IMPORTANCE OF CODE AND SOLUTION VERIFICATION IN CREDIBLE SIMULATIONS

A Dissertation

by

AARON MARTIN KRUEGER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yassin A. Hassan
Committee Members,	Rodolfo Vaghetto
	Maria D. King
	William H. Marlow
	Vincent A. Mousseau
Head of Department,	Michael Nastasi

December 2020

Major Subject: Nuclear Engineering

ABSTRACT

Current interest in verification, validation, and uncertainty quantification (VVUQ) has increased substantially over the past 30 years with increased awareness of potential inaccuracies of numerical simulations. This requires rigorous VVUQ analysis methods to correctly estimate the numeric and physics modeling uncertainties of numerical simulations for a given application. While methods exist that quantify these uncertainties, these methods are not the most rigorous in identifying code errors and estimating numerical and physical modeling uncertainties. The VVUQ methods presented in this dissertation describe three state-of-the-art improvements to current VVUQ methods: modified equation analysis method of manufactured solutions (MEAMMS) code verification, identifying and characterizing the "Asymptotic Point", and how this impacts validation and uncertainty quantification. MEAMMS code verification builds on the method of manufactured solutions (MMS) and the modified equation analysis (MEA) to identify code errors that are below, of the same order, or with certain implementations, higher than the numerical method. Previous code verification methods, such as MMS, do not identify these types of coding errors. Characterizing the asymptotic point for multiple manufactured solutions provides additional insight into how discretization error behaves for a variety of discretization sizes. This characterization is able to evaluate the performance of different discretization uncertainty methods inside the asymptotic range, near the asymptotic point, and outside the asymptotic range. Code and solution verification sensitivities show the importance of code and solution verification in validation and uncertainty quantification studies. By changing the amount of coding error and numerical uncertainty on a synthetic problem, the impact can be measured. This novel work aims at extending the field of VVUQ by developing tools and methodologies that improve the quality of computational

software's prediction capability.

DEDICATION

I would like to dedicate this work to my parents and sister. Without their constant love, support, encouragement, and sacrifices, I would not be the person I am today. I am truly blessed to call them family.

ACKNOWLEDGMENTS

I would like to thank Dr. Yassin Hassan for advising me throughout this process and the research opportunities he has provided me over the years. I would also like to thank Dr. Vincent Mousseau for mentoring me the past five years. Not only has he mentored me technically in V&V topics, but also in how to approach problems with wisdom and integrity. I would like to thank my other committee members, Dr. Maria King, Dr. Rodolfo Vaghetto, and Dr. William Marlow, for technical support and guidance throughout the project. Additionally, I would like to thank my colleagues at Sandia National Laboratories, especially Dr. Nathan Porter, Dr. Troy Haskin, and Lindsay Gilkey for their numerous technical and editorial comments throughout this work. I would like to thank my graduate school colleagues, Dr. Landon Brockmeyer, Dr. Jonathan Lai, John Mulloy, and Jesse Latimer, for staying up late with me too many times to finish school assignments or research. I know I will look back at these memories fondly. Lastly, I would like to thank the numerous friends and family that have supported me throughout my life, especially in graduate school. They have provided the necessary distraction, comfort, and guidance throughout my life, for which I am grateful.

CONTRIBUTORS AND FUNDING SOURCES

This work was supported by a dissertation committee consisting of Professor Yassin Hassan (advisor) and Professors Rodolfo Vaghetto and William Marlow of the Department of Nuclear Engineering, Professor Maria King of the Department of Biological and Agricultural Engineering, and Dr. Vincent Mousseau of Sandia National Laboratories.

All work conducted for the dissertation was completed by the student, under the advisement of Professor Yassin Hassan of the Department of Nuclear Engineering and Dr. Vincent Mousseau of Sandia National Laboratories.

This research was partially supported by the Consortium for Advanced Simulation of Light Water Reactors (CASL) (www.casl.gov), an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In the last few decades, the amount of computational power available to scientists and engineers has drastically increased the use of computational fluid dynamics (CFD) simulation and system analysis codes to solve nuclear engineering problems [1]. From accident analysis to design optimization to safety margin determination, simulations aim to improve the design and analysis of nuclear power plants. However, the credibility of simulation results has been a concern since the beginning of numerical simulations. As CFD simulations and system codes grow in complexity, the ability to determine the trustworthiness becomes even harder. As a result, a subset within the CFD and system code community has pushed for more rigorous methods to check if the equations in the software are solved correctly. This is referred to as verification. Additionally, validation evaluates if the equations being solved are correctly chosen for the problem [2].

Within verification lies two distinct types: code verification and solution or calculation verification. Code verification is "the process of determining that the numerical algorithms are correctly implemented in the computer code and of identifying errors in the software" [3] while solution verification is "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" [1]. Validation on the other hand determines if the numerical simulation result is "close enough" to experimental data. In addition to verification and validation, uncertainty quantification assesses the total uncertainty, which combines the uncertainties from the solution verification and validation process. Code verification, solution verification, validation, and uncertainty quantification all need tools and methodologies to be done correctly by the larger CFD and system code communities to produce credible simulations. This is the primary purpose of the verification and vali-

dation (V&V) community. By applying V&V tools and methodologies to nuclear power simulations, evidence is produced to provide simulation credibility. The next few sections look at the progression of V&V tools and methodologies to assess the current state-of-the-art as well as the limitations of these methods. It also describes the progression of nuclear power system codes and simulations in their use of V&V tools and methodologies. Then, new tools and methodologies are presented to improve the current state-of-the-art of the prediction capabilities of nuclear power plant codes and simulations.

1.1 Origins of Numerical Simulations

One of the first uses of using numerical techniques for solving engineering problems was by L. F. Richardson. He approximated differential equations using difference equations to solve the stress on a masonry dam [4]. Using the difference equations rather than the differential equations, solutions were able to be obtained on irregular geometries where exact solutions did not exist. He also developed the Richardson extrapolation, which is an algorithm to determine the mesh independent solution of differential equations. Using the difference between the numerical solution on the finest grid and the coarsest grid, the leading local truncation error (LTE) was calculated and removed from the solution to increase the accuracy. L. R. Richardson et al. continued this work in [5]. In future sections, the Richardson extrapolation will be used to estimate the discretization error for more complicated simulations.

It is important to remember that this work was completed in 1911 and 1927, respectively. Since this was before the development of modern computers, numerical calculations had to be completed by hand, which was extremely prohibitive for large or complicated calculations. As computers begin to develop and more widely used, the use of numerical simulations to solve engineering problems also become more widely used, including in the nuclear engineering field.

1.2 Development of V&V Concepts

With the development of more complex software to model complex physical phenomena, ensuring the software was coded well enough to be useful in prediction became a new field of research. Methodologies first started comparing to physical experiments, which evolved into the topic of validation. With numerical schemes such as the second-order temporal scheme developed by John Crank and Phyllis Nicolson [6] being applied to complex engineering problems, ensuring the code was bug free became a complicated task in and of itself, which evolved into code verification and solution verification. Unfortunately, there is no true system-level software test to ensure the software is working correctly [1] and it is extremely difficult to assess the impact of code bugs [7]. Software quality assurance (SQA) tools and methodologies, in addition to code and solution verification, have been developed to reduce the number of code bugs.

1.2.1 Software Quality Assurance

Software developers and V&V developers have different hierarchies for SQA and code verification. Software developers typically include code verification as part of SQA while Oberkampf and Roy consider as SQA as part of code verification [1] with numerical algorithm testing as separate from SQA, but included as the other part of code verification. Based on previous software development experience [8, 9, 10], the numerical algorithm testing has not been a priority and is often neglected. Therefore, this dissertation separates SQA and code verification as two distinct activities, but both are necessary in the development of computational software. Separating SQA from code verification also has the ability to rank the maturity of both activities separately, which will help improve the implementation of numerical algorithm testing. SQA involves the testing and documentation of the code that does not impact the numerical algorithm, which is the same definition as Oberkampf and Roy [1]. Code verification is based on what Oberkampf and Roy refer to

numerical algorithm testing [1].

SQA includes regularly timed static testing and dynamic testing [11], unit testing [12], component testing [13], regression testing, and code coverage testing. Just as important is documenting the results of the testing, as well as theory manuals to state the theoretical performance and limitations of the software being developed. These SQA processes reduce the probability of code bugs as well as reduce the time of finding code bugs when the testing indicates the presence of a code bug. Similar processes are in code verification, but focus on the testing of the numerical algorithm rather than the code as a whole.

1.2.2 Code Verification

Code verification activity methodologies have been developed extensively, but can be difficult to implement, especially for commercial codes. The methodologies can be classified into different levels of rigor. From Oberkampf and Roy [1], the following list are code verification activities with increasing rigor:

- Simple Tests
- Code-to-Code Comparison
- Convergence Tests
- Order of Accuracy Tests

While different applications and customer requirements dictate which level of rigor should be applied, the most rigorous code verification type should be applied to nuclear power plant safety codes. This is because of the large consequence of software prediction failure, such as bridge, reactor, or airplane failure. Below are the outlines of different levels of code verification rigor. These levels will be referred to later in this chapter to classify the current rigor of code verification in different thermal hydraulic nuclear power

plant simulation software. It should be noted that all of these tests can be useful, regardless of rigor level. Rigor level is determined by which test is the final test the code passes.

1.2.2.1 Simple Tests

Simple tests are tests based on intuition, such as symmetry problems or conservation tests. Symmetry tests are problems where the boundary conditions are set such that the exact solution is unknown, but the behavior should be symmetric. This allows for testing if the software can predict the symmetry. The downside of this method is that the numerical solution can't be compared to the exact solution because it's unknown. Conservation tests are problems where the flux into the domain should match the flux out of the domain. As with the symmetry problems, the numerical solution can't be compared to the exact solution. In addition, these types of tests have poor code coverage because only a small subset of boundary conditions and physics can be tested.

1.2.2.2 Code-to-Code Comparison

Code-to-code comparisons can be more rigorous than simple tests, but only for one situation. The only situation where code-to-code comparisons work as a final code verification test is when the physics is modeled exactly the same and the numerical scheme and grid are exactly the same. This tends only to apply to different versions of the same code, such as testing the solutions between two different revisions. All other situations tend to have slight physics modeling or numerical schemes that impact the comparison that suggest a code bug when it most likely indicates a non-equal comparison. The next code verification activities all require access to an exact solution, which improves the rigor of the testing immensely.

1.2.2.3 Convergence Tests

Convergence tests assess whether uniform refinement of the grid converges to the exact solution. This proves that the numerical solution is consistent with the physical model. As recommended by Oberkampf and Roy, convergence tests should be the minimum requirement for rigorous code verification [1].

1.2.2.4 Order of Accuracy Tests

The order of accuracy test calculates the rate (also referred to as order) of convergence and compares it to the theoretical order of convergence. This test requires the exact solution to calculate the most accurate observed order of accuracy based on absolute error. Additionally, relative error is able to be used, but the fidelity of the observed order of accuracy is reduced. This method has the ability to identify bugs that are one order lower than the theoretical order of this method. While this is the state-of-the-art code verification method, it does not identify code bugs that are of the same order as the theoretical order of the method [14].

1.2.2.5 Analytical Solutions

Historically, analytic solutions have been used to quantify error in the numerical method since the beginning of numerical analysis. The obvious downside is that analytical solutions to problems that exercise all portions of the code are difficult to impossible to derive. While still useful, a different approach to obtain an exact solution provides an even more useful tool.

1.2.2.6 Manufactured Solutions

The current most rigorous method to obtain an exact solution is called the method of manufactured solutions (MMS). According to Roache [15], the first known use of MMS for code verification purposes was Steinberg and Roache [16], but the first mention of

"manufactured solution" was Oberkampf et al. [17]. This method uses a proposed solution to the physical model and then derive a source term that makes the proposed solution true. This allows for extensive testing of all aspects of the numerical method. Once a convincing amount of code verification evidence has been documented, the coding error uncertainty is minimized. While this method is the most rigorous, it can not be applied to "black box" codes, since it requires intimate knowledge of the code.

1.2.3 Solution Verification

After code verification activities have been completed, solution verification activities can begin. Solution verification is estimating the numerical error due to discretizing the physical models within the software for problems of interest when an exact solution is unknown. This includes iteration error, round off error, discretization error, and sampling error for stochastic methods. While there are many components to solution verification, if code verification is completed properly, only discretization error and sampling error (for stochastic codes) need to be explicitly measured. Round off error can be minimized using computer hardware and software that uses 16 digits of precision. Iteration error can be minimized by setting the iterative convergence tolerance to be a few magnitudes lower than discretization error, and sampling error can be minimized by sampling enough to reduce the impact by a few magnitudes lower than discretization error. This may be an iterative process in and of itself, but this ensures a small impact of other sources of error.

To assess discretization error, an understanding of discretizing the physical models in both space and time is necessary. The physical models are discretized using a Taylor series. Since accounting for all high ordered terms is computationally expensive, terms are truncated and referred to as local truncation error (LTE). Discretization error is based off the transported LTE terms in the Taylor series [18]. This idea was used by Richardson to improve the accuracy of the solution. It was also used to assess the stability of numerical

methods. Hirt first used the modified equation analysis to assess the LTE and whether or not the LTE would be a stabilizing error or a destabilizing error [19]. The first instance of using MEA to assess accuracy was completed by Cyrus and Fulton [20]. Even though this work was completed in 1968, most researchers were not concerned with accessing discretization error until the late 1980s and early 1990s [2]. The prominent work from this era were post-processing methods based on the Richardson extrapolation. The standard solution verification method developed in this era was the grid convergence index (GCI) by Roache. Recently, other post-processing solution verification methods have been developed such as the factors of safety for Richardson extrapolation method [21], least squares GCI method [22, 23], robust verification analysis [24], and StREEQ [25].

1.2.3.1 Grid Convergence Index

GCI was developed to provide a uniform was of reporting numerical uncertainty [2]. This was done by estimating the error using the difference between Richardson extrapolated quantity of interest (QoI) and the QoI calculated on the finest grid. This estimated error is then multiplied by a safety factor to account for uncertainties in this estimation. Since the original derivation used Eq. 1.1 as the error model, uncertainty in the C_1 or p coefficients are increased if the local truncation error does not match this model.

$$Error = C_1 h^p \tag{1.1}$$

The error model is not appropriate when there are other dominant terms or the solution is not in the asymptotic region. The factor of safety is bifurcated into two different safety factors based on the behavior of the numerical scheme. When the observed order of accuracy is "close enough" to the theoretical order of accuracy $F_s = 1.25$, but when it's far from the theoretical order, $F_s = 3$. Roache never provided a metric for the definition of "close enough" and left it up to the user, but Oberkampf and Roy [1] suggested to use 10% as the metric for "close enough". The rest of the solution verification methods are based on

the error model in Eq. 1.1, but aim to improve the method of accounting for uncertainty.

1.2.3.2 Factors of Safety for Richardson Extrapolation

The factors of safety for Richardson extrapolation method aims to improve the factor of safety function from a sharp transition between 1.25 and 3 to a smooth transition where uncertainty is increased as the difference in observed and theoretical order of accuracy is increased. These increases are based on tuned coefficients from a database of problems across different physics.

1.2.3.3 Least Squares GCI

The least squares GCI (LSGCI) method is an improvement on GCI by informing the uncertainty model with more QoI solutions on different grids using a least squares calculation. Additional terms are included in the uncertainty calculation. Just like in the factors of safety method, LSGCI increases uncertainty as the difference in observed and theoretical order of accuracy is increased, but used different terms to increase the uncertainty.

1.2.3.4 Robust Verification Analysis

The robust verification method builds on the logic of the least squares GCI method, but uses many L-norms to calculate multiple optimized coefficients. The method then uses robust statistics to determine the median optimized result and determines the uncertainty based on the spread of the L-norm results. The additional robustness has the potential to work better than other methods outside the asymptotic range [26].

1.2.4 Validation

Another question of physical correctness was researched, which was the beginning of validation studies. Early validation studies were qualitative in nature, which included the use of the viewgraph norm. As the subject of validation evolved, quantitative measures of model form uncertainty was developed. To help identify specific models that have

large model form uncertainty, separate effects validation was developed to test individual physics to make sure the models perform adequately. To ensure that the models perform together well, integral testing is performed to understand the total model form uncertainty.

1.2.4.1 Separate Effects Validation

Separate effects testing allows for smaller, less complicated physics comparisons to test a physics model to identify initial points of large model form uncertainty and to reduce the likelihood of compensating model form error. To help with identifying the important physics that need to be tested, a phenomena identification and ranking table (PIRT) should be created to identify the important physics for the problem of interest [1]. PIRTs were originally developed for code scaling, applicability, and uncertainty evaluations of nuclear power plant's accident scenarios [27]. Using such a tool helps identify the important physics of the problem and the experiments that need to be completed for validation comparison. Additionally, a qualitative PIRT (QPIRT) was developed to better quantify the dominant physics [28, 29].

1.2.4.2 Integral Testing Validation

Integral testing is more application based with the combination of many single effects physics. An important part of integral testing is to collect experimental data that covers a large range of application space. This allows for a better validation evaluation by evaluating the model form error in more of the application space. This is a more difficult test for the multiple physics models utilized in the validation assessment when separate effects coefficients are determined first.

1.2.4.3 Validation Experiment Assessment

Through decades of validation assessments, it became clear that not all experiments are created equal. Oberkampf and Smith developed criteria for quality of validation exper-

iments for CFD applications [30]. While meeting the most rigorous criteria is difficult to achieve, it is meant to be an ideal set of criteria meant to evolve with experimental measurement development. It also specifies the data necessary to collect and provide to the computational physicist. When validation testing begins, it is important that the experimenter works with the computational physicist because computational physicist's model is ultimately being tested.

1.2.5 Uncertainty Quantification

While uncertainty quantification can be computed using many different methods, the most important part is to identify and quantify all sources of uncertainty. While identifying the dominating sources of uncertainty is difficult, not quantifying key sources of uncertainty can be devastating because of the potential consequences of engineering systems. Once all sources of large uncertainties are identified, there are two main things to consider: 1) is the uncertainty aleatory or epistemic? and 2) should local methods or global methods be used? This will define the process of computationally quantifying uncertainty.

1.2.5.1 Aleatory Uncertainty

Aleatory uncertainty is defined as the uncertainty due to inherent randomness [1]. Aleatory uncertainty in parameters tends to be associated with uncertainty in a systems initial conditions and boundary conditions. An example of aleatory uncertainty is the geometric dimension uncertainty associated with a manufacturing process. This type of uncertainty lends itself to being easily modeled as a probability distribution.

1.2.5.2 Epistemic Uncertainty

Epistemic uncertainty is defined as the uncertainty due to a lack of knowledge. Within epistemic uncertainty, there are two types: recognizable uncertainty and blind uncertainty. Recognizable uncertainty is an epistemic uncertainty for which there was a conscious deci-

sion to characterize its uncertainty to a certain precision. Blind uncertainty is an epistemic uncertainty for which there wasn't a conscious decision to characterize its uncertainty to a certain precision. Since blind uncertainty is impossible to characterize individually, it is important to reduce this uncertainty as much as possible. Since epistemic uncertainty is from a lack of knowledge, it doesn't behave well when its modeled as a probability distribution. Because of this, epistemic uncertainty is usually quantified by bounds (i.e. the value of a parameter is between value A and value B). While epistemic uncertainty is more difficult to model, it is important to reduce and characterize this type of uncertainty to have reliable predictions.

1.2.5.3 Local Method

Local uncertainty quantification is useful when the uncertainty around a specific point is desired. By perturbation input or values within the code, the impact can be quantified. This is very similar to a sensitivity analysis, but relies on the actual range of reasonable parameters. This is an important distinction because the most sensitive parameters are not necessarily the most significant. The other downside of this method is that information near the nominal value can be understood. To get a better understanding of the uncertainty for values far from nominal, a global uncertainty method should be employed.

1.2.5.4 Global Method

Global uncertainty quantification is useful when the uncertainty of the entire range of values is desired. This is done by randomly sampling a probability distribution to select the parameter input or values within the code. This uncertainty is propagated through the code to the outputs. This is completed using Monte Carlo or Latin hypercube sampling. While the uncertainty throughout the range of values is determined, the computational cost is significantly more than local uncertainty quantification, especially if the code is computationally intensive such as a CFD simulation.

1.3 Use of Codes for Nuclear Power Plants

Computational software has been used for nuclear power plant design, safety analysis, and licensing since the 1960s. With the increase in computational power, the use and complexity of the software has increased throughout the 1960s, 1970s, and 1980s. During this time, the software was tested and compared with experimental data to determine if the simulation was providing correct results. While validation activity was being completed, code and solution verification was not. There were two reasons for this: 1) the field of V&V was in it's infancy and the understanding of which components of the code needed to be tested were still being determined and 2) the Nuclear Regulatory Commission's (NRC) definitions for V&V terms and concepts are currently 5-10 years behind the field of V&V [31], even though it lead the field in the 1970s and 1980s. Therefore, most code focused on validation rather than both verification and validation to determine if the code was working as intended. Below is a detailed history of V&V activities for prominent thermal hydraulic simulation software.

1.3.1 RELAP

RELAP is one of the first thermal-hydraulics codes and has been developed by Idaho National Laboratory for the best estimate of transient simulations of light water reactor coolant systems during accident scenarios [32]. First starting out as a code to simulate small break loss of coolant accidents for pressurized water reactors using three control volumes, RELAP5-3D claims to have the ability to also simulate large break loss of coolant accidents, operator transients, boiling water reactors, test reactors, molten salt reactors, liquid metal reactors, high temperature gas-cooled reactors, supercritical fluid reactors, and more recently small modular reactors and traveling wave reactors in three dimensions [33]. In addition, the plan to develop RELAP-7 is underway and plans to utilize modern V&V methodologies [34].

Code verification activity for RELAP-5/3D is unclear with only mentioning roughly 2400 test case comparisons to ensure that the code is performing as intended [33]. This could either fall under the category of SQA or code verification depending on the test type. In addition, solution verification assessments have not been widespread in the literature, leading to assume that there is a general lack of estimating numerical uncertainty in RELAP calculations. This appears to change for RELAP-7 with the use of manufactured solutions to assess spatial and temporal convergence [34] although only one simple test for verifying the order of accuracy has been completed [35].

Some validation has been completed since the releases of RELAP-5/3D with comparisons to 54 test cases [33]. These test cases include phenomenological, separate effects, and integral testing to assess validation. Volume 3 of the RELAP user manual documents the results of the tests. RELAP-7 plans on assessing the same validation tests as RELAP-5/3D and includes test specifications that require comparisons to single-phase, two-phase, heat conduction, and component tests [34].

UQ has been completed using the code scaling, applicability, and uncertainty (CSAU) method ever since the NRC revised the acceptance criteria of emergency core cooling systems (ECCS) [36]. This approach changes the historical conservative analysis with a best estimate approach, which requires analysis of the uncertainty in the best estimate to assess safety margin. CSAU was first used on a large break loss of coolant accident in 1990 [37].

1.3.2 MELCOR

MELCOR is an engineering thermal-hydraulic software that models the progression of severe accidents in light water reactors. It is developed by Sandia National Laboratories for the NRC. It claims to have the ability to model the thermal hydraulic behavior in the reactor coolant system, reactor cavity, containment, and confinement buildings. Specific

phenomena include core heatup, core degradation, core relocation, core-concrete attack, hydrogen production, hydrogen transport, and hydrogen combustion, and fission product release and transport [38].

MELCOR was first developed in the early 1980's and since the beginning, V&V has been included in the development of the code. Code verification was completed using analytical solutions, but only checked for convergence by comparing the MELCOR output with the analytic solution [39, 40]. While this was state-of-the-art at the time, it continues to be the only code verification completed. Modern solution verification tools and methodologies have also not rigorously been utilized to determine the numerical uncertainty. One of the only instances of assessing the numerical uncertainty was performed by shuffling the order of the flow paths, which caused variance in the solution [41, 42]. This uncertainty was characterized, but uncertainty due to spatial or temporal discretization have not been historically assessed.

Validation for light water reactors has been an ongoing development since the early development of the code and is the strongest component of V&V [8, 9, 10, 39]. MELCOR has two types of validation studies: separate effects and integral testing with an emphasis on covering the most important physics [9]. This helps determine the adequacy of different physics models.

Early versions of uncertainty quantification of MELCOR was determining numerical sensitivities of the QoI based on some perturbation. Currently, analyses with large number of runs with different inputs to determine error bars have been applied to the NRC State-of-the-Art Reactor Consequence Analyses (SOARCA) study. SOARCA used best estimate with uncertainty quantification to present results rather than a PRA-style conservation-based results [43]. This focused on realism rather than the worse-case scenario. This uncertainty quantification using MELCOR allowed for accessing internal parameters, input, and control sequences to perform a complete Monte Carlo uncertainty analysis. The

Monte Carlo Analysis provided the average outcome as well as uncertainty bands to make informed decisions.

1.3.3 GOTHIC

GOTHIC is a general thermal-hydraulic analysis software package developed and maintained by Zachry Nuclear Engineering (formerly Numerical Applications Incorporated) for the Electric Power Research Institute for nuclear power plant design, safety, and licensing. It claims to solves the conservation of mass, momentum, and energy in multi-phase with the ability for multiple directional fidelity (1D, 2D, or 3D) or lumped parameter analysis.

Since GOTHIC is a proprietary code, the code verification can not be assessed readily, but evidence of solution verification is found in some instances [44].

Validation has been documented to included separate effects testing [45]. The separate effects tests include condensation heat transfer on a vertical flat plate and evaporative heat transfer from a hot pool to a dry atmosphere. Since GOTHIC is a proprietary code, integral testing documentation was not readily available in the literature.

While sensitivity studies for GOTHIC have been found [46], there is a lack of a full uncertainty quantification. Again, this could be due to the proprietary nature of the code.

1.3.4 Computational Fluid Dynamics

Computational fluid dynamics (CFD) has been a new addition to thermal-hydraulic codes that increases the fidelity of thermal-fluid predictions. Nuclear initiatives such as the consortium for advanced simulation of light water reactors (CASL) have utilized CFD codes such as STAR-CCM+ [47] to perform high fidelity simulations to train faster running lower fidelity codes [48, 49]. With the increase in fidelity comes an increase in grid complexity and therefore an increase in V&V complexity. An example of an increase in V&V complexity involves verifying the numerical order on unstructured grids. Since the stencil used won't be able to perfectly cancel out lower ordered terms, determining the

theoretical order of the numerical method is non-trivial. This makes completing code and solution verification complicated because code verification needs the theoretical order of accuracy and solution verification uses the theoretical order of accuracy to determine the discretization uncertainty. Nevertheless, V&V is an important step in ensuring credible predictions for engineering problems.

Code and solution verification for CFD has been largely developed by high consequence applications such as the aerospace and nuclear weapons fields. It was these fields that pushed for the development of capable V&V tools and methodologies to be applied on codes developed by the national labs or the national aeronautics and space administration (NASA). Commercial codes, however, have historically been design to be robust rather than accurate, which means that the code verification applied to codes like STAR-CCM+ have only begun to use tools such as MMS to verify the correct implementation of the numerical method. Most of STAR-CCM+'s tests focus on qualitative convergence to an analytical solution rather than the more rigorous order of convergence test. Solution verification is more widely applied than thermal-hydraulic system codes and is even a requirement for some journals [50]. While solution verification is frequently used, the problem of which solution verification method to use becomes a difficult question. In addition, most solution verification methods are based on the Richardson extrapolation and require being in the asymptotic region to perform correctly. Since CFD resolves or models more physics, they tend to have a larger range of scales. This makes achieving the asymptotic region extremely difficult.

Validation assessments for commercial codes like STAR-CCM+ are typically included within the testing requirements to ensure that physical models are matching certain benchmark experiments. These tests include both separate effects and integral testing. In addition, for large engineering projects, additional experiments are completed to validation models for specific conditions. Unfortunately, these tests are typically not completed with

validation in mind and do not provide the CFD user with data that completely tests the physical models. The other issue with validation assessments are from a lack of code and solution verification before the validation assessment is completed. This allows for code bugs in the numerical scheme and large numerical uncertainty to exist and impact the predictive capability of the code.

