Chapter 2**

  This chapter introduces the Fast Fluid Dynamics (FFD) algorithm, which serves as the primary numerical solution method for analyzing fluid flows in this thesis. It outlines the fundamental concepts and the governing equations of fluid flows, followed by their discretization through the Finite Differences Method. This chapter lays the groundwork for the subsequent numerical analysis and solution of fluid flow problems throughout the thesis.

- **Chapter 3**

  The nudging method is introduced and thoroughly explained, with a focus on its integration into the FFD solver. This chapter provides a comprehensive understanding of the nudging method and its practical implementation within the context of fluid dynamics analysis.

- **Chapter 4**

  This chapter investigates the lid-driven cavity benchmark test for incompressible flows. The case study is employed for validating the FFD solver and evaluating its performance in comparison to existing literature and OpenFOAM simulations. In addition, this chapter examines and evaluates the practical application of nudging.

- **Chapter 5**

  This chapter focuses on the application of the FFD algorithm in the dynamic context of a street canyon. Additionally, various turbulence models are explored, comparing their impact on accuracy and computational efficiency. The chapter concludes with an analysis of the nudging method's integration.

- **Chapter 6**

  The key findings and conclusions of the thesis are presented, accompanied by recommendations for future work in the studied applications and developed codes.

2. Fast Fluid Dynamics

2.1 Governing Equations

The Navier-Stokes equations are a set of partial differential equations that describe the behavior of fluid flow. They are derived from the fundamental principles of fluid mechanics, including Newton's Second Law of motion. The equations account for the conservation of momentum within a fluid, and they also account for the viscous forces that arise due to internal friction within the fluid. The full form of the Navier-Stokes equations for an incompressible fluid can be written as follows:

$$\nabla \cdot \vec{U} = 0 \tag{2.1}$$

$$\overbrace{\frac{\partial \vec{U}}{\partial t} + \underbrace{(\vec{U} \cdot \nabla)\vec{U}}_{\substack{\text{Acceleration/}\\\text{Convection}}}}^{\substack{\text{Material}\\\text{Derivative}}} = \underbrace{-\frac{1}{\rho}\nabla P}_{\substack{\text{Pressure}\\\text{Gradient}}} + \overbrace{\nu \nabla^2 \vec{U}}^{\substack{\text{Viscous}\\\text{Force/}\\\text{Diffusion}}} + \underbrace{\vec{f}}_{\substack{\text{External}\\\text{Forces}}} \tag{2.2}$$

where:
$\vec{U}$: the 2D velocity field (u, v)
P: pressure
ρ: fluid density
ν: kinematic viscosity of the fluid
$\vec{f}$: external forces vector

These equations describe how the velocity field $\vec{U}$ and pressure P in a fluid evolve over time, subject to the forces and initial/boundary conditions. The first equation, the continuity equation, enforces mass conservation by ensuring that the divergence of the velocity field is zero, whereas the second equation, the momentum equation, accounts for the acceleration of fluid particles, the forces due to pressure difference, the viscous forces and any external forces (e.g gravity or thermal forces).

2.2 Finite Difference Method

The Finite Difference Method (FDM) is a numerical technique used to solve partial differential equations (PDEs), such as the Navier-Stokes equations. In the context of FDM, the continuous spatial and time domains are divided into a grid of discrete points where the dependent variables are computed and stored. At each of these grid points, the derivatives are written in the form of differences, which relate the value at each point to its neighbors, using the truncated Taylor series expansion. As a result, the initial set of equations is transformed into a system of algebraic equations, which is subsequently solved with numerical methods. [22]

The truncated Taylor series expansion (2.3) plays a pivotal role in FDM. It expresses a function as an infinite series of derivatives evaluated at a particular point, x_0, and is truncated to a finite number of terms to make it computationally feasible. The remainder term $R_n(x)$ represents the truncation error, which accounts for the difference between the true function and its truncated approximation.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + \underbrace{R_n(x)}_{\substack{\text{Truncation}\\\text{Error}}} \quad (2.3)$$

The truncation error heavily depends on the term $(x - x_0)$, which in FDM, is the distance between two neighboring points on the grid. A small distance, meaning a fine mesh, implies that higher order terms are negligible, except in situations where higher derivatives are locally large [22]. Increasing the number of terms in the truncated Taylor series generally enhances the accuracy of the approximation, as it involves the information from more neighbors. However, this improvement comes at the cost of increased computational resources.

Contrary to the finite volume method (FVM), the nature of FDM approach does not automatically satisfy the conservation principles and is best suited for structured and orthogonal grids. To address this limitation, a staggered grid is often chosen as it is equivalent to FVM and obeys global conservation, as pointed out by Konangi et al. [23].

2.2.1 Mesh Configuration

The FFD algorithm is implemented using Finite Differences Methods on a staggered grid, introduced by Harlow and Welsh [24] in 1965. This is different from a collocated grid arrangement, where all variables are stored in the same positions. In a "velocity" staggered mesh, pressure is typically stored at the cell centroid and the velocity components are stored at the center of the edges of the grid cells. The staggering is not a diagonal procedure; rather, each velocity component has its own shifted grid, with the u-grid shifted to the right (x-direction) and the v-grid shifted up (y-direction). As a result, there are essentially three different grids.

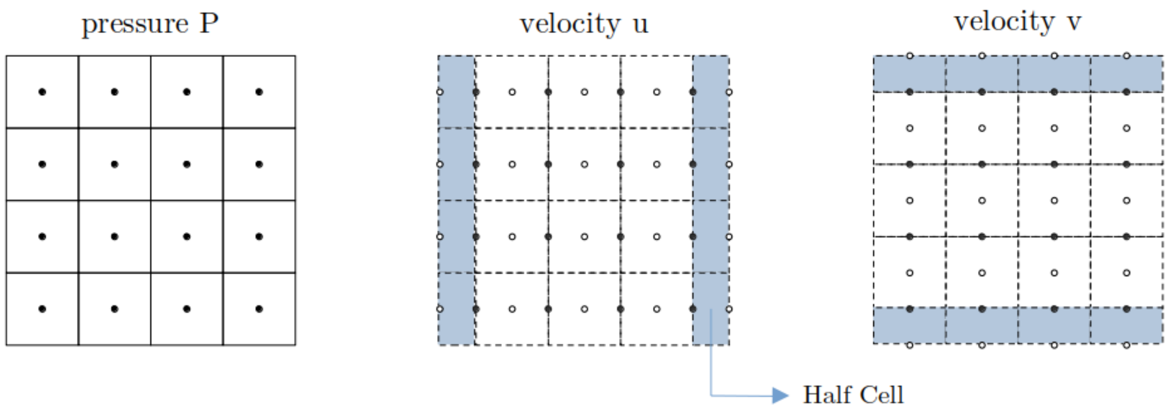


Figure 2.1: Schematic of Staggered Grid - Shifted Velocity Grids

Staggered grids are a useful solution to prevent odd-even decoupling problems in CFD, eliminating checkerboard patterns in solutions. However, they come with the disadvantage of storing different variables in separate locations, making it challenging to manage distinct control volumes and track metrics effectively. In contrast, modern codes often opt for collocated storage due to the Rhie & Chow [25] interpolation method. This widely used technique on collocated meshes helps suppress non-physical pressure oscillations linked to checkerboard effects, while maintaining computational efficiency.

The shifting of the original grid leads to the appearance of half cells near the boundaries of the velocity grids. This introduces implications for the indexing of the nodes and the dependent variables due to the presence of an additional node.

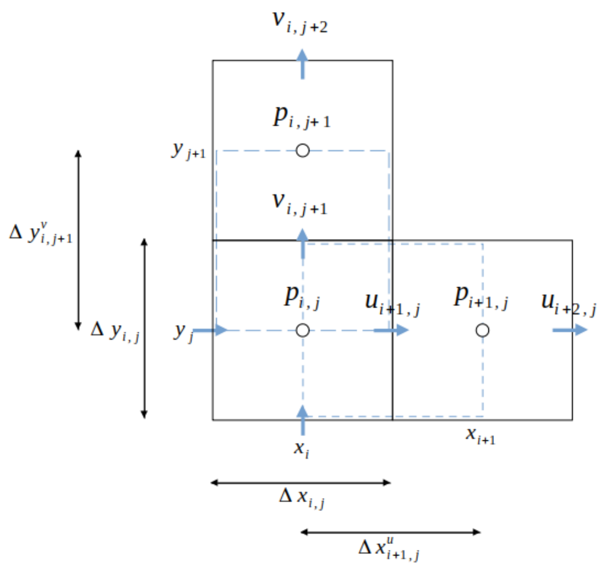


Figure 2.2: Example of Sizing and Shifting of a Single Cell on a Staggered Grid

The grid points are identified by an index i that runs in the x-direction and an index j for the y-direction. The edges of each cell are located so as to position the grid point at the cell center, contrary to having the edges midway between two adjacent grid points, as shown in the figure below.

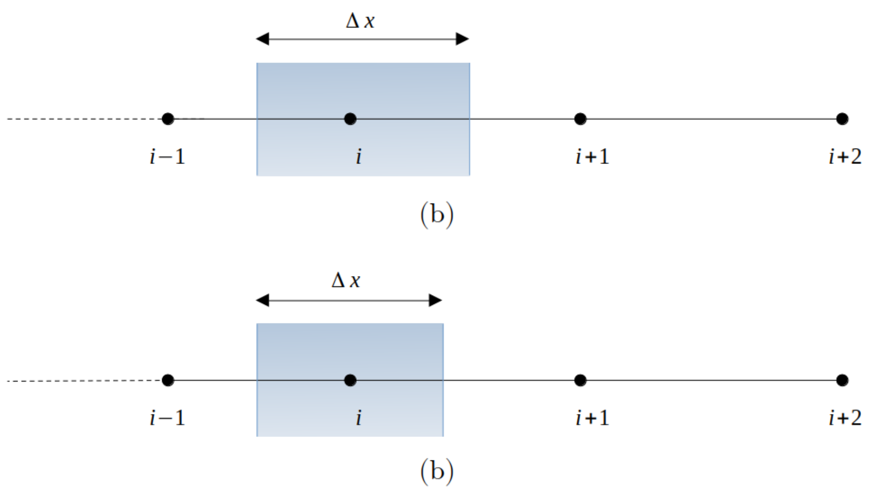


Figure 2.3: 1D Case of Cell Size for (a) Mid-Distance Edges and (b) Node at Cell Center

Staggered grids are particularly well-suited for FFD applications due to their strategic placement of variables. In this grid structure, an effective approximation of the first derivatives of velocity is achieved by employing centered differences around half grid points. This approach not only ensures good accuracy in the calculations but also mitigates the risk of numerical instabilities. As a result, first derivatives "live" at the cell centers [26], where the pressure grid points are originally located. This alignment proves to be pivotal for the pressure correction method, which is a key component in the later stages of FFD simulations. For instance, the following equation of a centered differences around $u_{i+1/2,j}$ node is a good approximation for the first derivative of u at the position of pressure $P_{i,j}$.

$$\left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2},j} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x_{i,j}} \tag{2.4}$$

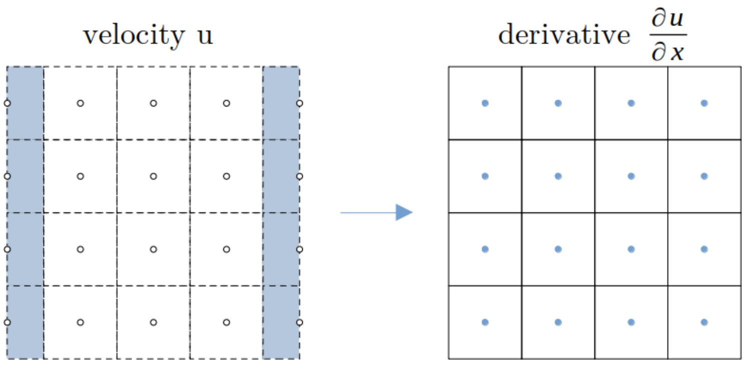


Figure 2.4: First Derivative's Computational Location with a Staggered Grid Approximation

Furthermore, the arrangement of staggered grids facilitates the application of a straightforward averaging interpolation method, without compromising accuracy. As a result, the translation of velocity values to the center of the original grid is usually accomplished through a simple averaging interpolation of the velocity values at the edges of the corresponding cell. This approach not only helps maintain accuracy but also plays a crucial role in effectively handling non-linear convective terms.

$$u_{i+\frac{1}{2},j} = \frac{u_{i+1,j} + u_{i,j}}{2}, u_{i,j+\frac{1}{2}} = \frac{u_{i,j+1} + u_{i,j}}{2} \tag{2.5}$$

$$v_{i,j+\frac{1}{2}} = \frac{v_{i,j+1} + v_{i,j}}{2}, v_{i+\frac{1}{2},j} = \frac{v_{i+1,j} + v_{i,j}}{2} \tag{2.6}$$

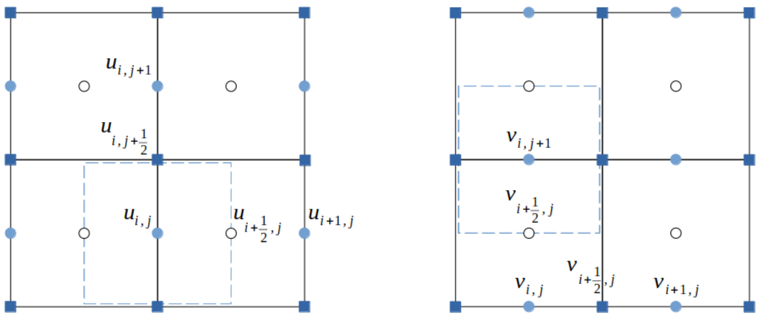


Figure 2.5: Positions of Common Interpolated Values in Staggered Grid

2.2.2 Boundary Conditions Handling

In the implementation of boundary conditions, an additional layer of *ghostnodes* is generated around the perimeter of the original mesh. This augmentation facilitates the handling of Neumann boundary conditions (BCs). For no-slip velocity BCs, Neumann BCs are commonly employed for pressure, resulting in the "mirroring" of pressure values at the boundary and the corresponding ghost node cells. In the case of Dirichlet velocity BCs, the staggered grid configuration allows for the averaging of velocity values on each side of the boundary, as illustrated in Figure 2.6, while for pressure Dirichlet conditions, it is accepted that the edge and the center of the cell have the same pressure. In the example case:

$$u_{west} = u_{1,3} \tag{2.7}$$

$$u_{east} = \frac{u_{2,1} + u_{2,2}}{2} \tag{2.8}$$

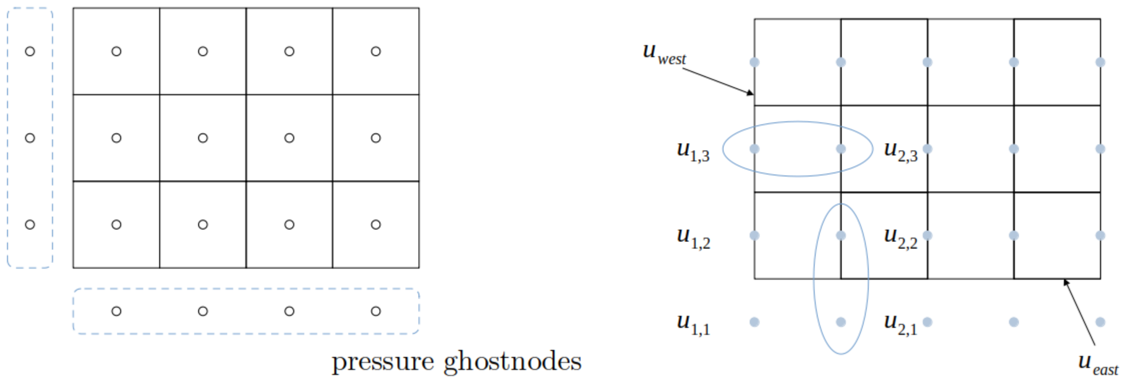


Figure 2.6: Handling of Boundary Conditions [27]

2.3 FFD Algorithm

FFD adopts a fractional step method for solving the Navier-Stokes equations, which decouples the pressure and the velocity. At first, intermediate velocity fields are computed from the momentum equation's forces, separately, ignoring the incompressibility constraint. Subsequently, a pressure correction method is used to project the intermediate velocity field into a a divergence-free vector field to obtain the pressure and update the velocity field.

$$\frac{\partial \vec{U}}{\partial t} = \nu \nabla^2 \vec{U} + \vec{f} \tag{2.9}$$

$$\frac{\partial \vec{U}}{\partial t} + (\vec{U} \cdot \nabla)\vec{U} = 0 \tag{2.10}$$

$$\nabla^2 P = \frac{\rho}{\Delta t} \nabla \cdot \vec{U} \tag{2.11}$$

$$\frac{\partial \vec{U}}{\partial t} = -\frac{1}{\rho} \nabla P \tag{2.12}$$

The calculation procedure of FFD is as follows:

Step 1. Diffusion/Source equation (2.9) is solved implicitly with Gauss-Seidel iterative method to obtain an initial intermediate velocity field.

Step 2. Convective equation (2.10) is solved by the Semi-Lagrangian method proposed by M. Dorostkar [28] to obtain the second intermediate velocity field

Step 3. Poisson equation of pressure (2.11) is also solved implicitly with Gauss-Seidel iterative method

Step 4. Pressure Gradient equation (2.12) is solved explicitly, by marching through time with a time step Δt

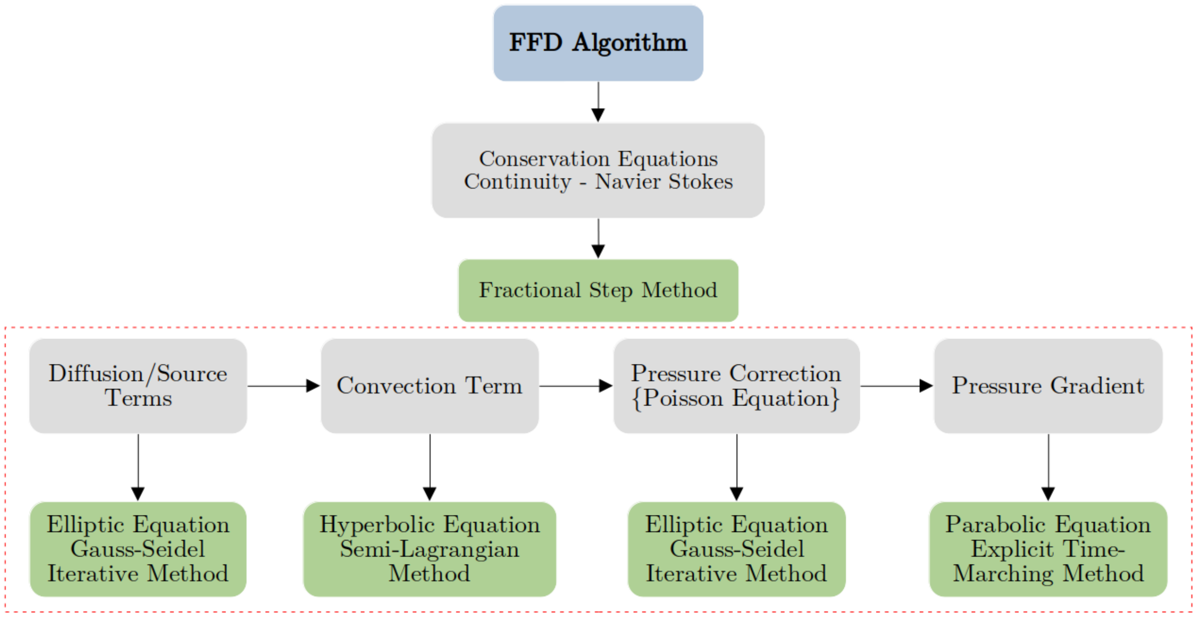

FFD Algorithm
Conservation Equations Continuity - Navier Stokes
Fractional Step Method
Diffusion/Source Terms
Convection Term
Pressure Correction {Poisson Equation}
Pressure Gradient
Elliptic Equation Gauss-Seidel Iterative Method
Hyperbolic Equation Semi-Lagrangian Method
Elliptic Equation Gauss-Seidel Iterative Method
Parabolic Equation Explicit Time-Marching Method

Figure 2.7: Schematic of Fast Fluid Dynamics Algorithm [28]

2.3.1 Source and Diffusion Term

The FFD algorithm begins with the equation for external forces and the diffusion of the flow (2.9). In this work, the only force taken into consideration is the gravitational force along the y-axis.

$$\vec{f} = (0, -g) \tag{2.13}$$

where:
g: gravity's acceleration

The diffusive term accounts for the internal frictional forces and, thus, the turbulence present in the flow. Based on the application, there are appropriate turbulence models.
The equation is descritized using a first order forward scheme for the time derivative (2.14) and a second order central difference (2.15) approximation for the Laplace operator.

$$\frac{\partial \vec{U}}{\partial t} = \frac{\vec{U}^{\star}_{i,j} - \vec{U}^{n}_{i,j}}{\Delta t} \tag{2.14}$$

$$\nabla^2 \vec{U} = \frac{\vec{U}^{\star}_{i+1,j} - 2\vec{U}^{\star}_{i,j} + \vec{U}^{\star}_{i-1,j}}{\Delta x^2_{i,j}} + \frac{\vec{U}^{\star}_{i,j+1} - 2\vec{U}^{\star}_{i,j} + \vec{U}^{\star}_{i,j-1}}{\Delta y^2_{i,j}} \tag{2.15}$$

The "star" notation ($\star$) is used for the intermediate velocity fields, and the upper index n denotes the timestep numbering.

Equation (2.15) is used for meshes with equal spacing between the nodes. For non-uniform structured grid cases, the spatial derivatives are derived from the Taylor series. For the 1-dimensional case, the Taylor series expansion for $f(x_i + \Delta x_{i+1})$ and $f(x_i - \Delta x_i)$ in terms of $f(x_i)$ is:

$$f(x_{i+1}) = f(x_i) + \Delta x_{i+1} f'(x_i) + \frac{\Delta x^2_{i+1}}{2} f''(x_i) + O(\Delta x^3_i) \tag{2.16}$$

$$f(x_{i-1}) = f(x_i) - \Delta x_i f'(x_i) + \frac{\Delta x^2_i}{2} f''(x_i) - O(\Delta x^3_i) \tag{2.17}$$

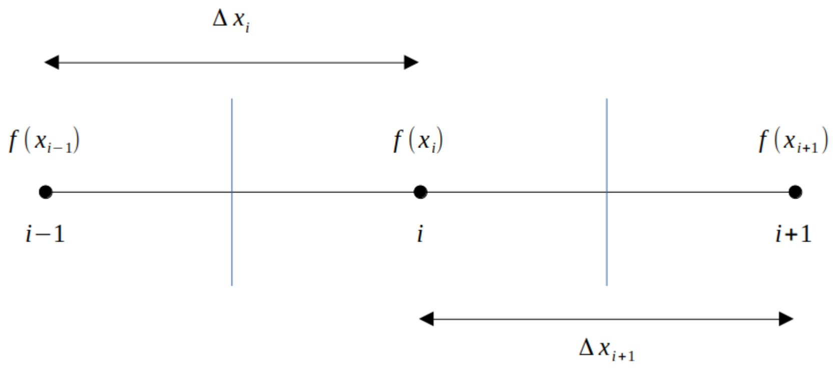


Figure 2.8: Schematic of the 1D Taylor Series Expansion

By summing the two Taylor expressions and eliminating the first derivative terms, the second derivative equals:

$$\begin{aligned} f''(x_i) = 2\left[\frac{f(x_{i+1})}{\Delta x_{i+1}(\Delta x_{i+1} + \Delta x_i)} - \frac{f(x_i)}{\Delta x_{i+1}\Delta x_i} + \frac{f(x_{i-1})}{\Delta x_i(\Delta x_{i+1} + \Delta x_i)}\right] \\ + \frac{\Delta x_{i+1} - \Delta x_i}{3} f'''(x_i) + O(\Delta x^2_i) \end{aligned} \tag{2.18}$$

However, the resulting scheme exhibits first-order accuracy, and may lead to significant numerical instabilities when the step size is not adequately small or when the expansion ratio between adjacent cells is excessively large.

The general discretized form for equation (2.9) is as follows:

$$\begin{aligned}\frac{\vec{U}^{\star}_{i,j}-\vec{U}^{n}_{i,j}}{\Delta t} =& 2\nu\left[\frac{\vec{U}_{i+1,j}}{\Delta x_{i+1,j}(\Delta x_{i+1,j}+\Delta x_{i,j})}+\frac{\vec{U}_{i,j+1}}{\Delta y_{i,j+1}(\Delta y_{i,j+1}+\Delta y_{i,j})}\right]\\ &-2\nu\left[\frac{1}{\Delta x_{i+1,j}\Delta x_{i,j}}+\frac{1}{\Delta y_{i,j+1}\Delta y_{i,j}}\right]\vec{U}_{i,j}\\ &+2\nu\left[\frac{\vec{U}_{i-1,j}}{\Delta x_{i,j}(\Delta x_{i+1,j}+\Delta x_{i,j})}+\frac{\vec{U}_{i,j-1}}{\Delta y_{i,j}(\Delta y_{i,j+1}+\Delta y_{i,j})}\right]+\vec{f}\end{aligned} \tag{2.19}$$

$$\Rightarrow \alpha_1\vec{U}^{\star}_{i+1,j}+\alpha_2\vec{U}^{\star}_{i,j+1}+\beta_0\vec{U}^{\star}_{i,j}+\gamma_1\vec{U}^{\star}_{i-1,j}+\gamma_2\vec{U}^{\star}_{i,j-1}=-\left(\frac{\vec{U}^{n}_{i,j}}{\Delta t}+\vec{f}\right) \tag{2.20}$$

where:

$$\alpha_1 = 2\nu\left[\frac{1}{\Delta x_{i+1,j}(\Delta x_{i+1,j}+\Delta x_{i,j})}\right] \qquad \gamma_1 = 2\nu\left[\frac{1}{\Delta x_{i,j}(\Delta x_{i+1,j}+\Delta x_{i,j})}\right]$$

$$\alpha_2 = 2\nu\left[\frac{1}{\Delta y_{i,j+1}(\Delta y_{i,j+1}+\Delta y_{i,j})}\right] \qquad \gamma_2 = 2\nu\left[\frac{1}{\Delta y_{i,j}(\Delta y_{i,j+1}+\Delta y_{i,j})}\right]$$

$$\beta_0 = -2\nu\left[\frac{1}{\Delta x_{i+1,j}\Delta x_{i,j}}+\frac{1}{\Delta y_{i,j+1}\Delta y_{i,j}}\right]-\frac{1}{\Delta t}$$

Each linear equation is solved twice, once for each velocity component (u, v), at every grid point of the mesh. Equation (2.20) is written in matrix form and solved implicitly for the intermediate velocity $(\star)$ field, using a simple Gauss-Seidel Solver.

$$\underset{n\times n}{A^u}\cdot\underset{n\times 1}{u^\star}=\underset{n\times 1}{B^u}, \qquad \underset{n\times n}{A^v}\cdot\underset{n\times 1}{v^\star}=\underset{n\times 1}{B^v} \tag{2.21}$$

where:
n: the number of nodes in the mesh

In the developed solver, the grids utilized in applications, adhere to an orthogonal structured grid format, yet the overall code structure resembles that of an unstructured mesh. The indexing commences from the lower-left corner node, systematically tracing cells vertically while assigning unique identification numbers. An ID number is an identifier assigned to each cell in the mesh, in order to handle data efficiently and manipulate specific cells during the simulation. The index of each velocity value and each column of matrix A correspond to the ID number of the respective cell. Coefficient matrix A is a sparse square matrix $n \times n$, where each row represents a cell with ID number i and contains at most five **non-zero** elements, a set of $(\alpha_1, \alpha_2, \beta_0, \gamma_1, \gamma_2)^i$ for each cell. Four of these elements correspond to contributions from the four neighbors. The vertical tracing employed in the indexing process reveals a distinctive pattern: the *northern* and *southern* neighbors exhibit an ID difference of ± 1 compared to the examined cell. Consequently, in the matrix structure, these neighboring elements are strategically positioned one place apart from the diagonal entry. This positioning explains why parameters $\alpha_2, \beta_0, \gamma_2$ are grouped together in equation (2.22). The diagonal elements are strictly negative and the matrix is diagonally dominant, which enhances the likelihood of convergence when applying the Gauss-Seidel method. Near mesh boundaries, grid points may have fewer than four neighbors and the diagonal element is adjusted to satisfy boundary conditions while ensuring it is non-zero. Matrices A^u and A^v share similar expressions but differ due to their reliance on cell size. This distinction arises from using staggered grids, where velocity components are computed on separate meshes. The general form of matrix A is:

$$A = \begin{bmatrix} \beta_0^1 & \alpha_2^1 & 0 & 0 & 0 & \alpha_1^1 & \cdots & 0 \\ \gamma_2^2 & \beta_0^2 & \alpha_2^2 & 0 & \alpha_1^2 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \gamma_1^{n-2} & 0 & \gamma_2^{n-2} & \beta_0^{n-2} & \alpha_2^{n-2} & 0 \\ 0 & \gamma_1^{n-1} & 0 & 0 & 0 & \gamma_2^{n-1} & \beta_0^{n-1} & \alpha_2^{n-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & \gamma_2^n & \beta_0^n \end{bmatrix} \tag{2.22}$$

The velocity field (u, v) and vectors B are of length n with each index associated with the cell ID.

$$u^\star = \begin{bmatrix} u_1^\star \\ u_2^\star \\ \vdots \\ u_n^\star \end{bmatrix}, \quad v^\star = \begin{bmatrix} v_1^\star \\ v_2^\star \\ \vdots \\ v_n^\star \end{bmatrix}, \quad B^u = -\begin{bmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_n^n \end{bmatrix}, \quad B^v = -\begin{bmatrix} v_1^n - g \\ v_2^n - g \\ \vdots \\ v_n^n - g \end{bmatrix} \tag{2.23}$$

2.3.2 Convection Term

The semi-Lagrangian scheme is a numerical method used for solving the advection equation, which describes the transport of a scalar quantity ϕ (such as temperature, concentration, or velocity) by a fluid flow. In the Navier-Stokes equations ϕ corresponds to the velocity components (u, v). The advection equation can be written as follows:

$$\frac{\partial \phi}{\partial t} + \vec{U} \cdot \nabla \phi = 0 \tag{2.24}$$

The semi-Lagrangian scheme employs the method of characteristics, which involves the tracing of particle trajectories within the fluid flow to estimate the values of the transported quantity at future time increments, as along a fluid path, the scalar quantity ϕ is assumed to remain constant.

$$\frac{d\phi}{d\vec{S}} = 0 \tag{2.25}$$

where:

$\vec{S}$: characteristic curve

The “arrival points” are typically positioned at the grid points of the mesh (Eulerian points), while the values of the transported quantity at the “departure points” are calculated with an interpolation scheme of the surrounding Eulerian points, as they do not necessarily coincide with the grid points [29]. The scheme is advantageous for its stability, accuracy in capturing sharp gradients, and its ability to handle advection-dominated phenomena without the limitations of the Courant-Friedrichs-Lewy (CFL) stability condition. However, its poor performance in mass and momentum conversation is most likely attributed to the selected interpolation scheme. [30].

2.3.2.1 Method of Characteristics

In mathematics, the method of characteristics is a technique for solving hyperbolic partial differential equations, by reducing the PDEs to a system of 1^{st} order ordinary differential equations (ODEs) [31]. A characteristic curve of the PDE (2.24) is by definition a solution of the differential equation (2.26).

$$\frac{d\vec{S}}{dt} = \vec{U} \tag{2.26}$$

Assuming a constant velocity within one time step $[t^n, t^{n+1}]$, the integral function of the characteristic curve can be written as:

$$\begin{gathered}\vec{U}_{(t,\vec{x})} = \vec{U} + O(\Delta t) \\ \int\limits_{t^n}^{t^{n+1}} \vec{S} = \int\limits_{t^n}^{t^{n+1}} \vec{U} dt \Rightarrow \vec{S}^n = \vec{S}^{n+1} - \vec{U}\Delta t\end{gathered} \tag{2.27}$$

Characteristic Curve with Improved Accuracy

Equation (2.27) is the conventional way of calculating the departure points in the semi-Lagrangian scheme. In his work, Mohammad M. Dorostkar [28] proposed a non-linear treatment of the velocity, which he considers a function of time, so as to improve the accuracy of the method by taking into account the curvature of the characteristic.

$$\begin{gathered}\vec{U}_{(t,\vec{x})} = \vec{U} + \vec{a}t + O(\Delta t^2) \\ \int\limits_{t^n}^{t^{n+1}} \vec{S} = \int\limits_{t^n}^{t^{n+1}} (\vec{U} + \vec{a}t) dt \Rightarrow \vec{S}^n = \vec{S}^{n+1} - \vec{U}\Delta t - \vec{a}\frac{\Delta t^2}{2}\end{gathered} \tag{2.28}$$

where:
$\vec{a}$: acceleration of the arrival point

For 2-dimensional problems the departure point's location is calculated as:

$$\begin{aligned} x_d &= x_a - u\Delta t - a_x \frac{\Delta t^2}{2} \\ y_d &= y_a - v\Delta t - a_y \frac{\Delta t^2}{2} \end{aligned} \tag{2.29}$$

with acceleration:

$$\begin{aligned} a_x &= \frac{u^n - u^{n-1}}{\Delta t} + O(\Delta t^2) \\ a_y &= \frac{v^n - v^{n-1}}{\Delta t} + O(\Delta t^2) \end{aligned} \tag{2.30}$$

While the proper way of calculating acceleration is between consecutive timesteps, because in FFD, the equations are split and solved separately, u^n and v^n essentially represent the intermediate velocity values from the diffusion equation, denoted as $u^\star$ and $v^\star$.

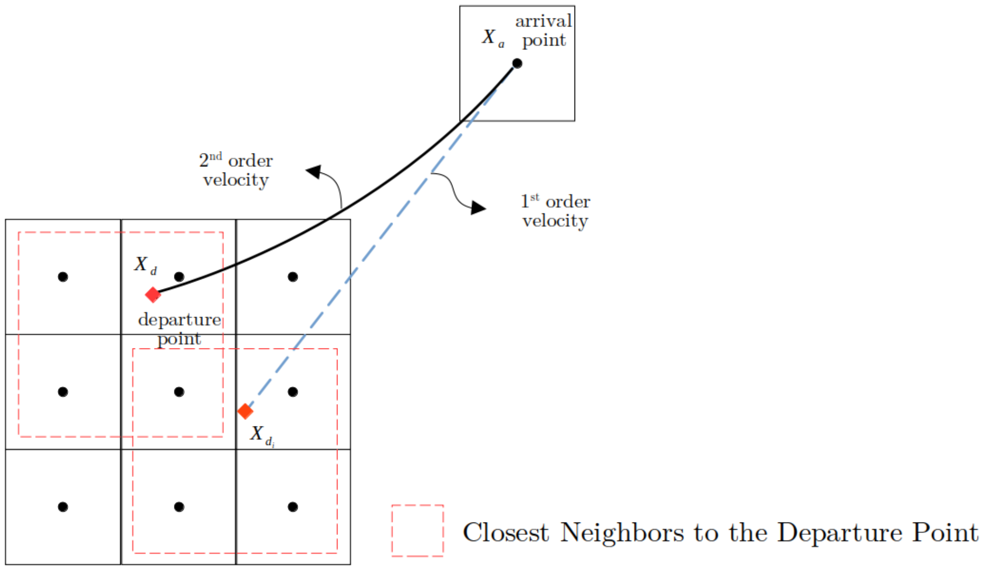


Figure 2.9: Characteristic Curve for the 1st and 2nd Order Velocity Models [28]

Wall Treatment

The trajectory equation is approximated with 2nd order accuracy, but due to truncation errors, there may be instances where departure points fall outside the computational domain. As a common practice, these Lagrangian points are re-positioned onto the boundary of the domain. However, when a particle's departure point is already situated on the solid boundary, it cannot migrate to an arrival point located away from that boundary. Consequently, this situation can lead to an unusually high concentration of departure points along the boundary [32]. As an alternative method, M. Jin et al. [33] implemented a special treatment for the near wall region. This approach assumed that the velocity component normal to the wall exhibited a linear variation between the solid boundary and the first adjacent grid point. This linear variation causes the velocity to gradually decrease to zero as it approaches the wall's surface during the trajectory tracing, as shown in Figure 2.10. For the trace-back from the arrival point to the first-grid point, the 1st order accuracy scheme is used, where the velocity is constant (2.27).

