Verification of Compressible and Incompressible Computational Fluid Dynamics Codes and Residual-based Mesh Adaptation

Aniruddha Choudhary

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Christopher J. Roy, Chair
Wayne L. Neu
Danesh Tafti
Heng Xiao

November 11, 2014
Blacksburg, Virginia

Keywords: Code verification, Order of accuracy, Method of manufactured solutions, Boundary conditions, Multiphase flows, Mesh adaptation, Truncation error

Verification of Compressible and Incompressible Computational Fluid Dynamics Codes and Residual-based Mesh Adaptation

Aniruddha Choudhary

ABSTRACT

Code verification is the process of ensuring, to the degree possible, that there are no algorithm deficiencies and coding mistakes (bugs) in a scientific computing simulation. In this work, techniques are presented for performing code verification of boundary conditions commonly used in compressible and incompressible Computational Fluid Dynamics (CFD) codes. Using a compressible CFD code, this study assesses the subsonic inflow (isentropic and fixed-mass), subsonic outflow, supersonic outflow, no-slip wall (adiabatic and isothermal), and inviscid slip-wall. The use of simplified curved surfaces is proposed for easier generation of manufactured solutions during the verification of certain boundary conditions involving many constraints. To perform rigorous code verification, general grids with mixed cell types at the verified boundary are used. A novel approach is introduced to determine manufactured solutions for boundary condition verification when the velocity-field is constrained to be divergence-free during the simulation in an incompressible CFD code. Order of accuracy testing using the Method of Manufactured Solutions (MMS) is employed here for code verification of the major components of an open-source, multiphase flow code - MFIX. The presence of two-phase governing equations and a modified SIMPLE-based algorithm requiring divergence-free flows makes the selection of manufactured solutions more involved than for single-phase, compressible flows. Code verification is performed here on 2D and 3D, uniform and stretched meshes for incompressible, steady and unsteady, single-phase and two-phase flows using the two-fluid model of MFIX.

In a CFD simulation, truncation error (TE) is the difference between the continuous governing equation and its discrete approximation. Since TE can be shown to be the local source term for the discretization error, TE is proposed as the criterion for determining which regions of the computational mesh should be refined/coarsened. For mesh modification, an error equidistribution strategy to perform r-refinement (i.e., mesh node relocation) is employed. This technique is applied to 1D and 2D inviscid flow problems where the exact (i.e., analytic) solution is available. For mesh adaptation based upon TE, about an order of magnitude improvement in discretization error levels is observed when compared with the uniform mesh.

GRANT INFORMATION

The Loci/CHEM verification study was supported by the National Aeronautics and Space Administration's Constellation University Institutes Program (CUIP) with Claudia Meyer of NASA Glenn Research Center serving as program manager and Kevin Tucker and Jeffrey West of NASA Marshall Space Flight Center serving as technical monitors.

The MFIX verification study was supported by the National Energy Technology Laboratory (NETL) through URS Corp. under contract T034:4000.3.671.238.003.413 with Mike Bergen as the URS subcontract technical representative.

The mesh adaptation study was partially funded by the Air Force Office of Scientific Research (AFOSR) Computational Mathematics Program managed by Dr. Fariba Fahroo under Grant FA9550-12-1-0173.

To my parents

Acknowledgements

I am thankful to my doctoral advisor, Dr. Christopher J. Roy. I learned from him various skills required to conduct proper engineering research. I am thankful to him for his methodical, patient, and consistent guidance throughout the course of this work. I also thank him along with the Aerospace and Ocean Engineering (AOE) department at Virginia Tech for providing uninterrupted funding and seamless access to the excellent computing resources of the AOE CFD Laboratory. I am thankful to my committee members, Dr. Wayne L. Neu, Dr. Danesh Tafti, and Dr. Heng Xiao, for serving on my doctoral committee.

I thank my recent and past colleagues at the AOE CFD laboratory, especially, Joseph Derlaga, Tyrone Phillips, Pavan S. Veluri, Brent Pickering, and Ian T. Voyles, for several helpful inputs during my research work. I am also thankful to other past members of the CFD lab. - Matthew Kurzen, Santhip Kanholy, Jacob Freeman, Kyle Knight and Ravi Duggirala.

I am thankful to Dr. Edward A. Luke of Mississippi State University, collaboration with whom was crucial in troubleshooting several issues during the verification of Loci/CHEM. I would like to thank the members of National Energy Technology Laboratory (NETL), Morgantown, West Virginia, with whom we had several useful interactions during the MFIX code verification project - Mehrdad Shahnam, Jordan Musser, Jean-François Dietiker, Rahul Garg, Tingwen Li, and Aytekin Gel.

I am thankful to Steve Edwards and Jonathan Spence (AOE System Administrators) for their prompt support in providing the necessary help with my computing requirements.

I am thankful to Rachel Hall Smith (AOE Graduate Coordinator) for her consistent help regarding various administrative requirements of the department and the graduate school.

The author also acknowledges Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to some of the results reported within this paper. URL: http://www.arc.vt.edu

Finally, I would like to thank the countless participants of the open-source software development community for providing the numerous computational tools that have contributed, in many ways, to both this work and this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

With increased use of computational tools for engineering simulations of physical systems, it becomes important to perform verification and validation studies for various aspects of a computational simulation. For a Computational Fluid Dynamics (CFD) simulation, verification and validation activities are useful in assessing the correctness of the code, quantifying the numerical accuracy of the simulation, and determining the applicability of the selected mathematical model.

According to the Guide for the Verification and Validation of Computational Fluid Dynamics Simulations [1], the terms "Verification" and "Validation" are defined as:

- *Verification*: The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- *Validation*: The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

In simpler words, verification deals with the mathematics of the simulation and involves assessing the correctness of the computer code and numerical algorithms as well as the accuracy of the numerical solution. Whereas, validation deals with the physics of the model and assesses whether the mathematical model selected satisfactorily predicts the physics of interest. Roache [28] put this distinction even more simply as: "Verification is a mathematics issue; not a physics issue".

There are two fundamental aspects to verification, code verification and solution verification which are defined as follows [23]:

- *Code verification*: The process of determining that the numerical algorithms are correctly implemented in the computer code and of identifying errors in software.
- *Solution verification*: The process of determining the correctness of the input data, the numerical accuracy of the solution obtained, and the correctness of the output data for a particular simulation.

In other words, code verification is the process of ensuring, to the degree possible, that there are no coding mistakes (bugs) and algorithm inconsistencies. Whereas, solution verification focuses on identifying and estimating various kinds of errors present in numerical simulations: e.g., round-off error, numerical iterative error, discretization error.

The first part of code verification deals with numerical algorithm verification (focus of the current work) whereas the second part about identifying errors in software falls under the topic of Software Quality Assurance. There are different criteria for assessing code verification: expert judgement, quantification of error, assessment of consistency/convergence, and testing of order of accuracy. Out of these, the order of accuracy test is considered to be the most rigorous one [23, 30].

Order of accuracy test requires the evaluation of discretization error on multiple grid levels. Discretization error is defined as the difference between the numerical solution to the discretized equations and the exact solution to the partial differential (or integral) equations. Evaluation of discretization error requires knowing the exact solution for the governing equations which is generally not known for problems of practical interest. In such scenarios, a mathematically rigorous technique called the Method of Manufactured Solutions (MMS) can be used where a solution is "manufactured" and used as an exact solution [26]. MMS is based upon the philosophy that code verification deals with the mathematics of the problem and hence arbitrary functions (with certain requirements as discussed later) can be selected as exact solutions.

The current work deals with code verification using method of manufactured solutions for boundary conditions (BCs) commonly implemented in compressible and incompressible CFD simulations, and for various features commonly implemented in a multiphase CFD code. The topic of verification is ultimately related to quantifying errors – specifically, the discretization error, in a computational simulation. Discretization error (DE) occurs in every CFD solution and is one of the main contributors to the overall uncertainty in a CFD prediction. Hence, the current work also proposes an approach to reduce discretization error in the solution using mesh adaptation (or targeted local refinement).

1.2 Previous work

Knupp and Salari [19] presented a detailed account of MMS-based code verification for incompressible and compressible Navier-Stokes codes. Their work addressed the verification

of Dirichlet BC (where exact value from the manufactured solution is specified), Neumann BC (where exact value of the gradient from the manufactured solution is specified), and mixed BC (a combination of Dirichlet and Neumann BCs). In this method the user specifies the exact values from the manufactured solutions at the boundaries instead of allowing the boundary condition implementation of the code to evaluate the boundary solution. This method is useful to accomplish code verification of governing equations inside the computational domain, and simple BCs (Dirichlet, Neumann, or mixed). For specialized boundary conditions used in practical flow simulations such as various kinds of wall, inflow, and outflow BCs, the authors in [19] suggested a more rigorous approach. When a boundary condition requires that a solution variable must meet a specific criterion on that boundary then the manufactured solution must be constructed adhering to that criterion. For example, for a free-slip wall BC implementation where the normal velocity component at the wall boundary is required to be zero the manufactured solution must be constructed such that its normal velocity component is zero at that boundary.

Bond et al. [4] presented detailed methodology for MMS-based boundary condition code verification. Their analysis included slip-wall BC, no-slip wall BCs (adiabatic and isothermal), and outflow BCs (subsonic, supersonic, and mixed) over non-orthogonal computational domains with hexahedral cells. In [4], a finite volume, unstructured, compressible CFD code, SIERRA/Premo [32] was used to verify the slip-wall BC and outflow BCs (subsonic and supersonic) using Euler equations, and the no-slip wall BCs (adiabatic and isothermal) using Navier-Stokes and RANS equations.

Veluri et al. [35] verified the no-slip wall BCs (adiabatic and isothermal), slip-wall BC, isentropic subsonic inflow BC, subsonic outflow BC, and supersonic outflow BC for an unstructured, compressible CFD code, Loci/CHEM [21], using the method of Bond et al. [4]. However, many of these tests employed manufactured solutions to work on meshes with planar boundary surfaces. In [35], the boundary conditions which were verified on curved surfaces (i.e., no-slip wall BC and slip-wall BC) used hexahedral cells at these surfaces.

In [10], Choudhary et al. presented the use of simple curved surfaces (a circular arc in 2D or a spherical cap in 3D) with hexahedral cells at the boundary for verification of inflow BCs (subsonic isentropic and subsonic fixed-mass) and outflow BCs (subsonic and supersonic).

Recently, Folkner et al. [15] suggested an approach where the different boundary conditions are verified along with the flow governing equations using a single manufactured solution. Their approach involves a selection matrix-based unifying framework for implementation of different boundary conditions which is yet to be seen in widespread use.

Code verification of multiphase flows is not as common in the literature as that for single-phase flows. This is because compared to single-phase flows, the presence of at least twice as many governing equations in multiphase flows, complex interphase interaction terms, and multiple constitutive relations make it difficult to obtain manufactured solutions or simple exact solutions for these equations. In [16], Grace and Taghipour discussed the importance of verification and validation activities for CFD models as applied to fluidized beds and

other dense multiphase flow systems. In addition, they correctly concluded from a survey of articles claiming "verification" or "validation" for numerical models simulating fluidized beds that these terms have often been used incorrectly.

There have been some MMS-based multiphase code verification studies in a multi-material context. "Multiphase" in this sense refers to the presence of materials in the domain with different physical properties thus resulting in solution discontinuities at the material interface. Brady et al. [6] presented a way to apply MMS to the finite volume multiphase code OSM which is a structured, Cartesian code for solving the heat equation. Manufactured solutions were generated using Heaviside and Dirac-delta functions to include the presence of moving interfaces in the domain for a typical immiscible two-phase system. They concluded that with such a discontinuity in material properties, the order of accuracy must reduce to first order for a second or higher order discretization scheme. This conclusion is also supported by Banks et al. [3] who showed that the formally second order accuracy of the discrete system reduces to first order in the presence of nonlinear discontinuities and to non-integer values below one for linear discontinuities. Roache et al. [27] used MMS to verify a finite-difference ground flow code with discontinuous conductivities in the domain by selecting the manufactured solutions such that they explicitly satisfy the geological boundary conditions. Crockett et al. [11] applied MMS to verify a multi-material heat equation solver that uses a Cartesian cut-cell/embedded boundary method to represent the interface between the materials. In [27] and [11], the interface locations are considered to be fixed and known a priori.

Shunn et al. [31] used MMS to verify an unstructured variable density flow-solver for a miscible two-fluid system with manufactured solutions reflective of the physical behavior common to combustion problems such as convective propagation of density fronts and mixing of species through diffusion. Physically-realistic manufactured solutions for incompressible, single-phase flows, were also proposed by Eça et al. [13, 14] for code verification of turbulent, wall-bounded flows.

Vedovoto et al. [34] performed a MMS-based code verification study of a pressure-based finite volume numerical scheme suited to variable density, single-phase flows generally encountered in combustion applications. In their work, the authors selected a manufactured solution mimicking the propagation of a corrugated flame front separating heavy from light gases.

In all these studies, the manufactured solutions proposed satisfy the necessary criteria such as divergence-free velocity field, wall boundary conditions, or consistency with employed turbulence functions. There are some advantages in using such physically-realistic manufactured in cases such as turbulence model verification [14] where the function and roles of different terms change based upon the nature of the solution. However, the selected manufactured solution should not just be realistic but also exhibit enough variations to ensure that all terms in the governing equations are exercised during the verification test [25].

Mesh adaptation is a widely studied field of research in the scientific computing community. For mesh adaptation approaches in CFD, there are several extensive reviews available (e.g.,

see [2], [22] and [18]). Performing mesh adaptation involves two distinct aspects: (1) a criterion for driving the adaptation, and (2) a mechanism to perform mesh modification. A variety of measures or adaptation monitors (also referred to as adaptation criteria, adaptation parameters, or weight functions) have been used in the past to determine which regions should be refined or coarsened. Once an adaptation criterion has been identified, the local mesh can be modified by moving nodes from one region to another (r-refinement), selectively refining/coarsening cells by adding/deleting nodes (h-refinement), or increasing/decreasing the formal order of accuracy of the method (p-refinement).

Hawken et al. [18] rightly observed that in many cases the adaptation criteria are chosen "heuristically" with "little or no justification" for them. Adaptation based upon solution features such as solution gradients, solution curvature, or even physical flow features (e.g., shock waves, expansion fans, contact discontinuities, vortices, and boundary layers) are known to fail in presence of multiple solution features (see [12] for one such example).

Adaptation based upon discretization error neglects the fact that DE can also be transported (i.e., convected, diffused, and propagated as acoustic waves) from other regions of the domain. Thus, it is entirely possible that the total DE may be large in a region where the local contributions to the DE are small. An example of the failure of mesh adaptation based on the total discretization error can be found in [17]. Adaptation based on recovery or reconstruction-based indicators (e.g., Zienkiewicz-Zhu patch recovery technique [38]) shows good results for linear elliptic problems in finite elements, but this improvement is not expected to carry over to hyperbolic or mixed-character problems.

Laflin [20] and McRae [22] demonstrated adaptation based on solution interpolation error (i.e., error in higher-order accurate interpolation from a patch of neighboring cells) to be successful for a wide range of 2D problems on structured meshes. Both recovery-based methods and adjoint-based methods for mesh adaptation use the solution residual weighted by the adjoint sensitivities as the adaption monitor, thus providing targeted mesh adaption for the chosen system response quantity (e.g., lift, drag, and moments).

Venditti and Darmofal [36, 37] successfully applied the method in finite-volume form to inviscid and viscous flow over airfoils at various Mach numbers. While adjoint-based methods show promising results, the implementation of these methods can be mathematically complex, code intrusive, and computationally expensive since it requires the solution for both dual and primal problems.

The error transport equations [29, 24] show that the discretization error is locally generated by the truncation error as well as transported from other regions of the domain. The truncation error (or TE) is defined as the difference between the discretized equations and the original continuous partial differential (or integral) equations. Since the truncation error serves as the local source of the discretization error in the domain, reducing the truncation error should result in a significant reduction in the discretization error. TE as an adaptation monitor is based upon systematic mathematical derivation and includes the effect of solution behavior, mesh resolution, and mesh quality in the adaptation process [8]. Borsboom

[5] applied a similar approach of residual-based adaptation to a scalar, 1D problem with an exact solution mimicking Sod's [33] 1D shock tube solution. For various grid sizes, Borsboom obtained 10 to 1000 times reduction in DE in comparison with the uniform mesh since the presence of a solution discontinuity in the domain usually results in large improvement with adaptation. Preliminary results for simple 1D and 2D Burgers' equations and Euler problems were shown to give about an order of magnitude improvement in DE (over uniform grids) when compared to feature-based adaptation monitors [29, 7, 8, 9].

1.3 Highlights/ Contributions

During the verification of boundary conditions in CFD codes, four objectives are accomplished. First, the method suggested by Bond et al. [4] is applied to different inflow, outflow, and wall BCs. For each of these BCs, the boundary constraints as well as the implementation of the constraints within the code is slightly different from their previous implementations in the literature. Second, the use of simplified boundary surfaces (e.g., spherical caps) is demonstrated which makes the derivation of manufactured solutions under several constraints much simpler than the use of general surfaces. Third, the use of grids with hybrid (i.e., consisting of mixed cell types) boundaries is shown to uncover implementation errors that could be masked by using cells of same types. Fourth, during the verification of the incompressible code, a new curl-based method is proposed to derive the manufactured solutions under the constraint of divergence-free velocity field.

The code verification study of the incompressible, multiphase flows assesses the numerical scheme implemented in the discretization of the two-fluid model governing equations within the open-source code, MFIX. The current work is restricted to 3D, incompressible, laminar, two-phase flows. Multiple objectives are accomplished within the context of this code verification study. First, compared to previous works in the literature, mathematically general manufactured solutions consisting of sinusoidal functions are used ensuring a rigorous verification of all the discretized terms of the governing equations. Second, the algorithm implemented in MFIX requires that the velocity field be divergence-free for the selected manufactured solutions. This is accomplished by introducing a novel, curl-based method to derive manufactured solution for code verification of incompressible flows. Third, the commonly used boundary conditions, i.e., no-slip wall, free-slip wall, and pressure outflow are tested using manufactured solutions which satisfy the divergence-free velocity field constraints as well as the boundary constraints. Note that this discussion is same as the discussion for incompressible flow boundary condition verification presented earlier. Fourth, the temporal order of accuracy for first-order and second-order time-stepping schemes is tested for unsteady simulations.

In the final part of the dissertation, the focus is on developing a TE/residual-based mesh adaptation procedure and applying it to benchmark 1D and 2D Euler problems in CFD. In this process, a Fortran library of subroutines called Structured Adaptation Module (or

SAM) is developed to perform r-refinement (i.e., adaptation via mesh relocation) for equidistribution of a selected adaptation monitor. To test the proposed method, 1D and 2D Euler problems are selected for which the exact (i.e., analytical) solutions are available. The Euler problems considered include 1D Burgers equation, a quasi-1D flow through a converging diverging nozzle, and supersonic flows turning past a sharp corner. The 1D Burgers equation case is used to compared the TE-based adaptation monitor against the feature-based adaptation monitors where approximately 10 to 100 times reduction is seen the solution error (compared to that on a uniform mesh). Whereas, for the Euler problems, approximately an order of magnitude reduction is seen in the solution error.

1.4 Outline

This dissertation is in the "manuscript" format (also known as the "multi-paper" format) where the main chapters of the dissertation are in the format as accepted by peer-reviewed journals.

This first chapter briefly introduces the topics of code verification, method of manufactured solutions, and suggests through relevant literature review the lack of comprehensive and rigorous code verification methods for boundary conditions and multiphase flows in CFD. This chapter also provides a brief introduction to mesh adaptation and why a residual-based driver for adaptation is a good candidate as an indicator of where the mesh must be refined/-coarsened. The code verification of boundary conditions is presented in Chapter 2, where the techniques presented are applied to both compressible and incompressible flow codes. This chapter presents the manufactured solutions required for rigorously verifying different boundary conditions commonly used in CFD codes demonstrating the use of boundaries with mixed cell types and spherical surfaces. Chapter 3 introduces the governing equations for the two-fluid model employed in the incompressible multiphase code, MFIX, and presents increasingly rigorous method of manufactured solution test cases to perform code verification. In Chapter 4, the details of the equidistribution approach for performing truncation error (or residual) based mesh adaptation are presented along with the results for 1D and 2D Euler problems. Chapter 5 provides conclusions highlighting the key findings of this work and some ideas for future work.

1.5 Attribution

Though the three main chapters, i.e., Chapters 2, 3, and 4, have multiple authors, most of the work is attributed to the first author. The first author (Aniruddha Choudhary) is responsible for the development of most of the methodology (unless indicated by references), developing necessary computer codes, running computational simulations, and generating all

of the results reported in this dissertation. The contributions of all the authors in different chapters are described as follows:

- Chapter 2: Code verification of boundary conditions. The first author (Aniruddha Choudhary) is responsible for generation of manufactured solutions, generation of grids, designing and running computational simulations using Loci/CHEM and MFIX/TFM codes, generation, and interpretation of results. The second author (Christopher J. Roy) is responsible for providing guidance regarding the framework adopted to perform code verification using manufactured solutions. The third author (Edward A. Luke) is responsible for providing assistance with installing and running the Loci/CHEM computer code. He also modified the Loci/CHEM computer code when necessary to correct any inconsistencies uncovered during the verification process. The fourth author (Pavan S. Veluri) is responsible for providing the Fortran code that subdivides a structured grid into generalized grids with mixed cell types as reported in Sec. 2.4.

- Chapter 3: Code verification of multiphase flows. The first author (Aniruddha Choudhary) is responsible for conducting literature review regarding multiphase code verification, designing the suite of test cases, generating the appropriate divergence-free manufactured solutions, running the computational simulations, debugging/modifying the code where necessary, and generating the reported results and conclusions. The second author (Christopher J. Roy) provided appropriate guidance regarding code verification during various steps. The third author (Jean-François Dietiker), the fourth author (Mehrdad Shahnam), and the fifth author (Rahul Garg) are all responsible for providing assistance in setting up the initial tests cases in MFIX, in debugging MFIX when bugs or algorithm inconsistencies were encountered, and in understanding relevant aspects of the two-fluid model implemented within MFIX.

- Chapter 4: Residual-based mesh adaptation. The first author (Aniruddha Choudhary) is responsible for developing the Fortran library of equidistribution-based adaptation subroutines, called Structured Adaptation Module (or SAM), for performing mesh adaptation for all the presented application problems, and for generating all the reported results. The second author (Christopher J. Roy) presented the residual-based mesh adaptation in [29] along with preliminary results for the 1D Burgers equation problem and provided crucial inputs on the current work.

Bibliography

[1] AIAA. AIAA Guide for the verification and validation of computational fluid dynamics simulations. Technical report, Jan 1998. URL `http://dx.doi.org/10.2514/4.472855.001`.

[2] Timothy J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(34):243 – 273, 1997. ISSN 0168-874X. doi: http://dx.doi.org/10.1016/S0168-874X(96)00032-7. URL `http://www.sciencedirect.com/science/article/pii/S0168874X96000327`. Adaptive Meshing, Part 2.

[3] J.W. Banks, T. Aslam, and W.J. Rider. On sub-linear convergence for linearly degenerate waves in capturing schemes. *Journal of Computational Physics*, 227(14): 69857002, Jul 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2008.04.002. URL `http://dx.doi.org/10.1016/j.jcp.2008.04.002`.

[4] Ryan B. Bond, Curtis C. Ober, Patrick M. Knupp, and Steven W. Bova. Manufactured solution for computational fluid dynamics boundary condition verification. *AIAA Journal*, 45(9):2224–2236, 2007. ISSN 0001-1452. doi: 10.2514/1.28099. URL `http://arc.aiaa.org/doi/abs/10.2514/1.28099`.

[5] Mart Borsboom. Development of an error-minimizing adaptive grid method. *Applied Numerical Mathematics*, 26(12):13 – 21, 1998. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/S0168-9274(97)00077-9. URL `http://www.sciencedirect.com/science/article/pii/S0168927497000779`.

[6] P.T. Brady, M. Herrmann, and J.M. Lopez. Code verification for finite volume multiphase scalar equations using the method of manufactured solutions. *Journal of Computational Physics*, 231(7):29242944, Apr 2012. ISSN 0021-9991. doi: 10.1016/j.jcp.2011.12.040. URL `http://dx.doi.org/10.1016/j.jcp.2011.12.040`.

[7] Aniruddha Choudhary and Christopher Roy. Efficient residual-based mesh adaptation for 1D and 2D CFD applications. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2011. doi: 10.2514/6.2011-214. URL `http://dx.doi.org/10.2514/6.2011-214`.

[8] Aniruddha Choudhary and Christopher Roy. A truncation error-based approach to understanding and improving mesh quality in CFD. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2012. doi: 10.2514/6.2012-607. URL `http://dx.doi.org/10.2514/6.2012-607`.

[9] Aniruddha Choudhary and Christopher J. Roy. Structured mesh r-refinement using truncation error equidistribution for 1D and 2D Euler problems. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2013. doi: 10.2514/6.2013-2444. URL `http://dx.doi.org/10.2514/6.2013-2444`.

[10] Aniruddha Choudhary, Christopher J Roy, Edward A. Luke, and Subrahmanya P. Veluri. Issues in verifying boundary conditions for 3D unstructured CFD codes. American Institute of Aeronautics and Astronautics, 2011. doi: doi:10.2514/

6.2011-386810.2514/6.2011-3868. URL http://dx.doi.org/10.2514/6.2011-3868. doi:10.2514/6.2011-3868.

[11] R.K. Crockett, P. Colella, and D.T. Graves. A Cartesian grid embedded boundary method for solving the Poisson and heat equations with discontinuous coefficients in three dimensions. *Journal of Computational Physics*, 230(7):24512469, Apr 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.12.017. URL http://dx.doi.org/10.1016/j.jcp.2010.12.017.

[12] Richard P. Dwight. Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation. *Journal of Computational Physics*, 227(5):2845 – 2863, 2008. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2007.11.020. URL http://www.sciencedirect.com/science/article/pii/S002199910700513X.

[13] L. Ea, M. Hoekstra, A. Hay, and D. Pelletier. A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow. *International Journal of Computational Fluid Dynamics*, 21(3-4):175188, Mar 2007. ISSN 1029-0257. doi: 10.1080/10618560701553436. URL http://dx.doi.org/10.1080/10618560701553436.

[14] L. Ea, M. Hoekstra, and G. Vaz. Manufactured solutions for steady-flow Reynolds-averaged NavierStokes solvers. *International Journal of Computational Fluid Dynamics*, 26(5):313332, Jun 2012. ISSN 1029-0257. doi: 10.1080/10618562.2012.717617. URL http://dx.doi.org/10.1080/10618562.2012.717617.

[15] D. Folkner, A. Katz, and V. Sankaran. Design and verification methodology of boundary conditions for finite volume schemes. *Computers & Fluids*, 96:264275, Jun 2014. ISSN 0045-7930. doi: 10.1016/j.compfluid.2014.03.028. URL http://dx.doi.org/10.1016/j.compfluid.2014.03.028.

[16] John R. Grace and Fariborz Taghipour. Verification and validation of CFD models and dynamic similarity for fluidized beds. *Powder Technology*, 139(2):99110, Jan 2004. ISSN 0032-5910. doi: 10.1016/j.powtec.2003.10.006. URL http://dx.doi.org/10.1016/j.powtec.2003.10.006.

[17] Xubin Gu and Tom Shih. Differentiating between source and location of error for solution-adaptive mesh refinement. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2001. doi: 10.2514/6.2001-2660. URL http://dx.doi.org/10.2514/6.2001-2660.

[18] D.F Hawken, J.J Gottlieb, and J.S Hansen. Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations. *Journal of Computational Physics*, 95(2):254 – 302, 1991. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(91)90277-R. URL http://www.sciencedirect.com/science/article/pii/002199919190277R.

[19] Patrick Knupp and Kambiz Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman & Hall/CRC, 2003.

[20] K. R. Laflin. *Solver-Independent r-Refinement Adaptation for Dynamic Numerical Simulations*. PhD thesis, North Carolina State University, 1997.

[21] E. A. Luke, X.-L. Tong, J. Wu, L. Tang, and P. Cinnella. CHEM: A chemically reacting flow solver for generalized grids, 2003. URL `http://www.tetraresearch.com/locichem/about-locichem/`.

[22] D.Scott McRae. r-refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 189(4):1161 – 1182, 2000. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/S0045-7825(99)00372-2. URL `http://www.sciencedirect.com/science/article/pii/S0045782599003722`. Adaptive Methods for Compressible {CFD}.

[23] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press Cambridge, 2010.

[24] W.L. Oberkampf and C.J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, 2010.

[25] Dominique Pelletier and Patrick J. Roache. *Verification and Validation of Computational Heat Transfer*, page 417442. John Wiley & Sons, Inc., Dec 2000. ISBN http://id.crossref.org/isbn/9780471348788. doi: 10.1002/9780470172599.ch13. URL `http://dx.doi.org/10.1002/9780470172599.ch13`.

[26] P. J. Roache and S. Steinberg. Symbolic manipulation and computational fluid dynamics. *AIAA Journal*, 22(10):1390–1394, October 1984. ISSN 0001-1452. doi: 10.2514/3.8794. URL `http://dx.doi.org/10.2514/3.8794`.

[27] P. J. Roache, P. Knupp, S. Steinberg, and R. L. Blaine. Experience with benchmark test cases for groundwater flow, in: Benchmark test cases for computational fluid dynamics. *ASME FED H00598-1990*, 93:49–56, 1990.

[28] Patrick J. Roache. *Fundamentals of Verification and Validation*. Hermosa Publishers, 2009. ISBN 0913478121, 9780913478127.

[29] Christopher Roy. Strategies for driving mesh adaptation in CFD (invited). In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2009. doi: 10.2514/6.2009-1302. URL `http://dx.doi.org/10.2514/6.2009-1302`.

[30] Christopher J. Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205(1):131 – 156, 2005. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2004.10.036. URL `http://www.sciencedirect.com/science/article/pii/S0021999104004619`.

[31] Lee Shunn, Frank Ham, and Parviz Moin. Verification of variable-density flow solvers using manufactured solutions. *Journal of Computational Physics*, 231(9):38013827, May 2012. ISSN 0021-9991. doi: 10.1016/j.jcp.2012.01.027. URL `http://dx.doi.org/10.1016/j.jcp.2012.01.027`.

[32] Thomas Smith, Curtis Ober, and Alfred Lorber. SIERRA/Premo-a new general purpose compressible flow simulation code. In *Fluid Dynamics and Co-located Conferences*. American Institute of Aeronautics and Astronautics, June 2002. doi: 10.2514/6.2002-3292. URL `http://dx.doi.org/10.2514/6.2002-3292`.

[33] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(78)90023-2. URL `http://www.sciencedirect.com/science/article/pii/0021999178900232`.

[34] Joo Marcelo Vedovoto, Aristeu da Silveira Neto, Arnaud Mura, and Luis Fernando Figueira da Silva. Application of the method of manufactured solutions to the verification of a pressure-based finite-volume numerical scheme. *Computers & Fluids*, 51(1):8599, Dec 2011. ISSN 0045-7930. doi: 10.1016/j.compfluid.2011.07.014. URL `http://dx.doi.org/10.1016/j.compfluid.2011.07.014`.

[35] Subrahmanya P. Veluri, Christopher J. Roy, and Edward A. Luke. Comprehensive code verification techniques for finite volume CFD codes. *Computers & Fluids*, 70(0):59–72, 2012. ISSN 0045-7930. doi: http://dx.doi.org/10.1016/j.compfluid.2012.04.028. URL `http://www.sciencedirect.com/science/article/pii/S0045793012001697`.

[36] David A Venditti and David L Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204 – 227, 2000. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2000.6600. URL `http://www.sciencedirect.com/science/article/pii/S0021999100966002`.

[37] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176 (1):40 – 69, 2002. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2001.6967. URL `http://www.sciencedirect.com/science/article/pii/S0021999101969670`.

[38] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992. ISSN 1097-0207. doi: 10.1002/nme.1620330703. URL `http://dx.doi.org/10.1002/nme.1620330703`.

Chapter 2

Code Verification of Boundary Conditions for Compressible and Incompressible CFD Codes

(In review with *Computers and Fluids*, Elsevier)

Aniruddha Choudhary[a], Christopher J. Roy[a], Edward A. Luke[b], Subrahmanya P. Veluri[c]

[a] Aerospace and Ocean Engineering Department, Virginia Tech, Blacksburg, VA 24061, USA

[b] Computer Science and Engineering Department, Mississippi State University, Mississippi State, MS 39762, USA

[c] ANSYS Inc., Lebanon, NH 03766, USA

Abstract

To establish confidence in the code, a rigorous assessment of boundary condition implementation is necessary. In this work, techniques are presented for performing code verification of boundary conditions commonly used in compressible and incompressible Computational Fluid Dynamics (CFD) codes. Using a compressible CFD code, this study assesses the subsonic inflow (isentropic and fixed-mass), subsonic outflow, supersonic outflow, no-slip wall (adiabatic and isothermal), and inviscid slip-wall. A novel approach is introduced to determine manufactured solutions for boundary condition verification when the velocity-field is constrained to be divergence-free during the simulation in an incompressible CFD code. The use of simplified curved surfaces is proposed for easier generation of manufactured solutions during the verification of certain boundary conditions involving many constraints. To perform rigorous code verification, general grids with mixed cell types at the verified boundary are used. It is found that the use of planar boundaries or only hexahedral cells at the verified

boundary can mask sources of errors in the boundary condition implementation.

Keywords Code verification; Method of manufactured solutions; Order of accuracy; Boundary conditions

2.1 Introduction

With increased use of computational tools for fluid dynamics simulations, it becomes important to perform verification of various aspects of a computational simulation such as conservation equations, closure models, and boundary conditions. Verification deals with the mathematics of the simulation and involves assessing the correctness of the computer code and numerical algorithms, as well as the accuracy of the numerical solution. There are two fundamental aspects to verification: code verification and solution verification. Code verification is the process of ensuring, to the degree possible, that there are no coding mistakes (bugs) or algorithm inconsistencies. Solution verification focuses on identifying and estimating various kinds of errors present in numerical simulations due to round-off, statistical sampling, iteration, and discretization error.

Different criteria for performing code verification, in order of increasing rigor, are: simple tests, code-to-code comparisons, discretization error quantification, convergence tests, and order of accuracy tests [23]. The order of accuracy test determines whether or not the discretization error is reduced at an expected rate given by the formal order of the implemented numerical scheme. Evaluation of discretization error requires knowing the exact solution for the governing equations which is generally not known for problems of practical interest. A technique called the method of manufactured solutions (MMS) can be used where a solution is "manufactured" and used as an exact solution [20]. This manufactured solution exactly solves the modified governing equations consisting of the MMS source terms, where the MMS source terms are generated by substituting the manufactured solution in the original governing equations. The method of manufactured solutions was first applied for code verification by Roache and Steinberg [20] where it was used to verify a code for generation of 3D transformations for elliptic partial differential equations (PDEs). The books by Knupp and Salari [11], Roache [21], and Oberkampf and Roy [15] provide a comprehensive discussion of code verification, order of accuracy testing, and MMS.

The current work is focused on the code verification of boundary conditions (BCs) commonly used in compressible and incompressible computational fluid dynamics (CFD) simulations. Knupp and Salari [11] presented a detailed account of MMS-based code verification for incompressible and compressible Navier-Stokes codes. Their work addressed the verification of Dirichlet BC (where exact value from the manufactured solution is specified), Neumann BC (where exact value of the gradient from the manufactured solution is specified), and mixed BC (a combination of Dirichlet and Neumann BCs). In this method the user specifies the exact values from the manufactured solutions at the boundaries instead of allowing the

boundary condition implementation of the code to evaluate the boundary solution. This method is useful to accomplish code verification of governing equations inside the computational domain, and simple BCs (Dirichlet, Neumann, or mixed). For specialized boundary conditions used in practical flow simulations such as various kinds of wall, inflow, and outflow BCs, the authors in [11] suggested a more rigorous approach. When a boundary condition requires that a solution variable must meet a specific criterion on that boundary then the manufactured solution must be constructed adhering to that criterion. For example, for a free-slip wall BC implementation where the normal velocity component at the wall boundary is required to be zero the manufactured solution must be constructed such that its normal velocity component is zero at that boundary.

Bond et al. [3] presented detailed methodology for MMS-based boundary condition code verification. Their analysis included slip-wall BC, no-slip wall BCs (adiabatic and isothermal), and outflow BCs (subsonic, supersonic, and mixed) over non-orthogonal computational domains with hexahedral cells. In [3], a finite volume, unstructured, compressible CFD code, SIERRA/Premo [25] was used to verify the slip-wall BC and outflow BCs (subsonic and supersonic) using Euler equations, and the no-slip wall BCs (adiabatic and isothermal) using Navier-Stokes and RANS equations. Veluri et al. [31] verified the no-slip wall BCs (adiabatic and isothermal), slip-wall BC, isentropic subsonic inflow BC, subsonic outflow BC, and supersonic outflow BC for an unstructured, compressible CFD code, Loci/CHEM [12], using the method of Bond et al. [3]. However, many of these tests employed manufactured solutions to work on meshes with planar boundary surfaces. In [31], the boundary conditions which were verified on curved surfaces (i.e., no-slip wall BC and slip-wall BC) used hexahedral cells at these surfaces. In [5], we presented the use of simple curved surfaces (a circular arc in 2D or a spherical cap in 3D) with hexahedral cells at the boundary for verification of inflow BCs (subsonic isentropic and subsonic fixed-mass) and outflow BCs (subsonic and supersonic). Recently, Folkner et al. [7] suggested an approach where the different boundary conditions are verified along with the flow governing equations using a single manufactured solution. Their approach involves a selection matrix-based unifying framework for implementation of different boundary conditions which is yet to be seen in widespread use.

In the current work we accomplish four objectives. First, we apply the method of [3] to different inflow, outflow, and wall BCs identifying the mathematical constraints for derivation of manufactured solutions for each boundary condition separately. Second, we demonstrate the use of boundary surfaces in the form of spherical caps to verify certain boundary conditions which makes the derivation of manufactured solutions under constraints much simpler than the use of general boundary surfaces. Third, we present the use of grids with boundary surfaces having cells of mixed types to verify boundary condition in a rigorous manner that could uncover implementation errors masked by using boundaries having cells of the same types. Fourth, we introduce two new methods to derive manufactured solutions under the constraint of divergence-free velocity field.

The paper is organized as follows: In Sec. 2.2, we present the governing equations and details of the flow solvers used for analysis. In Sec. 2.3, we give an overview of the methodol-

ogy used for MMS-based code verification of boundary conditions. Sec. 2.4 presents different grid types and boundary surfaces used for rigorous code verification of boundary conditions. Sec. 2.5 presents results of this work including: boundary constraints, derivation of manufactured solutions, results of order of accuracy tests, some issues uncovered, and a discussion of our findings for each of the cases. Finally, we summarize our work in Sec. 2.6. Before proceeding further, we address what boundary conditions and what is it that we are attempting to verify with code verification procedures.