Uncertainty quantification in CFD has become a hot area of research due to international benchmark problems such as the Generic Mixing Experiment (GEMIX) [51]. This benchmark is part of the Organization for Economic Co-operation and Development (OECD) CFD initiative for nuclear reactor safety, which is also called OECD/CFD4NRS. GEMIX is an experiment that measured the mixing of two fluid layers in a channel and provided CFD users with not only mean values of the experiment, but also the variance. Three open cases and one blind case provided the ability to assess model form uncertainty in the three open cases and input and numerical uncertainty of the blind case to make predictions for the blind case. Participants assessed the uncertainty using different methods and provided the results to the benchmark organizers. As an example, one of the participants used the ASME V&V 20 [52] methodology to perform the uncertainty quantification [53]. This type of study informed the organizers of which uncertainty quantification methods, tools, and techniques provided the best prediction and uncertainty results. Studies such as these aim to not only improve the state-of-the-art for uncertainty quantification, but also improve the state-of-the-practice for uncertainty quantification in CFD applications.

1.3.5 Applying V&V Concepts to Nuclear Power Plant Codes and Simulations

As stated above, the state-of-the-practice of V&V for nuclear power plant codes and simulations is behind the state-of-the-art tools and methodologies in the V&V field. A need for a more rigorous way to test the numerical method in thermal-hydraulic and CFD codes is necessary to ensure accurate results. Additionally, solution verification needs

to be applied, especially in thermal-hydraulics codes. The solution verification methods used should be robust and provide realistic, conservative numerical uncertainty estimates inside, near, and outside the asymptotic range. While validation and uncertainty quantification has historically been widely applied to nuclear power plant analysis, the code and solution verification has not been rigorously applied. Therefore, work should be completed to show how much of an impact code and solution verification play in the ability to complete a validation and uncertainty quantification analysis. The next sections describe tools, methodologies, and tests for better analyzing simulations. These sections not only make computational predictions for nuclear power plants more accurate and with less uncertainty, but shows the impact of what happens when verification has not been properly executed.

1.4 Code Verification Improvements

Using a manufactured solution (MS), all system variable are equated to arbitrary smooth functions and substituted into the partial differential equation. This provides a source term that makes the original function for a particular system variable correct. This can be used as an exact solution when an analytical solution is not available. Since the exact solution is known, the numerical method implemented in the code can be tested to ensure the correct implementation of the numerical method and reduce the number of possible code bugs. This is completed by running the simulation at different discretization sizes and observing the rate of convergence. If the observed rate of convergence matches well with the theoretical order of accuracy, the probability of code bugs is low, but if it doesn't match, code bugs might exist in the code or the simulations were not in the asymptotic range. The problem with MMS is that it doesn't test all aspects of the numerical scheme, such as the theoretical local truncation error. Therefore, new methods are needed to identify these type of code bugs as well as lower order code bugs.

Chapter 3 will present a method to not only verify the correct order of accuracy, but also verify the correct amount of discretization error. This is completed by combining the modified equation analysis (MEA) and MMS, which we refer to as MEAMMS (pronounced memes). MEAMMS is able to calculate the discretization error in two ways: 1) leading-order cancellation method called leading order MEAMMS and 2) high-order cancellation method called higher order MEAMMS. The high-order cancellation method has already been tested as part of the author's previous work [14], but for a much simpler problem.

1.5 Solution Verification Improvements

There are many methods of solution verification that estimate the numerical uncertainty of a simulation, but the most popular type is based on the Richardson extrapolation. The Richardson extrapolation used the leading term of the truncated Taylor series to estimate the numerical error. This works well when the leading term is "large" compared to all other higher order terms and the numerical uncertainty is well behaved. When simulations have well behaved discretization uncertainty, the simulation is generally referred to be in the asymptotic range [1]. While the definition is straightforward, determining where the asymptotic region starts is not well defined or characterized. What is even less characterized is how well different solution verification methods perform inside, near, and well outside the start of the asymptotic range or as referred to in this study as the *asymptotic point*.

Chapter 4 will quantitatively characterize the performance of GCI for calculations inside, near, and outside the asymptotic range. The problem is complex enough to be realistic and uses the non-linear form of the coupled shallow water equations.

1.6 Importance of Code and Solution Verification

Validation and uncertainty quantification can be completed using a number of methods and tools. In addition, evaluating validation experiments and estimating and characterizing the largest uncertainties for a system is crucial for good predictive capabilities. However, if proper SQA, code verification, and solution verification has not been completed prior to the validation quantification assessment, the predictive capabilities will be extensively reduced. This has a negative impact on the extrapolation data with no evidence that it happened.

Chapter 5 will quantitatively show the importance of SQA, code verification, and solution verification in the validation and uncertainty quantification assessment. To show this, a simulation with code bugs and large numerical uncertainties will be completed in addition to a simulation with no code bugs and small numerical uncertainties. Then, a calibration and validation assessment will be completed for both simulations. The predictive capabilities of each simulation will be quantitatively assessed by predicting the QoI outside the validation state space. This assessment shows that the simulation with code bugs and large numerical uncertainties will be inferior to the simulation with no code bugs and small numerical uncertainties.

2. BACKGROUND EQUATIONS

Before the MEAMMS methodology, solution verification analysis, or the VVUQ work is presented, a basic description of the equations is described. A new code is developed to fully show off the MEAMMS method as well as present the asymptotic point calculation for solution verification analysis. This code solves the discretized shallow water equations, which provides the opportunity to test MEAMMS and the asymptotic point calculation on a non-linear coupled equation set. Below is the derivation of the shallow water equations for use in the code as well as a modified shallow water equation to implement viscous effects.

2.1 Shallow Water Equations

The shallow water equations are a simplified version of the multiphase Euler equations by assuming that the pressure's equation of state is a function of height, gravity, and a constant density and that the change in energy as a function of time is zero. For more information about the shallow water equations, see pg. 35 of [54]. The multiphase isothermal Euler equations are shown in Eq. 2.1 through Eq. 2.4.

$$\frac{\partial \alpha\rho}{\partial t} + \frac{\partial \alpha\rho u}{\partial x} = 0 \tag{2.1}$$

$$\frac{\partial \alpha\rho u}{\partial t} + \frac{\partial \alpha\rho u^2}{\partial x} + \alpha\frac{\partial P}{\partial x} = 0 \tag{2.2}$$

$$\frac{\partial e}{\partial t} = 0 \tag{2.3}$$

$$P = \alpha\rho \mathbf{g} h \tag{2.4}$$

Starting with the one-dimensional multiphase Euler equations, the chain rule is applied so that partial derivatives are a function of just one variable, which is shown in Eq. 2.5 through Eq. 2.7.

$$\rho\frac{\partial\alpha}{\partial t}+\alpha\frac{\partial\rho}{\partial t}+\alpha\rho\frac{\partial u}{\partial x}+\rho u\frac{\partial\alpha}{\partial x}+\alpha u\frac{\partial\rho}{\partial x}=0 \tag{2.5}$$

$$\alpha\rho\frac{\partial u}{\partial t}+\alpha u\frac{\partial\rho}{\partial t}+\rho u\frac{\partial\alpha}{\partial t}+\alpha u^2\frac{\partial\rho}{\partial x}+2\alpha\rho u\frac{\partial u}{\partial x}+\rho u^2\frac{\partial\alpha}{\partial x}+\alpha\frac{\partial P}{\partial x}=0 \tag{2.6}$$

$$\frac{\partial e}{\partial t}=0 \tag{2.7}$$

Next, density is assumed to be the constant ρ_0, so derivatives of ρ in space and time are zero. Therefore, Eq. 2.5 and Eq. 2.6 are simplified to Eq. 2.8 and Eq. 2.9.

$$\rho_0\frac{\partial\alpha}{\partial t}+\alpha\cancelto{0}{\frac{\partial\rho}{\partial t}}+\alpha\rho_0\frac{\partial u}{\partial x}+\rho_0 u\frac{\partial\alpha}{\partial x}+\alpha u\cancelto{0}{\frac{\partial\rho}{\partial x}}=0 \tag{2.8}$$

$$\alpha\rho_0\frac{\partial u}{\partial t}+\alpha u\cancelto{0}{\frac{\partial\rho}{\partial t}}+\rho_0 u\frac{\partial\alpha}{\partial t}+\alpha u^2\cancelto{0}{\frac{\partial\rho}{\partial x}}+2\alpha\rho_0 u\frac{\partial u}{\partial x}+\rho_0 u^2\frac{\partial\alpha}{\partial x}+\alpha\frac{\partial P}{\partial x}=0 \tag{2.9}$$

Next, the equation of state for pressure (Eq. 2.4) is substituted into the pressure term of Eq. 2.9 and shown in Eq. 2.10.

$$\alpha\rho_0\frac{\partial u}{\partial t}+\rho_0 u\frac{\partial\alpha}{\partial t}+2\alpha\rho_0 u\frac{\partial u}{\partial x}+\rho_0 u^2\frac{\partial\alpha}{\partial x}+\alpha\rho_0\mathbf{g}\frac{\partial h}{\partial x}=0 \tag{2.10}$$

By factoring a velocity out of specific terms in the momentum equation, the mass equation can be separated out of the momentum equation and because the mass equation equals zero, the factored out terms are canceled out of the momentum equation (see Eq. 2.11). Additionally, all terms in the mass include a constant density and all terms in the momentum equation include a constant density times the void fraction. This can be further simplified out of the mass and momentum equations, which are shown in Eq. 2.12 through Eq. 2.13.

$$\alpha\rho_0\frac{\partial u}{\partial t}+\alpha\rho_0 u\frac{\partial u}{\partial x}+\alpha\rho_0\mathbf{g}\frac{\partial\alpha}{\partial x}+u\cancelto{0}{\left(\rho_0\frac{\partial\alpha}{\partial t}+\alpha\rho_0\frac{\partial u}{\partial x}+\rho_0 u^2\frac{\partial\alpha}{\partial x}\right)}=0 \tag{2.11}$$

$$\frac{\partial \alpha}{\partial t} + \alpha \frac{\partial u}{\partial x} + u \frac{\partial \alpha}{\partial x} = 0 \tag{2.12}$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \mathbf{g} \frac{\partial h}{\partial x} = 0 \tag{2.13}$$

To convert α, a non-dimentional quantity, in Eq. 2.12 and Eq. 2.13 into a dimensional one, a total height of the domain, l is used to convert the normalized α in terms of a dimentional one. This relationship is shown in Eq. 2.14. Equation 2.14 is then substituted into Eq. 2.12 and Eq. 2.13 to form Eq. 2.15 and Eq. 2.16.

$$\alpha = \frac{h}{l} \tag{2.14}$$

$$\frac{1}{l}\frac{\partial h}{\partial t} + \frac{h}{l}\frac{\partial u}{\partial x} + \frac{u}{l}\frac{\partial h}{\partial x} = 0 \tag{2.15}$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \mathbf{g} \frac{\partial h}{\partial x} = 0 \tag{2.16}$$

Since each term in Eq. 2.15 and Eq. 2.16 include $\frac{1}{l}$, each equation is multiplied by l on each side. The resulting equations are the shallow water equations, which is shown in Eq. 2.17 and Eq. 2.18.

$$\frac{\partial h}{\partial t} + h \frac{\partial u}{\partial x} + u \frac{\partial h}{\partial x} = 0 \tag{2.17}$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \mathbf{g} \frac{\partial h}{\partial x} = 0 \tag{2.18}$$

2.2 Shallow Water Equations with Radionuclide Transport

For the last portion of the dissertation, the code needs to model atmospheric transport of fission products, C. The atmospheric transport equations are the shallow water equations with the fission product distribution modeled using an advection and diffusion equation. The shallow water equations are

$$\frac{\partial h}{\partial t} + h \frac{\partial u}{\partial x} + u \frac{\partial h}{\partial x} = 0 \text{ and} \tag{2.19}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \mathbf{g}\frac{\partial h}{\partial x} = 0, \tag{2.20}$$

are coupled to the fission product equation,

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} + k\frac{\partial^2 C}{\partial x^2} = 0, \tag{2.21}$$

through velocity, u, in the advection term. The fission products, C, is measured as ${}^{atoms}/_{m^3}$. In addition to the temporal and advective terms, there's a diffusive term to model the turbulent diffusion of the fission products. Using this set of equations represents all the components necessary for studies such as dose assessment studies at site boundaries. These studies require models that are calibrated and validated, so it is an ideal equation set to study the components of the VVUQ framework.

3. MEAMMS

Verification is the process of checking that the equations in the software are programmed correctly. Previous work has developed the MMS to verify the order of accuracy of codes, but MMS only identifies coding errors that are one order lower than the order of the numerical method, which leaves the potential to impact the solution. Therefore, a more rigorous method needs to be developed to identify coding errors that are of the same order as the numerical method. There has been significant work completed by the author to show the feasibility of creating a more rigorous code verification method [14], but the previous work does not address non-linear coupled equation problems. In this section, the development of MEAMMS is presented, including why this method addresses coding errors that are of the same order as the numerical method, as well as applying the method to a non-linear coupled equation set.

3.1 Verification

MMS is based on the "p" verification method, which compares the theoretical order of accuracy with the observed order of accuracy. At least two simulations with different levels of discretization size are needed. Using the exact solution provided my MMS, the exact error is calculated by

$$\varepsilon_h = f_h - f_{exact} \tag{3.1}$$

where ε_h is the exact error, h is the characteristic discretization size, f_h is the QoI on the h^{th} grid, and f_{exact} is the exact solution of the QoI. Using the Richardson extrapolation [4, 5], the assumed form of the error, which is based on the leading order local truncation term, is shown in

$$\varepsilon_h = g_p h^p + \mathcal{O}\left(h^{p+1}\right) \tag{3.2}$$

where g_p is the p^{th} order coefficient, p is the order of the numerical method, and $\mathcal{O}\left(h^{p+1}\right)$ represents the higher-order terms. When the solution is in the asymptotic region, the higher-order terms can be neglected. The resulting equation is shown in

$$\varepsilon_h = g_p h^{p_{obs}}. \tag{3.3}$$

To calculate the grid-independant solution, a second solution is made by coarsening the discretization size by two. The resulting equation that describes this error is shown in

$$\varepsilon_{2h} = g_p \left(2h\right)^{p_{obs}}. \tag{3.4}$$

Combining section 3.3 and section 3.4 yields

$$p_{obs} = \frac{\ln\left(\frac{\varepsilon_{2h}}{\varepsilon_h}\right)}{\ln\left(2\right)}, \tag{3.5}$$

Now that the observed order of accuracy is calculated, the theoretical order of accuracy (p_{th}) is determined. Using MEA, the leading LTE term can be calculated. The order of this term defines the theoretical order of accuracy. Since the order of accuracy is quite sensitive to coding errors, comparing the observed order of accuracy to the theoretical order of accuracy can highlight these errors. Oberkampf and Roy [1] have suggested a tolerance within 10% identifies that the numerical scheme does not have coding errors that MMS can highlight. It should be noted that the behavior of the observed order of accuracy is a function of the leading order LTE dominance compared to the higher order terms. This means that the solution needs to be within the asymptotic range.

3.1.1 MMS

As stated above, MMS requires an exact solution. An exact solution can be from either an analytic solution or a manufactured solution when an analytical solution is not available. Since there are many equation configurations in a code that do not have an analytical solution, manufactured solutions are valuable to ensure that the these equations are implemented correctly. There are two steps to calculate a manufactured solution. The

first is to choose a function that represents the exact solution, $\phi(x,t)$. The second step is to substitute the exact solution into the PDE to determine the resulting right-hand-side of the equation, which is shown by

$$\mathcal{L}(\phi(x,t)) = Q(x,t), \tag{3.6}$$

where $\mathcal{L}(\phi)$ is the nonlinear operator, $\phi(x,t)$ is the manufactured solution, and $Q(x,t)$ is the source term. It should be noted that for MMS to work, the code needs to be able to import source terms. Now that an exact solution is available, the observed order of accuracy can be calculated.

3.1.2 MEA

Truncation error results from the terms dropped from a numerical solution when approximating a PDE. One way of calculating the LTE is using MEA. MEA was originally developed for stability calculations [19], but has also been used in accuracy calculations [20]. In this study, MEA is used to calculate the theoretical LTE for comparison to the LTE produced by the code. This process more thoroughly tests the performance of the numerical scheme.

3.1.3 MEAMMS Development

MEAMMS was developed by the author in [55], which gives a comprehensive derivation of MEAMMS. The summary of the derivations is that when the theoretical leading LTE is subtracted from the code, the observed order of accuracy should increase. When this does not happen, it means that there is a coding error in the code.

3.2 Demonstration Case

Before solving the shallow water equations numerically, an exact solution to the equations need to be derived. Using MMS, the functional form of u and h are known. MMS uses the philosophy of starting from the end and working backwards to find the exact so-

lution. By allowing the user to set the functional form of u and h, source terms to the shallow water equations can be derived, which force the exact solution to be the user specified function. Using Eq. 2.17 and Eq. 2.18, source terms are included on the right side of the shallow water equations, which are shown in Eq. 3.7 and Eq. 3.8.

$$\frac{\partial h}{\partial t} + h\frac{\partial u}{\partial x} + u\frac{\partial h}{\partial x} = Q_{Mass} \tag{3.7}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \mathbf{g}\frac{\partial h}{\partial x} = Q_{Mom} \tag{3.8}$$

3.2.1 Code Implementation

The code was built as a coupling of two separate code languages for fast an easy solving of the discrete equations while easily calculating higher order Taylor series terms. This lead to the coupling of two Mathematica scripts with a Python wrapper. The first Mathematica script implemented eight numerical schemes to solve the shallow water equations. The numerical schemes are all based on the second order in time and space Crank-Nicolson central difference scheme with the addition or subtraction of diffusion in either time or space. Using input flags, a first-order diffusion term is added, subtracted, or zeroed out. The second Mathematica script calculates the LTE symbolically for each of the different numerical scheme as well as calculate the MMS exact solutions and MMS source terms. The Python wrapper then interprets the numerical scheme equations and solves the residual form of the shallow water equations. It also interprets the LTE, exact solutions, and MMS source terms. This framework allows for the numerical and exact solutions to be calculated and compared to ensure the proper implementation of the numerical scheme and the asymptotic point for solution verification analysis.

3.2.1.1 Grid Method

Before the numerical scheme is derived, the implied grid and time step orientation should be explained. To avoid "checkerboarding," the conservation of mass state variables

are computed at a separate location than the conservation of momentum state variables. This is apposed to co-located mass and momentum equations, which has the downside of having oscillating solutions between the even and odd grid. The staggered grid shown in Figure 3.1 provides a visualization of the scalar (mass) and vector (momentum) cells and the location of the state variables relative to other state variables. This orientation is the basis of the derivation of the numerical scheme and the calculation of LTE.

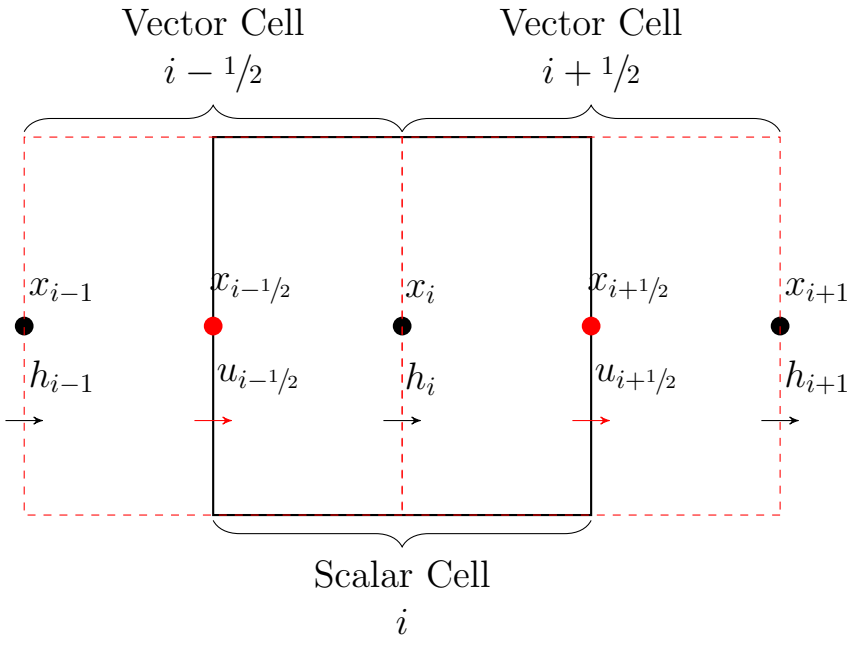

Figure 3.1: Visualization of Staggered Scalar and Momentum Grid

3.2.1.2 Numerical Scheme Implementation

As stated before, the numerical scheme is based on the Crank-Nicolson central difference scheme with the ability to add to subtract a first-order diffusion term in space, time, or both. The discretization scheme is split up into multiple term to simplify the derivation. To start, the $u\frac{\partial h}{\partial x}$ advection term in the mass equation was discretized as a central difference steady state term plus a first-order diffusion term to have the possibility of having a downwind or upwind discretization scheme, which is shown in Eq. 3.9. Since the $h\frac{\partial u}{\partial x}$

already has the necessary terms located at the correct location to be second order, there is not an additional diffusion term added, which is shown in Eq. 3.10. These terms were then substituted into the transient equation, which is shown in Eq. 3.11. The transient equation is second order in time, but has the ability to add diffusion is space. The discrete equation also includes the LTE term to ensure the PDE and discrete equations match exactly up to the order of the LTE.

$$Steady_{Mass_1} = V\left(\frac{u_{i+\frac{1}{2}} + u_{i-\frac{1}{2}}}{2}\right)\left[\left(\frac{h_{i+1} - h_{i-1}}{2\Delta x}\right) + A\left(\frac{h_{i+1} - 2h_i + h_{i-1}}{2\Delta x}\right)\right] \tag{3.9}$$

$$Steady_{Mass_2} = Vh_i\left(\frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x}\right) \tag{3.10}$$

$$\begin{aligned}&\frac{h_i^{n+1} - h_i^n}{\Delta t} + \frac{1}{2}\left(Steady_{Mass_1}^{n+1} + Steady_{Mass_1}^{n}\right)\\ &+ \frac{B}{2}\left(Steady_{Mass_1}^{n+1} - Steady_{Mass_1}^{n}\right) + \frac{1}{2}\left(Steady_{Mass_2}^{n+1} + Steady_{Mass_2}^{n}\right)\\ &+ \frac{B}{2}\left(Steady_{Mass_2}^{n+1} - Steady_{Mass_2}^{n}\right) + LTE_{Mass} = Q_{Mass}\end{aligned} \tag{3.11}$$

The momentum equation is discretized in a similar manner in Eq. 3.12 through Eq. 3.14.

$$Steady_{Mom_1} = Vu_{i+\frac{1}{2}}\left[\left(\frac{u_{i+\frac{3}{2}} - u_{i-\frac{1}{2}}}{2\Delta x}\right) + A\left(\frac{u_{i+\frac{3}{2}} - 2u_{i+\frac{1}{2}} + u_{i-\frac{1}{2}}}{2\Delta x}\right)\right] \tag{3.12}$$

$$Steady_{Mom_2} = V\mathbf{g}\left(\frac{h_{i+1} - h_i}{\Delta x}\right) \tag{3.13}$$

$$\begin{aligned}&\frac{u_{i+\frac{1}{2}}^{n+1} - u_{i+\frac{1}{2}}^{n}}{\Delta t} + \frac{1}{2}\left(Steady_{Mom_1}^{n+1} + Steady_{Mom_1}^{n}\right)\\ &+ \frac{B}{2}\left(Steady_{Mom_1}^{n+1} - Steady_{Mom_1}^{n}\right) + \frac{1}{2}\left(Steady_{Mom_2}^{n+1} + Steady_{Mom_2}^{n}\right)\\ &+ \frac{B}{2}\left(Steady_{Mom_2}^{n+1} - Steady_{Mom_2}^{n}\right) + LTE_{Mom} = Q_{Mom}\end{aligned} \tag{3.14}$$

To visualize the different possible numerical scheme and the stability of each scheme, Figure 3.2 shows the schemes in terms of the diffusion addition and subtraction in space and time.

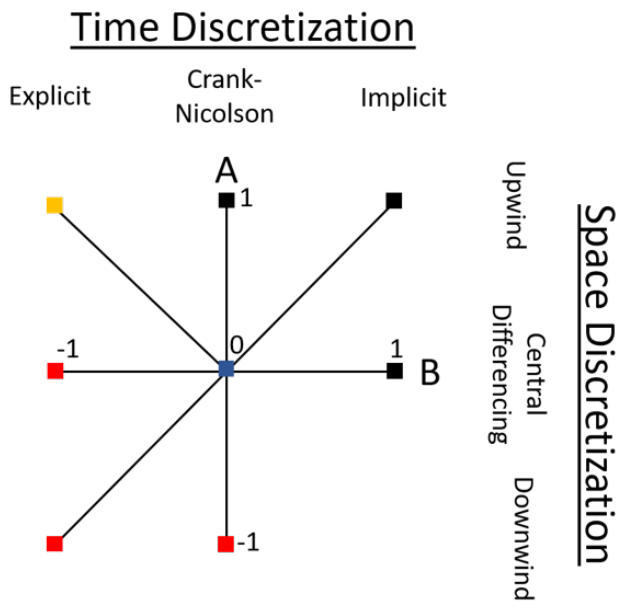

Figure 3.2: Visualization of the Eight Numerical Schemes Implimented

3.2.2 Newton's Method

To demonstrate how the non-linear terms were formulated, an example derivation is performed for the implicit upwind scheme (A=1 and B=1). Equation 3.15 and Eq. 3.16 are the implicit upwind schemes for the mass and momentum equations.

$$V\left(\frac{h_i^{n+1}-h_i^n}{\Delta t}\right)+V\left(\frac{u_{i+\frac{1}{2}}^{n+1}+u_{i-\frac{1}{2}}^{n+1}}{2}\right)\left(\frac{h_i^{n+1}-h_{i-1}^{n+1}}{\Delta x}\right) + V\left(h_i^{n+1}\right)\left(\frac{u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n+1}}{\Delta x}\right)+LTE_{Mass}=Q_{Mass} \tag{3.15}$$

$$V\left(\frac{u_{i+\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}}{\Delta t}\right)+V\left(u_{i+\frac{1}{2}}^{n+1}\right)\left(\frac{u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n+1}}{\Delta x}\right) + Vg\left(\frac{h_{i+1}^{n+1}-h_i^{n+1}}{\Delta x}\right)+LTE_{Mom}=Q_{Mom} \tag{3.16}$$

Newton's method uses a first-order Taylor series to make its linear approximation in

the state variable. Since it is a linear solver, multiple iteration updates of h^k and u^k are necessary for nonlinear problems between each time step to resolve nonlinearities.

$$\delta h = h^{n+1} - h^k \tag{3.17}$$

$$\delta u = u^{n+1} - u^k \tag{3.18}$$

Incorporating Eq. 3.17 and Eq. 3.18 into Eq. 3.9 and Eq. 3.10, assuming second-order terms are negligible, and simplifying:

$$\begin{aligned}
&\delta h_i \left[\frac{V}{\Delta t} + \frac{V}{\Delta x}\left(\frac{3u^k_{i+1/2}}{2} - \frac{u^k_{i-1/2}}{2}\right)\right] - \delta h_{i-1}\left[\frac{V}{2\Delta x}\left(u^k_{i+1/2} + u^k_{i-1/2}\right)\right] \\
&+\delta u_{i+1/2}\left[\frac{V}{\Delta x}\left(\frac{3h^k_i}{2} - \frac{h^k_{i-1}}{2}\right)\right] + \delta u_{i-1/2}\left[\frac{V}{\Delta x}\left(\frac{3h^k_i}{2} - \frac{h^k_{i-1}}{2}\right)\right] = VQ_{Mass} - VLTE_{Mass} \\
&- V\left(\frac{h^k_i - h^n_i}{\Delta t}\right) - \frac{V}{2}\left(u^k_{i+1/2} + u^k_{i-1/2}\right)\left(\frac{h^k_i - h^k_{i-1}}{\Delta x}\right) - Vh^k_i\left(\frac{u^k_{i+1/2} - u^k_{i-1/2}}{\Delta x}\right)
\end{aligned} \tag{3.19}$$

and

$$\begin{aligned}
&\delta u_{i+1/2}\left[\frac{V}{\Delta t} + \frac{V}{\Delta x}\left(2u^k_{i+1/2} - u^k_{i-1/2}\right)\right] - \delta u_{i-1/2}\left[\left(\frac{V}{\Delta x}\right)\left(u^k_{i+1/2}\right)\right] \\
&+ \delta h_{i+1}\left[\left(\frac{V\mathbf{g}}{\Delta x}\right)\right] - \delta h_i\left[\left(\frac{V\mathbf{g}}{\Delta x}\right)\right] = VQ_{Mom} - VLTE_{Mom} \\
&- V\left(\frac{u^k_{i+1/2} - u^n_{i+1/2}}{\Delta t}\right) - Vu^k_{i+1/2}\left(\frac{u^k_{i+1/2} - u^k_{i-1/2}}{\Delta x}\right) - V\mathbf{g}\left(\frac{h^k_{i+1} - h^k_i}{\Delta x}\right).
\end{aligned} \tag{3.20}$$

Eq. 3.19 and Eq. 3.20 can then be represented in matrix form, which is shown in

$$J\delta X = -res(X) \tag{3.21}$$

where

$$J_{i,j} = \frac{\partial res_j(X)}{\partial X_i}. \tag{3.22}$$

The Jacobian matrix (J) is then multiplied by δX. This process updates in the state vector and the terms on the right are the residuals. Once the equations are posed in this form, we have a simple linear system to solve for the state vector update, δX.

