ABSTRACT

Title of Document: FUNDAMENTAL THEORY OF SCIENTIFIC COMPUTER SIMULATION REVIEW

Joshua S. Kaizer, Doctor of Philosophy, 2013

Directed By: Professor Marino di Marzo
Department of Mechanical Engineering

"How much work is required to trust the results of a scientific computer simulation when human lives are at stake?" For many, this question is purely academic. But for those dealing with high consequence simulations, such as the simulations performed to demonstrate the safety of a nuclear power plant, the question is very pertinent and answering it is the domain of scientific computer simulation review. While simulation review has been performed for many years, it is rarely seen as a field of study in its own right. Consequently, requirements are often developed for a specific simulation and a particular use. If either the simulation or the use changes, new requirements must developed. To help solve this problem, some fundamental theory of scientific computer simulation review is proposed. This fundamental theory included the generation of a basic vocabulary, a formalization of the concept of maturity, the creation of simulation hierarchy, and the development of an assessment framework.

A basic vocabulary was generated to define the many common and important concepts used in simulation review. By focusing on review in general, the resulting vocabulary captures many of the ideas important to all simulation reviews and provides a better means of discussing the trustworthiness of their results. The concept of maturity was formalized into Maturity Theory. This theory provides a detailed analysis of maturity, which was used to better understand the tools available in simulation review. The Hierarchy for Scientific Computer Simulations was created to capture the various components of a generic simulation and define the relationships between those components. This Hierarchy provides a methodology which can be used to organize and represent many simulations. The Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations was developed using the concepts from Maturity Theory and the Hierarchy. This framework provides a structure which establishes the boundaries of simulation review, highlights many of the assumptions of a simulation which need to be supported if the results of the simulation are to be trusted, and establishes a method for its continually evolution. By contributing this fundamental theory, these advancements better established the field scientific computer simulation review.

FUNDAMENTAL THEORY OF SCIENTIFIC COMPUTER SIMULATION REVIEW

By

Joshua Stephen Kaizer

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Marino di Marzo, Chair
Professor James Baeder, Deans Representative
Professor Kenneth Kiger
Professor Gary Pertmer
Dr. Ralph Landry
Dr. William Oberkampf

Dedication

To Gilbert who rightly said,

The poet only asks to get his head into the heavens. It is the logician who seeks to get the heavens into his head. And it is his head that splits. (Orthodoxy, 1908)

Acknowledgements

I want to thank Dr. Kevin Heller who has been the greatest help to me in writing this dissertation. Through many hours of discussion, reading, and comments, he greatly improved my efforts. Additionally, I owe Dr. Anthony Attard and Andrew Ireland a debt of gratitude for the many hours of conversation and feedback they provided.

None of this could have been possible without Dr. Lawrence Hochreiter. It was his original concern which put my feet on this path and his guidance which gave me the tools to pursue this degree. I would like to thank Dr. Marino di Marzo for advising me through in this process and Dr. Ralph Landry for his many insights into my work. Additionally, I would like to thank Dr. William Oberkampf whose patience and knowledge have guided me through this process. He helped a young and hopeful engineer develop his ideas into a dissertation.

I would like to thank the U.S. Nuclear Regulatory Commission for providing me with the opportunity to perform this research. I would like to thank the management of NRC, chiefly Anthony Mendiola, William Ruland, and Eric Leeds, who gave me this opportunity and the management of Sheena Whaley and Tim McGinty who continued in supporting me. My coworkers in Nuclear Performance and Code Review bore the burden of my absence and to them I am especially grateful.

I would like to thank my father for always encouraging me and believing that I can succeed even though I did not always believe it myself. I would like to thank my mother who always told me "You can't trust anybody but Jesus, sucker" … which is a philosophical corner stone of this dissertation. I would like to especially thank Krista, my wife, for taking over the parenting of our baby daughter Anastasia while I was away writing and supporting me through this process. Her support was more necessary than any other. Finally, I would like to thank Anastasia for learning that getting a good night's sleep requires not whacking yourself in the face.

Most importantly, this dissertation will have its fair share of mistakes, blunders, and errors and one man is to blame, me.

Nomenclature

AS	Assignment Statement
CC	Computer Code
CG	Coded Group
CPF	Coded Physical Function
CSC	Complete Set of Codes
CT	Complex Term
FPE	Final Physical Equation
$\mathbf{I_{AS}}$	Input to the Assignment Statement
$\mathbf{I_{CC}}$	Input to the Computer Code
$\mathbf{I_{CG}}$	Input to the Coded Group
$\mathbf{I_{CPF}}$	Input to the Coded Physical Function
$\mathbf{I_{CSC}}$	Input to the Complete Set of Codes
$\mathbf{O_{AS}}$	Output from the Assignment Statement
$\mathbf{O_{CC}}$	Output from the Computer Code
$\mathbf{O_{CG}}$	Output from the Coded Group
$\mathbf{O_{CPF}}$	Output from the Coded Physical Function
$\mathbf{O_{CSC}}$	Output from the Complete Set of Codes
OPE	Original Physical Equation
SciCS	Scientific Computer Simulation
ST	Simple Term
T	Term

Table of Contents

List of Tables

List of Figures

1. Introduction to Scientific Computer Simulation Review

This chapter introduces the reader to the field of scientific computer simulation review. The first section is used to define a scientific computer simulation. Then in the next section, scientific computer simulation review is formally defined and discussed. In the third section, a brief history of simulations and simulation review is given. This is followed by a survey of the current research being done in simulation review, focusing especially on maturity assessment frameworks. Finally, this chapter concludes with a summary of the contributions made by this dissertation to the field of scientific computer simulation review.

1.1. What is a Scientific Computer Simulation?

The jargon of the Modeling and Simulation (M&S) community is similar to the jargon of any other community where terms commonly used take on different shades of meaning depending on who is speaking and to whom. Sometimes, the terms are used to communicate a very specific concept, and other times the same terms are used to communicate a general idea. Thus, any one term can have multiple definitions, and the definition the author intended may not always be discernible from the context. Such a case would certainly create confusion.

The following is such a list of terms whose precise definitions may vary from user to user. At a minimum, the reader should be vaguely familiar with some of these terms.

- analyst
- analytical model
- calculation
- calculational methodology
- case
- code
- code analysis
- code review
- computation
- computational fluid dynamics
- computational models
- computational physics
- computer analysis
- computer calculation
- computer code
- computer model
- computer program
- computer simulation
- conceptual model
- deck
- empirical model
- evaluation model
- first principle model
- mathematical models
- methodology
- model
- modeling and simulation
- physics model
- run
- scientific computer simulation
- sensitivity analysis
- simulation
- source code
- submodel
- validation
- verification
- uncertainty quantification

To avoid potential confusion, none of the above words (or their like) will be used without being first precisely defined[1]. Where appropriate, the author has attempted to choose definitions which would have broad agreement among the M&S community. At the very least the definitions will be apt for communicating the ideas of this dissertation. To lay the foundations for these definitions, it is appropriate that the first words defined will be those of "model" and "simulation" themselves.

1.1.1. Introduction to Model and Simulation

A universal definition of the word "model" is very hard to find. This is likely due to its use in a wide variety of fields as demonstrated by its Wikipedia page ("Model" 2013). This webpage contains no common definition for the word and is only a list of links to specific uses of the word model (e.g., mathematical model, scale model, building model, model aircraft, car model, miniature model, conceptual model, business model, economic model, etc...).

Because of this wide variability, many M&S texts begin not with a definition of a model but with a definition of a system (Neelamkavil 1987, Bossel 1994, and Zeigler, Praehofer, and Kim 2000). Such texts will then define a model in terms of a system. For example, Neelamkavil defines a **system** as "a set or assemblage of entities (elements or components) interrelated to each other and to the whole as to achieve a common goal." He then defines a **model** as a "simplified representation of a system (or process or theory) intended to enhance our ability to understand, predict, and possibly control the behavior of the *system*" (emphasis mine). Such definitions do address the wide variability that is associated with the term "model" (e.g., how someone who wears fashionable clothes and walks up and down a runway relates to a system described by mathematical equations). While such broad definitions are necessary when discussing any possible type of model, they are not needed when the discussion is focused, as is this discussion on scientific computer simulations. Maki and Thompson (2006) take a more focused approach by not attempting to define the general term "model", but start their definitions with a "mathematical model". Not all models are mathematical models, but as those are the only ones Maki

[1] When a word is defined, the lexicographer faces a dilemma: should the definition of word reflect how it is actually used (descriptive) or should the definition prescribe how the word should be used (prescriptive)? For those wishing to capture the common meanings of words and phrases, this is quite a dilemma, but for scientific works such as this, it is less so. The goal of this dissertation is to communicate an idea, not publish a list of words with universal agreement as to their definitions. While it is certainly advantageous to use words and concepts that are already known to the audience, it is more important that the ideas in this dissertation are clearly communicated. Thus, many of the definitions in this dissertation may not be common to other works, although the author has attempted to use common definitions where possible.

and Thompson are interested in, they are the only ones that are focused on. This dissertation takes a hybrid of the two approaches, by providing a general definition for "model" but not as general as Neelamkavil.

In general, a **Model** is a representation of the behavior of something. Likewise, a **Simulation** is an imitation of the behavior of something. The important distinction drawn here between "model" and "simulation" is the distinction between representation of a behavior and the imitation of a behavior. A model is the representation. It continually exists as a representation, whether written down on paper, coded into a computer language, or as a figment of someone's imagination. A model can be changed, but does not change by its own accord. On the other hand, the simulation is the imitation. The simulation is the exercise of the model with certain inputs. The simulation can be the calculation on the back of an envelope, the result of a computer run, or what someone believes will happen. These concepts are simple, but they are the necessary foundation for more complex concepts, such as that of a scientific computer simulation.

1.1.2. Introduction to Scientific Computer Simulation

One way to understand a scientific computer simulation is to first understand a simulation and then understand the modifiers "scientific" and "computer". Understanding the first modifier "scientific" is straight forward. This modifier implies that the "something" whose behavior is represented or imitated has to do with the physical universe and the way it behaves. Thus a **Scientific Model** is a representation of the behavior of something in the physical universe (e.g., using a string tied at one end to represent a sound wave). Likewise a **Scientific Simulation** is an imitation of the behavior of something in the physical universe (e.g., moving the string to imitate sound hitting a wall).

Understanding the second modifier "computer" is more complicated. First, all computer models are mathematical models at heart[2]. According to Maki and Thompson (2006), a **Mathematical Model** is a representation of the behavior of something using mathematical concepts, symbols, and relations, and a **Mathematical Simulation** is an imitation of the behavior of something by mathematical means.

Maki and Thompson further defined a computer model as mathematical model coded into a computer program. While this definition is technically correct, it does not capture the important distinction between computer models and mathematical models. Many people associate computer models with electronic computers; however,

[2] This is certainly true for all computer models in scientific computer simulation, but it is also true for all other computer models through the use of lambda calculus.

computers can also be human or machine. Thus, while the computer program could be source code used on an electronic computer, it could also be instructions to a room full of people.

What is important about a computer simulation is not what device performs the simulation but how the simulation has changed from what was originally intended. As Feynman (1986) pointed out in his lecture on computer heuristics, computers are essentially large filing systems that need to follow a very specific set of instructions as they cannot think for themselves. They are used because what a computer lacks in mathematical ability it more than makes up for in speed. This usually means that the mathematical model must be re-written to account for the lack in ability of the computer (whether it be electronic, man, or machine), and this re-writing is the basis for defining a computer model. A **Computer Model** is a mathematical model which has been re-written such that it can be solved by some device which has a limited mathematical capability. Likewise a **Computer Simulation** is an imitation of the behavior of something expressed as a mathematical model on a device with limited mathematical capability.

Finally, these two modifiers can be put together to fully define a scientific computer simulation. A **Scientific Computer Model** is a representation of the behavior of something in the natural universe through mathematical means on a device with limited mathematical capability. Likewise, a **Scientific Computer Simulation** is an imitation of the behavior of something in the natural universe expressed as a mathematical model on a device with limited mathematical capability.

1.1.2.1. Purposes of a Scientific Computer Simulation

Scientific computer simulations can be used for multiple reasons, but generally all of the reasons are grouped into two broad categories: advancement of scientific knowledge and advancement of technical knowledge (Oberkampf and Roy 2010). **Scientific Knowledge** is knowledge generated solely for the improved understanding of the Universe and humankind's place in it. **Technical Knowledge** is the generation of knowledge used in some way for the creation of applied knowledge or the creation of a new physical systems or processes. In other words, simulations are used to learn about the world around us or to make decisions about that world.

1.2. **What is Scientific Computer Simulation Review?**

No matter why a simulation is used (either to learn or to make a technical decision) the goal of every scientific computer simulation is the same: to adequately predict the behavior of the physical universe. A Simulation performed to reveal details about the movement of galaxies in deep space and a simulation performed to decide if the movement of cars over a bridge would cause it to fail have something in common: both must be deemed trustworthy before they are used. That is, someone must decide that the specific simulation has adequately predicted the behavior of the physical universe. While the criteria to make such a decision would be different in either case, such a decision must be reached.

In essence, before any simulation is used, a decision maker must decide that it is appropriate to use the simulation. If it is a technical decision, that decision maker may be a high level manager. However, even if it is a simulation used to advance technical knowledge, a decision maker (often the analyst performing the simulation) must decide that the specific simulation is adequate. Ultimately, someone must decide if the results of the simulation should be trusted. Because this decision is the the dividing line between using the results of a simulation or ignoring the results of a simulation, it is called the Ultimate Question. The **Ultimate Question of Scientific Computer Simulation** is: *Can the results of the specific simulation be trusted for the intended purpose?*

Answering the Ultimate Question is not only about demonstrating the results of the specific simulation can be trusted, but that the results can be trusted enough for the intended use. Thus, the Ultimate Question can be separated into two independent questions:

(1) How trustworthy are the results of the specific scientific computer simulation?
(2) How trustworthy do the results need to be for the intended purpose?

The first question (1) is fundamental to the scientific computer simulation itself and ideally could be answered without knowing the intended purpose of the simulation. The second question (2) is fundamental to the intended purpose (or requirements) of the simulation and ideally could be answered without knowing anything about the scientific computer simulation.

1.2.1. The Fundamental Question of a Scientific Computer Simulation

The **Fundamental Question of a Scientific Computer Simulation** is: *How trustworthy are the results of the specific scientific computer simulation?* Before attempting to answer this question, the observation by George Box needs to be kept in mind: "all models are wrong, but some are useful" (Box and Draper 1987). At the time Box was referring specifically to statistical models, but his statement is none the less true for all types of models and simulations. This statement is paraphrased below:

All scientific computer simulations are wrong, but some are useful.

This is not a pessimistic statement or an argument for inaction; rather it is the recognition of the fact that there will always be a difference between the behavior of the physical universe and the simulation's imitation of it. An ideal (or perfect) simulation would imitate that behavior perfectly, but such a simulation does not exist. Because "*all scientific computer simulations are wrong, but some are useful*", it is not a question of "is it wrong?" but a question of "how wrong is it?"

There are two approaches used when answering this Fundamental Question, each based on a different initial assumption. The **Passive Approach** is to initially assume that the simulation is correct unless specific evidence suggests otherwise. Using this approach, only those portions of the simulation which are thought to be questionable are reviewed. Conversely, the **Active Approach** is to initially assume that the entire simulation is incorrect unless specific evidence suggests otherwise. Using this approach, the entire simulation is questioned. This approach is generally considered the philosophy of skepticism, of Verification and Validation, and of science itself.

Answering the Fundamental Question of a Simulation can be very resource intensive, and therefore a combination of both the active and passive approaches is generally used. That is, rarely is the simulation assumed to be either entirely correct or entirely in error. Instead, certain parts of the simulation are assumed to be correct (or correct enough) unless there is evidence to say otherwise, and other parts are assumed to be in error thus prompting a need to develop evidence to the contrary. Typically, the amount of evidence obtained demonstrating that the simulation is correct (or correct enough) is entirely dependent upon the intended purpose of the simulation. That is, the amount of time and effort spent answering the Fundamental Question of a Simulation is usually decided by the answer to Fundamental Question on the Requirements.

1.2.2. The Fundamental Question on the Requirements of a Scientific Computer Simulation

The **Fundamental Question on the Requirements of a Scientific Computer Simulation** is: *How trustworthy do the results of a scientific computer simulation need to be for the intended purpose?* Even though it is independent, the answer to this question will often dictate how much effort is used in answering the Fundamental Question of a Simulation.

Oberkampf and Roy (2010) correctly observed that trusting the results of simulation for some intended purpose is similar to making a bet. Based on some body of supporting evidence, the decision maker is betting that the results of the simulation are true, and is willing to risk the consequences if the results of the simulation are not true. (For the purposes of this dissertation, a **Decision Maker** is any individual who answers the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*) in whole or in part.) Until a simulation is used in a decision making process, its only possible consequences are the loss of the resources spent on performing the simulation. It is only after the results of the simulation are used for some purpose that incorrect results may have greater consequences.

A distinction is often made between high and low consequence simulations. A **High Consequence Simulation** is a simulation which, if wrong, is likely to results in significant loss of resources including human lives. Likewise, a **Low Consequence Simulation** is a simulation which, if wrong, is likely to result in only minor loss of resources. While such definitions are common, they can be misleading as the simulation does not determine its consequences. The possible consequences are determined by the intended purpose of the simulation. Thus, the same simulation could be a high consequence simulation if used for one intended purpose and a low consequence simulation if used for another.

Therefore, it is better to consider a simulation as having either a high consequence purpose or a low consequence purpose. A **High Consequence Purpose** is when the intended purpose of the simulation is such that an incorrect simulation would likely result in significant loss of resources including human lives. Conversely, a **Low Consequence Purpose** is when the intended purpose of the simulation is such that an incorrect simulation would likely result in only minor loss of resources.

1.2.3. Scientific Computer Simulation Review: Answering the Ultimate Question

All of the topics addressed in this section are the domain of scientific computer simulation review. **Scientific Computer Simulation Review** is the process of determining how trustworthy the results of a scientific computer simulation are, how trustworthy the results need to be for some intended purpose, and based on this information, if the specific simulation should be trusted for the intended purpose. In other words, it is the act of answering the Fundamental Question of a Simulation and the Fundamental Question on the Requirements and using those answers to answer the Ultimate Question.

1.3. History of Scientific Computer Simulation Review

In some form or another, scientific computer simulation review has and continues to be performed on every scientific computer simulation. However, recently this process has become more formalized, suggesting that scientific computer simulation review could be thought of as a field of study in its own right, independent of any specific type of simulation.

1.3.1. History of Scientific Computer Simulations

The first scientific computer simulations were performed when humans started using mathematics to imitate the world around them. While it is not possible to know the actual first time this occurred, a good starting point is in the 1600's with Sir Isaac Newton. Newton's laws of physics not only described the behavior of the physical universe but did so in mathematical form. Additionally, the calculus discovered by Newton and Leibniz allowed for the development of much more complicated representations which were previously unobtainable.

The span between the 1600's to the turn of the 20th century saw many advances in both science and mathematics. Scientists discovered how the physical universe behaved and then expressed that behavior using calculus. However, scientists' ability to represent the universe exceeded their ability to imitate it. For example, the Navier-Stokes equations were published by 1850, but there was no way to solve such a complex equation without making many simplifying assumptions. While this is still somewhat true today, a breakthrough at the turn of the 20th century greatly increased humanity's ability to perform simulations.

The modern era of scientific computer simulation began in 1911 when Lewis Fry Richardson submitted a paper to the Royal Society where he performed a scientific computer simulation which calculated the stresses on a dam (Richardson 1911). Richardson used the step-by-step arithmetic method for solving difference equations which had been recently applied to partial differential equations by Runge (1895), Sheppard (1899), Huen (1900), Kutta (1901), and Ganz (1903). He applied their numerical procedure of solving ordinary and partial differential equations and to solve the equations needed to calculate stresses on a dam. Thus, it was a scientific simulation, but it was also a computer simulation as he used rooms full of human computers to perform the arithmetic[3].

This milestone marked the beginning of the modern era of scientific computer simulations as Richardson needed to do everything that is done today. He needed a

[3] One of the quickest computers was able to perform around 2000 operations a week (Richardson 1911).

mathematical equation which represented some phenomena, he needed to discretize that equation so that it could be solved by "computers", and he needed to get his "computers" to produce a solution. While modern simulations are much more complex and use many more operations, all of the elements of a modern simulation were present in this calculation by Richardson.

A second milestone in the modern era of scientific computer simulations occurred during WWII with the Manhattan Project. The effort to create an atomic bomb required an immense amount of computation, and such an amount could not be met by only using the human mind. Thus, human computers were first replaced by mechanical computers and then electric computers. This milestone greatly increased the number of operations which could be performed in a simulation.

The importance of the Manhattan Project on scientific computer simulations cannot be overstated. The war and the development of the atomic bomb were not only instrumental in creating the need for faster computers and more advanced scientific computer simulations, but as detailed by Smyth (1948), the research into the atomic bomb created the very foundation for the U.S. national laboratories and the Atomic Energy Commission. The national laboratories and the nuclear community are two of the main drivers of innovation in scientific computer simulations today.

1.3.2. Informal Scientific Computer Simulation Review

Every scientific computer simulation goes through some sort of review. Often the review is informal, in that no formal procedures or guidelines are followed. Frequently, the analyst who performs the simulation is also the analyst that performs the review. (For the purposes of this dissertation, an **Analyst** is any individual involved in the simulation or simulation review process.) For example, a student performing the simulation for homework will not submit the results of the simulation to be graded until he or she is satisfied that the results can be trusted, at least enough to get a satisfactory grade.

Analysts may not be consciously reviewing the simulation, but any analyst who performs a simulation is at least somewhat aware that he or she must answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). The review process concludes when the analyst passes along the results of the simulation to someone else, as at that point they are indorsing those results and believe they should be trusted.

Such informal processes have been and will continue to be practiced for a majority of simulation review. The informal process is heavily dependent on the skill level of the reviewer and the amount of resources available for the review. For simulations which

are used for a low consequence purpose, such a process may be appropriate. However, simulations used for a high consequence purpose often require a formal review process.

1.3.3. Formal Scientific Computer Simulation Review

Formal simulation review began when analysts started documenting the procedure used for informal reviews. By documenting their review process, analysts could finally communicate what they were looking for in a simulation and how they went about answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). One of the earliest such publicly available documents was generated by the U.S. Nuclear Regulatory Commission (NRC) in 1989 (NRC 1989).

In January of 1974, Appendix K to title 10 of the Code of Federal Regulations (CFR) Chapter 50 was promulgated. Appendix K gave specific and very conservative requirements for performing a certain scientific computer simulation that all nuclear power plants needed to perform. A decade later in 1987, the NRC decided to allow plants to perform the same simulation, but instead of using the very conservative requirements of Appendix K, they could use a best estimate approach if they captured their uncertainties. Guidance for such an approach was initially provided in Regulatory Guide 1.157 (NRC 1989).

Regulatory Guide 1.157 described the "models, correlations, data, model evaluation procedures, and methods that are acceptable to the NRC staff for meeting the requirements for a realistic or best-estimate calculation and for estimating the uncertainty in that calculation." In other words the Regulatory Guide described how to demonstrate that the results of the specific simulation were appropriate for the intended use (i.e., how to answer the Ultimate Question).

The concepts developed in Reg Guide 1.157 went into generating the Code Scaling, Applicability, and Uncertainty (CSAU) methodology (Boyack, B., *et al.* 1989). CSAU is a methodology which was developed to help quantify the uncertainty of complex computer simulations. It was one of the first documents which was independent of a specific simulation type (mostly) and focused on simulations in a broad sense. The document's impact can still be seen in the guidance the NRC uses today for simulation review as provided to its own staff in the Standard Review Plan, Section 15.0.2 (NRC 2007) and to the nuclear industry in Regulatory Guide 1.203 (NRC 2005).

Outside of the nuclear power industry, formal simulation review seemed to be a result of Computational Fluid Dynamics (CFD). CFD codes provided a platform from

which many different scenarios could be simulated. Because of the broad application, interest turned toward demonstrating that these simulations were accurate, which lead to the formalization of the disciplines of Verification and Validation (V&V). These practices were later combined with Uncertainty Quantification and are commonly known as VV&UQ.

While there are many important references in the history of VV&UQ, there are five which standout as key references. First, the two guidance documents produced by AIAA in 1998 and ASME 2009 describe how a V&V assessment should be performed. Second, the text books published by Roache (Roache 1998 and Roache 2009) provide a thorough background and history to the subject with Roache's additional insight into the different facets of the topics.

The fifth reference stands out from others for one main reason, its audience. Oberkampf and Roy's text (2010) could be considered the first textbook in VV&UQ. First, it is one of the few books which address all three topics. Second, it discusses the history, definitions, theory, and current practices of VV&UQ and attempts to bridge the gaps between the topics. Finally, and most importantly, the book was not written assuming the audience was intimately familiar with the topics. Rather, the authors wrote assuming the audience had the background of an undergraduate in engineering and built upon that knowledge base.

1.3.4. Latest Developments in Scientific Computer Simulation Review

Recently, the focus of simulation review has shifted towards assessing the maturity of scientific computer simulations and especially the development of frameworks for that purpose. The NRC has developed a partial framework for performing simulation reviews and complete frameworks have been developed by Sandia National Laboratories (SNL) and NASA.

1.4. Maturity Assessment in Scientific Computer Simulation Review

Recently, a major focus in scientific computer simulation review has been the development of frameworks which can be used to assess the maturity [4]of scientific computer simulations. A framework is built from a set of attributes where each attribute has specific criteria that correspond to a different maturity level. The process of using the framework to assess the maturity of a specific simulation involves determining what the specific simulation's level is in each attribute, (i.e., what criteria does that simulation match).

While multiple maturity assessment frameworks had been developed in the past to assess computer software, many of the early frameworks were not focused on scientific computer simulations specifically. Rather they were focused on commercial products and ensuring the software performed as the customer expected. In these situations, the customer acted as final judge on the performance of the software.

While there can be many customers for a scientific computer simulation, there is only one judge, nature. While commercial companies could have many requirements for a simulation, nature really only has one requirement: predict exactly what she would do in the same situation. While it is possible to fool companies, managers, and ourselves, into thinking that the simulation we performed met our requirements, as Feynman (1986) concluded in his personal observations on the Challenger Disaster "Nature cannot be fooled".

1.4.1. Survey of Early Maturity Assessment Frameworks

The earliest frameworks generally considered are the Capability Maturity Model Integration (CMMI) (CMMI 2010) and the Technology Readiness Levels (TRLs) (GAO 1999). Both of these frameworks started development in the 1980s and a brief summary is given here. For a more complete survey of these and other maturity assessment frameworks see Oberkampf and Roy (2010).

The CMMI framework is based in the Institute of Electrical and Electronics Engineering (IEEE) mindset. That mindset has a more general view of concepts (such as verification and validation) than is held by both the M&S and nuclear communities. CMMI is focused on helping companies develop the products and

[4] Ideally, the concept of maturity would be fully defined before maturity assessment frameworks are discussed. However, key concepts of maturity and maturity theory have not been developed and are one of the research contributions of this dissertation.

software their customers want instead of determining if software correctly predicts behavior in the physical universe (i.e., nature).

The TRL framework was intended "to determine the readiness of technologies to be incorporated into a weapon or another type of system." While some of the concepts in this framework can be carried over to scientific computer simulations, the framework itself was developed to examine the maturity of products and technologies and determine if those products were ready to be used by NASA or the Air Force.

1.4.2. Survey of Frameworks focused on Scientific Computer Simulations

The need for frameworks specifically developed for scientific computer simulation became apparent in the mid 2000's, first by Harmon and Youngblood (2003, 2005) and Pilch, *et al.* (2004). Harmon and Youngblood proposed a framework for assessing the validation process for simulations. While Harmon and Youngblood are credited with developing the first framework, it is Pilch, *et al.*'s framework that is really the starting point for scientific computer simulation review. Harmon and Youngblood's framework was specifically focused on the validation aspect while Pilch, *et al.*'s framework was focused on the all aspects of the simulation.

Pilch's framework was further developed by Oberkampf, Pilch, and Trucano (2007) at SNL and became the Predicative Capability Maturity Model (PCMM). Around the same time, Thomas Zang and others at NASA collaborated with SNL and produced their own framework in a NASA standard for modeling and simulation as a response to the 2003 Space Shuttle Columbia accident (NASA[1] 2008).

Independently from this work, the U.S. Nuclear Regulatory Commission had been developing its own guidance on assessing simulations. In 2000, the NRC released draft guidance which included a partially developed assessment framework (Standard Review Plan, Section 15.0.2, 2000).

1.4.2.1. U.S. Nuclear Regulatory Commission Guidance

In 2000, the U.S. Nuclear Regulatory Commission released draft guidance on how to review a scientific computer simulation (i.e., what were the requirements), specifically simulations which were used to predict how nuclear power plants would respond in specific accident scenarios. This guidance was developed from CSAU **(**Boyack, B., *et al.* 1989) and the experiences of the senior NRC staff that had spent their careers reviewing computer simulations. Initially, draft guidance was released

for those reviewing the simulations in the form of the Standard Review Plan, Section 15.0.2 (NRC[2] 2000) and for those performing the simulations in the form of a Regulatory Guide 1.203 (NRC[1] 2000). Both of these documents were later formally published in 2007 and 2005.

Neither of these guidance documents provides a complete assessment framework as they do not provide complete assessment criteria. This was done intentionally as the guidance was meant to be applicable to a broad range of events. However the guidance does provide the basis for criteria and a list of attributes of the simulation to be assessed. Standard Review Plan, Section 15.0.2, separates the simulation into the following seven attributes as represented in Table 1-1.

Table 1-1: NRC's Maturity Attributes

NRC Attributes
Documentation
Evaluation Model
Accident Scenario Identification Process
Code Assessment
Uncertainty Analysis
Quality Assurance Plan
COL Action Items and Certification Requirements and Restrictions

1.4.2.2. Harmon and Youngblood

Harmon and Youngblood (2003, 2005) are credited with developing the first complete framework for assessing the maturity of scientific computer simulations. The framework is considered complete because they not only provided the different attributes which should be assessed but also criteria needed to assess each attribute.

Their framework was specifically focused on assessing the maturity of the validation process for simulation models. The framework focused on the simulations performed by the Department of Defense (DoD), going so far as to adopt the DoD encompassing view of validation (allowing expert opinion and the results of other simulations to validate a simulation instead of requiring experimental data). The framework contained five maturity attributes as represented in Table 1-2.

Table 1-2: Harmon and Youngblood's Maturity Assessment Framework

Harmon and Youngblood Attributes	Levels
The Conceptual Model of the Simulation	1 - 6
Verification Results from Intermediate Development Products	1 - 6
The Validation Referent	1 - 6
The Validation Criteria	1 - 6
The Simulation Results	1 - 6

Each attribute was assessed and assigned a level based on the following criteria:

- **Level 1** – we have no idea of the maturity
- **Level 2** – it works, trust me
- **Level 3** – it represents the right entities and attributes
- **Level 4** – it does the right things, its representation are complete enough
- **Level 5** – for what it does, its representations are accurate enough
- **Level 6** – I'm confident this simulation is valid.

For this framework, the same criteria were used for each attribute.

1.4.2.3. Pilch, Trucano, Peercy, Hodges, and Froehlich

Pilch *et al.* (2004) are credited with developing the first framework for assessing the entire scientific computer simulation. The framework was created specifically for nuclear weapons design codes with the nine attributes and four levels per attribute, as represented in Table 1-3.

Table 1-3: Pilch *et al.*'s Maturity Assessment Framework

Pilch *et al*. Attributes	Levels
Request for Service	1 - 4
Project Plan Development	1 - 4
Technical Plan Development	1 - 4
Technical Plan Review	1 - 4
Application-Specific Calculation Assessment	1 - 4
Solution Verification	1 - 4
Uncertainty Quantification	1 - 4
Qualification and Acceptance Testing	1 - 4
Documentation and Archiving	1 - 4

Unlike Harmon and Youngblood, each of Pilch *et al.*'s attributes had assessment criteria specific to that attribute. Thus, the criteria corresponding to level 3 in Solution Verification was different from the criteria corresponding to level 3 in Uncertainty Quantification. The criteria were attribute specific, but a general description for the criteria can be given as follows:

- **Level 1** – appropriate for R&D tasks
- **Level 2** – appropriate for preliminary design and support
- **Level 3** – appropriate for qualification support
- **Level 4** – appropriate for qualification of system components

By using maturity levels specific to certain attributes, the framework was able to relay more information to the decision maker.

Pilch *et al.*'s framework was not only the first formal and complete maturity assessment framework for scientific computer simulations, but it was also the first attempt at defining the boundaries of scientific computer simulation review. Whether consciously aware of it or not, Pilch *et al.* were claiming that only these nine attributes were important in determining if a specific simulation (in this case M&S for nuclear weapons) can be trusted. While it is obvious that this list is not all-inclusive, the concept that there could be such an all-inclusive list, or at least a list that had captured all of the important attributes, was a new idea. This fact should not be overlooked as up to this point, most discussions of simulation review were experience based being either a list of best practices or a list of warnings to not repeat some set of errors. But by developing a maturity framework focused on scientific computer simulations in general and not one specific aspect of simulations, Pilch *et al.* made the bold suggestion that there may be some finite set of attributes and defined set criteria which could be used to determine the trustworthiness of a specific scientific computer simulation.

1.4.2.4. Predictive Capability Maturity Model

The Predictive Capability Maturity Model (PCMM) was developed at SNL by Oberkampf, Pilch, and Trucano (2007) along with collaborations with NASA. It is perhaps best to think of PCMM as a broader application of Pilch *et al.*'s original framework.[5] The framework was constructed to focus more on the computational aspects of M&S with six attributes and four levels per attribute, as represented in Table 1-4. A complete description of the framework is given in Section 5.3.1 of this dissertation.

Table 1-4: Oberkampf, Pilch, and Trucano's Maturity Assessment Framework

PCMM Attributes	Levels
Representation and Geometric Fidelity	0 - 3
Physics and Material Model Fidelity	0 - 3
Code Verification	0 - 3
Solution Verification	0 - 3
Model Validation	0 - 3
Uncertainty Quantification and Sensitivity Analysis	0 - 3

Each attribute was assigned a maturity level based criteria specific to that attribute, but roughly corresponding to the following:

- **Level 0** – little or no assessment of the completeness
- **Level 1** – some informal assessment of the completeness
- **Level 2** – some formal assessment of the completeness
- **Level 3** – formal assessment of the completeness

1.4.2.5. NASA Standard for Models and Simulation

NASA also developed a maturity framework in their standard for models and simulation in collaborations with SNL. They initially published an interim standard in 2006 (NASA 2006) and their final standard in 2008 (NASA[1] 2008). Most recently, an overview of the philosophy and requirements of their standard was published in the Journal of Aircraft (Blattnig *et al.* 2013).

The NASA standard was developed as a response to the findings of the 2003 Space Shuttle Columbia accident report. The findings from the Columbia Accident

[5] Two of the authors of Pilch *et al*'s. original framework (Pilch and Trucano) were also authors on PCMM.

Investigation Board (CIAB) which were particularly relevant to the development are given as follows (CAIB 2003):

> "F.6.3-10: The Team's assessment of possible tile damage was performed using an impact simulation that was well outside Crater's test database. The Boeing analyst was inexperienced in the use of Crater and the interpretation of its results. Engineers with extensive Thermal Protection System expertise at Huntington Beach were not actively involved in determining if Crater results were properly interpreted."
>
> "F.6.3-11: Crater initially predicted tile damage deeper than the actual tile depth, but engineers used their judgment to conclude that damage would not penetrate the densified layer of tile. Similarly, RCC damage conclusions were based primarily on judgment and experience rather than analysis."
>
> "F6-3-13: The assumption (and their uncertainties) used in the analysis were never presented or discussed in full to either the Mission Evaluation Room or the Mission Management Team."

NASA's framework has eight attributes with five levels per each attribute (each with their own criteria), as represented in Table 1-5. A complete description of the framework is given in Section 5.3.2 of this dissertation.

Table 1-5: NASA's Maturity Assessment Framework

NASA Attributes	Levels
Verification: Were the models implemented correctly, and what was the numerical error/uncertainty?	0 - 4
Validation: Does the M&S results compare favorably to the referent data, and how close is the referent to the real-world system?	0 - 4
Input Pedigree: How confident are we of the current input data?	0 - 4
Results Uncertainty: What is the uncertainty in the current M&S result?	0 - 4
Results Robustness: How thoroughly is the sensitivity of the current M&S results known?	0 - 4
Use History: Have the current M&S been successfully used before?	0 - 4
M&S Management: How well managed were the M&S process?	0 - 4
People Qualifications: How qualified were the personnel?	0 - 4

One of the most useful documents about maturity framework development was produced by NASA to support this standard. The document detailed the development and rationale behind the framework, including how it was made, what was included, what was not included, and what internal struggles existed (NASA 2009). The document was generated to capture that thought behind the standard and serves as one of the few technical documents that captures the "story" behind the work.

1.5. Research Contributions of this Dissertation

The main objective of this dissertation is to better establish scientific computer simulation review as an independent field of study. While there has been much attention given to specific aspects of simulation review (i.e., Verification, Validation, and Uncertainty Quantification), these are only some of the tasks that are important in answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). As a whole, simulation review has been performed for decades, but formalization of the topic has barely extended beyond organizational Quality Assurance Programs (QAP) and best practice guides.

In order to achieve the objective of better establishing simulation review as an independent field of study, the research conducted in this dissertation was performed on the following three key topics:

1. Maturity Theory – Chapter 2
2. Hierarchy of Scientific Computer Simulation Components – Chapter 3
3. Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations – Chapters 4, 5, and 6

1.5.1. Maturity Theory and its Application to Scientific Computer Simulation Review

Chapter 2 develops basic maturity theory. The need for such basic theory became evident after research into maturity assessment frameworks revealed that many key concepts (such as maturity itself) were not specifically defined. Conversations with experts who developed such frameworks revealed common misunderstandings of the maturity frameworks themselves, and it is the opinion of the author that many of the misunderstandings had more to do with a lack of knowledge about maturity than any failing of the frameworks. Thus, this chapter attempts to address that knowledge gap.

1.5.2. Hierarchy of Scientific Computer Simulation Components

Chapter 3 describes the formalization of a Hierarchy of Scientific Computer Simulation Components. The need for such a hierarchy became apparent as there was no formalized organizational structure commonly used to describe scientific computer simulations. Research into this area revealed there is a *de facto* organizational structure, (i.e., a manner in which many analysts thought of simulations). However, this structure was not formally defined. Therefore, the structure was formalized as a hierarchy in this chapter. An added advantage of the formalization of the hierarchy

allowed for additional insights to be gained into the different components of scientific computer simulations.

1.5.3. Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

Chapters 4, 5, and 6 describe the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations. The current maturity assessment frameworks were generated with significant effort by multiple experts in various fields of study over many years. Therefore, it did not seem likely that a single author with neither the breadth nor depth of experience of the original authors could suggest any useful changes for the current frameworks. Instead, the author of this dissertation chose to develop a Theoretical/Logical Framework.

Perhaps the largest challenge in creating a maturity assessment framework is developing a framework which is cost effective. There are almost an infinite number of attributes which could be assessed, but discerning which attributes should be assessed and which can be ignored is a very complicated problem. If the framework is too large, it would be would be too expensive and never used. If the framework is too small, it would likely ignore many important attributes. Much of the work which went into the current maturity frameworks (PCMM and NASA) went into solving this problem. Therefore, instead of focusing on the economical aspect of the framework, the author chose to develop a framework which focused on capturing the almost infinite number of attributes that could possibly be assessed. Thus, the framework would be theoretical in nature. While such a framework may or may not be useful for assessing simulations, it would be extremely useful for understanding the many different aspects of the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

Chapter 4 discusses the development of the Theoretical/Logical Framework. Chapter 5 discusses how the Theoretical/Logical Framework is populated with assessment sets. Finally, Chapter 6 provides the maturity assessments of the Theoretical/Logical Framework. Additionally, the Theoretical/Logical Framework is used to assess a simple homework problem in Appendix A.

2. Maturity and Scientific Computer Simulation Review

This chapter introduces the reader to maturity theory and how it can be used to better understand scientific computer simulation review. The first section is used to define and discuss some basic concepts in maturity theory. The second section describes scientific computer simulation review as a four stage process using those concepts. Finally, the third section concludes and summarizes the contributions of this chapter.

2.1. Introduction to Maturity Theory

The following section introduces the concept of maturity, develops its underlying theory, and provides common definitions. While the concept of maturity in computer simulations is quite prevalent, it has never actually been defined. For example, while both the Capability Maturity Model Integration (CMMI) (SEI 2010) and NASA use the concept of maturity (or as NASA calls it "credibility"), neither actually define it. NASA has even stated that they purposefully do not provide a technical definition for "credibility" (their synonymy for maturity) and state that they "use the word in the usual sense of the English language" (NASA[2] 2008). It should be noted that developing a definition for maturity is not a menial task. As this section demonstrates, a definition for the word "maturity" is only suggested after a significant amount of maturity theory is given.

Developers and practitioners of maturity assessment are often concerned about a decision maker misinterpreting the meaning of the assessment's results. While some misinterpretations are simply due to the nature of the complex problem being addressed, others are systematic. These systematic misinterpretations are often the consequence of decision makers not understanding basic maturity theory and are independent of the specific framework chosen. While the basic concept of maturity is intuitive in nature, little has been written about it, about maturity frameworks, or about maturity assessment. Therefore, this section was written to provide this missing background of basic maturity theory. This theory is applicable to any use of maturity and not specific scientific computer simulations.

2.1.1. Introduction to Maturity

Maturity is defined as "the quality or state of being mature, fully developed" ("Maturity," 2013). Aside from being circular, this definition only replaces the word "maturity" with the phrase "fully developed". This definition does capture the essence of maturity, but to have a firm grasp of what maturity is (and is not), a more detailed definition is required.

Maturity is a deceptively familiar concept. It is familiar because most people use words such as "mature" and "immature" on an almost daily basis and are able to effectively communicate with each other. It is deceptive because the concept is much more complex than the binary descriptors of "mature" and "immature". Maturity has specific components, attributes, and rules which must be obeyed if the concept is to be used correctly and precisely. Interestingly, like the rule of grammar, most people are able to follow its rules without even realizing it.

2.1.2. Components of Maturity

The following is an example of a statement using the concept of maturity: "Josh is emotionally mature". This generic statement reveals one key aspect about maturity: it is a descriptive summary of an object. In other words, when the concept of maturity is connected with some object, that object is being both summarized and described in some manner. In that sense, maturity is similar to a measurement.

The following is an example of a statement using the concept of measurement: "The length of screw is 1.2 inches." As maturity is a descriptive summary of an object, a measurement (1.2 inches) is a descriptive summary (length) of an object (the screw). This similarity suggests that one way to better understand maturity is to better understand a measurement.

Formally defined, a **Measurement** is the assignment of a number to an object in a systematic way as a means of representing properties of the object (Allen and Yen 1979). Using this definition, a measurement can be separated into four distinct components:

(1) The object (e.g., the screw)
(2) The property of the object (e.g., the length of the screw)
(3) The systematic process used to generate the measurement (e.g., using a ruler)
(4) The measurement itself (e.g., 1.2 inches)

These four components of a measurement can be used to generate the following four corresponding components of maturity:

2.1.2.1. Maturity Object

As a measurement is a descriptive summary of some object, a maturity is a descriptive summary of the maturity object. The **Maturity Object** is the object whose maturity is being assessed. It is important to remember that any statements about maturity are not statements in a vacuum; they must be made in reference to the maturity object.

In the sentence "Josh is emotionally mature", Josh is the maturity object.

2.1.2.2. Maturity Attribute

A measurement isn't a summary of the entire object, but only of one property of that object. Likewise a maturity isn't a summary of the entire maturity object, but of one attribute of that object, the maturity attribute. The **Maturity Attribute** is the specific attribute of the maturity object that is being assessed. It is important to remember that any statements about maturity are not statements about an entire object but about a specific attribute of that object. The maturity attribute can be referred to the maturity category or maturity element.

In the sentence "Josh is emotionally mature", emotional maturity is the maturity attribute.

2.1.2.3. Maturity Assessment Set

As a measurement requires a systematic process to assign a number to a specific property of an object, maturity assessment requires the maturity assessment set to assign a maturity level to a specific attribute of the maturity object. The **Maturity Assessment Set** is the set of all of the maturity levels (or criteria) of a specific attribute of an object. The maturity assessment set is used to determine what maturity level the specific attribute has obtained. Typically, the maturity levels are arranged in order of increasing maturity. The maturity assessment set can be referred to as the assessment set, maturity criteria, or criteria.

In the sentence "Josh is emotionally mature", an example of the criteria that could be used is represented in Table 2-1.

Table 2-1: Maturity Assessment Set for Emotional Maturity

Maturity Object	Person
Maturity Attribute	Emotional
Maturity Levels	
Level	**Criteria**
Immature	The person is not humble.
Mature	The person is humble.

2.1.2.4. Assessed Maturity Level

As a measurement is the result of a systematic process, a maturity is the result of the assessing an attribute against some criteria and is referred to as the assessed maturity level. The **Assessed Maturity Level** is the maturity level a specific attribute has obtained in a given maturity assessment set. Given a specific attribute and a specific assessment set, and information about the object, the assessed level is the maturity level that attribute of the object has achieved.

In the sentence "Josh is emotionally mature", the assessed maturity level is "mature". That is, because Josh is humble he is assessed to be emotionally mature. Using the criteria of Table 2-1, the other possible level would have been "immature".

2.1.2.5. Summary of the Components of Maturity

Table 2-2 provides a comparison between the components of a measurement and of maturity.

Table 2-2: Comparing the Components of a Measurement to a Maturity

Component	Measurement	Maturity
(1)	Object	Maturity Object
(2)	Property of the Object	Maturity Attribute
(3)	Systematic Process used to generate the Measurement	Maturity Assessment Set
(4)	Measurement	Assessed Maturity Level

These components can be used to better understand some basic concepts of maturity assessment.

2.1.3. Basic Aspects of Maturity Assessment

Maturity Assessment is the act of using a specified maturity assessment set to determine the achieved maturity level for a specific maturity attribute of the given maturity object. Maturity assessment can either refer to the assessment of a single attribute using one assessment set or the assessment of multiple attributes of the same object using different assessments sets for each attribute. While the phrase "maturity assessment" may be unfamiliar, the concept of maturity assessment is very familiar. Almost all decision making processes from deciding what to eat to deciding what car to purchase can be described in terms of maturity assessment. One example of this is provided in Example 2.1 below. This sub-section discusses some basic concepts of maturity assessment which may not be readily apparent at a first glance.

Example 2.1: Maturity Assessment in Buying a Car

Suppose you plan on buying a new car. You start the process by thinking about the qualities of the car which are most important to you. You decide the car needs to be cheap, look cool, and be safe. With this in mind, you go see what is for sale.

After traveling to many dealerships you have narrowed your choices down to three cars: A, B, and C. Then you go back home and weigh the pros and cons of each car. Car A looked cool but was expensive and not as safe as Car B. Car C was red (and you hate the color red), but it was pretty safe and was the cheapest. This information can be thought of in terms of maturity.