The velocity's component normal to the solid boundary is:

$$w = \begin{cases} \dfrac{z - z_0}{z_1 - z_0} w_1, & t < t^\star \\ w_a, & t^\star \leq t \leq t^{n+1} \end{cases} \tag{2.31}$$

By integrating equation (2.31) the relocated departure point is calculated as:

$$z_d = z_0 + (z_1 - z_0) \exp\left(-\frac{w_1}{z_1 - z_0}\left(\Delta t - \frac{z_a - z_1}{w_a}\right)\right) \tag{2.32}$$

where:
z: coordinate in the direction normal to the solid boundary
z_0: z-coordinate of the solid boundary
z_1: z-coordinate of the first-grid node
z_1: z-coordinate of the departure point
w_1: velocity at first-grid node
w_a: velocity at arrival point

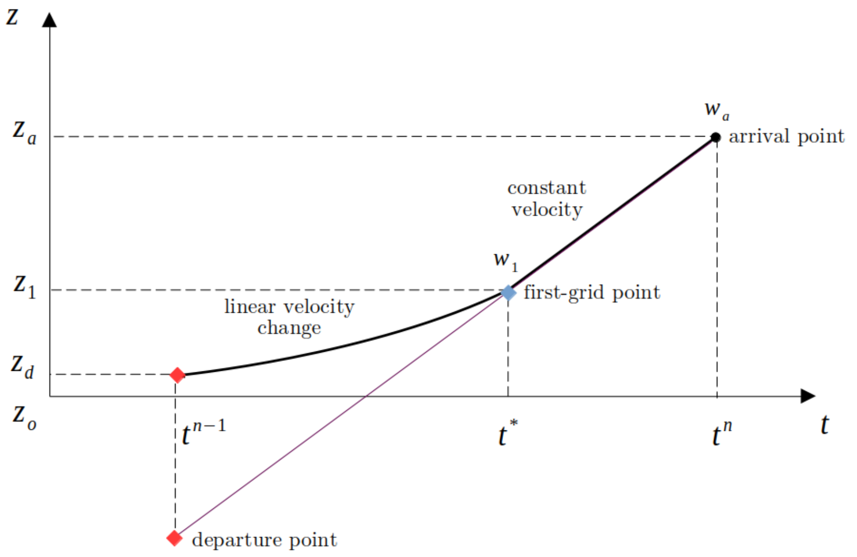


Figure 2.10: Treatment of Departure Points Located Out of Bounds [34]

In 2-dimensional problems within the (x, y) plane, this approach is employed when one of the coordinates of the departure point extends beyond the domain boundaries, while the other coordinate follows the conventional trajectory equation. For cases, where both coordinates of the departure points are simultaneously out of bounds, the point is translated on top of the boundary. Additionally, the velocity w_1 at the first adjacent grid node is interpolated with second-order accuracy from the surrounding nodes.

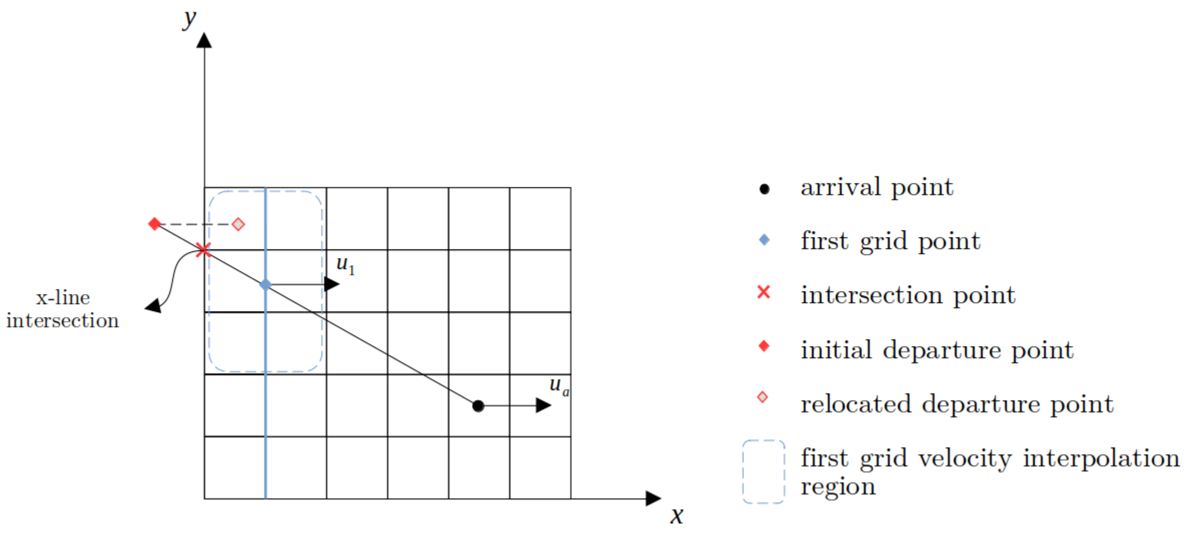


Figure 2.11: Wall Treatment in 2D for X-Line Intersection

2.3.2.2 Interpolation Scheme

The method of characteristics dictates that the scalar quantity ϕ is equal at both the arrival and departure points over a time step Δt. Therefore, after computing the departure point coordinates, ϕ_d requires an interpolation scheme. The proposed method employs an 4^{th} order semi-Lagrangian scheme with a computing cost of a 3^{rd} order interpolation. This is achieved by applying backward and forward 3^{rd} order interpolations in an alternating sweep manner, as described in [35].

In 1D semi-Lagrangian applications on non-uniform grids, the interpolated value of ϕ_d is determined using the Taylor series for the neighboring cell about x_d. The generalized formula is the following polynomial [36]:

$$\phi_d \approx \sum_{i=1}^{nb} \left[\left(\prod_{\substack{1 \le j \le nb \\ j \ne i}}^{nb} \frac{(x_d - x_j)}{(x_i - x_j)} \right) \phi_j \right] \tag{2.33}$$

For a uniform grid, the 3^{rd} order schemes are:

- Forward Interpolation

$$\phi_d = \frac{s_2 s_3}{2\Delta x^2}\phi_i + \frac{s_1 s_3}{\Delta x^2}\phi_{i+1} - \frac{s_1 s_2}{2\Delta x^2}\phi_{i+2} + E_f \tag{2.34}$$

where:

$$\begin{aligned} s_1 &= x_d - x_i \\ s_2 &= x_{i+1} - x_d \\ s_3 &= x_{i+2} - x_d \end{aligned}$$

The truncation error is:

$$E_f = \frac{s_1 s_2 s_3}{6}\frac{\partial^3 \phi_d}{\partial x^3} + O(\Delta x^4) \tag{2.35}$$

- Backwards Interpolation

$$\phi_d = -\frac{s_1 s_3}{2\Delta x^2}\phi_{i-1} + \frac{s_2 s_3}{\Delta x^2}\phi_i + \frac{s_1 s_2}{2\Delta x^2}\phi_{i+1} + E_b \tag{2.36}$$

where:

$$\begin{aligned} s_1 &= x_d - x_i \\ s_2 &= x_d - x_{i-1} \\ s_3 &= x_3 - x_{i+1} \end{aligned}$$

The truncation error is:

$$E_b = -\frac{s_1 s_2 s_3}{6}\frac{\partial^3 \phi_d}{\partial x^3} + O(\Delta x^4) \tag{2.37}$$

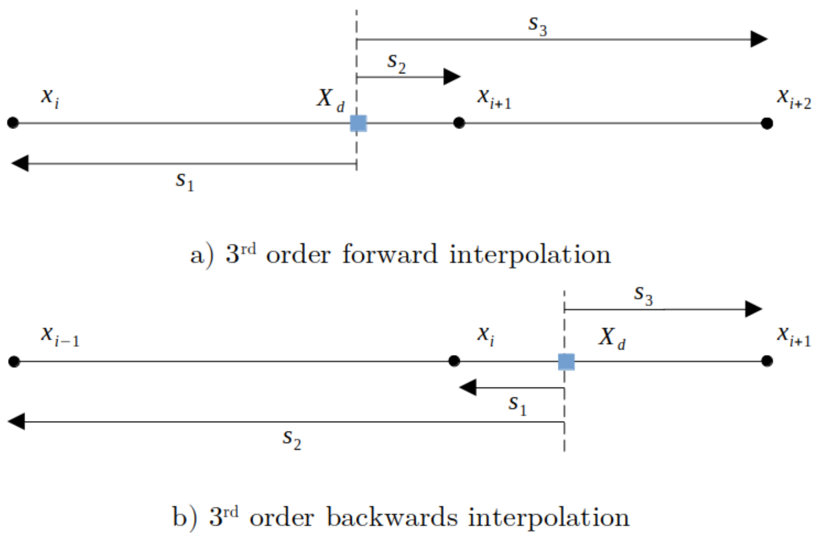


Figure 2.12: Schematic of the 1D Semi-Lagrangian Method for a) Forwards Interpolation and b) Backwards Interpolation [28]

The truncation errors are of 3^{rd} order, but by alternating between forward and backward interpolation, the leading errors are proved to cancel out at the end of $2\Delta t$ (which completes one sweep), and the final error is of 4^{th} order [35].

When applying the scheme on 2D grids, the 1D formulas (2.34) and (2.36) are used, in the manner demonstrated in Figure 2.13. Initially, three interpolations in the x-direction are performed to calculate the intermediate scalar values $\phi^\star$, which serve as reference points for the subsequent interpolation in the y-direction. The specific interpolation region varies depending on the timestep of the simulation, alternating between the blue and red regions in the figure for even and odd timesteps, respectively.

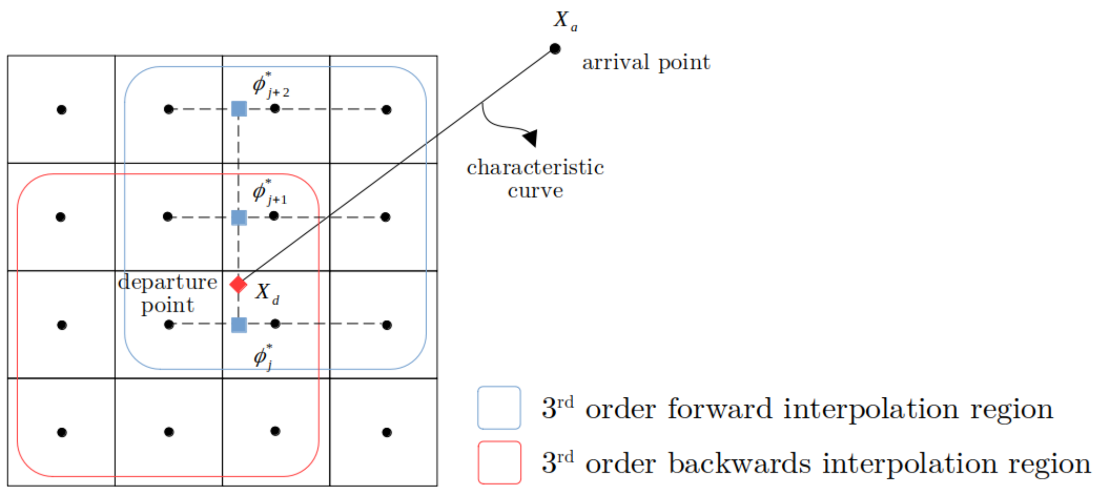


Figure 2.13: Schematic of the 2D Semi-Lagrangian Method [28]

2.3.3 Pressure Correction

Chorin's projection method is a numerical technique used to decouple the velocity and pressure equations in simulations of incompressible fluids. This is necessary because the Navier-Stokes equations lack an independent equation for pressure; instead, pressure acts as a constraint to ensure mass conservation of the velocity field. Chorin's method involves solving the Poisson equation for pressure and subsequently updating the velocity field based on the pressure gradient. The resulting field is divergence-free, which satisfies the continuity equation for incompressible fluids.

2.3.3.1 Poisson Equation

Poisson equation for pressure is derived by applying the divergence operator to the incompressible Navier-Stokes equations.

Assuming a velocity field $\vec{v}$ that satisfies the momentum and mass equations at a time moment $(n+1)$. The time-discretized equations are as follows:

$$\nabla \cdot \vec{v}^{n+1} = 0 \tag{2.38}$$

$$\frac{\vec{v}^{n+1} - \vec{v}^{n}}{\Delta t} = -\frac{1}{\rho}\nabla P^{n+1} - (\vec{v}^{n} \cdot \nabla)\vec{v}^{n} + \nu \nabla^2 \vec{v}^{n} \tag{2.39}$$

Note that the velocity is explicitly solved, simplifying the left side and resulting equations. This approach is chosen to keep the equations more straightforward and manageable.

To facilitate decoupling, an intermediate velocity field $\vec{v}^{\star}$ is introduced, which "splits" the time derivative

$$\boxed{\frac{\vec{v}^{n+1} - \vec{v}^{\star}}{\Delta t}} + \boxed{\frac{\vec{v}^{\star} - \vec{v}^{n}}{\Delta t}} = \boxed{-\frac{1}{\rho}\nabla P^{n+1}} - \boxed{(\vec{v}^{n} \cdot \nabla)\vec{v}^{n} + \nu \nabla^2 \vec{v}^{n}} \tag{2.40}$$

This leads to two sets of equations:

$$\frac{\vec{v}^{\star} - \vec{v}^{n}}{\Delta t} = -(\vec{v}^{n} \cdot \nabla)\vec{v}^{n} + \nu \nabla^2 \vec{v}^{n} \tag{2.41}$$

$$\frac{\vec{v}^{n+1} - \vec{v}^{\star}}{\Delta t} = -\frac{1}{\rho}\nabla P^{n+1} \tag{2.42}$$

Equation (2.41) calculates the velocity field $\vec{v}^{\star}$, that does not inherently satisfy the mass conversation as the pressure field is not taken into consideration. As a result it cannot be equal to velocity vector $\vec{v}^{n+1}$.

The divergence operator is applied to equation (2.42), resulting in the Poisson equation for pressure. This equation calculates the pressure distribution necessary to ensure the incompressibility of the fluid in the simulation.

$$\cancelto{\textit{incompressible}}{\nabla \cdot \vec{v}^{n+1}} - \nabla \cdot \vec{v}^{\star} = -\frac{\Delta t}{\rho}\nabla^2 P^{n+1} \Rightarrow \nabla^2 P^{n+1} = \frac{\Delta t}{\rho}\nabla \cdot \vec{v}^{\star} \tag{2.43}$$

Similarly, to the diffusion term, the Poisson equation is descritized with a central 2nd order scheme on the pressure grid of the staggered meshes.

$$\alpha_1 P^{\star}_{i+1,j} + \alpha_2 P^{\star}_{i,j+1} + \beta_0 P^{\star}_{i,j} + \gamma_1 P^{\star}_{i-1,j} + \gamma_2 P^{\star}_{i,j-1} = \frac{\rho}{\Delta t}\frac{\partial \vec{U}^{\star\star}_{i}}{\partial x_i} \tag{2.44}$$

where:

$$\alpha_1 = 2\left[\frac{1}{\Delta x_{i+1,j}(\Delta x_{i+1,j} + \Delta x_{i,j})}\right] \qquad \gamma_1 = 2\left[\frac{1}{\Delta x_{i,j}(\Delta x_{i+1,j} + \Delta x_{i,j})}\right]$$

$$\alpha_2 = 2\left[\frac{1}{\Delta y_{i,j+1}(\Delta y_{i,j+1} + \Delta y_{i,j})}\right] \qquad \gamma_2 = 2\left[\frac{1}{\Delta y_{i,j}(\Delta y_{i,j+1} + \Delta y_{i,j})}\right]$$

$$\beta_0 = -2\left[\frac{1}{\Delta x_{i+1,j}\Delta x_{i,j}} + \frac{1}{\Delta y_{i,j+1}\Delta y_{i,j}}\right]$$

The matrix form is then formulated and and addressed using the Gauss-Seidel iterative method. It is important to note that for both the diffusion and pressure correction steps, no relaxation or scheme was employed in the Gauss-Seidel method. Convergence criteria were defined by either achieving a relative error within the range of $10^3 - 10^4$ or ensuring that the number of iterations remained below a specified threshold.

The spatial derivative is expressed in the Einstein notation, where the repeated index indicates the summation of the corresponding terms.

$$\frac{\partial \vec{U}_i^{\star\star}}{\partial x_i} = \frac{\partial u^{\star\star}}{\partial x} + \frac{\partial v^{\star\star}}{\partial y} \tag{2.45}$$

where:
$U_i^{\star\star}$: the second intermediate field after the convection step is solved

The velocity gradient is descretized with a 1^{st} order forward scheme:

$$\frac{\partial u^{\star\star}}{\partial x} = \frac{u^{\star\star}_{i+1,j} - u^{\star\star}_{i,j}}{\Delta x_{i,j}} \tag{2.46}$$

$$\frac{\partial v^{\star\star}}{\partial y} = \frac{v^{\star\star}_{i+1,j} - v^{\star\star}_{i,j}}{\Delta y_{i,j}} \tag{2.47}$$

2.3.3.2 Pressure Gradient Term

The corrected pressure field derived from Poisson equation is later used to update the velocity field, by simply marching through time, without the need for iterations or complex algorithms.

$$\vec{U}^{n+1} = \vec{U}^{\star\star} - \frac{\Delta t}{\rho}\nabla P^{\star} \tag{2.48}$$

$$u^{n+1} = u^{\star\star} - \frac{\Delta t}{\rho}\left(\frac{\partial P^{\star}}{\partial x}\right) \tag{2.49}$$

$$v^{n+1} = v^{\star\star} - \frac{\Delta t}{\rho}\left(\frac{\partial P^{\star}}{\partial y}\right) \tag{2.50}$$

The gradient term of pressure is discretized with a 1^{st} order forward scheme:

$$\frac{\partial P^\star}{\partial x} = \frac{P^\star_{i+1,j} - P^\star_{i,j}}{\Delta x_{i,j}} \tag{2.51}$$

$$\frac{\partial P^\star}{\partial y} = \frac{P^\star_{i,j+1} - P^\star_{i,j}}{\Delta y_{i,j}} \tag{2.52}$$

The first iteration of FFD is thus completed and the algorithm commences anew with the velocity and pressure fields of the previous timestep as initial conditions in order to speed up the convergence.

2.4 Turbulence Models

Turbulence in fluid flows refers to the chaotic and irregular motion of fluid elements. When a fluid, such as air or water, flows over a surface, it can exhibit different types of flow patterns. In laminar flow, the fluid moves smoothly in parallel layers with minimal disruption. However, as the speed of the fluid or the complexity of the flow increases, turbulence may occur.

Turbulent flow is characterized by random and irregular fluctuations in velocity, pressure, and other flow properties. These fluctuations create swirling eddies and vortices within the fluid, leading to a more complex and less predictable flow pattern compared to laminar flow. Turbulence is often accompanied by increased mixing, heat transfer, and energy dissipation.

Until now, the flow has been characterized as laminar, with no consideration given to turbulent behavior. However, for the purpose of this thesis, which focuses on addressing the application of atmospheric flows and urban micro-climates, two distinct turbulent models will be utilized. These models have been documented in relevant literature, particularly in the context of indoor simulations.

Turbulence is introduced into the diffusion term of the Navier-Stokes equations as an additional viscosity.

$$\frac{\partial \vec{U}}{\partial t} + (\vec{U} \cdot \nabla)\vec{U} = -\frac{1}{\rho}\nabla P + (\nu + \nu_t)\nabla^2 \vec{U} + \vec{f} \tag{2.53}$$

2.4.1 Constant Turbulence Model

In 2019 Tan et al. developed an FFD solver for Data-Center Floor Plenums. To mitigate the computational cost and simplify implementation, they opted for a practical approach, regarding turbulence, by introducing a fixed turbulent viscosity set at 100 times the molecular viscosity, as documented in [11].

$$\nu_t = 100\ \nu \tag{2.54}$$

Given that accuracy was not the primary objective of the investigation, this model was selected for its simplicity and reported a relative error of 3.6% compared with the standard CFD with $k - \epsilon$ turbulence model.

2.4.2 Zero-Equation Model

Chen et al. [37] introduced a zero-equation turbulence model, designed to eliminate the need for solving an additional differential equation, thereby offering a computationally economical alternative. It originates from Prandtl's mixing length hypothesis, deveoped in the early 20^{th} century. The key idea behind Prandtl'a mixing length hypothesis is to relate the turbulent eddy viscosity to the characteristic length scale of the turbulent eddies within the flow.

$$u \sim l_{mox} \left| \frac{\partial U}{\partial y} \right| \tag{2.55}$$

$$\mu_t = \rho l_{mix}^2 \left| \frac{\partial U}{\partial y} \right| \tag{2.56}$$

The concept assumes that the turbulent mixing can be represented by a length scale, known as the mixing length and is used to estimate the eddy viscosity. While the hypothesis has limitations and is not universally accurate for all turbulent flows, it provides a practical approach in modeling free shear flows. After some modifications to make the hypothesis mode applicable, the following model was developed:

$$\nu_t = 0.03874 \; L \; U \tag{2.57}$$

where:

L: a length scale, which is equal to the shortest distance inside the domain from the wall

U: the local mean velocity

The model has shown promising results in simulating indoor scenarios involving natural, forced, and mixed convection.

3. Data Assimilation Method - Nudging

Data Assimilation has been introduced in Chapter 1 as the merging of experimental data with numerical models, in order to reduce computational costs without compromising the accuracy of the results. The primary emphasis of this thesis will be on nudging, a particularly inexpensive Data Assimilation method.

3.1 General Principles

Nudging, also known as Newtonian relaxation, is a dynamic method employed in various fields, such as meteorology and computational modeling, to adjust a model's state toward observed data [14]. This technique involves introducing a feedback term into the model equation that is proportional to the misfit between the model and the observations. The feedback term introduced in nudging serves as a control mechanism for the evolution of a dynamical system. This concept shares similarities with PID control, where feedback terms are adjusted based on the error between the desired and actual states.

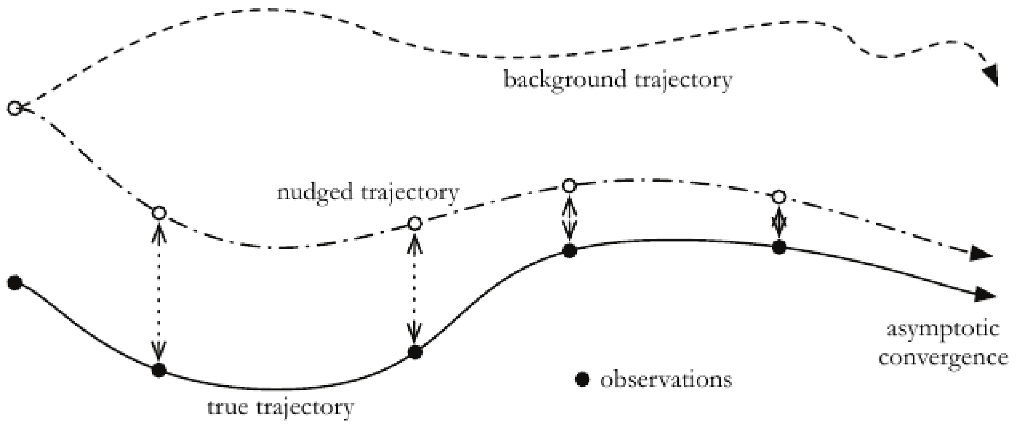


Figure 3.1: Schematic of Nudging Method[14]

The following system of ordinary differential equations govern a dynamic system with state variable $\mathbf{x}$

$$\frac{d\mathbf{x}}{dt} = M(\mathbf{x}(t)), \ \mathbf{x}(t=0) = \mathbf{x}_0 \tag{3.1}$$

If $\mathbf{y}^o(t)$ are a set of observations of the system's state at locations $\mathbf{x}_k$, distributed over time, the observation error is calculated as:

$$\epsilon^o(t) = \mathbf{y}^o(t) - H(\mathbf{x}(t)) \tag{3.2}$$

where:
H : the observation operator or measurement matrix

Operator H transfers a state $\mathbf{x}$ to the same time and location as the observations $\mathbf{y}^o$, so that $\mathbf{y}^o(t)$ and $H(\mathbf{x}(t))$ are comparable.

The nudged system of equations incorporates a gain matrix $\mathbf{K}$ that when small, $\mathbf{x}$ simply solves the model equations. Parameter $\mathbf{K}$ can be either a matrix or a scalar and is case dependent, meaning it cannot be selected automatically but relies on numerical experimentation.The strength of the nudging term determines how much influence the observations have on the model's evolution.

The nudged system is:

$$\frac{d\mathbf{x}}{dt} = M(\mathbf{x}(t)) \underbrace{\boxed{+\mathbf{K}(\mathbf{y}^o(t) - H(\mathbf{x}(t)))}}_{\text{feedback-like term}}, \ \mathbf{x}(t=0) = \mathbf{x}_0 \tag{3.3}$$

For small observation errors that remain constant over time and are independent of spatial variations, nudging proves to be a reliable and effective approach. However, when dealing with substantial observational errors, the use of an ill-defined gain matrix $\mathbf{K}$ can lead to suboptimal results, necessitating a more sophisticated selection approach. Examples of such advanced methods include the incorporation of a Kalman filter gain matrix or the implementation of an optimal nudging scheme [14]. While these methods offer improved accuracy, it's important to note that they also come with an increased computational cost.

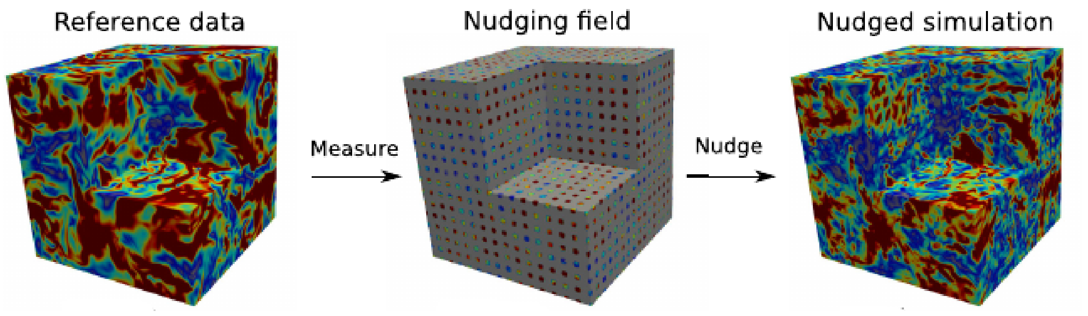


Figure 3.2: Visualization of the Nudging Process [38]

3.2 Application to Navier-Stokes Equations

The nudging term is added to the momentum Navier-Stokes equations as a forcing term.

$$\frac{\partial \vec{U}_a}{\partial t} + (\vec{U}_a \cdot \nabla)\vec{U}_a = -\frac{1}{\rho}\nabla P + \nu \nabla^2 \vec{U}_a + \vec{f} + \mathbf{K}(\vec{U}_{obs} - H(\vec{U}_a)) \tag{3.4}$$

where:
$\vec{U}_a$: the estimated velocity field
$\vec{U}_{obs}$: the velocity measurements

For simplicity the gain matrix $\mathbf{K}$ is characterized by a scalar parameter α to adjust the weight of the nudging and the adjoint of the observation operator, H^T, to project the observation error at the observation locations. The value of parameter α has been researched extensively and Di Leoni et al. [38] reported that α should scale as the inverse of the time step of the simulations.

$$\alpha \approx \frac{1}{\Delta t} \tag{3.5}$$

The nudged Navier-Stokes equations are:

$$\frac{\partial \vec{U}_a}{\partial t} + (\vec{U}_a \cdot \nabla)\vec{U}_a = -\frac{1}{\rho}\nabla P + \nu \nabla^2 \vec{U}_a + \vec{f} + \alpha H^T(\vec{U}_{obs} - H(\vec{U}_a)) \tag{3.6}$$

3.2.1 FFD Implementation

The FFD algorithm, extensively detailed in Chapter 2, employs a time-splitting method where each component of the Navier-Stokes equations is addressed independently. The nudging term can be incorporated into either the diffusion equation or the pressure correction Poisson equation.

$$\frac{\partial \vec{U}_a}{\partial t} = \nu \nabla^2 \vec{U}_a + \vec{f} + \alpha H^T(\vec{U}_{obs} - H(\vec{U}_a))$$
$$\nabla^2 P = \frac{\rho}{\Delta t}\nabla \cdot \vec{U}_a + \alpha H^T(\vec{U}_{obs} - H(\vec{U}_a))$$

Given that the experimental data represent velocity measurements, it is preferable to introduce the nudging feedback into the diffusion equation. This choice is motivated by the fact that the external forces are already accounted for in the diffusion equation, and solving it implicitly in relation to velocity can enhance the stability of the algorithm.

3.2.2 Discretization

The finite differences framework for the nudging process does not require the nudging points to coincide with the grid nodes of the mesh. However, for the sake of simplicity in the coding process, observations derived from experimental values in the literature are translated to the nearest mesh nodes. Moreover, rather than projecting the observation error to the simulation grid, the observation vector is populated with zeros, and the whole nudging process takes place exclusively on the mesh locations. The nudged equation is thus simplified in the following manner:

$$\frac{\partial \vec{U}_a}{\partial t} = \nu \nabla^2 \vec{U}_a + \vec{f} + \alpha(\vec{U}_{obs} - \vec{U}_a) \tag{3.7}$$

where $\vec{U}_{obs}, \vec{U}_a$ are of size n number of mesh points.

Additionally, given that the applications that are being examined in the present thesis are steady state problems, the observations are not time dependent but correspond to the steady state solution.

The discretization is identical with the method presented in Chapter 2, where an implicit scheme is implemented. The observation vector is considered at the previous timestep and modifies the constant vector $\vec{B}$ of the system of equations.

$$\underset{n\times n}{A} \cdot \underset{n\times 1}{\vec{U}^{\star}} = \underset{n\times 1}{B} \tag{3.8}$$

where:
n: the number of nodes in the mesh

For a uniform mesh, the diffusion equation becomes:

$$\frac{\vec{U}^{\star}_{a,i,j} - \vec{U}^{n}_{a,i,j}}{\Delta t} = \nu \left[\frac{\vec{U}^{\star}_{a,i+1,j} - 2\vec{U}^{\star}_{a,i,j} + \vec{U}^{\star}_{a,i-1,j}}{\Delta x^2} + \frac{\vec{U}^{\star}_{a,i,j+1} - 2\vec{U}^{\star}_{a,i,j} + \vec{U}^{\star}_{a,i,j-1}}{\Delta y^2} \right] + \vec{f} + \alpha(\vec{U}_{obs,i,j} - \vec{U}^{n}_{a,i,j}) \tag{3.9}$$

4. Benchmark Test: Lid-Driven Cavity

4.1 Problem Description

The lid-driven cavity is a classic benchmark test that is widely used for validating and comparing different numerical methods and solvers, as it has been extensively researched and documented for both laminar and turbulent cases and there are available analytical solutions for certain Reynolds numbers. It involves a simple square cavity domain with three stationary walls and a top lid that is moving at a constant velocity u_0. Despite its simplicity, the lid-driven cavity problem exhibits interesting flow physics, including the formation of vortices and flow patterns, making it a valuable tool for studying the behavior of fluid flow in confined domains.

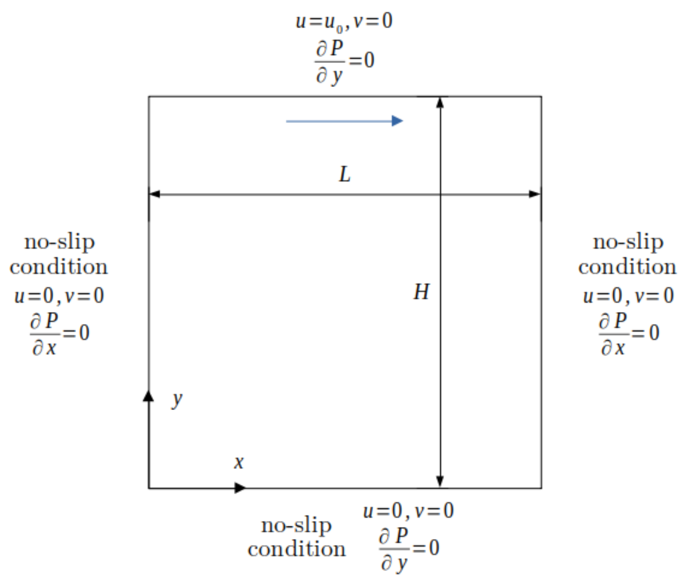

$u=u_0, v=0$
$\frac{\partial P}{\partial y}=0$
L
no-slip condition
$u=0, v=0$
$\frac{\partial P}{\partial x}=0$
H
no-slip condition
$u=0, v=0$
$\frac{\partial P}{\partial x}=0$
y
x
no-slip condition
$u=0, v=0$
$\frac{\partial P}{\partial y}=0$

Figure 4.1: Boundary Conditions for the 2D Lid-Driven Cavity Test

For simulating the cavity lid, it is assumed that there is not a gravitational field or any other external forces acting on the system, in accordance with the simulations conducted by Ghia et al [39]. The flow is considered laminar with a constant viscosity ν. The Reynolds number is thus calculated as:

$$Re = \frac{u_0 H}{\nu} \tag{4.1}$$

It is common practice to either use dimensionless variables or normalize every parameter to unity, in order to standardize comparisons. As such the upper lid velocity is set to $u_0 = 1\, m/s$ and the domain's dimensions are $1 \times 1\, m^2$

Regarding boundary conditions (Figure 4.1), the simulation employs a no-slip Dirichlet boundary condition for velocity at the fixed walls. For the pressure field, Neumann boundary conditions are imposed along every edge of the cavity. In order to establish the pressure level in the flow, a single reference point of pressure, denoted as P_{ref}, is usually applied at the inner left corner of the cavity, which acts as a pressure constraint.

Otherwise, the Navier Stokes equations have an infinite number of solutions, since they only solve with respect to the pressure gradient.

4.2 Numerical Solution of Fast Fluid Dynamics Algorithm

This section focuses on evaluating the effectiveness and reliability of the C++ code developed for the FFD algorithm. The objective is to assess the algorithm's accuracy, stability, and efficiency in simulating fluid dynamics phenomena.

All the simulations in this thesis were performed on a PC system with AMD Ryzen 5 2500U CPU @ 2 GHz and 12 GB RAM. The C++ code was compiled using GCC 11.4.0 with standard C++17 compliance and Level-3 optimization settings.

4.2.1 Validation of CityFFD Back & Forth Sweep Interpolation Method

As previously discussed in Chapter 2, the back & forth sweep interpolation method used in the semi-Lagrangian scheme of the convection term was based on the research work of Mohammad M. Dorostkar [28] and the CityFFD model application. As an initial validation step, it is crucial to assess the C++ solver using conditions and parameters akin to those proposed in this work. While the solving techniques remain similar, it's worth noting that the algorithm for solving the system of linear equations utilizes the Gauss-Seidel method instead of the multigrid method. This choice does not necessarily impact the results and maintains compatibility and simplicity, even though it may not provide the same level of computational efficiency as the multigrid method.

Similarly, to the CityFFD 2D cavity-lid testing, the simulation's timestep is set to $\Delta t = 0.005\ sec$ and the selected Reynolds number is $Re = 1000$. The domain is divided into structured and uniform grids of sizes 64×64 and 128×128. The convergence criterion is set to 10^{-6}, which is the maximum value of the difference between the current and the previous timestep. This criterion is well-suited for this application, given the known tendency of the laminar variation in the cavity benchmark to reach a steady state.