2.1.1 A note on boundary conditions in CFD

Physically, boundary conditions at the domain boundary determine the interaction between the physical phenomenon in the domain interior and the domain exterior. Mathematically, boundary conditions are mathematical relations that, together with the initial conditions, can ensure (at least locally) that the initial boundary value problem (IBVP) for Navier-Stokes and/or Euler equations is well-posed (i.e., a stable, unique solution exists). Simple extrapolation of variables at the domain boundary is not sufficient as it does not account for the physical phenomenon and the mathematical nature of the governing equations at the boundary [10]. Thompson [29, 30] defined a time-dependent boundary condition for a time-dependent IBVP as a precise mathematical relation that can be used to determine $\partial\vec{U}/\partial t$, where $\vec{U}$ is the conserved variable vector, along the domain boundary and is not obtainable solely from the governing equation of $\vec{U}$ in the domain interior. For 1D Euler equations, a compatible set of boundary conditions can be derived that would ensure a unique determination of $\partial\vec{U}/\partial t$ at the domain boundary. Navier-Stokes equations have mixed hyperbolic-parabolic character and the assessment of well-posedness is not straightforward. In general, the determination of compatible boundary conditions for IBVPs (here, Euler and Navier-Stokes equations) does not guarantee an existence of a stable, unique solution. See [18] for a summary of boundary conditions that may ensure a well-posed problem.

Efficient and robust techniques based upon mathematical theory and application based evidence are widely used to determine boundary conditions in CFD. For example, at the wall, one generally imposes no-slip conditions on velocity for Navier-Stokes equations along with either isothermal or adiabatic conditions for the state variables. For a complete description of different boundary conditions, the interested readers are referred to the pioneering works by Yee [33], Yee, Beam and Warming [34], and the book by Hirsch [10]. In the present work, the FUN3D technical note [4] and the Loci/CHEM user's guide [13] are referred to for an overview of the implementation techniques of different boundary conditions.

The various aspects to be considered while selecting, implementing, and assessing a boundary condition can be summarized as follows [19]:

1. Physical: A boundary condition must satisfy the physical phenomenon. For example, for 1D Euler equations, at a subsonic inflow, the physics of the boundary imposes, (a)

$\vec{v} \cdot \vec{n} = 0$, and (b) $|\vec{v} \cdot \vec{n}| < c$, where $\vec{v}$ is the velocity vector, $\vec{n}$ is the outward normal at the boundary, and c is the local speed of sound.

2. Mathematical: A boundary condition should be mathematically well-posed. For example, for 1D Euler equations, at a subsonic inflow, an analysis using compatibility conditions (see [8] for derivation) shows that (a) two variables must be defined, and (b) information on density must be directly or indirectly obtainable for the problem to be well-posed. This suggests a use of either a fixed-mass inflow (mass flow rate and density specified) or an isentropic inflow (total pressure and total temperature specified) formulation amongst others. For 2D, and 3D flows, information on flow directionality must either be provided or assumed at the subsonic inflow.

3. Numerical: Includes numerical approximation of the mathematical condition (e.g.,one-sided extrapolation), and additional numerical requirements. Stability and accuracy of the boundary scheme must also be assessed.

4. Application: With all above aspects, the BC must ultimately be satisfactorily efficient and general to handle a wide range of application-based flow problems.

For cell-centered finite-volume methods, the boundary flux is usually developed using either a standard weak formulation or a scheme-consistent weak formulation. In a standard (or a scheme-inconsistent) formulation the boundary flux is directly computed at the boundary making the boundary scheme different than the finite-volume scheme applied at the interior faces. In a scheme-consistent formulation, the boundary condition is applied at the ghost cells (i.e., computational cells just outside the edge of the physical domain) which is used along with the interior scheme to compute the boundary flux [17]. Whether a standard or a scheme-consistent formulation has been implemented, in as far as boundary condition code verification is concerned, the goal is to verify its implementation in the code as intended by the developer. This requires us to identify the relevant physical, mathematical, and numerical constraints of a boundary condition while constructing the manufactured solutions. For example, a one-sided extrapolation of a solution variable to the boundary is usually formulated based upon a finite-difference approximation of a derivative condition such as zero gradient. In this case, the zero gradient condition becomes one of the constraints under which the manufactured solution must be derived. Together, the physical, mathematical, and numerical constraints of a boundary condition are referred to as boundary constraints and are the primary objects during our analysis.

2.2 Codes and governing equations

2.2.1 Compressible solver: Loci/CHEM

For the discussion of subsonic inflow, outflow, and wall boundary conditions using compressible flows, we use Loci/CHEM [12] which is a finite-volume, compressible CFD code for generalized 3D unstructured grids and employs a density-based solver capable of solving turbulent, multiphase, multispecies, chemically reacting 3D flows. Loci/CHEM is developed upon the Loci framework, a rule-based programming framework developed for large-scale, massively parallel, multidisciplinary simulations [14].

The governing equations for CHEM include the 3D unsteady Favre-averaged Navier-Stokes equations (referred to herein as baseline governing equations) and several turbulence models. The steady-state form of equations is obtained from the baseline governing equations by setting the time derivative terms to zero. The Euler equations are obtained by removing the viscous terms, while laminar Navier-Stokes equations are obtained by specifying constant viscosity and neglecting turbulence models in the baseline governing equations. The 3D, laminar Navier-Stokes equations along with the constitutive relations as verified in this work are given in Eqs. 2.1-2.6 using Cartesian tensor notations, where i, j, and k correspond to the components in the three coordinate directions, x, y and z (other symbols have usual thermodynamic connotations).

Continuity:

$$\frac{\partial \rho u_j}{\partial x_j} = 0 \tag{2.1}$$

Momentum:

$$\frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \tag{2.2}$$

Energy:

$$\frac{\partial \rho e_T u_j}{\partial x_j} = -\frac{\partial q_j}{\partial x_j} - \frac{\partial p u_j}{\partial x_j} + \frac{\partial \tau_{jk} u_k}{\partial x_j} \tag{2.3}$$

Stress tensor:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \left(\frac{\partial u_i}{\partial x_i} \right) \delta_{ij} \tag{2.4}$$

Laminar heat flux:

$$q_j = -\frac{\mu}{Pr} c_p \frac{\partial T}{\partial x_j} \tag{2.5}$$

Thermodynamic relations:

$$h_T = e + \frac{u_i u_i}{2} + \frac{p}{\rho}, \quad e = c_v T, \quad p = \rho R T \tag{2.6}$$

Formally second order upwind flux-differencing schemes of the Roe family are employed with flux reconstruction at the cell center using weighted least-square gradient technique. Loci/CHEM is capable of handling generalized grids, i.e., grids composed of arbitrary polyhedra, including tetrahedra, prisms, pyramids, and hexahedra, making adaptive mesh refinement feasible. Loci/CHEM has been comprehensively verified to be second order for baseline governing equations, unsteady time-stepping accuracy, and some boundary conditions [31, 22, 9, 32].

2.2.2 Incompressible solver: MFIX/TFM

For the verification of no-slip wall BC, slip wall BC, and pressure outflow BC using incompressible flows, we use MFIX (Multiphase Flow with Interphase eXchanges) [1] which is an open-source, general-purpose, hydrodynamic code that can simulate heat transfer and chemical reactions for fluids containing multiple solid phases. It is generally used for flows common in energy conversion and chemical reactor processes such as bubbling, circulating, and spouted fluidized beds. Both the continuum and discrete approaches to model the multiphase interactions are available in MFIX. In a continuum approach, the different phases are mathematically described as interpenetrating continua. The continuum approach is also referred to as the Eulerian-Eulerian method or the two-fluid method (TFM). Methods where the carrier (or surrounding) phase is treated as a continuum and the dispersed phase is treated as discrete entities (i.e., particles or parcels of particles) are called Lagrangian-Eulerian methods or Continuum Discrete Methods.

In the current discussion of boundary condition verification techniques, we are concerned with the two-fluid model with the assumption of steady-state, single-phase flow. In this sense, the MFIX governing equations reduce simply to the incompressible Navier-Stokes equations where the continuity equation is given by Eq. 2.1 and the momentum conservation equation is given by Eq. 2.2 but with the density term, ρ, treated as a constant. This is a description of only the relevant governing equations for single-phase, incompressible flows as verified in the current work. For a detailed description of multiphase governing equations and models in MFIX, see [2].

MFIX uses formally second-order centered and upwinding methods for spatial discretization featuring a staggered-grid approach where momentum variables (e.g., fluid and solid velocities) reside at the face centers while scalar variables (e.g., pressure, temperature, volume-fraction) reside at the cell centers. MFIX uses a modified SIMPLE-based algorithm and employs a pressure projection method imposing a divergence-free velocity for incompressible flow [27]. The two-fluid model of MFIX has been verified to be second order accurate for the

single-phase and two-phase baseline governing equations on 3D, stretched Cartesian meshes using the centered discretization scheme [6].

2.3 Methodology

The order of accuracy test for code verification determines whether the observed order of accuracy for the numerical scheme matches the formal order of accuracy as the mesh is refined into the asymptotic range. The formal order is obtained from a truncation error analysis of the discrete equations [23]. The observed order is determined directly from the simulations by observing the rate at which the norms (e.g., L_1, L_2, L_∞) of the solution discretization error decrease as the grid is systematically refined. The asymptotic range is that range of discretization sizes (e.g., Δx, Δt) where the higher order terms of the truncation and discretization error are negligible compared to the lowest order term (i.e., the term which defines the formal order of the scheme). The observed order of accuracy will generally fail to match the formal order when there is a coding mistake (bug), any algorithm inconsistency, a discontinuity in the solution, large iterative or round-off errors, or if the solution is not in the asymptotic range.

To calculate the observed order of accuracy, the numerical solution must be calculated at multiple grid levels. The number of grid levels required depends on whether the solution is in the asymptotic range and usually varies from one problem to another. Systematic mesh refinement [15] is defined as uniform and consistent refinement over the spatial domain. Uniform refinement ensures that the mesh has been refined along all coordinate directions equally over the entire domain and consistent refinement ensures that the mesh quality either stays constant or improves with mesh refinement.

Systematic mesh refinement for structured grids is usually performed by selecting the finest grid and then uniformly coarsening the grid along all the coordinate directions by the factor selected for grid refinement. Alternatively, systematically refined mesh levels can be obtained by selecting a consistent set of mesh transformation equations from an evenly spaced computational space (ξ, η, ζ) to the physical space (x, y, z), performing mesh refinement on the computational domain, and then applying the mesh transformation to the refined computational domain. If the transformation functions used are continuous then such a refinement is systematic. This latter process is used in the current study to generate hybrid unstructured systematic meshes by starting from a family of systematically refined structured meshes, then splitting structured cells from parts of the domain into multiple unstructured cells. This technique was previously implemented in generating sets of systematically-refined unstructured grids by Veluri et al. [31].

A few compressible CFD codes (e.g, Premo [3, 24], WIND [24], and Loci/CHEM [31, 32]) and an incompressible code (MFIX [6]) have been comprehensively verified using MMS applying the systematic mesh refinement approach. Herbert and Luke [9] used an alterna-

tive, statistical approach to MMS which employs grid shrinking and successfully verified the Loci/CHEM CFD code for multi-species, laminar Navier-Stokes equations using both statistical and traditional MMS. Thomas et al. [28] used a similar approach of computational windows to isolate and verify specific elements of the computational scheme applied to the FUN3D CFD code. These are only a few examples of MMS applied to CFD codes (see Refs. [11, 21, 15] for more examples).

The procedure for applying MMS with order of accuracy test is summarized as follows [23, 15]:

1. Choose the form of the governing equations.
2. Choose the form of the manufactured solution.
3. Derive the modified governing equations.
4. Solve the discrete form of the modified governing equations on multiple, systematically-refined meshes.
5. Evaluate the global discretization error in the numerical solution.
6. Apply the order of accuracy test to determine if the observed order of accuracy matches the formal order.

MMS is based upon the philosophy that code verification deals with mathematics of a given problem and hence arbitrary functions (with certain continuity and other requirements) can be selected as exact solutions. Manufactured solutions need not be physically realistic, but should be selected based upon certain criteria outlined below [23, 15]:

1. Manufactured solutions should be smooth, analytic functions of spatial (or temporal, in case of time accuracy verification) coordinates.
2. The derivatives of a manufactured solution should not vanish within the domain, including the cross-derivative terms (if they appear in the governing equations).
3. Care should be taken that one term does not dominate the other terms (e.g., a manufactured solution for a Navier-Stokes code verification should be such that the convective and the diffusive terms are of the same order of magnitude).
4. The manufactured solutions should be realizable within the code (e.g., non-positive temperature values in the MS cannot be accepted if the code uses square root of temperature to calculate the speed of sound).

The manufactured solution for each boundary condition is generated by modifying the baseline manufactured solution to satisfy the boundary constraints. The baseline manufactured solution selected is a combination of sine and cosine functions and takes the following general form

$$\phi(x,y,z) = \phi_0 + \phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_z f_{\phi z}\left(\frac{a_{\phi z}\pi z}{L}\right) + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right) + \phi_{yz} f_{\phi yz}\left(\frac{a_{\phi yz}\pi yz}{L^2}\right) + \phi_{zx} f_{\phi zx}\left(\frac{a_{\phi zx}\pi zx}{L^2}\right) \tag{2.7}$$

where L represents a characteristic length, $\phi = [\rho, u, v, w, p]^T$ represents manufactured solutions for any of the primitive variables, and the functions, $f_{\phi x}(\cdot)$, $f_{\phi y}(\cdot)$, etc. represent sine or cosine functions. The sinusoidal functions used in the current work are given in Table 2.2. The constants, ϕ_0, ϕ_x, $a_{\phi x}$, etc. are selected based upon the criteria for selecting manufactured solutions mentioned before. Since trigonometric functions have been used, it is ensured that there is reasonable periodicity of solutions and their derivatives within the domain. (See Ref. [11, 21, 15] for more examples of continuous manufactured solutions.) The characteristic length parameter can be used to modify the periodicity of manufactured solutions and their derivatives within the domain. In the current study, we select the characteristic length as equal to the domain length. The manufactured solution constants used for compressible flow and incompressible flow cases are given in Table 2.3 and Table 2.4, respectively. The 2D manufactured solutions can be obtained from the 3D manufactured solutions (Eq. 2.7) by removing the terms involving the third coordinate direction.

The general approach adapted in this work to obtain manufactured solutions for the purpose of code verification of boundary conditions is based upon the approach developed by Bond et al. [3]. The method primarily involves multiplying a baseline standard manufactured solution as described in Eq. 2.7 by a function representing the boundary surface. This procedure is explained here with the help of a simple example in 2D. A baseline 2D manufactured solution for the steady-state form of governing equations can be written from Eq. 2.7 as follows

$$\phi(x,\ y) = \phi_0 + \phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right) \tag{2.8}$$

Suppose the goal is to derive a manufactured solution for ϕ such that it satisfies the following conditions at a given boundary expressed as $F(x,y) = C$ in the 2D domain, where C is some scalar constant:

1. $\phi = \phi_0$, and
2. the derivatives of ϕ normal to the boundary up to order $m-1$ are zero.

The manufactured solution satisfying these conditions is obtained by multiplying the sinusoidal parts of the manufactured solution from Eq. 2.8 with $(C - F(x,y))^m$ as shown in

Eq. 2.9

$$\phi_{BC}(x,y) = \phi_0 + (C - F(x,y))^m \left(\phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right)\right) \tag{2.9}$$

In further discussion and derivations of boundary condition manufactured solutions, we specify the baseline manufactured solution of Eq. 2.7 with the following notation

$$\phi = \phi_0 + \phi_1 \tag{2.10}$$

where ϕ_0 is the base term, and ϕ_1 is the variable term consisting of sinusoidal functions. The base term, ϕ_0, can be a constant or a variable function depending on the requirements of the boundary condition being verified.

The MMS-based code verification approach discussed here can be applied to both compressible and incompressible flow codes given the selected manufactured solutions meet the criteria described earlier (i.e., smooth, analytic, continuous, balanced, and realizable). Note that, there is no separate evolution equation for pressure and temperature in the Navier-Stokes equations. This is addressed differently in compressible flows than in incompressible flows. For compressible flows, the continuity equation is a transport equation for density which is directly related to the fluid pressure in the flow via the equation of state. For incompressible flows, the density is constant and the continuity equation is simply a constraint on the velocity field which is expressed as the divergence free condition. MFIX employs a SIMPLE-based algorithm [16, 26] to solve for pressure using a pressure-correction (or pressure-projection) method. Assuming an initial pressure field, the momentum equations are solved to obtain an intermediate velocity field. A pressure-correction equation is then formulated and solved under the assumption of continuity (divergence-free flow) and momentum conservation. The pressure-correction solution is then used to correct the initial pressure field and the intermediate velocity field. This process is iteratively repeated until all conservation equations are satisfied within a given tolerance. Though possible, it would require modifications at several steps in the MFIX code to allow for a general manufactured solution to be used for code verification. Modification of important steps of the algorithm is not advised during a code verification exercise. In other words, a manufactured solution without a divergence-free velocity field is not effectively realizable within this code violating one of the criteria for selection of manufactured solutions.

To derive the manufactured solutions for 3D flows satisfying the divergence-free condition, we propose a new curl-based approach. Since the divergence of the curl of a 3D vector field is identically zero, one can select the manufactured solution for the 3D velocity field, $\vec{V}$, as

$$\vec{V} = \nabla \times \vec{H} \tag{2.11}$$

where $\vec{H} = \{u(x,y,z), v(x,y,z), w(x,y,z)\}^T$ is a general 3D vector field consisting of functions of the form described in Eq. 2.7. The pressure manufactured solution is selected directly

using Eq. 2.7. Furthermore, for verification of boundary conditions with incompressible flows, the manufactured solutions must be constructed with boundary constraints as well as the divergence-free constraint which is discusses in Sec.2.5.

2.4 Mesh types

For rigorous boundary condition code verification, the most general grid on which a boundary condition should be verified must have

1. a surface function for the boundary with sufficient curvature,
2. cells with variable quality metrics (skewness, stretching, etc.), and
3. cells of different types (hexahedral, tetrahedral, prismatic cells) at the boundary.

For compressible flow verification using Loci/CHEM, the 3D hybrid grids consist of hexahedral cells, tetrahedral cells, prismatic cells, and curved surfaces as boundaries (with skewness and stretching). Fig. 2.1(a) shows a general hybrid mesh used in this work for which the mesh generation equations are given as

$$\begin{aligned} x &= 0.03\sin(2\pi\eta) + 0.03\sin(2\pi\zeta) + 0.04\ \eta + 0.05\ \zeta + \xi, \\ y &= -0.03\sin(2\pi\xi) - 0.1\xi + \eta, \quad \text{and} \quad z = -0.03\sin(2\pi\xi) - 0.1\xi + \zeta \end{aligned} \tag{2.12}$$

where ξ, η, and ζ are coordinates on an evenly spaced grid of unit dimensions. The computational domain for all the grids discussed in this work assumes a domain reference length of $L = 1$ m in each coordinate direction. The inflow and wall BCs are tested on the $S_{\xi 0}$ surface as defined in Eq. 2.13 which is obtained by substituting $\xi = 0$ into Eq. 2.12. Whereas, the outflow BCs are tested on the $S_{\xi 1}$ surface as defined in Eq. 2.14 which is obtained by substituting $\xi = 1$ into Eq. 2.12.

$$S_{\xi 0}(x, y, z) \equiv x - 0.03\sin(2\pi y) - 0.03\sin(2\pi z) - 0.04y + 0.05z = 0 \tag{2.13}$$

$$\begin{aligned} S_{\xi 1}(x, y, z) \equiv x &- 0.03\sin(2\pi(y + 0.1)) - 0.03\sin(2\pi(z + 0.1)) \\ &- 0.04(y + 0.1) - 0.05(y + 0.1) - 1 = 0 \end{aligned} \tag{2.14}$$

The numerical constants selected in Eq. 2.12 ensure sufficient curvature for the surface to be tested. For example, the difference in maximum and minimum elevation over the 3D wave representing $S_{\xi 0}$ as shown in Fig. 2.1(b) is about 9% of the domain length. Grids with even more complexity (e.g., larger cell quality variations) are possible, but may lead to iterative non-convergence and/or require unfeasibly large number of mesh nodes to achieve asymptotic convergence.

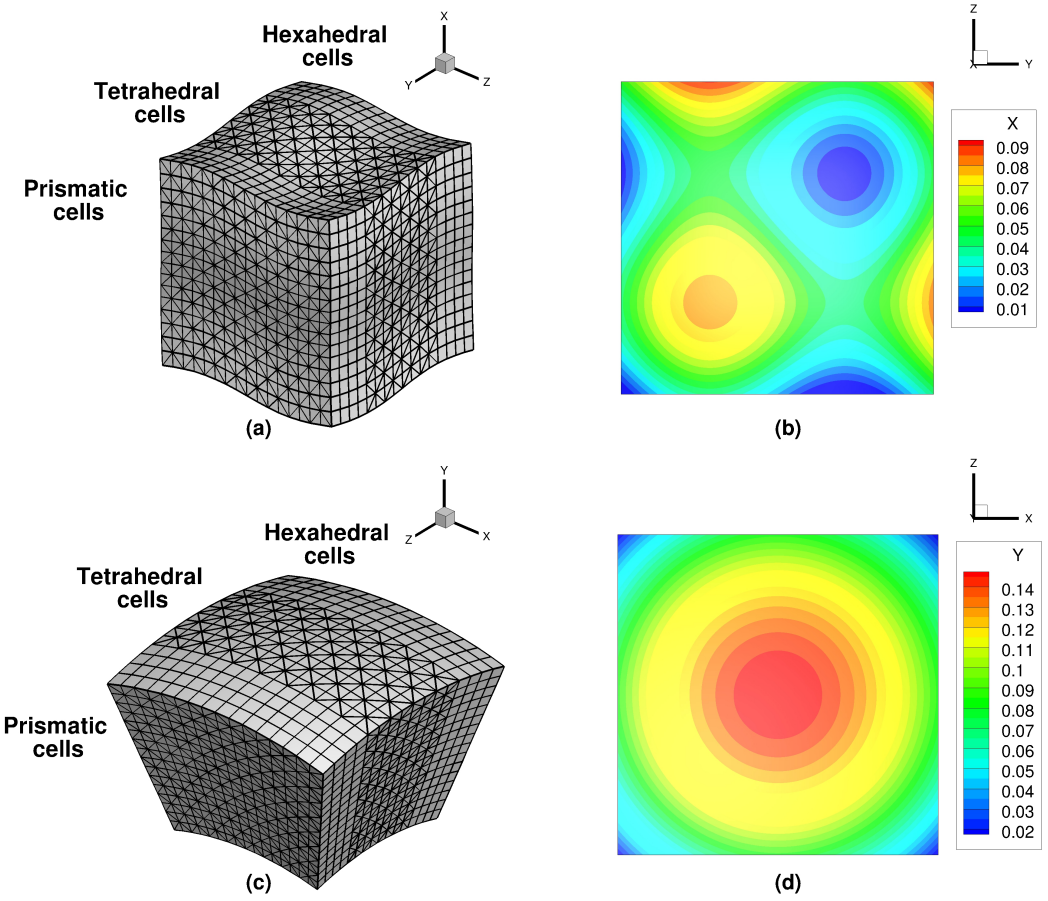

Figure 2.1: Grids used for boundary condition verification with the compressible solver: (a) general 3D mesh with hybrid boundaries at $\xi = 0$ and $\xi = 1$, (b) $S_{\xi 0}$ boundary for the general 3D mesh, (c) annular 3D mesh with hybrid boundaries at $r = r_{min}$ and $r = r_{max}$, and (d) S_{r0} boundary for the annular 3D mesh. Note: ξ corresponds to x in subfigures (a)(b) while r corresponds to y in subfigures (c)(d).

The boundary conditions addressed here require multiple constraints to be satisfied by the manufactured solution at the curved surface. A general curved boundary as shown in Fig. 2.1(b) makes it difficult to correctly and intuitively devise a manufactured solution for some of the boundary conditions (discussed in Sec. 2.5). An annular hybrid grid where the verified boundary surfaces follow equations of a spherical cap as shown in Fig. 2.1(c) is used for easy selection of manufactured solutions. The mesh nodes are generated using the following set of equations

$$\begin{aligned} &x = 0.5 + r\sin(\theta), \quad z = 0.5 + r\sin(\gamma), \quad \text{and} \\ &y = -1.8 + \sqrt{(x-0.5)^2 + (z-0.5)^2 + r^2}, \end{aligned} \tag{2.15}$$

where θ and γ each vary uniformly in the range $[-\pi/12, \pi/12]$, and r varies uniformly in the range $[1.95, 2.95]$. For the annular hybrid mesh, the inflow and wall BCs are tested on the S_{r0} surface as defined in Eq. 2.16 while the outflow BCs are tested on the S_{r1} surface as defined in Eq. 2.17. The subscripts "$r0$" and "$r1$" in Eqs. 2.16 and 2.17 correspond to the substitution of $r = r_{min} = 1.95$ and $r = r_{max} = 2.95$ into Eq. 2.15, respectively. It is evident that these spherical cap surfaces are parts of a sphere centered at $(x, y, z) = (0.5, -1.8, 0.5)$. The difference in maximum and minimum elevation over the spherical cap as shown in Fig. 2.1(d) is about 13% of the domain length.

$$S_{r0}(x, y, z) \equiv \sqrt{(x-0.5)^2 + (y+1.8)^2 + (z-0.5)^2} - 1.95 = 0 \tag{2.16}$$

$$S_{r1}(x, y, z) \equiv \sqrt{(x-0.5)^2 + (y+1.8)^2 + (z-0.5)^2} - 2.95 = 0 \tag{2.17}$$

The mesh nodes generated from Eq. 2.12 for the general grid and from Eq. 2.15 for the annular grid are then appended with appropriate nodal connectivity information such that mixed type cells are present on the boundaries under consideration. In this process, no new nodes are added, 50% of the hexahedral cell are each subdivided into five tetrahedral cells each, and 25% of the hexahedral cells are each subdivided into two prismatic cells each. This process is also used to generate grids with all hexahedral, all tetrahedral, or all prismatic cells where necessary.

MFIX employs a structured, staggered-grid solver and is equipped to work with 2D and 3D Cartesian grids with stretched cells. The MFIX algorithm under test is currently not equipped to work with grids having curved, skewed or rotated cells. For incompressible, divergence-free flow verification using MFIX, the 2D and 3D stretched Cartesian grids selected are shown in Fig. 2.2(a) and Fig. 2.2(b), respectively. The mesh nodes are generated using an internal layer grid equation as

$$\begin{aligned} &x = \xi + 2.5(0.4 - \xi)(1 - \xi)\xi, \quad y = \eta + 2.5(0.4 - \eta)(1 - \eta)\eta \quad \text{and} \\ &z = \zeta + 2.5(0.4 - \zeta)(1 - \zeta)\zeta, \end{aligned} \tag{2.18}$$

where ξ, η, and ζ are coordinates on an evenly spaced grid of unit dimensions. The maximum stretching factor (i.e., cell size ratio of two consecutive cells) ranges from 1.72 on the coarsest mesh (9x9x9) to 1.04 on the finest mesh (129x129x129) ensuring sufficient variation in the cell quality at the boundaries. The planar boundary surfaces on these grids are simply given as

$$\begin{aligned} &S_{x0}(x,y,z) \equiv x = 0, \quad S_{y0}(x,y,z) \equiv y = 0, \quad S_{z0}(x,y,z) \equiv z = 0 \\ &S_{x1}(x,y,z) \equiv x = 1, \quad S_{y1}(x,y,z) \equiv y = 1, \quad \text{and} \quad S_{z1}(x,y,z) \equiv z = 1. \end{aligned} \tag{2.19}$$

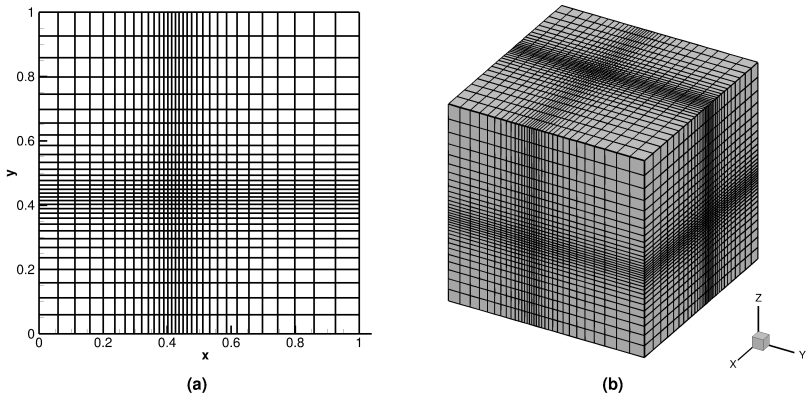

Figure 2.2: Grids used for BC verification with the incompressible solver: (a) 2D stretched Cartesian mesh, and (b) 3D stretched Cartesian mesh.

In this work, results for only the most complex grid type tested are presented. Simplified grid types are discussed only where necessary. A successful verification on a complex grid type (e.g., 3D general hybrid grid with curved boundaries) implies successful verification on the corresponding simpler grid types (e.g., 3D Cartesian grid with planar boundaries).

2.5 Results and discussion

2.5.1 Baseline governing equations

Code verification is performed for each boundary condition separately by allowing the boundary condition algorithm implemented in the code to determine the values at the concerned boundary. The solution variables on all other boundaries are specified using the Dirichlet values from the selected manufactured solution. Thus, code verification of the baseline governing equations using Dirichlet specification of all solution variable at all boundaries is a necessary step before proceeding to boundary condition code verification. This ensures that

the code has been verified in the interior of the domain and that any errors observed during boundary condition verification result from issues in the boundary condition being tested.

The compressible flow baseline governing equations presented in Sec.2.2.1 are verified on 3D general hybrid and annular hybrid grids using subsonic and supersonic manufactured solutions. Baseline manufactured solutions are selected and code verification is performed with Dirichlet specification of all the solution variables at all domain boundaries. The implementation of Dirichlet BC in Loci/CHEM sets the outer state from the user-specified conditions, determines the inner state using interior state and characteristic information, and then combines the two states using Roe's flux scheme to formulate the boundary flux. Fig. 2.3(a) shows a representative plot for the baseline density manufactured solution. The order of accuracy approaches the formal (second) order of accuracy of the code for both L_2 and L_∞ norms of the discretization error as shown in Fig. 2.3(b) and Fig 2.3(c), respectively. Similarly, for supersonic manufactured solutions all variables are established to be second order accurate for 3D general hybrid and annular hybrid meshes.

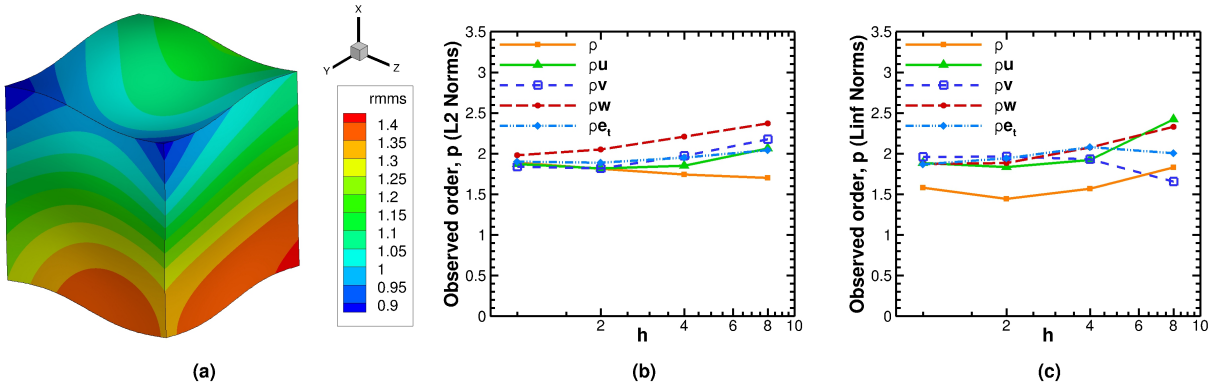

Figure 2.3: Baseline compressible flow code verification on the general 3D hybrid mesh: (a) density manufactured solution, (b) order of accuracy with L_2 norms of the discretization error, and (c) order of accuracy with L_∞ norms of the discretization error.

The incompressible flow baseline governing equations discussed in Sec. 2.2.2 are verified on 3D stretched Cartesian grids using manufactured solutions with the divergence-free velocity field of the form described in Eq. 2.11 and pressure manufactured solution of the form described in Eq. 2.7. The resulting x-, y-, z-velocity, and pressure manufactured solutions are shown in Fig. 2.4(a)-(d). The constants selected for the manufactured solutions demonstrate sufficient variation in all solution variables over the domain. The observed order approaches the formal order for both L_2 and L_∞ norms of discretization error as shown in Fig. 2.4(e)(f).

For boundary condition verification, we focus on the strictest of the global error norms, namely the L_∞ norms, as L_1 or L_2 error norms will be dominated by positive results from the rest of the domain possibly masking any issues at the boundaries within feasible number

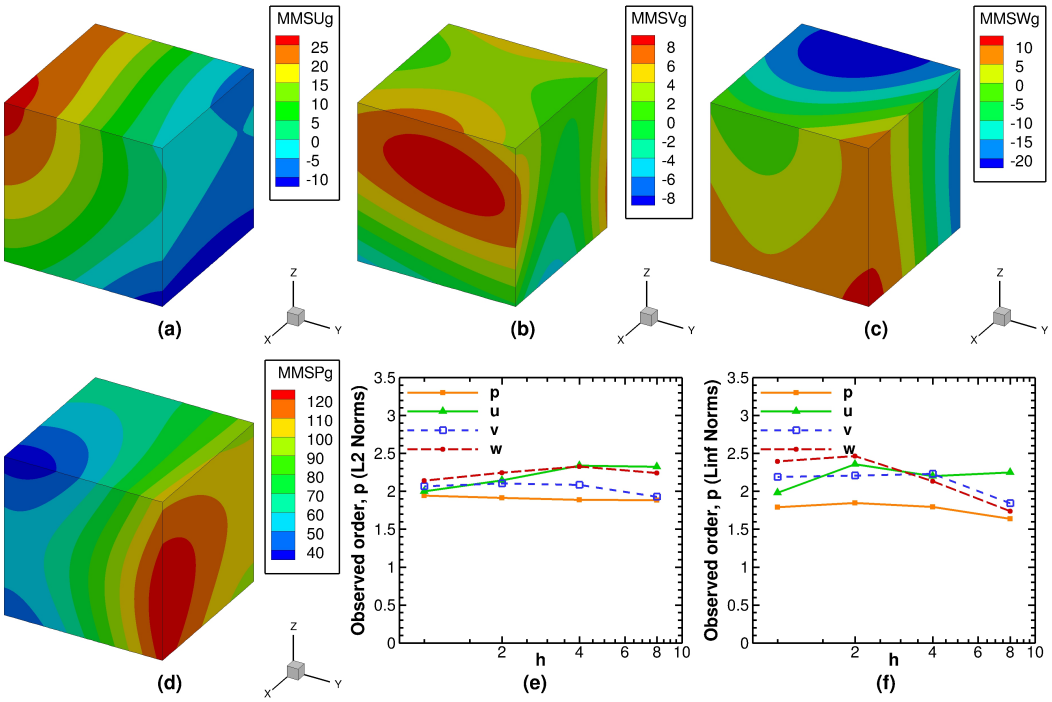

Figure 2.4: Baseline incompressible flow code verification on 3D stretched Cartesian mesh: (a) x-velocity, (b) y-velocity, (c) z-velocity, and (d) pressure manufactured solutions; order of accuracy using (e) L_2 and (f) L_∞ norms of the discretization error.

of mesh levels. Unless mentioned otherwise, all order verification results presented in this work use solutions on five mesh levels with 9×9×9, 17×17×17, 33×33×33, 65×65×65, and 129×129×129 mesh nodes.

2.5.2 Compressible flow

Subsonic inflow boundary conditions

The two subsonic inflow boundary conditions discussed here are isentropic inflow and fixed-mass inflow. These boundary conditions are verified for laminar Navier-Stokes equations using the compressible flow solver. The isentropic inflow BC is a subsonic inflow boundary condition which requires input specification of: (a) total pressure, p_t, and total temperature, T_t, as constants at the boundary and (b) a direction vector for the flow (which is assumed to be normal if flow directionality is not specified). The mathematical formulation for the isentropic inflow BC [4, 13] assumes that the flow through the boundary is isentropic as well as adiabatic. Based upon the specified p_t and T_t values, and assumption of adiabatic flow (i.e., total enthalpy is conserved), a quadratic equation is formulated to evaluate the speed of sound, and consequently flow velocity, Mach number, pressure, and temperature to determine the right (domain interior) state at the boundary. The left (domain exterior) state is determined using the interior conditions.

One possible approach for obtaining the manufactured solution for the isentropic inflow BC is to select the variables such that

1. ρ and p are constant at the boundary,
2. the velocity magnitude ($\sqrt{u^2 + v^2 + w^2}$ in 3D) is constant at the boundary,
3. the gradients of ρ and p normal to the boundary are zero,
4. u, v, and w are such that velocity vector is normal to the boundary,
5. the gradients of u, v, and w normal to the boundary are zero.

Conditions 1 and 2 ensure that constant p_t and T_t requirements are met, condition 3 is necessary to ensure that the flow through the boundary is adiabatic and isentropic, and condition 4 ensures a normal inflow velocity (assuming normal flow during this test). Condition 5 is required because the implementation of this boundary condition under the physical conditions of isentropic, adiabatic flow assumes zero gradients for velocity variables near the boundary.

Deriving a manufactured solution from these boundary constraints for a general surface is extremely complicated. It can be shown that the simplest possible manufactured solution on a general curved surface satisfies conditions 1-4 while violates condition 5. The use of such a

manufactured solution on a general curved boundary can be shown to give erroneous results. For example, consider a 2D grid where the equation for the isentropic inflow boundary curve is given as

$$F_{BC}(x,y) \equiv y - x\sin\left(\frac{5\pi}{180}\right) - 0.05\sin(2\pi x) = 0. \tag{2.20}$$

A simple manufactured solution giving adiabatic, isentropic and normal flow to the curved boundary of Eq. 2.20 is given as (in SI units)

$$\rho = 1, \quad p = 1\times 10^5, \quad u = \frac{c_0\partial_x(F_{BC})}{\Psi}, \quad \text{and } v = \frac{c_0\partial_y(F_{BC})}{\Psi} \tag{2.21}$$

where, $c_0 = 80$ m/s is the magnitude of velocity at the boundary, and $\Psi = \sqrt{[\partial_x(F_{BC})]^2 + [\partial_y(F_{BC})]^2}$. The velocity streamlines for the manufactured solution of Eq. 2.21 are shown in Fig. 2.5(a) which has non-zero gradients of the velocity variables (i.e., u and v) normal to the boundary. Consequently, the numerically converged solution results in an incorrect solution for velocity as shown in Fig. 2.5(b) and Fig. 2.5(c) where the velocity is normal yet non-isentropic at the curved boundary. Incorrect boundary solution results in an incorrect solution for the entire flow over the domain.

