Contents

Chapter 1

Verification and Validation

Computer simulations are in many engineering applications a cost-efficient way for conducting design and performance optimalization of physical problems. However, thrusting blindly numbers generated from a computer code can prove to be naive. It doesn't take a lot of coding experience before one realizes the many things that can brake down and produce unwanted or unexpected results. Therefore, *credability* of computational results are essential, meaning the simulation is worthy of belief or confidence [3]. *Verification and validation* (V&V) is the main approach for assessing and the reliability of computational simulations [10]. A thorough discussion of (V&V) concepts and terminology during the last century can be found in [3]. In this thesis, the definitions provided by the *American Society of Mechanical Engineers guide for Verification and Validation in Computational Solid Mechanics* [8] are followed.

Definition 1.1. Verification: The process of determining that a computational model accurately represents the underlying mathematical model and its solution.

Definition 1.2. Validation: The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

Simplified *verification* considers if one solves the equations right, while *validation* is checking if one solves the right equations for the given problem [5].
To test a computational code for all possible parameters, conditions and applications are simply too time consuming. Verification and validation are therefore ongoing processes, with no clear boundary of completeness unless additional requirements are specified [5]. The goal of this chapter is to verify our implementations using the method of manufactured solution (MMS), addressing validation in a later chapter.

1.1 Verification of Code

Within scientific computing a mathematical model is often the baseline for simulations of a particular problem of interest. For scientists exploring physical phenomena, the mathematical model is often on the form of systems of partial differential equations (PDE´s). A computer program therefore must evaluate mathematical

identities such a differential operators and functions in order to produce accurate solutions of the governing PDE´s. Through verification of code, the ultimate goal is to ensure a computer program truly represents the mathematical model. To accumulate sufficient evidence that a mathematical model is solved correctly by a computer code, it must excel within predefined criteria. If the acceptance criterion is not satisfied, a coding mistake is suspected. Should the code pass the preset criteria, the code is considered verified. Of the different classes of test found in [5], *Order-of-accuracy* (OAA) is regarded as the most rigorous acceptance criterion for verification [7], [5], [2]. In addition to error estimation and convergence of the numerical solution, the method ensure the discretization error E is reduced in coordinance with the *formal order of accuracy* expected from the numerical scheme. The formal order of accuracy is defined to be the theoretical rate at which the truncation error of a numerical scheme is expected to reduce. The *observed order of accuracy* is the actual rate produced by our code. The order of convergence is calculated xAssuming a PDE of space and time, order-of-accuracy tests are conducted separatly of
By monitoring the dicretization error E by spatial and temporal refinements, one assumes the asymptotic behavior,

$$E = E_x + E_t = u_e - u_h = C\Delta t^p + D\Delta x^l$$

where C is a constant, h represents the spatial or temporal resolution, and p is the convergence rate of the numerical scheme. For order of convergence tests, the code is assumed to be verified if the discretization error is proportional to h^p.

1.1.1 Method of manufactured solution

The basis of a convergence test is how to find an exact/reference solution, in order to compute the discretization error E. However solutions of PDE´s are limited, and often simplifications of the original problem are needed to produce analytically solutions. *The method of manufactured solutions* provides a simple yet robust way of making analytic solutions for PDE´s. Let a partial differential equation of interest be on the form

$$\mathbf{L}(\mathbf{u}) = \mathbf{f}$$

Here $\mathbf{L}$ is a differential operator, $\mathbf{u}$ is variable the of interest, and $\mathbf{f}$ is some sourceterm. In the method of manufactured solution one first manufactures a solution $\mathbf{u}$ for the given problem. In general, the choice of $\mathbf{u}$ will not satisfy the governing equations, producing a sourceterm $\mathbf{f}$ after differentiation by $\mathbf{L}$. The produced source term will cancel any imbalance formed by the manufactured solution $\mathbf{u}$ of the original problem. Therefore, the manufactured solution can be constructed without any physical reasoning, proving code verificaion as a purely a mathematical exercise were our only interest is to verify the solution [4]. If the MMS is not chosen properly the test will not work, therfore some guidelines for rigirous verification have been proposed in [2, 7, 4].

- The manufactured solution (MS), should be composed of smooth analytic functions such as exponential, trigonometric, or polynomials.
- The MS should should have sufficient number of derivatives, exercising all terms and derivatives of the PDE´s.