Maturity	Price	Style	Safety
Low	A	C	A
Middle	B	B	C
High	C	A	B

That is, in each of the attributes (price, style, and safety), each car can be compared to the others. For example, for *price* car A had the lowest maturity because it was the most expensive and car C had the greatest maturity because it was the least expensive. Thus, when you make your decision, you are comparing the different maturity levels for the three objects (cars) under consideration. Unfortunately, there is one final challenge that remains, what decision do you make? Maturity cannot make the decision for you and you (the decision maker) must decide which attributes are most important.

2.1.3.1. Maturity is about an Attribute not an Object

Any meaningful discussion of maturity is about the maturity of an attribute and not the maturity of the object. For example, saying "Josh is emotionally mature" is a statement which carries a specific meaning about the attribute of emotional maturity. On the other hand, a statement such as "Josh is mature" without further clarification or context, conveys little to no meaning. Missing from this statement is in what way Josh is mature. Does the person making this statement mean that Josh has reached a certain age, that Josh has developed a specific aspect of his personality, or something else entirely?

The above statement is similar to the following statement made using the concept of a measurement instead of maturity: "The screw is 1.2." This statement corresponds to "Josh is mature". In both statements the object is identified as is the final value of the assessment/measurement. However, what each statements lacks is the attribute or the property of the object under consideration. In other words, which property of the screw is 1.2? The length? The weight? The diameter? Not specifying the attribute in a maturity statement is the same as not specifying the property in a measurement statement.

This fact can be confusing as some objects are very strongly associated with one specific attribute. Thus, most considerations of that object are really considerations of that one attribute. A common example of this is when someone may say "The wine is mature", they most likely mean "The taste (flavor, aroma, texture) of the wine is mature". This is a case where the taste attribute of wine is so closely associated with wine itself, that a consideration of wine is almost exclusively a consideration of its taste and not of any other attribute (e.g., density, viscosity, temperature, etc.).

2.1.3.1. Maturity Assessments can be Informal or Formal

No one would think to use a measurement that is informal or undefined, but informal maturity assessments are commonplace. An **Informal Maturity Assessment** is an assessment where the assessment criteria are not formally defined or captured in any fashion. Almost all uses of the word "mature" (or any derivation thereof) involve an informal assessment. For example, suppose a person comments that "Josh is emotionally mature." That person is most likely making that determination based on some criteria known only to them (which may not even be consciously known) and not based on some assessment criteria written down anywhere.

An informal assessment is highly subjective as another person would likely use different criteria for same attribute and may result in a different assessed maturity level. In other words, person A would say "Josh is emotionally mature" while person B would say "Josh is emotionally immature". This difference in the assessed maturity level is due to the different criteria each person is using. Even if the same person is assessing the same attribute, they may apply different criteria for each assessment. Thus, at one point person A would say "Josh is emotionally mature" and at another point they may say "Josh is emotionally immature". This could be the result of a change in the maturity object, but it could also be the result of the same person applying a different set of assessment criteria. To ensure that the same assessment criteria are applied, the criteria should be formalized.

A **Formal Maturity Assessment** is an assessment whose assessment criteria are formally defined and captured in some fashion. For example, a maturity assessment which uses the criteria described in Table 2-1 would be considered a formal assessment becuase those criteria have been captured. Generally, formal maturity assessments appear as a maturity table. A **Maturity Table** is a table which contains the assessment set and may contain any number of other important features such as the maturity object, the maturity attribute, and tracking information. Table 2-1 is an example of a maturity table. The tables in Example 2.1 are not maturity tables as they do not provide the criteria which were used to assess the maturity.

2.1.3.2. The Maturity Assessment Set should contain all possible Maturity Criteria

Consider the measurement of the length of a screw. The length of any screw must be an element from the set of real, positive numbers, as represented in Eq. 2.1.

$$Length \, \epsilon \, \{\Re +\} \qquad 2.1$$

Likewise, the assessed maturity level must also be an element from some set, the set of all possible maturity criteria. The **Set of All Possible Maturity Criteria** (or set of all possible maturity levels) is the set of all of the maturity criteria which correspond to any possible level of the specific attribute of the given object. This set could be finite or infinite.

The maturity assessment set should contain all necessary criteria such that any appropriate object could be assessed. For example, the criteria described in Table 2-1 should be able to be used to assess any person's emotional maturity. If there is one person whose emotional maturity doesn't correspond to one of the criterion (or levels), then the criteria are incomplete.

2.1.3.3. The number of Maturity Levels in a Maturity Assessment Set can vary

The number of maturity levels in any given maturity assessment set can vary. When assessing maturity, many people may think of two levels (mature and immature), but many more levels are possible. Furthermore, two sets of criteria can be created for the same attribute, where each set has a different number of levels which may be appropriate under different circumstances. Often, choosing the number of levels is a decision made when an assessment set is created.

When a maturity assessment set is created, choosing the number of assessment criteria (or maturity levels) is typically accomplished by one of two methods. Either a set number of levels are arbitrarily chosen or some natural demarcation in the attribute is used.

The advantage of choosing the number of levels beforehand is that it can be ensured that the number of levels in the assessment set will not be too many or too few for useful communication. A popular choice for the number of levels is based on 'the magic number of seven, plus or minus two' (Miller 1956). However, forcing a certain number of levels can result in subjective assessment criteria where a division

between two adjacent levels is somewhat arbitrary. On the other hand, using a natural demarcation in the attribute usually results in very objective criteria with logical distinctions between two adjacent maturity levels but can easily result in too many or too few levels which can be confusing. These pluses and minuses are summarized in Table 2-3.

Table 2-3: Pluses and Minuses in Determining Maturity Levels

How are the number of levels determined?	Advantages	Disadvantages
Pre-Selected	Will never result in too many or too few levels	Level distinctions will likely be somewhat arbitrary
Natural Demarcation of Attribute	Levels will be distinct	May results in too many or too few levels to be useful

For example, suppose a bank decided to develop a maturity assessment set for someone's credit score. As the credit score is a whole number from 350 to 850, one simple way to create the criteria is to allow the attribute itself to determine the number of levels. Thus, each level corresponds to a distinct value of the credit score. This maturity table is given in Table 2-4.

Table 2-4: Maturity Table for Credit Score with Number of Levels "Chosen" by the Attribute

Maturity Object	Person
Maturity Attribute	Financial – Credit Score
Maturity Levels	
Level	**Criteria**
1	Credit Score equals 350.
2	Credit Score equals 351.
⋮	
501	Credit Score equals 850.

The criteria are very objective with a logical distinction between each of them, but there are over 500 levels. While the levels are distinct, there are too many of them to be useful to a decision maker. Suppose the bank decides to develop another set of

assessment criteria, but this time forcing the number of levels to be 7. This new maturity table is given in Table 2-5.

Table 2-5: Maturity Table for Credit Score with Pre-Chosen Number of Levels

Maturity Object	Person
Maturity Attribute	Financial – Credit Score
Maturity Levels	
Level	**Criteria**
1	Credit Score less than 500.
2	Credit Score between 500 - 579.
3	Credit Score between 580 - 619.
4	Credit Score between 620 - 659.
5	Credit Score between 660 - 699.
6	Credit Score between 700 - 759.
7	Credit Score between 760 - 850.

With only seven levels, the results of this assessment could be much more easily used. However, the distinction between adjacent levels is somewhat obscured and arbitrary. For example, the person with a credit score of 660 has the same maturity as someone with a credit score 39 points higher (level 5), but someone with a credit score of only 1 point lower than 600 has a lower maturity (level 4). Even with the somewhat arbitrary groupings, the method of pre-selecting the number of levels in creating the maturity assessment set and performing the maturity assessment is usually preferred.

2.1.3.4. Maturity Levels are an Ordered Set

One of the most common misunderstandings about maturity assessment is the fact that maturity levels are an ordered set and do not have an associated scale. While each maturity level in a particular assessment can be assigned a number, (1,2,3,…), it

could also be assigned a letter (A,B,C,...), a word (most mature, very mature, somewhat mature, …) or any other set of symbols (♐, ♒, ❖, …). It is often convenient to represent maturity levels as numbers, but this can easily lead to misunderstanding or misuse because numbers have an associated scale and maturity levels do not.

Saying a number has an associated scale means that someone could say that 2 is twice as big as 1 and 6 is half as big as 12 and so on. However, maturity levels do not have a scale. That is, an object with a maturity level of 4 is not twice as mature as another object with a level 2. The object with a maturity level of 4 is more mature than the object with a maturity level of 2, but exactly how much more mature cannot be known.

Maturity levels are an ordered (or ranked) set. The numbers 1,2,3,4 and 5 are also an ordered set, but an ordered set with an associated scale. Examples of ordered sets without associated scales are the alphabet, the "4 star" movie rating system, and the set of words {immature, somewhat immature, somewhat mature, mature}. An ordered set is a set that is placed in order from least to greatest (or vice versa). To create the ordered set of maturity levels, the criteria of a given attribute are placed in the order of lowest to highest maturity (i.e., they are ranked).

It is easy for someone to struggle with the concept that the numbers associated with maturity levels do not have an assocaited scale as almost every number in someone's life is associated with a scale. Ten dollars is twice as much as five dollars, three gallons of gas is half as much as six gallons, four miles is twice as long as two miles, etc. Ordered sets without associated scales are rare. When someone does encounter them, the ordered sets are generally not sets of numbers but of something else (e.g., stars used to rate a restaurant, thumbs used to rate a movie). Decision makers need to keep this in mind lest they make a decision under false pretenses, as Example 2.2 demonstrates.

Example 2.2: How Maturity Levels can be Misleading

The fact that maturity levels are an ordered set without a scale is particularly important to decision makers. If the decision maker is unaware of the limitations of ordered sets, he or she can be easily misled and may make a decision based on false pretenses, which could end in disaster.

Suppose that you are a bank manager and there are two clients who have come to you for a loan. Unfortunately you can only give one of them a loan, so you ask a clerk to create some maturity assessment sets and determine the financial maturity of each client. The clerk makes up one assessment set which focuses on the attribute "Amount Saved Per Year" and another which focuses on the attribute "Credit Score". After the clerk performs the maturity assessment, he hands you the following results (remember, the higher the number the higher the maturity):

	Client A	**Client B**
Amount Saved	5	7
Credit Score	501	411

Based solely on this table, would you give the loan to A or B?

You may be thinking that the loan should go to A. B may have a higher "Amount Saved" maturity, but it is only by two levels higher. Whereas A's "Credit Score" maturity is 90 levels higher. This thought process is common because we are used to thinking in terms of a scale, but it is incorrect for maturity levels as they have no scale.

Consider the "Credit Score" assessment represented by Table 2-4 (the maturity table with 501 levels). Also consider the "Credit Score" assessment represented by Table 2-5 (the maturity table with 7 levels). While A and B have maturities 90 levels apart using Table 2-4, they have the same maturity level using Table 2-5. Thus, the 90 level difference between A and B is actually almost no difference at all; each client has outstanding credit.

Now consider the "Amount Saved" maturity table given in Table 2-6 in the following section. While level 5 and 7 seem close, the criteria tell another story. Someone with a maturity level of 5 is only saving $30 a year. One the other hand, someone with a maturity level of 7 is saving more than $10,000 a year. Thus level 7 represents a significant increase in maturity compared to level 5.

With this information in hand, who would you now give a loan to? Client A who has excellent credit but does not save, or client B who also has excellent credit and saves a large amount of income? This example demonstrates how easy it is to be misled by assuming maturity levels have a scale.

2.1.4. Defining Maturity

With the concept of maturity introduced, the components of maturity defined and basic concepts of maturity assessment given, maturity itself can be defined. The suggested definition for maturity is as follows:

> **Maturity** is a measurement of rank of an attribute in the spectrum of the very worst to the very best achievable values of that attribute.

Maturity implies some value system, thus the attribute is one in which can be ranked in a value system. The very worst would be the least desirable state and the very best would be the most desirable state. Thus, it is best to think of the maturity as attribute of an object where that object's attribute exists in the spectrum of all possible values of that attribute.

2.1.5. Maturity Assessment Sets and Frameworks

Maturity assessment sets are used to determine the maturity of a single attribute. These assessment sets can be grouped into larger sets called sets of assessment sets. A **Set of Maturity Assessment Sets** is a set of one or more maturity assessment sets, where each set is used to assess a different attribute of the same object. For example, a set of assessment sets could be generated from the maturity tables of "Emotional Maturity" and of "Credit Score" described by Table 2-1 and Table 2-5. Even though those two assessment sets have nothing in common, they could both be used to assess the maturity of the same object (a person), and therefore can be combined into a set of maturity assessment sets.

Often, the set of maturity assessment sets will have a common focus. For example the set generated from the maturity tables "Credit Score" and "Amount Saved" (given in Table 2-5 and Table 2-6) has a common focus, a person's finances, and therefore could be considered a maturity assessment framework. A **Maturity Assessment Framework** is a set of maturity assessment sets where each assessment set shares a common focus with all of the other assessments sets in the group. The **Focus of the Framework** is the aspect of the maturity object which each maturity assessment set is focused upon. For example, the focus of the framework given by the maturity assessment sets of "Credit Score" and "Amount Saved" is "finances".

Table 2-6: Maturity Table for "Amount Saved Per Year"

Maturity Object	Person
Maturity Attribute	Financial – Amount Saved
Maturity Levels	
Level	**Criteria**
1	The person in not saving any money.
2	The person is saving at least $5 a year.
3	The person is saving at least $10 a year.
4	The person is saving at least $20 a year.
5	The person is saving at least $30 a year.
6	The person is saving at more than $30 but less than $10,000 a year.
7	The person is saving at more than $10,000 a year.

2.1.6. The Attributes of Maturity Criteria

These next three sub-sections turn the concept of maturity assessment in on itself by examining the different attributes of an individual maturity criterion, of maturity assessment sets, and of maturity assessment frameworks. This sub-section focuses on the two attributes of an individual maturity criterion: Distinct and Exact.

2.1.6.1. Distinct

A **Distinct Criterion** is a criterion which is distinct from all other criteria in a given assessment set. Such criteria do not overlap, but have clear lines of distinction. For example, if an object corresponds to multiple criteria in the same assessment, then those criteria are not distinct from each other. This can be somewhat confusing as criteria can be written such that they are cumulative. **Cumulative Criteria** are criteria written in such a way that an object which meets the criteria of any specific level of maturity will also meet the criteria of every lower level of maturity. The

opposite of cumulative criteria are non-cumulative criteria. **Non-cumulative Criteria** are criteria which are written in such a way that an object which meets the criteria of any specific level will not meet the criteria of any level that is lower in maturity. Cumulative criteria can still be distinct as long as there are clear lines of separation.

2.1.6.2. Exact

An **Exact Criterion** is a criterion defined exactly with no need for interpretation. Ambiguities in the criterion will negatively impact the assessment as inexact criteria can introduce significant subjectivity into the assessment process.

2.1.7. The Attributes of Maturity Assessment Sets

This sub-section focuses on the six attributes of maturity assessment sets: Complete, Detailed, Economical, Objective, Well-Defined, and Well-Spaced. These attributes are the result of grouping individual criterion into sets and their interactions with each other. Additionally, the assessment set can take on the attributes of the criterion which make up the assessment set:

- a **Distinct Assessment Set** is an assessment set made up of distinct criteria, and
- an **Exact Assessment Set** is an assessment set made up of exact criteria.

2.1.7.1. Complete

A **Complete Assessment Set** is an assessment set whose criteria correspond to all possible maturity objects. If there is an object which does not correspond to at least one of the criteria in the assessment set, then that assessment set is not complete. In other words, the assessment set should contain the set of all possible maturity criteria.

2.1.7.2. Detailed

A **Detailed Assessment Set** is an assessment set which has a sufficient number of distinct criteria such that important distinctions in maturity of the given attribute can be realized. This is not a consideration of the number maturity levels but the amount of useful detail. For example, the attribute assessment represented by Table 2-4 has

many more levels than that of Table 2-5, but it is not necessarily more detailed as the additional information does not seem useful.

2.1.7.3. Economical

An **Economical Assessment Set** is an assessment set which can be performed at a relatively low cost. The cost of performing an assessment will often be the main factor in deciding if the assessment will be performed. The cost is driven by the time, money and resources needed to determine which of the criteria in the maturity assessment set corresponds to the specific attribute of the given object.

2.1.7.4. Focused

A **Focused Assessment Set** is an assessment set where each of the criteria in the assessment is directly related to the maturity attribute. This attribute is as much about the maturity attribute as it is the assessment criteria; as a clearly defined and focused attribute will result in clearly defined and focused criteria. On the other hand, a maturity attribute which is not focused will result in an criteria which are unrelated to the attribute.

2.1.7.5. Objective

An **Objective Assessment Set** is an assessment set where two different individuals given the same set and the same information about an object would determine the same maturity level. Assessments are performed by individuals who may have varying backgrounds, expertise, and opinions. The results of a perfectly objective assessment would not be influenced by the individual performing the assessment. Conversely, the results of a perfectly non-objective assessment would be entirely based on the individual performing the assessment. In general, the objectivity of an assessment is determined by its criteria being distinct and being exact.

The objectivity of an assessment set can also be thought of in terms of qualitative (less objective) or quantitative (more objective). Assessment sets which are completely objective are those which are completely quantitative, but these sets may not always provide the most useful information to the decision maker.

2.1.7.6. Well-Defined

A **Well-Defined Assessment Set** is an assessment set where the criteria completely capture all that is intended by the maturity attribute being assessed. Making a well-defined assessment usually involves better defining the attribute being assessed instead of adjusting the criteria. For example, any assessment of "Emotional Maturity" using the criteria given in Table 2-1 would not be a well-defined assessment. There are many other aspects to emotional maturity not captured in these criteria. Attempting to fix this problem by creating better criteria would likely not help as there are many aspects to emotional maturity which could not be captured by a single maturity assessment set. Instead, the attribute of "Emotional Maturity" should be made the focus of a framework and the attribute of the given criteria should be changed to "Is the individual humble?"

2.1.7.7. Well-Spaced

A **Well-Spaced Assessment Set** is an assessment set where the increase in maturity from one maturity level to the next is approximately the same. As stated before, maturity levels are an ordered set and do not have a scale. However, those creating the criteria should strive to make the increases in maturity between levels approximately the same. For example, consider the levels of the attribute assessment represented by "Amount Saved" given in Table 2-6. It is obvious that the increase in maturity from level 3 to 4 is small compared to the increase in maturity from level 6 to 7, thus the assessment set "Amount Saved" is not well-spaced.

2.1.8. The Attributes of Maturity Assessment Frameworks

This sub-section focuses on the two attributes of maturity assessment frameworks: Independent and Thorough. These attributes are the result of grouping individual assessments sets into sets and their interactions with each other. A framework can also take on the attributes of the criteria which make up the assessment sets of the framework:

- a **Distinct Framework** is a framework made up of distinct assessment sets, and
- an **Exact Framework** is framework made up of exact assessment sets.

Likewise, the framework can take on the attributes of the assessment sets that make up the framework:

- a **Complete Framework** is a framework made up of complete assessments,
- a **Detailed Framework** is a framework made up of detailed assessments,
- an **Economical Framework** is a framework made up of economical assessments,
- a **Focused Framework** is a framework made up of focused assessments,
- an **Objective Framework** is a framework made up of objective assessments,
- a **Well-Defined Framework** is a framework made up of well-defined assessments, and
- a **Well-Spaced Framework** is a framework made up of well-spaced assessments.

2.1.8.1. Independent

An **Independent Framework** is a framework where the assessment sets are independent from each other. Thus, the resulting level of one assessment would be completely independent from the resulting level of another assessment.

2.1.8.2. Thorough

A **Thorough Framework** is a framework whose focus has been completely defined by the attributes of the individual assessment set. In all but the simplest of cases, a framework is never completely thorough as there are usually an infinite number of attributes which could be used to describe the focus. For example, the 'financial' framework described above by combining the assessments sets of "Amount Saved" and "Credit Score" is hardly a thorough framework as there are many more attributes which fall under the 'financial' focus.

2.1.9. Common Maturity Frameworks

This sub-section provides examples of common maturity assessment frameworks. While the phrase "maturity assessment framework" is relatively new, the concept is very old. Frameworks are used quite often to aid in decision making even though the decision maker may not realize it, such as the framework used in purchasing a car given in Example 2.1.

2.1.9.1. Common Informal Maturity Assessment Frameworks

The following are examples of common informal maturity assessment frameworks. Any one of these frameworks could be made formal by writing down the specific attributes and assessment criteria.

1. The four star system used to rate movies
2. The two thumbs up system use to rate movies
3. The three star system use to rate restaurants
4. The rating system for cooking competitions[6]
5. The five star system for Amazon
6. The five star system for Netflix

Notice that in each of these cases, the resulting rating conforms to an ordered set. A movie rated 3 ½ stars is not 87.5% as good as a 4 star movie, it simply means that it is better than a 3 star, but not as good as a 4 star. Because these are informal frameworks, the attributes assessed and the assessment criteria are left entirely up to the analyst performing the assessment (i.e., the critic).

2.1.9.2. Formal Maturity Frameworks

Formal maturity assessment frameworks are not as common as informal frameworks, but there are still many used every day. The creation of a formal assessment framework generally takes a great deal of resources as there needs to be some way to

[6] Often, cooking contests are not completely informal as some of the attributes are formalized. For example, Iron Chef uses the attributes of taste, presentation, and originality. Judges are instructed to rate taste on a scale of 0 – 10, presentation from 0 – 5, and originality from 0 – 5. However the criteria for each level are still left up to the judge. These scores are then combined to determine the winner.

ensure that the focus of the framework is adequately described by the attributes and criteria in the framework. The following three formal assessments sets will be discussed below.

1. The "4 Cs" used to assess the value of diamonds
2. The pain scale used to assess the pain of a doctor's patient.
3. The APGAR score used to assess the health of newborn babies.

The 4 C's

The "4 C's" is an assessment framework used by the Gemological Institute of America to determine the maturity of a diamond ("4 Cs Guide", 2013). The "4 Cs" are the following attributes of a diamond:

- Color
- Clarity
- Cut
- Carat

The focus of the framework is to determine the monetary value of the diamond. The maturity tables for this framework are given in Table 2-7, Table 2-8, Table 2-9, and Table 2-10.[7]

[7] As written, the attribute assessments for color, clarity, and cut have decreasing maturity. That is, the higher the maturity level the less mature the attribute and the lower the cost of the diamond. On the other hand, the attribute assessment for carat weight has increasing maturity. That is, the higher the maturity level the more mature the attribute and the higher the cost of the diamond.

Table 2-7: Maturity Table for a Diamond's Color

Maturity Object	Diamond
Maturity Attribute	Color
Maturity Levels	
1	D – Determined by expert opinion and color chart
2	E – Determined by expert opinion and color chart
⋮	⋮
22	Y – Determined by expert opinion and color chart
23	Z – Determined by expert opinion and color chart

Table 2-8: Maturity Table for a Diamond's Clarity

Maturity Object	Diamond
Maturity Attribute	Clarity
Maturity Levels	
1	Internally Flawless
2	An expert can only detect flaws using a 10x loupe magnifying glass when viewing the bottom of the diamond.
3	An expert can only detect flaws using a 10x loupe magnifying glass when viewing the top of the diamond.
⋮	⋮
9	Inclusions are easily seen.
10	Inclusions are large and easily seen and may impact the brilliance or structural integrity of the diamond.

Table 2-9: Maturity Table for a Diamond's Cut

Maturity Object	Diamond
Maturity Attribute	Cut
Maturity Levels	
1	Excellent – Determined by many other attributes
2	Very good – Determined by many other attributes
3	Good – Determined by many other attributes
4	Fair – Determined by many other attributes
5	Poor – Determined by many other attributes

Table 2-10: Maturity Table for a Diamond's Carat Weight

Maturity Object	Diamond
Maturity Attribute	Carat Weight
Maturity Levels	
1	0.01 Carats
2	0.02 Carats
⋮	⋮
N	Highest achievable diamond weight (around 3200 carats).

The Pain Scale[8]

[8] While the pain scale is only one maturity assessment, it still fits the criteria of a framework. It is a set of one element.

Many doctors have a scale which is used by patients to help communicate the pain they are currently feeling. The scale usually consists of 11 levels (0 – 10) with 0 being "no pain" and 10 being "worst pain ever". This is an interesting scale in that while the levels are provided, little criteria is actually given, allowing the patient to determine the criteria for themselves. Table 2-11 is an example of the pain scale.

Table 2-11: Maturity Table for Pain Sclae

Maturity Object	Person
Maturity Attribute	Pain
Maturity Levels	
0	No Pain
1	Some Pain
⋮	⋮
10	Worst Pain Ever

The Apgar Test

The final example of a formal framework is the Apgar test (Apgar 1953). The Apgar test is used to determine the health of a newborn baby. The baby is assessed immediately after birth and at certain intervals after that (e.g., a 10 minute score or 20 minute score). The Apgar test is a maturity framework with five attributes and three levels in each attribute as represent in Table 2-12.

Table 2-12: The Apgar Maturity Assessment Framework

Maturity Object	Baby				
Maturity Attribute	Complexion	Pulse Rate	Reflex Irritability	Muscle Tone	Breathing
Maturity Levels					
0	**Blue or pale all over**	**Absent**	**No response to stimulation**	**None**	**Absent**
1	**Blue at extremities, body pink**	**< 100**	**Grimace/feeble cry when stimulated**	**Some flexion**	**Weak, irregular, grasping**
2	**No cyanosis, body and extremities pink**	**≥ 100**	**Cry or pull away when stimulated**	**Flexed arms and legs that resist extension**	**Strong, lusty cry**

The resulting maturity levels from each attribute are combined (added) to create a combine maturity level which gives an overall indication of the health of the baby. The combined maturity assessment set represented by this combined maturity level is given in Table 2-13.

Table 2-13: The Combined Maturity assessment set for the APGAR Score

Maturity Object	Baby
Maturity Attribute	APGAR Score
Maturity Levels	
1	Score of 3 or below.
2	Score of 4 – 6.
3	Score of 7 or above.

Babies with a maturity level of 1 are critically low, level 2 are fairly low and level 3 are normal. This combined level can help decision makers such as doctors and nurses determine if certain actions need to be immediately taken.

2.1.10. Warnings about Combining Maturity Levels

The phrase "comparing apples and oranges" is often used to describe a situation where two objects are not comparable, but are compared anyway. Unfortunately, "comparing apples and oranges" is a necessity in decision making. Decision makers must often decide if cost trumps performance, reliability trumps timeliness, quality trumps quantity, etc. For example, sometimes a decision maker may choose to increase cost to gain extra performance or alternately may decide to reduce the cost even though it means a reduction in performance. This same challenge of making trade-offs can be seen when decision makers are using the results of maturity assessment framework to make a decision.

The results of a maturity assessment framework are the assessed maturity levels of each attribute of the object. While this information is certainly helpful to the decision maker, it does not make the decision on its own. Therefore, it is a common practice to combine the maturity levels from each of the attributes (i.e., assessment sets) in the framework into one overall number which could be used to make a decision. Such a process is a maturity combination process.

A **Maturity Combination Process** is a process used to combine maturity levels from one or more attributes of the same object to generate one maturity level. Generally, this process is a weighting function of some sort, but could really be any process which used the maturity level of each attribute as an input and resulted in a single combined maturity level. A **Combined Maturity Level** is the maturity level which is the result of the maturity combination process. There is one important aspect of this combined maturity level, it is not assessed. That is, the level is the result of some numerical averaging procedure and not the result of applying specific assessment criteria.

However, the combined level is still a maturity level, and therefore it must also be the level of some maturity attribute and correspond to some maturity assessment criteria. The **Combined Maturity Attribute** is the attribute of the object which is being assessed in the combined process. The **Combined Maturity Assessment Set** is the assessment criteria which describes all maturity levels of that combined maturity attribute. Combining maturity levels is dangerous because neither the combined maturity attribute nor the combined assessment criteria are known.

The combined level is a maturity level, but a maturity level in a vacuum. Maturity levels only have meaning because they are an expression of where the attribute of a given object exists in the spectrum of all possible values of that attribute. However, a combined maturity level is merely a number. The combined maturity attribute is not known and neither are all the possible values of that maturity attribute. Thus, the normally clear meaning of maturity level is completely obscured when such levels are combined.

Combing maturity levels is often a necessary evil. However, decision makers need to understand the pitfalls or else they may misinterpret the data which may lead to disaster.

2.2. The Four Stages of Scientific Computer Simulation Review

The review of any scientific computer simulation requires answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). As discussed earlier, this can be separated into the following two Fundamental Questions:

- the Fundamental Question of a Simulation (*How trustworthy are the results of the specific scientific computer simulation?*), and
- the Fundamental Question on the requirements (*How trustworthy do the results of a scientific computer simulation need to be for the intended purpose?*).

While this separation provides insight into the questions which need to be answered, it does not provide a method for achieving those answers. Fortunately, the concept of maturity does provide such a method.

Using the concept of maturity, scientific computer simulation review (i.e., answering the Ultimate Question) can be explained as having four separate stages. The **Four Stages of Scientific Computer Simulation Review** are given as follows:

1. Maturity Framework Development – Develop a maturity assessment framework which captures all appropriate attributes and criteria.
2. Maturity Requirements Determination – Use the developed framework to determine the required maturity levels of each attribute for the particular use of the simulation.
3. Maturity Level Assessment – Use the developed framework to determine the achieved maturity level of each attribute for the specific simulation.
4. Maturity Judgment Stage – Compare the Maturity Assessments from stage 3 with Maturity Requirements from stage 2 to decide if the simulation is adequate for the given use.

While the above terms are new, the concepts are old, and these same four stages are performed (more or less) in practically all computer simulation reviews. However, here they are called out and formally defined to better compartmentalize the simulation review process.

2.2.1. Maturity Framework Development

The first stage of review is the maturity framework development. **Maturity Framework Development** is the process of generating the maturity assessment framework which will be used for the review of the simulation. This step generally involves either choosing or generating the maturity attributes and the maturity assessment criteria. For simulation review, the real focus of the framework is the

Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*), but each maturity assessment set can be focused on different aspects the simulation.

One major advantage of making the framework formal is that such a framework can be reviewed and modified by others. Simulation reviews performed with informal frameworks are practically "new" every time and some things can be missed and other things which are not very important can be stressed. The advantage of using a formal framework is the increased consistency and the ability to evolve the framework as more knowledge is gained.

This stage of simulation review is very important as it is the foundation of the other stages. The same framework is used to determine the Maturity Requirements and to assess the Maturity Levels. Also, it is the other levels of the framework which are not required or not assessed that are used by decision makers to make the Maturity Judgment.

Maturity framework development is generally a process of either choosing which assessment sets should be included in the framework or developing such assessment sets. Choosing the assessment sets is very important as this choice will set the tone for the entire review process.

2.2.2. Maturity Requirements Determination

The second stage in simulation review is maturity requirements determination. **Maturity Requirements Determination** is the process of determining the minimum required maturity level of each attribute in the given framework for the intended use of a simulation. This choice is often highly subjective and multiple levels may be acceptable under different conditions.

This stage is placed before Maturity Level Assessment to avoid confirmation bias, which should be one of the largest concerns in the review process. Confirmation bias is prevalent wherever people need information to answer questions on which they have some opinion, and engineers and scientists are not immune to this (Nickerson 1998). Typically, the individual or team performing the simulation review has some vested interest in either demonstrating the simulation should or should not be trusted. One step towards minimizing this bias is by independently performing each of the stages of simulation review. However, as it is likely that the same analyst will perform multiple stages, it is recommended that the maturity requirements are determined before the simulation's maturity is assessed because this ensures that the requirements will not be made to match the assessed simulation.

2.2.3. Maturity Level Assessment

The third stage in simulation review is maturity level assessment. **Maturity Level Assessment** is the process of determining the maturity level of each attribute in the given framework for a specific simulation. It is recommended that this stage either be performed after the Maturity Requirements Determination or independently from it. While this stage can be subjective due to the framework itself, it is often less subjective than determining the required levels.

2.2.4. Maturity Judgment

The fourth and final stage in simulation review is maturity judgment. **Maturity Judgment** is the process of determining if the required maturity levels of a situation are met by the assessed maturity levels of a simulation. Ideally, this process is simply ensuring that each attribute has achieved its required minimum level, but this is rarely the case. Usually, there are tradeoffs (attributes above their required levels may make up for attributes below their required levels) and re-evaluations of the minimum required levels. This is a very important stage which is prime for further research.

2.3. **Conclusion**

Chapter 2 developed basic maturity theory and then applied that theory to scientific computer simulation review. Maturity is a measurement of a specific attribute of an object. It is the rank of that attribute in the spectrum of the very worst to the very best achievable values of that attribute. This measurement requires four distinct components (maturity object, maturity attribute, maturity assessment set, and assessed maturity level). A further examination of these components produces some basic concepts which are important in understanding maturity assessment (e.g., Maturity Levels do not have a scale). Maturity assessments can be combined to create maturity assessment frameworks and it is these frameworks which are vital in many decision making processes, including scientific computer simulation review.

Using the concepts developed in maturity theory, scientific computer simulation review can be understood as a four stage process. In the first stage, a maturity assessment framework is developed. In the second stage, the framework is used to determine the maturity requirements of the intended use of the simulation. In the third stage, the framework is used to assess the simulation itself. Finally, in the fourth stage, the requirements are compared to the assessed levels in order to make a reach a conclusion on the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). This dissertation will focus on developing such a framework. The first step is to create an organizational structure for simulations and this step is performed in Chapter 3.

3. Hierarchy of Scientific Computer Simulation Components

Scientific computer simulations are mathematical imitations of real world physical processes, but what defines the boundaries of such an imitation? Is the simulation the equations which describe the imitation? Is it the source code used to perform the calculations? Is it the results of those calculations? Is it the documentation describing the process? Is it all of the above?

The focus of this chapter is one way to understand the boundaries of a simulation, by thinking of it as a collection of components. These components are best understood through the Hierarchy of Scientific Computer Simulation Components proposed here.

The first section addresses the six phases of performing a scientific computer simulation. These six phases are similar in some respects to the proposed hierarchy but ultimately have a much different purpose. The second section discusses a similar hierarchy used in biology (i.e., the Hierarchy of Life) and provides some background about similar organizational structures used in computer science. The third section defines the Hierarchy of Scientific Computer Simulation Components and defines each component (or level) in the hierarchy. The fourth section provides additional detail about the different components and how they interface with each other. Finally, the fifth section concludes and summarizes the contributions of this chapter.

3.1. The Six Phases of a Scientific Computer Simulation

Generally, a scientific computer simulation is separated into six phases. These "Six Phases of Computer Simulation" describe how a simulation is performed (Oberkampf and Roy 2010, Section 3.4). The six phases are summarized as follows:

1. **Conceptual Modeling Phase** – Specify the laws of physics which are of interest, how the physical system uses those laws, and identify any assumptions and approximations made in defining the system.
2. **Mathematical Modeling Phase** – Represent the important laws of physics and the physical system using mathematical equations.
3. **Discretization and Algorithm Selection Phase** – Represent those mathematical equations in a way that can be solved by a computer.
4. **Computer Programming Phase** – Code the discretized equations into a computer program
5. **Numerical Solution Phase** – Run the program to obtain the numerical results of the equations.

6. **Solution Representation Phase** – Calculate the quantities of interest from the numerical results.

Each phase can be thought of as a mapping of the information in the preceding phase into the new phase. For example, phase 2 takes the conceptual model and maps it to equations needed in phase 3. While these six phases do provide a way to organize the different *tasks* needed in creating a simulation, the phases are not intended to organize the different *components* that make up a simulation. Therefore, a new organizational tool is warranted.

The goal of this new tool is to create an organizational structure such that an analyst who understands the structure could use that structure to understand any scientific computer simulation. Therefore, the structure is heavily influenced by the organizational structures commonly used in the source code of specific higher level languages (mainly FORTRAN and MATLAB).

3.2. Focus on the Hierarchy as an Organizational Tool

A hierarchy is defined as an arrangement of items in which the items are represented as being "above", "below", or "at the same level" as another ("Hierarchy" 2013). The advantage of such an arrangement is that even the most complex entities can be easily understood through the components which they are made from. One such hierarchy which has been successfully used in science is the Hierarchy of Life.

3.2.1. Hierarchy of Life

The Hierarchy of Life is the organization of the different components which make up living organisms. One representation[9] of that hierarchy begins with "Organelles" and ends with an "Organism" and is given in Figure 1.

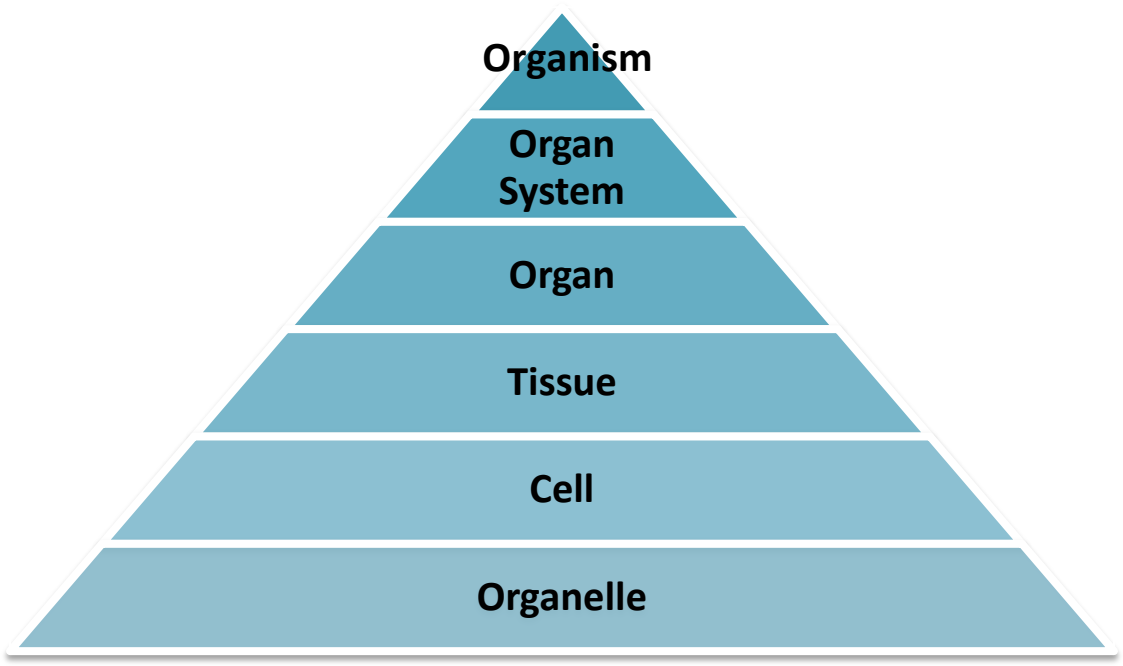


Figure 1: Hierarchy of Life

There are three significant features of this hierarchy

(1) The hierarchy is applicable to every organism
(2) There is no component of an organism that is not defined as a level in the hierarchy
(3) Each level can be understood as a collection of the levels below it.

[9] Other representations are possible that begin with lower levels (atoms) and end at higher levels (ecosystem).

These three features make this a very powerful tool which can be used to gain better understanding and insight into living organisms.

Developing a similar hierarchy for scientific computer simulations requires developing a set of components which comprise the various levels of the hierarchy and defining each component such that it is a set of the components at the next lower level (e.g., all tissues are sets of cells). This process is discussed in the next sub-section and is guided by the definitions provided in Computer Science literature and by the organizational structure of the source code commonly used for many scientific computer simulations.

3.2.2. Hierarchies in Computer Science

As defined in Chapter 1, a scientific computer simulation is the imitation of the behavior of something in the natural universe expressed as a mathematical model on a device with limited mathematical capability. That expression is commonly in the form of a computer program or algorithm. An **Algorithm** is an unambiguous description of a finite set of operations which specifics a sequence of operations that always halts (Brainerd and Landweber 1974). To perform such an algorithm on a computer, the algorithm must be in machine language. **Machine Language** is the sequence of bits that directly controls a processor, causing it to add, compare, move data from one place to another, and so forth at appropriate times (Scott 2009).

However, an algorithm is almost never written directly in machine language, as any but the simplest of algorithms would be too complex for an analyst to understand. Instead, the algorithm is written using a "higher level" language. A **Higher Level Language** is the expression of an algorithm in a logical fashion which can be turned into machine language by a compiler. Scott (2009) identifies six basic types of higher level languages, with those of von Neumann and object-oriented being the most popular (e.g., C and FORTRAN).

One of the challenges in defining a hierarchy common to all scientific computer simulations is that those simulations can be expressed in various higher level languages and those languages will each have their own set of rules for organization. However, some universal components have been identified. For example, Halstead (1977) organizes source code by thinking of it as a collection of operands and operators. **Operands** are the variables or constants employed in the specific implementation of the algorithm. **Operators** are the symbols or combinations of symbols that affect the value or ordering of an operand. While these components are universal, they are too focused on aspects of source code to make appropriate levels for the intended hierarchy.

Yourdon and Constantine (1977) organize source code by thinking of it as a collection of statements and modules. A **Statement** is the smallest unit of a program which is a well-defined, complete instruction, command, declaration, etc. A **Module** is a lexically contiguous sequence of program statements, bounded by elements, having an aggregate identifier. Yourdon and Constantine build on these components and discuss the various ways to structure source code. While Yourdon and Constantine's components do make good levels in a hierarchy, their definitions are broad in order to account for the many uses of source code. Because the focus of this hierarchy is only source code which is used to represent a scientific computer simulation, the scope of these terms can be narrowed.

3.2.3. Focus on Scientific Computer Simulations

The goal of the hierarchy proposed here is to provide an organizational structure for scientific computer simulations by defining the different components of those simulations in a hierarchical fashion (i.e., ever component is a set of the components directly below it). While the hierarchy uses many similar terms and concepts currently in use (e.g., statement, code, etc.) there is not necessarily a direct one-to-one relationship between the components of the hierarchy and the components in source code. The different components used in source code are the result of the rules of the higher level language the code is written in. The hierarchy described here does not necessarily conform to the rules of any specific language (however it is very similar to FORTRAN and MATLAB). That is, instead of being focused on how a scientific computer simulation should be represented in a specific language, the hierarchy is focused on how any simulation could be understood in the mind of an analyst.

3.3. Components of the Hierarchy of Scientific Computer Simulation Components

The **Hierarchy of Scientific Computer Simulation Components** is a hierarchy which arranges the different components of a scientific computer simulation into six levels: Term, Assignment Statement, Coded Physical Function, Coded Group, Computer Code, and Complete Set of Codes. Similar to the Hierarchy of Life's ability to organize any life form, this hierarchy was developed to organize scientific computer simulations. This hierarchy is represented in Figure 2.

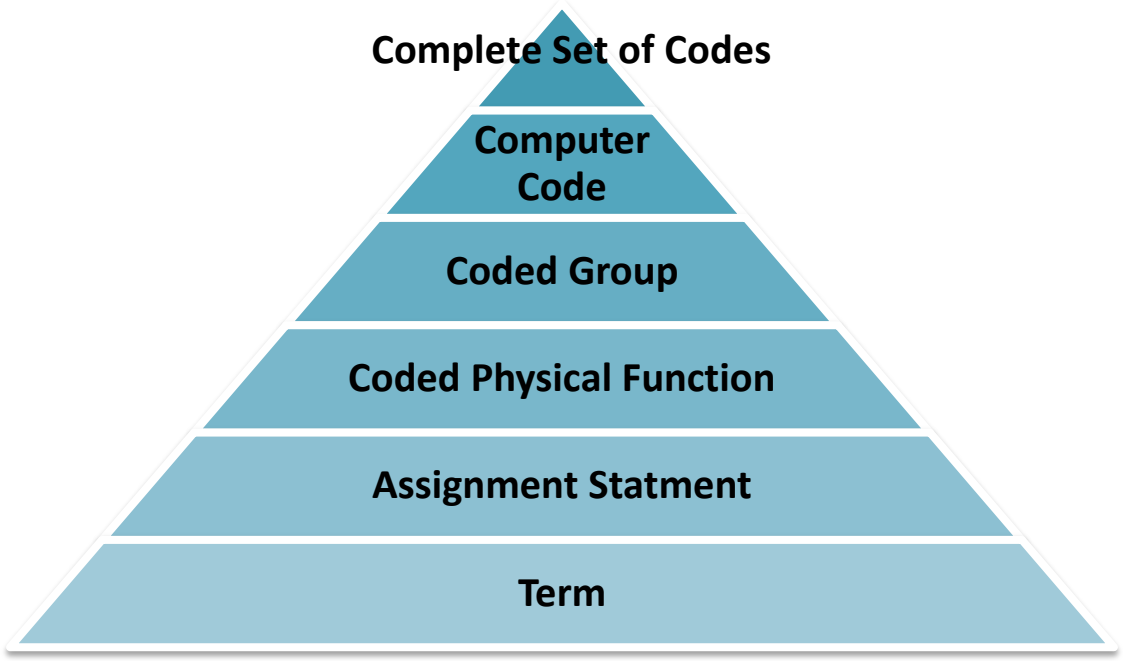


Figure 2: Hierarchy of Scientific Computer Simulations

Each of the levels [10]in the hierarchy was defined to maintain the three features from the Hierarchy of Life:

(1) The hierarchy is applicable to every scientific computer simulation
(2) There is no component of scientific computer simulation that is not defined as a level in the hierarchy
(3) Each level can be understood of as a collection of the levels below it.

The components and structure of the hierarchy are taken from the informal organizational structure of the source code from many scientific computer simulations.

[10] It is important to note that a scientific computer simulation is not a level in the hierarchy. This was done on purpose and its reasons will become clear at the end of the chapter.

3.3.1. Term

A **Term** is defined to be an expression which is a separable part of an Assignment Statement (the next higher level in the hierarchy). It is the smallest component of a scientific computer simulation generally considered and should be thought of as being consistent with the concept of a "term" in mathematics. It is related to the concepts of operands and operators as defined in software science (Halstead 1977), but it is not identical to these concepts. A Term can be a single operand, but a term can also be a collection of operands and operators. Generally, a term is best thought of as a logical collection of constants, variables, and operators.

There are two types of terms, simple and complex. A **Simple Term** is a term which cannot be further separated into other terms (i.e., an operand). A **Complex Term** is a term which can be further separated into other terms.

Example 3.1 demonstrates different types terms that can appear in a simulation.

Example 3.1: Term

The following is a line of code from MATLAB.

```
y = 3*x + 17;
```

The terms are `y`, `3*x`, and `17`. Of these three, `y` and `17` are simple terms as they cannot be further separated. The term `3*x` is a complex term as it can be further separated into the simple terms `3` and `x`.

Note that `3*x + 17` could also be considered a term (e.g., the term on the Right Hand Side) and would be a complex term.

3.3.2. Assignment Statement

An **Assignment Statement** is a set of terms which are used to assign a value to a specific dependent variable. The concept of an Assignment Statement comes from that of a statement as defined by Yourdon and Constantine (1979). The variable whose value is calculated may be a vector or matrix, in which case the same Assignment Statement is called repeatedly and each call assigns a value to a different element of the variable. Assignment statements can be written on multiple lines of code or take advantage of a loop operation (`for`, `while`, etc.).

The concept of "Assignment Statement" originates in computer science and represents the operation of the specified group of terms. As all Assignment

Statements are functions, it must contain at least two terms, the Left Hand Side (which is the term or terms whose values are being assigned) and the Right Hand Side (which is the collection of terms and operators used in the calculation). Example 3.2 demonstrates different types of Assignment Statements that can appear in a simulation.