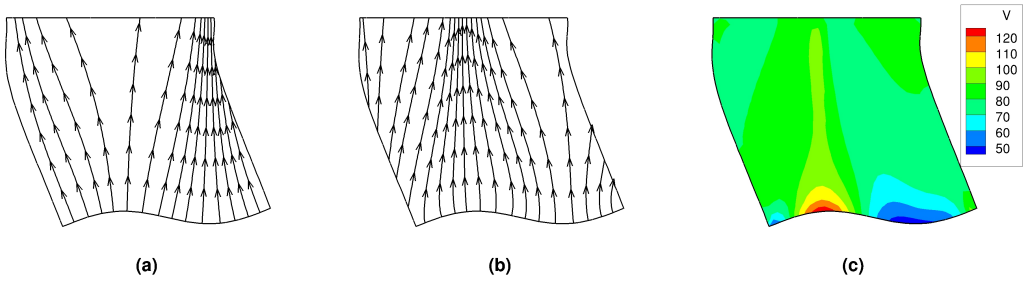

Figure 2.5: (a) Velocity streamlines for selected manufactured solution, (b) Velocity streamlines for iteratively converged numerical solution, and (c) velocity magnitude for the numerically converged solution, when isentropic inflow BC is used with the 2D manufactured solution of Eq. 2.21.

To address this issue, a set of annular grids such that the concerned boundary is a spherical cap as shown in Fig. 2.1(d) is used for the isentropic inflow BC verification. This allows us to determine the base terms of manufactured solutions for the x-, y-, and z-velocity components by multiplying the respective direction cosines of the normal vector to the surface with the constant velocity magnitude at the inflow boundary. For example, the x-direction cosine for the concerned surface, S_{r0}, is multiplied to the velocity magnitude, c_0, to get the base term as $u_0 = c_0 l_x$, where l_x is the x-direction cosine of the normal vector to the surface, S_{r0},

defined as

$$l_x = \frac{\partial_x \left(S_{r0}\right)}{\sqrt{\left[\partial_x \left(S_{r0}\right)\right]^2 + \left[\partial_y \left(S_{r0}\right)\right]^2 + \left[\partial_z \left(S_{r0}\right)\right]^2}} \tag{2.22}$$

The variable term of the manufactured solution is obtained as discussed in Eq. 2.9, by multiplying the variable part of the baseline manufactured solution (e.g., u_1 for x-velocity) with the surface function raised to the second power. The combination of base term and the variable term thus obtained meets all 5 conditions required for the manufactured solution to test the isentropic inflow BC. Finally, the manufactured solution for verification of the isentropic inflow BC on 3D annular grids can be written as

$$\begin{aligned}
&\rho = \rho_0 + \left(S_{r0}\right)^2 \rho_1, \quad p = p_0 + \left(S_{r0}\right)^2 p_1, \quad u = \frac{c_0 \partial_x \left(S_{r0}\right)}{\Psi} + \left(S_{r0}\right)^2 u_1, \\
&v = \frac{c_0 \partial_y \left(S_{r0}\right)}{\Psi} + \left(S_{r0}\right)^2 v_1, \quad w = \frac{c_0 \partial_z \left(S_{r0}\right)}{\Psi} + \left(S_{r0}\right)^2 w_1, \\
&\text{where } \Psi = \sqrt{\left[\partial_x \left(S_{r0}\right)\right]^2 + \left[\partial_y \left(S_{r0}\right)\right]^2 + \left[\partial_z \left(S_{r0}\right)\right]^2}
\end{aligned} \tag{2.23}$$

where velocity magnitude of $c_0 = 80$ m/s ensures subsonic flow. For the manufactured solution of Eq. 2.23, Figs. 2.6(a) and 2.6(b) show that there is small variation in state variables at the inflow boundary, while Fig. 2.6(c) shows the normal direction of the inflow velocity vector. The observed order of accuracy plot in Fig. 2.6(e) shows that this boundary condition is verified to be second-order accurate on 3D grids with hexahedral cells at the boundary.

Similar to the isentropic inflow BC, the fixed-mass inflow BC is applicable only to subsonic inflow conditions and requires the specification of (a) mass flux, $\dot{m}/A$, through the boundary, (b) total temperature, T_t, at the boundary, and (c) a direction vector for the inflow at the boundary (assumed normal to the boundary here). The mathematical formulation is similar to that for isentropic inflow BC and assumes adiabatic, isentropic flow through the boundary surface. Consequently, the manufactured solution for fixed-mass inflow BC is same as shown in Eq. 2.23 for isentropic inflow BC and 3D annular grids with hexahedral cells are used for verification. During the simulation, fixed-mass inflow BC is specified using constant values of mass flux, $\dot{m}/A = 80$ kg/m^2/s, and total temperature, $T_t = 350.65$ K, calculated using the selected manufactured solutions. The flow is ensured to be subsonic by selecting the velocity magnitude, $c_0 = 80$ m/s. Initial verification of the fixed-mass BC for Loci/CHEM gave negative results. This was found to be because of an incorrect implementation of the boundary condition which was preserving total enthalpy instead of total temperature at the boundary. Although not shown here for brevity, the corrected version of the code was then verified to be second order accurate on 3D annular grids.

Both the subsonic inflow boundary conditions discussed in this section are verified to be second order accurate using laminar Navier-Stokes equations on annular grids with only hexahedral cells at the inflow boundary. Cases with hybrid boundaries failed the order test for

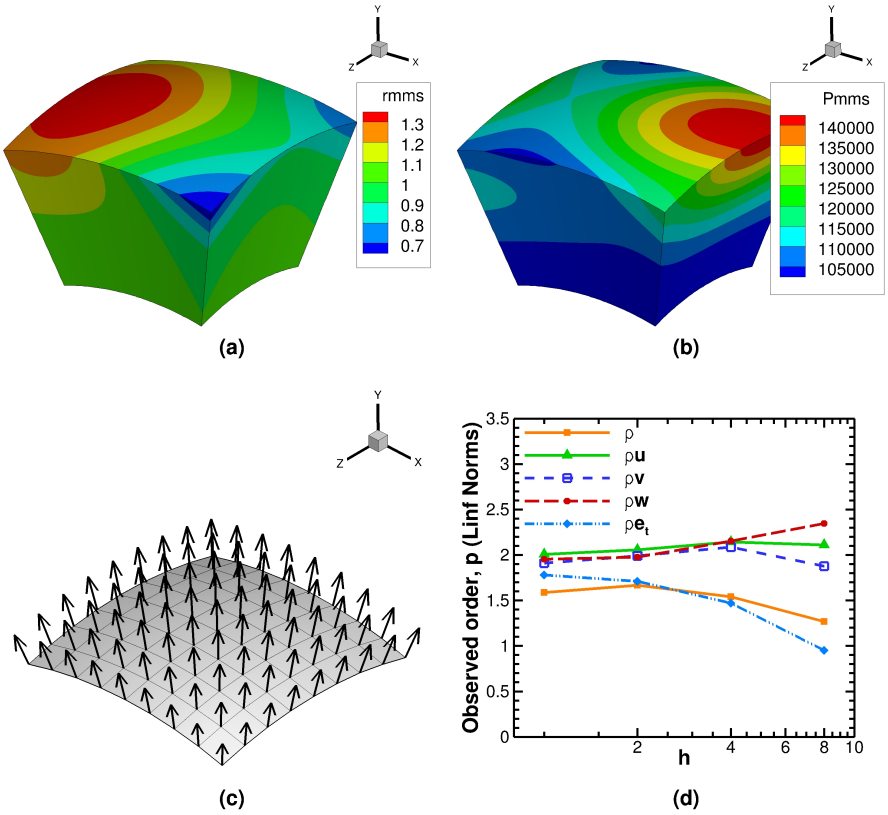

Figure 2.6: Isentropic inflow manufactured solution on 3D annular grid with hexahedral cell boundary: (a) density, (b) pressure, (c) velocity vectors, and (d) observed order of accuracy using L_∞ norms of discretization error.

these boundary conditions when laminar Navier-Stokes equations were solved while showed iterative non-convergence for Euler equations indicating that there are some unknown issues in using hybrid cells at the boundaries while ensuring isentropic, adiabatic, subsonic inflow. When Euler equations were used along with only hexahedral cells at the inflow boundary, the discretization errors were readily found to decrease at second order with mesh refinement for the same manufactured solutions presented in Eq. 2.23.

Outflow boundary conditions

The two outflow boundary conditions addressed here are the subsonic outflow and the supersonic extrapolation-based outflow. Both these boundary conditions are tested for laminar Navier-Stokes equations using the compressible flow solver on 3D general grids with hybrid boundaries shown in Fig. 2.1(a).

The subsonic outflow boundary condition imposes a user-specified constant static pressure at the outflow. The implementation of this boundary condition involves extrapolating the velocity and temperature from the interior which is mathematically imposed using a zero derivative, usually zero gradient or zero curvature, condition on these variables. The velocity extrapolation in Loci/CHEM is done with zero velocity gradient at the boundary which is code specific. A CFD code may assume zero velocity curvature (i.e., second derivative of velocity variables are zero) at the subsonic outflow boundary in which case the manufactured solutions presented here will require appropriate modification. Density is updated using the extrapolated temperature and the specified static pressure under the assumption that flow is adiabatic and isentropic through the outflow boundary. The outflow velocity vector need not be normal to the boundary but must be subsonic and flowing out of the domain.

One possible approach to select the manufactured solutions based upon the subsonic outflow boundary constraints is as follows:

1. the gradients of u, v, and w normal to the boundary are zero,
2. ρ and p are constant at the boundary,
3. the gradients of ρ and p normal to the boundary are zero,
4. u, v, and w are selected such that flow is subsonic and outward at the boundary.

Condition 1 is based upon the zero gradient implementation in the code which also ensures isentropic, adiabatic flow at the boundary. Conditions 2 and 3 impose the requirements of isentropic and adiabatic flow through the boundary on state variables. To ensure a subsonic outflow, the base terms of the manufactured solutions for u, v, and w components of the velocity vector are selected as constants. Such a selection of functions ensures that all solution variables are constant at the boundaries and that their derivatives normal to the

boundaries are zero. Finally, the manufactured solution for the verification of the subsonic outflow BC is given as

$$\rho = \rho_0 + (S_{\xi 1})^2 \rho_1, \quad p = p0 + (S_{\xi 1})^2 p_1, \quad u = u0 + (S_{\xi 1})^2 u_1,$$
$$v = v0 + (S_{\xi 1})^2 v_1, \quad w = w0 + (S_{\xi 1})^2 w_1. \tag{2.24}$$

For boundaries with hexahedral cells using Euler equations, all variables are verified to be second order accurate as shown in Fig. 2.7(a)(b). For hybrid boundaries using laminar Navier-Stokes equations, it is found that the observed orders of accuracy for mass and energy variables are less than second order and around 1.5 as shown in Fig. 2.7(d). Examining the errors in density solution, as shown in Fig. 2.7(c), and pressure solution (not shown here), it is found that large errors are present at the outflow boundary for density and pressure solution, and at the interfaces between the cells of different kinds. The source of the this error is currently unknown. We think that the problem could be related to a first order approximation relating the face center locations and the cell center locations of tetrahedral and prismatic cells.

The supersonic extrapolation-based BC is used to establish zero gradients on velocity, pressure and temperature variables at the boundary for supersonic outflow conditions. There are no other constraints or user specifications required for this boundary condition except that flow should be supersonic and going outward. The manufactured solution used for verification is similar to that in Eq. 2.24 using the constants for supersonic compressible flow given in Table 2.3. Fig. 2.8(a) shows the outflow velocity vectors, and Fig. 2.8(b) shows the observed order of accuracy using L_∞ norms of discretization error. This boundary condition is verified to be second order accurate on the 3D general grid with hybrid boundaries.

Wall boundary conditions

The three wall boundary conditions addressed in this section are the adiabatic no-slip wall, the isothermal no-slip wall, and the slip-wall. The no-slip wall boundary conditions are used for modeling viscous walls and are tested here using the laminar Navier-Stokes equations on 3D general hybrid grids. The formulation requires no-slip condition on all velocity variables at the boundary. For adiabatic no-slip wall BC, the temperature gradients normal to the wall must be zero while for isothermal no-slip wall BC, the temperature values must approach a constant value at the boundary. The manufactured solutions for velocity variables are obtained by multiplying the boundary surface equation to the baseline velocity manufactured solutions of Eq. 2.7. For the adiabatic no-slip wall BC, the manufactured solutions for the state variables are selected to ensure adiabatic conditions as shown in Eq. 2.25. For isothermal no-slip wall BC, the density manufactured solution is formulated in terms of the temperature and pressure manufactured solutions using the perfect gas equation as shown in Eq. 2.26 since a temperature manufactured solution could not be directly input in

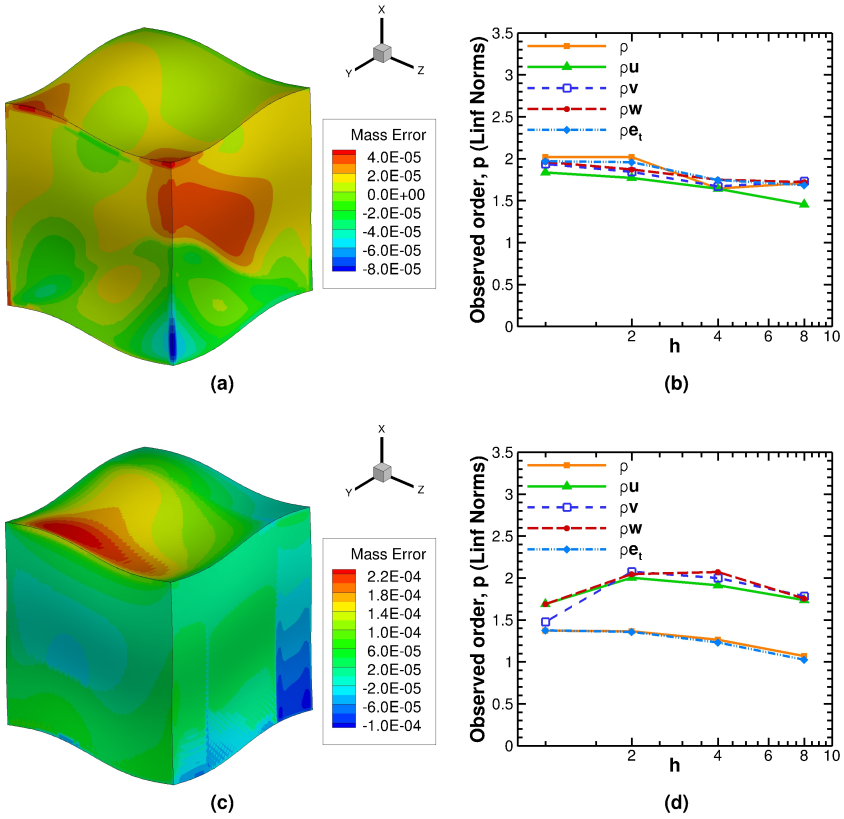

Figure 2.7: Subsonic outflow boundary condition verification on 3D general grids: (a) mass errors, (b) observed order of accuracy using L_∞ norms of discretization error, for hexahedral cells using Euler equations; (c) mass errors, (d) observed order of accuracy using L_∞ norms of discretization error, for hybrid boundaries using laminar Navier-Stokes equations.

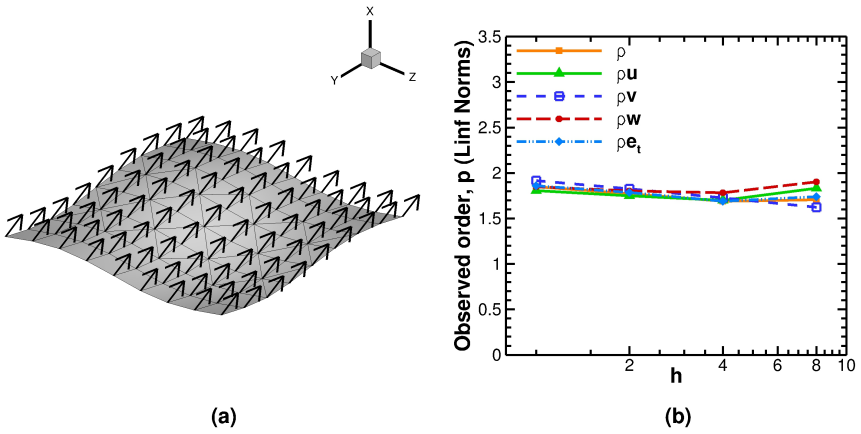

Figure 2.8: Supersonic extrapolation-based outflow boundary condition verification on 3D general hybrid grid with laminar Navier-Stokes equations: (a) velocity vectors at the outflow boundary, and (b) observed order of accuracy using L_∞ norms of discretization error.

Loci/CHEM manufactured solution input routines.

$$\begin{aligned} &\rho = \rho_0 + (S_{\xi 0})^2 \rho_1, \quad p = p_0 + (S_{\xi 0})^2 p_1, \\ &u = S_{\xi 0}\left(u_0 + u_1\right), \quad v = S_{\xi 0}\left(v_0 + v_1\right), \quad w = S_{\xi 0}\left(w_0 + w_1\right). \end{aligned} \tag{2.25}$$

$$\begin{aligned} &\rho = \frac{p_0 + S_{\xi 0} p_1}{R\left(T_0 + S_{\xi 0} T_1\right)}, \quad p = p_0 + S_{\xi 0} p_1, \\ &u = S_{\xi 0}\left(u_0 + u_1\right), \quad v = S_{\xi 0}\left(v_0 + v_1\right), \quad w = S_{\xi 0}\left(w_0 + w_1\right). \end{aligned} \tag{2.26}$$

Both the no-slip wall boundary conditions are verified to be second order accurate on general 3D grids with hexahedral cells as shown in Fig. 2.9. For grids with hybrid boundaries, both adiabatic and isothermal no-slip wall BCs fail the order accuracy test for mass and pressure variables as shown in Figs. 2.10(b) and (d) using the L_∞ norms of discretization error. The errors are found to be located near the interfaces of hexahedral-tetrahedral and tetrahedral-prismatic cells as shown in Figs. 2.10(a) and (c). The L_2 norms of discretization error for these cases show near second order accuracy possibly indicating that the errors in state variables are concentrated only at the mixed cell interfaces of the no-slip wall.

The slip wall BC is implemented using a second order characteristic-based impermeable wall formulation. This boundary condition is tested here with Euler equations on 3D annular grids with hexahedral cells at the boundary. For hybrid grids, slip-wall boundary condition is more robust when implemented with a symmetry wall formulation instead which is not investigated in this work. Here, the slip wall BC is employed with an impermeable wall formulation for inviscid flow over stationary walls where the boundary constraints prescribe

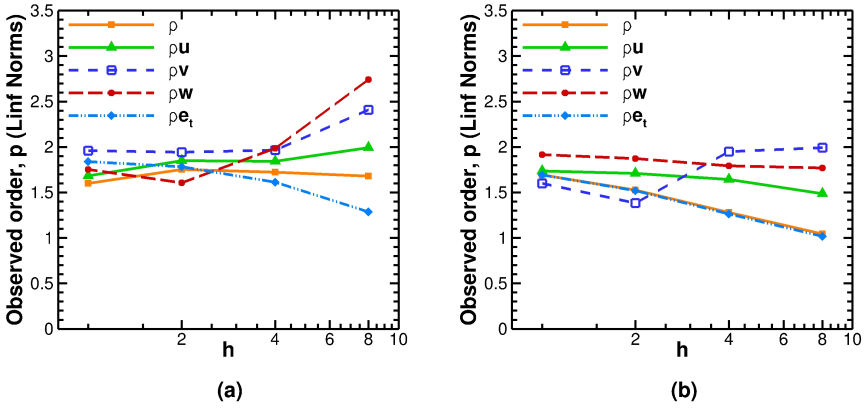

Figure 2.9: No-slip wall boundary condition verification on 3D grids with hexahedral cells using laminar Navier-Stokes equations: observed order of accuracy using L_∞ norms of discretization error for (a) adiabatic wall conditions, and (b) isothermal wall conditions.

that the flow should have both zero mass flux and zero heat flux through the wall. This translates to requirements on the manufactured solutions that flow must be tangent to the boundary and temperature gradients normal to the wall must be zero. The tangency requirement for velocity can be written as:

$$\nabla S_{r0} \cdot \vec{V} = 0 \Rightarrow u l_x + v l_y + w l_z = 0 \tag{2.27}$$

where, l_x, l_y, l_z, are the direction cosines of the normal vector to the boundary surface as described in Eq. 2.22. Note that Eq. 2.27 represents a plane in 3D since there could be infinite vectors tangential to a curved surface at a given point. Thus, we impose an additional constraint that the velocity vector, $\vec{V}$, should be parallel to a fixed vector, $\vec{d} = \hat{i} + \hat{k}$, where $\hat{i}$ and $\hat{k}$ are Cartesian unit vectors in x and z directions, respectively. This selection of $\vec{d}$ as a fixed vector (i.e., not changing with respect to (x, y, z)) leads to some loss of generality, but a more general expression should be used carefully and any discontinuities in the manufactured solutions must be avoided. Based upon these two conditions, the relationship between the three components of velocity at the slip-wall is given as:

$$v = -\frac{u\,(l_x + l_z)}{l_y} \quad \text{and } u = w. \tag{2.28}$$

Finally, a possible manufactured solution for slip-wall verification can be given as

$$\begin{aligned} &\rho = \rho_0 + (S_{r0})^2 \rho_1, \quad p = p_0 + (S_{r0})^2 p_1, \quad u = \frac{y'}{\Psi}\,(u_0 + u_1)\,, \\ &v = -\frac{(x' + z')}{\Psi}\,(u_0 + u_1)\,, \quad w = \frac{y'}{\Psi}\,(u_0 + u_1)\,, \\ &\text{where } \Psi = \sqrt{(x' + z')^2 + 2(y')^2}. \end{aligned} \tag{2.29}$$

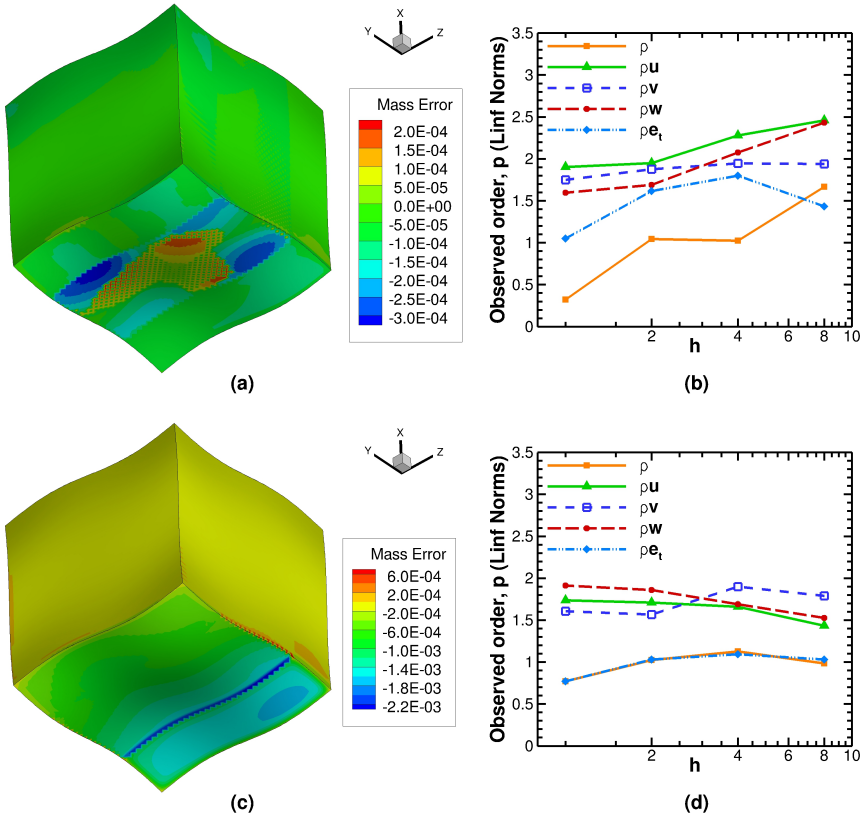

Figure 2.10: No-slip wall boundary condition verification on 3D general hybrid grid with laminar Navier-Stokes equations: (a) mass errors, and (b) observed order of accuracy using L_∞ norms of discretization error, for adiabatic conditions; (c) mass errors, and (d) observed order of accuracy using L_∞ norms of discretization error, for isothermal conditions.

where, $x' = (x - 0.5)$, $y' = (y + 1.8)$, and $z' = (z - 0.5)$ are the scaled direction cosines of the unit normal vector at the concerned spherical cap surface. The baseline function, $(u_0 + u_1)$, in Eq. 2.29 is the same for all the velocity variables which ensures that the velocity vector satisfies tangential flow at the boundary, $\nabla S_{r0} \cdot \vec{V} = 0$. The scaling parameter, Ψ, is selected simply for convenience such that $\sqrt{u^2 + v^2 + w^2} = u_0 + u_1$. The expressions for state variables (i.e., pressure and density) follow the same derivation as in Eq. 2.23 for isentropic subsonic inflow.

As shown in Fig. 2.11(a), the direction of the flow over the surface is such that there is some variation in all three coordinate directions at the wall boundary. Second order accuracy is achieved as demonstrated by Fig. 2.11(b) for slip-wall when hexahedral cells are used at the boundary.

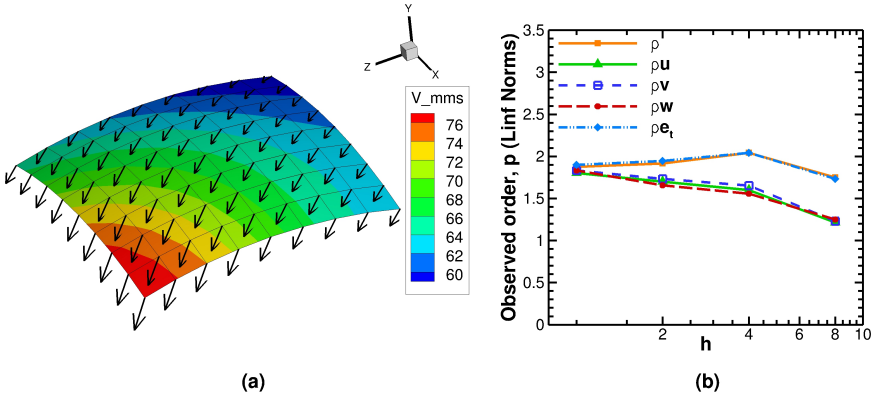

Figure 2.11: Slip wall boundary condition verification on 3D annular grid with hexahedral cells using Euler equations: (a) velocity vectors at the slip wall boundary, and (b) observed order of accuracy using L_∞ norms of discretization error.

2.5.3 Incompressible (divergence-free) flow

The three boundary conditions discussed here for divergence-free flows are the no-slip wall, the slip wall, and the pressure outflow. The selection of manufactured solutions for no-slip wall is discussed in greater detail than for the other two boundary conditions.

Wall boundary conditions

Apart from the divergence-free flow condition, the implementation in the incompressible code, MFIX, prescribes that for a no-slip wall the velocity must be zero at the wall. For a slip-wall, to ensure tangency at the boundary, the implementation in the code sets the

normal component of the velocity vector as zero while the tangential components of velocity are set equal to those in the ghost cell resulting in a zero gradient normal to the wall for these (tangential) components. There are no constraints on pressure at the wall boundaries. The mathematical formulation of these constraints and the boundaries on which these BCs are verified are presented in Table 2.1.

Table 2.1: Mathematical constraints on the no-slip wall, slip-wall, and pressure outflow boundary conditions in the incompressible solver

	No-slip wall	Slip wall	Pressure outflow
Tested boundary	$S_{x0} \equiv x = 0$	$S_{x0} \equiv x = 0$	$S_{y1} \equiv y = 1$
Constraints	$\nabla \cdot \vec{V} = 0$	$\nabla \cdot \vec{V} = 0$	$\nabla \cdot \vec{V} = 0$
	$\vec{V} = 0$, at S_{x0}	$\vec{V} \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla u \cdot \nabla S_{y1} = 0$, at S_{y1}
		$\nabla v \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla v \cdot \nabla S_{y1} = 0$, at S_{y1}
		$\nabla w \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla w \cdot \nabla S_{y1} = 0$, at S_{y1}
			$\nabla p \cdot \nabla S_{y1} = 0$, at S_{y1}

Consider the no-slip wall boundary condition verification on the 2D stretched Cartesian mesh where the tested boundary is selected as, $S_{x0}(x, y) \equiv x = 0$. The manufactured solution satisfying the boundary constraints and divergence-free constraint can be derived as follows. Let u and v be the incompressible velocity manufactured solutions that satisfy the no-slip wall BC at S_{x0} such that

$$u = u_1 x, \quad \text{and } v = v_1 x. \tag{2.30}$$

where u_1, v_1 are functions in x and y. Then in order to satisfy the divergence-free condition,

$$\frac{\partial u_1 x}{\partial x} + \frac{\partial v_1 x}{\partial y} = 0 \Rightarrow \frac{\partial v_1}{\partial y} = -\frac{\partial u_1}{\partial x} - \frac{u_1}{x}. \tag{2.31}$$

Selecting $u_1 = u_0 x \left(1 + \sin\left(\pi(x+y)^2\right)\right)$ where u_0 is a constant, the function v_1 can be found as

$$v_1 = \int \left(-\frac{\partial u_1}{\partial x} - \frac{u_1}{x}\right) dy + f_v(x). \tag{2.32}$$

where $f_v(x)$ is the integration constant selected as $5u_0$ here to ensure a dominantly upward flow. The gauge pressure manufactured solution can be a general function. Finally, the set of manufactured solutions for this case is given as

$$\begin{aligned} u &= u_0 x^2 \left(1 + \sin\left(\pi(x+y)^2\right)\right), \\ v &= u_0 x \left(5 - 3y + \frac{x}{2}\cos\left(2\pi\left(x+y\right)\right) + \frac{1}{2\pi}\sin\left(2\pi\left(x+y\right)\right)\right), \\ p &= p_0 x^2 \left(1 + \sin\left(\pi\left(x+y\right)\right)\right). \end{aligned} \tag{2.33}$$

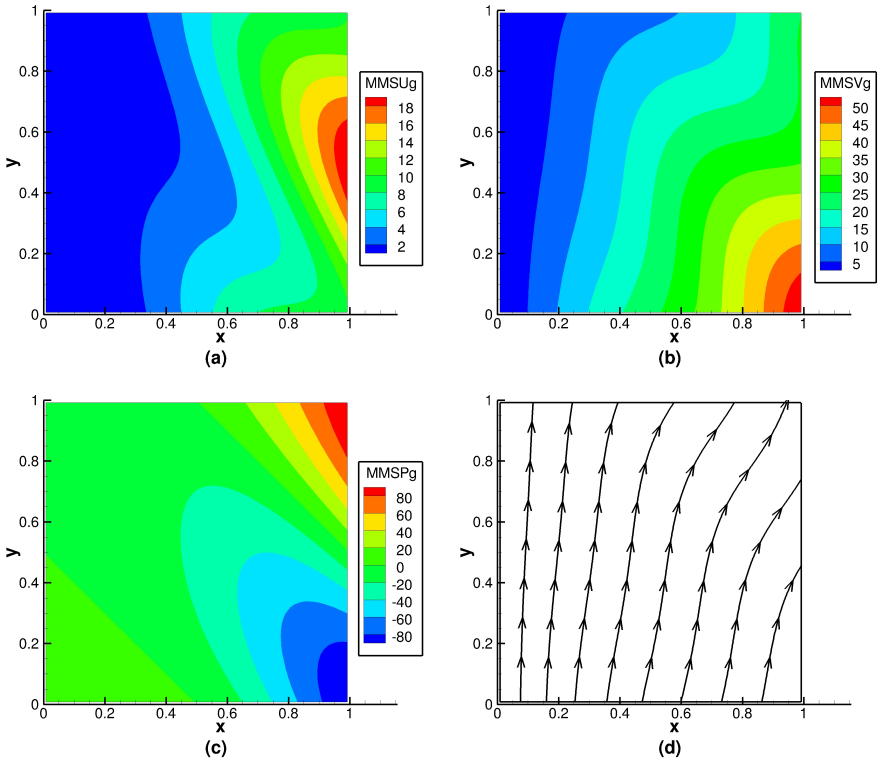

Figure 2.12: No-slip wall boundary condition manufactured solution on 2D stretched Cartesian mesh with S_{x0} as the no-slip boundary: (a) x-velocity, (b) y-velocity, (c) pressure, and (d) velocity streamlines.

For $u_0 = 10$ m/s, and $p_0 = 100$ Pa, the resulting manufactured solutions for x-velocity, y-velocity, and pressure variables are shown in Fig. 2.12.

It is evident that the procedure just described for verification of incompressible boundary conditions is cumbersome, involves trial-and-error in selection of the general functions and various constants to ensure reasonable flow variation in the domain, and is not easily extendible to 3D grids. For verification on 3D grids, a novel, more general method is presented next that uses the zero curl property of divergence-free velocity fields.

Assume that the velocity vector for the manufactured solution has the form, $\vec{V} = \vec{V}_1 S$, where $\vec{V}_1$ is a general vector function to be determined, and $S \equiv S(x, y, z) = 0$ is a general surface. This manufactured solution for velocity field, $\vec{V}$, satisfies the no-slip wall conditions at S. Requiring $\vec{V}$ to be divergence-free and using the mathematical identity that the divergence of a curl is zero, we select

$$\vec{V} = \vec{V}_1 S = \vec{\nabla} \times \vec{G} \Rightarrow \vec{V}_1 = \frac{\vec{\nabla} \times \vec{G}}{S} \tag{2.34}$$

where $\vec{G}$ is a general 3D vector field. Since the manufactured solution must be well defined over the domain, and S goes to zero at the corresponding boundary, S must be a multiplicative factor in $\vec{\nabla} \times \vec{G}$. Substituting $\vec{G} = S^2 \vec{H}$ in Eq. 2.34, where $\vec{H}$ is another general vector field, we get

$$\vec{V} = S^2 \vec{\nabla} \times \vec{H} + 2S \nabla S \times \vec{H}. \tag{2.35}$$

Here, $\vec{H}$ is selected as the general sinusoidal vector field $\{u, v, w\}^T$ described using the functions of Eq. 2.7 with the constants provided for incompressible flow in Table 2.4. The manufactured solution in Eq. 2.35 meets all the constraints for no-slip wall as presented in Table 2.1.

Following a similar method, manufactured solution is derived for the slip wall verification, the derivation of which is omitted in this discussion for for brevity. Note from Table 2.1 that the constraints on the tangential components of velocity at the slip-wall require that the gradients of v and w (i.e., ∇v and ∇w) in the direction normal to the surface (i.e., ∇S_{x0}) are set as zero. Satisfying these constraints results in a velocity vector, $\vec{V}$, that has a factor of S_{x0}^2 in the variable part. Finally, a possible manufactured solutions for slip-wall is given in Eq. 2.36.

$$\vec{V} = \vec{V}_0 + S_{x0}^3 \vec{\nabla} \times \vec{H} + 3S_{x0}^2 \nabla S_{x0} \times \vec{H} \tag{2.36}$$

where, $\vec{V}_0 = \{0, v_0, w_0\}^T$ consists of non-zero scalar constants for v_0 and w_0.

For wall boundary conditions, convergence issues were encountered for the selected manufactured solutions. This issue is likely due to the use of a source-term driven flow instead of a boundary condition driven flow and the fact that a pressure-projection algorithm is

being used. Therefore, currently we have verified the no-slip wall and the slip wall boundary condition for the most complex grid type (3D stretched Cartesian mesh) for momentum equations only by specifying the pressure solution instead of solving for it which is possible in a SIMPLE-based algorithm. The no-slip wall and slip wall boundary conditions are verified to be second order accurate for the momentum equations as shown in Fig. 2.13(a)(b).

Pressure outflow boundary condition

For the pressure outflow BC, pressure and all three velocity components are required to have zero gradients normal to the outflow boundary. The constraints shown in Table 2.1 suggest that the velocity manufactured solutions for pressure outflow can be similar to that shown in Eq. 2.36 for the slip-wall. The pressure manufactured solution follows the basic derivation for functions requiring zero gradient normal to the boundary using Eq. 2.9 with $m = 2$ and $S_{y1} \equiv y = 1$ as the tested boundary. A possible manufactured solution for verification of the pressure outflow can be written as:

$$\begin{aligned} p &= p_0 + S_{y1}^2 p_1, \\ \vec{V} &= \vec{V_0} + S_{y1}^3 \vec{\nabla} \times \vec{H} + 3S_{y1}^2 \nabla S_{y1} \times \vec{H} \end{aligned} \tag{2.37}$$

where, $\vec{V_0} = \{u_0, v_0, w_0\}^T$ consists of non-zero scalar constants. The pressure outflow BC is verified to be second order accurate for momentum and pressure solutions as shown in Fig. 2.13(c) where no convergence issues were encountered since the problem has a proper outflow boundary.

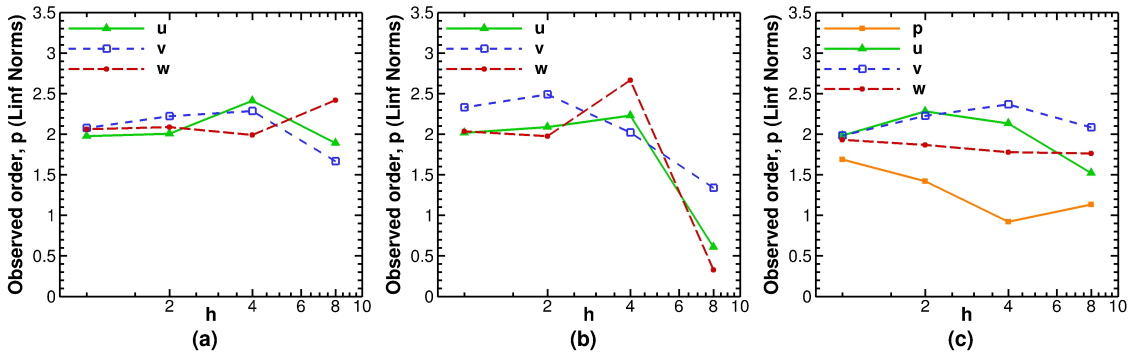

Figure 2.13: Observed orders of accuracy for boundary condition verfication using incompressible laminar Navier-Stokes equations : (a) no-slip wall, (b) slip-wall, and (c) pressure outflow.