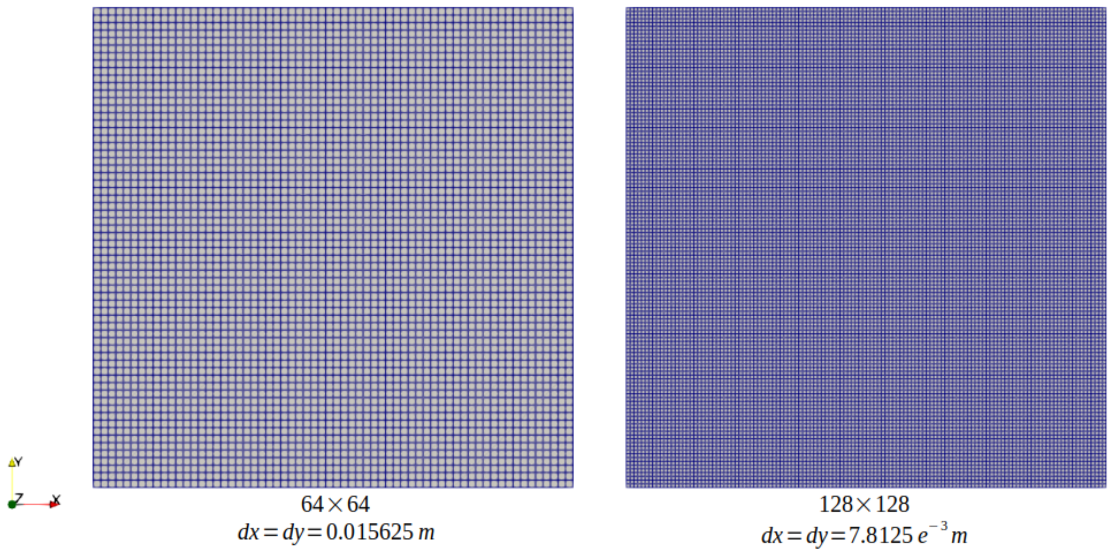


Figure 4.2: Uniform and Structured Grids of 64 × 64 and 128 × 128 for the 2D Lid-Driven Cavity

To facilitate comparisons, the velocity profiles along the lines $x = 0.5$ and $y = 0.5$ are presented for the horizontal and vertical components respectively, as shown in Figure 4.3. Ghia et al. [39] have provided a tabular version of the results for reference, which are based on a vorticity-stream function formulation coupled with an implicit multigrid method. Despite its age, the reliability of the Ghia et al. [39] lid-driven cavity benchmark persists due to its extensively validated results, consistent agreement with newer studies, well-defined geometry and conditions, numerical stability, and enduring educational value.

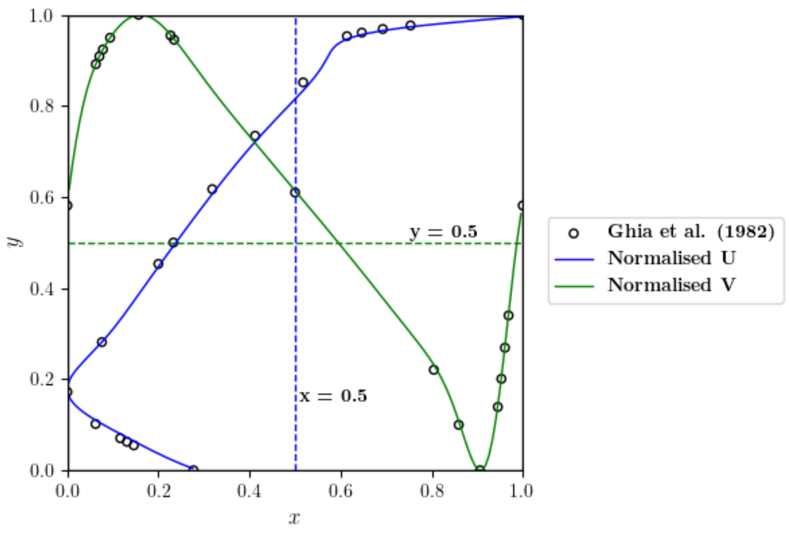


Figure 4.3: Ghia et al.'s [39] Tabular Velocity Values along Cavity's Midlines and Interpolated Profiles for Better Visualization

In CityFFD [28], the profile of the horizontal velocity component u is used as a validation criteria. Figure 4.4 shows the high accuracy of the proposed back & forth interpolation scheme, while the conventional linear method fails to converge satisfactorily to Ghia's [39] solution even on the finer mesh.

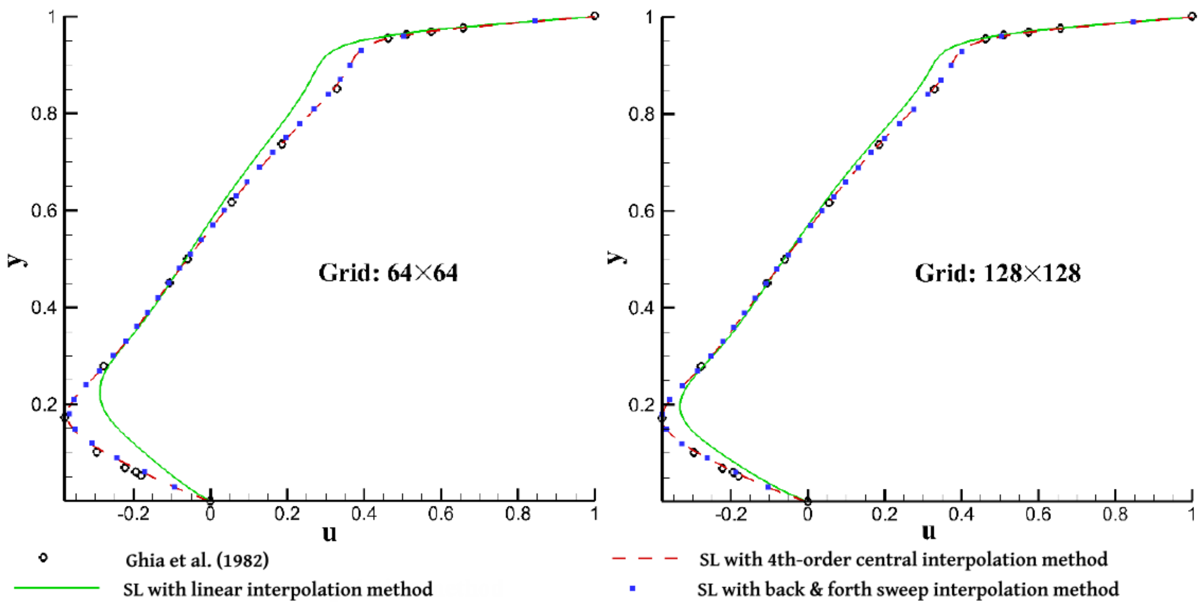


Figure 4.4: CityFFD [28] Lid-Driven Cavity Benchmark Results for Horizontal Velocity u for $Re = 1000$, $\Delta t = 0.005$ *sec*

The solver developed in the present thesis successfully replicates the results of CityFFD, as shown in the accompanying figures. Figure 4.5 mirrors the behavior observed in Figure 4.4 and verifies that the back & forth interpolation scheme applied in the convection term can achieve the same accuracy of the 4^{th} order central scheme. As expected, the finer mesh is able to capture the velocity profiles with greater precision, as it facilitates better calculations of gradients and reduces numerical errors.

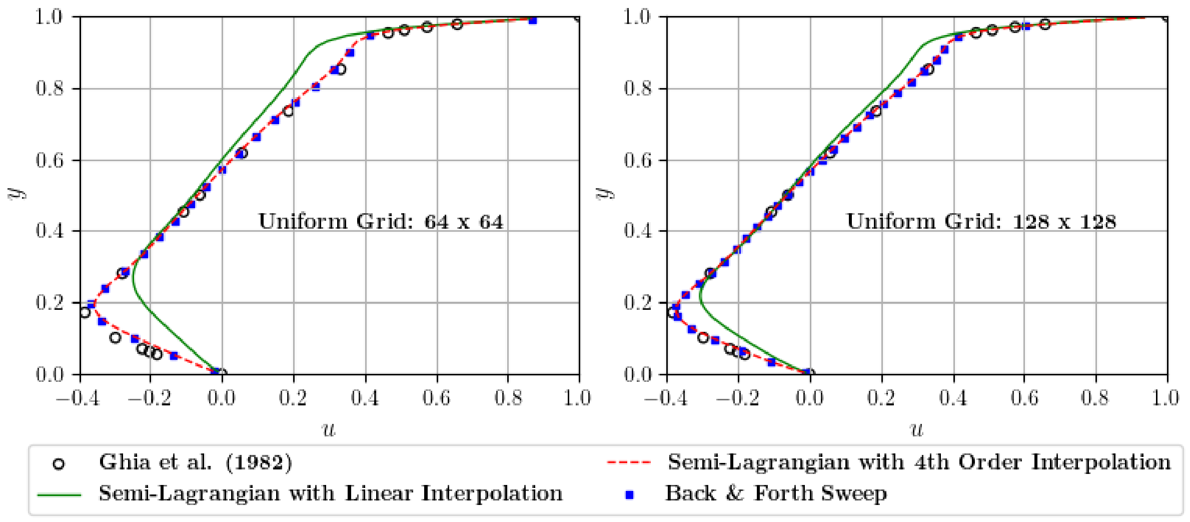


Figure 4.5: Present Thesis Lid-Driven Cavity Benchmark Results for Horizontal Velocity u for $Re = 1000$, $\Delta t = 0.005$ sec

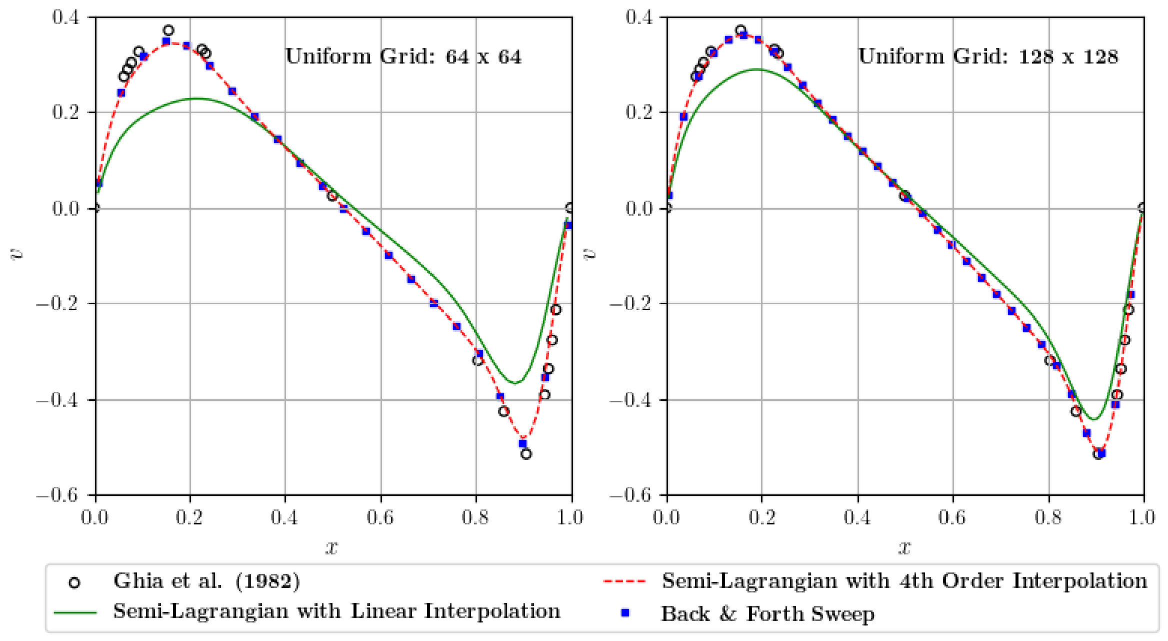


Figure 4.6: Present Thesis Lid-Driven Cavity Benchmark Results for Vetical Velocity v for $Re = 1000$, $\Delta t = 0.005$ sec

Upon examining each interpolation scheme individually and comparing them to M. Dorostkar's work [28] in Figures 4.7 and 4.8, certain discrepancies become apparent. To facilitate a statistical comparison, both the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) were calculated. It is crucial to acknowledge that the error calculations themselves may introduce uncertainties due to the misalignment of the observations ($\hat{y}_i$) and predicted values (y_i). In this analysis, an interpolation method was employed to align the data points and the observed values, which were extracted from the graphs using a plotting tool. These steps introduce an additional layer of potential error, and the results should be interpreted with caution, taking into consideration the interpolation process and the reliance on graphical data extraction.

$$MAE = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n} \tag{4.2}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{4.3}$$

where :
n: the number of observations
y_i: predicted values (thesis results)
$\hat{y}_i$: observed values (M. Dorostkar [28] results)

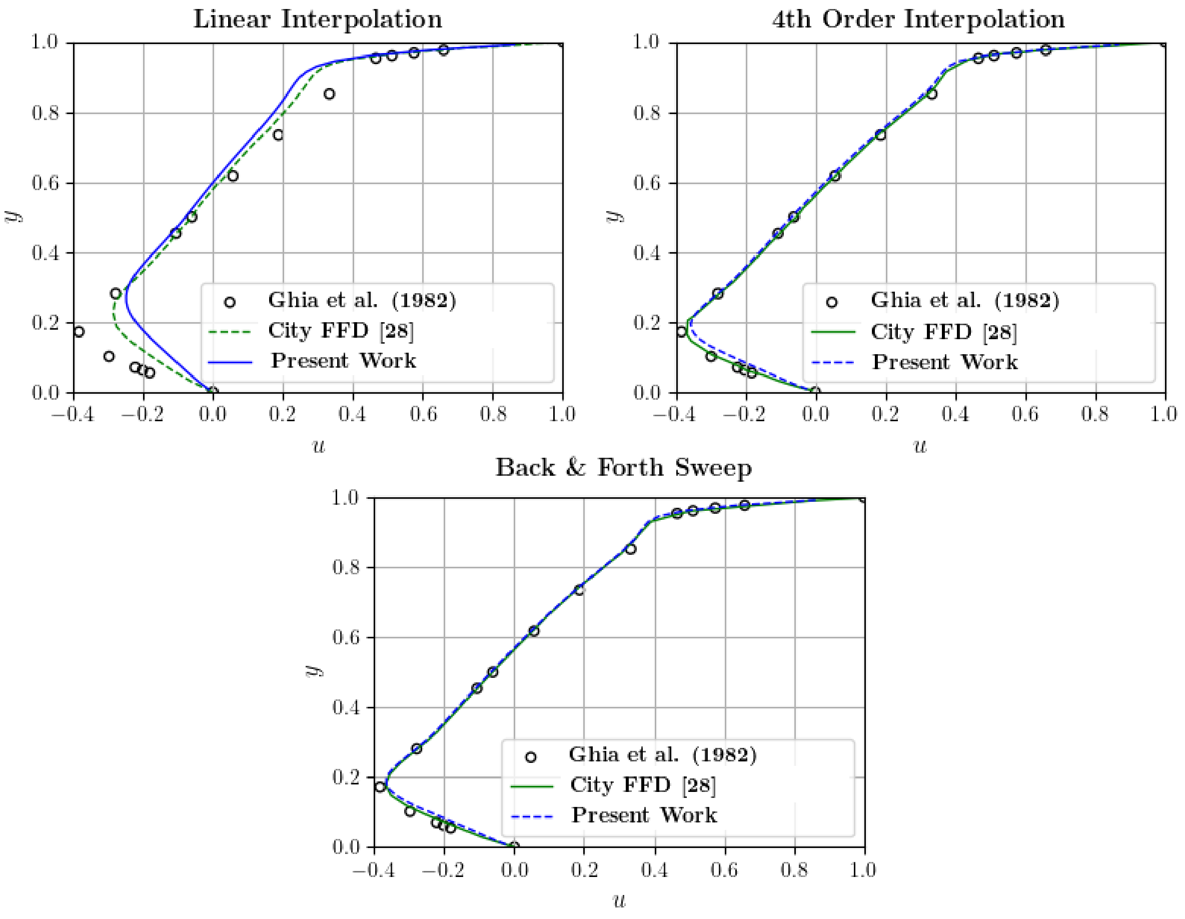


Figure 4.7: Comparison of CityFFD [28] and Thesis u Profiles for Different Interpolation Schemes for $Re = 1000$, $\Delta t = 0.005\ sec$, Grid: 64×64

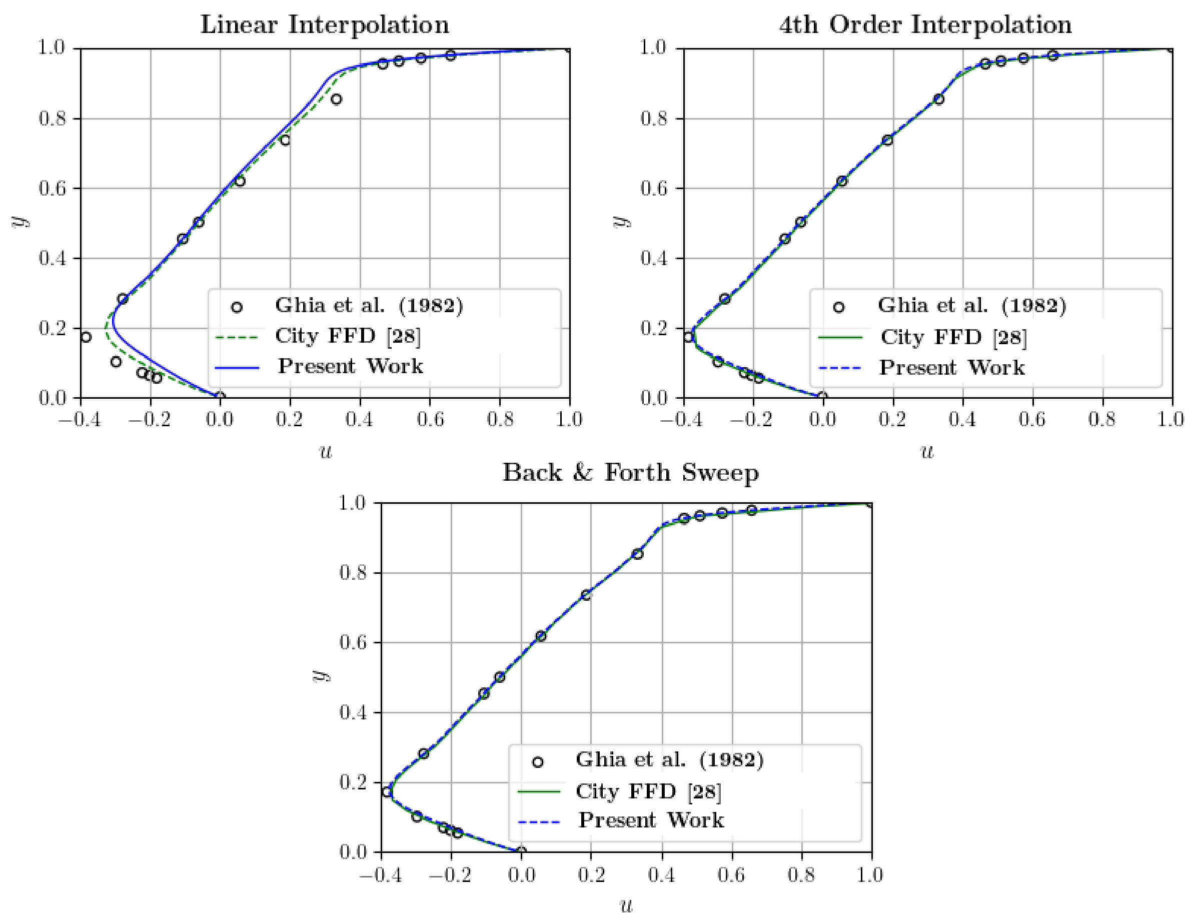


Figure 4.8: Comparison of CityFFD [28] and Thesis u Profiles for Different Interpolation Schemes for $Re = 1000$, $\Delta t = 0.005\ sec$, Grid: 128×128

Table 4.1: Comparative Error of CityFFD and Present Thesis Applications for Different Interpolation Schemes

Schemes	Grid	MAE [%]	RMSE
Linear	64×64	2.54184	0.029347
	128×128	1.96834	0.02252
4th Order Central	64×64	1.55463	0.01938
	128×128	1.38028	0.01717
Back & Forth Sweep	64×64	0.80235	0.01136
	128×128	0.72321	0.01059

The results indicate an absolute error of approximately $\sim 2\%$ for the linear cases and $\sim 0.8\%$ for the back-and-forth sweep interpolation method. The small magnitude of these errors suggests that they may be attributable to coding parameters. Notably, the linear case exhibits the most pronounced struggle, indicating a potential disparity in the coding implementation. Although the same interpolation scheme is employed, variations could arise from how departure points lacking sufficient neighboring cells for interpolation are handled. Moreover, given that this scheme inherently possesses higher numerical error, disparities between codes are likely to be magnified. Other factors contributing to these differences could include the treatment of ghostnodes, the convergence criteria employed by the linear solver, and potential relaxation factors. Additionally, the absence of information regarding the mesh used, such as whether it was collocated or staggered, complicates the expectation of identical results.

In the context of a lid-driven cavity, vortices typically emerge due to the shear forces induced by the motion of the lid against the stationary fluid. These vortices tend to manifest near the corners and along the edges of the cavity, where the interaction between the moving lid and the stationary walls with the fluid generates regions of high vorticity. The viscous corner eddies at the intersection of two solid boundaries are known as "Moffatt vortices", forming a sequence of vortices with rapidly decreasing size and intensity towards the corner [40]. However, due to computational limitations, accurately resolving tertiary and quaternary vortices, which occur at extremely small scales, remains challenging. Consequently, most simulations primarily focus on capturing the larger-scale secondary vortices. The specific patterns of vortices depend mainly on the Reynolds number. At lower Reynolds numbers, the flow tends to be more steady and laminar, whereas at higher Reynolds numbers, turbulence becomes more prominent, leading to complex vortex structures.

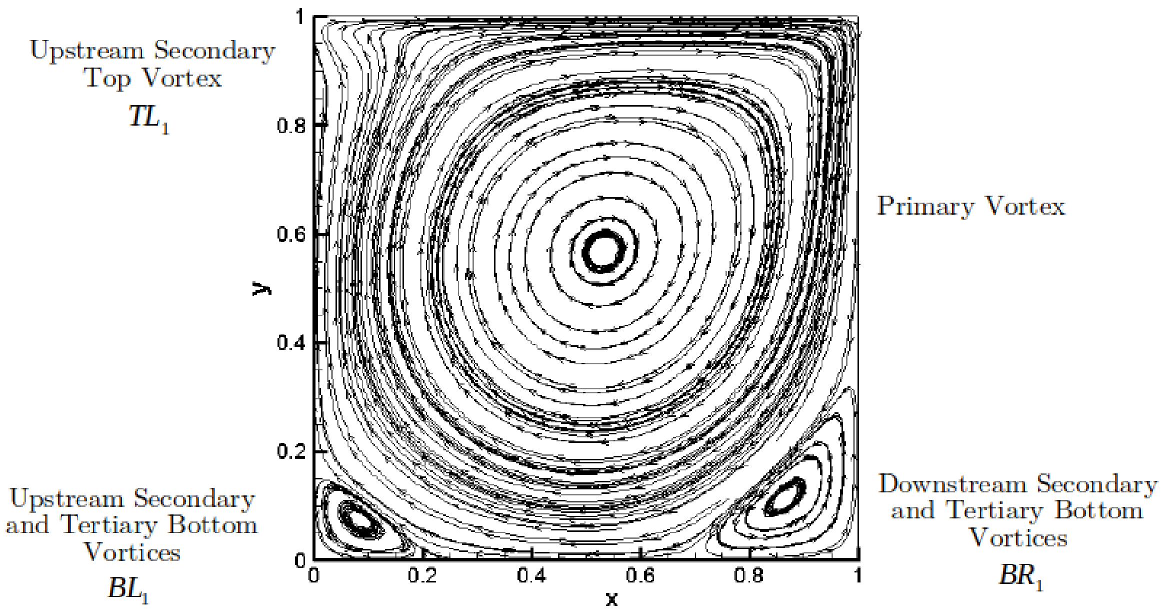


Figure 4.9: Velocity Streamlines for CityFFD [28] with Back & Forth Sweep Interpolation Method for $Re = 1000$, $\Delta t = 0.005\ sec$

In Figure 4.10, the velocity streamlines of each interpolation method for grids 64×64 and 128×128 are presented. Regardless of the grid resolution, the solver adeptly captures the secondary small-scale vortices at the lower corners of the domain for both linear and higher-order interpolation schemes. Moreover, the streamlines at the top left corner of the cavity, where the upstream secondary top vortex is located, are curved more pronouncedly for the higher-order schemes.

Interestingly, the impact of grid resolution appears to be less significant compared to the influence of the chosen interpolation scheme, except notably for the linear scheme. This suggests that for high-order schemes, finer meshes do not lead to substantial changes in capturing the underlying physics of the flow. However, for the linear interpolation scheme, on the fine mesh of 128×128, the primary central vortex is positioned lower, closer to the geometric center of the cavity, resembling the behavior seen in the higher-order schemes. Nonetheless, there are notable alterations in velocity magnitude. Conversely, on a 64×64 grid, the primary vortex appears tilted towards the upper right corner.

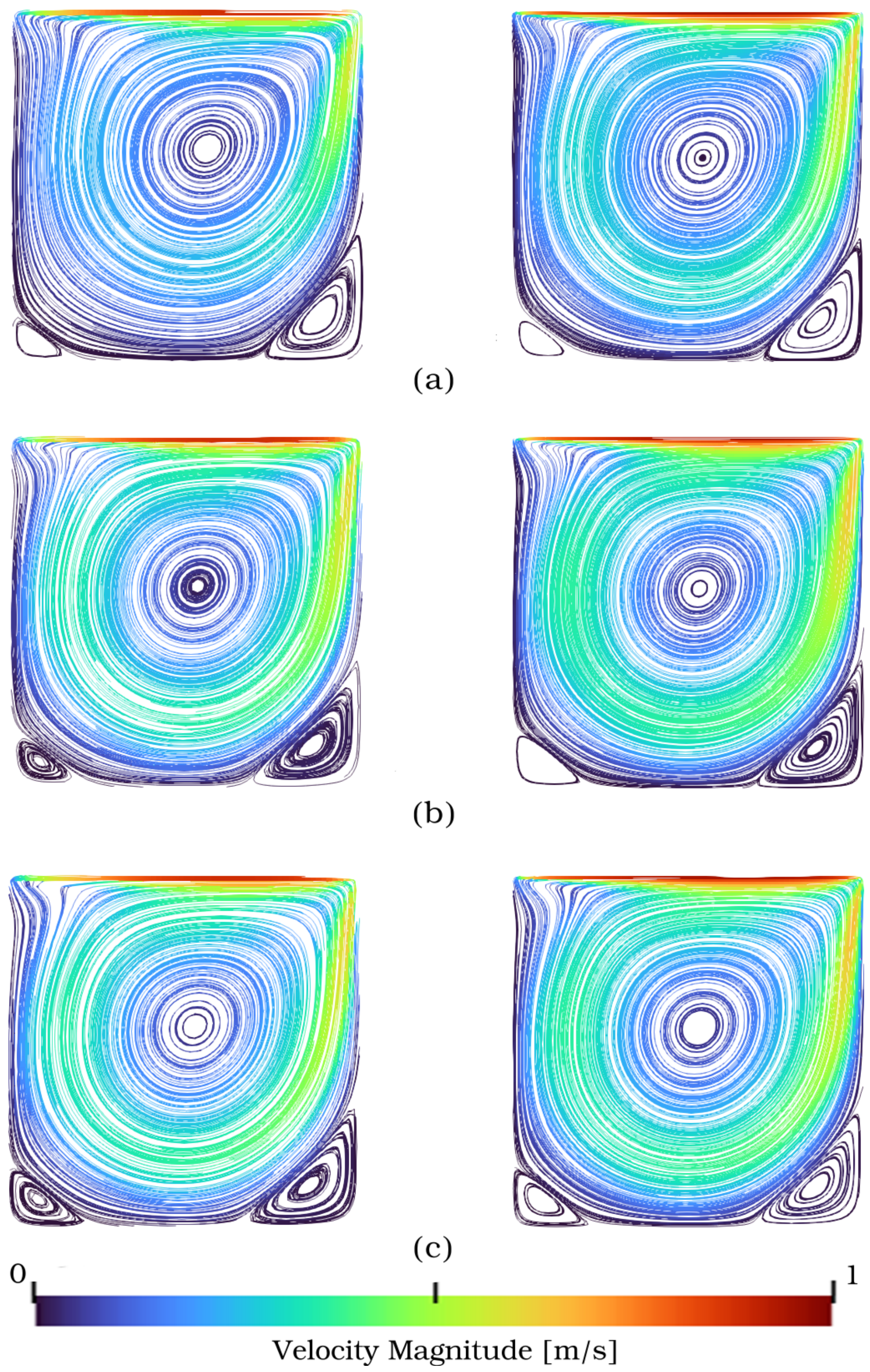


Figure 4.10: Velocity Streamlines for Grids 64×64 (left) and 128×128 (right) for (a) Linear Scheme ; (b) 4^{th} Order Central Scheme ; (c) Back & Forth Sweep

Finally, the solver's performance was found to be comparable to the results obtained using CityFFD. Specifically, for the coarse grid of size 64×64, the current work demonstrated an average improvement in computational speed of approximately $\sim 35\%$ compared to CityFFD. For the fine mesh 128×128, the performance remained superior, showing an $\sim 18\%$ increase in speed.

In the case of CityFFD [28], reducing the cell size to half its original size required approximately $3\times$ more computational time, whereas the present code exhibited a steeper increase, requiring approximately $4 - 5\times$ more computational time. It's essential to note that these numbers are not directly comparable quantities, as CityFFD [28] utilizes multigrid methods for solving the elliptic equations, which inherently tend to be faster. Additionally, the processor employed in the CityFFD [28] work, the Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, has been benchmarked to be approximately 10% more powerful, compared to the current processor unit, AMD Ryzen 5 2500U CPU @ 2 GHz. Despite this, the recorded clock times remained smaller than those of CityFFD, possible due to the Gauss-Seidel having limited iterations and lax convergence criteria. However, subsequent doubling of the number of nodes would probably result in suboptimal outcomes.

Regarding the performance of each interpolation scheme, the linear scheme exhibits the fastest performance as it involves the fewest cells for interpolation. In contrast, the 4^{th} order central scheme consumes approximately 50% more computational resources. The back-and-forth sweep method achieves similar accuracy to the 4^{th} order scheme with approximately $15 - 20\%$ less computing time, aligning with the findings of M. Dorostkar [28].

Table 4.2: Clock Time of CityFFD and Present Thesis Applications for Different Interpolation Schemes

Schemes	CityFFD		Present Thesis		Difference [%]
	Grid	Clock Time [sec]	Grid	Clock Time [sec]	
Linear	64×64	21	64×64	11.13	**-47.00**
	128×128	67	128×128	47.28	-29.43
4th Order Central	64×64	26	64×64	18.08	-30.46
	128×128	82	128×128	72.52	-11.56
Back & Forth Sweep	64×64	22	64×64	14.29	-35.05
	128×128	73	128×128	61.87	-15.25

In Table 4.2 clock time refers to the approximate processor time that is consumed by the program, during execution. Simulation time represents the physical time that the transient solver actually solved, in order to reach steady state.

4.2.2 Model Validation for Various Reynolds Numbers

The validated model with the back & forth interpolation scheme is tested at various Reynolds numbers and compared with the tabular results of Ghia et al. [39]. In Figures 4.11 and 4.12, the u and v profiles are respectively depicted along the centerlines of the cavity for Reynolds numbers $Re = 100, 400, 1000, 3200, 5000$. Two grid resolutions, namely 64×64 and 128×128, are considered. For lower Reynolds numbers, the profiles generated by the different meshes exhibit almost indistinguishable characteristics.

However, as the Reynolds number increases, so does the complexity of the flow. Consequently, high Reynolds number flows require finer grids to capture the details of the flow accurately. As a result, differences in the profiles become more pronounced at $Re = 3200$ and $Re = 5000$, especially for the coarse grid of 64×64.

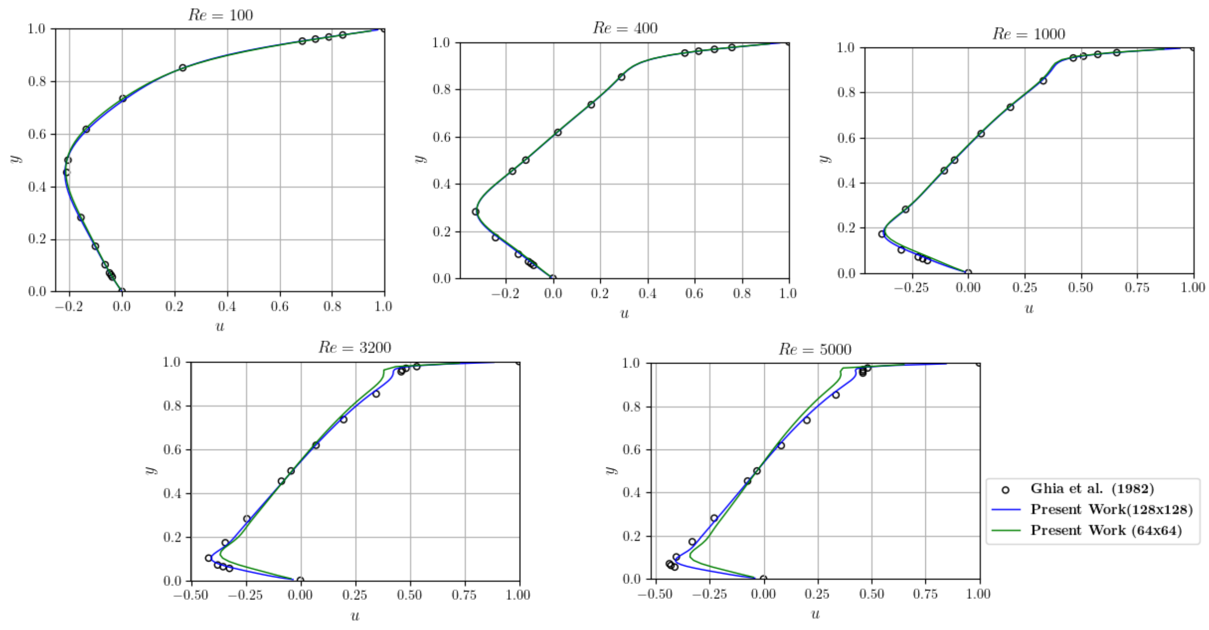


Figure 4.11: Horizontal Profile u with Back & Forth Sweep Interpolation for Various Reynolds Numbers and Uniform Mesh for Grids: 64×64 and 128×128

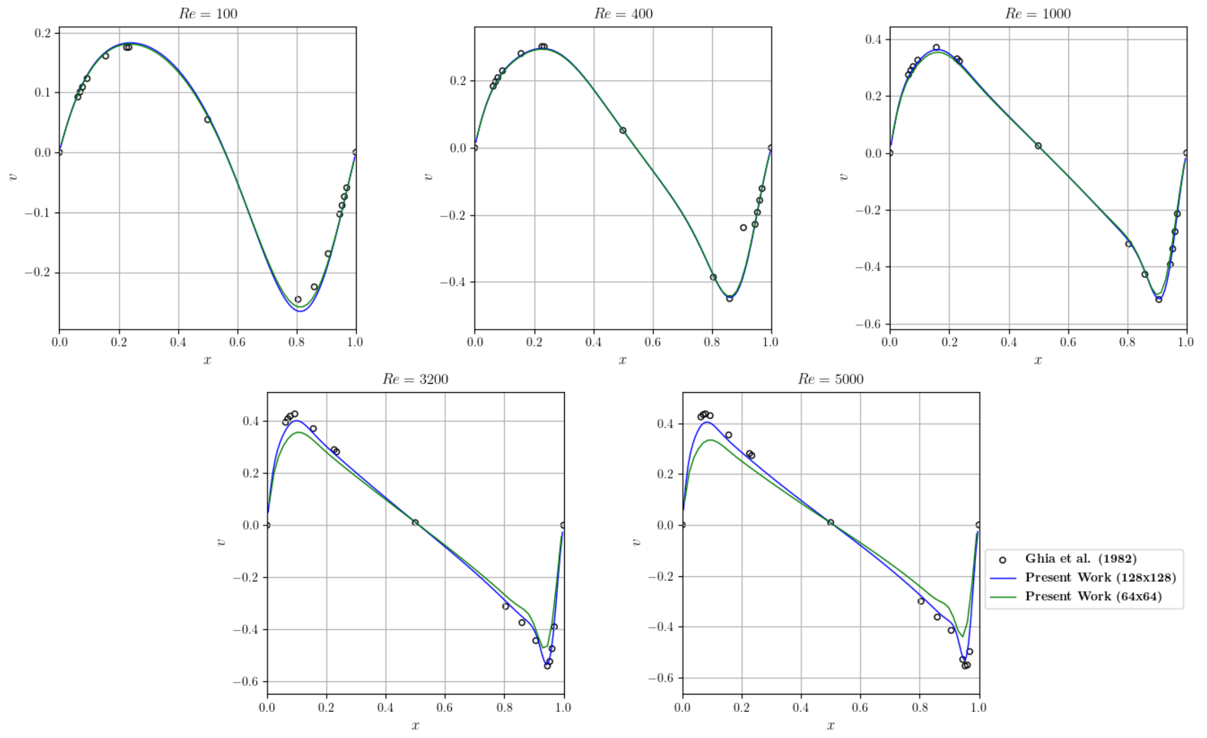


Figure 4.12: Vertical Profile v with Back & Forth Sweep Interpolation for Various Reynolds Numbers and Uniform Mesh for Grids: 64×64 and 128×128

Additionally, when the Reynolds number increases, the core of a vortex tends to exhibit characteristics of solid-body rotation. This means that the fluid within the core rotates as if it were a solid object, with circular streamlines and a consistent vorticity throughout. This behavior becomes particularly evident when observing velocity profiles, as at high Reynolds numbers, these profiles become increasingly linear within the primary vortex, indicating uniform rotation of the fluid. This phenomenon arises due to the dominance of inertial effects in the flow at high Reynolds numbers.