The state vector is then updated

$$X^{k+1} = X^k + \delta X, \tag{3.23}$$

until

$$max\left(X^k\right) < \mathcal{E}_{tol_{Nonlinear}}. \tag{3.24}$$

Once the nonlinear tolerance is met, X^{n+1} is set to be X^{k+1}. This completes the timestep.

3.2.3 Manufactured Solutions Source Terms

It is important to note that some functions of u and h are more useful than others. Since the source terms are based on derivatives of u and h, functions that allow for easy calculation of derivatives are desired. In the example for ASME V&V 20's sample calculation, trigonometric functions are used, such as sine and cosine [52]. Additionally, adding coefficients to the functions allow for the user to set the length scales and time scales of the problem. By setting the timestep and cell size small relative to the length and time scales, the numerical problem is within the asymptotic region. Equation 3.25 and Eq. 3.26 are the chosen functional forms of u and h for this particular study.

$$u = u_1 + u_2 \cos\left(k_u x + \omega_u t\right) \tag{3.25}$$

$$h = h_1 + h_2 \sin\left(k_h x + \omega_h t\right) \tag{3.26}$$

Now that the functional form of u and h are selected, the derivatives that are included in Eq. 3.7 and Eq. 3.8 need to be calculated. The derivatives for u and h in space and time are shown in Eq. 3.27 through Eq. 3.30.

$$\frac{\partial u}{\partial x} = -u_2 k_u \sin\left(k_u x + \omega_u t\right) \tag{3.27}$$

$$\frac{\partial u}{\partial t} = -u_2 \omega_u \sin\left(k_u x + \omega_u t\right) \tag{3.28}$$

$$\frac{\partial h}{\partial x} = h_2 k_h \cos\left(k_h x + \omega_h t\right) \tag{3.29}$$

$$\frac{\partial h}{\partial t} = h_2 \omega_h \cos\left(k_h x + \omega_h t\right) \tag{3.30}$$

Eq. 3.27 through Eq. 3.30 are then substituted back into Eq. 3.7 and Eq. 3.8 to calculate Q_{Mass} and Q_{Mom}, which is shown in Eq. 3.31 and Eq. 3.32.

$$\begin{aligned} Q_{Mass} = &\left(h_2 \omega_h \cos\left(k_h x + \omega_h t\right)\right) - \left(h_1 + h_2 \sin\left(k_h x + \omega_h t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{3.31}$$

$$\begin{aligned} Q_{Mom} = &-\left(u_2 \omega_u \sin\left(k_u x + \omega_u t\right)\right) - \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \mathbf{g}\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{3.32}$$

Now that Q_{Mass} and Q_{Mom} are known, Eq. 3.31 and Eq. 3.32 are substituted back into Eq. 3.7 and Eq. 3.8, which is shown in Eq. 3.33 and Eq. 3.34. This means that the exact solution to Eq. 3.33 and Eq. 3.34 are Eq. 3.25 and Eq. 3.26, which is useful for debugging and verifying the order of accuracy in numerical problems.

$$\begin{aligned} \frac{\partial h}{\partial t} + h\frac{\partial u}{\partial x} + u\frac{\partial h}{\partial x} = &\left(h_2 \omega_h \cos\left(k_h x + \omega_h t\right)\right) \\ &- \left(h_1 + h_2 \sin\left(k_h x + \omega_h t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{3.33}$$

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \mathbf{g}\frac{\partial h}{\partial x} = &-\left(u_2 \omega_u \sin\left(k_u x + \omega_u t\right)\right) \\ &- \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \mathbf{g}\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{3.34}$$

3.2.4 Local Truncation Error Calculation

MEAMMS combines the power of MEA with the MMS to calculate higher-order LTE terms numerically to ensure the numerical method is implemented correctly. Since the

MEA calculated the symbolic form of LTE and MMS provides the symbolic form of the exact solution, the high-order derivatives can be computed using the exact solution provided by the MMS. To show how this works, the LTE for implicit upwind scheme is derived below.

Using the implicit upwind numerical scheme shown in Eq. 3.15 and Eq. 3.16, the LTE_{Mass} and LTE_{Mom} is calculated by substituting Taylor series back into the equation, which is referred to MEA. For the conservation of mass equation, the Taylor series is set about h_i^{n+1} and u_i^{n+1}, so all terms not located at h_i^{n+1} and u_i^{n+1} need to be approximated to be at h_i^{n+1} and u_i^{n+1} using a Taylor series expansion. For the conservation of momentum equation, the Taylor series is set about $h_{i+\frac{1}{2}}^{n+1}$ and $u_{i+\frac{1}{2}}^{n+1}$, so all terms not located at $h_{i+\frac{1}{2}}^{n+1}$ and $u_{i+\frac{1}{2}}^{n+1}$ need to be approximated to be at $h_{i+\frac{1}{2}}^{n+1}$ and $u_{i+\frac{1}{2}}^{n+1}$ using a Taylor series expansion. Equation 3.35 through Eq. 3.38 are the Taylor series for the conservation of mass equation and Eq. 3.39 through Eq. 3.42 are the Taylor series for the conservation of momentum equation.

$$h_i^n = h_i^{n+1} - \Delta t \left.\frac{\partial h}{\partial t}\right|_i^{n+1} + \frac{\Delta t^2}{2} \left.\frac{\partial^2 h}{\partial t^2}\right|_i^{n+1} - \cdots + \frac{\Delta t^n}{n!} \left.\frac{\partial^n h}{\partial t^n}\right|_i^{n+1} \tag{3.35}$$

$$h_{i-1}^{n+1} = h_i^{n+1} - \Delta x \left.\frac{\partial h}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{2} \left.\frac{\partial^2 h}{\partial x^2}\right|_i^{n+1} - \cdots + \frac{\Delta x^n}{n!} \left.\frac{\partial^n h}{\partial x^n}\right|_i^{n+1} \tag{3.36}$$

$$u_{i+\frac{1}{2}}^{n+1} = u_i^{n+1} + \frac{\Delta x}{2} \left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8} \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!} \left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1} \tag{3.37}$$

$$u_{i-\frac{1}{2}}^{n+1} = u_i^{n+1} - \frac{\Delta x}{2} \left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8} \left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} - \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!} \left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1} \tag{3.38}$$

$$u_{i+\frac{1}{2}}^n = u_{i+\frac{1}{2}}^{n+1} - \Delta t \left.\frac{\partial u}{\partial t}\right|_{i+\frac{1}{2}}^{n+1} + \frac{\Delta t^2}{2} \left.\frac{\partial^2 u}{\partial t^2}\right|_{i+\frac{1}{2}}^{n+1} - \cdots + \frac{\Delta t^n}{n!} \left.\frac{\partial^n u}{\partial t^n}\right|_{i+\frac{1}{2}}^{n+1} \tag{3.39}$$

$$u_{i-\frac{1}{2}}^{n+1} = u_{i+\frac{1}{2}}^{n+1} - \Delta x \left.\frac{\partial u}{\partial x}\right|_{i+\frac{1}{2}}^{n+1} + \frac{\Delta x^2}{2} \left.\frac{\partial^2 u}{\partial x^2}\right|_{i+\frac{1}{2}}^{n+1} - \cdots + \frac{\Delta x^n}{n!} \left.\frac{\partial^n u}{\partial x^n}\right|_{i+\frac{1}{2}}^{n+1} \tag{3.40}$$

$$h_{i+1}^{n+1} = h_{i+\frac{1}{2}}^{n+1} + \frac{\Delta x}{2}\left.\frac{\partial h}{\partial x}\right|_{i+\frac{1}{2}}^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 h}{\partial x^2}\right|_{i+\frac{1}{2}}^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_{i+\frac{1}{2}}^{n+1} \tag{3.41}$$

$$h_{i}^{n+1} = h_{i+\frac{1}{2}}^{n+1} - \frac{\Delta x}{2}\left.\frac{\partial h}{\partial x}\right|_{i+\frac{1}{2}}^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 h}{\partial x^2}\right|_{i+\frac{1}{2}}^{n+1} - \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_{i+\frac{1}{2}}^{n+1} \tag{3.42}$$

Eq. 3.35 through Eq. 3.42 are substituted into Eq. 3.15 and Eq. 3.16, which is shown in Eq. 3.43 and Eq. 3.44 and simplified in Eq. 3.45 and Eq. 3.46.

$$\begin{aligned}
&V\left[\frac{h_i^{n+1} - \left(h_i^{n+1} - \Delta t\left.\frac{\partial h}{\partial t}\right|_i^{n+1} + \frac{\Delta t^2}{2}\left.\frac{\partial^2 h}{\partial t^2}\right|_i^{n+1} - \cdots + \frac{\Delta t^n}{n!}\left.\frac{\partial^n h}{\partial t^n}\right|_i^{n+1}\right)}{\Delta t}\right] \\
&+ \frac{V}{2}\left[\left(u_i^{n+1} + \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}\right)\right. \\
&\left.+ \left(u_i^{n+1} - \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} - \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}\right)\right]\left[\left(\frac{h_i^{n+1}}{\Delta x}\right)\right. \\
&\left.- \left(\frac{h_i^{n+1} - \Delta x\left.\frac{\partial h}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{2}\left.\frac{\partial^2 h}{\partial x^2}\right|_i^{n+1} - \cdots + \frac{\Delta x^n}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_i^{n+1}}{\Delta x}\right)\right] \\
&+ V\left(h_i^{n+1}\right)\left[\left(\frac{u_i^{n+1} + \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}}{\Delta x}\right)\right. \\
&\left.- \left(\frac{u_i^{n+1} - \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_i^{n+1} + \frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} - \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}}{\Delta x}\right)\right] + LTE_{Mass} = Q_{Mass}
\end{aligned} \tag{3.43}$$

$$
V\left(\frac{u_{i+\frac{1}{2}}^{n+1}-\left(u_{i+\frac{1}{2}}^{n+1}-\Delta t\ \frac{\partial u}{\partial t}\Big|_{i+\frac{1}{2}}^{n+1}+\frac{\Delta t^2}{2}\ \frac{\partial^2 u}{\partial t^2}\Big|_{i+\frac{1}{2}}^{n+1}-\cdots+\frac{\Delta t^n}{n!}\ \frac{\partial^n u}{\partial t^n}\Big|_{i+\frac{1}{2}}^{n+1}\right)}{\Delta t}\right)
$$

$$
+V\left(u_{i+\frac{1}{2}}^{n+1}\right)\left(\frac{u_{i+\frac{1}{2}}^{n+1}-\left(u_{i+\frac{1}{2}}^{n+1}-\Delta x\ \frac{\partial u}{\partial x}\Big|_{i+\frac{1}{2}}^{n+1}+\frac{\Delta x^2}{2}\ \frac{\partial^2 u}{\partial x^2}\Big|_{i+\frac{1}{2}}^{n+1}-\cdots+\frac{\Delta x^n}{n!}\ \frac{\partial^n u}{\partial x^n}\Big|_{i+\frac{1}{2}}^{n+1}\right)}{\Delta x}\right)
$$

$$
+V\mathbf{g}\left(\frac{\left(h_{i+\frac{1}{2}}^{n+1}+\frac{\Delta x}{2}\ \frac{\partial h}{\partial x}\Big|_{i+\frac{1}{2}}^{n+1}+\frac{\Delta x^2}{8}\ \frac{\partial^2 h}{\partial x^2}\Big|_{i+\frac{1}{2}}^{n+1}+\cdots+\frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\ \frac{\partial^n h}{\partial x^n}\Big|_{i+\frac{1}{2}}^{n+1}\right)}{\Delta x}\right.
$$

$$
\left.-\frac{\left(h_{i+\frac{1}{2}}^{n+1}-\frac{\Delta x}{2}\ \frac{\partial h}{\partial x}\Big|_{i+\frac{1}{2}}^{n+1}+\frac{\Delta x^2}{8}\ \frac{\partial^2 h}{\partial x^2}\Big|_{i+\frac{1}{2}}^{n+1}-\cdots+\frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\ \frac{\partial^n h}{\partial x^n}\Big|_{i+\frac{1}{2}}^{n+1}\right)}{\Delta x}\right)+LTE_{Mom}=Q_{Mom}
\tag{3.44}
$$

$$
\begin{aligned}
&\underbrace{V\left.\frac{\partial h}{\partial t}\right|_i^{n+1} + V u_i^{n+1}\left.\frac{\partial h}{\partial x}\right|_i^{n+1} + V h_i^{n+1}\left.\frac{\partial u}{\partial x}\right|_i^{n+1} - Q_{Mass}}_{PDE} \\
&+ \underbrace{V\left(-\frac{\Delta t}{2}\left.\frac{\partial^2 h}{\partial t^2}\right|_i^{n+1} + \cdots - \frac{\Delta t^{n-1}}{n!}\left.\frac{\partial^n h}{\partial t^n}\right|_i^{n+1}\right)}_{Temporal\ Truncation\ Error} \\
&+ \underbrace{V\left(u_i^{n+1}\right)\left(-\frac{\Delta x}{2}\left.\frac{\partial^2 h}{\partial x^2}\right|_i^{n+1} + \cdots - \frac{\Delta x^{n-1}}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_i^{n+1}\right)}_{Spatial\ Truncation\ Error} \\
&+ \underbrace{V\left(\frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \cdots + \frac{\Delta x^n}{2n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}\right)\left(\left.\frac{\partial h}{\partial x}\right|_i^{n+1}\right)}_{Spatial\ Truncation\ Error} \\
&+ \underbrace{V\left(\frac{\Delta x^2}{8}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^{n+1} + \cdots + \frac{\Delta x^n}{2n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}\right)}_{Spatial\ Truncation\ Error} \\
&* \underbrace{\left(-\frac{\Delta x}{2}\left.\frac{\partial^2 h}{\partial x^2}\right|_i^{n+1} + \cdots - \frac{\Delta x^{n-1}}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_i^{n+1}\right)}_{Spatial\ Truncation\ Error} \\
&+ \underbrace{V\left(h_i^{n+1}\right)\left(\frac{\Delta x^2}{6}\left.\frac{\partial^3 u}{\partial x^3}\right|_i^{n+1} + \cdots + \frac{\Delta x^{n-1}}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_i^{n+1}\right) + LTE_{Mass}}_{Spatial\ Truncation\ Error} \\
&= 0
\end{aligned}
\tag{3.45}
$$

$$
\begin{aligned}
&\underbrace{V\left.\frac{\partial u}{\partial t}\right|_{i+\frac{1}{2}}^{n+1} + V\left(u_{i+\frac{1}{2}}^{n+1}\right)\left.\frac{\partial u}{\partial x}\right|_{i+\frac{1}{2}}^{n+1} + V\mathbf{g}\left.\frac{\partial h}{\partial x}\right|_{i+\frac{1}{2}}^{n+1} - Q_{Mom}}_{PDE} \\
&\underbrace{-V\left(\left.\frac{\Delta t}{2}\frac{\partial^2 u}{\partial t^2}\right|_{i+\frac{1}{2}}^{n+1} + \cdots - \left.\frac{\Delta t^{n-1}}{n!}\frac{\partial^n u}{\partial t^n}\right|_{i+\frac{1}{2}}^{n+1}\right)}_{Temporal\ Truncation\ Error} \\
&\underbrace{-V\left(u_{i+\frac{1}{2}}^{n+1}\right)\left(\left.\frac{\Delta x}{2}\frac{\partial^2 u}{\partial x^2}\right|_{i+\frac{1}{2}}^{n+1} + \cdots - \left.\frac{\Delta x^{n-1}}{n!}\frac{\partial^n u}{\partial x^n}\right|_{i+\frac{1}{2}}^{n+1}\right)}_{Spatial\ Truncation\ Error} \\
&\underbrace{+V\mathbf{g}\left(\left.\frac{\Delta x^2}{24}\frac{\partial^3 h}{\partial x^3}\right|_{i+\frac{1}{2}}^{n+1} + \cdots + 2\frac{\left(\frac{\Delta x}{2}\right)^{n-1}}{n!}\left.\frac{\partial^n h}{\partial x^n}\right|_{i+\frac{1}{2}}^{n+1}\right) + LTE_{Mom}}_{Spatial\ Truncation\ Error} = 0
\end{aligned}
\tag{3.46}
$$

After subtracting the PDE from Eq. 3.45 and Eq. 3.46, the resulting LTE for mass and momentum are shown in Eq. 3.47 and Eq. 3.48.

$$
\begin{aligned}
LTE_{Mass} = &V\left(\left.\frac{\Delta t}{2}\frac{\partial^2 h}{\partial t^2}\right|_{i}^{n+1} - \cdots + \left.\frac{\Delta t^{n-1}}{n!}\frac{\partial^n h}{\partial t^n}\right|_{i}^{n+1}\right) \\
&+ V\left(u_i^{n+1}\right)\left(\left.\frac{\Delta x}{2}\frac{\partial^2 h}{\partial x^2}\right|_{i}^{n+1} - \cdots + \left.\frac{\Delta x^{n-1}}{n!}\frac{\partial^n h}{\partial x^n}\right|_{i}^{n+1}\right) \\
&- V\left(\left.\frac{\Delta x^2}{8}\frac{\partial^2 u}{\partial x^2}\right|_{i}^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_{i}^{n+1}\right)\left(\left.\frac{\partial h}{\partial x}\right|_{i}^{n+1}\right) \\
&+ V\left(\left.\frac{\Delta x^2}{8}\frac{\partial^2 u}{\partial x^2}\right|_{i}^{n+1} + \cdots + \frac{\left(\frac{\Delta x}{2}\right)^n}{n!}\left.\frac{\partial^n u}{\partial x^n}\right|_{i}^{n+1}\right)\left(\left.\frac{\Delta x}{2}\frac{\partial^2 h}{\partial x^2}\right|_{i}^{n+1} - \cdots\right. \\
&\left. + \left.\frac{\Delta x^{n-1}}{n!}\frac{\partial^n h}{\partial x^n}\right|_{i}^{n+1}\right) - V\left(h_i^{n+1}\right)\left(\left.\frac{\Delta x^2}{6}\frac{\partial^3 u}{\partial x^3}\right|_{i}^{n+1} + \cdots + \left.\frac{\Delta x^{n-1}}{n!}\frac{\partial^n u}{\partial x^n}\right|_{i}^{n+1}\right)
\end{aligned}
\tag{3.47}
$$

$$
\begin{aligned}
LTE_{Mom} = V & \left(\frac{\Delta t}{2} \left. \frac{\partial^2 u}{\partial t^2} \right|_{i+\frac{1}{2}}^{n+1} + \cdots - \frac{\Delta t^{n-1}}{n!} \left. \frac{\partial^n u}{\partial t^n} \right|_{i+\frac{1}{2}}^{n+1} \right) \\
& + V \left(u_{i+\frac{1}{2}}^{n+1} \right) \left(\frac{\Delta x}{2} \left. \frac{\partial^2 u}{\partial x^2} \right|_{i+\frac{1}{2}}^{n+1} + \cdots - \frac{\Delta x^{n-1}}{n!} \left. \frac{\partial^n u}{\partial x^n} \right|_{i+\frac{1}{2}}^{n+1} \right) \\
& - V \mathbf{g} \left(\frac{\Delta x^2}{24} \left. \frac{\partial^3 h}{\partial x^3} \right|_{i+\frac{1}{2}}^{n+1} + \cdots + 2 \frac{\left(\frac{\Delta x}{2}\right)^{n-1}}{n!} \left. \frac{\partial^n h}{\partial x^n} \right|_{i+\frac{1}{2}}^{n+1} \right)
\end{aligned}
\tag{3.48}
$$

3.3 Problem Setup

To show how MMS order of accuracy testing compares to MEAMMS testing, a test problem was set up using the numerical schemes in Figure 3.2. MMS and MEAMMS using the leading order was only able to test numerical methods listed in Table 3.1 as stable because both code verification methods requires a stable solution to test the order of accuracy. MEAMMS with higher order terms was able to test all numerical schemes and only requires one computation rather than two or more. Unstable methods do not converge to the exact solution as Δx approaches zero, while stable methods converge. Sometimes the stability is dependent on the mesh settings, as with the explicit upwind scheme. While A-stable cases are between stable and unstable, making the method unpredictable with regards to stability.

The one-dimensional domain is characterized by a length and constant cross-sectional area. Flow through the domain enters the left boundary and exits the right boundary. The time-dependent boundary conditions for the left and right boundaries are determined using ghost cells at the boundary. Each ghost cell value is set by the manufactured solution. The manufactured solution also provides the initial condition throughout the domain. The boundary conditions and the initial conditions provide enough information to track the height and the velocity of the wave through the domain. For the MMS test problem, the manufactured solution uses coefficients listed in Table 3.2. For the MEAMMS test

Table 3.1: Description of the Numerical Schemes Implemented Within the Code

Temporal Scheme	Spatial Scheme	B	A	Temporal Order	Spatial Order	Stability Condition
Explicit	Downwind	1	1	1	1	Unstable
	Upwind	1	-1	1	1	$U_1 \Delta t \, / \, \Delta x < 1$
	Central Difference	1	0	1	2	Unstable
Implicit	Downwind	-1	1	1	1	Unstable
	Upwind	-1	-1	1	1	Stable
	Central Difference	-1	0	1	2	Stable
Crank Nicolson	Downwind	0	1	2	1	Unstable
	Upwind	0	-1	2	1	Stable
	Central Difference	0	0	2	2	A-Stable

problems, the manufactured solution uses coefficients listed in Table 3.3

Table 3.2: MMS Calculation Setup

Quantity	Value
Domain Length (m)	6.28
End Time (s)	1.0
Cross-Sectional Area (m^2)	0.04
h_1	2.150
h_2	0.220
u_1	1.220
u_2	0.133
k_h	0.00112
k_u	0.00116
ω_h	0.00155
ω_u	0.00110
g	0.990

While refinement studies with exact solutions require only two grids, four grids (coarse through very fine) were chosen to observe that the order of accuracy was relatively stable and converging to the theoretical order of accuracy. Table 3.4 shows the cell and timestep

Table 3.3: MEAMMS Calculation Setup

Quantity	Value
Domain Length (m)	6.28
End Time (s)	1.0
Cross-Sectional Area (m^2)	0.04
h_1	2.150
h_2	0.220
u_1	1.220
u_2	0.133
k_h	0.112
k_u	0.116
ω_h	0.155
ω_u	0.110
g	0.990

sizes for each refinement level.

Table 3.4: Refinement Setup

Quantity	# of Cells	Δ x	# of Timesteps	Δ t
Coarse	10	0.6280	10	0.1000
Medium	20	0.3140	20	0.0500
Fine	40	0.1570	40	0.0250
Very Fine	80	0.0785	80	0.0125

While the Richardson extrapolation only requires two grids, another method to calculate the observed order of accuracy is to use a more complex error model, which requires more data to inform the error model. This method removes the assumption of the numerical method to converge to the exact PDE solution [56]. This is important because even if the numerical scheme is suppose to converge to the correct value, the incorrect implementation of the numerical method could converge to the incorrect solution. To model the constant error, the error model is updated to add a constant error term to the truncation

error in Eq. 3.2, which is shown in Eq. 3.49. The zeroth-order error term (ε_0) captures the zeroth-order error without impacting the p^{th} ordered term. Unlike the conventional MMS error model, the more complex error model helps the code developer to identify which term the error is impacting.

$$\varepsilon_h = \varepsilon_0 + g_p h^p + \mathcal{O}\left(h^{p+1}\right) \tag{3.49}$$

Using the error model in Eq. 3.49, the coefficients are solved by using the least squares optimization method used in [56].

3.4 MMS Order of Accuracy Test Results

The first code verification method to test the code was MMS with the simplified error model. MMS is only able to test for coding errors that impacts the order of accuracy one order lower than the theoretical order of the numerical method since it can't distinguish coding errors that don't impact the order of accuracy. Using a constant Courant, Friedrichs, and Lewy (CFL) number [57], the grid and timestep are refined to calculate the observed order of accuracy. Based on the results from the refinement study, the order of accuracy is listed in Table 3.5 and shown as the slope in Figure 3.3 for height and Figure 3.4 for velocity. These results show that the stable numerical scheme within the code passes the MMS order of accuracy test, which is the current state-of-the-art code verification testing.

Table 3.5: Order of Accuracy Calculation without LTE Source Term

Numerical Scheme	p_{1_h}	p_{2_h}	p_{1_u}	p_{2_u}	Theoretical Order
Explicit Upwind	0.945	0.949	0.961	0.981	1.000
Implicit Upwind	0.973	0.950	0.955	0.975	1.000
Implicit Central Difference	1.002	0.963	0.967	0.985	1.000
Crank Nicolson Upwind	0.959	0.948	0.958	0.9785	1.000
Crank Nicolson Central Difference	1.956	1.979	2.020	2.047	2.000

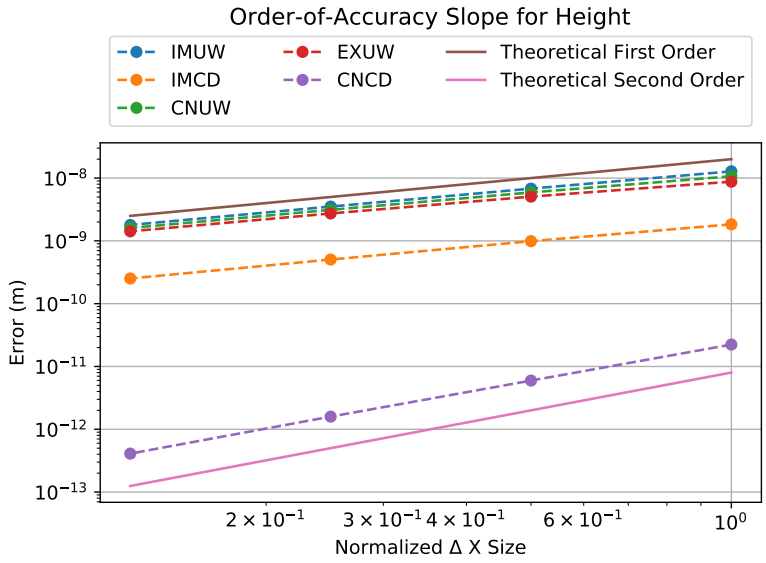

Figure 3.3: MMS Order of Accuracy Slope Results for the Implemented Stable Numerical Schemes (Height)

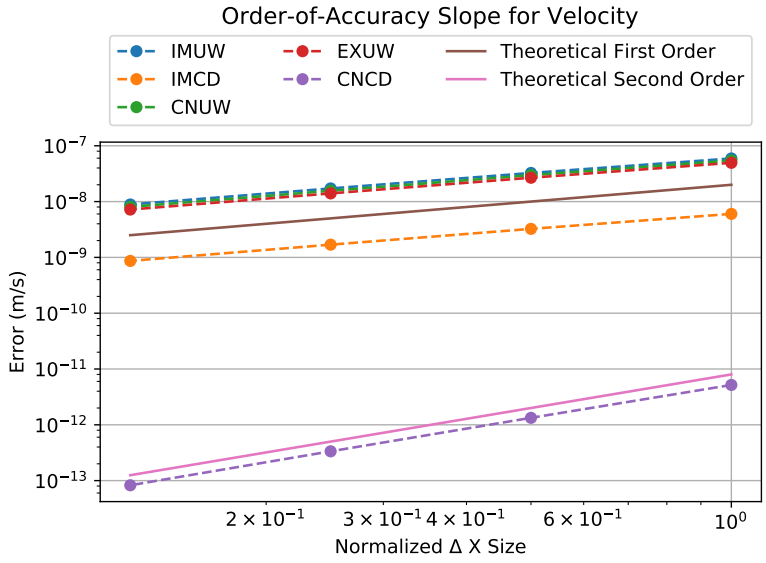

Figure 3.4: MMS Order of Accuracy Results for the Implemented Stable Numerical Schemes (Velocity)

3.5 MEAMMS Test Results With Leading Order Terms

Now that the observed order of accuracy matches the theoretical order of accuracy, the code has been verified to have no coding errors that impact the order of accuracy. Since there is a probability of coding errors of the same order as the numerical method, the leading LTE is added as a source term. If there are no zeroth-order coding errors or coding errors of the same order as the numerical method, the LTE will cancel the leading truncation error produced by the code and increase the order of accuracy. Table 3.6 shows that the observed order of accuracy has increased by one for all first-order schemes and increased by two for the second order scheme because all odd LTE terms are zero. These results are confirmed in Figure 3.5 for height and Figure 3.6 for velocity by the discretization error slope matching the theoretical discretization slope.