2.6 Conclusions

In this work, we presented a code verification study of different boundary conditions commonly used in compressible and incompressible CFD simulations. We applied the techniques proposed by Bond et al. [3] to subsonic inflow, subsonic outflow, supersonic outflow, no-slip wall, and slip-wall boundary conditions with laminar Navier-Stokes equations by identifying the boundary constraints and deriving appropriate manufactured solutions for each case. Since a given boundary condition can be implemented with different formulations (such as Neumann or extrapolation-based), many of the manufactured solutions presented vary from previous works and add to the database of boundary condition verification techniques.

Verification techniques for the subsonic inflow boundary conditions were presented for the first time where we proposed the use of simplified curved boundaries (e.g., spherical caps) in order to easily derive the manufactured solutions satisfying the relevant boundary constraints. Furthermore, we proposed the use of general meshes with hybrid boundaries (i.e., boundaries containing cells of mixed types) for rigorous code verification. In the case of no-slip wall boundary conditions, we identified errors at the interface of different cell types at the wall boundary which remained masked with similar analysis on boundaries with all hexahedral cells. Similar issues were encountered when tetrahedral and prismatic cells were used at the boundaries while testing the subsonic inflow and subsonic outflow boundary conditions.

Lastly, we presented two methods for performing boundary condition code verification when a divergence-free velocity field is an added constraint. A new, curl-based method to derive manufactured solutions for divergence-free flows was proposed along with the verification of slip-wall, no-slip wall, and pressure outflow boundary conditions for an incompressible flow solver that employs a pressure-projection algorithm.

The methods we presented here are general and can be applied to verify boundary conditions in other compressible and incompressible CFD codes. For instances where the boundary conditions have been implemented under different assumptions, these methods should be modified accordingly. On the same note, it is our recommendation that code developers interested in boundary condition verification document the exact nature of boundary conditions implemented in their code specifying the physical, mathematical, numerical aspects, as well as boundary flux formulation procedures for finite-volume schemes.

Acknowledgements

The Loci/CHEM verification study was supported by the National Aeronautics and Space Administration's Constellation University Institutes Program (CUIP) with Claudia Meyer of NASA Glenn Research Center serving as program manager and Kevin Tucker and Jeffrey West of NASA Marshall Space Flight Center serving as technical monitors. The MFIX

verification study was supported by the National Energy Technology Laboratory (NETL) through URS Corp. under contract T034:4000.3.671.238.003.413 with Mike Bergen as the URS subcontract technical representative. The authors would like to thank the following members of NETL for multiple communications regarding MFIX - Mehrdad Shahnam, Jordan Musser, Jean-Francois Dietiker, Rahul Garg, Tingwen Li, and Aytekin Gel. The authors also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: http://www.arc.vt.edu

Bibliography

[1] MFIX - Multiphase Flow with Interphase eXchanges. URL `https://mfix.netl.doe.gov/`.

[2] S. Benyahia, M. Syamlal, and T.J. O'Brien. Summary of MFIX equations 2012-1, January 2012. URL `https://mfix.netl.doe.gov/documentation/Theory.pdf`.

[3] Ryan B. Bond, Curtis C. Ober, Patrick M. Knupp, and Steven W. Bova. Manufactured solution for computational fluid dynamics boundary condition verification. *AIAA Journal*, 45(9):2224–2236, 2007. ISSN 0001-1452. doi: 10.2514/1.28099. URL `http://arc.aiaa.org/doi/abs/10.2514/1.28099`.

[4] Jan-Rene Carlson. Inflow/outflow boundary conditions with application to FUN3D, NASA/TM ; 2011-217181. Technical report, Langley Research Center, Oct 2011.

[5] Aniruddha Choudhary, Christopher J Roy, Edward A. Luke, and Subrahmanya P. Veluri. Issues in verifying boundary conditions for 3D unstructured CFD codes. American Institute of Aeronautics and Astronautics, 2011. doi: doi:10.2514/6.2011-386810.2514/6.2011-3868. URL `http://dx.doi.org/10.2514/6.2011-3868`. doi:10.2514/6.2011-3868.

[6] Aniruddha Choudhary, Christopher J Roy, Jean-Franois Dietiker, Mehrdad Shahnam, and Rahul Garg. Code verification for multiphase flows using the method of manufactured solutions. In *Proceedings of the ASME 2014 4th Joint US-European Fluids Engineering Division Summer Meeting*, 2014.

[7] D. Folkner, A. Katz, and V. Sankaran. Design and verification methodology of boundary conditions for finite volume schemes. *Computers & Fluids*, 96:264275, Jun 2014. ISSN 0045-7930. doi: 10.1016/j.compfluid.2014.03.028. URL `http://dx.doi.org/10.1016/j.compfluid.2014.03.028`.

[8] Francesco Grasso and Carlo Meola. *Euler and Navier-Stokes equations for compressible flows: Finite-volume methods*, book section 4, pages 235–237. Academic Press, London,

1996. ISBN 978-0-12-553010-1. doi: 10.1016/B978-012553010-1/50005-0. URL http://www.sciencedirect.com/science/article/pii/B9780125530101500050.

[9] Shelley Hebert and Edward Luke. Honey, I shrunk the grids! a new approach to CFD verification. American Institute of Aeronautics and Astronautics, Jan 2005. ISBN http://id.crossref.org/isbn/978-1-62410-064-2. doi: 10.2514/6.2005-685. URL http://dx.doi.org/10.2514/6.2005-685.

[10] Charles Hirsch. *Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows*. John Wiley & Sons Inc, 1990. ISBN 9780471924524.

[11] Patrick Knupp and Kambiz Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman & Hall/CRC, 2003.

[12] E. A. Luke, X.-L. Tong, J. Wu, L. Tang, and P. Cinnella. CHEM: A chemically reacting flow solver for generalized grids, 2003. URL http://www.tetraresearch.com/locichem/about-locichem/.

[13] E. A. Luke, X.-L. Tong, J. Wu, P. Cinnella, and R. Chamberlain. CHEM 3.2: A finite-rate viscous chemistry solver the user guide,. Report, Mississippi State University, 2010.

[14] Edward A. Luke and Thomas George. Loci: a rule-based framework for parallel multi-disciplinary simulation synthesis. *J. Funct. Program.*, 15(3):477–502, 2005. ISSN 0956-7968. doi: 10.1017/s0956796805005514.

[15] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press Cambridge, 2010.

[16] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Series on Computational Methods in Mechanics and Thermal Science. CRC Press, New York, USA, 1980. ISBN 0891165223, 978-0891165224.

[17] Tyrone Phillips. *Residual-Based Discretization Error Estimation for Computational Fluid Dynamics*. dissertation, Virginia Tech, October 2014.

[18] T.J Poinsot and S.K Lele. Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics*, 101(1):104129, Jul 1992. ISSN 0021-9991. doi: 10.1016/0021-9991(92)90046-2. URL http://dx.doi.org/10.1016/0021-9991(92)90046-2.

[19] T. H. Pulliam. Notes on solution methods in computational fluid dynamics. Report, VKI Fluid Mechanics Lecture Series, January 20-24 1986.

[20] P. J. Roache and S. Steinberg. Symbolic manipulation and computational fluid dynamics. *AIAA Journal*, 22(10):1390–1394, October 1984. ISSN 0001-1452. doi: 10.2514/3.8794. URL http://dx.doi.org/10.2514/3.8794.

[21] Patrick J. Roache. *Fundamentals of Verification and Validation*. Hermosa Publishers, 2009. ISBN 0913478121, 9780913478127.

[22] Christopher Roy, Eric Tendean, Subramanya Veluri, Rifki Rifki, Edward Luke, and Shelley Hebert. Verification of RANS turbulence models in Loci-CHEM using the method of manufactured solutions. In *Fluid Dynamics and Co-located Conferences*. American Institute of Aeronautics and Astronautics, June 2007. doi: 10.2514/6.2007-4203. URL http://dx.doi.org/10.2514/6.2007-4203.

[23] Christopher J. Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205(1):131 – 156, 2005. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2004.10.036. URL http://www.sciencedirect.com/science/article/pii/S0021999104004619.

[24] Christopher John Roy, CC Nelson, TM Smith, and CC Ober. Verification of Euler/-NavierStokes codes using the method of manufactured solutions. *International Journal for Numerical Methods in Fluids*, 44(6):599–620, 2004. ISSN 1097-0363.

[25] Thomas Smith, Curtis Ober, and Alfred Lorber. SIERRA/Premo-a new general purpose compressible flow simulation code. In *Fluid Dynamics and Co-located Conferences*. American Institute of Aeronautics and Astronautics, June 2002. doi: 10.2514/6.2002-3292. URL http://dx.doi.org/10.2514/6.2002-3292.

[26] M. Syamlal. MFIX documentation: Numerical technique, Tech. Rep. DOE/MC31346-5824, NTIS/DE98002029. Web Page June 6, 2014, National Energy Technology Laboratory, 1998. URL https://mfix.netl.doe.gov/documentation/numerics.pdf.

[27] M. Syamlal, W. Rogers, and T.J. O'Brien. MFIX documentation: theory guide, Tech. Rep. DOE/METC-95/1013, NTIS/DE95000031. Web Page May 28, 2014, National Energy Technology Laboratory, 1993. URL https://mfix.netl.doe.gov/documentation/Theory.pdf.

[28] J. L. Thomas, B. Diskin, and C. L. Rumsey. Towards verification of unstructured-grid solvers. *AIAA Journal*, 46(12):3070–3079, December 2008. ISSN 0001-1452. doi: 10.2514/1.36655. URL http://arc.aiaa.org/doi/abs/10.2514/1.36655.

[29] Kevin W Thompson. Time dependent boundary conditions for hyperbolic systems. *Journal of Computational Physics*, 68(1):124, Jan 1987. ISSN 0021-9991. doi: 10.1016/0021-9991(87)90041-6. URL http://dx.doi.org/10.1016/0021-9991(87)90041-6.

[30] Kevin W Thompson. Time-dependent boundary conditions for hyperbolic systems, II. *Journal of Computational Physics*, 89(2):439461, Aug 1990. ISSN 0021-9991. doi: 10.1016/0021-9991(90)90152-q. URL http://dx.doi.org/10.1016/0021-9991(90)90152-Q.

[31] Subrahmanya P. Veluri, Christopher J. Roy, and Edward A. Luke. Comprehensive code verification techniques for finite volume CFD codes. *Computers & Fluids*, 70(0):59–72, 2012. ISSN 0045-7930. doi: http://dx.doi.org/10.1016/j.compfluid.2012.04.028. URL http://www.sciencedirect.com/science/article/pii/S0045793012001697.

[32] Subrahmanya Pavan Kumar Veluri. *Code Verification and Numerical Accuracy Assessment for Finite Volume CFD Codes*. dissertation, Virginia Tech, 2010. URL http://hdl.handle.net/10919/28715.

[33] H. C. Yee. Numerical approximation of boundary conditions with applications to inviscid equations of gas dynamics. Report NASA-TM-81265, A-8480, NASA Ames Research Center, 1981. URL http://ntrs.nasa.gov/?R=19810011307.

[34] H. C. Yee, R. M. Beam, and R. F. Warming. Boundary approximations for implicit schemes for one-dimensional inviscid equations of gasdynamics. *AIAA Journal*, 20(9): 1203–1211, 1982. ISSN 0001-1452. doi: 10.2514/3.51181. URL http://arc.aiaa.org/doi/abs/10.2514/3.51181.

Appendix 2.A Functions and constants in manufactured solutions

Note: The amplitude constants employ SI units while the frequency constants are dimensionless. For incompressible manufactured solutions, pressure (p) is the gauge pressure.

Table 2.2: Sinusoidal functions and frequency constants used in the compressible and incompressible manufactured solutions.

Variable, ϕ	$f_{\phi x}$	$f_{\phi y}$	$f_{\phi z}$	$f_{\phi xy}$	$f_{\phi yz}$	$f_{\phi zx}$
ρ	cos	sin	sin	cos	sin	cos
u	sin	cos	cos	cos	sin	cos
v	sin	cos	cos	cos	sin	cos
w	cos	sin	cos	sin	sin	cos
p	cos	cos	sin	cos	sin	cos
T	cos	sin	sin	cos	sin	cos
Variable, ϕ	$a_{\phi x}$	$a_{\phi y}$	$a_{\phi z}$	$a_{\phi xy}$	$a_{\phi yz}$	$a_{\phi zx}$
ρ	0.75	0.45	0.8	0.65	0.75	0.5
u	0.5	0.85	0.4	0.6	0.8	0.9
v	0.8	0.8	0.5	0.9	0.4	0.6
w	0.85	0.9	0.5	0.4	0.8	0.75
p	0.4	0.45	0.85	0.75	0.7	0.8
T	0.75	1.25	0.5	0.65	0.75	0.5

Table 2.3: Amplitude constants used in the subsonic (supersonic) compressible manufactured solutions.

Variable, ϕ	ϕ_0	ϕ_x	ϕ_y	ϕ_z	ϕ_{xy}	ϕ_{yz}	ϕ_{zx}
ρ	1	0.15	-0.1	0.1	0.08	0.05	0.12
u	70 (800)	5 (50)	-15 (-150)	-10 (-100)	7 (70)	4 (40)	-4 (-40)
v	90 (800)	-5 (-50)	10 (100)	5 (50)	-11 (-110)	-5 (-50)	5 (50)
w	80 (800)	-10 (-100)	10 (100)	12 (120)	-12 (-120)	11 (110)	5 (50)
p (10^5)	1	0.2	0.5	0.2	-0.25	-0.1	0.1
T	350	10	-30	20	10	-12	15

Table 2.4: Amplitude constants used in the subsonic, incompressible manufactured solutions.

Variable, ϕ	ϕ_0	ϕ_x	ϕ_y	ϕ_z	ϕ_{xy}	ϕ_{yz}	ϕ_{zx}
u	7	3	-4	-3	2	1.5	2
v	9	-5	4	5	-3	2.5	3.5
w	8	-4	3.5	4.2	-2.2	2.1	2.5
p	100	20	-50	20	-25	-10	10

Chapter 3

Code Verification for Multiphase Flows Using the Method of Manufactured Solutions

Aniruddha Choudhary[a], Christopher J. Roy[a], Jean-François Dietiker[b], Mehrdad Shahnam[b], Rahul Garg[b]
[a] Aerospace and Ocean Engineering Department, Virginia Tech, Blacksburg, VA 24061, USA
[b] National Energy Technology Laboratory, Morgantown, WV 26506, USA

Abstract

Code verification is the process of ensuring, to the degree possible, that there are no algorithm deficiencies and coding mistakes (bugs) in a scientific computing simulation. Order of accuracy testing using the Method of Manufactured Solutions (MMS) is a rigorous technique that is employed here for code verification of the major components of an open-source, multiphase flow code - MFIX. Code verification is performed here on 2D and 3D, uniform and stretched meshes for incompressible, steady and unsteady, single-phase and two-phase flows using the two-fluid model of MFIX. Currently, the algebraic gas-solid exchange terms are neglected as these can be verified via techniques such as unit-testing. The no-slip wall, free-slip wall, and pressure outflow boundary conditions are verified. Temporal order of accuracy for first-order and second-order time-marching schemes during unsteady simulations is also assessed. The presence of two-phase governing equations and a modified SIMPLE-based algorithm requiring divergence-free flows makes the selection of manufactured solutions more involved than for single-phase, compressible flows. A newly-developed curl-based manufactured solution is used to generate manufactured solutions that satisfy the divergence-free constraint during the verification of the incompressible governing equations. Manufactured

solutions with constraints due to boundary conditions as well as due to divergence-free flow are derived in order to verify the boundary conditions.

Keywords Multiphase flows; Code verification; Method of manufactured solutions; Order of accuracy

Nomenclature

C_{pg}	Specific heat of the fluid phase
C_{pm}	Specific heat of the mth solids phase
f_{gi}	Fluid flow resistance due to porous media
g	Subscript indicating gas phase
g_i	Acceleration due to gravity
g_t	Coefficient of temporal terms in the spatial-temporal discretization error expression
g_x	Coefficient of spatial terms in the spatial-temporal discretization error expression
h_t	Normalized temporal discretization
h_x or h	Normalized spatial discretization (measure of grid size)
ΔH_g	Heat of reaction in the fluid phase
ΔH_m	Heat of reaction in the mth solids phase
I_{gmi}	Gas-solid interaction force
I_{kmi}	Solid-solid interaction force
J_m	Collisional dissipation term for solid phase m
m	Index of the mth solids phase
n	Index of the nth chemical species
M	Total number of solids phases
N_m	Total number of chemical species for solid phase m
$\hat{p}$	Spatial observed order of accuracy
P_g	Pressure in the fluid phase
$\hat{q}$	Temporal observed order of accuracy
q_{gi}	Fluid-phase conductive heat flux
q_{mi}	Solids-phase m conductive heat flux
R_{gn}	Rate of production of the nth chemical species in the fluid phase
R_{mn}	Rate of production of the nth chemical species in the mth solids phase
T_g	Thermodynamic temperature of the fluid phase
T_m	Thermodynamic temperature of the solids phase m
T_{Rg}	Fluid phase radiation temperature
T_{Rm}	Solids phase m radiation temperature

	Greek symbols
e_g	Volume fraction of the fluid phase (void fraction)
e_m	Volume fraction of the mth solids phase
γ_{gm}	Fluid-solids heat transfer coefficient corrected for interphase mass transfer
γ_{Rg}	Fluid-phase radiative heat transfer coefficient
Θ_m	Granular temperature of phase m
Θ_s	Solid phase granular temperature
μ_g	Molecular viscosity of the fluid phase
μ_m	Molecular viscosity of the solid phase m
Π_m	Energy exchange term for the solid phase m
ρ_g	Microscopic (material) density of the fluid phase
ρ_m	Microscopic (material) density of the mth solids phase
τ_{gij}	Fluid-phase stress tensor
τ_{mij}	Solids phase m stress tensor
$\phi_0, \phi_x \ldots$	Amplitude constants in the manufactured solutions
$a_{\phi x}, a_{\phi y} \ldots$	Frequency constants in the manufactured solutions

3.1 Introduction

With increased use of computational tools for engineering simulations of physical systems, it becomes important to perform verification and validation studies for various aspects of a computational simulation. For a Computational Fluid Dynamics (CFD) simulation, verification and validation activities are useful in assessing the correctness of the code, quantifying the numerical accuracy of the simulation, and determining the applicability of the selected mathematical model. Verification deals with the mathematics of the simulation and involves assessing the correctness of the computer code and numerical algorithms as well as the accuracy of the numerical solution. Validation deals with the physics of the model and assesses whether the mathematical model selected satisfactorily predicts the physics of interest.

There are two fundamental aspects to verification: code verification and solution verification. Code verification is the process of ensuring, to the degree possible, that there are no coding mistakes (bugs) and algorithm inconsistencies. Solution verification focuses on identifying and estimating various kinds of errors present in numerical simulations: round-off error, numerical iterative error and discretization error. The current work is concerned with performing code verification of various features commonly implemented in a multiphase CFD code. The different criteria for assessing code verification are: expert judgment, error quantification, consistency/convergence, and order of accuracy. Out of these, the order of accuracy test is considered to be the most rigorous one [21, 29]. Order of accuracy test requires the evaluation of discretization error on multiple grid levels. Discretization error is defined as the difference between the numerical solution to the discretized equations and the exact solution to the partial differential (or integral) equations. Evaluation of discretization

error requires knowing the exact solution for the governing equations which is generally not known for problems of practical interest. A technique called the Method of Manufactured Solutions (MMS) is used where a solution is "manufactured" and used as an exact solution [25]. This manufactured solution exactly solves the original governing equations modified by introducing source terms that are generated by substituting the manufactured solutions in the original governing equations. MMS is based upon the philosophy that code verification deals with the mathematics of the problem and hence arbitrary functions (with certain requirements as discussed later) can be selected as exact solutions. The books by Roache [27], Knupp and Salari [19], and Oberkampf and Roy [21] provide a comprehensive discussion of code verification, MMS, and order of accuracy tests.

CFD simulations of fluid-solids multiphase systems can be categorized into two basic types: (1) continuum approaches, and (2) Lagrangian-Eulerian approaches. In a continuum approach, which is also referred to as the Eulerian-Eulerian method or the Two-Fluid Method (TFM), the different phases are mathematically described as interpenetrating continua and the governing equations for mass, momentum, and energy are obtained by averaging quantities over a control volume. The interaction between the phases is modeled using various sub models commonly referred to as constitutive relations or closure models. The constitutive relations can be used to formulate interphase exchange terms describing various physical interactions such as momentum transfer and heat transfer between fluids and solid, or solid and solid phases. (See [20, 14, 22] for discussion on modeling of interphase exchange terms) Special constitutive relations (such as interphase drag models, solid-stress models) are needed for practical problems of interest and are developed from experimental data, or theoretical modeling, or first-principles based numerical simulations (such as direct numerical simulations, e.g. [33]). Although assessment and improvement of multiphase constitutive models are important processes in multiphase fluid dynamics, they are not the main focus of the current work. Methods where the carrier (or surrounding) phase is treated as a continuum and the dispersed phase is treated as discrete entities (i.e., particles or parcels of particles) are called Lagrangian-Eulerian methods or continuum discrete methods. Code verification of Lagrangian-Eulerian methods is not addressed in the current study, though some of the ideas presented here for code verification of the two-fluid model can be used to verify specific components of the other models (for example, the carrier-phase flow simulation within the continuum discrete model).

Code verification of multiphase flows is not as common in the literature as that for single-phase flows. This is because compared to single-phase flows, the presence of at least twice as many governing equations in multiphase flows, complex interphase interaction terms, and multiple constitutive relations make it difficult to obtain manufactured solutions or simple exact solutions for these equations. In [15], Grace and Taghipour discussed the importance of verification and validation activities for CFD models as applied to fluidized beds and other dense multiphase flow systems. In addition, they correctly concluded from a survey of articles claiming "verification" or "validation" for numerical models simulating fluidized beds that these terms have often been used incorrectly.

There have been some MMS-based multiphase code verification studies in a multi-material context. "Multiphase" in this sense refers to the presence of materials in the domain with different physical properties thus resulting in solution discontinuities at the material interface. Brady et al. [6] presented a way to apply MMS to the finite volume multiphase code OSM which is a structured, Cartesian code for solving the heat equation. Manufactured solutions were generated using Heaviside and Dirac-delta functions to include the presence of moving interfaces in the domain for a typical immiscible two-phase system. They concluded that with such a discontinuity in material properties, the order of accuracy must reduce to first order for a second or higher order discretization scheme. This conclusion is also supported by Banks et al. [3] who showed that the formally second order accuracy of the discrete system reduces to first order in the presence of nonlinear discontinuities and to non-integer values below one for linear discontinuities. Roache et al. [26] used MMS to verify a finite-difference ground flow code with discontinuous conductivities in the domain by selecting the manufactured solutions such that they explicitly satisfy the geological boundary conditions. Crockett et al. [8] applied MMS to verify a multi-material heat equation solver that uses a Cartesian cut-cell/embedded boundary method to represent the interface between the materials. In [26] and [8], the interface locations are considered to be fixed and known a priori.

Shunn et al. [30] used MMS to verify an unstructured variable density flow-solver for a miscible two-fluid system with manufactured solutions reflective of the physical behavior common to combustion problems such as convective propagation of density fronts and mixing of species through diffusion. Physically-realistic manufactured solutions for incompressible, single-phase flows, were also proposed by Eça et al. [10, 11] for code verification of turbulent, wall-bounded flows. Vedovoto et al. [35] performed a MMS-based code verification study of a pressure-based finite volume numerical scheme suited to variable density, single-phase flows generally encountered in combustion applications. In their work, the authors selected a manufactured solution mimicking the propagation of a corrugated flame front separating heavy from light gases. In all these studies, the manufactured solutions proposed satisfy the necessary criteria such as divergence-free velocity field, wall boundary conditions, or consistency with employed turbulence functions. There are some advantages in using such physically-realistic manufactured in cases such as turbulence model verification [11] where the function and roles of different terms change based upon the nature of the solution. However, the selected manufactured solution should not just be realistic but also exhibit enough variations to ensure that all terms in the governing equations are exercised during the verification test [24].

The focus of current work is MMS-based code verification of the numerical scheme implemented in the discretization of two-fluid model governing equations. We use mathematically general manufactured solutions consisting of sinusoidal functions thus ensuring a rigorous verification of all the discretized terms of the governing equations. The algorithm implemented in the code being investigated (i.e., MFIX) requires that the velocity field be divergence-free for the selected manufactured solutions. We accomplish this by introducing a novel, curl-based method to derive manufactured solution for code verification of incompressible

flows. We also verify three of the mostly commonly used boundary conditions, i.e., no-slip wall, free-slip wall, and pressure outflow. While verifying these boundary conditions, we derive the manufactured solutions to satisfy the divergence-free velocity field constraint as well as the boundary constraints. We also examine the temporal order of accuracy for first-order and second-order time-stepping schemes during unsteady simulations. Various lessons learned during this verification study are discussed alongside the results of order of accuracy tests. The current work is restricted to 3D, incompressible, laminar, two-phase flows. The emphasis is put upon systematically developing rigorous manufactured solution functions for the verification of two-fluid governing equations and approaching code verification from a mathematical point of view as opposed to using less rigorous or physically realistic manufactured solutions. We make the simplification of omitting the verification of momentum transfer models (i.e., drag models), solid-stress models, and any chemical reactions or phase transformations. These constitutive models may be sufficiently verified through other approaches (e.g., unit testing) since these are are generally algebraic in nature. For example, Garg et al. [13] verified an interphase interaction model (called the spring-dashpot model) in MFIX-DEM (the Lagrangian-Eulerian solver in MFIX) using a suite of simple problems having analytical solutions.

3.2 Code and governing equations

MFIX (Multiphase Flow with Interphase eXchanges) [2, 32] is an open-source, general-purpose, hydrodynamic code that can simulate heat transfer and chemical reactions for fluids containing multiple solid phases. It is generally used for flows which commonly occur in energy conversion and chemical reactor processes such as bubbling, circulating, and spouted fluidized beds. MFIX features different modeling methodologies to perform multiphase flow simulations such as Lagrangian-Eulerian methods (DEM and PIC models), quadrature based method (QMOM), and Eulerian-Eulerian method (or two-fluid model). Various drag models, frictional-stress models and kinetic theory models are available as constitutive relations in the multiphase framework of MFIX. For a complete description of multiphase governing equations and various constitutive relations employed in MFIX, see Ref. [4]. In Eqs. 3.1-3.9 below, we present only the governing equations which are relevant to the current discussion. (Einstein summation convention is implied for only i and j indices)

Gas continuity equation

$$\frac{\partial}{\partial t}\left(\varepsilon_g \rho_g\right) + \frac{\partial}{\partial x_i}\left(\varepsilon_g \rho_g U_{gi}\right) = \sum_{n=1}^{N_g} R_{gn} \tag{3.1}$$

Solids continuity equation

$$\frac{\partial}{\partial t}\left(\varepsilon_m \rho_m\right) + \frac{\partial}{\partial x_i}\left(\varepsilon_m \rho_m U_{mi}\right) = \sum_{n=1}^{N_m} R_{mn} \tag{3.2}$$

Note that in a SIMPLE-based algorithm, the continuity equations are not directly solved but instead lead to the formulation of a Poisson equation for pressure correction. In the absence of species reaction terms, R_{gn} and R_{mn}, Eq. 3.1 and Eq. 3.2 reduce to the divergence-free condition for steady-state, incompressible flows.

Gas momentum equation

$$\frac{\partial}{\partial t}\left(\varepsilon_g \rho_g U_{gi}\right) + \frac{\partial}{\partial x_j}\left(\varepsilon_g \rho_g U_{gj} U_{gi}\right) = -\varepsilon_g \frac{\partial}{\partial x_i}\left(P_g\right) + \frac{\partial}{\partial x_j}\left(\tau_{gij}\right) - \left(\sum_{m=1}^{M} I_{gmi}\right) + f_{gi} + \varepsilon_g \rho_g g_i \tag{3.3}$$

Solids momentum equation

$$\frac{\partial}{\partial t}\left(\varepsilon_m \rho_m U_{mi}\right) + \frac{\partial}{\partial x_j}\left(\varepsilon_m \rho_m U_{mj} U_{mi}\right) = -\varepsilon_m \frac{\partial}{\partial x_i}\left(P_g\right) + \frac{\partial}{\partial x_j}\left(\tau_{mij}\right) + I_{gmi} - \left(\sum_{k=1}^{M} I_{kmi}\right) + \varepsilon_m \rho_m g_i \tag{3.4}$$

In Eqs. 3.3 and 3.4, the porous media force terms (f_{gi}), gas-solid (I_{gmi}) and solid-solid (I_{kmi}) momentum exchange terms are modeled with algebraic models and thus not considered during the verification study. The gravity term ($\varepsilon_g \rho_g g_i$) has also been neglected. For laminar flows, the gas and solid stress models are simplified to include all the derivative terms while the algebraic closure model terms are neglected as shown in Eqs. 3.5 and 3.6.

Gas stress model

$$\tau_{gij} = 2\mu_g S_{gij} \quad \text{where, } S_{gij} = \frac{1}{2}\left(\frac{\partial U_{gi}}{\partial x_j} + \frac{\partial U_{gj}}{\partial x_i}\right) - \frac{1}{3}\left(\frac{\partial U_{gi}}{\partial x_i}\right)\delta_{ij} \tag{3.5}$$

Solids stress model

$$\tau_{mij} = 2\mu_m S_{mij} \quad \text{where, } S_{mij} = \frac{1}{2}\left(\frac{\partial U_{mi}}{\partial x_j} + \frac{\partial U_{mj}}{\partial x_i}\right) - \frac{1}{3}\left(\frac{\partial U_{mi}}{\partial x_i}\right)\delta_{ij} \tag{3.6}$$

Gas energy equation

$$\varepsilon_g \rho_g C_{pg}\left[\frac{\partial T_g}{\partial t} + U_{gj}\frac{\partial T_g}{\partial x_j}\right] = -\frac{\partial q_{gi}}{\partial x_i} + \sum_{m=1}^{M} \gamma_{gm}\left(T_m - T_g\right) - \Delta H_g + \gamma_{Rg}\left(T_{Rg}^4 - T_g^4\right) \tag{3.7}$$

Solids energy equation

$$\varepsilon_m \rho_m C_{pm}\left[\frac{\partial T_m}{\partial t} + U_{mj}\frac{\partial T_m}{\partial x_j}\right] = -\frac{\partial q_{mi}}{\partial x_i} - \gamma_{gm}\left(T_m - T_g\right) - \Delta H_m + \gamma_{Rm}\left(T_{Rm}^4 - T_m^4\right) \tag{3.8}$$

In Eqs. 3.7 and 3.8, the gas-solid heat transfer terms (γ_{gm}), the heat of reaction terms (ΔH_g and ΔH_m), and radiative heat transfer terms (γ_{Rg} and γ_{Rm}) have not been considered for

verification. The gas and solids heat conduction terms are given as, $q_{gi} = -\kappa_g \partial T_g / \partial x_i$ and $q_{mi} = -\kappa_m \partial T_m / \partial x_i$, respectively.

Solids granular energy equation

$$\frac{3}{2}\varepsilon_m \rho_m \left[\frac{\partial \Theta_m}{\partial t} + \frac{\partial \left(\Theta_m U_{mj}\right)}{\partial x_j}\right] = \frac{\partial}{\partial x_i}\left(\kappa_m \frac{\partial \Theta_m}{\partial x_i}\right) + \tau_{mij}\frac{\partial U_{mi}}{\partial x_j} + \Pi_m - \rho_m \varepsilon_m J_m \tag{3.9}$$

In Eq. 3.9, the collisional dissipation (Π_m) and granular energy exchange (J_m) terms are purely algebraic functions and not considered for verification.

The above two-fluid governing equations for the gas and solid phases are implemented within MFIX in a finite-volume formulation where the domain is discretized using structured cells. The solution variables are stored in a staggered-grid formulation where the momentum variables (e.g., fluid and solid velocities) reside at the face centers while scalar variables (e.g., pressure, temperature, volume-fraction) reside at the cell centers. For discretization of spatial derivative terms, various formally second-order centered and upwind schemes are available. For discretization of temporal derivative terms, first order (Euler implicit) and second order (2-step implicit Runge-Kutta) schemes are available. MFIX uses a modified SIMPLE-based algorithm and employs a pressure projection method which imposes a divergence-free velocity constraint for incompressible flows.

3.3 Methodology

3.3.1 Order of accuracy test

The order of accuracy test determines whether the observed order of accuracy for the numerical scheme matches the formal order of accuracy as the mesh is systematically refined into the asymptotic range. The observed order is determined directly from the simulations by observing the rate at which the norms (L_1, L_2, L_∞) of the (solution) discretization error reduce as the grid is refined. The formal order is usually obtained from a truncation error analysis of the discrete equations [29]. The asymptotic range is that range of discretization sizes (e.g., Δx, Δt) where the higher order terms of the truncation error are negligible compared to the lowest order term (i.e., the term which defines the formal order of the scheme). The observed order of accuracy will generally fail to match the formal order when there is a coding mistake (bug), any algorithm inconsistency, a discontinuity in the solution, large iterative or round-off errors, or if the solution is not in the asymptotic range.

To calculate the observed order of accuracy, the numerical solution must be calculated on multiple systematically refined mesh levels. Systematic mesh refinement [21] is defined as uniform and consistent refinement over the spatial domain. Uniform refinement ensures that the mesh has been refined along all coordinate directions equally over the entire domain and consistent refinement ensures that the mesh quality either stays constant or improves with

mesh refinement. The number of mesh levels required depends on whether the solution is in the asymptotic range and usually varies from one problem to another.

Systematic mesh refinement for structured grids is usually performed by selecting the finest grid and then uniformly coarsening the grid along all the coordinate directions by the factor selected for grid refinement. Alternatively, systematically refined mesh levels can be obtained by selecting a consistent set of mesh transformation equations from an evenly spaced computational space (ξ, η, ζ) to the physical space (x, y, z), performing mesh refinement on the computational domain, and then applying the mesh transformation to the refined computational domain. If the transformation functions used are continuous then such a refinement is systematic.

3.3.2 Mesh types for verification

MFIX employs a structured, staggered-grid solver and is equipped to work with 2D and 3D Cartesian grids with stretched cells. The algorithm currently under test is not equipped to work with grids having curved, skewed or rotated cells. For incompressible, divergence-free flow verification, the 2D and 3D stretched Cartesian grids selected are shown in Fig. 3.1(a) and Fig. 3.1(b), respectively. The mesh nodes are generated using an internal layer grid equation as

$$\begin{aligned} &x = \xi + 2.5\,(0.4 - \xi)\,(1 - \xi)\,\xi, \quad y = \eta + 2.5\,(0.4 - \eta)\,(1 - \eta)\,\eta \quad \text{and} \\ &z = \zeta + 2.5\,(0.4 - \zeta)\,(1 - \zeta)\,\zeta, \end{aligned} \tag{3.10}$$

where ξ, η, and ζ are coordinates on an evenly spaced grid of unit dimensions. The maximum stretching factor (i.e., cell size ratio of two consecutive cells) ranges from 1.72 on the coarsest mesh ($9 \times 9 \times 9$) to 1.04 on the finest mesh ($129 \times 129 \times 129$) ensuring sufficient variation in the cell quality in the domain. The planar boundary surfaces on these grids are simply given as

$$\begin{aligned} &S_{x0}\,(x, y, z) \equiv x = 0, \quad S_{y0}\,(x, y, z) \equiv y = 0, \quad S_{z0}\,(x, y, z) \equiv z = 0 \\ &S_{x1}\,(x, y, z) \equiv x = 1, \quad S_{y1}\,(x, y, z) \equiv y = 1, \quad \text{and} \quad S_{z1}\,(x, y, z) \equiv z = 1. \end{aligned} \tag{3.11}$$

3.3.3 MMS procedure

The method of manufactured solutions was first applied for code verification by Roache and Steinberg (1984) [25] where it was used to verify a code for generation of 3D transformations for elliptic PDEs. A few CFD codes (e.g, Premo [5, 28], WIND [28], and Loci/CHEM [36]) have been comprehensively verified using the method of manufactured solutions. Hebert and Luke [16] used an alternative, statistical approach to MMS which employs grid shrinking and successfully verified the Loci/CHEM CFD code for multi-species, laminar Navier-Stokes

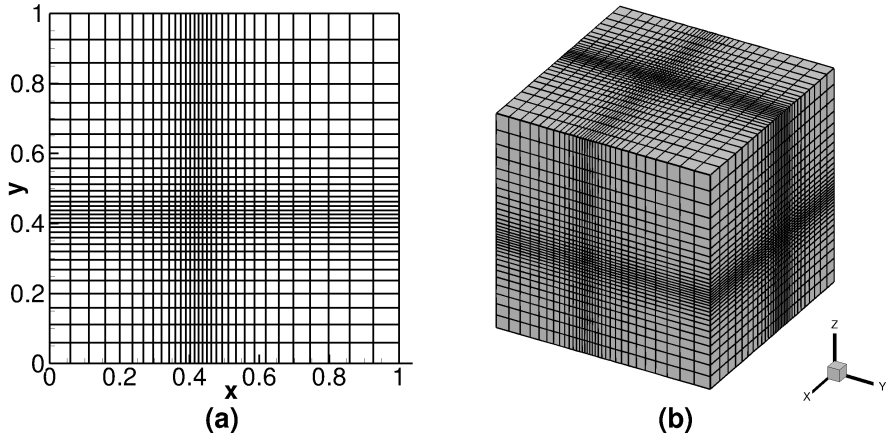

Figure 3.1: Grids used for code verification: (a) 2D stretched Cartesian mesh, and (b) 3D stretched Cartesian mesh.

equations using both statistical and traditional MMS. Thomas et al. [34] used a similar approach of computational windows to isolate and verify specific elements of the computational scheme applied to the FUN3D CFD code. These are only a few examples of MMS applied to CFD codes (see Refs. [19, 27, 21] for more examples).

The procedure for applying MMS with order of accuracy test is summarized as follows [29, 21]:

1. Choose the form of the governing equations.
2. Choose the form of the manufactured solution.
3. Derive the modified governing equations.
4. Solve the discrete form of the modified governing equations on multiple, systematically-refined meshes.
5. Evaluate the global discretization error in the numerical solution.
6. Apply the order of accuracy test to determine if the observed order of accuracy matches the formal order.