Example 2: Assignment Statements

The following are lines of code which are examples of Assignment Statements in MATLAB. Statements (1) – (5) are Assignment Statements while statement (6) is not.

```
(1)     imeth = 1;%input('Enter method (1): ');
(2)     dt = diffno*dx^2/anu; % time step
(3)     du(1) = 0;
(4)     du(jmax) = 0;
(5)     for j=2:jmax-1
          du(j) = -0.5*cfl*(u(j+1)-u(j-1))+0.5*(u(j+1)-2*u(j)+u(j-1));
        end
(6)     % allow for boundary conditions
```

Operationally, comments are not part of the Assignment Statement as a change in the comment will not impact the resulting numerical value of the dependent variable of the function. While comments are often placed near Assignment Statements (as given in (1) and (2)), it is best to think of comments as an extension of the documentation for a computer simulation instead of part of the Assignment Statement itself.

3.3.3. Coded Physical Function

A *term* is the smallest component of the proposed hierarchy and an *Assignment Statement* is the smallest "complete" unit of a computer program, but it is the *Coded Physical Function* which is the most fundamental component of a scientific computer simulation. That is, while computer simulations are written as collections of Terms in Assignment Statements they are thought of as collections of Coded Physical Functions. The phrase *Coded Physical Function* is new, but the concept should be familiar.

On the surface, a **Coded Physical Function**[11] is a set of Assignment Statements. This concept is further developed in the following subsections.

[11] This definition is not underlined because it will be added to later.

3.3.3.1. Review of Functions

This first subsection will review basics of functions. These basic definitions are important as they provide the foundation for the discussion which follows. The following definitions for function, domain, and range are provided by Stewart (1995).

A **Function** is a rule that assigns to each element x in a set $\mathbb{A}$ exactly one element, called $f(x)$ in set $\mathbb{B}$. The **Domain of the Function** is the set of elements in $\mathbb{A}$. The **Range of the Function** is the set of elements in $\mathbb{B}$.

Another way to think of the range of the function is the space described by the dependent variables of the function. The **Dependent Variables** are all variables in the function which are considered output as they are assigned a value by the function. Another way to think of the domain of the function is the space described by the independent variables of the function. The **Independent Variables** are all variables in the function which are considered input as they are needed to assign the value to the dependent variables.

It is easy to only think in terms of functions when dealing with scientific computer simulations as those are the only objects which show up in source code. However, many of those functions did not start as functions, but rather as equations.

3.3.3.2. Introduction to Physical Equations

An **Equation** is a mathematical statement which states the equality of two sets of terms. Because equations do not have independent or dependent variables they do not have a domain or a range. Thus the term variable space is introduced. The **Variable Space** of an equation is an *N*-dimensional space which contains all possible values of the variables in that equation (assuming there are *N* variables).

While people often talk of equations being "solved", for scientific computer simulations, an equation must be functionalized before it is solved. **Functionalization** is the process of turning an equation into a function. Typically this process is performed by choosing the dependent variable (or variables) and algebraically manipulating the equation until the dependent variable is alone on the Left Hand Side (LHS). Except for all but the simplest cases, an equation can usually be functionalized into many different functions.

Scientific computer simulations are concerned with solving one type of equations, physical equations. A **Physical Equation** is an equation whose constants and variables are related to quantities in the physical universe and whose form is an attempt to define some relationship between those quantities under certain conditions.

Similarly, a **Physical Function** is the functionalization of the physical equation into a specific functional form. It is very important to understand the distinction between an equation and a physical equation (or a function and a physical function).

Physical equations are subsets of equations. That is, all physical equations are equations, but not all equations are physical equations. All equations must obey the laws of their mathematical operators, so must physical equations. However, physical equations have three major characteristics which distinguish them from other types of equations:

(1) Physical equations usually have associated units.

> While the variables and constants in an equation are associated with certain spaces or sets (e.g., set of all real numbers, set of all complex numbers, etc.), the variables and constants in a physical equation are often associated with multiple spaces, where the equivalent value of the variable in one space is often a different numeric value in another space. For example, 100, 212, and 373.15 are not numerically equal, but they are all the represent the same value of the physical property of temperature in three different spaces (Celsius, Fahrenheit, and Kelvin).

(2) Physical equations claim to represent a physical truth.

> All equations claim to represent a mathematical truth. Given some set of numbers, a certain operation can be performed which results in another set of numbers. Such truth claims are considered *a priori* as they do not require any evidence outside of the equation itself to demonstrate it is true. However, this is not the case for physical equations. While the mathematical form of the physical equation must be correct (*a priori*), the fact that it is a physical equation means it is making a claim about the way the physical universe works and such a claim requires evidence (*a posteriori*).

(3) Physical equations are only applicable in a restricted application space.

> The variable space of any equation is typically very large. For example, the value of the variables is often restricted to the set of real numbers, or some other large set. However, the value of the variables in a physical equation is often further restricted. For example, the value of velocity in a physical equation will have an upper and lower bound of the speed of light. While a value higher than the speed of light would not violate any mathematical constraints on the equation, it would violate the physical constraint that an object's velocity cannot reach the speed of light. In other words, certain values would be outside of the application space of the physical equation. The **Application Space** is the mathematical space over which the physical

equation is claimed to be valid. When the physical equation is functionalized, the application space becomes the application range and the application domain. The **Application Domain** is the space described by all of the independent variables in which the physical function is claimed to be valid. The **Application Range** is the space described by all of the dependent variables in which the physical function is claimed to be valid. Typically, the application domain and application range of a physical function are much smaller than the domain and range of the same mathematical function. That is mathematically, the function can operate in a larger space than it should be physically applied. A demonstration of the use of the application space can be found later in Example 3.5.

Because physical equations require *a posteriori* knowledge, their origin must come from outside of pure reason alone. Typically, physical equations are grouped by where they originate from, resulting in the following three types of physical equations:

(1) First Principle Physical Equations
(2) Empirical Physical Equations
(3) Definitional Physical Equations

Each of these three origins is further discussed in the following sub-sections. Example 3.3 demonstrates the difference between an equation and a physical equation.

Example 3.3: Equation vs. Physical Equation

The following is an example of an equation.

$$3 \cdot x + 2 \cdot y + 1 \cdot z = 9$$

While we could change the variables to physical quantities, that does not make this a physical equation. For example, we could substitute pressure (P) for the x-term, velocity (v) for the y-term, and elevation (z) for the z-term, but the result is not a physical equation.

$$3 \cdot P + 2 \cdot v + 1 \cdot z = 9$$

The above equation does have variables that are related to quantities in the physical universe and does attempt to describe a relationship between those quantities, but this relationship is fictional. That is, this relationship was not arrived at through observing how these quantities behave; rather it was randomly generated.

While all equations are *a priori* a physical equation claims to represent the behavior of the physical universe and therefore is also *a posteriori*. One such a relationship which is a physical equation is the Bernoulli equation, given below.

$$\frac{P}{\rho} + \frac{v^2}{2} + g \cdot z = A$$

When we say this is a physical equation, we mean that the terms in the equation and its form are were influenced by some evidence in the physical universe.

3.3.3.3. First Principle Physical Equations

A **First Principles Physical Equation** is a physical equation which is a mathematical representation of a principle considered true by the scientific community under specific conditions. These first principles are not axioms which can be used to describe the behavior of the universe. Instead, they are the best descriptions of nature based on many observations. Often, they require certain assumptions along with a set of axioms in mathematics in order to express the relationship. While they are not always applicable, they are broadly applicable and almost never challenged where they are commonly used (e.g., the Conservation of Mass would not be applied to the uranium in a nuclear reactor, but it is rightly applied to the water flowing through the reactor to cool it).

A **First Principle** is a principle (i.e., law or concept) from a set of principles which is considered true by the scientific community under specific conditions. An example of a first principle is the concept that "mass is conserved in a closed system". A first

principle equation is a mathematical representation of that principle. Such a mathematical representation for this example is given in Eq. 3.1

$$\iiint_{V(t)} \boldsymbol{\rho} \cdot \boldsymbol{dV} = \boldsymbol{A} \tag{3.1}$$

It is important to stress that the mathematical representation is not necessarily the same as the actual first principle. This distinction is stressed because many equations which are called first principle equations are actually further derivations of first principle equations. That is, they are equations with further assumptions not found in the first principle. For example, another common form of the conservation of mass equation is given by Eq. 3.2.

$$\frac{\boldsymbol{d}}{\boldsymbol{dt}} \iiint_{V(t)} \boldsymbol{\rho} \cdot \boldsymbol{dV} = \iiint_{V(t)} \frac{\boldsymbol{\partial \rho}}{\boldsymbol{\partial t}} \cdot \boldsymbol{dV} + \iint_{S(t)} \boldsymbol{\rho} \cdot \vec{\boldsymbol{u}} \odot \vec{\boldsymbol{n}} \cdot \boldsymbol{dS} = \mathbf{0} \tag{3.2}$$

This equation is not a direct representation of the first principle, but adds multiple assumptions that the first principle of the conservation of mass does not make including the following (Kundu and Cohen 2004):

1. The velocity field is continuous.
2. Reynolds Transport theorem is correct.

These assumptions are generally considered reasonable and are almost never questioned, but they are still assumptions not made in the first principle of the conservation of mass. Typically, equations are still considered first principle equations if the assumptions made are assumed to be minor. However, in many cases the additional assumptions are considered major and the equation is called a pseudo-first principle equation or a semi-empirical equation. A **Pseudo-First Principle Physical Equation** or **Semi-Empirical Physical Equation** is an equation which is based on a first principle equation, but includes the introduction of sub-models. One example is the Reynolds Averaged Navier Stokes given by Eq. 3.3. This equation is based on derivations of the conservation of mass and momentum, but some of these derivations have significant assumptions associated with them which are not made by the first principles themselves.

$$\frac{\boldsymbol{\partial}\overline{\vec{\boldsymbol{u}}}}{\boldsymbol{\partial t}} + \boldsymbol{\nabla} \odot \left(\overline{\vec{\boldsymbol{u}}} \cdot \overline{\vec{\boldsymbol{u}}}\right) + \boldsymbol{\nabla} \odot \left(\overline{\vec{\boldsymbol{u}'}} \cdot \overline{\vec{\boldsymbol{u}'}}\right) = -\frac{\mathbf{1}}{\boldsymbol{\rho}} \cdot \boldsymbol{\nabla} \overline{\boldsymbol{p}} + \boldsymbol{\nu} \cdot \boldsymbol{\nabla}^{\mathbf{2}} \overline{\vec{\boldsymbol{u}}} \tag{3.3}$$

To clarify this potential confusion, the concept of a genesis equation is introduced. A **Genesis Equation** is the exact representation of the relationship described by the first principle in the form of an equation and contains the minimum number of additional assumptions necessary to represent that principle. In other words, every first principle should have one (and only one) genesis equation. Such an equation is

defined to draw a distinction between the assumptions made by the first principle itself and any additional assumptions made in deriving the specific form the first principle equation. For example, both Eq. 3.1 and Eq. 3.2 are first principle equations; however, only Eq. 3.1 is the genesis equation as Eq. 3.2 has additional assumptions not made by the first principle.

Ideally, any first principle equation (pseudo- or otherwise) would be able to trace its derivation from the genesis equation to its current form, listing not only the steps taken in the derivation, but also the all of the assumptions of the new form of the equation. An example of such a scheme is provided in the Appendices B,C, and D of this dissertation.

Appendix B provides a listing of the first principles and their genesis equations. While this listing is only suggested, it would be very helpful if there was one common starting point for all derivations of first principle equations instead of multiple starting points. Appendix C provides some common derivations of first principle equations. These derivations provide the following information for each derivation of the first principle equation:

- **Equation ID** – This is the identification number for this particular equation.
- **Parent ID** – This is the identification number of the parent equation (i.e., the equation from which the current equation was derived.) This number could be used to trace any specific derivation of a first principle equation back to the genesis equation.
- **Starting Assumptions** – This is a list of all of the assumptions which are in the parent equation.
- **Description of this step** – This is a brief description of the derivation which is being performed to the parent equation in this step
- **Assumption of this step** – This is a list of all of the additional assumptions which are made during the derivation of this step.
- **New Equation** – The final form of the equation after the derivation of this step. It is suggested that this form is given in vector and indices notation.

An example table for this information is given Table 3-1.

Table 3-1: Derivation Table

Equation ID	Identification number of this equation.
Parent ID	Identification number of the parent equation.
Starting Assumptions	A list of each assumption made in the derivation of the parent equation going all the way back to the genesis equation.
Description of this step	A brief description of this step.
Assumptions of this step	A list of each assumption made in this step.
New Equation (vector)	The new equation represented in vector notation.
New Equation (indices)	The new equation represented in indices notation.

Appendix D provides some common derivations of pseudo-first principle equations (e.g., Navier-Stokes).

The goal in providing these appendices is that the derivation of first principle equations will become more formalized, with a listing of all known assumptions of that particular form of the equation readily available. Such additional formalization would greatly aid the act of ensuring that the assumptions of a physical equation were appropriately satisfied.

3.3.3.4. Empirical Physical Equations

An **Empirical Physical Equation** is a physical equation whose form, constants, and variables are closely related to the experimental observation of some physical quantity. Typically, these equations are the results of mathematical fits to specific data sets. They are not meant to be the broad description of nature, but narrow descriptions of a specific aspect of nature under very specific conditions.

Per their definition, an empirical physical equation describes some empirical or observed relationship. Typically this means that certain physical quantities are varied and the impact on another physical quantity is measured. As such, most empirical physical equations are developed as functions and therefore may not be able to be rearranged into other functional forms (i.e., with other variables being the dependent variable). For example, Bernoulli's equation describes the relationship between pressure, velocity, and gravity. Thus, this equation can be rearranged to have any one of those three physical quantities be the dependent variable. Conversely, when a Critical Heat Flux (CHF) correlation is developed, the CHF value is measured at various temperatures, pressures, and flow rates. The final CHF correlation has CHF

as the dependent variable as a function of the temperature, pressure and flow rate. Rearranging such an equation so that the temperature would be a function of the CHF, pressure, and flow rate is mathematically possible but is most likely technically incorrect. This example demonstrates the power of a first principle physical equation compared to an empirical physical equation. The first principle equation represents a general behavior of nature and can transformed into any number of functional relationships, however the empirical equation represents a single functional relationship and should not be transformed into any other function.

The form of an empirical equation is arbitrary. Typically, an analyst will have certain amount of data to fit and any fit to that data is an empirical equation. Mathematically, *N* data points can always be exactly fit by an *N*-degree polynomial. However, the analyst usually has other constraints to consider besides ensuring that the given data is perfectly fit. Often, something is believed about the underlying physical process (i.e., it is linear, it is quadratic) and the analyst will attempt to capture that information in the form of the equation.

While the focus on first principle equations is to ensure the derivation and any additional assumptions are correct, the focus on empirical equations is to ensure the data is correct and that the empirical relationship represented by the data is an actual relationship in the physical universe. The concern with empirical equations is that the data and any relationship derived from the data are missing some key component which is important where the equation will be applied. This concern is hard to alleviate, as the data is often obtained in an experiment which is under different conditions than the actual use of the equation.

3.3.3.5. Definitional Physical Equation

A **Definitional Physical Equation** is a physical equation whose form, constants, and variables are chosen to define some physical quantity. Some examples of definitional physical equations are the equation used to convert temperature form Fahrenheit to Celsius, the equation used to calculate an average temperature on a metal plate, and the equation used to calculate mass flow rate from density, velocity, and flow area. These are not first principle equations in that their origin is not a first principle and they are not empirical equations as their origin is not a fit to data.

It is important to distinguish between two types of definitional physical equations, those which are always true (known) and those which are assumed to be true (assumed). A **Known Definitional Physical Equation** is a definitional physical equation which is unconditionally true given some set of basic underlying assumptions are true. For example, the equation which describes how to convert Fahrenheit to Celsius or how to calculate the mass flow rate from density, velocity,

and flow area (assuming that velocity is a continuous field). These equations are known and true because the definition they describe never changes.

An **Assumed Definitional Physical Equation** is a definitional physical equation which is not unconditionally true but is usually assumed to be true. These equations are bit harder to understand as it is generally not the form of the equation which is in question but the meaning of the variables. For example, the average temperature on the surface of a plate could be calculated by finding the mean of three temperatures on the surface. However, the equation which averaged these temperatures would be an assumed definitional physical equation. It is assumed, not because the form of the equation is in question (as the equation for a mean is known), but because it is assumed that the mean of those three temperatures represents the average temperature on the surface of a plate. Maybe one temperature should have been weighted more than the others, maybe other temperatures should have been used, maybe only one temperature should be used, etc… To reiterate, it is usually not the form of an assumed definitional physical equation which makes it assumed, rather it is assumed because what the variables represent is assumed. Example 3.3 provides a demonstration of the difference between a known and an assumed definitional physical equation.

Example 3.3: Absolute vs. Assumed Definitional Physical Equations

Unit conversion is a good example of an *known definitional physical equation*, such as the unit conversion from Fahrenheit to Celsius as given below.

$$T(°C) = \frac{T(°F) - 32}{1.8}$$

This physical equation is unconditionally true as it is a correct representation of the temperature in Celsius as a function of the temperature in Fahrenheit.

An averaged value is a good example of an *assumed definitional physical equation*, such as using the mean of three temperatures to represent the average temperature as given below.

$$T_{Average} = \frac{T_1 + T_2 + T_3}{3}$$

While the form of the equation is correct (i.e., this is the correct way to calculate a mean of three quantities), the assumption that the mean of these three temperatures is the average temperature may not be correct.

3.3.3.6. Definition of a Coded Physical Function

The following three steps must be taken to so that the physical equation can be solved by a computer:

1. The physical equation must be written as a physical function. In some instances, this means selecting which of the variables will be the dependent variable (i.e., functionalization).
2. The physical equation may have to be re-written in a discretized form so it can be solved by computer (i.e., discretization).
3. The physical function must be coded in a specific computer language.

Generally the steps occur in order, but it is also possible that the dependent variable chosen is one of the variables produced in the discretization process.

Finally, what appears in the computer simulation is not the physical equation, or even the physical function, but a Coded Physical Function. A **Coded Physical Function** is a single physical function which has been discretized and written as some combination of Assignment Statements in a specific computer language.

Usually, a physical function is discretized by using Taylor series expansion so that it can be solved by a computer. While such a technique is very powerful, the resulting discretized function is different from the original physical function. Therefore, the solution of the original physical function and that of the discretized function will also be different. The question of how different these two solutions are from one another is addressed by topic of Verification.

It is important to stress that while the *Term* is the smallest component of a computer simulation, the *Coded Physical Function* is the fundamental component. It is fundamental because when most people think about, discuss, or analyze computer simulations they are generally focused on the Coded Physical Functions. The name *Coded Physical Function* was specifically chosen to highlight this and to remind analysts of the following three facts:

1. Coded physical functions are coded in a computer language and are not the original mathematical function (i.e., they are *coded*).
2. Coded physical functions are physical functions and therefore related to physical quantities in the physical universe which have certain constraints (i.e., they are *physical*).
3. Coded physical functions must be functions and not simply equations (i.e., they are *functions*).

Each of these facts results in certain constraints on the Coded Physical Functions which are addressed in the following section. Examples of Coded Physical Functions are given in Example 3.4.

Example 3.4: Coded Physical Functions

The following are four examples of Coded Physical Functions in MATLAB:

(1)
```
P = n*R*T/t;
```

(2)
```
dt = diffno*dx^2/anu;
```

(3)
```
du(1) = 0;
du(jmax) = 0;
for j=2:jmax-1
   du(j) = -0.5*cfl*(u(j+1)-u(j-1))+0.5*(u(j+1)-2*u(j)+u(j-1));
end
```

(4)
```
if x > 0
   P = 5*x + 3;
else
   P = x^2 + 1;
end
```

(4) could also be written as:

```
P = (x>0)*(5*x + 3) + (x<=0)*(x^2 + 9);
```

In examples (1) and (2) the Coded Physical Functions are written as single Assignment Statements. While in example (3) three Assignment Statements are needed for one Coded Physical Function. Example (4) is an oddity as it is two different Coded Physical Functions, `(5*x + 3)` and `(x^2 + 9)`, written as one Assignment Statement. However, as the example shows, it can also be written as two Assignment Statements.

3.3.3.7. Constraints on a Coded Physical Function

A **Constraint** is a limitation placed on the variable space of an equation. Any constraints placed on the variable space of an equation would also apply to any function made from that equation and would appear as a constraint on the domain or the range. The following are four types of constraints frequently associated with Coded Physical Functions:

1. **Mathematical Constraint** – Any constraint on the variable space that arises from the rules of mathematics. The function itself can be thought of as a mathematical constraint on the range as the function limits the values the dependent variable may take.

2. **Physical Constraint** – Any constraint on the variable space that is a result of the permissible values of the physical quantities in question. These constraints are often expressed in terms of the application space of the physical equation.

3. **Discretization Constraint** – Any constraint on the variable space that is the result of using a computer with finite precision to perform calculations which would otherwise be continuous.

4. **Discretionary Constraint** – Any constraint on the variable space which an analyst chooses to enforce. This constraint is not forced by any stipulations of mathematics, nature, or the computer; rather it is one which the analyst chooses to enforce on the equation.

Example 3.5 provides an example of each of the different types of constraints.

Example 3.5: Constraints

Consider a Coded Physical Function made from Bernoulli's equation such that the height term is the dependent variable:

$$\frac{P}{\rho} + \frac{v^2}{2} + g \cdot z = A \qquad \rightarrow \qquad \texttt{z = (A - P/rho - v^2/2)/g;}$$

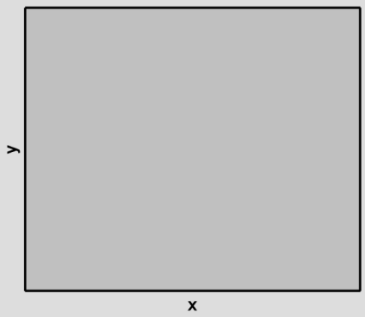
The constraints easiest to identify are the *physical constraints*, as these are solely dependent on the physical quantities in the function. The physical constraints are as follows:

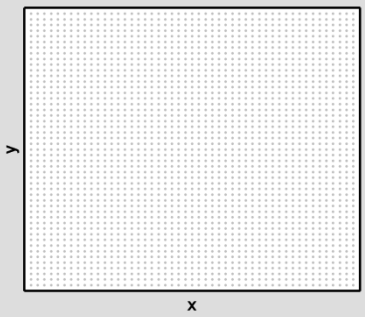
(1) $P \geq 0$ (there is no such thing as a negative pressure)
(2) $\rho > 0$ (there is no such thing as a negative or zero density)
(3) $-c \leq v \leq c$ (the velocity cannot be greater than the speed of light)
(4) $g \in \Re$ (acceleration due to gravity is a real number)
(5) $z \in \Re$ (height is a real number)
(6) $A = k$(this must be a constant)

Next, the *mathematical constraints* are identified, as these are solely dependent on the form of the function. The mathematical constraints are as follows:

(1) $g \neq 0$ (division by zero is not allowed)

The *discretization constraints* are not so easily identified. These are more prevalent where the parent equation is partial differential in nature, but there are still discretization constraints on this function. The constraints are seen in the possible values that can be taken by values in the domain and the resulting values in the range. This can best be illustrated by the following figures.





The figure on the left is part of the domain of the actual function. That is, it can be any value in the xy-plane (where x and y are any two of the independent variables in the function). The figure on the right is part of the domain of the actual Coded Physical Function. Notice that the domain is no longer continuous and only the values at discrete points are available. This is a result of the finite precision of the computer. Not all values which are mathematically possible can appear in the domain as the computer can not represent those values.

Finally, an *arbitrary constraint* would be any other constraint on this Coded Physical Function which was not a physical, mathematical, or discretization constraint. For example, suppose the analyst only wanted to consider situations where the pressure was above 100. Than the arbitrary constraint would be as follows:

(1) $P > 100$

3.3.3.8. Coded Physical Functions appearing outside the Source Code

It is important to remember that not all Coded Physical Functions will appear in the source code. Quite often, very important calculations may be performed outside of the source code. These calculations are generally used to provide input to the simulations or used to manipulate the results of the output. Even though these calculations are not part of the source code, they are still considered Coded Physical Functions and they are part of the simulation.

3.3.4. Coded Group

A **Coded Group** is a set of one or more Coded Physical Functions. These may also be known as sub-routines, models, groups, or sub-programs, but they do not have to be. The term "Coded Group" was chosen as those other terms have other specific meanings.

The Coded Physical Functions must be assigned to a Coded Group by an analyst. Thus, the Coded Group is the most variable of all the components. Some analysts may choose to think of each Coded Physical Function as its own group, some may place the functions into a group based on some criteria (i.e., the functions in a group are part of a sub-routine or generally considered together as a model), and some analysts may choose not to use Coded Groups at all. Coded Groups are an organizational tool and an analyst can choose not to use them[12]. Example 3.6 provides a demonstration of Coded Groups.

Example 3.6: Coded Group

The following are lines of code from MATLAB:

```
% preallocate for speed
du = zeros(jmax,1);

% initial data:
u  = uinite;
u1 = uinite;
u2 = uinite;
uSpace = u;

% Diffusion Number
diffNo = nu*dt/(dx^2);
```

Each line is an Assignment Statement and also a Coded Physical Function. We could break these lines up by their comments and consider this section of code to have three Coded Groups. We could also consider all of these functions to be in the same group (in this case called "Calculate Variables") or we may not bother assigning the functions to any specific group altogether. Using groups is a choice that can make understanding the simulation easier, but it is not required.

3.3.5. Computer Code

A **Computer Code** is a set of one or more Coded Groups. That is, when an analyst is referring to a Computer Code, he or she is referring to some set of Coded Groups and wishes to consider the set instead of the individual groups. Like Coded Groups, the number of Coded Physical Functions in a Computer Code is somewhat arbitrary, as almost any Computer Code could be split into two codes and any two codes could be made into a single Computer Code. Of the levels discussed so far, most readers will likely find themselves most familiar with this level, as the Computer Code is intimately connected with the source code.

[12] If Coded Groups are not used, then every Coded Physical Function can be considered its own group or all Coded Physical Functions in a computer code can be considered one group.

Not all Coded Physical Functions contained in a code are exercised during every execution of that code. Often, the analyst is able to choose which functions to exercise or the code itself may choose which functions depending on the calculated values of quantities in the code. These choices represent two different selection techniques, external selection and internal selection. **External Selection** is when the external analyst chooses which Coded Physical Functions are exercised in a code execution. These are commonly made through modeling options. For example, if a Computer Code has two heat transfer models and the analyst must choose one, then the Computer Code requires an external selection. **Internal Selection** is when the code itself chooses which Coded Physical Functions are exercised in a code execution. For example, a flow map being used to select different heat transfer correlations based on some fluid conditions is an internal selection.

These two concepts are needed for comparing the output of multiple executions of the same Computer Code. Namely, even though it may be the same source code, the actual Coded Physical Functions which are used could change between code executions and an example of this is demonstrated in Example 3.7.

3.3.5.1. Forms of Computer Codes

It is helpful to think of the Computer Code as a set of Coded Physical Functions with both internal and external selections. This means that a code has both a static form which is often indefinite and a dynamic form which must be explicit. The **Static Form of a Computer Code** is the set of all Coded Physical Functions which exist in the Computer Code. Generally, when an analyst refers to a code, he or she is referring to the static form of that Computer Code. The **Dynamic Form of a Computer Code** is the set of all Coded Physical Functions which are executed at during a specific execution of the code. The dynamic form must be a subset of the static form.

Before a code can be executed it must be made explicit. In the static form of most computer codes, there are often options as to what Coded Physical Functions will be used during a specific execution of the code (e.g., model selection). Because multiple Coded Physical Functions are available for use, the analyst must often select the specific Coded Physical Functions which will be used during one specific execution. If the analyst has not made that selection, the code is considered an indefinite computer code. An **Indefinite Computer Code** is a code where the selection of at least one Coded Physical Function is required before the code will execute. If the analyst has made the necessary selections, the code is considered an explicit computer code. An **Explicit Computer Code** is a code where all necessary selections of Coded Physical Functions have been made. Such a code may still require values for input parameters (e.g., pressure, temperature, etc.), but once those values are given

the code can execute. Example 3.7 provides a demonstration of these different forms of a Computer Code.

Example 3.7: Example of a Computer Code

The following is a simple Computer Code from MATLAB, where the inputs to the code are `X`, `A`, `P`, `rho`, `v`, and `g`.

The *static* form of the Computer Code is given as follows:

```
if X = 1
    z = (A - P/rho - v^2/2)/g;
else
    z = A + P + rho +v - g;
end
```

Depending on the value of `X` chosen, the *dynamic* from is one of the following:

```
z = (A - P/rho - v^2/2)/g;
```

Or

```
z = A + P + rho +v - g;
```

Because the value `X` must be chosen by an analyst, the *static* form of the code is also *indefinite*. That is, there are two Coded Physical Functions which could be used in the code execution and the analyst must choose one of the two to execute by either setting the value for `X` as "0" or "1".

This example was specifically chosen to demonstrate that the *static* form of a code can produce very different *dynamic* forms. Depending on the value of "`X`", the dynamic form will either calculate "`z`" using the Bernoulli equation or using some random combination of variables. The dynamic form using Bernoulli's equation should be trusted much more than the dynamic form using the random combination. It should be dynamic form of the Computer Code and not the static form which is focused upon when answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

3.3.5.2. Equivalencies among Computer Codes

How can an analyst be certain that a code which is tested will be the same code which is used in the work environment? For example, suppose someone tested the code given in Example 3.7 with `X = 1` but actually used it with `X = 2`. They may have thought that such testing would apply since it was the same code, but this is not true. While it is easy to differentiate between the dynamic forms of this simple example, most real codes are much more complex. To help with this situation, the following equivalencies for Computer Codes are defined.

1. **Statically Equivalent** – Two Computer Codes are statically equivalent if they have the have the same static form. That is, the codes have identical sets of Coded Physical Functions which are programmed to be executed in the same way. Typically, a code is only statically equivalent with the same version of itself, but could be equivalent with its expression in another computer language.

2. **Explicitly Equivalent** – Two Computer Codes are explicitly equivalent if they are first statically equivalent and second if they have the same external selections made. Many QA programs are set up to make codes explicitly equivalent such that the same physical models are used in the same manner.

3. **Dynamically Equivalent** – Two Computer Codes are dynamically equivalent if they are first explicitly equivalent and second if the internal selections made in each code are identical. Dynamically equivalence can be a confusing concept, especially if the computer simulation is time dependent. If the simulations are time dependent, do the codes need to make the same internal selections at the same time step or just at some point during the code run to be dynamically equivalent? In general, the only way to ensure that a code is dynamically equivalent with another code is if those codes are explicitly equivalent and contain no internal selections (which essentially never occur in real-world simulations).

These three equivalencies can be used to better define what is meant when an analyst says that two codes are "the same". These concepts are meant to highlight that fact that static equivalence is much different than dynamic equivalence and sometimes it is more appropriate to think of two simulations which were performed with the same Computer Code as resulting from two different Computer Codes.

Such equivalencies are often helpful when discussing version control of a computer code. Newer versions of the same computer code may or may not contain the same Coded Physical Functions, thus, it is likely that the same computer code is not statically equivalent with a previous version of itself.

3.3.6. Complete Set of Codes

A **Complete Set of Codes** is a set of one or more Computer Codes used to perform a simulation. The set is complete in the sense that every Coded Physical Function needed to perform the simulation must be included in the set. In reality, there are two complete sets, the set the analyst knows about and the actual complete set. The **Actual Complete Set of Codes** is the actual set of codes which is used to perform the

simulation. The **Recognized Set of Codes** is the set of codes which the analyst believes have been used to perform the simulation. By definition, the recognized set is a subset of the actual set. For example, suppose a simulation relied on the output from code A, but the analyst did not recognize the fact that code A was used. Then while the actual set would contain code A while the recognized set would not.

3.3.6.1. Includes every Coded Physical Function used in the Simulation

The Complete Set of Codes should capture every Coded Physical Function that is used in performing the simulation. The Coded Physical Function is the fundamental component of the hierarchy and also of the simulation. While most of the Coded Physical Functions are given in source code, some calculations may be performed elsewhere. Even though these calculations may not be written in a computer language, each of them can still be considered a Coded Physical Function. Therefore any set of them can be considered a Coded Group and also a Computer Code. In other words, the Complete Set of Codes is not simply about capturing the all of the Coded Physical Functions in source code, but capturing every Coded Physical Function used in the simulation.

Identifying calculations outside of source code can be very difficult. This difficulty is most apparent when considering the input to a simulation and the output which results from a simulation. Often the input to a simulation is calculated elsewhere. However, that calculation would be part of the simulation. Even if the input is a measurement, that measurement is still usually the result of some calculation. Similarly, the results of the simulation can be manipulated well after any code execution. Any calculations with those results are usually considered part of the simulation itself. These aspects of the input and output are important to keep in mind when attempting to define the boundaries of the simulation.

3.3.6.2. Fully Described Complete Set of Codes

A **Fully Described Complete Set of Codes**[13] is a Complete Set of Codes, where each code in the set is an Explicit Computer Code and there is some guidance which directs the analyst on how to choose the input. Thus the fully described set includes all of the calculations needed to perform the simulation, prescribes any necessary external selections, and provides a methodology for choosing other input values. If

[13] This concept draws heavily from "Evaluation Methodology" as defined in 10 CFR 50.46, Code of Federal Regulations.

any of these aspects are missing, if there external selections which need to be made or if there is no methodology for choosing input, the complete set is not fully described.

For example, if the code used in Example 3.7 were the only code used in a simulation, then the code would also be the Complete Set of Codes. However, that code would not be fully described unless guidance were given which directed the analyst in the external section of the value of `X` and provided the analyst with guidance on how to choose the inputs `A`, `P`, `rho`, `v`, and `g`.

3.3.7. Scientific Computer Simulation

The components defined above are all of the components which make up a scientific computer simulation. However, notice that the highest level in the hierarchy is the Complete Set of Codes and not the scientific computer simulation itself. This is because each level in the hierarchy is defined as a set of the levels beneath it. However, a scientific computer simulation is not simply the Complete Set of Codes.

A scientific computer simulation is the Complete Set of Codes but it also requires input for the complete set and the resulting output. This need for the input and output is unique to a simulation as the definition for all other components do not require input or output and can be thought of as collections of lower level components. The definition suggested here for a scientific computer simulation (in terms of its components) is as follows:

> A **Scientific Computer Simulation** is a Complete Set of Codes with one complete set of inputs and all resulting outputs.

Figure 3 and Figure 4 are used to demonstrate the difference between a Complete Set of Codes and a Simulation using the simplest possible example of each.

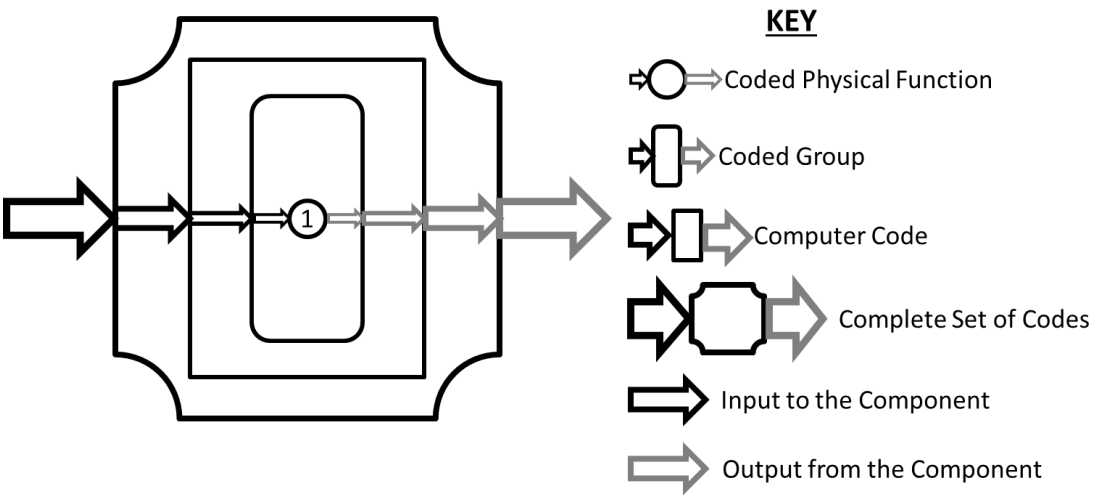


Figure 3: Simplest Complete Set of Codes

Figure 3 is the simplest possible Complete Set of Codes. It is a Complete Set of Codes which contains one Computer Code. That Computer Code contains one Coded Group. Finally, that Coded Group contains one Coded Physical Function. The arrows indicate the pathways of communication between each of the components. The black empty arrows indicate the flow of the input the grey empty arrows indicate the flow of the output. The arrows are empty to signify that no input is actually provided and no output is actually calculated. That is, this is a Complete Set of Codes and not a Scientific Computer Simulation.

For this simple case, the input to the Complete Set of Codes is the input to the Computer Code which is the input to the Coded Group which is the input to the Coded Physical Function. Likewise, the output of the Coded Physical Function is the output of the Coded Group, which is the output of the Computer Code, which is the output of the Complete Set of Codes.

For this Complete Set of Codes to become an actual Scientific Computer Simulation, some input would need to be provided which resulted in some output, as given in Figure 4.

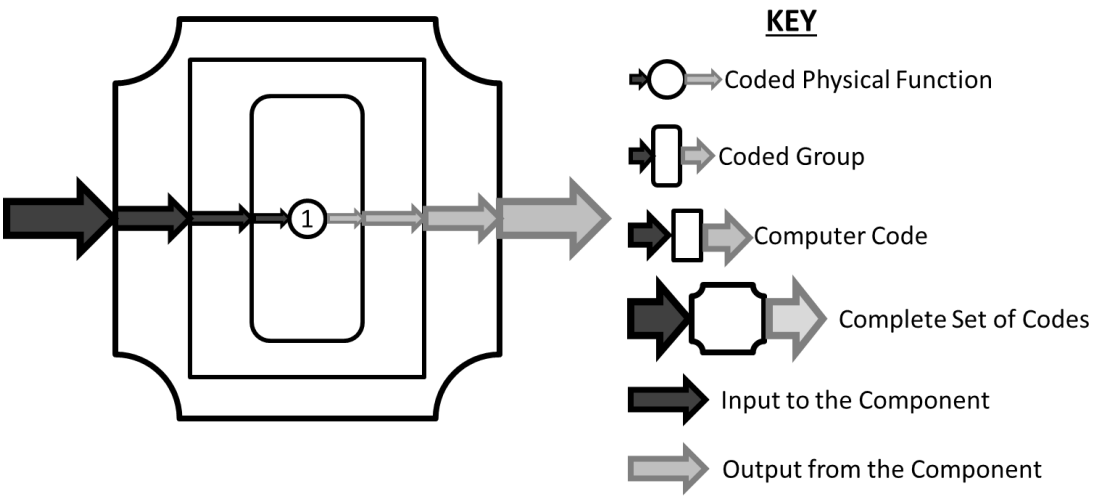


Figure 4: Simplest Scientific Computer Simulation

Figure 4 is the simplest possible Scientific Computer Simulation. It is based on the simplest possible Complete Set of Codes given in Figure 3, but also contains the input to the Complete Set of Codes and the resulting output. The arrows are no longer empty but filled to signify that input is provided and output is calculated. An example of a more complex Scientific Computer Simulation is represented in Figure 5.

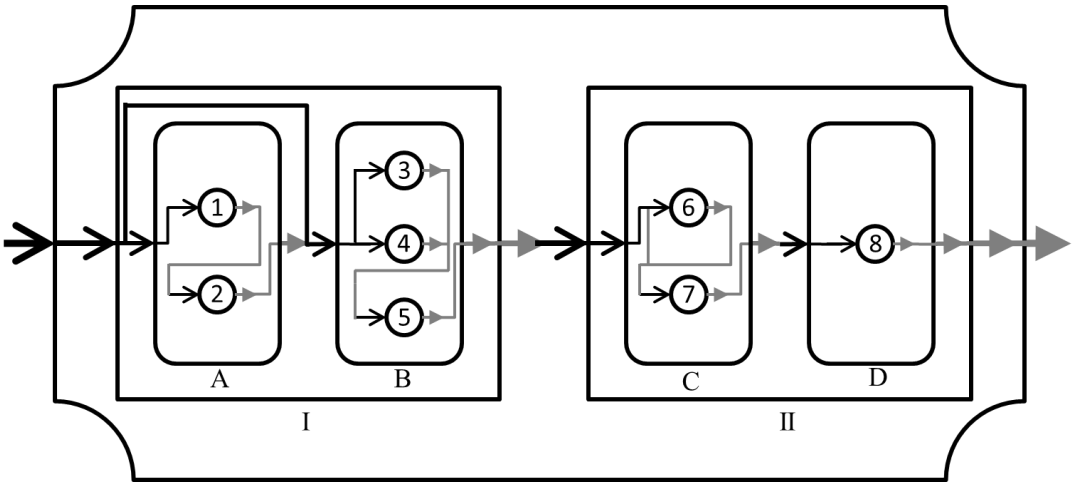


Figure 5: Example Scientific Computer Simulation

This simulation has 8 Coded Physical Functions (1-8), 4 Coded Groups (A-D), and 2 Computer Codes (I and II). This simple example is useful to further demonstrate the concept that the Coded Physical Function is the fundamental component of a simulation. The identical simulation as that above could also have been created using one Computer Code instead of two, as given in Figure 6.

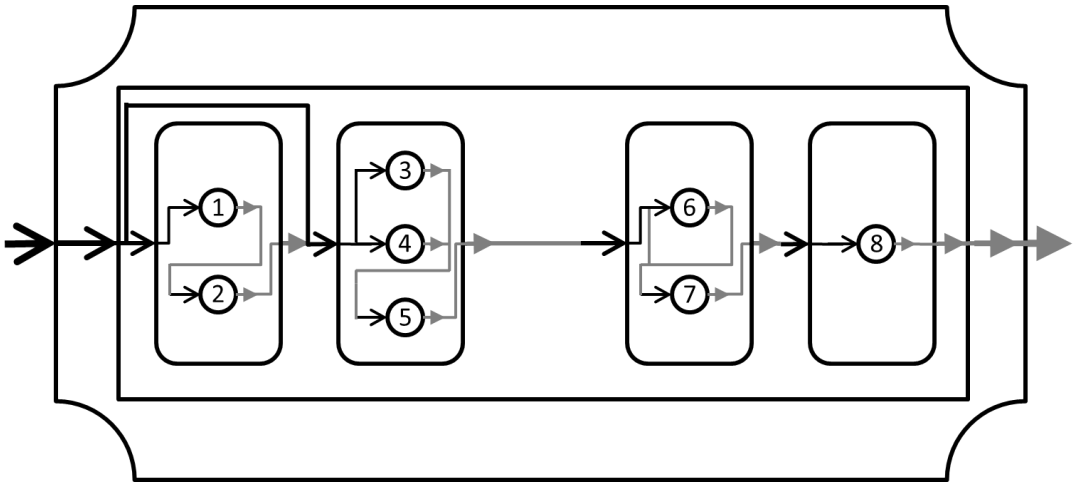


Figure 6: Variation #1 on the Example Scientific Computer Simulation

Notice that the same input provided to both the simulation represented in Figure 5 and Figure 6 would produce the same result. This is because the same Coded Physical Functions are used in the same manner. This can be taken to an extreme by placing all of the Coded Physical Functions into one Coded Group as given in Figure 7.

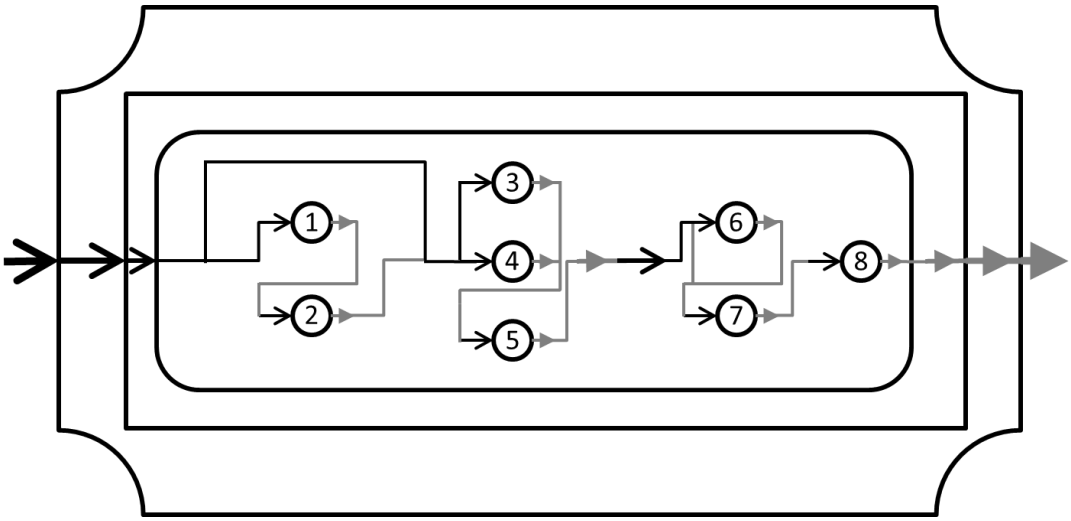


Figure 7: Variation #2 on the Example Scientific Computer Simulation

In Figure 7, the same simulation is given, except the Complete Set of Codes contains only one Computer Code, which contains only one Coded Group, which contains all of the Coded Physical Functions. This is the reason that the Coded Physical Function is the fundamental component of a simulation. Computer Codes and Coded Groups are useful for organizing Coded Physical Functions, but that organization is there to help the analysts understanding. It is the Coded Physical Functions and their relationships with each other which will determine the simulation's results.

3.4. **Details of the Components**

This section uses the Example Scientific Computer Simulation of Figure 5 for discussion, which is given again here for convince.

