IMPLEMENTATION AND VALIDATION OF TVD SCHEMES FOR MODELLING OF MULTIPHASE FLOW

by

Qianyu Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
University of Alberta

Abstract

This work focuses on the implementation and validation of the implicit Total Variation Diminishing (TVD) schemes for modelling of two-dimensional multiphase flow. The TVD schemes are designed to mitigate the numerical diffusion at the interface where discontinuities of fluid properties exist while preserving the boundedness of the solutions. The volume of fluid approach was taken. The one-dimensional code was implemented in FORTRAN 77 and MATLAB, and the two-dimensional code was implemented in FORTRAN 77. The in-house developed code can be expanded into three-dimensions using similar algorithms and structures. The programs can be served as computational tools in the future studies of multiphase flow.

To validate the code, a one-dimensional passive scalar convection-diffusion problem, the lids-driven cavity (LDC) problem and the cylinder convection problem were investigated. The code results were compared with the solutions from a commercial computational fluid dynamics software ANSYS Fluent and the analytical solutions. In the one-dimensional case, with a grid size of 0.125 m, the TVD code produced an average relative error in the order of 10^{-1}, while Fluent produced an average relative error in the order of 10^{0} with its most accurate discretization scheme. The maximum error between the code prediction and the Fluent prediction for the LDC problem was 7% using the second-order upwind scheme at a grid size of 0.01 m. The relative errors for the first and third-order schemes were close to 1%.

In the cylinder advection studies, the TVD schemes were effective in the cylinder volume conservation. Among the four TVD schemes tested, the SUPERBEE flux limiter function produced the most accurate results while consuming the longest time; whereas the Lin-Lin flux limiter function was the best in terms of the usage of computational resources. With a 0.02 m mesh size, the SUPERBEE method gave an error of 7% in cylinder volume after 80 seconds of elapsed time.

Acknowledgements

During my time as a Master's student at the University of Alberta, I have received enormous support from my supervisors, **Dr. Alexandra Komrakova** and **Dr. Petr Nikrityuk**. Dr. Petr Nikrityuk introduced me to the field of computational fluid dynamics and aroused my interest in this field. Dr. Alexandra Komrakova helped me through the tough transition from petroleum engineering, in which area I pursued my undergraduate degree, to mechanical engineering. Their enthusiasm and passion in their field of study motivated me through my own research. They were always open for discussions, and have always treated the students with respect. They have made my graduate life repletive and enjoyable. I would like to give my sincerest gratitude to them for taking great care of me.

I would also like to thank my group members, Zhe Chen, Hongbo Shi, Xiaomeng Li and Yiran Lu for helping me through the countless technical problems I encountered. They have created a harmonious work environment so I could concentrate on my research and was never distracted by any conflict. They have become my friends in life but not just colleagues at work.

Last but not least, I would like to express my appreciation to my family; my dearest husband, Xuefei Han for his patience and support; my father Danjun Zhang and my mother Xiaowen Zhang for their constant trust and encouragement. I am fortunate to have a family that respect all my decisions. Their unconditional love is indispensable for me.

Table of Contents

Abstract ii

Acknowledgments iii

1 Introduction to Multiphase Flow 1

1.1 Numerical Models for Two-Phase Flow 2

1.2 Applications and Recent Publications 4

1.3 Overview on Interface Reconstruction Techniques 5

1.4 Conclusions 6

2 Introduction to TVD 7

2.1 Background of TVD Schemes 9

2.2 Applications of TVD Schemes 10

2.3 TVD Validation Cases 11

2.3.1 Dam-Break Problems 11

2.3.2 Bubble-in-Liquid Problems 17

2.3.3 Other Cases 20

2.4 TVD Deficiencies 21

2.5 Objectives 21

3 1D Formulation and Validation 22

3.1 Introduction to TVD Criteria 22

3.2 Formulation of the Convection-Diffusion Equation 28

3.3 Comparison of Results from Various Methods 31

3.4 Verification on Order of Accuracy 34

3.5 Validation 35

3.5.1 ANSYS Fluent 35

4 2D Formulation and Validation **38**
4.1 Formulation of the Convection-Diffusion Equation 38
4.2 2D Lids-Driven Cavity Problem 40
4.3 2D Cylinder Advection 46
4.4 Comparison between various TVD schemes 57
4.5 Time Complexity Analysis 62

5 Conclusions **64**
5.1 Future works 65

A 1D FORTRAN Code **70**

B 1D MATLAB Code **79**

C Sample Output from 1D Code **87**

List of Tables

2.1 Summary of TVD validation cases, adapted from [1] 12
2.2 Summary of TVD validation cases in dam-break problems 16
2.3 Summary of TVD validation cases in bubble-in-liquid problems . . . 19

3.1 Conversion from shape sensing functions to flux limiter functions . . 26
3.2 Conversion from shape sensing functions to flux limiter functions for TVD schemes 27

4.1 Summary of mean relative errors (%) as the grid refines 42
4.2 Summary of maximum relative discrepancies (%) as the grid refines 45

List of Figures

2.1 Flow domain for false diffusion illustration, adapted from [2] 8
2.2 Comparison of numerical solutions obtained from TVD schemes with the QUICK results, UDS results and the exact solution respectively on a 50 x 50 grid, adapted from [2] 8
2.3 Example of discretized points for TV illustration, adapted from [2] . 9
2.4 Schematic of two-dimensional idealized dam-break, adapted from [1] 13
2.5 Side view of two-dimensional dam-break in channels, adapted from [3] 14
2.6 Top view of three-dimensional partial dam-break, adapted from [4] . 15
2.7 Schematics of initial conditions in bubble-in-liquid problems, adapted from (a) [1] (b) [1] (c) [5] (d) [6] (the schematics are not to scale) . . 17

3.1 Example of one-dimensional control volume discretization around node P on a uniform grid 22
3.2 $r - \psi$ diagram for conventional schemes 23
3.3 $r - \psi$ diagram for TVD schemes 24
3.4 NVD curves for conventional schemes 25
3.5 NVD curves for TVD schemes 25
3.6 Convection-diffusion problem simulation domain with boundary conditions implemented in code 28
3.7 Normalized ϕ variations from UDS, CDS, LUDS, QUICK and TVD schemes for (a) 10 CVs (b) 20 CVs (c) 40 CVs (d) 80 CVs 32
3.8 Maximum percentage error variation with Peclet number for code results 33
3.9 Order of accuracy check for code results 34
3.10 Convection-diffusion problem simulation domain with boundary conditions in Fluent 35
3.11 Comparison between code and Fluent results obtained from LUDS for (a) 10 CVs (b) 20 CVs (c) 40 CVs (d) 80 CVs 36

3.12 Maximum percentage error variation with Peclet number for Fluent results 36
3.13 Order of accuracy check for Fluent results 37

4.1 Example of two-dimensional control volume discretization around node P on a uniform grid 38
4.2 Schematic of the lids-driven cavity (LDC) problem 40
4.3 Normalized temperature profile for the lids-driven cavity (LDC) problem 41
4.4 Normalized temperature plots from various methods along $x = 0.5$ m: (a) 1250 CVs (c) 5000 CVs (e) 20000 CVs and along $x = 1.5$ m: (b) 1250 CVs (d) 5000 CVs (f) 20000 CVs 43
4.5 Comparison between results from code and Fluent along $x = 0.5$ m: (a) 1250 CVs (c) 5000 CVs (e) 20000 CVs and along $x = 1.5$ m: (b) 1250 CVs (d) 5000 CVs (f) 20000 CVs 44
4.6 Illustration of the stair-step approximation 47
4.7 Initial schematic of the cylinder convection problem 47
4.8 Volume fraction contour plots obtained with a 50×500 grid at $t = 40$ s for: CDS, UDS, LUDS, QUICK and TVD respectively 49
4.9 Volume fraction contour plots obtained with a 50×500 grid at $t = 80$ s for: CDS, UDS, LUDS, QUICK and TVD respectively 50
4.10 Volume fraction contour plots obtained with a 100×1000 grid at $t = 40$ s for: CDS, UDS, LUDS, QUICK and TVD respectively . . . 51
4.11 Volume fraction contour plots obtained with a 100×1000 grid at $t = 80$ s for: CDS, UDS, LUDS, QUICK and TVD respectively . . . 52
4.12 Normalized volume averaged fraction obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various schemes 53
4.13 Normalized cylinder volume obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various schemes 55
4.14 Illustration of the fluctuation in the stair-step approximated volume 56
4.15 Volume fraction contour plots obtained with a 50×500 grid at $t = 40$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively . . 57
4.16 Volume fraction contour plots obtained with a 50×500 grid at $t = 80$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively . . 58
4.17 Volume fraction contour plots obtained with a 100×1000 grid at $t = 40$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively 59

4.18 Volume fraction contour plots obtained with a 100×1000 grid at $t = 80$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively 60
4.19 Normalized cylinder volume obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various TVD schemes 61
4.20 The correlation between real time elapse and numerical time elapse for the cylinder advection studies. The dashed lines are for the 50×500 grid and solid lines for the 100×1000 grid: (a) for conventional schemes (b) for TVD schemes 63

List of Abbreviations

CDS	Central Difference Scheme
CFD	Computational Fluid Dynamics
CFL	Courant-Friedriech-Levi
CV	Control Volume
DEM	Discrete Element Method
DNS	Direct Numerical Simulation
LDC	Lids-Driven Cavity
LPT	Lagrangian Particle tracking
LS	Level-Set
LUDS	Linear Upwind Difference Scheme
MUSCL	Monotonic Upwind Scheme for Conservation Laws
NS	Navier-Stokes
PDF	Probability Distribution Function
PLIC	Piecewise Linear Interface Calculation
QUICK	Quadratic Upstream Interpolation for Convective Kinematics
SIP	Strongly Implicit Procedure
SLIC	Simple Linear Interface Calculation
TDMA	Tri-Diagonal Matrix Algorithm
TVD	Total Variation Diminishing
TVNI	Total Variation NonIncreasing
U	Unit length
UDS	Upwind Difference Scheme
VOF	Volume of Fluid

List of Symbols

b	water column width
E	east node
e	east face of control volume
h	water column height
N	north node
n	north face of control volume
P	central node
r	ratio of upwind gradient to downwind gradient
S	south node
s	south face of control volume
W	west node
w	west face of control volume

Greek letters

α	volume fraction
β	compressibility parameter
ε	fluid volume fraction
φ	level-set function
ϕ	transport property
ψ	flux limiter function

Subscripts

g	gas phase
p	particle
s	solid phase

Chapter 1

Introduction to Multiphase Flow

Fluid flow is present in all aspect of our life. It is the fundamental of meteorology, hydro-geology, aeronautics, and many more other subjects. Its significance becomes more appreciated with the recent advancement and growing demand for renewable energy. Fluid flow is able to carry kinematic energy as in windmills and water dams; and in combination with thermal energy, it is capable of storing and transporting heat [7]. Multiphase flow is the fluid flow consisting of two or more distinct phases. The phases refer to either physical states (solid, liquid and gas) or distinct liquids that are immiscible. In general, in the heterogeneous mixture, the two phases can be dispersed or separated. Water cascade exposed in the atmosphere is an example of separated flow. Particle-laden flow, fluidized bed and atomization are examples of dispersed flow.

The understanding of multiphase fluid dynamics was initiated from the observations and develops through experimentations. Exact solutions for some simplified problems can be found, but it is unrealistic to solve most of the practical applications analytically. Thus, many scholars were dedicated to finding empirical models to describe the flow behaviours. For instance, the well established Richardson and Zaki [8] correlation for fluidized bed; and the Beggs and Brill [9] model for liquid-gas pipe flow. On the other hand, some other researchers use the aid of computers to promote the understanding of the fundamentals of flow. Such study is called computational fluid dynamics (CFD). In CFD, the flow is represented by rudimentary equations to be discretized and solved iteratively by computers. After validating the CFD solutions with experimental results, the developed CFD tool is commonly used to predict more complex flows.

In contrast to single-phase flow, multiphase flow introduces the complication of the interface and the handling of the surface tension force. The sharp change in fluid properties at the interface poses exceptional challenges to researchers [10]. In addition, unlike the solid-fluid flows, in which the solid components stay relatively constant in shape and size, the fluid-fluid flows deal with interfaces that are dynamic. Therefore, it is crucial to control and minimize the numerical errors, such as the spurious current and false diffusion, which will be defined in Chapter 2.

1.1 Numerical Models for Two-Phase Flow

Before introducing the approaches used for multiphase flow calculations, a brief discussion about the flow description theorems is necessary. In Eulerian description, the fluid is treated as a continuum. The flow evaluation is based on a fixed volume in space, and the constitutive equation is the conservation of momentum. In Lagrangian description, the particles are traced by their trajectories, while the particle force balance is solved.

Therefore, by definition, the Euler-Lagrange approach deals with one of the phases using the Eulerian description and the other phase using the Lagrangian description. As a result, this approach is appropriate for flows that are comprised of a continuous phase and a dispersed phase, which is scattered and occupies only a small fraction of the total fluid volume [11]. This approach is also known as the Lagrangian particle tracking (LPT) method [12] or discrete element method (DEM).

In the Euler-Euler approach, both phases are treated as continua. The volume of fluid (VOF) method and the Eulerian method are base on the Euler-Euler approach. The Eulerian method requires solving multiple sets of momentum equations, one for each phase. This method is rather sophisticated hence rarely used. In the VOF method, only one set of momentum equations is needed. The volume fraction α, which is a passive scalar, is obtained by solving an advection equation. A passive scalar is a quantity that is independent of both pressure and temperature. The volume fraction for a particular phase is the volume occupied by that phase divided by the total volume.

The fraction is dimensionless and is expected to be bounded within zero and one. In a control volume, a fraction of zero means that the phase is absolutely absent, whereas a fraction of unity indicates that the phase is the only phase present. Ideally, for immiscible fluids, α should be either exactly zero or exactly one in all cells except at the interface. The molecular diffusion is negligible, so the physical interface is thin. The interface is located at $\alpha = 0.5$. In numerical practices, numerical diffusion

from discretization errors is inevitable; as a result, a thick layer of cells has α in between zero and one, indicating unphysical mixing of the two phases, and smeared interface. The interface smearing cannot be eliminated but can be controlled and mitigated by using a finer mesh, or a higher order discretization scheme.

Moreover, fractions of all phases should sum up to one. The fluid properties used in solving the constitutive equations are weighted functions of the properties of individual fluids, based on their fractions. Supposing in a two-phase system, there exists an intensive scalar property ϕ. The ϕ values for the two phases are ϕ_1 and ϕ_2 respectively. Let ε be the volume fraction of phase one, the volume fraction of phase two is then $(1-\varepsilon)$. The ϕ value for the resulting mixture, denoted by ϕ_m, is demonstrated in Equation 1.1.

$$\phi_m = \varepsilon\phi_1 + (1-\varepsilon)\phi_2 \tag{1.1}$$

It should be noted that solid particles can also be regarded as continuum [13]. With the presence of large quantity of solid particles, tracking individual particles with direct numerical simulation (DNS) becomes impractical. Resultantly, some averaging techniques were developed to summarize the behaviours of the solid particles. In this case, the concentration of the solid species is monitored, instead of the fluid volume fraction that is used in conventional VOF. The constitutive equations imitate the equations used for fluid-fluid simulations.

There are also approaches that focus on solving the interface instead of tracking the fluid volume, such as the markers-and-cells (MAC) method and the level-set method. In the MAC method, the computational domain is discretized into a Eulerian grid, and the fluids are highlighter by frictional marker particles, which are traced with the Lagrangian description. Unlike the VOF method, in which the domain always has to be entirely filled by fluid, the MAC method allows cavity cells, denoted by "null" cells. To simplify a two-phase system with large density and viscosity ratios, the phase with lower density and viscosity is treated as an "inert" phase. With such modification, the interface transforms into a free surface, and only one of the fluids needs to be marked and traced.

Recent development in MAC permits marker particles at only the interface. McKee et al. [14] asserted that such approach significantly saves the computational resources and is efficient for complex problems.

The advantage of the MAC method is that it produces minimum smearing at the interface. Consequently, the interface curvature calculations and resultantly, the surface tension calculations become relatively straightforward. However, since the governing equations are solved by the finite difference method rather than the finite

volume method, which is used in VOF, the mass conservation is not guaranteed and has to be carefully manipulated.

The level-set (LS) method uses a continuous function to represent the interface; this auxiliary function is so called the level-set function φ. The level-set function is dependent on variables x and t, with x being a location in space and t being the time. For a two-phase flow, one of the phases has negative φ and the other has positive φ; the interface is located at where the φ equals to zero. The LS method is efficient in solving changing interface topology, but it suffers greatly from discretization and mass conservation errors. However, due to the fact that curvature calculations in the LS method are trivial since it can be derived directly from the function formulation, it is commonly used in conjunction with other methods. Huang and Zhang [6] used the LS method to simulate a rising bubble. Wang et al. [5] investigated rising and merging of multiple bubbles using a hybrid MAC and LS scheme. Aniszewski et al. [15] used a coupled LS-VOF method to model a bubble advection.

Provided a brief discussion on some of the frequently used models for solving two-phase flows, this paper will be focusing on the VOF method.

1.2 Applications and Recent Publications

Bubble behaviours are of great interest to researchers and were studied intensively. It is commonly used in the industry to facilitate separation or promote heat transfer. Samkhaniani and Ansari [16], when investigating a case of vapour bubble condensation, pointed out that bubbles are subjected to frequent variations in size, shape, velocity and collapse rate. Due to the nature of the VOF formulations, fluid properties at the interface changes abruptly from zero to unity. This characteristic introduced errors in the calculation of the bubble's surface geometry thus results in unphysical disturbance in the flow, known as the "parasitic currents".

In fact, it is quite challenging to maintain the shape of a bubble that is immersed in a quiescent liquid, in the absence of any physical forces or velocities, using numerical simulations. Aniszewski et al. [15] conducted the static droplet test for a spherical droplet placed at the center of a computational domain. The density and viscosity ratios of the fluids inside and outside of the droplet were kept at unity. The authors [15] observed fictional velocity and bubble distortion, which may eventually lead to breakage of the bubble. Mesh refinement is ineffective in remitting this issue. The authors also noticed that the coupled LS-VOF methods are affected by convergence difficulties when it comes to the calculation of interface curvature and surface tension [15].

One approach to mitigate the bubble distortion caused by the spurious current is to use an artificial statistical filter to post-process the interface, as described in Samkhaniani and Ansari's work [16]. Another approach to reduce the impact of abrupt transitions at the interface is to give the interface a predefined thickness, normally a few cell lengths. With this treatment, the smearing is introduced artificially, and the property transitions are made smooth. However, Shardt et al. [17] concluded that this artificial interface smearing results in overlapped droplet boundaries, and thereby causing non-physical coalescence. To better resolve the interfaces in order to obtain a more physically meaningful solution, numerical schemes that are capable of mitigating the false diffusion and retain the interface sharpness is in demand.

Grosshans et al. [12] studied the atomization of a liquid jet using the VOF method. A liquid jet imitating diesel injection in combustion chambers was considered. The authors [12] examined the impacts of numerical parameters, liquid-gas density and viscosity ratios with the VOF simulations. They evaluated the probability distribution function (PDF) of droplet diameters while adjusting the input parameters. They concluded that high numerical diffusion occurs when the grid is coarse and such diffusion suppresses turbulence. The density ratio tested was from 10 to 30 and the viscosity ratio tested was from 1 to 7. With such small ratios, the difference in the numerical results is significant. Therefore, it can be inferred that VOF results are susceptible to property ratios of the two phases, which can be as great as 1000 in real flows.

1.3 Overview on Interface Reconstruction Techniques

Aniszewski et al. [15] compared four different interface reconstruction methods when investigating the static bubble problem, and found that the results are to a large extent dependent on the methods selected. Interface reconstruction techniques are impacted by mesh types. A mesh can be either fixed or moving [18], and the fixed mesh can be either structured or unstructured. A moving grid allows local mesh refinement along the curvature of the interface, in an effort to minimize smearing at the interface. However, with a small movement of the interface, such grids can be deformed drastically resulting in poorer grid quality. Elghobashi [19] stated that such method is inappropriate for turbulent flows.

Fixed unstructured grids can be easily fitted to irregular domains, but it suffers from problems concerning convergence rate and solution accuracy. On the other hand, a uniform Cartesian grid is preferable since it is more computationally ef-

ficient and is easier to implement. Some commonly used surface approximation techniques for VOF results on a uniform Cartesian grid are the stair-step approximation, the simple line interface calculation (SLIC), which produces a piecewise constant interface approximation [20], and the piecewise linear interface calculation (PLIC).

A detailed illustration of the stair-step approximation is included in Section 4.3. The SLIC algorithm is first-order accurate whereas the PLIC algorithm is second-order accurate. In the PLIC algorithm, the vector $\mathbf{n}$ normal to the interface is calculated by:

$$\mathbf{n} = \frac{\nabla \varepsilon}{|\nabla \varepsilon|} \tag{1.2}$$

The interface in a specific cell is mapped such that its normal is equivalent to the calculated $\mathbf{n}$. Rider and Kothe [20] stated that high-order schemes should be used to calculate the gradient of $\mathbf{n}$. Therefore, an accurate discretization method would also benefit the preciseness of interface reconstruction.

1.4 Conclusions

Multiphase fluid modelling is challenging because resolving the interface between phases is difficult. These difficulties are mainly caused by the discontinuity in fluid properties at the interface. Therefore, high-order numerical schemes producing accurate and bounded solutions are in demand.

Chapter 2

Introduction to TVD

TVD stands for "Total Variation Diminishing". In fluid dynamics, solving the Navier-Stokes (NS) equations is challenging. Such equations can be solved analytically in few idealized cases, but they pose extreme challenges when applied to real-world scenarios, because of the scenarios' nonlinearity and complexity. In these scenarios, the equations are then discretized in attempt to obtain a numerical approximation. Some well-known discretization schemes include the first-order accurate upwind difference scheme (UDS) the second-order accurate central difference scheme (CDS) and the third-order accurate quadratic upstream interpolation for convective kinematics (QUICK) scheme [2].

However, researchers in the CFD field encountered difficulties in the spatial discretization of the convective term in the NS equations. It was observed that, although low-order schemes such as UDS are always conservative and thus able to generate stable iterative results, the results exhibit enormous numerical diffusion [2]. In turbulent flows, false diffusion leads to unphysical results [21] [22]. With grid refinement, the problem of numerical diffusion remitted, with an extensive increase in the computational cost [2]. High-order schemes such as the QUICK scheme cause dispersion and undesirable oscillations in the solutions [2]. To eliminate the spurious oscillations, the TVD schemes are introduced.

Versteeg et al. [2] modelled a two-dimensional source-free diffusion-free property ϕ, to compare the results of the Van Leer TVD scheme to the QUICK scheme, the UDS and the exact solution. The rectangular computational domain is 1 m $\times$ 1 m in size and its schematic is presented in Figure 2.1. The authors [2] set the velocity field to be uniform and diagonal on a 50 x 50 uniform grid. Property ϕ has a value of 100 along the left and upper boundaries, and a value of zero along the right and bottom boundaries.

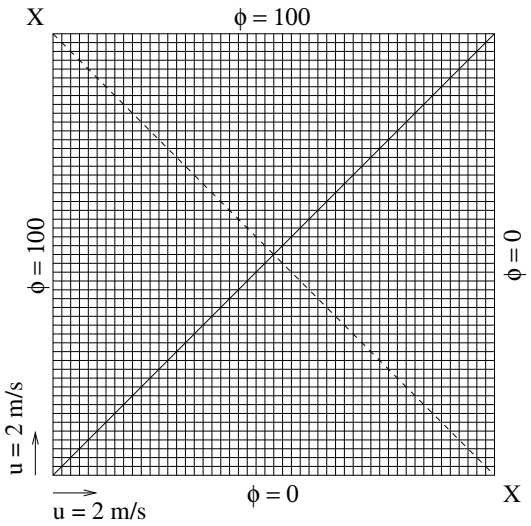


Figure 2.1: Flow domain for false diffusion illustration, adapted from [2]

The ϕ profile is plotted along the X-X diagonal line in Figure 2.2. Since there is no physical diffusion, property ϕ is expected to have a sharp transition from 100 to zero, as illustrated by the exact solution. From the figure, the TVD scheme produces the most reliable result compared to the exact solution. The result from the QUICK scheme wiggles near the interface and the result from the UDS shows unrealistic smearing, which represents the artificial diffusion introduced by numerical approximations.

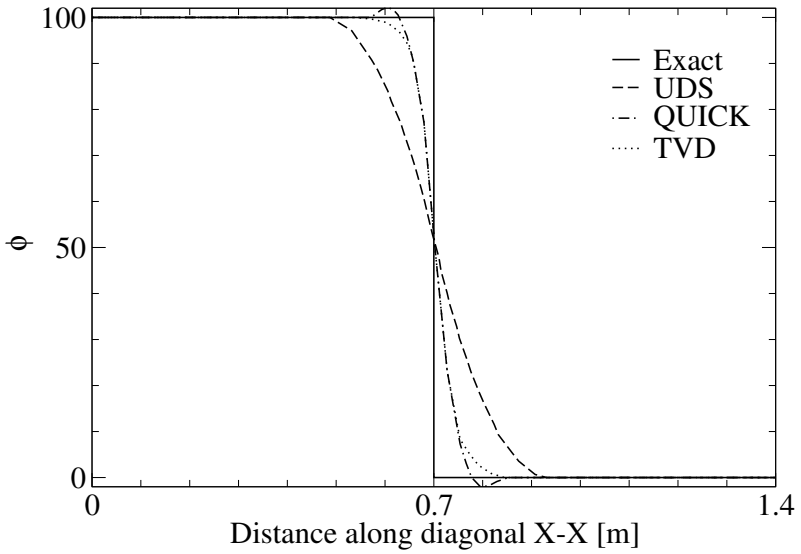


Figure 2.2: Comparison of numerical solutions obtained from TVD schemes with the QUICK results, UDS results and the exact solution respectively on a 50 x 50 grid, adapted from [2]

2.1 Background of TVD Schemes

Harton [23] first introduced the concept of TVNI (total variation non-increasing), which is later known as TVD. In general, TVD refers to a class of methods that ensures numerical stability and avoids unphysical wiggling in numerical solutions. Total variation is defined as the sum of differences between adjacent discretized points [24]. For example in Figure 2.3, the total variation of a transport property ϕ can be written as Equation 2.1.

$$TV(\phi) = |\phi_2 - \phi_1| + |\phi_3 - \phi_2| + |\phi_4 - \phi_3| + |\phi_5 - \phi_4| = |\phi_3 - \phi_1| + |\phi_5 - \phi_3| \quad (2.1)$$

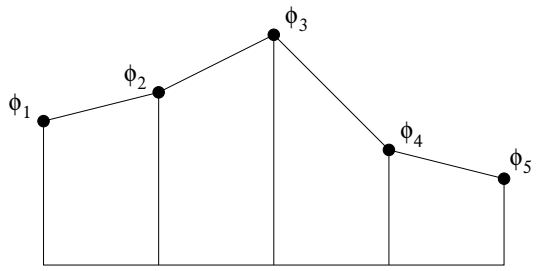


Figure 2.3: Example of discretized points for TV illustration, adapted from [2]

Harten [23] stated that a property must satisfy monotonicity preservation, to be oscillation-free. This indicates that the property must not create new local extremas and any existing extremas must not increase in magnitude [23]. For a property to satisfy monotonicity preserving, its total variation must not increase [23] [24].

Nikrityuk [25] conducted a comparative analysis of different methods on solving the convective term in the one-dimensional convection-diffusion equation. The numerical solutions were compared to the exact solution. With a homogeneous velocity field, the solutions obtained by implicit TVD schemes are oscillation free regardless of the number of control volumes. At a hundred control volumes, the TVD solution coincides with the analytical solution. In another test case with a non-homogeneous sinusoidal velocity field, when the CDS fails to capture the trend exhibited by the analytical solution, TVD schemes successfully tracks the step changes in the property of interest.