Table 3.6: MEAMMS Order of Accuracy Calculation with Leading LTE Source Term

Numerical Scheme	p_{1_h}	p_{2_h}	p_{1_u}	p_{2_u}	Theoretical Order
Explicit Upwind	1.960	1.978	1.865	1.873	2.000
Implicit Upwind	1.954	1.972	1.886	1.872	2.000
Implicit Central Difference	1.964	1.983	1.931	1.942	2.000
Crank Nicolson Upwind	1.957	1.975	1.844	1.852	2.000
Crank Nicolson Central Difference	3.967	3.986	3.924	3.951	4.000

3.6 MEAMMS Test Results With Higher Order Terms

The MEAMMS with higher order terms builds on the testing of the MEAMMS leading order. So instead of canceling out the leading truncation error, all truncation error is canceled to lower than round-off error. While calculating enough LTE terms to cancel the truncation error to this level requires work, there is the benefit of ensuring no higher

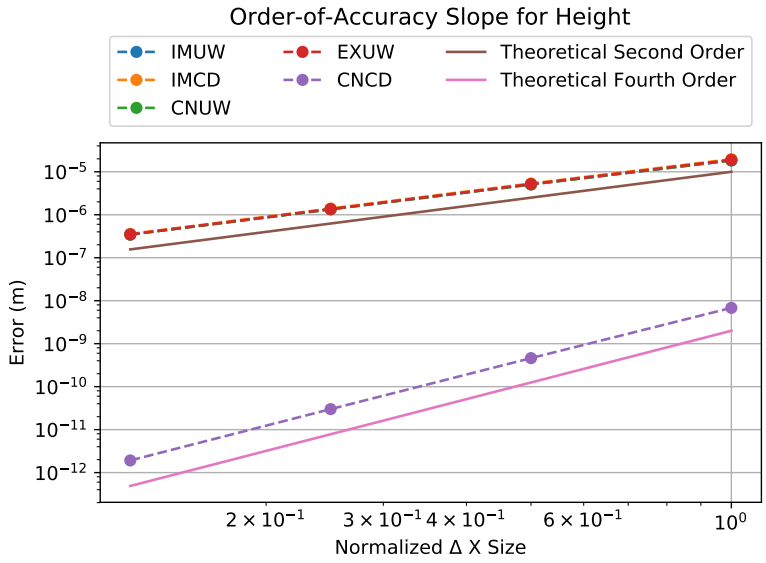

Figure 3.5: MEAMMS Order of Accuracy Slope Results for the Implemented Stable Numerical Schemes (Height)

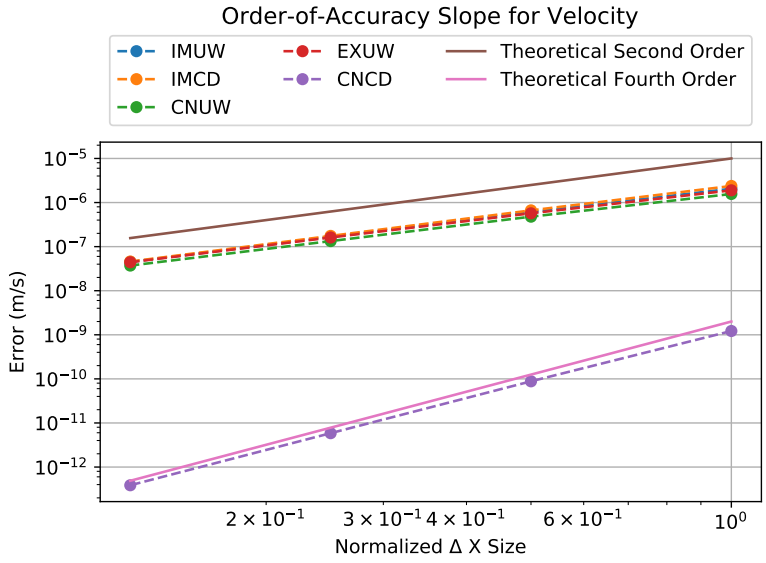

Figure 3.6: MEAMMS Order of Accuracy Results for the Implemented Stable Numerical Schemes (Velocity)

order coding errors, requires only one simulation result, and the spatial grid and timestep size does not have to be in the asymptotic range as long as high enough terms are used to compute a converging result. This state-of-the-art code verification method is the most rigorous test that can be applied a code that utilizes a numerical method. The code was able to pass the higher order MEAMMS test, which is shown in Table 3.7 where the discretization error is lower than round off error after 10 and 20 timesteps. This is confirmed when the discretization error is plotted spatially after 20 timesteps, which is shown in Figure 3.7 for height and Figure 3.8 for velocity.

Table 3.7: Discretization Error After Multiple Timesteps With Higher Order LTE Source Terms

Numerical Scheme	10 Steps $\|\|h\|\|_2$	20 Steps $\|\|h\|\|_2$	10 Steps $\|\|u\|\|_2$	20 Steps $\|\|u\|\|_2$
Explicit Downwind	$4.500 * 10^{-15}$	$2.087 * 10^{-13}$	$1.906 * 10^{-15}$	$7.369 * 10^{-14}$
Explicit Upwind	$4.622 * 10^{-16}$	$1.813 * 10^{-16}$	$2.126 * 10^{-16}$	$3.738 * 10^{-16}$
Explicit Central Difference	$5.588 * 10^{-16}$	$1.901 * 10^{-15}$	$3.452 * 10^{-16}$	$1.236 * 10^{-15}$
Implicit Downwind	$2.513 * 10^{-14}$	$7.548 * 10^{-13}$	$1.978 * 10^{-14}$	$4.655 * 10^{-13}$
Implicit Upwind	$4.441 * 10^{-16}$	$4.054 * 10^{-16}$	$2.794 * 10^{-16}$	$3.452 * 10^{-16}$
Implicit Central Difference	$1.445 * 10^{-15}$	$2.630 * 10^{-15}$	$1.261 * 10^{-15}$	$2.006 * 10^{-15}$
Crank-Nicolson Downwind	$1.010 * 10^{-14}$	$6.642 * 10^{-13}$	$9.233 * 10^{-15}$	$2.162 * 10^{-13}$
Crank-Nicolson Upwind	$4.441 * 10^{-16}$	$4.622 * 10^{-16}$	$3.569 * 10^{-16}$	$3.511 * 10^{-16}$
Crank-Nicolson Central Difference	$6.784 * 10^{-16}$	$1.404 * 10^{-15}$	$7.364 * 10^{-16}$	$8.835 * 10^{-16}$

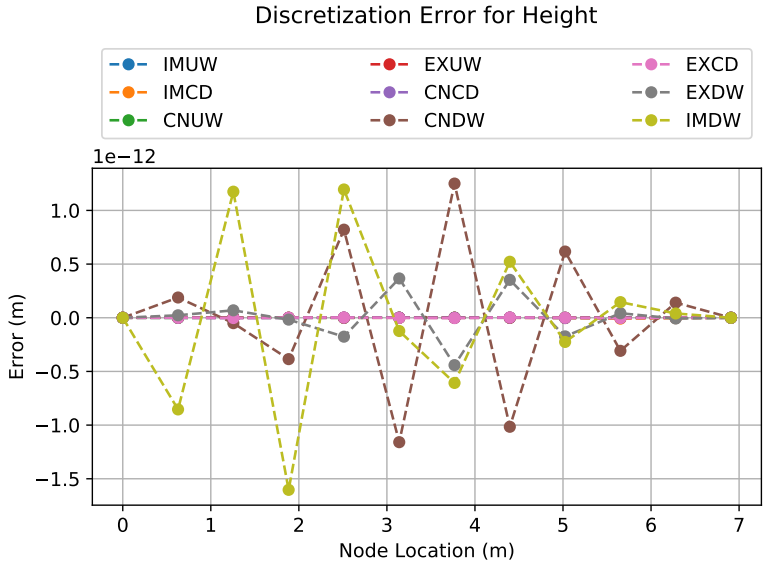

Figure 3.7: Spatial Distribution of Discretization Error After 20 Timesteps with Higher Order LTE Source Terms (Height)

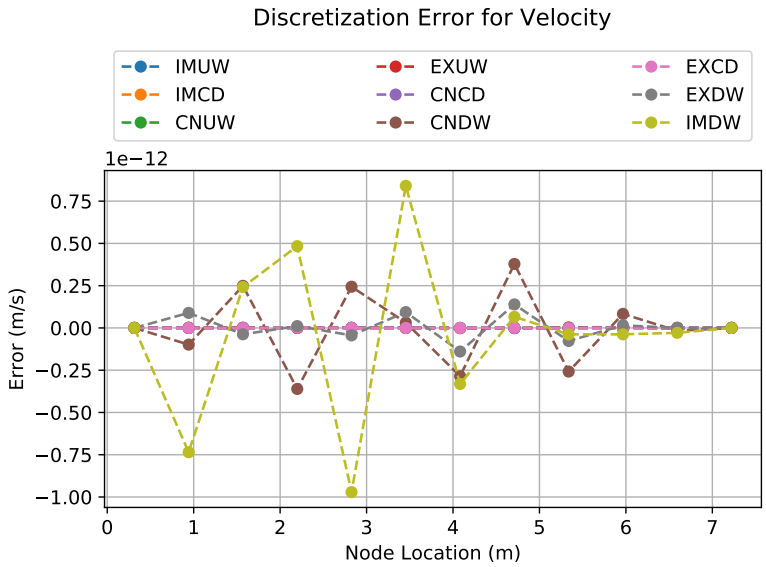

Figure 3.8: Spatial Distribution of Discretization Error After 20 Timesteps with Higher Order LTE Source Terms (Velocity)

3.7 Example of Where MMS Fails

To test the power and robustness of MEAMMS, coding errors are deliberately placed in the code to test each code verification method. There are two types of coding errors that are placed within the code: a zeroth-order coding error and a first-order coding error. Calculating the observed order of accuracy with the MMS method using the simplified error model shouldn't be able to detect first-order coding errors on a first-order method. It should be able to detect lower order coding errors, but the simple model doesn't distinguish between zeroth-order coding errors and other ordered coding errors (or lack of ordered coding errors). Calculating the observed order of accuracy calculated using the complex error model and fitting the model with a least squares method [56] still doesn't detect first-order coding errors on a first-order method, but should be able to distinguish zeroth-order coding errors and other coding errors (or lack of ordered coding errors). MEAMMS will be able to detect both types of coding errors because both types of coding errors impact the truncation error, which MEAMMS detects if there are changes to the truncation error.

3.7.1 Zeroth-Order Coding Error

A possible zeroth-order coding error is an additional constant added to the computed solution. Zeroth-order coding errors alone do not impact the rate of convergence, but instead impacts the value that the computed solution converges to. This means that regardless of the size of Δx or Δt, the error still exists. To test the code verification methods, a constant error was added to the velocity QoI. This test is performed on the implicit upwind numerical scheme using four tests: MMS, LSMMS, leading order MEAMMS, and higher order MEAMMS. A summary of the first three methods and the observed order of accuracies for those methods are shown in Table 3.8. Also, the observed order of accuracy plots are shown in Figures 3.9 through 3.14. Since higher order MEAMMS detect errors by a reduction of discretization error to approximately round-off error using one simulation,

the results are presented in separate format in Figure 3.15 and Figure 3.16.

Table 3.8: Order of Accuracy Calculation with Zeroth-Order Coding Error

Code Verification Method	p_{2_h}	p_{2_u}	No Code Bug
MMS	$-2.830 * 10^{-3}$	$-5.077 * 10^{-3}$	1
LSMMS	0.000	0.000	1
Leading Order MEAMMS	$7.196 * 10^{-5}$	$-8.153 * 10^{-6}$	2

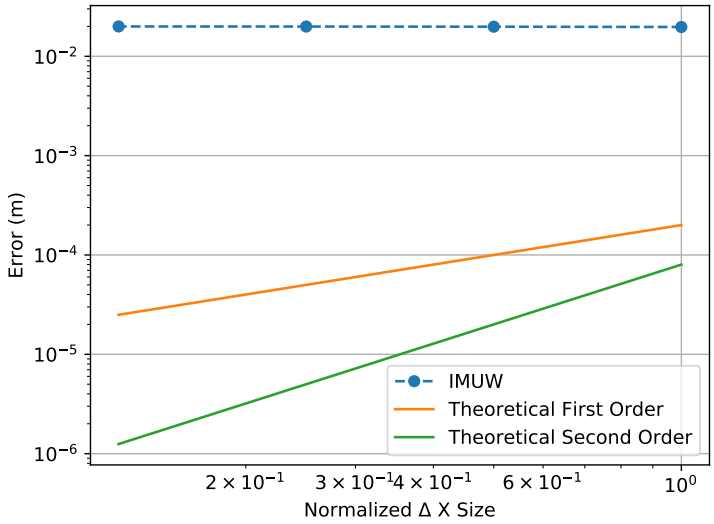

Figure 3.9: MMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Height)

All code verification methods analyzed are able to detect a zeroth-order code bug, but identify them in different ways. MMS with the simple error model identified the coding error by degrading the observed order of accuracy. LSMMS with the complex error model identified the coding error by having a non-constant value for ε_0. The leading order MEAMMS identified the coding error because the theoretical LTE doesn't cancel the

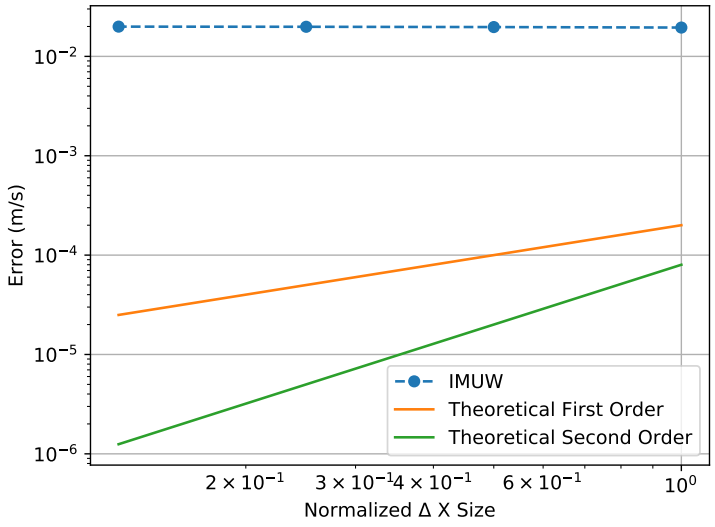

Figure 3.10: MMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Velocity)

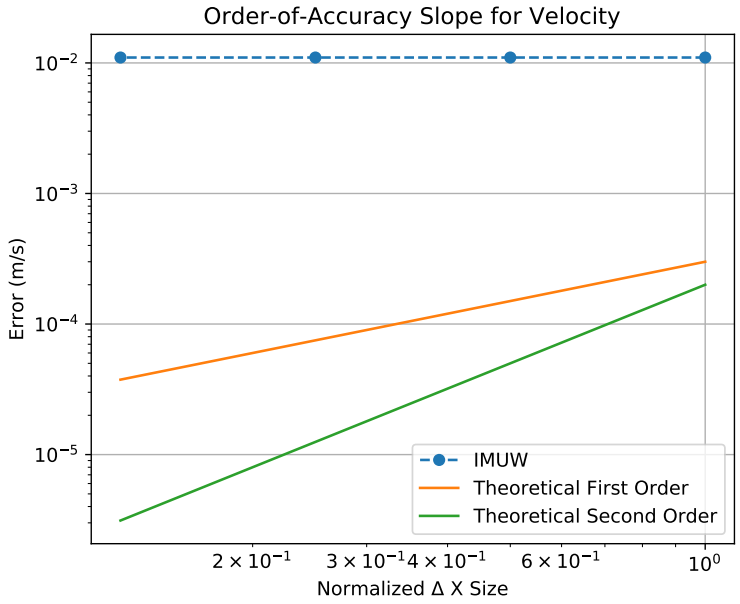

Figure 3.11: Least Squares MMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Height)

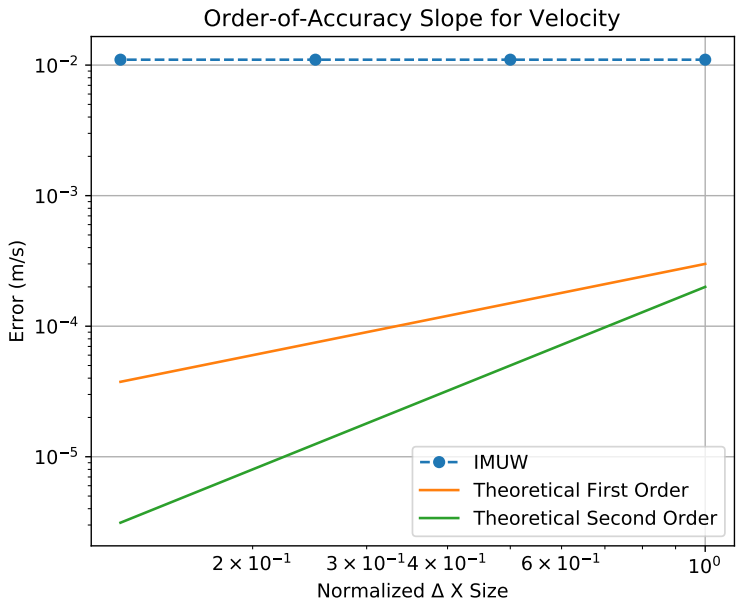

Figure 3.12: Least Squares MMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Velocity)

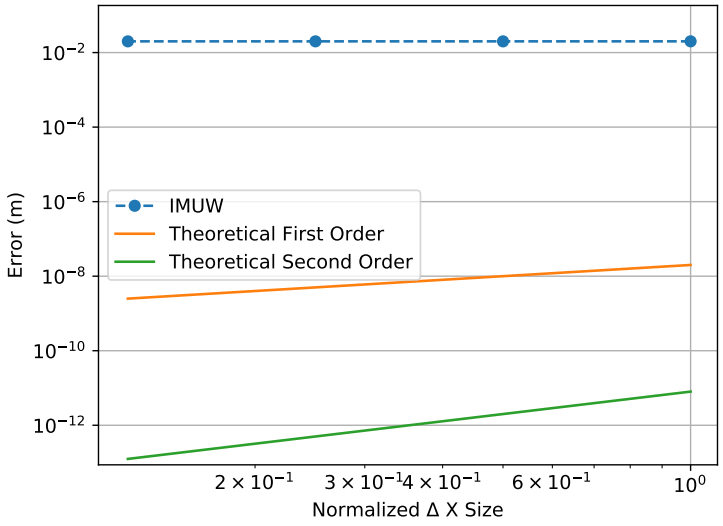

Figure 3.13: Leading Order MEAMMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Height)

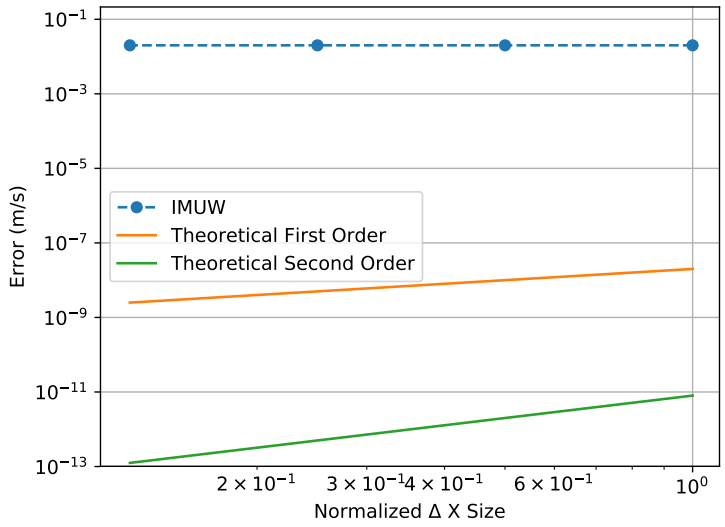

Figure 3.14: Leading Order MEAMMS Order of Accuracy Slope Results for Implicit Upwind with Zeroth-Order Coding Error (Velocity)

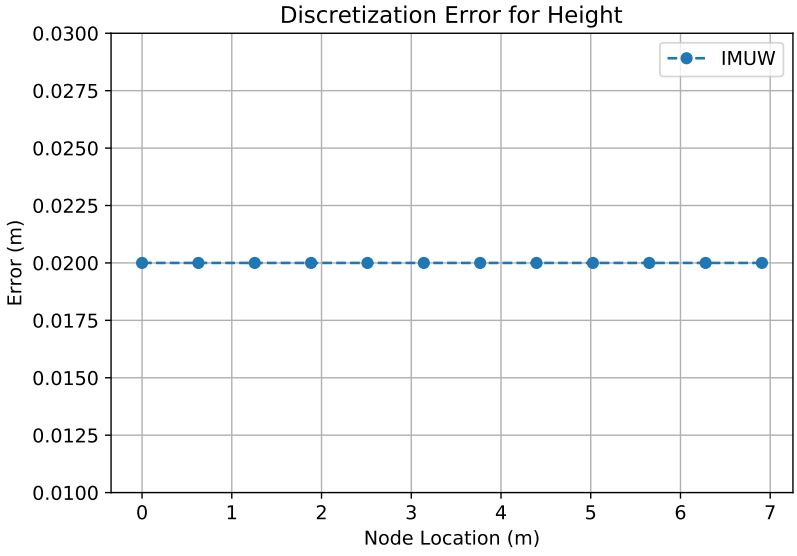

Figure 3.15: Higher Order MEAMMS Discretization Error Results After 10 Timesteps for Implicit Upwind with Zeroth-Order Coding Error (Height)

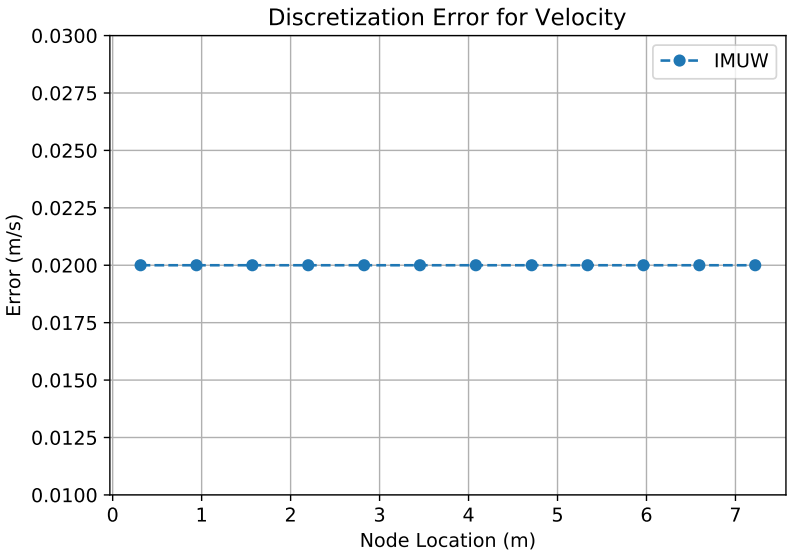

Figure 3.16: Higher Order MEAMMS Discretization Error Results After 10 Timesteps for Implicit Upwind with Zeroth-Order Coding Error (Velocity)

first-order error produced by the code perfectly and raise the observed order of accuracy to two. Similarly, the higher order MEAMMS identified the coding error because the discretization error after 10 timesteps isn't approximately at the level of round off.

3.7.2 First-Order Coding Error

The first-order coding error is a boundary condition error that was originally part of an early draft of the code. This is as simple as an incorrect equation or indexing on the ghost cells. By adding an additional, unnecessary term to the boundary, the code should still converge to the correct solution and still converge at the correct rate, but there will be a difference in the truncation error at the boundary. This test is performed on the implicit upwind numerical scheme using four tests: MMS, LSMMS, leading order MEAMMS, and higher order MEAMMS. A summary of the first three methods and the observed order of accuracies for those methods are shown in Table 3.9. Also, the observed order of accuracy plots are shown in Figures 3.17 through 3.22. Since higher order MEAMMS

detect errors by a reduction of discretization error to approximately round-off error using one simulation, the results are presented in separate format in Figure 3.23 and Figure 3.24.