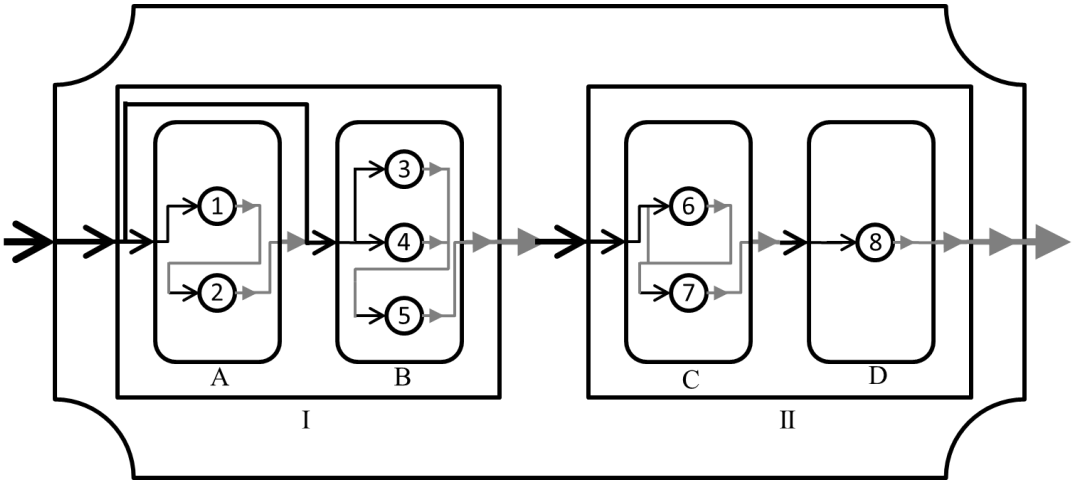


Figure 5: Example Scientific Computer Simulation

Before additional details of the components are given, three additional concepts should be defined in terms of components: parent, siblings, and children. A **Parent Component** is the component one level higher in the hierarchy which contains the component under consideration. Conversely, **Children Components** are all of the components one level lower in the hierarchy which are contained in the component under consideration. The following parental/children relationships exist in the simulation represented by Figure 5:

- Computer Code-I (CC-I) is the parent of Coded Group-A (CG-A) and CG-B (i.e., CG-A and CG-B are children of CC-I)
- CC-II is the parent of CG-C and CG-D (i.e., CG-C and CG-D are children of CC-I)
- CG-A is the parent of Coded Physical Function-1 (CPF-1) and CPF-2 (i.e., CPF-1 and CPF-2 are children of CG-A)
- CG-B is the parent of CPF-3, CPF-4, and CPF-5 (i.e., CPF-3, CPF-4, and CPF-5 are children of CG-B)
- CG-C is the parent of CPF-6 and CPF-7 (i.e., CPF-6 and CPF-7 are children of CG-C)
- CG-D is the parent of CPF-8 (i.e., CPF-8 is a child of CG-D)

Sibling Components are any components which are in the same level as the component under consideration and share the same parents. The following sibling relationships exist in the simulation represented by Figure 5:

- CC-I and CC-II are siblings

- CG-A and CG-B are siblings
- CG-C and CG-D are siblings
- CPF-3, CPF-4, and CPF-5 are siblings
- CPF-6 and CPF-7 are siblings

3.4.1. Input and Output of the Components

Typically, the input for each component must be selected from the input to its parent component and the output from all of its siblings (including itself). The input provided by the analyst is the input to the Complete Set of Codes. If a specific Coded Physical Function requires some of that input, then that input is passed to the Coded Physical Function by way of the Computer Code and Coded Group which contains that function. Likewise, if the output from a specific Coded Physical Function can be passed to the Coded Group and the Computer Code which contains that function, if that output is desired for some purpose. The following input/output relationships exist in the simulation represented by Figure 5:

Table 3-2: Input/Output Relationships

Component	Obtains Input from…	Obtains Output from…
CPF-1	Input to CG-A	Calculates Output
CPF-2	Output from CPF-1	Calculates Output
CPF-3	Input to CG-B	Calculates Output
CPF-4	Input to CG-B	Calculates Output
CPF-5	Output from CPF- 3 and CPF-4	Calculates Output
CPF-6	Input to CG-C and Output from CPF-6	Calculates Output
CPF-7	Output from CPF-6	Calculates Output
CPF-8	Input to CG-D	Calculates Output
CG-A	Input to CC-I	Output from CPF-2
CG-B	Input to CC-I and Output from CG-A	Output from CPF-5
CG-C	Input to CC-II	Output from CPF-7
CG-D	Output from CG-C	Output from CPF-8
CC-I	Input to Complete Set of Codes	Output from CG-B
CC-II	Output from CC-I	Output from CG-D
Complete Set of Codes	Analyst	Output from CC-II

In general, the input to a component must come from the input to its parent component or the output of its sibling components (including itself). Likewise, the output to a component must come from its children components, however for completeness, the following two exceptions to these rules should be kept in mind.

1. The input to a simple term must be selected from the input to its parent Assignment Statement as a simple term cannot obtain output from its siblings.
2. The output from a complex term is a function of the simple terms which make up that complex term.

The actual input to any component can be something as simple as a single numerical value, but is likely to be much more complex as Scientific Computer Simulations often require extensive meshes and other information often contained in large matrices. This input is initially passed to the Complete Set of Codes, where it may be separated and passed separately to multiple Complete Codes, the Coded Groups inside those codes, and the Coded Physical Functions inside those groups. Table 3-4 provides additional details for each component in the hierarchy. The abbreviations given in Table 3-3 are used in Table 3-4.

Table 3-3: Abbreviations for Components

Object	Abbreviation
Scientific Computer Simulation	**SciCS**
Complete Set of Codes	**CSC**
Computer Code	**CC**
Coded Group	**CG**
Coded Physical Function	**CPF**
Assignment Statement	**AS**
Term	**T**
Simple Term	**ST**
Input to Component X	I_X
Output from Component X	O_X
Number of Computer Codes in the Complete Set of Codes	N_{CC}
Number of Coded Groups Codes in a specific Computer Code	N_{CG}
Number of Coded Physical Functions in a specific Coded Group	N_{CPF}
Number of Assignment Statements in a specific Coded Physical Functions	N_{AS}
Number of Terms in a specific Assignment	N_T
Number of Simple Terms in a specific Term	N_{ST}

Each of the columns of the table is described as follows:

- The first column of the table provides the component under consideration (e.g., Complete Set of Codes or Assignment Statement).
- The second column of the table describes how the component under consideration is defined. For example, a Coded Group is defined as a set N of Coded Physical Functions which is expressed in mathematical notation as $\mathbf{CG} \equiv \{\mathbf{CPF}_{1,}\mathbf{CPF}_{2,} \cdots \mathbf{CPF}_{N}\}$.
- The third column of the table describes the input for the component under consideration. The description assumes that the particular component being described is the i[th] component in its parent's group. For example, input for the i[th] Coded Group in a Computer Code can be represented as $\mathrm{I}_{\mathbf{CG},\mathrm{i}} = \{\mathrm{I}_{\mathbf{CC}}, \mathrm{O}_{\mathbf{CG},1}, \mathrm{O}_{\mathbf{CG},2}, \cdots, \mathrm{O}_{\mathbf{CG},\mathrm{N}}\}$. That is, the input for that i[th] Coded Group must either be obtained from the input to the Computer Code which contains the Coded Group $(\mathrm{I}_{\mathbf{CC}})$ or the output from all of the other Coded Groups in the same Computer Code $(\mathrm{O}_{\mathbf{CG},1}, \mathrm{O}_{\mathbf{CG},2}, \cdots, \mathrm{O}_{\mathbf{CG},\mathrm{N}})$, including its own output.
- Finally, the fourth column of the table describes the output from the component under consideration. For example, output from the i[th] Coded Group in a Computer Code can be represented as $\mathrm{O}_{\mathbf{CG},\mathrm{i}} = \{\mathrm{O}_{\mathbf{CPF},1}, \mathrm{O}_{\mathbf{CPF},2}, \cdots, \mathrm{O}_{\mathbf{CPF},\mathrm{N}}\}$. That is, the output from that i[th] Coded Group must be obtained from the output from the N number of Coded Physical Functions in that Coded Group $(\mathrm{O}_{\mathbf{CPF},1}, \mathrm{O}_{\mathbf{CPF},2}, \cdots, \mathrm{O}_{\mathbf{CPF},\mathrm{N}})$.

Table 3-4: Details of the Components in the Hierarchy

Component	Defined as a Set of…	Input for the i^{th} component	Output from the i^{th} component
Complete Set of Codes	$\mathbf{CSC} \equiv \{\mathbf{CC}_1, \mathbf{CC}_2, \cdots \mathbf{CC}_{N_{CC}}\}$	$I_{\mathbf{CSC}} =$ User Specified	$O_{\mathbf{CSC}} = \{O_{\mathbf{CC},1}, O_{\mathbf{CC},2}, \cdots, O_{\mathbf{CC},N_{CC}}\}$
Computer Code	$\mathbf{CC} \equiv \{\mathbf{CG}_1, \mathbf{CG}_2, \cdots \mathbf{CG}_{N_{CG}}\}$	$I_{\mathbf{CC},i} = \{I_{\mathbf{CSC}}, O_{\mathbf{CC},1}, O_{\mathbf{CC},2}, \cdots, O_{\mathbf{CC},N_{CC}}\}$	$O_{\mathbf{CC},i} = \{O_{\mathbf{CG},1}, O_{\mathbf{CG},2}, \cdots, O_{\mathbf{CG},N_{CG}}\}$
Coded Group	$\mathbf{CG} \equiv \{\mathbf{CPF}_1, \mathbf{CPF}_2, \cdots \mathbf{CPF}_{N_{CPF}}\}$	$I_{\mathbf{CG},i} = \{I_{\mathbf{CC}}, O_{\mathbf{CG},1}, O_{\mathbf{CG},2}, \cdots, O_{\mathbf{CG},N_{CG}}\}$	$O_{\mathbf{CG},i} = \{O_{\mathbf{CPF},1}, O_{\mathbf{CPF},2}, \cdots, O_{\mathbf{CPF},N_{CPF}}\}$
Coded Physical Function	$\mathbf{CPF} \equiv \{\mathbf{AS}_1, \mathbf{AS}_2, \cdots \mathbf{AS}_{N_{AS}}\}$	$I_{\mathbf{CPF},i} = \{I_{\mathbf{CG}}, O_{\mathbf{CPF},1}, O_{\mathbf{CPF},2}, \cdots, O_{\mathbf{CPF},N_{CPF}}\}$	$O_{\mathbf{CPF},i} = \{O_{\mathbf{AS},1}, O_{\mathbf{AS},2}, \cdots, O_{\mathbf{AS},N_{AS}}\}$
Assignment Statement	$\mathbf{AS} \equiv \{\mathbf{T}_1, \mathbf{T}_2, \cdots \mathbf{T}_{N_T}\}$	$I_{\mathbf{AS},i} = \{I_{\mathbf{CPF}}, O_{\mathbf{AS},1}, O_{\mathbf{AS},2}, \cdots, O_{\mathbf{AS},N_{AS}}\}$	$O_{\mathbf{AS},i} = \{O_{\mathbf{T},1}, O_{\mathbf{T},2}, \cdots, O_{\mathbf{T},N_T}\}$
Term	$\mathbf{T} \equiv \{\mathbf{ST}_1, \mathbf{ST}_2, \cdots \mathbf{ST}_{N_{ST}}\}$	$I_{\mathbf{T},i} = \{I_{\mathbf{AS}}\}$	$O_{\mathbf{T},i} = f(\mathbf{ST}_1, \mathbf{ST}_2, \cdots \mathbf{ST}_{N_{ST}})$

3.4.2. Input and Output of the Scientific Computer Simulation

Before the input and output of a scientific computer simulation can be defined, a concept of coupled components must be introduced. A **Coupled Component** is any component whose input is obtained from the output of any other component (including itself); further, the coupling between two components extends to their parents. For example, suppose Coded Physical Function A in Computer Code 1 uses the output from Coded Physical Function B in Computer Code 2. Function A is said to be coupled to function B. However, the code group and the Computer Code which contain Function A are also coupled with the Coded Physical Function B, the code group which contains function B, and the Computer Code which contains function B.

This concept of coupled components provides the foundation for the definition for the input to the simulation. The **Input to the Scientific Computer Simulation** is any input to a Coded Physical Function which is not the output of any Coded Physical Function in the Complete Set of Codes. That is, the input to the simulation is every input value that is not calculated by the simulation. Each of these inputs should be specified in the simulation's input, including values hardwired into the source code.

The **Output from the Scientific Computer Simulation** is the output from every Coded Physical Function in the Complete Set of Codes. In other words, any quantity that is calculated in the simulation be thought of as the output from the simulation as it must be an output of one of the Coded Physical Functions in the simulation. However, usually only a subset of all the calculated quantities are thought of as output from the simulation, this subset is the selected output. The **Selected Output from the Scientific Computer Simulation** is any output from any Coded Physical Function in the Complete Set of Codes which is used for some purpose by an analyst. It is not necessarily the output from every Coded Physical Function in the simulation, but only that set of Coded Physical Functions whose output is desired. In other words, there certain outputs from Coded Physical Functions, Coded Groups, and Computer Codes which are not often considered part of the output from the Complete Set of Codes (i.e., the simulation). Logically, these values must still be output; they are just not the selected output as they are not of interest for some reason or another. Notice that in Figure 5, Figure 6, and Figure 7, only the grey arrows designate only the selected output.

These definitions for input and output allow a clear distinction to be drawn between different simulations. Simulations with different input values are different simulations. Also, simulations with different Coded Physical Functions are also different simulations. While this may seem obvious, it leads a surprising conclusion

that any other calculations which use the results of a simulation are not part of that original simulation.

Suppose that a simulation is performed to predict the temperature along the surface of a plate. Some input is chosen and the output from the Complete Set of Codes is obtained. That input, those Complete Set of Codes, and the resulting output is a simulation which will be called simulation A for this example. Further suppose that an analyst wanted to now average the calculated temperature across the surface of the plate to obtain an average surface temperature. While that calculation would rely on the results of simulation A, it would not be a part of simulation A itself. The calculation of the average surface temperature was not one of the Coded Physical Functions defined in the Complete Set of Codes, but was an additional calculation performed by the analyst. The analyst could consider the input, the Complete Set of Codes plus the average calculation, and the resulting output plus the calculated average surface temperature a different simulation, say simulation B. But using the definition of a scientific computer simulation, it would not be the same simulation as the original.

3.5. Conclusions on the Hierarchy of Scientific Computer Simulation Components

The Hierarchy of Scientific Computer Simulation Components is not the only way to understand the boundaries of a simulation, but it is one useful way. It is a formalization of the hierarchy unofficially used by many engineers who perform simulations, as its structure is aligned with some of the higher level languages those engineers use. The lower levels of the hierarchy are directly related to components in source code, where the other levels are simply defined as collections of those components. There are two very important aspects to the hierarchy.

The first aspect is the concept of the Coded Physical Function. While this level of the hierarchy is made up of lower levels, it is seen as the fundamental unit of all scientific computer simulations. Most simulations and most of science are focused on these functions, their development, their behavior, and if they can be trusted to predict some physical phenomenon. In one sense, a scientific computer simulation can be understood as a collection of Coded Physical Functions with some given input and the resulting output. The levels above the Coded Physical Function (Coded Group and Computer Code) provide a convenient way to arrange and organize the Coded Physical Functions into sets, allowing analysts to considering only a certain set of Coded Physical Functions instead of all of them at once. However, identical simulation can be represented by any number of Coded Groups and Computer Codes, as long the same Coded Physical Functions are used in the same manner. Thus, while Coded Groups and Computer Codes are important in understanding the arrangement of the simulation, it is the Coded Physical Functions which actually "make-up" the simulation.

The second aspect is that the hierarchy provides one way in which a simulation can be thought of by the analyst. That is, any simulation could be expressed and understood in terms of the hierarchy. While that expression could be translated into the source code, it does not need to be in order to be useful. It is the goal of a higher level language to express the problem in an ordered and logical manner so that a computer can perform the simulation and it is the goal of the hierarchy to express the problem in an order and logical manner so that analysts can understand the simulation.

The hierarchy is a very important aspect of the maturity framework developed in this dissertation. The next step in creating this framework is to use the hierarchy and develop the structure of the framework and this step is performed in Chapter 4.

4. Developing the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

This chapter focuses on developing the structure of the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations. This structure is based on the components defined in the Hierarchy of Scientific Computer Simulation Components and Maturity Theory.

The first section focuses on the objectives of current maturity assessment frameworks. These objectives are important because the proposed Theoretical/Logical Framework is created by changing these objectives. These changes results in a framework which contains all possible maturity assessment sets needed to answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). The second section uses a thought experiment to introduce the concept of an Ideal Simulation. This concept is expanded in the third section and a set of necessary and sufficient conditions are generated from the Hierarchy which must be true for an Ideal Simulation. In the fourth section, these necessary and sufficient conditions are applied to a real-world (non-ideal) simulation. In the fifth section, these conditions are used as the structure for the Theoretical/Logical Framework. Finally, the sixth section concludes and summarizes the contributions of this chapter.

4.1. Concentration on Framework Development

The first stage of scientific computer simulation review is the development of the maturity assessment framework. This framework is then used in stage 2 to determine the maturity requirements for the intended purpose of the simulation, in stage 3 to assess the maturity of the specific simulation, and in stage 4 to make the judgment as to whether or not that requirements have been met by the specific simulation. While each stage has its own particular challenges, the developed framework impacts all stages and is therefore a major focus of simulation review as well as this dissertation.

4.1.1. Analysis of the Current Practical Frameworks

Two formal frameworks which are focused on scientific computer simulations are PCMM (Section 1.4.2.4) (Oberkampf, *et al.* 2007) and NASA's framework (Section 1.4.2.5) (NASA[1] 2008). Both of these frameworks are practical frameworks. A **Practical Maturity Assessment Framework** is a formal framework developed with

the intent to be used to assess the maturity of real-world scientific computer simulations. Practical frameworks, such as PCMM and NASA's framework were generated with significant effort by multiple experts in various fields of study over several years. The two objectives of practical frameworks are summarized as follows:

(1) The practical framework should contain all of the important maturity assessment sets which could be used to help answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*)

(2) The practical framework must be a cost effective use of resources.

In other words, not only does the framework have to consider all of the important attributes, but it must do so economically, else the framework would not be used and therefore not practical.

4.1.2. Developing a Theoretical/Logical Framework

One of the most enlightening documents about developing a framework was published by NASA (NASA 2009). In this document, NASA detailed the steps behind the development of their framework, the struggles, the disagreements, and how resolution was mostly achieved. It should not be surprising that developing a practical framework which satisfies both of the objectives above takes cooperation among a large number of experts in many fields of study as well as a significant time and resource investment. While frameworks are being further developed in this manner, any practical framework developed by an individual would likely fall short of these current frameworks. Such a framework would be susceptible to the limits of that individual's knowledge and particular biases. For this reason, the framework suggested in this dissertation is not considered a practical framework.

Instead of developing a practical framework, this dissertation focuses on developing a theoretical/logical framework. The difference between the theoretical/logical framework and a practical framework can best be seen through the objectives of the proposed framework:

(1) The framework should be structured to contain all of the possible maturity assessments sets which could be used to help answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

(2) The framework should be structured such that all of assumptions made when answering trusting a simulation (i.e., answering the Ultimate Question in the affirmative) have been taken into account.

Objective (1) makes the framework theoretical as the number of maturity assessment sets which could be used to help answer the Ultimate Question may be infinite. Objective (2) makes the framework logical as some method is required to ensure that all possible assumptions have been accounted for and no assumption has been ignored. The development of the specific Theoretical/Logical Framework proposed here is focused on capturing and organizing all of the maturity assessments sets which are necessary and sufficient in answering the Ultimate Question. It is possible that there are an infinite number of such sets which the framework would need to contain. Thus, the framework must not only be able to organize the potentially infinite number of maturity assessment sets, but it must do so in a way which demonstrates some sufficiency of all sets.

4.1.2.1. Necessary and Sufficient

Because the Theoretical/Logical Framework makes substantial use of the concepts of necessary and sufficient, a brief review of those concepts is provided here. A **Necessary Condition** is a circumstance in whose absence a specific event could not occur (Copi 1986). Thus, a necessary condition for being a bachelor is being male, as only males are defined as bachelors. However, simply being male is not sufficient for being a bachelor. A **Sufficient Condition** is a circumstance in whose presence a specific event must occur (Copi 1986). Thus, being a male and also being unmarried are sufficient conditions for being a bachelor, as a bachelor is defined as an unmarried male.

4.1.2.2. Necessary and Sufficient Maturity Assessment Sets

Using the above definitions, the concepts of necessary and sufficient can be applied to maturity attributes and maturity assessment sets. A **Necessary Maturity Attribute** is an attribute which must be assessed or else the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*) cannot be answered. For example, assessing the maturity of a simulation's peer review is necessary, as it is need to answer the Ultimate Question. However, assessing this single attribute is not sufficient and does not completely address the Ultimate Question on its own. **Sufficient Maturity Attributes** are the set of attributes which must be assed to in order to fully answer the Ultimate Question. The development of the

Theoretical/Logical Framework will focus on developing and organizing both necessary and sufficient maturity attributes and maturity assessment sets.

It is difficult (and likely not possible) to prove that any set of maturity attributes considered were sufficient for answering the Ultimate Question. Demonstrating that such a set would be sufficient would require complete and perfect knowledge of the simulation and the physical phenomena it predicted. While this level of knowledge does not exist for real-world simulations, it is useful to imagine a scenario where it did exist as given in the following thought experiment.

4.2. Thought Experiment: Simulating an Asteroid Crashing into Earth

The following thought experiment demonstrates a concept about simulations, resources, and trust: given two simulations of the same scenario, the simulation which required more resources (computational power, manpower, time, etc...) is *generally* considered the more trustworthy of the two. The word *generally* is stressed as this is far from being an absolute truth. Additional resources allow for the use of higher fidelity models, more testing, more iterations, more feedback from experts, etc…

4.2.1. Thought Experiment

Suppose that Bob works for NASA and discovers that an asteroid will crash into Earth sometime in the next few years. Bob performs a simulation to determine the results of the impact. He then shows his discovery and the results of his analysis to his boss. Bob's boss passes the information to his boss and so on until they are seen by the NASA Administrator.

The NASA Administrator tells NASA's Chief Scientist to perform a better simulation and provides all the resources of NASA to do so. So the Chief Scientist gathers a team of experts from NASA and performs another simulation using all the resources that NASA has at its disposal. After the simulation is completed, the Chief Scientist presents the results to the President of the United States.

The President tells NASA's Chief Scientist to perform a better simulation and provides all the resources of US to do so. So the Chief Scientist gathers a large group of experts from across the US and performs another simulation using all the resources that the US has at its disposal. After the simulation is completed, the Chief Scientist presents the results to the United Nations.

The Secretary-General of the United Nations tells NASA's Chief Scientist to perform an even better simulation and provides all the resources of the world to do so. So the Chief Scientist gathers an even large group of experts from across the world and performs another simulation using all the resources that the world has at its disposal. After the simulation is completed, the Chief Scientist presents the results to the rest of the world.

After the Chief Scientist has presented the results, two very interesting things happen. First, a man steps out of a blue box, walks up, and hands the Chief Scientist results from an even better simulation which was performed at some point in the future.

Finally, a dove descends from the heavens carrying the results of an even better simulation than the simulation from the future.

4.2.2. Five Basic Types of Simulations

This thought experiment has five different simulations, each more trusted than the one before. Each simulation is predicting the exact same event for the exact same purpose. The only difference between the simulations is how much resources are required to perform it. Again, this is not to say more resources *necessarily* make a more trustworthy simulation, only that they *probably do*. The resource levels for each of the simulations are given below in increasing order:

(1) **Personal Simulation** – a simulation performed by an individual. Personal simulations are limited by what a single person can do with the resources available to them. This is highly dependent on the individual and the available resources. The first simulation Bob performed was a Personal Simulation.

(2) **Organizational Simulation** – a simulation performed by many individuals or an organization. Organizational simulations take advantage of the expertise and resources available to an organization. This will likely result in a more extensive simulation which uses much more man power. The simulation performed by NASA was an Organizational Simulation.

(3) **National Simulation** – a simulation performed by many organizations. National simulations take advantage of expertise and resources of multiple organizations up to and including that of entire nations. The Manhattan project is a good example of a national simulation as many organizations were needed to simulate an atomic bomb. The simulations performed by the United States and the United Nations were National Simulations.

(4) **Best Humanly Possible Simulation** – a simulation performed using all the resources of humanity (including all future resources). In other words, suppose every human being decided that simulating one event was the most important thing in their life and everyone worked towards this goal until the end of the human species. While such a simulation is obviously fictional, it is helpful because it represents an upper limit as no simulation could possibly be better than this one. The simulation results from the man in the blue box were the results from the Best Humanly Possible Simulation at any time in the future.

(5) **Ideal Simulation** – a simulation which is performed perfectly and whose results are true. It is important to remember that even the Best Humanly Possible Simulation is likely to fall short of providing absolute truth. An Ideal Simulation would require complete and perfect knowledge. Humanity does not currently have this level of knowledge and it is likely that such a level of knowledge is not obtained during our existence. Such a simulation could only be performed by a god[14] who does have such complete and perfect knowledge and can express such knowledge perfectly. The simulation results from the dove had the results were the results from an Ideal Simulation.

While each of these different types of simulations can be further developed, it is the fictional simulations of Best Humanly Possible and Ideal which are of most interest. The Best Humanly Possible Simulation represents the very best achievable simulation. In other words, (by the definition of maturity) it is the most mature simulation. Such a conceptual simulation is therefore very useful in defining the highest maturity level of a specific attribute. The **Highest Maturity Level of Any Attribute** would be the level achieved if all of the resources of humanity were focused on performing a single simulation until the end of humanity. It is important to recognize that, by definition, this level of maturity could be achieved but no higher level could be achieved, thus making it the highest.

While the Best Humanly Possible Simulation is vital in developing the highest maturity level, it is the Ideal Simulation which provides a logical method for determining a set of maturity attributes which are sufficient for answering the Ultimate Question.

[14] This is in no way meant to be a philosophical or theological argument for or against a deity. The word "god" was chosen because it is the closest word which represents everything that would be required for an Ideal Simulation: absolute knowledge, perfection, and infinite ability.

4.3. **Consideration of an Ideal Scientific Computer Simulation**

The answer to Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*) is generally a function of many things, including the personal assessment of the decision maker. The goal of maturity assessment is to provide the decision maker with clearer and more objective information, but the answer will still likely contain a great deal of subjectivity. However, when considering an Ideal Simulation, there is no subjectivity in the answer. By definition, the answer to the Ultimate Question for any Ideal Simulation is "yes".

4.3.1. Focus on the Fundamental Question of Scientific Computer Simulation

For an Ideal Simulation, the answer to the Ultimate Question is "yes". But answering the Fundamental Questions (*How trustworthy are the results of the specific scientific computer simulation?* & *How trustworthy do the results of a scientific computer simulation need to be for the intended purpose?*) cannot be given as a simple "yes" or "no". However, what can be said is that the results of an Ideal Simulation are true[15]. In other words, no matter what the requirements are for the intended purpose, the results of an Ideal Simulation can be trusted. Because this statement relates to both of the Fundamental Questions, it is called the Fundamental Statement. The **Fundamental Statement of Scientific Computer Simulations** is: *The results of the scientific computer simulation are true*. It should always be remembered that this statement is likely only true for Ideal Simulations.

By itself, the Fundamental Statement is rather broad statement and not very useful. However, the Fundamental Statement could be expressed as a set of more narrowly defined necessary and sufficient statements. That is, each statement in the set would be a necessary condition for the Fundamental Statement to be true and the complete set of all the statements would be a sufficient condition for the Fundamental Statement to be true. This set of necessary and sufficient statements could be very useful in better understanding the Fundamental Statement and the Ultimate Question.

[15] The following discussion uses the concept of "truth". For physical phenomena, there are generally two points of view on this topic: truth is deterministic or truth is non-deterministic. While the author has his own opinions on the matter, the discussion is applicable to either point of view.

4.3.2. Applying the Hierarchy of Scientific Computer Simulation Components

The goal of this section is to deconstruct the Fundamental Statement (*The results of a Scientific Computer Simulation are true*) into a set of more narrowly focused necessary and sufficient statements. A table is used to help track the deconstruction process with the following three columns:

- a unique identification number (**ID**) for each statement
- the statement itself
- the component of interest for each statement from the Hierarchy of Scientific Computer Simulation Components.

As the Fundamental Statement is the originating statement, it is identified as statement **0** and its component of interest is a Scientific Computer Simulation (**SciCS**). This information is captured in Table 4-1.

Table 4-1: Fundamental Statement of an Ideal Scientific Computer Simulation

ID	Statement	Component of Interest
0	The results of the Scientific Computer Simulation are true.	**SciCS**

There are many ways to generate a set of necessary and sufficient statements from the Fundamental Statement. A useful deconstruction of statement **0** is achieved by applying the definition of a Scientific Computer Simulation, as defined in the hierarchy. This deconstruction is given in the following sub-section.

4.3.2.1. Applying the Definition of a Scientific Computer Simulation

As defined in the hierarchy, a Scientific Computer Simulation (**SciSC**) is the set of the Complete Set of Codes (**CSC**) needed to perform the simulation as well as the given input and resulting output. This definition has three aspects which are very important:

- The Scientific Computer Simulation contains not only the Complete Set of Codes which are used to perform the simulation, but must also contain some selected Input to the Complete Set of Codes ($\mathbf{I_{CSC}}$) and the resulting Output from the Complete Set of Codes ($\mathbf{O_{CSC}}$). If either the inputs or the outputs are missing, the simulation is incomplete.
- The input to the Complete Set of Codes must be selected by the analyst.
- Generally, only a subset of the entire output form the Complete Set of Codes is of interest and therefore the analyst must make this choice correctly (e.g.,

the analyst should not accidently choose pressure when she meant to choose temperature).

These aspects of a Complete Set of Codes are summarized in Figure 8.

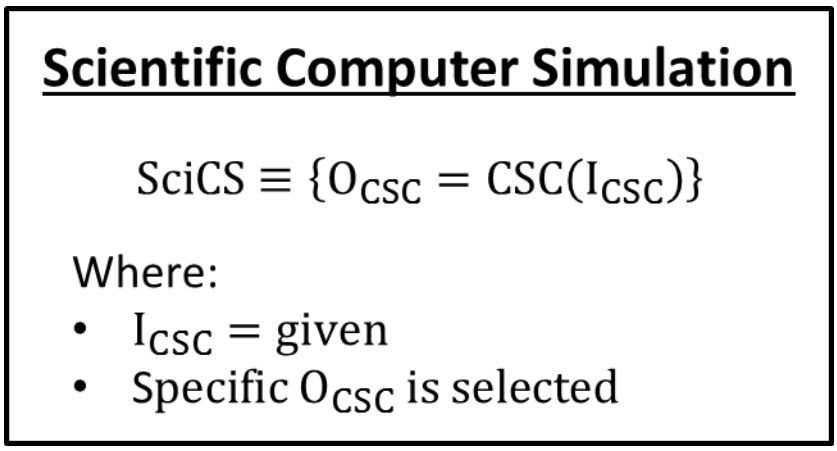


Figure 8: Important Aspects of a Scientific Computer Simulation

Using these aspects of a Scientific Computer Simulation, the following three statements can be generated which are both necessary and sufficient conditions for the Fundamental Statement (*The results of the scientific computer simulation are true*):

- The Input provided for the Complete Set of Codes is correct.
- The Complete Set of Codes calculates the correct output for given input.
- The specific output selected from all of the output from the Complete Set of Codes has been selected correctly.

That is, if any of the above statements are false, then the Fundamental Statement must also be false (i.e., the statements are necessary conditions). However, if all of the statements are true, then the Fundamental Statement must also be true (i.e., the set of statements is a sufficient condition). As these statements are the first deconstruction using the hierarchy they are identified as statement **1.1**, **1.2**, and **1.3**. The component of interest for each statement is the Complete Set of Codes. This information is captured in Table 4-2, which is the first deconstruction of the Fundamental Statement given in Table 4-1.

Table 4-2: Necessary and Sufficient Statements from the 1st Deconstruction of the Fundamental Statement

ID	Statement	Component of Interest
1.1	The Input provided for the Complete Set of Codes is correct.	**CSC**
1.2	The Complete Set of Codes calculates the correct output for given input.	**CSC**
1.3	The specific output selected from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

An additional and useful deconstruction of statement **1.2** is achieved by applying the definition of a Complete Set of Codes, as defined in the Hierarchy of Scientific Computer Simulation Components. This deconstruction is given in the following sub-section.

4.3.2.2. Applying the Definition of a Complete Set of Codes

As defined in the hierarchy, a Complete Set of Codes ($\mathbf{CSC}$) is the set of all Computer Codes ($\mathbf{CC}$) needed to perform the simulation. This definition has three aspects which are very important for each Computer Code in the Complete Set of Codes:

- The Input for each Computer Code ($\mathbf{I_{CC}}$) must be selected [16]from the Input for the Complete Set of Codes ($\mathbf{I_{CSC}}$) and the output from all of the Computer Codes in the complete set.
- The Output from each Computer Code ($\mathbf{O_{CC}}$) is the result of the Computer Code's operation on a given input.
- The Output from the Complete Set of Codes ($\mathbf{O_{CSC}}$) must be selected from the outputs of the Computer Codes in the complete set.

These aspects of a Complete Set of Codes are summarized in Figure 9.

[16] The input for the Complete Set of Codes is *provided* from a source external to the simulation, but the input for the Computer Code (and all subsequent lower levels) is not provided but *selected* from some a source internal to the simulation.

Complete Set of Codes

$$CSC \equiv \{CC_1, CC_2, \cdots CC_N\}$$

Where:

- $I_{CC,i} = \{I_{CSC}, O_{CC,1}, O_{CC,2}, \cdots, O_{CC,N}\}$
- $O_{CC,i} = CC_i(I_{CC,i})$
- $O_{CSC} = \{O_{CC,1}, O_{CC,2}, \cdots, O_{CC,N}\}$

Figure 9: Important Aspects of a Complete Set of Codes

Using these aspects of a Complete Set of Codes, the following three statements can be generated which are both necessary and sufficient conditions for statement **1.2** (*The Complete Set of Codes calculates the correct output for given input*):

- The input selected for each Computer Code is correctly selected from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.
- Each Computer Code calculates the correct output for given input.
- The output from the Complete Set of Codes is correctly selected from the output from the Computer Codes in the Complete Set of Codes.

That is, if any of the above statements are false, then statement **1.2** must also be false and if all of the statements are true, then statement **1.2** must also be true. As these statements are the second deconstruction using the hierarchy they are identified as statement **2.1**, **2.2**, and **2.3**. The component of interest for statements **2.1** and **2.2** is the Computer Code and for statement **2.3** is Complete Set of Codes. This information is captured in Table 4-3, which is the second deconstruction of the Fundamental Statement (*The results of the scientific computer simulation are true*) originally given in Table 4-1.

Table 4-3: Necessary and Sufficient Statements from the 2nd Deconstruction of the Fundamental Statement

#	Statement	Component of Interest
1.1	The input provided for the Complete Set of Codes is correct.	**CSC**
2.1	The input selected for each Computer Code is correctly selected from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
2.2	Each Computer Code calculates the correct output for given input.	**CC**
2.3	The output from the Complete Set of Codes is correctly selected from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
1.3	The specific output selected from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

As a reminder, the input to a Computer Code is either from the input to the Complete Set of Codes or the output from other Computer Codes. Statement **1.1** ensures that any input from the Complete Set of Codes is correct and Statement **2.2** ensures that any input which is the output from a Computer Code is also correct. Thus, these statements ensure that the input provided to each Computer Code is correct.

An additional and useful deconstruction of statement **2.2** is achieved by applying the definition of a Computer Code, as defined in the Hierarchy of Scientific Computer Simulation Components. This deconstruction is given in the following sub-section.

4.3.2.3. Applying the Definition of a Computer Code

As defined in the hierarchy, a Computer Code ($\mathbf{CC}$) is a chosen set of Coded Groups ($\mathbf{CG}$). This definition has three aspects which are very important for each Coded Group in the specified Computer Code under consideration:

- The Input for each Coded Group ($\mathbf{I_{CG}}$) must be selected from the Input to the Computer Code ($\mathbf{I_{CC}}$) and the output from the all of the Coded Groups in the code
- The Output from each Coded Group ($\mathbf{O_{CG}}$) is the result of the Coded Group's operation on a given input.
- The Output from the Computer Code ($\mathbf{O_{CC}}$) must be selected from the outputs of the Coded Groups in the Computer Code.

These aspects of a Complete Set of Codes are summarized in Figure 10.

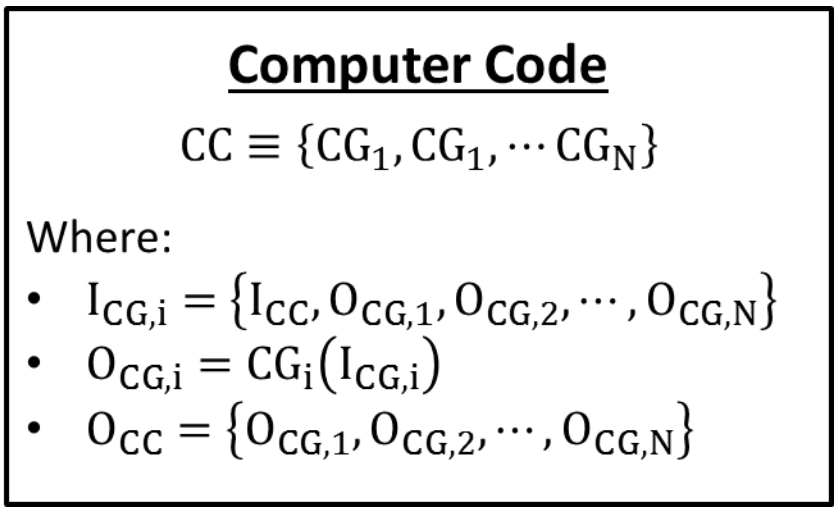


Figure 10: Important Aspects of a Computer Code

Using these aspects of a Computer Code, the following three statements can be generated which are both necessary and sufficient conditions for statement **2.2** (*Each Computer Code calculates the correct output for given input*):

- The input selected for each Coded Group is correctly selected from the input to the Computer Code and the output from the Coded Groups in the Computer Code.
- Each Coded Group calculates the correct output for given input.
- The output from the Computer Code is correctly selected from the output from the Coded Groups in the Computer Code.

That is, if any of the above statements are false, then statement **2.2** must also be false and if all of the statements are true, then statement **2.2** must also be true. As these statements are the third deconstruction using the hierarchy they are identified as statement **3.1**, **3.2**, and **3.3**. The component of interest for statements **3.1** and **3.2** Coded Group and for statement **3.3** is the Computer Code. This information is captured in Table 4-4, which is the third deconstruction of the Fundamental Statement (*The results of the scientific computer simulation are true*) originally given in Table 4-1.

Table 4-4: Necessary and Sufficient Statements from the 3rd Deconstruction of the Fundamental Statement

#	Statement	Component of Interest
1.1	The input provided for the Complete Set of Codes is correct.	**CSC**
2.1	The input selected for each Computer Code is correctly selected from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
3.1	The input selected for each Coded Group is correctly selected from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CM**
3.2	Each Coded Group calculates the correct output for given input.	**CM**
3.3	The output from the Computer Code is correctly selected from the output from the Coded Groups in the Computer Code.	**CC**
2.3	The output from the Complete Set of Codes is correctly selected from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
1.3	The specific output selected from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

Notice that the input to a Code Group is either from the input to the Complete Set of Codes (through the input to the Computer Code which contains the or the output from other Computer Codes. Statement **1.1** ensures that any input from the Complete Set of Codes is correct and Statement **2.2** ensures that any input provided to the Computer Code which is the output from a Computer Code is also correct. Thus, these statements ensure that the input provided to each Computer Code is correct.

An additional and useful deconstruction of statement **3.2** is achieved by applying the definition of a Coded Group, as defined in the Hierarchy of Scientific Computer Simulation Components. This deconstruction is given in the following sub-section.

4.3.2.4. Applying the Definition of a Coded Group

As defined in the hierarchy, a Coded Group (**CG**) is a chosen set of Coded Physical Functions (**CPF**). This definition has three aspects which are very important for each Coded Physical Function in the specified coded model under consideration:

- The Input for each Coded Physical Function ($\mathbf{I_{CPF}}$) must be selected from the Input to the Coded Group ($\mathbf{I_{CG}}$) and the output from the all of the Coded Physical Functions in the Coded Group
- The Output from each Coded Physical Function ($\mathbf{O_{CPF}}$) is the result of the Coded Physical Function's operation on a given input.
- The Output from the Coded Group ($\mathbf{O_{CG}}$) must be selected from the outputs of the Coded Physical Functions in the Coded Group.

These aspects of a Complete Set of Codes are summarized in Figure 10.

Coded Group

$$CG \equiv \{CPF_1, CPF_1, \cdots CPF_N\}$$

Where:

- $I_{CPF,i} = \{I_{CG}, O_{CPF,1}, O_{CPF,2}, \cdots, O_{CPF,N}\}$
- $O_{CPF,i} = CPF_i(I_{CPF,i})$
- $O_{CG} = \{O_{CPF,1}, O_{CPF,2}, \cdots, O_{CPF,N}\}$

Figure 11: Important Aspects of a Coded Group

Using these aspects of a Coded Group, the following three statements can be generated which are both necessary and sufficient conditions for statement **3.2** (*Each Coded Group calculates the correct output for given input*):

- The input selected for each Coded Physical Function is correctly selected from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.
- Each Coded Physical Function calculates the correct output for given input.
- The output from the Coded Group is correctly selected from the output from the Coded Physical Functions in the Computer Code.

That is, if any of the above statements are false, then statement **3.2** must also be false and if all of the statements are true, then statement **3.2** must also be true. As these statements are the fourth deconstruction using the hierarchy they are identified as statement **4.1**, **4.2**, and **4.3**. The component of interest for statements **4.1** and **4.2** is Coded Physical Functions and for statement **4.3** is the Coded Group. This information is captured in Table 4-5, which is the fourth deconstruction of the

Fundamental Statement (*The results of the scientific computer simulation are true*) originally given in Table 4-1.

Table 4-5: Necessary and Sufficient Statements from the 4th Deconstruction of the Fundamental Statement

#	Statement	Component of Interest
1.1	The input provided for the Complete Set of Codes is correct.	**CSC**
2.1	The input selected for each Computer Code is correctly selected from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
3.1	The input selected for each Coded Group is correctly selected from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CG**
4.1	The input selected for each Coded Physical Function is correctly selected from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.	**CPF**
4.2	Each Coded Physical Function calculates the correct output for given input.	**CPF**
4.3	The output from the Coded Group is correctly selected from the output from the Coded Physical Functions in the Computer Code.	**CG**
3.3	The output from the Computer Code is correctly selected from the output from the Coded Groups in the Computer Code.	**CC**
2.3	The output from the Complete Set of Codes is correctly selected from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
1.3	The specific output selected from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

While an additional deconstruction of statement **4.2** is achievable by applying the definition of a Coded Physical Function, as defined in the Hierarchy of Scientific Computer Simulation Components, this deconstruction does not seem useful and is therefore not performed.

4.4. Consideration of a Real-World Scientific Computer Simulation

The statements given in Table 4-5 are necessary and sufficient conditions for the Fundamental Statement (*The results of a Scientific Computer Simulation are true*). That is, if any of the nine statements are false, then Fundamental Statement must also be false and if all nine of the statements are true, then Fundamental Statement must also be true. The truth of these statements is impacted both by the Complete Set Codes (e.g., the given source code) and the input chosen for a particular simulation. Thus, even if the Complete Set of Codes is perfect, the Fundamental Statement may not be true if the analyst has not chosen the input correctly. For an Ideal Simulation, all of the statements are true by definition, but this is not the case for a real-world simulation. However, the statements do still have a significant value.

Suppose an analyst were given a simulation and wanted to answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). If that analyst could prove the nine statements in the above table were true for that simulation, then the answer to the Ultimate Question would be "yes". While it seems highly unlikely that someone could actually prove one of the statements true, much less all nine, decision makers are consistently answering the Ultimate Question in the affirmative.

While the following statement is true: "if the nine statements are true then the answer to the Ultimate Question is 'yes'", its converse is not necessarily true "if a decision maker answers 'yes' to the Ultimate Question then the nine statements are true". However, when a decision maker does answer "Yes" to the Ultimate Question, he or she is assuming that the Fundamental Statement and the nine necessary and sufficient statements in Table 4-5 are true or at least true enough for the intended purpose of the specific simulation.

4.4.1. Fundamental Assumption of Scientific Computer Simulations

When a decision maker answers "yes" to the Ultimate Question, that decision maker is assuming that the results of the simulation are true, or at least true enough for the intended use. That is, the decision maker is assuming the Fundamental Statement (*The results of the scientific computer simulation are true*) can be assumed for that simulation; this is the fundamental assumption. The **Fundamental Assumption of Scientific Computer Simulations** is: *The assumption that the results of the scientific computer simulation are true (or true enough).* That is, it is the assumption made when a decision maker answers "yes" to the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

The Fundamental Assumption is considered "fundamental" for three reasons. First, the assumption's similarity to the Fundamental Statement, second, the assumption's association with the Fundamental Questions (*How trustworthy are the results of the specific scientific computer simulation?* & *How trustworthy do the results of a scientific computer simulation need to be for the intended purpose*), and third, the assumption itself is fundamental to every use of a scientific computer simulation.

The Fundamental Assumption is considered "an assumption" because no decision maker has perfect knowledge. While every simulation is likely to have some amount of supporting evidence, that evidence will likely always fall short of absolute proof. **Supporting Evidence** is any evidence which supports the conclusion that the results of a scientific computer simulation are true (i.e., any evidence which supports the Fundamental Assumption). However extensive that evidence may be, it is usually very far from proving that the assumption can be trusted beyond any doubt. While decision makers will require different amounts of evidence based on the potential consequences of trusting a simulation, the act of trusting a simulation is always done without perfect and complete knowledge of the situation and is therefore still an assumption.

4.4.2. Essential Assumptions of Scientific Computer Simulations

Remember that the set of nine statements given in Table 4-5 form a necessary and sufficient set of conditions for the Fundamental Statement (*The results of the scientific computer simulation are true*). Similarly, they also form a necessary and sufficient set of conditions for the Fundamental Assumption, as the Fundamental Assumption is merely the assumption that the Fundamental Statement is true. Therefore, these nine statements become nine assumptions which are essential in answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?).* The **Essential Assumptions of Scientific Computer Simulations** are the set of assumptions which are both necessary and sufficient to the Fundamental Assumption. These assumptions, like the fundamental assumption, are assumed anytime the results of a simulation are trusted (whether the analyst is aware of it or not).

The statements given in Table 4-5 are turned into the Essential Assumptions and are given in Table 4-6. The component of interest remains the same, but the statement itself slightly changes to account for the phrase "or true enough" in the Fundamental Assumption. Likewise the identification number of each statement is simplified and a modifier is added to indicate that the statements are now Equivalent Assumptions (**EA**).

Table 4-6: Original Essential Assumptions

ID	Assumption	Component of Interest
EA-1	The input provided for the Complete Set of Codes is correct (or correct enough).	**CSC**
EA-2	The input selected for each Computer Code is correctly selected (or correct enough) from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
EA-3	The input selected for each Coded Group is correctly selected (or correct enough) from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CG**
EA-4	The input selected for each Coded Physical Function is correctly selected (or correct enough) from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.	**CPF**
EA-5	Each Coded Physical Function calculates the correct output (or correct enough) for given input.	**CPF**
EA-6	The output from the Coded Group is correctly selected (or correct enough) from the output from the Coded Physical Functions in the Computer Code.	**CG**
EA-7	The output from the Computer Code is correctly selected (or correct enough) from the output from the Coded Groups in the Computer Code.	**CC**
EA-8	The output from the Complete Set of Codes is correctly selected (or correct enough) from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
EA-9	The specific output selected (or correct enough) from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

4.4.3. Characteristics of the Essential Assumptions

There are certain characteristics of the Essential Assumptions which may not be immediately obvious. First, the assumptions are mostly independent from each other; second, some of the assumptions are made multiple times in a given simulation; and third, the importance of each assumption varies. These three characteristics are discussed below.