Table 4.3 shows the Root Mean Square Error (RMSE) between the solver employed in this thesis for the fine grid case and the benchmark results from Ghia et al. [39]. Notably, the RMS error is higher by an order of magnitude at high Reynolds numbers compared to the lower Reynolds number cases, but it still remains remarkably small. Furthermore, the solver encounters difficulties in achieving convergence as the Reynolds number increases, as presented in Table 4.4.

Table 4.3: Comparative Error of Present Thesis Results to Ghia et al. [39] Simulations for $\Delta t = 0.005\ sec$, Grid: 128×128

Reynolds Number	RMSE	
	U-Profile	V-Profile
100	0.0042212	0.0099060
400	0.0056102	0.0035536
1000	0.011064	0.0058725
3200	0.022208	0.020566
5000	0.032113	0.033482

Table 4.4: Simulation and Clock Time of Present Thesis Solver for Various Reynolds Numbers for $\Delta t = 0.005\ sec$, Grid:128×128

Reynolds Number	Simulation Time[sec]	Clock Time [sec]
100	15.135	23.25
400	29.425	43.53
1000	39.55	61.87
3200	107.515	168.88
5000	201.925	320.34

The following contours illustrate the velocity streamlines resolved by the current solver for the examined Reynolds numbers. As the Reynolds number increases, notable changes occur within the flow dynamics of the cavity. Initially, at low Reynolds numbers, viscous forces dominate, leading to steady and straightforward flow patterns. However, as the Reynolds number escalates, the flow becomes more intricate and as result, multiple vortices and recirculation zones are formed. Literature on the subject [41] highlights that the two-dimensional flow is not stable at high Reynolds numbers as, beyond a critical value, smaller-scale vortices are shed into the cavity from the downstream end of the moving wall.

From the velocity streamlines it is evident, that as the Reynolds number increases, the flow dynamics within the cavity undergo significant changes. Initially, the primary vortex is displaced from the cavity's geometric center, as shown in Figure 4.13 for $Re = 100$, but gradually migrates towards it as the Reynolds number grows. This phenomenon can be attributed to the increasing dominance of inertial forces over viscous forces, resulting in a more organized flow pattern.

As the flow progresses downstream of the lid, there's a notable acceleration, which leads to a significant asymmetry in the vortices formed at the lower corners of the domain. Specifically, the vortex located at the bottom right (BR_1) tends to be larger in size compared to its counterpart on the bottom left (BL_1). At lower Reynolds numbers, the bottom left vortex may even be negligible. However, as the Reynolds number increases, the separated vortices at the bottom gain strength and size.

The emergence of an upper upstream eddy signifies a notable shift in the flow's topology. The presence of this vortex becomes apparent even at early Reynolds numbers, such as $Re = 400$, where a subtle curvature is observed at the upper left corner. Ghia et al. [39] emphasizes that this vortex is different in nature, compared to the two lower eddies, which are an infinite sequence of eddies.

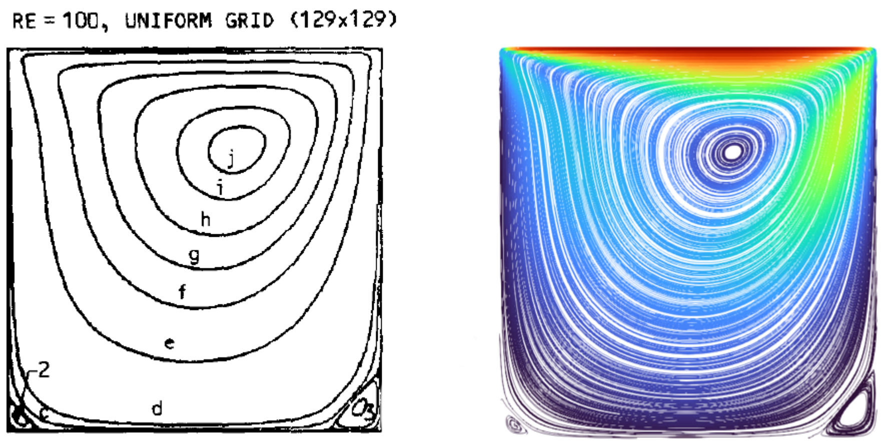


Figure 4.13: Comparison of Present Work Velocity Streamlines to Ghia et al. [39] for $Re = 100$, $\Delta t = 0.005\,sec$, Grid: 128×128

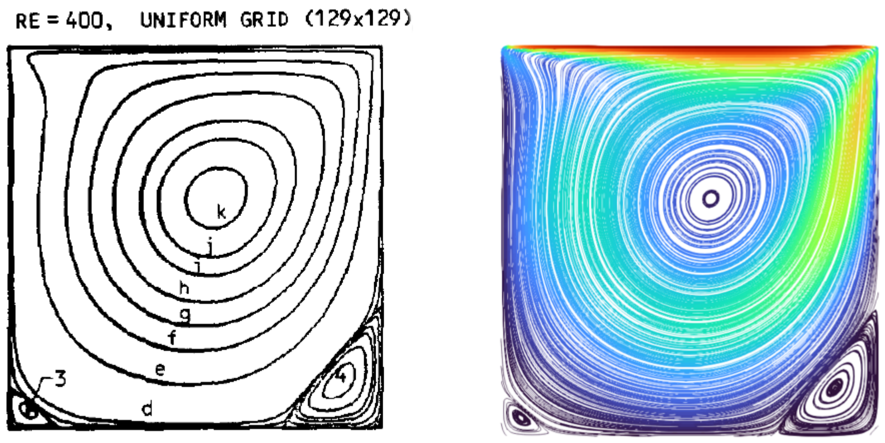


Figure 4.14: Comparison of Present Work Velocity Streamlines to Ghia et al. [39] for $Re = 400$, $\Delta t = 0.005\,sec$, Grid: 128×128

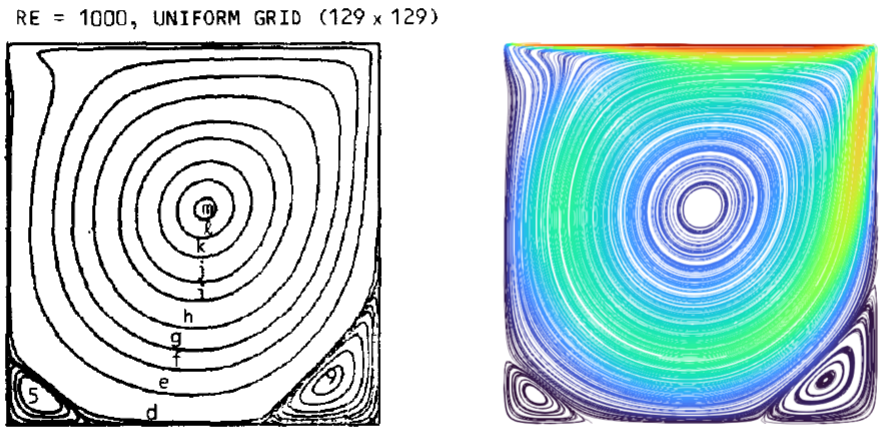


Figure 4.15: Comparison of Present Work Velocity Streamlines to Ghia et al. [39] for $Re = 1000$, $\Delta t = 0.005\,sec$, Grid: 128×128

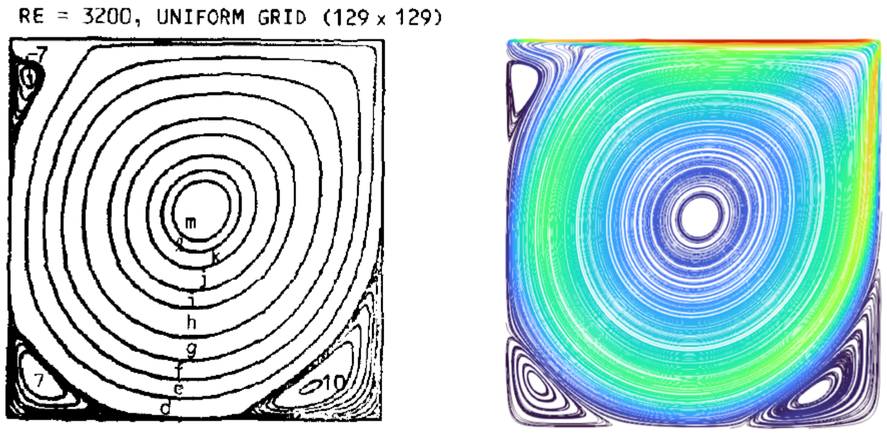


Figure 4.16: Comparison of Present Work Velocity Streamlines to Ghia et al. [39] for $Re = 3200$, $\Delta t = 0.005\,sec$, Grid: 128×128

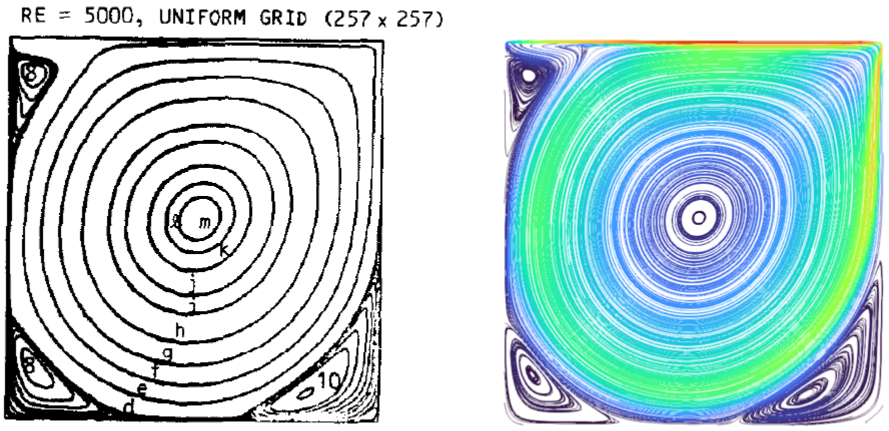


Figure 4.17: Comparison of Present Work Velocity Streamlines to Ghia et al. [39] for $Re = 5000$, $\Delta t = 0.005\,sec$, Grid: 128×128

4.3 Performance and Stability Analysis

In Fast Fluid Dynamics (FFD), algorithmic performance stands out as a key selling point, making it necessary to conduct a thorough investigation of the solver's overall efficiency. The performance of each individual FFD step is examined, and the overall performance is compared to OpenFOAM. In addition, various timesteps are explored, in order to validate the stability of the semi-Lagrangian scheme. This approach ensures a comprehensive understanding of FFD's capabilities and allows for meaningful comparisons with established solvers like OpenFOAM.

4.3.1 Performance Evaluation

The cavity case is revisited across five distinct Reynolds numbers, employing grid resolutions of 64×64 and 128×128.

Initially, the focus is on the number of FFD cycles required for the case to attain a steady-state solution. Notably, it is observed that the grid resolution exerts minimal impact on simulation iterations, suggesting a mesh-independent solution within the selected range of mesh sizes. In instances where simulation cycles differ, this variability may be ascribed to the convergence criteria of the linear system solvers. For instance, a low number of maximum iterations might result in insufficient solution convergence, thereby delaying the overall convergence process. As expected, cases with higher Reynolds numbers require the most FFD cycles due to the increased complexity of the flow dynamics.

Additionally, employing linear regression analysis on the data reveals an R^2 value of approximately ~ 0.98. The coefficient of determination, denoted by R^2, is utilized to assess how well the model fits the data, with a value close to 1 indicating a strong fit. However, it's crucial to note that this correlation is specific to the case under consideration and cannot be generalized to other cases or employed for predictions within the same case. It is an observation that lacks deterministic patterns.

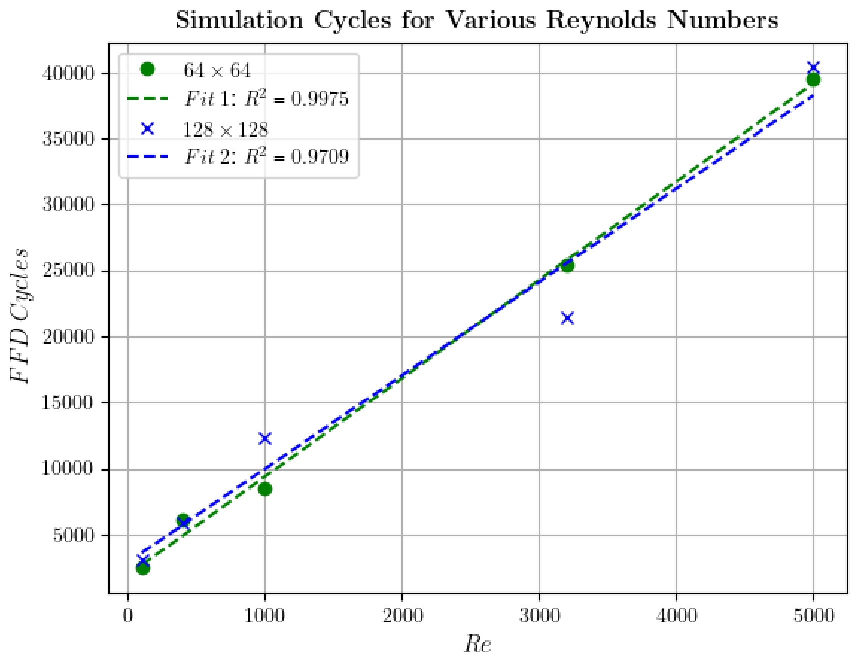


Figure 4.18: FFD Cycles for Various Reynolds Numbers for Present Work Simulations of $\Delta t = 0.005\ sec$ and Grids: 64×64, 128×128

Table 4.5: Number of FFD Cycles Needed for Convergence for $\Delta t = 0.005\,sec$

Reynolds Number	Grid	
	64×64	128×128
100	2457 (12.285 *sec*)	3027 (15.135 *sec*)
400	6062 (30.31 *sec*)	5884 (29.42 *sec*)
1000	8483 (42.315 *sec*)	12292 (61.46 *sec*)
3200	25358 (126.79 *sec*)	21503 (107.515 *sec*)
5000	39500 (197.5 *sec*)	40358 (201.925 *sec*)

In Chapter 2, the FFD algorithm is described, which decomposes the Navier-Stokes equations into four distinct steps. Table 4.6 records the clock times for each of these steps.

Table 4.6: Computational Cost of Each of FFD's Algorithm Step for Present Thesis Simulations for $\Delta t = 0.005\,sec$

Reynolds Number	Grid	Diffusion [sec]	Convection [sec]	Pressure Correction [sec]	Pressure Gradient [sec]	Clock Time [sec]
100	64×64	1.178	2.709	0.527	0.0744	4.66
	128×128	6.097	13.842	2.804	0.480	24.03
400	64×64	2.874	6.617	1.293	0.177	11.16
	128×128	11.889	26.966	5.451	0.910	46.00
1000	64×64	3.364	8.840	1.706	0.186	14.29
	128×128	15.874	34.154	6.905	1.038	61.46
3200	64×64	8.819	27.756	5.272	0.714	42.90
	128×128	38.539	93.017	18.578	2.815	154.195
5000	64×64	12.588	43.849	8.316	1.129	66.30
	128×128	69.791	184.653	36.913	5.642	298.74

Notably, the convection step, which employs the method of characteristics, requires the most computational resources, accounting for approximately $55 - 65\%$ of the total simulation time. It is important to highlight that despite its significant time consumption, the convection step does not involve any iterations. Instead, it computes the departure points for each node in the mesh and interpolates the surrounding neighbors for velocity calculation. The poor performance is most likely attributed to the coding, such as unnecessary assignments and excessive initialization of variables. It should be mentioned, though, that the convection step is perfect for parallelization, which would significantly minimize the computational cost of the FFD algorithm as a whole.

The diffusion and pressure correction steps each demand approximately $\sim 23.9\%$ and $\sim 12.2\%$ of the computational workload, respectively. In the diffusion step, two linear systems of equations are solved—one for each velocity component—making it comparable in computational cost to the pressure correction step. Essentially, the pressure correction computational cost is equal to one Gauss-Seidel iterative process. While solving the pressure equation is inherently more complex and could theoretically require more computational time, the minimum iterations factor serves to minimize the clock time.

Considering the relatively slow nature of the Gauss-Seidel iterative method, there is potential for improvement in computational efficiency by exploring alternative solvers like the multi-grid method.

Finally, the pressure gradient term has a negligible impact on the performance of the solver, as it simply calculates the pressure gradient and updates the velocity for each mesh node.

Increasing the Reynolds number or refining the mesh does not significantly alter the distribution of computational resources during the simulation, as evidenced by the data presented in Table 4.7. However, halving the cell size, and consequently quadrupling the number of nodes, results in approximately a $4-5\times$ increase in the time required for each step. Specifically, diffusion time is multiplied on average by 4.78, convection time by 4.12, pressure correction time by 4.31 and lastly pressure gradient time by 5.22. This indicates that the algorithm's complexity scales proportionally with the number of nodes.

Table 4.7: Computational Cost of Each of FFD's Algorithm Step as Percentage of the Total Clock Time for Simulations for $\Delta t = 0.005\, sec$

Reynolds Number	Grid	Diffusion [%]	Convection [%]	Pressure Correction [%]	Pressure Gradient [%]
100	64×64	25.27	58.13	11.30	1.59
	128×128	25.37	57.60	11.67	1.99
400	64×64	25.75	59.26	11.58	1.59
	128×128	**25.85**	58.62	11.85	**1.98**
1000	64×64	23.54	61.86	11.94	1.30
	128×128	25.83	55.57	11.23	1.68
3200	64×64	20.56	64.69	12.28	1.66
	128×128	24.99	60.32	12.05	1.83
5000	64×64	18.98	**66.14**	**15.54**	1.70
	128×128	23.36	61.81	12.35	1.89

4.3.2 Performance Comparison with OpenFOAM Simulation

The performance comparison to the CityFFD model, as conducted in previous paragraphs, lacks the depth necessary to draw significant conclusions, as it merely validates the current solver. To provide a more comprehensive analysis, the performance of OpenFOAM for the lid-driven cavity case is also evaluated. This case serves as a standard benchmark within the OpenFOAM tutorials folder. By comparing these two approaches, the aim is to assess whether the Fast Fluid Dynamics (FFD) algorithm indeed outperforms conventional methods.

In OpenFOAM environment, Pressure-Implicit with Splitting of Operators (PISO) algorithm is one the most popular transient solvers. PISO employs a methodology consisting of one predictor step followed by two corrector steps, all aimed at upholding mass conservation through predictor-corrector iterations. Unlike approaches that tackle all coupled equations simultaneously or through iterative sequences, PISO breaks down the operators into an implicit predictor and several explicit corrector steps. This approach isn't typically viewed as iterative, as only a minimal number of corrector steps are typically required to achieve the desired precision. Each time step involves predicting velocity, then subsequently correcting pressure and velocity to refine the solution.

While a comprehensive explanation of PISO can be found in relevant literature, it's essential to note that this thesis does not delve into the intricacies of the algorithm itself but rather acknowledges its significance in CFD simulations within the OpenFOAM framework.

The OpenFOAM case was meticulously set up to replicate the conditions of the FFD algorithm as closely as possible. This involved substituting the solvers for linear systems with plain Gauss-Seidel solvers, ensuring that convergence criteria and minimum iteration requirements were consistent. Additionally, the grid resolution and timestep parameters were matched to ensure equitable conditions. To maintain fairness in time recording, unnecessary file writing and logging between timesteps were avoided in both cases. This approach allowed for a focused evaluation of solver efficiency and computational speed under comparable conditions. Finally, the simulation was stopped upon reaching a steady state, which was determined by observing that the velocity equation no longer required further iterations for solutions.

In the OpenFOAM cases, the Reynolds numbers from the previous sections were reexamined with grid resolutions of 64×64 and 128×128, alongside a simulation timestep of $\Delta t = 0.005\ sec$. The results obtained from these simulations are presented in Table 4.8 for comprehensive comparison and evaluation.

An immediate observation reveals that the FFD algorithm outperforms the conventional OpenFOAM method. On average, it showcases a reduction in computational cost of approximately 29.39%, with the highest recorded improvement reaching 35.50%. However, upon closer examination, no clear evidence emerges to directly correlate the improved performance with either the Reynolds number or the grid resolution. The specific simulation times are omitted from the table as they closely resemble those of the FFD method, with no significant deviations observed.

Table 4.8: Clock Time of Thesis Solver and OpenFOAM Lid-Driven Cavity Case for Similar Simulation Parameters

Reynolds Number	Present Work		OpenFOAM		Difference [%]
	Grid	Clock Time [sec]	Grid	Clock Time [sec]	
100	64×64	4.58	64×64	6.78	−32.45
	128×128	23.25	128×128	34.05	−31.72
400	64×64	11.10	64×64	14.97	−25.85
	128×128	43.53	128×128	64.42	−32.43
1000	64×64	14.29	64×64	20.67	−30.87
	128×128	61.87	128×128	85.79	−27.88
3200	64×64	43.43	64×64	64.43	−32.59
	128×128	168.88	128×128	262.65	**-35.50**
5000	64×64	61.32	64×64	79.48	−22.84
	128×128	320.34	128×128	410.00	−21.87

While the improved performance is a significant advantage of the FFD algorithm, it is worth noting that the literature reports, as discussed in Chapter 1, suggest even greater improvements, often ranging from $5 - 10\times$ better performance. This underscores the potential of the FFD algorithm and highlights the need for further investigation into optimizing its implementation in the present thesis for even more substantial gains in computational efficiency. It's important to acknowledge that the present implementation may lack in coding proficiency in C++, whereas OpenFOAM is a simulation library maintained by professional programmers. This expertise in development and optimization could be a contributing factor to the observed performance disparities between the present FFD algorithm and the literature claims.

4.3.3 Stability Analysis for Timestep

The semi-Lagrangian scheme used for solving the convection term is a stable method that is not constrained by the Courant-Friedrichs-Lewy (CFL) number. CFL is a dimensionless number used in numerical simulations to access the stability and accuracy of numerical solutions for time-dependent partial differential equations. The CFL number must be minimized for explicit schemes, ensuring it remains below a certain value, typically around 1. However, for inherently more stable implicit schemes, a slightly larger CFL number may be deemed acceptable.

So far, the simulations have maintained a constant timestep of $\Delta t = 0.005$ *sec*. To evaluate the stability of the solver, a progressive increase in timestep was implemented for the $Re = 1000$ scenario, conducted on a grid resolution of 64×64. Remarkably, the solver was able to handle timesteps up to $\Delta t = 0.14$ *sec*. However, upon reaching this timestep, the Courant number surged to an extremely large value, causing the solver to fail. The maximum recorded CFL number during the simulations peaked at approximately ~ 32 for a timestep of $\Delta t = 0.1325$ *sec*. In contrast, the OpenFOAM's PISO simulation, under similar parameters, successfully handled a timestep of $\Delta t = 0.02$ *sec*, as its Courant number remained close to 1. Moreover, as the timestep increased, the FFD solver demonstrated exceptional speed, albeit at the expense of solution accuracy.

Table 4.9: Comparison of Solver and OpenFOAM Clock and Simulations Times for Various Timesteps for $Re = 1000$, Grid 64×64

Timestep[sec]	Present Work		OpenFOAM		Max CFL	RMSE
	Clock Time[sec]	Simulation Time[sec]	Clock Time[sec]	Simulation Time[sec]		
$\Delta t = 0.001$	58.33	34.126	95.81	30.435	0.056	0.024656
$\Delta t = 0.005$	14.29	42.415	24.47	41.045	0.28	0.031229
$\Delta t = 0.01$	8.88	47.47	14.28	47.78	0.57	0.041682
$\Delta t = 0.02$	5.44	56.82	9.629	67.18	1.13	0.061845
$\Delta t = 0.05$	3.16	78.9	–	–	2.81	0.11085
$\Delta t = 0.1$	3.47	166.9	–	–	5.52	0.17041
$\Delta t = 0.12$	3.96	229.68	–	–	6.59	0.19162
$\Delta t = 0.13$	3.54	215.41	–	–	7.14	0.20275
$\Delta t = 0.1325$	3.78	243.27	–	–	32.71	0.20496
$\Delta t = 0.14$	–	–	–	–	–	–

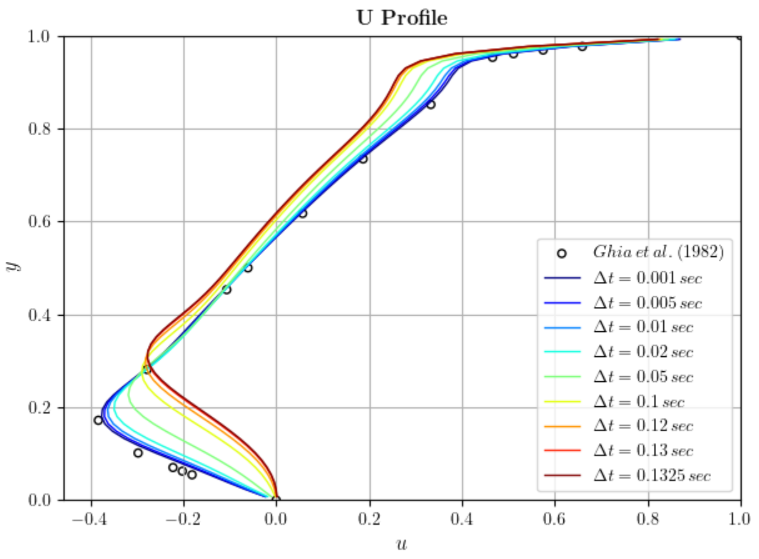


Figure 4.19: Horizontal Velocity Profile u for Present Work Simulations for Various Timesteps for $Re = 1000$, Grid: 64×64

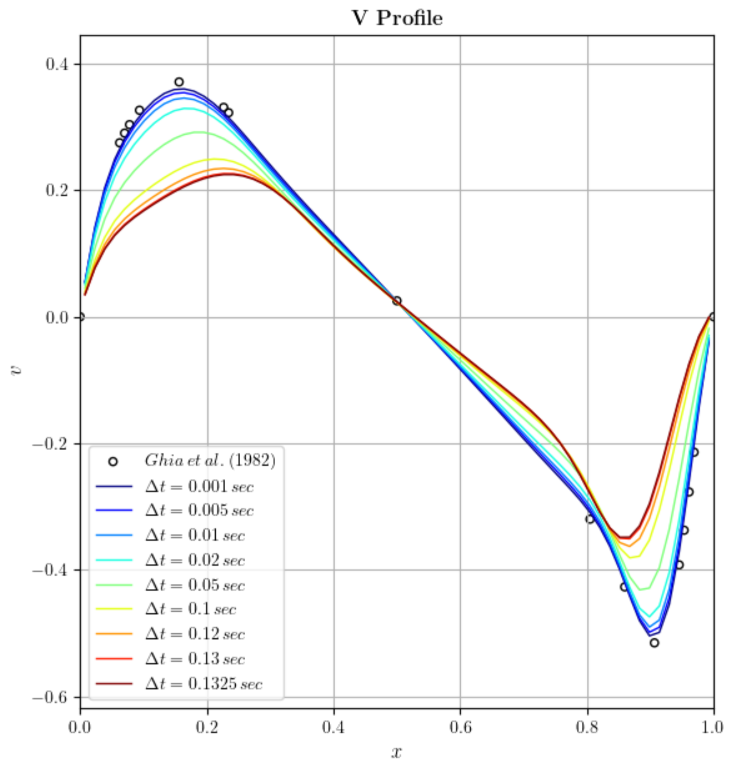


Figure 4.20: Vertical Velocity Profile v for Present Work Simulations for Various Timesteps for $Re = 1000$, Grid: 64×64

The provided velocity profiles demonstrate that as the timestep increases beyond $\Delta t = 0.02 - 0.05\ sec$, the solution of the flow progressively diverges from the reference solutions documented by Ghia et al. [39]. Interestingly, for timesteps below this threshold, $\Delta t = 0.02 sec$, which coincidentally aligns with the maximum timestep OpenFOAM could simulate, the velocity profiles closely match Ghia et al.'s [39] solutions. The streamlines depicted in Figure 4.21 further illustrate this point, with the primary vortex consistently located at the geometric center of the cavity across timesteps below $\Delta t = 0.02$ *sec*. Additionally, the sizes of the lower eddies maintain their size, compared to the most accurate solution for timestep $\Delta t = 0.001$ *sec*.

For timesteps exceeding $\Delta t = 0.1$ *sec*, corresponding to Courant numbers larger than 5.5, the flow exhibits a more rapid damping, likely due to numerical dissipation. Consequently, the primary vortex starts to drift towards the upper corner of the cavity, as the velocity magnitude is lower and cannot maintain the vortex's rotation. This movement creates additional space for the lower right vortex (BL_1) to expand in size, further distorting the primary vortex.

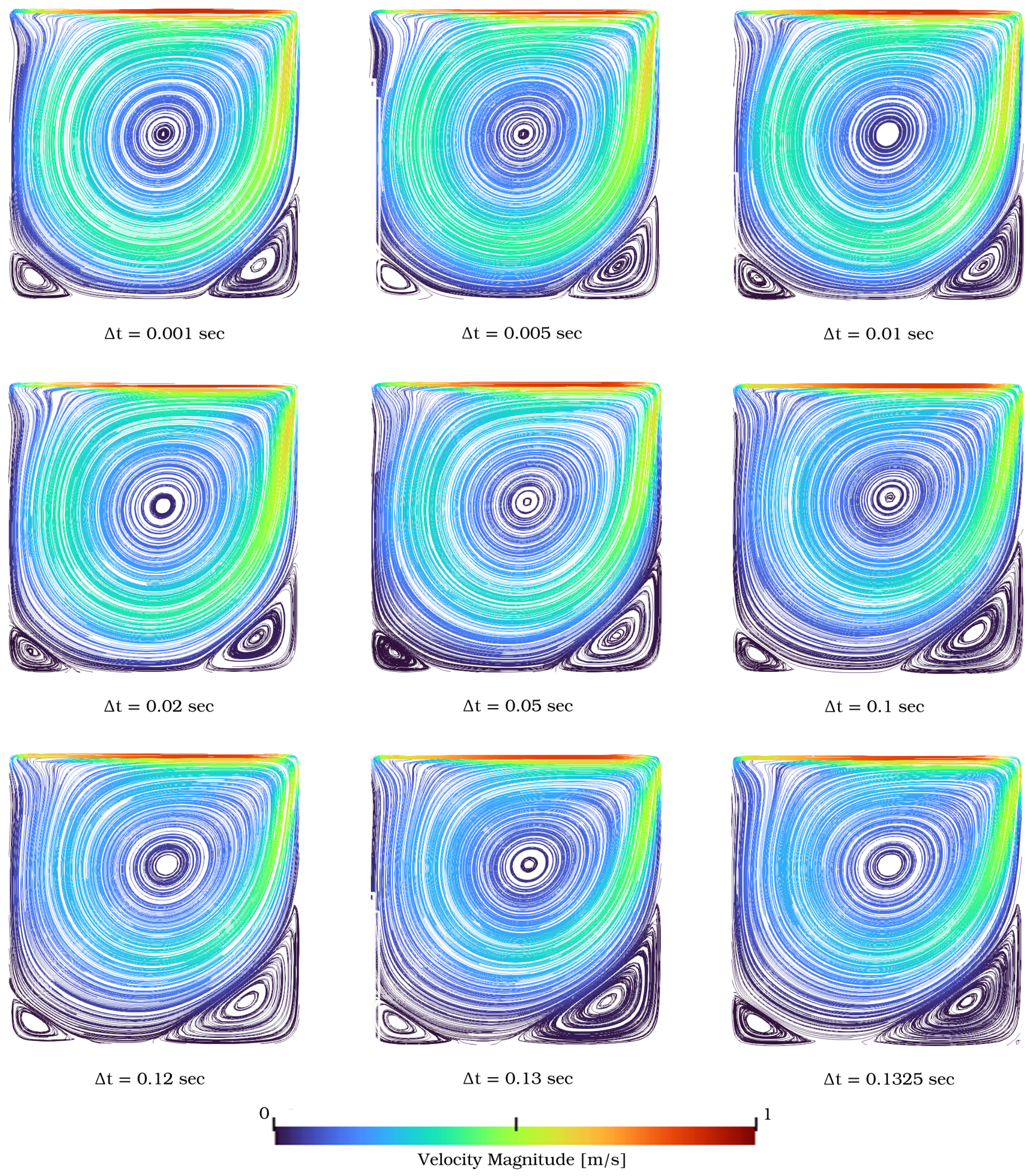


Figure 4.21: Velocity Streamlines for Present Work Simulations for Back & Forth Sweep for Various Timesteps for $Re = 1000$, Grid: 64×64

4.4 Nudging

In Chapter 3, nudging was introduced as a cost-effective and fast technique for integrating experimental results into numerical simulations. To validate the effectiveness of this method, the velocity profiles provided by Ghia et al. [39] served as the reference observation values. The investigation of the method's impact involved utilizing various sets of observations, as depicted in the figure below. Specifically, the values at each centerline, comprising 16 observation points each, were individually employed. Subsequently, these observations were combined to assess the overall improvement in accuracy achieved through the nudging technique and its consistency across different observation sets.