Table 3.9: Order of Accuracy Calculation with First-Order Coding Error

Code Verification Method	p_{2_h}	p_{2_u}	No Code Bug
MMS	0.951	1.077	1
LSMMS	0.758	0.9449	1
Leading Order MEAMMS	1.063	1.026	2

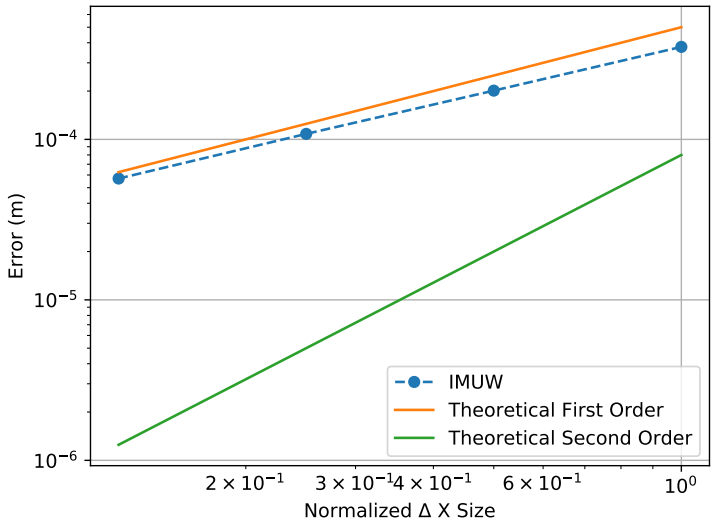

Figure 3.17: MMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Height)

Table 3.9 shows that both MMS error models did not detect the coding error because the observed order of accuracy matches the theoretical order of accuracy. However, leading order MEAMMS and higher order MEAMMS did detect the coding error. The leading order MEAMMS identified the coding error because the theoretical LTE doesn't cancel the

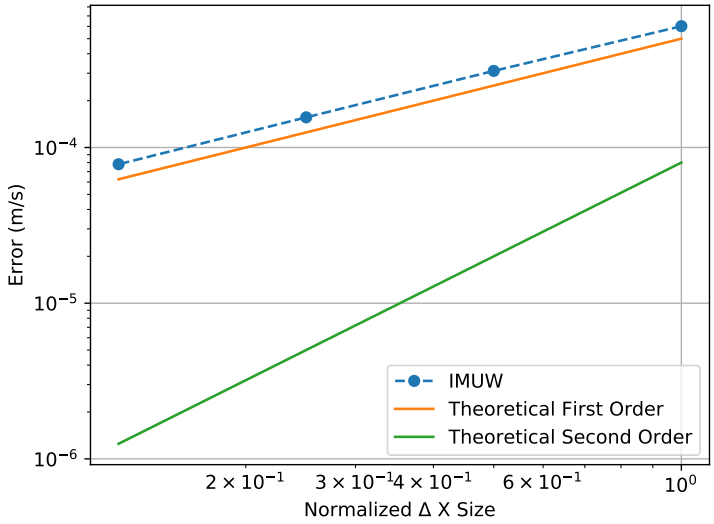

Figure 3.18: MMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Velocity)

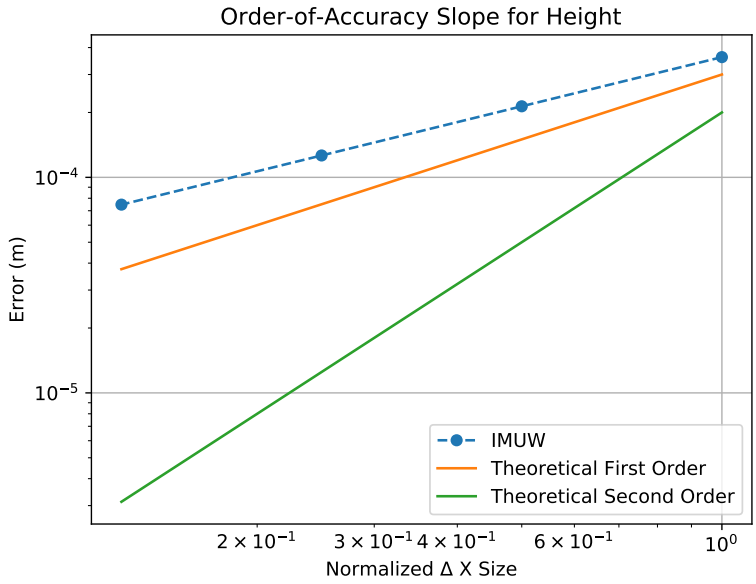

Figure 3.19: Least Squares MMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Height)

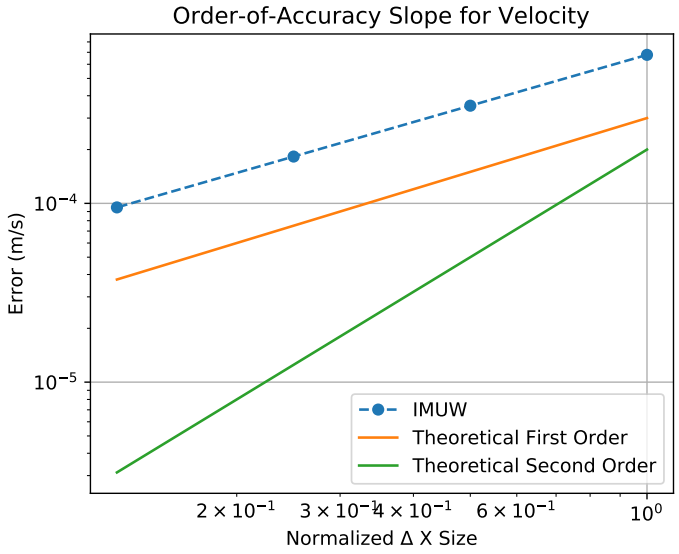

Figure 3.20: Least Squares MMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Velocity)

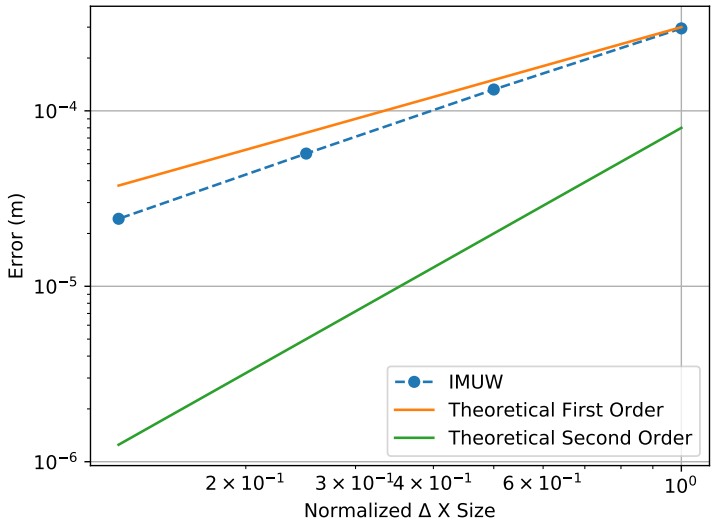

Figure 3.21: Leading Order MEAMMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Height)

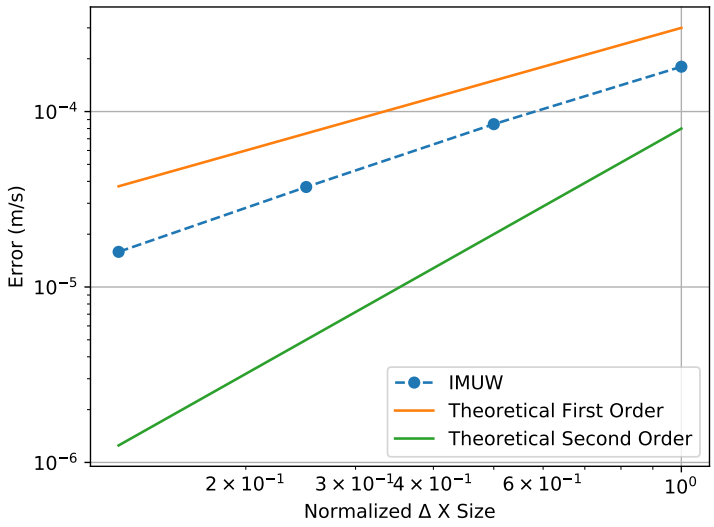

Figure 3.22: Leading Order MEAMMS Order of Accuracy Slope Results for Implicit Upwind with First-Order Coding Error (Velocity)

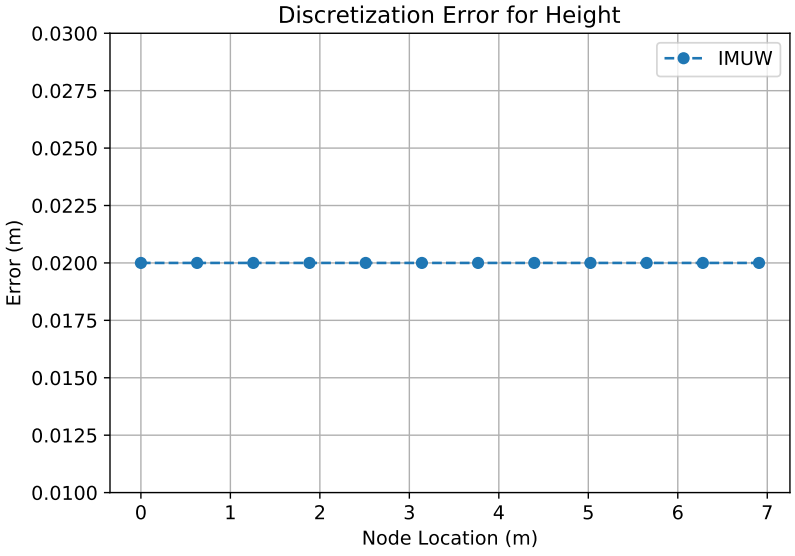

Figure 3.23: Higher Order MEAMMS Discretization Error Results After 10 Timesteps for Implicit Upwind with First-Order Coding Error (Height)

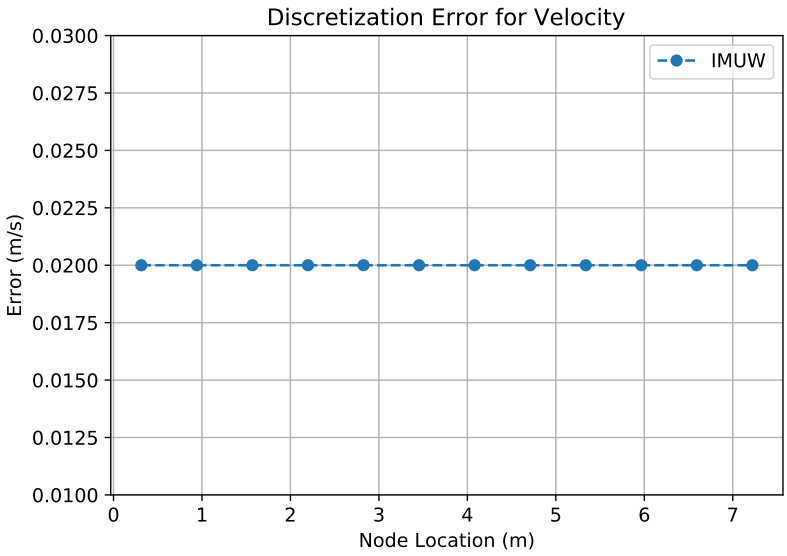

Figure 3.24: Higher Order MEAMMS Discretization Error Results After 10 Timesteps for Implicit Upwind with First-Order Coding Error (Velocity)

first-order error produced by the code perfectly and raise the observed order of accuracy to two. Similarly, the higher order MEAMMS identified the coding error because the discretization error after 10 timesteps isn't approximately at the level of round off.

3.8 Coarse Code Verification

An important note in completing code verification activities is that the solutions need to be within the asymptotic region. The asymptotic region starts when the leading order error becomes the dominant source of the total error. When this happens, the observed order of accuracy will start to converge to the theoretical order of accuracy when coding errors are removed. To show how this works, a coarse grid and large timestep was chosen and then the timestep and spatial step was reduced. As the timesteps and spatial steps are reduced, the solution becomes increasingly accurate and eventually enters the asymptotic region when the observed order of accuracy matches the theoretical order of accuracy to within a tolerance. The parameters in Table 3.10 shows the problem setup for the test. Table 3.11

shows the observed and theoretical order of accuracy with decreasing the timestep and spatial step.

Table 3.10: Code Bug Calculation Setup

Quantity	Value
Domain Length (m)	6.28
End Time (s)	1.0
Cross-Sectional Area (m^2)	0.04
h_1	2.150
h_2	0.220
u_1	1.220
u_2	0.133
k_h	1.12
k_u	1.16
ω_h	1.55
ω_u	1.10
g	0.990

Table 3.11: Order of Accuracy Calculation with Decreasing Timestep and Spatial Step Size

$\frac{\Delta x}{L}$	p_{2_h}	Theoretical Order
0.5	0.769	1.000
0.05	0.836	1.000
0.005	0.973	1.000

As the LTE is added, the observed order of accuracy becomes more accurate and the solutions start to be inside the asymptotic region. This shows that grid and timestep refinement studies need to be completed to determine if the simulation is inside the asymptotic region. In addition, the convergence of the observed order of accuracy ideally would be monitored to ensure that the particular problem didn't get the write answer for the wrong

reason.

Another method to get more accurate solutions is to increase the order of the method. Using the symbolic equations for the theoretical LTE, different ordered solutions are calculated on a grid of 10 cells using 10 timesteps and a discretization error is calculated. Table 3.12 shows the discretization error as a function of LTE order.

Table 3.12: Discretization Calculation with Increasing LTE

Leading LTE	$\|\|h\|\|_2$	$\|\|u\|\|_2$
First	$5.473 * 10^{-2}$	$1.385 * 10^{-2}$
Second	$1.101 * 10^{-2}$	$2.303 * 10^{-3}$
Third	$1.939 * 10^{-3}$	$6.089 * 10^{-4}$
Fourth	$3.014 * 10^{-4}$	$6.727 * 10^{-5}$
Fifth	$3.266 * 10^{-5}$	$1.110 * 10^{-5}$
Sixth	$4.476 * 10^{-6}$	$1.007 * 10^{-6}$

This explains the use of higher order methods to compute solutions, even though the boundary conditions become more complex with increasing the order of the method.

3.9 Code Verification Conclusion and Future Work

The use of MMS order of accuracy tests has increased the rigorousness of code verification testing, but has shown to miss key coding errors. As presented, MEAMMS catches the coding errors that MMS misses, which impacts the accuracy. Through MEAMMS the code is tested to ensure it matches the numerical scheme intended for the application. While calculating the LTE is a sizable effort, MEAMMS is meant for codes that require an additional level of testing that MMS can not match.

Future work includes stability analyses, applying MEAMMS to problems with source terms that include state variable information, apply MEAMMS to a non-uniform grid, and apply MEAMMS to an unstructured grid. The stability analysis would be beneficial to

perform to add additional characteristics to match up with how the code performs, especially in the presence of coding errors. To make the problem more realistic, source terms that include state variables should be tested to ensure applying MEAMMS to problems with source terms are straight forward. The next step to making the problem more realistic is to apply MEAMMS to a problem with non-uniform spacing. This would then cover all structured grid problems for system codes and CFD codes. Lastly, to make this method applicable to all problems, MEAMMS should be applied to an unstructured grid. This can be challenging due to the sheer number of LTE terms and the lack of cancellation of LTE method due to equal, but opposite LTE.

4. SOLUTION VERIFICATION

Solution verification is a crucial step in a VVUQ framework. It allows the user to quantify the numerical uncertainty produced when the physics equations are numerically solved when there is no exact solution. While there has been a significant effort to improve these methods in the past three decades, there are still questions about their performance near and outside the asymptotic region. This work aims to characterize the performance of solution verification methods inside, near, and outside the asymptotic region for steady state and transient problems. To accomplish this, the asymptotic point is derived by comparing the L2 Norm of the leading LTE terms with the L2 Norm of the higher order LTE terms. Problems are run on manufactured problems, so an exact solution and an exact error is available, but is not used as part of the solution verification algorithm. This allows for a comparison between the estimated numerical uncertainty with the exact discretization error. This work is to continue solution verification analysis and derivation of the asymptotic point from the previous work [58].

4.1 Introduction

Solution verification is the process of quantifying the numerical uncertainty for a numerically calculated QoI when the exact discretization error is unknown. Discretization error comes from truncating higher order terms in the numerical scheme and being transported through the numerical solution. Computing the discretization error is prohibitively complex and computationally expensive to calculate without an error transport equation, which requires editing the source code and is not currently implemented in most, if not all, commercial simulation codes. Therefore, the discretization error is treated as an uncertainty. There are multiple ways of estimating this uncertainty with positives and negatives with each method. This work focuses on solution verification methods that are based on

the Richardson extrapolation, which assumes that the solutions are inside the asymptotic range. Often times for real world applications, the solutions are not inside the asymptotic region because it requires prohibitively fine grids or small time steps. Therefore, there is a need to evaluate the performance of solution verification methods inside, near, and outside the asymptotic region.

4.2 Solution Verification Method Development

To better understand solution verification, let's look at three important details of solution verification: the asymptotic region, the solution verification derivation, and various error metrics.

4.2.1 Derivation of Asymptotic Point

Since solving PDEs analytically for real world problems is most likely impossible, numerical approximations of the PDEs are necessary. For finite volume schemes, a Taylor series is used as the approximation method. For the Taylor series to be convergent, the solution must be inside the asymptotic region. The asymptotic range starts when the leading order LTE term becomes dominant compared to all other higher order terms. For example, the Taylor series,

$$\phi_{i+1} = \phi_i + \Delta x \left.\frac{\partial \phi}{\partial x}\right|_i + \frac{\Delta x^2}{2} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i + \frac{\Delta x^3}{6} \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i + \ldots, \tag{4.1}$$

can be used to approximate the derivative $\frac{\partial \phi}{\partial x}$. By rearranging the terms in 4.1, the following derivative is approximated by

$$\underbrace{\left.\frac{\partial \phi}{\partial x}\right|_i}_{\text{Exact Derivative}} = \underbrace{\frac{\phi_{i+1} - \phi_i}{\Delta x}}_{\text{Approximated Derivative}} \underbrace{- \frac{\Delta x}{2} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_i}_{\text{Leading Order LTE}} \underbrace{- \frac{\Delta x^3}{6} \left.\frac{\partial^3 \phi}{\partial x^3}\right|_i - \ldots.}_{\text{Higher Order LTE}} \tag{4.2}$$

By evaluating the leading order LTE term and the higher order terms using the manufactured solution, the asymptotic point is defined when the ratio between the leading and higher order LTE terms is equal to one. This is shown in

$$LTE_{\text{ratio}} = \frac{-\frac{\Delta x}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_i}{-\frac{\Delta x^3}{6}\left.\frac{\partial^3 \phi}{\partial x^3}\right|_i + \ldots}. \tag{4.3}$$

For this analysis, there are three main regions of interest: inside, near, and outside the asymptotic region. Inside of the asymptotic range is where the leading order error dominates the higher order solution. This is where the numerical solution is clearly converging to the exact solution at close to the theoretical rate of convergence. The left boundary of inside the asymptotic range is where $\Delta x = 0$, while the right boundary of inside of the asymptotic region is defined to be 10% of the Δx that corresponds to the asymptotic point. This is further illustrated by Table 4.1.

Table 4.1: Asymptotic Ratio Table

LTE Term	Start of Asymptotic Region	Asymptotic Point	Outside Asymptotic Region
Leading Order LTE Term	10	1	0.1
Higher Order LTE Term	1	1	1

The outside of the asymptotic region is where the solution is characterized as random because the higher order terms are dominating the leading order error and the numerical solution is not converging to the exact solution. The left boundary of outside of the asymptotic region is defined to be the Δx that corresponds to the asymptotic point, while the right boundary is defined as the largest Δx that is outside the asymptotic region. Near the asymptotic region is defined to be between the two regions, where the leading order error and the higher order error is roughly the same magnitude. These regions are classification zones and data within these zones are evaluated separately. This is due to some solution verification methods performing differently than others in different regions. A

visual description of these regions are shown in Figure 4.1.

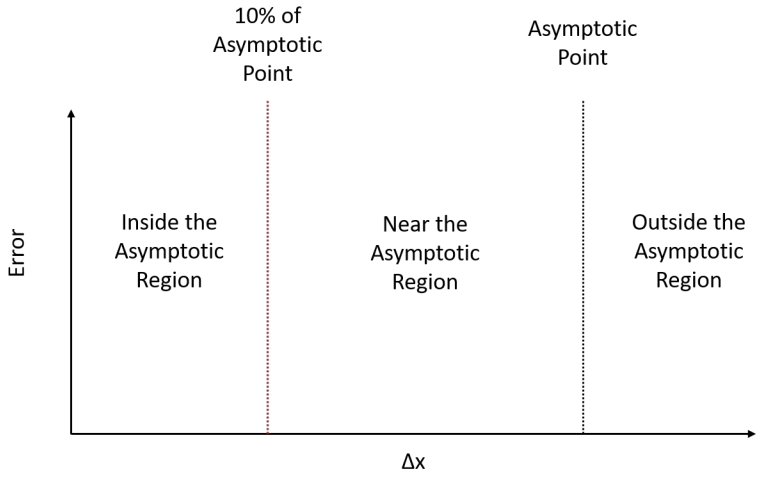

Figure 4.1: The Three Regions Analyzed: Inside, Near, and Outside the Asymptotic Range.

4.2.2 Derivation of Solution Verification Methods

This analysis evaluates the performance of GCI, which is based on the Richardson extrapolations. The basic concept of GCI is to estimate the difference between the extrapolated numerical solution and the finest solution and add a factor of safety that is dependent on how well behaved the extrapolation matches the theoretical performance of the extrapolation. To derive GCI, an explanation of the Richardson extrapolation is necessary and is loosely based on the derivation in [1].

The Richardson extrapolations is based on the work of L. F. Richardson [4, 5] where he observed how the numerical approximation starts to converge to the grid independent solution at a rate based on the order of the leading LTE terms. As an example, let's take three numerical approximations from three successively refined grids,

$$f_1 = \tilde{f} + c_1 h^p + c_2 h^{p+1} + \mathcal{O}\left(h^{p+2}\right), \tag{4.4}$$

$$f_2 = \tilde{f} + c_1 (rh)^p + c_2 (rh)^{p+1} + \mathcal{O}\left(h^{p+2}\right), \text{ and} \tag{4.5}$$

$$f_3 = \tilde{f} + c_1 \left(r^2 h\right)^p + c_2 \left(r^2 h\right)^{p+1} + \mathcal{O}\left(h^{p+2}\right), \tag{4.6}$$

where f_1, f_2, and f_3 are QoIs on successively coarsened grids, $\tilde{f}$ is the grid independent or Richardson extrapolated QoI value, c_1 and c_2 are constant coefficients, h is the characteristic grid size, r is the coarsening factor, and p is the order of the numerical method. Neglecting higher order terms, Eqs. 4.4, 4.5, and 4.6 are simplified to

$$f_1 = \tilde{f} + c_1 h^p, \tag{4.7}$$

$$f_2 = \tilde{f} + c_1 (rh)^p \text{, and} \tag{4.8}$$

$$f_3 = \tilde{f} + c_1 \left(r^2 h\right)^p. \tag{4.9}$$

To solve for the order of accuracy, p, and ultimately $\tilde{f}$, Eq. 4.8 is subtracted from Eq. 4.9 and Eq. 4.7 is subtracted from Eq. 4.8, which is shown in

$$f_3 - f_2 = c_1 r^p h^p \left(r^p - 1\right) \text{ and} \tag{4.10}$$

$$f_2 - f_1 = c_1 h^p \left(r^p - 1\right). \tag{4.11}$$

Dividing Eq. 4.10 by Eq. 4.11 then results into

$$\frac{f_3 - f_2}{f_2 - f_1} = r^p. \tag{4.12}$$

Applying logarithmic rules allows for solving for p,

$$p = \frac{ln\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{ln\left(r\right)}. \tag{4.13}$$

Once p is known, $\tilde{f}$ is now solved for using

$$\tilde{f} = f_1 - \frac{f_2 - f_1}{r^p - 1}. \tag{4.14}$$

$\tilde{f}$ is only exact when the derivation assumptions are met. These include neglecting higher order terms and c_1 being a constant. Since these assumptions are not perfectly true,

$\tilde{f}$ is not exact and therefore not used as the prediction of the numerical code. This is where GCI comes into play. GCI uses $\tilde{f} - f_1$, which is the difference between the Richardson extrapolated QoI value and the QoI on the finest grid, to estimate the discretization error for the finest mesh. Using this error and a factor of safety, uncertainty bars are applied to the f_1 solution, which represents the 5th and 95th percentiles, which is shown in

$$GCI = F_s \frac{|f_2 - f_1|}{r^p - 1}. \tag{4.15}$$

The factor of safety, F_s, is dependent on how well behaved p_{obs} is to the theoretical order of accuracy. The factor of safety is 1.25 for well behaved solutions and 3.0 for ill-behaved solutions. The original implimentation left it to engineering judgement to determine if the solution was behaved or ill-behaved, but Oberkampf and Roy suggested a difference of 10% between the observed and theoretical order of accuracy to determine if the solution was behaved or ill-behaved [1]. Eq. 4.15 is an engineering approximation that determines the error bounds for the discretization error. Using GCI, the numerical uncertainty will be quantified using Oberkampf and Roy's version of GCI.

4.2.3 Comparison Metrics

To evaluate solution verification methods, a variety of metrics are used to compare each method. Each metric has a specific strength in summarizing a large set of comparison data in one number. The normalized L_1 norm, L_2 norm, and $L_{\inf}$ norm are all based on the normalized P norm equation,

$$||X||_p = \frac{1}{n^{1/p}} \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{4.16}$$

where p is the value of the norm, x_i is the difference between the exact error and the estimated error, and n is the length of x_i. Below are detailed descriptions of the metrics used in this work.

4.2.3.1 L_1 Norm

The normalized L_1 norm is a measure of the average absolute difference of the error vector, X, which is shown in

$$||X||_1 = \frac{1}{n}\left(\sum_{i=1}^{n}|x_i|\right). \tag{4.17}$$

This norm is useful when quantifying an error vector by describing the average absolute error for a given process.

4.2.3.2 L_2 Norm

The normalized L_2 norm, also known as the root-mean-square (RMS) error, is a measure of the square root of the average of the square vector, X^2, which is shown in

$$||X||_2 = \frac{1}{n^{1/2}}\left(\sum_{i=1}^{n}|x_i|^2\right)^{1/2}. \tag{4.18}$$

This norm is useful for quantifying the spread of the error vector and is sensitive to outlier data, which is important in highlighting larger disagreements.

4.2.3.3 L_∞ Norm

The L_∞ norm is a measure of the maximum error in the error vector. This is derived by taking the limit of Eq. 4.16 as $P->\inf$, which simplifies to

$$||X||_\infty = max\left(x_i\right). \tag{4.19}$$

This norm is useful when quantifying a maximum value of the error vector. This highlights problem areas in the vector that could be hidden by the L_1 and L_2 norms because it only finds the outliers.

4.2.3.4 Using Comparison Metrics

Since these test problems utilize manufactured solutions, the estimated discretization error with uncertainty bars can be compared to the exact discretization error. This allows

for solution verification methods to be evaluated for their performance. Their performance is based on how well the discretization uncertainty bound contains the exact discretization error. When the exact discretization error is outside the bound, the distance between the exact discretization error and the nearest bound is calculated. This calculation is performed spatially for the steady state case and spatially at the end time for the transient case. This comparison will illuminate the performance of the solution verification methods inside, near, and outside the asymptotic range.

4.3 Description of Test Problems

To fully understand the performance of the solution verification method, it is applied to a manufactured test problem. This test problem uses the shallow water equations with source terms. The first test is a steady state problem to get a baseline assessment of the solution verification method. To understand how the solution verification method performs when both spatial and temporal components of discretization error, the second test is a transient test. This test uses manufactured solutions that are a function of space and time. The manufactured solutions are functions of sine and cosine. Since sine and cosine functions change throughout the spatial and temporal domain, the area of interest is the whole domain. The exact test cases with derivation of the manufactured solutions are shown below.

4.3.1 Steady State

The manufactured solutions and manufactured solution source terms are derived in a similar manner as in the code verification section. The QoIs for this problem are velocity, u, and wave height, h and are shown in

$$u = u_1 + u_2 \cos\left(k_u x\right) \tag{4.20}$$

$$h = h_1 + h_2 \sin\left(k_h x\right). \tag{4.21}$$

To avoid wave overlap, the velocity QoI is a cosine function while the height is a sine function. This is the same manufactured solution as Eq. 3.25 and Eq. 3.26, but the temporal coefficient is set to zero to remove the functional dependence on time.

Once the QoI functions are chosen, the relevant derivatives are calculated in

$$\frac{\partial u}{\partial x} = -u_2 k_u \sin\left(k_u x\right) \tag{4.22}$$

$$\frac{\partial u}{\partial t} = 0 \tag{4.23}$$

$$\frac{\partial h}{\partial x} = h_2 k_h \cos\left(k_h x\right) \tag{4.24}$$

$$\frac{\partial h}{\partial t} = 0. \tag{4.25}$$

By substituting Eqs. 4.22 through 4.25 into Eq. 4.20 and Eq. 4.21, the source terms are calculated and shown in

$$Q_{Mass} = -\left(h_1 + h_2 \sin\left(k_h x\right)\right)\left(u_2 k_u \sin\left(k_u x\right)\right) + \left(u_1 + u_2 \cos\left(k_u x\right)\right)\left(h_2 k_h \cos\left(k_h x\right)\right) \tag{4.26}$$

$$Q_{Mom} = -\left(u_1 + u_2 \cos\left(k_u x\right)\right)\left(u_2 k_u \sin\left(k_u x\right)\right) + \mathbf{g}\left(h_2 k_h \cos\left(k_h x\right)\right). \tag{4.27}$$

4.3.2 Transient

The manufactured solutions and manufactured solution source terms for the transient sine and cosine test problem are derived in the exact same manner as in the code verification section, but presented again here for completeness. The QoIs for this problem are velocity, u, and wave height, h and are shown in

$$u = u_1 + u_2 \cos\left(k_u x + \omega_u t\right) \tag{4.28}$$

$$h = h_1 + h_2 \sin\left(k_h x + \omega_u t\right). \tag{4.29}$$

To avoid wave overlap, the velocity QoI is a cosine function while the height is a sine function. This is the same manufactured solution in Eq. 3.25 and Eq. 3.26.

Once the QoI functions are chosen, the relevant derivatives are calculated in

$$\frac{\partial u}{\partial x} = -u_2 k_u \sin\left(k_u x + \omega_u t\right) \tag{4.30}$$

$$\frac{\partial u}{\partial t} = -u_2 \omega_u \sin\left(k_u x + \omega_u t\right) \tag{4.31}$$

$$\frac{\partial h}{\partial x} = h_2 k_h \cos\left(k_h x + \omega_h t\right) \tag{4.32}$$

$$\frac{\partial h}{\partial t} = h_2 \omega_h \cos\left(k_h x + \omega_h t\right). \tag{4.33}$$

By substituting Eqs. 4.30 through 4.33 into Eq. 4.28 and Eq. 4.29, the source terms are calculated and shown in

$$\begin{aligned} Q_{Mass} = &\left(h_2 \omega_h \cos\left(k_h x + \omega_h t\right)\right) - \left(h_1 + h_2 \sin\left(k_h x + \omega_h t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{4.34}$$

$$\begin{aligned} Q_{Mom} = &-\left(u_2 \omega_u \sin\left(k_u x + \omega_u t\right)\right) - \left(u_1 + u_2 \cos\left(k_u x + \omega_u t\right)\right)\left(u_2 k_u \sin\left(k_u x + \omega_u t\right)\right) \\ &+ \mathbf{g}\left(h_2 k_h \cos\left(k_h x + \omega_h t\right)\right) \end{aligned} \tag{4.35}$$

4.3.3 Initial and Boundary Conditions

Using the manufactured solutions, the solution is known at the start of the problem for both the steady state and transient problems. This is used to initialize the computational simulation through

$$h\left(x, 0\right) = h_1 + h_2 sin\left(k_h x\right), \text{ and} \tag{4.36}$$

$$u\left(x, 0\right) = u_1 + u_2 sin\left(k_u x\right). \tag{4.37}$$

This allows for starting with zero error, but as the problem evolves over space for the steady state case and space and time for the transient case, discretization error starts to accumulate throughout the computational domain.