The most significant advantage of TVD schemes is that they avoid undesirable wiggles in the numerical solution while maintaining the minimum degree of smearing in the solution obtained when high-order schemes are used. The solutions obtained with TVD schemes are generally more realistic and reliable compared with which are achieved with other schemes.

2.2 Applications of TVD Schemes

TVD schemes are commonly used in solving problems that involve discontinuities. It had first found its applications in gas dynamics to simulate pressure, density and velocity discontinuities [23]. Sandham and Yee [26] discovered that TVD schemes can be used to solve high-speed turbulent flow and compressible mixing layer problems. The authors [26] suggested that explicit TVD schemes captured the shocks, but the solution accuracy is dependent on the flux limiter functions being used. The flux limiter function is a feature that characterizes a discretization scheme and gives information on whether a method is TVD or not. The criteria for a scheme to be TVD will be discussed in detail in Chapter 3.

Nandi and Date [1] [27] formulated and validated a fully implicit method for simulation of flows with interface. The authors used the "single fluid formalism", which treated two immiscible fluids as a single fluid with an abrupt change in fluid properties at the interface, to simulate the multiphase flow behaviour. They used the TVD scheme to evaluate the convective terms in the convection-diffusion equation and had thus managed to decrease the degree of smearing at the interface.

Inspired by Nandi and Date [1] [27], Pirker [28] used experimental methods to further investigate the sloshing resonance phenomena. Wang et al. [29] cited for the stability of the implicit methods in their article regarding the modelling of gas storage in underground aquifers. The authors did not apply the TVD schemes to solve the associated governing equations; instead, they used CDS on a non-uniform grid. However, more in-depth future studies in this area may involve the use of TVD schemes.

In addition, TVD schemes have applications in material science and engineering to model heat flow [25]. Date [30] concluded that TVD schemes result in a reduction in numerical diffusion when temperature discontinuity is present.

2.3 TVD Validation Cases

Validation is an essential step in the numerical study process. By comparing the obtained simulation results to the analytical solutions, the experimental results, or simulation results acquired by other authors using different approaches, the validity of the algorithms, as well as the quality of the computer code prepared to effectuate the algorithms, can be checked.

Nandi and Date [1] illustrated potential applications of TVD in six research problems: one regarding liquid-liquid interface, and the others regarding liquid-gas (air) interface [1]. These research problems include the large-scale Rayleigh-Taylor instability, dam break, and sloshing problems; as well as some small-scale problems involving splashing droplet on a liquid surface, bursting bubble through a water-air interface and merging of bubbles surrounded by liquid [1]. The authors adopted single machine precision for two-dimensional problems and double machine precision for three-dimensional problems.

The authors [1] compared their simulation outcomes to previous results employing different schemes and the available experimental data. The conclusions are summarized in Table 2.1. It is shown that TVD predictions are in close agreement with the experiment results. In three-dimensional cases, TVD results in significantly less error in comparison with the level-set method developed by Takahira et al. [31]. The details of the dam break problems and bubble-in-liquid problems are elaborated in the subsections. Similar cases analyzed by other researchers are also presented. A range of input properties and the associated outputs are discussed in detail.

2.3.1 Dam-Break Problems

Three major types of dam break problems were of interest to the researchers: two-dimensional idealized dam-break, two-dimensional dam-break in channels, and three-dimensional partial dam-break. The researchers also investigated some other relevant two-dimensional problems, including dam-break in converging-diverging channels and the oblique hydraulic jump problem. Results from these studies were summarized in Table 2.2 at the end of the dam-break section, and the details can be found in the subsections.

Table 2.1: Summary of TVD validation cases, adapted from [1]

Problem	**Dimensions**	**Precision**	**Maximum Error**	**Comparison Literature**	**Comparison Method(s)**
Rayleigh Taylor	2D	Single	0.46%	Rudman [32]	Volume Tracking
Collapse of Water Column	2D	Single	0.43%	Jun & Spalding [33]	Explicit Van Leer
			N/A	Martin & Moyce [34]	Experiment
Sloshing	2D	Single	0.45%	Andrillon & Alessandrini [35]	Experiment
Splashing	2D	Single	0.002%	Puckett [36]	VOF
Bursting	3D	Double	0.078%	Takahira [31]	Level-set
Merging	3D	Double	0.054%	Takahira [31]	Level-set

Two-Dimensional Idealized Dam-Break

A schematic of this type of problem is shown in Figure 2.4, in which h stands for the water column height and b is the water column width. Nandi and Date [1] evaluated this problem using their formulation with the assumption of both fluids (water and air) being incompressible. The simulation domain was 4×2.2 m^2 consisting of 40×22 control volumes. The height and width of the water column are 2 m and 1 m respectively. Time step of 0.001 s was used. The gas to liquid density ratio was 1.205 : 998.1 and the viscosity ratio was 101 : 1.81 [1]. The volume fraction of the two fluids was tracked. With the Lin-Lin flux limiter, the authors obtained a solution with 0.43% relative error compared with the solutions obtained with the explicit Van Leer TVD approach [33].

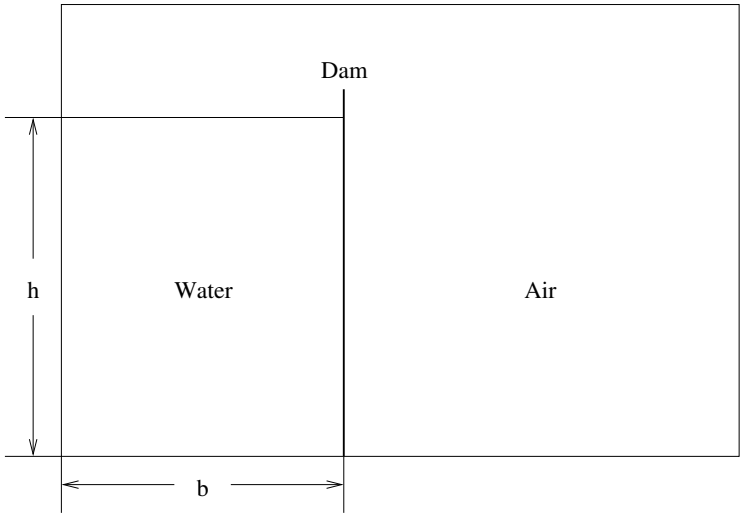


Figure 2.4: Schematic of two-dimensional idealized dam-break, adapted from [1]

Chang et al. [4] took the compressibility of air into consideration and conducted a sensitivity study on the compressibility parameter β. Initially, the water column is 10 m in height and 500 m in width. The grid number was refined to 401×61 to achieve the grid-independent results. Different TVD limiter functions, including the Min-Mod, the SUPERBEE and the monotonic upwind scheme for conservation laws (MUSCL) functions, were tested and the results obtained were compared [4]. An exact solution was obtained in this simplified case, and the relative errors were calculated as the means of comparison. The authors [4] concluded that the SUPERBEE method produced the least relative error and thus the most accurate results.

Two-Dimensional Dam-Break in Channels

A generalized side view of this type of problem is presented in Figure 2.5. Wood and Wang [3] used the second-order explicit MacCormack TVD scheme to add an artificial diffusion into the algorithm, to eliminate wiggles in the evaluation of the shallow water equations (SWE). Experiments were performed prior to the simulation, and the simulation domain was fitted to the experimental dimensions [3]. Initially, the water column height was 0.889 m and the water depth was set to 2.54×10^{-4} m in the channel [3]. The computational domain covers a 4.9×2.7 m^2 area, and local grid refinement was employed in the channel and near the dam (the water intake) in the water reservoir [3]. Compared to the experimental results, the authors overestimated water depth in the channel upstream of the channel bend, while underestimating it around the channel bend [3].

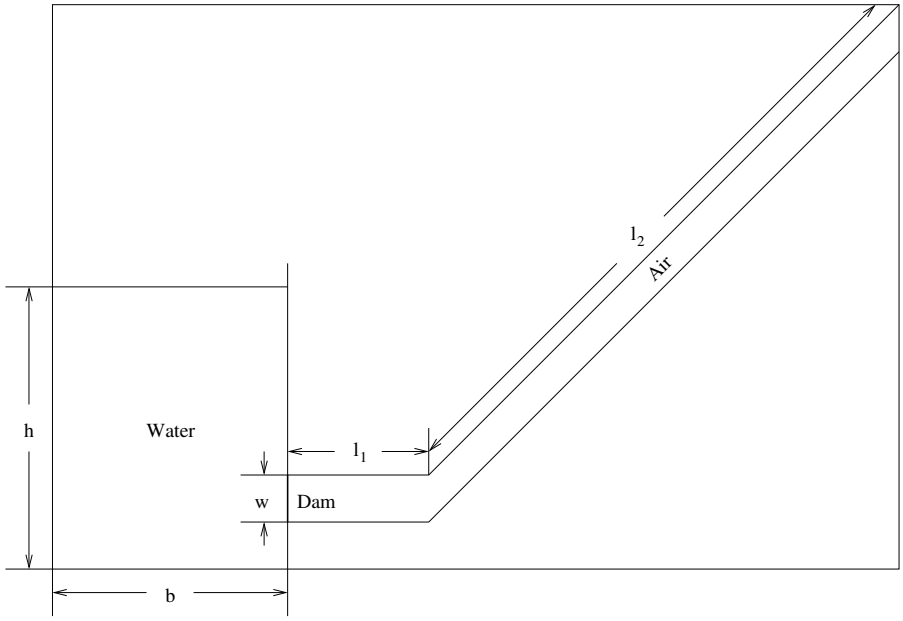


Figure 2.5: Side view of two-dimensional dam-break in channels, adapted from [3]

A combination of the idealized dam-break problem and the dam-break in channels problem gives a problem of dam-break onto a slope. In Chang's [4] work, a reservoir of size 2.25×0.25 m^2 was considered. A total of 401×51 grids were simulated. The air and water densities were 1.23 kg/m^3 and 10^3 kg/m^3 respectively, and the corresponding viscosities were 1.78×10^{-5} Ns/m^2 and 1.34×10^{-3} Ns/m^2. The SUPERBEE TVD method was used. The authors [4] concluded that the simulated water depth variation over time was smoother than the measurements. This can be due to excessive numerical dissipation in the computation [4].

Three-Dimensional Partial Dam-Break

A generalized top view of this type of problem is presented in Figure 2.6. Chang et al. [4] considered a symmetrical 0.4 m breach on a $3 \times 2 \times 1.2$ m^3 domain, where the reservoir has a dimension of $1 \times 2 \times 0.6$ m^3. A total of $151 \times 101 \times 51$ grids was employed. The SUPERBEE TVD flux limiter was used. The simulation results show reasonable agreement with previous experimental and simulation results [4].

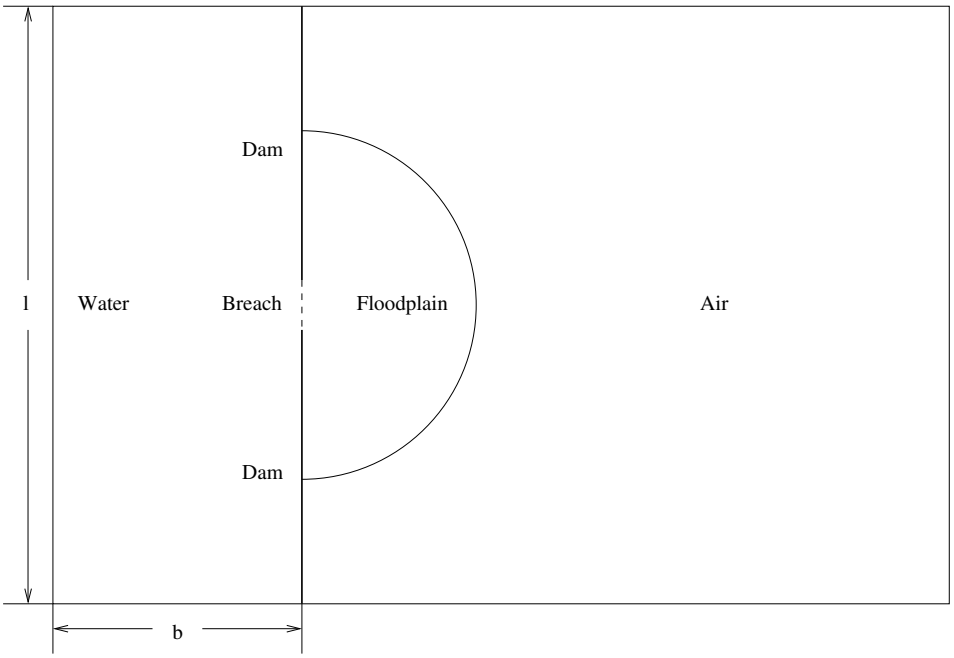


Figure 2.6: Top view of three-dimensional partial dam-break, adapted from [4]

Song et al. [37] modeled a problem with a non-symmetrical 75 m breach on a $200 \times 200 \times 10$ m^3 domain. The unstructured Delaunay triangular meshes were used, and a total of 7394 triangles were included in the simulation [37]. The reservoir width was 100 m, and the dam width was 15 m. The breach was 75 m in length, and the top of the breach was 30 m apart from the top of the reservoir boundary. The roughness of the walls was neglected. Initially, the water depth was 10 m upstream of the dam and 5 m downstream of the dam. The simulation results were plotted in 3D as well as in a contour plot of the water depth. The authors [37] concluded qualitatively that the results agreed with the results obtained from previous simulations. A more complex case, which introduces a rectangular downstream barrier, was also analyzed. In this scenario, the flow detours around the barrier.

Table 2.2: Summary of TVD validation cases in dam-break problems

Author(s)	**2D/3D**	**Compute Domain**	**Grid Specs**	**Time Step**	ρ_g/ρ_l	μ_g/μ_l	**Flux Limiter**	**Error**
Two-Dimensional Idealized Dam-Break								
Nandi & Date [1]	2D	4×2.2 m^2	40×20	0.001 s	1.205 : 998.1	1.81 : 101	Lin-Lin	0.43%
Chang et. al [4]	2D	500×10 m^2	401×61	N/A	1.23 : 1000	1.78 : 134	MUSCL	1.22%
Two-Dimensional Dam-Break in Channels								
Wood & Wang [3]	2D	4.9×2.7 m^2	N/A	0.001 s	N/A	N/A	MacCormack	N/A
Chang et. al [4]	2D	2.25×0.25 m^2	401×51	N/A	1.23 : 1000	1.78 : 134	SUPERBEE	N/A
Three-Dimensional Partial Dam-Break								
Chang et. al [4]	3D	$3 \times 2 \times 1.2$ m^3	$151 \times 101 \times 51$	N/A	1.23 : 1000	1.78 : 134	SUPERBEE	N/A
Song et. al [37]	3D	$200 \times 200 \times 10$ m^3	7394	N/A	N/A	N/A	Runge-Kutta	N/A

2.3.2 Bubble-in-Liquid Problems

In this section, the two case studies performed by Nandi and Date [1] involving bubbles (three-dimensional bubble bursting and bubble merging), a two-dimensional problem of a single rising bubble, as well as another two-dimensional problem considering merging and bursting simultaneously, are discussed. The schematics of initial conditions of these problems are presented in Figure 2.7 from (a) to (d) respectively.

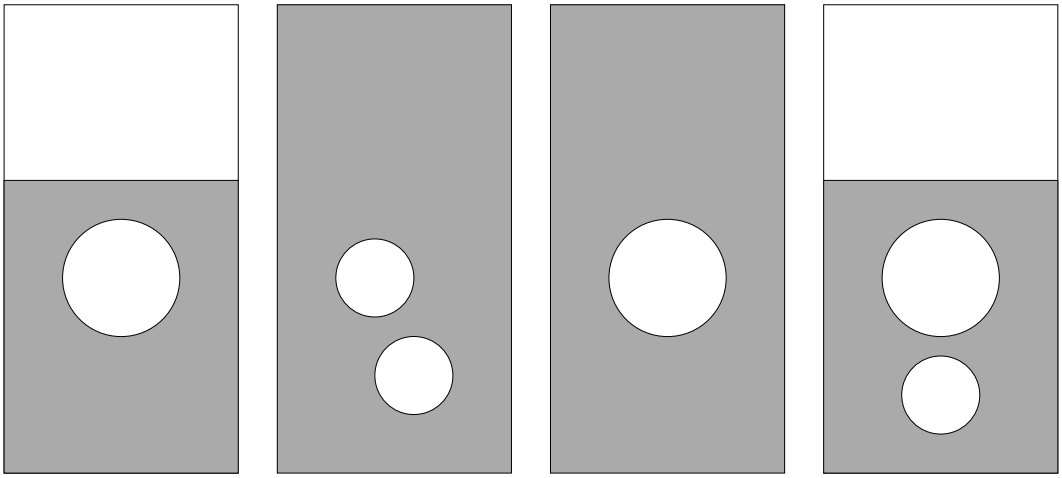


Figure 2.7: Schematics of initial conditions in bubble-in-liquid problems, adapted from (a) [1] (b) [1] (c) [5] (d) [6] (the schematics are not to scale)

Three-Dimensional Bursting of Bubble

Nandi and Date [1] used the fully implicit Lin-Lin TVD scheme to study the bursting of a gas bubble at the liquid-gas interface. The computational domain was $6 \times 6 \times 12$ U^3 and the mesh dimensions were $30 \times 30 \times 60$. The liquid-gas contact surface was located at 4 U (one-third of the domain was filled with liquid). A bubble with a radius of $r = 1$ U was placed at $(3, 3, 2.8)$ U initially. The time step used was $\Delta t = 5.645 \times 10^{-4}$ s, or dimensionless time step $\Delta\tau = 0.001$. The dimensionless time is defined as $\tau = t \times \frac{U_{ref}}{D}$ where $U_{ref} = \sqrt{0.64gD}$. At the onset of the simulation, the bubble rose due to the buoyancy force. The bubble burst once it reached the free liquid surface; ripples and water neck were observed on the liquid surface subsequently. In the simulation, gas to liquid density and viscosity ratios were set to 0.001 and 0.01 respectively. The maximum volume error was found to be 0.078%. The authors observed improvement in results over results from previous work carried by Takahira et al. [31].

Three-Dimensional Merging of Rising Bubbles

Nandi and Date [1] also validated their TVD formulation with the study of the three-dimensional merging of rising bubbles. The computational domain was $4 \times 4 \times 12$ U^3 and the mesh size was $30 \times 30 \times 90$. There was no free water surface. At the beginning of the simulation, the two bubbles were located at $(0.25, 0, -4.5)$ U and $(-0.25, 0, -2.3)$ U. The time step used was $\Delta t = 4.516 \times 10^{-4}$ s, or dimensionless time step $\Delta\tau = 0.001$. The dimensionless time is defined as $\tau = t \times \frac{U_{ref}}{D}$ where $U_{ref} = \sqrt{gD}$. The gas to liquid density and viscosity ratios tested were 0.001 and 0.01816 respectively. In this case, the maximum volume error was 0.054%, which was improved significantly compared with the error of 2% obtained by Takahira et al. [31], while using a more refined grid of $50 \times 50 \times 150$.

Two-Dimensional Rising bubble

Huang and Zhang [6] investigated a two-dimensional axisymmetric rising bubble problem with two TVD schemes: the SUPERBEE TVD scheme and the Min-Mod TVD scheme. The gas to liquid density ratio was 0.001, and the viscosity ratio was 0.01 [6]. It was observed that the down face velocity of the bubble exceeded the up face velocity of the bubble, so the initially circular gas bubble deforms into a ring-shaped bubble over time. The computational domain tested was 6×10 U^2 and the bubble radius was 1 U. The grid resolution was 121×200 [6]. The maximum relative error for the volume of the bubble at time 1.8 s was 9.65%. The authors [6] concluded a good agreement of their work with the previous works.

Two-Dimensional Merging and Bursting of Rising Bubbles

Wang et al. [5] used the TVD Runge-Kutta scheme to solve the NS equations. The authors used the level-set method with ghost technique to capture the oil-water interface (with a density ratio of $1 : 1.205$) in the problem of merging bubbles [5]. A two-dimensional domain of 4×3 U^2 with 80×60 grids was considered and the time step was 0.002 s [5]. The top one-fourth of the computational domain was occupied by oil whereas the rest was occupied by water [5]. The two bubbles of different sizes were initially placed at the x-midline but one above the other in y. The top and bottom bubbles have radii of 0.5 and 0.4 respectively [5]. At time zero, the two bubbles started to rise and merge; and at time 0.25 s, the merged bubble burst at the oil-water interface. The authors compared the simulation results to results produced by the commercial CFD software "ANSYS Fluent" and had found close agreement between the two solutions [5].

Table 2.3: Summary of TVD validation cases in bubble-in-liquid problems

Author(s)	**2D/3D**	**Compute Domain**	**Grid Specs**	**Time Step, Δt**	ρ_g/ρ_l	μ_g/μ_l	**Flux Limiter**	**Error**
Nandi & Date [1]	3D	$6 \times 6 \times 12$ U^3	$30 \times 30 \times 60$	5.645×10^{-4} s ($\Delta\tau = 0.001$)	0.001	0.01	Lin-Lin	0.078%
Nandi & Date [1]	3D	$4 \times 4 \times 12$ U^3	$30 \times 30 \times 90$	4.516×10^{-4} s ($\Delta\tau = 0.001$)	0.001	0.01816	Lin-Lin	0.054%
Huang & Zhang [6]	2D	6×10 U^2	121×200	N/A	1 : 1000	1 : 100	SUPERBEE Min-Mod	9.65%
Wang et. al [5]	2D	4×3 U^2	80×60	0.002 s	1 : 1.205	N/A	Runge-Kutta	N/A

2.3.3 Other Cases

Date [38] used the Lin-Lin [39] TVD scheme to predict compressible shocks. The author [38] considered a two-dimensional problem of flow through a plane channel with a bump on one wall on a non-staggered grid. It was concluded that when the UDS failed to predict transonic and supersonic flows accurately on a mesh of 116×33, the implicit TVD scheme successfully captured the shocks on a coarser mesh of 80×20.

In later years, Date [30] employed the Lin-Lin [39] TVD scheme again, to simulate gas flow in a two-dimensional convergent-divergent nozzle on an unstructured mesh. The Mach number and pressure were predicted. The pressure values were compared with the experimental values [40]. A R^2 value of 0.96 was observed for both pressure along the nozzle wall and pressure along the center line. Date [30] suggested that this type of simulation can be used to obtain a specific Mach number at the nozzle exit, thus facilitate nozzle designs.

Song et al. [37] solved the conservative form of the SWE to study the two-dimensional oblique hydraulic jump problem. The second-order MUSCL scheme was used for spacial discretization, and the TVD Runge-Kutta scheme was used for time discretization. The oblique hydraulic jump problem studies the increase in flow height at the deflected channel wall when a supercritical flow goes through a converging channel [41]. The unstructured Delaunay triangular meshes were employed on a 20 m $\times$ 10 m computational domain. One of the channel walls was deflected by a 8.95° angle. Initially, the fluid flowed at 1 m/s down the channel, and the flow depth was 1 m. The outlet boundary was a free-flow boundary, while the channel walls were free-slip solid boundaries. The simulation solutions were compared with the analytical solutions and errors of 0.07%, 0.2% and 5×10^{-3}% were observed in the hydraulic jump angle, the flow depth at the jump and the velocity across the jump respectively.

2.4 TVD Deficiencies

Despite all the advantages TVD schemes present, it has some drawbacks. Sandham and Yee [26] mentioned that TVD schemes fail to capture the vortex growth due to the "clipping phenomena". The "clipping phenomena" is a risk posed by TVD schemes, which smooths out most of the true extremas in the numerical solution [26]. The authors [26] suggested that the clipping effect affects the results significantly when the flow growth rate is sensitive to changes in the Reynolds number.

Other than the accuracy in the predictions, Yee et al. [42] mentioned that compared to the conventional implicit methods, implicit TVD scheme requires roughly 2.5 times more CPU time per time step. Lien and Leschziner [24] found that the UMIST TVD scheme requires 15% more CPU time than the QUICK scheme.

2.5 Objectives

The objective of the current study is to implement and validate the fully implicit TVD method developed by Nandi and Date [1] [27]. After the implementation, this software can be used to solve two-dimensional multiphase immiscible flow problems. This implementation facilitates future researches in the CFD field, especially in areas involving multiphase immiscible flows.

Besides, the current study distinguishes and compares various TVD schemes in terms of their formulations, the accuracy of results, as well as the required computational time.

Chapter 3

1D Formulation and Validation

3.1 Introduction to TVD Criteria

Two functions that are most commonly used to determine whether a scheme is TVD or not are the flux limiter function $\psi(r)$ [2] and the shape sensing function $f_c(\xi)$ [25]. These two functions can be visualized in two types of diagrams: the $r - \psi$ diagram and the normalized variable diagram (NVD).

Considering a standard one-dimensional control volume as shown in Figure 3.1, the central point and the adjacent points west and east of it are denoted by upper-case letters P, W and E. The cell faces are denoted by lower-case letters w and e. The size of the control volume is δx.

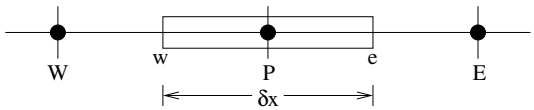


Figure 3.1: Example of one-dimensional control volume discretization around node P on a uniform grid

The independent variable r in the flux limiter function is the ratio between upwind gradient and downwind gradient (Equation 3.1) [2], where ϕ_U, ϕ_{UU} and ϕ_D are values of the transport property ϕ upstream, upstream of upstream and downstream of the cell face, respectively.

$$r = \frac{\phi_U - \phi_{UU}}{\phi_D - \phi_U} \tag{3.1}$$

As a result, the flux limiter function reveals the balance between upwind and downwind contributions. For flux in the positive direction, r is $(\phi_P - \phi_W)/(\phi_E - \phi_P)$; and the property value at the east face ϕ_e can be expressed in the generalized form:

$$\phi_e = \phi_P + \frac{1}{2}\psi(r) \times (\phi_E - \phi_P) \tag{3.2}$$

The $r-\psi$ diagram that displays the relationship between the flux limiter function ψ and the variable r as defined in Equation 3.1, is presented in Figure 3.2. Sweby [43] suggested that a TVD scheme must satisfy $\psi \leq 2r$ in the range $0 < r < 1$ and $\psi \leq 2$ in the range $r \geq 1$. It can be seen that UDS is unconditionally TVD; therefore solutions obtained by UDS are always bounded. Boundedness means that the value of ϕ stays within the boundary values in the absence of source; in other words, the normalized ϕ value belongs to $[0, 1]$ when the source term equals to zero.

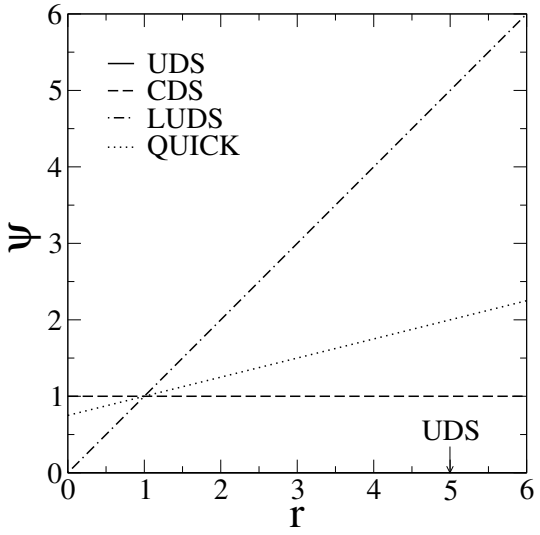


Figure 3.2: $r - \psi$ diagram for conventional schemes

However, UDS's low (first) order of accuracy causes large numerical diffusion. Sweby [43] suggested that the $r - \psi$ diagram also gives information on the order of accuracy of a scheme. The scheme is of second-order accurate if its $r - \psi$ curve lies between $\psi = 1$ and $\psi = r$, and passes through the point $(1, 1)$. Several high-order TVD schemes are presented in Figure 3.3. According to Sweby's criteria, all the schemes presented in Figure 3.3 are second-order accurate.