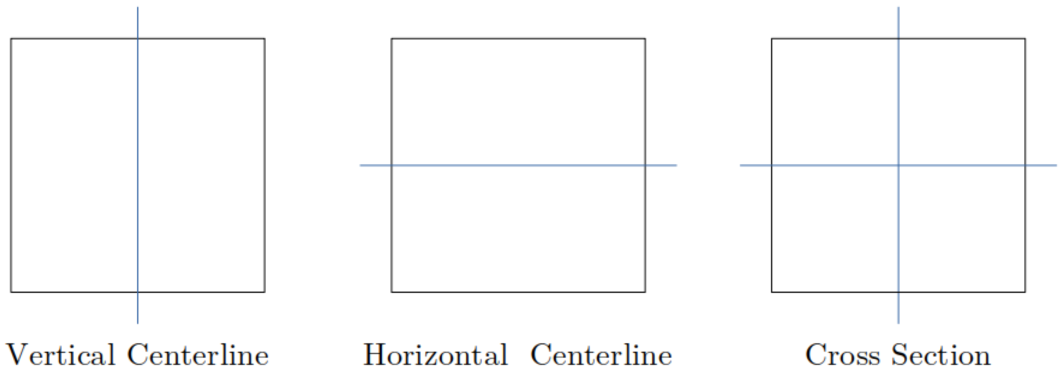


Figure 4.22: Configuration of Nudging Observation's Positioning

The nudged equation is reiterated here for clarity. To begin, it is essential to establish the nudging factor, denoted as α. As mentioned earlier, a practical estimation for α is derived from the simulation timestep, as discussed in Leoni et al. [38]. In all preceding simulations, the timestep has consistently been set to $\Delta t = 0.005\, sec$, so according to the approximation, the optimal α value hovers around $\alpha \sim 200$.

$$\frac{\partial \vec{U}_a}{\partial t} = \nu \nabla^2 \vec{U}_a + \vec{f} + \alpha(\vec{U}_{obs} - \vec{U}_a) \tag{4.4}$$

To effectively determine the appropriate value for the parameter α, a brief parametric analysis is carried out. Figure 4.23 illustrates a comparison of various α values, taking into consideration key criteria such as simulation time, clock time, and the Root Mean Square Error (RMSE) between the nudged cross-section simulation on the coarse grid (64×64) and the high-order fine grid (128×128) simulation for Reynolds number $Re = 1000$. This analysis aims to identify the optimal α value that strikes a balance between computational efficiency and accuracy. Although values in the vicinity of 200 exhibit a small RMSE, the overall optimal point, primarily based on clock time considerations, is found to be $\alpha \sim 95$. For larger parameters, the code has difficulty converging and for very large values, it fails to converge altogether. It is crucial to note that parameter α is inherently case-specific, and further experimentation revealed its sensitivity to different scenarios. For instance, when dealing with cross-section nudging at a consistent Reynolds number, such as $Re = 5000$, which inherently presents convergence challenges, the chosen α value did not converge. In stark contrast, convergence is attained for the vertical centerline nudging under the same conditions. This underscores a significant drawback of the nudging method, that it is not an automated process and relies heavily on user input.

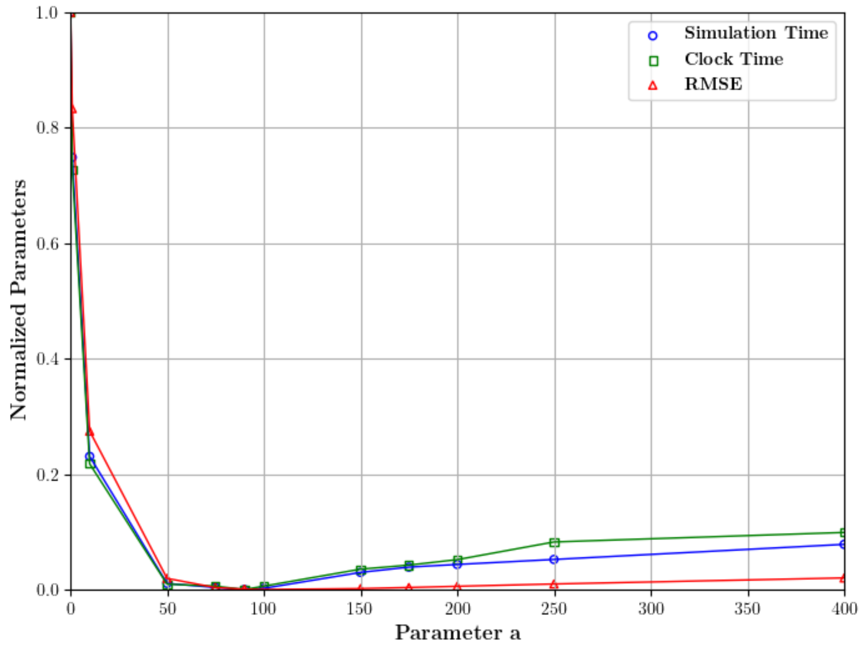


Figure 4.23: Evaluation of Various α values for $Re = 1000$ based on Simulation Time, Clock Time for Cross-Section Simulation on Grid 64×64 and RSME for u and v Profiles Compared to Back & Forth Sweep on Grid 128×128

The optimal value of $\alpha \sim 95$ for cross-section nudging has been implemented for both the vertical and horizontal centerline cases. In Figure 4.23, it can be observed that the root mean square error remains relatively stable around this value and up to $\alpha \sim 200$. This suggests that the effectiveness of the nudging technique is consistent across these parameters. Consequently, it seems reasonable to infer that the other two nudging cases do not demonstrate significant deviations.

In the following figures, the velocity profiles for each assimilation case are depicted. Upon initial inspection, a striking improvement is evident in the nudged profiles compared to the original linear interpolation simulation. This improvement signifies the nudging term's capability to compensate for flow physics that might not be accurately simulated due to numerical errors. Notably, in all three cases, the nudged profiles closely match the profiles derived from the simulation on the fine grid using the back & forth sweep method, which was significantly more computationally expensive.

As expected, when using exclusively the set of observations from one centerline, the corresponding velocity profile showed the most significant improvement. For instance, when assimilating the u profile on the vertical centerline, the horizontal velocity exhibited better results compared to the vertical velocity. However, it should be noted that even though the vertical profile was not as accurately matched as the horizontal one, it still showed considerable improvement compared to the poor linear interpolation case. This is further confirmed by the comparative errors presented in Table 4.10, where the RMSE values are approximately $8\times$ smaller for the u profile and around $5\times$ smaller for the v profile, compared to the linear interpolation case. These substantial reductions in RMSE highlight the effectiveness of the nudging technique in significantly enhancing the accuracy of velocity profiles, particularly when utilizing observations from a single centerline. Additionally, nudging for the cross-section case yields the smallest overall error, with a mere 1% difference compared to the back & forth sweep scheme on a finer 128×128 grid.

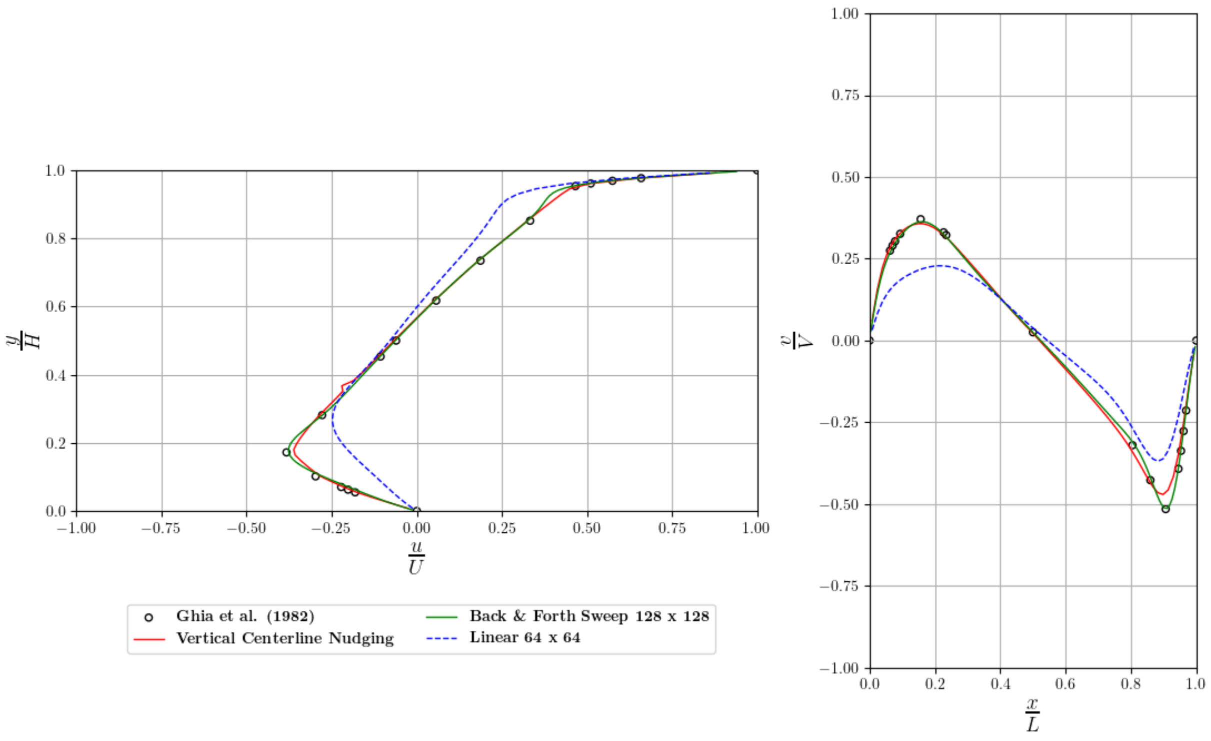


Figure 4.24: Velocity Profiles at Cavity Centerlines for Nudging Location at Vertical Centerline for $Re = 1000$, $\alpha = 95$, $\Delta t = 0.005\ sec$

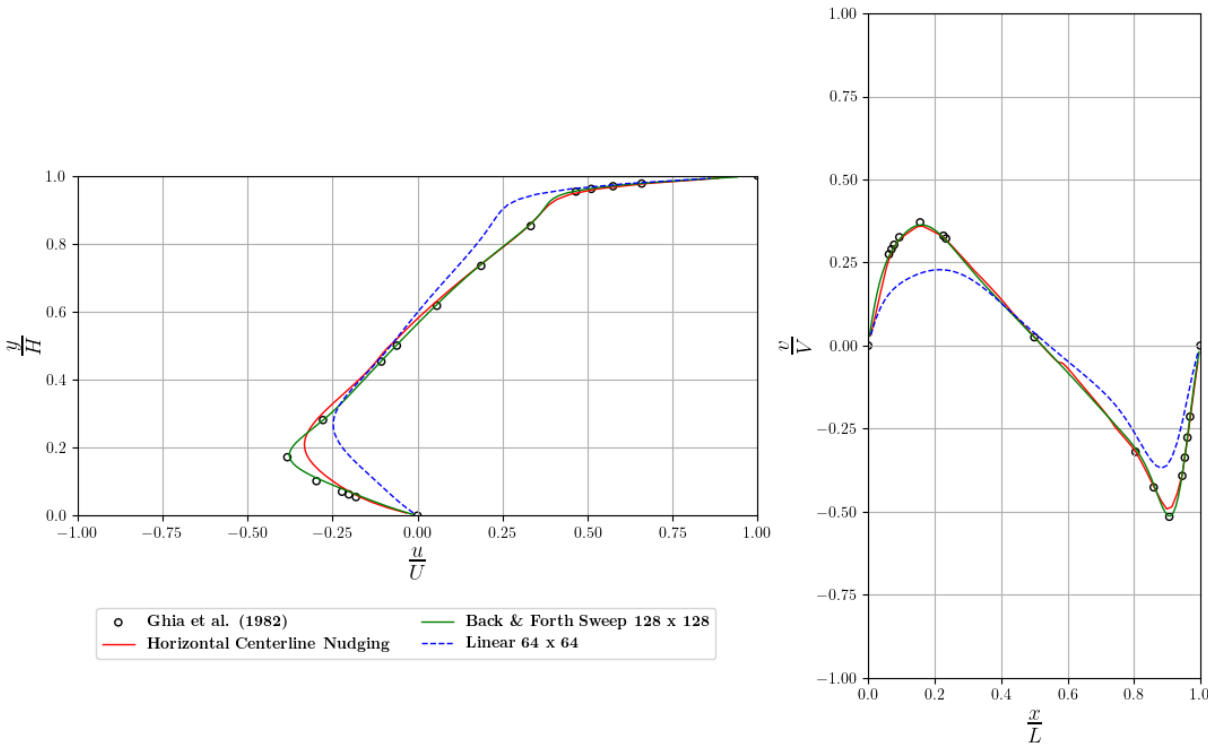


Figure 4.25: Velocity Profiles at Cavity Centerlines for Nudging Location at Horizontal Centerline for $Re = 1000$, $\alpha = 95$, $\Delta t = 0.005\ sec$

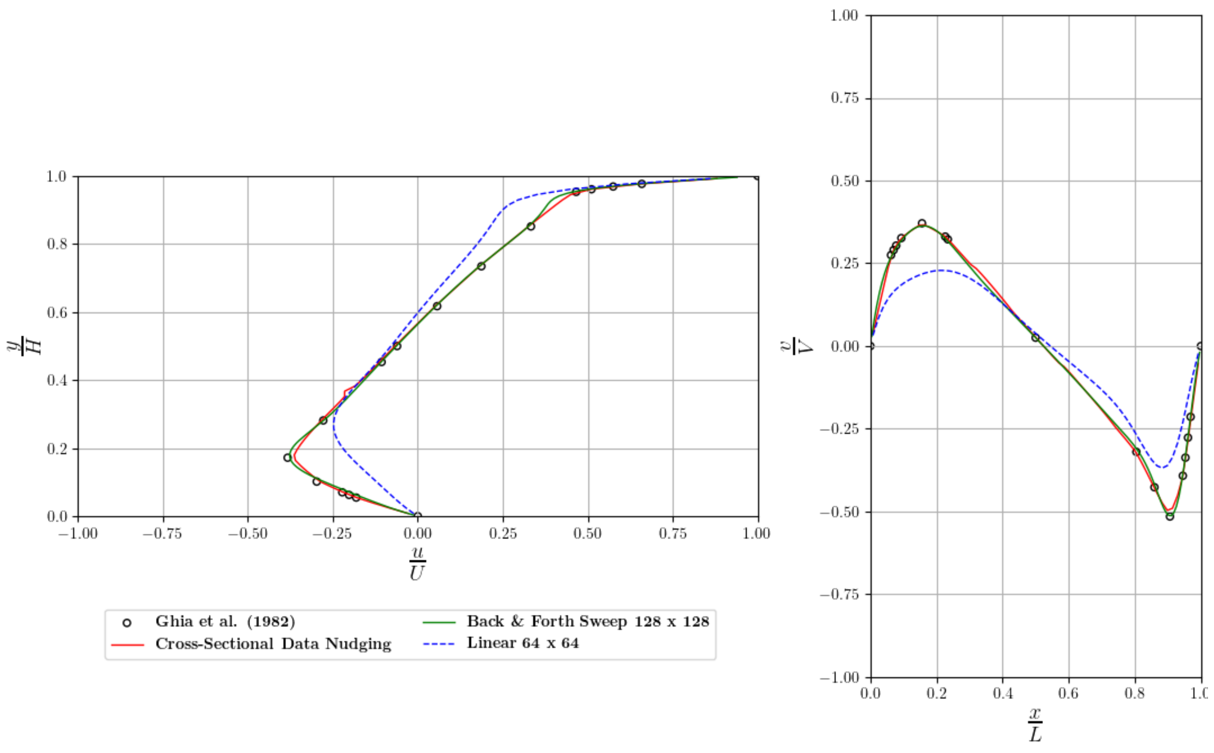


Figure 4.26: Velocity Profiles at Cavity Centerlines for Nudging Location at the Cross-Section for $Re = 1000$, $\alpha = 95$, $\Delta t = 0.005$ *sec*

Table 4.10: Comparative Error of Nudging Results for Grid 64×64 to Back & Forth Sweep Interpolation Method for $Re = 1000$, $\Delta t = 0.005$ *sec*, Grid 128×128

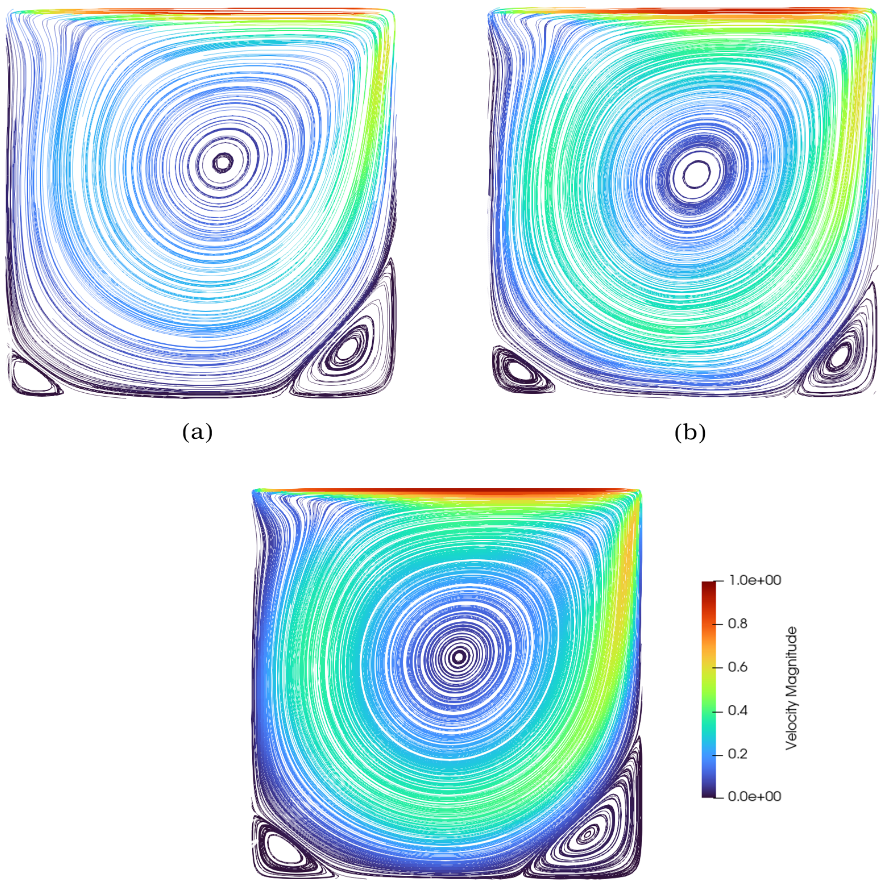
Case	RMSE	
	U-Profile	V-Profile
Linear Scheme 64×64	0.084931	0.081421
Vertical Centerline Nudging	0.010477	0.016494
Horizontal Centerline Nudging	0.020177	0.011528
Cross-Sectional Nudging	0.010188	0.011308

Performing nudging techniques not only improved the accuracy of the solution, but it also significantly lowered the time needed for the solution to reach a steady state. This reduction in computational time is a crucial advantage, as it enhances the efficiency of the simulation process. By effectively guiding the model towards a more accurate representation of the flow dynamics, nudging minimizes the transient phase and accelerates the convergence to a stable solution. This is evidenced by the simulation times needed by each case, with the nudged cases requiring the least number of FFD cycles to converge. In comparison to the original linear interpolation case conducted on the coarse grid, nudging accelerated the simulation by a factor of 2 to 4, depending on the number of observation values integrated into the simulation. Notably, cross-sectional nudging, which involves the highest density of observation points, exhibited the swiftest convergence and lowest clock time. Finally, given that the velocity profiles closely resemble the results of the back & forth sweep method on the fine grid, a comparison of computational time is warranted. Remarkably, nudging on cases with only one set of observations needed $11 - 13\times$ less computational resources, while cross-sectional performed $22\times$ faster.

Table 4.11: Simulation and Clock Time for the Nudging Cases for $Re = 1000$, $\Delta t = 0.005\,sec$

Case	Simulation Time[sec]	Clock Time [sec]
Back & Forth Sweep 128×128	39.55	61.87
Linear Scheme 64×64	39.055	11.13
Vertical Centerline Nudging	18.04	4.602
Horizontal Centerline Nudging	23.41	5.406
Cross-Sectional Nudging	10.64	2.832

With the relatively minor discrepancies between the nudged cases, the streamlines appear similar. To better visualize the impact of nudging on the simulation, the velocity streamlines of the cross-section nudged case are compared to the linear interpolation for the same grid and the back & forth sweep method on 128×128. In contrast to the linear case, the nudged case effectively repositions the primary vortex closer to the geometric center of the cavity, resulting in smaller lower vortices. Although the positioning resembles that of the high-order case on the fine grid, there are notable differences, particularly in the magnitude of the velocity, which affects the size of the bottom-right (BL_1) vortex. Overall, the nudged case proves to be a viable alternative to high-order schemes, even when applied to coarser grids.



(c)

Figure 4.27: Velocity Streamlines for (a) Linear Scheme on Grid 64×64 ; (b) Cross-Section Nudging ; (c) Back & Forth Sweep on Grid 128×128 for $Re = 1000$, $\Delta t = 0.005\,sec$

In real-life scenarios and experimental testing, the number of measurement points is often significantly fewer than the 17 points per velocity profile documented by Ghia et al. [39]. In the following figures, the number of observed values has been reduced to three points per cavity centerline, for two different testing locations (cases (a) and (b)). Similar, to previous simulations, nudging is conducted on a 64×64 grid, employing linear scheme interpolation for convection. Nonetheless, the results exhibit a poor performance compared to prior cases. This is primarily due to the limited number of points, which exerts a concentrated impact, affecting only a small subset of nodes in the mesh. Consequently, the observed values induce a highly localized effect, influencing the flow solely in close proximity to them. Ideally, it would be necessary to employ appropriate interpolating methods and weighting functions to distribute the observations across as many cells as possible. Despite the nudged cases displaying a lower error when compared with the linear case, the velocity profiles still lack smoothness and exhibit unnatural spikes around the observation points. Additionally, the topology of the observation points significantly affects the performance of nudging, as seen in Table 4.12, warranting further investigation.

Table 4.12: Simulation and Clock Time and RMSE Compared to Back & Forth Sweep on 128×128 for the Nudging Cases (a) and (b) for $Re = 1000$, $\Delta t = 0.005\, sec$

Case	Simulation Time[sec]	Clock Time [sec]	RMSE
Back & Forth Sweep 128×128	39.55	61.87	–
Linear Scheme 64×64	39.055	11.13	0.082906
Case (a)	33.535	10.35	0.053389
Case (b)	26.225	7.73	0.048756

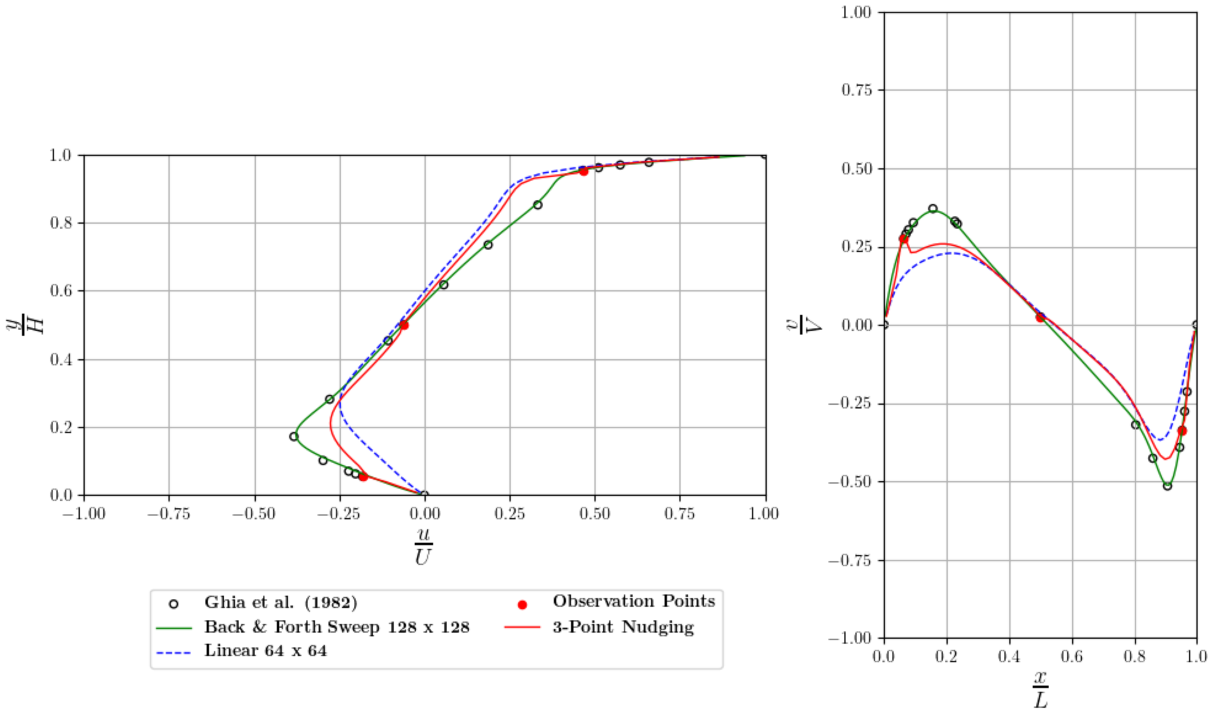


Figure 4.28: Case (a) - Velocity Profiles at Cavity Centerlines for 3-Point Nudging Locations at Cross-Section for $Re = 1000$, $\alpha = 200$, $\Delta t = 0.005\, sec$

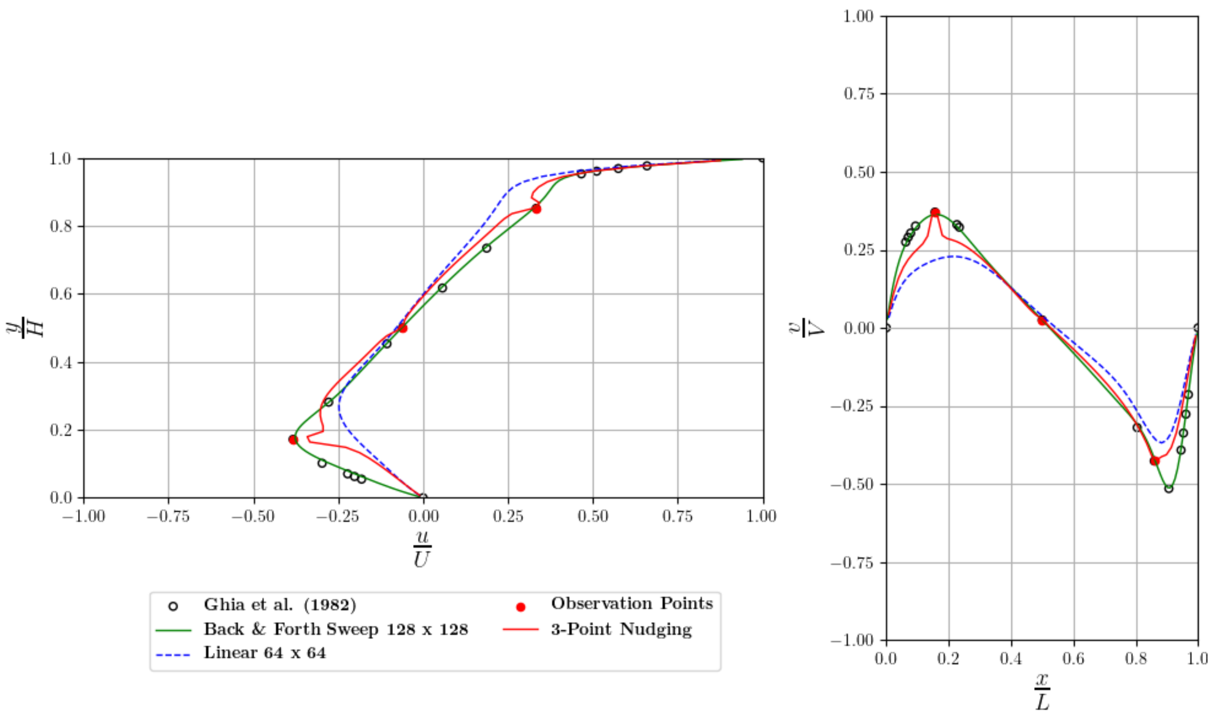


Figure 4.29: Case (b) - Velocity Profiles at Cavity Centerlines for 3-Point Nudging Locations at Cross-Section for $Re = 1000$, $\alpha = 200$, $\Delta t = 0.005\ sec$

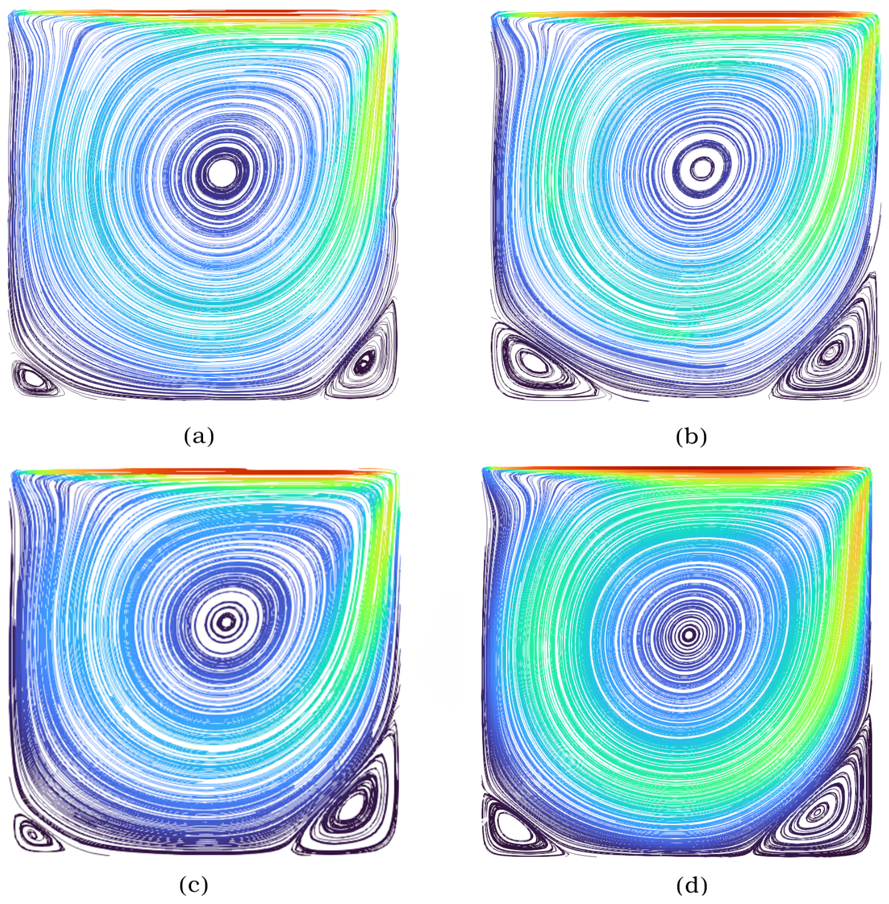


Figure 4.30: Velocity Streamlines for (a) Case **(a)** ; (b) Case **(b)** ; (c) Linear Scheme on Grid 64 × 64 (d) Back & Forth Sweep on Grid 128 × 128 for $Re = 1000$, $\Delta t = 0.005\ sec$

5. Application: Street Canyon

5.1 Problem Description

In recent years, there has been a growing trend among researchers to employ Computational Fluid Dynamics (CFD) for modeling microclimates and addressing environmental challenges in diverse urban settings, ranging from generic layouts to real-world cityscapes. The analysis of aerodynamics in urban areas, driven by concerns such as air quality, pedestrian comfort, energy consumption of buildings, and the dispersion of contaminants, plays a pivotal role in urban planning and design. For these reasons, researchers have extensively studied street canyons using both wind tunnel experiments and numerical models.

Street canyons, formed by closely situated buildings around narrow streets in densely populated cities, present unique challenges and opportunities for urban planners, as they can significantly impact the dispersion of pollutants and traffic emissions due to buildings hindering the wind flow. Consequently, it is crucial to gain accurate prediction of street pollution when designing cities that prioritize optimal natural ventilation and comfort.

In 2D simulations, employing a *T-shape* domain is a widely adopted practice due to its simplicity and computational efficiency, serving as a viable alternative to modeling a complex array of buildings and street canyons. Figure 5.1 illustrates that in a sequence of buildings, the flow fields tend to exhibit few differences beyond a certain number of structures. The use of a *T-shape* configuration can yield comparable results and capture the behavior in the downstream street canyons, albeit with slightly weaker flows [42].

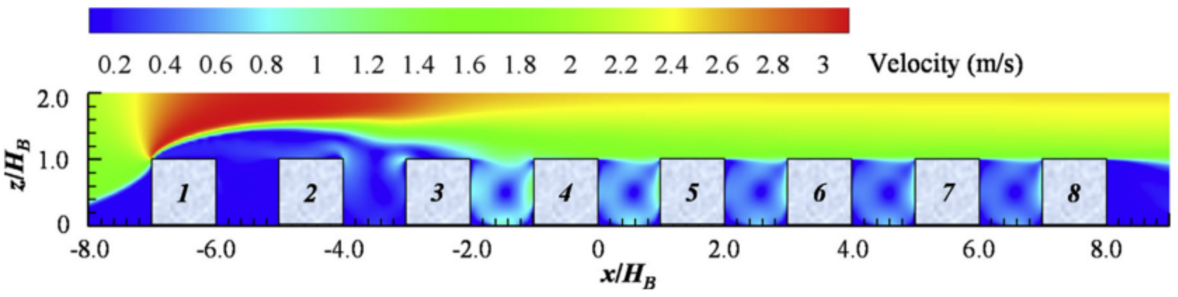


Figure 5.1: Predicted Velocity Contour on the Vertical Centerplane of the Street Canyon Model [42]

5.1.1 Boundary Conditions

Figure 5.2 shows a *T-shape* computational domain, the configuration of which is based on the work of Xiaomin et al. [43]. The upstream side of the street canyon is called the leeward wall with a height of H_1, and the downstream side is the windward wall with a height of H_2. The ratios of these building heights, H_1/H_2, and the ratio to the canyon width, H_1/W, are common geometric parameters examined in street canyon simulations.

Regarding the boundary conditions, a no-slip boundary condition is enforced on the wall surfaces, and wall functions are not employed. A symmetry plane is implemented at the top of the domain to ensure a parallel flow; thus, the vertical component of velocity is set to zero, and the fluxes of all other variables are also set to zero, preserving the inflow velocity profile. At the domain entrance, a velocity inlet is implemented with a Dirichlet condition for the horizontal velocity (u), either as a specified velocity profile or a constant freestream velocity. At the outlet, the flow is assumed to be fully developed, and a constant static pressure is applied. All other boundaries have a Neumann condition for pressure, and the outlet serves as the reference point for the pressure level. In cases where the gravity field is active, the outlet pressure must incorporate the hydrostatic pressure $+\rho g h$.

