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 [1]. *Verification and validation* (VV) is the main approach for assessing and the reliability of computational simulations [6]. The terminology of (VV) have proven unconsistent across differnt engineering disciplines due to the variety of views regarding the fundaments of the method. A thorough review considering the development of (VV) concepts and terminology during the last century are studied in [1], where several attempts of definitions of *verification* and *validation* by different scientific communities are considered. In this thesis, the definitions provided by the *American Society of Mechanical Engineers guide for Verification and Validation in Computational Solid Mechanics* [**?**] 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. [4]
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 [4].
We will in this thesis use the more detailed description found in [4].

> The code author defines precisely what continuum partial differential equations and continuum boundary conditions are being solved, and convincingly demonstrates that they are solved correctly, i.e., usually with some order of accuracy, and always consistently, so that as some

> measure of discretization (e.g. the mesh increments) $\nabla \to 0$, the code produces a solution to the continuum equations; this is Verification.
>
> — *Roache, P.J.*

The goal of this chapter is to verify our implementations using the method of manufactured solution (MMS).

1.1 Verification of Code by the method of manufactured solutions

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, forming a basis for the given problem. To measure accuracy, an exact solution for the given problem is necessarily. However accurate solutions of PDE´s are often limited, and often simplifications of the original problem are needed in order to produce analytically solutions for comparison.
The method of manufactured solutions provides a simple yet robust way of making analytic solutions for PDE´s. The problem is not attacked in a traditional way, by finding an analytic solution by retaining the overall physics defined by the model. 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. The beauty of such an approach is that the manufactured solution can be constructed without any physical reasoning, proving code verificaion as a purely a mathematical exercise were we are only interested if we are solving our equation right [3]. Even though the manufactured solution $\mathbf{u}$ can be chosen independently, some guidelines have been proposed for rigirous verification([**?**], [5], [3]).

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

The guidelines presented are not limitation for choosing a MS, but rather improvements for ensuring the representation of the mathematical model is thoroughly tested.
The sourceterm $\mathbf{f}$ with respect to the selected solution $\mathbf{u}$ is then used as input in the implementation, yielding a numerical solution. Verification of code and calculation is then performed on the numerical solution against the manifactured solution $\mathbf{u}$.

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 [5]. 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 of them.

Verification can be devided into *verification of code* and *verification of calculation* [[3] [1]
Several *criteria* for evuationthe accuracy can be chosen for verification [5]. 1,2,3,4,5. NEVN ALLE OF LEGG VEKT PÅ DE TO SISTE , KAP 5 boka
For the purpose of verification of calculation we need to calculate the error of our numerical simulation. Let $\mathbf{u}_h$ denote our numerical solution and $\mathbf{u}$ be our exact solution. By letting $||\cdot||$ be the L^2 norm, we define the error as

$$E = ||\mathbf{u} - \mathbf{u}_h||$$

Assuming our computational mesh is constructed by equilateral triangles, and that our simulations are solved with a constant timestep, the total error contribution from the temporal and spatial discretized PDE can be written as

$$E = A\delta x^l + B\delta t^k$$

Where A and B are constants, and l and k denote the expected convergencerate... FYLL INN REF FRA ANNET KAP OM EXPECTED CONVERGENCERATE
. In order to evalute properties of either the spatial or temporal discretization, we must reduce the numerical error contribution of the discretization not of interest. Say we would like to evaluate the convergencerate of the spatial discretization, then the temporal error must be reduced in order to not poute..
Even though the method of MMS a certain freedom in the construction of a manufactured solution, certain guidelines have been proposed ([7], [5], [3]).

- To ensure theoretical order-of-accuracy, the manufactured solution should be constructed of polynomials, exponential or trigonometric functions to construct smooth solutions.