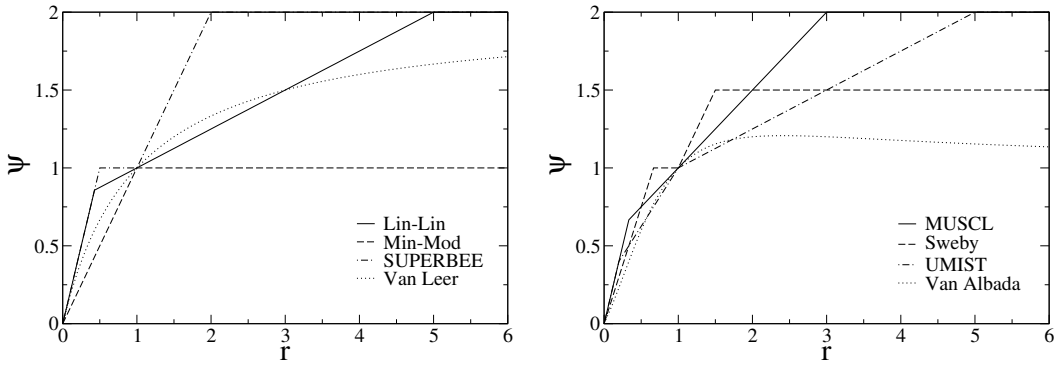


Figure 3.3: $r - \psi$ diagram for TVD schemes

Similarly, the independent variable ξ in the shape sensing function is the ratio between upwind gradient and overall gradient (Equation 3.3) [27].

$$\xi = \frac{\phi_U - \phi_{UU}}{\phi_D - \phi_{UU}} \tag{3.3}$$

For flux in the positive direction, ξ is $(\phi_P - \phi_W)/(\phi_E - \phi_W)$; and the property value at the east face ϕ_e can be expressed in the generalized form:

$$\phi_e = \phi_P + f_c(\xi) \times (\phi_E - \phi_W) \tag{3.4}$$

The NVD is a plot of normalized face value ξ_f over ξ. ξ_f is defined as $(\phi_f - \phi_{UU})/(\phi_D - \phi_{UU})$ and it is the sum of ξ and $f_c(\xi)$. A TVD scheme must satisfy $\xi \leq \xi_f \leq 1$ in the region $0 \leq \xi \leq 1$. For $\xi < 0$ or $\xi > 1$, a TVD scheme must have ξ_f equals to ξ; in other word, only UDS is bounded in in such regions. A NVD for a list of conventional schemes is plotted in Figure 3.4.

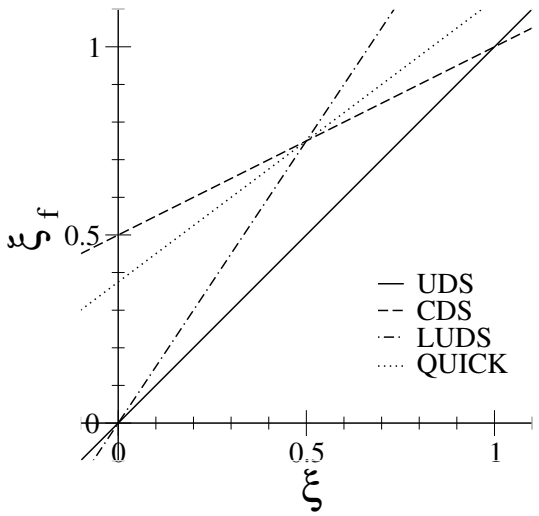


Figure 3.4: NVD curves for conventional schemes

Leonard [44] stated that for a uniform grid, a scheme with NVD curve passing through the point $(0.5, 0.75)$ is at least second-order accurate; if the scheme's NVD curve passes through the point with a slope of 0.75, the scheme is third-order accurate. Some high-order TVD schemes are presented in Figure 3.5. According to Leonard's criteria, all the schemes presented in Figure 3.5 are second-order accurate, except for the Lin-Lin scheme, which is third-order accurate.

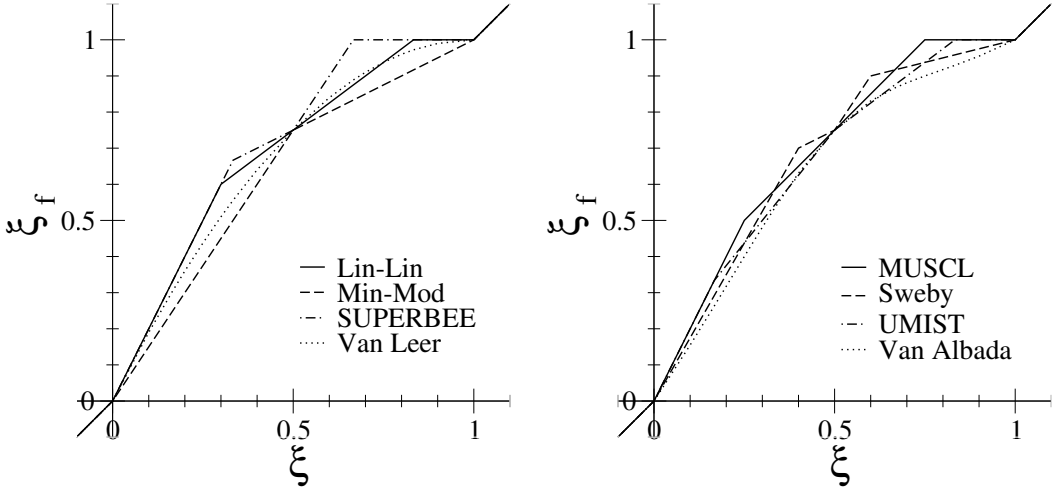


Figure 3.5: NVD curves for TVD schemes

A systematic comparison between the two characteristic functions was not conducted previously. In fact, the shape sensing function f_c and the flux limiter function ψ can be related by Equation 3.5 and the corresponding independent variables r and ξ can be converted using Equation 3.6. Formulations of the shape sensing function and the flux limiter function for various schemes and conditions are summarized in Table 3.1.

$$\psi = \frac{2f_c(\xi)}{1-\xi} \tag{3.5}$$

$$r = \frac{\xi}{1-\xi} \tag{3.6}$$

Table 3.1: Conversion from shape sensing functions to flux limiter functions

Schemes or Conditions	$f_c(\xi)$	$\psi(r)$
UDS	0	0
CDS	$\frac{1}{2} - \frac{\xi}{2}$	1
LUDS	$\frac{\xi}{2}$	r
QUICK	$\frac{3}{8} - \frac{\xi}{4}$	$\frac{3}{4} + \frac{r}{4}$
TVD Bound for $0 < r < 1$ [43]	ξ	$2r$
TVD Bound for $r \geq 1$ [43]	$1 - \xi$	2

TVD schemes are developed such that it stays within the TVD boundaries presented in Table 3.1 under all circumstances. The detailed formulation of a number of TVD schemes in terms of both the shape sensing functions and the flux limiter functions are listed in Table 3.2.

Table 3.2: Conversion from shape sensing functions to flux limiter functions for TVD schemes

Schemes	$f_c(\xi)$		$\psi(r)$
Lin-Lin [27]	$0,$	if $\xi \leq 0$	$max[0, min(2r, \frac{3}{4} + \frac{r}{4}, 2)]$
	$\xi,$	if $0 < \xi \leq \frac{3}{10}$	
	$\frac{3}{8} - \frac{\xi}{4},$	if $\frac{3}{10} < \xi \leq \frac{5}{6}$	
	$1 - \xi,$	if $\frac{5}{6} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
Min-Mod [2]	$0,$	if $\xi \leq 0$	$max[0, min(r, 1)]$
	$\frac{\xi}{2},$	if $0 < \xi \leq \frac{1}{2}$	
	$\frac{1}{2} - \frac{\xi}{2},$	if $\frac{1}{2} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
SUPERBEE [2]	$0,$	if $\xi \leq 0$	$max[0, min(2r, 1), min(r, 2)]$
	$\xi,$	if $0 < \xi \leq \frac{1}{3}$	
	$\frac{1}{2} - \frac{\xi}{2},$	if $\frac{1}{3} < \xi \leq \frac{1}{2}$	
	$\frac{\xi}{2},$	if $\frac{1}{2} < \xi \leq \frac{2}{3}$	
	$1 - \xi,$	if $\frac{2}{3} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
Van Leer [2]	$0,$	if $\xi \leq 0$	$\frac{r+\|r\|}{1+r}$
	$\xi - \xi^2,$	if $0 < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
MUSCL [2]	$0,$	if $\xi \leq 0$	$max[0, min(2r, \frac{r}{2} + \frac{1}{2}, 2)]$
	$\xi,$	if $0 < \xi \leq \frac{1}{4}$	
	$\frac{1}{4},$	if $\frac{1}{4} < \xi \leq \frac{3}{4}$	
	$1 - \xi,$	if $\frac{3}{4} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
Sweby [2]	$0,$	if $\xi \leq 0$	$max[0, min(\beta r, 1), min(r, \beta)]$
	$\frac{\beta\xi}{2},$	if $0 < \xi \leq \frac{1}{\beta+1}$	
	$\frac{1}{2} - \frac{\xi}{2},$	if $\frac{1}{\beta+1} < \xi \leq \frac{1}{2}$	
	$\frac{\xi}{2},$	if $\frac{1}{2} < \xi \leq \frac{\beta}{\beta+1}$	
	$\frac{(1-\xi)\beta}{2},$	if $\frac{\beta}{\beta+1} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
UMIST [2]	$0,$	if $\xi \leq 0$	$max[0, min(2r, \frac{1}{4} + \frac{3r}{4}, \frac{3}{4} + \frac{r}{4}, 2)]$
	$\xi,$	if $0 < \xi \leq \frac{1}{6}$	
	$\frac{1}{8} + \frac{\xi}{4},$	if $\frac{1}{6} < \xi \leq \frac{1}{2}$	
	$\frac{3}{8} - \frac{\xi}{4},$	if $\frac{1}{2} < \xi \leq \frac{5}{6}$	
	$1 - \xi,$	if $\frac{5}{6} < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	
Van Albada [2]	$0,$	if $\xi \leq 0$	$\frac{r+r^2}{1+r^2}$
	$\frac{\xi-\xi^2}{2+4\xi^2-4\xi},$	if $0 < \xi \leq 1$	
	$0,$	if $\xi \geq 1$	

3.2 Formulation of the Convection-Diffusion Equation

A variety of numerical schemes were evaluated considering discretization of the one-dimensional steady-state convection-diffusion Equation 3.7; where ρ is the fluid density, u is the flow velocity, ϕ is the property of interest, x is the length in the flow direction, and Γ is the diffusivity. The problem is visualized in Figure 3.6. Defining the product of fluid density and velocity at a specified cell face as flow flux f, namely, $f = \rho u$, Equation 3.7 can be expressed into Equation 3.8 using a control volume approach.

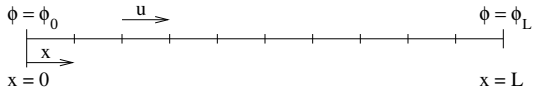


Figure 3.6: Convection-diffusion problem simulation domain with boundary conditions implemented in code

$$\rho u \frac{\partial \phi}{\partial x} = \Gamma \frac{\partial^2 \phi}{\partial x^2} + Q \tag{3.7}$$

$$f_e \phi_e - f_w \phi_w = (\Gamma \frac{\partial \phi}{\partial x})_e - (\Gamma \frac{\partial \phi}{\partial x})_w + Q\Delta x \tag{3.8}$$

The second-order partial differential diffusive term $\Gamma \frac{\partial^2 \phi}{\partial x^2}$ is discretized using CDS as a standard operation [30]. The convective term $\rho u \frac{\partial \phi}{\partial x}$ were discretized using CDS, UDS, LUDS, QUICK and TVD. The source term is ignored.

Coding was carried out using shape sensing functions. The deferred correction was utilized to linearize the discretized equations while moving the non-linear terms to the source term. This can be illustrated using Equation 3.4. In Equation 3.4, ϕ_P was treated as the linear term that represents the contribution from the first-order upwind scheme. Meanwhile, the non-linear term $f_c(\xi) \times (\phi_E - \phi_W)$ was moved to the source term. The solution was then obtained iteratively. The deferred correction was utilized to ensure main diagonal dominance in the solution matrix, thus promotes convergence of the solution.

Moving all terms in Equation 3.8 to the same side of equation gives Equation 3.9. Discretization of individual terms in the equation are presented in Equations 3.10 to 3.12.

$$f_e \phi_e - f_w \phi_w - ((\Gamma \frac{\partial \phi}{\partial x})_e - (\Gamma \frac{\partial \phi}{\partial x})_w) = 0 \tag{3.9}$$

$$
\begin{aligned}
f_e\phi_e =&(\phi_P + f_{ce}^{+}(\phi_E - \phi_W))AMAX(f_e, 0) \\
&- (\phi_E - f_{ce}^{-}(\phi_{EE} - \phi_P))AMAX(-f_e, 0) \qquad (3.10) \\
f_w\phi_w =&(\phi_W + f_{cw}^{+}(\phi_P - \phi_{WW}))AMAX(f_w, 0) \\
&- (\phi_P - f_{cw}^{-}(\phi_E - \phi_W))AMAX(-f_w, 0) \qquad (3.11) \\
(\Gamma\frac{\partial\phi}{\partial x})_e =&\Gamma_e\frac{\phi_E - \phi_P}{x_E - x_P} \quad \& \quad (\Gamma\frac{\partial\phi}{\partial x})_w = \Gamma_w\frac{\phi_P - \phi_W}{x_P - x_W} \qquad (3.12)
\end{aligned}
$$

With deferred correction, Equation 3.9 can be linearized into:

$$
A_E\phi_E + A_P\phi_P + A_W\phi_W = b_P \qquad (3.13)
$$

Denoting the convective contribution to coefficients as A^c and the diffusive contribution to coefficients as A^d. The coefficients A_W, A_P and A_E in Equation 3.13 are then the sums of A^c and A^d. The term b_p is the artificial source term induced by deferred correction. Plugging Equations 3.10 to 3.12 into Equation 3.9 and organizing equation gives:

$$
A_E^c = - AMAX(-f_e, 0),\ A_W^c = -AMAX(0, f_w) \quad \& \quad A_P^c = -(A_E^c + A_W^c) \qquad (3.14)
$$

$$
A_E^d = -\frac{\Gamma_e}{x_E - x_P},\ A_W^d = -\frac{\Gamma_w}{x_P - x_W} \quad \& \quad A_P^d = -(A_E^d + A_W^d) \qquad (3.15)
$$

$$
\begin{aligned}
b_p =&f_{ce}^{+}(\phi_W - \phi_E)AMAX(f_e, 0) - f_{ce}^{-}(\phi_{EE} - \phi_P)AMAX(-f_e, 0) \\
&+ f_{cw}^{+}(\phi_P - \phi_{WW})AMAX(f_w, 0) - f_{cw}^{-}(\phi_W - \phi_E)AMAX(-f_w, 0) \qquad (3.16)
\end{aligned}
$$

Note that for UDS, the source term b_p equals to zero since the shape sensing function has a value of zero.

Fixed value boundary conditions (the Dirichlet boundary conditions) was used. Solution matrix contains only internal nodes. At the west-most internal node, ϕ_W is known and is moved to the source term. Similarly, at the east-most internal node, ϕ_E is known and is moved to the source term. Supposing there are N control volumes, there would be $N + 1$ nodes in total and $N - 1$ internal nodes. Mark the nodes with subscript number from 1 to $N + 1$; ϕ_1 and ϕ_{N+1} are known and ϕ_2 to

ϕ_N are to be calculated. The solution matrix has the form:

$$\left[\begin{array}{ccccccccc|c} A_{P,2} & A_{E,2} & 0 & 0 & ... & 0 & 0 & 0 & 0 & \phi_2 \\ A_{W,3} & A_{P,3} & A_{E,3} & 0 & ... & 0 & 0 & 0 & 0 & \phi_3 \\ 0 & A_{W,4} & A_{P,4} & A_{E,4} & ... & 0 & 0 & 0 & 0 & \phi_4 \\ & & & & ... & & & & & ... \\ 0 & 0 & 0 & 0 & ... & A_{W,N-2} & A_{P,N-2} & A_{E,N-2} & 0 & \phi_{N-2} \\ 0 & 0 & 0 & 0 & ... & 0 & A_{W,N-1} & A_{P,N-1} & A_{E,N-1} & \phi_{N-1} \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & A_{W,N} & A_{P,N} & \phi_N \end{array}\right] = \left[\begin{array}{c} b_{P,2} - A_{W,2}\phi_1 \\ b_{P,3} \\ b_{P,4} \\ ... \\ b_{P,N-2} \\ b_{P,N-1} \\ b_{P,N} - A_{E,N}\phi_{N+1} \end{array}\right] \tag{3.17}$$

The code performing the calculations were written in FORTRAN and MATLAB and are attached in Appendix A and Appendix B. The user has the option to select from either uniform or progressively changing grid width by modifying the expansion factor. With the expansion factor equals to one, the grid is uniform. With the expansion fact less than one, the grid size decreases at a constant rate; similarly, with the expansion factor greater than one, the grid size increases at a constant rate. The expansion factor has to be positive. Other required input parameters are: the output file name; fluid density, velocity and diffusivity; boundary values; discretization scheme for the convective term; minimum and maximum x; total number of nodes and the matrix solver. Four matrix solvers, successive over-relaxation (SOR), Gauss-Seidel (GS), Jacobi and Tri-Diagonal Matrix Algorithm (TDMA) are available. The output file contains the x value, ϕ value, exact solution and residual at every nodal point. It also provides information on matrix iteration and convergence. A sample output file is attached in Appendix C.

3.3 Comparison of Results from Various Methods

In the current study, a reference case was generated to demonstrate the difference in results obtained from various methods. The tested fluid was air at standard conditions ($\rho = 1$ kg/m^3, $c_p = 1000$ J/kg·K and $k = 0.02$ W/m·K). The fluid flows from $x = 0$ m to $x = 1$ m at velocity equals to 0.001 m/s. The UDS, CDS, LUDS, QUICK and Lin-Lin TVD schemes were used to discretize the convective term. Boundary values were set to zero at $x = 0$ m and one at $x = 1$ m. Four control volume sizes were tested, corresponding to 11, 21, 41 and 81 nodal points uniformly distributed along the flow direction (expansion factor equals to one). TDMA solver was selected for the fastest convergence. The threshold residual for convergence was set to 10^{-15}.

Analytical solution of this problem is presented in Equation 3.18. ϕ_0 and ϕ_L are the known ϕ values at the boundaries. Pe is the Peclet number defined as $\rho u \Delta x/\Gamma$.

$$\phi(x) = \phi_0 + \frac{\exp(\frac{Pe \cdot x}{L})}{\exp(Pe) - 1}(\phi_L - \phi_0) \tag{3.18}$$

In Equation 3.18, Pe is the global Peclet number; in other words, Δx is the difference between x_{max} and x_{min}. The global Peclet number is the ratio between convective strength and diffusive strength [25]. Local Peclet number is calculated using the mesh size as Δx. The mesh size is equivalent to the size of a single control volume. With increasing mesh size, the local Pe number grows. In the reference case, the thermal diffusivity of air is 2×10^{-5}, thus the global Pe is 50. For 10, 20, 40 and 80 control volumes, the mesh sizes are 0.1, 0.05, 0.025 and 0.0125 respectively; the resulting local Pes are 5, 2.5, 1.25 and 0.625 accordingly.

Results from all schemes as well as from the analytical solution are plotted in Figure 3.7. The vertical axis ϕ^* is the normalized ϕ calculated by $(\phi - \phi_{min})/(\phi_{max} - \phi_{min})$ and the horizontal axis x^* is the normalized x calculated by $(x - x_{min})/(x_{max} - x_{min})$. For coarse grid containing 10 CVs, local Pe values are large, and the results obtained from CDS and QUICK scheme fluctuate wildly. Other schemes give non-fluctuating solutions, but the residual from UDS is significant. The TVD result is oscillation-free and gives the smallest error among results from all other schemes.

When the grid was refined to 20 CVs, errors from all methods dropped. Results from QUICK scheme becomes stable, while results from CDS keeps oscillating. When the grid was further refined to 40 and 80 CVs, all solutions became more accurate and oscillation-free.

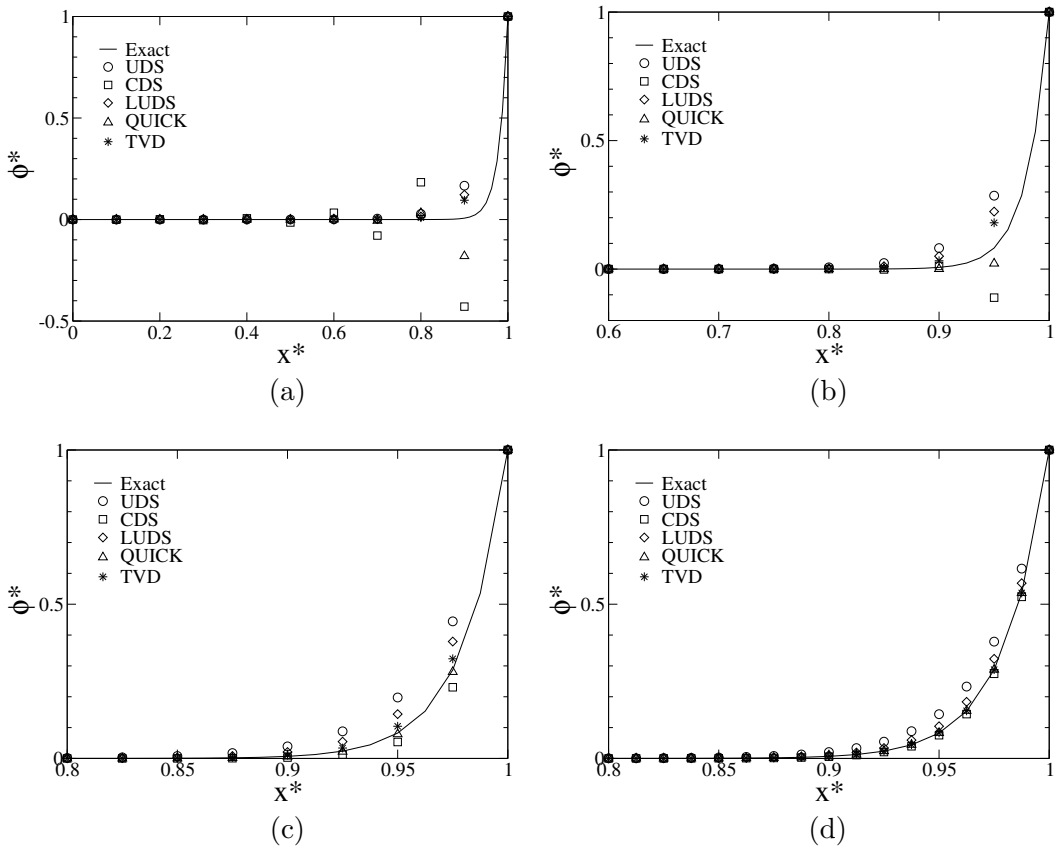


Figure 3.7: Normalized ϕ variations from UDS, CDS, LUDS, QUICK and TVD schemes for (a) 10 CVs (b) 20 CVs (c) 40 CVs (d) 80 CVs

The average relative error was plotted against the local Peclet number in Figure 3.8. Maximum relative error was not used because it always appears when the exact solution firstly switches from zero to a non-zero but extremely small value, and it is not representative of the actual deviation of the prediction from the exact solution. To avoid infinitely large errors, a small number was added to the numerator. The small number was set to be 10^{-6} in the current studies. The average relative error formulation is presented in Equation 3.19, where N is the total number of nodal values in the domain.

$$Error\ \% = \frac{1}{N} \sum \frac{|\phi_{exact} - \phi_{numerical}|}{\phi_{exact} + small} \times 100\% \tag{3.19}$$

For all the schemes, the average relative error always drops as local Pe drops. Since the Peclet number is directly proportional to the mesh size, this trend infers the error reduction with grid refinement.

At large local Pe, the CDS gives the most significant error among all schemes, even though its order of accuracy exceeds the UDS's order of accuracy. The CDS is unsuitable for convection dominated flow since it is unable to differentiate flows from different directions. Thus, it takes upwind and downwind contributions identically. In convection dominated flows, the property value should be affected more by its upstream node value. Therefore, upwind schemes are more appropriate when large Pe is present.

It is also observed that at large local Pe, the third-order accurate QUICK scheme gives the second largest error. The error is caused by the fluctuation in ϕ^* presented in Figure 3.7. It is thus evident that high-order schemes like QUICK produce results that is quite unstable when the mesh is coarse. On the contrary, the TVD scheme gives the most accurate result. As local Pe drops, the QUICK error reduces rapidly and at $Pe = 0.625$, it matches the error from TVD. The lowest percentage error achieved with the code is 10^{-2}.

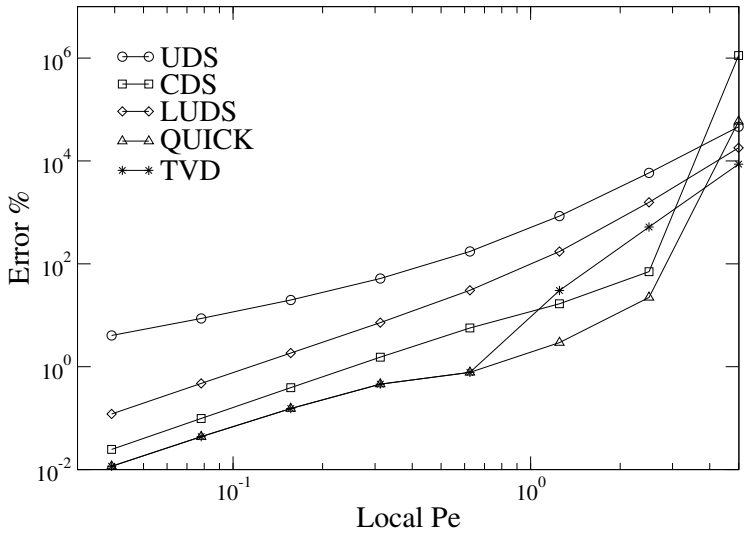


Figure 3.8: Maximum percentage error variation with Peclet number for code results

3.4 Verification on Order of Accuracy

Order of accuracy of a scheme depicts the rate of error reduction as the grid refines. For instance, the error of a first-order scheme is supposed to drop one order of magnitude when the mesh size is refined by an order of magnitude. The error is defined as:

$$Error = \frac{\sum |\phi_{exact} - \phi_{numerical}|}{N} \tag{3.20}$$

A plot of error for various schemes versus mesh size is presented in Figure 3.9. Comparing the slope of the schemes with the slope of the lines representing different orders of accuracy, it is verified that UDS is first-order accurate, and the CDS and LUDS are second-order accurate.