The selected manufactured solutions need not be physically realistic, but should be selected based upon certain criteria outlined below [29, 21]:

1. Manufactured solutions should be smooth, analytic functions of spatial (or temporal, in case of time accuracy verification) coordinates.

2. The derivatives of a manufactured solution should not vanish within the domain, including the cross-derivative terms (if they appear in the governing equations).

3. Care should be taken that one term does not dominate the other terms (e.g., a manufactured solution for a Navier-Stokes code verification should be such that the convective and the diffusive terms are of the same order of magnitude).

4. The manufactured solutions should be realizable within the code (e.g., non-positive temperature values in the MS cannot be accepted if the code uses square root of temperature to calculate the speed of sound).

3.3.4 Baseline manufactured solution

The baseline manufactured solution selected for the verification study is a combination of sine and cosine functions and takes the following general form [21]

$$\begin{aligned}\phi\left(x,y,z\right) = \phi_0 + \phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_z f_{\phi z}\left(\frac{a_{\phi z}\pi z}{L}\right) \\ + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right) + \phi_{yz} f_{\phi yz}\left(\frac{a_{\phi yz}\pi yz}{L^2}\right) + \phi_{zx} f_{\phi zx}\left(\frac{a_{\phi zx}\pi zx}{L^2}\right)\end{aligned} \tag{3.12}$$

where L represents a characteristic length, $\phi = [P_g, u_g, v_g, w_g, u_s, v_s, w_s, T_g, T_s, \Theta_s]^T$ represents the set of primitive variables being tested for order of accuracy, and the functions $f_{\phi x}(\cdot)$, $f_{\phi y}(\cdot)$, etc. represent sine or cosine functions. The sinusoidal functions used in the current work are given in Table 3.2. The constants, ϕ_0, ϕ_x, $a_{\phi x}$, etc. are selected based upon the criteria for selecting manufactured solutions mentioned before. Since trigonometric functions have been used, it is ensured that there is reasonable periodicity of solutions and their derivatives within the domain. (See Ref. [19, 27, 21] for more examples of manufactured solutions.) The manufactured solution constants used here are given in Table 3.3. The 2D manufactured solutions can be obtained from the 3D manufactured solutions (Eq. 3.12) by removing the terms involving the third coordinate direction. The characteristic length parameter can be used to modify the periodicity of manufactured solutions and their derivatives within the domain. In the current study, we select the characteristic length as equal to the domain length.

3.3.5 Boundary condition manufactured solution

The general approach for verification of boundary conditions involves obtaining the manufactured solution such that it satisfies all the boundary constraints as implemented within the code. For general velocity fields (i.e., without any divergence-free constraint) this method primarily involves multiplying a baseline standard manufactured solution as described in Eq. 3.12 by a function representing the boundary surface. This procedure has been previously used [5, 36, 7, 12] for the boundary condition code verification with general flows

and is briefly explained in the current section with the help of a simple example in 2D. The techniques to perform verification for divergence-free flows are introduced in Sec. 3.4.5. A baseline 2D manufactured solution for the steady-state form of governing equations can be written from Eq. 3.12 as follows

$$\phi\left(x,\ y\right) = \phi_0 + \phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right) \tag{3.13}$$

Suppose the goal is to derive a manufactured solution for ϕ such that it satisfies the following conditions at a given boundary expressed as $F\left(x,y\right) = C$ in the 2D domain, where C is some scalar constant:

1. $\phi = \phi_0$, and
2. the derivatives of ϕ normal to the boundary up to order $m-1$ are zero.

The manufactured solution satisfying these conditions is obtained by multiplying the sinusoidal parts of the manufactured solution from Eq. 3.13 with $\left(C - F\left(x,y\right)\right)^m$ as shown in Eq. 3.14

$$\phi_{BC}\left(x,y\right) = \phi_0 + \left(C - F\left(x,y\right)\right)^m \left(\phi_x f_{\phi x}\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_{\phi y}\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_{xy} f_{\phi xy}\left(\frac{a_{\phi xy}\pi xy}{L^2}\right)\right) \tag{3.14}$$

In further discussion and derivations of manufactured solutions, we specify the baseline manufactured solution of Eq. 3.12 with the following notation

$$\phi = \phi_0 + \phi_1 \tag{3.15}$$

where ϕ_0 is the base term, and ϕ_1 is the variable term consisting of sinusoidal functions. The base term, ϕ_0, can be a constant or a variable function depending on the requirements of the case being verified.

3.3.6 Divergence-free manufactured solution

The MMS-based code verification approach discussed here can be applied to both compressible and incompressible flow codes given the selected manufactured solutions meet the criteria described earlier (i.e., smooth, analytic, continuous, balanced, and realizable). However, the solution for pressure is addressed differently in compressible flows than in incompressible flows. For compressible flows, the continuity equation is a transport equation for density which is directly related to the fluid pressure in the flow via the equation of state. For incompressible flows, the density is constant and the continuity equation is simply a constraint on the velocity field which is expressed as the divergence-free condition. MFIX employs a SIMPLE-based algorithm [23, 31] to solve for pressure using a pressure-correction (or

pressure-projection) method. Assuming an initial pressure field, the momentum equations are solved to obtain an intermediate velocity field. A pressure-correction equation is then formulated and solved under the assumption of continuity (divergence-free flow) and momentum conservation. The pressure-correction solution is then used to correct the initial pressure field and the intermediate velocity field. This process is iteratively repeated until all conservation equations are satisfied within a given tolerance. Though possible, it would require modifications at several steps in the MFIX code to allow for a general manufactured solution to be used for code verification. Modification of important steps of the algorithm is not advised during a code verification exercise. In other words, a manufactured solution without a divergence-free velocity field is not effectively realizable within this code violating one of the criteria for selection of manufactured solutions.

To derive the manufactured solutions for 3D flows satisfying the divergence-free condition, we propose a new curl-based approach. Since the divergence of the curl of a 3D vector field is identically zero, one can select the manufactured solution for the 3D velocity field, $\vec{V}$, as

$$\vec{V} = \nabla \times \vec{H} \tag{3.16}$$

where $\vec{H} = \{u_g(x,y,z), v_g(x,y,z), w_g(x,y,z)\}^T$ is a general 3D vector field consisting of functions of the form described in Eq. 3.12. The pressure manufactured solution is selected directly using Eq. 3.12. Furthermore, for verification of boundary conditions with incompressible flows, the manufactured solutions must be constructed with boundary constraints as well as the divergence-free constraint which is discusses in Sec. 3.4.

3.4 Results and discussion

In this section, various MMS test cases are described with results. The problems are tested in an increasing order of complexity starting from a 2D rectangular channel single-phase flow case where the analytical solution is known, followed by 2D and 3D, single-phase and two-phase MMS cases. Temporal order verification of time-stepping schemes and verification of boundary conditions is performed for the single-phase equations with divergence-free constraint. Results for boundary condition verification include techniques to erive manufactured solutions for no-slip wall, free-slip wall, pressure outflow cases.

The results presented herein are obtained with double precision computations and the normalized residuals are iteratively converged to 1e-12. This ensures that the iterative errors are negligible in comparison with the discretization error and thus do not significantly affect the order of accuracy. Additionally, it is also ensured that the code provides same results when the simulation is performed using different compilers and grid-decompositions for parallel computations since not all simulations are run on the same software and/or hardware configurations. The difference in results for the discretization error norms using different grid decompositions and compilers are found to be of the order of 1e-10.

3.4.1 2D rectangular channel flow

It is helpful to use the method of exact solutions for cases with analytical solutions available prior to performing a rigorous MMS-based code verification exercise. This establishes some degree of confidence in the code and guides the test engineers towards modifications such as including the MMS source terms and MMS boundary conditions. In the case of plane Poiseuille flow (i.e., pressure driven) through a rectangular channel, the 2D, laminar, steady-state, Navier-Stokes equations reduce to a second order linear ODE:

$$\nu_g \frac{d^2 u_g}{dy^2} = \frac{1}{\rho_g} \frac{dP_g}{dx} \tag{3.17}$$

Eq. 3.17 is analytically solvable when dp/dx = constant and no-slip wall boundary conditions are assumed at the channel walls (at $y = 0$ and $y = H$ as shown in Fig. 3.2(a)). The analytical solution is given as:

$$u_g\,(y) = -\frac{dP_g}{dx} \frac{1}{2\rho_g \nu_g} y(H - y) \tag{3.18}$$

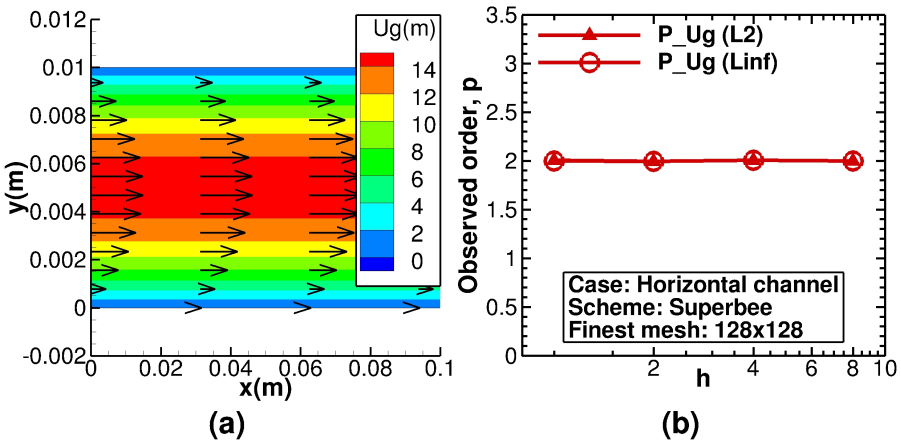

Figure 3.2: Code verification for 2D rectangular channel flow: (a) numerical solution for x-velocity, and (b) observed order of accuracy using L_2 and L_∞ norms of discretization error.

This problem is solved for single-phase calculations, with cyclic (periodic) boundary conditions along x-direction and second order scheme for numerical discretization. The flow parameters are selected as: channel length, $L = 0.2$ m; channel height, $H = 0.01$ m; fluid density, $\rho_g = 1$ kg/m^3; fluid viscosity, $\mu_g = 0.001$ Pa $\cdot$ s; and pressure drop across the channel length, $\Delta P_g = 240$ Pa. The solution for the x-velocity variable, u, results in a second order accurate solution consistent with the simplicity of this problem. Fig. 3.2(b) shows

the observed order of accuracy for the x-velocity variable using L_2 and L_∞ norms of the discretization error. Herein, h is the grid size measure which means that for a refinement factor of 2, h for the finest (128×128 cells) to coarsest (8×8 cells) mesh level will be $1, 2, 4, 8,$ and 16, respectively. Pressure having linear variation and y-velocity being zero results in exact calculations for a second order scheme and thus observed orders for these variables are undefined.

3.4.2 2D steady-state, single-phase flows

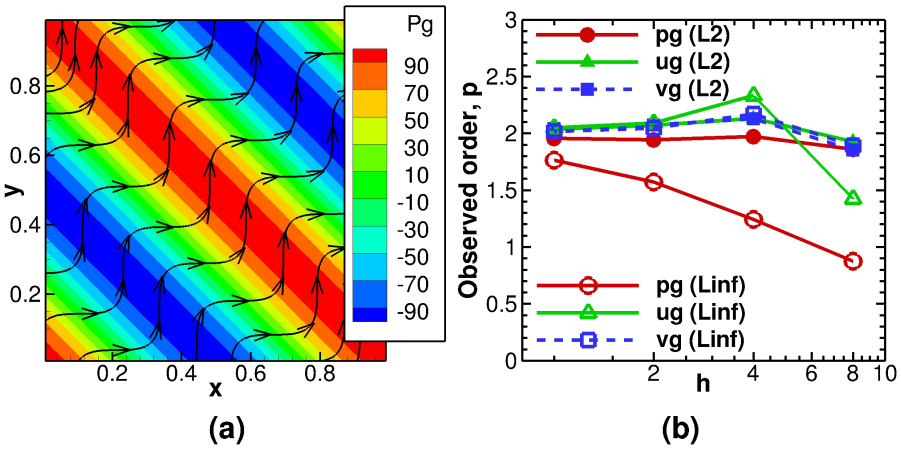

Figure 3.3: Verification for 2D, steady-state, single-phase flows using simple solenoidal manufactured solution: (a) pressure contours and velocity streamlines, and (b) observed order of accuracy using L_2 and L_∞ norms of discretization error.

For the verification of steady-state, single-phase flows on 2D grids, a simple sinusoudal divergence-free manufactured solution [35] is selected as defined in Eq. 3.19 and shown in Fig. 3.3.

$$u_g = u_{g0} \sin\left(2\pi\left(x+y\right)\right)^2 \quad v_g = v_{g0} \cos\left(2\pi\left(x+y\right)\right)^2 \quad P_g = P_{g0} \cos\left(2\pi\left(x+y\right)\right) \tag{3.19}$$

where, $u_{g0} = v_{g0} = 5.0$ m/s, and $P_{g0} = 100$ Pa. During initial testing, it was found that the calculation for the cross-terms of the strain-tensor in the momentum equations was not performed within the steady-state sub-iterations leading to incorrect results. These errors did not show up in cases with zero shear at the boundaries or for unsteady simulations. This issue was subsequently fixed for steady-state simulations. Another important finding was that the Superbee scheme (a second order upwind scheme) resulted in first order accuracy at the west, south, and bottom (for 3D) boundaries when these boundaries were set as inflows. This was an artifact of MFIX using the staggered grid with different data structure at west/south/bottom boundaries vs. east/north/top boundaries. One must be careful to make

special modifications in the scheme at the boundaries while using upwind approaches. In the rest of the analysis in this work, a central scheme has been used for numerical calculations for which the solution was found to be second order accurate as shown in Fig. 3.3.

Note that, on a staggered grid, the manufactured solutions and the MMS source terms must be evaluated where the finite-volume approximation of that variable is assumed to be located. For example, Fig. 3.4 shows a representative staggered-grid where the x-momentum and y-momentum variable are computed at the face-centers while the scalar variables (such as pressure) are solved at the cell centers. Thus, the manufactured solution and source term for the u_g variable must be evaluated and compared with the numerical solution at the location $(i-1/2, j)$, $(i+1/2, j)$ etc., where (i, j) is the cell-center.

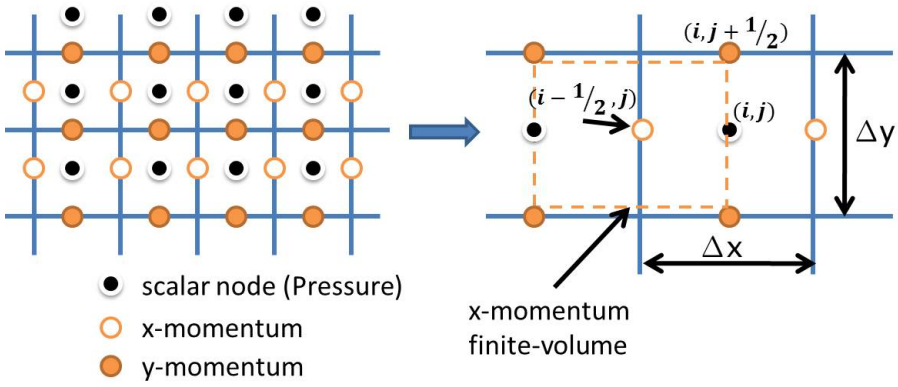

Figure 3.4: An example staggered-grid showing location of scalar and vector variables.

3.4.3 3D steady-state, single-phase flows

For code verification of 3D, single-phase, steady-state flows we use the curl-based manufactured solution presented in Eq. 3.16 for velocity while for pressure, the baseline manufactured solution shown in Eq. 3.12 is used. The x-velocity, y-velocity, z-velocity and pressure manufactured solutions are shown in Fig. 3.5. The constants selected for the manufactured solutions as given in Table 3.3 demonstrate sufficient variation in all concerned variables over the domain. The order of accuracy test results in second order accuracy for L_2 norms as shown in Fig. 3.5(a)-(d). However, the pressure variable shows an order of 1.7 in the L_∞ norms as shown in Fig. 3.5(e)(f). This was confirmed using a finer mesh level ($256 \times 256 \times 256$ cells) and it was found that the observed order of accuracy in pressure calculations does not reduce.

The larger errors in the pressure variable (though still approaching second order accuratcy) are found to be related to the quality of the mesh used for verification. For uniform meshes,

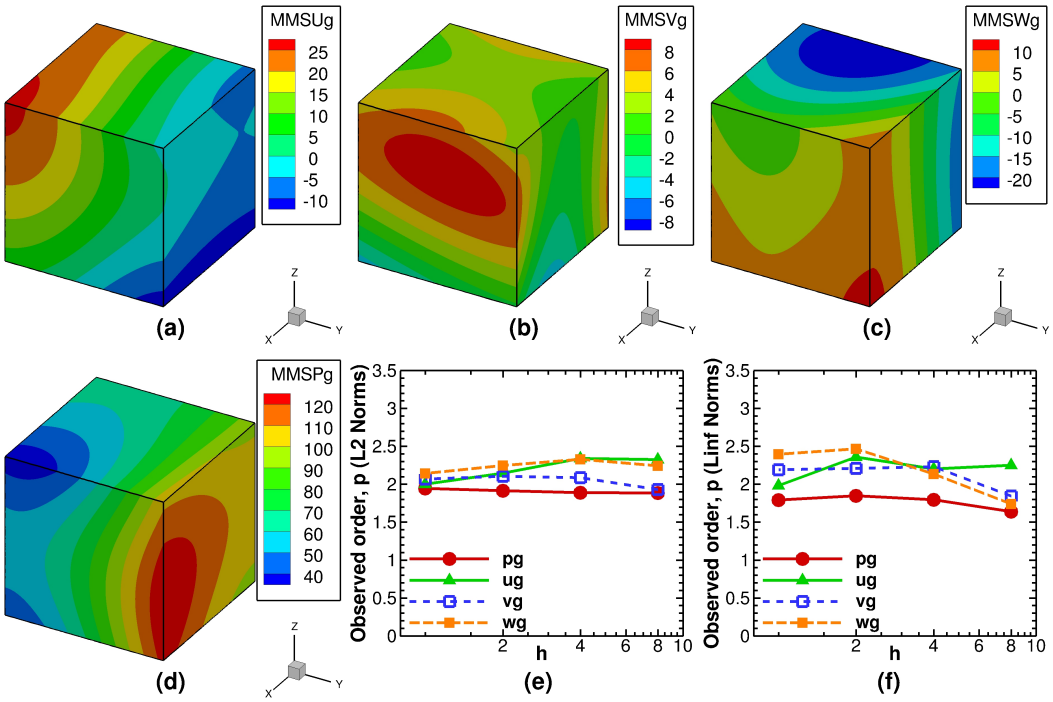

Figure 3.5: Verification for 3D, steady-state, single-phase flows using curl-based manufactured solution: (a) x-velocity, (b) y-velocity, (c) z-velocity, (d) pressure contours; and observed order of accuracy using (e) L_2 and (f) L_∞ norms of discretization error.

it was verified that all variables achieve an order of two, in L_2 and L_∞ norms (not shown here for conciseness). For the grid shown in Fig. 3.6(a), which is defined using the cubic transformation given in Eq. 3.10, the larger errors only appear at the boundaries as seen in Fig. 3.6(b). For this type of grid, the stretching factor assumes a value of one at the boundary due to a common assumption in this and other CFD codes that the ghost cell size equals the first interior cell size. Fig. 3.6(c) show another grid where constant stretching factors are used within three segments (i.e., segment 1: $0 \leq x \leq 0.25$, segment 2: $0.25 \leq x \leq 0.75$, and segment 3: $0.75 \leq x \leq 1.0$) along each coordinate direction. The stretching factor is exactly one at every segment intersection (i.e., at $x = 0.25$, $x = 0.75$, etc.). The large pressure errors are clearly seen from Fig. 3.6(d) to be along these locations of discontinuous variation in stretching factor. This analysis suggests that not having a smoothly varying mesh and having large stretching factor discontinuities at the boundaries due to the common assumption about ghost cell sizes may results in large local errors. Usually, this is not a problem for physical flows where large flow variations at the boundaries are very uncommon.

3.4.4 3D steady-state, two-phase flows

Fewer cases with analytical solutions are available for multiphase flows. One such case is the steady fully-developed two-phase flow consisting of solid dispersed phase in a fluid carrier phase between two stationary parallel walls [9, 18]. An exact solution is available when constant volume fraction and a favorable constant pressure gradient are assumed. However, the governing equations for this problem are difficult to replicate using MFIX governing equations (i.e., Eqs. 3.1-3.9) without making significant changes to the code which is counterproductive in a code verification study. Instead, the 3D two-phase flow is directly verified using general manufactured solutions.

Extension of the 3D, single-phase, steady-state relations from the previous section to two-phase flows is relatively straightforward; however, there is more work involved due to the presence of all variables for a constant volume-fraction, incompressible flow case (i.e., $\phi = [P_g, u_g, v_g, w_g, u_s, v_s, w_s, T_g, T_s, \Theta_s]^T$. The source terms are generated using the Mathematica symbolic manipulation software [1] but can contain hundreds of terms. This does not result in a significant computational overhead since the manufactured solutions and the source terms are evaluated only once during each time-step (or only once during the simulation for steady-state flows).

The manufactured solutions for both fluid and solid velocity fields are required to be divergence-free, and can be defined using either the curl-based approach mentioned in Eq. 3.16 or a simple solenoidal velocity field similar to that in Eq. 3.19 with a constant value for w_g in the third direction. In this case different constant values constants could be used for fluid and solid velocity manufactured solutions. The manufactured solutions for pressure, fluid temperature, solid temperature, and granular temperature are selected using the general form described in Eq. 3.12 and are shown in Figs. 3.7(a)(b)(c). All governing

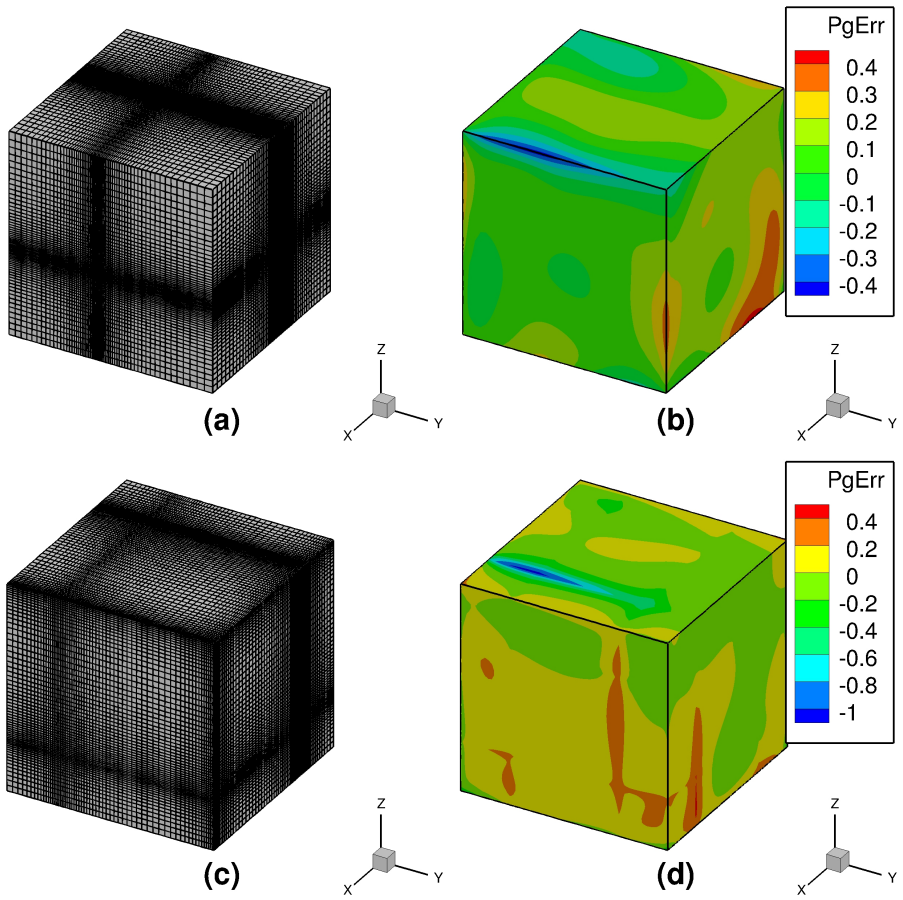

Figure 3.6: Effect of using smooth and discontinuous stretching factors: (a) mesh and (b) errors in pressure, using grid with smooth stretching; (a) mesh and (b) errors in pressure, using grid with discontinuous stretching. (Note: Grids contain 64×64 interior cells)

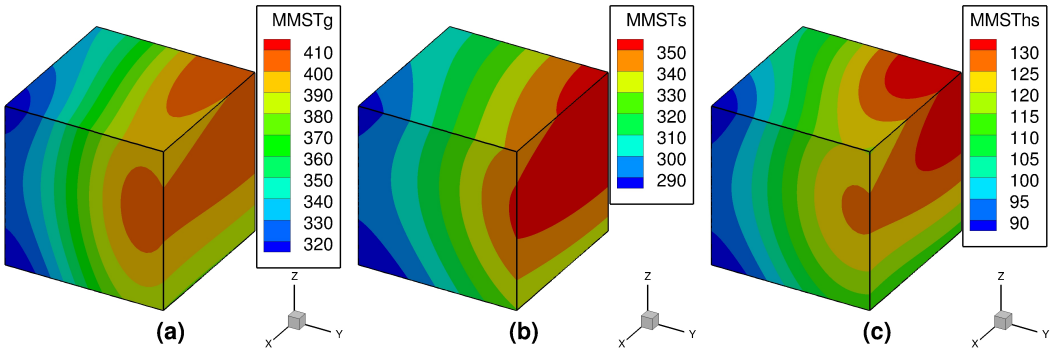

Figure 3.7: Manufactured solution for 3D, steady-state, two-phase flow verification: (a) gas temperature, (b) solid temperature, (c) solid granular energy.

equations described using Eqs. 3.1-3.9 are used during the numerical calculations in this problem. The order of accuracy is confirmed to be second order for this case using both L_2 and L_∞ norms as seen from Fig. 3.8(a) and (b), respectively.

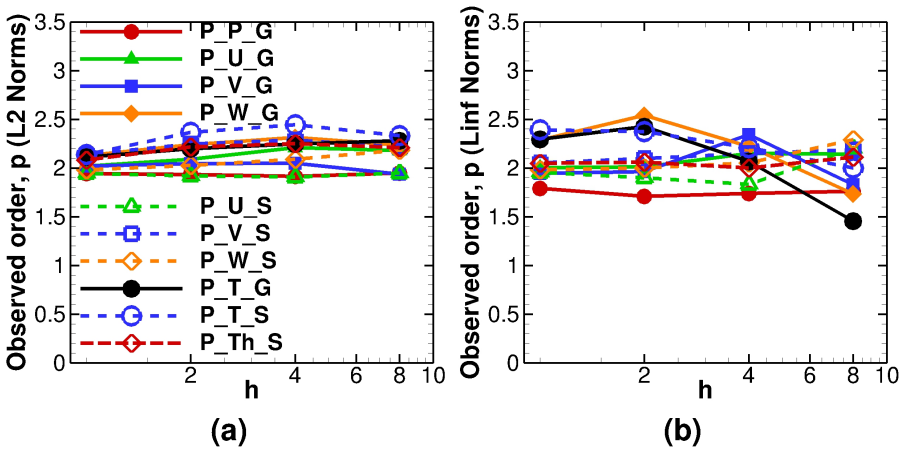

Figure 3.8: Observed order of accuracy for 3D, steady-state, two-phase flow verification: (a) L_2 norms, and (b) L_∞ norms of the discretization error.

3.4.5 Boundary condition verification

The three boundary conditions discussed here for divergence-free flows are the no-slip wall, the slip wall, and the pressure outflow. The selection of manufactured solutions for no-slip wall is discussed in greater detail than for the other two boundary conditions. (In this and the next section, since the boundary conditions and temporal schemes are assessed using single-phase divergence-free flows, variables u, v, w, and P are used for u_g, v_g, w_g, and P_g, respectively, to avoid unnecessary subscripts.)

Wall boundary conditions

Apart from the divergence-free flow condition, the implementation in the incompressible code, MFIX, prescribes that for a no-slip wall the velocity must be zero at the wall. For a slip-wall, to ensure tangency at the boundary, the implementation in the code sets the normal component of the velocity vector as zero while the tangential components of velocity are set equal to those in the ghost cell resulting in a zero gradient normal to the wall for these (tangential) components. There are no constraints on pressure at the wall boundaries. The mathematical formulation of these constraints and the boundaries on which these BCs are verified are presented in Table 3.1.

Table 3.1: Mathematical constraints on the no-slip wall, slip-wall, and pressure outflow boundary conditions in the incompressible solver.

	No-slip wall	Slip wall	Pressure outflow
Tested boundary	$S_{x0} \equiv x = 0$	$S_{x0} \equiv x = 0$	$S_{y1} \equiv y = 1$
Constraints	$\nabla \cdot \vec{V} = 0$	$\nabla \cdot \vec{V} = 0$	$\nabla \cdot \vec{V} = 0$
	$\vec{V} = 0$, at S_{x0}	$\vec{V} \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla u \cdot \nabla S_{y1} = 0$, at S_{y1}
		$\nabla v \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla v \cdot \nabla S_{y1} = 0$, at S_{y1}
		$\nabla w \cdot \nabla S_{x0} = 0$, at S_{x0}	$\nabla w \cdot \nabla S_{y1} = 0$, at S_{y1}
			$\nabla p \cdot \nabla S_{y1} = 0$, at S_{y1}

Consider the no-slip wall boundary condition verification on the 2D stretched Cartesian mesh where the tested boundary is selected as, $S_{x0}(x, y) \equiv x = 0$. The manufactured solution satisfying the boundary constraints and divergence-free constraint can be derived as follows. Let u and v be the incompressible velocity manufactured solutions that satisfy the no-slip wall BC at S_{x0} such that

$$u = u_1 x, \quad \text{and } v = v_1 x. \tag{3.20}$$

where u_1, v_1 are functions in x and y. Then in order to satisfy the divergence-free condition,

$$\frac{\partial u_1 x}{\partial x} + \frac{\partial v_1 x}{\partial y} = 0 \Rightarrow \frac{\partial v_1}{\partial y} = -\frac{\partial u_1}{\partial x} - \frac{u_1}{x}. \tag{3.21}$$

Selecting $u_1 = u_0 x \left(1 + \sin\left(\pi(x+y)^2\right)\right)$ where u_0 is a constant, the function v_1 can be found as

$$v_1 = \int \left(-\frac{\partial u_1}{\partial x} - \frac{u_1}{x}\right) dy + f_v(x). \tag{3.22}$$

where $f_v(x)$ is the integration constant selected as $5u_0$ here to ensure a dominantly upward flow. The gauge pressure manufactured solution can be a general function. Finally, the set of manufactured solutions for this case is given as

$$\begin{aligned} u &= u_0 x^2 \left(1 + \sin\left(\pi(x+y)^2\right)\right), \\ v &= u_0 x \left(5 - 3y + \frac{x}{2}\cos\left(2\pi\left(x+y\right)\right) + \frac{1}{2\pi}\sin\left(2\pi\left(x+y\right)\right)\right), \\ P &= P_0 x^2 \left(1 + \sin\left(\pi\left(x+y\right)\right)\right). \end{aligned} \tag{3.23}$$

For $u_0 = 10$ m/s, and $P_0 = 100$ Pa, the resulting manufactured solutions for x-velocity, y-velocity, and pressure variables are shown in Fig. 3.9.

It is evident that the procedure just described for verification of incompressible boundary conditions is cumbersome, involves trial-and-error in selection of the general functions and

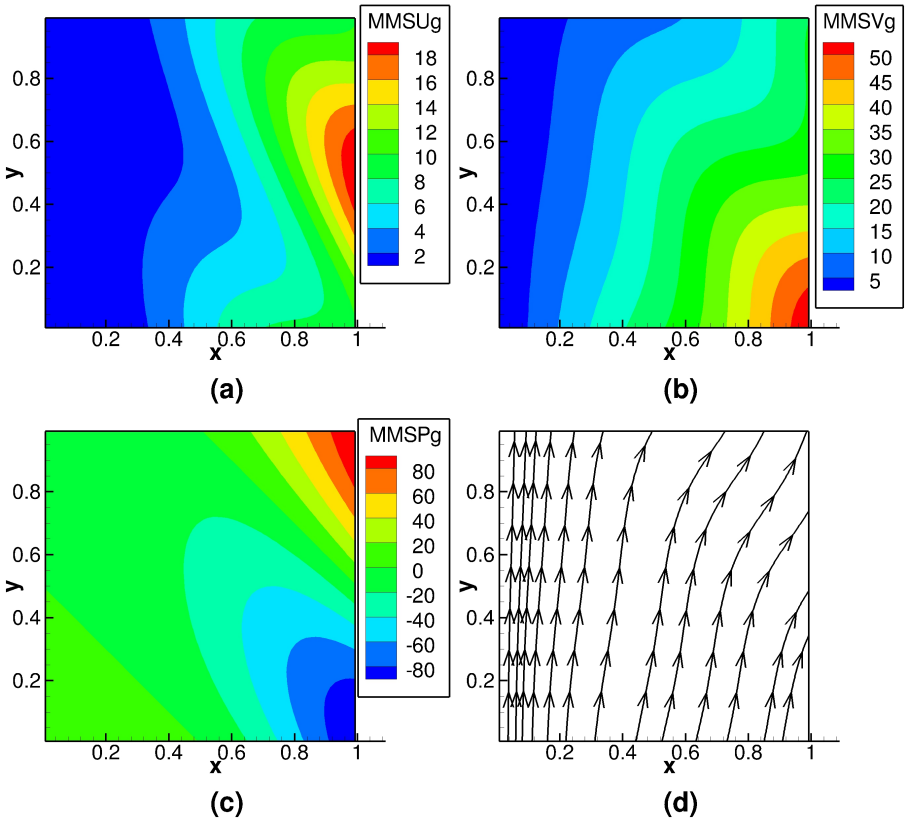

Figure 3.9: No-slip wall boundary condition manufactured solution on 2D stretched Cartesian mesh with S_{x0} as the no-slip boundary: (a) x-velocity, (b) y-velocity, (c) pressure, and (d) velocity streamlines.

various constants to ensure reasonable flow variation in the domain, and is not easily extendible to 3D grids. For verification on 3D grids, a novel, more general method is presented next that uses the zero curl property of divergence-free velocity fields.

Assume that the velocity vector for the manufactured solution has the form, $\vec{V} = \vec{V}_1 S$, where $\vec{V}_1$ is a general vector function to be determined, and $S \equiv S(x, y, z) = 0$ is a general surface. This manufactured solution for velocity field, $\vec{V}$, satisfies the no-slip wall conditions at S. Requiring $\vec{V}$ to be divergence-free and using the mathematical identity that the divergence of a curl is zero, we select

$$\vec{V} = \vec{V}_1 S = \vec{\nabla} \times \vec{G} \Rightarrow \vec{V}_1 = \frac{\vec{\nabla} \times \vec{G}}{S} \tag{3.24}$$

where $\vec{G}$ is a general 3D vector field. Since the manufactured solution must be well defined over the domain, and S goes to zero at the corresponding boundary, S must be a multiplicative factor in $\vec{\nabla} \times \vec{G}$. Substituting $\vec{G} = S^2 \vec{H}$ in Eq. 3.24, where $\vec{H}$ is another general vector field, we get

$$\vec{V} = S^2 \vec{\nabla} \times \vec{H} + 2S \nabla S \times \vec{H}. \tag{3.25}$$

Here, $\vec{H}$ is selected as the general sinusoidal vector field $\{u, v, w\}^T$ described using the functions of Eq. 3.12 with the constants provided for incompressible flow in Table 3.3. The manufactured solution in Eq. 3.25 meets all the constraints for no-slip wall as presented in Table 3.1.

Following a similar method, manufactured solution is derived for the slip wall verification, the derivation of which is omitted in this discussion for for brevity. Note from Table 3.1 that the constraints on the tangential components of velocity at the slip-wall require that the gradients of v and w (i.e., ∇v and ∇w) in the direction normal to the surface (i.e., ∇S_{x0}) are set as zero. Satisfying these constraints results in a velocity vector, $\vec{V}$, that has a factor of S_{x0}^2 in the variable part. Finally, a possible manufactured solutions for slip-wall is given in Eq. 3.26.

$$\vec{V} = \vec{V}_0 + S_{x0}^3 \vec{\nabla} \times \vec{H} + 3S_{x0}^2 \nabla S_{x0} \times \vec{H} \tag{3.26}$$

where, $\vec{V}_0 = \{0, v_0, w_0\}^T$ consists of non-zero scalar constants for v_0 and w_0.

For wall boundary conditions, convergence issues were encountered for the selected manufactured solutions. This issue is likely due to the use of a source-term driven flow instead of a boundary condition driven flow and the fact that a pressure-projection algorithm is being used. Therefore, currently we have verified the no-slip wall and the slip wall boundary condition for the most complex grid type (3D stretched Cartesian mesh) for momentum equations only by specifying the pressure solution instead of solving for it which is possible in the algorithm implementation within this code. The no-slip wall and slip wall boundary conditions are verified to be second order accurate for the momentum equations as shown in Fig. 3.10(a)(b).

Pressure outflow boundary condition

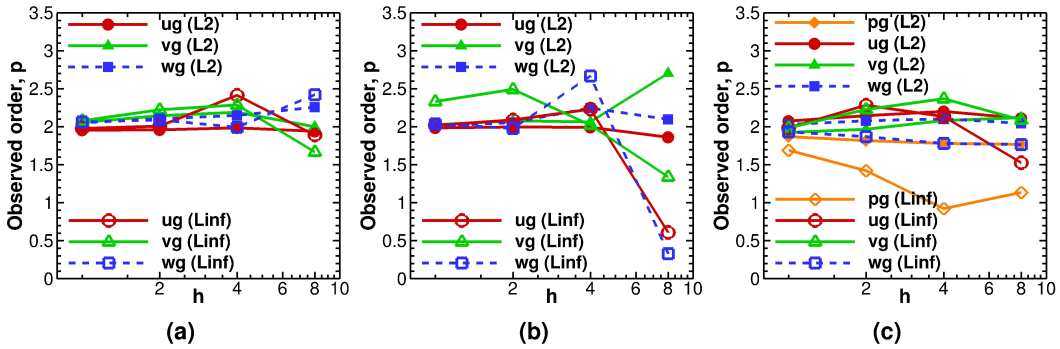

Figure 3.10: Observed order of accuracy for 3D incompressible boundary condition verification: (a) no-slip wall, (b) slip-wall, and (c) pressure outflows.