For the boundary conditions, the manufactured solution is also used. Since the code is implicit upwind, the left boundary is defined as the exact solution as the computational solution, which is shown in

$$h\left(0,t\right) = h_1 + h_2 sin\left(\omega_h t\right), \text{ and} \tag{4.38}$$

$$u\left(0,t\right) = u_1 + u_2 sin\left(\omega_u t\right). \tag{4.39}$$

Since this is a one dimensional problem solved using a first-order method, only one boundary needs to be defined.

4.3.4 Numerical Settings

The range of discretization sizes span across outside, near, and inside the asymptotic range. The steady state and transient simulations use identical simulation settings except that the temporal coefficients (ω_h and ω_u) in the manufactured solutions are set to zero. This ensures that the steady state manufactured solutions do not change as a function of time.

4.3.4.1 Steady State

The steady state simulation settings are based on the computational limitations of only being able to run 29 simulations where the number of cells increase by 20% for each refinement. Based on this limitations, the length scales of the problem were selected such that the middle of the near asymptotic region is in the middle of the plot. This provides the ideal settings to analyze all three different regions. The settings for the steady state simulations are shown in Table 4.2.

4.3.4.2 Transient

The transient simulation settings are based on the computational limitations of only being able to run 25 simulations where the number of cells and time steps increase by 20% for each refinement. This is less than the number of refinement levels of the steady state

Table 4.2: Steady State Calculation Setup

Quantity	Value
Domain Length (m)	1.0
End Time (s)	1.0
Cross-Sectional Area (m^2)	0.04
h_1	1.150
h_2	0.022
u_1	1.220
u_2	0.0133
k_h	12.12
k_u	12.55
ω_h	0.00
ω_u	0.00
$\mathbf{g}$	9.81
N_{coarse}	4
N_{Sims}	29
r	1.2

study because the transient simulations cost significantly more computationally. Based on this limitations, the length scales and time scales of the problem were selected such that the middle of the near asymptotic region is in the middle of the plot. This provides the ideal settings to analyze all three different regions. The settings for the steady state simulations are shown in Table 4.3.

4.4 Results

Solution verification methods are used to estimate the discretization error and include uncertainty. For real world applications, simulations often do not fully resolve the QoI that are calculated. This means that the solutions are calculated outside the asymptotic range. To better understand the performance of solution verification methods like GCI, two separate simulation data sets are ran inside, near, and outside the asymptotic range to determine the performance. The first set of simulations is a steady state case, where only spatial discretization error exist. The second set of simulations is a transient case, where

Table 4.3: Transient Calculation Setup

Quantity	Value
Domain Length (m)	1.0
End Time (s)	1.0
Cross-Sectional Area (m^2)	0.04
h_1	1.150
h_2	0.022
u_1	1.220
u_2	0.0133
k_h	12.12
k_u	12.55
ω_h	12.16
ω_u	12.30
$\mathbf{g}$	9.81
N_{coarse}	4
N_{Sims}	25
r	1.2

both spatial and temporal discretization error exists.

4.4.1 Steady State

For the steady state case, the solution for both height and velocity are computed for a range of different discretizations. Using this data, the relative error between the solution on one mesh and the next refined mesh is computed. Additionally, the exact discretization is also computed as a reference. Only the performance metrics have access to the exact discretization data because in real world applications, the exact discretization error is unknown. The relative and exact discretization error for height and velocity are shown in Figure 4.2 and Figure 4.3, respectively.

In addition to showing the relative and exact discretization error, the three different asymptotic ranges are shown. To the right of the black dotted lines is outside the asymptotic range, between the red and black dotted lines is near the asymptotic range, and to the left of the red dotted lines inside the asymptotic range. Additionally, reference slopes

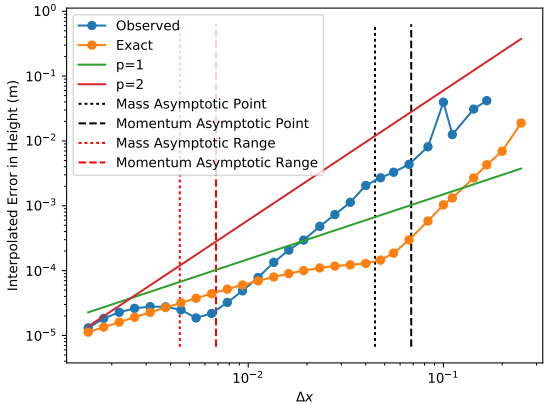

Figure 4.2: Error in Height as a Function of Discretization Size for Steady State Problem

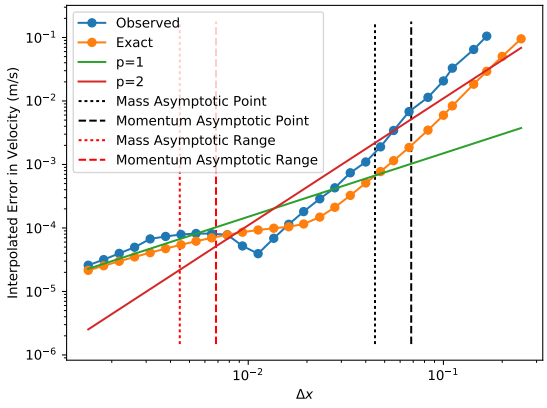

Figure 4.3: Error in Velocity as a Function of Discretization Size for Steady State Problem

are shown to put into perspective the observed order of accuracy in each of the ranges. When the simulation data is outside the asymptotic range, the observed order of accuracy is much larger than the theoretical first-order slope. This is because the higher order LTE is dominating the leading order LTE. As the solution starts to get near the asymptotic range, the exact discretization error starts to match the theoretical performance because the leading LTE terms is starting to dominate. Since the relative discretization error is calculated between two meshes with similar level of refinement, it takes a more refinement than the exact discretization error data set for the leading LTE terms to dominate. Once the solutions are fully inside the asymptotic region, both the relative and exact discretization error starts to match the first-order of accuracy slope. This is confirmed when visualizing just the the order of accuracy, which is shown in Figure 4.4 for height and Figure 4.5 for velocity.

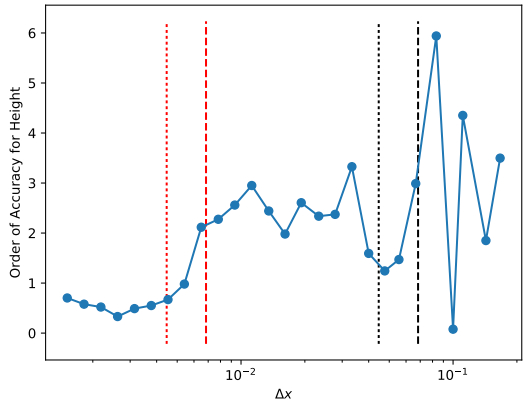

Figure 4.4: Observed Order of Accuracy of Height as a Function of Discretization Size for Steady State Problem

Both these figures show the order of accuracy very chaotic outside the asymptotic region, smoother near the asymptotic region, and convergent to the theoretical order of accuracy of one. This should give a good data set to see how the GCI performs in these

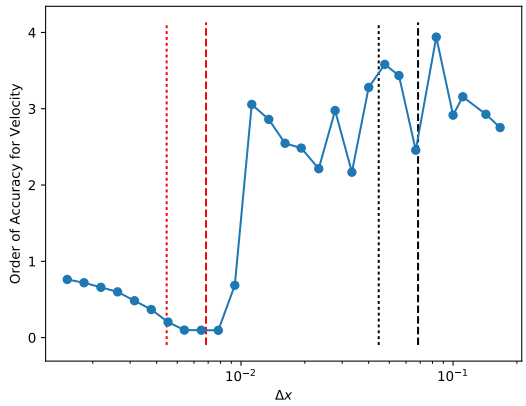

Figure 4.5: Observed Order of Accuracy of Velocity as a Function of Discretization Size for Steady State Problem

three asymptotic regimes. While the error and observed order of accuracy data looks chaotic, the discretization uncertainty calculated by GCI bounds the exact discretization error everywhere for Height (Figure 4.6). For velocity, there are only a few points that are not bounded (Figure 4.7) in the near asymptotic range regime.

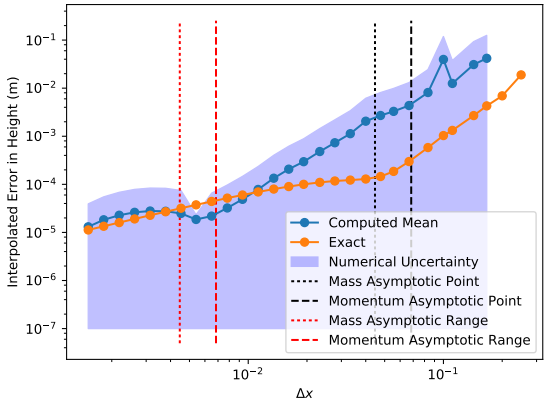

Figure 4.6: Error with GCI Uncertainty in Height as a Function of Discretization Size for Steady State Problem

This is broken up by asymptotic regimes. These results are initially surprising because of how chaotic the order of accuracy is, but upon further inspection in how the GCI cal-

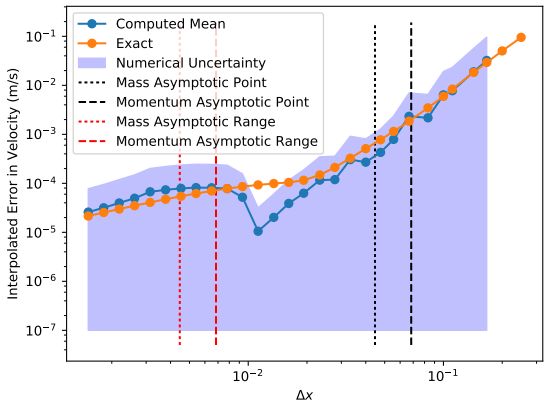

Figure 4.7: Error with GCI Uncertainty in Velocity as a Function of Discretization Size for Steady State Problem

culation is performed as formulated by [1], the observed order of accuracy used in the equations is bounded between the theoretical order of accuracy and 0.5. This effectively smooths out the observed order of accuracy, making the GCI computation more reliable.

4.4.2 Transient

For the transient case, the discretization error is a combination of spatial and temporal errors, which should make it the more difficult for the solution verification methods to estimate the discretization error. However, the results suggest that the additional entropy from the temporal term smooths the solution. As with the steady state case, the QoIs are the discretization errors in height and velocity and are shown in Figure 4.8 and Figure 4.9, respectively.

As with the steady state case, the higher order LTE terms are dominating the leading order LTE, which leads to a higher order discretization error outside the asymptotic region. This dominance becomes less near the asymptotic range, and finally, inside the asymptotic range, the discretization error converges to the theoretical order of accuracy, which is one.

The observed order of accuracy plots shown in Figure 4.10 and Figure 4.11 show that there are more data points that are convergent to the theoretical order of accuracy. This

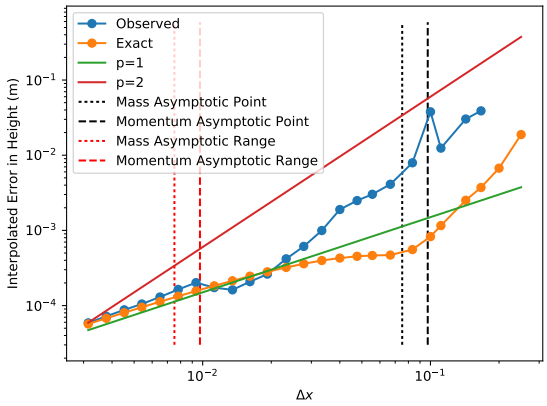

Figure 4.8: Error in Height as a Function of Discretization Size for Transient Problem

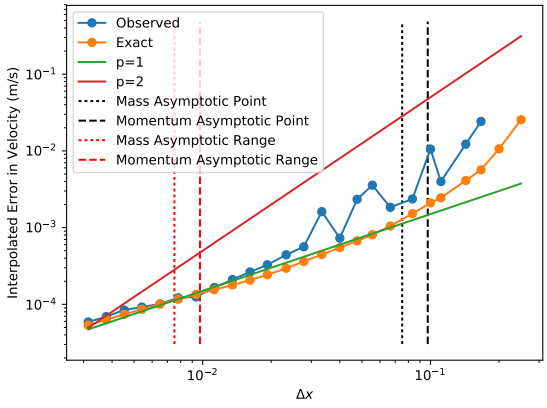

Figure 4.9: Error in Velocity as a Function of Discretization Size for Transient Problem

is most likely due to the additional diffusion from the temporal error term. The observed order of accuracy is chaotic outside the asymptotic regions, starts to being smooth near the asymptotic region, and convergent to the theoretical order of accuracy inside the asymptotic region.

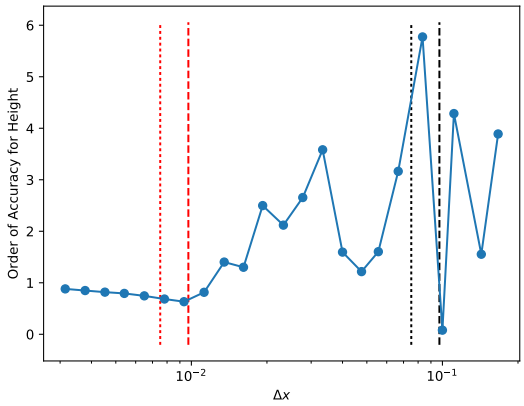

Figure 4.10: Observed Order of Accuracy of Height as a Function of Discretization Size for Transient Problem

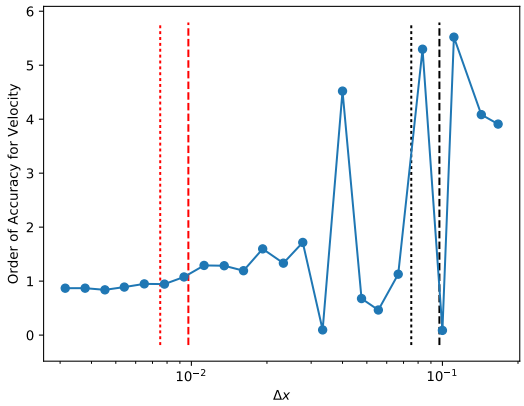

Figure 4.11: Observed Order of Accuracy of Velocity as a Function of Discretization Size for Transient Problem

The GCI discretization uncertainty bound for the transient problem actually performs

better than the steady state problem, which is shown in Figure 4.12 for height and Figure 4.13 for velocity. These results are even more surprising because capturing both temporal and spatial discretization error in the uncertainty bound. This is caused by having a much smoother computed mean discretization error, which could be cause by the addition of temporal discretization error within this problem.

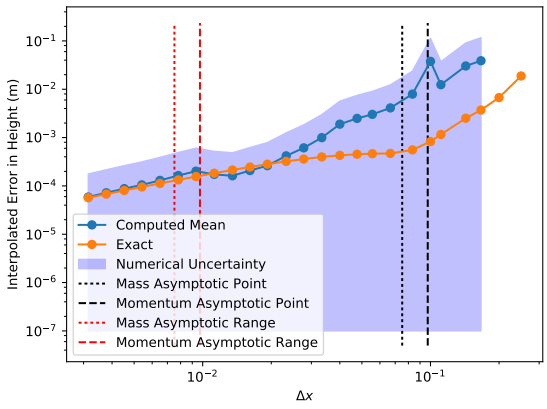

Figure 4.12: Error with GCI Uncertainty in Height as a Function of Discretization Size for Transient Problem

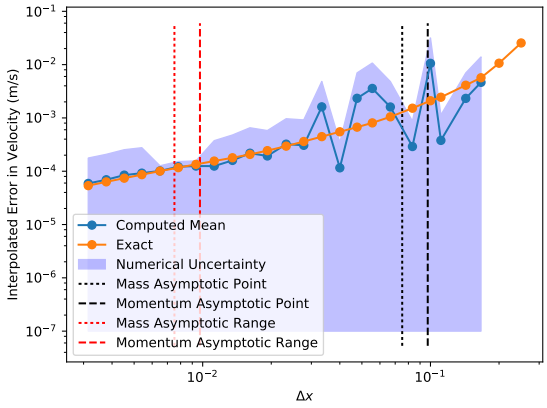

Figure 4.13: Error with GCI Uncertainty in Velocity as a Function of Discretization Size for Transient Problem

The L_1, L_2, and L_∞ values confirm that the GCI discretization uncertainty bounds the exact error for all regimes, which are all zero.

4.5 Conclusion and Future Work

Understanding solution verification method's performance in a variety of computational regimes is an important step towards delivering credible predictions. Since the regimes (inside, near, and outside the asymptotic range) have different behavior, it is important to ensure the solution verification methods bound the exact discretization error without too much conservatism.

This work aims at starting this analysis. With the author's definitions of inside, near, and outside the asymptotic range, the solution is put into perspective of how dominant the leading order LTE is. By testing the performance of GCI on a steady state and transient case, GCI's performance is tested for spatial discretization error and both spatial and temporal discretization error. For both data sets, GCI correctly bounds the exact discretization error in all regimes. It does, however, have more chaotic behavior near and outside the asymptotic range. Therefore, it is important that other solution verification methods perform better than GCI outside and near the asymptotic range for cases that are more chaotic than this data set.

Future work would be to assess more solution verification methods, such as least squares GCI (LSGCI) [22], factor of safety method [21], robust multi-regression [24], and StREEQ [25] to determine if any of the approaches above perform better. Additionally, applying these methods on data sets that have a large difference in spatial and temporal scales of the mass and momentum equations would be more realistic than these data sets where the the scales are roughly the same for the mass and momentum equations.

5. VERIFICATION AND VALIDATION CONSIDERATIONS

For engineering applications, it is often is the case that computational codes are used to make predictions when little experimental data exists. In the nuclear power plant industry, full scale experiments are expensive as well as potentially hazardous when investigating accident phenomena. Because of this, scaled experiments are the only feasible way to understand the physics for the plant. These experiments are used to inform the models used within computational codes. One key assumption is that the models are applicable when scaling to the full system to predict full system QoIs. The credibility of these predictions is reduced when verification and validation (V&V) activities are not followed when developing simulations. Unfortunately, it is common practice to skip code and solution verification and jump right to calibration and validation. While assessing if the models match well with scaled experimental results and calibrating the model when useful, not assessing and reducing numerical error has the unintended consequence of being calibrated into the model. This results in the model being a function of discretization error and coding errors, rather than strictly a function of physics. Completing each component of V&V is of crucial importance when using codes for predictive capabilities. This study will show that when each component of the V&V process is not properly completed, the predictive capability of a simulation is degraded.

5.1 Introduction

To make credible predictions, assessing or reducing each source of error for the QoI is necessary. Each step in the V&V framework assesses different sources of error. The V&V framework consists of SQA, code verification, solution verification, calibration, and validation. SQA reduces coding errors that reduce the quality of the code. SQA processes, such as documenting the code or performing regression tests, reduce unintended

code changes. Code verification checks the equations within the code are coded as intended, which reduces code bug error. Solution verification assesses numerical error for the given application and makes sure it's acceptable for the given application. Validation and calibration assesses and reduces model form error. When one of these errors are not quantified and accounted for, it reduces the credibility of the prediction being made.

The two most unquantified class of errors are code bug errors and numerical errors. The reason these two error are typically unquantified is the historical perspective that the ultimate test of quality is quantitatively matching experimental data. While the ramifications of only quantifying model form error might be limited when the predictions are interpolated inside of the experimental data set, it causes large problems when the predictions are extrapolated outside of the experimental data set. The reason for this is scaling concepts applied to continuous equations are not necessarily scaled in the same way as the discrete equations where code bug errors and numerical errors exist.

5.2 Problem Description

To show the issue of not properly completing code and solution verification, an example problem is developed. The example problem mimics a problem in nuclear power plant accident dose calculation at the site boundary. Since the full scale experiment is too costly and hazardous to complete, scaled experiments inform computational tools to predict full scale phenomena. To understand how code and solution verification impacts the predictive capability of calculating the dose at the boundary, two different simulations are produced. The correct VVUQ simulation completes code verification using MEAMMS, solution verification using GCI, validation and calibration on the scaled experiment. These calibrated coefficients are then used to simulate the full scale problem. The incorrect VVUQ simulations only performs validation and calibration on the scaled experiment. These calibrated coefficients are then used to simulate the full scale problem. Using synthetic experimental

data for the scaled and full scale problem, the predictive capability of both simulations are measured. The difference in the predictive capability is solely due to the use of code and solution verification. The following describes how the synthetic experimental data is generated as well as the computational simulation set up.

5.2.1 Synthetic Experimental Data Generation

To model the atmospheric transport of fission products, the shallow water equations in Eq. 2.19 and Eq. 2.20 and the advection diffusion equation in Eq. 2.21 are used. To understand how to set up the scaled experiment, these equations are non-dimensionalized to determine the problem set up in the scaled and full scale experiment.

5.2.1.1 Non-Dimensional Equation Set

To start the non-dimensionalization process, each variable and coefficient is normalized by a characteristic value of the variable. For instance, for flow between the power plant and the site boundary, the characteristic length can be set to the distance between the power plant and the site boundary. This means the scaled experiment is based on the total length of the set up. This process is done for every variable. The normalized variables are derived in

$$t^* = \frac{t}{t_0}, \tag{5.1}$$

$$h^* = \frac{h}{h_0}, \tag{5.2}$$

$$x^* = \frac{x}{x_0}, \tag{5.3}$$

$$C^* = \frac{C}{C_0}, \tag{5.4}$$

$$u^* = \frac{u}{u_0} = \frac{x^*}{t^*}, \tag{5.5}$$

$$\mathbf{g}^* = \frac{\mathbf{g}}{\mathbf{g}_0} = \frac{h^*}{t^{*2}}, \text{ and} \tag{5.6}$$

$$\nu^* = \frac{\nu}{\nu_0} = \frac{x^{*2}}{t^*}. \tag{5.7}$$

The non-dimensionalized variables are then substituted into the conservation equations, Eq. 2.19, Eq. 2.20, and Eq. 2.21, which is shown in

$$\left(\frac{h_0}{t_0}\right)\frac{\partial h^*}{\partial t^*} + \left(\frac{h_0 u_0}{x_0}\right) h^* \frac{\partial u^*}{\partial x^*} + \left(\frac{u_0 h_0}{x_0}\right) u^* \frac{\partial h^*}{\partial x^*} = 0, \tag{5.8}$$

$$\left(\frac{u_0}{t_0}\right)\frac{\partial u^*}{\partial t^*} + \left(\frac{u_0^2}{x_0}\right) u^* \frac{\partial u^*}{\partial x^*} + \left(\frac{\mathbf{g}_0 h_0}{x_0}\right) \mathbf{g}^* \frac{\partial h^*}{\partial x^*} = 0, \text{ and} \tag{5.9}$$

$$\left(\frac{C_0}{t_0}\right)\frac{\partial C^*}{\partial t^*} + \left(\frac{C_0 u_0}{x_0}\right) u^* \frac{\partial C^*}{\partial x^*} + \left(\frac{\nu_0 C_0}{x_0^2}\right) \nu^* \frac{\partial^2 C^*}{\partial x^{*^2}} = 0. \tag{5.10}$$

Now to divide by the coefficients in front of time

$$\frac{\partial h^*}{\partial t^*} + \left(\frac{t_0}{h_0}\right)\left(\frac{h_0 u_0}{x_0}\right) h^* \frac{\partial u^*}{\partial x^*} + \left(\frac{t_0}{h_0}\right)\left(\frac{u_0 h_0}{x_0}\right) u^* \frac{\partial h^*}{\partial x^*} = 0, \tag{5.11}$$

$$\frac{\partial u^*}{\partial t^*} + \left(\frac{t_0}{u_0}\right)\left(\frac{u_0^2}{x_0}\right) u^* \frac{\partial u^*}{\partial x^*} + \left(\frac{t_0}{u_0}\right)\left(\frac{\mathbf{g}_0 h_0}{x_0}\right) \mathbf{g}^* \frac{\partial h^*}{\partial x^*} = 0, \text{ and} \tag{5.12}$$

$$\frac{\partial C^*}{\partial t^*} + \left(\frac{t_0}{C_0}\right)\left(\frac{C_0 u_0}{x_0}\right) u^* \frac{\partial C^*}{\partial x^*} + \left(\frac{t_0}{C_0}\right)\left(\frac{\nu_0 C_0}{x_0^2}\right) \nu^* \frac{\partial^2 C^*}{\partial x^{*^2}} = 0. \tag{5.13}$$

After some simplification, the resulting equations are

$$\frac{\partial h^*}{\partial t^*} + \left(\frac{t_0 u_0}{x_0}\right) h^* \frac{\partial u^*}{\partial x^*} + \left(\frac{u_0 t_0}{x_0}\right) u^* \frac{\partial h^*}{\partial x^*} = 0, \tag{5.14}$$

$$\frac{\partial u^*}{\partial t^*} + \left(\frac{t_0 u_0}{x_0}\right) u^* \frac{\partial u^*}{\partial x^*} + \left(\frac{t_0 \mathbf{g}_0 h_0}{u_0 x_0}\right) \mathbf{g}^* \frac{\partial h^*}{\partial x^*} = 0, \text{ and} \tag{5.15}$$

$$\frac{\partial C^*}{\partial t^*} + \left(\frac{t_0 C_0 u_0}{x_0}\right) u^* \frac{\partial C^*}{\partial x^*} + \left(\frac{t_0 \nu_0}{x_0^2}\right) \nu^* \frac{\partial^2 C^*}{\partial x^{*^2}} = 0. \tag{5.16}$$

Using the identity $u_0 = \frac{x_0}{t_0}$, an additional simplification is completed on

$$\frac{\partial h^*}{\partial t^*} + h^* \frac{\partial u^*}{\partial x^*} + u^* \frac{\partial h^*}{\partial x^*} = 0, \tag{5.17}$$

$$\frac{\partial u^*}{\partial t^*} + u^* \frac{\partial u^*}{\partial x^*} + \left(\frac{\mathbf{g}_0 h_0}{u_0^2} \right) \mathbf{g}^* \frac{\partial h^*}{\partial x^*} = 0, \text{ and} \tag{5.18}$$

$$\frac{\partial C^*}{\partial t^*} + u^* \frac{\partial C^*}{\partial x^*} + \left(\frac{\nu_0}{u_0 x_0} \right) \nu^* \frac{\partial^2 C^*}{\partial x^{*^2}} = 0. \tag{5.19}$$

The terms in brackets can then be replaced by non-dimensional numbers, which are shown in

$$\frac{\partial h^*}{\partial t^*} + h^* \frac{\partial u^*}{\partial x^*} + u^* \frac{\partial h^*}{\partial x^*} = 0, \tag{5.20}$$

$$\frac{\partial u^*}{\partial t^*} + u^* \frac{\partial u^*}{\partial x^*} + \left[\frac{1}{Fr^2} \right] \mathbf{g}^* \frac{\partial h^*}{\partial x^*} = 0, \text{ and} \tag{5.21}$$

$$\frac{\partial C^*}{\partial t^*} + u^* \frac{\partial C^*}{\partial x^*} + \left[\frac{1}{Pe} \right] \nu^* \frac{\partial^2 C^*}{\partial x^{*^2}} = 0. \tag{5.22}$$

where $Fr = \frac{u}{\sqrt{\mathbf{g}h}}$ is the Froude number and $Pe = \frac{ux}{\nu}$ is the Péclet number. This process allows for understanding how the results can be scaled from the scaled experiment to the full scale experiment. The next section derives the exact solution and the method to add error to mimic manufactured experimental measurement error for the scaled experiment.

5.2.1.2 Scaled Experiment

The scaled experiment represents an experiment that is relatively inexpensive to generate a large amount of data as opposed to the full scale where generating only a few data points is feasible. Using the large amount of scaled data, physical coefficients in the code are able to be calibrated. To fully control the problem, the data is generated from a very fine numerical solution and is used instead of an exact solution. The very fine numerical solution allows for a better understanding of the prediction performance. The numerical

solution for the shallow water equations are of the same manufactured solutions as previous test problems. The sine and cosine functions represent the atmospheric portion of the test problem by modeling the height and velocity of atmospheric equations. The functions representing height and velocity are

$$u = u_1 + u_2 \cos\left(k_u x + \omega_u t\right) \text{ and} \tag{5.23}$$

$$h = h_1 + h_2 \sin\left(k_h x + \omega_h t\right). \tag{5.24}$$

The fission product concentration released during an accident can be represented as a pulse. To ensure the pulse is smooth across the entire domain, the pulse is initialized as a Gaussian functions shown in

$$C = C_1 + C_2 e^{-\frac{\left(k_C x - \omega_C t\right)^2}{2\delta^2}}. \tag{5.25}$$

The manufactured solution is used to initialize the solution and the boundaries are set far from the pulse to ensure the boundaries are not impacting the solution. The manufactured solution coefficients, characteristic values, and non-dimensional coefficients for the scaled problem are shown in Table 5.1.