4.4.3.1. Independence of Essential Assumptions

The Essential Assumptions are mostly independent from each other. The assumptions were written such that one assumption being true of false would not necessarily impact the other assumptions being true or false, even though it may have a great impact on the results of the simulation. For example, assume all of the assumptions are true except assumption EA-1 (*The input provided to the Complete Set of Codes is correct*). This assumption being false does not impact any of the other assumptions. That is, just because the input provided is wrong does not mean that the input selected by any component would be selected incorrectly. The process of selecting the input or output requires choosing the "correct box". The process of providing the correct input or calculating the correct output requires making sure the number placed in the "box" is correct. Thus, a Computer Code can choose the "correct box" independently of the number inside being correct.

4.4.3.2. Instances of Essential Assumptions

Some of the assumptions are made multiple times in a given simulation. Each component of the simulation assumes all associated Essential Assumptions are true. For example, the component of interest for EA-4 and EA-5 is the Coded Physical Function. That means that these assumptions are made for each and every Coded Physical Function in the simulation. If the simulation has 100 Coded Physical Functions, then each of these assumptions is made 100 times.

4.4.3.3. Importance of Essential Assumptions

While each Essential Assumption is important, some assumptions are more important than others in answering the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). That is not to say that in general assumption EA-2 is more or less important than assumption EA-4. Instead, it is more likely that specific instances of one assumption may be more important than specific instance of other assumptions.

For example, suppose a simulation has five Coded Physical Functions. Thus, the simulation will have five instances of EA-5. Suppose that the second Coded Physical Function has a large impact on the final result of the simulation, whereas the fourth has only a minimal impact. Then, EA-5 associated with the second Coded Physical Function will be a much more important assumption than EA-5 associated with the fourth Coded Physical Function.

Determining which instances of the Essential Assumptions are more important would be very useful. One possible method for doing so is by performing a sensitivity analysis where the simulation is adjusted such that the specific assumption is false in some way (i.e., some attributes for a particular Essential Assumption are forced to be at a lower maturity level in order to determine that impact on the simulation's results).

4.4.4. Deconstructing Essential Assumptions

The Essential Assumptions given in Table 4-6 can be thought of as a deconstruction of the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*. That is, the Essential Assumptions form a set of necessary and sufficient conditions for the Fundamental Assumption. Because they are more detailed than the Fundamental Assumption, the Essential Assumptions provide a clearer path to helping decision makers determine if a simulation should be trusted. That is, instead of decision makers asking themselves if there is enough evidence to support the Fundamental Assumption which is very broad, they can ask themselves if there is enough evidence to support the Essential Assumptions which are much more focused.

Generally, if someone is trying to decide if an assumption should be made, it is easier if that assumption has a narrower focus rather than a broader focus. This is the reasoning behind deconstructing the Fundamental Assumption into the Essential Assumptions. While the focuses of the Essential Assumptions are narrower, it would be even more helpful to make them narrower still. This is the goal in deconstruction the Essential Assumptions.

4.4.4.1. Rules and Guidelines for Deconstruction

The only rule for deconstruction of an Essential Assumption is that the resulting new set of assumptions must form a set of necessary and sufficient conditions for the original assumption. While this is the only true rule, the individual performing the deconstruction should strive to make each assumption in the new set independent from the other assumptions in that set and independent from all of the current Essential Assumptions. This recommendation will result in a better set of Essential Assumptions. An example of this deconstruction process is given in the following sub-section.

4.4.4.2. Deconstructing EA-5

An example of the deconstruction process is given by deconstructing Essential Assumption EA-5 (T*he Coded Physical Function calculates the correct output for the given input*). While an additional deconstruction of EA-5 is achievable by applying the definition of a Coded Physical Function, as defined in the Hierarchy of Scientific Computer Simulation Components, this deconstruction does not seem useful. Thinking of a Coded Physical Function in terms of the Assignment Statements which make up that function is not very useful in this case as Assignment Statements are greatly influenced by the computer language and the chosen representation style of the Coded Physical Function in that language. Thus, the same function could be represented multiple ways. For simulation review, a Coded Physical Function's representation in source code does not matter (as long as that representation is correct). Therefore, this deconstruction is not based on what Assignment Statements makeup the Coded Physical Function, but how to ensure that the Coded Physical Function is performing the calculation correctly.

Every Coded Physical Function has its origins in a physical equation. That physical equation often requires some amount of derivation to change it into its final form. Then, that final form must be changed through functionalization and discretization into the Coded Physical Function. Using these aspects of a Coded Physical Function, the following three statements can be generated which are both necessary and sufficient conditions for Essential Assumption EA-5 (*The Coded Physical Function calculates the correct output for the given input*):

- The Original form of the Physical Equation (**OPE**) is correct (or correct enough).
- The derivations and assumptions used to obtain the Final Physical Equation (**FPE**) from the original physical equation are correct (or correct enough).
- The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct(or correct enough).

That is, if any of the above assumptions are false, then EA-5 must also be false and if all of the assumptions are true, then EA-5 must also be true. As these assumptions are further deconstructions of EA-5, they are identified as assumptions **EA-5.1**, **EA-5.2**, and **EA-5.3**. The focus of all three assumptions is the Coded Physical Function. This information is captured in Table 4-7, which is the first deconstruction EA-5.

Table 4-7: Deconstruction of Essential assumption EA-5

ID	Statement	Component of Interest
EA-5.1	The original form of the physical equation (OPE) is correct (or correct enough).	**CPF**
EA-5.2	The derivations and assumptions used to obtain the final physical equation (FPE) from the original physical equation are correct (or correct enough).	**CPF**
EA-5.3	The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct (or correct enough).	**CPF**

4.4.4.3. Current Set of Essential assumptions

With the deconstruction of EA-5, the current set of Essential Assumptions is given below in Table 4-8.

Table 4-8: Current Set of Essential Assumptions

#	Statement	Component of Interest
EA-1	The input provided for the Complete Set of Codes is correct (or correct enough).	**CSC**
EA-2	The input selected for each Computer Code is correctly selected (or correct enough) from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
EA-3	The input selected for each Coded Group is correctly selected (or correct enough) from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CG**
EA-4	The input selected for each Coded Physical Function is correctly selected (or correct enough) from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.	**CPF**
EA-5.1	The original form of the physical equation (OPE) is correct (or correct enough).	**CPF**
EA-5.2	The derivations and assumptions used to obtain the final physical equation (FPE) from the original physical equation are correct (or correct enough).	**CPF**
EA-5.3	The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct (or correct enough).	**CPF**
EA-6	The output from the Coded Group is correctly selected (or correct enough) from the output from the Coded Physical Functions in the Computer Code.	**CG**
EA-7	The output from the Computer Code is correctly selected (or correct enough) from the output from the Coded Groups in the Computer Code.	**CC**
EA-8	The output from the Complete Set of Codes is correctly selected (or correct enough) from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
EA-9	The specific output selected (or correct enough) from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

4.5. Generating the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

The following are the two objectives of the Theoretical/Logical Framework:

(1) The framework should be structured to contain all of the possible maturity assessments sets which could be used to help answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

(2) The framework should be structured such that all of assumptions made when answering trusting a simulation (i.e., answering the Ultimate Question in the affirmative) have been taken into account.

To satisfy objective (1), the framework is structured such that it could organize a potentially infinite number of maturity assessment sets. This is done using an organizational scheme based on set theory, as a set can contain an infinite number of elements. Thus, the Theoretical/Logical Framework is organized into groups [17]of maturity assessment sets.

To satisfy objective (2), each assessment set in the framework is placed into a group depending on which Essential Assumption that assessment set supports. As the Essential Assumptions form a necessary and sufficient set of assumptions for the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*, then any assessment set which supports this assumption must also support at least one of Essential Assumptions. (Note that if the assessment set does not support the Fundamental Assumption, it is of no concern in simulation review).

Each group of maturity assessment sets is, by definition, its own framework. That is, each group is a set of maturity assessment sets which have a common focus, their associated Essential Assumption. Thus each group forms an Essential Maturity Framework. An **Essential Assumption Framework** is a group of maturity assessment sets which have the common focus of determining of supporting a specific Essential Assumption. Thus, each essential framework can be used to determine the specific requirements for the intended purpose of the simulation (i.e., identify what is "correct enough") and each framework can be used to determine the resulting maturity levels of the specific simulation.

[17] These are actually sets, but the term "group" is used instead of "set" to avoid confusion with the maturity assessment sets.

While the Essential Assumptions are necessary and sufficient for demonstrating that the Fundamental Assumption is true, it is likely that there are infinitely many maturity assessment sets needed to demonstrate that a given Essential Assumption is correct (or correct enough).

4.5.1. Focuses of the Theoretical/Logical Framework

The Theoretical/Logical Framework is made up of Essential Assumption Frameworks, each with a focus of a different Essential Assumption. These different focuses are given in Table 4-9.

Table 4-9: Focuses of the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

ID	Focus	Component of Interest
EA-1	The input provided for the Complete Set of Codes is correct (or correct enough).	**CSC**
EA-2	The input selected for each Computer Code is correctly selected (or correct enough) from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
EA-3	The input selected for each Coded Group is correctly selected (or correct enough) from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CG**
EA-4	The input selected for each Coded Physical Function is correctly selected (or correct enough) from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.	**CPF**
EA-5.1	The original form of the physical equation (OPE) is correct.	**CPF**
EA-5.2	The derivations and assumptions used to obtain the final physical equation (FPE) from the original physical equation are correct.	**CPF**
EA-5.3	The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct.	**CPF**
EA-6	The output from the Coded Group is correctly selected (or correct enough) from the output from the Coded Physical Functions in the Computer Code.	**CG**
EA-7	The output from the Computer Code is correctly selected (or correct enough) from the output from the Coded Groups in the Computer Code.	**CC**
EA-8	The output from the Complete Set of Codes is correctly selected (or correct enough) from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
EA-9	The specific output selected (or correct enough) from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

4.5.2. Observations on the Theoretical/Logical Framework

This sub-section provides some observations on the recently defined Theoretical/Logical Framework. These observations include a discussion on how to determine what is "true enough", a discussion on the independence of the Essential Assumption Frameworks, and finally a discussion on the maturity assessment sets in each Essential Framework.

4.5.2.1. How to define "correct enough"

As defined above, the Theoretical/Logical Framework was created to demonstrate the validity of the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*. The use of the framework does this by organizing the maturity assessment sets into groups, where each group focuses on a different Essential Assumption. The use of the framework in the computer simulation review process actually solves one of the major problems of the Fundamental Assumption, how to define "correct enough"?

The Fundamental Assumption recognizes that many simulations can and should be used without their results being completely true and accounts for this by allowing their results to be "true enough". Precisely defining "true enough" could be very difficult, but a similar concept has already been addressed, the Fundamental Question on the Requirements (*How trustworthy do the results of a scientific computer simulation need to be for the intended purpose?*). This question is answered in the second stage of simulation review (Maturity Requirements Determination). The choice of what is "correct enough" is made by a decision maker. The analyst and the decision maker can be the same person, or they can be different people.

It is important to realize that all of the maturity levels for a specific simulation could be correct (or correct enough) and the simulation may still provide results which should not be trusted. This can either be due to the decision maker not choosing "correct enough" appropriately or the fact that there are other attributes (i.e., other maturity assessment sets) which are currently missing from the framework and that if those attributes (i.e., assessment sets) were known, they would demonstrate that some set of the Essential Assumptions were not correct (or correct enough). While the Essential Assumptions are easier to understand and more focused than the Fundamental Assumption, one key assumption made whenever a framework is used is that the framework contains all of the important maturity assessment sets needed to assess the simulation. This is even true of the Theoretical/Logical Framework

4.5.2.2. Independence of Each Essential Maturity Framework

Because the focus of each Essential Assumption Framework is an Essential Assumption, and the Essential Assumptions are mostly independent, the Essential Frameworks are also mostly independent. Thus, the assessed maturity of any Essential Framework need not be related to the assessed maturity of any other framework. This independence is a result of the deconstruction process used on the Fundamental Assumption and is beneficial. By making the Essential Frameworks independent, it means that each of the frameworks can be used to assess the simulation separately and even by separate analysts.

4.5.2.3. Completeness of Each Essential Assumption Framework

There are three requirements of maturity assessment sets which make up any given Essential Assumption Framework:

(1) Each maturity assessment in an Essential Framework must have the focus of justify the Essential Assumption associated with that framework.

(2) Each maturity assessment set is necessary for determining the Essential Assumption of the framework is valid.

(3) The set of all maturity assessment sets is sufficient for demonstrating the Essential Assumption of the framework is valid.

Requirements (1) and (2) are much easier to meet than (3). As discussed before, it is likely that an infinite number of assessment sets would be needed to prove that any Essential Assumption is valid. Therefore, the entire Theoretical/Logical Framework and each Essential Framework is likely incomplete in that additional maturity assessment sets are needed to demonstrate that the Essential Assumptions are correct (or correct enough).

The need for additional assessment sets is separate from the

4.6. Conclusions on Developing the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

The Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations is framework which can be used to answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*). Specifically, it is a framework composed of other frameworks, each with the focus of a specific Essential Assumption. The biggest advantage of this multiple framework approach is that if each Essential Assumption can be demonstrated to be correct (or correct enough), the answer to the Ultimate Question must be "yes".

The Theoretical/Logical Framework was not developed for direct application; instead it was developed to provide a structure which can capture all possible maturity assessment sets. Many of these sets would not be used in a practical framework, but in order to make such a determination if a set should or should not be used for the review of a specific simulation, that set must be captured somewhere.

Additionally, the framework provides a list of all of the assumptions which are made when the results of a simulation are trusted (i.e., a decision maker answers "yes" to the Ultimate Question). That is, the list of Essential Assumptions is demonstrated to be both necessary and sufficient conditions for the Fundamental Assumptions (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*. Thus, these assumptions are made even if the analyst or decision is unaware that they are making them. It is hoped that by seeing additional detail on the assumptions made, a decision maker will better understand everything that needs to be considered when answering the Ultimate Question.

The next step is to develop the maturity assessment sets which will populate the Theoretical/Logical Framework. The process used to develop these assessment sets is discussed in Chapter 5 and the assessment sets themselves are provided in Chapter 6.

5. Developing Maturity Assessment Sets

The Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations was developed with the following two objectives:

(1) The framework should be structured to contain all of the possible maturity assessments sets which could be used to help answer the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

(2) The framework should be structured such that all of assumptions made when answering trusting a simulation (i.e., answering the Ultimate Question in the affirmative) have been taken into account.

To meet these two objectives, the Theoretical/Logical Framework was developed using the Hierarchy of Scientific Computer Simulation Components to deconstruct the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*. The Fundamental Assumption was deconstructed into a set of Essential Assumptions, each of which became the focus of its own Essential Assumption Framework.

The first section provides a thought experiment which is used to better understand a maturity assessment set. The second section examines the creation of maturity assessments from either currently known assessments sets or from concepts. The third section provides a review of the current maturity assessments sets used in the PCMM and NASA frameworks. The fourth section discusses the topics of Verification, Validation, and Uncertainty Quantification and how they fit in the Theoretical/Logical Framework. Finally, the fifth section concludes and summarizes the contributions of this chapter.

5.1. Thought Experiment: Creating a List of Every Reason Why a Scientific Computer Simulation Should be Trusted

There are many methods which can be used to generate a maturity assessment set. One such method is through the use of the following thought experiment. The thought experiment explores the concept of Supporting Evidence, how such evidence can be used to support the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough))*, and how such evidence is compared to other similar evidence in determining which is more mature.

5.1.1. Thought Experiment

Suppose that Bob works for NASA and has just completed performing a scientific computer simulation. Now he wants his boss to use that simulation to make a certain decision, so he decides to make a list of all the reasons why the results of the simulation should be trusted. That is, Bob takes the Active Approach and assumes that the entire simulation is incorrect unless specific evidence suggests otherwise. He then seeks out this specific evidence and uses to demonstrate that the simulation can be trusted.

He writes down specific statements like the source code was peer reviewed by 2 other reviewers, physical model X was verified using a common benchmark, physical model Y is known by the industry to be a reliable model, etc. In other words, Bob's list contained all of the reasons why he thought that the Fundamental Assumption was valid for that simulation.

After creating this first list, Bob decides to perform a new simulation and create another list. However, this time he chooses to simulate a completely different scenario using completely different physical models and a completely different source code. Again, after he completes the simulation he creates another list of all the reasons why the results of the new simulation should be trusted. While these simulations have nothing in common, Bob notices that the lists for each simulation have similar sounding statements. This similarity inspires Bob into doing the impossible.

Bob decides to perform every scientific computer simulation that could possibly be performed. After obtaining the results of these infinitely many simulations, he creates infinitely many lists of the reasons why the results of each simulation should be trusted. Bob then combines all of the infinitely many lists into one master list. This master list contains every reason why any possible simulation should be trusted.

Bob wants to go one step further and come up with an organization scheme for this master list. So he decides to place all of the reasons into groups. In each group he places reasons which are similar to every other reason in that group, but differ by degree. For example, one group would contain the reasons "The source code was peer reviewed by 2 people" and "The source code was peer reviewed by 3 people". Since all the reasons in the group differ by degree, Bob is able to order the reasons from worst to best.

Finally, Bob notices that every reason in a group answers the same question. For the two reasons given above, they and every reason in their group answer the question

"How many people peer reviewed the source code?" So Bob decides to name each group using the question which every reason in the groups answers.

5.1.2. Insight into Maturity Assessment Sets

The groups of reasons which are similar and only differ by degree are maturity assessment sets. The names of the groups, the question which every reason in the group answers, are the attributes of the assessment sets. In this thought experiment, Bob has generated every possible maturity assessment set and has identified all of the possible levels (or criteria) of each assessment set. For example, a maturity assessment set can be made from the two reasons given above and this assessment set is given in Table 5-1.

Table 5-1: Example Maturity Assessment

Maturity Object	Scientific Computer Simulation
Maturity Attribute	How many people peer reviewed the source code?
Maturity Levels	
0	No one
1	One Person
2	Two People
3	Three People
⋮	⋮
N	N People

Thus, this thought experiment demonstrates that one way to think of a maturity assessment set is that it is a group of reasons why the simulation should be trusted which are similar to each other and only differ by degree and the maturity attribute of

that assessment set is the question which every reason in the set answers. These insights into maturity assessment sets are helpful in creating new assessment sets.

5.2. Creating Maturity Assessment Sets

The creation of a new maturity assessment set typically follows one of two paths. Either the new assessment set is evolved from a current assessment set (or sets) or the new assessment set is created from a concept for which no assessment set exists. No matter which process is used, the attributes of maturity discussed in Sections 2.1.6, 2.1.7, and 2.1.8 should be kept in mind to guide the analyst in creating the best assessment sets (and frameworks) possible. However, before any of these methods can be discussed, a common format for expressing maturity tables is required.

5.2.1. Format of a Maturity Assessment Table

Maturity assessment tables have been used through this dissertation, and while they each tend to follow the basic format given below in Table 5-2, there have been some exceptions.

Table 5-2: General Maturity Assessment Attribute Table

Maturity Object	Object Title
Maturity Attribute	Attribute Title
Maturity Levels	
1	Maturity level of one extreme (Lowest or Highest)
⋮	⋮
N	Maturity level of the other extreme (Highest or Lowest)

For example, Table 2-7, Table 2-8, and Table 2-9 which gave the assessment sets for the 4Cs framework had decreasing levels of maturity instead of increasing levels of maturity. Additionally, the above table captures many of the important aspects of the maturity assessment set, but not all of them. Therefore, a more formalized maturity assessment table is suggested and is given in Table 5-3. The table is split into two main regions, rows which provide background information for the Maturity Assessment Set and the Maturity Assessment Set itself.

Table 5-3: Suggested Format of a Maturity Assessment Table

Maturity Attribute		Name of the Maturity Attribute					
ID	**v.**	ID #	Rev #	**Parent**	**v.**	ID #	Rev #
Reference		Reference of the attribute assessment given in this table					
Parent Reference		Reference of the parent attribute assessment					
Other Aspect		Relevant Information					
Maturity Assessment Set							
-1		Does not apply					
0		Lowest possible maturity level (generally the null-case)					
1		Next to lowest possible maturity level					
⋮		⋮					
N		Highest possible maturity level					

The background rows provide important information for tracking the development of the Maturity Assessment Set and other useful information about the assessment set itself. Information commonly found in the background rows are:

- **Maturity Attribute** – The name of the maturity attribute. It is recommended that the name be the question which every maturity level answers.
- **ID** and **v.** – The identification and version numbers are used to specify a specific assessment set. While a numerical identifier is recommended, the only requirement is that the ID and version be unique for the given assessment set. One way to determine if the change to an assessment set requires a change to the ID or the version is provided by the following rule of thumb: A change to the attribute requires a change to the ID and a change to anything but the attribute requires a change to the version.
- **Parent ID** and **v.** – Along with the ID and version of the current assessment set, the table also provides a place for the ID and version of the parent assessment set. This is the set (or sets) which were used in creating the current assessment set. This information is useful in understanding the development and tracking the history of a given assessment set.
- **Reference** – This is the document reference which contains the given assessment set. If the assessment set is taken from some outside source, this

should be a reference to that source. If the assessment set is being developed in the current document, this cell should read "current".

- **Parent Reference** – This is the document reference of the parent assessment set. This information is useful in understanding the development and tracking the history of a given assessment set.
- **Other Aspect** – There are numerous other aspects which may be added to the maturity table. Whatever other aspects are added, they should be clearly defined before being used.

Below the background rows, the Maturity Assessment Set is provided as two columns of the numerical level and the criteria.

- **Numerical Maturity Level** – The format suggested here for providing the assessment set is using a numerical ordering scheme starting at level "-1" which has the associated criteria that the given assessment set does not apply. This is followed by level "0" which has the associated criteria of the lowest possible level of maturity for the given assessment set. The levels are incremented by 1 until level "N" is reached which has the highest possible level of maturity for the assessment set.
- **Maturity Criteria** – Each numerical maturity level must have associated criteria. The criteria is used to define which maturity level has been achieved for the given maturity object in the specific attribute. However, sometimes the maturity criteria are also referred to as the maturity levels. This is because each criterion represents a different level of maturity.

Ultimately, all parts of the maturity table are optional except for the assessment set, as it is the assessment set that defines the maturity table. The format suggested here is similar to the maturity assessment frameworks of PCMM and NASA.

5.2.2. Evolving from Known Maturity Assessment Sets

The **Evolution of a Maturity Assessment Set** is the process of creating a new assessment set from one or more known assessment sets. This process is commonly done in the creation of every assessment set, as even an assessment set created from a concept is usually evolved until it becomes satisfactory. Typically, there are two methods to evolve an assessment set: Aggregate and Abridge or Divide and Distill.

Aggregate and Abridge is the evolution process where two or more parent maturity assessment sets are aggregated and result in an assessment set whose criteria are an abridgment of the assessment sets of the parents. In this process, information is lost as the finer detail of the parent assessment sets is combined into a less detailed aggregated set. While this loss of information is not ideal, it is often necessary in a world with limited resources.

Aggregating multiple assessment sets into one set does not have the same dangers as combining maturity levels into one level. This is because the meaning of a maturity level of an aggregated assessment is clearly defined in the criteria of the aggregated set whereas the meaning of a combined maturity level has no clear definition and no directly associated criteria. While it may be tempting to combine the maturity levels of one or more assessment sets, all currently known combination process (e.g., weighted means) rely not only on the levels being combined have a scale, but all of the levels combined having the same scale. Maturity levels do not an associated scale and therefore any combination process is misleading at best and dangerous at worst. Instead of the combing the assessed maturity levels of certain assessment sets, the sets themselves should be aggregated and abridged into a new assessment set which can then be used to assess the simulation.

Divide and Distill is the evolution process where the parent maturity attribute is divided into multiple parts and criteria of the parent maturity assessment set are distilled to capture the levels of the new attributes. In this process, detailed information is usually gained. This is likely the most commonly used process in creating assessment sets as both the attribute and the criteria are divided and distilled until the resulting assessment set captures the intended information.

5.2.3. Example: Evolving a Maturity Attribute Assessment

Table 5-4 is a maturity assessment set taken from the PCMM framework. Examination of the assessment set reveals that it could be evolved into multiple assessment sets by dividing the attribute into three attributes and distilling the criteria.

Table 5-4: Example Maturity Assessment from PCMM

Maturity Attribute		**Representation and geometric fidelity** What features are neglected because of simplifications and stylizations?					
ID	**v.**	PCMM-1	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Little or no representational or geometric fidelity for the system and boundary conditions					
1		• Significant simplification or stylization of the system and boundary conditions • Geometry or representation of major components is defined					
2		• Limited simplification or stylization of major components and boundary conditions • Geometry or representation is well defined for major components and some minor components • Some peer review conducted					
3		• Essentially no simplification or stylization of components in the system and boundary conditions • Geometry or representation of all components is at the detail of *as built*, e.g., gaps, material interfaces, fasteners • Independent peer review conducted					

The attribute can be divided into the following three attributes:

1. What is the level of simplification or stylization of the system and boundary conditions?
2. How well is the geometry or representation defined?
3. What is the level of peer review?

These attributes along with their distilled criteria are given in

Table 5-5, Table 5-6, and Table 5-7.

Table 5-5: Divided PCMM Assessment (1 of 3)

Maturity Attribute		What is the level of simplification or stylization of the system and boundary conditions?					
ID	**v.**	PCMM-1.1	0	**Parent**	**v.**	PCMM-1	0
Reference		Current					
Parent Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
-1		Does not apply					
0		Judgment only					
1		Significant simplification or stylization of the system and boundary conditions					
2		Limited simplification or stylization of major components and boundary conditions					
3		Essentially no simplification or stylization of components in the system and boundary conditions					

Table 5-6: Divided PCMM Attribute Assessment (2 of 3)

Maturity Attribute		How well is the geometry or representation defined?					
ID	**v.**	PCMM-1.2	1	**Parent**	**v.**	PCMM-1	0
Reference		Current					
Parent Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
-1		Does not apply					
0		Little or no representational or geometric fidelity for the system and boundary conditions					
1		Geometry or representation of major components is defined					
2		Geometry or representation is well defined for major components and some minor components					
3		Geometry or representation of all components is at the detail of *as built*, e.g., gaps, material interfaces, fasteners					

Table 5-7: Divided PCMM Attribute Assessment (3 of 3)

Maturity Attribute		What is the level of Peer Review?					
ID	**v.**	PCMM-1.3	0	**Parent**	**v.**	PCMM-1	0
Reference		Current					
Parent Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
-1		Does not apply					
0		No peer review conducted					
1		Some peer review conducted					
2		Independent peer review conducted					

The results of the divide and distill process are criteria which are more distinct and exact as well as assessment sets which are more focused.

5.2.4. Creating from a Concept

Creation from a Concept is the process of creating a new assessment set for a specific maturity attribute from a general concept. This can be done in many ways, but generally the first assessment set created is not what is desired and therefore the maturity assessment set is evolved using the Divide and Distill process.

The creation process suggested here focuses on the maturity attribute and uses the following six steps:

1. Describe the Maturity Attribute
2. Write the Maturity Question
3. Identify all Possible Maturity Levels (i.e., Criteria) of the Attribute
4. Order the Maturity Levels (i.e., Criteria) from Least to Greatest Maturity
5. Create the Maturity Assessment Table
6. Analyze the Assessment

Each of these steps is further described below, but performing the steps can be very challenging. The thought experiment given at the beginning of this chapter is helpful in creating assessment sets from a general concept.

5.2.4.1. Step 1 – Describe the Maturity Attribute

The goal of this step is not to write the maturity attribute explicitly; instead, it is to think about what attribute should be about. There are many different attributes that could be used to assess the maturity of the given object, so what aspect of that maturity should be captured by this attribute?

5.2.4.2. Step 2 – Write the Maturity Question

After the analyst has some idea of the specific aspect of maturity he or she wishes to capture, the analyst should attempt to write out an appropriate maturity attribute. It is suggested that the attribute be written as a very specific question, where the answers to the question are the criteria which make up the assessment set and are also the possible answers of any simulation.

5.2.4.3. Step 3 – Identify All Possible Maturity Levels (i.e., Criteria) of the Attribute

Once the attribute has been chosen, the levels of the attribute need to be determined. If the attribute has been written as a question, these levels are all the possible answers to that question. This step is not concerned with placing those levels in any order and is solely concerned with obtaining a complete listing of the levels.

5.2.4.4. Step 4 – Order the Maturity Levels (i.e., Criteria) from Least to Greatest Maturity

The last major step in the creation of an assessment set is ranking the levels in order of increasing maturity. All of the possible levels were identified in step 3 and now those levels must be placed in order of increasing maturity. This process should be as objective and independent as possible. That is, one analyst with a reasonable background in the material should produce the same order for the levels as another analyst.

5.2.4.5. Step 5 – Creating the Maturity Assessment Table

Once the levels are ordered, a maturity assessment table can be created. The table can also contain any other aspects of the assessment set the analyst finds relevant but

should at least including the maturity attribute and some type of identification number.

5.2.4.6. Step 6 – Analyzing the Assessment

Once the assessment table has been created, it can be analyzed to determine if the maturity attribute is really being answered by the maturity levels. At this point, an analyst will likely wish to improve the assessment set by using the evolution process descried above. Some useful questions to ask are:

- Does the attribute have the correct focus?
- Why is level 1 less mature than all the others? Is it possible to have a level that would be less mature?
- Where is level "N" more mature than all others? Is it possible to have a level that would be more mature?

5.2.5. Example: Creating an Attribute Assessment for EA-5.1

In this example, the six step process is used to create a maturity assessment set for Essential Assumption EA-5.1 (*The original form of the physical equation is correct*).

5.2.5.1. Step 1 – Describe the Maturity Attribute

The attribute is focused on ensuring the original form of the physical equation is correct (or correct enough). However, even this focus is somewhat broad, as it could be supported with many types of evidence. Therefore, the focus is narrowed to the source of the original form of the first principle physical equation and how trusted is that source.

5.2.5.2. Step 2 – Write the Maturity Question

Once the focus is narrowed to the source of the original form, the attribute can be written as the following question:

> What is the reference of the original form of the first principle physical equation?

5.2.5.3. Step 3 – Identify All Possible Maturity Levels (i.e., Criteria) of the Attribute

The above question can have many answers and it may not be possible to prove that all of the answers have been identified. Thus, this assessment set may need to be modified in the future if a new answer is discovered. For the time being, the following list captures most of the answers to this question:

- Text book
- Journal Article
- Personal Memory
- Personal Notes
- Internet
- Genesis Equation
- Don't know

Notice that the above levels have not been placed in any order.

5.2.5.4. Step 4 – Order the Maturity Levels (i.e., Criteria) from Least to Greatest Maturity

Ordering (or ranking) the maturity levels (or criteria) in order from lowest to highest maturity should result in an objective ranking. The analyst should consider how others would also rank the levels. If the levels are quantitatively different the ranking process becomes mathematic, however the levels are usually qualitatively different which may result in personal bias.

In this case, the ranking is done by how much the source is trusted; with the lowest trusted source having the lowest maturity and the highest trusted source having the highest maturity.

The lowest level of maturity (level 0) is "Don't know". It is reasonable to assume that the origin of the original physical equation must be higher than this level, even if the analyst performing the assessment doesn't know it. However, one of the reasons to perform maturity assessment is to assess not only the state of maturity of a simulation, but the current level of knowledge about that simulation. Therefore, the lowest level in most assessment sets is "Don't know" or "None".

Determining the next level (level 1) is more complicated. Some could argue that "Personal Memory" is the next lowest maturity, while others may say it would be the "Internet" or "Personal Notes". Since a good argument could be made for any one of these three being more or less mature than the other two, these three should be grouped in the same level.

Combining these three into one level is further confirmed when examining the last three possible levels. No matter how an analyst would rank "Personal Memory", "Personal Notes", or "Internet", that analyst would likely rank those sources below "Text books", "Journal Articles", and a "Genesis Equation". Therefore, level 1 is defined as "Personal Memory", "Personal Notes", or "Internet".

Determining the next level (level 2) presents a similar challenge. It seems that "Text books", "Journal Articles", and a "Genesis Equation" may all have the same maturity. However, further consideration reveals that while "Text books" and "Journal Articles" are close to the same level of maturity, they are less mature than a Genesis Equation. As defined, a "Genesis Equation" is specifically written to be the closest representation of a first principle as possible. It has no further assumption than those of the first principle. On the other hand, the equations found in text books and journals are likely to have further assumptions and those assumptions may or may not be justified. Therefore level 2 is defined as "Text book" or "Journal Articles" and the highest level (level 3) is defined as "Genesis Equation".

5.2.5.5. Step 5 – Creating the Maturity Assessment Table

With the maturity levels ranked, a maturity assessment table can created and is given in Table 5-3. Notice that a row was added to identify what Essential Assumption this attribute assessment set is supporting.

Table 5-8: Maturity Attribute Assessment Table for EA-5.1

Maturity Attribute		What is the reference of the original form of the first principle physical equation?					
ID	**v.**	Example-1	0	**Parent**	**v.**		
Reference		Current					
EA		EA-5.1					
Maturity Assessment Set							
-1		Does not apply					
0		Don't know.					
1		Internet, Personal Notes, or Memory					
2		Text book or journal article					
3		Genesis Equation					

5.2.5.6. Step 6 – Analyzing the Assessment

The attribute was focused on how much the source of the original form of the physical equation could be trusted. This attribute could be rewritten as multiple attributes (i.e., multiple assessment sets), each with a tighter focus and more detailed assessment criteria. The following questions are examples of those which could be used to make this assessment set better:

- What type of journal did the journal article come from? (highly respected or fly-by-night)
- How commonly referenced is the article? (De facto standard or completely unknown)
- Is the textbook considered a classic? (Used undergraduates everywhere or a professor's pet project)
- What edition of the textbook is it? (Tenth Edition with few errors or a first edition with many)

While answering these and other questions are very important in creating a good assessment set, most good questions arise from the creation process and are only thought of after the assessment set has been created. Therefore, creating good assessment sets is a process of constant revision.

5.3. Current Maturity Assessments

It is often easier to evolve a known assessment set than to create one from a concept. Therefore, this section provides the assessment sets used in the PCMM and NASA frameworks. These assessment sets have not been modified, expect they are given in the format of the maturity table described above.

5.3.1. PCMM Maturity Assessment Framework

The following are the maturity assessment sets of the PCMM framework as recorded in Table 15.3 of Oberkampf and Roy (2010). The framework is separated into the following six attributes with one assessment set in each area:

1. Representation and Geometric Fidelity
2. Physics and Material Modeling Fidelity
3. Code Verification
4. Solution Verification
5. Model Verification
6. Uncertainty Quantification

5.3.1.1. PCMM Assessments

Table 5-9: PCMM-1 Maturity Table

Maturity Attribute		**Representation and geometric fidelity** What features are neglected because of simplifications and stylizations?					
ID	**v.**	PCMM-1	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Little or no representational or geometric fidelity for the system and boundary conditions					
1		• Significant simplification or stylization of the system and boundary conditions • Geometry or representation of major components is defined					
2		• Limited simplification or stylization of major components and boundary conditions • Geometry or representation is well defined for major components and some minor components • Some peer review conducted					
3		• Essentially no simplification or stylization of components in the system and boundary conditions • Geometry or representation of all components is at the detail of *as built*, e.g., gaps, material interfaces, fasteners • Independent peer review conducted					

Table 5-10: PCMM-2 Maturity Table

Maturity Attribute		**Physics and Material Model Fidelity** How fundamental are the physics and material models and what is the level of model calibration?					
ID	**v.**	PCMM-2	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Model forms are either unknown or fully empirical • Few, if any, physics-informed models • No coupling of models					
1		• Some models are physics based and are calibrated using data from related systems • Minimal ad hoc coupling of models					
2		• Physics-based models for all important processes • Significant calibration needed using separate effects tests (SETs) and integral effects tests (IETs) • One-way coupling of models • Some peer reviews conducted					
3		• All models are physics based • Minimal need for calibrations using SETs or IETs • Sound physical basis for extrapolation and coupling of models • Full, two-way coupling of models • Independent peer review conducted					

Table 5-11: PCMM-3 Maturity Table

Maturity Attribute		**Code Verification** Are algorithm deficiencies, software errors, and poor SQE practices corrupting the simulation results?					
ID	**v.**	PCMM-3	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Minimal testing of any software elements • Little or no SQE procedures specified or followed					
1		• Code is managed by SQE procedures • Unit regression testing conducted • Some comparisons made with benchmarks					
2		• Some algorithms are tested to determine the observed order of numerical convergence • Some Features & Capabilities (F&C) are tested with benchmark solutions • Some peer reviews conducted					
3		• All important algorithms are tested to determine the observed order of numerical convergence • All important F&Cs are tested with rigorous benchmark solutions • Independent peer review conducted					

Table 5-12: PCMM-4 Maturity Table

Maturity Attribute		**Solution Verification** Are numerical solution errors and human procedural errors corrupting the simulation results?					
ID	**v.**	PCMM-4	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Numerical errors have an unknown or large effect on simulation results					
1		• Numerical effects on relevant System Response Quantities (SRQs) are qualitatively estimated • Input/output (I/O) verified only by the analysts					
2		• Numerical effects are quantitatively estimated to be small on some SRQs • I/O independently verified • Some peer reviews conducted					
3		• Numerical effects are determined to be small on all important SRQs • Important simulations are independently reproduced • Independent peer review conducted					

Table 5-13: PCMM-5 Maturity Table

Maturity Attribute		**Model Validation** How carefully is the accuracy of the simulation and experimental results assessed at various tiers in a validation hierarchy?					
ID	**v.**	PCMM-5	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Few, if any, comparisons with measurements from similar systems or applications					
1		• Quantitative assessment of accuracy of SRQs not directly relevant to the application of interest • Large or unknown experimental uncertainties					
2		• Quantitative assessment of predictive accuracy for some key SRQs from IETs and SETs • Experimental uncertainties well characterized for most SETs, but poorly for IETs • Some peer review conducted					
3		• Quantitative assessment of predictive accuracy for all important SRQs from IETs and SETs at conditions/geometries directly relevant to the application • Experimental uncertainties are well characterized from all IETs and SETs • Independent peer review conducted					

Table 5-14: PCMM-6 Maturity Table

Maturity Attribute		**Uncertainty Quantification and Sensitivity Analysis** How thoroughly are uncertainties and sensitivities characterized and propagated?					
ID	**v.**	PCMM-6	0	**Parent**	**v.**		
Reference		Oberkampf, W.L., and C.J. Roy. (2010) *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge.					
Maturity Assessment Set							
0		• Judgment only • Only deterministic analyses are conducted • Uncertainties and sensitivities are not addressed					
1		• Aleatory and Epistemic (A&E) uncertainties propagated, but without distinction • Informal sensitivity studies conducted • Many strong UQ/SA assumption made					
2		• A&E uncertainties segregated, propagated and identified in SRQs • Quantitative sensitivity analyses conducted for most parameters • Numerical propagation errors are estimated and their effect known • Some strong assumptions made • Some peer review conducted					
3		• A&E uncertainties comprehensively treated and properly interpreted • Comprehensive sensitivity analyses conducted for parameters and models • Numerical propagation errors are demonstrated to be small • No significant UQ/SA assumptions made • Independent peer review conducted					

5.3.1.2. PCMM Framework Summary

A summary of the PCMM framework is provided in Table 5-15.

Table 5-15: PCMM Framework Summary

Categories	Attribute	Levels
Representation and Geometric Fidelity	What features are neglected because of simplifications and stylizations?	0 – 4
Physics and Material Model Fidelity	How fundamental are the physics and material models and what is the level of model calibration?	0 – 4
Code Verification	Are algorithm deficiencies, software errors, and poor SQE practices corrupting the simulation results?	0 – 4
Solution Verification	Are numerical solution errors and human procedural errors corrupting the simulation results?	0 – 4
Model Validation	How carefully is the accuracy of the simulation and experimental results assessed at various tiers in a validation hierarchy?	0 – 4
Uncertainty Quantification and Sensitivity Analysis	How thoroughly are uncertainties and sensitivities characterized and propagated?	0 – 4

5.3.2. NASA Maturity Assessment Framework

The following are the maturity assessments developed from the NASA's Standard for Models and Simulation as described by NASA[1] (2009). These maturity assessments are taken from Table 1 – Table 4 in Appendix B. The framework is separated into the following three areas with multiple assessment sets per area:

1. M&S Development
2. M&S Operation
3. Supporting Evidence

Additionally, a Technical Review assessment is performed for each assessment in M&S Development and M&S Operation

5.3.2.1. M&S Development

Table 5-16: NASA-1 Maturity Table

Maturity Attribute		**Verification** Were the models implement correctly, and what was the numerical error/uncertainty?					
ID	**v.**	NASA-1	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Conceptual and mathematical models verified – Favorable evidence for verification of conceptual and mathematical models.					
2		Unit and regression testing of key features – Favorable results from unit and regression testing of key features of the computational model.					
3		Formal numerical error estimation – Some formal method is used to assess numerical errors associated with unit testing with significant coverage of the code.					
4		Numerical errors for all important features – Reliable error estimation methods are used to quantitatively assess numerical errors. These estimates show that the errors are small from test suites, which exercise all important algorithms, all important features and capabilities, and all important couplings (physics, groups, etc.) of the full compotation model.					

Table 5-17: NASA-2 Maturity Table

Maturity Attribute		**Validation** Did the M&S results compare favorably to the referent data, and how close is the referent data to the real-world system?					
ID	**v.**	NASA-2	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Conceptual and mathematical models agree with simple referents – M&S conceptual and mathematical models compare favorably with "general problem" and "textbook" referents.					
2		Results agree with experimental data or other M&S on unit problems – M&S results compare favorably for unit problems at validation points by comparison of M&S results to an acceptable referent, which is either experimental measurements or higher fidelity M&S results.					
3		Results agree with experimental data for problems of interest – M&S results compare favorably for problems of interest at validation points by comparison of M&S results to an acceptable referent, which is experimental measurement on problems of interest.					
4		Results agree with real-world data – M&S results compare favorably for the real world system at validation points by comparison of M&S results to an acceptable referent, which is measurements on the real-world system.					

5.3.2.2. M&S Operation

Table 5-18: NASA-3 Maturity Table

Maturity Attribute		**Input Pedigree** How confident are we of the current input data?					
ID	**v.**	NASA-3	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Input data traceable to informal documentation – The input data is traceable to informal documentation.					
2		Input data traceable to formal documentation – The input data is traceable to formal documentation.					
3		Input data agree with experimental data for problems of interest – The input data compare favorably with acceptable measured referent data from other problems of interest. Uncertainty associated with the input data is known.					
4		Input data agree with real world data – The input data compare favorably with measured data from the real-world system. Uncertainty associated with the input data is known.					

Table 5-19: NASA-4 Maturity Table

Maturity Attribute		**Results Uncertainty** What is the uncertainty in the current M&S results?					
ID	**v.**	NASA-4	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Qualitative estimates – Uncertainty estimates are qualitative.					
2		Deterministic analysis or expert opinion - Uncertainty estimates are quantitative and based on determinist analysis or expert opinion.					
3		Non-deterministic analysis – Uncertainty estimates are quantitative and based on nondeterministic analysis.					
4		Non-deterministic and numerical analysis – Uncertainty estimates are quantitative and based on nondeterministic and numerical analysis.					

Table 5-20: NASA-5 Maturity Table

Maturity Attribute		**Results Robustness** How thoroughly are the sensitivities of the current M&S results known?					
ID	**v.**	NASA-5	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Qualitative Estimates – Sensitivity of M&S results for the real-world system is estimated by analogy with the quantified sensitivity of similar problems of interest.					
2		Sensitivity known for few parameters – Sensitivity of M&S results for the real-world system is quantitatively known for a few variables and parameters.					
3		Sensitivity known for many parameters – Sensitivity of M&S results for the real-world system is quantitatively known for a many variables and parameters.					
4		Sensitivity known for most parameters; key sensitivities identified – Sensitivity of M&S results for the real-world system is quantitatively known for most of the variables and parameters, including all of the most sensitive variables and parameters.					

5.3.2.3. Supporting Evidence

Table 5-21: NASA-6 Maturity Table

Maturity Attribute		**Use History** Have the current M&S been used successfully before?					
ID	**v.**	NASA-6	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Passes simple tests – Specific scenarios have been created to test applications, or results compare favorably with outputs from other similar tools.					
2		Used before for critical decisions – Used previously to perform analysis upon which critical decisions have been made.					
3		Previous predictions were later validated by mission data – Post-decision real-world events have been accurately represented in results (e.g., validated by mission data).					
4		De facto standard.					

Table 5-22: NASA-7 Maturity Table

Maturity Attribute		**M&S Management** How well managed were the M&S processes?					
ID	**v.**	NASA-7	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Managed process – The M&S roles and responsibilities have been defined.					
2		Established process – The M&S effort has established a documented process for M&S development and operation.					
3		Predictable process – The M&S effort is measuring repeatability of the M&S results generated by the M&S process.					
4		Continual process improvement – The M&S effort is using measurements on M&S processes to improve the repeatability of the M&S results.					

Table 5-23: NASA-8 Maturity Table

Maturity Attribute		**People Qualifications** How qualified were the personnel?					
ID	**v.**	NASA-8	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Engineering or science degree – Possess engineering or science degree, has been introduced to the topic of M&S, and has been exposed to generic recommended practices in M&S.					
2		Formal M&S training and experience, and recommend practice training – Possess engineering or science degree, has received formal training in formulation of M&S and generic training in recommended practices for M&S, and has developed M&S products.					
3		Advanced degree or extensive M&S experience, and recommended practice knowledge – Possesses advanced engineering or science degree or extensive work experience, has general M&S training, has specific experience with the M&S being reviewed, and has been trained on specific recommended practices relevant to the current application.					
4		Extensive experience in and use of recommended practices for this particular M&S – Possesses advanced engineering or science degree or extensive work experience, has extensive experience with the development and use of the M&S being reviewed, and has employed specific recommended practices relevant to current application.					

5.3.2.4. Technical Review

Table 5-24: NASA-9 Maturity Table

Maturity Attribute		What was the level of Technical Review?					
ID	**v.**	NASA-9	0	**Parent**	**v.**		
Reference		NASA (2009). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.					
Maturity Assessment Set							
0		Insufficient Evidence.					
1		Favorable informal internal peer review.					
2		Favorable formal internal peer review.					
3		Favorable external peer review.					
4		Favorable external peer review accompanied by independent *factor evaluation*.					

5.3.2.5. NASA Framework Summary

A summary of the NASA framework is provided in Table 5-25.

Table 5-25: NASA Framework Summary

Category	Attribute	Levels
M&S Development	Were the models implement correctly, and what was the numerical error/uncertainty?	0 – 4
	What was the level of Technical Review?	0 – 4
	What is credibility of the validation that has been performed?	0 – 4
	What was the level of Technical Review?	0 – 4
M&S Operation	How confident are we of the current input data?	0 – 4
	What was the level of Technical Review?	0 – 4
	What is the uncertainty in the current M&S results?	0 – 4
	What was the level of Technical Review?	0 – 4
	How thoroughly are the sensitivities of the current M&S results known?	0 – 4
	What was the level of Technical Review?	0 – 4
Supporting Evidence	Have the current M&S been used successfully before?	0 – 4
	How well managed were the M&S processes?	0 – 4
	How qualified were the personnel?	0 – 4

5.3.3. Summary

Both the PCMM and NASA frameworks focus on modeling issues, Verification, Validation, and Uncertainty Quantification (VV&UQ). These are typically the tasks which produce the most important Supporting Evidence when an analyst is attempting to justify the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are true (or true enough).*) There are numerous references which describe these tasks in detail (including Oberkampf and Roy 2010). However, a brief description of each task and how it fits into the Theoretical/Logical Framework is provided in the following section.