- The solution should be utilized by every term in the PDE of interest, such that no term yields zero. (få frem at en løsning må velges slik at ingen differentials blir 0

- Certain degree to be able to calculate expected order of convergence (Få frem at må ha "grad nok" til å kunne regne convergencerate)

Fluid structure interaction consists of several buildingblocks of fluid and structure equations describing forces exerted from one another. With this in mind a verification of the full FSI code can be tedious as implementation errors yielding non-desired results can be hard to find. We will therefore provide verification of each buildingblock until we reach the total system of equations.

In the following sections we will overlook the implemented solvers. Unlesss specified, all simulations are implemented on an unit square. Simulation parameters will be reported,
For construction of the sourceterm **f** the Unified Form Language (UFL) [2] provided in FEniCS Project will be used. UFL provides a simple yet powerfull method of declaration for finite element forms. An example will be provided in the Fluid Problem section.

1.1.1 Fluid Problem

One question which arises during the construction of the manufactured solution is, which formulation of the Navier-Stokes equation do we want to calculate the sourceterm. From a numerical point of view constructing the sourceterm from the Eulerian formulation and then map the equation would be feasable. Such an apporach limits the evaluation of computational demanding routines such as the generation of the deformation gradient $\hat{F}$ and its Jacobian $\hat{\mathrm{J}}$. Even though refinement studies of spatial and temporal discretizations are often computed on small problems, such speed-ups are important when running larger simulations. Recall from Chapter ??? the ALE formulation of the Navier Stokes equation.

$$\rho_f \hat{\mathrm{J}} \frac{\partial \hat{u}}{\partial t} + \hat{\mathrm{J}} \hat{F}^{-1}(\hat{u} - \hat{w}) \cdot \nabla \hat{u} - \nabla \cdot \hat{\mathrm{J}} \sigma \hat{F}^{-T} = f$$
$$\hat{\mathrm{div}}(\hat{\mathrm{J}} \hat{F}^{-1} \hat{\mathrm{u}}) = 0$$

Algorithm 1.1: Descriptive Caption Text

```
u_x = "cos(x[0])*sin(x[1])*cos(t_)"
u_y = "-sin(x[0])*cos(x[1])*cos(t_)"
p_c = "sin(x[0])*cos(x[1])*cos(t_)"

f = rho*diff(u_vec, t_) + rho*dot(grad(u_vec), (u_vec - w_vec)) -
div(sigma_f(p_c, u_vec, mu))
```

We will on the basis of the presented guidelines define the manufactured solution.

$$u = sin(x + y + t)^2$$
$$v = cos(x + y + t)^2$$
$$p = cos(x + y + t)$$

1.2 Validation of Code

From a thorough process of verifying our code, we can pursue validation activities on the assumption that our computational model compute accurate solutions. As we have experienced verification of code can be a tedious task, but its complexity is reduced to issues of mathematical and numerical nature. When it comes to validation on the other hand, numerous potential problems must be assessed. Does the mathematical model describe the the physical process of interest? What about the influence of of experimental measurement methods and their uncertainty ? But a as pointed out by /oberkampf. nevn paper of bok , a successful validation rise thrust in our mathematical computation... FIX
In the literature several different and often conflicting definitions of validation have been proposed /Rykiel . In his thorough work Rykiel also points out a main concern over that in all the confusion, there has never been more arising demands of validating models describing the real world.

/Refsgaard and Henriksen (2004) has propsed the following definition which we will use

Definition 1.3. Model Validation:
Substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model

This means as /Rykiel earlier proposed, the method of validation is not some method "for certifying the truth of current scientific understanding ... Validation means a model is acceptable for its intended use it meets specific performance requirements".
SKRIV NOE OM TUREK HVA HVA DET EXPERIMENTET TESTER OSV

Bibliography

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

[2] Fenics Project. Unified Form Language (UFL) Documentation. 2016.

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

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

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

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

[7] Stanly Steinberg and Patrick J. Roache. Symbolic manipulation and computational fluid dynamics. *Journal of Computational Physics*, 57(2):251–284, 1985.