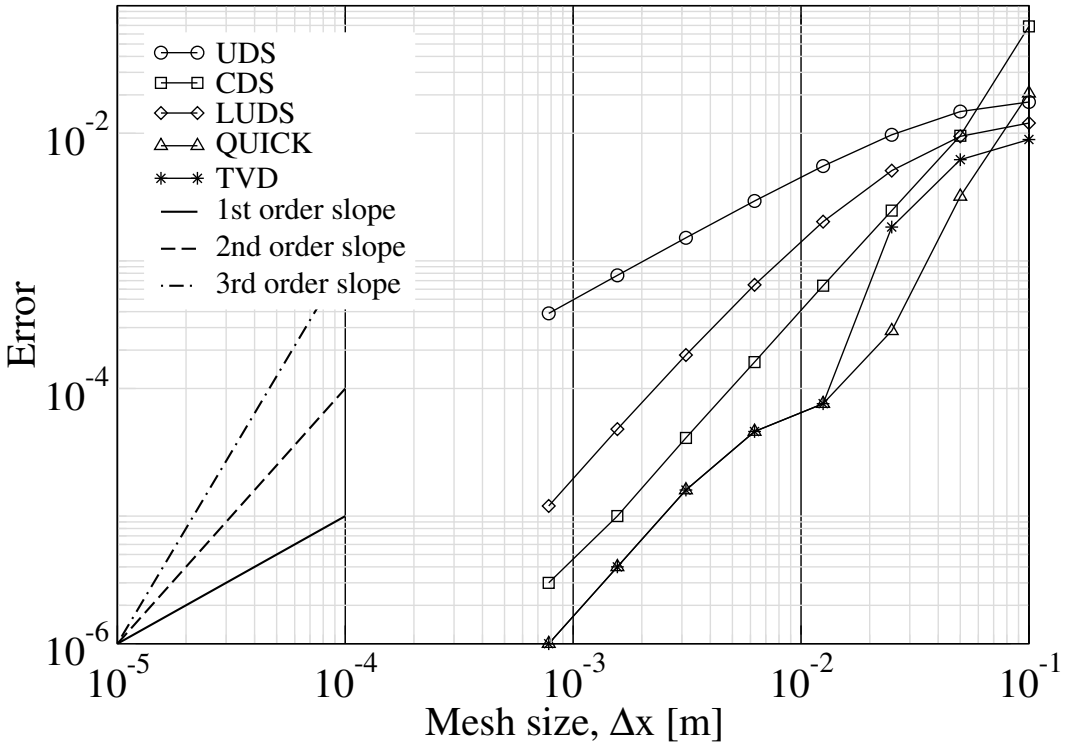


Figure 3.9: Order of accuracy check for code results

3.5 Validation

3.5.1 ANSYS Fluent

A pseudo one-dimensional case was tested in ANASYS Fluent. The simulation domain is two-dimensional and has size 10 m $\times$ 1 m. The temperature was set to 300 K at the top and 280 K at the bottom. The left and right boundaries are symmetry boundaries. The horizontal direction is uniform and fluid flow from the bottom to the top at a velocity of $0.001m/s$. A schematic of the mesh and the problem statements is presented in Figure 3.10. For comparison purposes, the tested fluid properties were the same as the inputs for the code ($\rho = 1$ kg/m^3, $c_p = 1000$ J/kg·K and $k = 0.02$ W/m·K). Four mesh sizes, 10×100, 20×200, 40×400 and 80×800 were tested. The results were compared to the code outputs for 10, 20, 40, 80 CVs respectively.

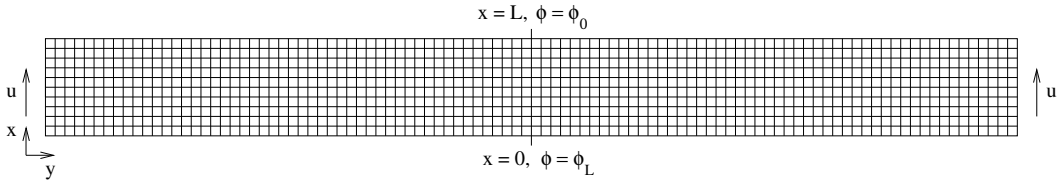


Figure 3.10: Convection-diffusion problem simulation domain with boundary conditions in Fluent

Results for the three upwind difference schemes: UDS, LUDS and QUICK were compared. The UDS results from Fluent and the code were precisely the same; however, results for LUDS and QUICK scheme showed large deviation. It was observed that Fluent's QUICK scheme always produces non-oscillation results. The exact formulation was not provided in Fluent's theory manual; thereby, the code results and Fluent's findings are not comparable. Results for LUDS are plotted in Figure 3.11, along with the exact solutions. It was observed that results from code were always more accurate than results from Fluent.

The mean relative error is again plotted against local Peclet number in Figure 3.12. The maximum number of control volume plotted is 80, because the solution failed to converge with further mesh refinement. The error drops as local Pe drops. However, the error for LUDS and QUICK are nearly overlapping. This proves the conjecture that Fluent algorithm for LUDS and QUICK deviates from the true LUDS and QUICK formulations.

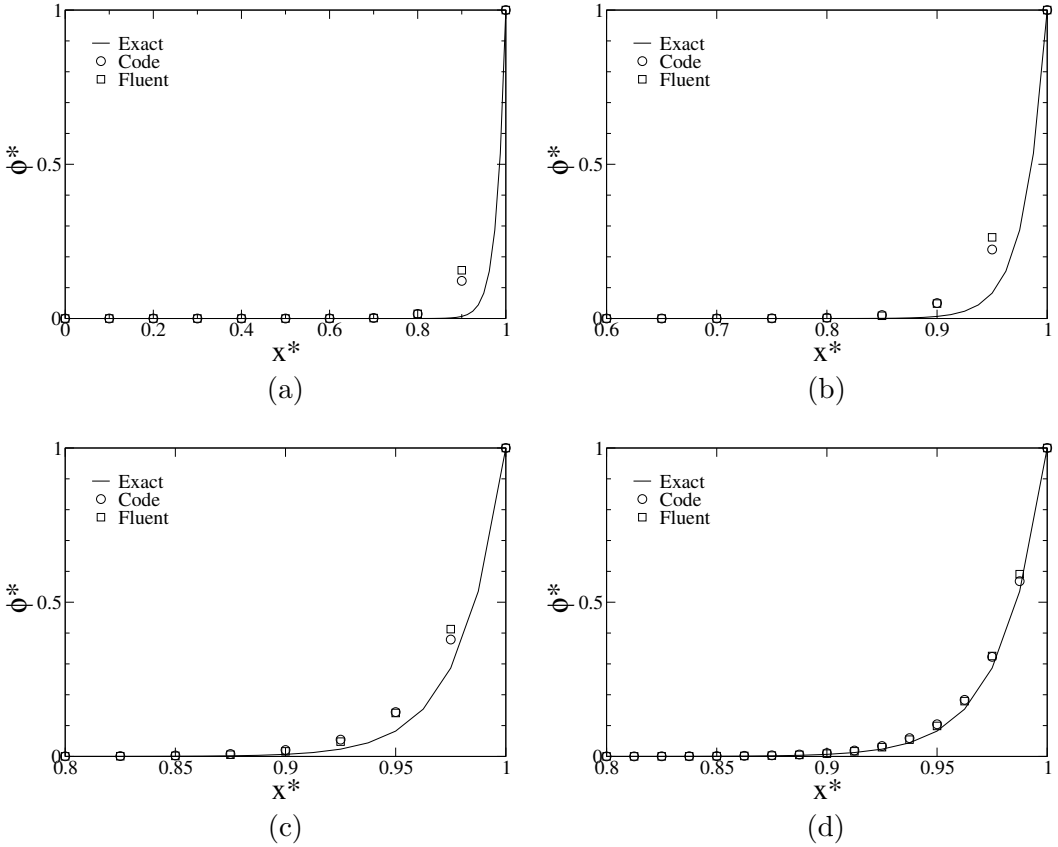


Figure 3.11: Comparison between code and Fluent results obtained from LUDS for (a) 10 CVs (b) 20 CVs (c) 40 CVs (d) 80 CVs

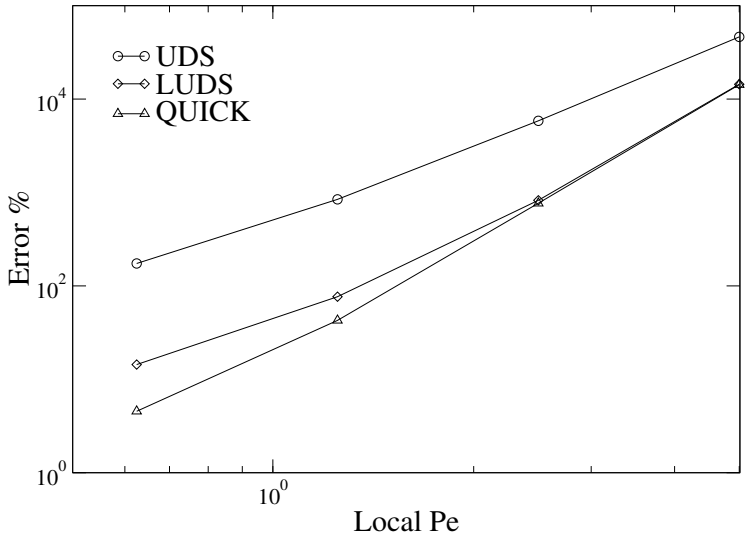


Figure 3.12: Maximum percentage error variation with Peclet number for Fluent results

To better understand the reason behind the discrepancy, a plot of error for UDS, LUDS and QUICK from Fluent calculations versus mesh size is presented in Figure 3.13. Comparing with Figure 3.9, it can be seen that, for QUICK, the Fluent error is dropped to 10^{-3} at 80 CVs, while the Code error is as low as 10^{-4}. It is thereby suspected that Fluent uses a blending formulation for LUDS and QUICK to prevent oscillation in high local Pe regions.

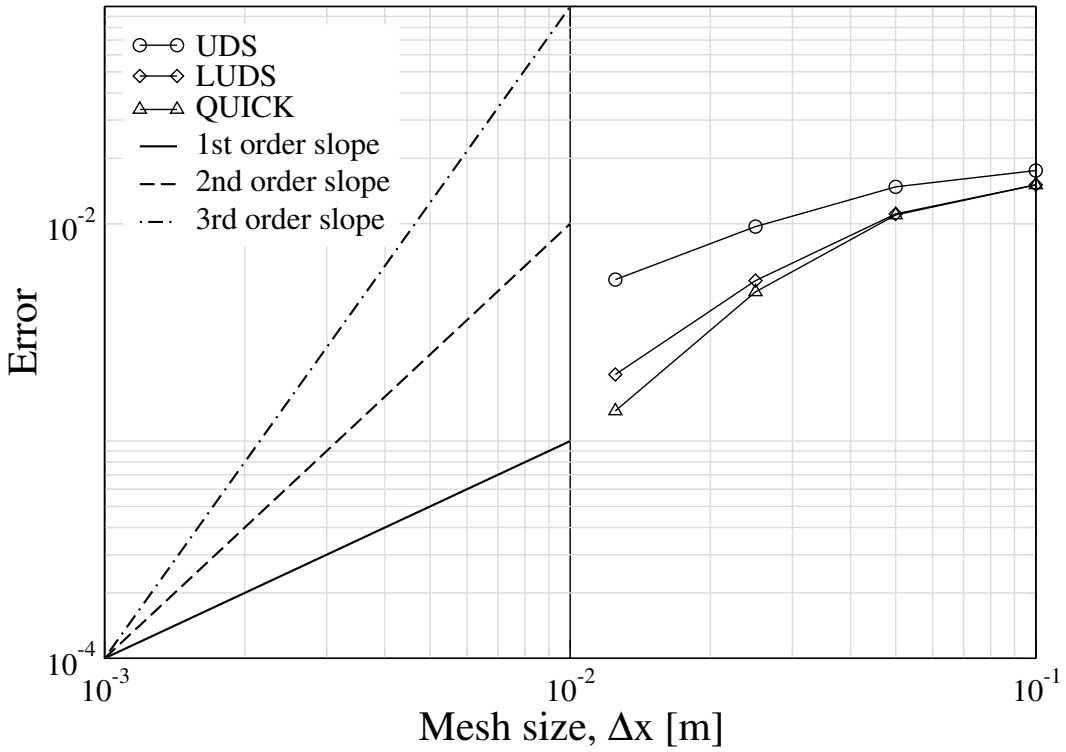


Figure 3.13: Order of accuracy check for Fluent results

Chapter 4

2D Formulation and Validation

4.1 Formulation of the Convection-Diffusion Equation

Considering a standard two-dimensional control volume as shown in Figure 4.1, the central point and the adjacent points west, east, north and south of it are denoted by upper-case letters P, W, E, N and S. The cell faces are denoted by lower-case letters w, e, n and s. The size of the control volume is $\delta x \times \delta y$.

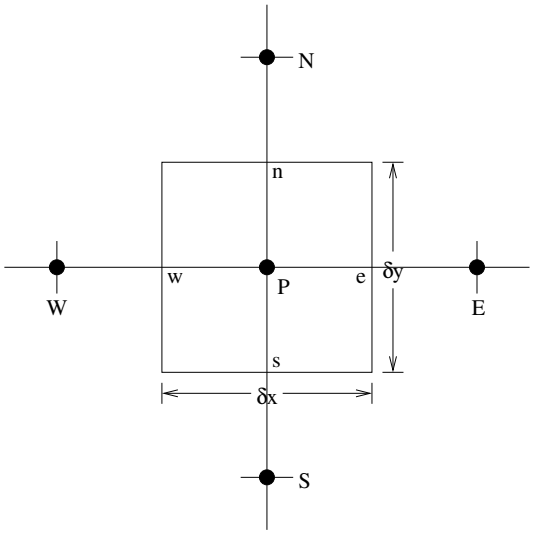


Figure 4.1: Example of two-dimensional control volume discretization around node P on a uniform grid

The two-dimensional steady-state convection-diffusion equation, without the

source term, is presented in Equation 4.1.

$$\rho(u_x\frac{\partial\phi}{\partial x}+u_y\frac{\partial\phi}{\partial y})=\Gamma(\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}) \tag{4.1}$$

The two dimensions were treated separately; namely, the flow was artificially divided into the x-component and y-component. The flux in the x-direction was solved first; the flux in the y-direction was solved subsequently. The x-direction formulation is the same as in the one-dimensional case; the y-direction formulation is very similar. For clarity, formulations for flux in the y-direction are presented in Equations 4.2 to 4.4.

$$\begin{aligned} f_n\phi_n =&(\phi_P+f_{cn}^{+}(\phi_N-\phi_S))AMAX(f_n,0) \\ &-(\phi_N-f_{cn}^{-}(\phi_{NN}-\phi_P))AMAX(-f_n,0) \end{aligned} \tag{4.2}$$

$$\begin{aligned} f_s\phi_s =&(\phi_S+f_{cs}^{+}(\phi_P-\phi_{SS}))AMAX(f_s,0) \\ &-(\phi_P-f_{cs}^{-}(\phi_N-\phi_S))AMAX(-f_s,0) \end{aligned} \tag{4.3}$$

$$(\Gamma\frac{\partial\phi}{\partial y})_n=\Gamma_n\frac{\phi_N-\phi_P}{y_N-y_P} \quad \& \quad (\Gamma\frac{\partial\phi}{\partial y})_s=\Gamma_s\frac{\phi_P-\phi_S}{x_P-x_S} \tag{4.4}$$

The linearized equation and its coefficients are presented in Equations 4.5 to 4.9; where the superscript c denotes convection and the superscript d denotes diffusion.

$$A_N\phi_N+A_P\phi_P+A_S\phi_S=b_P \tag{4.5}$$

$$A_N=A_N^c+A_N^d=-AMAX(-f_n,0)-\frac{\Gamma_n}{x_N-x_P} \tag{4.6}$$

$$A_S=A_S^c+A_S^d=-AMAX(0,f_s)-\frac{\Gamma_s}{x_P-x_S} \tag{4.7}$$

$$A_P=-(A_N+A_S) \tag{4.8}$$

$$\begin{aligned} b_p=&f_{cn}^{+}(\phi_S-\phi_N)AMAX(f_n,0)-f_{cn}^{-}(\phi_{NN}-\phi_P)AMAX(-f_n,0) \\ &+f_{cs}^{+}(\phi_P-\phi_{SS})AMAX(f_s,0)-f_{cs}^{-}(\phi_S-\phi_N)AMAX(-f_s,0) \end{aligned} \tag{4.9}$$

4.2 2D Lids-Driven Cavity Problem

The two-dimensional lids-driven cavity (LDC) problem was studied to validate the program. The schematic of a typical LDC problem is shown in Figure 4.2. The LDC problem studies the flow and temperature distribution inside a closed rectangular cavity. Either, or both of the top and bottom boundaries of the domain are "lids" moving toward the positive x-direction at a constant speed U_{lid}. The top and bottom lids are at constant temperatures of T_t and T_b respectively. The left and right of the domain are solid walls that are no-slip and adiabatic.

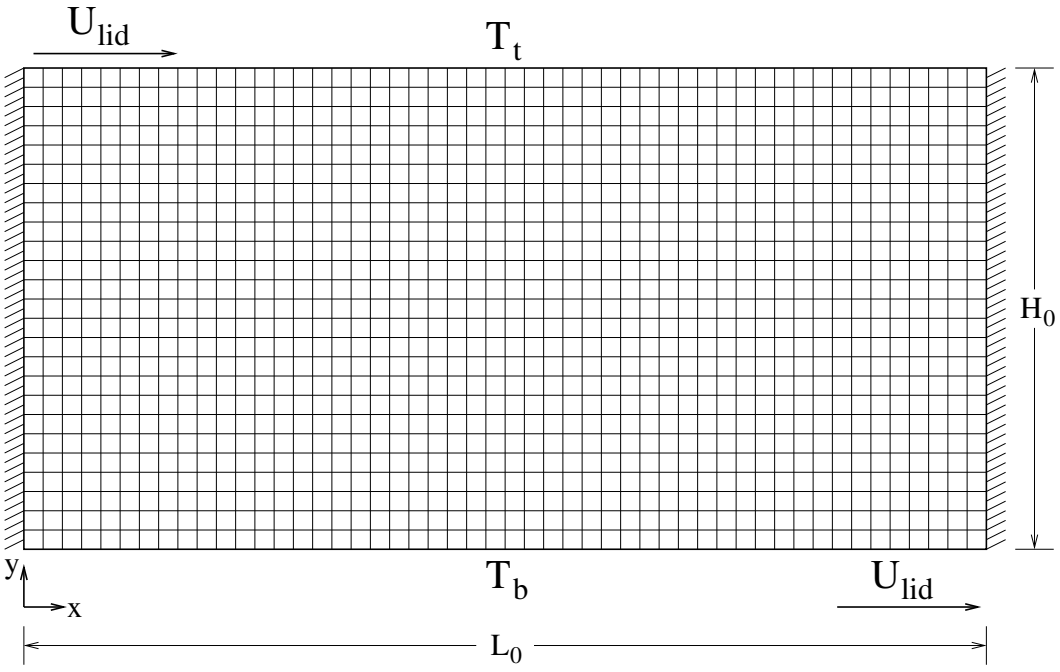


Figure 4.2: Schematic of the lids-driven cavity (LDC) problem

In order to solve this problem, three fundamental equations are solved simultaneously. These three equations are: the mass conservation equation (Equation 4.10), the Navier-Stokes equation (Equation 4.11), and the temperature convection-diffusion equation (Equation 4.12). The system is at equilibrium, so the $\frac{\partial u}{\partial t}$ term is eliminated. Fluid properties are assumed constant. The viscous heating effect and the buoyancy effect are also neglected.

$$\nabla \cdot \vec{u} = 0 \tag{4.10}$$

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \vec{u} \tag{4.11}$$

$$\frac{\partial T}{\partial t} + (\vec{u} \cdot \nabla)T = \frac{k}{\rho c_p} \nabla^2 T \tag{4.12}$$

The domain is 2×1 m^2 and both lids are moving at $U_{lid} = 0.007$ m/s. The top lid is held at $T_t = 320$ K and the bottom lid at $T_b = 300$ K. The tested fluid was air with $\rho = 1.225$ kg/m^3, $c_p = 1006.43$ J/kg·K, $k = 0.0242$ W/m·K, and dynamic viscosity $\mu = 1.7894 \times 10^{-5}$ kg/m·s. The p and ν in Equation 4.11 stand for pressure and fluid kinematic viscosity respectively. Kinematic viscosity is the quotient of the fluid's dynamic viscosity and density. The Reynolds number is $Re = \frac{\rho U_{lid} H}{\mu} = \frac{1.225 \times 0.007 \times 1}{1.7894 \times 10^{-5}} = 479.2$. The flow is laminar and the domain can be considered symmetric under current flow specifications.

Square control volumes were used for discretization. Three mesh sizes where tested: $\Delta x = \Delta y = 0.04$ m, $\Delta x = \Delta y = 0.02$ m and $\Delta x = \Delta y = 0.01$ m; these mesh sizes corresponds to 25×50, 50×100 and 100×200 total control volumes. The corresponding Peclet numbers calculated by $(\rho u_{lid} \Delta x c_p)/k$ are 14.3, 7.1 and 3.6 respectively for the systematically refined grids. The simulated temperature contour plot is similar for all cases; therefore, only one plot is presented for visualization. Figure 4.3 shows the temperature contour for the finest grid with Lin-Lin TVD.

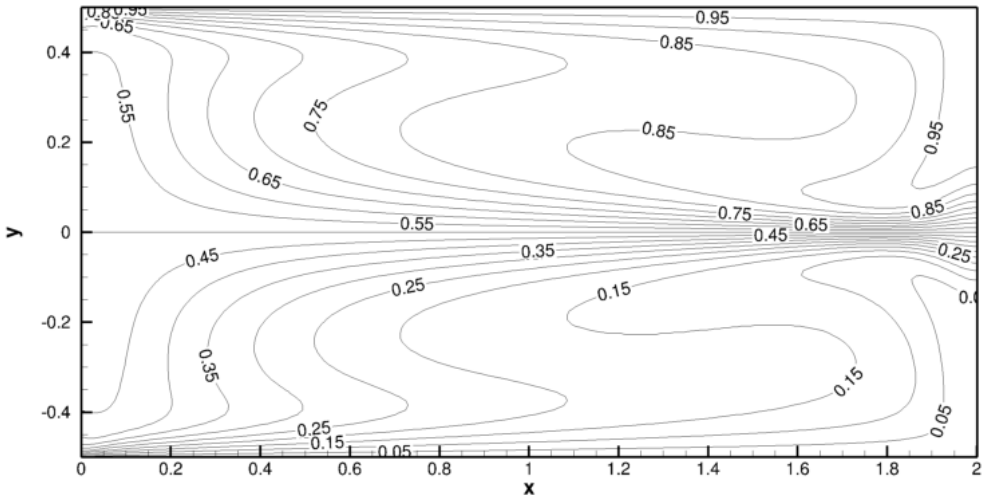


Figure 4.3: Normalized temperature profile for the lids-driven cavity (LDC) problem

The contour plot confirms geometrical and physical symmetry of the domain. No vortices are present. The density of the contour lines provides information on the gradient of temperature. Near the left wall, temperature change along the y-direction is more gradual near the symmetry line at $y = 0$ and more abrupt near the top and bottom lids. As opposed to which was observed near the right wall, temperature change along the y-direction is more gradual near the lids and more abrupt crossing the symmetry line.

In order to conduct a quantitative analysis, temperatures were taken from two lines in the domain: one at $x = 0.5$ m and the other at $x = 1.5$ m. Results from the upwind schemes (UDS, LUDS, QUICK and TVD) are plotted in Figure 4.4. The horizontal axis is y, and the vertical axis is the normalized temperature T^* calculated by $(T - T_b)/(T_t - T_b)$.

In the current study, spurious oscillations are not observed, and all the solutions are bounded. As presented in Figure 4.4 (a), with the coarsest mesh tested, along $x = 0.5$ m, all schemes result in similar trends. However, the temperature magnitude varies. While UDS and LUDS over-predict temperature for $y < 0$ and under-predict temperature for $y > 0$, TVD produces results that are in close agreement with the QUICK scheme.

When the mesh is refined, as presented in Figure 4.4 (c) and (e), the gap between results from various schemes diminishes. With twenty thousand control volumes, the solution from low-order schemes overlaps with TVD, with the expense of excessive computational resources. All schemes give comparable temperature profiles along $x = 1.5$ m as shown in Figure 4.4 (b), (d) and (f).

Similar to high order scheme, the TVD solution is less dependent on the mesh size. The mean relative error is calculated by taking the average of $|\frac{T_1 - T_2}{T_1}| \times 100\%$; where T_1 is the solution temperature for the coarser grid, and T_2 is the solution temperature for the finer grid. The corresponding errors are listed in Table 4.1. The relative error calculated using the coarsest mesh with $\Delta x = 0.04$ m and the finest mesh with $\Delta x = 0.01$ m is 0.0333% lies between the results from LUDS and QUICK. Overall, the TVD scheme is able to give reasonably accurate predictions with a limited number of control volumes.