For the pressure outflow BC, pressure and all three velocity components are required to have zero gradients normal to the outflow boundary. The constraints shown in Table 3.1 suggest that the velocity manufactured solutions for pressure outflow can be similar to that shown in Eq. 3.26 for the slip-wall. The pressure manufactured solution follows the basic derivation for functions requiring zero gradient normal to the boundary using Eq. 3.14 with $m = 2$ and $S_{y1} \equiv y = 1$ as the tested boundary. A possible manufactured solution for verification of the pressure outflow can be written as:

$$\begin{aligned} P &= P_0 + S_{y1}^2 P_1, \\ \vec{V} &= \vec{V}_0 + S_{y1}^3 \vec{\nabla} \times \vec{H} + 3 S_{y1}^2 \nabla S_{y1} \times \vec{H} \end{aligned} \tag{3.27}$$

where, $\vec{V}_0 = \{u_0, v_0, w_0\}^T$ consists of non-zero scalar constants. The pressure outflow BC is verified to be second order accurate for momentum and pressure solutions as shown in Fig. 3.10(c) where no convergence issues were encountered since the problem has a proper outflow boundary.

3.4.6 Temporal order verification

Code verification for unsteady flows requires a combined order of accuracy test for both spatial and temporal terms as both the grid and the time-step are refined simultaneously. There are different methods in literature to perform temporal order verification [17]. The method used in the current discussion follows the description in Refs. [21, 36].

Neglecting the higher order terms, the discretization error for a scheme with spatial and temporal terms, can be written as

$$\varepsilon_{h_x}^{h_t} = g_x h_x^{\hat{p}} + g_t h_t^{\hat{q}} \tag{3.28}$$

where $\hat{p}$ and $\hat{q}$ are the observed orders of accuracy in space and time, respectively, g_x and g_t are the coefficients of spatial and temporal terms, respectively, and h_x and h_t are normalized spatial and temporal discretizations, respectively. The following steps are employed to determine the temporal order of accuracy of the code.

1. A constant time-step is selected making the temporal discretization error term in Eq. 3.28 a constant with respect to grid refinement. Discretization error is found on three systematically refined mesh levels to determine g_x and $\hat{p}$ using Eq. 3.29 and Eq. 3.30.

$$\hat{p} = \frac{ln\left(\frac{\left\|\varepsilon^{h_t}_{r_x^2 h_x}\right\| - \left\|\varepsilon^{h_t}_{r_x h_x}\right\|}{\left\|\varepsilon^{h_t}_{r_x h_x}\right\| - \left\|\varepsilon^{h_t}_{h_x}\right\|}\right)}{ln\left(r_x\right)} \tag{3.29}$$

$$g_x = \frac{\left\|\varepsilon^{h_t}_{r_x h_x}\right\| - \left\|\varepsilon^{h_t}_{h_x}\right\|}{h_x^{\hat{p}}\left(r_x^{\hat{p}} - 1\right)} \tag{3.30}$$

   where, r_x is the refinement factor for spatial refinement.

2. Similarly, a constant grid-size is selected and discretization error is found for three different time-step values to determine g_t and $\hat{q}$ in the temporal part of the discretization error expression with temporal refinement factor of r_t.

3. To ensure that the temporal and spatial errors are of similar orders, h_x and h_t are selected such that $g_x h_x^{\hat{p}} \approx g_t h_t^{\hat{q}}$. This selection needs trial and error, taking cost of computation and numerical stability into consideration.

4. For the selection of spatial and temporal discretization sizes and known formal orders, multiple cases are run to determine the temporal observed order of accuracy for the combined spatial-temporal refinements.

Unsteady equations are solved for multiple grid-levels and time steps using time dependent manufactured solutions as given in Eq. 3.31 for 3D, single-phase flows to obtain observed spatial and temporal accuracy orders.

$$\begin{aligned} u &= u_0 \sin\left(2\pi\left(x+y+z+t\right)\right)^2 \\ v &= u_0 \cos\left(2\pi\left(x+y+z+t\right)\right)^2 \\ w &= w_0 \\ p &= p_0 \cos\left(2\pi\left(x+y+z+t\right)\right) \end{aligned} \tag{3.31}$$

These relatively simple unsteady manufactured solutions are used instead of curl-based functions since unsteady calculations can be very expensive, especially for levels with small cell

sizes and small time-steps. The variable time-stepping algorithm available in MFIX is turned off to ensure that the selected time step is being used for these calculations.

The first order (Euler implicit) time-stepping method in MFIX is verified to be first order accurate in time and second order accurate in space as shown in Figs. 3.11(a) and (b), respectively. However, the second order time-stepping method failed the order of accuracy test due to a known simplification for run stability in the second step of the 2-step implicit Runge-Kutta method.

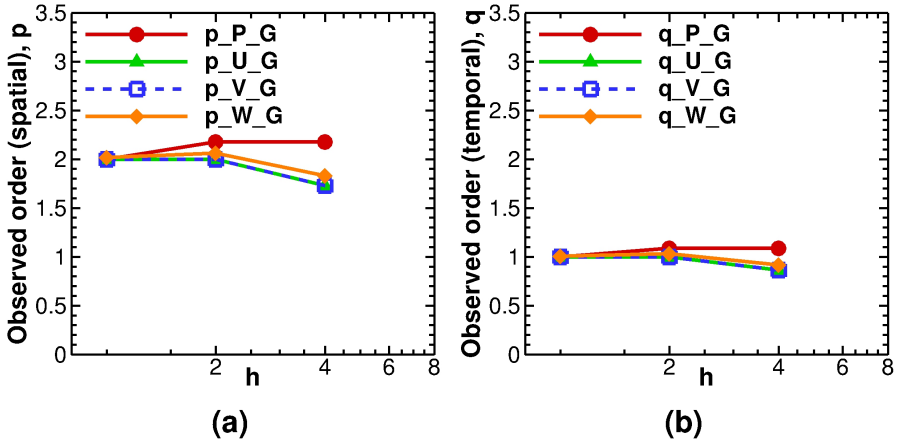

Figure 3.11: Combined spatial-temporal order verification for Euler-implicit time-stepping method. (a) Spatial scheme, and (b) temporal scheme observed orders of accuracy using L_2 norms of discretization error when Euler implicit (formally first order) time-stepping and central (formally second order) discretization of spatial terms are employed in a combined mesh and time-step refinement study.

3.5 Conclusions

A code verification study employing the method of manufactured solutions was presented for the governing equations of the two-fluid model used in multiphase flows. Different features of the code were verified such as the discretized terms of momentum, pressure-correction, and energy equations, for 2D and 3D, steady/unsteady, single-phase and two-phase incompressible flows. A newly-developed curl-based method to derive manufactured solutions was introduced for divergence free flows during the verification of baseline governing equations as well as during boundary condition verification. No-slip and free-slip wall boundary conditions were verified to be second order accurate for momentum equations while the pressure outflow boundary condition was found to be second order accurate for momentum as well as pressure-correction equations on 3D grids using single-phase flow equations.

Simplifications were made by assuming incompressible flow, constant viscosity, constant volume fraction, and by neglecting the algebraic constitutive relations that describe the essential interactions between phases in a multiphase flow simulation. Even with these simplifications, many useful lessons were learned from this code verification study about the implementation of discrete governing equations in this code. The multiphase code, MFIX, was found to be second order accurate after addressing certain issues. Some issues were uncovered in the implementation of Superbee scheme at the boundaries, in the evaluation of cross-terms of the strain-tensor during steady-state simulations, and in the second order time-stepping method. It was observed that the common practice of setting the ghost cell size equal to the first interior cell size leads to some reduction in solution accuracy without completely failing the order test.

Code verification of multiphase flows is challenging, especially in cases where the interface between the phases is resolved using discrete elements (such as in Lagrangian-Eulerian methods). Though current study addresses only continuum governing equations for code verification and is similar to single-phase flow code verification in many ways, the emphasis is put upon using rigorous, mathematical methods such as order of accuracy testing for code verification. The verification of algebraic constitutive models based upon unit-testing in conjunction with the MMS-based verification of discretized terms of the governing equation presented here, along with other practices such as model validation and error quantification are ultimately necessary to achieve greater confidence in multiphase flow simulations.

Acknowledgements

The authors would like to thank Tingwen Li and Jordan Musser of NETL, Morgantown, WV, and Aytekin Gel of ALPEMI Consulting, LLC., Phoenix, AZ for numerous discussions over the period of this study. The MFIX verification study was supported by the National Energy Technology Laboratory (NETL) through URS Corp. under contract T034:4000.3.671.238.003.413 with Mike Bergen as the URS subcontract technical representative. The authors also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper. URL: http://www.arc.vt.edu

Bibliography

[1] Mathematica, version 9.0, 2012. URL `http://www.wolfram.com/mathematica/`.

[2] MFIX - Multiphase Flow with Interphase eXchanges. URL `https://mfix.netl.doe.gov/`.

[3] J.W. Banks, T. Aslam, and W.J. Rider. On sub-linear convergence for linearly degenerate waves in capturing schemes. *Journal of Computational Physics*, 227(14): 69857002, Jul 2008. ISSN 0021-9991. doi: 10.1016/j.jcp.2008.04.002. URL `http://dx.doi.org/10.1016/j.jcp.2008.04.002`.

[4] S. Benyahia, M. Syamlal, and T.J. O'Brien. Summary of MFIX equations 2012-1, January 2012. URL `https://mfix.netl.doe.gov/documentation/Theory.pdf`. Accessed: May 28, 2014.

[5] Ryan B. Bond, Curtis C. Ober, Patrick M. Knupp, and Steven W. Bova. Manufactured solution for computational fluid dynamics boundary condition verification. *AIAA Journal*, 45(9):2224–2236, 2007. ISSN 0001-1452. doi: 10.2514/1.28099. URL `http://arc.aiaa.org/doi/abs/10.2514/1.28099`.

[6] P.T. Brady, M. Herrmann, and J.M. Lopez. Code verification for finite volume multiphase scalar equations using the method of manufactured solutions. *Journal of Computational Physics*, 231(7):29242944, Apr 2012. ISSN 0021-9991. doi: 10.1016/j.jcp.2011.12.040. URL `http://dx.doi.org/10.1016/j.jcp.2011.12.040`.

[7] Aniruddha Choudhary, Christopher J Roy, Edward A. Luke, and Subrahmanya P. Veluri. Issues in verifying boundary conditions for 3D unstructured CFD codes. American Institute of Aeronautics and Astronautics, 2011. doi: doi:10.2514/6.2011-386810.2514/6.2011-3868. URL `http://dx.doi.org/10.2514/6.2011-3868`. doi:10.2514/6.2011-3868.

[8] R.K. Crockett, P. Colella, and D.T. Graves. A Cartesian grid embedded boundary method for solving the Poisson and heat equations with discontinuous coefficients in three dimensions. *Journal of Computational Physics*, 230(7):24512469, Apr 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.12.017. URL `http://dx.doi.org/10.1016/j.jcp.2010.12.017`.

[9] Donald A. Drew and Stephen L. Passman. *Theory of Multicomponent Fluids*. Springer New York, 1999. ISBN http://id.crossref.org/isbn/978-0-387-22637-8. doi: 10.1007/b97678. URL `http://dx.doi.org/10.1007/b97678`.

[10] L. Ea, M. Hoekstra, A. Hay, and D. Pelletier. A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow. *International Journal of Computational Fluid Dynamics*, 21(3-4):175188, Mar 2007. ISSN 1029-0257. doi: 10.1080/10618560701553436. URL `http://dx.doi.org/10.1080/10618560701553436`.

[11] L. Ea, M. Hoekstra, and G. Vaz. Manufactured solutions for steady-flow Reynolds-averaged NavierStokes solvers. *International Journal of Computational Fluid Dynamics*, 26(5):313332, Jun 2012. ISSN 1029-0257. doi: 10.1080/10618562.2012.717617. URL `http://dx.doi.org/10.1080/10618562.2012.717617`.

[12] D. Folkner, A. Katz, and V. Sankaran. Design and verification methodology of boundary conditions for finite volume schemes. *Computers & Fluids*, 96:264275, Jun 2014. ISSN 0045-7930. doi: 10.1016/j.compfluid.2014.03.028. URL `http://dx.doi.org/10.1016/j.compfluid.2014.03.028`.

[13] Rahul Garg, Janine Galvin, Tingwen Li, and Sreekanth Pannala. Open-source MFIX-DEM software for gassolids flows: Part Iverification studies. *Powder Technology*, 220(0):122 – 137, 2012. ISSN 0032-5910. doi: http://dx.doi.org/10.1016/j.powtec.2011.09.019. URL `http://www.sciencedirect.com/science/article/pii/S003259101100502X`. Selected Papers from the 2010 {NETL} Multiphase Flow Workshop.

[14] D. Gidaspow. *Multiphase Flow and Fluidization: Continuum and Kinetic Theory Descriptions*. Academic Press, 1994. ISBN 9780122824708. URL `http://store.elsevier.com/product.jsp?isbn=9780122824708`.

[15] John R. Grace and Fariborz Taghipour. Verification and validation of CFD models and dynamic similarity for fluidized beds. *Powder Technology*, 139(2):99110, Jan 2004. ISSN 0032-5910. doi: 10.1016/j.powtec.2003.10.006. URL `http://dx.doi.org/10.1016/j.powtec.2003.10.006`.

[16] Shelley Hebert and Edward Luke. Honey, I shrunk the grids! a new approach to CFD verification. American Institute of Aeronautics and Astronautics, Jan 2005. ISBN http://id.crossref.org/isbn/978-1-62410-064-2. doi: 10.2514/6.2005-685. URL `http://dx.doi.org/10.2514/6.2005-685`.

[17] James Kamm, William Rider, and Jerry Brock. Combined space and time convergence analysis of a compressible flow algorithm. American Institute of Aeronautics and Astronautics, Jun 2003. ISBN http://id.crossref.org/isbn/978-1-62410-086-4. doi: 10.2514/6.2003-4241. URL `http://dx.doi.org/10.2514/6.2003-4241`.

[18] C. Kleinstreuer. *Two-Phase Flow: Theory and Applications*. CRC Press, 2003. ISBN 9781591690009.

[19] Patrick Knupp and Kambiz Salari. *Verification of Computer Codes in Computational Science and Engineering*. Chapman & Hall/CRC, 2003.

[20] C. K. K. Lun, S. B. Savage, D. J. Jeffrey, and N. Chepurniy. Kinetic theories for granular flow: inelastic particles in couette flow and slightly inelastic particles in a general flowfield. *Journal of Fluid Mechanics*, 140:223–256, 3 1984. ISSN 1469-7645. doi: 10.1017/S0022112084000586. URL `http://journals.cambridge.org/article_S0022112084000586`.

[21] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press Cambridge, 2010.

[22] P. J. Oliveira and R. I. Issa. On the numerical treatment of interphase forces in two-phase flow. *Numerical Methods in Multiphase Flows*, ASME FED, 185:131–140, 1994.

[23] S.V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Series on Computational Methods in Mechanics and Thermal Science. CRC Press, New York, USA, 1980. ISBN 0891165223, 978-0891165224.

[24] Dominique Pelletier and Patrick J. Roache. *Verification and Validation of Computational Heat Transfer*, page 417442. John Wiley & Sons, Inc., Dec 2000. ISBN http://id.crossref.org/isbn/9780471348788. doi: 10.1002/9780470172599.ch13. URL `http://dx.doi.org/10.1002/9780470172599.ch13`.

[25] P. J. Roache and S. Steinberg. Symbolic manipulation and computational fluid dynamics. *AIAA Journal*, 22(10):1390–1394, October 1984. ISSN 0001-1452. doi: 10.2514/3.8794. URL `http://dx.doi.org/10.2514/3.8794`.

[26] P. J. Roache, P. Knupp, S. Steinberg, and R. L. Blaine. Experience with benchmark test cases for groundwater flow, in: Benchmark test cases for computational fluid dynamics. *ASME FED H00598-1990*, 93:49–56, 1990.

[27] Patrick J. Roache. *Fundamentals of Verification and Validation*. Hermosa Publishers, 2009. ISBN 0913478121, 9780913478127.

[28] C. J. Roy, C. C. Nelson, T. M. Smith, and C. C. Ober. Verification of Euler/NavierStokes codes using the method of manufactured solutions. *International Journal for Numerical Methods in Fluids*, 44(6):599620, Feb 2004. ISSN 1097-0363. doi: 10.1002/fld.660. URL `http://dx.doi.org/10.1002/fld.660`.

[29] Christopher J. Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205(1):131 – 156, 2005. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2004.10.036. URL `http://www.sciencedirect.com/science/article/pii/S0021999104004619`.

[30] Lee Shunn, Frank Ham, and Parviz Moin. Verification of variable-density flow solvers using manufactured solutions. *Journal of Computational Physics*, 231(9):38013827, May 2012. ISSN 0021-9991. doi: 10.1016/j.jcp.2012.01.027. URL `http://dx.doi.org/10.1016/j.jcp.2012.01.027`.

[31] M. Syamlal. MFIX documentation: Numerical technique, Tech. Rep. DOE/MC31346-5824, NTIS/DE98002029. Web Page June 6, 2014, National Energy Technology Laboratory, 1998. URL `https://mfix.netl.doe.gov/documentation/numerics.pdf`.

[32] M. Syamlal, W. Rogers, and T.J. O'Brien. MFIX documentation: Theory guide, Tech. Rep. DOE/METC-95/1013, NTIS/DE95000031. Web Page May 28, 2014, National Energy Technology Laboratory, 1993. URL `https://mfix.netl.doe.gov/documentation/Theory.pdf`.

[33] S. Tenneti, R. Garg, and S. Subramaniam. Drag law for monodisperse gassolid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres. *International Journal of Multiphase Flow*, 37(9):10721092, Nov 2011. ISSN 0301-9322. doi: 10.1016/j.ijmultiphaseflow.2011.05.010. URL `http://dx.doi.org/10.1016/j.ijmultiphaseflow.2011.05.010`.

[34] J. L. Thomas, B. Diskin, and C. L. Rumsey. Towards verification of unstructured-grid solvers. *AIAA Journal*, 46(12):3070–3079, December 2008. ISSN 0001-1452. doi: 10.2514/1.36655. URL `http://arc.aiaa.org/doi/abs/10.2514/1.36655`.

[35] Joo Marcelo Vedovoto, Aristeu da Silveira Neto, Arnaud Mura, and Luis Fernando Figueira da Silva. Application of the method of manufactured solutions to the verification of a pressure-based finite-volume numerical scheme. *Computers & Fluids*, 51(1):8599, Dec 2011. ISSN 0045-7930. doi: 10.1016/j.compfluid.2011.07.014. URL `http://dx.doi.org/10.1016/j.compfluid.2011.07.014`.

[36] Subrahmanya P. Veluri, Christopher J. Roy, and Edward A. Luke. Comprehensive code verification techniques for finite volume CFD codes. *Computers & Fluids*, 70(0):59–72, 2012. ISSN 0045-7930. doi: http://dx.doi.org/10.1016/j.compfluid.2012.04.028. URL `http://www.sciencedirect.com/science/article/pii/S0045793012001697`.

Appendix 3.A Functions and constants in manufactured solutions

Note: The amplitude constants employ SI units while the frequency constants are dimensionless. For incompressible manufactured solutions, pressure (p_g) is the gauge pressure.

Table 3.2: Sinusoidal functions and frequency constants used in the compressible and incompressible manufactured solutions.

Variable, ϕ	$f_{\phi x}$	$f_{\phi y}$	$f_{\phi z}$	$f_{\phi xy}$	$f_{\phi yz}$	$f_{\phi zx}$
u_g	sin	cos	cos	cos	sin	cos
v_g	sin	cos	cos	cos	sin	cos
w_g	cos	sin	cos	sin	sin	cos
P_g	cos	cos	sin	cos	sin	cos
T_g	cos	cos	sin	cos	sin	cos
T_s	cos	cos	sin	cos	sin	cos
Θ_s	cos	cos	sin	cos	sin	cos
Variable, ϕ	$a_{\phi x}$	$a_{\phi y}$	$a_{\phi z}$	$a_{\phi xy}$	$a_{\phi yz}$	$a_{\phi zx}$
u_g	0.5	0.85	0.4	0.6	0.8	0.9
v_g	0.8	0.8	0.5	0.9	0.4	0.6
w_g	0.85	0.9	0.5	0.4	0.8	0.75
P_g	0.4	0.45	0.85	0.75	0.7	0.8
T_g	0.75	1.25	0.8	0.65	0.5	0.6
T_s	0.5	0.9	0.8	0.5	0.65	0.4
Θ_s	0.8	1.25	0.7	0.5	0.6	0.7

Table 3.3: Amplitude constants used in the subsonic, incompressible manufactured solutions.

Variable, ϕ	ϕ_0	ϕ_x	ϕ_y	ϕ_z	ϕ_{xy}	ϕ_{yz}	ϕ_{zx}
u_g	7	3	-4	-3	2	1.5	2
v_g	9	-5	4	5	-3	2.5	3.5
w_g	8	-4	3.5	4.2	-2.2	2.1	2.5
P_g	100	20	-50	20	-25	-10	10
T_g	350	10	-30	20	-12	10	8
T_s	300	15	-20	15	-10	12	10
Θ_s	100	5	-10	12	-8	10	7

Chapter 4

Structured Mesh Adaptation Using Truncation Error Equidistribution for 1D and 2D CFD Problems

Aniruddha Choudhary & Christopher J. Roy
Aerospace and Ocean Engineering Department, Virginia Tech, Blacksburg, VA 24061, USA

Abstract

In a CFD simulation, truncation error (TE) is the difference between the continuous governing equation and its discrete approximation. Since TE can be shown to be the local source term for the discretization error, TE is proposed as the criterion for determining which regions of the computational mesh should be refined/coarsened. For mesh modification, an error equidistribution strategy to perform r-refinement (i.e., mesh node relocation) is employed. This technique is applied to 1D and 2D inviscid flow problems where the exact (i.e., analytic) solution is available. For mesh adaptation based upon TE, about an order of magnitude improvement in discretization error levels is observed when compared with the uniform mesh. For a simpler problem (1D Burgers equation), we find TE-based adaptation to give better results compared to feature-based adaptation (e.g., gradient, curvature). It is shown that this procedure can be extended to problems without exact solutions by estimating instead of evaluating TE.

Keywords Mesh adaptation; Truncation error; Residual; r-Refinement; Equidistribution

4.1 Introduction

Discretization error (DE) occurs in every Computational Fluid Dynamics (CFD) solution and is one of the main contributors to the overall uncertainty in a CFD prediction. DE is defined as the difference between the exact solution to the discrete equations and the exact solution to the continuous governing equations. For a given numerical scheme, discretization error can be reduced by solving the discrete equations on a mesh with improved resolution and/or improved quality cells. Performing uniform mesh refinement over the entire domain to improve mesh resolution is generally inefficient and computationally expensive since it can result in highly refined cells/elements in regions of the domain where refinement is not required. Mesh adaptation is targeted local refinement for reducing the solution error and is a much better strategy than uniform refinement. Mesh adaptation is a widely studied field of research in the scientific computing community. For mesh adaptation approaches in CFD, there are several extensive reviews available (e.g., see [3], [17] and [14]).

Performing mesh adaptation involves two distinct aspects: (1) a criterion for driving the adaptation, and (2) a mechanism to perform mesh modification. A variety of measures or adaptation monitors (also referred to as adaptation criteria, adaptation parameters, or weight functions) have been used in the past to determine which regions should be refined or coarsened. Once an adaptation criterion has been identified, the local mesh can be modified by moving nodes from one region to another (r-refinement), selectively refining/coarsening cells by adding/deleting nodes (h-refinement), or increasing/decreasing the formal order of accuracy of the method (p-refinement). For general unstructured grid methods the h-refinement approach is the most popular one. For structured grid methods the r-refinement approach is most often used. p-refinement has not found widespread use for CFD problems but is more popular in the finite-element community [3].

Hawken et al. [14] rightly observed that in many cases the adaptation criteria are chosen "heuristically" with "little or no justification" for them. Adaptation based upon solution features such as solution gradients, solution curvature, or even physical flow features (e.g., shock waves, expansion fans, contact discontinuities, vortices, and boundary layers) are known to fail in presence of multiple solution features (see [11] for one such example). Adaptation based upon discretization error neglects the fact that DE can also be transported (i.e., convected, diffused, and propagated as acoustic waves) from other regions of the domain. Thus, it is entirely possible that the total DE may be large in a region where the local contributions to the DE are small. An example of the failure of mesh adaptation based on the total discretization error can be found in [13]. Adaptation based on recovery or reconstruction-based indicators (e.g., Zienkiewicz-Zhu patch recovery technique [28]) shows good results for linear elliptic problems in finite elements, but this improvement is not expected to carry over to hyperbolic or mixed-character problems. Laflin [16] and McRae [17] demonstrated adaptation based on solution interpolation error (i.e., error in higher-order accurate interpolation from a patch of neighboring cells) to be successful for a wide range of 2D problems on structured meshes. Both recovery-based methods and adjoint-based methods for mesh adaptation use

the solution residual weighted by the adjoint sensitivities as the adaption monitor, thus providing targeted mesh adaption for the chosen system response quantity (e.g., lift, drag, and moments). Venditti and Darmofal [26, 27] successfully applied the method in finite-volume form to inviscid and viscous flow over airfoils at various Mach numbers. While adjoint-based methods show promising results, the implementation of these methods can be mathematically complex, code intrusive, and computationally expensive since it requires the solution for both dual and primal problems.

The truncation error is defined as the difference between the discretized equations and the original continuous partial differential (or integral) equations. The error transport equations [23, 19] show that the discretization error is locally generated by the truncation error as well as transported from other regions of the domain. Since the truncation error serves as the local source of the discretization error in the domain, reducing the truncation error should result in a significant reduction in the discretization error. TE as an adaptation monitor is based upon systematic mathematical derivation and includes the effect of solution behavior, mesh resolution, and mesh quality in the adaptation process [8]. Borsboom [6] applied a similar approach of residual-based adaptation to a scalar, 1D problem with an exact solution mimicking Sod's [25] 1D shock tube solution. For various grid sizes, Borsboom obtained 10 to 1000 times reduction in DE in comparison with the uniform mesh since the presence of a solution discontinuity in the domain usually results in large improvement with adaptation. By our research group, preliminary results for simple 1D and 2D Burgers' equations and Euler problems were shown to give about an order of magnitude improvement in DE (over uniform grids) when compared to feature-based adaptation monitors [23, 7, 8, 9].

The objective of the current work is to develop a TE/residual-based mesh adaptation procedure and apply it to benchmark 1D and 2D Euler problems in CFD. The proposed TE-based mesh adaptation methodology is explained in detail in Section 4.2. A Fortran library, called Structured Adaption Module (SAM), is developed to perform equidistribution of the selected adaptation monitor on structured meshes using r-refinement techniques. The mathematical background and implementation of the adaptation process in SAM is briefly discussed. The work presented in this paper is restricted to r-refinement on structured grids only, although many of the lessons learned should be extendible to unstructured grids as well as to other refinement techniques (e.g., h- and p- refinement).

To test the proposed TE-based adaptation method, we select 1D and 2D Euler problems for which the exact (i.e., analytic) solutions are available. This allows us to evaluate (as opposed to estimate) the truncation error by substituting the exact solution into the discrete governing equations. The Euler problems considered include (1) quasi-1D nozzle flow, and (2) supersonic expansion fan. A simpler problem, 1D Burgers equation, is used to compare the TE-based adaptation monitor against feature-based adaptation monitors. The details of these application problems and the codes used for flow solution are presented in Section 4.3.

Results for the selected application problems are presented and discussed in Section 4.4. Improvement in errors due to mesh adaptation vary between different problems but in gen-

eral, an order of magnitude improvement in discretization error levels from the uniform mesh is achieved for TE-based adaptation. For practical applications, accurate estimation of the truncation error is important (see [22] for details on truncation error estimation). Since estimation of truncation error is a detailed topic, we present only a representative result for mesh adaptation when TE is estimated instead of directly evaluated. Finally, the work is concluded in Section 4.5 with a summary of the results, observations about the proposed method, various current challenges and ideas for extending this method to more complex problems.

4.2 Methodology

4.2.1 Adaptation monitor

Truncation error as adaptation monitor

The motivation for selecting the truncation error as the adaptation monitor stems from analyzing the Generalized Truncation Error Expression (GTEE) [23, 19] which relates the expression for the continuous governing equation, $L(\cdot)$, its discrete approximation, $L_h(\cdot)$, and the truncation error expression, $\tau_h(\cdot)$. Following the convention of [22], the GTEE can be expressed as shown in Eqn. 4.1.

$$L_h(I^h w) = I^h L(w) + \tau_h(w) \tag{4.1}$$

Here, w is some continuous function and I^h is an operator that restricts the continuous function w to a discrete domain. For example, the restriction of a continuous function corresponds to the cell average of the function on the discrete domain when a finite-volume formulation is employed and to nodal values when a finite-difference formulation is employed.

By substituting the exact solution to the discrete equation, u_h, into Eqn. 4.1, subtracting the continuous equation, $L(\tilde{u}) = 0$, where $\tilde{u}$ is the exact solution to the continuous equation, and employing the definition for discretization error as, $\varepsilon_h = u_h - I^h\tilde{u}$, we get the continuous error transport equation as shown in Eqn. 4.2. In Eqn. 4.2, I_h^q prolongs a discrete function to a q^{th} order polynomial. By substituting the exact solution to the continuous equation, $\tilde{u}$, into the GTEE, and subtracting the discrete equation, $L_h(u_h) = 0$, we get the discrete error transport equation as shown in Eqn. 4.3.

$$L(I_h^q \varepsilon_h) = -I_h^q \tau_h(u_h) \tag{4.2}$$

$$L_h(\varepsilon_h) = -\tau_h(I^h\tilde{u}) \tag{4.3}$$

Note that the derivation of both Eqs. 4.2 and 4.3 assume simple linearization for the continuous and discrete governing equations (e.g., $L(I^h u_h) - L(\tilde{u}) \approx L(I^h u_h - \tilde{u})$). More sophisticated linearization can also be used [20].

The continuous and discrete error transport equation show that the discretization error can be both locally generated by the truncation error and also transported from other regions of the domain. This latter error transport is also known as pollution error in the finite element community. Since the truncation error serves as the local source term in the discretization error transport equations, reducing the truncation error should result in a commensurate reduction in the discretization error, at least for smooth structured grids. This may not be true for unstructured meshes, where the adaptation monitor based on TE can be "noisy" and may prevent a practically useful identification of the region where the truncation error is being generated.

Evaluation/estimation of truncation error

By substituting the exact solution to the continuous differential equation, $\tilde{u}$, into Eqn. 4.1, we get TE in terms of the discrete residual as shown in Eqn. 4.4.

$$\tau_h(\tilde{u}) = L_h(I^h\tilde{u}) \tag{4.4}$$

Since exact solution is available for the problems considered in the current work, we use Eqn. 4.4 to directly evaluate the TE by operating the discrete equations on the exact solution. By substituting the exact solution to the discretized equation, u_h, into Eqn. 4.1, we get the TE in terms of the continuous residual as shown in Eqn. 4.5.

$$\tau_h(I_h^q u_h) = -I^h L(I_h^q u_h) \tag{4.5}$$

Eqs. 4.4 and 4.5 present the equivalency of residuals and truncation error (or its estimate thereof). The continuous residual of Eqn. 4.5 provides us with a method to estimate the TE by using a prolongation of the numerical solution, $I_h^q u_h$ to a continuous space. For example, adjoint methods [27] use an embedded grid approach and insert the coarse mesh numerical solution into the finer mesh discrete operator to estimate truncation error. For finite volume methods, where the solution is known as an average over the cell, solution reconstruction between different meshes should be performed in a finite-volume consistent manner. By this we mean that the restriction of the prolongation of a cell averaged finite volume solution should return that same value, i.e., $I^h I_h^q u_h = u_h$. One such reconstruction method is the k-exact method [5, 4], where "k" represents the order of the polynomial used in the reconstruction. For a representative case with adaptation based upon estimate TE, we use the continuous residual k-exact method that is implemented in the CFD code employed here for Euler solutions (Refer to [22] for implementation details).

4.2.2 Mesh relocation

Equidistribution statement

The principle of error equidistribution can be used as the underlying concept to develop strategies for mesh adaptation and has been around for many years [24, 15]. The objective of function equidistribution over a domain is to find partitions (or subdomains) such that the integral of that function over each partition is a constant value. When this is achieved, the function is considered *equidistributed* over the new discrete domain (i.e., mesh). Mathematically, the equidistribution statement can be formulated as follows (shown here for one dimension). For higher dimensions, the extension of the theory is straightforward, while the implementation is more involved. Consider $\phi(x)$ to be a positive continuous function over the 1D domain, $D : a \leq x \leq b$, then the goal of equidistribution over a discrete domain with N nodes is to find partitions, $\Pi : a = x_1 < x_2 \ldots < x_N = b$, of D such that

$$\int_{x_i}^{x_{i+1}} \phi(x)\,\mathrm{d}x = \frac{1}{N-1}\int_a^b \phi(x)\,\mathrm{d}x \tag{4.6}$$

where, $i = 1, 2 \ldots N-1$. Defining $\gamma(x) = \int_a^x \phi(x)\,\mathrm{d}x$, for an equidistributed mesh we have

$$\gamma(x_{i+1}) = \text{constant} = \frac{i}{N-1}\gamma(b) = c_i \tag{4.7}$$

Eqn. 4.7 indicates that the equidistribution problem is essentially a root-finding problem. Though for small 1D meshes, a root-finding technique (e.g., false position) can be used, we use an iterative approach since it provides easy extension to higher dimensions, can be coupled with existing explicit/implicit linear solvers, and allows for inclusion of stability measures (such as under-relaxation and partial convergence) for stability in the adaptation process.

For an iterative method, the elemental step in the equidistribution process is shown in Fig. 4.1 where the node is relocated to achieve equal area over two cells, $A_1 = A_2$ under the curve of the function, $W(x)$. If $\phi(x)$ is a positive, continuous function dependent only on

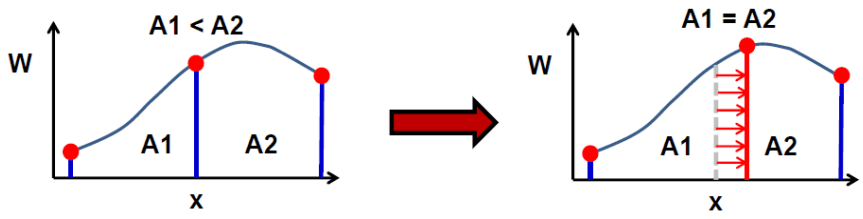

Figure 4.1: Elemental step for 1D equidistribution of a function, $W(x)$, using mesh node relocation.

the continuous coordinates (i.e., x, y, z) then achieving equidistribution once is sufficient. However, TE is a highly nonlinear function of discrete coordinates (i.e., mesh node locations) and varies as the nodes are relocated. Hence, the equidistribution process needs to be repeated multiple times when the adaptation monitor is based upon TE. Moreover, achieving complete equidistribution for each instance of the truncation error is counter-productive and results in large oscillations during the adaptation process. Instead, stability in the adaptation process can be achieved by performing partial equidistribution (less iterations than full convergence) for each instance of the truncation error. In conjunction with a linear/explicit iterative method, this results in movement of mesh nodes towards where truncation error is larger resulting in higher mesh resolution in such regions of the domain.

Adaptation process

The adaptation process used here is outlined as follows:

Step 1. Evaluate/estimate TE on the current mesh as averages over each cell.

Step 2. Create a positive, non-zero, smooth weight function at the nodes using the TE of Step 1.

Step 3. Check for equidistribution before proceeding to mesh relocation. Go to Step-8 if the weight function is equidistributed over the given mesh within a given tolerance.

Step 4. Relocate the mesh for one iteration of the explicit/linear solver.

Step 5. Interpolate the weight function of Step 2 on the relocated mesh of Step 4.

Step 6. Repeat Steps 4-5 for a few iterations.

Step 7. Interpolate the original solution onto the new mesh. Repeat Steps 1-5.

Step 8. End of mesh adaptation

Steps 4-5 are referred to as the inner iterations while Steps 2-7 are referred to as the outer iterations. The process of mesh adaptation for complicated adaptation monitors (TE/residuals, adjoint-based indicators) requires techniques such as performing partial inner iterations, under-relaxation for inner and/or outer iterations, and specifying a reasonable adaptive convergence criterion.

When a system of coupled conservation equations are present such as in Euler and Navier-Stokes problems, each equation has its own truncation error. One way to combine the truncation error from different equations is shown in Eqn. 4.8:

$$\tau_{c,i} = \sum_{n=1}^{neq} \frac{|\tau_{n,i}|}{|\max^0(\tau_{n,i})|} \tag{4.8}$$

where neq is the number of conservation equations in the system, $\tau_{n,i}$ is the truncation error for the n^{th} conservation equation at some mesh node location, i, (e.g., for 1D), $\tau_{c,i}$ is the combined truncation error, and $\max^0(\tau_{n,i})$ is the maximum value of truncation error from a given conservation equation over the entire domain on the mesh prior to any mesh adaptation. The scaling factor, $\max^0(\tau_{n,i})$, ensures that the truncation error from different equations are scaled equally relative to each other and the scaling constants do not vary with adaptation. More sophisticated methods for combining the truncation error from various equations are certainly possible but are not currently investigated.

The combined truncation error, $\tau_{c,i}$, is smoothened to promote generation of smooth meshes. Meshes with poor quality cells result in poor solutions as well as noisy truncation error that could cause oscillations during the mesh relocation process. To remove the high-frequency noise from the weight function we perform a few passes of a simple elliptic smoothening (low-pass filter) operation as shown in Eqn. 4.9 (for 1D).

$$W_i^{k+1} = \frac{1}{\eta + 2} \sum_{k=0}^{\text{kmax-1}} W_{i-1}^k + \eta W_i^k + W_{i+1}^k \tag{4.9}$$

where k is the index for smoothening passes, kmax is the maximum number of smoothening passes, W is the weight function set to τ_c for TE-based adaptation when $k = 0$, and η is some constant ($\eta = 4$ in the current work). The downside of using many smoothening passes (i.e., large kmax) is that the smoothening can destroy significant features of the original monitor. Better strategies in dealing with smoothening of function to remove the noise and to keep the underlying structure of the weight function are expected to give better results.