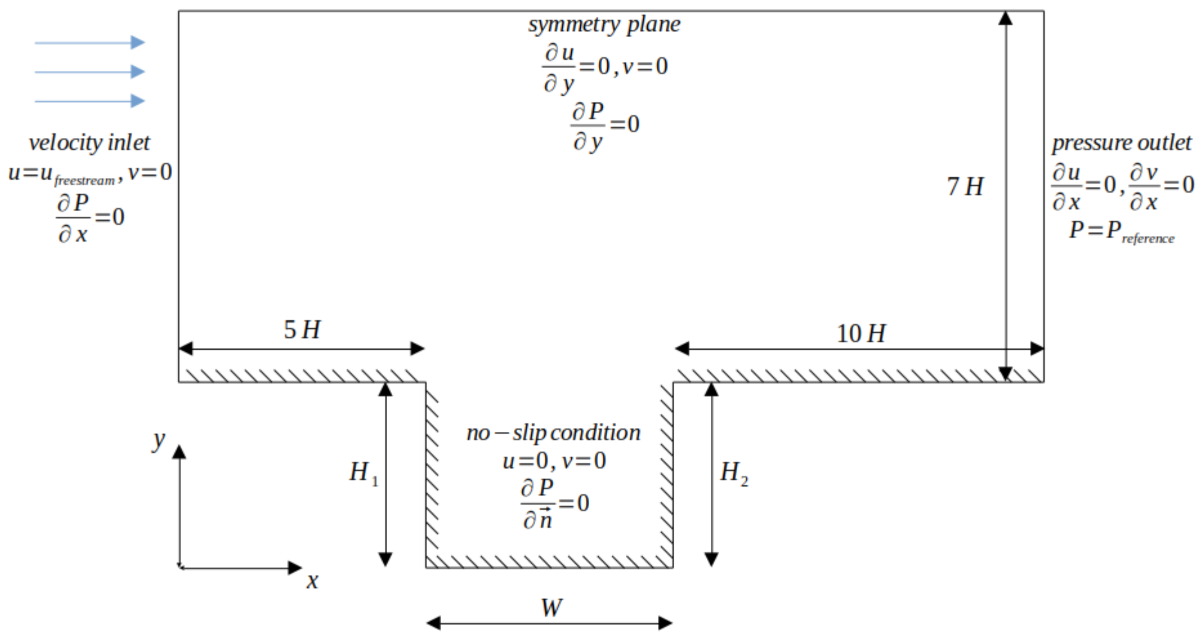

symmetry plane
$\frac{\partial u}{\partial y}=0, v=0$
$\frac{\partial P}{\partial y}=0$
velocity inlet
$u=u_{freestream}, v=0$
$\frac{\partial P}{\partial x}=0$
pressure outlet
$\frac{\partial u}{\partial x}=0, \frac{\partial v}{\partial x}=0$
$P=P_{reference}$
$7H$
$5H$
$10H$
no−slip condition
$u=0, v=0$
$\frac{\partial P}{\partial \vec{n}}=0$
H_1
H_2
y
x
W

Figure 5.2: Boundary Conditions for the 2D Street Canyon Test

5.1.2 Mesh Configuration

The computational grid plays a pivotal role in ensuring the stability and accuracy of discretization schemes employed in CFD solvers. A well-designed grid is essential to minimize errors, particularly in capturing critical physical phenomena like shear layers and vortices. Achieving a fine resolution on the grid is crucial for an accurate representation of these phenomena. This requires careful consideration of grid stretching or compression, especially in regions exhibiting high gradients near boundaries such as walls. To maintain a low truncation error, it is advisable to limit grid stretching, and a recommended maximum expansion ratio between two consecutive cells in high-gradient regions is typically suggested to be below $1.3 - 1.5$. Hoffmann and Chiang [44] proposed the following stretching function for the y direction, which will be used for the placement of the first node closest to the wall y_1.

$$y_i = \frac{(2\alpha+\beta)(\frac{\beta+1}{\beta-1})^{\frac{\gamma_i-\alpha}{1-\alpha}} + 2\alpha - \beta}{(2\alpha+1)\left[1+(\frac{\beta+1}{\beta-1})^{\frac{\gamma_i-\alpha}{1-\alpha}}\right]} \tag{5.1}$$

where:

$\gamma_i = (i-1)\Delta y$

The rest of the refinement zone is defined by a geometric series, according to which, the cell center is positioned at:

$$y_i = G(1 - G^i) + \frac{y_H}{2} G^i \tag{5.2}$$

where:
G: the growth factor of the geometric progression; estimated at ~ 1.1
y_H: the height of the first cell, closest to the wall (calculated by Hoffmann equation)

The total thickness of the refinement zone is:

$$y_T = \sum_{k=1}^{N-1} y_H G^k = y_H \left(\frac{1 - G^N}{1 - G} \right) \tag{5.3}$$

where:
N: the total number of refinement layers

The C++ solver requires user-defined values for the cell size parameters, Δx and Δy, which dictate the grid resolution in regions outside the refinement layers. In Figure 5.3, an example mesh for cases $H_1/H_2 = 1, H_1/W = 1$ with $\Delta x = \Delta y = 0.0375$ is presented. It is important to note that the top boundary, outlet, and inlet sections do not undergo any refinement treatment, in contrast to the dense mesh employed atop the walls and within the confines of the street canyon. This strategic differentiation in mesh resolution enhances computational efficiency and accuracy, concentrating computational resources where they are most needed while maintaining a coarser grid in less critical regions.

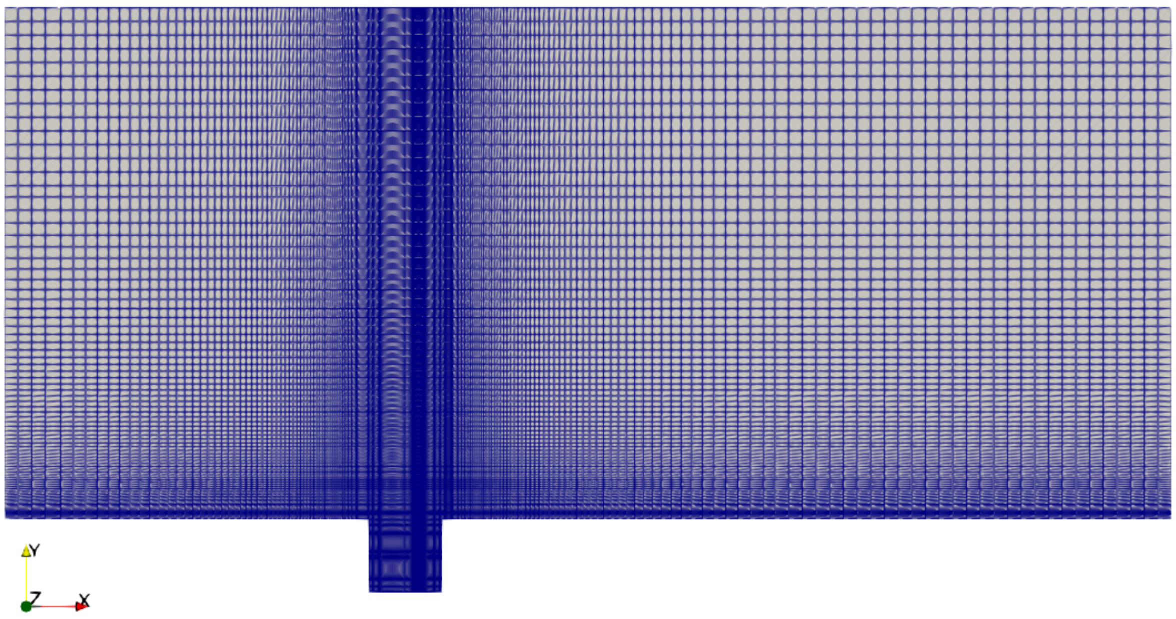


Figure 5.3: Grid: $\Delta x = \Delta y = 0.0375\ m$ for $H_1/H_2 = 1, H_1/W = 1$

5.2 Numerical Solution of Fast Fluid Dynamics Algorithm

5.2.1 Case: Allegrini Experiment

The primary objective of the street canyon case is to reproduce the findings documented in the experimental study conducted by Allegrini et al. [45], wherein a street canyon with a height ratio of $H_1/H_2 = 1$ for various Reynolds numbers and heating scenarios was investigated. The experiments were carried out in a wind tunnel and the measurements were extracted with a PIV method for the model in Figure 5.4. It is important to note that this thesis specifically focuses on the isothermal case, where there is an absence of buoyancy or turbulence induced by temperature differentials.

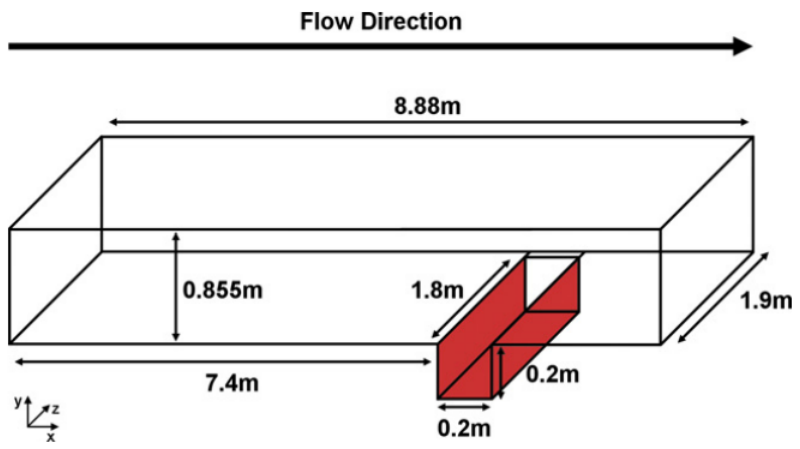


Figure 5.4: Dimensions of Wind Tunnel Model for Allegrini Experiment [45]

The employed solver utilizes the back & forth sweep interpolation method and incorporates two different turbulence models, as described in Chapter 2. The tested case is for Reynolds number $Re = 9000$, which for a street canyon of height $H = 0.2\,m$, corresponds to a freestream velocity $u_{freestream} = 0.68 m/s$. To ensure a detailed capture of the physics involved, a timestep of $\Delta t = 0.001\ sec$ was chosen based on relevant literature.

The velocity profiles documented in Allegrini et al.'s work [45] were extracted to a set of points using an online plotting tool and are presented in Figures 5.5 and 5.6. These points will serve as the reference values for later comparisons and analyses. It's important to note that these data incorporate not only the experimental errors but also any inaccuracies introduced by the plotting tool itself.

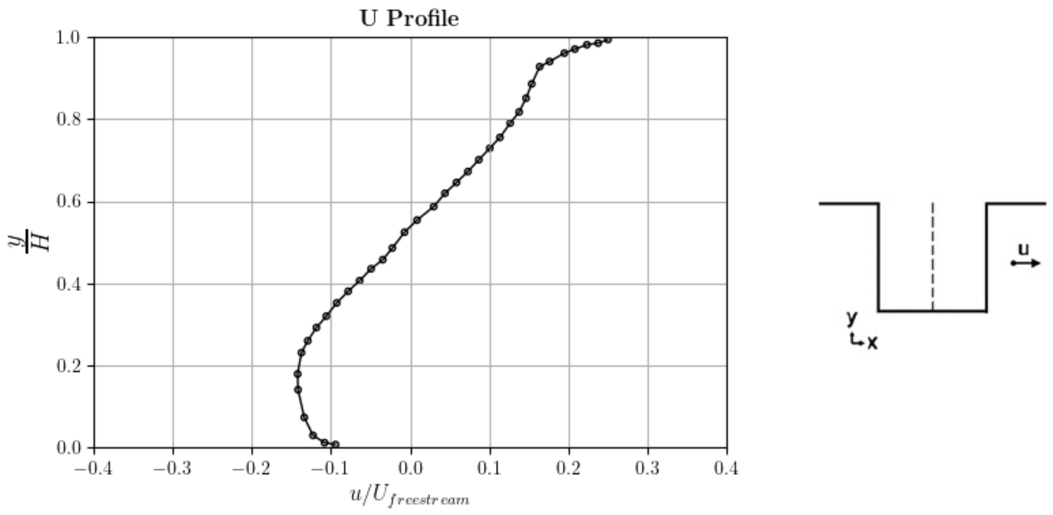


Figure 5.5: Normalized Horizontal Velocity on the Horizontal Centerline of the Street Canyon for $Re = 9000$ Derived from Allegrini's Experiment [42]

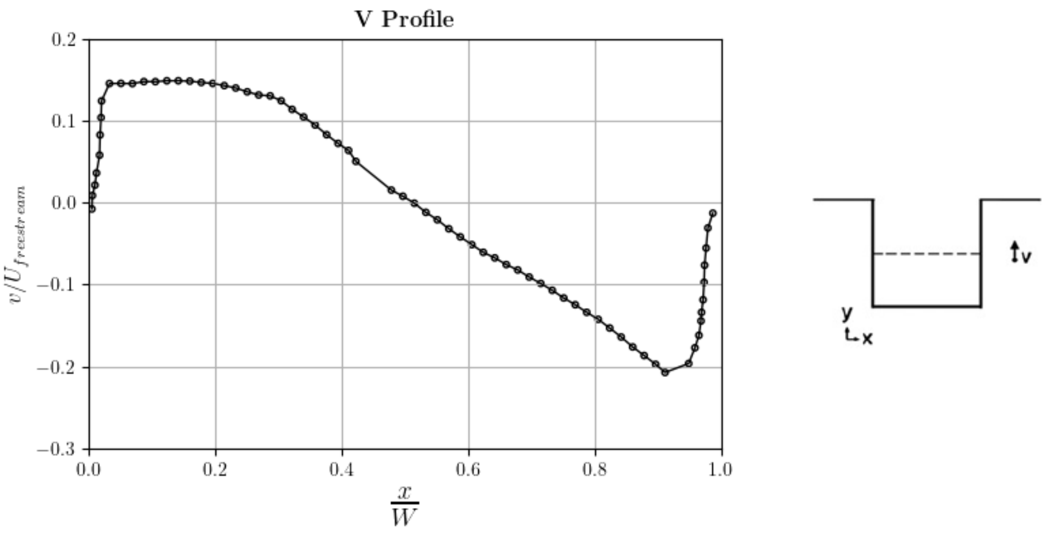


Figure 5.6: Normalized Vertical Velocity on the Vertical Centerline of the Street Canyon for $Re = 9000$ Derived from Allegrini's Experiment [42]

The velocity profiles and accompanying streamlines exhibit similarities to those observed in the lid-driven case, investigated in Chapter 4. However, rather than being driven by a moving lid, the motion within the canyon is induced by the shear force exerted by the airflow above it. This airflow, aimed at simulating an urban landscape, introduces turbulence, rendering the flow non-laminar, unlike the laminar conditions in the cavity problem. Moreover, considering that the reference data stem from real-life experiments, gravitational acceleration is also factored in, albeit with minimal influence on the overall flow dynamics.

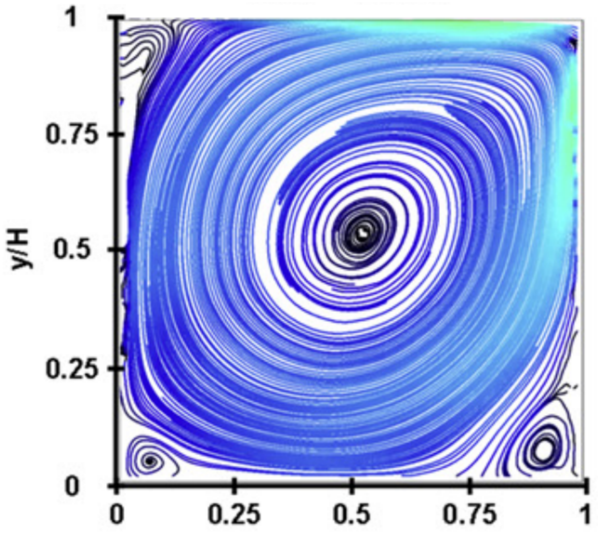


Figure 5.7: Velocity Streamlines for Isothermal Street Canyon Case and $Re = 9000$ Derived from Allegrini's Experiment [42]

5.2.2 Turbulence Model $v_t = 100v$

In this section, the efficiency of the turbulence model $\nu_t = 100\nu$ is examined. Given that the turbulent viscosity remains constant irrespective of spatial or temporal constraints, the computational cost of this particular model is negligible compared to the laminar case, which is its primary advantage.

The augmented viscosity utilized in the simulation serves as a stabilizing factor, akin to the artificial viscosity technique. This augmentation effectively mitigates high-frequency oscillations and averts numerical instabilities. Consequently, the solver is able to achieve steady state convergence in $\sim 51\, sec$ of simulation time in approximately $53\, min$. The max recorded CFL number was 1.107.

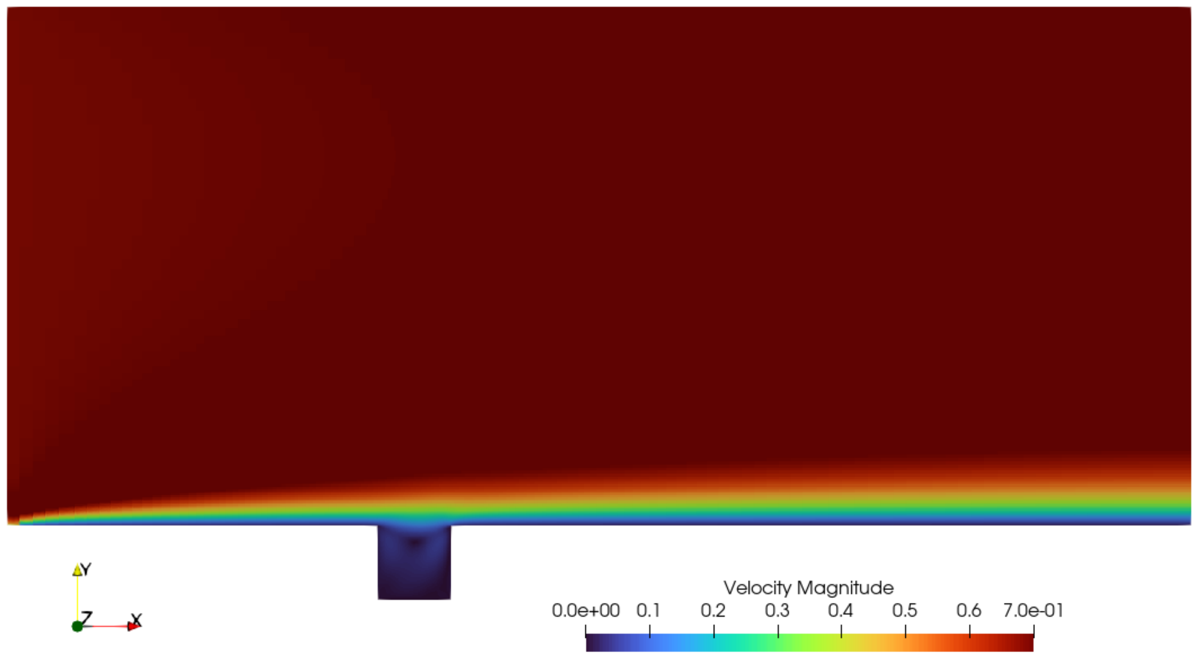


Figure 5.8: Velocity Contour of Simulation Domain for Turbulence Model
$v_t = 100v$, $Re = 9000$, $\Delta t = 0.001\, sec$

The multiplication of viscosity by a factor of 100 has a dual effect. While it helps in stabilizing the flow to achieve a steady solution, it significantly alters the Reynolds number of the case. The model essentially, simulates a flow with a Reynolds number at approximately ~ 90, where viscous forces dominate, resembling laminar flow characteristics. This is evidenced by the boundary layer created, along the flow in the domain, as illustrated in Figure 5.8. At lower Reynolds numbers, such as in this case, the boundary layer remains laminar, and the streamwise velocity undergoes uniform changes as one moves away from the wall. Additionally, the thickness of the boundary layer appears to be larger than expected for the conditions being examined.

In Figure 5.9, the velocity streamlines are depicted and a notable deviation from the findings of Allegrini et al. [45] is observed. In their study, the primary vortex is situated near the geometric center of the cavity. In contrast, the FFD solver manages to align the vortex with the vertical axis but struggles to do so with the horizontal axis. This discrepancy primarily arises from the unresolved lower bottom eddies by the solver. One would anticipate the presence of three eddies—two at the bottom of the canyon and one in the top left corner. Nevertheless, in this instance, only the lower-right vortex manifests, albeit with significant disparities in size compared to the experimental setup. Additionally, there's a noticeable difference in the velocity magnitude, with the FFD flow exhibiting substantially lower velocities. This discrepancy in velocity magnitude could be attributed to the excessive damping imposed by the turbulence model and the semi-Lagrangian scheme utilized in the simulation.

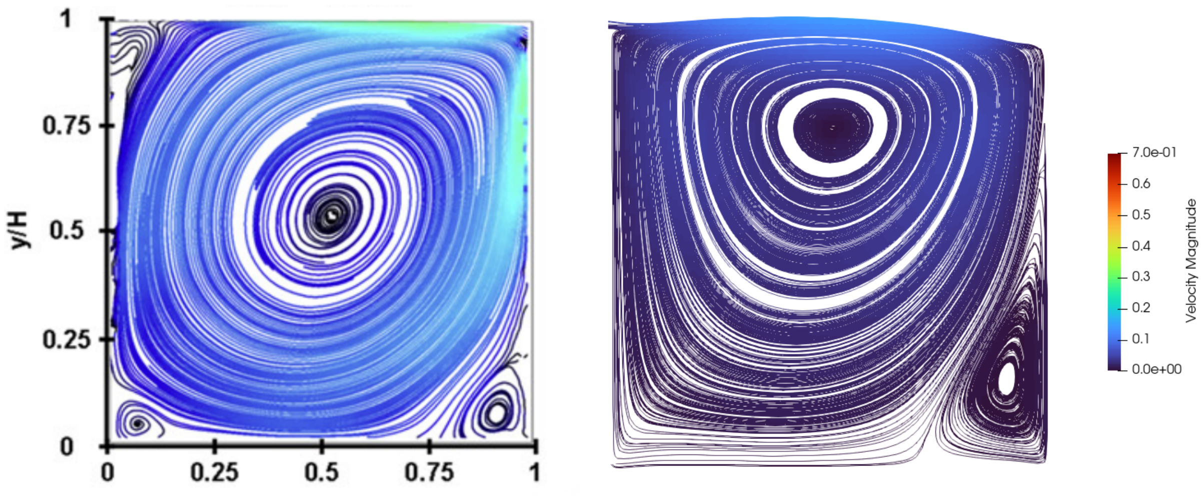


Figure 5.9: Velocity Streamlines for Turbulence Model $v_t = 100v$, $Re = 9000$, $\Delta t = 0.001\ sec$

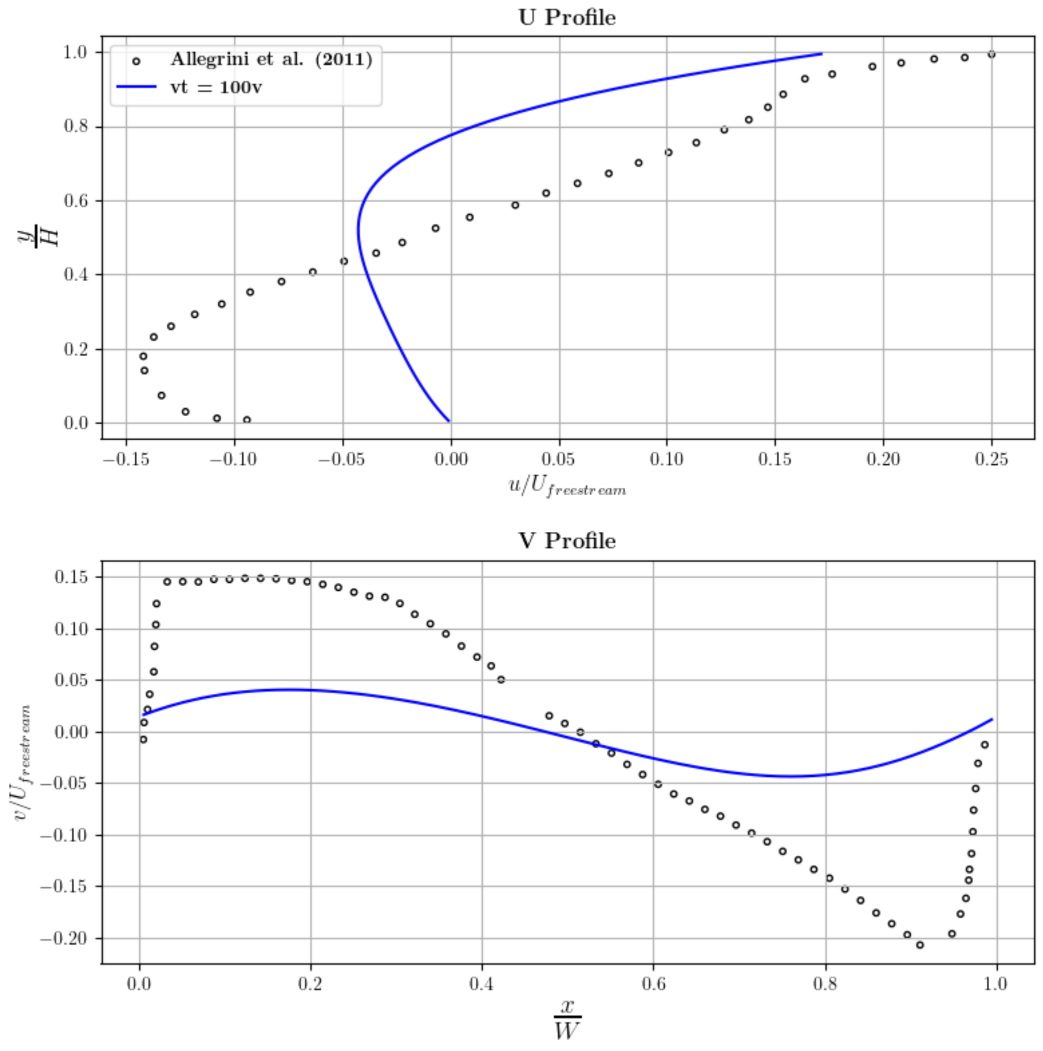


Figure 5.10: Velocity Profiles for Turbulence Model $v_t = 100v$, $Re = 9000$, $\Delta t = 0.001\ sec$

This is further evidenced by the velocity profiles in Figure 5.10, where the velocities are observed to be $2 - 3\times$ smaller than the maximum velocity recorded in Allegrini's [45] work. Moreover, the displacement of the primary vortex is also noticeable in the u profile by the peak value's placement. Although the v profile captures the nature of the vortex, there is a significant difference in magnitude compared to the experimental results. Lastly, the treatment of the walls appears different, with the FFD capturing smaller velocity gradients near the walls.

5.2.3 Zero Equation Turbulence Model

Contrary to the simplified $\nu_t = 100\nu$ turbulence model, which assumes a constant turbulent viscosity throughout the flow domain, the zero equation model offers a more sophisticated approach by considering the actual flow dynamics and behavior. This model, also known as the algebraic turbulence model, is termed "zero equation" because it directly solves for the turbulent viscosity without explicitly solving any additional transport equations. The equation is reiterated here for clarity.

$$\nu_t = 0.03874\ L\ U \tag{5.4}$$

where:
L: a length scale
U: the local mean velocity

During the model testing phase, a series of simulations were conducted across a range of refinement levels in mesh resolution and timesteps. However, a notable challenge emerged as the model failed to attain a steady-state solution, primarily due to pronounced oscillations in the flow above the street canyon, particularly within the shearing zone. The final results cover a simulation time of up to 90 sec and required approximately 102 min to compute. The maximum CFL number recorded during the simulation was 1.26.

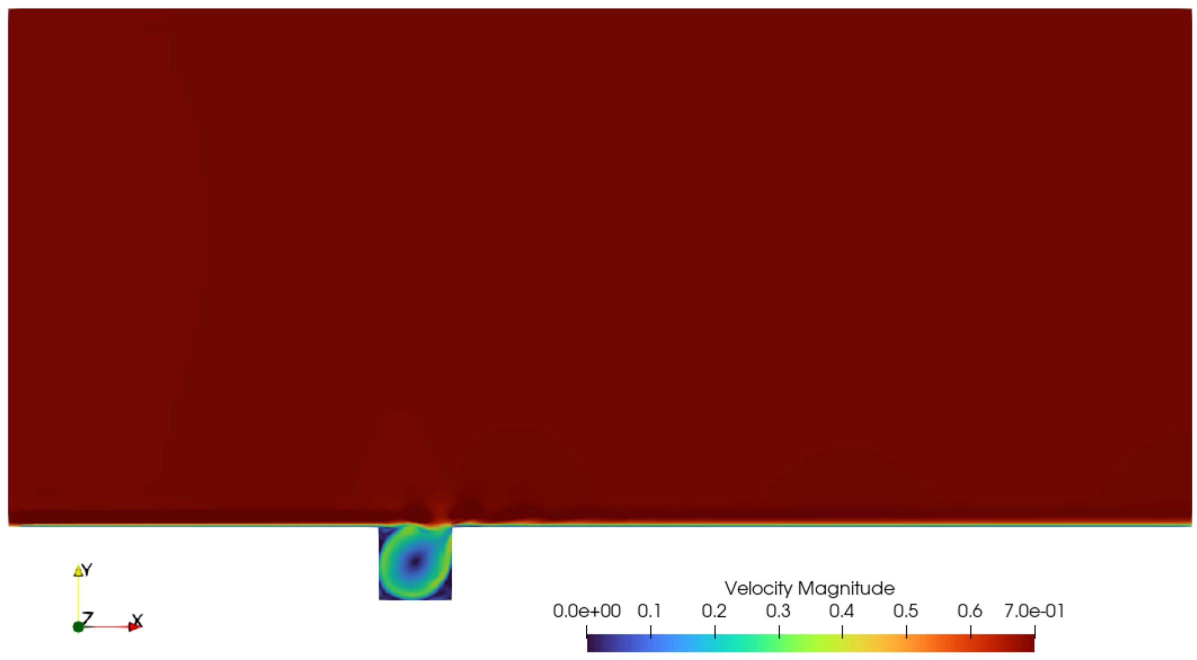


Figure 5.11: Velocity Contour for Zero Equation Turbulence Model, $Re = 9000$, $\Delta t = 0.001\ sec$

In contrast to the previous turbulence model, there's a marked difference in the boundary layer formed on the walls downstream of the canyon. Notably, the boundary layer's height is significantly reduced, primarily attributed to the higher Reynolds value. This reduction highlights a more accurate representation of the flow's dynamics, indicating improved resolution and capturing of turbulent effects.

Significant improvements are also evident in the behavior of the flow inside the street canyon, as shown in the streamlines in Figure 5.12 for the time moment 90 *sec*. The positioning of the primary vortex at the geometric center of the cavity closely resembles the findings of Allegrini et al.'s experiment [45]. Additionally, the model adeptly captures the existence of three additional eddies situated at the corners of the street canyon, mirroring the experimental setup. However, it's important to note that while the general placement of these vortices corresponds to the experiment, there are noticeable differences in their strength and size compared to the observed data.

In Figure 5.13, the velocity streamlines for the last 10 *sec* of the simulation are presented. While the primary vortex remains stable in its position, consistent with Allegrini's [45] results, the three additional vortices exhibit constant fluctuations. Of particular note is the upper-left vortex, situated in the shearing region of the flow, where these fluctuations are most prominent. It is also evident that these fluctuations follow an almost periodic pattern. For example, the velocity streamlines for times 80 *sec*, 83 *sec*, 87 *sec*, 90 *sec* display similar vortex geometries and placements.

Finally, the velocity profiles in Figure 5.14 suggest that the model tends to overestimate the turbulence within the flow, as evidenced by velocity magnitudes that are $2-4\times$ greater than expected. Particularly affected is the profile of the horizontal velocity u, which deviates significantly from the anticipated behavior, especially within the shearing zone. However, contrary to the $\nu_t = 100\nu$ turbulence model, the velocity at the bottom wall of the cavity aligns well with experimental observations, as does the center of the vortex, where $u = 0$. Regarding the profile of the v component of the velocity, although its magnitude appears higher in comparison, it adeptly captures the general behavior of the flow. Specifically, the linear portion of the plot, representing the primary vortex, closely mirrors the corresponding positions observed in the experiments, particularly along the windward wall.