Table 4.1: Summary of mean relative errors (%) as the grid refines

Schemes \ Δx, [$\times 10^{-2}$ m]	$4 \to 2$	$2 \to 1$	$4 \to 1$
UDS	0.0490	0.0198	0.0689
LUDS	0.0256	0.0085	0.0357
QUICK	0.0186	0.0052	0.0240
TVD	0.0263	0.0064	0.0333

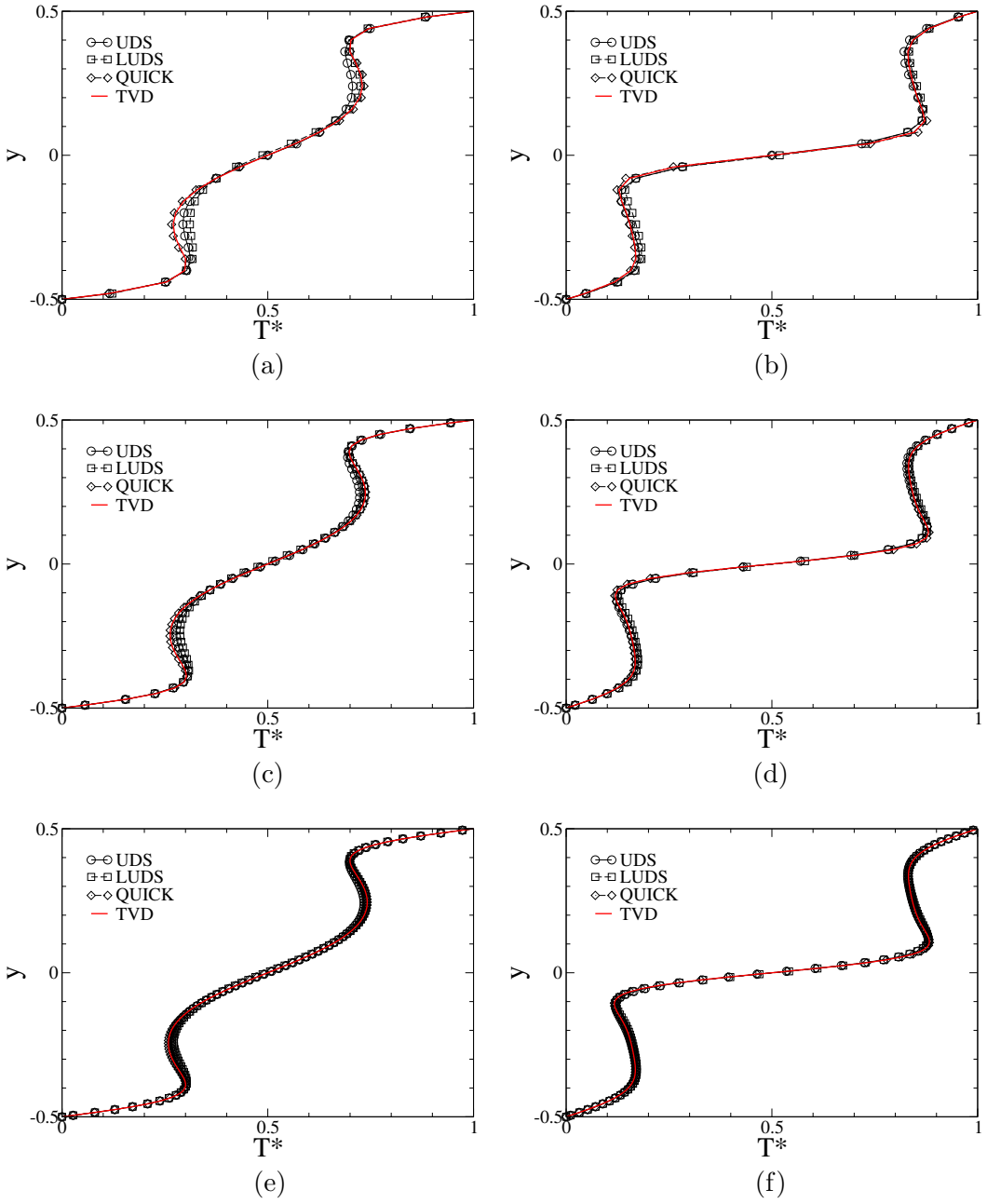


Figure 4.4: Normalized temperature plots from various methods along $x = 0.5$ m: (a) 1250 CVs (c) 5000 CVs (e) 20000 CVs and along $x = 1.5$ m: (b) 1250 CVs (d) 5000 CVs (f) 20000 CVs

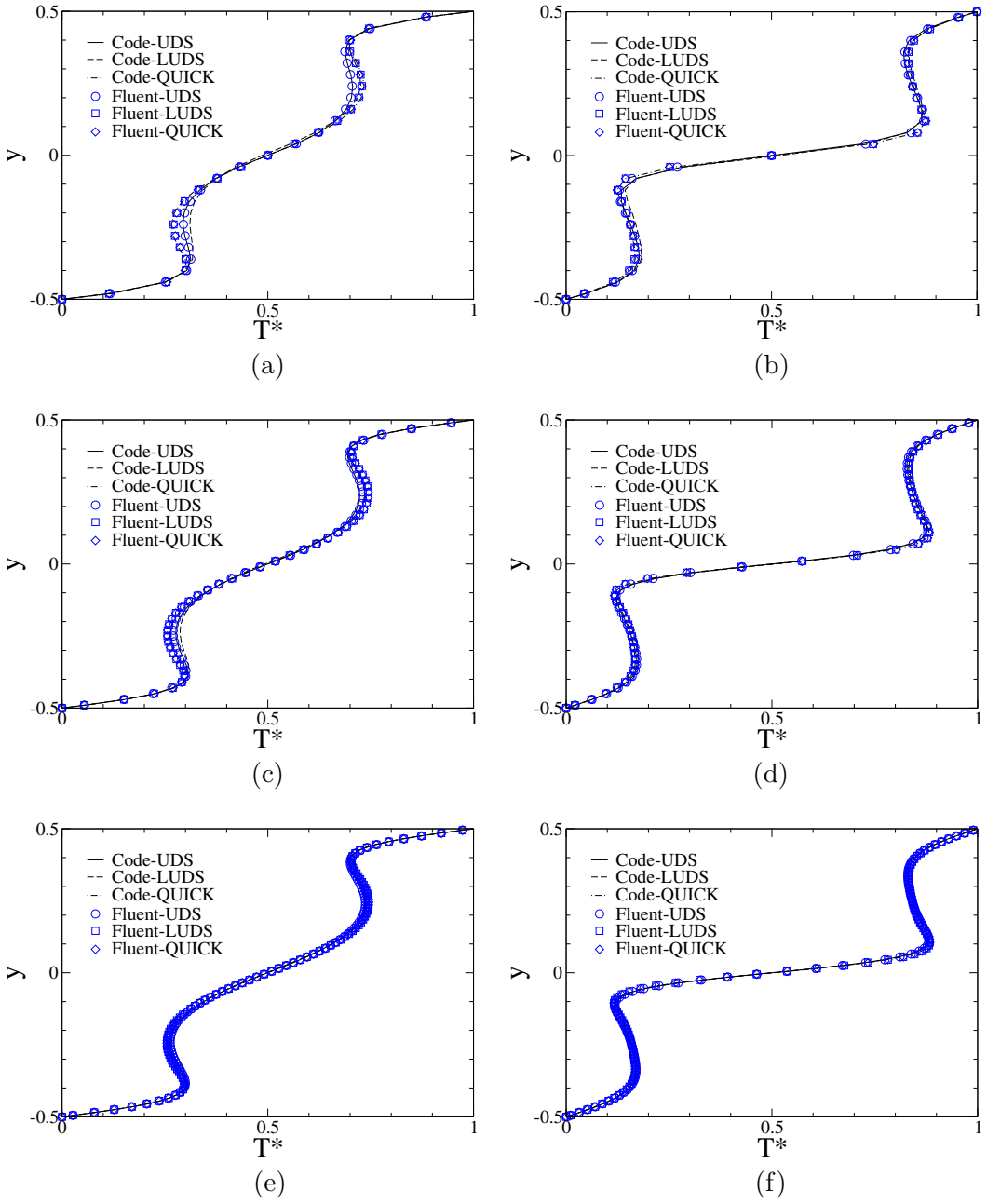


Figure 4.5: Comparison between results from code and Fluent along $x = 0.5$ m: (a) 1250 CVs (c) 5000 CVs (e) 20000 CVs and along $x = 1.5$ m: (b) 1250 CVs (d) 5000 CVs (f) 20000 CVs

The UDS, LUDS and QUICK solutions were validated against the results from Fluent. The overall trend align well with the trend from the code results as presented in Figure 4.5. The maximum relative discrepancies were calculated with Equation 4.13 and was summarized in Table 4.2.

$$Error\ \% = \max |\frac{T^*_{Fluent} - T^*_{code}}{T^*_{Fluent}}| \times 100\% \tag{4.13}$$

Table 4.2: Summary of maximum relative discrepancies (%) as the grid refines

Schemes \ Δx, [$\times 10^{-2}$ m]	4	2	1
UDS	5.6863	3.4278	1.9069
LUDS	16.4903	12.3029	6.6917
QUICK	3.7169	3.6264	2.1455

The relative error decreases as the grid refines because theoretically, both the code results and the Fluent results are approaching the exact solution asymptotically, even though the exact solution is unavailable for this case. In one dimension, the UDS implemented in code and the Fluent UDS produces the exact same results. However, in the LDC problem, the difference is not negligible. Such difference can be caused by different algorithms in calculation of the velocity field, and is not discussed here.

Furthermore, the QUICK scheme implemented in code and Fluent produces the most similar results. Due to QUICK's high-order of accuracy, when its solution does not oscillate, it is capable of producing the best approximation among the three upwind schemes. The percentage discrepancy is the largest for LUDS.

Although it is generally a good practice to validate simulation results with commercial software solutions, results from commercial software needs to be used with caution since the exact implementations are encapsulated from the users. Developing an in-house software therefore gives user the option to manage the implementations and truly understand the algorithms.

4.3 2D Cylinder Advection

A two-dimensional unsteady problem of cylinder advection was considered. This is a simplification of bubble advection in three dimensions. Since the third dimension is ignored, the circle extends infinitely in the third dimension, transforming into a cylinder. Instead of the temperature convection-diffusion equation (Equation 4.12), the convection-diffusion of volume fraction ε was solved (Equation 4.14). The flow field has a uniform velocity; as a result, the momentum equation was not solved.

$$\frac{\partial \varepsilon}{\partial t} + (\vec{u} \cdot \nabla)\varepsilon = D\nabla^2\varepsilon \tag{4.14}$$

Both the cylinder and the surrounding are fluids with the same properties. The two fluids considered are immiscible, so physical diffusion is assumed absent. D is the diffusive coefficient of volume fraction, which should have a value of exactly zero. For numerical purposes, D was set to 10^{-15}. The physical value of ε is confined to the range of zero to one. The flow velocity was set to $U = 0.1$ m/s in the z-direction and zero in the r-direction. The strongly implicit procedure (SIP) solver was used for the matrix calculations.

The Cartesian grid was used. The stair-step approximation was applied. The stair-step approximation is the most simplistic approach for surface reconstruction. Some other commonly used options are the simple linear interface calculation (SLIC) and the piecewise linear interface calculation (PLIC) [25].

An illustration of the stair-step approximation is provided in Figure 4.6. A circle containing fluid A (the shaded area) with radius R is immersed in fluid B (the non-shaded area), and placed on a two-dimensional discretized mesh with $\Delta x = \Delta y = R/5$. Fluid A has a volume fraction of one whereas fluid B has a volume fraction of zero. In a control volume, if $\varepsilon \geq 0.5$, the entire cell is treated as if it was filled by fluid A; to the contrary, if $\varepsilon < 0.5$, the cell is considered entirely occupied by fluid B. The thicker line in Figure 4.6 represents the reconstructed interface given by the method. Such a technique is incapable of providing a precise prediction of the surface, and the resulted surface smoothness is closely related to the mesh size adopted. The surface curvature calculations was not included in the code since the surface tension calculations were not implemented.

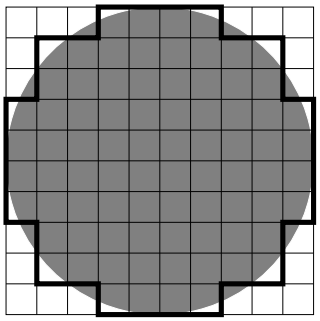


Figure 4.6: Illustration of the stair-step approximation

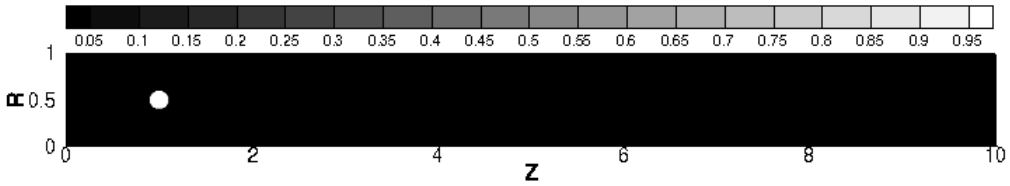


Figure 4.7: Initial schematic of the cylinder convection problem

In the current simulation, the computational domain is 10×1 m^2, as demonstrated in Figure 4.7. The colour bar depicts the volume fraction of the cylinder. The radius of the cylinder is set to 0.1 m. Initially, the cylinder is located at $(1, 0.5)$ m. The top and bottom boundaries were set to constant ε; while the left and right boundaries were set to no ε-flux, namely, $\frac{\partial \varepsilon}{\partial z}$ equals to zero.

Explicit time discretization was used. In convection dominated flow, the Courant-Friedrichs-Lewy (CFL) number, defined by the ratio of flow velocity U and dimensionless velocity u^* has to be less than one. The dimensionless velocity is equal to the mesh size divided by the time step. Due to the fact that a fluid particle cannot travel through more than one control volume in a single time step, the time step Δt is restricted to be less than $\Delta z/U$, which is the time needed for a particle to travel through one control volume.

Two meshes with Δx equal to 0.02 m and 0.01 m respectively were tested. The corresponding Peclet numbers are 2×10^{12} and 10^{12} for the coarse grid and the finer grid respectively. the At the given velocity, the critical time steps calculated by the CFL restriction are 0.2 s and 0.1 s respectively. To guarantee convergence, Δt was set to 0.02 s. Physically, the cylinder is expected to hold its shape and maintain its volume while moving through the domain because there is no external force present.

The simulation was carried out for an 80-s period. The CDS, UDS, LUDS, QUICK and Lin-Lin TVD scheme were assessed. Figure 4.8 is the solution ε contour plot at $t = 40$ s and Figure 4.9 is the solution ε contour plot at the end of simulation period for the 50×500 grid. The commercial software Tecplot 360 was used to post-process the data and prepare the plots. The scales of the plots were not unified on intention. The span of the volume fraction illustrates the boundedness of the solution. Since the field was initialized with $0 < \varepsilon < 1$, any solution that is beyond this range is unbounded and thus non-physical.

It was observed that, for all the schemes tested, the cylinder was deformed into a slender bar extending in the z-direction. The degree of deformation differs. The solution remained unaltered in the top 1/3 and bottom 1/3 of the simulation domain.

For both mesh sizes tested, only the UDS and TVD scheme were able to produce bounded solutions. The CDS produced a mirrored image of the cylinder tailed two meters behind the actual cylinder. This mirrored image of the cylinder imitated the modelled shape of the cylinder but had a negative ε value. The absolute value of ε inside the mirrored cylinder is close to the value of ε inside the actual cylinder. Aside from the mirrored image, the resulted volume was scattered throughout the simulation domain. Overall, the CDS gave the most unbounded solution.

Even though the UDS provided a bounded solution, the numerical diffusion caused a massive reduction in the cylinder volume. At the end of the simulation period, there was nearly no trace of the cylinder remaining as seen in Figure 4.9.

The LUDS has an intermediate ability in predicting the shape and volume of the cylinder, but it gave a region of negative ε right ahead of the actual cylinder resulting in unboundedness. In addition, there appeared to be segments of diffused volumes near the cylinder. The QUICK scheme well preserved the shape of the cylinder due to its high order of accuracy. However, the cylinder was immediately followed by a region of negative ε, and similar to the LUDS, there were segmented regions with $0 < \varepsilon < 1$. TVD produced the most reliable result. With a 50×500 grid, the cylinder was deformed slightly, and the cylinder surface is somewhat smeared.

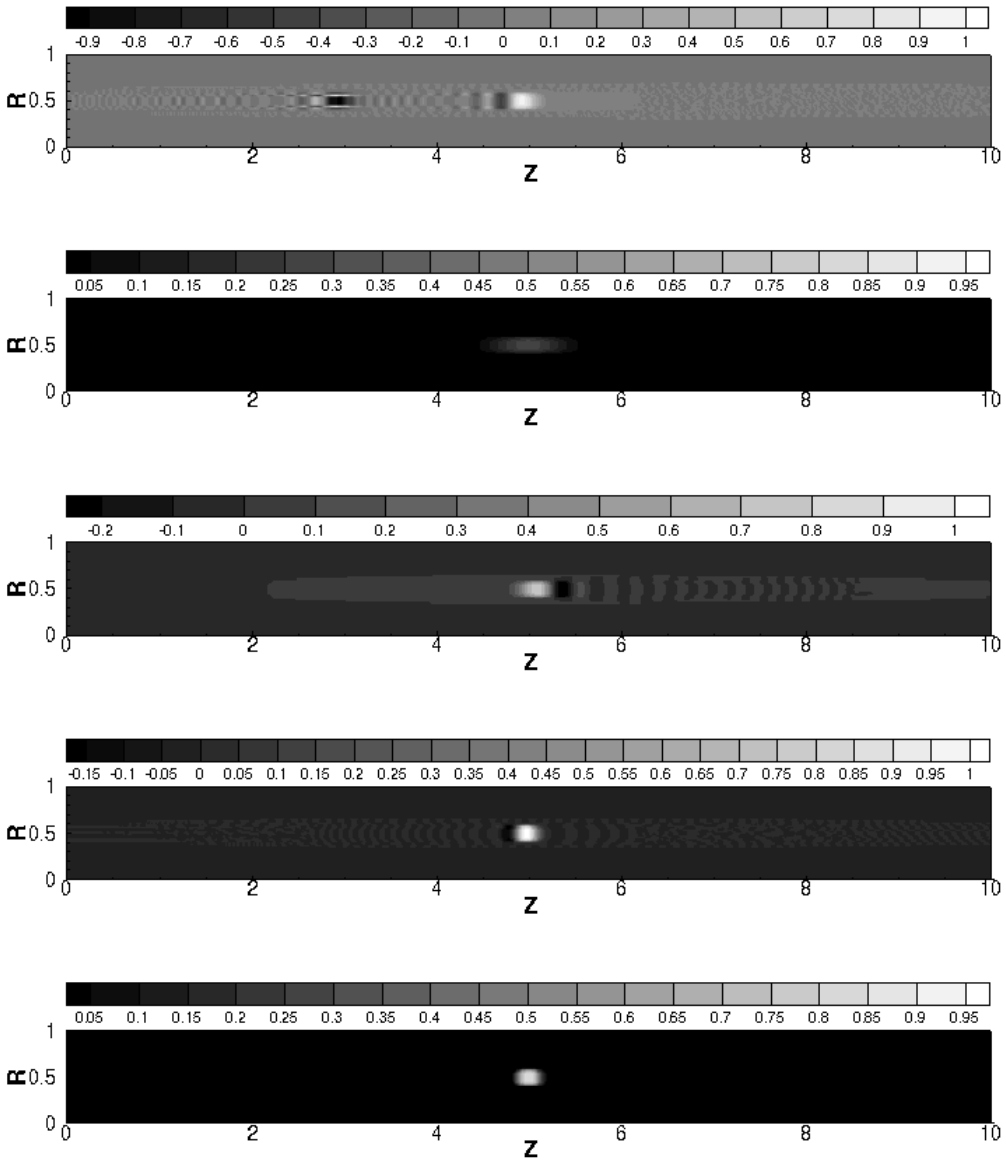


Figure 4.8: Volume fraction contour plots obtained with a 50×500 grid at $t = 40$ s for: CDS, UDS, LUDS, QUICK and TVD respectively

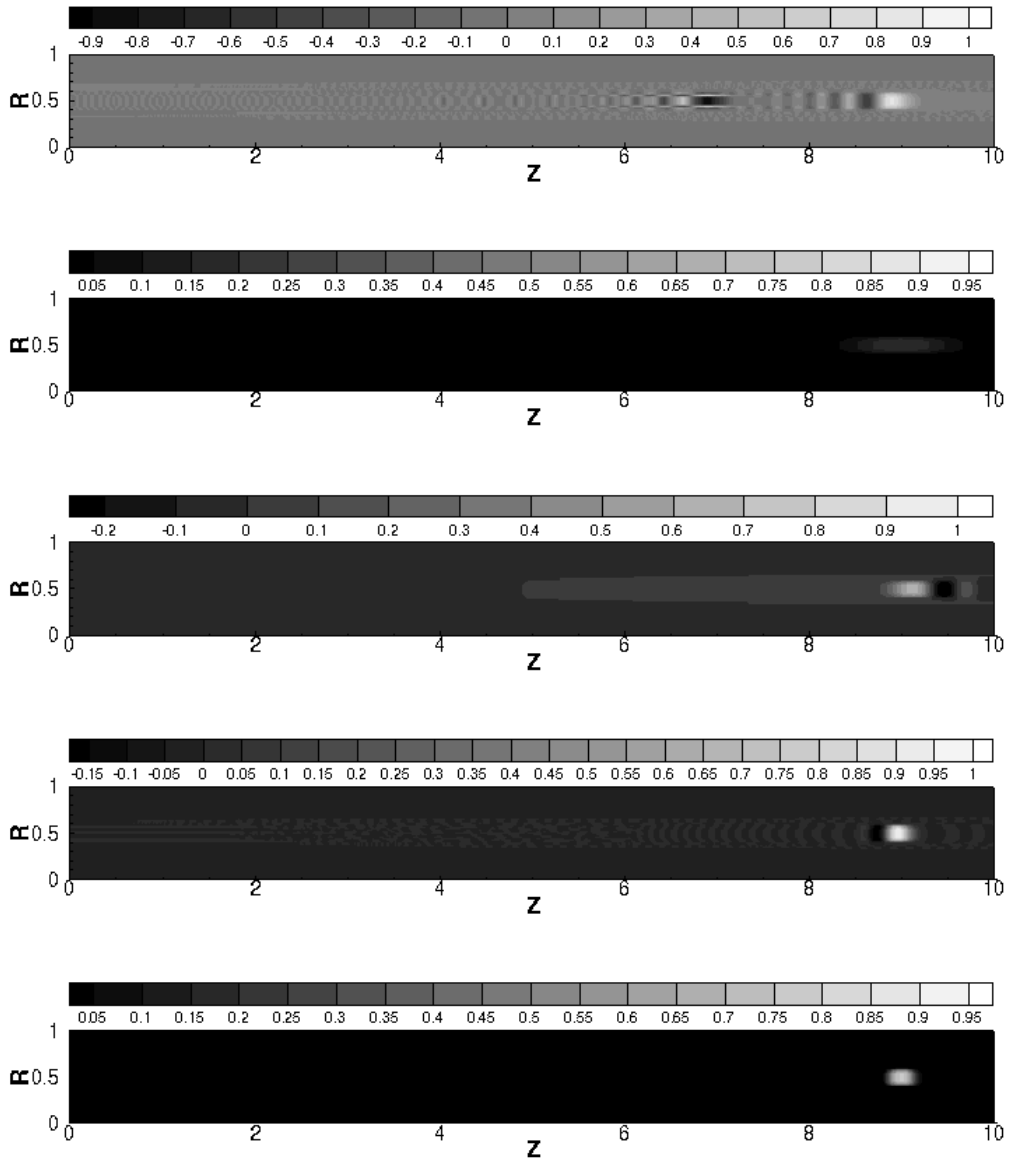


Figure 4.9: Volume fraction contour plots obtained with a 50×500 grid at $t = 80$ s for: CDS, UDS, LUDS, QUICK and TVD respectively

After mesh refinement, solutions from all schemes improved in terms of volume preservation as showed in Figure 4.10 and Figure 4.11. There was no visible deformation for the QUICK and TVD scheme. However, the problem of unboundedness for the CDS, LUDS and QUICK did not mitigate at all, meaning that the unphysical oscillation became more severe.

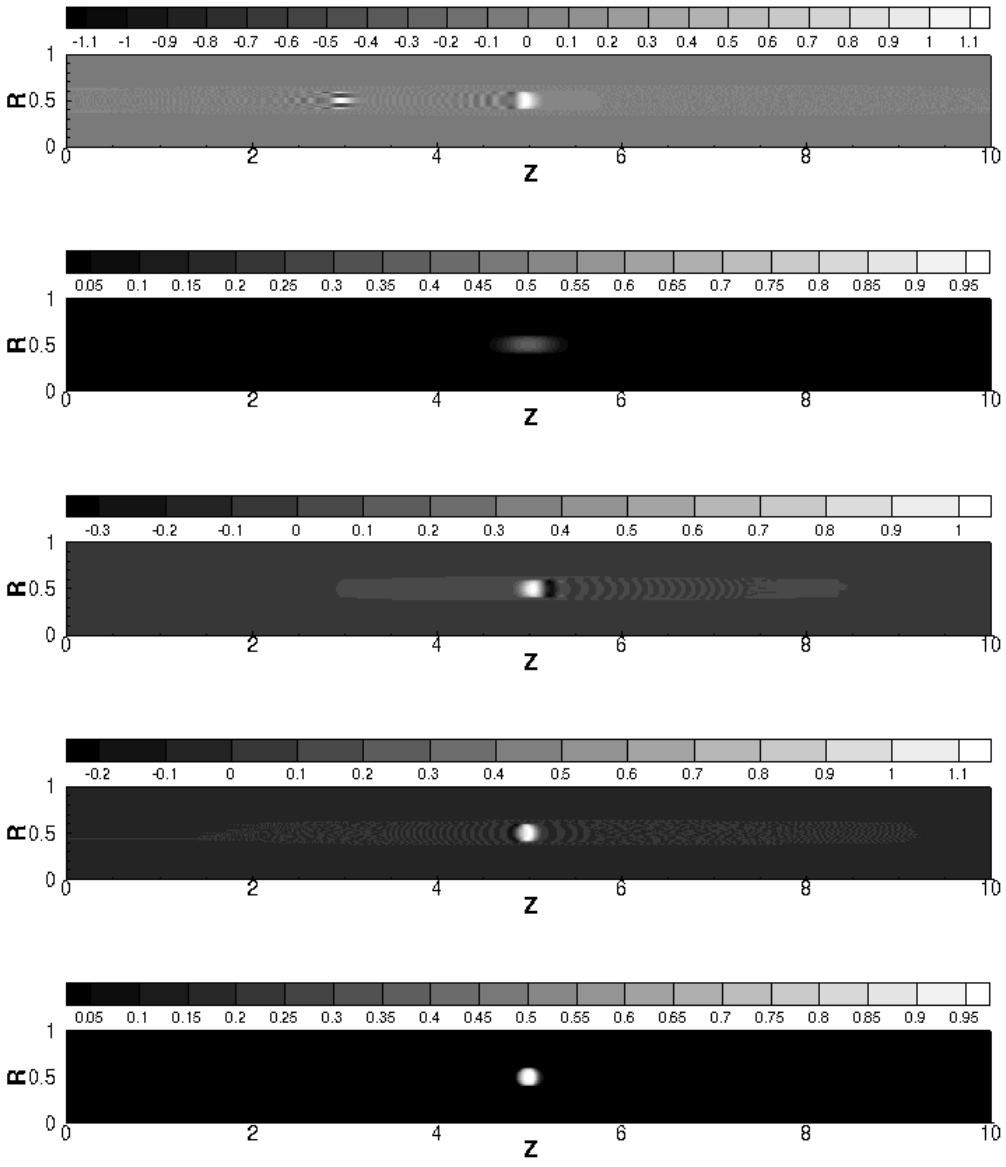


Figure 4.10: Volume fraction contour plots obtained with a 100×1000 grid at $t = 40$ s for: CDS, UDS, LUDS, QUICK and TVD respectively

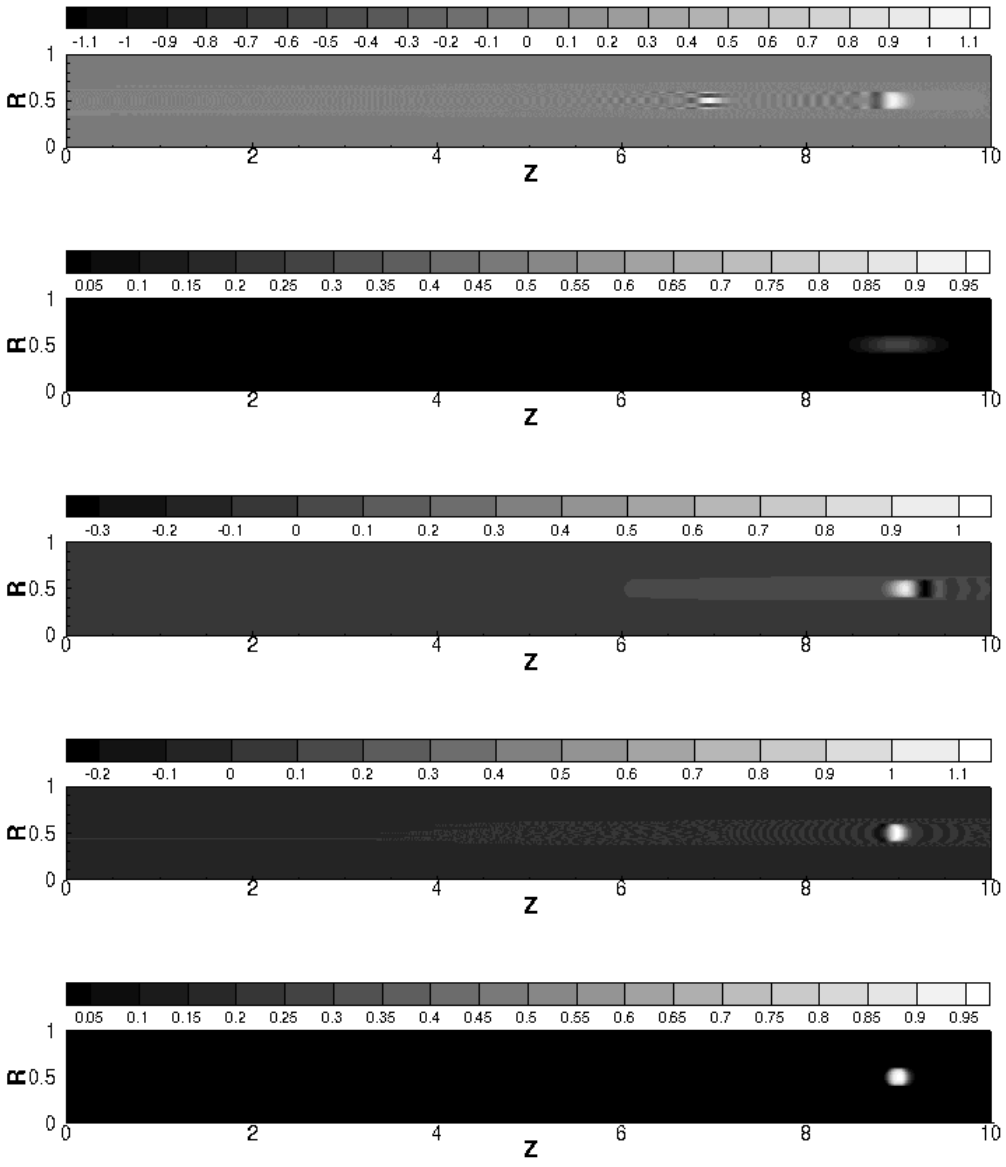
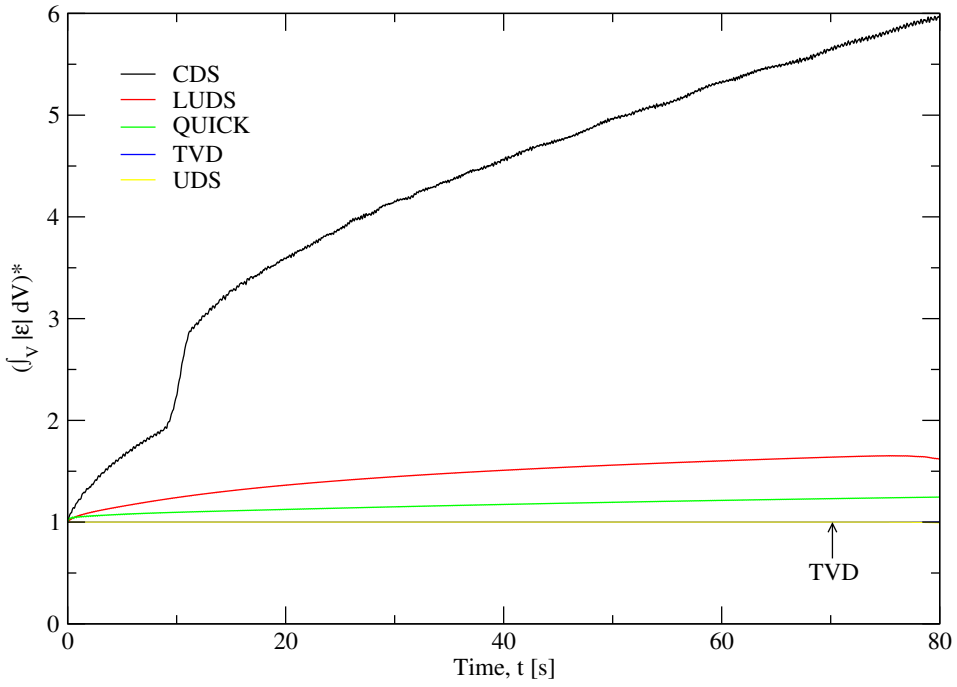


Figure 4.11: Volume fraction contour plots obtained with a 100×1000 grid at $t = 80$ s for: CDS, UDS, LUDS, QUICK and TVD respectively

To quantitatively analyze the results, two parameters were plotted against time: the volume averaged ε (Figure 4.12) and the stair-step approximated volume V^* of the cylinder (Figure 4.13). Both of these parameters were normalized with the initial values since both of these parameters should remain unchanged theoretically.