Table 5.1: Scaled Settings

Quantity	Unit	Dimensional Value	Characteristic Value	Characteristic Value Equation	Non-Dimensional Value	Non-Dimensional Value Equation
x	$[Length]$	25.000	25.000	$x_0 = \frac{x}{1}$	1.000	$x^* = \frac{x}{x_0}$
t	$[Time]$	6.000	6.000	$t_0 = \frac{t}{1}$	1.000	$t^* = \frac{t}{t_0}$
h_1	$[Height]$	0.0115	0.0115	$h_{1_0} = \frac{h_1}{1}$	1.000	$h_1^* = \frac{h_1}{h_{1_0}}$
h_2	$[Height]$	0.0022	0.0115	$h_{2_0} = h_{1_0}$	0.191	$h_2^* = \frac{h_2}{h_{2_0}}$
u_1	$\left[\frac{Length}{Time}\right]$	1.220	4.167	$u_{1_0} = \frac{x_0}{t_0}$	0.293	$u_1^* = \frac{u_1}{u_{1_0}}$
u_2	$\left[\frac{Length}{Time}\right]$	0.0133	4.167	$u_{2_0} = \frac{x_0}{t_0}$	0.003	$u_2^* = \frac{u_2}{u_{2_0}}$
c_1	$[Fraction]$	0.0001	0.500	$c_{1_0} = \frac{c_1}{1}$	0.0002	$c_1^* = \frac{c_1}{c_{1_0}}$
c_2	$[Fraction]$	0.500	0.500	$c_{2_0} = c_{1_0}$	1.000	$c_2^* = \frac{c_2}{c_{2_0}}$
k_h	$\left[\frac{1}{Length}\right]$	0.312	0.040	$k_{h_0} = \frac{1}{x_0}$	7.800	$k_h^* = \frac{k_h}{k_{h_0}}$
k_u	$\left[\frac{1}{Length}\right]$	0.316	0.040	$k_{u_0} = \frac{1}{x_0}$	7.900	$k_u^* = \frac{k_u}{k_{u_0}}$
k_C	$\left[\frac{1}{Length}\right]$	0.100	0.040	$k_{C_0} = \frac{1}{x_0}$	2.500	$k_h^* = \frac{k_C}{k_{C_0}}$
ω_h	$\left[\frac{1}{Time}\right]$	0.155	0.167	$\omega_{h_0} = \frac{1}{t_0}$	0.930	$\omega_h^* = \frac{\omega_h}{\omega_{h_0}}$
ω_u	$\left[\frac{1}{Time}\right]$	0.100	0.167	$\omega_{u_0} = \frac{1}{t_0}$	0.600	$\omega_u^* = \frac{\omega_u}{\omega_{u_0}}$
ω_C	$\left[\frac{1}{Time}\right]$	0.210	0.167	$\omega_{C_0} = \frac{1}{t_0}$	1.260	$\omega_C^* = \frac{\omega_C}{\omega_{C_0}}$
δ	$[Non - Dim]$	0.100	1.000	$\delta_0 = 1$	0.100	$\delta^* = \frac{\delta}{\delta_0}$
$\mathbf{g}$	$\left[\frac{Height}{Time^2}\right]$	9.81	0.0003	$\mathbf{g}_0 = \frac{h_0}{t_0^2}$	$30,710$	$\mathbf{g}^* = \frac{\mathbf{g}}{\mathbf{g}_0}$
ν	$\left[\frac{Length^2}{Time}\right]$	1.100	104.167	$\nu_0 = \frac{x_0^2}{t_0}$	0.011	$\nu^* = \frac{\nu}{\nu_0}$

Based on these values, the two non-dimensional parameters are $Fr = 3.632$ and $Pe = 27.727$. These values are necessary to ensure the correct balance of physics is preserved between the scaled and full scale experiment.

A very fine case is used as the exact solution for the fission product concentration. To ensure the numerical error is low, the first-order LTE is calculated to be small relative to the physical diffusion coefficient, ν. Using the leading order LTE calculated in Mathematica, which is the same as Eq. 3.45 except height is replaced by fission product concentration, the modified equation PDE for the fission product conservation of mass equation is shown in

$$\underbrace{V\frac{\partial C}{\partial t} + Vu\frac{\partial C}{\partial x} + VC\frac{\partial u}{\partial x} - V\nu\frac{\partial^2 C}{\partial x^2}}_{PDE}\underbrace{-\frac{V}{2}\Delta t\frac{\partial^2 C}{\partial t^2} - \frac{V}{2}\Delta x u\frac{\partial^2 C}{\partial x^2}}_{LTE}. \tag{5.26}$$

To combine all of the second order derivatives, the second derivative in time needs to be substituted with terms solely in space. To do this, an analysis similar to the derivation of the CFL number is performed [57]. First, the fission product transport equation is shown in

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} + C\frac{\partial u}{\partial x} - \nu\frac{\partial^2 C}{\partial x^2} = 0. \tag{5.27}$$

Taking the a first derivative in time of Eq. 5.27 and rearranging the second order in time derivative to the left and all other terms to the right, which is shown in Eq. 5.28, shows how the temporal LTE terms how to convert temporal derivatives to mix derivatives. Next, taking the frist derivative in space, which is shown in Eq. 5.29, and second derivative in space, which is shown in Eq. 5.30, provide the mixed terms necessary to convert the mixed derivatives in Eq. 5.27 to be a function of derivatives in space, which is shown in Eq. 5.31.

$$\frac{\partial^2 C}{\partial t^2} = -u\frac{\partial^2 C}{\partial t\partial x} - C\frac{\partial^2 u}{\partial t\partial x} + \nu\frac{\partial^3 C}{\partial t\partial x^2} \tag{5.28}$$

$$\frac{\partial^2 C}{\partial t\partial x} = -u\frac{\partial^2 C}{\partial x^2} - C\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^3 C}{\partial x^3} \tag{5.29}$$

$$\frac{\partial^2 C}{\partial t \partial x^2} = -u\frac{\partial^3 C}{\partial x^3} - C\frac{\partial^3 u}{\partial x^3} + \nu\frac{\partial^4 C}{\partial x^4} \tag{5.30}$$

$$\frac{\partial^2 C}{\partial t^2} = u^2\frac{\partial^2 C}{\partial x^2} + uC\frac{\partial^2 u}{\partial x^2} - u\nu\frac{\partial^3 C}{\partial x^3} - C\frac{\partial^2 u}{\partial t \partial x} - \nu u\frac{\partial^3 C}{\partial x^3} - \nu C\frac{\partial^3 u}{\partial x^3} + \nu^2\frac{\partial^4 C}{\partial x^4} \tag{5.31}$$

Eq. 5.31 is now substituted back into Eq. 5.26. Since we only want to understand how the leading order LTE terms impact the amount of diffusion in the problem, all non-second order derivatives are removed. These diffusive terms are shown in

$$-V\nu\frac{\partial^2 C}{\partial x^2} - \frac{V}{2}\Delta t u^2\frac{\partial^2 C}{\partial x^2} - \frac{V}{2}\Delta t u C\frac{\partial^2 u}{\partial x^2} + \frac{V}{2}\Delta t C\frac{\partial^2 u}{\partial t \partial x} - \frac{V}{2}\Delta x u\frac{\partial^2 C}{\partial x^2}. \tag{5.32}$$

Since diffusive terms in velocity do not directly impact the diffusiveness of the fission product concentration, these derivatives are also removed, which is shown in Eq. 5.33. These terms are split up into physical diffusion terms and numerical diffusive terms. When these two terms are present in the computational simulation, the result is an effective diffusive coefficient, ν_{model}, which is shown in Eq. 5.33. When the ratio between numerical diffusion and physical diffusion is low, the physical results dominate the solution, which is ideal for the experimental case. While this can be computationally expensive, it is necessary to avoid numerical bias in the experimental solution. This ratio is shown in Eq. 5.35. Using this equation and requiring it to be less than 7.5%, the time and space discretization for the experimental setup is determined to be $\Delta t = 0.06(s)$ and $\Delta x = 0.05(m)$.

$$-V\frac{\partial^2 C}{\partial x^2}\left(\underbrace{\nu}_{\text{Physical Diffusion}} + \underbrace{\frac{1}{2}\Delta t u^2 + \frac{1}{2}\Delta x u}_{\text{Numerical Diffusion}}\right). \tag{5.33}$$

$$\nu_{model} = \nu + \frac{1}{2}\Delta t u^2 + \frac{1}{2}\Delta x u. \tag{5.34}$$

$$\nu_{\text{ratio}} = \frac{\frac{1}{2}\Delta t u^2 + \frac{1}{2}\Delta x u}{\nu} \tag{5.35}$$

Now that the numerical error of the experimental results have been shown to be small, the experimental setup description can be described. The QoI being measured is the fission

product concentration at the point specified in Figure 5.1 as a function of time. This represents measuring the fission product concentration as a function of time at the site boundary in the full scale case.

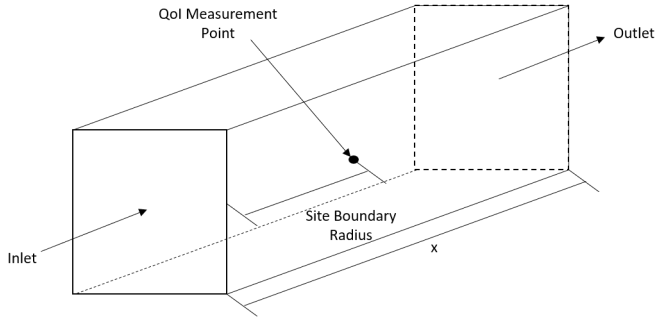

Figure 5.1: Visualization of the Experimental Setup

To mimic experimental measurement error, the height, velocity, and fission product concentration are distorted by adding normally distributed random noise. Since it is typical to have measurement error on the order of 1%, the magnitude of the noise is set to 1% of the first-order coefficient. This process is described using

$$QoI_{Measurement} = QoI\,(x,t) + \mathcal{N}\,(0,\sigma) \tag{5.36}$$

Based on the coefficients in Table 5.1 and the manufactured solutions in Eq. 5.23, Eq. 5.24, and the initial conditions in Eq. 5.25, the resulting scaled experimental values for the fission product concentration is shown in Figure 5.2. To understand how the measurement error impacts the experimental values, both the before and after measurement uncertainty data is shown.

Using these data sets, the ν_{model} in the code can be calibrated to the data set, which will change based on the ν_{ratio} value for a particular coarseness of the grid. This ν_{model} is then used in the prediction phase when the full scale computed simulation results are compared to the full scale simulation results.

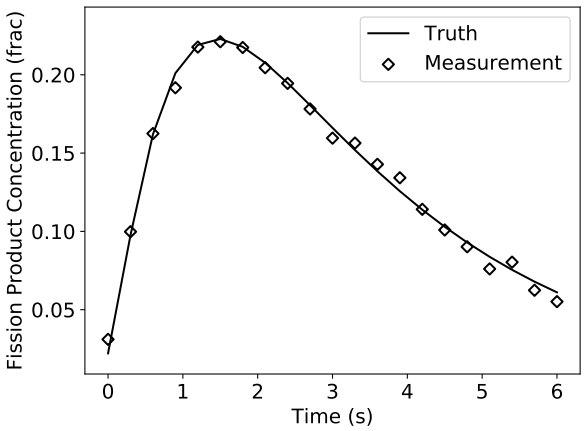

Figure 5.2: Scaled Experimental Data for Fission Product Concentration

5.2.1.3 Full Scale Experiment

For the full scale problem, each coefficient in the manufactured solution is based on the scaled setup. Specifically, the full scale dimensionalized values are defined from the small scale experiment non-dimensionalized values, the non-dimensional parameters Fr and Pe, and the full scale characteristic values. Based on these values, they are converted to the dimensionalized full scale values. The dimensionalized values, the characteristic values, and the non-dimensionalized values are shown in Table 5.2.

Table 5.2: Full Scale Settings

Quantity	Unit	Dimensional Value	Characteristic Value	Characteristic Value Equation	Non-Dimensional Value	Non-Dimensional Value Equation
x	$[Length]$	250.0	250.0	$x_0 = \frac{x}{1}$	1.000	$x^* = \frac{x}{x_0}$
t	$[Time]$	18.97	18.97	$t_0 = \frac{t}{1}$	1.000	$t^* = \frac{t}{t_0}$
h_1	$[Height]$	0.115	0.115	$h_{1_0} = \frac{h_1}{1}$	1.000	$h_1^* = \frac{h_1}{h_{1_0}}$
h_2	$[Height]$	0.0220	0.115	$h_{2_0} = h_{1_0}$	0.191	$h_2^* = \frac{h_2}{h_{2_0}}$
u_1	$\left[\frac{Length}{Time}\right]$	3.858	13.18	$u_{1_0} = \frac{x_0}{t_0}$	0.293	$u_1^* = \frac{u_1}{u_{1_0}}$
u_2	$\left[\frac{Length}{Time}\right]$	0.0421	13.18	$u_{2_0} = \frac{x_0}{t_0}$	0.003	$u_2^* = \frac{u_2}{u_{2_0}}$
c_1	$[Concentration]$	0.0001	0.500	$c_{1_0} = \frac{c_1}{1}$	0.0002	$c_1^* = \frac{c_1}{c_{1_0}}$
c_2	$[Concentration]$	0.500	0.500	$c_{2_0} = c_{1_0}$	1.000	$c_2^* = \frac{c_2}{c_{2_0}}$
k_h	$\left[\frac{1}{Length}\right]$	0.0312	0.004	$k_{h_0} = \frac{1}{x_0}$	7.800	$k_h^* = \frac{k_h}{k_{h_0}}$
k_u	$\left[\frac{1}{Length}\right]$	0.0316	0.004	$k_{u_0} = \frac{1}{x_0}$	7.900	$k_u^* = \frac{k_u}{k_{u_0}}$
k_C	$\left[\frac{1}{Length}\right]$	0.010	0.004	$k_{C_0} = \frac{1}{x_0}$	2.500	$k_h^* = \frac{k_C}{k_{C_0}}$
ω_h	$\left[\frac{1}{Time}\right]$	0.049	0.0527	$\omega_{h_0} = \frac{1}{t_0}$	0.930	$\omega_h^* = \frac{\omega_h}{\omega_{h_0}}$
ω_u	$\left[\frac{1}{Time}\right]$	0.032	0.0527	$\omega_{u_0} = \frac{1}{t_0}$	0.600	$\omega_u^* = \frac{\omega_u}{\omega_{u_0}}$
ω_C	$\left[\frac{1}{Time}\right]$	0.066	0.0527	$\omega_{C_0} = \frac{1}{t_0}$	1.260	$\omega_C^* = \frac{\omega_C}{\omega_{C_0}}$
δ	$[Dimensionless]$	0.100	1.000	$\delta_0 = 1$	0.100	$\delta^* = \frac{\delta}{\delta_0}$
$\mathbf{g}$	$\left[\frac{Length}{Time^2}\right]$	9.81	0.0003	$\mathbf{g}_0 = \frac{h_0}{t_0^2}$	$30,710$	$\mathbf{g}^* = \frac{\mathbf{g}}{\mathbf{g}_0}$
ν	$\left[\frac{Length^2}{Time}\right]$	34.79	$3,294$	$\nu_0 = \frac{x_0^2}{t_0}$	0.011	$\nu^* = \frac{\nu}{\nu_0}$

Using the same measurement error methodology in the small scale experiment and based on the coefficients in Table 5.2 and the manufactured solutions in Eq. 5.23, Eq. 5.24, and initial condition from Eq. 5.25, the resulting full scale experimental values are scaled up from the scaled experiments. The full scale fission product concentration true values and experimental values are shown in Figure 5.3.

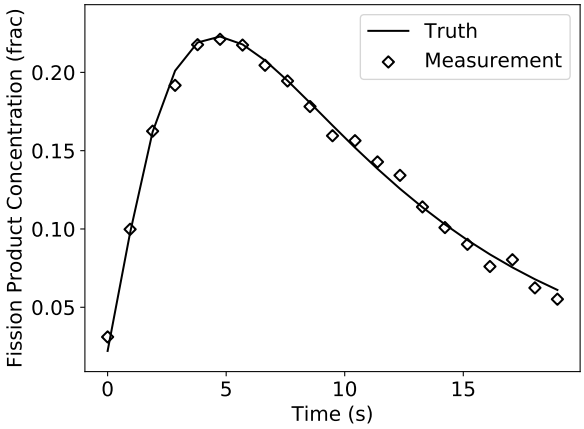

Figure 5.3: Full Scale Experimental Data for Fission Product Concentration

Since the full scale experiment only has a few data points, a recalibration of the model isn't feasible. Instead, it allows for determining the performance of the prediction at the full scale based on scaled experiment calibration. In the next section, two different computational frameworks are shown. One with all components of V&V and one with just calibration and validation.

5.2.2 Computational Solution

For this large scale prediction study, there are two different computational analyses that are run: one that follows all of the V&V processes and one that only completes calibration and validation. In an effort to have a variety of cases, the analysis that only completes calibration and validation have two different ν_{ratio}s to represent a variety of numerical error conditions. In addition, to study the effect of numerical error scaling, the Δx and Δt

scaling process is adjusted to scale incorrectly. This represents simulation software that adjusts the mesh on the fly if certain cells are ill-conditioned.

For both analyses, the computational simulation is computed using the implicit upwind numerical scheme. The boundary conditions and initial conditions for the shallow water equation are set by the manufactured solution. This provides a reasonable result for the height and velocity QoIs. The initial conditions for the fission product transport equation is given by Eq. 3.45. Results are saved such that the time dependent fission product concentration is able to be compared to experimental results. Using this simulation set up as well as the coefficients set up in Table 5.1 for the scaled simulation and Table 5.2 for the full scale simulation, the computational results are generated.

5.2.2.1 Proper V&V Process

To ensure the predictive capability of the simulations are adequate for the problem, all components of the V&V process are followed, except for SQA. This includes code verification, solution verification, calibration, and validation. This analysis represents the proper way to make computational predictions. It should be treated as the base analysis, while the other analysis represents realistic ways to computational simulations are used to incorrectly make predictions.

Code Verification The code verification technique used on the code is higher-order MEAMMS, which is presented in an earlier chapter. Higher-order MEAMMS provides confidence that the code doesn't include coding errors that are lower than the order of the numerical method, like regular MMS, but also coding errors that are of the same order and higher than the numerical method. This provides increased confidence that there are no coding errors in the code and reduces coding error to zero.

Solution Verification To ensure the numerical error doesn't impact the prediction, the k_{ratio} is set to 0.375 with a $\Delta x = 1.0$ and a $\Delta t = 0.3$. This ensures that the discretization

error is much smaller than the physical phenomenon. In addition, the numerical error is computed using GCI to quantify the numerical error.

Model Calibration and Validation The diffusion model is calibrated using the scaled case experimental data. The calibration used a least-squares minimization technique by adjusting ν to minimize the difference between the experimental and numerical solution. This results in a diffusive coefficient that well characterizes the physics of the problem. Once the calibrated ν is determined, the model form error is calculated using an L_2 norm metric, which is shown in

$$\varepsilon_{\text{MFE}} = \sqrt{\sum_{i=1}^{N} \left(QoI_{exp_i} - QoI_{comp_i}\right)^2} \tag{5.37}$$

where ε_{MFE} is the L_2 Norm of the model form error.

5.2.2.2 Improper V&V Process

To represent various missteps throughout the V&V process, two cases are used to represent realistic scenarios that degrade the predictive quality of simulation results, which are shown in Table 5.3. This includes skipping code verification, solution verification, and incorrectly scaling the numerical error. Below are the descriptions how these cases are sent up in the V&V context.

Table 5.3: Improper V&V Cases

Code Verification	Solution Verification	Mesh Scaling Degradation
None	$\nu_{\text{ratio}} \approx 0.68$	150%
None	$\nu_{\text{ratio}} \approx 1$	150%

Code Verification To represent a code with coding errors that would pass MMS, but not MEAMMS, a first-order error is added to the base code to represent a code that has not gone through rigorous code verification. These errors represent errors that in the small

scale case would not be blatant errors, but instead errors that produce reasonable, but incorrect results.

Solution Verification Since solution verification measures the numerical error, skipping this step could allow for a large numerical error to exist in the computed solution. To represent skipping this step, two different cases are computed. The first case is where Δx and Δt are set such that $\nu_{\text{ratio}} \approx 0.68$. This represents the case when the numerical error is of the same magnitude as the physics of the simulation. The second case is where Δx and Δt are set such that $\nu_{\text{ratio}} \approx 1$. This represents the case when the numerical error equals the physics of the simulation. Since solution verification is not completed, these errors would be unknown.

Model Calibration and Validation The model calibration and validation is identical to the method presented in the proper V&V procedure, but since the code has coding error and numerical error in the solution, the calibration process includes this in the computation result. This might at first appear to be a good thing, but since the calibrated model is making a prediction for the full scale case, this calibrated model doesn't represent the underlying physics.

The two cases where improper V&V is implemented represent a variety of realistic scenarios that happen when using real world software. In the next section, the proper V&V results are compared to the improper V&V results to show the impact on predictive quality.

5.3 Results

The results of both the scaled and full scale simulations are presented here. The first set of results are based on following the proper V&V process while the second set of results are based on following an improper V&V process. Table 5.4 shows all the simulations completed for this study.

Table 5.4: All Simulation Cases

Case	Scale	Proper V&V Process?	k_{ratio}	Mesh Scaling Degradation	Results Location
Case 1	Scaled	Yes	$\nu_{\text{ratio}} \approx 0.38$	None	Section 5.3.1
Case 2	Scaled	No	$\nu_{\text{ratio}} \approx 0.68$	None	Section 5.3.2
Case 3	Scaled	No	$\nu_{\text{ratio}} \approx 1.0$	None	Section 5.3.2
Case 4	Full Scale	Yes	$\nu_{\text{ratio}} \approx 0.38$	None	Section 5.3.3
Case 5	Full Scale	No	$\nu_{\text{ratio}} \approx 0.68$	150%	Section 5.3.3
Case 6	Full Scale	No	$\nu_{\text{ratio}} \approx 1.0$	150%	Section 5.3.3

Case 1 provides the ν_{Model} for Case 4, while Cases 2 and 3 provide ν_{Model} for Cases 5 and 6, respectively. To understand the impact of a proper V&V process, Case 4, 5, and 6 are compared to the full scale experimental data to determine the predictive capability loss of Cases 5 and 6 when a proper V&V process isn't followed.

5.3.1 Proper V&V Process

The proper V&V process involves code verification, solution verification, calibration, and validation. Each component of the V&V process measures or reduces the error due to coding error, numerical error, and model form error, respectively to each V&V process. The next sections provide the measurement of each of these errors.

5.3.1.1 Code Verification

For code verification, the code is tested using the MEAMMS code verification methodology. MEAMMS adds the theoretical LTE into the numerical solution with the idea that if it cancels out the observed LTE, the code doesn't have coding errors that are lower than the order of the LTE. When coding errors are minimized, the uncertainty due to coding errors is drastically reduced. Figure 5.4 and Figure 5.5 show the spatial errors at the last timestep of a MEAMMS analysis. Since the error reduction is quite large (greater than 10^{10}) when up to 20th order LTE is added to the numerical solution, the probability of coding errors is

practically zero. Therefore the uncertainty due to coding errors is also zero.

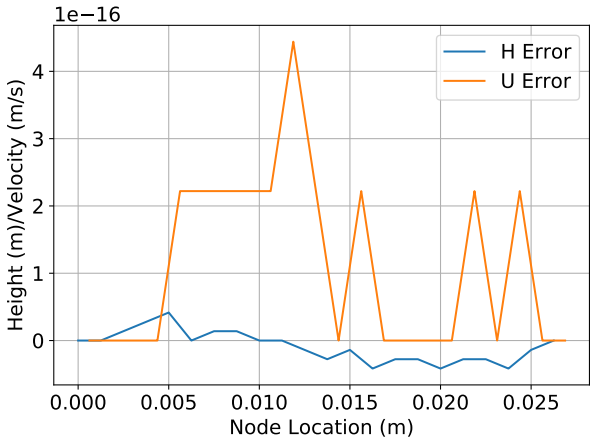

Figure 5.4: Code Verification for Height and Velocity for Proper V&V Case

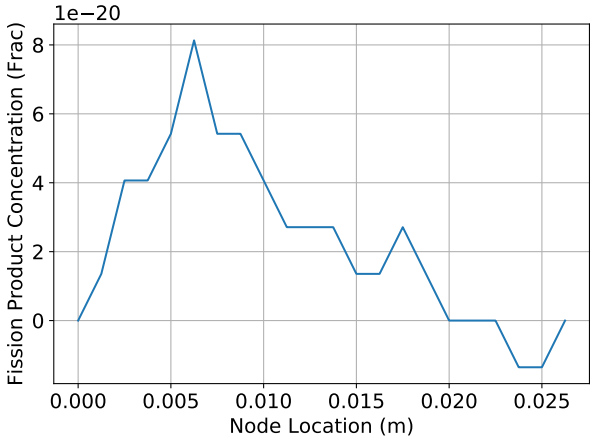

Figure 5.5: Code Verification for Fission Product Concentration for Proper V&V Case

5.3.1.2 Solution Verification

The next source of error that to quantify is numerical error. This is measured using the solution verification tool, GCI. This is widely accepted as the standard solution verification method because of it's simplicity and ease of use. GCI uses the rate at which the error is decreasing as the mesh sizing is reduced and determines an extrapolated solution when

the mesh size is zero. Using the difference between the extrapolated solution and the solution on the finest grid, this difference is multiplied by a factor of safety to add an error bound on the difference. This is then used as an error quantification of the numerical error (See Section 4.2.2 for derivation). For the proper V&V case, the GCI value is a very small number. This uncertainty is small relative to the scale of the numerical solution in Figure 5.6.

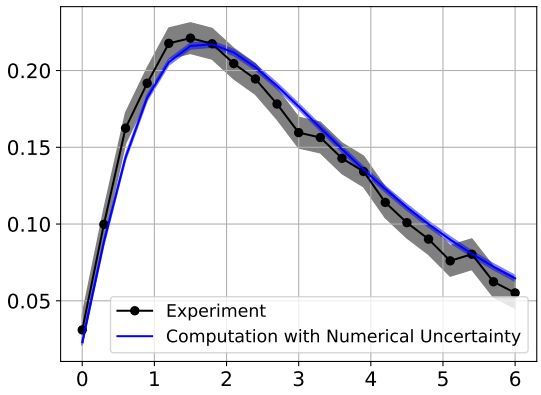

Figure 5.6: Solution Verification for Fission Product Concentration for Proper V&V Case

5.3.1.3 Model Calibration and Validation

The next step is to calibrate the computational model's diffusivity to experimental data. To do this, multiple different model diffusion coefficients are computed and then compared to experimental data. The coefficient that produces the lowest L_2 norm is used as the model's coefficient for the prediction case. Since the experimental diffusivity and the numerical diffusivity are known, an approximate estimate of the calibrated model diffusion coefficient can be calculated to minimize the calibration effort. The predicted diffusion coefficient is $\nu = 0.62$ with the actual calibrated diffusion coefficient is $\nu = 0.9$. The prediction is slightly off due to the lack of calibration on the advective term. The calibration test runs are shown in Figure 5.7.

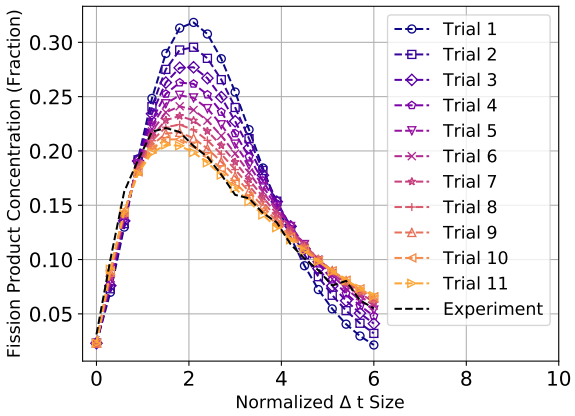

Figure 5.7: Calibration of k for Case 1

Now that the diffusion model is calibrated, the model form error is computed during the validation portion. For this validation analysis, the validation metric is the L_2 Norm using Eq. 5.37. The model form error is shown in Figure 5.8.