5.4. Verification, Validation, and Uncertainty Quantification

Perhaps there are no tasks more closely associated with scientific computer simulation review than verification, validation, and uncertainty quantification (VV&UQ). Both PCMM and the NASA framework have these tasks as key elements in maturity assessment. However, it is not immediately apparent how these tasks are used in the Theoretical/Logical Framework. That is, how are VV&UQ used to support the Fundamental Assumption (*The assumption that the results of the scientific computer simulation are correct (or correct enough)*)?

Additional clarity is needed because the Theoretical/Logical Framework was not formed based on what Supporting Evidence is commonly provided. Instead, it was based on what evidence is needed to support the Essential Assumptions. VV&UQ are the tasks most associated with providing supporting evidence, and this section discusses which Essential Assumptions these tasks support.

5.4.1. Defining Validation

Validation can be thought of as the accumulation of evidence which assesses the claim that a mathematical function can predict a real physical quantity (Oberkampf and Roy 2010). Thus, validation is a never ending process as more evidence can always be obtained to bolster this claim. However, many times analysts will use the concept of validation in the past tense and claim that a specific function is validated. While this use is not consistent with the concept of validation as an accumulation of evidence, it does provide some insight into what the evidence must include.

By claiming that the validation for a function is complete (i.e., the function is validated) the analyst is claiming that no future evidence would suggest that the real physical quantity could not be predicted by the given mathematical function. The only way for such a claim to be complete, is if there is evidence for every aspect of the assumption that the mathematical function can predict the real physical quantity.

To better understand how validation fits into the Theoretical/Logical Framework, the Fundamental Question of Validation is generated based on a generic problem, that question is used generate the Fundamental Assumption of Validation, and that assumption is deconstructed into the set of Necessary Assumptions of Validation.

5.4.1.1. A Generic Validation Problem

For this generic validation problem, suppose that there is some real physical quantity ($\mathbf{q}$) which exists in the real universe ($\mathbb{U}$). Further suppose that an analyst wishes to predict that quantity using a physical function ($\mathbf{PF}$) which exists in some application space ($\mathbb{A}$). The application space can be the domain of the physical function, but is often some restricted domain.

The analyst would validate the predictions of the physical function by making a certain number ($\mathbf{N}$) of measurements ($\mathbf{m_i}$) in the measurement space ($\mathbb{M}$). The analyst could then calculate the error between these measurements and the predictions of the physical function ($\mathbf{e_i = m_i - PF_i}$) at similar points in the measurement and application spaces. Based on this set of measured errors ($\mathbf{E}$, where $\{\mathbf{e_1}, \mathbf{e_2}, \cdots \mathbf{e_N}\} \in \mathbf{E}$), the analyst would determine if the physical function can be used to predict the real world quantity in the application space. If the set of measured errors satisfies some criteria, the function is often considered to be validated. However, as mentioned above, this usage of the term validation assumes that no future evidence would contradict this finding.

For the purposes of definition, assume that the analyst could determine the error at every point in the application space. That is, instead of performing $\mathbf{N}$ measurements and predictions and obtaining the error sample, set $\mathbf{E}$, the analyst could perform $\mathbf{N}_{\infty\infty}$[18] measurements and predictions and obtain the total population of errors in the entire application space ($\boldsymbol{\mathcal{E}}$, where $\{\mathbf{e_1}, \mathbf{e_2}, \cdots \mathbf{e}_{\mathbf{N}_{\infty\infty}}\} \in \boldsymbol{\mathcal{E}}$). The definitions for the above quantities are restated in Table 5-26 for convenience.

[18] The double infinity is used to signify that the number of points in any fixed space is not only infinite, but uncountable.

Table 5-26: Definitions of Quantities in a Generic Validation Problem

Quantity	Definition
$\mathbf{q}$	The real physical quantity which exists in the real universe.
$\mathbb{U}$	The space the real physical quantity exists in. This can be thought of as the mathematical space which is made up of all the variables which can impact the value of **q.**
PF	The physical function which is used to predict the value **q**.
$\mathbb{A}$	The space over which the physical function will be applied to predict the predict the value **q**.
N	The number of measurements taken.
$\mathbf{m_i}$	The values of each measurement taken.
$\mathbb{M}$	The space over which the physical function will be applied to predict the predict the value **q**.
$\mathbf{e_i = m_i - PF_i}$	The difference between the measurement value and the predicted value of each measurement taken.
$\{\mathbf{e_1}, \mathbf{e_2}, \cdots \mathbf{e_N}\} \in \mathbf{E}$	The set of all measured errors.
$\mathbf{N_{\infty\infty}}$	The total number of points in the application space.
$\{\mathbf{e_1}, \mathbf{e_2}, \cdots \mathbf{e_{N_{\infty\infty}}}\} \in \mathcal{E}$	The population of all measured errors in the application space.

Using the concepts given above, the fundamental question of validation can be defined. The **Fundamental Question of Validation** is: *Can a real physical quantity which exists in the real universe be predicted by a physical function in some application space with adequate accuracy?* When a physical function is said to be validated, this question becomes the fundamental assumption of validation. The **Fundamental Assumption of Validation** is: *A real physical quantity exists in the physical universe and can be predicted by a physical function in some application space with adequate accuracy.*

Ideally, the Fundamental Assumption of Validation could be deconstructed into a set of Essential Validation Assumptions. However, this is not the case as a set of assumptions can be generated which are necessary for the Fundamental Assumption of Validation to be true, but it cannot be proven that this set is sufficient. The **Necessary Assumptions of Validation** are the assumptions which must be true if the Fundamental Assumption of Validation is true. While each assumption in the set of assumptions is necessary, it cannot be proven that the entire set is sufficient.

5.4.1.2. Determining the Necessary Assumptions of Validation

Declaring that a function is validated is making the Fundamental Assumption of Validation. This assumption can be deconstructed into the following seven Necessary

Assumptions of Validation. That is, if an analyst is calming that a function has been validated, that analyst is also calming that the following assumptions are true. Evidence should be provided which justifies each of these assumptions if a function is assumed to be validated. Additionally, this is not a complete set in that other necessary assumptions may be needed to form a sufficient set for the Fundamental Assumption of Validation.

The **First Necessary Assumption of Validation** is the assumption that the physical quantity which is predicted is a real physical quantity and actually exists (or appears to exist[19]) in the real universe. This assumption is almost never addressed as nearly every physical quantity in validation is familiar and almost always uncontested (e.g., temperature, pressure, flow, etc…). However, the history of science does contain physical quantities and phenomena which have been later proved not to exist (e.g., aether). This is more likely to be an issue with newly discovered physical quantities than those which are commonly used.

The **Second Necessary Assumption of Validation** is the assumption that the real physical quantity which is in the real universe could be predicted by some function in the application space. If it is not possible for any function in the application space to predict the real physical quantity, than the physical function cannot predict the real physical quantity as it is a function in the application space.

The **Third Necessary Assumption of Validation** is the assumption that the real physical quantity which exists in the real universe can be adequately approximated by the measured quantity in its measured space. Notice that this necessary assumption is almost identical to the Fundamental Assumption of Validation. Instead of assuming that the real physical quantity is predicted by the physical function, this assumption is assuming that the real physical quantity can be predicted (or approximated) by the measured data. In other words, at some point the analyst must assume that something can approximate the real physical quantity. This assumption is meant to remind analysts the real physical quantity is not the same thing as the measured value.

The **Fourth Necessary Assumption of Validation** is the assumption that the measured space is similar enough to the application space. While the predictions will always be made in the application space, the evidence for the validation is always from the measured space. The measured space includes all of the variables which impacted the experiment. The experimenter will be aware of most of these variables, but not all. As the application space is usually very similar to the measured space, any variables that were important to the experiment, but the experimenter was unaware of (which are still in the measured space), would be missing from the application space. If the application space is a subset of the measurement space, this

[19] There can be philosophical debate as to where quantities like temperature, pressure, and flow are real quantities or only perceived by humans. Therefore, the caveat is added to avoid this debate. If the quantity is real or not is immaterial to this argument, as long as the can be treated as real.

assumption is considered more reasonable, but if the measurement space is a subset of the application space, this is considered a major assumption and may not be defensible.

The **Fifth Necessary Assumption of Validation** is the assumption that the set of all predicted errors in the application space is an adequate representative sample from the population of all possible predicted errors. This assumption is usually addressed by demonstrating that a sufficient number of measurements have been taken (i.e., the value **N** is large enough). Saying that the sample of measured errors is a representative sample of the population of all possible errors means the following:

(1) Any additional measured errors would not significantly change the mean of the distribution of **E**.
(2) Any additional measured errors would not significantly change the variance of the distribution of **E**.
(3) Any additional measured errors would not significantly change the shape of the distribution of **E**.

As it is not possible to prove any of the above statements are true, an analyst will typically try to demonstrate that the elements in **E** can be described as selections from a known distribution. Thus, the elements in population $\boldsymbol{\mathcal{E}}$ could logically also be described by that known distribution.

The **Sixth Necessary Assumption of Validation** is the assumption that any possible predicted error in the application space is adequately independent of its location in the application space. This assumption generally has little to no supporting evidence. If the magnitude of the error is dependent on the location in the application space, then there is some hidden bias. Such information would be very important to an analyst, but testing for this information is extremely difficult. Even if an analyst wanted to justify this assumption, it is difficult. For example, if the application space contained two dimensions $\boldsymbol{X}$ and $\boldsymbol{Y}$, the analyst could determine that there is no correlation in the $\boldsymbol{X}$ or $\boldsymbol{Y}$ dimension separately, but it would be much more difficult for the analyst to show there was no correlation in the $\boldsymbol{XY}$ plane. This problem is only exacerbated as the number of dimension in the application space increases.

The **Seventh Necessary Assumption of Validation** is the assumption that the magnitude of the error in the population of all possible errors is below some required value. This assumption is usually supported by demonstrating that the maximum (or some statistic) from the measured error set is below some stated value.

5.4.1.3. Conclusions on Validation

The Necessary Assumptions of Validation (**VL**) are used to support the Fundamental Assumption of Validation (*A real physical quantity exists in the physical universe and*

can be predicted by a physical function in some application space). These assumptions can be used to demonstrate that a physical function is correctly predicting (or correct enough) a real physical quantity and are therefore used to support Essential Assumption EA-5 (*Each Coded Physical Function calculates the correct output for given input*). Each of these assumptions should be justified for every instance of a Coded Physical Function in the Theoretical/Logical Framework. The seven assumptions are given in Table 5-27.

Table 5-27: The Seven Necessary Assumption of Validation

ID	Assumption	Component of Interest
VL-1	The real physical quantity which is predicted is a real physical quantity and actually exists or appears to exist in the real universe.	**CPF**
VL-2	The real physical quantity which is in the real universe could be predicted by some function in the application space.	**CPF**
VL-3	The real physical quantity which is in the real universe can be adequately approximated by the measured quantity in its measured space.	**CPF**
VL-4	The measured space is similar enough to the application space.	**CPF**
VL-5	The set of all predicted errors in the application space is an adequate representative sample from the population of all possible predicted errors.	**CPF**
VL-6	Any possible predicted error in the application space is adequately independent of its location in the application space.	**CPF**
VL-7	The magnitude of the error in the population of all possible errors is below some required value.	**CPF**

5.4.2. Defining Verification

Verification can be thought of as the accumulation of evidence which assesses the claim that the solution to the mathematical functions represented in the simulation are correct (or correct enough) when compared with the true solution of those same functions. Thus, like validation, verification is an accumulation of evidence and a never ending process as more evidence can always be obtained to bolster these claims. However, also like validation, many times analysts will use the concept of verification in the past tense and claim that a specific function is verified. While this use is not consistent with the concept of verification as an accumulation of evidence, it does provide some insight into what the evidence must include.

Typically, a verification activity is thought of has having one of two focuses, to either verify the code or to verify the specific solution (Oberkampf and Roy 2010):

- Code Verification – The process of determining that the numerical algorithms are correctly implemented in the computer code and of identifying errors in the software (ASME 2006).
- Solution Verification – The process of determining the correctness of the input data, the numerical accuracy of the solution obtained, and the correctness of the output data for a particular simulation.

While different tasks must be performed for code verification and solution verification, the entire goal of these tasks and both types of verification is to demonstrate the mathematical functions represented in the simulation are correct (or correct enough) when compared with the true solution of those same functions. Thus, a common concept of verification is used in this section. To better understand how verification fits into the Theoretical/Logical Framework, the Fundamental Question of Verification is generated based on a generic problem, that question is used generate the Fundamental Assumption of Verification, and that assumption is deconstructed into the set of Necessary Assumptions of Verification.

5.4.2.1. A Generic Verification Problem

For the generic verification problem, suppose that there is a continuous function ($\mathbf{CF}$) which exists in the continuous space ($\mathbb{C}$). Further suppose that an analyst wishes to predict that continuous function using a discretized function ($\mathbf{DF}$) which exists in the discretized space ($\mathbb{D}$).

The analyst would verify the predictions of the discretized function by making a certain number ($\mathbf{N}$) of calculations in the discretized space. The analyst could then calculate the error between the continuous function and the discretized function

($\mathbf{e_i = CF_i - DF_i}$) at certain points in the discretized space. Based on the set of all calculated errors ($\{\mathbf{e_1, e_2, \cdots e_N}\} \in \mathbf{E}$), the analyst would determine if the discretized function has been verified in the discretized space.

For the purposes of definition, assume that the analyst could determine the error every point in the discretized space. That is, instead of comparing the continuous and discretized function at **N** points and obtaining the error sample, set **E**, the analyst could compare the continuous and discretized function $\mathbf{N_{\infty\infty}}$[20] points and obtain the total population of errors in the entire discretization space $\left(\{\mathbf{e_1, e_2, \cdots e_{N_{\infty\infty}}}\} \in \boldsymbol{\mathcal{E}}\right)$. These definitions for the above quantities are restated in Table 5-28 for convenience.

Table 5-28: Definitions of Quantities in a Generic Verification Problem

Quantity	Definition
$\mathbf{CF}$	The continuous physical function.
$\mathbb{C}$	The space of the continuous physical function.
$\mathbf{DF}$	The discretized physical function.
$\mathbb{D}$	The space of the discretized physical function.
$\mathbf{N}$	The number of points in the discretized space in which the discretized function is compared to the continuous function
$\mathbf{e_i = CF_i - DF_i}$	The calculated error which is defined as the difference between the continuous function and the discretized function.
$\{\mathbf{e_1, e_2, \cdots e_N}\} \in \mathbf{E}$	The set of all measured errors.
$\mathbf{N_{\infty\infty}}$	The total number of points in the discretization space.
$\left\{\mathbf{e_1, e_2, \cdots e_{N_{\infty\infty}}}\right\} \in \boldsymbol{\mathcal{E}}$	The population of all error in the discretization space.

Using the above definitions, the fundamental question of verification can be defined. The **Fundamental Question of Verification** is: *Can a continuous function which exists in a continuous space can be adequately predicted by a discretized function in a discretized space?* When a physical function is said to be verified, this becomes the fundamental assumption of verification. The **Fundamental Assumption of Verification** is: *A continuous function which exists in a continuous space can be adequately predicted by a discretized function in a discretized space*.

Ideally, the Fundamental Assumption of Verification could be deconstructed into a set of Essential Verification Assumptions. However, this is not the case as a set of assumptions can be generated which are necessary for the Fundamental Assumption of Verification to be true, but it cannot be proven that this set is sufficient. The **Necessary Assumptions of Verification** are the assumptions which must be true if

[20] The double infinity is used to signify that the number of points in any fixed space is not only infinite, but uncountable.

the Fundamental Assumption of Verification is true. While each assumption in the set of assumptions is necessary, it cannot be proven that the entire set is sufficient.

5.4.2.2. Determining the Necessary Assumptions of Verification

The following five Necessary Assumptions of Verification have been identified. Each of these assumptions must be true if the Fundamental Assumption of Verification is to be true. However, other necessary assumptions may be needed to form a sufficient set.

The **First Necessary Assumption of Verification** is the assumption that there are no mistakes or "bugs" in the source code or the execution of a simulation which can negatively impact the results of the simulation. While it may be impossible to prove that the source code and its execution are "bug" free, this is assumed when the simulation is trusted. Generally, the evidence which supports this assumption is based on the code's Quality Assurance program and any analysis which demonstrates a mathematical verification.

Often, "bugs" which are not captured by the mathematical verification are considered "bugs" which do not impact the results of the simulation. In other words, if the "bug" doesn't impact the results of the simulation, it does not matter. This is a necessary conclusion, but not one which is logically sound. The only logically sound conclusion which can be drawn from demonstrating that a simulation gives the correct mathematical output for a given set of inputs is that, if the simulation is given those specific inputs, it will result in correct mathematical output[21]. If the simulation is given any other set of inputs, it must be assumed that the results would be mathematically correct.

The **Second Necessary Assumption of Verification** is the assumption that the discretized space contains all dimensions from the continuous space of the mathematical function. This assumption is generally trivial as the discretized space is made from the continuous space.

The **Third Necessary Assumption of Verification** is the assumption that the discretization error is appropriately small. Demonstrating that the discretization error is appropriately small is the focus of Solution Verification. It is important to remember that the discretization error which can be calculated from the Taylor Series expansion which results in the discretized function is only a lower limit to what the error could be, but the error is likely to be much higher in practice. For example, a second order accurate method may really be first order accurate when used in a

[21] And even this is dependent the same non-deterministic values.

simulation. Approaches such as Richardson Extrapolation should be used to determine the true order of accuracy.

The **Fourth Necessary Assumption of Verification** is the assumption that the iteration error is appropriately small.

The **Fifth Necessary Assumption of Verification** is the assumption that the round-off error is appropriately small. The round-off error is usually controlled by the machine precision of the computer which is performing the simulation.

5.4.2.3. Conclusions on Verification

The Necessary Assumptions of Verification (**VR**) are used to support the Fundamental Assumption of Verification (*A continuous function which exists in a continuous space can be predicted by a discretized function in a discretized space*.). The justification of these assumptions can be used to demonstrate that a physical function has been discretized correctly and are used to support Essential Assumption EA-5.3 (*The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct*). Each of these assumptions should be justified for each instance of a Coded Physical Function in the Theoretical/Logical Framework. These six assumptions are given in Table 5-29.

Table 5-29: The Five necessary Assumption of Verification

ID	Assumption	Component of Interest
VR-1	There are no mistakes or "bugs" in the source code or the execution of a simulation.	Source Code
VR-2	The discretized space contains all dimensions from the continuous space.	**CPF**
VR-3	The discretization error is appropriately small.	**CPF**
VR-4	The iteration error is appropriately small.	**CPF**
VR-5	The round-off error is appropriately small.	**CPF**

5.4.3. Defining Uncertainty Quantification

Uncertainty quantification can be thought of as any evidence which supports the assumption that the statistical variability in the results of the simulation has been appropriately captured. Typically, assuming that the uncertainties of a simulation have been captured means one of the follow:

(1) The assumption that the variability of the inputs to the simulation is appropriately captured.
(2) The assumption that the variability of the results of each Coded Physical Function in the simulation due to the variability in the inputs is appropriately captured.
(3) The assumption that the variability of the results of each Coded Physical Function in the simulation due to any error in the Coded Physical Function itself is appropriately captured. That is, this assumption is a recognition that even given "true" input values, the Coded Physical Function will likely not calculate the true value of the predicted quantity, but some value close to that true value.

To better understand how uncertainty quantification fits into the Theoretical/Logical Framework, three Fundamental Questions of Uncertainty Quantification are generated based on a generic problem and these three questions are used to generate the three Fundamental Assumptions of Uncertainty Quantification. However, further deconstruction of these assumptions into sets of necessary assumptions is not developed here, but should be done in future work.

5.4.3.1. A Generic Uncertainty Quantification Problem

For this generic uncertainty quantification problem, suppose that there is a simulation which has some input $(\mathbf{I})$. This simulation is made up of $\mathbf{N}$ Coded Physical Functions, the i-th being represented by $\mathbf{CPF_i}$. Each Coded Physical Function has some input $\left(\mathbf{I_{CPF,i}}\right)$ and produces some output $\left(\mathbf{O_{CPF,i}}\right)$. These definitions for the above quantities are restated in Table 5-30 for convenience.

Table 5-30: Definitions of Quantities in a Generic Uncertainty Quantification Problem

Quantity	Definition
$\mathbf{I}$	Input used for the simulation.
$\mathbf{N}$	Number of Coded Physical Functions in the simulation.
$\mathbf{CPF_i}$	The i-th Coded Physical Function in the simulation.
$\mathbf{I_{CPF,i}}$	The inputs to the i-th Coded Physical Function in the simulation.
$\mathbf{O_{CPF,i}}$	The outputs from the i-th Coded Physical Function in the simulation.

Using the above concepts, three **Fundamental Questions of Uncertainty Quantification** can be defined as follows:

(1) The **Frist Fundamental Question of Uncertainty Quantification** is: *How much can the inputs to the simulation vary?*
(2) The **Second Fundamental Question of Uncertainty Quantification** is: *How much can the results of each Coded Physical Function in the simulation vary due to variability in the input to that function?*
(3) The **Third Fundamental Question of Uncertainty Quantification** is: *How much can the results of each Coded Physical Function in the simulation vary due to variability in the function itself?*

When the uncertainty of a simulation is assumed to be quantified, these three questions become the **Fundamental Assumptions of Uncertainty Quantification**:

(1) The **Frist Fundamental Assumption of Uncertainty Quantification** is: *The variability of the inputs of simulation are appropriately captured*.
(2) The **Second Fundamental Assumption of Uncertainty Quantification** is: *The variability of the results of each Coded Physical Function in the simulation due to the variability in the inputs is appropriately captured*.
(3) The **Third Fundamental Assumption of Uncertainty Quantification** is: *The variability of the results of each Coded Physical Function in the simulation due to the variability of the function itself is appropriately captured*.

Decomposition of these fundamental assumptions is beyond the scope of this dissertation.

5.4.3.2. Conclusions on Uncertainty Quantification

The Fundamental Assumptions of Uncertainty Quantification (FAUQ) are given below in Table 5-31.

Table 5-31: Fundamental Uncertainty Quantification Sets

#	Assumption
FAUQS-1	The variability of the inputs of the simulation is appropriately captured.
FAUQS-2	The variability of the results of each Coded Physical Function in the simulation due to the variability in the inputs is appropriately captured.
FAUQS-3	The variability of the results of each Coded Physical Function in the simulation due to the variability of the function itself is appropriately captured.

5.4.4. Summary on Verification and Validation the Theoretical/Logical Framework

Verification and Validation (V&V) provide justification for the assumption that the Coded Physical Functions are correct. Verification provides direct evidence which supports assumption EA-5.3 (*The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct*).

Validation provides direct evidence which supports assumption EA-5 (*Each Coded Physical Function calculates the correct output for given input*). This also includes the assumptions which are a result of the deconstruction of this Essential Assumption:

- EA-5.1 (*The original form of the physical equation* (OPE) *is correct*)
- EA-5.2 (*The derivations and assumptions used to obtain the final physical equation from the original physical equation are correct*)
- EA-5.3 (*The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct*)

However, the power of V&V lies in the fact that while they provide direct evidence for these Essential Assumptions, they provide secondary evidence for all of the Essential Assumptions. In other words, if an analyst can demonstrate that a specific Coded Physical Function is calculating the correct output for a given input (i.e., demonstrating EA-5 is a valid assumption), that analyst has probably also provided evidence to support the assumption that the input to that specific Coded Physical Function has been correctly chosen (EA-4) and also evidence to suggest that the input to any parents of the Coded Physical Function are also correctly selected.

The link between V&V and the other Essential Assumptions is obvious, but it is not clearly defined. V&V would likely provide evidence for some of the Essential Assumptions, but not all of them or all instances of them.

5.5. **Conclusions on Developing Maturity Assessment Sets**

The thought experiment discussed in this chapter provides a foundation for understanding maturity assessments; they are a collection of reasons why a scientific computer simulation should be trusted which can be grouped because they only differ in degree (i.e., in maturity level). They represent the set of all possible answers to the same question, and that question is the maturity attribute.

Creating assessment sets, especially important or practical assessment sets, is a very complicated process. However this process can be separated into evolving a current assessment set or by creating an assessment set from some concept. Evolving assessment sets can either be done to gain information and create new assessment sets (Divide and Distill) or can be done to consolidate information and summarize multiple assessment sets (Aggregate and Abridge).

The assessment sets used in the current frameworks are mostly focused on assessing the performance of VV&UQ activities for the specific simulation. The concepts of VV&UQ can be further understood by examining the Fundamental Assumptions of each. These assumptions can be further understood by deconstructing them into their Necessary Assumptions. These Necessary Assumptions, when justified, can then be used to support an Essential Assumption.

The link between UQ and the Theoretical/Logical Framework is a subject for future work, but the link between V&V and the Theoretical/Logical Framework is very clear. V&V represent much of the evidence which support the most important Essential Assumption, EA-5 (T*he Coded Physical Function calculates the correct output for the given input*).

The first round of assessment sets which are used in the Theoretical/Logical Framework are provided in Chapter 6 and these sets are used to assess a simple homework problem in Appendix A.

6. Maturity Assessment Sets of the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations

This chapter provides the first round of the maturity assessment sets of the Theoretical/Logical Framework. These assessment sets were created by focusing on the Essential Assumptions derived in Chapter 4 and the Necessary Assumptions of Verification and Validation derived in Chapter 5.

The first section is the assessment sets created from the Necessary Assumption of Validation. The second section is the assessment sets created from the Necessary Assumption of Verification. The third section is the assessment sets created for Peer Review. The fourth section is the assessment sets created form the Essential Assumptions. Finally, the fifth summarizes of all of the assessment sets and provides the first version of the Theoretical/Logical Framework.

6.1. **Validation**

The following section provides the maturity assessments which can be used as supporting evidence for each of the Necessary Assumptions of Validation. These assumptions were originally given in Table 5-27. A copy of that table is given below for convince.

Table 5-27: The Seven Necessary Assumption of Validation

ID	Assumption	Component of Interest
VL-1	The real physical quantity which is predicted is a real physical quantity and actually exists or appears to exist in the real universe.	**CPF**
VL-2	The real physical quantity which is in the real universe could be predicted by some function in the application space.	**CPF**
VL-3	The real physical quantity which is in the real universe can be adequately approximated by the measured quantity in its measured space.	**CPF**
VL-4	The measured space is similar enough to the application space.	**CPF**
VL-5	The set of all predicted errors in the application space is an adequate representative sample from the population of all possible predicted errors.	**CPF**
VL-6	Any possible predicted error in the application space is adequately independent of its location in the application space.	**CPF**
VL-7	The magnitude of the error in the population of all possible errors is below some required value.	**CPF**

6.1.1. 1st Necessary Assumption of Validation

Maturity Attribute		How common is the prediction of the physical quantity?					
ID	**v.**	VL-1-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

Maturity Attribute		How common is the physical quantity predicted under these specific conditions?					
ID	**v.**	VL -1-2	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

6.1.2. 2nd Necessary Assumption of Validation

Maturity Attribute		How common is the prediction of the physical quantity using the given set of dependent variables?					
ID	**v.**	VL-2-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

6.1.3. 3rd Necessary Assumption of Validation

Maturity Attribute		How common is this physical quantity measurement made using the given method?					
ID	**v.**	VL-3-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-3					
Maturity Assessment Set							
-1		Does not apply					
0		Don't know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

Maturity Attribute		How accurate is the measurement generally considered?					
ID	**v.**	VL-3-2	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-3					
Maturity Assessment Set							
-1		Does not apply					
0		Don’t know.					
1		Measurement is more of a guess of the value of the real physical quantity.					
2		Measurement contains some error which is unquantified.					
3		Measurement contains some error which is quantified.					
4		Measurement is considered to be completely accurate.					

Maturity Attribute		How much computation is needed to obtain the measured value?					
ID	**v.**	VL-3-3	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-3					
Maturity Assessment Set							
-1		Does not apply					
0		Don’t know.					
1		Measurement value requires significant computation.					
2		Measurement value requires some computation.					
3		Measurement value requires an insignificant amount of computation and is often considered a direct measurement.					
4		Measurement value requires no computation and is a direct measurement.					

Maturity Attribute		How is the uncertainty on the measurement determined?					
ID	**v.**	VL-3-4	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-3					
Maturity Assessment Set							
-1		Does not apply					
0		Don't know.					
1		No uncertainty is determined.					
2		Measurement uncertainty is obtained from a reference.					
3		Measurement uncertainty is obtained through some testing.					
4		Measurement uncertainty is obtained through a rigorous testing process.					
5		There measurement is exact and has no uncertainty.					

6.1.4. 4th Necessary Assumption of Validation

Maturity Attribute		How common is this physical quantity measured in the given application space?					
ID	**v.**	VL-4-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-4					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

6.1.5. 5th Necessary Assumption of Validation

Maturity Attribute		How thoroughly sampled is the application space?					
ID	**v.**	VL-5-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The application space is sparsely sampled.					
2		The application space is highly sampled in certain regions.					
3		The application space is highly sampled in certain regions which are the most important.					
4		The application space is highly sampled.					

Maturity Attribute		How widely varying are the measurement values in the application space?					
ID	**v.**	VL-5.2	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The measurement values are widely varying in the application space.					
2		The measurement values follow a general trend in the application space.					
3		The measurement values display an obvious and gentle trend in the application space.					

Maturity Attribute		How widely varying are the predicted values over the application space?					
ID	**v.**	VL-5-3	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The predicted values are widely varying in the application space.					
2		The predicted values follow a general trend in the application space.					
3		The predicted values display an obvious and gentle trend in the application space.					

Maturity Attribute		Is there evidence to suggest that the error sample has the same mean as the error population?					
ID	**v.**	VL-5-4	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		All of the errors were used to determine the error mean.					
2		Some small portion of the errors was reserved and the mean of that reserved set was compared to the error mean.					
3		A large portion of the errors was reserved and the mean of that reserved set was compared to the error mean.					

Maturity Attribute		Is there evidence to suggest that the error sample has the same variance as the error population?					
ID	**v.**	VL-5-5	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		All of the errors were used to determine the error variance.					
2		Some small portion of the errors was reserved and the mean of that reserved set was compared to the error variance.					
3		A large portion of the errors was reserved and the mean of that reserved set was compared to the error variance.					

Maturity Attribute		Is there evidence to suggest that the error sample has the same distribution as the error population?					
ID	**v.**	VL-5-6	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-5					
Maturity Assessment Set							
(No maturity levels are known for this criteria)							

6.1.6. 6th Necessary Assumption of Validation

No maturity assessments are known for this assumption.

6.1.7. 7th Necessary Assumption of Validation

Maturity Attribute		Is the magnitude of the error small compared with the measured and predicted values?					
ID	**v.**	VL-7-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-7					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The magnitude of the error is on the same order or greater than the measured and predicted values.					
2		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 2.					
3		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 5.					
4		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 10.					
5		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 20.					
6		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 50.					
7		The magnitude of the error is on the smaller than the measured and predicted values by at least a factor of 100.					

6.2. **Verification**

The following section provides the maturity assessments which can be used as supporting evidence for each of the Necessary Assumptions of Verification. These assumptions were originally given in Table 5-29. A copy of that table is given below for convince.

Table 5-29: The Five necessary Assumption of Validation

ID	Assumption	Component of Interest
VR-1	There are no mistakes or "bugs" in the source code or the execution of a simulation.	Source Code
VR-2	The discretized space contains all dimensions from the continuous space.	**CPF**
VR-3	The discretization error is appropriately small.	**CPF**
VR-4	The iteration error is appropriately small.	**CPF**
VR-5	The round-off error is appropriately small.	**CPF**
VR-6	The magnitude of the error in the population of all possible errors is below some required value.	**CPF**

6.2.1. 1st Necessary Assumption of Verification

No maturity assessments are currently given for this assumption.

6.2.2. 2nd Necessary Assumption of Verification

Maturity Attribute		Does the discretized space bound the continuous space?					
ID	**v.**	VR-2-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VR-2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Some dimensions in the continuous space are not represented in the discretized space.					
2		All important dimensions in the continuous space are represented in the discretized space.					
3		All dimensions in the continuous space are represented in the discretized space.					

6.2.3. 3rd Necessary Assumption of Verification

Maturity Attribute		How is the discretization error determined?					
ID	**v.**	VR -3-1	0	**Parent**	**v.**		
Reference		Current					
Assumption		VL-3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Order of accuracy is determined from Taylor Series exapnsion.					
2		Richardson Extrapolation is used.					
3		Richardson Extrapolation is used with all 5 assumption met.					

6.2.4. 4th Necessary Assumption of Verification

No maturity assessments are currently given for this assumption.

6.2.5. 5th Necessary Assumption of Verification

No maturity assessments are currently given for this assumption.

6.3. **Peer Review Attributes**

The following are maturity assessments which can be used as supporting evidence for Peer Review.

Maturity Attribute		How thorough was the peer review?					
ID	**v.**	PR-1	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review took less than 1% of the time it took to complete the original work.					
2		The peer review took less than 5% of the time it took to complete the original work.					
3		The peer review took less than 10% of the time it took to complete the original work.					
4		The peer review took less than 20% of the time it took to complete the original work.					
5		The peer review took more than 20% of the time it took to complete the original work.					

Maturity Attribute		How independent was the peer review?					
ID	**v.**	PR-2	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review was performed by the same analyst who did the original work.					
2		The peer review was performed by an analyst(s) who shared the same boss as the analyst who did the original work.					
3		The peer review was performed by an analyst(s) in the same organization as the analyst who did the original work or by a contractor of that company.					
4		The peer review was performed by an analyst(s) which was independent of the company of the analyst who did the original work.					

Maturity Attribute		How important was the peer review?					
ID	**v.**	PR-3	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review had little importance and was more of a formality.					
2		The peer review is only used to discover major problems and minor discrepancies and issues are mostly ignored.					
3		The peer reviewer has the ability to not allow the use of this simulation, but this power is rarely exercised.					
4		The peer reviewer has the ability to not allow the use of this simulation and this power is commonly exercised.					

Maturity Attribute		How detailed was the peer review?					
ID	**v.**	PR-4	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review consisted of double checking certain values.					
2		The peer review consisted of double checking all values.					
3		The peer review consisted of double checking all values using a formal structure.					

Maturity Attribute		How different was the peer review?					
ID	**v.**	PR-5	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review consisted of reviewing the work.					
2		The peer review consisted of performing the same work using the same methods to ensure a similar result.					
3		The peer review consisted of performing the same work using different methods to ensure a similar result.					

Maturity Attribute		How many individuals were involved in the peer review?					
ID	**v.**	PR-6	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer review was performed by one person.					
2		The peer review was performed by a team of 3 or less.					
3		The peer review was performed by a team of 5 or less.					
4		The peer review was performed by a team of 10 or less.					
5		The peer review was performed by a team of 20 or less.					
6		The peer review was performed by a team of more than 20.					

Maturity Attribute		What was the knowledge base of each peer reviewer?					
ID	**v.**	PR-7	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer reviewer has a bachelor’s degree in a relevant field.					
2		The peer reviewer has a master’s degree in a relevant field.					
3		The peer reviewer has a PhD degree in a relevant field.					
4		The peer reviewer has performed significant research in the particular aspect they are peer reviewing equivalent to a thesis.					

Maturity Attribute		What was the historical base of each peer reviewer?					
ID	**v.**	PR-8	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer reviewer has less than 1 year of experience peer reviewing this specific aspect.					
2		The peer reviewer has less than 2 years of experience peer reviewing this specific aspect.					
3		The peer reviewer has less than 5 years of experience peer reviewing this specific aspect.					
4		The peer reviewer has less than 10 years of experience peer reviewing this specific aspect.					
5		The peer reviewer has less than 20 years of experience peer reviewing this specific aspect.					
6		The peer reviewer has more than 20 years of experience peer reviewing this specific aspect.					

Maturity Attribute		What was the experience base of each peer reviewer?					
ID	**v.**	PR-9	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer reviewer has never performed the action they are peer reviewing.					
2		The peer reviewer has only performed the action they are peer reviewing occasionally.					
3		The peer reviewer has performed the action they are peer reviewing on many occasions.					
4		The peer reviewer has spent a career performing the action they are peer reviewing.					

Maturity Attribute		How involved was each peer reviewer?					
ID	**v.**	PR-10	0	**Parent**	**v.**		
Reference		Current					
Focus		Entire Peer Review					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The peer reviewer was only involved in a minimal capacity.					
2		The peer reviewer was during some of the peer review.					
3		The peer reviewer was during most of the peer review.					
4		The peer reviewer was during all of the peer review.					

Maturity Attribute		Did peer reviewer believe they had enough resources to perform an appropriate peer review?					
ID	**v.**	PR-11	0	**Parent**	**v.**		
Reference		Current					
Focus		Each Peer Reviewer					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The amount of resources used for the peer review was significantly lacking.					
2		The amount of resources used for the peer review was somewhat lacking.					
3		The amount of resources used for the peer review was adequate.					
4		The amount of resources used for the peer review was more than adequate.					

Maturity Attribute		Did the peer reviewer believe their feedback was understood?					
ID	**v.**	PR-12	0	**Parent**	**v.**		
Reference		Current					
Focus		Each Peer Reviewer					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		Feedback was ignored or discounted.					
2		Feedback was not understood.					
3		Feedback was mostly understood.					
4		Feedback was fully understood.					

Maturity Attribute		Did the peer reviewer believe their feedback was handled appropriately?					
ID	**v.**	PR-13	0	**Parent**	**v.**		
Reference		Current					
Focus		Each Peer Reviewer					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		None of my feedback was handled appropriately.					
2		Some feedback was handled appropriately.					
3		Most feedback was handled appropriately.					
4		All feedback was handled appropriately.					

6.4. **Essential Assumptions**

The following section provides the maturity assessments which can be used as supporting evidence for the Essential Assumptions. These assumptions were originally given in Table 4-8. A copy of that table is given below for convince.

Table 4-8: Current Set of Essential Assumptions

#	Statement	Component of Interest
EA-1	The input provided for the Complete Set of Codes is correct (or correct enough).	**CSC**
EA-2	The input selected for each Computer Code is correctly selected (or correct enough) from the input to the Complete Set of Codes and the output from the Computer Codes in the Complete Set of Codes.	**CC**
EA-3	The input selected for each Coded Group is correctly selected (or correct enough) from the input to the Computer Code and the output from the Coded Groups in the Computer Code.	**CG**
EA-4	The input selected for each Coded Physical Function is correctly selected (or correct enough) from the input to the Coded Group and the output from the Coded Physical Functions in the Coded Group.	**CPF**
EA-5.1	The original form of the physical equation (OPE) is correct.	**CPF**
EA-5.2	The derivations and assumptions used to obtain the final physical equation (FPE) from the original physical equation are correct.	**CPF**
EA-5.3	The discretization and functionalization used to obtain the Coded Physical Function from the final physical equation are correct.	**CPF**
EA-6	The output from the Coded Group is correctly selected (or correct enough) from the output from the Coded Physical Functions in the Computer Code.	**CG**
EA-7	The output from the Computer Code is correctly selected (or correct enough) from the output from the Coded Groups in the Computer Code.	**CC**
EA-8	The output from the Complete Set of Codes is correctly selected (or correct enough) from the output from the Computer Codes in the Complete Set of Codes.	**CSC**
EA-9	The specific output selected (or correct enough) from all of the output from the Complete Set of Codes must be selected correctly.	**CSC**

Component – this identifies the component in the Hierarchy of Scientific Computer Simulation Components which is being assessed.

ID – Notice that the ID of each of the following Assessment Sets is identified

6.4.1. Maturity Tables for Essential Assumption 1

Maturity Attribute		How simple is the input?					
ID	**v.**	EA-1-1	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes					
Assumption		EA-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Input is a specification of over 500 values.					
2		Input is a specification of fewer than 500 values.					
3		Input is a specification of fewer than 100 values.					
4		Input is a specification of fewer than 10 values.					
5		Input is a specification of 1 value.					

Maturity Attribute		What is the origin of the input?					
ID	**v.**	EA-1-2	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes, each input					
Assumption		EA-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		Input is the best guess of an analyst or a group of analysts.					
2		Input is obtained from historical precedent.					
3		Input is obtained from a formal written process.					

Maturity Attribute		How detailed is the input?					
ID	**v.**	EA-1-3	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes, each input					
Assumption		EA-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		Input is free form.					
2		Input is chosen from a large set (> 10 choices).					
3		Input is chosen from a small set (< 10 choices).					

Maturity Attribute		Was the input peer reviewed?					
ID	**v.**	EA-1-4	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes, each input					
Assumption		EA-1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" input was peer reviewed.					
4		This input was peer reviewed.					

6.4.2. Maturity Tables for Essential Assumption 2

Maturity Attribute		How was the selection of the input to each Computer Code verified?					
ID	**v.**	EA-2-1	0	**Parent**	**v.**		
Reference		Current					
Component		Computer Code, each input					
Assumption		EA-2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The input selection was verified through some general verification process.					
2		The input selection was verified through some process which made use of this specific Computer Code.					
3		The input selection was verified through some process which made use of this specific Computer Code and this specific input.					

Maturity Attribute		Was the input peer reviewed?					
ID	**v.**	EA-2-2	0	**Parent**	**v.**		
Reference		Current					
Component		Computer Code, each input					
Assumption		EA-2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" input was peer reviewed.					
4		This input was peer reviewed.					

6.4.3. Maturity Tables for Essential Assumption 3

Maturity Attribute		How was the selection of the input to each Coded Group verified?					
ID	**v.**	EA-3-1	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Group, each input					
Assumption		EA-3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The input selection was verified through some general verification process.					
2		The input selection was verified through some process which made use of this specific Coded Group.					
3		The input selection was verified through some process which made use of this specific Coded Group and this specific input.					

Maturity Attribute		Was the input peer reviewed?					
ID	**v.**	EA-3-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Group, each input					
Assumption		EA-3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" input was peer reviewed.					
4		This input was peer reviewed.					

6.4.4. Maturity Tables for Essential Assumption 4

Maturity Attribute		How was the selection of the input to each Coded Physical Function verified?					
ID	**v.**	EA-4-1	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Function, each input					
Assumption		EA-4					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The input selection was verified through some general verification process.					
2		The input selection was verified through some process which made use of this specific Coded Physical Function.					
3		The input selection was verified through some process which made use of this specific Coded Physical Function and this specific input.					

Maturity Attribute		Was the input peer reviewed?					
ID	**v.**	EA-4-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Function, each input					
Assumption		EA-4					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" input was peer reviewed.					
4		This input was peer reviewed.					

6.4.5. Maturity Tables for Essential Assumption 5.1

Maturity Attribute		What is the origin of each original physical equation?					
ID	**v.**	EA-5.1-1	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Internet, Personal Notes, or Memory					
2		Text book or journal article					
3		Genesis Equation					

Maturity Attribute		Are the assumptions of the original physical equation satisfied?					
ID	**v.**	EA-5.1-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Assumptions of the Original Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Some of the assumptions are satisfied.					
2		Most of the assumptions are satisfied.					
3		All of the assumptions are satisfied.					

Maturity Attribute		How are the assumptions satisfied?					
ID	**v.**	EA-5.1-3	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Assumptions of the Original Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The assumption is assumed to be satisfied, but no justification is given.					
2		The assumption is generally considered to be satisfied under the given circumstances.					
3		Administrative controls ensure the assumption will be satisfied.					
4		The assumption is satisfied by the coding itself and the simulation will result in an error message if the assumption is no longer satisfied.					
5		The assumption is universally assumed to be satisfied in almost all situations of interest.					
6		The assumption is satisfied by the coding itself and the simulation will not produce a result (or will stop) if the assumption is not satisfied.					

Maturity Attribute		How common is the use of the original physical equation for this type of simulation?					
ID	**v.**	EA-5.1-4	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Original Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

Maturity Attribute		Was the original physical function validated?					
ID	**v.**	EA-5.1-5	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Original Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		No validation was performed.					
2		A general validation was performed.					
3		A “nearby” original physical equation was validated.					
4		This original physical equation was validated.					

Maturity Attribute		Was the original physical equation peer reviewed?					
ID	**v.**	EA-5.1-6	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Assumptions of the Original Equation					
Assumption		EA-5.1					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A “nearby” original physical equation was peer reviewed.					
4		This original physical equation was peer reviewed.					

6.4.6. Maturity Tables for Essential Assumption 5.2

Maturity Attribute		Are the derivation steps from the original physical equation to the final physical equation clearly defined?					
ID	**v.**	EA-5.2-1	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The derivation is only qualitatively discussed.					
2		Each major derivation step is given.					
3		Each derivation step is given.					
4		Each step in the derivation process is clearly defined using the format (or one similar) to that used in Appendix B.					

Maturity Attribute		Is each assumption of the final physical equation satisfied?					
ID	**v.**	EA-5.2-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Assumptions of the Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		Some of the assumptions are satisfied.					
2		Most of the assumptions are satisfied.					
3		All of the assumptions are satisfied.					

Maturity Attribute		How is the assumption satisfied?					
ID	**v.**	EA-5.2-3	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Assumptions of the Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		The assumption is assumed to be satisfied, but no justification is given.					
2		The assumption is generally considered to be satisfied under the given circumstances.					
3		Administrative controls ensure the assumption will be satisfied.					
4		The assumption is satisfied by the coding itself and the simulation will result in an error message if the assumption is no longer satisfied.					
5		The assumption is universally assumed to be satisfied in almost all situations of interest.					
6		The assumption is satisfied by the coding itself and the simulation will not produce a result (or will stop) if the assumption is not satisfied.					

Maturity Attribute		How common is the use of the final physical equation for this type of simulation?					
ID	**v.**	EA-5.2-4	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		First of a kind use.					
2		Rarely used.					
3		Commonly used.					
4		De facto standard.					

Maturity Attribute		Was the final physical function validated?					
ID	**v.**	EA-5.2-5	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		No validation was performed.					
2		A general validation was performed.					
3		A “nearby” final physical equation was validated.					
4		This final physical equation was validated.					

Maturity Attribute		Was the derivation from the original physical equation to the final physical equation peer reviewed?					
ID	**v.**	EA-5.2-6	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation, Final Equation					
Assumption		EA-5.2					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A “nearby” derivation was peer reviewed.					
4		This derivation was peer reviewed.					

6.4.7. Maturity Tables for Essential Assumption 5.3

Maturity Attribute		How was the overall verification for the simulation performed?					
ID	**v.**	EA-5.3-1	0	**Parent**	**v.**		
Reference		Current					
Component		Simulation					
Assumption		EA-5.3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don’t know.					
1		No verification was performed.					
2		The verification was a comparison to another simulation.					
3		The verification was a comparison to test data or benchmarks.					
4		The verification was a comparison to a known solution.					
5		The verification was a comparison to a known solution using Method of Manufactured Solutions or a similar process.					

Maturity Attribute		Was the final physical function validated?					
ID	**v.**	EA-5.3-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation					
Assumption		EA-5.3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No validation was performed.					
2		A general validation was performed.					
3		A "nearby" final physical equation was validated.					
4		This final physical equation was validated.					

Maturity Attribute		Was the Coded Physical Function peer reviewed?					
ID	**v.**	EA-5.3-3	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Physical Equation					
Assumption		EA-5.3					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" final physical equation was peer reviewed.					
4		This final physical equation was peer reviewed.					