To deeply verify the robustness of the method of manufactured solution, a report regarding code verification by MMS for CFD was published by Salari and Knupp [7]. This thorough work applied the method for both compressible and incompressible time-dependent Navier-Stokes equation. To prove its robustness the authors deliberate implemented code errors in a verified Navier-Stokes solver by MMS presented in the report. In total 21 blind testcases where implemented, where different approaches of verification frameworks were tested. Of these, 10 coding mistakes that reduces the observed order-of-accuracy was implemented. Here the method of manufactured solution captured all coding mistakes, except one. This mistake would, accordingly to the co-author , been captured if his guidelines for conducting MMS had been followed. In general, computing the source term f can be quite challenging and error prone. Therefore, symbolic computation of the sourceterm is advantigous to overcome mistakes which can easily occur when calulating by hand. For construction of the sourceterm **f**, the Unified Form Language (UFL) [1] provided in FEniCS Project will be used.

1.1.2 Verification of the fluid-structure interaction solver by MMS

In general the MMS does not need to match any physical capabilities. However, when considering multiphysics problems, such as FSI, the equations has to meet the mathematical criteria of the interface.

1. Kinematic boundary condition $\hat{\mathbf{v}}_s = \hat{\mathbf{v}}_f$, enforced strongly by a continious velocity field in the fluid and solid domain.
2. Dynamic boundary condition $\sigma_s \cdot \mathbf{n} = \sigma_f \cdot \mathbf{n}$, enforced weakly by omitting the boundary integrals from the weak formulation in problem.

The choice of a MMS is therefore not trivial, as it must fulfill condition 1 and 2, in addition to the divergence-free condition in the fluid, and avoiding cancellation of the ALE-convective term $/pder\hat{T}_f t$. The struggle is reflected of the abscence of research, regarding MMS for coupled FSI solvers in the litterature. The challenge are often disregarded, such as [9], where the verification process is conducted on the fluid and structure solver separately. Instead, the correctness of the coupling is evaluated by the code validation. The approach clearly ease the process, assuming verification of each codeblock is "sufficient" to declare the code verified. It must be stressed that solving each problem individually is not true verification, in reference to a monolithic approach where the problems are solved at the same time.
The construction of a MMS for a monolithic FSI problem is therefore out of the scope of this thesis. Conducting verification on the fluid and structure separatly is not , but considered "good enough" to show the mathematical model is discretized accuratly.

1.2 Validation

Through *verification*, one can assure that a scientific code implements a mathematical model correctly. However, c orrectness is unnecessarly, if the model fails to serve as an accurate representation of the physical problem of interest. By definition 1.2, *Validation* is the act of demonstrating that a mathematical model is applicable for its intended use with a certain degree of accuracy. This demonstration is not intended to portray the model as an absolute truth, nor the best model available [6]. The acceptence criteria of validation is based on the numerical solution produced, by comparison with existing experiment data. The dependency of thrusting experiments, makes *validation* assess a wide range of issuses [10]

- The relevance of mathematical model compared to the experiment.
- Assurence of that the experiments was conducted correctly, in accordinance with prescribed parameters, initial and boundary conditions e.t
- Uncertainty of experimental measurements

Comparing numerical results with existing experiments,raise some issues in the validation process. First, reproducing of experimental resuts... This is a test to so

Bibliography

[1] Martin Alnes, Anders Logg, Kristian Olgaard, Marie Rognes, and Garth Wells. Unified Form Language: A domain-specific language for weak formulations of partial differential equations. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua(212):4207–4211, 2015.

[2] Stéphane Étienne, D Tremblay, and Dominique Pelletier. Code Verification and the Method of Manufactured Solutions for Fluid-Structure Interaction Problems. *36th AIAA Fluid Dynamics Conference and Exhibit*, (June):1–11, 2006.

[3] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing.* Cambridge University Press, Cambridge, 2010.

[4] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, 124(1):4, 2002.

[5] P.J. Roache. *Verification and Validation in Computational Science and Engineering.* Computing in Science Engineering, Hermosa Publishers, 1998, 8-9, 1998.

[6] Edward J. Rykiel. Testing ecological models: The meaning of validation. *Ecological Modelling*, 90(3):229–244, 1996.

[7] Kambiz Salari and Patrick Knupp. Code Verification by the Method of Manufactured Solution. Technical report, Sandia National Laboratories, 2000.

[8] LE Schwer. Guide for verification and validation in computational solid mechanics. *American Society of Mechanical Engineers*, PTC 60(V&V 10):1–15, 2006.

[9] Jason P Sheldon, Scott T Miller, and Jonathan S Pitt. Methodology for Comparing Coupling Algorithms for Fluid-Structure Interaction Problems. *World Journal of Mechanics*, 4(February):54–70, 2014.

[10] Ian Sommerville. Verification and Validation. Technical Report February, 2006.