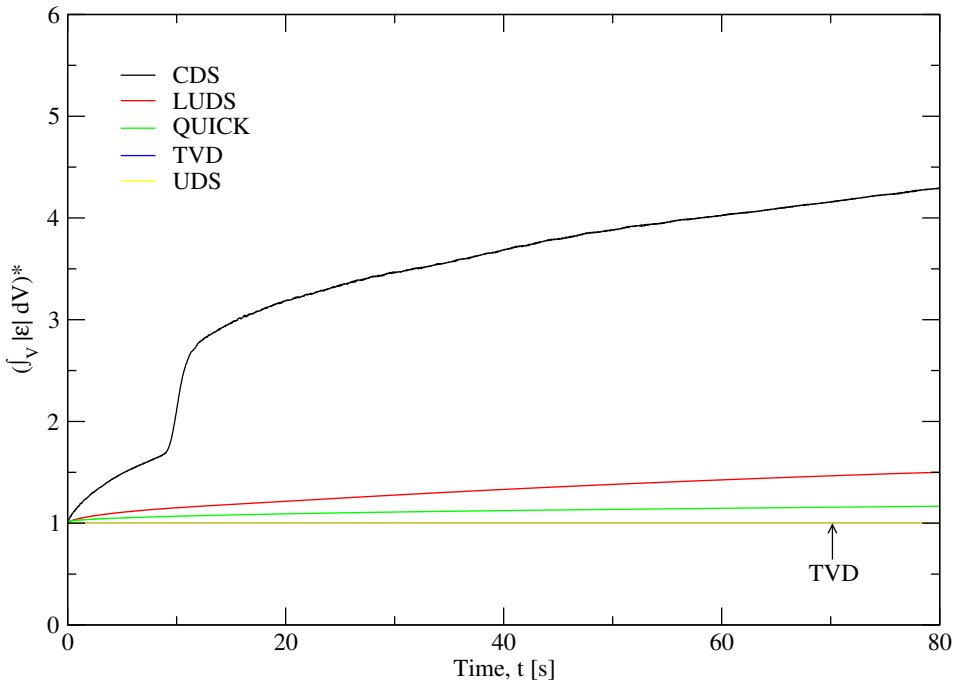
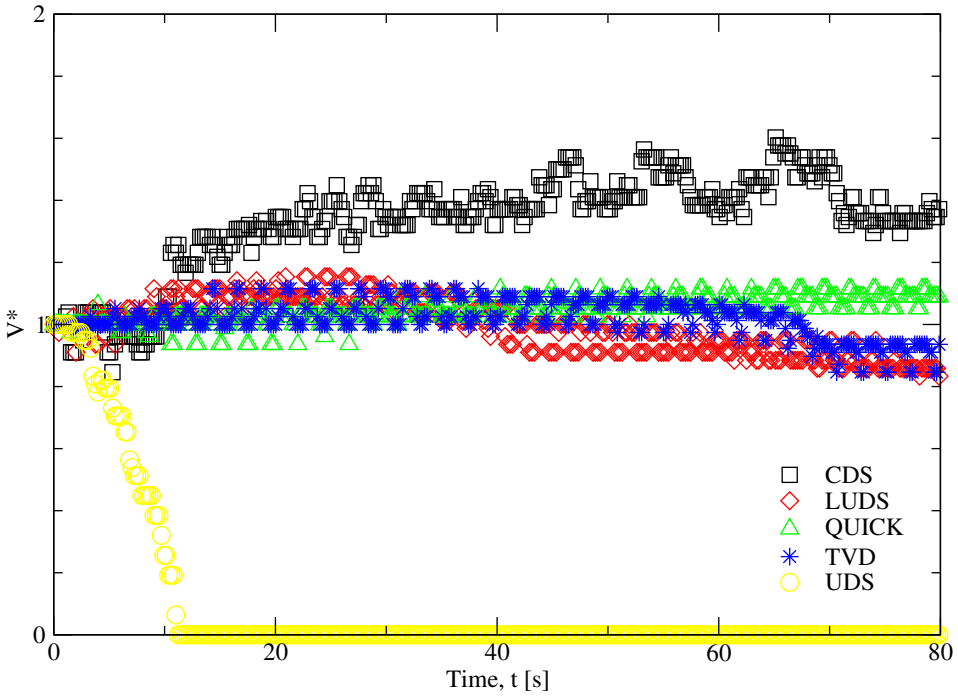


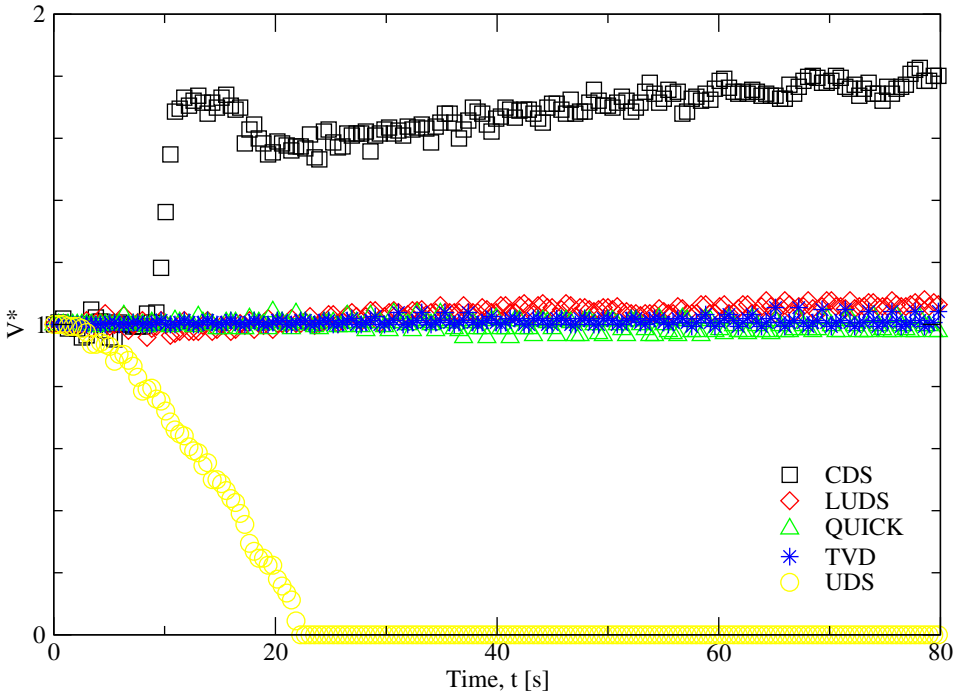
Figure 4.12: Normalized volume averaged fraction obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various schemes

The normalized volume averaged ε plots provide information on the boundedness of the solutions. Since the UDS and TVD solutions are both bounded, their $(\int_V |\varepsilon| dV)^*$ values stay unity. The LUDS solution is more unbounded than the QUICK solution in the current test case. The CDS solution is the most unbounded. At $t = 10$ s, $(\int_V |\varepsilon| dV)^*$ for CDS jumped abruptly. This is due to the first appearance of the mirrored image of the cylinder. Since the absolute value of ε was used, the appearance of the mirrored image resulted in fictional mass creation.

After the grid refinement, $(\int_V |\varepsilon| dV)^*$ lowered in magnitude for all schemes. Along with the observation from the contour plots, it is clear that even though the solution range expanded as the grid is refined (ε becomes more negative), the volume containing the unbounded values decreased. As a result, the overall integration dropped.



(a)



(b)

Figure 4.13: Normalized cylinder volume obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various schemes

The normalized volume plots provide information on the volume conservation of the cylinder. Scatters are used instead of lines to avoid overlapping lines. Point skips of 10 and 20 are applied to the plots for the coarse grid and the refined grid respectively. The values fluctuate violently due to the nature of the stair-step approximation. Such fluctuation can be illustrated in Figure 4.14. Two circles of the same size are placed on the same mesh but at different locations. Note that the approximated volume of the circle on the left is equivalent to 80 CVs and the approximated volume of the circle on the right is equal to only 69 CVs; while the actual size of the circle is 78.5 CVs. The errors are then 2% and 12% respectively. Since this fluctuation is directly related to the size of the control volume, it can be reduced and controlled by mesh refinement. In the illustration, if the mesh is refined uniformly by a factor of two, the error reduces to 0.6%.

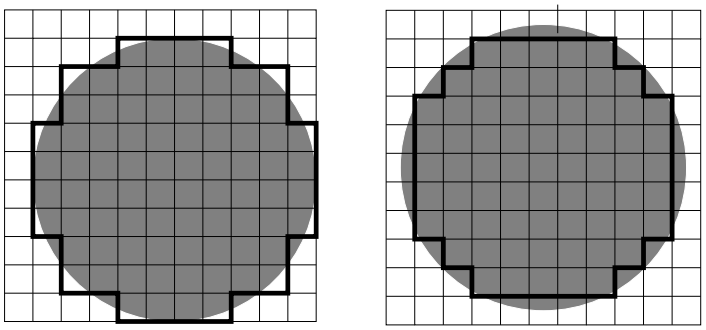


Figure 4.14: Illustration of the fluctuation in the stair-step approximated volume

For the 50×500 mesh, using the 0.5 volume fraction threshold, the cylinder modelled by the UDS vanished in 11 seconds. V^* predicted by LUDS grew in the first 20 seconds, then declined and stabilized to approximately 0.9 of the initial volume. Cylinder volume predicted by QUICK stayed relatively constant in the first 30 seconds then elevated $1.1V_0$, where V_0 is the initial cylinder volume. V^* predicted by TVD stayed relatively constant in the first 60 seconds, then reduced to 0.9.

With grid refinement, the fluctuation in V^* was mitigated as expected. All solutions improved except for the CDS result. The cylinder modelled by the UDS vanished in 22 seconds. It can be concluded that UDS is not appropriate for convection dominated problems due to enormous numerical diffusion. The QUICK and TVD schemes were able to maintain the volume, and V^* remained unity throughout the simulation period. On the whole, the observations from the plots align with the observations from the ε contours.

4.4 Comparison between various TVD schemes

There are unlimited variations of the flux limiter function, thus an unlimited number of TVD schemes. The results produced by different TVD schemes are inconsistent. Aside from the Lin-Lin TVD, three other famous TVD schemes: Min-Mod, SUPERBEE and Van Leer were tested in the two-dimensional cylinder convection case, to promote the understanding of TVD.

Figure 4.15 is the solution ε contour plot at $t = 40$ s and Figure 4.16 is the solution ε contour plot at the end of simulation period for the 50×500 grid. Figure 4.17 is the solution ε contour plot at $t = 40$ s and Figure 4.18 is the solution ε contour plot at the end of simulation period for the 100×1000 grid.

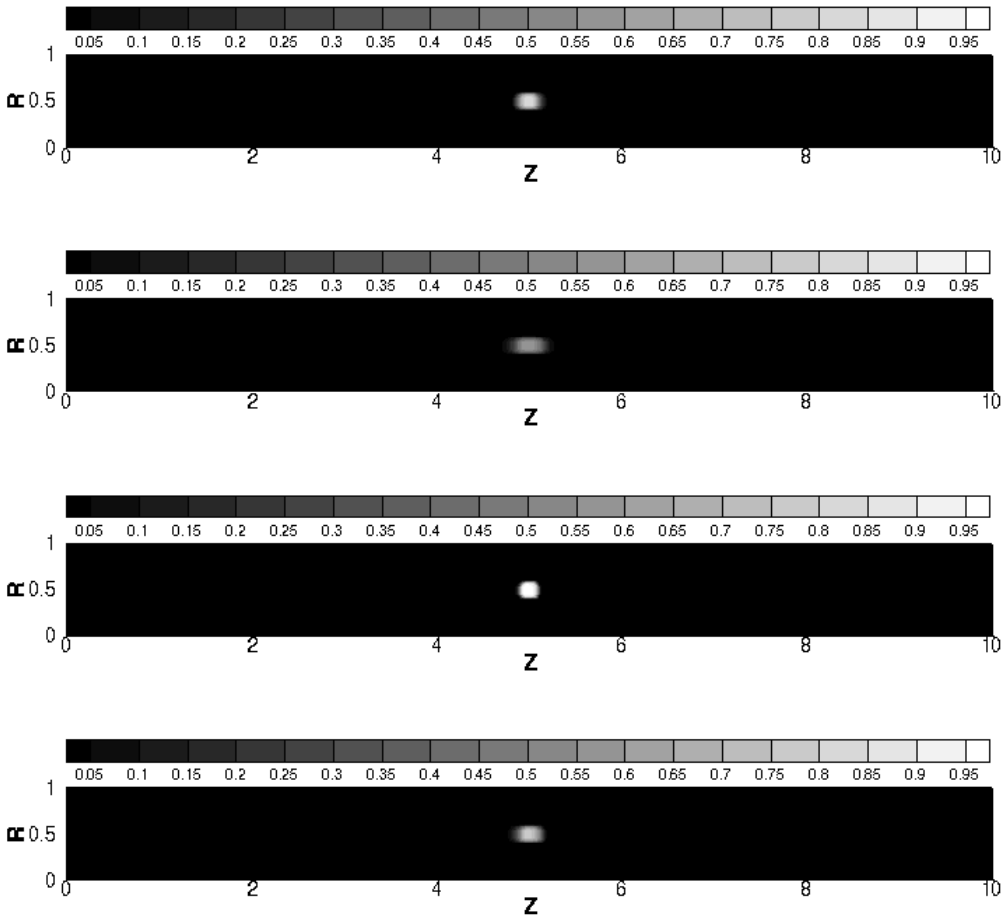


Figure 4.15: Volume fraction contour plots obtained with a 50×500 grid at $t = 40$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively

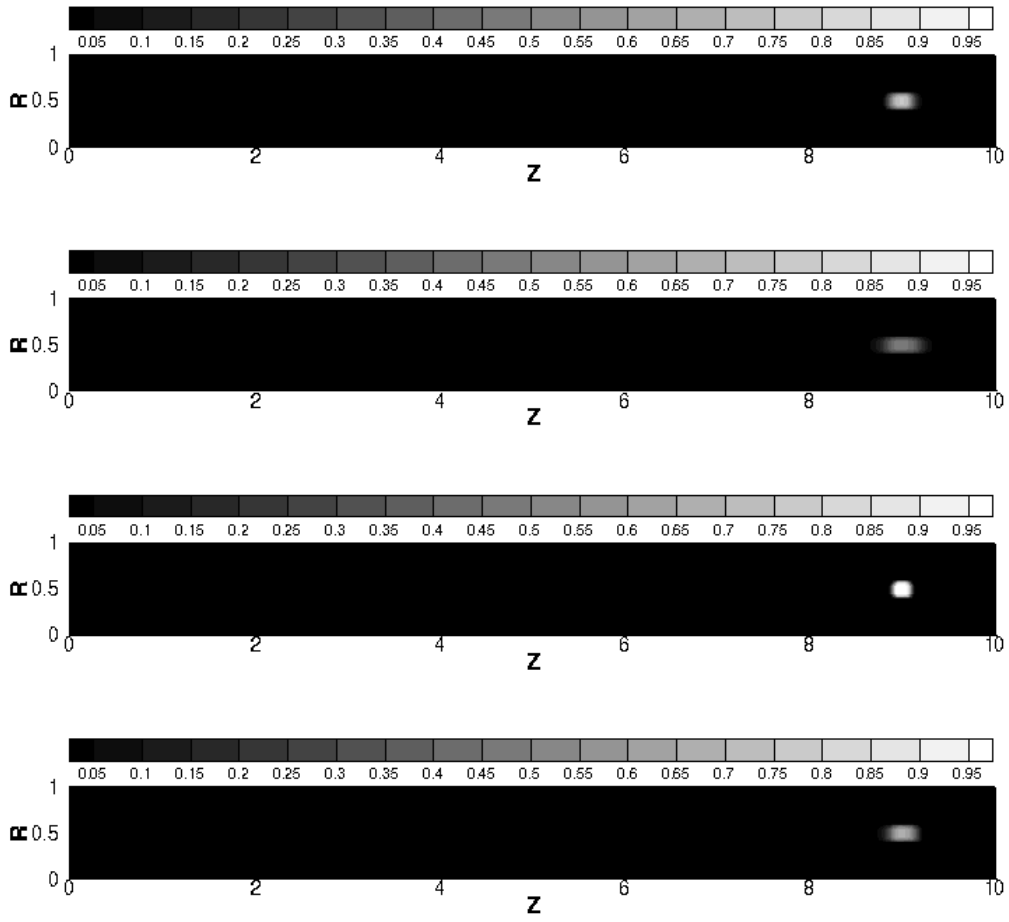


Figure 4.16: Volume fraction contour plots obtained with a 50×500 grid at $t = 80$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively

The performance of these schemes can be ranked according to the resolution of the cylinder. SUPERBEE produced the best resolution of the cylinder, followed by Lin-Lin and Van Leer. SUPERBEE was able to conserve the cylinder perfectly even in the coarser mesh. The Min-Mod TVD scheme produced the worst resolution. From the $r - \psi$ diagram (Figure 3.3) and the NVD diagram (Figure 3.5), it is clear that the SUPERBEE curve is the furthest away from the UDS line, whereas the Min-Mod curve is the closest to the UDS line.

With mesh refinement, cylinder volume conservation and solution resolution improved for all TVD schemes.

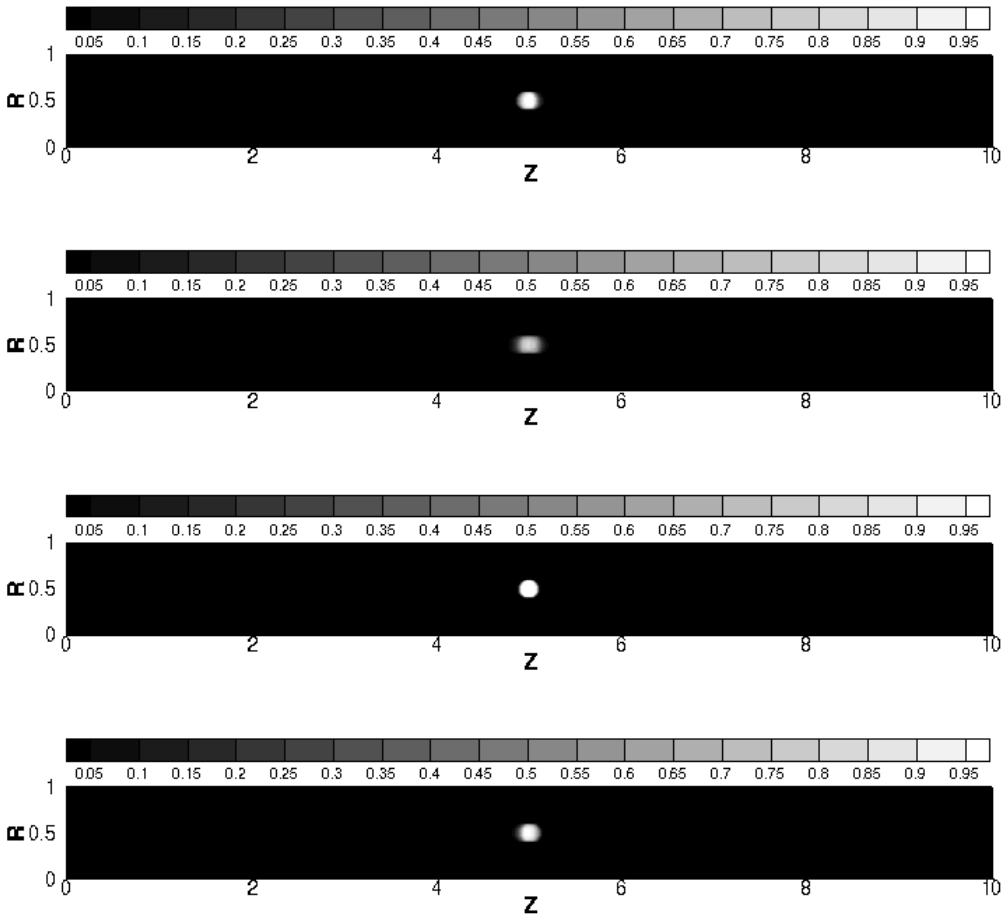


Figure 4.17: Volume fraction contour plots obtained with a 100×1000 grid at $t = 40$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively

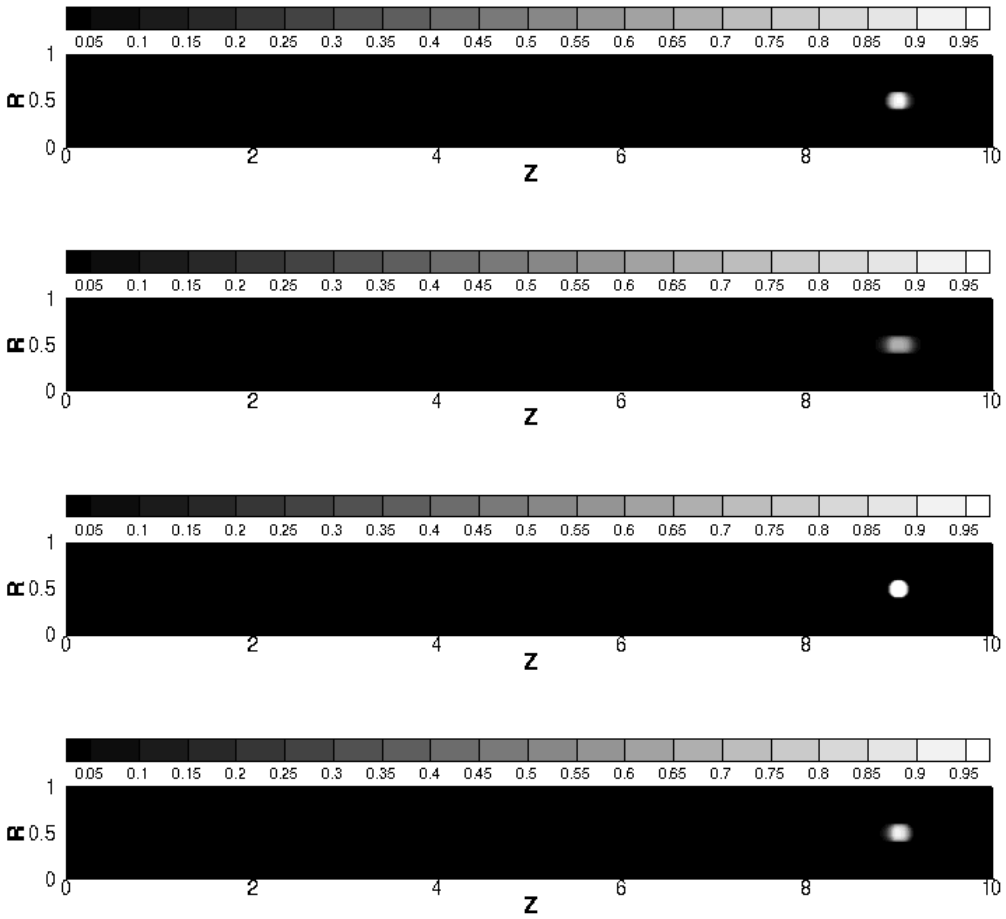
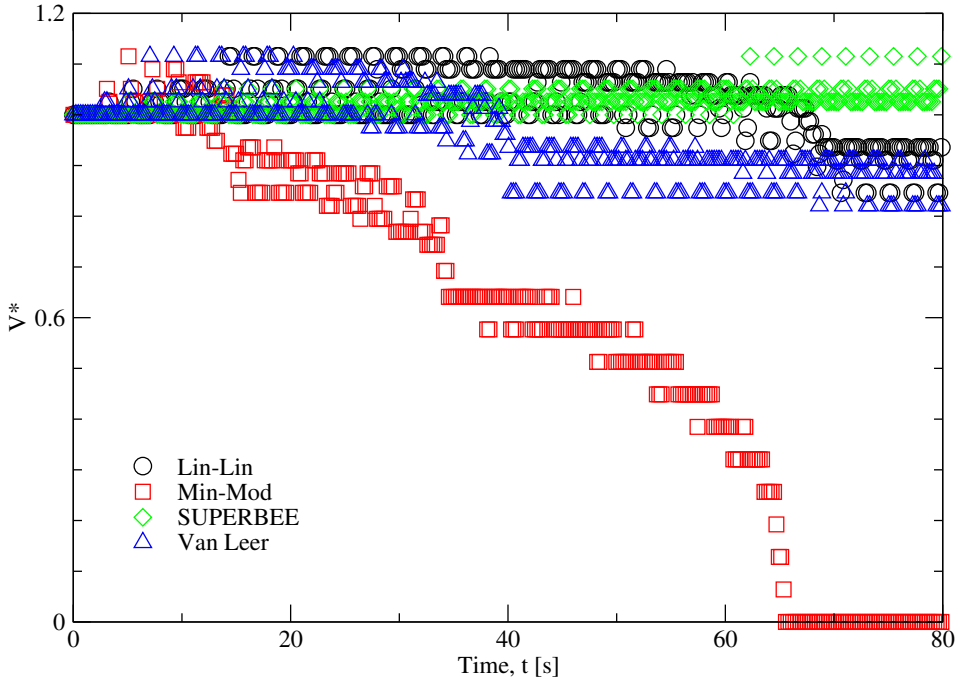


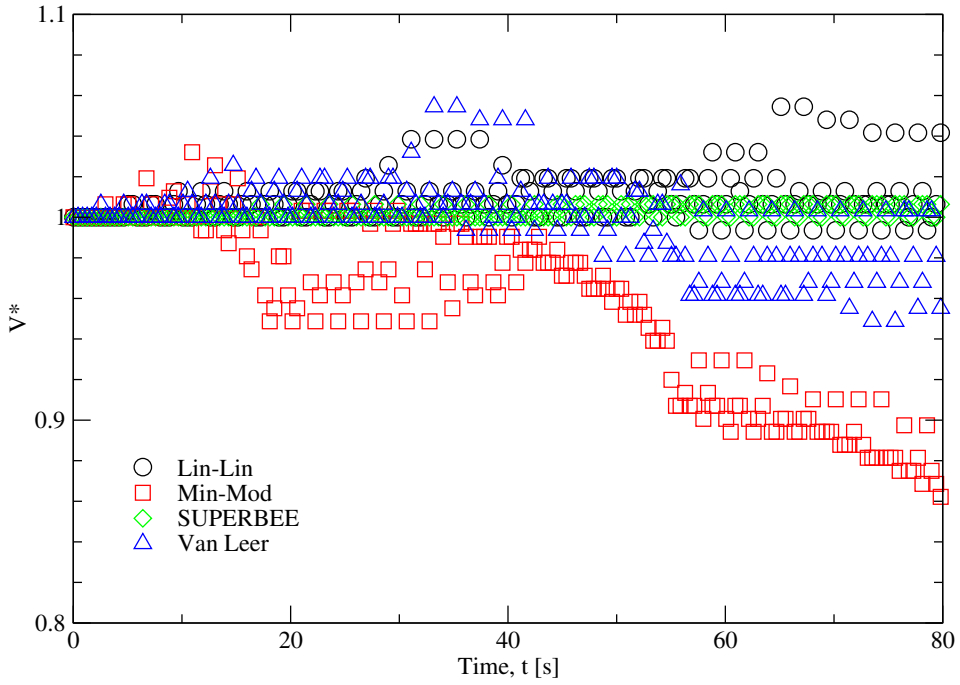
Figure 4.18: Volume fraction contour plots obtained with a 100×1000 grid at $t = 80$ s for: Lin-Lin, Min-Mod, SUPERBEE and Van Leer respectively

Since all TVD schemes produce bounded solutions, the normalized volume averaged fraction plots were not prepared. The V^* plots are presented in Figure 4.19. For the coarser grid, the Min-Mod TVD scheme was unable to hold the cylinder volume, so V^* declined to zero at 65 s. Van Leer result was slightly worse than the Lin-Lin result; it resulted in about 15% reduction in cylinder volume. SUPERBEE gave an increase in the cylinder volume at 80 seconds.