The determination of when to stop mesh adaptation, i.e., the stopping criterion, is based upon a measure of equidistribution achieved called the weight equidistribution error (WEE) which is defined as (for 1D):

$$\text{WEE}_i = \left| \frac{\sigma - W_i \delta x_i}{\sigma} \right| \qquad \text{where,} \qquad \sigma = \frac{\sum_D W_i \delta x_i}{N} \tag{4.10}$$

where, N is the number of cells, W_i is the weight function over the i-th cell, δx_i is the cell size of the i-th cell, and D represents the domain over which the summation is performed. A global measure of WEE such as average (L_1 norm) or maximum (L_∞ norm) is used to determine when weight equidistribution error reaches within 5% or lower for most cases.

After preparing the weight function, mesh relocation during the inner iterations is performed using an explicit iterative method based upon the center of mass (CoM) approach for mesh movement [12]. Other approaches in the literature include line-by-line or bilinear mesh relocation using the spring analogy where the weight function is used to determine a spring constant for linear springs connecting the nodes [2][18]. In our previous analysis [9], we found the center of mass approach to be more stable and more easily extendible to higher dimension than the spring-based approach. The spring-based approach often led to undamped dynamical motion of the mesh during adaptive iterations. For a 2D domain,

the relocation is performed on a uniformly spaced computational mesh. The relocated node positions in the computational space along with the transformation metrics of the previous step are used to obtain the (x, y) coordinates of the relocated mesh nodes in the physical space. During the mesh movement, it is necessary to interpolate the weight functions after each relocation step to ensure that the weight function remains fixed with respect to the physical mesh. This process is called weight function redistribution for which a simple bi-linear interpolation can be used. The mesh movement algorithm used in the current work closely follows the description in [17] and [16].

To perform the mesh adaptation in tandem with the numerical simulations, we use a newly developed library of Fortran procedures called Structured Adaptation Module (SAM). SAM currently employs the mesh relocation technique described in this section over 1D and 2D structured meshes and can be coupled with a flow solver code of interest. For general applications, mesh adaptation must be performed after a certain level of solution iterative convergence is reached and this is usually repeated multiple times. However, for current analysis of TE-based adaptation, we simply evaluate (or estimate) the truncation error using the flow solver code, perform mesh adaptation using SAM, and then re-evaluate (or re-estimate) truncation error using the flow solver code. While this process is computationally inefficient, it suffices to demonstrate the potential of the approach. This process is repeated multiple times until equidistribution is achieved. Finally, the flow solution is obtained on the adapted mesh and compared against the solution on the initial, un-adapted mesh. Currently, the adaptation process has not been optimized for computational efficiency since exact solution is available for all the problems considered and hence TE can be evaluated without solving for the numerical flow solution.

4.3 Application problems

This section discusses the governing equations for the application problems used in this paper to demonstrate the proposed use of TE as the adaptation monitor.

4.3.1 1D Burgers equation

We select the 1D Burgers equation for our first analysis because it is a simple problem with a single governing equation. Burgers equation resembles the Navier-Stokes equation in the sense that it has a non-linear convection term and a diffusion term. The chosen exact solution to Burgers equation is a smooth viscous shock, which can be modified if needed by changing the Reynolds number of the flow in order to study the effect of discontinuity sharpness. We use the 1D Burgers equation here to compare the TE-based adaptation process against gradient and curvature-based adaptation as well as to study the effect of mesh size and strength of the viscous shock.

The governing equation for the steady-state form of 1D Burgers' equation is given as

$$L(u) \equiv u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{4.11}$$

where ν is the viscosity coefficient. An exact solution for the 1D Burgers' equation is given as

$$\tilde{u} = -2\tanh\left(x\frac{Re}{2l_{ref}}\right) \tag{4.12}$$

where Re is the Reynolds number for the flow, defined here as $Re = u_{ref}l_{ref}/\nu$, where l_{ref} is the domain length and u_{ref} is the reference velocity. The exact solution is shown in Fig. 4.2(a). For all results presented for 1D Burgers equation, $l_{ref} = 8$ m and $u_{ref} = 2$ m/s. The 1D domain spans from $x = -4$ m to $x = 4$ m. The code for 1D Burgers equation employs a second order, central, finite-difference scheme which is solved using a tridiagonal matrix algorithm (Thomas algorithm). The exact solution for this problem is given in Fig. 4.2(a) from which it can be seen that there is a viscous shock at the center, $x = 0$ m, while the solution has negligible variation near the boundaries.

4.3.2 Quasi-1D nozzle flow

The flow through a quasi-1D nozzle employs the standard Euler equations. In a finite-volume formulation, the momentum source terms are modified to include the nozzle cross-section area effect as shown in Eqn. 4.13 and 4.14.

$$L(\vec{U}) \equiv A_i\frac{\partial \vec{U}_i}{\partial t} + \frac{\delta(\vec{F}A)_i}{\delta x_i} = \vec{S}_i \tag{4.13}$$

where, i represents a finite-volume cell, A represents the nozzle cross-section area, $\delta(\cdot)_i$ is the discrete operator to approximate the finite-volume quantity, and

$$\vec{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho e_T \end{pmatrix} \quad \vec{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u h_T \end{pmatrix} \quad \text{and, } \vec{S} = \begin{pmatrix} 0 \\ p\frac{dA}{dx} \\ 0 \end{pmatrix} \tag{4.14}$$

where, symbols have usual thermodynamic meanings.

The converging-diverging nozzle geometry as described by the area function given in Eqn. 4.15 extends between $x = [-2, 2]$ m, has a sinusoidal profile in the middle, and uniform cross-section parts at both ends.

$$A(x) = \begin{cases} 0.2 + 0.4\left(1 + \sin(\pi(x - 0.5))\right) & \text{if } |x| \leq 1, \\ 1 & \text{if } 1 < |x| \leq 2 \end{cases} \tag{4.15}$$

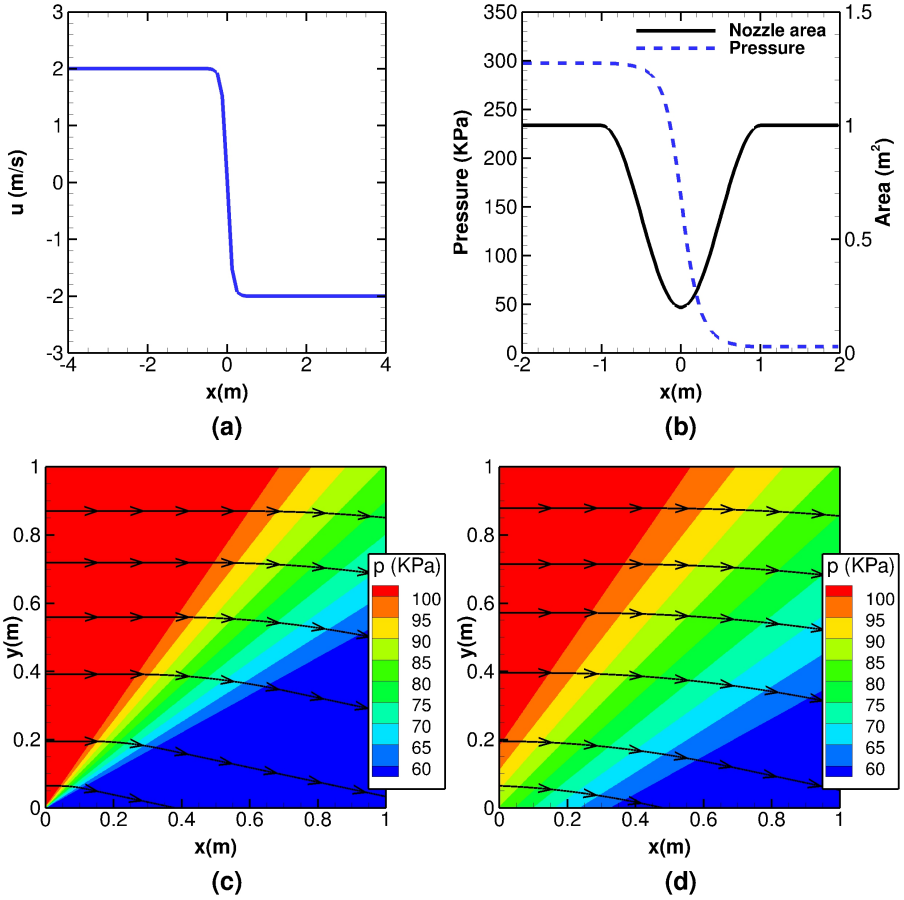

Figure 4.2: Exact solutions for 1D and 2D and 2D application problems: (a) 1D Burgers equation ($Re = 128$); (b) quasi-1D nozzle flow; supersonic expansion fan flow (c) singularity-dominated case, and (d) edge-dominated case.

It should be noted that this choice of area function leads to curvature discontinuities at $x = \pm 1$. The flow is set to be isentropic with a stagnation temperature of $T_0 = 600$ K and a stagnation pressure of $P_0 = 300$ KPa resulting in a subsonic to supersonic flow with a Mach number of 1 at the throat. The exact solution as known from the isentropic Mach-area relations and the nozzle profiles are shown in Fig. 4.2(b).

A second order, finite-volume, flow solver employing roe-flux scheme and upwind-biased MUSCL extrapolation is used for flow solution. Physical boundary conditions are implemented, i.e., isentropic inflow at $x = -2$ m and supersonic (extrapolation-based) outflow at $x = 2$ m.

4.3.3 Supersonic expansion fan

For 2D Euler flows, the governing equations employed is the standard Euler equations in their weak (i.e., integral) form as given in Eqn. 4.16.

$$\frac{\partial}{\partial t}\int_{\Omega} \vec{Q}\,\mathrm{d}\Omega + \oint_{\partial\Omega} \vec{F}\,\mathrm{d}s = \int_{\Omega} \vec{S}\,\mathrm{d}\Omega \tag{4.16}$$

where $\vec{Q}$ is the conserved variables vector, and $\vec{F}$ is the inviscid flux vector given as

$$\vec{Q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_T \end{pmatrix} \quad \text{and, } \vec{F} = \begin{pmatrix} \rho V_n \\ \rho u V_n + n_x p \\ \rho v V_n + n_y p \\ \rho h_T V_n \end{pmatrix} \tag{4.17}$$

$\vec{S}$ is a general source term vector which can consist of body forces, chemical reaction terms, or artificial source terms (such as manufactured solution source terms).

The flow solver employed for 2D Euler simulations is SENSEI (Structured, Euler/Navier-Stokes Explicit-Implicit solver) [10] which is a structured grid, cell-centered, second-order finite-volume CFD solver. The code base of SENSEI is designed to incorporate the residual-based error estimation and grid adaptation. For example, evaluation of TE (using discrete residual) when exact solution is available and various TE-estimation methods [22, 20] are available which is key to our TE-adaptation process.

The first application problem selected utilizes the Prandtl-Meyer expansion fan which is a 2D supersonic flow past a sharp convex corner resulting in an infinite number of Mach waves creating a "fan" shape. Here, the flow consists of a Mach 1.2 flow at atmospheric pressure and standard temperature going through a 12 ° downward expansion turn. The computational domain extends between $0 \leq (x, y) \leq 1$ and consists of straight boundaries. The exact solution for this problem is know using the Prandtl-Meyer function and consists of a uniform flow region at the inflow and outflow, sudden flow variation (discontinuity in solution gradient) at the fan leading and trailing edges, and a region of smooth variation

between the fan edges. Two cases are considered with different locations of the expansion fan vertex: (a) the singularity case, where the vertex of the expansion fan is at the domain origin, and (b) the edge case, where the leading/trailing edges of the expansion fan intersect with the domain boundaries. The singularity and the edge cases are shown in Figs. 4.2(c) and (d), respectively. These cases show different adaptation behavior due to the truncation error being dominated by the singularity in the former case and being dominated by the fan edges in the latter case. To obtain the steady-state flow solution for this problem, we use Roe's flux scheme with fully-upwinded MUSCL extrapolation for the evaluation of left and right state of solution variables. The boundary conditions are prescribed using the exact solutions.

4.4 Results

Results of the analysis done are presented in this section. While all trends that were analyzed are discussed, only the most representative case is shown using figures for conciseness.

4.4.1 1D Burgers equation

All results discussed for 1D Burgers equation are adaptively converged to an L_1 norm (or average) of the WEE of $< 0.1\%$. The maximum number of smoothing passes in the smoothing operator of Eqn. 4.9 are kept at 5 passes for coarser meshes (9 and 17 nodes) and at 10-15 passes for finer meshes (33 and 65 nodes). For $N = 65$ (where N is the number of mesh nodes) and a high Reynolds number of $Re = 128$, the different adaptation monitors compared are truncation error, solution gradient ($\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}$), and solution curvature ($\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2}$). It can be seen from Figs. 4.3(a) and (b) that while it is possible to achieve very small levels of equidistribution error norms, this is not necessary since the peak TE does not decrease significantly after average WEE reaches within about 1%. Fig. 4.3(b) shows that reduction in peak TE is much larger when TE-based adapted is employed compared to the feature-based adaptation. The benefit of using TE-based adaptation here is more clearly evident in Fig. 4.3(e) showing the truncation error and in Fig. 4.3(c) showing the node spacing for different adaptation monitors. (Note that the solution is skew-symmetric about the center as seen in Fig. 4.3(d) for TE over the domain, meaning only half of the domain needs to be analyzed.) While the gradient-based method adapts to resolve the large gradient at $x = 0$ and the curvature-based method adapts to resolve the large curvature near the center, the TE-based method results in an overall reduction of truncation error and thus discretization error (Fig. 4.3(f)). Essentially, TE can be seen as combining the contribution of solution gradient, curvature, higher derivatives, as well as effects of mesh quality (e.g., stretching factor in 1D) and mesh resolution [8].

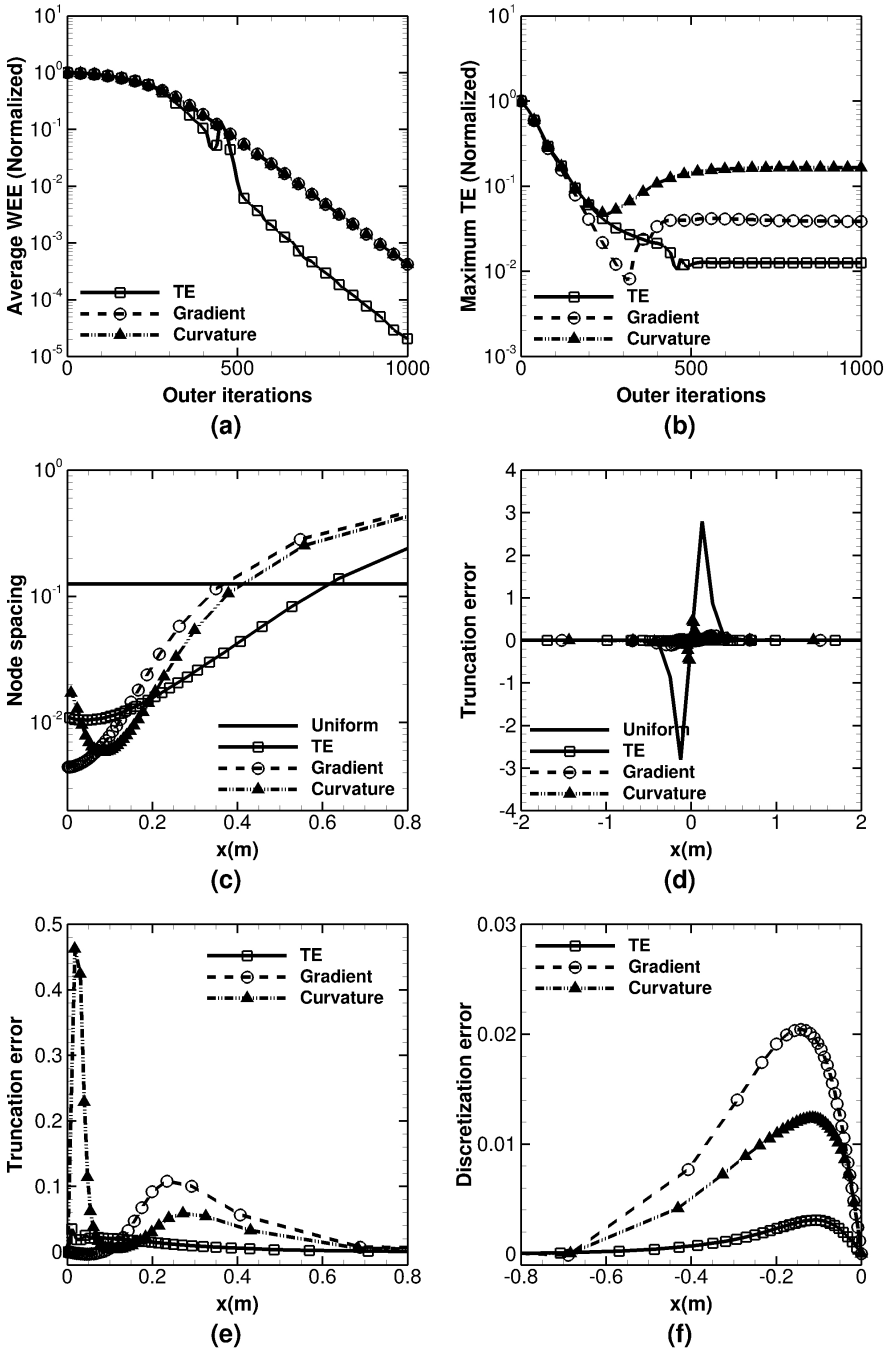

Figure 4.3: Adaptation for 1D Burgers equation with $Re = 128$ and $N = 65$ using truncation error, gradient, and curvature as the adaptation monitors: (a) average equidistribution error vs. adaptive iterations; (b) peak truncation error vs. adaptive iterations; (c) node spacing on the adapted meshes (enlarged view of $0 \leq x \leq 0.8$); truncation error on uniform and adapted meshes for (d) the complete domain, (e) enlarged view of $0 \leq x \leq 0.8$; and (f) discretization error on adapted meshes (enlarged view of $-0.8 \leq x \leq 0$)

.

Table 4.1: For 1D Burgers equation, improvement factors in maximum truncation error and discretization error for different mesh sizes, N, and Reynolds numbers, Re.

	TE Improvement				DE Improvement			
Re	$N = 9$	$N = 17$	$N = 33$	$N = 65$	$N = 9$	$N = 17$	$N = 33$	$N = 65$
8	1.1	1.2	1.1	1.1	1.4	1.5	1.5	1.5
16	2.2	2.4	2.5	2.4	2.8	2.3	2.4	2.4
32	2.0	6.0	7.2	8.0	6.5	9.9	6.9	7.1
64	–	6.7	20.7	25.4	–	33.7	38.7	26.7
128	–	–	26.5	78.6	–	–	**148.9**	**154.6**

Quantitatively, for $N = 65$ and $Re = 128$, TE-based adaptation results in approximately 78 times reduction in peak TE and approximately 154 times reduction in peak DE when compared to the uniform (un-adapted) mesh. For higher Reynolds number cases, the DE reductions were more than two orders of magnitude compared to the uniform mesh. The effect of different mesh sizes and the strength of viscous shock (i.e., Reynolds number) is shown in Table 4.1. Trivially, for a given mesh size, as the strength of the viscous shock increases, the factor of improvement in both peak TE and peak DE increases since the peak error on the un-adapted mesh is higher for large Reynolds number. More conclusively, for a given Reynolds number, the improvement factor for coarser meshes is significantly less than that for finer meshes. This is the phenomenon of node starvation where there are simply not enough mesh nodes available in the domain to resolve the feature properly. For a few pathological cases with a small number of mesh nodes ($N = 9$ and $N = 17$) and large Reynolds number, adaptation could not be converged. For other cases, approximately an order of magnitude or higher improvement in TE and DE is observed when enough mesh nodes are available to resolve a relatively strong solution feature.

4.4.2 Quasi-1D nozzle flow

For quasi-1D nozzle flow, the adaptive convergence criterion of WEE $< 1\%$ could not be achieved for several cases while the peak TE is found to stagnate after about an order of magnitude reduction as seen in Fig. 4.4(a). Weight function smoothing is performed to reasonable levels (i.e., between 5-15 passes depending on the mesh size). For the mesh size of $N = 65$ nodes, the node spacing of Fig. 4.4(b) shows that the adaptation process resolves the nozzle throat which is the region of largest TE in this problem and the curvature discontinuity in the nozzle area function at $x = \pm 1$ m. As desired, large cells are found in the straight sections of the nozzle. After a certain level of TE reduction, further equidistribution simply results in small oscillations between concentrating nodes at the throat versus at curvature discontinuities without any significant reduction in TE norms. While approximately an order of magnitude reduction is observed in peak TE and peak DE values, at some locations in the domain the discretization error reduces by much higher factors such as near the outflow

region as shown in Fig. 4.4(d). Note that the discretization error for mass, momentum, and energy equations refer to the errors in primitive variables - density (ρ), velocity (u), and pressure (P), respectively. Using conserved variables for analysis (i.e., ρ, ρu, and ρe_T) showed similar error improvement factors to that for primitive variables.

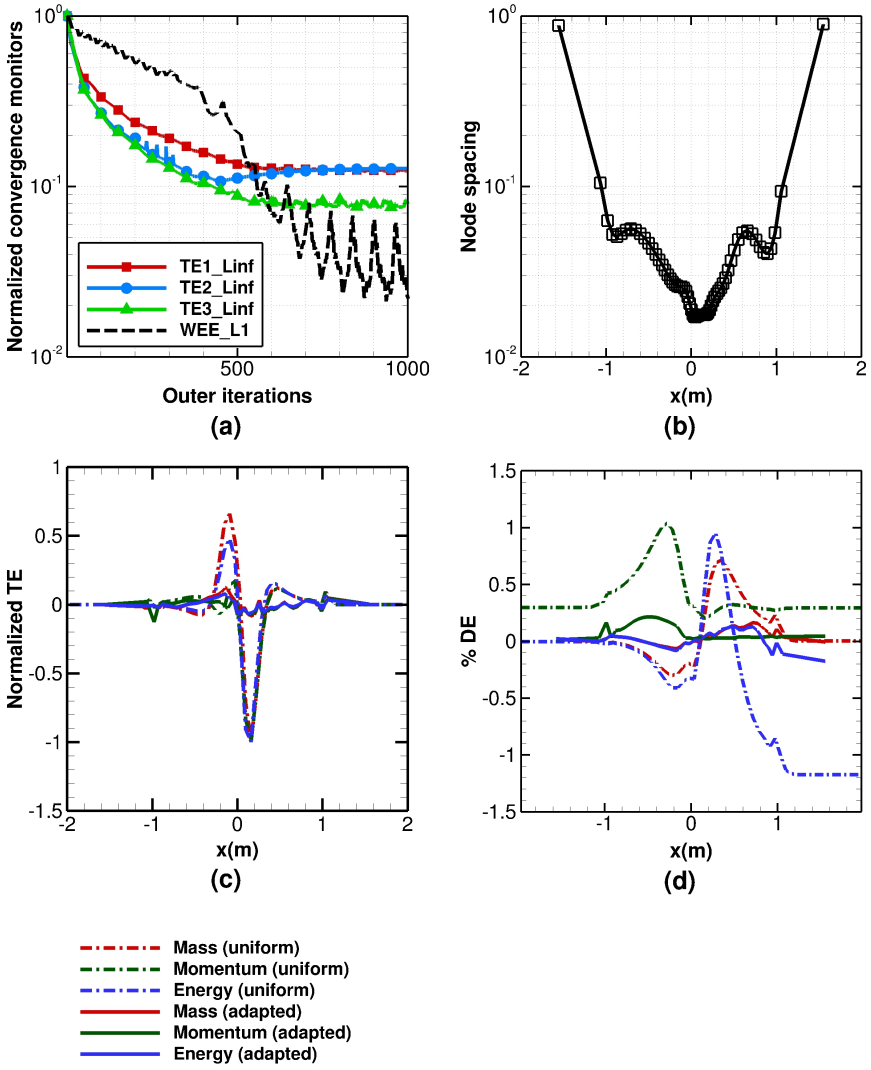

Figure 4.4: Adaptation for quasi-1D nozzle flow with $N = 65$ nodes using truncation error as the adaptation monitor: (a) adaptive convergence monitors vs. adaptive iterations; (b) node spacing on the adapted mesh; (c) truncation error on uniform vs. adapted mesh, and (d) percentage discretization error obtained on uniform vs. adapted mesh.

Dealing with a system of conservation equations for Euler problems makes the weight function determination challenging. Though Eqn. 4.8 is used in the current work, it is not conclusively the best approach. Adapted meshes obtained by using TE from individual

conservation equations are compared against the adapted mesh obtained from combined TE-based adaptation in Fig. 4.5(a) for $N = 33$. It appears that momentum equation behaves differently in comparison to the other two equations and adapts to a different mesh. As seen from Table 4.2, only combined TE produces an overall reduction of about a factor of 5 in peak discretization error for all the primitive variables. Currently, a multimesh strategy is being explored where truncation error from each governing equation is used to perform mesh adaptation separately such that each governing equation gets solved on the mesh most optimized for its individual solution.

With an increase in the number of mesh nodes, no significant change is observed in peak error improvement factors as shown in Table 4.3 with the only exception being the $N = 17$ case. For $N = 17$, we again see node starvation in Fig. 4.5(b) as the relevant regions of the domain (i.e., throat, and curvature discontinuities) get properly resolved only with larger mesh sizes (65 and 129 nodes).

Though it cannot be ruled out that initial meshes may have an effect on the final adapted mesh obtained with this method, we observed no such issues for the few cases tested. For the three different grids considered (uniform, clustered near the throat, and clustered near the boundaries) as the initial meshes (shown in Fig. 4.5(c)), we obtain almost identical adapted meshes (Fig. 4.5(d)).

Table 4.2: For quasi-1D nozzle flow, improvement factors in peak discretization error for different adaptation monitors ($N = 33$).

	DE Improvement		
Monitor	ρ	u	P
Combined TE	4.3	6.6	4.9
Mass TE	0.8	5.4	1.0
Mtm. TE	2.3	7.2	2.7
Energy TE	0.8	5.8	1.0

Table 4.3: For quasi-1D nozzle flow, improvement factors in peak discretization error for different mesh sizes (with combined TE as the adaptation monitor).

	DE Improvement		
N	ρ	u	P
17	1.6	5.2	1.9
33	4.3	6.6	4.9
65	4.8	6.8	5.2
129	4.6	7.4	5.1

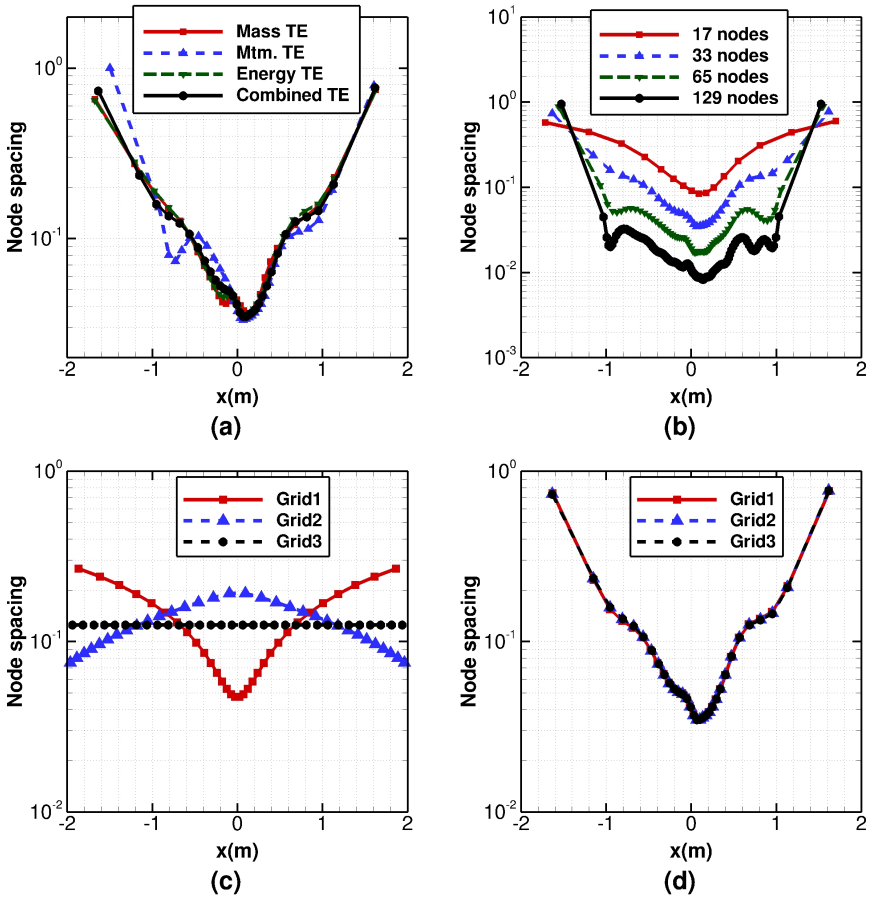

Figure 4.5: Various comparisons for quasi-1D nozzle flow using TE-based adaptation: (a) node spacing on adapted meshes for adaptation monitor based upon combined TE vs TE from individual governing equations, (b) adapted meshes obtained when different mesh sizes are used, (c) three test grids with different initial grid spacing, and (d) adapted grids obtained for the three test grids.

4.4.3 Supersonic expansion fan

For the singularity case (where the leading and trailing edges of the expansion pass through the origin) an adaptive convergence criterion of WEE $< 1\%$ is readily achieved as shown in Fig. 4.6(a) where a mesh with 65×65 nodes is being used. With adaptive iterations, cells with large TE get significantly smaller. This results in large reduction of the area weighted truncation error which consequently results in reduction of discretization error over the domain. Fig. 4.6(a) shows that approximately 50 times reduction in the maximum cell area weighted truncation error is achieved within 30 adaptive iterations. It can be seen that the adapted mesh of Fig. 4.6(b) has been driven primarily by the singularity that exists at $(x, y) = (0, 0)$. The large region of TE near the singularity in Fig. 4.6(c) is reduced to a significantly smaller size with adaptation as shown in Fig. 4.6(d). Similarly, regions of large discretization error in Fig. 4.6(e) that are located along the edges of the fan (due to transported error from the singularity region) significantly reduce in size as mesh nodes are relocated towards the region of large TE (i.e., the singularity) as shown in Fig. 4.6(f).

Adaptation for the edge driven case shows a more complex behavior than the singularity case and results in node oscillations during the adaptation process. Adaptive convergence is shown in Fig. 4.7(a) for 65×65 nodes mesh where reduction in truncation error at outer iteration of 300 is achieved by reducing the number of inner iterations in the adaptation process. The locations where the fan edges intersect the $x = 0$ and $y = 0$ boundaries of the domain become regions of relevance along with the leading and trailing edges of fan itself. The adapted mesh of Fig. 4.7(b) is driven by these competing regions and makes the adaptation behavior more complex than the singularity case. Figs. 4.7(c)-(f) again show significant reduction of both TE and DE over the domain with mesh adaptation.

The comparison of adapted vs uniform meshes is better seen when discretization error for different solution variables are compared along certain slices of the domain (e.g., $y = 0.5$ and $x = 0.5$). Figs. 4.8(a) and (b) show significant gains in discretization error on most of the $y = 0.5$ slice. The improvement is better for the singularity case as seen from Fig. 4.8(a). For the edge case shown in Fig. 4.8(b), the trailing edge of fan has some error likely transported from where it intersects the inflow boundary. A similar trend is seen in Figs. 4.8(c) and (d) for the $x = 0.5$ slice.

4.5 Conclusions

Since truncation error acts as the local source of discretization error in the domain, it was found that mesh adaptation performed to reduce the TE in the domain results in a commensurate reduction of discretization error. Compared to feature-based monitors, TE combines the effect of solution derivatives as well as mesh quality metrics and was found to perform better for the cases analyzed. Results in improvement due to TE-based adaptation var-

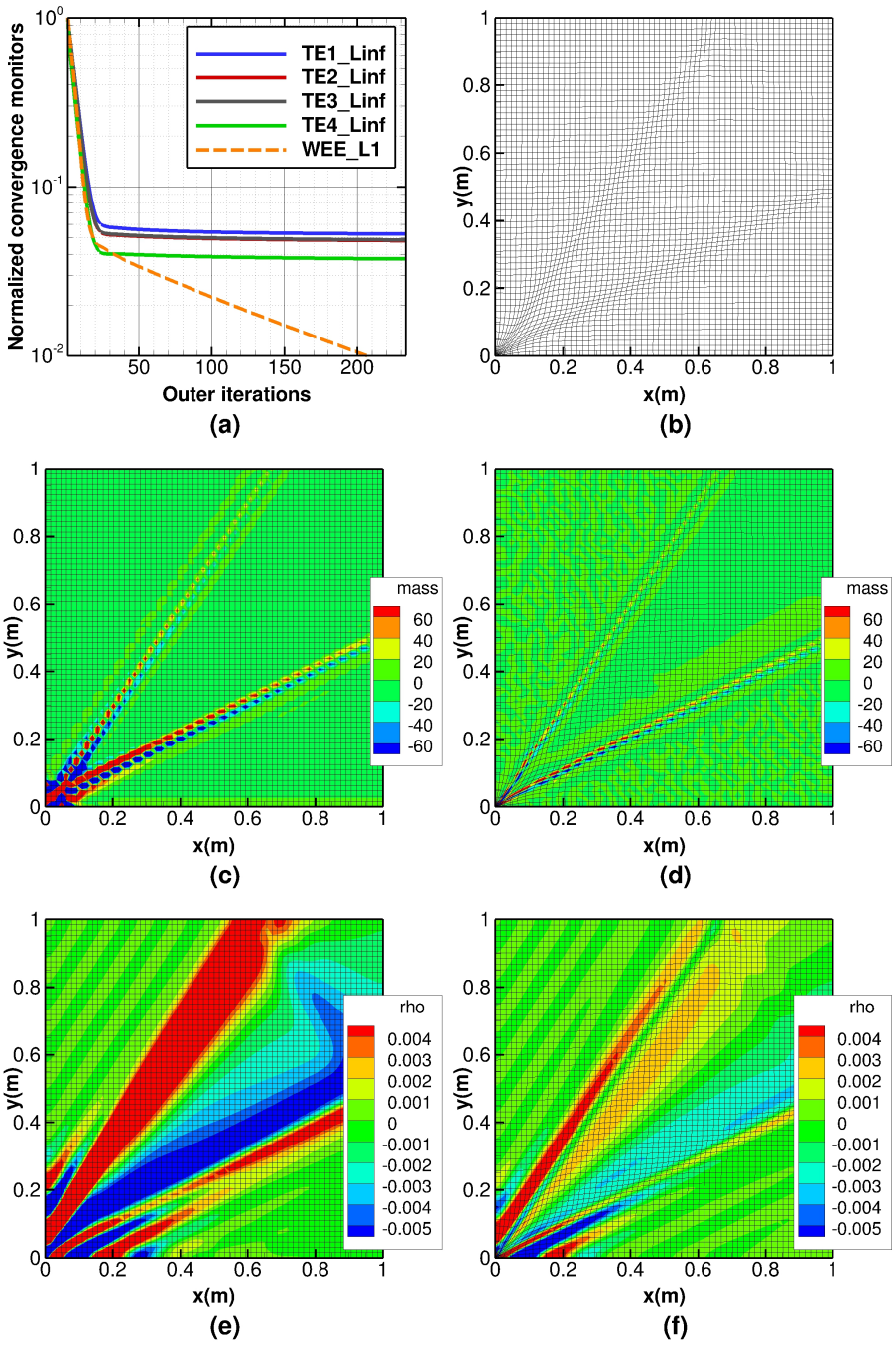

Figure 4.6: TE-based adaptation for supersonic expansion flow when adaptation is dominated by singularity (65×65 nodes): (a) normalized convergence monitors vs. adaptive iterations; (b) adapted mesh; mass truncation error on (c) initial uniform mesh, (d) adapted mesh; and density discretization error on (e) initial uniform mesh, (f) adapted mesh.

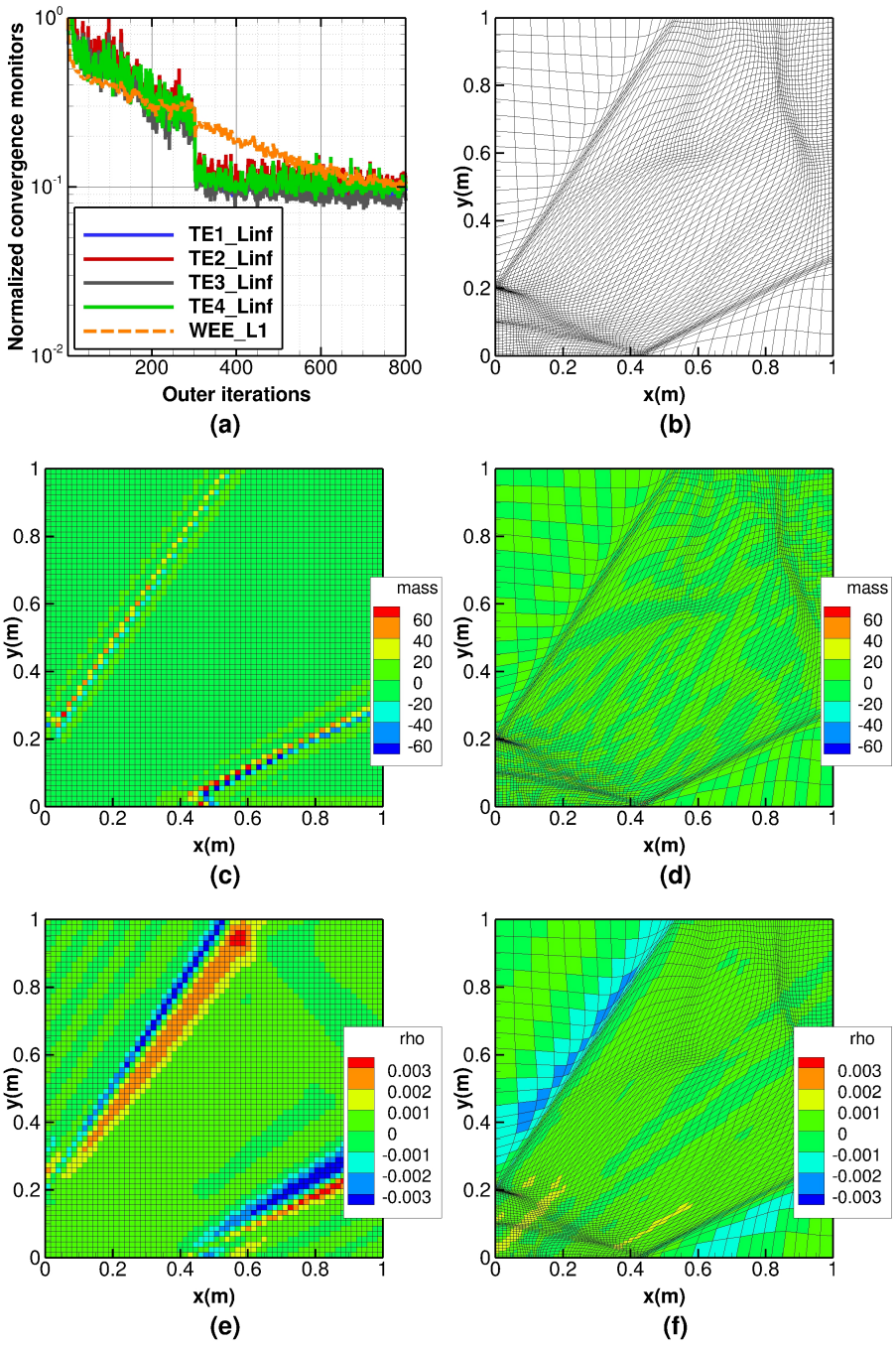

Figure 4.7: TE-based adaptation for supersonic expansion flow when adaptation is dominated by fan edges (65×65 nodes): (a) normalized convergence monitors vs. adaptive iterations; (b) adapted mesh; mass truncation error on (c) initial uniform mesh, (d) adapted mesh; and density discretization error on (e) initial uniform mesh, (f) adapted mesh.