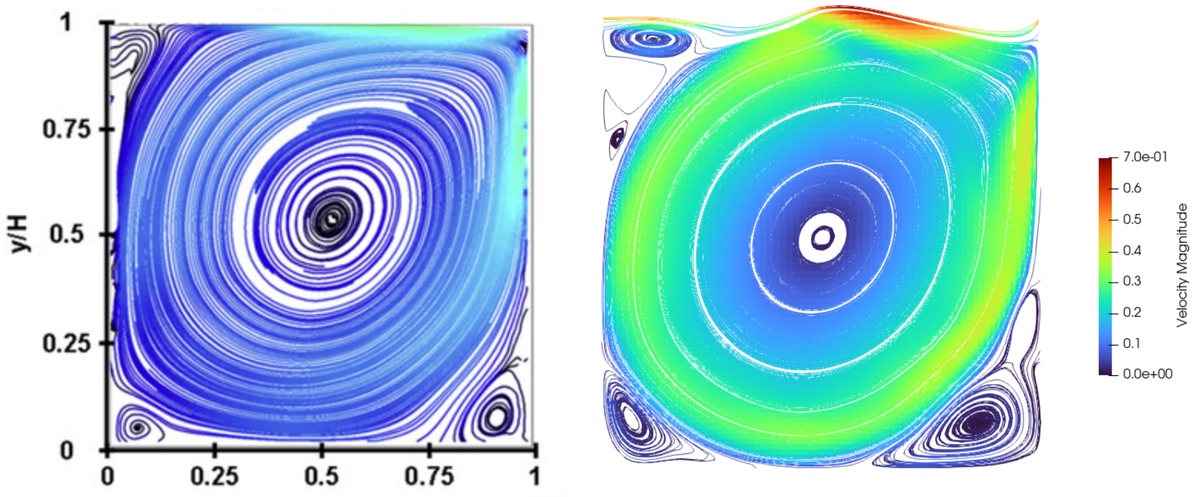


Figure 5.12: Velocity Streamlines for Zero Equation Turbulence Model, $Re = 9000$, $T = 90\ sec$, $\Delta t = 0.001\ sec$

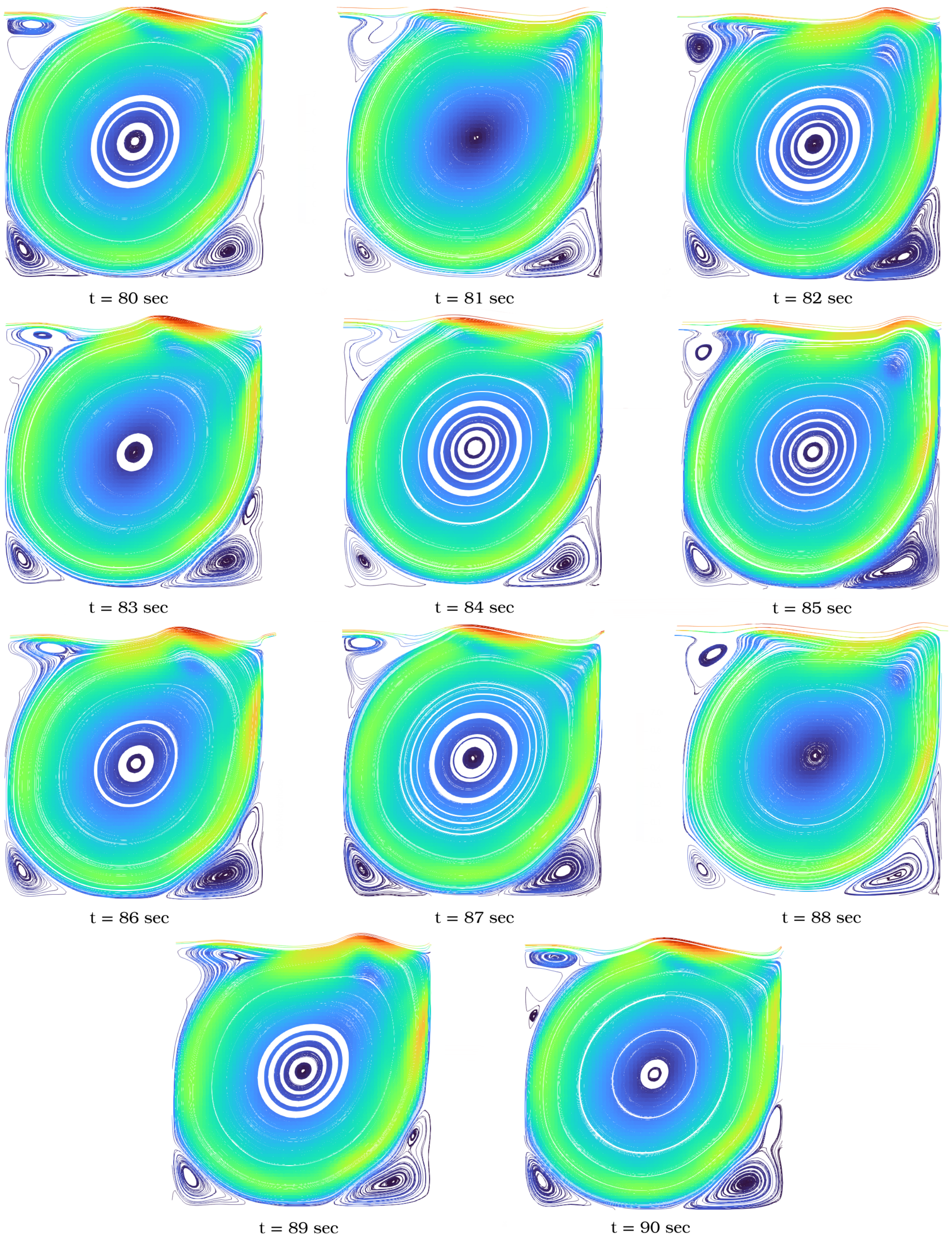


Figure 5.13: Velocity Streamlines for Various Timesteps for Zero Equation Turbulence Model, $Re = 9000$, $\Delta t = 0.001\ sec$

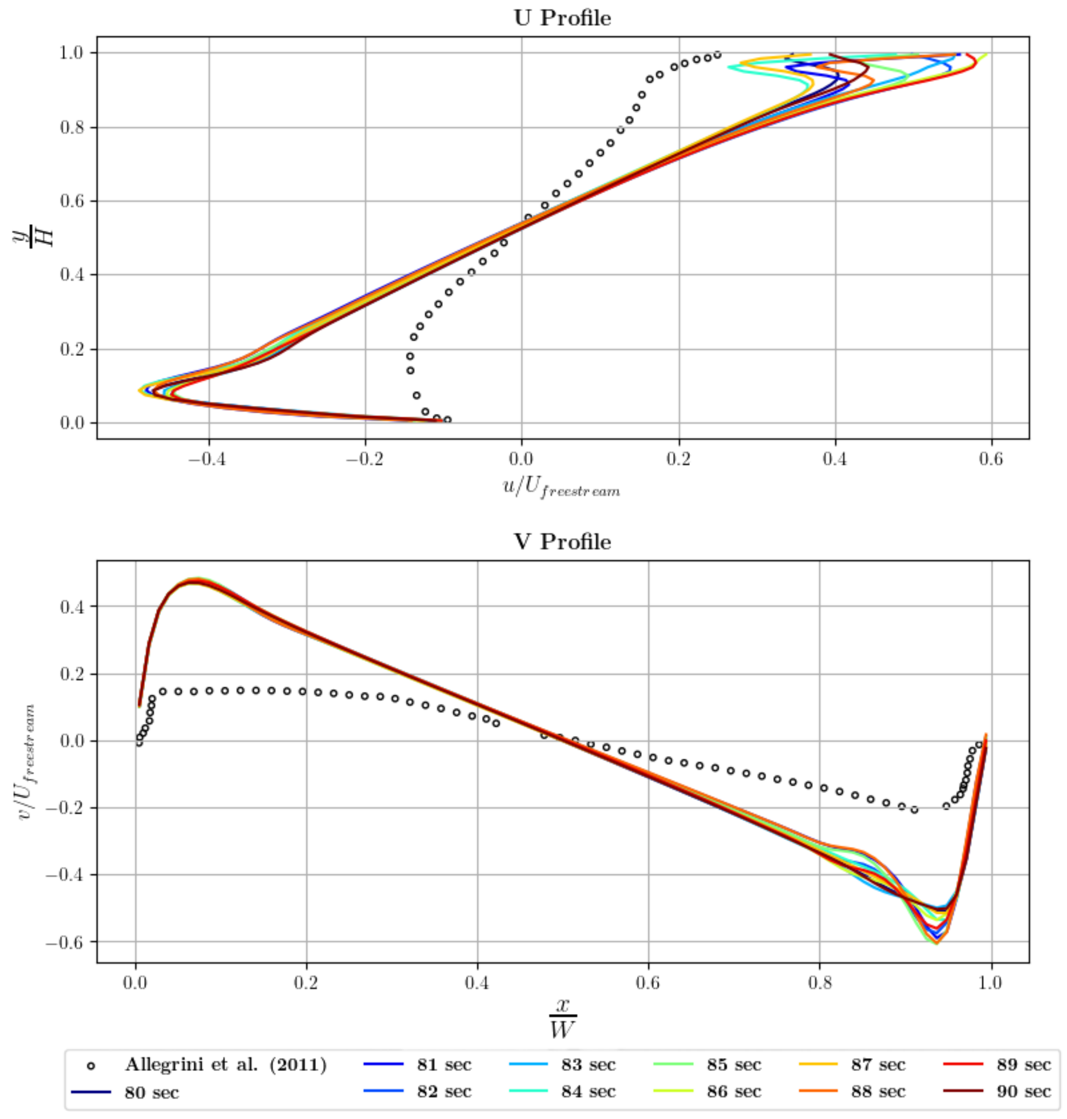


Figure 5.14: Velocity Profiles for Various Timesteps for Zero Equation Turbulence Model, $Re = 9000$, $\Delta t = 0.001$ *sec*

5.3 Performance Analysis

5.3.1 OpenFOAM Simulation

For the OpenFoam simulation, the $k - \epsilon$ turbulence model was utilized, requiring the solution of two additional equations for kinetic energy (k) and its dissipation (ϵ). This specific model is computationally demanding, making it challenging to compare directly with the turbulence model integrated into the FFD algorithm. Nevertheless, the results are presented here to underscore the discrepancy in accuracy between $k - \epsilon$ and the previous turbulence models. The OpenFOAM simulation was performed for 85 *sec*.

Similar to the zero equation model, the OpenFOAM simulation fails to converge to a steady-state solution. This issue becomes evident in Figure 5.15,where the initial residuals oscillate, indicating the inherent unsteadiness of the problem. However, despite this convergence issue, the velocity profiles and streamlines closely resemble those observed in the experimental work conducted by Allegrini et al. [45].

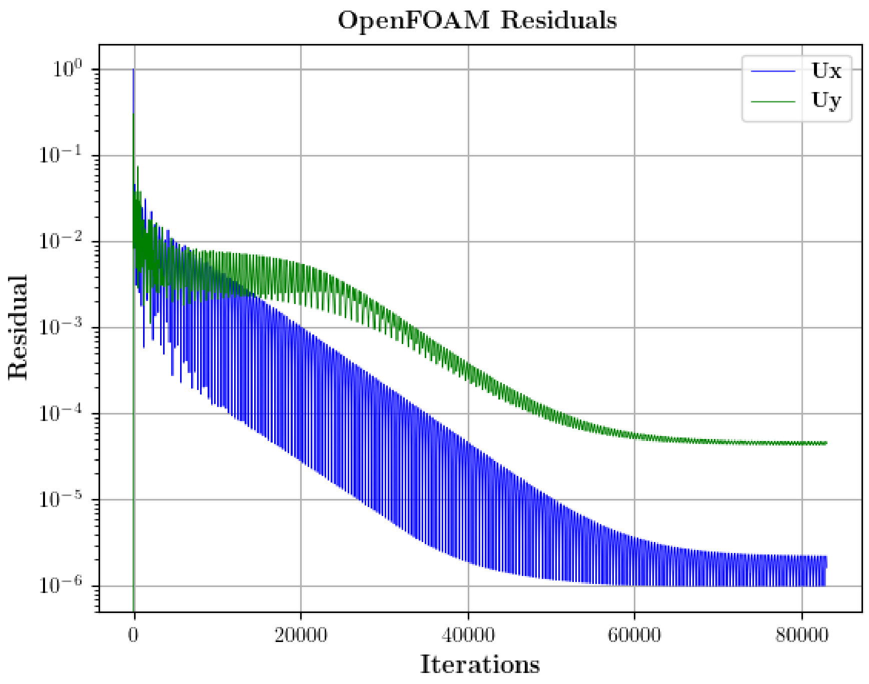


Figure 5.15: Velocity Residuals for OpenFOAM Simulation for Turbulence Model $k-\epsilon$, $Re = 9000$, $\Delta t = 0.001$ *sec*

The velocity streamlines mirror those depicted in Figure 5.7, exhibiting similar characteristics in terms of velocity magnitude and vortex formation. The freestream flow appears detached from the flow within the street canyon, leading to a distinct behavior where the u velocity component becomes nearly parallel in the upper region, akin to the lid-driven cavity benchmark. However, despite the similarities, it's worth noting that even the $k-\epsilon$ model struggles to perfectly match the velocity profiles observed in the actual flow.

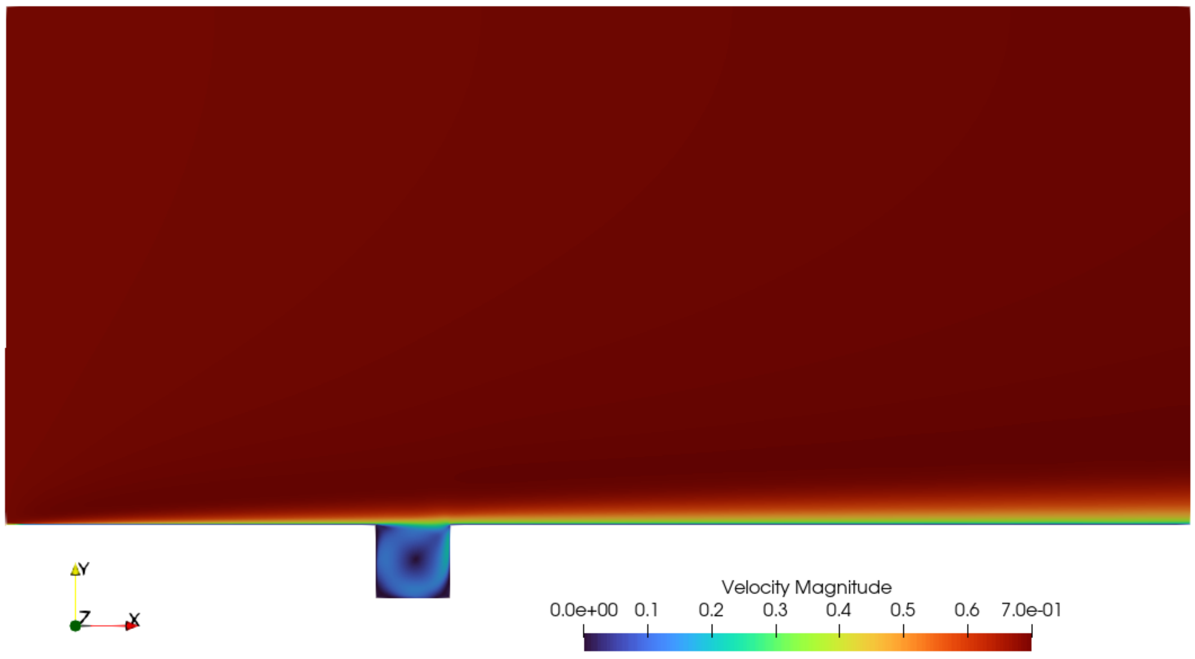


Figure 5.16: Velocity Contour for OpenFOAM Simulation for Turbulence Model $k-\epsilon$, $Re = 9000$, $\Delta t = 0.001$ *sec*

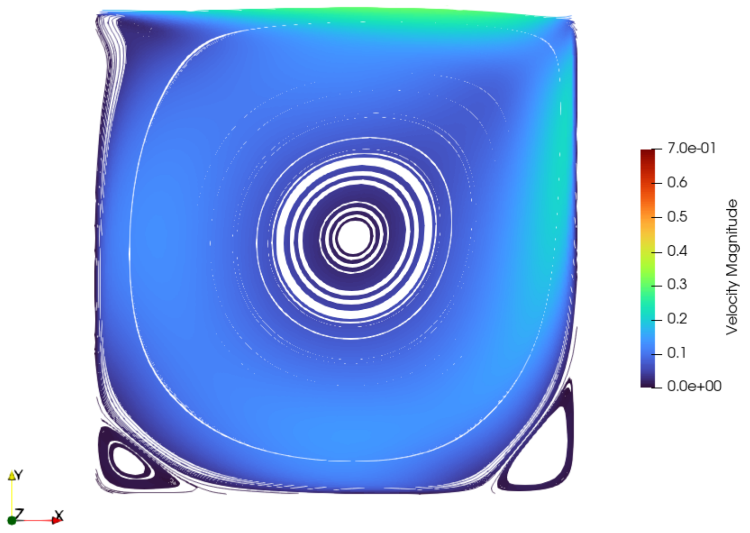


Figure 5.17: Velocity Streamlines for OpenFOAM Simulation for Turbulence Model $k - \epsilon$, $Re = 9000$, $\Delta t = 0.001$ sec

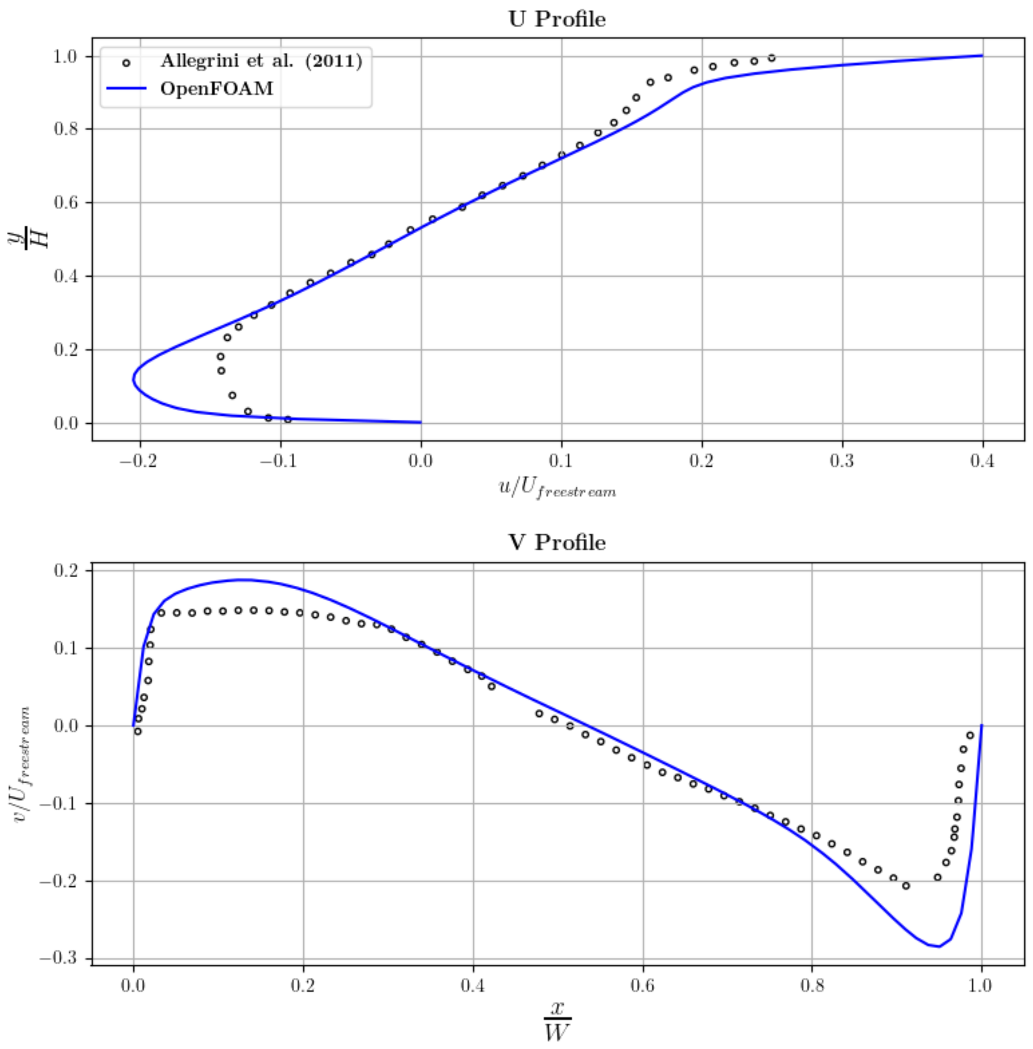


Figure 5.18: Velocity Profiles for OpenFOAM Simulation for Turbulence Model $k - \epsilon$, $Re = 9000$, $\Delta t = 0.001$ sec

5.3.2 Performance Evaluation

To effectively compare the performance of each turbulence model, especially considering the unsteadiness of the simulation, the analysis focuses on the initial 1000 timesteps, corresponding to the first 1 *sec* of the simulation. It is worth noting again that the FFD algorithm utilizes the back & forth sweep interpolation method, which was introduced in Chapter 2.

In Table 5.1, the clock times for each turbulence model are recorded under identical grid resolution and simulation parameters. It is observed that there is no significant difference in time between the laminar case and the $\nu_t = 100\nu$ turbulence model, as expected. However, when employing the zero-equation model, there is an increase of 8.8% in computational cost. This additional cost arises from the extra calculations needed to interpolate the velocity at each mesh node for the velocity magnitude required by the model. Similarly, OpenFOAM's $k-\epsilon$ model exhibits the highest computational cost among all models. However, it is noteworthy that the computational costs of the two latter models are surprisingly close in terms of clock time, which may require further investigation.

Table 5.1: Clock Time for Simulations with Different Turbulence Models for the First 1000 Timesteps and $Re = 9000$, $\Delta x_{max} = 0.035\,m$, 28321 Cells, $\Delta t = 0.001\,sec$

Turbulence Model Case	Clock Time[sec]
FFD Laminar Case	61.01
$\nu_t = 100\nu$	60.07
Zero Equation Model	66.37
OpenFOAM $k-\epsilon$	67.48

Table 5.2: Computational Cost of Each of FFD's Algorithm Step for Simulations for the First 1000 Timesteps and $Re = 9000$, $\Delta x_{max} = 0.035\,m$, 28321 Cells, $\Delta t = 0.001\,sec$

Turbulence Model Case	Diffusion [sec]	Convection [sec]	Pressure Correction [sec]	Pressure Gradient [sec]	Clock Time [sec]
FFD Laminar Case	2.473	57.396	1.651	0.278	63.02
	3.92 %	91.08 %	2.62 %	0.44 %	–
$\nu_t = 100\nu$	3.162	55.864	1.619	0.274	62.1
	5.09 %	89.96 %	2.61 %	0.44 %	–
Zero Equation Model	8.187	56.731	1.668	0.297	68.13
	12.02 %	83.27 %	2.45 %	0.44 %	–

In Chapter 4, where the cavity benchmark was investigated, the FFD solver was found to be approximately 30% faster than OpenFOAM. However, this trend does not seem to hold true in the current analysis. Upon examining each of FFD's individual steps in Table 5.2, it was found that the convection step demonstrated a dramatic increase in computational cost, requiring approximately 87% of the total computational cost. This is in stark contrast to the lid-driven cavity application, where the convection step accounted for only about 55% of the total computational cost. Such a significant difference indicates that the coding of the convection library may be suboptimal, as the performance of the code deteriorates significantly as the number of nodes increases.

To conduct a fair comparison between the FFD and OpenFOAM solvers, the turbulence models are omitted, and the laminar case is simulated with identical parameters and grid resolutions. Doubling the number of nodes from the previous comparison, it becomes apparent that the computational cost of the FFD algorithm not only matches but exceeds that of OpenFOAM by almost twofold. Notably, the convection step accounts for approximately 95% of the total clock time. This discrepancy underscores a significant performance issue with the current FFD solver, particularly when dealing with larger grid sizes.

Table 5.3: Clock Time for Laminar Case Simulations for the First 1000 Timesteps and $Re = 9000$, Uniform Grid: $\Delta x = 0.02\,m$, 46166 Cells, $\Delta t = 0.001\,sec$

Laminar Case's Solver	Clock Time[sec]
FFD	95.81
OpenFOAM	41.66

5.4 Nudging

The implementation of the nudging method aims to assess the potential improvement in the physical representation of the flow by assimilating Allegrini's [45] experimental data. In Chapter 4, it was established that nudging effectively compensated for inaccuracies in FFD and expedited convergence towards a steady-state solution, and employing a greater number of observation points during the nudging process resulted in accelerated convergence.

In the street canyon application, both velocity profiles from Allegrini's [45] experiments are assimilated for each turbulence model to assess the impact of nudging on the solution. However, to mitigate computational costs, the linear interpolation scheme is employed during the semi-Lagrangian method in the convection equation, which introduces an additional level of numerical inaccuracies. Despite the inherent limitations of the linear interpolation scheme, the RMS error values presented in Table 5.4 are comparable to those achieved using the high-order back & forth sweep method.

Nudging is applied to both turbulence models discussed in the preceding section. However, for the zero-equation model, which failed to attain a steady-state solution and is computationally demanding, nudging is implemented primarily for the sake of inclusivity. It's worth noting that nudging relies on the experimental data provided by Allegrini et al.[45], extracted from Figures 5.5 and 5.6, comprising approximately of 30 observation points for each velocity profile.

For the $\nu_t = 100\nu$ turbulence model, a few values of parameter α are investigated for the cross sectional nudging. It is crucial to emphasize that when comparing the nudging source term to the source term of the diffusion equation, their magnitudes are of similar size, with the nudging source term being consistently smaller than the source term. This observation indicates that nudging neither eliminates the original diffusion equation nor imposes boundary conditions akin to a identity equation.

In Figure 5.19, the nudged equations exhibit a significant improvement compared to the original simulation. As expected, increasing the parameter α yields better results in velocity profiles, as the nudged source term gradually outweighs the diffusion source term. However, neither the simulation time nor the RMSE demonstrate a significant improvement proportional to the increase in α, as shown in Table 5.4. What remains significant is the reduction in total error by approximately $3-6\times$, coupled with achieving convergence in fewer iterations.

Table 5.4: Simulation Time and RMSE for Nudged Cases for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$

Nudging Parameter α	Simulation Time[sec]	RMSE
500	43.976	0.024477
1000	41.404	0.019207
1250	38.62	0.017949
2500	37.253	0.015143
Linear Case ($\alpha = 0$)	58.960	0.091278
Back & Forth Sweep Case ($\alpha = 0$)	51.051	0.090657

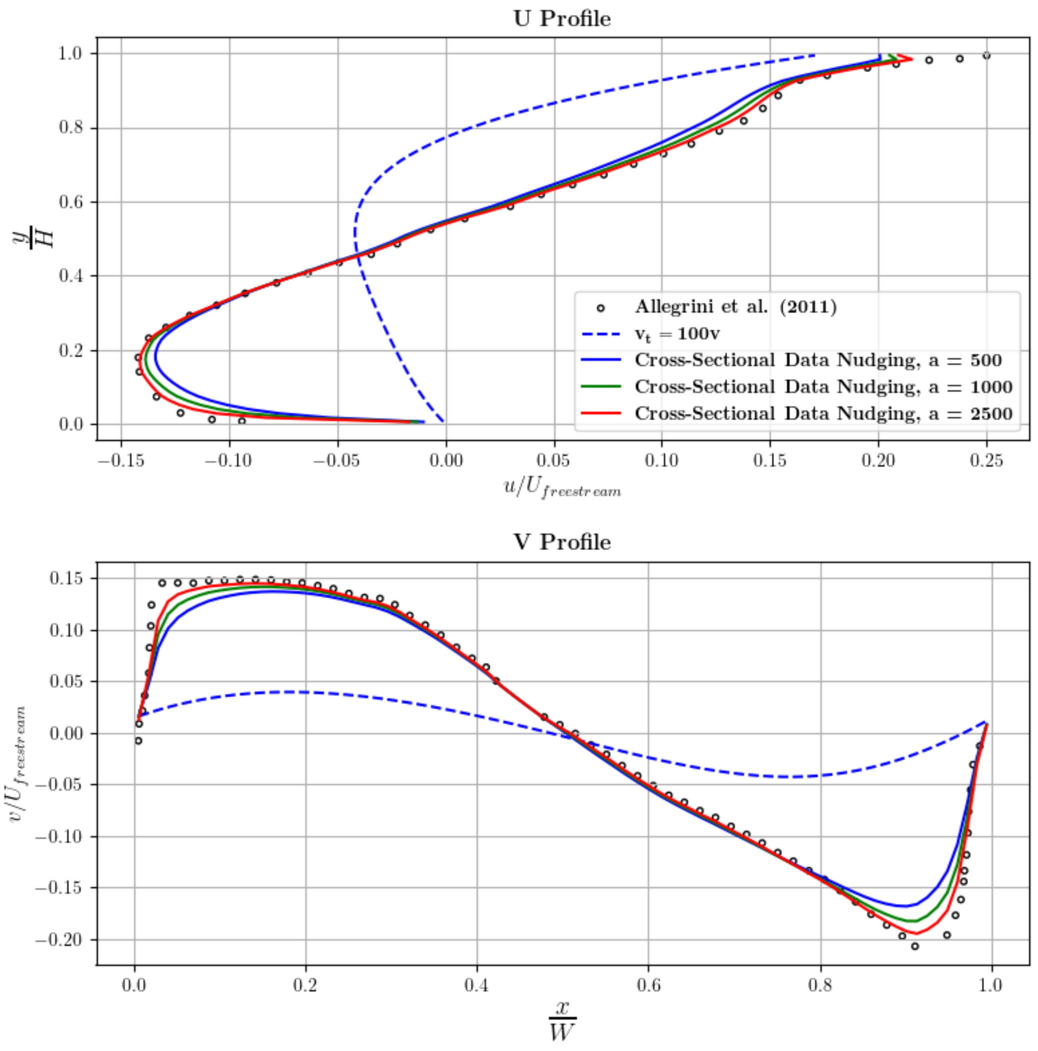


Figure 5.19: Velocity Profiles for Cross-Sectional Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$

Despite achieving good agreement in the velocity profiles with the experimental data, the velocity streamlines highlight a weakness in the model. A comparison with Allegrini's [45] velocity contour in Figure 5.7 reveals that the nudged simulations fail to capture the lower left vortex, a similar limitation observed in the non-nudged FFD simulation utilizing the same turbulence model. This discrepancy could be attributed to the concentration of observation points along the centerlines of the street canyon, resulting in a lack of additional information for locations farther away. Secondly, it's plausible that the nudging process isn't fully capable of bypassing the underlying physics of the flow, potentially being overruled by the effects of the current turbulence model.

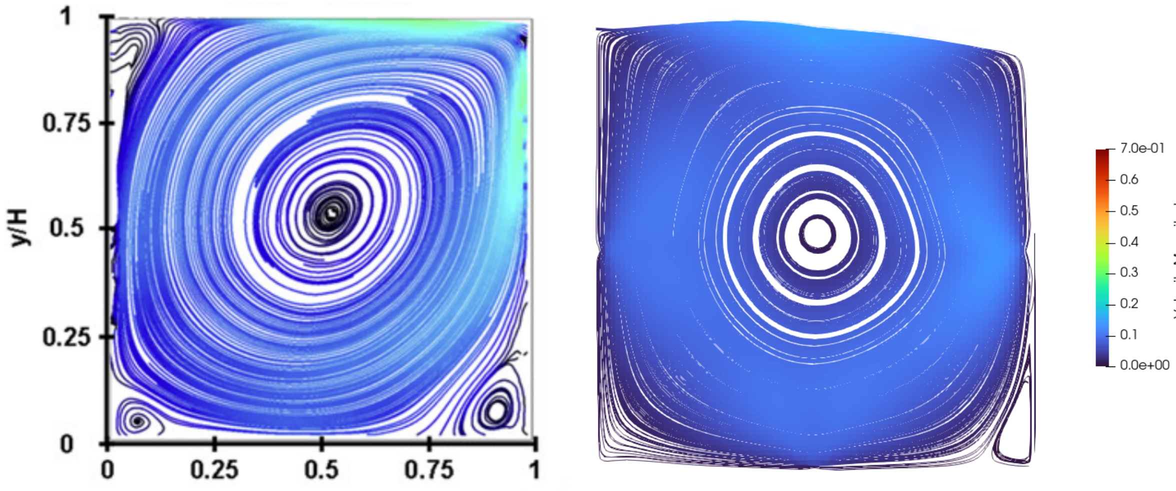


Figure 5.20: Velocity Streamlines for Cross-Sectional Nudging Simulation for Turbulence Model $v_t = 100v$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

Furthermore, restricting the nudging to either the u or v profile yields the anticipated outcomes, with the corresponding velocity profile exhibiting the most significant improvement. However, this doesn't imply that the other non-nudged velocity component remains unaffected and does not show improvement. Table 5.5 highlights this, where the RMSE compared to the experimental data is very small for both velocity components, but notably smaller for the nudged profile.

Table 5.5: Simulation Time and RMSE Compared to Allegrini's Data [45] for Nudged Cases and for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

Case	Simulation Time[sec]	RMSE	
		U-Profile	V-Profile
Vertical Centerline Nudging (Nudged u)	38.496	0.018299	0.056315
Horizontal Centerline Nudging (Nudged v)	36.967	0.038052	0.013831
Cross-Sectional Nudging	37.253	0.016751	0.013536
Linear Case ($\alpha = 0$)	58.960	0.088309	0.094248
Back & Forth Sweep Case ($\alpha = 0$)	51.051	0.087422	0.093893

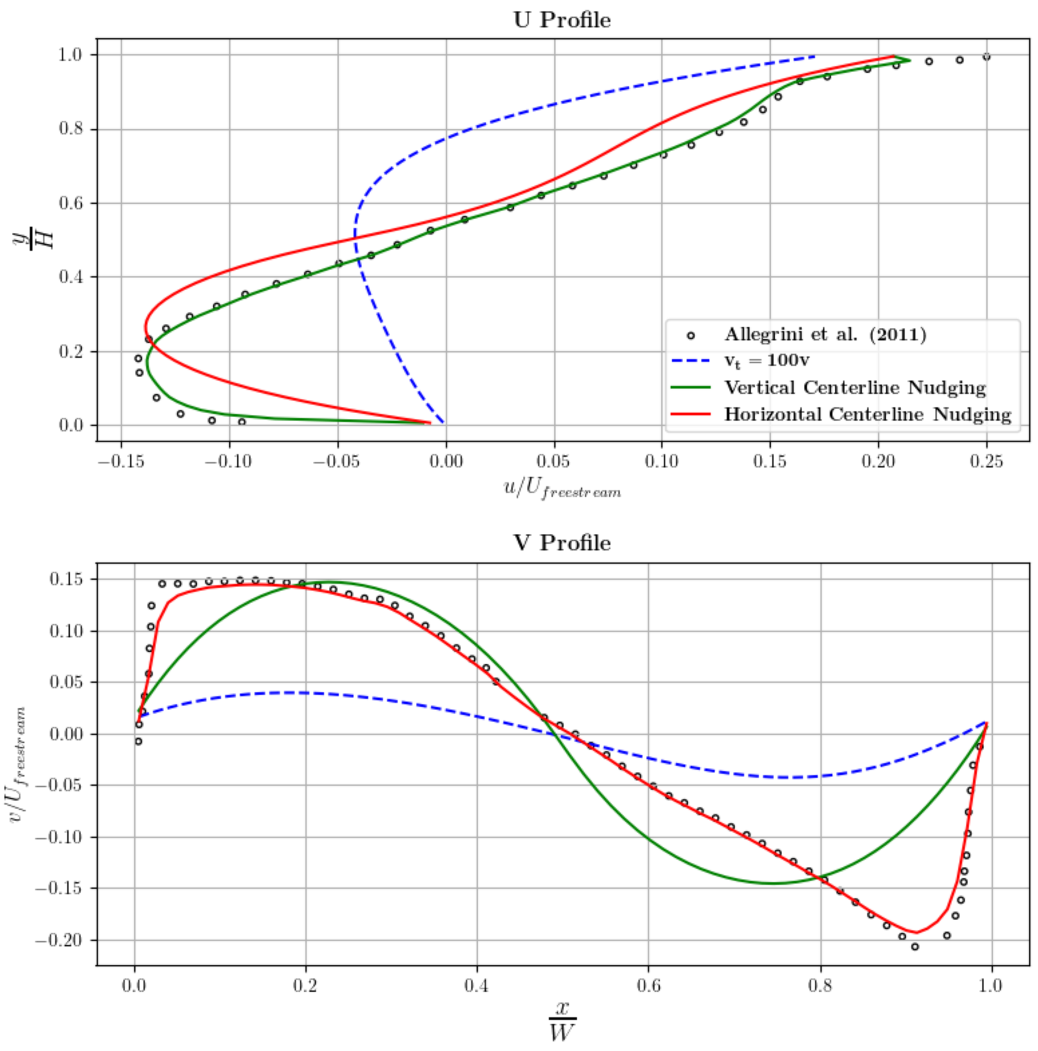


Figure 5.21: Velocity Profiles for Vertical (a) and Horizontal (b) Centerline Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

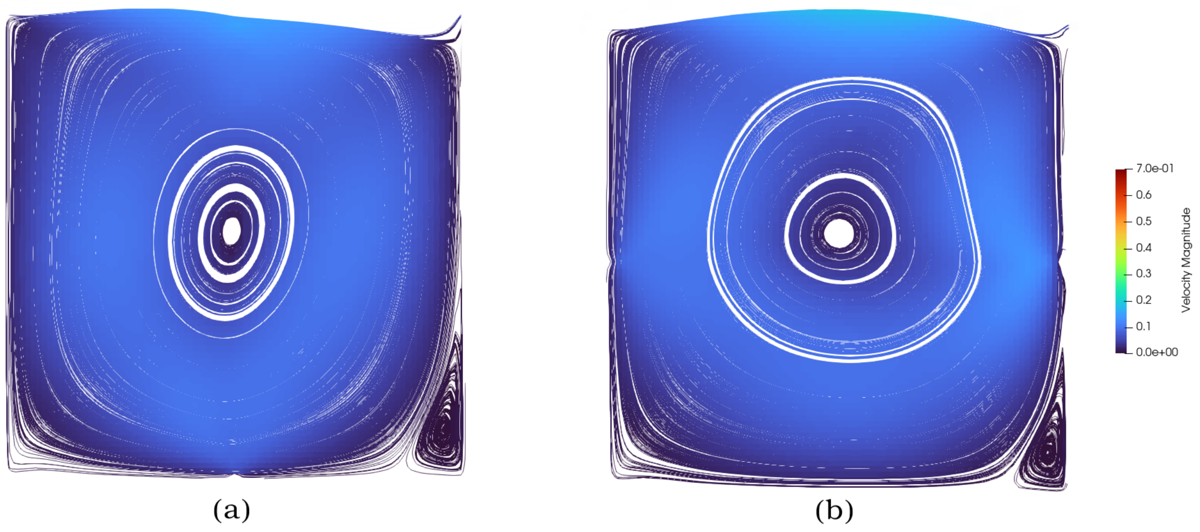


Figure 5.22: Velocity Streamlines for Vertical (a) and Horizontal (b) Centerline Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

By limiting the number of nudging locations to 9 points per velocity profile, convergence is achieved at approximately the same simulation time, as the cross-sectional nudging case. However, the RMSE increases, indicating that the velocity profiles start to deviate from the experimental data. Similar to the lid-driven cavity, the profiles exhibit unnatural spikes in the observation points vicinity and struggle to match the original experimental values. It is reasonable to assume that further reducing the number of observation points would exacerbate the discrepancy between simulation and experiment.