For the refined grid, Min-Mod TVD scheme predicted a volume reduction of approximately 13%. All other TVD schemes resulted in solutions that wiggle near unity.



(a)



(b)

Figure 4.19: Normalized cylinder volume obtained with (a) a 50×500 grid and (b) a 100×1000 grid for various TVD schemes

4.5 Time Complexity Analysis

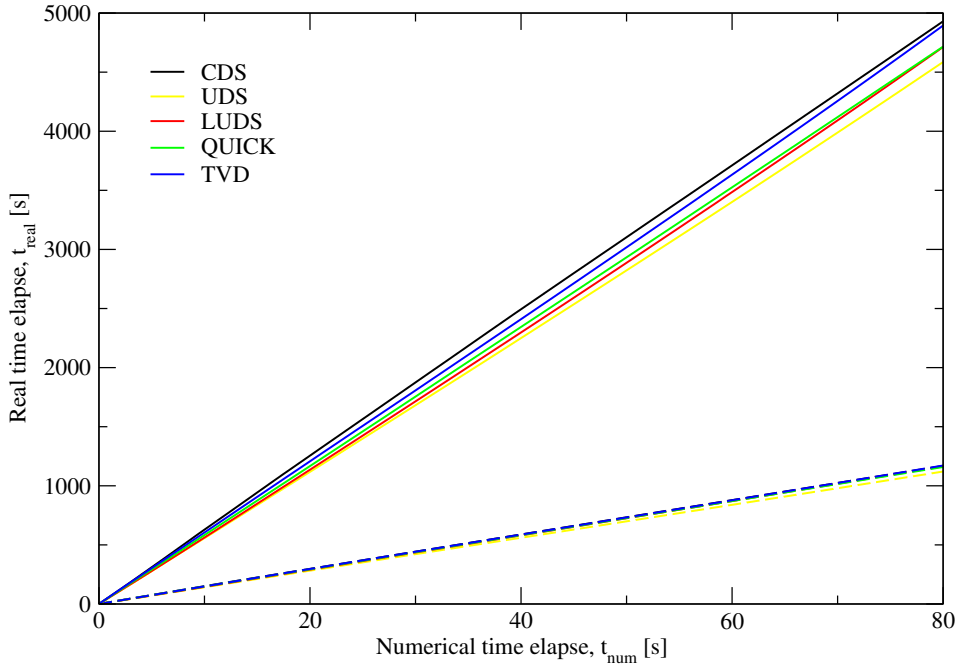
The computational time is a major concern and constraint of CFD simulations. The time performance of a program is determined by the number of basic operations required to solve the problem. In the current studies, as the source term becomes more non-linear, the number of iterations needed to solve the solution matrix at a specific numerical time step increases thus results in rising computation time.

Figure 4.20 illustrated the change in real time elapse with the change in numerical time elapse. Linear correlations are observed for all discretization schemes analyzed. The CDS takes the longest time while providing the worst results. It is confirmed that CDS is inappropriate in solving terms that are biased by flow direction. For the upwind terms, it is generally true that the computational time is in direct relation with the order of the scheme and the accuracy of the output; that is, $t_{QUICK} > t_{LUDS} > t_{UDS}$ as presented in Figure 4.20 (a).

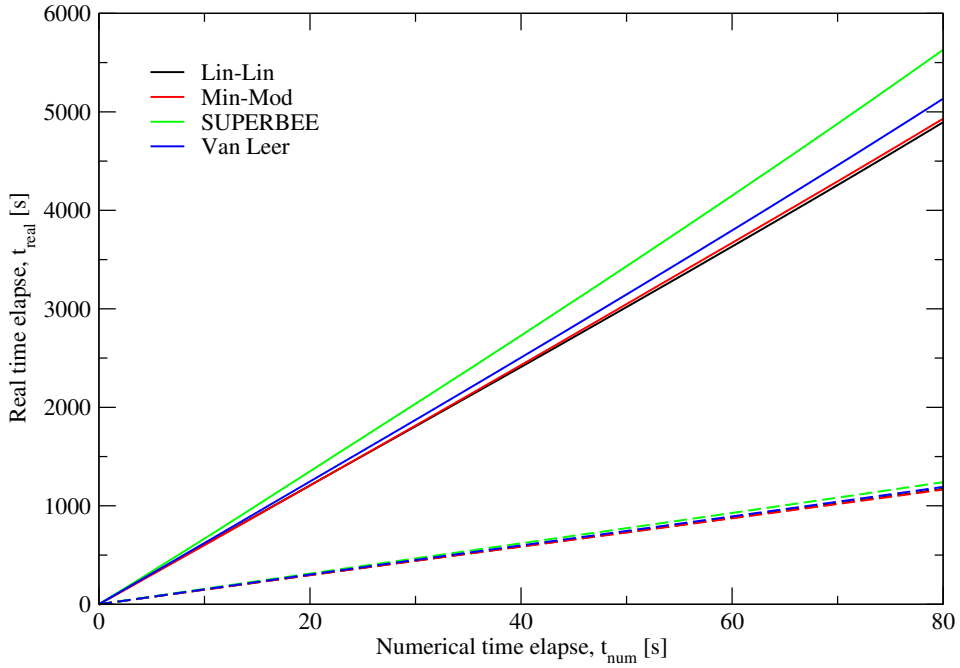
Figure 4.20 (b) provides information on the TVD schemes. The SUPERBEE flux limiter, while providing the most resolved solution, requires the largest amount of computational resources. The Van Leer solution and the Lin-Lin solution have similar levels of accuracy; however, the Lin-Lin flux limiter requires the least computational resources. The time performance of the Min-Mod method is similar to which of the Lin-Lin method, but it produces the most smeared solution as analyzed in the previous section.

It is shown that the time-dependence on non-linearity of the source term is insignificant in coarse meshes and when numerical time elapse is small. However, the difference in computational time accumulates as time propagates. More importantly, the computational time is directly proportional to the number of control volumes present in the mesh. It takes approximately 6×10^{-4} second per control volume per physical time using a time step of $\Delta t = 0.02$ s. Although increasing the time step would decrease the total number of time frames, it would increase the difficulty in convergence at each time step. Namely, more iterations are required to achieve the desired precision and consequently, more time is needed. The time step should also obey the CFL restriction. When CFL number is higher than one, even if one manages to obtain a converged solution, the solution is not physically meaningful.

It should be noted that the current timing only accounted for the calculations performed on a single passive scalar convection-diffusion equation. The solutions of the Navier-Stokes equation and the temperature convection-diffusion equations were not activated.



(a)



(b)

Figure 4.20: The correlation between real time elapse and numerical time elapse for the cylinder advection studies. The dashed lines are for the 50×500 grid and solid lines for the 100×1000 grid: (a) for conventional schemes (b) for TVD schemes

Chapter 5

Conclusions

In the current studies, the algorithms for multiple implicit TVD schemes were implemented in FORTRAN 77 and MATLAB. The codes use the Eulerian-Eulerian volume of fluid (VOF) method to model two-phase fluid flow. The numerical solutions were validated using the one-dimensional convection-diffusion problem, the two-dimensional lids-driven cavity (LDC) problem, as well as the two-dimensional cylinder advection problem. The numerical results were compared with the exact solutions, whenever available, and the solutions from a commercial computational fluid dynamics (CFD) software ANSYS Fluent. For the one-dimensional simulation, the smallest relative error achieved by the code was $1.16 \times 10^{-2}\%$ compared to 4.55% calculated from the Fluent results.

In the LDC study case, the TVD results are in best agreement with the QUICK results. With a grid size of 0.04 m, the maximum relative discrepancy was 5.67%, which reduces to 0.12% when the grid was refined. A large deviation in the results was observed when comparing the results from the three upwind schemes with the Fluent results. This indicates that LUDS implementation in the code and in Fluent are inconsistent. The commercial software is convenient to use and is capable of analysing complicated problems. However, due to its ambiguity in concept definitions and the encapsulation of its implementations, it is doubtable whether the solution could be trusted or not.

In the cylinder advection study investigation, the TVD schemes were proven to provide accurate and bounded results, and their performances exceed which of the conventional upwind schemes (first-order UDS, second-order LUDS and third-order QUICK scheme). A comparison between four different TVD schemes was conducted. The stair-step volume approximation technique was applied. The volume conservation plot shows that the Min-Mod flux limiter function failed to preserve the cylinder

volume in a coarse mesh. The SUPERBEE flux limiter function results in the most resolved solutions and the smallest volume variation (7%) in the coarse grid while consuming the longest computational time. After the grid refinement, all TVD flux limiters except for the Min-Mod flux limiter give 100% volume conservation. The Lin-Lin TVD flux limiter is the most efficient in terms of calculation speed. The difference in time consumption is caused by the level of non-linearity in the matrix source term.

5.1 Future works

- Additional validation cases against the current code should be performed. The results should be compared with some experimental data in addition to other numerical solutions.
- In order to solve more complex problems, calculations in the third dimension must be added to the code. The algorithm is similar to which of the other two dimensions.
- Implementing the TVD algorithm in the Navier-Stokes solver and the second-order interface reconstruction method.
- With surface reconstruction, implementing the surface tension calculations. The code can then be used to solve more realistic problems.
- Expanding the code to adapt the multiphase VOF method instead of just the two-phase VOF method.

Bibliography

[1] Kausik Nandi and AW Date. Validation of fully implicit method for simulation of flows with interfaces using primitive variables. *International Journal of Heat and Mass Transfer*, 52(13):3225–3234, 2009.

[2] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[3] Amanda Wood and Keh-Han Wang. Modeling dam-break flows in channels with 90 degree bend using an alternating-direction implicit based curvilinear hydrodynamic solver. *Computers & Fluids*, 114:254–264, 2015.

[4] W-Y Chang, L-C Lee, H-C Lien, and J-S Lai. Simulations of dam-break flows using free surface capturing method. *Journal of Mechanics*, 24(4):391–403, 2008.

[5] Xueyao Wang, Shi Liu, Changchun Liu, Fan Jiang, Zhihong Li, and Zhengzhong Ma. Capturing moving interfaces by levelset method with ghost technique. *Journal of Thermal Science*, 16(4):376–381, 2007.

[6] Jun-tao HUANG and Hui-sheng ZHANG. A level set method for simulation of rising bubble. *sign*, 1:l9, 2004.

[7] Etienne Guyon and Marie Yvonne Guyon. Taking fluid mechanics to the general public. *Annual review of fluid mechanics*, 46:1–22, 2014.

[8] JF Richardson and WN Zaki. The sedimentation of a suspension of uniform spheres under conditions of viscous flow. *Chemical Engineering Science*, 3(2): 65–73, 1954.

[9] Dale H Beggs, James P Brill, et al. A study of two-phase flow in inclined pipes. *Journal of Petroleum technology*, 25(05):607–617, 1973.

[10] Donald Allen Drew. Mathematical modeling of two-phase flow. *Annual review of fluid mechanics*, 15(1):261–291, 1983.

[11] *ANSYS FLUENT 12.0/12.1 Documentation*, 2009.

[12] H Grosshans, A Movaghar, L Cao, M Oevermann, R-Z Szász, and Laszlo Fuchs. Sensitivity of vof simulations of the liquid jet breakup to physical and numerical parameters. *Computers & Fluids*, 136:312–323, 2016.

[13] William D Fullmer and Christine M Hrenya. The clustering instability in rapid granular and gas-solid flows. *Annual Review of Fluid Mechanics*, 49:485–510, 2017.

[14] Sean McKee, Murilo F Tomé, Valdemir G Ferreira, José A Cuminato, Antonio Castelo, FS Sousa, and Norberto Mangiavacchi. The mac method. *Computers & Fluids*, 37(8):907–930, 2008.

[15] Wojciech Aniszewski, Thibaut Ménard, and Maciej Marek. Volume of fluid (vof) type advection methods in two-phase flow: a comparative study. *Computers & Fluids*, 97:52–73, 2014.

[16] N Samkhaniani and MR Ansari. Numerical simulation of bubble condensation using cf-vof. *Progress in Nuclear Energy*, 89:120–131, 2016.

[17] Orest Shardt, JJ Derksen, and Sushanta K Mitra. Simulations of droplet coalescence in simple shear flow. *Langmuir*, 29(21):6201–6212, 2013.

[18] Ruben Scardovelli and Stéphane Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual review of fluid mechanics*, 31(1):567–603, 1999.

[19] Said Elghobashi. Direct numerical simulation of turbulent flows laden with droplets or bubbles. *Annual Review of Fluid Mechanics*, 51:217–244, 2019.

[20] William J Rider and Douglas B Kothe. Reconstructing volume tracking. *Journal of computational physics*, 141(2):112–152, 1998.

[21] PG Huang, BE Launder, and MA Leschziner. Discretization of nonlinear convection processes: a broad-range comparison of four schemes. *Computer Methods in Applied Mechanics and Engineering*, 48(1):1–24, 1985.

[22] MA Leschziner. Practical evaluation of three finite difference schemes for the computation of steady-state recirculating flows. *Computer Methods in Applied Mechanics and Engineering*, 23(3):293–312, 1980.

[23] Ami Harten. High resolution schemes for hyperbolic conservation laws. *Journal of computational physics*, 49(3):357–393, 1983.

[24] Fue-Sang Lien and MA Leschziner. Upstream monotonic interpolation for scalar transport with application to complex turbulent flows. *International Journal for Numerical Methods in Fluids*, 19(6):527–548, 1994.

[25] Petr A Nikrityuk. *Computational thermo-fluid dynamics: in materials science and engineering*. John Wiley & Sons, 2011.

[26] ND Sandham and Helen C Yee. A numerical study of a class of tvd schemes for compressible mixing layers. 1989.

[27] Kausik Nandi and AW Date. Formulation of fully implicit method for simulation of flows with interfaces using primitive variables. *International Journal of Heat and Mass Transfer*, 52(13):3217–3224, 2009.

[28] S Pirker, A Aigner, and G Wimmer. Experimental and numerical investigation of sloshing resonance phenomena in a spring-mounted rectangular tank. *Chemical engineering science*, 68(1):143–150, 2012.

[29] Bao Hui Wang, Xiang Zhen Yan, and Xiu Juan Yang. Research on numerical simulation of gas storage field in aquifer based on fully implicit method. In *Advanced Materials Research*, volume 383, pages 4467–4474. Trans Tech Publ, 2012.

[30] Anil W Date. *Introduction to computational fluid dynamics*. Cambridge University Press, 2005.

[31] Tomonori Horiuchi, Sanjoy Banerjee, and Hiroyuke Takahira. An improved three-dimensional level set method for gas-liquid two-phase flows. *Journal of Fluids Engineering(Transactions of the ASME)*, 126(4):578–585, 2004.

[32] Murray Rudman. Volume-tracking methods for interfacial flow calculations. *International journal for numerical methods in fluids*, 24(7):671–691, 1997.

[33] Liu Jun. Numerical simulation of flow with moving interface. *PhysicoChemical Hydrodynamics*, 10(5):625, 1988.

[34] JC Martin and WJ Moyce. Part iv. an experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 244(882):312–324, 1952.

[35] Yann Andrillon and Bertrand Alessandrini. A 2d+ t vof fully coupled formulation for the calculation of breaking free-surface flow. *Journal of marine science and technology*, 8(4):159–168, 2004.

[36] Elbridge Gerry Puckett, Ann S Almgren, John B Bell, Daniel L Marcus, and William J Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130 (2):269–282, 1997.

[37] Lixiang Song, Jianzhong Zhou, Qiang Zou, Jun Guo, and Yi Liu. Two-dimensional dam-break flood simulation on unstructured meshes. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*, pages 465–469. IEEE, 2010.

[38] AW Date. Solution of navier-stokes equations on nonstaggered grid at all speeds. *Numerical Heat Transfer, Part B*, 33(4):451–467, 1998.

[39] Chung-Hsiung Lin and CA Lin. Simple high-order bounded convection scheme to model discontinuities. *AIAA journal*, 35(3):563–565, 1997.

[40] Mary L Mason, Lawrence Elwood Putnam, and Richard J Re. The effect of throat contouring on two-dimensional converging-diverging nozzles at static conditions. 1980.

[41] Tae Hoon Yoon and Seok-Koo Kang. Finite volume model for two-dimensional shallow water flows on unstructured grids. *Journal of Hydraulic Engineering*, 130(7):678–688, 2004.

[42] HC Yee, RF Warming, and A Harten. Implicit total variation diminishing (tvd) schemes for steady-state calculations. *Journal of Computational Physics*, 57(3): 327–360, 1985.

[43] Peter K Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM journal on numerical analysis*, 21(5):995–1011, 1984.

[44] Brian Phillip Leonard. Simple high-accuracy resolution program for convective modelling of discontinuities. *International Journal for Numerical Methods in Fluids*, 8(10):1291–1318, 1988.