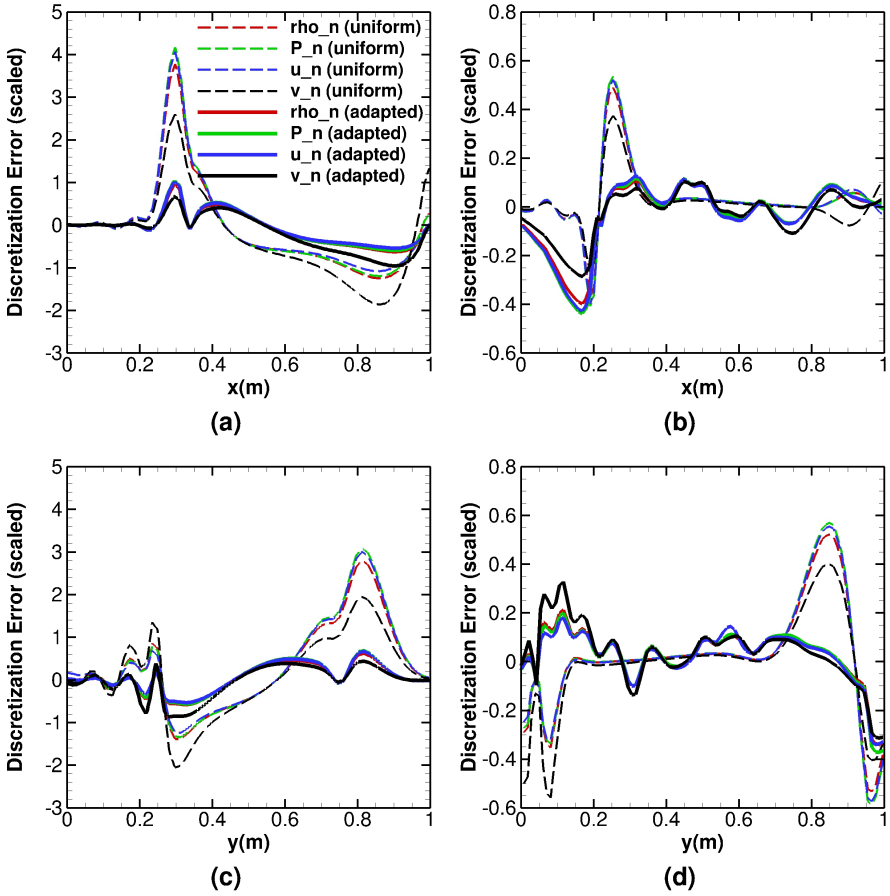

Figure 4.8: Comparison of scaled DE values obtained on adapted vs. uniform meshes for expansion fan problems. Along $y = 0.5$ slice for (a) singularity case, (b) edge case, Along $x = 0.5$ slice for (c) singularity case, (d) edge case.

ied from approximately 10 times reduction in discretization error (for Euler flow problem) to approximately 100 times reduction (for Burgers equation) when compared to a uniform mesh. The reduction in discretization error was found to be even larger in local regions of the domain instead of the reduction in a global quantity (e.g., norms of the errors). The method was shown to work for 1D and 2D Euler problems Flow problems with singularities and other discontinuities were handled by the adaptation monitor as expected. In applying this method to real problems, the biggest challenge is to obtain accurate truncation error estimates when the exact solution is not available.

An error equidistribution strategy was implemented to perform r-refinement for 1D and 2D structured grids and some nuances of the equidistribution process were discussed. Though equidistribution is a simple and computationally inexpensive way to perform mesh relocation, it has some drawbacks as currently implemented. Mesh relocation here was performed primarily by "pulling" the mesh nodes towards the region of large error rather than combining it with other techniques such as rotating the cells when required. An optimization method where truncation error (or its measure) is the objective function to be minimized and mesh nodes are the unknowns, though computationally very expensive, was shown to give several orders of magnitude reduction in discretization error when tested for simple 1D and 2D Burgers equations [1, 21]. Another alternative could be to employ more sophisticated mesh movement mechanisms involving linear and torsional springs to account for rotation of cells.

Our process relied heavily on having a smooth expression for the truncation error over the domain. When truncation error was not smooth, some degree of smoothness was achieved by using a low-pass filter as described in Eqn. 4.9. However, as the mesh adapts, the truncation error becomes more noisy and unsuitable for a smooth and convergent adaptation, especially for complex problems. We believe that addressing this problem of smoothening the weight function enough without removing the underlying information would improve the results as well as the computational efficiency of the process.

Acknowledgements

Part of this work was funded by the Air Force Office of Scientific Research (AFOSR) Computational Mathematics Program managed by Dr. Fariba Fahroo under Grant FA9550-12-1-0173. The authors would like to acknowledge Joseph Derlaga (for providing the quasi-1D nozzle solver) and Tyrone Phillips (for providing the TE estimation routines).

Bibliography

[1] Edward Alyanak, Christopher Roy., and Aniruddha Choudhary. Investigation of optimal grid distributions for Burgers' equation. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2011. doi: 10.2514/6.2011-888. URL `http://dx.doi.org/10.2514/6.2011-888`.

[2] Dale A. Anderson. Equidistribution schemes, Poisson generators, and adaptive grids. *Appl. Math. Comput.*, 24(3):211–227, December 1987. ISSN 0096-3003. doi: 10.1016/0096-3003(87)90085-3. URL `http://dx.doi.org/10.1016/0096-3003(87)90085-3`.

[3] Timothy J. Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(34):243 – 273, 1997. ISSN 0168-874X. doi: http://dx.doi.org/10.1016/S0168-874X(96)00032-7. URL `http://www.sciencedirect.com/science/article/pii/S0168874X96000327`. Adaptive Meshing, Part 2.

[4] Timothy Barth. Recent developments in high order k-exact reconstruction on unstructured meshes. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 1993. doi: 10.2514/6.1993-668. URL `http://dx.doi.org/10.2514/6.1993-668`.

[5] Timothy Barth and Paul Frederickson. Higher order solution of the euler equations on unstructured grids using quadratic reconstruction. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 1990. doi: 10.2514/6.1990-13. URL `http://dx.doi.org/10.2514/6.1990-13`.

[6] Mart Borsboom. Development of an error-minimizing adaptive grid method. *Applied Numerical Mathematics*, 26(12):13 – 21, 1998. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/S0168-9274(97)00077-9. URL `http://www.sciencedirect.com/science/article/pii/S0168927497000779`.

[7] Aniruddha Choudhary and Christopher Roy. Efficient residual-based mesh adaptation for 1D and 2D CFD applications. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2011. doi: 10.2514/6.2011-214. URL `http://dx.doi.org/10.2514/6.2011-214`.

[8] Aniruddha Choudhary and Christopher Roy. A truncation error-based approach to understanding and improving mesh quality in CFD. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2012. doi: 10.2514/6.2012-607. URL `http://dx.doi.org/10.2514/6.2012-607`.

[9] Aniruddha Choudhary and Christopher J. Roy. Structured mesh r-refinement using truncation error equidistribution for 1D and 2D Euler problems. In *Fluid Dynamics*

and Co-located Conferences, pages –. American Institute of Aeronautics and Astronautics, June 2013. doi: 10.2514/6.2013-2444. URL http://dx.doi.org/10.2514/6.2013-2444.

[10] Joseph M. Derlaga, Tyrone Phillips, and Christopher J. Roy. SENSEI computational fluid dynamics code: A case study in modern Fortran software development. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2013. doi: 10.2514/6.2013-2450. URL http://dx.doi.org/10.2514/6.2013-2450.

[11] Richard P. Dwight. Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation. *Journal of Computational Physics*, 227(5):2845 – 2863, 2008. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2007.11.020. URL http://www.sciencedirect.com/science/article/pii/S002199910700513X.

[12] Peter R. Eiseman. Adaptive grid generation. *Computer Methods in Applied Mechanics and Engineering*, 64(13):321 – 376, 1987. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/0045-7825(87)90046-6. URL http://www.sciencedirect.com/science/article/pii/0045782587900466.

[13] Xubin Gu and Tom Shih. Differentiating between source and location of error for solution-adaptive mesh refinement. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2001. doi: 10.2514/6.2001-2660. URL http://dx.doi.org/10.2514/6.2001-2660.

[14] D.F Hawken, J.J Gottlieb, and J.S Hansen. Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations. *Journal of Computational Physics*, 95(2):254 – 302, 1991. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(91)90277-R. URL http://www.sciencedirect.com/science/article/pii/002199919190277R.

[15] J. Kautsky and N. Nichols. Equidistributing meshes with constraints. *SIAM Journal on Scientific and Statistical Computing*, 1(4):499–511, 1980. doi: 10.1137/0901036. URL http://dx.doi.org/10.1137/0901036.

[16] K. R. Laflin. *Solver-Independent r-Refinement Adaptation for Dynamic Numerical Simulations*. PhD thesis, North Carolina State University, 1997.

[17] D.Scott McRae. r-refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 189(4):1161 – 1182, 2000. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/S0045-7825(99)00372-2. URL http://www.sciencedirect.com/science/article/pii/S0045782599003722. Adaptive Methods for Compressible {CFD}.

[18] K. Nakahashi and G. S. Deiwert. Three-dimensional adaptive grid method. *AIAA Journal*, 24(6):948–954, June 1986. ISSN 0001-1452. doi: 10.2514/3.9369. URL http://dx.doi.org/10.2514/3.9369.

[19] W.L. Oberkampf and C.J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, 2010.

[20] Tyrone Phillips and Christopher J. Roy. Defect correction and error transport discretization error estimation for applications in cfd. In *AIAA Aviation*, pages –. American Institute of Aeronautics and Astronautics, June 2014. doi: 10.2514/6.2014-2572. URL http://dx.doi.org/10.2514/6.2014-2572.

[21] Tyrone Phillips, Christopher Roy, Edward Alyanak, and Carl Ollivier Gooch. Optimal mesh adaption for Burgers' equation. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2012. doi: 10.2514/6.2012-2710. URL http://dx.doi.org/10.2514/6.2012-2710.

[22] Tyrone Phillips, Joseph M. Derlaga, Christopher J. Roy, and Jeffrey Borggaard. Finite volume solution reconstruction methods for truncation error estimation. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2013. doi: 10.2514/6.2013-3090. URL http://dx.doi.org/10.2514/6.2013-3090.

[23] Christopher Roy. Strategies for driving mesh adaptation in CFD (invited). In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2009. doi: 10.2514/6.2009-1302. URL http://dx.doi.org/10.2514/6.2009-1302.

[24] R. D. Russell and J. Christiansen. Adaptive mesh selection strategies for solving boundary value problems. *SIAM Journal on Numerical Analysis*, 15(1):pp. 59–80, 1978. ISSN 00361429. URL http://www.jstor.org/stable/2156563.

[25] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(78)90023-2. URL http://www.sciencedirect.com/science/article/pii/0021999178900232.

[26] David A Venditti and David L Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204 – 227, 2000. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2000.6600. URL http://www.sciencedirect.com/science/article/pii/S0021999100966002.

[27] David A. Venditti and David L. Darmofal. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176

(1):40 – 69, 2002. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2001.6967. URL http://www.sciencedirect.com/science/article/pii/S0021999101969670.

[28] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992. ISSN 1097-0207. doi: 10.1002/nme.1620330703. URL http://dx.doi.org/10.1002/nme.1620330703.

Chapter 5

Conclusions

A code verification study of different boundary conditions commonly used in compressible and incompressible CFD simulations was presented. Also, a code verification study employing the method of manufactured solutions was presented for the governing equations of the two-fluid model used in multiphase flows.

The techniques proposed by Bond et al. [2] for boundary condition verification were applied to subsonic inflow, subsonic outflow, supersonic outflow, no-slip wall, and slip-wall boundary conditions with laminar Navier-Stokes equations by identifying the boundary constraints and deriving appropriate manufactured solutions for each case. Since a given boundary condition can be implemented with different formulations (such as Neumann or extrapolation-based), many of the manufactured solutions presented vary from previous works and add to the database of boundary condition verification techniques.

Verification techniques for the subsonic inflow boundary conditions were presented for the first time where the use of simplified curved boundaries (e.g., spherical caps) was proposed in order to easily derive the manufactured solutions satisfying the relevant boundary constraints. Furthermore, the use of general meshes with hybrid boundaries (i.e., boundaries containing cells of mixed types) was proposed for rigorous code verification. In the case of no-slip wall boundary conditions, errors were identified at the interface of different cell types at the wall boundary which remained masked with similar analysis on boundaries with all hexahedral cells. Similar issues were encountered when tetrahedral and prismatic cells were used at the boundaries while testing the subsonic inflow and subsonic outflow boundary conditions.

Different features of an open-source, incompressible, multiphase code were verified including the discretized terms of momentum, pressure-correction, and energy equations, for 2D and 3D, steady/unsteady, single-phase and two-phase incompressible flows. A newly-developed curl-based method to derive manufactured solutions was introduced for divergence free flows during the verification of baseline governing equations as well as during boundary

condition verification. No-slip and free-slip wall boundary conditions were verified to be second order accurate for momentum equations while the pressure outflow boundary condition was found to be second order accurate for momentum as well as pressure-correction equations on 3D grids using single-phase flow equations.

Simplifications were made by assuming incompressible flow, constant viscosity, constant volume fraction, and by neglecting the algebraic constitutive relations that describe the essential interactions between phases in a multiphase flow simulation. Even with these simplifications, many useful lessons were learned from this code verification study about the implementation of discrete governing equations in this code. The multiphase code, MFIX, was found to be second order accurate after addressing certain issues. Some issues were uncovered in the implementation of Superbee scheme at the boundaries, in the evaluation of cross-terms of the strain-tensor during steady-state simulations, and in the second order time-stepping method. It was observed that the common practice of setting the ghost cell size equal to the first interior cell size leads to some reduction in solution accuracy without completely failing the order test.

The code verification methods presented here are general and can be applied to verify boundary conditions in other compressible and incompressible CFD codes. For instances where the boundary conditions have been implemented under different assumptions, these methods should be modified accordingly. On the same note, it is recommended that code developers interested in boundary condition verification document the exact nature of boundary conditions implemented in their code specifying the physical, mathematical, numerical aspects, as well as boundary flux formulation procedures for finite-volume schemes

Code verification of multiphase flows is challenging, especially in cases where the interface between the phases is resolved using discrete elements (such as in Lagrangian-Eulerian methods). Though current study addresses only continuum governing equations for code verification and is similar to single-phase flow code verification in many ways, the emphasis is put upon using rigorous, mathematical methods such as order of accuracy testing for code verification. The verification of algebraic constitutive models based upon unit-testing in conjunction with the MMS-based verification of discretized terms of the governing equation presented here, along with other practices such as model validation and error quantification are ultimately necessary to achieve greater confidence in multiphase flow simulations.

Since truncation error acts as the local source of discretization error in the domain, it was found that mesh adaptation performed to reduce the TE in the domain results in a commensurate reduction of discretization error. Compared to feature-based monitors, TE combines the effect of solution derivatives as well as mesh quality metrics and was found to perform better for the cases analyzed. Results in improvement due to TE-based adaptation varied from approximately 10 times reduction in discretization error (for Euler flow problems) to approximately 100 times reduction (for Burgers equation) when compared to a uniform mesh. The reduction in discretization error was found to be even larger in local regions of the domain instead of the reduction in a global quantity (e.g., norms of the errors). The

method was shown to work for 1D and 2D Euler problems Flow problems with singularities and other discontinuities were handled by the adaptation monitor as expected. In applying this method to real problems, the biggest challenge is to obtain accurate truncation error estimates when the exact solution is not available.

An error equidistribution strategy was implemented to perform r-refinement for 1D and 2D structured grids and some nuances of the equidistribution process were discussed. Though equidistribution is a simple and computationally inexpensive way to perform mesh relocation, it has some drawbacks as currently implemented. Mesh relocation here was performed primarily by "pulling" the mesh nodes towards the region of large error rather than combining it with other techniques such as rotating the cells when required. An optimization method where truncation error (or its measure) is the objective function to be minimized and mesh nodes are the unknowns, though computationally very expensive, was shown to give several orders of magnitude reduction in discretization error when tested for simple 1D and 2D Burgers equations [1, 3]. Another alternative could be to employ more sophisticated mesh movement mechanisms involving linear and torsional springs to account for rotation of cells.

The adaptation process relied heavily on having a smooth expression for the truncation error over the domain. When truncation error was not smooth, some degree of smoothness was achieved by using a low-pass filter as described in Eqn. 4.9. However, as the mesh adapts, the truncation error becomes more noisy and unsuitable for a smooth and convergent adaptation, especially for complex problems. It is possible that addressing this problem of smoothening the weight function enough without removing the underlying information would improve the results as well as the computational efficiency of the process.

Bibliography

[1] Edward Alyanak, Christopher Roy., and Aniruddha Choudhary. Investigation of optimal grid distributions for Burgers' equation. In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2011. doi: 10.2514/6.2011-888. URL `http://dx.doi.org/10.2514/6.2011-888`.

[2] Ryan B. Bond, Curtis C. Ober, Patrick M. Knupp, and Steven W. Bova. Manufactured solution for computational fluid dynamics boundary condition verification. *AIAA Journal*, 45(9):2224–2236, 2007. ISSN 0001-1452. doi: 10.2514/1.28099. URL `http://arc.aiaa.org/doi/abs/10.2514/1.28099`.

[3] Tyrone Phillips, Christopher Roy, Edward Alyanak, and Carl Ollivier Gooch. Optimal mesh adaption for Burgers' equation. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2012. doi: 10.2514/6.2012-2710. URL `http://dx.doi.org/10.2514/6.2012-2710`.

Appendix A

Verification of time-step accuracy in MFIX

More details about the temporal verification covered in Sec. 3.4.6 are presented here. Neglecting the higher order terms, the discretization error for a scheme with spatial and temporal terms can be written as

$$\varepsilon_{h_x}^{h_t} = g_x {h_x}^{\hat{p}} + g_t {h_t}^{\hat{q}} \tag{A.1}$$

where $\hat{p}$ and $\hat{q}$ are the observed orders of accuracy in space and time, respectively, g_x and g_t are the coefficients of spatial and time terms, respectively, and h_x and h_t are normalized spatial and temporal discretizations respectively [2].

Following steps are employed to determine the temporal order of accuracy of the code as described in [2]:

1. A fixed constant time-step is selected making the temporal discretization error term a constant. Discretization error (DE) is found on three mesh levels with sytematically refined grid-spacings to determine g_x and $\hat{p}$.

2. A fixed mesh is selected and DE is found for three cases with different time-step values. Thus, g_t and $\hat{q}$ in the temporal part of the discretization error expression Eq. A.1 can be obtained.

3. To ensure that the temporal and spatial errors are of similar orders, h_x and h_t are selected such that $g_x {h_x}^{\hat{p}} \approx g_t {h_t}^{\hat{q}}$. This selection needs trial and error, keeping the cost of computation and stability into consideration.

4. For these selections of spatial and temporal discretization sizes and known formal orders (easier when $r_x = r_t$), 2 cases are run to determine the temporal observed order of accuracy for the combined spatial-temporal refinements.

This process is shown below for the Euler-implicit time-stepping:

Step 1 Using constant time-step ($\delta t = 0.1$ sec), and three different mesh spacings ($N = 8 \times 8 \times 8$, $N = 16 \times 16 \times 16$, and $N = 32 \times 32 \times 32$), find g_x and $\hat{p}$ from

$$\hat{p} = \frac{ln\left(\frac{\left\|\varepsilon^{h_t}_{r_x^2 h_x}\right\| - \left\|\varepsilon^{h_t}_{r_x h_x}\right\|}{\left\|\varepsilon^{h_t}_{r_x h_x}\right\| - \left\|\varepsilon^{h_t}_{h_x}\right\|}\right)}{ln\left(r_x\right)} \tag{A.2}$$

$$g_x = \frac{\left\|\varepsilon^{h_t}_{r_x h_x}\right\| - \left\|\varepsilon^{h_t}_{h_x}\right\|}{h_x^{\hat{p}}\left(r_x^{\hat{p}} - 1\right)} \tag{A.3}$$

where, r_x is the refinement factor for spatial refinement. This is calculated in Table A.1.

Table A.1: Determination of $\hat{p}$ and g_x

	P_g	u_g	v_g	w_g
$\hat{p}$	1.40	2.55	2.55	2.63
g_x	7.99	9.29	9.29	3.87

Step 2 Similarly, using fixed mesh spacing ($N = 8 \times 8 \times 8$) and three different time-steps ($\delta t = 0.1$ sec, $\delta t = 0.05$ sec, and $\delta t = 0.025$ sec), g_t and $\hat{q}$ are obtained. This is calculated in Table A.2

Table A.2: Determination of $\hat{q}$ and g_t

	P_g	u_g	v_g	w_g
$\hat{q}$	1.76	0.97	0.97	0.69
g_t	22.04	3.95	3.95	-0.01

Step 3 Based on the obtained values of $\hat{p}$, $\hat{q}$, g_x, and g_t, four cases are selected for combined temporal and spatial refinement study such that $g_x {h_x}^{\hat{p}} \approx g_t {h_t}^{\hat{q}}$. Based upon the suggestions in [2], for first order temporal methods and second order spatial methods, the temporal and spatial refinements factors are selected as $r_t = 4$ and $r_x = 2$, respectively. These cases are described in Table A.3, where N_x and N_t are the number of grid points along one dimension and number of time-steps, respectively.

Step 4 Finally, observed orders for the combined refinement study suggest that Euler implicit time-stepping method is first order accurate as shown in Tables A.4 and A.5.

Table A.3: Selected spatial and temporal scales for combined spatial-temporal verification.

	Case	h_x	N_x	h_t	N_t
Selected	A	0.125	8	0.2	5
Refined	B	0.0625	16	0.05	20
Refined	C	0.03125	32	0.0125	80
Refined	D	0.015625	64	0.003125	320

Table A.4: L_2 norms of discretization error obtained using combined spatial-temporal refinement.

Case	P_g	u_g	v_g	w_g
A	1.53E+00	7.37E-01	7.37E-01	1.59E-02
B	3.38E-01	2.22E-01	2.22E-01	3.07E-03
C	7.47E-02	5.56E-02	5.56E-02	6.93E-04
D	1.87E-02	1.39E-02	1.39E-02	1.66E-04

Table A.5: Spatial and temporal orders of accuracy for Euler-implicit time-stepping

Case	Spatial orders ($r_x = 2$)				Temporal orders ($r_t = 4$)			
(A,B)	2.176	1.731	1.731	2.375	1.088	0.866	0.866	1.187
(B,C)	2.178	1.996	1.996	2.146	1.089	0.998	0.998	1.073
(C,D)	1.998	2.003	2.003	2.065	0.999	1.001	1.001	1.033

Appendix B

MMS verification cases for MFIX

Table B.1: MMS-based code verification cases for MFIX

#	Case ID	Details	Notes
1	horizontal-channel-superbee	2D, single-phase, steady, Superbee scheme, uniform mesh	2^{nd} order
2	vertical-channel-superbee	2D, single-phase, steady, Superbee scheme, uniform mesh	2^{nd} order
3	sinusoidal-2d-superbee	2D, single-phase, steady, Superbee scheme, uniform mesh	Found 1^{st} order errors at W, S, and B boundary cells due to data structure; Subsequent cases tested with Central scheme only; Found that stress tensor trace and cross terms must be evaluated at each subiteration; Found errors in physical properties at the ghost cells
4	sinusoidal-2d	2D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order

Continued on next page

Table B.1 – *Continued from previous page*

#	Case ID	Details	Notes
5	curlbased-3d	3D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order; Stretched meshes have large (but 2^{nd} order) errors at boundaries for pressure variable
6	sinusoidal-3d-unsteady-ee	3D, single-phase, unsteady, Central spatial scheme, Euler implicit temporal scheme, uniform mesh	1^{st} order
7	sinusoidal-3d-unsteady-cn	3D, single-phase, unsteady, Central spatial scheme, Crank-Nicholson temporal scheme, uniform mesh	Temporal scheme is Implicit mid-point Scheme or 2-step Implicit Runge-Kutta; Pressure variable not extrapolated in the second-step; 1^{st} order temporal
8	curlbased-3d-twophase	3D, two-phase, steady, Central scheme, stretched mesh	2^{nd} order; Pressure, gas momentum, solid momentum, gas temperature, solid temperature, and granular temperature verified; variable volume-fraction and continuity equations not tested
9	bc-nsw-2d	No-slip wall BC, 2D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order; Pressure equation not tested
10	bc-fsw-2d	Free-slip wall BC, 2D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order; Pressure equation not tested
11	bc-po-2d	Pressure outflow BC, 2D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order

Continued on next page

Table B.1 – *Continued from previous page*

#	Case ID	Details	Notes
12	bc-nsw-3d	No-slip wall BC, 3D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order; Pressure equation not tested
13	bc-fsw-3d	Free-slip wall BC, 3D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order; Pressure equation not tested
14	bc-po-3d	Pressure outflow BC, 3D, single-phase, steady, Central scheme, stretched mesh	2^{nd} order

Note: Results are as reported on June 26^{th}, 2014.

Appendix C

Structured Adaptation Module (SAM): User's guide

Date: November 24th, 2014

Author and Contact: Aniruddha Choudhary (aniruddhac@gmail.com, anirudd@vt.edu)

C.1 Short introduction

The Structured Adaptation Module (henceforth referred to as SAM) is a suite of procedures which can be used by any code for performing r-refinement of structured grids using a provided weighting function (also referred to as adaptation monitor, or adaptation criterion) in the literature. SAM uses a weight- function equidistribution approach to perform mesh adaptation. r-refinement is a mesh relocation technique which redistributes the grids nodes such that the number of nodes remain constant. This is in contrast with h-refinement, where grid-nodes are added or removed from the domain depending on the adaptation criterion, and p-refinement, where the order of the basis function is modified for performing mesh adaptation. h-refinement is more suitable for unstructured mesh adaptation, p-refinement is more used in the Finite Element community, while r-refinement is popular with both structured and unstructured meshes. There are also hybrid mesh refinement techniques. SAM currently only works with r-refinement. Henceforth, any reference to mesh adaptation with SAM refers to this type of refinement. For more details about the equidistribution method and r-refinement, refer to [1] and [3].

Note: All floating-point calculations in the code are with double precision, e.g.:

```
Integer, Parameter :: dp = Selected_Real_Kind(p=13,r=200)
Real(dp)           :: x
```

C.2 Installing and using SAM

To use SAM, one should include "Use SAM" inside the procedure (called interfacing subroutine) that wishes to have access to the publicly accessible subroutines for mesh adaptation. There are also other utilities available within the codebase for performing related calculations such as evaluation of a function norms.

The interfacing subroutine should have the following:

- Access to the SAM library, `libsam.a`
- Size of the grid: e.g., `imax`, `jmax`, `kmax` in 3D
- Access to a grid file in the form of a multidimensional array (e.g, `xin(:,:,:)`)
- Access to a weighting function in the form of a multidimensional array (e.g., `Fin(:,:,:)`). The weighting function can be mesh-node located (for 1D and 2D) or cell-center located (for 1D only).
- Access to an array of the same size and type as `xin` to receive the adapted grid (e.g., `xout(:,:,:)`)
- Access to an array of the same size and type as `Fin` to receive the redistributed weight function over the adapted mesh (e.g., `Fout(:,:,:)`)

SAM uses various internal (private) modules located in `HEAD/src/`. All of these modules need to be compiled and linked while compiling the solver code. For performing adaptation, the user only needs to understand the publicly accessible subroutines which are described in Sec. C.3.

Apart from the interfacing subroutine, the user must make sure that there exists a namelist called `SAM_control.nml` in the same folder from where the main (solver or interfacing code) executable is executed. Using this namelist, one can set the various options for the mesh adaptation procedure. Details of various options and their default values are given in Sec. C.4.

After obtaining the SAM code base, following (or similar) steps can be taken to create the library.

```
./bootstrap
mkdir OPT
cd OPT/
../configure --prefix=‘pwd‘ FC=/home/roycfd/Software/gcc/gcc.4.8.3/bin/gfortran
make
```

To compile the interfacing or solver code with the SAM library:

```
cd ..
cd examples/
cd Example_2D/
gfortran -I ../../OPT/src/ example_2d.f95 ../../OPT/src/libsam.a
./a.out
```

C.3 Using SAM's public subroutines

Various publicly accessible subroutines are described in this section.

C.3.1 SAM_Set_Problem

Usage (1D): `CALL SAM_Set_Problem(imax)`
Description: Sets problem size, i.e. the size of the mesh array as `nmesh = imax`
and the size of the weight function array as
`nwt = imax` if `SAM_weight` is set as '`mesh_node`'
`nwt = imax-1` if `SAM_weight` is set as '`cell_center`'

Usage (2D): `CALL SAM_Set_Problem(imax, jmax)`
Description: Sets the array sizes for array variables holding mesh as:
`nxmesh = imax`
`nymesh = jmax`
and for weight function as:
`nxwt = imax`
`nywt = jmax`

Note: Currently 2D routines are only coded for mesh-node located weight functions.

C.3.2 SAM_Adapt_Mesh

Usage (1D): `CALL SAM_Adapt_Mesh(xin, Fin, xout, Fout)`
Description: Adapts the current mesh (`xin`) so as to equidistribute the given weight function (`Fin`) on the adapted mesh.

The input function (`Fin`) must be positive and non-zero or else the code returns to the calling procedure without any action. Consider using the utility `SAM_Make_Positive_NonZero`.

Usage (2D): `CALL SAM_Adapt_Mesh(xin, Fin, xout, Fout)`
Description: Same as for 1D description

C.3.3 SAM_Make_Positive_NonZero

Usage (1D): `CALL SAM_Make_Positive_NonZero(Fin)`
Description: Takes a discrete function (`Fin`) and makes it positive and non-zero.

If any of the function values are near-zero (i.e., less than 1.0e-8) then all the values are shifted by `0.001*MAXVAL(Fin)`.

Usage (2D): `CALL SAM_Make_Positive_NonZero(Fin)`
Description: Same as for 1D description

C.3.4 SAM_Smoothen

Usage (1D): `CALL SAM_Smoothen(F, max_pass)`
Description: Smoothes the function (`F`) using an elliptic smoother (low-pass filtering) for `max_pass` number of smoothening runs. Other features may be added later.

Hint: `5<max_pass<20` is a good value for usual cases. Too much smoothening will not perform much adaptation. Too little smoothening may cause adaptation process to not converge well. If you wish to set this value via `SAM_control.nml` (i.e, `SAM_smoothening_passes`) then `max_pass` must be provided a negative (-1, -2 etc.) value here.

Usage (1D): `CALL SAM_Smoothen(F, max_pass)`
Description: Same as for 1D description

C.3.5 SAM_Write_Data

Usage (1D): `CALL SAM_Write_Data(xin, Fin, fn, filename, zn)`
Description: Simply writes data to a tecplot readable file as:
(1) Mesh size (`imax` etc.), zone number in the heading
(2) `x, y, z, Fin(x,y,z)`
Zone number and file. id number should be provided by the calling routine. The file name (which should be a `Character(len=20)` variable) should be provided by the calling routine. When multiple calls of `SAM_Write_Data` are made with the same file number, the file is simply appended.

Usage (2D): `CALL SAM_Write_Data(xin, Fin, fn, filename, zn)`
Description: Same as for 1D description

C.3.6 SAM_Function_Norms

Usage (1D): `CALL SAM_Function_Norms(fsize, F, fnorms)`
Description: Calculates L1, L2 and Linfinity norms for a function `F` of size `fsize` and stored in the 1x3 array `fnorms` to get norm results as:
`fnorms(1)` = L1 norm of `F`
`fnorms(2)` = L2 norm of `F`
`fnorms(3)` = Linf norm of `F`

Usage (2D): `CALL SAM_Function_Norms(fsizex, fsizey, F, fnorms)`
Description: Same as for 1D description

C.3.7 SAM_Calculate_Move_2D

(Only for 2D operations)

Usage (2D): `CALL SAM_Calculate_Move_2D(X1, X2, Move)`
Description: Returns an array, `Move`, containing the distances for each node between the input `X1` and `X2` meshes, i.e.,
`Move(i,j) = || X1(i,j) - X2(i,j) ||`

C.3.8 SAM_Equidistribution_Error

(Only for 1D and 2D cell-centered functions and `F` must be a cell-centered function)

Usage (1D): `CALL SAM_Equidistribution_Error(fsize, x, F,err_eqdbF)`
Description: Given an input array `F` of size `fsize` over a mesh `x` this returns an array `err_eqdbF` of the same size as `F` containing the relative error from the equidistributed value.

Usage (2D): Call `SAM_Equidistribution_Error(fsizex,fsizey,x,F,err_eqdbF)`
Description: Same as for 1D description.

C.4 Creating `SAM_control.nml`

A sample of SAM_control.nml could be found in the SAM code base. Various parameters that can be set in the control file are discussed in Table C.4. See the notes following the table for more details about these parameters.

Table C.1: Parameters and options in SAM

#	Parameter (Type)	Description	Options	[Default]
1	`SAM_Dimension (Char)`	Dimension of the adaptation problem	'1D', '2D'	['1D']
2	`SAM_Relocation (Char)`	Relocation method	'`regula_falsi`', '`spring_analogy`', '`center_of_mass`'	['`center_of_mass`']
3	`SAM_tolerance (Real)`	Tolerance for adaptive convergence		[1E-6]
4	`SAM_itermax (Integer)`	Maximum number of iterations within the adaptation cycle		[10000]
5	`SAM_urelax (Real)`	Under-relaxation factor between two inner adaptation iterations		[1.0]
6	`SAM_urelax_post (Real)`	Under-relaxation factor between two outer adaptation iterations		[1.0]
7	`SAM_history (Logical)`	`.TRUE.` to obtain adaptive history (i.e., adaptation convergence vs. number of inner iterations)	`.TRUE.`, `.FALSE.`	[`.FALSE.`]
8	`SAM_history_f (Integer)`	Frequency for outputting adaptive history		[20]
9	`SAM_Weight (Char)`	Location of the weight function	'`mesh_node`', '`cell_center`'	['`mesh_node`']

Continued on next page

Table C.1 – *Continued from previous page*

#	Parameter (Type)	Description	Options	[Default]
10	`SAM_fileout` `(Logical)`	`.TRUE.` to obtain inner adaptation data (i.e, weight function vs. current mesh nodes)	`.TRUE.`, `.FALSE.`	[`.FALSE.`]
11	`SAM_fileout_f` `(Integer)`	Frequency of inner adaptation data		[20]
12	`SAM_quadrature` `(Char)`	Quadrature for interpolation while mesh-movement	'`fvm_discrete`', '`fvm_linear`'	['`fvm_discrete`']
13	`SAM_smoothening_passes` `(Integer)`	Number of smoothening passes for function smoothening		[5]
14	`SAM_nwtfix (Integer)`	Number of boundary weight function variables to fix during smoothening		[1]

- For `SAM_Relocation`, the '`regula_falsi`' is an exact root-finding method and is only available for 1D functions. The 'center_of_mass' option is the least oscillatory of the three especially when the weight function changes with mesh movement.
- For `SAM_itermax`, when root-finding method is used for relocation, inner relocation iterations are not required. However, there are iterations within each root-finding call. Thus, for root-finding, `SAM_itermax` is used to change the maximum number of iterations within each each root-finding call.
- Inner iteration or inner cycle refers to a single mesh relocation. Inner iterations when allowed to be performed for `SAM_itermax` number of times or until `SAM_tolerance` is reached constitute a complete call to SAM for mesh adaptation, also known as the outer iteration. Furthermore, multiple outer iterations are usually needed for weight functions that vary drastically with mesh movement (e.g., a truncation error-based weight function).
- For `SAM_urelax`, exploring different values can help in reducing oscillations in the inner cycles. For small value of `SAM_urelax`, consider increasing `SAM_itermax`.

- `SAM_urelax_post` must usually be kept at 1.0 if full equidistribution is desired. However, this variable is very useful if the adaptation monitor (or weight function) is being created from a quantity that is highly mesh dependent (e.g., truncation error). In such case, small value for `SAM_urelax_post` must be set or equivalently (and more efficiently), less number of inner iterations should be made during each outer cycle.

- `SAM_quadrature` is only used for 1D adaptation with `SAM_weight`='`cell_center`'. In '`fvm_discrete`' option, cell-center value is assumed to be constant over the entire cell. In '`fvm_linear`' option, the cell-center values are joined by straight lines (linear interpolation) to create the function and special zero-order treatment is done at the boundaries.

- `SAM_nwtfix` is implemented for 1D adaptation only. If `SAM_nwtfix`=0, then smoothening occurs at the boundaries. If `SAM_nwtfix>=1`, then no smoothening occurs for `SAM_nwtfix` number of weight function variables at the boundaries.

Bibliography

[1] Aniruddha Choudhary and Christopher J. Roy. Structured mesh r-refinement using truncation error equidistribution for 1D and 2D Euler problems. In *Fluid Dynamics and Co-located Conferences*, pages –. American Institute of Aeronautics and Astronautics, June 2013. doi: 10.2514/6.2013-2444. URL `http://dx.doi.org/10.2514/6.2013-2444`.

[2] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press Cambridge, 2010.

[3] Christopher Roy. Strategies for driving mesh adaptation in CFD (invited). In *Aerospace Sciences Meetings*, pages –. American Institute of Aeronautics and Astronautics, January 2009. doi: 10.2514/6.2009-1302. URL `http://dx.doi.org/10.2514/6.2009-1302`.