Table 5.6: Simulation Time and RMSE Compared to Allegrini's Data [45] for 9-Point Nudging and Cross-Sectional for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

Case	Simulation Time[sec]	RMSE	
		U-Profile	V-Profile
9-Point Nudging	37.07	0.022860	0.021672
Cross-Sectional Nudging	37.253	0.016751	0.013536
Linear Case ($\alpha = 0$)	58.960	0.088309	0.094248
Back & Forth Sweep Case ($\alpha = 0$)	51.051	0.087422	0.093893

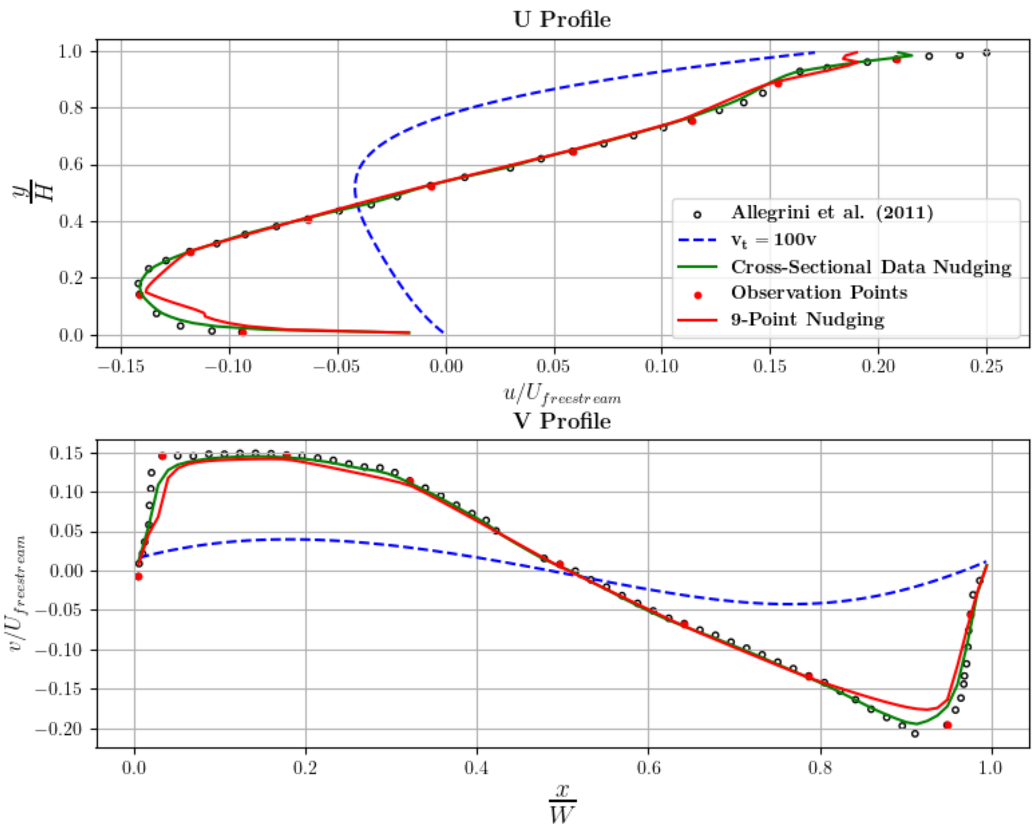


Figure 5.23: Velocity Profiles for 9-Point Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

Utilizing 3 points for each velocity profile, nudging is performed at two distinct locations. Table 5.7 illustrates the RMS errors in comparison to those reported by Allegrini et al. [45]. Notably, while the error reduction is evident when compared to the linear and back-and-forth sweep interpolation methods, the extent of improvement is somewhat less pronounced in contrast to cross-sectional nudging utilizing all observation points.

Table 5.7: Simulation Time and RMSE Compared to Allegrini's Data [45] for 3-Point Nudging for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001$ sec, $\alpha = 2500$

Case	Simulation Time[sec]	RMSE	
		U-Profile	V-Profile
Cross-Sectional Nudging	37.253	0.016751	0.013536
Case (a)	47.529	0.058392	0.082697
Case (b)	50.779	0.075421	0.084974
Linear Case ($\alpha = 0$)	58.960	0.088309	0.094248
Back & Forth Sweep Case ($\alpha = 0$)	51.051	0.087422	0.093893

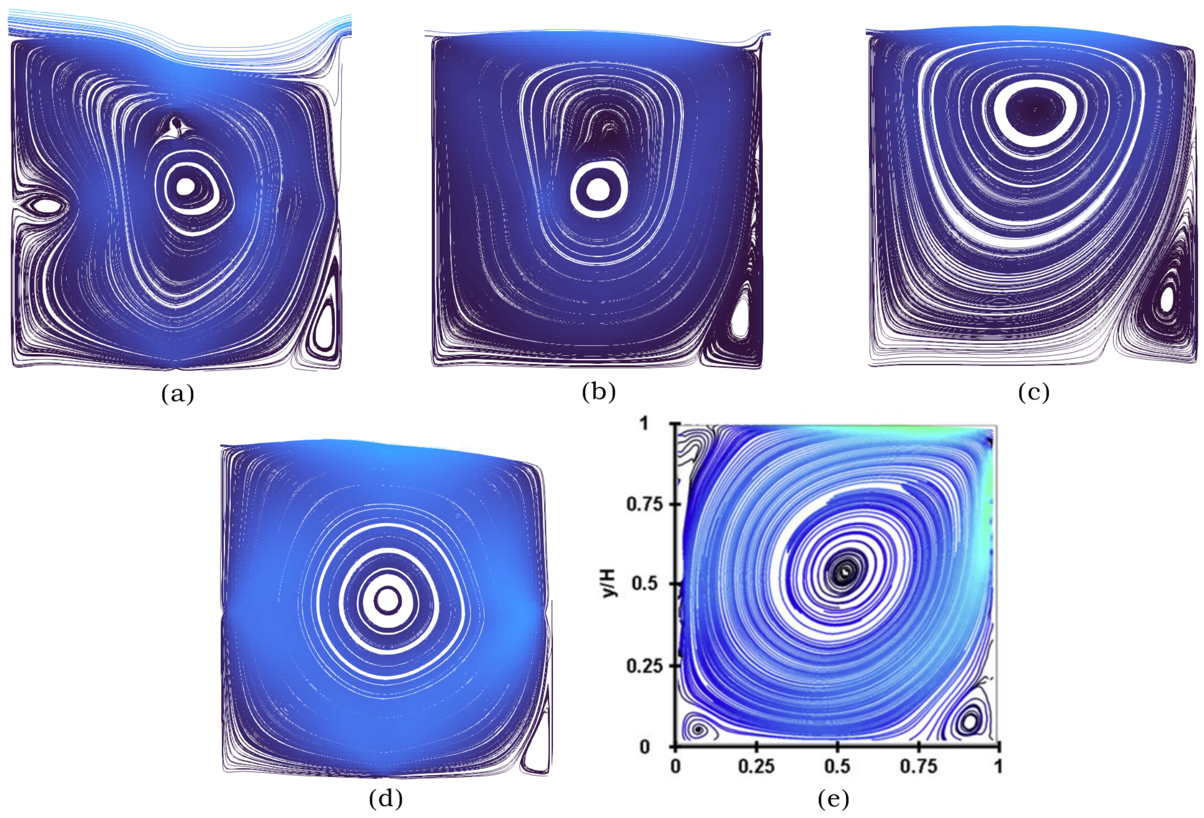


Figure 5.24: Velocity Streamlines for (a) Case **(a)** ; (b) Case **(b)** ; (c) Back & Forth Sweep ; (d) Cross-Sectional Nudging for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001$ sec, $\alpha = 2500$; (e) Allegrini Case [45]

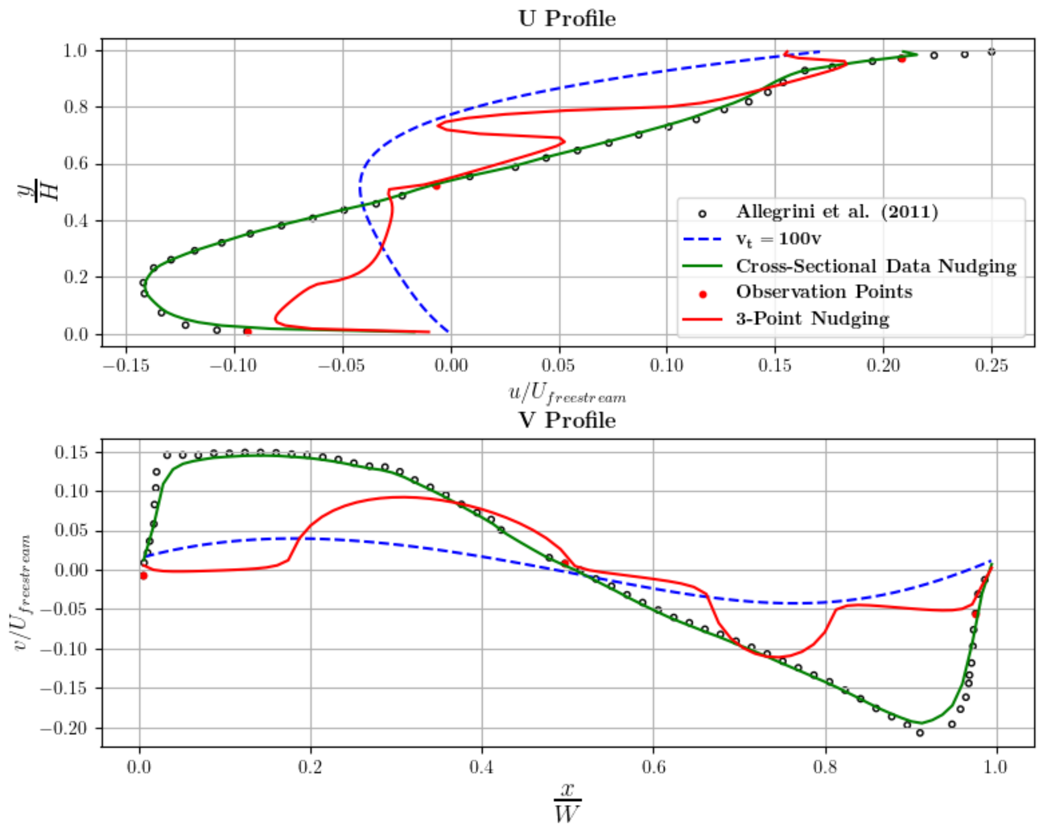


Figure 5.25: Case (a) - Velocity Profiles for 3-Point Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

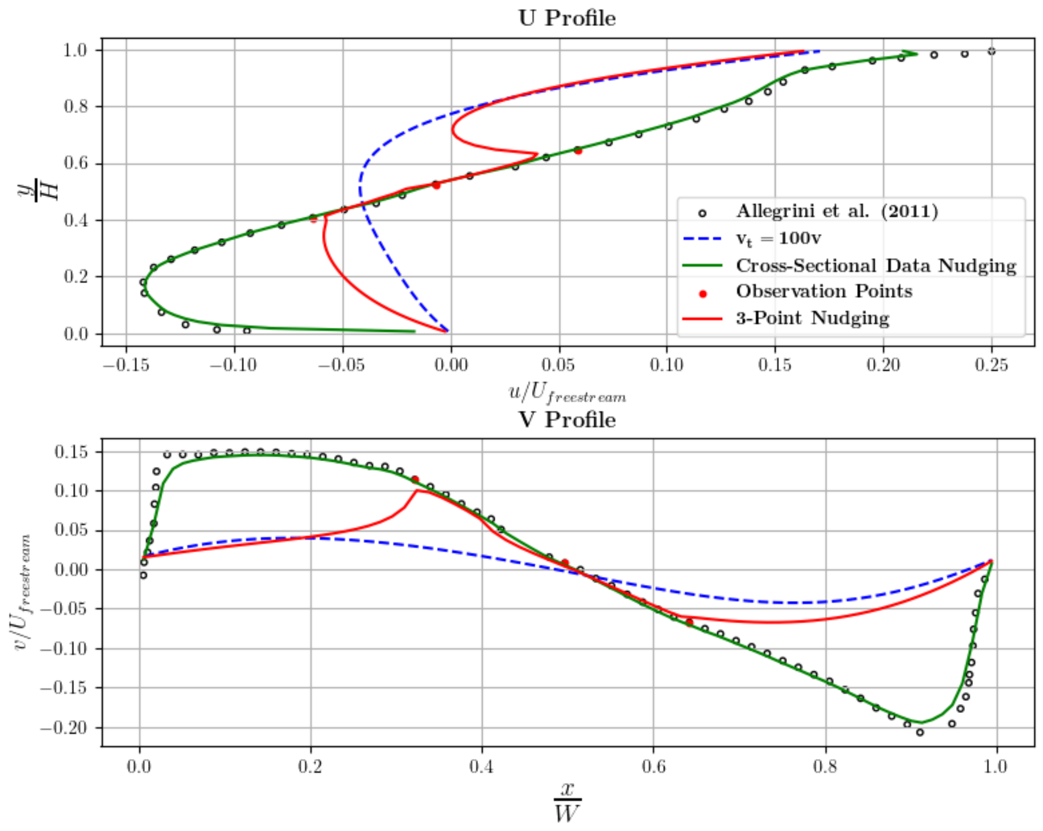


Figure 5.26: Case (b) - Velocity Profiles for 9-Point Nudging Simulation for Turbulence Model $\nu_t = 100\nu$, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 2500$

The velocity profiles depicted in Figures 5.25 and 5.26 reveal poor performance of the nudging technique. The profiles fail to align with all the observation points used in the process and exhibit local peaks around these points. Furthermore, the profiles demonstrate high levels of unnatural behavior, a characteristic also confirmed by the velocity contours. Interestingly, although case (a) yields a lower RMSE, the overall results are inferior compared to case (b). This suggests that RMSE alone may not be a reliable metric for assessing the effectiveness of the nudging technique. Finally, it's worth noting that the positioning of the nudging points has a dramatic impact on the flow behavior.

For the zero equation model, nudging was applied in conjunction with the cross-sectional data using a nudging parameter of $\alpha = 1000$. However, the simulation failed to converge due to repetitive instabilities occurring in the shearing zone of the flow, particularly on top of the street canyon. This instability is apparent in the velocity profiles depicted in Figure 5.27, where the profiles for different timesteps align well with the experimental data and with each other, except for the horizontal profile at the top.

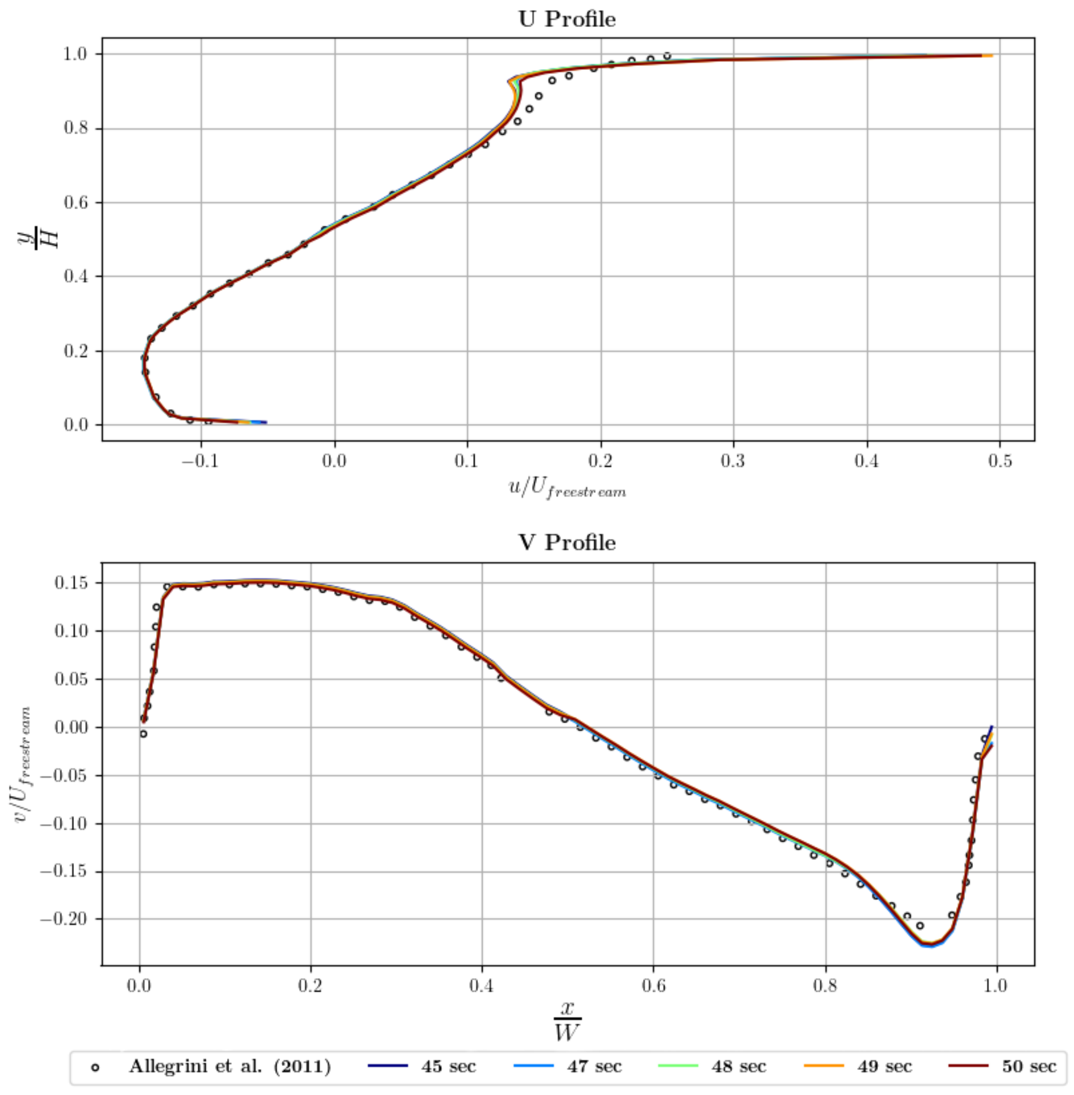


Figure 5.27: Velocity Profiles for Cross-Sectional Nudging Simulation for Zero Equation Turbulence Model, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 1000$

In contrast, the velocity streamlines shown in Figure 5.28 reveal an unexpected result. While the zero equation model was able to capture the lower left vortex in the FFD simulation using the back-and-forth sweep interpolation method, the nudged simulation fails to do so. This discrepancy could stem from the lower-order scheme used or from limitations inherent in the nudging process itself. Nevertheless, despite this difference, the streamlines match well with Allegrini's [45] streamlines in Figure 5.7 in terms of magnitude, thereby highlighting the weakness of $\nu_t = 100\nu$ turbulence model.

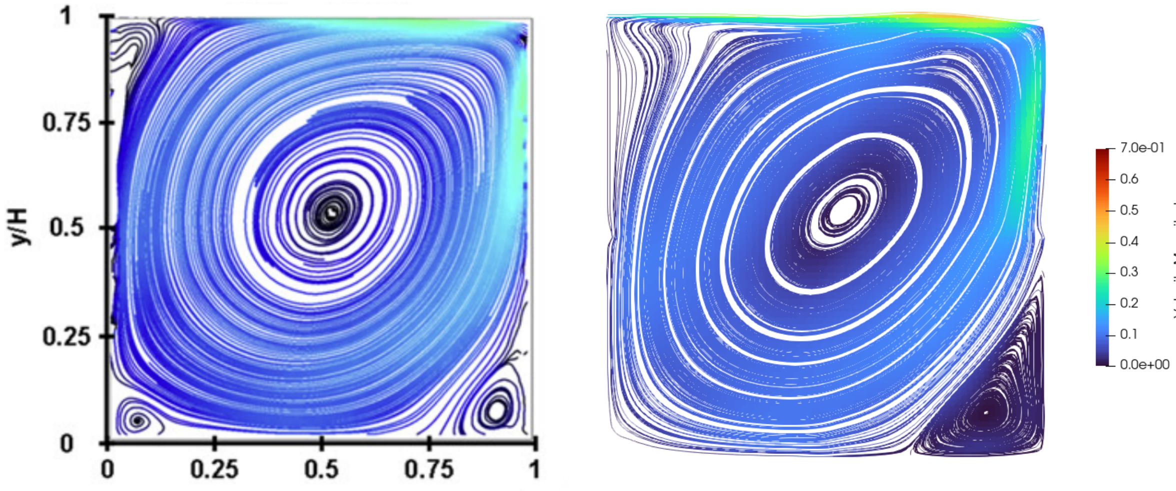


Figure 5.28: Velocity Streamlines for Cross-Sectional Nudging Simulation for Zero Equation Turbulence Model, $Re = 9000$, $\Delta t = 0.001\ sec$, $\alpha = 1000$

6. Conclusions and Future Work

6.1 Summary & Conclusions

In this thesis, an innovative Fast Fluid Dynamics (FFD) solver was developed using C++. The primary objective was to investigate the effectiveness of coupling this solver with a data assimilation technique known as nudging to enhance the accuracy of fluid flow simulations.

Initially, the focus was on developing and validating the FFD solver. It underwent rigorous testing using a lid-driven cavity benchmark, which is a common test case in computational fluid dynamics. The solver demonstrated exceptional performance in terms of both flow accuracy and computational speed, attributed to a novel back-and-forth interpolating scheme in the semi-Lagrangian method. Notably, it even outperformed corresponding simulations conducted using OpenFOAM, a widely used open-source CFD software package. Moreover, the solver demonstrated excellent stability across various time steps, thanks to the semi-Lagrangian scheme employed in the convection equation.

With the solver's robust performance established, the subsequent step involved integrating nudging algorithms into the solver framework. This allowed for the investigation of how nudging could further improve the accuracy of fluid flow simulations by assimilating observational data into the model. The results verified that nudging was capable of compensating for the numerical errors of computational solutions and was able to guide the solution to mirror the results of well documented fine-grid simulations. The results indicated that nudging effectively compensated for numerical errors in computational solutions and guided the solution to closely match results from well-documented fine-grid simulations. However, determining the optimal nudging parameter, α, required a trial-and-error approach. Furthermore, narrowing down the number of observation points used during nudging was explored, revealing a decrease in the method's efficiency. Although results still exhibited smaller errors, the positioning and topology of observation points were found to be critical factors influencing the effectiveness of the nudging method.

The process was repeated for a second application, focusing on an urban street canyon, where two different turbulence models were investigated. Unfortunately, both models exhibited poor results, attributed to either excessive damping of the flow or inherent instability in turbulence modeling. Additionally, during this investigation, the FFD solver demonstrated poor computational efficiency. It was observed that as the number of nodes in the mesh increased, the solver's performance gradually deteriorated. This degradation in performance was primarily due to challenges related to the convection equation. Despite the solver's initial success in the lid-driven cavity benchmark, the complexities of urban flow dynamics posed significant challenges that the solver struggled to effectively address. Even when assimilating experimental data through nudging, the solver successfully matched the velocity profiles in the canyon's centerlines but failed to correct the rest of the flow. This observation underscores the importance of both the placement and number of observation points in nudging. It highlights the nuanced nature of data assimilation techniques and the need for careful consideration of observational data distribution to effectively guide simulations.

6.2 Future Work

Based on the analyses conducted in the present thesis, several areas warrant further investigation for future research. First and foremost, enhancing the computational efficiency of the proposed C++ solver is imperative. One promising avenue for achieving this goal is to implement a more efficient solver for solving the elliptic equations, such as the multigrid method. Moreover, addressing the computational demands associated with the convection equation is crucial. This aspect of the FFD solver appears to consume the most computational resources and therefore requires a thorough debugging session to identify and rectify any bottlenecks in the processes. Once these issues have been pinpointed and addressed, optimizing the convection equation to run in parallel could lead to a substantial improvement in the solver's performance. Parallelization would allow for more efficient utilization of computational resources, potentially resulting in significant speedups and improved scalability.

In terms of accuracy, non-laminar applications pose a significant challenge. Exploring alternative turbulence models is necessary to improve solver accuracy in such scenarios. Wall functions could be a computationally efficient approach to handle boundary layers without requiring excessively fine grid resolutions near the boundaries. Additionally, incorporating literature proposals such as finite volume methods or artificial compressibility corrections could enhance model accuracy. These approaches offer potential solutions to improve the representation of complex flow phenomena and should be pursued to ensure the solver's reliability across a broader range of applications.

Furthermore, extending the FFD solver to 3D cases and non-structured meshes is essential for its applicability to a wider range of real-world problems. This expansion would enable the solver to handle more complex geometries and flow patterns, enhancing its versatility and usefulness in various engineering problems.

Finally, the nudging process requires further fine-tuning. The selection of the nudging parameter, α, appears to be particularly challenging as it is highly case-specific and cannot be automatically determined. There is a need to devise a method for selecting the optimal value without incurring significant additional computational costs. Additionally, in all the simulations presented in the thesis, nudging was performed using steady-state data. Exploring the possibility of incorporating real-time measurements from observations and running the solver simultaneously should be investigated. This approach could potentially lead to better predictive capabilities and more reliable results, as the simulation would be dynamically updated.

Bibliography

[1] J. Stam, "Stable Fluids," *ACM SIGGRAPH 99*, vol. 1999, 11 2001. [Online]. Available: https://www.researchgate.net/publication/2486965_Stable_Fluids

[2] A. Staniforth and J. Côté, "Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review," *Monthly Weather Review*, vol. 119, no. 9, pp. 2206 – 2223, 1991. [Online]. Available: https://journals.ametsoc.org/view/journals/mwre/119/9/1520-0493_1991_119_2206_slisfa_2_0_co_2.xml

[3] N. Max, R. Crawfis, and D. Williams, "Visualizing Wind Velocities by Advecting Cloud Textures," *Proceedings Visualization '92*, pp. 179–184, 4 1992. [Online]. Available: https://ieeexplore.ieee.org/document/235210

[4] R. Fedkiw, J. Stam, and H. Jensen, "Visual Simulation of Smoke," *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 15–22, 06 2001. [Online]. Available: https://dl.acm.org/doi/10.1145/383259.383260

[5] Harris and Mark, "Real-Time Cloud Simulation and Rendering," 07 2005. [Online]. Available: https://www.researchgate.net/publication/262285593_Real-time_Cloud_Simulation_and_Rendering

[6] W. Zuo and Q. Chen, "Real-Time or Faster-than-Real-Time Simulation of Airflow in Buildings," *Indoor Air*, vol. 19, pp. 33–44, 03 2009. [Online]. Available: https://www.researchgate.net/publication/23976130_Real-time_or_faster-than-real-time_simulation_of_airflow_in_buildings

[7] W. Zuo, J. Hu, and Q. Chen, "Improvements in FFD Modeling by Using Different Numerical Schemes," *Numerical Heat Transfer Part B-fundamentals*, vol. 58, pp. 1–16, 08 2010. [Online]. Available: https://www.researchgate.net/publication/263407897_Improvements_of_Fast_Fluid_Dynamics_for_Simulating_Air_Flow_in_Buildings

[8] W. Zuo and Q. Chen, "Simulations of Air Distributions in Buildings by FFD on GPU," *HVAC&R Research*, vol. 16, pp. 785–798, 11 2010. [Online]. Available: https://www.researchgate.net/publication/254306414_Simulations_of_Air_Distributions_in_Buildings_by_FFD_on_GPU

[9] W. Liu, R. You, J. Zhang, and Q. Chen, "Development of a Fast Fluid Dynamics-Based Adjoint Method for the Inverse Design of Indoor Environments," *Journal of Building Performance Simulation*, vol. 10, pp. 1–18, 11 2016. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/19401493.2016.1257654

[10] Y. Xue and W. Liu, "Inverse Design of Built Environment by a Fast Fluid Dynamics-based Genetic Algorithm," 01 2019, pp. 2880–2885. [Online]. Available: https://www.researchgate.net/publication/342385454_Inverse_Design_of_Built_Environment_by_a_Fast_Fluid_Dynamics-based_Genetic_Algorithm

[11] W. Tian, J. Vangilder, X. Han, C. Healey, M. Condor, and W. Zuo, "A New Fast Fluid Dynamics Model for Data-Center Floor Plenums," vol. 125, no. 1, 01 2019. [Online]. Available: https://www.researchgate.net/publication/330542284_A_New_Fast_Fluid_Dynamics_Model_for_Data-Center_Floor_Plenums

[12] T. Dai, S. Liu, J. Liu, N. Jiang, W. Liu, and Q. Chen, "Evaluation of Fast Fluid Dynamics with Different Turbulence Models for Predicting Outdoor Airflow and Pollutant Dispersion," *Sustainable Cities and Society*, vol. 77, p. 103583, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670721008489

[13] H. Kato and S. Obayashi, "Integration of CFD and Wind Tunnel by Data Assimilation," *Journal of Fluid Science and Technology*, vol. 6, no. 5, pp. 717–728, 2011. [Online]. Available: https://www.jstage.jst.go.jp/article/jfst/6/5/6_5_717/_article/-char/en

[14] M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2016. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611974546

[15] M. Meldi and A. Poux, "A Reduced Order Model based on Kalman Filtering for Sequential Data Assimilation of Turbulent Flows," *Journal of Computational Physics*, vol. 347, pp. 207–234, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999117304965

[16] H. Kato and S. Obayashi, "Integration of CFD and Wind Tunnel by Data Assimilation," *Journal of Fluid Science and Technology*, vol. 6, no. 5, pp. 717–728, 2011. [Online]. Available: https://www.jstage.jst.go.jp/article/jfst/6/5/6_5_717/_article/-char/en

[17] P. Chandramouli, E. Memin, and D. Heitz, "4D Large Scale Variational Data Assimilation of a Turbulent Flow with a Dynamics Error Model," *Journal of Computational Physics*, vol. 412, p. 109446, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999120302205

[18] S. Symon, N. Dovetta, B. McKeon, D. Sipp, and P. Schmid, "Data Assimilation of Mean Velocity from 2D PIV Measurements of Flow over an Idealized Airfoil," *Experiments in Fluids*, vol. 58, 04 2017.

[19] T. Hayase and S. Hayashi, "State Estimator of Flow as an Integrated Computational Method With the Feedback of Online Experimental Measurement," *Journal of Fluids Engineering*, vol. 119, no. 4, pp. 814–822, 12 1997. [Online]. Available: https://doi.org/10.1115/1.2819503

[20] K. Imagawa and T. Hayase, "Numerical Experiment of Measurement-Integrated Simulation to Reproduce Turbulent Flows with Feedback Loop to Dynamically Compensate the Solution using Real Flow Information," *Computers & Fluids*, vol. 39, no. 9, pp. 1439–1450, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004579301000099X

[21] M. Zauner, V. Mons, O. Marquet, and B. Leclaire, “Nudging-Based Data Assimilation of the Turbulent Flow around a Square Cylinder,” *Journal of Fluid Mechanics*, vol. 937, p. A38, 2022. [Online]. Available: https://hal.science/hal-03709219

[22] J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics*. Springer International Publishing, 2020, p. 426. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-56026-2#bibliographic-information

[23] S. Konangi, N. K. Palakurthi, and U. Ghia, “Von Neumann Stability Analysis of First-Order Accurate Discretization Schemes for One-Dimensional (1D) and Two-Dimensional (2D) Fluid Flow Equations,” *Computers & Mathematics with Applications*, vol. 75, no. 2, pp. 643–665, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089812211730620X

[24] F. H. Harlow and J. E. Welch, “Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface,” *The Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 12 1965. [Online]. Available: https://doi.org/10.1063/1.1761178

[25] C. M. Rhie and W. L. Chow, “Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation,” *AIAA Journal*, vol. 21, no. 11, pp. 1525–1532, 1983. [Online]. Available: https://doi.org/10.2514/3.8284

[26] N. Stoop, “Lecture 12,” MIT OpenCourseWare, Massachusetts Institute of Technology, Spring 2016, computational Science And Engineering II.

[27] B. Seibold, “Lecture 23,” MIT OpenCourseWare, Massachusetts Institute of Technology, Spring 2009, numerical Methods for Partial Differential Equations.

[28] M. Mortezazadeh, L. L. Wang, M. Albettar, and S. Yang, “CityFFD – City Fast Fluid Dynamics for Urban Microclimate Simulations on Graphics Processing Units,” *Urban Climate*, vol. 41, p. 101063, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212095521002935

[29] J. C. Andrew Staniforth, “Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review,” *Review Articles in Monthly Weather Review*, vol. 135, p. 2206–2223, 09 1991. [Online]. Available: https://journals.ametsoc.org/view/journals/mwre/113/3/1520-0493_1985_113_0388_aslasi_2_0_co_2.xml

[30] M. Mortezazadeh and L. L. Wang, “An Adaptive Time-Stepping Semi-Lagrangian Method for Incompressible Flows,” *Numerical Heat Transfer, Part B: Fundamentals*, vol. 75, no. 1, pp. 1–18, 2019. [Online]. Available: https://www.researchgate.net/publication/332433395_An_adaptive_time-stepping_semi-Lagrangian_method_for_incompressible_flows

[31] A. Salih, “Method of Characteristics,” Indian Institute of Space Science and Technology, Thiruvananthapuram, Tech. Rep., 06 2016.

[32] N. Wood, A. Staniforth, and A. White, “Determining Near-Boundary Departure Points in Semi-Lagrangian Models,” *Quarterly Journal of the Royal Meteorological Society*, vol. 135, pp. 1890 – 1896, 10 2009. [Online]. Available: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.478

[33] M. Jin, W. Zuo, and Q. Chen, “Simulating Natural Ventilation in and Around Buildings by Fast Fluid Dynamics,” *Numerical Heat Transfer Part A - Applications*, vol. 64, pp. 273–289, 08 2013. [Online]. Available: https://www.researchgate.net/publication/258806507_Simulating_Natural_Ventilation_in_and_Around_Buildings_by_Fast_Fluid_Dynamics

[34] Jin, Mingang and Zuo, Wangda and Chen, Qingyan, “Improvements of Fast Fluid Dynamics for Simulating Air Flow in Buildings,” *Numerical Heat Transfer Part B: Fundamentals*, vol. 62, 01 2012. [Online]. Available: https://www.researchgate.net/publication/263407897_Improvements_of_Fast_Fluid_Dynamics_for_Simulating_Air_Flow_in_Buildings

[35] M. Mortezazadeh Dorostkar and L. Wang, “A High-Order Backward Forward Sweep Interpolating Algorithm for Semi-Lagrangian Method,” *International Journal for Numerical Methods in Fluids*, vol. 84, 01 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.4362

[36] G. Stewart, *Afternotes on Numerical Analysis*, ser. Lectures on Elementary Numerical Analysis. Society for Industrial and Applied Mathematics, 1996. [Online]. Available: https://books.google.gr/books?id=XvdSuKcKpmwC

[37] Q. Chen and W. Xu, “A Zero-Equation Turbulence Model for Indoor Airflow Simulation,” *Energy and Buildings*, vol. 28, pp. 137–144, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778898000206

[38] P. Clark Di Leoni, A. Mazzino, and L. Biferale, “Synchronization to Big Data: Nudging the Navier-Stokes Equations for Data Assimilation of Turbulent Flows,” *Phys. Rev. X*, vol. 10, p. 011023, Feb 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.011023

[39] U. Ghia, K. Ghia, and C. Shin, “High-Re Solutions for Incompressible Flow using the Navier-Stokes Equations and a Multigrid Method,” *Journal of Computational Physics*, vol. 48, no. 3, pp. 387–411, 1982. [Online]. Available: https://www.semanticscholar.org/paper/High-Re-solutions-for-incompressible-flow-using-the-Ghia-Ghia/211b45b6a06336a72ca064a6e59b14ebc520211c

[40] S. Biswas and J. Kalita, “Moffatt Vortices in the Lid-Driven Cavity Flow,” *Journal of Physics: Conference Series*, vol. 759, 01 2016. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/759/1/012081/pdf

[41] H. Kuhlmann and F. Romanò, “The Lid-Driven Cavity,” *Computational Methods in Applied Sciences*, 04 2018. [Online]. Available: https://www.researchgate.net/publication/324413434_The_Lid-Driven_Cavity

[42] Z. Ai and C. Mak, “CFD Simulation of Flow in a Long Street Canyon under a Perpendicular Wind Direction: Evaluation of Three Computational Settings,” *Building and Environment*, vol. 114, pp. 293–306, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132316305273

[43] X. Xiaomin, H. Zhen, and W. Jiasong, “The Impact of Urban Street Layout on Local Atmospheric Environment,” *Building and Environment*, vol. 41, no. 10, pp. 1352–1363, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132305002003

[44] K. Hoffmann and S. Chiang, *Computational Fluid Dynamics*, ser. Computational Fluid Dynamics. Engineering Education System, 2000, no. v. 1. [Online]. Available: https://books.google.gr/books?id=98gjAAAACAAJ

[45] J. Allegrini, V. Dorer, and J. Carmeliet, “Wind Tunnel Measurements of Buoyant Flows in Street Canyons,” *Building and Environment*, vol. 59, pp. 315–326, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132312002399