Appendix A

1D FORTRAN Code

```
C#############################################################
      PROGRAM FD
C#############################################################
C     This program solves the one-dimensional convection-
C     diffusion equation with Dirichlet boundary conditions
C     on both ends. The diffusive term is calculated using
C     CDS and the convective term using CDS, UDS, LUDS,
C     QUICK and TVD (Lin-Lin).The solutions obtained are
C     compared with the exact solution. The code is adapted
C     from Milovan Peric's code in the book "Computational
C     Methods for Fluid Dynamics".
C#############################################################
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX),U(NX),
     *       FluxC(NX)
      DIMENSION FIEX(NX)
      CHARACTER FILOUT*10,STRING1*45,STRING2*40,STRING3*34
C
C.....OPEN FILES
      PRINT *,' ENTER OUTPUT FILE NAME:   '
      READ(*,1) FILOUT
    1 FORMAT(A10)
      OPEN (UNIT=8,FILE=FILOUT)
      OPEN (UNIT=1,FILE='rez.dat')
C
C.....READ INPUT DATA
      PRINT *,' ENTER: DEN(SITY), VEL(OCITY), DIF(FUSION COEFF.)   '
      READ(*,*) DEN,VEL,DIF
      PRINT *,' ENTER BOUNDARY VALUES: FI0, FIN   '
      READ(*,*) FI0,FIN
      STRING1=' CHOOSE CONVECTION SCHEME: 1 - CDS, 2 - UDS, '
      STRING2='3 - UWDS, 4 - PDS, 5 - LUDS, 6 - QUICK, '
      STRING3='7 - DEFFERED CORRECTION, 8 - TVD   '
      PRINT *, STRING1//STRING2//STRING3
      READ(*,*) IC
C
C.....DEFINE THE GRID: EX - EXPANSION FACTOR;
```

```
C       N - NUMBER OF NODES INCL. BOUNDARY ONES
        PRINT *,' ENTER: XMIN, XMAX, EX, N '
        READ(*,*) XMIN,XMAX,EX,N
        NM=N-1
        IF(EX.EQ.1.) THEN
          DX=(XMAX-XMIN)/REAL(N-1)
        ELSE
          DX=(XMAX-XMIN)*(1.-EX)/(1.-EX**(N-1))
        ENDIF
        X(1)=XMIN
        DO I=2,N
          X(I)=X(I-1)+DX
          DX=DX*EX
        ENDDO
C
C.....INITIALIZE FIELDS
        DO I=1,N
          FI(I)=0.
          U (I)=VEL
          FluxC(I)=DEN*U(I)
        ENDDO
        FI(1)=FI0
        FI(N)=FIN
        DENVEL=DEN*VEL
        ZERO=0.
C
        DO I=2,NM
          DENVEL_e=(FluxC(I)+FluxC(I+1))/2.
          DENVEL_w=(FluxC(I)+FluxC(I-1))/2.
          PEcell_e=DENVEL_e*(X(I+1)-X(I))/DIF
          PEcell_w=DENVEL_w*(X(I)-X(I-1))/DIF
C
C.....CENTRAL DIFFERENCE CONVECTION APPROX. (CDS)
          IF(IC.EQ.1) THEN
            AEC= DENVEL_e/2.
            AWC=-DENVEL_w/2.
C
C.....UPWIND CONVECTION APPROX. (UDS)
          ELSEIF(IC.EQ.2 .OR. IC.GE.4) THEN
            AEC= AMIN1(DENVEL_e,ZERO)
            AWC=-AMAX1(DENVEL_w,ZERO)
C
C.....UPWIND WEIGHTED DIFFERENCE APPROX. (UWDS)
          ELSEIF(IC.EQ.3) THEN
            AEC_cds= DENVEL_e/2.
            AWC_cds=-DENVEL_w/2.
            AEC_uds= AMIN1(DENVEL_e,ZERO)
            AWC_uds=-AMAX1(DENVEL_w,ZERO)
C
            gama_e=(PEcell_e*PEcell_e)/(5+PEcell_e*PEcell_e)
            gama_w=(PEcell_w*PEcell_w)/(5+PEcell_w*PEcell_w)
C
            AEC=(1.-gama_e)*AEC_cds+gama_e*AEC_uds
            AWC=(1.-gama_w)*AWC_cds+gama_w*AWC_uds
C
          ENDIF
```

```
C
C.....CENTRAL DIFFERENCE DIFFUSION APPROX. (CDS)
          AED=-DIF/(X(I+1)-X(I))
          AWD=-DIF/(X(I)-X(I-1))
C
          IF(IC.EQ.4) THEN
              AED=AED*AMAX1(0.,(1.-0.1*DABS(PEcell_e))**5)
              AWD=AWD*AMAX1(0.,(1.-0.1*DABS(PEcell_w))**5)
          ENDIF
C
C.....ASSEMBLE COEFFICIENT MATRIX
          AE(I)=AEC+AED
          AW(I)=AWC+AWD
          AP(I)=-AW(I)-AE(I)
C
          Q(I)=0.
        ENDDO
C
C.....BOUNDARY CONDITIONS
        Q(2)=Q(2)-AW(2)*FI(1)
        AW(2)=0.
        Q(NM)=Q(NM)-AE(NM)*FI(N)
        AE(NM)=0.
C
C.....SOLVE EQUATION SYSTEM
        PRINT *,' CHOOSE SOLVER: 1 - JACOBI, 2 - GS, 3 - GSOR, 4 - TDMA'
        READ(*,*) IS
C         IF(IC .GT. 4) IS=3
        IF(IS.EQ.1) THEN
          CALL JACOBI(IC)
        ELSEIF(IS.EQ.2) THEN
          CALL GS(IC)
        ELSEIF(IS.EQ.3) THEN
          CALL GSOR(IC)
        ELSEIF(IS.EQ.4) THEN
          CALL TDMA(IC)
        ENDIF
C
C.....CALCULATE EXACT SOLUTION AND ERROR NORM
        ERROR=0.
        FIEX(1)=FI0
        FIEX(N)=FIN
        PE=DENVEL*(XMAX-XMIN)/DIF
        print *, 'Pe',PE
        RX=1./(XMAX-XMIN)
        DO I=2,NM
          FIEX(I)=FI0+((DEXP(PE*X(I)*RX)-1.)/(DEXP(PE)-1.))*(FIN-FI0)
          print *, 'FIEX(I)',I,FIEX(I)
C
          ERROR=ERROR+ABS(FIEX(I)-FI(I))
        ENDDO
        ERROR=ERROR/REAL(N)
C
C.....PRINT THE RESULT
        WRITE(8,*) ' '
        WRITE(8,*) ' PECLET NUMBER:  PE = ',PE
```

```
      WRITE(8,*) ' LOcal PECLET NUMBER:  PE = ',PE*DX/(XMAX-XMIN)
      WRITE(8,*) ' ERROR NORM = ',ERROR
      IF(IC.EQ.1) WRITE(8,*) ' CDS USED FOR CONVECTION '
      IF(IC.EQ.2) WRITE(8,*) ' UDS USED FOR CONVECTION '
      IF(IC.EQ.3) WRITE(8,*) ' UWDS USED FOR CONVECTION '
      IF(IC.EQ.4) WRITE(8,*) ' PDS USED FOR CONVECTION '
      IF(IC.EQ.5) WRITE(8,*) ' LUDS USED FOR CONVECTION '
      IF(IC.EQ.6) WRITE(8,*) ' QUICK USED FOR CONVECTION '
      IF(IC.EQ.7) WRITE(8,*) ' DEFERRED CORRECTION USED FOR CONVECTION '
      IF(IC.EQ.8) WRITE(8,*) ' TVD USED FOR CONVECTION '
      IF(IS.EQ.1) WRITE(8,*) ' JACOBI SOLVER '
      IF(IS.EQ.2) WRITE(8,*) ' GAUSS-SEIDEL SOLVER '
      IF(IS.EQ.3) WRITE(8,*) ' GSOR SOLVER '
      IF(IS.EQ.4) WRITE(8,*) ' TDMA SOLVER '
      WRITE(8,*) '  '
      WRITE(8,*) '      X          FI_EXACT          FI          ERROR '
      WRITE(8,*) '  '
      DO I=1,N
        WRITE(8,65) X(I),FIEX(I),FI(I),FIEX(I)-FI(I)
        WRITE(1,65) X(I),FIEX(I),FI(I),FIEX(I)-FI(I)
      ENDDO
   65 FORMAT(1P4E16.5)
      STOP
      END
C
C###############################################################
      SUBROUTINE TDMA(IC)
C###############################################################
C     INITIAL VALUES OF VARIABLES BPR(I) AND V(I) ASSUMED ZERO!
C
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX),FluxC(NX)
      DIMENSION BPR(NX),V(NX),Q2(NX)
C
      DO I=1,NX
        Q2(I)=Q(I)
      ENDDO
C
      DO IT=1,1000
C
C.....CALCULATE NEW SOLUTION
        DO I=3,NM
          DENVEL_e=(FluxC(I)+FluxC(I+1))/2.
          DENVEL_w=(FluxC(I)+FluxC(I-1))/2.
C
C.....UPDATE NEW SOURCE TERM
          IF(IC .GE. 5) THEN
            CALL GETQ(IC,I,Q2(I),DENVEL_e,DENVEL_w)
          ENDIF
C
        ENDDO
C
C.....CALCULATE 1./U_P (BPR) AND MODIFIED SOURCE TERM (V)
        DO I=2,NM
          BPR(I)=1./(AP(I)-AW(I)*AE(I-1)*BPR(I-1))
```

```
            V(I)=Q(I)-AW(I)*V(I-1)*BPR(I-1)
          ENDDO
C
C.....CALCULATE VARIABLE VALUES - BACKWARD SUBSTITUTION
          DO I=NM,2,-1
            FI(I)=(V(I)-AE(I)*FI(I+1))*BPR(I)
          ENDDO
C
C.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
          CALL GETRES(RES,IT)
          IF(RES.LT.1.E-15) RETURN
C
        ENDDO
        print *, 'ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED'
        RETURN
        END
C
C#############################################################
        SUBROUTINE JACOBI(IC)
C#############################################################
        include 'float.h'
        PARAMETER (NX=200)
        COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX),FluxC(NX)
        DIMENSION FIO(NX),Q2(NX)
C
        DO I=1,NX
          Q2(I)=Q(I)
        ENDDO
C
        DO IT=1,1000
C
C.....SAVE OLD SOLUTION
          DO I=1,N
            FIO(I)=FI(I)
          ENDDO
C
          DO I=3,NM
            DENVEL_e=(FluxC(I)+FluxC(I+1))/2.
            DENVEL_w=(FluxC(I)+FluxC(I-1))/2.
C
C.....UPDATE NEW SOURCE TERM
            IF(IC .GE. 5) THEN
              CALL GETQ(IC,I,Q2(I),DENVEL_e,DENVEL_w)
            ENDIF
C
          ENDDO
C
C.....CALCULATE NEW SOLUTION
          DO I=2,NM
            FI(I)=(-AE(I)*FIO(I+1)-AW(I)*FIO(I-1)+Q(I))/AP(I)
          ENDDO
C
C.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
          CALL GETRES(RES,IT)
          IF(RES.LT.1.E-4) RETURN
C
```

```
      ENDDO
      print *, 'ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED'
      RETURN
      END
C
C#############################################################
      SUBROUTINE GS(IC)
C#############################################################
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX),FluxC(NX)
      DIMENSION Q2(NX)
C
      DO I=1,NX
        Q2(I)=Q(I)
      ENDDO
C
      DO IT=1,1000
C
C.....CALCULATE NEW SOLUTION
        DO I=3,NM
          DENVEL_e=(FluxC(I)+FluxC(I+1))/2.
          DENVEL_w=(FluxC(I)+FluxC(I-1))/2.
C
C.....UPDATE NEW SOURCE TERM
          IF(IC .GE. 5) THEN
            CALL GETQ(IC,I,Q2(I),DENVEL_e,DENVEL_w)
          ENDIF
C
        ENDDO
C
        DO I=2,NM
          FI(I)=(-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I))/AP(I)
        ENDDO
C
C.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
        CALL GETRES(RES,IT)
        IF(RES.LT.1.E-4) RETURN
C
      ENDDO
      print *, 'ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED'
      RETURN
      END
C
C#############################################################
      SUBROUTINE GSOR(IC)
C#############################################################
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX),FluxC(NX)
      DIMENSION Q2(NX)
C
      PRINT *,' ENTER OVERRELAXATION PARAMETER: OM   '
      READ(*,*) OM
C
      DO I=1,NX
```

```
          Q2(I)=Q(I)
        ENDDO
C
        DO IT=1,1000
C
C.....CALCULATE NEW SOLUTION
          DO I=3,NM
            DENVEL_e=(FluxC(I)+FluxC(I+1))/2.
            DENVEL_w=(FluxC(I)+FluxC(I-1))/2.
C
C.....UPDATE NEW SOURCE TERM
            IF(IC .GE. 5) THEN
              CALL GETQ(IC,I,Q2(I),DENVEL_e,DENVEL_w)
            ENDIF
C
          ENDDO
C
          DO I=2,NM
            FI(I)=FI(I)+
     *            OM*((-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I))/AP(I)-FI(I))
          ENDDO
C
C.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
          CALL GETRES(RES,IT)
          IF(RES.LT.1.E-15) RETURN
C
        ENDDO
        print *, 'ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED'
        RETURN
        END
C
C#############################################################
        SUBROUTINE few_pm(fe_p,fe2_p,IC)
C#############################################################
        include 'float.h'
C
C.....LUDS
        IF(IC .EQ. 5) THEN
          fe2_p=fe_p/2.
C
C.....QUICK
        ELSEIF(IC .EQ. 6) THEN
          fe2_p=3./8.-fe_p/4.
C
C.....TVD
        ELSEIF(IC .EQ. 8) THEN
          IF(fe_p .LT. 0. .OR. fe_p .GT. 1.) THEN
            fe2_p=0.
          ELSEIF(fe_p .GE. 0. .AND. fe_p .LE. 0.3) THEN
            fe2_p=fe_p
          ELSEIF(fe_p .GT. 0.3 .AND. fe_p .LE. 5./6.) THEN
            fe2_p=3./8.-fe_p/4.
          ELSEIF(fe_p .GT. 5./6. .AND. fe_p .LE. 1.) THEN
            fe2_p=1.-fe_p
          ENDIF
        ENDIF
```

```
      RETURN
      END
C
C#############################################################
      SUBROUTINE GETQ(IC,I,Q2,DENVEL_e,DENVEL_w)
C#############################################################
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX)
C
      ZERO=0.
      small=1e-15
C
C.....Def Cor. Scheme
      IF(IC .EQ. 7) THEN
        Q(I)=Q2+(FI(I  )*AMAX1( DENVEL_e,ZERO)
     *          + FI(I+1)*AMAX1(-DENVEL_e,ZERO)
     *          - FI(I+1)*0.5*DENVEL_e
     *          - FI(I-1)*AMAX1( DENVEL_w,ZERO)
     *          - FI(I  )*AMAX1(-DENVEL_w,ZERO)
     *          + FI(I-1)*0.5*DENVEL_w)
C
C.....LUDS, QUICK, TVD
      ELSEIF(IC .GE. 5) THEN
C
C.....Date p. 64, f(\eta)=f((Fi_U - Fi_UU)/(Fi_D - Fi_UU))
        fe_p=(FI(I  )-FI(I-1))/(FI(I+1)-FI(I-1)+small)
        fe_m=(FI(I+1)-FI(I+2))/(FI(I  )-FI(I+2)+small)
C
        fw_p=(FI(I-1)-FI(I-2))/(FI(I  )-FI(I-2)+small)
        fw_m=(FI(I  )-FI(I+1))/(FI(I-1)-FI(I+1)+small)
C
        CALL few_pm(fe_p,fe2_p,IC)
        CALL few_pm(fe_m,fe2_m,IC)
        CALL few_pm(fw_p,fw2_p,IC)
        CALL few_pm(fw_m,fw2_m,IC)
C
        IF(I .EQ. NM) THEN
          Q(I)=Q2+0.5*(DENVEL_e+DABS(DENVEL_e))*fe2_p*(FI(I-1)-FI(I+1))
C     *            +0.5*(DENVEL_e-DABS(DENVEL_e))*fe2_m*(FI(I+2)-FI(I  ))
     *            +0.5*(DENVEL_w+DABS(DENVEL_w))*fw2_p*(FI(I  )-FI(I-2))
     *            +0.5*(DENVEL_w-DABS(DENVEL_w))*fw2_m*(FI(I-1)-FI(I+1))
C
        ELSE
          Q(I)=Q2+0.5*(DENVEL_e+DABS(DENVEL_e))*fe2_p*(FI(I-1)-FI(I+1))
     *            +0.5*(DENVEL_e-DABS(DENVEL_e))*fe2_m*(FI(I+2)-FI(I  ))
     *            +0.5*(DENVEL_w+DABS(DENVEL_w))*fw2_p*(FI(I  )-FI(I-2))
     *            +0.5*(DENVEL_w-DABS(DENVEL_w))*fw2_m*(FI(I-1)-FI(I+1))
        ENDIF
      ENDIF
      RETURN
      END
C
C#############################################################
      SUBROUTINE GETRES(RES,IT)
C#############################################################
```

```
      include 'float.h'
      PARAMETER (NX=200)
      COMMON N,NM,FI(NX),AE(NX),AW(NX),AP(NX),Q(NX),X(NX)
C
      RES=0.
C
      DO I=2,NM
        RES=RES+ABS(-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I)-AP(I)*FI(I))
      ENDDO
      WRITE(8,*) IT,' ITER., RES = ',RES
      WRITE(*,*) IT,' ITER., RES = ',RES
      RETURN
      END
C
```

Subfile 'float.h'

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

Appendix B

1D MATLAB Code

```
clc; clear;
%#############################################################
% PROGRAM FD
% This program solves the one-dimensional convection-
% diffusion equation with Dirichlet boundary conditions
% on both ends. The diffusive term is calculated using
% CDS and the convective term using CDS, UDS, LUDS,
% QUICK and TVD (Lin-Lin).The solutions obtained are
% compared with the exact solution.
%#############################################################
%
% OPEN FILES
FILOUT=input('ENTER OUTPUT FILE NAME: ','s');
fileID1=fopen(FILOUT,'w');
fileID2=fopen('rez_m.dat','w');
%
% READ INPUT DATA
TMP=str2num(input('ENTER: DEN(SITY), VEL(OCITY),...
    DIF(FUSION COEFF.)\n','s'));
DEN=TMP(1); VEL=TMP(2); DIF=TMP(3);
TMP=str2num(input('ENTER BOUNDARY VALUES: FI0, FIN \n','s'));
FI0=TMP(1); FIN=TMP(2);
METHODS=["CDS","UDS","UWDS","PDS","LUDS","QUICK",...
    "DEFERRED CORRECTION","TVD"];
PROMPT='CHOOSE CONVECTION SCHEME:';
for I=1:size(strlength(METHODS),2)
    if I == size(strlength(METHODS),2)
        PROMPT=[PROMPT,num2str(I),' - ',METHODS(I),'\n'];
    else
        PROMPT=[PROMPT,num2str(I),' - ',METHODS(I),',  '];
    end
end
IC=input(char(strjoin(PROMPT)));
%
% DEFINE THE GRID: EX - EXPANSION FACTOR;
% N - NUMBER OF NODES INCL. BOUNDARY ONES
TMP=str2num(input('ENTER: XMIN, XMAX, EX, N\n','s'));
XMIN=TMP(1); XMAX=TMP(2); EX=TMP(3); N=TMP(4);
```

```
NM=N-1;
if EX == 1
    DX=(XMAX-XMIN)/NM;
else
    DX=(XMAX-XMIN)*(1-EX)/(1-EX^NM);
end
X(1)=XMIN;
for I=2:N
    X(I)=X(I-1)+DX;
    DX=DX*EX;
end
%
% INITIALIZE FIELDS
for I=1:N+1
    FI(I)=0;
    U(I)=VEL;
    FluxC(I)=DEN*U(I);
end
FI(1)=FI0; FI(N)=FIN; DENVEL=DEN*VEL;
%
for I=2:NM
    DENVEL_e=(FluxC(I)+FluxC(I+1))/2;
    DENVEL_w=(FluxC(I)+FluxC(I-1))/2;
    PEcell_e=DENVEL_e*(X(I+1)-X(I))/DIF;
    PEcell_w=DENVEL_w*(X(I)-X(I-1))/DIF;
    %
    % CENTRAL DIFFERENCE CONVECTOIN APPROX. (CDS)
    if IC == 1
        AEC= DENVEL_e/2;
        AWC=-DENVEL_w/2;
        %
        % UPWIND CONVECTION APPROX. (UDS)
    elseif IC == 2 || IC >= 4
        AEC= min(DENVEL_e,0);
        AWC=-max(DENVEL_w,0);
        %
        % UPWIND WEIGHTED DIFFERENCE APPROX. (UWDS)
    elseif IC == 3
        AEC_cds= DENVEL_e/2;
        AWC_cds=-DENVEL_w/2;
        AEC_uds= min(DENVEL_e,0);
        AWC_uds=-max(DENVEL_w,0);
        %
        gama_e=(PEcell_e*PEcell_e)/(5+PEcell_e*PEcell_e);
        gama_w=(PEcell_w*PEcell_w)/(5+PEcell_w*PEcell_w);
        %
        AEC=(1-gama_e)*AEC_cds+gama_e*AEC_uds;
        AWC=(1-gama_w)*AWC_cds+gama_w*AWC_uds;
    end
    %
    % CENTRAL DIFFERENCE DIFFUSION APPROX. (CDS)
    AED=-DIF/(X(I+1)-X(I));
    AWD=-DIF/(X(I)-X(I-1));
    %
    if IC == 4
        AED=AED*max(0,(1-0.1*abs(PEcell_e))^5);
```

```
        AWD=AWD*max(0,(1-0.1*abs(PEcell_w))^5);
    end
    %
    % ASSEMBLE COEFFICIENT MATRIX
    AE(I)=AEC+AED;
    AW(I)=AWC+AWD;
    AP(I)=-AW(I)-AE(I);
    %
    Q(I)=0;
end
%
% BOUNDARY CONDITIONS
Q(2)=Q(2)-AW(2)*FI(1);
AW(2)=0;
Q(NM)=Q(NM)-AE(NM)*FI(N);
AE(NM)=0;
%
% SOLVE EQUATION SYSTEM
SOLVERS=["JACOBI","GAUSS-SEIDEL","GSOR","TDMA"];
PROMPT=' CHOOSE SOLVER:  ';
for I=1:size(strlength(SOLVERS),2)
    if I == size(strlength(SOLVERS),2)
        PROMPT=[PROMPT,num2str(I),' - ',SOLVERS(I),'\n'];
    else
        PROMPT=[PROMPT,num2str(I),' - ',SOLVERS(I),',  '];
    end
end
IS=input(char(strjoin(PROMPT)));
% if IC > 4; IS=3; end
switch IS
    case 1
        [FI]=JACOBI(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP);
    case 2
        [FI]=GS(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP);
    case 3
        [FI]=GSOR(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP);
    case 4
        [FI]=TDMA(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP);
end
%
% CALCULATE EXACT SOLUTION AND ERROR NORM
ERROR=0;
FIEX(1)=FI0; FIEX(N)=FIN;
PE=DENVEL*(XMAX-XMIN)/DIF;
fprintf('Pe %f\n',PE);
RX=1/(XMAX-XMIN);
for I=2:NM
    FIEX(I)=FI0+((exp(PE*X(I)*RX)-1)/(exp(PE)-1))*(FIN-FI0);
    fprintf('FIEX(I) %i  %.16E\n',I,FIEX(I));
    %
    ERROR=ERROR+abs(FIEX(I)-FI(I));
end
ERROR=ERROR/N;
%
% PRINT THE RESULT
fprintf(fileID1,'\n PECLET NUMBER: PE = %f\n',PE);
```

```
fprintf(fileID1,'LOCAL PECLET NUMBER: PE = %f\n',PE*DX/(XMAX-XMIN));
fprintf(fileID1,'ERROR NORM = %f\n',ERROR);
fprintf(fileID1,'%s USED FOR CONVECTION\n',METHODS(IC));
fprintf(fileID1,'%s SOLVER\n\n',SOLVERS(IS));
fprintf(fileID1,'      X            FI_EXACT            FI            ERROR...
    \n\n');
for I=1:N
    fprintf(fileID1,'%16.5E %16.5E %16.5E %16.5E\n',X(I),FIEX(I),...
        FI(I),FIEX(I)-FI(I));
    fprintf(fileID2,'%16.5E %16.5E %16.5E %16.5E\n',X(I),FIEX(I),...
        FI(I),FIEX(I)-FI(I));
end
```

Subfunction 'GETQ.m'

```
function [Q]=GETQ(IC,I,Q2,FI,NM,DENVEL_e,DENVEL_w)
%###############################################################
% SUBROUTINE GETQ(IC,I,Q2,DENVEL_e,DENVEL_w)
%###############################################################
%
small=1e-15;
%.....Def Cor. Scheme
if IC == 7
    Q=Q2+(FI(I  )*max( DENVEL_e,0)+FI(I+1)*max(-DENVEL_e,0)...
        - FI(I+1)*0.5* DENVEL_e  -FI(I-1)*max( DENVEL_w,0)...
        - FI(I  )*max(-DENVEL_w,0)+FI(I-1)*0.5* DENVEL_w);
    %
    %.....LUDS, QUICK, TVD
elseif IC >= 5
    %
    %.....Date p. 64, f(\eta)=f((Fi_U - Fi_UU)/(Fi_D - Fi_UU))
    %
    fe_p=(FI(I  )-FI(I-1))/(FI(I+1)-FI(I-1)+small);
    fe_m=(FI(I+1)-FI(I+2))/(FI(I  )-FI(I+2)+small);
    %
    fw_p=(FI(I-1)-FI(I-2))/(FI(I  )-FI(I-2)+small);
    fw_m=(FI(I  )-FI(I+1))/(FI(I-1)-FI(I+1)+small);
    %
    fe2_p=few_pm(fe_p,IC);
    fe2_m=few_pm(fe_m,IC);
    fw2_p=few_pm(fw_p,IC);
    fw2_m=few_pm(fw_m,IC);
    %
    if I == NM
        Q=Q2+0.5*(DENVEL_e+abs(DENVEL_e))*fe2_p*(FI(I-1)-FI(I+1))...
            +0.5*(DENVEL_w+abs(DENVEL_w))*fw2_p*(FI(I  )-FI(I-2))...
            +0.5*(DENVEL_w-abs(DENVEL_w))*fw2_m*(FI(I-1)-FI(I+1));
    else
        Q=Q2+0.5*(DENVEL_e+abs(DENVEL_e))*fe2_p*(FI(I-1)-FI(I+1))...
            +0.5*(DENVEL_e-abs(DENVEL_e))*fe2_m*(FI(I+2)-FI(I  ))...
            +0.5*(DENVEL_w+abs(DENVEL_w))*fw2_p*(FI(I  )-FI(I-2))...
            +0.5*(DENVEL_w-abs(DENVEL_w))*fw2_m*(FI(I-1)-FI(I+1));
    end
end
```

Subfunction 'GETRES.m'

```
function [RES]=GETRES(fileID1,NM,IT,AE,AW,AP,FI,Q)
RES=0;
for I=2:NM
    RES=RES+abs(-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I)-AP(I)*FI(I));
end
fprintf(fileID1,'%i ITER., RES = %.16E\n',IT,RES);
fprintf('%i ITER., RES = %.16E\n',IT,RES);
```

Subfunction '`few_pm.m`'

```
function [fe2_p]=few_pm(fe_p,IC)
%#############################################################
% SUBROUTINE few_pm(fe_p,fe2_p,IC)
%#############################################################
%
%.....LUDS
switch IC
    case 5
    fe2_p=fe_p/2;
    %
    %.....QUICK
    case 6
        fe2_p=3/8-fe_p/4;
    %
    %.....TVD
    case 8
        if fe_p < 0 || fe_p > 1
            fe2_p=0;
        elseif fe_p >= 0 && fe_p <= 0.3
            fe2_p=fe_p;
        elseif fe_p > 0.3 && fe_p <= 5/6
            fe2_p=3/8-fe_p/4;
        elseif fe_p > 5/6 && fe_p <=1
            fe2_p=1.-fe_p;
        end
end
```

Subfunction 'GS.m'

```
function [FI]=GS(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP)
%#############################################################
% SUBROUTINE GS
%#############################################################
%
Q2=Q;
%
for IT=1:1000
    %
    %.....CALCULATE NEW SOLUTION
    for I=3:NM
        DENVEL_e=(FluxC(I)+FluxC(I+1))/2;
        DENVEL_w=(FluxC(I)+FluxC(I-1))/2;
        %
        %.....UPDATE NEW SOURCE TERM
        if IC >= 5
            [Q(I)]=GETQ(IC,I,Q2(I),FI,NM,DENVEL_e,DENVEL_w);
```

```
        end
        %
    end
    %
    for I=2:NM
        FI(I)=(-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I))/AP(I);
    end
    %
    %.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
    [RES]=GETRES(fileID1,NM,IT,AE,AW,AP,FI,Q);
    if RES < 1E-4; return; end
end
fprintf('ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED\n')
```

Subfunction 'GSOR.m'

```
function [FI]=GSOR(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP)
format long
%#############################################################
% SUBROUTINE GSOR(IC)
%#############################################################
OM=input('ENTER OVERRELAXATION PARAMETER: OM  \n');
%
Q2=Q;
%
for IT=1:1000
    %
    %.....CALCULATE NEW SOLUTION
    for I=3:NM
        DENVEL_e=(FluxC(I)+FluxC(I+1))/2;
        DENVEL_w=(FluxC(I)+FluxC(I-1))/2;
        %
        %.....UPDATE NEW SOURCE TERM
        if IC >= 5
            [Q(I)]=GETQ(IC,I,Q2(I),FI,NM,DENVEL_e,DENVEL_w);
        end
        %
    end
    %
    for I=2:NM
        FI(I)=FI(I)+OM*((-AE(I)*FI(I+1)-AW(I)*FI(I-1)+Q(I))/...
            AP(I)-FI(I));
    end
    %
    %.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
    [RES]=GETRES(fileID1,NM,IT,AE,AW,AP,FI,Q);
    if RES < 1E-15; return; end
end
fprintf('ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED\n')
```

Subfunction 'JACOBI.m'

```
function [FI]=JACOBI(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP)
%#############################################################
% SUBROUTINE JACOBI
%#############################################################
```

```
%
Q2=Q;
%
for IT=1:1000
    %
    %.....SAVE OLD SOLUTION
    FIO=FI;
    %
    for I=3:NM
        DENVEL_e=(FluxC(I)+FluxC(I+1))/2;
        DENVEL_w=(FluxC(I)+FluxC(I-1))/2;
        %
        %.....UPDATE NEW SOURCE TERM
        if IC >= 5
            [Q(I)]=GETQ(IC,I,Q2(I),FI,NM,DENVEL_e,DENVEL_w);
        end
        %
    end
    %
    %.....CALCULATE NEW SOLUTION
    for I=2:NM
        FI(I)=(-AE(I)*FIO(I+1)-AW(I)*FIO(I-1)+Q(I))/AP(I);
    end
    %
    %.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
    [RES]=GETRES(fileID1,NM,IT,AE,AW,AP,FI,Q);
    if RES < 1E-4; return; end
end
fprintf('ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED\n')
```

Subfunction 'TDMA.m'

```
function [FI]=TDMA(fileID1,IC,Q,FI,NM,FluxC,AE,AW,AP)
%#################################################################
% SUBROUTINE TDMA
%#################################################################
% INITIAL VALUES OF VARIABLES BPR(I) AND V(I) ASSUMED ZERO!
%
Q2=Q;
BPR=zeros(NM);V=zeros(NM);
%
for IT=1:1000
    %
    %.....CALCULATE NEW SOLUTION
    for I=3:NM
        DENVEL_e=(FluxC(I)+FluxC(I+1))/2;
        DENVEL_w=(FluxC(I)+FluxC(I-1))/2;
        %
        %.....UPDATE NEW SOURCE TERM
        if IC >= 5
            [Q(I)]=GETQ(IC,I,Q2(I),FI,NM,DENVEL_e,DENVEL_w);
        end
        %
    end
    %
    %.....CALCULATE 1./U_P (BPR) AND MODIFIED SOURCE TERM (V)
```

```
    for I=2:NM
        BPR(I)=1./(AP(I)-AW(I)*AE(I-1)*BPR(I-1));
        V(I)=Q(I)-AW(I)*V(I-1)*BPR(I-1);
    end
    %
    %.....CALCULATE VARIABLE VALUES - BACKWARD SUBSTITUTION
    for I = NM:-1:2
        FI(I)=(V(I)-AE(I)*FI(I+1))*BPR(I);
    end
    %
    if IC>=5
        for I=3:NM
            [Q(I)]=GETQ(IC,I,Q2(I),FI,NM,DENVEL_e,DENVEL_w);
        end
    end
    %
    %.....CALCULATE RESIDUAL AND CHECK CONVERGENCE
    [RES]=GETRES(fileID1,NM,IT,AE,AW,AP,FI,Q);
    if RES < 1E-15; return; end
end
fprintf('ATENTION : THE SOLUTION HAS NOT BEEN CONVERGED\n')
```

Appendix C

Sample Output from 1D Code

The input are as follows:

```
ENTER OUTPUT FILE NAME: out
ENTER: DEN(SITY), VEL(OCITY), DIF(FUSION COEFF.)
1,0.001,0.00002
ENTER BOUNDARY VALUES: FI0, FIN
0,1
CHOOSE CONVECTION SCHEME: 1 - CDS, 2 - UDS, 3 - UWDS,
4 - PDS, 5 - LUDS, 6 - QUICK, 7 - DEFERRED CORRECTION,
8 - TVD
8
ENTER: XMIN, XMAX, EX, N
0,1,1,11
CHOOSE SOLVER: 1 - JACOBI, 2 - GAUSS-SEIDEL, 3 - GSOR,
4 - TDMA
4
```

The sample output 'out' with the input parameters listed above:

```
1 ITER., RES = 1.3888880849499747E-04
2 ITER., RES = 7.7160071121169582E-05
3 ITER., RES = 3.4292387680957279E-05
4 ITER., RES = 1.2299677686683212E-05
5 ITER., RES = 6.8685737229986442E-06
6 ITER., RES = 6.6658006859191125E-06
7 ITER., RES = 4.7267518749292399E-06
8 ITER., RES = 3.2931748302273433E-06
9 ITER., RES = 1.8985038582913801E-06
10 ITER., RES = 6.5902121637746811E-07
11 ITER., RES = 7.8801971068598852E-07
12 ITER., RES = 6.9422517067578522E-07
13 ITER., RES = 4.0738930977519601E-07
14 ITER., RES = 3.9005838266500344E-07
15 ITER., RES = 4.0289655957415377E-07
16 ITER., RES = 3.0587577319544280E-07
17 ITER., RES = 2.0753335680265375E-07
18 ITER., RES = 9.7810731108509703E-08

PECLET NUMBER: PE = 50.000000
```

LOCAL PECLET NUMBER: PE = 5.000000
ERROR NORM = 0.008935
TVD USED FOR CONVECTION
TDMA SOLVER

X	FI_EXACT	FI	ERROR
0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
1.00000E−01	2.84323E−20	−8.09630E−10	8.09630E−10
2.00000E−01	4.24816E−18	−5.66741E−09	5.66741E−09
3.00000E−01	6.30511E−16	−2.61244E−08	2.61244E−08
4.00000E−01	9.35762E−14	2.75349E−07	−2.75349E−07
5.00000E−01	1.38879E−11	5.16871E−06	−5.16870E−06
6.00000E−01	2.06115E−09	7.49621E−05	−7.49600E−05
7.00000E−01	3.05902E−07	8.61715E−04	−8.61409E−04
8.00000E−01	4.53999E−05	9.06567E−03	−9.02027E−03
9.00000E−01	6.73795E−03	9.50629E−02	−8.83249E−02
1.00000E+00	1.00000E+00	1.00000E+00	0.00000E+00