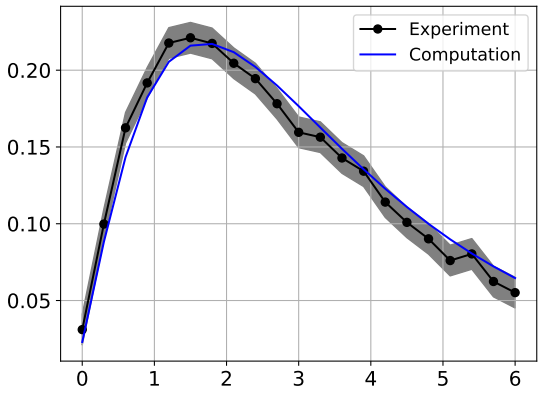

Figure 5.8: Validation for Case 1

5.3.2 Improper V&V Process

As with the proper V&V process, Cases 2 and 3's diffusion models are calibrated and model form error is quantified. However, code verification and solution verification activities are represented as if these activities were skipped to highlight their importance

for using computational models as a prediction tool. Below describes the V&V activities completed for the improper V&V cases (Case 2 and 3).

5.3.2.1 Code Verification

Since coding error is not assessed in the improper case, the code verification assessment isn't completed. This means that the coding error is incorrectly assumed to be zero. To reproduce the scenario of when this can impact the predictive capability of the simulation, Case 2 and Case 3 are run with the coding error described in the code verification chapter. This increases the coding error and unknown effects will occur when this code with the coding error is used to predict the full scale scenario (Cases 5 and 6).

5.3.2.2 Solution Verification

Since numerical error is not assessed in the improper case, the solution verification assessment isn't completed. This means that the numerical error is incorrectly assumed to be zero. To reproduce the scenario of when this can impact the predictive capability of the simulation, Case 2 has a large mesh size and therefore has a larger numerical error. Case 3 takes it a step further and even has a large mesh size than Case 2, so the numerical error is even larger. These large meshes push the solutions farther away from the experimental data, but the calibration process can reduce these effects. Unfortunately, this has unintended consequences when the full scale case is run.

5.3.2.3 Model Calibration and Validation

As with the proper V&V process, the improper V&V process cases' diffusion model is calibrated to the experimental data. For Case 2, the calibrated diffusion coefficient is $\nu = 0.8$ while the predicted calibrated diffusion coefficient is $\nu = 0.32$. For Case 3, the calibrated diffusion coefficient is $\nu = 0.6$ while the predicted calibrated diffusion coefficient is $\nu = 0.0063$. The calibration test cases for Case 2 and Case 3 are shown

in Figure 5.9 and Figure 5.10, respectively.

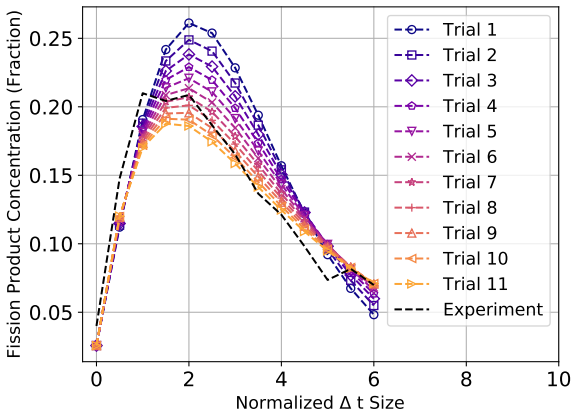

Figure 5.9: Calibration of ν for Case 2

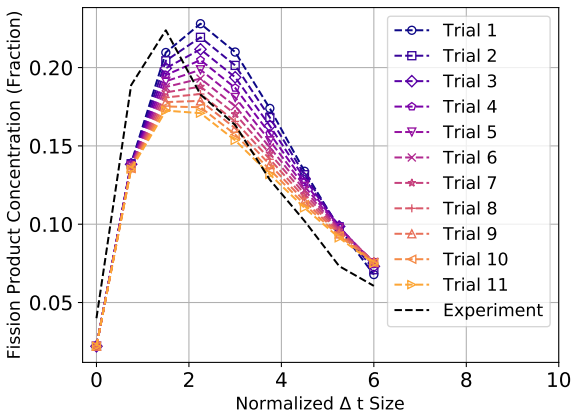

Figure 5.10: Calibration of ν for Case 3

Now that the models are calibrated, the model form error is calculated using the same methodology as the proper V&V case, which uses Eq. 5.37 to calculate the model form error. Based on this calculation, the model form error for Cases 2 and 3 are shown in Figure 5.11 and Figure 5.12, respectively.

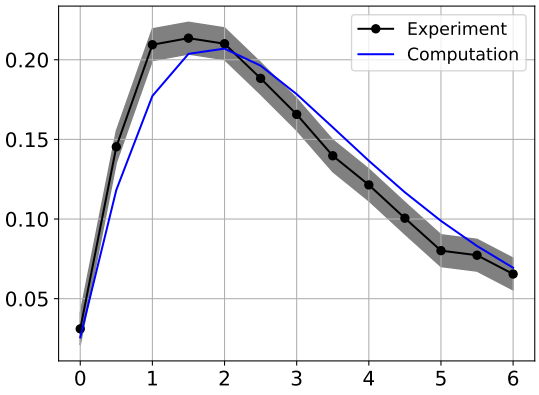

Figure 5.11: Validation for Case 2

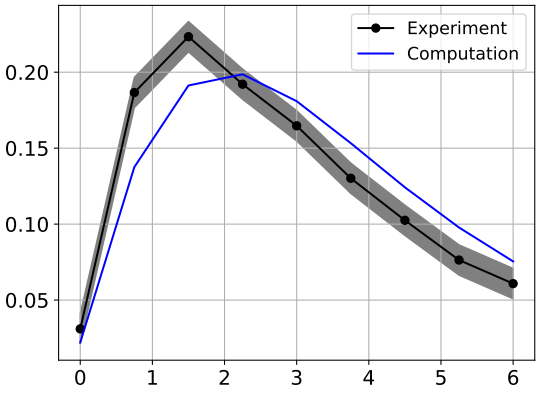

Figure 5.12: Validation for Case 3

5.3.3 Full Scale Comparison

To ensure the scaling is correctly completed, the full scale fission product concentration is compared to the scaled up version of the scaled fission product concentration. To scale up the scaled fission product concentration, a factor of 3.16 is applied to the time and a factor of 1.0 is applied to the concentration. As shown in Figure 5.13, the scaling is correctly done to 2 digits because the L_2 norm of the difference is 1.6%.

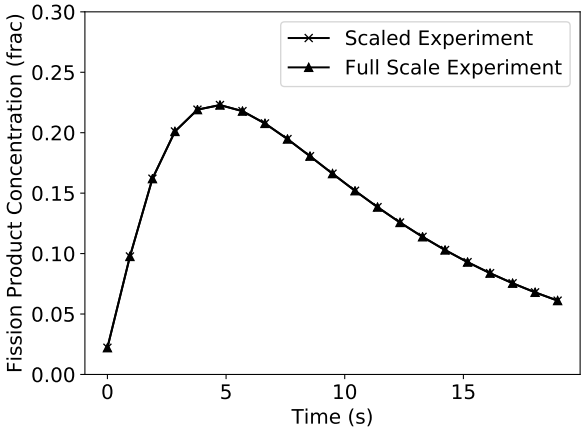

Figure 5.13: Comparison Between Full Scale and Scaled Up Scaled Fission Product Concentration

Based on the results from the previous sections, the calibrated model diffusion coefficients are shown in Table 5.5. To show how the calibration process cannot tell the difference between diffusion from the model diffusion and numerical diffusion, each of cases are compared to show how the calibration performed, which is shown in Figure 5.14.

Table 5.5: Scaled Calibrated Coefficients

Case	Diffusion Model Coefficient
Case 1	0.9
Case 2	0.8
Case 3	0.6

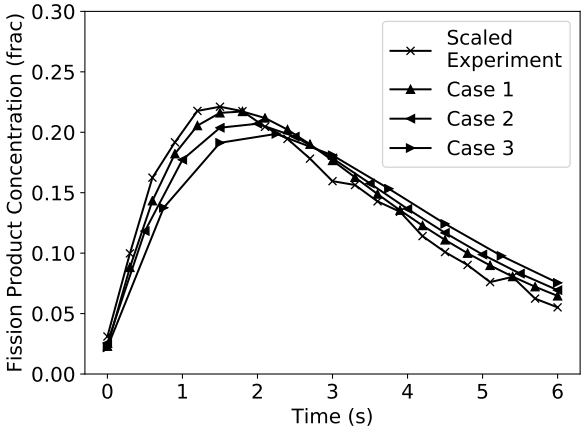

Figure 5.14: Scaled Comparison Between Cases 1, 2, and 3

This shows that the calibrated model diffusion coefficient canceled out the numerical diffusion. This cancellation is difficult to scale, so for the prediction cases, the numerical diffusion either needs to be low or scaled properly. While the numerical diffusion is low for Case 4, it is much larger for Cases 5 and 6. Also, the numerical diffusion for Cases 5 and 6 do not scale properly since there was no effort to quantify the numerical error during the solution verification portion of V&V activities. Based on the scaling factor of 31.62, the full scale calibrated diffusion coefficients and $L\infty$ Norm errors are shown in Table 5.6. Using these diffusion model coefficients, the full scale fission product concentration predictions for Cases 4, 5, and 6 are shown in Figure 5.15.

Table 5.6: Full Scale Calibrated Coefficients

Case	Diffusion Model Coefficient
Case 4	28.46
Case 5	25.30
Case 6	18.97

While Case 4 predicts the full scale experimental data well, Case 5 and especially

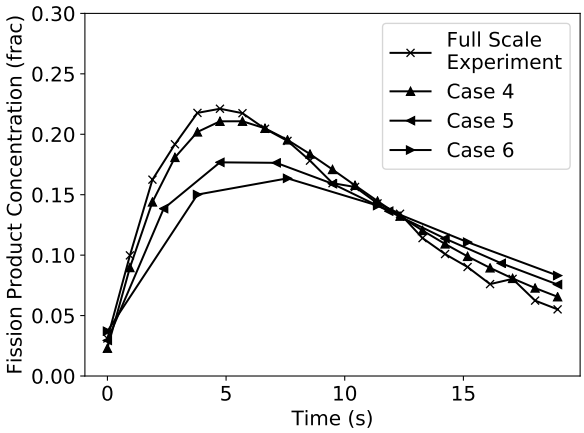

Figure 5.15: Full Scale Comparison Between Cases 4, 5, and 6

Case 6 under predicts the fission product concentration. This has large ramifications when the fission product concentration is used to calculation radiation dose at a reactor site boundary. Since the fission product concentrations are under predicted, this means the radiation dose is under predicted, which has serious public health consequences. Using this as an example, it shows that skipping code and solution verification can have serious consequences.

5.4 Conclusion and Future Work

Engineers use computational codes to make predictions where experimental data is not available due to cost, time, or public safety. To ensure that these predictions are credible, verification and validation are used to ensure errors are either reduce significantly or quantified to ensure that the errors will not impact decision making. Historically, validation has been viewed as the more necessary than verification, but this is not the case when scaling is involved. This is because when scaling is involved, predictiveness is critical and all uncertainties need to be characterized. As shown in the presented example, when verification activities are not completed, the predictive quality is reduced. This is because calibration changes model coefficients to offset the effects of coding errors or numerical errors.

Where these errors are offset and applied to the application scale, the numerical errors are not guaranteed to scale properly without quantifying the errors. As shown in Figure 5.15, the predicted fission product concentration when both verification and validation are used (Case 4) has a higher prediction quality than when verification is skipped (Cases 5 and 6). Therefore, code and solution verification can not be ignored and should be of equal importance to calibration and validation.

While this is a proof of concept, future work would include applying this to a real experiment and using production level software would add additional confidence that verification should be of importance to calibration and validation. Additionally, adding uncertainty quantification activities to this example would provide even more evidence that the uncertainty for Case 4 is computed correctly while would be severely under-predicting the uncertainty in Cases 5 and 6.

6. CONCLUSION

With the recent increase in computational power and the rise in cost of experiments, scientists and engineers have increased the use of computational fluid dynamics (CFD) simulations and high fidelity system analysis codes to solve nuclear engineering problems. Because of the increase in reliance of simulations and lower reliance on experiments, credibility of these simulations is of utmost importance. Therefore, code and solution verification play an important role in assessing credibility of these simulations.

In this work, the author examines various aspects of addressing the credibility in complexity of CFD and system codes. In chapter 3, a new, more rigorous code verification method is developed that is based on the method of manufacturing solutions (MMS). This new methods works by testing additional information in the code, namely the local truncation error (LTE). This additional code verification testing increases the confidence that coding errors do not exist within the code. In chapter 4, a solution verification method, GCI, is evaluated in newly defined solutions regimes: inside, near, and outside the asymptotic region. This allows for the solution verification method to be evaluated for different discretization sizes. This is important to evaluate because for complex reactor geometry, the range of spatial scales is quite large, making it difficult to have an ideal discretization grid that resolves the physics while not requiring a large amount of computational time. In chapter 5, a study is performed to evaluate the credibility of nuclear engineering-relevant simulations when code and solution verification is ignored or improperly conducted. In the first scenario where code and solution verification is performed, the prediction at the application-scale is accurate. In the second scenario where code and solution verification is not performed, the prediction at the application-scale is not accurate. This means that when code and solution verification is performed, the credibility of the sim-

ulations is much better than the credibility of the simulations without code and solution verification. Below is a more complete summary of the code verification method development work, solution verification analyzation work, and the prediction comparison work.

Code verification is an important part of building credible simulations. The current state-of-the-art is to use the method of manufactured solutions (MMS) to compare the observed order of accuracy with the theoretical order of accuracy. While this method is useful for finding some coding errors, MMS can not identify all coding errors. Therefore, it is important to develop a code verification method to identify the coding errors that MMS can not. Using the cobination of MMS and MEA to develop MEAMMS, the observed LTE has to match the theoretical LTE. This means that MEAMMS tests more of the characteristics than MMS. Additionally, higher order MEAMMS verifies the validity of lower order MEAMMS by showing that the process is mathematically consistent. Additionally, the study shows an example of when MMS cannot identify a coding error, but MEAMMS can. This work has produced a conference paper [14] and an accepted journal paper [55].

While this work identified a problem and proposed a solution to that problem, the example problems have been less than realistic. Future work will be to apply MEAMMS to more realistic test problems. This would start to address the question of how to apply MEAMMS to a production code.

Now that a more rigorous method of code verification is developed, evaluating solution verification methods in different regimes in an analytical way is an additional step in producing credible simulations. The most used solution verification method, the grid convergence index (GCI), is assessed inside, near, and outside the asymptotic region. The key evolution of this work is a mathematical definition of the start of the asymptotic region as well as outside the asymptotic region. Using the LTE calculated from the manufactured solution, the asymptotic point is calculated where the leading LTE is equal to the higher order LTE. This is the start of where the discretization goes from chaotic behavior to con-

vergent behavior. The start of the asymptotic region is then defined when the leading LTE is one magnitude larger than the higher order LTE. This allows for evaluating the GCI solution verification method in different convergence regions.

The reason for evaluating GCI in these regions is that real world problems often either cannot identify if the solution is inside the asymptotic region or obtaining a solution inside the asymptotic region is too computationally demanding. Therefore, understanding how solution verification methods perform in all regimes is important for producing credible simulations.

This work has produced a conference paper [58] with a journal paper in the works. Future work is to include more than one solution verification method in the analysis and to generate more complex data with differing length scales and time scales.

To additionally show the importance of code and solution verification in producing credible simulations, two simulation strategies are compared. In the first simulation strategy, code and solution verification is completed before calibration and validation. In the second simulation strategy, code and solution verification is skipped and jumped straight to calibration and validation. The test problem is to calibrate the turbulence model for a fission product transport simulation. For the case where code and solution verification is performed, coding error and numerical uncertainty is minimized first. Then calibration and validation is performed. This simulation is valid in both interpolating between scaled data and extrapolating in the full scale case.

For the case where code and solution verification is skipped, coding errors are added into the solution and a coarse computational grid is used during the scaled calibration and validation process. This approach matches the scaled data well, but not the full scale case. The resulting fission product concentration at the site boundary is correctly predicted by the case where code and solution verification is performed, while the fission product concentration at the site boundary is under predicted by the case where code and solution

verification is skipped. This shows that when code and solution verification is skipped, the credibility of the simulations suffers.

From the results shown in this work, it is clear that code and solution verification play an important role in the credibility of simulations. The importance of code and solution verification will increase as full scale experimental data becomes less available for advanced reactors, which increases the reliance on simulations. As simulations results are used more to make decisions, it is crucial that robust code and solution verification methods are used to minimize coding error and numerical uncertainty.

REFERENCES

[1] W. L. Oberkampf and C. J. Roy, *Verification and validation in scientific computing*. Cambridge, United Kingdom: Cambridge University Press, 2010. doi:10.1017/CBO9780511760396.

[2] P. J. Roache, "Perspective: a method for uniform reporting of grid refinement studies," *Transactions-American Society of Mechanical Engineers Journal of Fluids Engineering*, vol. 116, pp. 405–405, 1994.

[3] ASME, "Guide for verification and validation in computational solid mechanics," *American Society of Mechanical Engineers*, ASME Standard V&V 10-2006, New York, NY.

[4] L. F. Richardson, "The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 210, pp. 307–357, 1911. doi:10.1098/rsta.1911.0009.

[5] L. F. Richardson, B. J Arthur Gaunt, *et al.*, "VIII. the deferred approach to the limit," *Phil. Trans. R. Soc. Lond. A*, vol. 226, no. 636-646, pp. 299–361, 1927. doi:10.1098/rsta.1927.0008.

[6] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, pp. 50–67, Cambridge University Press, 1947. doi:10.1007/BF02127704.

[7] P. M. Knupp, C. C. Ober, and R. B. Bond, "Impact of coding mistakes on numerical error and uncertainty in solutions to PDEs," tech. rep., Sandia National Laboratories Report SAND2017-5341, 2007.

[8] L. Humphries, "MELCOR Quality Assurance Practices," in *MELCOR Software Quality Assurance Training*, 2011.

[9] L. Humphries, "MELCOR Code V&V and SQA," in *Workshop on Operational Experience and Advances in MELCOR Modeling, Shenzhen, China, November 19-23, 2012*.

[10] L. Humphries and J. Reynolds, "MELCOR Quality Assurance Practices," in *MELCOR Software Quality Assurance Training*, 2014.

[11] A. Abran, J. W. Moore, P. Bourque, R. Dupuis, and L. L. Tripp, *Guide to the software engineering body of knowledge: 2004 version SWEBOK*. IEEE Computer Society, 2004.

[12] S. Eddins, "Taking control of your code: Essential software development tools for engineers," in *International Conference on Image Processing, Atlanta, GA, Oct*, vol. 9, 2006.

[13] B. Kleb and W. A. Wood, "Computational simulations and the scientific method," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, no. 6, pp. 244–250, 2006.

[14] A. M. Krueger, V. A. Mousseau, and Y. A. Hassan, "Rigorous code verification: An additional tool to use with the method of manufactured solutions," in *ASME 2019 Verification and Validation Symposium*, 2019. doi:10.1115/VVS2019-5166.

[15] P. J. Roache, "Code verification by the method of manufactured solutions," *Transactions of the American Society of Mechanical Engineers Journal of Fluids Engineer-*

ing, vol. 124, no. 1, pp. 4–10, 2002. doi:10.1115/1.1436090.

[16] S. Steinberg and P. J. Roache, "Symbolic manipulation and computational fluid dynamics," *Journal of Computational Physics*, vol. 57, no. 2, pp. 251–284, 1985. doi:10.1016/0021-9991(85)90045-2.

[17] W. Oberkampf, F. Blottner, and D. Aeschliman, "Methodology for computational fluid dynamics code verification/validation," in *Fluid Dynamics Conference*, p. 2226, 1995.

[18] A. M. Krueger, V. A. Mousseau, and Y. A. Hassan, "Adding confidence to solution verification: Using MMS-informed MEA to better understand discretization error," in *2018 ANS Winter Meeting and Expo*, 2018.

[19] C. W. Hirt, "Heuristic stability theory for finite-difference equations," *Journal of Computational Physics*, vol. 2, no. 4, pp. 339–355, 1968. doi:10.1016/0021-9991(68)90041-7.

[20] N. J. Cyrus and R. E. Fulton, "Accuracy study of finite difference methods," Tech. Rep. NASA TN D-4372, National Aeronautics and Space Administration, 1968.

[21] T. Xing and F. Stern, "Factors of safety for richardson extrapolation," *Journal of Fluids Engineering*, vol. 132, no. 6, p. 061403, 2010.

[22] L. Eça and M. Hoekstra, "Discretization uncertainty estimation based on a least squares version of the grid convergence index," in *Proceedings of the Second Workshop on CFD Uncertainty Analysis, Instituto Superior Tecnico, Lisbon, Oct*, 2006.

[23] L. Eça and M. Hoekstra, "A procedure for the estimation of the numerical uncertainty of cfd calculations based on grid refinement studies," *Journal of Computational Physics*, vol. 262, pp. 104–130, 2014.

[24] W. Rider, W. Witkowski, J. R. Kamm, and T. Wildey, "Robust verification analysis," *Journal of Computational Physics*, vol. 307, pp. 146–163, 2016.

[25] G. A. Radtke, N. Martin, C. H. Moore, A. Huang, and K. L. Cartwright, "Robust verification of stochastic simulation codes," *Journal of Computational Physics*, Submitted in 2020.

[26] A. M. Krueger, "Estimation of discretization error for three dimensional CFD simulations using a taylor series modified equation analysis," Master's thesis, Texas A&M University, 2017.

[27] R. Shaw, T. Larson, and R. Dimenna, "Development of a phenomena identification and ranking table (PIRT) for thermal-hydraulic phenomena during a pwr lbloca," *NUREG: CR-5074, EG&G, Idaho*, 1988.

[28] J. P. Yurko and J. Buongiorno, "Quantitative phenomena identification and ranking table (QPIRT) for bayesian uncertainty quantification," *2012 International Congress on Advances in National Power Plants (ICAPP '12)*, June 24-28 2012.

[29] H. Luo, *Quantified PIRT and uncertainty quantification for computer code validation*. PhD thesis, Oregon State University, 2012.

[30] W. L. Oberkampf and B. L. Smith, "Assessment criteria for computational fluid dynamics model validation experiments," *Journal of Verification, Validation and Uncertainty Quantification*, vol. 2, no. 3, p. 031002, 2017.

[31] R. C. Schmidt, "A review of NRC regulatory requirements and statements and statements concering verification, validation, and uncertainty quantification of computer codes used in support of nuclear reactor license applications," Sandia National Laboratories Internal Letter Report, NEAMS FY09 Level 5 Milestone Report, Sept. 2009.

[32] RELAP5 Development Team, "RELAP5/MOD3 Code Manual Vol.1 Rev. 1," NUREG/CR-5535 report, Idaho National Laboratory, August 1995.

[33] G. Mesina, "A history of RELAP computer codes," *Nuclear Science and Engineering*, vol. 182, no. 1, pp. v–ix, 2016.

[34] C. L. Smith, Y.-J. Choi, and L. Zou, "RELAP-7 Software Verification and Validation Plan," inl/ext-14-33201, Idaho National Laboratory, September 25, 2014.

[35] J. Yoo and Y.-J. Choi, "RELAP-7 Software Verification and Validation Plan: Requirements Traceability Matrix (RTM) Update and Code Verification Strategy," INL/EXT-17-43199, Idaho National Laboratory, September 2017.

[36] A. Prosek and B. Mavko, "Evaluating code uncertainty—i: using the CSAU method for uncertainty analysis of a two-loop PWR SBLOCA," *Nuclear Technology*, vol. 126, no. 2, pp. 170–185, 1999.

[37] B. Boyack, I. Catton, R. Duffey, K. Katsma, G. Lellouche, S. Levy, G. Wilson, and N. Zuber, "Quantifying reactor safety margins part 1: an overview of the code scaling, applicability, and uncertainty evaluation methodology," *Nuclear Engineering and Design*, vol. 119, no. 1, pp. 1–15, 1990.

[38] L. Humphries, B. Beeny, F. Gelbard, D. Louie, and J. Phillips, "MELCOR Computer Code Manuals Vol. 2.2.9541: Reference Manual," SAND report 2017-0455, Sandia National Laboratories, January 2017.

[39] C. D. L. E. al., "MELCOR Validation and Verification 1986 Papers," SAND report 86-2689, Sandia National Laboratories, 1986.

[40] L. N. Kmetyk, "MELCOR Assessment: Gedanken Problems Volume 1," SAND report 92-0762, Sandia National Laboratories, 1992.

[41] R. M. Summers and R. K. C. Jr., "Diagnosis and Resolution of Numerical Sensitivities in MELCOR," in *Presented at 20th Water Reactor Safety Information Meeting, Bethesda, MD, October 22, 1992*.

[42] L. Humphries, "Quicklook Overview of Model Changes in MELCOR 2.2: Rev 5342 to Rev 9496," SAND report 2017-5599, Sandia National Laboratories, 2017.

[43] R. O. Gauntt, "Exercises in Severe Accident Analysis using MELCOR: Accident Walkthrough," in *Presented at IAEA Workshop on Models and Methods For Calculating Severe Accident Source Terms for AP1000, Haiyang, China, April 27-May 1, 2015*.

[44] K. Fernández Cosials, *Analysis and improvement of hydrogen mitigation strategies during a severe accident in nuclear containments*. PhD thesis, Industriales, 2017.

[45] T. L. George and A. Singh, "Separate effects tests for GOTHIC condensation and evaporative heat transfer models," *Nuclear Engineering and Design*, vol. 166(3), pp. 403–411, 1996.

[46] B. Beeny, R. Vaghetto, K. Vierow, and Y. Hassan, "MELCOR and GOTHIC analyses of a large dry PWR containment to support resolution of GSI-191," *Nuclear Technology*, vol. 196, no. 2, pp. 292–302, 2016.

[47] Siemens, *STAR-CCM+ Users' Guide*. 12.02.010-r8 ed., 2017.

[48] L. Gilkey, "STAR-CCM+ (CFD) calculations and validation," Sandia Technical Report SAND2017-12545 R, CASL, 2017.

[49] N. Gordon, "CTF (subchannel) calculations and validation," Sandia Technical Report SAND2017-12874 R, CASL, 2017.

[50] C. J. Freitas, "Editorial," *Journal of Fluids Engineering*, vol. 115, no. 3, pp. 339–340, 1993.

[51] J. Fokken, B. Krohn, R. Kapulla, B. Niceno, H. Prasser, and A. Badillo, "OECD/NEA CFD-UQ benchmark exercise: CFD prediction and uncertainty quantification of a GEMIX mixing layer test—final report," 2017.

[52] American Society of Mechanical Engineers (ASME), "Standard for verification and validation in computational fluid dynamics and heat transfer," Tech. Rep. V&V 20-2009, American Society of Mechanical Engineers, New York, USA, 2009.

[53] A. C. Rakhimov, D. Visser, and E. Komen, "Uncertainty quantification method for cfd applied to the turbulent mixing of two water layers," *Nuclear Engineering and Design*, vol. 333, pp. 1–15, 2018.

[54] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Berlin Heidelberg: Springer Science & Business Media, 3 ed., 2009. doi:10.1007/b79761.

[55] A. Krueger, V. Mousseau, and Y. Hassan, "LTE-informed code verification," *Journal of Verification, Validation, and Uncertainty Quantification*, Accepted in 2020.

[56] L. Eça, C. M. Klaij, G. Vaz, M. Hoekstra, and F. S. Pereira, "On code verification of RANS solvers," *Journal of Computational Physics*, vol. 310, pp. 418–439, 2016. doi:10.1016/j.jcp.2016.01.002.

[57] R. Courant, K. Friedrichs, and H. Lewy, "On the partial difference equations of mathematical physics," *IBM Journal of Research and Development*, vol. 11, pp. 215–234, March 1967. doi:10.1147/rd.112.0215.

[58] A. M. Krueger, V. A. Mousseau, and Y. A. Hassan, "Characterization of solution verification," in *ASME 2018 Verification and Validation Symposium*, 2018.