6.4.8. Maturity Tables for Essential Assumption 6

Maturity Attribute		How was the selection of each output from each Coded Group verified?					
ID	**v.**	EA-6-1	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Group, Output					
Assumption		EA-6					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The output selection was verified through some general verification process.					
2		The output selection was verified through some process which made use of this specific Coded Group.					
3		The output selection was verified through some process which made use of this specific Coded Group and this specific output.					

Maturity Attribute		Was the output peer reviewed?					
ID	**v.**	EA-6-2	0	**Parent**	**v.**		
Reference		Current					
Component		Coded Group, Output					
Assumption		EA-6					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" output was peer reviewed.					
4		This output was peer reviewed.					

6.4.9. Maturity Tables for Essential Assumption 7

Maturity Attribute		How was the selection of each output from each Computer Code verified?					
ID	**v.**	EA-7-1	0	**Parent**	**v.**		
Reference		Current					
Component		Computer Code, Output					
Assumption		EA-7					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		The output selection was verified through some general verification process.					
2		The output selection was verified through some process which made use of this specific Computer Code.					
3		The output selection was verified through some process which made use of this specific Computer Code and this specific output.					

Maturity Attribute		Was the output peer reviewed?					
ID	**v.**	EA-7-2	0	**Parent**	**v.**		
Reference		Current					
Component		Computer Code, Output					
Assumption		EA-7					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" output was peer reviewed.					
4		This output was peer reviewed.					

6.4.10. Maturity Tables for Essential Assumption 8

Maturity Attribute		Was the output peer reviewed?					
ID	**v.**	EA-8-1	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes, Output					
Assumption		EA-8					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" output was peer reviewed.					
4		This output was peer reviewed.					

6.4.11. Maturity Tables for Essential Assumption 9

Maturity Attribute		Was the selection of the output peer reviewed?					
ID	**v.**	EA-9-1	0	**Parent**	**v.**		
Reference		Current					
Component		Complete Set of Codes, selection of specific Output					
Assumption		EA-9					
Maturity Assessment Set							
-1		Does not apply.					
0		Don't know.					
1		No peer review was performed.					
2		A general peer review was performed.					
3		A "nearby" output was peer reviewed.					
4		This output was peer reviewed.					

6.5. Summary

The following are a summary of the maturity assessments used in the Theoretical/Logical Framework, the levels of each assessment, and the number of times each assessment is used.

6.5.1. Validation Summary

Table 6-1: Summary of Validation Assessments of the Theoretical/Logical Framework

Assumption	Attribute	Levels
VL-1	How common is the prediction of the physical quantity?	0 – 4
	How common is the physical quantity predicted under these specific conditions?	0 – 4
VL -2	How common is the prediction of the physical quantity using the given set of dependent variables?	0 – 4
VL -3	How common is this physical quantity measured in the given method?	0 – 4
	How accurate is the measurement generally considered?	0 – 4
	How much computation is needed to obtain the measured value?	0 – 4
	How is the uncertainty on the measurement determined?	0 – 5
VL -4	How common is this physical quantity measured in the given application space?	0 – 4
VL -5	How thoroughly sampled is the application space?	0 – 4
	How widely varying are the measurement values in the application space?	0 – 3
	How widely varying are the predicted values in the application space?	0 – 3
	Is there evidence to suggest that the error sample has the same mean as the error population?	0 – 3
	Is there evidence to suggest that the error sample has the same variance as the error population?	0 – 3
	Is there evidence to suggest that the error sample has the same distribution as the error population?	?
VL -6	Is there evidence to suggest that the magnitude of the error in the sample is independent of its location in the application space?	?
VL -7	Is the magnitude of the error small compared with the measured and predicted values?	0 – 7

6.5.2. Verification Summary

Table 6-2: Summary of Verification Assessments of the Theoretical/Logical Framework

Assumption	Attribute	Levels
VR-1	Is there evidence to suggest that there are no mistakes or bugs in the source code?	?
VR -2	Does the discretized space bound the continuous space?	0 – 3
VR -3	How is the discretization error determined?	0 – 3
VR -4	Is there evidence to suggest that the iteration error is appropriately small?	?
VR -5	Is there evidence to suggest that the round-off error is appropriately small?	?

6.5.3. Peer Review Summary

Table 6-3: Summary of the Peer Review Assessments of the Theoretical/Logical Framework

Attribute	Levels
How thorough was the peer review?	0 – 5
How independent was the peer reviewer?	0 – 4
How important was the peer review?	0 – 4
How detailed was the peer review?	0 – 3
How different was the peer review?	0 – 3
How many individuals were involved in the peer review?	0 – 6
What was the knowledge base of each peer reviewer?	0 – 4
What was the historical base of each peer reviewer?	0 – 6
What was the experience base of each peer reviewer?	0 – 4
How involved was each peer reviewer?	0 – 4
Did peer reviewer believe they had enough resources to perform an appropriate peer review?	0 – 4
Did the peer reviewer believe their feedback was understood?	0 – 4
Did the peer reviewer believe their feedback was handled appropriately?	0 – 4

6.5.4. Summary of Framework

Table 6-4: Summary of Theoretical/Logical Framework Maturity Assessments

Assumption	Attribute	Component of Interest	Levels
EA-1	How simple is the set of all inputs	CSC	0 – 5
	What is the origin of each input?	I_{CSC}	0 – 3
	How detailed is each input?	I_{CSC}	0 – 3
	Was the input peer reviewed?	I_{CSC}	0 – 4
	Peer Review Assessments	I_{CSC}	many
EA-2	How was the selection of the each input to each Computer Code verified?	I_{CC}	0 – 3
	Was the input peer reviewed?	I_{CC}	0 – 4
	Peer Review Assessments	I_{CC}	many
EA-3	How was the selection of each input to each Coded Group verified?	I_{CG}	0 – 3
	Was the input peer reviewed?	I_{CG}	0 – 4
	Peer Review Assessments	I_{CG}	many
EA-4	How was the selection of the each input to each Coded Physical Function verified?	I_{CPF}	0 – 3
	Was the input peer reviewed?	I_{CPF}	0 – 4
	Peer Review Assessments	I_{CPF}	many
EA-5	See Table Below	CPF	many
EA-6	How was the selection of each output from each Coded Group verified?	O_{CG}	0 – 3
	Was the output peer reviewed?	O_{CG}	0 – 4
	Peer Review Assessments	O_{CG}	many
EA-7	How was the selection of each output from each Computer Code verified?	O_{CC}	0 – 3
	Was the output peer reviewed?	O_{CC}	0 – 4
	Peer Review Assessments	O_{CC}	many
EA-8	Was the output peer reviewed?	O_{CSC}	0 – 4
	Peer Review Assessments	O_{CSC}	many
EA-9	Was the selection of the output peer reviewed?	O_{CSC}	0 – 4
	Peer Review Assessments	O_{CSC}	many

Table 6-5: Summary of Theoretical/Logical Framework Maturity Assessments (Part 2)

Assumption	Attribute	Component of Interest	Levels
EA-5.1	What is the origin of each original physical equation?	CPF	0 – 3
	Is each assumption of the original physical equation satisfied?	CPF	0 – 3
	How is the assumption satisfied?	CPF	0 – 6
	How common is the use of the original physical equation for this type of simulation?	CPF	0 – 4
	Was the original physical equation validated?	CPF	0 – 4
	Was the original physical equation peer reviewed?	CPF	0 – 4
	Peer Review Assessments	CPF	many
	Validation Assessments	CPF	many
EA-5.2	Are the derivation steps from the original physical equation to the final physical equation clearly defined?	CPF	0 – 4
	Is each assumption of the final physical equation satisfied?	CPF	0 – 3
	How is the assumption satisfied?	CPF	0 – 6
	How common is the use of the original physical equation for this type of simulation?	CPF	0 – 4
	Was the final physical equation validated?	CPF	0 – 4
	Was the derivation from the original physical equation to the final physical equation peer reviewed?	CPF	0 – 4
	Peer Review Assessments	CPF	many
	Validation Assessments	CPF	many
EA-5.3	How was the overall verification for the simulation performed?	CPF	0 – 5
	Was the Coded Physical Function equation verified?	CPF	0 – 4
	Was the process from the final physical equation to the Coded Physical Function equation peer reviewed?	CPF	0 – 4
	Peer Review Assessments	CPF	many
	Verification Assessments	CPF	many
	Validation Assessments	CPF	many

7. Conclusions and Recommendations for Future Work

This chapter provides the conclusions of this dissertation as well as the recommendations for future work.

7.1. Summary

Whether to advance scientific knowledge or make a technical decision, using the results of a scientific computer simulation for any purpose requires a decision to be made. Those results are generally not trusted unless the simulation has gone through some amount of review. While this review process is vital in trusting the results of a simulation, scientific computer simulation review has rarely been independently studied. While there are multiple references on how to model the phenomena which are predicted by a simulation, there are few references which discuss how to demonstrate those predictions are trustworthy.

The main objective of this dissertation is to better establish scientific computer simulation review as its own field of study which would in turn facilitate further discussions of scientific computer simulations and the trustworthiness of their results. This is achieved by contributing to the fundamental theory of scientific computer simulation review. This fundamental theory includes:

- the generation of a basic vocabulary which can be used to discuss the various concepts in simulation review,
- a formalization of the concept of maturity which is used to better understand the tools available in simulation review,
- the creation of a hierarchy which can be used to organize and represent many scientific computer simulations,
- and the development of an assessment framework which establishes the boundaries of simulation review, highlights many of the assumptions of a simulation which need to be supported, and establishes a method for its continual evolution.

The vocabulary generated in Chapters 1 - 5 defines many of the common and important concepts in simulation review. Definitions are given for concepts ranging from the very basic (e.g., scientific computer simulation review) to the very complex (e.g., Essential Assumption). As the concepts are defined, their definition is focused to those specific aspects of the concept which are important in simulation review in general and not necessarily to any one particular simulation. Thus, many familiar terms are redefined (e.g., Computer Code) while some new terms are introduced (e.g., Coded Physical Function). The breadth and depth of these definitions resulted in a basic vocabulary.

The discussion in Chapter 2 starts with the commonly used concept of maturity and formalizes this concept into a robust Maturity Theory. In this theory, the nature of maturity is examined and its basic components are discovered. These components reveal certain aspects about maturity assessment that are otherwise hidden. Both the components and the aspects are used to better understand the concepts of maturity assessment and maturity assessment frameworks, which are the tools of simulation review.

The Hierarchy of Scientific Computer Simulation Components in Chapter 3 is created by examining the structure of multiple source codes and identifying components common to almost all simulations. These components are defined in such a way that the hierarchy is applicable to many scientific computer simulations, that every component of a simulation is defined, and that each level of the hierarchy (i.e., each component) can be represented as a collection of the levels beneath it.

The Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations is developed in Chapters 4 – 6 by analyzing the components defined in the hierarchy and identifying the Essential Assumptions of each component (i.e., the assumptions that are necessary and sufficient for the component to be trusted). Because these components can be organized to represent almost any scientific computer simulation, their corresponding Essential Assumptions can also be organized to represents the necessary and sufficient assumptions which need to be true for that simulation to be trusted. Each Essential Assumption becomes the focus of its own maturity assessment framework where the goal of that framework is to contain the maturity assessment sets which are needed to validate the specific Essential Assumption.

7.2. **Conclusions**

By generating a basic vocabulary, a better means for the discussion of scientific computer simulation review is given. By formalizing Maturity Theory, the tools needed to understand and perform scientific computer simulation review are better understood. By creating the Hierarchy of Scientific Computer Simulation Components, a methodology which can be used to organize and represent many scientific computer simulations in the same fashion is established. By developing the Theoretical/Logical Maturity Assessment Framework for Scientific Computer Simulations, a framework which establishes the boundaries of simulation review is provided. By contributing this needed fundamental theory, these advancements better establish the field scientific computer simulation review.

7.3. Recommendations for Future Work

The following are the recommendations for future work.

7.3.1. Further developments of Scientific Computer Simulation Review

Perhaps the most important recommendation for future work is the further development of the other three stages of scientific computer simulation review. This dissertation did define four stages of simulation review (Maturity Framework Development, Maturity Requirements Determination, Maturity Level Assessment, and Maturity Judgment), but it only advanced one stage, Maturity Framework Development. Further development of the Maturity Requirements and Maturity Judgment stages are highly recommended and would be highly profitable as these stages are very important to simulation review but present very difficult challenges.

Maturity Requirements asks: How should the requirements of the intended purpose of a simulation should be determined? This question would be difficult to answer systematically, but one practical answer is determining the requirements of the intended purpose by assessing the maturity of many simulations currently deemed trustworthy. This process would require a well-tested maturity framework and access to many trusted simulations.

Maturity Judgment asks even more difficult questions: What is to be done with the great number of assessments? What if some assessments are below their required levels and others are above? Can these differences offset each other? Answering these questions is extremely difficult but also extremely important as they are answered when the results of many simulations are trusted. Further development of Maturity Theory could provide a logical foundation for helping to answer some of these questions. Also, further development of Maturity Requirements as described above could reveal how these questions are currently being answered.

7.3.2. Further development of Maturity Theory

The discussion in Chapter 2 provides a basic foundation for Maturity Theory. However, this discussion could be greatly expanded and further codify the rules of maturity. It is likely that concepts developed which focus on decision making and concepts developed in set theory (especially order theory) could provide a useful basis for extending Maturity Theory. While Maturity Theory is useful in

understanding the tools of simulation review, it is widely applicable to a great number of different fields and its further development potentially has the largest impact of any of the recommended future work.

7.3.3. Further development of Hierarchy of Scientific Computer Simulation Components

One of the conclusions of the application of the Theoretical/Logical Framework in Appendix A was that some of the assessment sets were not focused on a specific component in the Hierarchy. Rather, they seemed to be focused on a sub-component. Further developing the Hierarchy to define these sub-components and formalize their rules is needed in order to create a better and more accurate Theoretical/Logical Framework.

7.3.4. Further development of assessment sets in the Theoretical/Logical Framework

The Theoretical/Logical Framework was designed with its continual development in mind. It was not developed by creating a collection of good assessment sets; rather it was developed such that through the evolution process it would generate better assessment sets. This process requires continually application and analysis of the framework. One such application and analysis is provided in Appendix A.

This application revealed the following changes which should be incorporated in the next revision of the Theoretical/Logical Framework:

- Focusing the assessment sets on specific sub-components of the Hierarchy
- Further refining the validation assessments
- Ensuring the assessment sets are not focused on specific simulation requirements

While the Theoretical/Logical framework is not focused on being practically applied, a subset of its assessment sets could be made into a practical framework. The key question when creating such a framework is: What assessment set should be considered to maximize the understanding of maturity for the time and costs incurred in performing an assessment? In other words, along with the development of the assessment sets for the Theoretical/Logical framework, some thought should be given as to what would be needed to make the assessment sets economical.

7.3.5. Expanding the Necessary Assumptions of Verification, Validation, and Uncertainty Quantification

The concepts of VV&UQ are extremely important to simulation review. The further expansion of the Necessary Assumptions of Verification and Validation and the creation of the Necessary Assumptions of Uncertainty Quantification would provide additional insight and understanding into exactly what assumptions are made during these tasks and what supporting evidence they provide. Ideally, all three sets could be expanded into sets which contained all of the necessary and sufficient conditions for demonstrating their corresponding Fundamental Assumptions were valid. Additionally, the Necessary Assumptions of Uncertainty Quantification would allow it to become part of the Theoretical/Logical Framework.

7.3.6. Applying the Theoretical/LogicalMaturity Framework using a computer

The frameworks of PCMM and NASA represent practical frameworks. The framework developed here is a Theoretical/Logical Framework. The difference between the two has been well contrasted in this dissertation. As demonstrated by the application in Appendix A, the Theoretical/Logical Framework does provide many more maturity assessments that can be easily analyzed by a human. However, it seems that this wealth of information could be analyzed by a computer. While assessing hundreds or thousands of assessment sets would take substantial time for an analyst, it could take much less time for a computer.

Automating the process of maturity assessment and having that assessment performed by a computer should be further investigated. Even if it is ultimately discovered that results of such a procedure add little value, attempting to write the assessment sets so they could be performed by a computer will make them much more focused and objective and lead to a better overall framework.

A. Example Application of the Theoretical/Logical Framework

The following is the application of Theoretical/Logical Framework to homework 6 problem 5 of ENAE 684 from Fall 2010 at University of Maryland. The goal of this simulation was to use Lax's method with the point operator applied to linear convection, given as:

$$u_j^{n+1} = \frac{u_{j+1}^n + u_{j-1}^n}{2} - \frac{a \cdot \Delta t}{\Delta x} \cdot \left(\frac{u_{j+1}^n + u_{j-1}^n}{2} \right)$$

Where $a > 0$ with Dirichlet boundary conditions at $x = 0$ and a non-reflective boundary condition at the other end (i.e., Eulcer-excplicit w/ 2-pt backwards).

The initial condition for this simulation is discontinuous and is given as follows:

$$u(x,0) = \begin{cases} 0.5 & x < {}^{\pi}/_{2} \\ 1.5 & {}^{\pi}/_{2} < x < \pi \end{cases}$$

The specific simulation was performed with a CFL = 0.75 and using 61 mesh points.

A.1. Source Code

The source code for this homework was separated into two different m-files from MATLAB.

A.1.1. newwave.m

```
% newwave.m
%
% Written by Dr. James D. Baeder for ENAE684
% UNIVRSITY OF MARYLAND
%
% Linear advection equation
% u_t + a*u_x = 0 ;a>0
% Dirichlet at left and no reflection at right
% Numerical Method (with appropriate bc?s):
% 1) Lax's Method
% u_j^n+1 = u_j^n - 0.5*cfl*(u_j+1^n - u_j-1^n)
% + 0.5*(u_j+1^n - 2*u_j^n + u_j-1^n)
% Initial Solution:
% u(t=0,x) = 1.5 ; x>pi/2
% 0.5 ; x<=pi/2
% Boundary Condition:
% u(t,x=0) = 1 - 0.5*cos(-4*t)
% Exact Solution:
% Implemented in exact.m
% Numerical Solution:
% ?!?!?!?!?!?!?!
%
% Let's clear everything including graphics for figures 1
clear all;
close all;
figure(1);

% Let's get the input from the user!!!!
Imeth = 1;%input('Enter method (1): ');
cfl = 0.75;%input('Enter cfl number: ');
jmax = input('Enter # of mesh points (including end points): ');
% Set some variables for now
a = 1;
tfinal = 1.5*pi;
xend = pi;
% Calculate some variables
dx = xend/(jmax-1); % mesh spacing in x
dt = cfl*dx/a; % time step size
dtdx = dt/dx;
nsteps = round(tfinal/dt); % total number of time steps
npart = round(nsteps/6); % intermediate time step
% initialize mesh locations
```

```
x = (0:jmax-1)*dx;
% preallocate for speed
du = zeros(jmax,1);
% store exact solution at a few times
229prate=exact(x,a,dt*npart);
uinite=exact(x,a,0);
% initial data:
u=uinite;
% output some data
disp(['Number of time steps is ',num2str(nsteps)]);
% okay let's cycle through the time steps!
For n = 1:nsteps
    t = n*dt;
    uexact = exact(x,a,t);
    if(imeth==1)
        method='Lax';
        % Dirichlet BC at x = 0
        du(1) = exact(0,a,t)-exact(0,a,t-dt);
        % Interior operator
        for j=2:jmax-1
            du(j) = -0.5*cfl*(u(j+1)-u(j-1))+0.5*(u(j+1)-2*u(j)+u(j-
1));
        end
        % Euler-explicit backward: non-reflective boundary condition
at other end
        du(jmax) = -cfl*(u(jmax)-u(jmax-1));
        u=u+du;
    end
    if (n==npart) % save the solution in upart
        upart = u;
    end;
    % plot solution (initial, current and partway):
    figure(1)
    plot(x,uinite,'g',x,uexact,'b',x,u,'bo','LineWidth',2.0)
    if(n>=npart)
        hold on
        plot (x,229prate,'r',x,upart,'ro','LineWidth',2.0);
        hold off
    end;
    % calculate L1norm of error
    L1norm=sum(abs(u-uexact))/length(u);
    set(gca,'FontSize',14,'LineWidth',2.0,'FontWeight','demi');
    title([' Solution: ',method,'; M = ',num2str(jmax),...
        '; t = ',num2str(t),'; L1norm = ',num2str(L1norm)])
    xlabel('X'); ylabel('U');
    axis([0,xend,0.,2.])
end
disp(['L1norm = ',num2str(L1norm)]);
```

A.1.2. exact.m

```
function utrue = exact(x,a,t)
% exact.m
% The exact solution

m1=length(x);
utrue = zeros(m1,1);
for j=1:m1
% from initial condition
if(x(j)-a*t>=pi/2)
utrue(j)=1.5;
end
if(x(j)-a*t>=0 && x(j)-a*t<pi/2)
utrue(j)=0.5;
end
% from boundary condition
if(x(j)-a*t<=0)
utrue(j)=1-0.5*cos(-4*(x(j)/a-t));
end
end
```

A.2. Components of the Scientific Computer Simulation

The source code is separated into the components described below.

A.2.1. Complete Set of Codes

The Complete Set of Codes is named "Homework 6_5". The inputs for the Complete Set of Codes are provided in Table A-1 and the selected outputs are provided in Table A-2.

Table A- 1: Inputs for the Complete Set of Codes

Component ID	Input	Representation in the Source Code
I_{csc}-1	The Courant-Friedrichs-Levy number	`cfl`
I_{csc}-2	The starting x-location	N/A
I_{csc}-3	The ending x-location	`xend`
I_{csc}-4	The number of Mesh points	`jmax`
I_{csc}-5	The convection factor	`a`
I_{csc}-6	The starting time	N/A
I_{csc}-7	The ending time	`tfinal`

Table A- 2: Selected Outputs from the Complete Set of Codes

Component ID	Selected Output	Representation in the Source Code
O_{csc}-1	The mesh points	`x`
O_{csc}-2	The predicted velocity at each mesh point	`u`
O_{csc}-3	The exact velocity at each mesh point	`uexact`
O_{csc}-4	The L1 Norm of the error	`L1norm`

A.2.2. Computer Codes

"Homework 6_5" is made of two Computer Codes: "HW6_5" and "exact". A description of each code, along with its inputs and outputs is given below.

A.2.2.1. "HW6_5"

This is the main code used to perform the simulation. It applies Lax's method to linear convection. The inputs for this Computer Code are provided in Table A-3 and the selected outputs are provided in Table A-4. All of the inputs to the Computer Code are repeat inputs from the Complete Set of Codes. However, this Computer Code has additional outputs which are not outputs of the complete set.

Table A- 3: Inputs for "HW6_5"

Component ID	Input	Representation in the Source Code
I_{CC_1}-1 ($\equiv I_{CSC}$-1)	The Courant-Friedrichs-Levy number	`cfl`
I_{CC_1}-2 ($\equiv I_{CSC}$-2)	The starting x-location	N/A
I_{CC_1}-3 ($\equiv I_{CSC}$-3)	The ending x-location	`xend`
I_{CC_1}-4 ($\equiv I_{CSC}$-4)	The number of Mesh points	`jmax`
I_{CC_1}-5 ($\equiv I_{CSC}$-5)	The convection factor	`a`
I_{CC_1}-6 ($\equiv I_{CSC}$-6)	The starting time	N/A
I_{CC_1}-7 ($\equiv I_{CSC}$-7)	The ending time	`tfinal`

Table A- 4: Selected Outputs from "HW6_5"

Component ID	Output	Representation in the Source Code
O_{CC_1}-1 ($\equiv O_{CSC}$-1)	The mesh points	`x`
O_{CC_1}-2 ($\equiv O_{CSC}$-2)	The predicted velocity at each mesh point	`u`
O_{CC_1}-3 ($\equiv O_{CSC}$-3)	The exact velocity at each mesh point	`uexact`
O_{CC_1}-4 ($\equiv O_{CSC}$-4)	The L1 Norm of the error	`L1norm`
O_{CC_1}-5	The convection factor	`a`
O_{CC_1}-6	The current calculational time	`t`

A.2.2.2. "exact"

This code is used to calculate the exact solution. The inputs for this Computer Code are provided in Table A-5 and the selected outputs are provided in Table A-6.

Table A- 5: Inputs for "exact"

Component ID	Input	Representation in the Source Code
I_{CC_2}-1 (≡ O_{CSC}-1)	The mesh points	`x`
I_{CC_2}-2 (≡ I_{CSC}-5)	The convection factor	`a`
I_{CC_2}-3 (≡ O_{CC_1}-6)	The current calculational time	`t`

Table A- 6: Selected Outputs from "exact"

Component ID	Selected Output	Representation in the Source Code
O_{CC_2}-1 (≡ O_{CC_1}-3)	The exact velocity at each mesh point	`uexact`

A.2.3. Coded Groups

The source code could be separated into Coded Groups. However such separation does not seem useful in this example. Thus, each Computer Code is considered to have all of its Coded Physical Functions in a single Coded Group.

Conversely, this same problem could have been represented as contacting a single Computer Code with two main Coded Groups ("HW6_5" and "exact").

A.2.4. Coded Physical Functions

The following are the Coded Physical Functions used in each Computer Code:

A.2.4.1. Coded Physical Functions in “HW6_5”

Table A- 7: Coded Physical Functions in "HW6_5"

Component ID	Representation in the Source Code
CPF_1-1	`cfl = 0.75`
CPF_1-2	`jmax = 61`
CPF_1-3	`a = 1`
CPF_1-4	`tfinal = 1.5*pi`
CPF_1-5	`xend = pi`
CPF_1-6	`dx = xend/(jmax-1)`
CPF_1-7	`dt = cfl*dx/a;`
CPF_1-8	`dtdx = dt/dx`
CPF_1-9	`nsteps = round(tfinal/dt)`
CPF_1-10	`npart = round(nsteps/6)`
CPF_1-12	`x = (0:jmax-1)*dx`
CPF_1-13	`du = zeros(jmax,1)`
CPF_1-14	`u=uinite`

CPF_1-15	`for n = 1:nsteps`
	`  t = n*dt;`
	`end`

CPF_1-16	`for n = 1:nsteps`
	`  du(1) = exact(0,a,t)-exact(0,a,t-dt);`
	`  for j=2:jmax-1`
	`  du(j) = -0.5*cfl*(u(j+1)-u(j-+0.5*(u(j+1)-2*u(j)+u(j-1));`
	`  end`
	`  du(jmax) = -cfl*(u(jmax)-u(jmax-1));`
	`  end`
	`end`

CPF_1-17	`for n = 1:nsteps`
	`  L1norm=sum(abs(u-uexact))/length(u);`
	`end`

A.2.4.2. Coded Physical Functions for exact

Table A- 8: Coded Physical Functions in "exact"

Component ID	Representation in the Source Code
CPF_2-1	`m1=length(x)`
CPF_2-2	`utrue = zeros(m1,1)`

CPF_2-3	`for j=1:m1`
	`% from initial condition`
	`  if(x(j)-a*t>=pi/2)`
	`    utrue(j)=1.5;`
	`  end`
	`end`

CPF_2-4	`for j=1:m1`
	`% from initial condition`
	`  if (x(j)-a*t>=0 && x(j)-a*t<pi/2)`
	`    utrue(j)=0.5;`
	`  end`
	`end`

CPF_2-5	`for j=1:m1`
	`% from initial condition`
	`  if(x(j)-a*t<=0)`
	`     utrue(j)=1-0.5*cos(-4*(x(j)/a-t));`
	`   end`
	`end`

A.3. Scientific Computer Simulation

The scientific computer simulation is defined as the set containing the Complete Set of Codes ("Homework 6_5") along with the selected input provided in Table A-9 and the resulting output discussed in Table A-10.

Table A- 9: Inputs for the Complete Set of Codes

Component ID	Input	Value in Simulation
I_{CSC}-1	The Courant-Friedrichs-Levy number	0.75
I_{CSC}-2	The starting x-location	0
I_{CSC}-3	The ending x-location	π
I_{CSC}-4	The number of Mesh points	61
I_{CSC}-5	The convection factor	1
I_{CSC}-6	The starting time	0
I_{CSC}-7	The ending time	$1.5 \cdot \pi$

Table A- 10: Outputs from the Complete Set of Codes

Component ID	Selected Output	Value from Simulation
O_{CSC}-1	The mesh points	See Table A-11
O_{CSC}-2	The predicted velocity at each mesh point at each time	See Table A-11 and NOTE
O_{CSC}-3	The exact velocity at each mesh point	See Table A-11
O_{CSC}-4	The L1 Norm of the error	0.096091

NOTE: The main System Response Quantity of this simulation was the predicted velocity (O_{CSC}-2). While this velocity was predicted at each time step, it was not recorded at each time step. Thus, the value of the velocity given below is the final value calculated at the final time step for the simulation.

Table A- 11: Output from Simulation

x	u	uexact
0	0.5	0.5
0.05236	0.51655	0.51093
0.10472	0.5531	0.54323
0.15708	0.60764	0.59549
0.20944	0.67744	0.66543
0.2618	0.75918	0.75
0.31416	0.84913	0.84549
0.36652	0.94328	0.94774
0.41888	1.0376	1.0523
0.47124	1.128	1.1545
0.5236	1.2109	1.25
0.57596	1.2829	1.3346
0.62832	1.3412	1.4045
0.68068	1.3837	1.4568
0.73304	1.409	1.4891
0.7854	1.4165	1.5
0.83776	1.4061	1.4891
0.89012	1.3789	1.4568
0.94248	1.3364	1.4045
0.99484	1.2806	1.3346
1.0472	1.2144	1.25
1.0996	1.1407	1.1545
1.1519	1.0628	1.0523
1.2043	0.98411	0.94774
1.2566	0.90801	0.84549
1.309	0.83761	0.75
1.3614	0.77575	0.66543
1.4137	0.72485	0.59549
1.4661	0.68677	0.54323
1.5184	0.66284	0.51093
1.5708	0.6537	0.5
1.6232	0.65941	0.51093
1.6755	0.67935	0.54323
1.7279	0.71235	0.59549
1.7802	0.7567	0.66543
1.8326	0.81026	0.75
1.885	0.87054	0.84549
1.9373	0.93484	0.94774
1.9897	1.0004	1.0523
2.042	1.0643	1.1545
2.0944	1.124	1.25
2.1468	1.177	1.3346
2.1991	1.2213	1.4045
2.2515	1.2551	1.4568
2.3038	1.2774	1.4891
2.3562	1.2875	1.5
2.4086	1.2851	1.4891
2.4609	1.2708	1.4568
2.5133	1.2454	1.4045
2.5656	1.2103	1.3346
2.618	1.1671	1.25
2.6704	1.1179	1.1545
2.7227	1.0649	1.0523
2.7751	1.0105	0.94774
2.8274	0.95685	0.84549
2.8798	0.90635	0.75
2.9322	0.86102	0.66543
2.9845	0.82263	0.59549
3.0369	0.79263	0.54323
3.0892	0.77191	0.51093
3.1416	0.75985	0.5

A.4. Maturity Assessments from the Theoretical/Logical Framework

The following table provides a description of the important components of the simulation along with the number of assessment sets which can be used to assess each component.

Table A- 12: Description of the Simulation

Component	Number in Simulation	Number of Applicable Assessment Sets per Component
Complete Set of Codes	1	1
Input to the Complete Set of Codes	7	16
Output from the Complete Set of Codes	4	14
Inputs to the Computer Code	10	15
Outputs from the Computer Code	7	15
Inputs to the Coded Group	N/A	15
Outputs from the Coded Group	N/A	15
Inputs to the Coded Physical Functions	≈65	16
Outputs from the Coded Physical Functions	21	15
Coded Physical Functions	21	113

Even this simple simulation would result in over 4000 assessments. While performing all of these assessments for every component in the simulation is well beyond the scope of this dissertation, one complete set of assessments will be performed for each type of component in the simulation.

A.4.1. Common Assessment Identifications

The assessments for Peer Review and Validation are used multiple times. This section provides the link between the Assessment IDs and its corresponding attribute

Table A- 13: Peer Review Attributes

Assessment ID	Peer Review Attributes
PR-1	How thorough was the peer review?
PR-2	How independent was the peer review?
PR-3	How important was the peer review?
PR-4	How different was the peer review?
PR-5	How different was the peer review?
PR-6	How many individuals were involved in the peer review?
PR-7	What was the knowledge base of each peer reviewer?
PR-8	What was the historical base of the peer reviewer?
PR-9	What was the experience base of the peer reviewer?
PR-10	How involved was the peer reviewer?
PR-11	Did the peer reviewer believe they had enough resources to perform an appropriate peer review?
PR-12	Did the peer reviewer believe their feedback was understood?
PR-13	Did the peer reviewer believe their feedback was handled appropriately?

Table A- 14: Validation Attributes

Assessment ID	Validation Attributes
VL-1.1	How common is the prediction of the physical quantity?
VL-1.2	How common is the physical quantity predicted under these specific conditions?
VL-2.1	How common is the prediction of the physical quantity using the given set of dependent variables?
VL-3.1	How common is this physical quantity measured in the given method?
VL-3.2	How accurate is the measurement generally considered?
VL-3.3	How much computation is needed to obtain the measured value?
VL-3.4	How is the uncertainty on the measurement determined?
VL-4.1	How common is this physical quantity measured in the given application space?
VL-5.1	How thoroughly sampled is the application space?
VL-5.2	How widely varying are the measurement values in the application space?
VL-5.3	How widely varying are the predicted values in the application space?
VL-5.4	Is there evidence to suggest that the error sample has the same mean as the error population?
VL-5.5	Is there evidence to suggest that the error sample has the same variance as the error population?
VL-5.6[22]	Is there evidence to suggest that the error sample has the same distribution as the error population?
VL-6.1[24]	Is there evidence to suggest that the magnitude of the error in the sample is independent of its location in the application space?
VL-7.1	Is the magnitude of the error small compared with the measured and predicted values?

[22] Not used in assessment. No assessment set has been developed.

A.4.2. Assessments of a Complete Set of Codes

Table A- 15: Assessment for a Complete Set of Codes

Component ID	"Homework 6_5"
Component	Complete Set of Codes
Assumption	EA-1

EA-1-1
-1
0
1
2
3
4
5

Table A- 16: Attributes for the Complete Set of Codes

Assessment ID	Attribute
EA-1-1	How simple is the set of all inputs?

A.4.3. Assessments of the Input to a Complete Set of Codes

Table A- 17: Assessment for the Input to a Complete Set of Codes

Component ID	I_{CSC}-1 (The Courant-Friedrichs-Levy number)														
Component	Input for the Complete Set of Codes														
Assumption	EA-1														
EA-1-2	EA-1-3	EA-1-4	PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
		4	4	4	4			4	4	4	4	4	4	4	4
			5					5		5					
								6		6					

Table A- 18: Attributes for the Input to the Complete Set of Codes

Assessment ID	Attribute
EA-1-2	What is the origin of the input?
EA-1-3	How detailed is the input?
EA-1-4	Was the input peer reviewed?

For the Peer Review Attributes see Table A-13.

A.4.4. Assessments of the Output from a Complete Set of Codes

Table A- 19: Assessments for the Output for a Complete Set of Codes

Component ID	O_{CSC}-2 (The predicted velocity at each mesh point)												
Component	Output for the Complete Set of Codes												
Assumption	EA-8												
EA-8-1	PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4			4	4	4	4	4	4	4	4
			5					5		5			
								6		6			

Table A- 20: Attributes for the Output for a Complete Set of Codes

Assessment ID	Attribute
EA-8-1	Was the output peer reviewed?

For the Peer Review Attributes see Table A-13.

A.4.5. Assessments of the Input to a Computer Code

Table A- 21: Assessments for the Input to a Computer Code

Component ID		I_{CC_1}-4 (The number of mesh points)												
Component		Input to a Computer Codes												
Assumption		EA-2												
EA-2-1	EA-2-2	PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4			4	4	4	4	4	4	4	4
			5					5		5				
								6		6				

Table A- 22: Attributes for the Input to a Computer Code

Assessment ID	Attribute
EA-2-1	How was the selection of the input to each Computer Code verified?
EA-2-2	Was the input peer reviewed?

For the Peer Review Attributes see Table A-13.

A.4.6. Assessments of the Output from a Computer Code

Table A- 23: Assessments for the Output from a Computer Code

Component ID	O_{CC_1}-4 (L1 Norm of Error)													
Component	Output from the Computer Code													
Assumption	EA-7													
EA-7-1	EA-7-2	PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4			4	4	4	4	4	4	4	4
			5					5		5				
								6		6				

Table A- 24: Attributes for the Output from a Computer Code

Assessment ID	Attribute
EA-7-1	How was the selection of the output to each Computer Code verified?
EA-7-2	Was the output peer reviewed?

For the Peer Review Attributes see Table A-13.

A.4.7. Assessments of the Input to a Coded Physical Function

Table A- 25: Assessments for the Input to a Coded Physical Function

Component ID	CPF_1-16 (Coded Physical Function for `du`)
Component	Input to a Coded Physical Function
Assumption	EA-4

EA-4-1	EA-4-2	PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4			4	4	4	4	4	4	4	4
			5					5		5				
								6		6				

Table A- 26: Attributes for the Input to a Coded Physical Function

Assessment ID	Attribute
EA-4-1	How was the selection of the input to each Coded Physical Function verified?
EA-4-2	Was the input peer reviewed?

For the Peer Review Attributes see Table A-13.

A.4.8. Assessments of a Coded Physical Function

A.4.8.1. Assessments for EA-5.1

The assessments for EA-5.1 are split into the following three tables:

Table A- 27: Assessments for a Coded Physical Function, Assumption 5.1 (Part 1 of 3)

Component ID		CPF_1-16 (Coded Physical Function for `du`)			
Component		Coded Physical Functions			
Assumption		EA-5.1			
EA-5.1-1	EA-5.1-2	EA-5.1-3	EA-5.1-4	EA-5.1-5	EA-5.1-6
-1	-1	-1	-1	-1	-1
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
		4	4	4	4
		5			
		6			

Table A- 28: Attributes for a Coded Physical Function, Assumption 5.1

Assessment ID	Attribute
EA-5.1-1	What is the origin of the original physical equation?
EA-5.1-2	Are the assumptions of the original physical equation satisfied?
EA-5.1-3	How are the assumptions satisfied?
EA-5.1-4	How common is the use of the original physical equation for this type of simulation?
EA-5.1-5	Was the original physical equation validated?
EA-5.1-6	Was the original physical equation peer reviewed?

Table A- 29: Assessments for a Coded Physical Function, Assumption 5.1 (Part 2 of 3)

Component ID			CPF_1-16 (Coded Physical Function for `du`)									
Component			Coded Physical Functions									
Assumption			EA-5.1									
PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4			4	4	4	4	4	4	4	4
	5					5		5				
						6		6				

For the Peer Review Attributes see Table A-13.

Table A- 30: Assessments for a Coded Physical Function, Assumption 5.1 (Part 3 of 3)

Component ID	CPF_1-16 (Coded Physical Function for `du`)
Component	Coded Physical Functions
Assumption	EA-5.1

VL-1.1	VL-1.2	VL-2.1	VL-3.1	VL-3.2	VL-3.3	VL-3.4	VL-4.1	VL-5.1	VL-5.2	VL-5.3	VL-5.4	VL-5.5	VL-7.1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4					4
						5							5
													6
													7

For the Validation Attributes see Table A-14.

A.4.8.2. Assessments for EA-5.2

The assessments for EA-5.2 are split into the following three tables

Table A- 31: : Assessments for a Coded Physical Function, Assumption 5.2 (Part 1 of 3)

Component ID		CPF_1-16 (Coded Physical Function for `du`)			
Component		Coded Physical Functions			
Assumption		EA-5.2			
EA-5.2-1	EA-5.2-2	EA-5.2-3	EA-5.2-4	EA-5.2-5	EA-5.2-6
-1	-1	-1	-1	-1	-1
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4		4	4	4	4
		5			
		6			

Table A- 32: Attributes for a Coded Physical Function, Assumption 5.2

Assessment ID	Attribute
EA-5.2-1	Are the derivation steps from the original physical equation to the final physical equation clearly defined?
EA-5.2-2	Are the assumptions of the final physical equation satisfied?
EA-5.2-3	How are the assumption satisfied?
EA-5.2-4	How common is the use of the original physical equation for this type of simulation?
EA-5.2-5	Was the final physical equation validated?

EA-5.2-6	Was the derivation from the original physical equation to the final physical equation peer reviewed?

Table A- 33: Assessments for a Coded Physical Function, Assumption 5.2 (Part 2 of 3)

Component ID			CPF_1-16 (Coded Physical Function for `du`)									
Component			Coded Physical Functions									
Assumption			EA-5.2									
PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4			4	4	4	4	4	4	4	4
	5					5		5				
						6		6				

For the Peer Review Attributes see Table A-13.

Table A- 34: Assessments for a Coded Physical Function, Assumption 5.2 (Part 3 of 3)

Component ID			CPF_1-16 (Coded Physical Function for du)										
Component			Coded Physical Functions										
Assumption			EA-5.2										
VL-1.1	VL-1.2	VL-2.1	VL-3.1	VL-3.2	VL-3.3	VL-3.4	VL-4.1	VL-5.1	VL-5.2	VL-5.3	VL-5.4	VL-5.5	VL-7.1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4					4
						5							5
													6
													7

For the Validation Attributes see Table A-14.

A.4.8.3. Assessments for EA-5.3

The assessments for EA-5.3 are split into the following three tables:

Table A- 35: Assessments for a Coded Physical Function, Assumption 5.3 (Part 1 of 4)

Component ID	CPF_1-16 (Coded Physical Function for `du`)	
Component	Coded Physical Functions	
Assumption	EA-5.3	
EA-5.3-1	EA-5.3-2	EA-5.3-3
-1	-1	-1
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5		

Table A- 36: Assessments for a Coded Physical Function, Assumption 5.3

Assessment ID	Attribute
EA-5.3-1	How was the overall verification for the simulation performed?
EA-5.3-2	Was the Coded Physical Function equation verified?
EA-5.3-3	Was the process from the final physical equation to the Coded Physical Function equation peer reviewed?

Table A- 37: Assessments for a Coded Physical Function, Assumption 5.3 (Part 2 of 4)

Component ID			CPF_1-16 (Coded Physical Function for `du`)									
Component			Coded Physical Functions									
Assumption			EA-5.3									
PR-1	PR-2	PR-3	PR-4	PR-5	PR-6	PR-7	PR-8	PR-9	PR-10	PR-11	PR-12	PR-13
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4			4	4	4	4	4	4	4	4
	5					5		5				
						6		6				

For the Peer Review Attributes see Table A-13.

Table A- 38: Assessments for a Coded Physical Function, Assumption 5.3 (Part 3 of 4)

Component ID			CPF_1-16 (Coded Physical Function for `du`)										
Component			Coded Physical Functions										
Assumption			EA-5.3										
VL-1.1	VL-1.2	VL-2.1	VL-3.1	VL-3.2	VL-3.3	VL-3.4	VL-4.1	VL-5.1	VL-5.2	VL-5.3	VL-5.4	VL-5.5	VL-7.1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4					4
						5							5
													6
													7

For the Validation Attributes see Table A-14.

Table A- 39: Assessments for a Coded Physical Function, Assumption 5.3 (Part 4 of 4)

Component ID	CPF_1-16 (Coded Physical Function for `du`)
Component	Coded Physical Functions
Assumption	EA-5.3

VR-2.1	VR-3.1
-1	-1
0	0
1	1
2	2
3	3
	4
	5

Assessment ID	Attribute
VR-1.1[23]	Is there evidence to suggest that there are no mistakes or bugs in the source code?
VR-2.1	Does the discretized space bound the continuous space?
VR-3.1	How is the discretization error determined?
VR-4.1[26]	Is there evidence to suggest that the error sample has the same distribution as the error population?
VR-5.1[26]	Is there evidence to suggest that the magnitude of the error in the sample is independent of its location in the application space?

[23] Not used in assessment. No assessment set has been developed.

A.5. Conclusions and Summary

The application of the Theoretical/Logical Framework to even this simple problem resulted in more assessments than could be easily be understood or represented. This was expected as the major challenge in developing a practical framework is to narrow the assessment sets into only a few which provide an overall picture of a simulation. On the other hand, it is the goal of the Theoretical/Logical Framework to provide all possible assessment sets. Thus, an assessment using the Theoretical/Logical Framework would likely provide information than can be easily understood by humans. However, it is possible that this information could be processed by a computer.

In some cases, the application of the assessment sets was very straight forward and in others it was difficult. This difficulty is mostly attributed to the fact that many of the assessment sets were not focused on a component of the hierarchy, but a "sub-component". For example, it was unclear if the assessments focused on the input to a component were focused on the input as a whole or each individual input. Additionally, for many of the peer review assessments, the basic terminology in the peer review was not defined well enough to establish exactly how the component under consideration should be assessed.

This same confusion can be seen in the validation attributes which were either assessed at the very lowest or highest maturity levels and no levels in between.

A number of assessment sets were found to be dependent on the requirements of the simulation (e.g., many of the peer review assessment sets). As this simulation had no real requirements, assesses these maturities was difficult.

On one hand, the application of the framework demonstrated that many of the current maturity assessment sets did not provide the useful information which was intended. Even considering the sheer number of assessment performed, the assessed levels provided little insight into simulation. While one could argue that such insight may become apparent after repeated applications of the same framework to different simulation, this insight is deemed unlikely.

On the other hand, the reason it was so easily determined that many of the current maturity assessment sets did not provide useful information was because through the assessment process the flaws in the assessment sets were obvious. Those flaws were only obvious because of the significant theory developed in this dissertation which focused on what an assessment set should be and how the framework should work. It was easy to determine that the current version of the framework fell short of that goal and in many instances it was also easy to determine many of the fixes which would be needed. Maturity Theory and its application in the Theoretical/Logical Framework

was not focused on how to create maturity assessments which were good. Instead, it was focused on how to create maturity assessment sets which are better.

B. First Principles

This appendix provides a list of the first principles and the genesis equations for some of those first principles.

B.1. Conservation of Mass

Equation ID	**Genesis Equation for the Conservation of Mass**
Parent ID	Concept of the Conservation of Mass
Starting Assumptions	• ?
Description of this step	The Genesis Equation for the Conservation of Mass.
Assumptions this step	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Genesis Equation (vector)	$\iiint_{V(t)} \rho \cdot dV = constant$
Genesis Equation (indices)	$\iiint_{V(t)} \rho \cdot dV = constant$

B.2. Conservation of Momentum

Equation ID	**Genesis Equation for the Conservation of Momentum**
Parent ID	Concept of the Conservation of Momentum
Starting Assumptions	• ?
Description of this step	The Genesis Equation for the Conservation of Momentum.
Assumptions this step	• The velocity u is $u(\vec{x}, t)$, a continuous vector field.
Genesis Equation (vector)	$\frac{\partial}{\partial t}(m \cdot \vec{u}) = \sum \vec{F}_{External}$
Genesis Equation (indices)	$\frac{\partial}{\partial t}(m \cdot u_i) = \sum F_{i_{External}}$

B.3. Conservation of Energy

Equation ID	**Genesis Equation for the Conservation of Energy**
Parent ID	Concept of the Conservation of Energy
Starting Assumptions	• ?
Description of this step	The Genesis Equation for the Conservation of Energy.
Assumptions this step	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The specific energy **e** is $\mathbf{e}(\vec{x}, t)$, a continuous scalar field.
Genesis Equation (vector)	$\iiint_{V(t)} \rho \cdot \boldsymbol{e} \cdot dV = constant$
Genesis Equation (indices)	$\iiint_{V(t)} \rho \cdot \boldsymbol{e} \cdot dV = constant$

B.4. Others

The following are some concepts which are often considered first principles.

(1) Newton's 1st Law
(2) Newton's 2nd Law
(3) Newton's 3rd Law
(4) 0th Law of Thermodynamics
(5) 1st Law of Thermodynamics
(6) 2nd Law of Thermodynamics
(7) 3rd Law of Thermodynamics

C. Common Derivations of First Principle Equations

This Appendix provides some common derivations of the conservation of mass and momentum equations in the style recommended in this dissertation. Notice that for each version of the first principle equation, the assumption of that equation are captured, and the identification scheme allows the equation to be traced back to its Genesis Equation.

C.1. Common Derivations of the Conservation of Mass

Equation ID	**Mass-1**
Parent ID	Genesis Equation for the Conservation of Mass
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Description of this step	Change to differential form.
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{d}{dt}\iiint_{V(t)} \rho \cdot dV = 0$$
Genesis Equation (indices)	$$\frac{d}{dt}\iiint_{V(t)} \rho \cdot dV = 0$$

Equation ID	**Mass-2**
Parent ID	Mass-1
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Description of this step	Apply Reynolds Transport Theorem
Assumptions this step	• Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector.
Genesis Equation (vector)	$$\frac{d}{dt}\iiint_{V(t)} \rho \cdot dV = \iiint_{V(t)} \frac{\partial \rho}{\partial t} \cdot dV + \iint_{S(t)} \rho \cdot \vec{u} \odot \vec{n} \cdot dS = 0$$
Genesis Equation (indices)	$$\frac{d}{dt}\iiint_{V(t)} \rho \cdot dV = \iiint_{V(t)} \frac{\partial \rho}{\partial t} \cdot dV + \iint_{S(t)} \rho \cdot u_i \cdot n_i \cdot dS = 0$$

Equation ID	**Mass-3**
Parent ID	Mass-2
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector.
Description of this step	Apply Gauss's Theorem
Assumptions this step	• Volume must be infinitesimal.
Genesis Equation (vector)	$\iiint_{V(t)} \frac{\partial \rho}{\partial t} \cdot dV + \iiint_{V(t)} \nabla \odot (\rho \cdot \vec{u}) \cdot dV = 0$
Genesis Equation (indices)	$\iiint_{V(t)} \frac{\partial \rho}{\partial t} \cdot dV + \iiint_{V(t)} \frac{\partial}{\partial x_i} (\rho \cdot u_i) \cdot dV = 0$

Equation ID	**Mass-4**
Parent ID	Mass-3
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Combine terms inside the integral.
Assumptions this step	• None.
Genesis Equation (vector)	$\iiint_{V(t)} \left[\frac{\partial \rho}{\partial t} + \nabla \odot (\rho \cdot \vec{u})\right] \cdot dV = 0$
Genesis Equation (indices)	$\iiint_{V(t)} \left[\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} (\rho \cdot u_i)\right] \cdot dV = 0$

Equation ID	**Mass-5**
Parent ID	Mass-4
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Recognize the terms in the brackets must be zero for the integral to be zero.
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{\partial \rho}{\partial t} + \nabla \odot (\rho \cdot \vec{u}) = 0$
Genesis Equation (indices)	$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho \cdot u_i) = 0$

Equation ID	**Mass-6**
Parent ID	Mass-5
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Carry through the divergence
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{\partial \rho}{\partial t} + \vec{u} \odot (\nabla \rho) + \rho \cdot (\nabla \odot \vec{u}) = 0$
Genesis Equation (indices)	$\frac{\partial \rho}{\partial t} + u_i \cdot \frac{\partial \rho}{\partial x_i} + \rho \cdot \frac{\partial u_i}{\partial x_i} = 0$

Equation ID	**Mass-7**
Parent ID	Mass-6
Starting Assumptions	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • Reynolds Transport Assumptions? • The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Re-write the equation using the definition of a material derivative.
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{D\rho}{dt} + \rho \cdot (\nabla \odot \vec{u}) = 0$
Genesis Equation (indices)	$\frac{D\rho}{dt} + \rho \cdot \frac{\partial u_i}{\partial x_i} = 0$

C.2. Common Derivations of the Conservation of Momentum

Equation ID	**Momentum-1**
Parent ID	Genesis Equation for the Conservation of Momentum
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field.
Description of this step	Recognize the momentum can be obtained by integrating over the volume.
Assumptions this step	• The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Genesis Equation (vector)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot \vec{u} \cdot dV = \sum \vec{F}_{External}$
Genesis Equation (indices)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot u_i \cdot dV = \sum \vec{F}_{External}$

Equation ID	**Momentum-2**
Parent ID	Momentum-1
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Description of this step	Separate the external forces into body and surface forces.
Assumptions this step	• None
Genesis Equation (vector)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot \vec{u} \cdot dV = \vec{B} + \vec{F}_S$
Genesis Equation (indices)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot u_i \cdot dV = B_i + F_{S_i}$

Equation ID	**Momentum-3**
Parent ID	Momentum-2
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field.
Description of this step	Separate the external forces into body and surface forces.
Assumptions this step	• The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface.
Genesis Equation (vector)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot \vec{u} \cdot dV = \iiint_{V(t)} \vec{b} \cdot dV + \iint_{S(t)} \vec{T} \cdot dS$
Genesis Equation (indices)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot u_i \cdot dV = \iiint_{V(t)} b_i \cdot dV + \iint_{S(t)} T_i \cdot dS$

Equation ID	**Momentum-3.1**
Parent ID	Momentum-3
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface.
Description of this step	Apply Reynolds Transport Theorem to the LHS. Notice that this version is no longer the full momentum equation.
Assumptions this step	• Reynolds Transport Assumptions? • The vector n is a unit surface normal vector.
Genesis Equation (vector)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot \vec{u} \cdot dV = \iiint_{V(t)} \frac{\partial(\rho \cdot \vec{u})}{\partial t} \cdot dV + \iint_{S(t)} (\rho \cdot \vec{u}) \cdot \vec{u} \odot \vec{n} \cdot dS$
Genesis Equation (indices)	$\frac{d}{dt}\iiint_{V(t)} \rho \cdot u_i \cdot dV = \iiint_{V(t)} \frac{\partial(\rho \cdot u_i)}{\partial t} \cdot dV + \iint_{S(t)} \rho \cdot u_j \cdot u_i \cdot n_i \cdot dS$

Equation ID	**Momentum-3.2**
Parent ID	Momentum-3.1
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector.
Description of this step	Apply Gauss's Theorem to surface integral
Assumptions this step	• Volume must be infinitesimal.
Genesis Equation (vector)	$$\frac{d}{dt}\iiint_{V(t)} \rho\cdot\vec{u}\cdot dV = \iiint_{V(t)} \frac{\partial(\rho\cdot\vec{u})}{\partial t}\cdot dV + \iiint_{V(t)} \nabla\odot(\rho\cdot\vec{u}\cdot\vec{u})\cdot dV$$
Genesis Equation (indices)	$$\frac{d}{dt}\iiint_{V(t)} \rho\cdot u_i\cdot dV = \iiint_{V(t)} \frac{\partial(\rho\cdot u_i)}{\partial t}\cdot dV + \iiint_{V(t)} \frac{\partial}{\partial x_j}(\rho\cdot u_i\cdot u_j)dV$$

Equation ID	**Momentum-3.3**
Parent ID	Momentum-3.2
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Combine terms on the RHS inside the integral.
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{d}{dt}\iiint_{V(t)} \rho\cdot\vec{u}\cdot dV = \iiint_{V(t)} \left[\frac{\partial(\rho\cdot\vec{u})}{\partial t} + \nabla\odot(\rho\cdot\vec{u}\cdot\vec{u})\right] dV$$
Genesis Equation (indices)	$$\frac{d}{dt}\iiint_{V(t)} \rho\cdot u_i\cdot dV = \iiint_{V(t)} \left[\frac{\partial(\rho\cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\cdot u_i\cdot u_j)\right]\cdot dV$$

Equation ID	**Momentum-4**
Parent ID	Momentum-3 and Momentum 3.3
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Substitute the derivations of *Momentum-3.3* into *Momentum-3*.
Assumptions this step	• None.
Genesis Equation (vector)	$\iiint_{V(t)} \left[\frac{\partial(\rho\cdot\vec{u})}{\partial t} + \nabla\odot(\rho\cdot\vec{u}\cdot\vec{u})\right] dV = \iiint_{V(t)} \vec{b}\cdot dV + \iint_{S(t)} \vec{T}\cdot dS$
Genesis Equation (indices)	$\iiint_{V(t)} \left[\frac{\partial(\rho\cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\cdot u_i\cdot u_j)\right]\cdot dV = \iiint_{V(t)} b_i\cdot dV + \iint_{S(t)} T_i\cdot dS$

Equation ID	**Momentum-5**
Parent ID	Momentum-4
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Redefine the surface tension T so that it is equal to some σ times the unit normal at the surface n.
Assumptions this step	• None.
Genesis Equation (vector)	$\iiint_{V(t)} \left[\frac{\partial(\rho\cdot\vec{u})}{\partial t} + \nabla\odot(\rho\cdot\vec{u}\cdot\vec{u})\right] dV = \iiint_{V(t)} \vec{b}\cdot dV + \iint_{S(t)} \vec{\sigma}\odot\vec{n}\cdot dS$
Genesis Equation (indices)	$\iiint_{V(t)} \left[\frac{\partial(\rho\cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho\cdot u_i\cdot u_j)\right]\cdot dV = \iiint_{V(t)} b_i\cdot dV + \iint_{S(t)} \sigma_{ji}\cdot n_j\cdot dS$

Equation ID	**Momentum-5**
Parent ID	Momentum-4
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Apply Gauss's Theorem to the surface integral.
Assumptions this step	• Volume must be infinitesimal.
Genesis Equation (vector)	$$\iiint_{V(t)} \left[\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u})\right] dV = \iiint_{V(t)} \vec{b} \cdot dV + \iiint_{V(t)} \nabla \odot \vec{\sigma} \cdot dV$$
Genesis Equation (indices)	$$\iiint_{V(t)} \left[\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j)\right] \cdot dV = \iiint_{V(t)} b_i \cdot dV + \iiint_{V(t)} \frac{\partial \sigma_{ji}}{\partial x_j} dV$$

Equation ID	**Momentum-6**
Parent ID	Momentum-5
Starting Assumptions	• The velocity u is $u(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The density ρ is $\rho(\vec{\mathrm{x}},\mathrm{t})$, a continuous scalar field. • The body force b is $b(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field. • The surface tension T is $T(\vec{\mathrm{x}},\mathrm{t})$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Combine terms in the integral.
Assumptions this step	• None.
Genesis Equation (vector)	$$\iiint_{V(t)} \left[\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) - \vec{b} - \nabla \odot \vec{\sigma}\right] dV = 0$$
Genesis Equation (indices)	$$\iiint_{V(t)} \left[\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) - b_i - \frac{\partial \sigma_{ji}}{\partial x_j}\right] \cdot dV = 0$$

Equation ID	**Momentum-7**
Parent ID	Momentum-6
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Recognize that the term in the brackets must be zero for the integral to be zero.
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) - \vec{b} - \nabla \odot \vec{\sigma} = 0$$
Genesis Equation (indices)	$$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) - b_i - \frac{\partial \sigma_{ji}}{\partial x_j} = 0$$

Equation ID	**Momentum-8**
Parent ID	Momentum-7
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Use the moment of inertia of *Momentum-7* along with Reynolds Transport and Gauss to show that $\sigma_{ji} = \sigma_{ij}$.
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) - \vec{b} - \nabla \odot \vec{\sigma} = 0$$
Genesis Equation (indices)	$$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) - b_i - \frac{\partial \sigma_{ij}}{\partial x_j} = 0$$

Equation ID	**Momentum-9**
Parent ID	Momentum-8
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Define the mechanical pressure as the scalar $P = -{}^{1}/_{3} \cdot \sigma_{kk}$. Define the deviatoric stress as $\vec{d} = d_{ij} = \sigma_{ij} - {}^{1}/_{3} \cdot \sigma_{kk} \cdot \delta_{ij}$
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) = \vec{b} + \nabla \odot (\vec{d} - P \cdot \delta)$
Genesis Equation (indices)	$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) = b_i + \frac{\partial}{\partial x_j}(d_{ij} - P \cdot \delta_{ij})$

Equation ID	**Momentum-9.1**
Parent ID	Momentum-9
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Expand the LHS of *Momentum-9*
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) = \rho \cdot \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \frac{\partial \rho}{\partial t} + \rho \cdot \vec{u} \cdot (\nabla \odot \vec{u}) + \vec{u} \cdot [\nabla \odot (\rho \cdot \vec{u})]$
Genesis Equation (indices)	$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) = \rho \cdot \frac{\partial u_i}{\partial t} + u_i \cdot \frac{\partial \rho}{\partial t} + \rho \cdot u_j \cdot \frac{\partial}{\partial x_j}(u_i) + u_i \cdot \frac{\partial}{\partial x_j}(\rho \cdot u_j)$

Equation ID	**Momentum-9.2**
Parent ID	Momentum-9.1
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Rearrange the equation
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) = \rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] + \vec{u} \cdot \left[\frac{\partial \rho}{\partial t} + \nabla \odot (\rho \cdot \vec{u})\right]$$
Genesis Equation (indices)	$$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) = \rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] + u_i \cdot \left[\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_j)\right]$$

Equation ID	**Momentum-9.3**
Parent ID	Momentum-9.2
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Rearrange the term which is multiplying velocity on the RHS.
Assumptions this step	• None.
Genesis Equation (vector)	$$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) = \rho \cdot \left[\frac{\partial \rho}{\partial t} + \vec{u} \odot (\nabla \rho) + \rho \cdot (\nabla \odot \vec{u})\right] + \vec{u} \cdot [\vec{u} \odot (\nabla \rho) + \rho \cdot (\nabla \odot \vec{u})]$$
Genesis Equation (indices)	$$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) = \rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] + u_i \cdot \left[\frac{\partial \rho}{\partial t} + u_i \cdot \frac{\partial \rho}{\partial x_i} + \rho \cdot \frac{\partial u_i}{\partial x_i}\right]$$

D. Common Derivations of Pseudo-First Principle Equations

This Appendix provides some common derivations of equations related to first principle equations.

D.1. Common Derivations of the Navier-Stokes Equation

Equation ID	**NS-1**
Parent ID	Momentum-9.3 and Mass-6
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Remove *Mass-6* from *Momentum-9.3*
Assumptions this step	• None.
Genesis Equation (vector)	$\frac{\partial(\rho \cdot \vec{u})}{\partial t} + \nabla \odot (\rho \cdot \vec{u} \cdot \vec{u}) = \rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right]$
Genesis Equation (indices)	$\frac{\partial(\rho \cdot u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho \cdot u_i \cdot u_j) = \rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right]$

Equation ID	**NS-2**
Parent ID	NS-1 and Momentum-9
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Substitute the RHS of *NS-1* into the LHS of *Momentum-9*
Assumptions this step	• None.
Genesis Equation (vector)	$\rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] = \vec{b} + \nabla \odot \left(\vec{d} - P \cdot \delta\right)$
Genesis Equation (indices)	$\rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] = b_i + \frac{\partial}{\partial x_j}\left(d_{ij} - P \cdot \delta_{ij}\right)$

Equation ID	**NS-3**
Parent ID	NS-2
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Carry out the derivative on the stress term on the RHS of *NS-2*
Assumptions this step	• None.
Genesis Equation (vector)	$\rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] = \vec{b} + \nabla \odot \vec{d} - \nabla \odot (P \cdot \delta)$
Genesis Equation (indices)	$\rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] = b_i + \frac{\partial d_{ij}}{\partial x_j} - \frac{\partial\left(P \cdot \delta_{ij}\right)}{\partial x_j}$

Equation ID	**NS-4**
Parent ID	NS-3
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Reduce the pressure term on the RHS of *NS-3* using the definition of δ
Assumptions this step	• None.
Genesis Equation (vector)	$$\rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] = \vec{b} + \nabla \odot \vec{d} - \nabla P$$
Genesis Equation (indices)	$$\rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] = b_i + \frac{\partial d_{ij}}{\partial x_j} - \frac{\partial P}{\partial x_j}$$

Equation ID	**NS-5**
Parent ID	NS-4
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal.
Description of this step	Assume the fluid is a Newtonian fluid: $$d_{ij} = 2 \cdot \mu \cdot \left(e_{ij} - {}^{1}/_{3} \cdot e_{rr} \cdot \delta_{ij}\right)$$ Where $$e_{ij} = \frac{1}{2} \cdot \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$
Assumptions this step	• The fluid is a Newtonian fluid.
Genesis Equation (vector)	$$\rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] = -\nabla P + \vec{b} + 2 \cdot \mu \cdot \nabla \odot ??? [24]$$
Genesis Equation (indices)	$$\rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] = -\frac{\partial P}{\partial x_j} + b_i + 2 \cdot \mu \cdot \frac{\partial}{\partial x_j}\left(e_{ij} - {}^{1}/_{3} \cdot e_{rr} \cdot \delta_{ij}\right)$$

[24] This term step is not known by the author.

Equation ID	**NS-6**
Parent ID	NS-5
Starting Assumptions	• The velocity u is $u(\vec{x}, t)$, a continuous vector field. • The density ρ is $\rho(\vec{x}, t)$, a continuous scalar field. • The body force b is $b(\vec{x}, t)$, a continuous vector field. • The surface tension T is $T(\vec{x}, t)$, a continuous vector field over the surface. • Reynolds Transport Assumptions? • The vector n is a unit surface normal vector. • Volume must be infinitesimal. • The fluid is a Newtonian fluid.
Description of this step	Assume the flow is incompressible $(\nabla \odot \vec{u} = 0)$
Assumptions this step	• The flow is incompressible
Genesis Equation (vector)	$$\rho \cdot \left[\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot (\nabla \odot \vec{u})\right] = -\nabla P + \vec{b} + \mu \cdot (\nabla \odot \vec{u})^2$$
Genesis Equation (indices)	$$\rho \cdot \left[\frac{\partial u_i}{\partial t} + u_j \cdot \frac{\partial}{\partial x_j}(u_i)\right] = -\frac{\partial P}{\partial x_j} + b_i + \mu \cdot \frac{\partial^2 u_i}{\partial {x_j}^2}$$

Glossary/Index

Active Approach

To initially assume that the entire simulation is incorrect unless specific evidence suggests otherwise.

Page: 6

Actual Complete Set of Codes

The actual set of codes which is used to perform the simulation.

Page: 78

Aggregate and Abridge

The evolution process where two or more parent maturity attributes are aggregated and result in an assessment set whose criteria are an abridgment of the parent maturity assessment criteria.

Page: 129

Algorithm

An unambiguous description of a finite set of operations which specifics a sequence of operations that always halts (Brainerd and Landweber 1974).

Page: 57

Analyst

Any person involved in the simulation or simulation review process.

Page: 10

Application Domain

The space described by all of the independent variables in which the physical function is valid.

Page: 64

Application Range

The space described by all of the dependent variables in which the physical function is valid over.

Page: 64

Application Space

The mathematical space over which the physical equation is valid.

Page: 63

Arbitrary Constraint

Any constraint on the variable space which an analyst chooses to enforce. This constraint is not forced by any stipulations of mathematics or nature; rather it is one which the analyst arbitrarily chooses to enforce on the equation.

Page: 73

Assessed Maturity Level

The maturity level a specific attribute has obtained in a given maturity assessment set.

Page: 26

Assignment Statement

A set of terms which are used to assign a value to a specific dependent variable.

Page: 60

Assumed Definitional Physical Equation

A definitional physical equation which is assumed true for a particular instance.

Page: 70

Best Humanly Possible Simulation

A simulation performed using all the resources of humanity from now until its end as a species. In other words, suppose every human being decided that simulating one event was the most important thing in their life and everyone worked towards this goal for generations and generations. While such a simulation is obviously fictional, it is helpful because it represents an upper limit as no simulation could possibly be better than this one.

Page: 97

Children Components

Any components one level lower in the hierarchy which is contained in the component under consideration.

Page: 84

Coded Group

A set of one or more Coded Physical Functions.

Page: 74

Coded Physical Function

A single physical function which has been discretized and written as some combination of Assignment Statements in a specific computer language.

Page: 71

Combined Maturity Assessment Set

The assessment criteria which describes all maturity levels of that combined maturity attribute.

Page: 48

Combined Maturity Attribute

The attribute of the object which is being assessed in the combined process.

Page: 48

Combined Maturity Level

The assessed maturity level which is the result of the maturity combination process.

Page: 48

Complete Assessment Set

An assessment set whose criteria correspond to all possible maturity objects.

Page: 37

Complete Framework

A framework made up of complete assessment sets.

Page: 40

Complete Set of Codes

A set of one or more Computer Codes used to perform a simulation.

Page: 78

Complex Term

A term which can be further separated into other terms.

Page: 60

Computer Code

A set of one or more Coded Groups.

Page: 75

Computer Model

A mathematical model which has been re-written such that it can be solved by some device which has a limited mathematical capability.

Page: 4

Computer Simulation

An imitation of the behavior of something expressed as a mathematical model on a device with limited mathematical capability.

Page: 4

Constraint

A limitation placed on the variable space of an equation.

Page: 72

Coupled Components

Any component whose input is obtained from the output of any component (including itself); further, the coupling between two components extends to their parents.

Page: 89

Cumulative Criteria

Criteria written in such a way that an object which meets the criteria of any specific level of maturity will also meet the criteria of every lower level of maturity.

Page: 36

Creation from a Concept

The process of creating a new assessment set for a specific maturity attribute from a general concept.

Page: 134

Decision Maker

Any individual who answers Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*) in whole or in part.

Page: 7

Definitional Physical Equation

A physical equation whose form, constants, and variables are chosen to define some physical quantity.

Page: 69

Dependent Variables

The variables which the function is solved to obtain.

Page: 62

Detailed Assessment Set

An assessment set which has a sufficient number of distinct criteria such that important distinctions in maturity of the given attribute can be realized.

Page: 37

Detailed Framework

A framework made up of detailed assessment sets.

Page: 40

Discretization Constraint

Any constraint on the variable space that is the result of using a computer with finite precision to perform calculations which would otherwise be continuous.

Page: 73

Distinct Assessment Set

An assessment set made up of distinct criteria.

Page: 37

Distinct Criterion

A criterion which is from all other the other criteria in a given assessment set.

Page: 36

Distinct Framework

A framework made up of distinct assessment sets.

Page: 40

Divide and Distill

The evolution process where the parent maturity attribute is divided into multiple parts and criteria of the parent maturity assessment set are distilled to capture the levels of the new attributes.

Page: 130

Domain of the Function

The set of elements in $\mathbb{A}$, where each element x in a set $\mathbb{A}$ exactly one element, called f(x) in set $\mathbb{B}$ (Stewart 1995).

Page: 62

Dynamic Form of a Computer Code

The set of all Coded Physical Functions which are actually executed at some point during the execution of the code.

Page: 76

Dynamically Equivalent

Two Computer Codes are dynamically equivalent if they are first explicitly equivalent and second if the internal selections made in each code are identical. Dynamically equivalence can be a confusing concept, especially if the computer simulation is time dependent. If the simulations are time dependent, this raises the question of if the codes need to make the same internal selections at the same time step or just at some point during the code run. In general, the only way to ensure that a code is dynamically equivalent with another code is if those codes are explicitly equivalent and contain no internal selections.

Page: 78

Economical Assessment Set

An assessment set which can be performed at a relatively low cost.

Page: 38

Economical Framework

A framework made up of economical assessment sets.

Page: 40

Empirical Physical Equation

A physical equation whose form, constants, and variables are closely related to the experimental observation of some physical quantity.

Page: 68

Equation

A mathematical statement which states the equality of two sets of terms.

Page: 62

Essential Assumption Framework

A group of maturity assessment sets which have the common focus of determining of supporting a specific Essential Assumption.

Page: 117

Essential Assumptions of Scientific Computer Simulations

The set of assumptions which are both necessary and sufficient to the Fundamental Assumption.

Page: 110

Evolution of a Maturity Assessment Set

The process of creating a new assessment set from one or more known assessment sets.

Page: 129

Exact Assessment Set

An assessment set made up of exact criteria.

Page: 37

Exact Criterion

A criterion defined exactly with no need for interpretation.

Page: 37

Exact Framework

A framework made up of exact assessment sets.

Page: 40

Explicit Computer Code

A code where all necessary selections of Coded Physical Functions have been made.

Page: 76

Explicitly Equivalent

Two Computer Codes are explicitly equivalent if they are first statically equivalent and second if they have the same external selections made. Many QA programs are set up to make codes explicitly equivalent such that the same physical models are used in the same manner.

Page: 78

External Selection

When the external analyst chooses which Coded Physical Functions are exercised in a code run.

Page: 76

First Principle

A principle (i.e., law or concept) from a set of principles which is considered true by the scientific community under specific conditions.

Page: 65

First Principles Physical Equation

A physical equation which is a mathematical representation of a principle considered true by the scientific community under specific conditions.

Page: 65

Focus of the Framework

The aspect of the maturity object which each maturity assessment set is focused upon.

Page: 35

Focused Assessment Set

An assessment set where each of the criteria in the assessment is directly related to the maturity attribute.

Page: 38

Focused Framework

A framework made up of focused assessment sets.

Page: 40

Formal Maturity Assessment

An assessment whose assessment criteria are formally defined and captured in some fashion.

Page: 29

Four Stage of Scientific Computer Simulation Review

1. Maturity Framework Development – Develop a maturity assessment framework which captures all appropriate attributes and criteria.
2. Maturity Requirements Determination – Use the developed framework to determine the required maturity levels of each attribute for the particular use of the simulation.
3. Maturity Level Assessment – Use the developed framework to determine the achieved maturity level of each attribute for the specific simulation.
4. Maturity Judgment Stage – Compare the Maturity Assessments from stage 3 with Maturity Requirements from stage 2 to decide if the simulation is appropriate for the given use.

Page: 50

Fully Described Complete Set of Codes

A Complete Set of Codes, where the input to that complete set (external selections plus any other input) has been chosen or there is clear guidance which directs the analyst on how to choose the input.

Page: 79

Function

A rule that assigns to each element x in a set $\mathbb{A}$ exactly one element, called f(x) in set $\mathbb{B}$ (Stewart 1995).

Page: 62

Functionalization

The process of turning an equation into a function.

Page: 62

Fundamental Assumption of Scientific Computer Simulations

The assumption that the results of the scientific computer simulation are true (or true enough). That is, it is the assumption made when a decision maker answer "yes" to the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

Page: 109

Fundamental Assumption of Validation

A real physical quantity exists in the physical universe and can be predicted by a physical function in some application space.

Page: 162

Fundamental Assumption of Verification

A continuous function which exists in a continuous space can be predicted by a discretized function in a discretized space.

Page: 167

Fundamental Assumptions of Uncertainty Quantification

1. The Frist Fundamental Assumption of Uncertainty Quantification is: *The variability of the inputs of simulation are appropriately captured.*
2. The Second Fundamental Assumption of Uncertainty Quantification is: *The variability of the results of each Coded Physical Function in the simulation due to the variability in the inputs is appropriately captured.*
3. The Third Fundamental Assumption of Uncertainty Quantification is: *The variability of the results of each Coded Physical Function in the simulation due to the variability of the function itself is appropriately captured.*

Page: 171

Fundamental Question of a Scientific Computer Simulation

How trustworthy are the results of the specific scientific computer simulation?

Page: 6

Fundamental Question of Validation

Can a real physical quantity which exists in the real universe can be predicted by a physical function in some application space?

Page: 162

Fundamental Question of Verification

Can a continuous function which exists in a continuous space can be predicted by a discretized function in a discretized space?

Page: 167

Fundamental Question on the Requirements of a Scientific Computer Simulation

How trustworthy do the results of a scientific computer simulation need to be for the intended purpose?

Page: 7

Fundamental Questions of Uncertainty Quantification

1. The Frist Fundamental Question of Uncertainty Quantification is: *How much can the inputs to the simulation vary?*
2. The Second Fundamental Question of Uncertainty Quantification is: *How much can the results of each Coded Physical Function in the simulation vary due to variability in the input to that function?*
3. The Third Fundamental Question of Uncertainty Quantification is: *How much can the results of each Coded Physical Function in the simulation vary due to variability in the function itself?*

Page: 171

Fundamental Statement of Scientific Computer Simulations

The results of the scientific computer simulation are true. It should always be remembered that his statement is only definitionally true for Ideal Simulations.

Page: 99

Genesis Equation

The exact representation of the relationship described by the first principle in the form of an equation and contains the minimum number of additional assumptions necessary to represent that principle.

Page: 66

Hierarchy of Scientific Computer Simulation Components

A hierarchy which arranges the different components of a scientific computer simulation into six levels: Term, Assignment Statement, Coded Physical Function, Coded Group, Computer Code, and Complete Set of Codes.

Page: 59

High Consequence Purpose

When the intended purpose of the simulation is such that an incorrect simulation would likely result in significant loss of resources including human lives.

Page: 7

High Consequence Simulation

A simulation, which, if wrong, is likely to results in significant loss of resources including human lives.

Page: 7

Higher Level Language

The expression of an algorithm in a logical fashion which can be turned into machine language by a compiler.

Page: 57

Highest Maturity Level of Any Attribute

The level achieved if all of the resources of humanity were focused on performing a single simulation until the extinction of humans.

Page: 98

Ideal Simulation

A simulation which is performed perfectly and whose results are true. It is important to remember that even the Best Humanly Possible Simulation is likely to fall short of providing absolute truth. An Ideal Simulation would require complete and perfect knowledge. Humanity does not currently have this level of knowledge and it is likely that such a level of knowledge is not obtained during our existence.

Page: 98

Indefinite Computer Code

A code where the selection of at least one Coded Physical Function is required before the code will execute.

Page: 76

Independent Framework

A framework where the assessment sets are independent from each other.

Page: 40

Independent Variables

All variables in the function which are not the dependent variables; their values must be known before the function can be solved.

Page: 62

Informal Maturity Assessment

An assessment where the assessment criteria are not formally defined or captured in any fashion.

Page: 29

Input to the Scientific Computer Simulation

Any input to a Coded Physical Function which is not the output of any Coded Physical Function in the Complete Set of Codes.

Page: 89

Internal Selection

When the code itself chooses which coded physical Coded Physical Functions are exercised in a code run and there is no way to know by the input what those Coded Physical Functions would be.

Page: 76

Known Definitional Physical Equation

A definitional physical equation which is unconditionally true.

Page: 69

Low Consequence Purpose

When the intended purpose of the simulation is such that an incorrect simulation would likely result in only minor loss of resources.

Page: 7

Low Consequence Simulation

A simulation, which, if wrong, is likely to result in only minor loss of resources.

Page: 7

Machine Language

The sequence of bits that directly controls a processor, causing it to add, compare, move data from one place to another, and so forth at appropriate times (Scott 2009).

Page: 57

Mathematical Constraint

Any constraint on the variable space that arises from the rules of mathematics. The function itself can be thought of as a mathematical constraint on the range as the function limits the values the dependent variable may take.

Page: 72

Mathematical Model

A representation of the behavior of something using mathematical concepts, symbols, and relations.

Page: 3

Mathematical Simulation

An imitation of the behavior of something by mathematical means.

Page: 3

Maturity

A measurement of rank of an attribute in the spectrum of the very worst to the very best achievable values of that attribute.

Page: 35

Maturity Assessment

The act of using a specified maturity assessment set to determine the achieved maturity level for a specific maturity attribute of the given maturity object.

Page: 27

Maturity Assessment Framework

A set of maturity assessment sets where each assessment set shares a common focus with all of the other assessments sets in the group.

Page: 35

Maturity Assessment Set

The set of all of the maturity levels (or criteria) of a specific attribute of an object.

Page: 25

Maturity Attribute

The specific attribute of the maturity object that is being assessed.

Page: 25

Maturity Combination Process

A process used to combine maturity levels from one or more attributes of the same object to generate one maturity level.

Page: 48

Maturity Framework Development

The process of generating the maturity assessment framework which will be used for the review of the simulation.

Page: 50

Maturity Judgment

The process of determining if the required maturity levels of a situation are met by the assessed maturity levels of a simulation.

Page: 52

Maturity Level Assessment

The process of determining the maturity level of each attribute in the given framework for a specific simulation.

Page: 52

Maturity Object

The object whose maturity is being assessed.

Page: 25

Maturity Table

A table which contains the assessment set at a minimum and may contain any number of other important features such as the maturity object, the maturity attribute, and tracking information.

Page: 171

Measurement

The assignment of a number to an object in a systematic way as a means of representing properties of the object (Allen and Yen 1979).

Page: 24

Model

A representation of the behavior of something.

Page: 3

Module

A lexically contiguous sequence of program statements, bounded by elements, having an aggregate identifier (Yourdon and Constantine 1977).

Page: 58

National Simulation

A simulation performed by many organizations. National simulations take advantage of expertise and resources of multiple organizations up to and including that of entire nations. The Manhattan project is a good example of a national simulation as many organizations were needed to simulate an atomic bomb.

Page: 97

Necessary Assumptions of Validation

The assumptions which must be true if the Fundamental Assumption of Validation is true.

Page: 162

Necessary Assumptions of Verification

The assumptions which must be true if the Fundamental Assumption of Verification is true.

Page: 167

Necessary Condition

A circumstance in whose absence a specific event could not occur (Copi 1986).

Page: 94

Necessary Maturity Attribute

An attribute which must be assessed or else the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*) cannot be answered.

Page: 94

Non-cumulative Criteria

Criteria are written in such a way that an object which meets the criteria of any specific level will not meet the criteria of any level which is lower in maturity.

Page: 37

Objective Assessment Set

An assessment set where two different individuals given set and the same information about an object would determine the same maturity level.

Page: 38

Objective Framework

A framework made up of objective assessment sets.

Page: 40

Operands

The variables or constants employed in the specific implementation of the algorithm (Halstead 1977).

Page: 57

Operators

The symbols or combinations of symbols that affect the value or ordering of an operand (Halstead 1977).

Page: 57

Organizational Simulation

A simulation performed by many individuals or an organization. Organizational simulations take advantage of the expertise and resources available to an organization. This will likely result in a more extensive simulation which uses much more man power.

Page: 97

Output from the Scientific Computer Simulation

The output from every Coded Physical Function in the Complete Set of Codes.

Page: 89

Parent Component

The component one level higher in the hierarchy which contains the component under consideration.

Page: 84

Passive Approach

To initially assume that the simulation is correct unless specific evidence suggests otherwise.

Page: 6

Personal Simulation

A simulation performed by an individual. Personal simulations are limited by what a single person can do with the resources available to them. This is highly dependent on the individual and the available resources.

Page: 97

Physical Constraint

Any constraint on the constant space or variable space that is a result of the permissible values of the physical quantities in question. These constraints are often expressed in terms of the application space of the physical equation.

Page: 73

Physical Equation

An equation whose constants and variables are related to quantities in the physical universe and whose form is an attempt to define some relationship between those quantities under certain conditions.

Page: 62

Physical Function

The functionalization of the physical equation into a specific functional form.

Page: 63

Practical Maturity Assessment Framework

A formal framework developed with the intent to be used to assess the maturity of real-world scientific computer simulations.

Page: 92

Pseudo-First Principle Physical Equation

An equation which is based on a first principle equation, but includes the introduction of sub-models (see also Semi-Empirical Physical Equation).

Page: 66

Range of the Function

The set of elements in $\mathbb{B}$, where each element x in a set $\mathbb{A}$ exactly one element, called f(x) in set $\mathbb{B}$ (Stewart 1995).

Page: 62

Recognized Complete Set of Codes

The set of codes which the analyst believes have been used to perform the simulation.

Page: 79

Scientific Computer Model

A representation of the behavior of something in the natural universe through mathematical means on a device with limited mathematical capability.

Page: 4

Scientific Computer Simulation

An imitation of the behavior of something in the natural universe expressed as a mathematical model on a device with limited mathematical capability.

A Complete Set of Codes with one complete set of inputs and all resulting outputs.

Page: 4, 80

Scientific Computer Simulation Review

The process of determining how trustworthy the results of a scientific computer simulation are, how trustworthy the results need to be for some intended purpose, and based on this information if the specific simulation should be trusted for the intended purpose.

Page: 8

Scientific Knowledge

Knowledge generated solely for the improved understanding of the Universe and humankind's place in it.

Page: 4

Scientific Model

A representation of the behavior of something in the natural universe (e.g., using a string tied at one end to represent a sound wave).

Page: 3

Scientific Simulation

An imitation of the behavior of something in the natural universe (e.g., moving the string to imitate sound hitting a wall).

Page: 3

Selected Output from the Scientific Computer Simulation

Any output from any Coded Physical Function in the Complete Set of Codes which is used for some purpose by an analyst.

Page: 89

Semi-Empirical Physical Equation

An equation which is based on a first principle equation, but includes the introduction of sub-models (see also Pseudo-First Principle Physical Equation).

Page: 66

Set of All Possible Maturity Criteria

The set of all of the maturity criteria which correspond to any possible level of the specific attribute of the given object.

Page: 30

Set of Maturity Assessment Sets

A set of one or more maturity assessment sets, where each set is used to assess a different attribute of the same object.

Page: 35

Sibling Components

Any components which are in the same level as the component under consideration and share the same parent component.

Page: 84

Simple Term

A term which cannot be further separated into other terms (i.e., an operand).

Page: 60

Simulation

An imitation of the behavior of something.

Page: 3

Statement

The smallest unit of a program which is a well-defined, complete instruction, command, declaration, etc. (Yourdon and Constantine 1977).

Page: 58

Static Form of a Computer Code

The set of all Coded Physical Functions which exist in the Computer Code.

Page: 76

Statically Equivalent

Two Computer Codes are statically equivalent if they have the have the static form. That is, the codes have identical sets of Coded Physical Functions which are programmed to be executed in the same way. Typically, a code is only statically equivalent with the same version of itself, but could be equivalent with its expression in another computer language.

Page: 78

Sufficient Condition

A circumstance in whose presence a specific event must occur (Copi 1986).

Page: 94

Sufficient Maturity Attributes

The set of attributes which must be assed to completely address the Ultimate Question (*Can the results of the specific simulation be trusted for the intended purpose?*).

Page: 94

Supporting Evidence

Any evidence which supports the conclusion that the results of a scientific computer simulation are true (i.e., any evidence which supports the Fundamental Assumption).

Page: 110

Technical Knowledge

The generation of knowledge used in some way for the creation of applied knowledge or the creation of a new physical systems or processes.

Page: 4

Term

A defined to be an expression which is a separable part of an Assignment Statement (the next level in the hierarchy).

Page: 60

Thorough Framework

A framework whose focus has been completely defined by the attributes of the individual assessment set.

Page: 40

Ultimate Question of Scientific Computer Simulation

Can the results of the specific simulation be trusted for the intended purpose?

Page: 5

Variable Space

An equation is an N-dimensional space which contains all possible values of the variables for that equation, assuming there are N variables.

Page: 62

Well-Defined Assessment Set

An assessment set where the criteria completely capture all that is intended by the maturity attribute being assessed.

Page: 39

Well-Defined Framework

A framework made up of well-defined assessment sets.

Page: 40

Well-Spaced Assessment Set

An assessment set where the increase in maturity from one maturity level to the next is approximately the same.

Page: 39

Well-Spaced Framework

A framework made up of well-spaced assessment sets.

Page: 40

Bibliography

4 Cs Guide. (2006). Retrieved June 6, 2013 from http://gia4cs.gia.edu/en-us/four-cs-guide.htm

AIAA. (1998). *Guide for Verification and Validation of Computational Fluid Dynamics*. AIAA-G-077, Reston, VA, American Institute for Aeronautics and Astronautics

Allen, M. J., and W.M. Yen. (1979). *Introduction to Measurement Theory*. Monterey, California. Brooks/Cole Publishing Company

Apgar, V. (1953). "A proposal for a new method of evaluation of the newborn infant." *Current Researches in Anesthesia and Analgesia*, 32 (4): 260 – 267.

ASME. (2006). *Guide for verification and validation in computational solid mechanics*. ASME V&V 10-2006, New York, NY, American Society of Mechanical Engineers

ASME. (2009). *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*. ASME V&V 20-2009, New York, NY, American Society of Mechanical Engineers

Blattnig, S. R., J. M. Luckring, J. H. Morrison, A. J. Sylvester, R. K. Tripathi and T. A. Zang (2013). "NASA Standard for Models and Simulations: Philosophy and Requirements Overview." Journal of Aircraft 50(1): 20-28.

Bossel, H. (1994). *Modeling and Simulation*. 1st Ed., Wellesley, MA, A. K. Peters.

Box, G., and N. Draper (1987) *Empirical Modeling Building and Response Surfaces*. New York, Wiley

Boyack, B., et al. (1989). "Quantifying Reactor Safety Margins, Application of Code, Scaling, Applicability, and Uncertainty Evaluation Methodology to a Large Break, Loss-of-Coolant Accident," NUREG/CR-5349, USNRC

Brainerd, W.S, and L.H. Landweber. (1974). *Theory of Computation*. John Wiley and Sons, New York

CAIB (2003). *Columbia Accident Investigation Board Report, Volume 1*. Columbia Accident Investigation Board. *http://caib.nasa.gov*

CMMI (2010) *Capability Maturity Model Integration* http://cmmiinstitute.com/

Copi, I.M. (1986). *Introduction to Logic (Seventh Edition)*. New York, Macmillian Publishing Company

Feynman, R.P. (1985) "Richard Feynman Computer Heuristics Lecture." Online video clip. *YouTube*. YouTube, June 2, 2012. Web. May 14, 2013.

Feynman, R.P. (1986). *Report of the Presidential Commission on the Space Shuttle Challenger Accident, Volume 2, Appendix F – Personal Observations on the Reliability of the Shuttle*.

Ganz, R. (1903). "Über die numerische Auflösung von partiellen Differentialgleichungen," *Zeitschrift Math. u. Phys*. No. 48

GAO (1999). *Best Practices: Better Management of Technology Development Can Improve Weapon System Outcomes*. GAO/NSIAD-990162, Washington, DC, U.S. Government Accounting Office

Halstead, M.H. (1977). *Elements of Software Science*. Elsevier, New York

Harmon, S.Y. and S.M. Youngblood (2003). A proposed model for simulation validation process maturity. *Simulation Interoperability Workshop*, Paper No. 03S-SIW-127, Orlando, FL, Simulation Interoperability Standards Organization.

Harmon, S.Y. and S.M. Youngblood (2005). A proposed model for simulation validation process maturity. *The Journal of Defense Modeling and Simulation*. **2**(4), 179-190.

Hierarchy. (d.) (2013). In *Wikipedia*. Retrieved June 19, 2013, from http://en.wikipedia.org/wiki/Hierarchy

Huen, K. (1900). "Nue methode zur approximativen Integration der Differentialgleichungen einer unabhängigen Verä," *Zeitschrift Math. u. Phys*. No. 45

Kundu, P.K. and I.M. Cohen (2004). *Fluid Mechanics: 3rd Edition*, Elsevier Academic Press, Amsterdam

Kutta, W. (1901). "Beitrag zur näherungsweisen Integration totaler Differentialgleichen," *Zeitschrift Math. u. Phys*. No. 46

Nickerson, R. S. (1998). "Confirmation bias: A ubiquitous phenomenon in many guises," *Review of General Psychology*, Volume 2, No2, pp. 175-220.

Maki, D. P. and M. Thompson (2006). *Mathematical Modeling and Simulation*, Belmont, CA, Thomson Brooks/Cole.

Maturity. (d.) (2013). *In Merriam-Webster*. Retrieved March 5, 2013, from http://www.merriam-webster.com/dictionary/maturity

Model. (d.) (2013). *In Wikipedia*. Retrieved February 27, 2013, from http://en.wikipedia.org/wiki/Model

Miller, G.A. (1956). "The magic number seven, plus or minus two: some limits on our capacity for processing information." *Psychological Review*., vol 63, pp. 81 – 97.

NASA (2006). *Standard for Models and Simulations*. NASA-STD-(I)-7009, Washington, DC, National Aeronautics and Space Administration.

NASA[1] (2008). *Standard for Models and Simulations*. NASA-STD-7009, Washington, DC, National Aeronautics and Space Administration.

NASA[2] (2008). Towards a Credibility Assessment of Models and Simulations. *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. AIAA-2008-2156.

NASA (2009). *NASA Standard for Models and Simulations (M&S): Development Process and Rationale*. NASA/TM-2009-215775, Washington, DC, National Aeronautics and Space Administration.

Neelamkavil, F. (1987). *Computer Simulation and Modelling*. 1st Ed., New York, John Wiley.

NRC (1989). *Best Estimate Calculation of Emergency Core Cooling System Performance.* Regulatory Guide 1.157. Washington D.C., U.S. Nuclear Regulatory Commission

NRC[1] (2000). *Transient and Accident Analysis Methods.* Regulatory Guide 1.203 DRAFT, Washington D.C., U.S. Nuclear Regulatory Commission

NRC[2] (2000). *Review of Transient and Accident Analysis Methods*. Standard Review Plan Section 15.0.2 of NUREG-0800, DRAFT, Washington D.C., U.S. Nuclear Regulatory Commission

NRC[1] (2005). *Transient and Accident Analysis Methods.* Regulatory Guide 1.203, Washington D.C., U.S. Nuclear Regulatory Commission

NRC (2007). *Review of Transient and Accident Analysis Methods*. Standard Review Plan Section 15.0.2 of NUREG-0800, Washington D.C., U.S. Nuclear Regulatory Commission

Oberkampf, W.L., M. Pilch, and T.G. Trucano (2007). *Predictive Capability Maturity Model for Computational Modeling and Simulation*. SAND2007-5948, Albuquerque, NM, Sandia National Laboratories.

Oberkampf, W.L, and C.J. Roy. (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge

Pilch, M., T.G., Trucano, D.E. Peercy, A.L. Hodges, and G.K. Froehlich (2004). *Concepts for Stockpile Computing (OUO)*. SAND2004-2479 (Restricted Distribution, Official Use Only), Albuquerque, NM, Sandia National Laboratories.

Richardson, L.F. (1911). The Approximate Arithmetic 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* **210**, 301-357.

Roache (1998). *Verification and Validation in Computational Science and Engineering*. Hermosa, New Mexico

Roache (2009). *Fundamentals of Verification and Validation*. Hermosa, New Mexico

Runge (1895). "Über die numerische Auflösung von Differentialgleichungen," 'Math. Ann.,' Bd.. 46. Leipzig.

Scott, M.L. (2009). *Progamming Language Pragmatics*. Morgan Kaufman Publishers, Burlington, Massachusetts

Sheppard, W.F. (1899). "A Method for Extending the Accuracy of Mathematical Tables," *Proceedings of the London Mathematical Society*. s1-31(1): 70-99

Smyth, H.D. (1948). *Atomic Energy for Military Purposes*. Princeton University Press, Princeton

Stewart, J. (1995). *Calculus: 3rd Edition*. Brooks/Cole Publishing Company, Pacific Grove

Yourdon, E., and L.L. Constantine. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey

Zeigler, B. P., H. Praehofer and T. G. Kim (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2nd Ed., San Diego, CA, Academic Press.