What does computers do better than humans? What is it possible to compute? These questions have not been fully answered yet, and in the coming years we will most likely see that the boundaries for what computers can do will expand significantly. Many of the fundamental laws in nature have been known for quite some time, but still it is almost impossible to predict the behavior of water (H_2O) from quantum mechanics. The most sophisticated super computers runs for days and are only able to simulate the behavior of molecules in a couple of seconds, almost too short to extract meaningful thermodynamic properties. This leads to another interesting question: What does humans do better than machines? A large part of the answer to this question is *modeling*. Modeling is the ability to break a complicated, unstructured problem into smaller pieces that can be solved by computers or by other means. Modeling requires *domain knowledge*, one need to understand the system well enough to make the correct or the most efficient simplifications. The process usually starts with some experimental data that one would like to understand, it could be the increasing temperature in the atmosphere or sea, it could be changes in the chemical composition of a fluid passing through a rock. The modeler then makes a mental image, which includes a set of mechanisms that could be the cause of the observed data. These mechanisms then needs to be formulated mathematically. How can we know if a model of a system is good? First of all, a good model is a model that do not break any of the fundamental laws of nature, such as mass (assuming non relativistic effects) and energy conservation. Even if you are searching for new laws of nature, you have to make sure that your model respect the existing laws, because then a deviation from your model and the observations could be a hint of the new physics you are searching for. Secondly, the model must be able to match the observable data, with a limited set of variables. The variables should be determined from data, and then the model should be able to make some predictions that can be tested. Thus, the true purpose of the model is not only to match experimental data, but serve as a framework where the underlying mechanisms of the process can be understood. This is done by making model predictions, test them, and improve the model.

In this course our main focus will be on how to use computers to solve models. We will show you through numerical projects how a mathematical model of a physical system can be made, and you will have the possibility to explore the model. Computers are extremely useful, they can solve problems that would be impossible to solve by hand. However, it is extremely important to know about the limitations and strength of various algorithms. One need to have a toolbox of various algorithms that can be employed depending on the problem one are studying. Sometimes speed is not an issue, and one can use simpler algorithms, but in many cases *speed is an issue*. Thus it is important to not waste computational time when it is not needed, we will encounter examples of this many times in this course.

Why should you spend time learning about algorithms that have been implemented already in a software that most likely can be downloaded for free? There are many answers to this question, some more practical and some that goes deeper. Lets start with the practical considerations: Often you encounter a

problem that needs to be solved by a computer, it could be to fit an analytical model to data. Once you have this problem, one could ask Mr. Google for a solution, after a web search you will quickly realize that there are numerous ways of achieving what you want. By educating yourself within the most basic numerical methods, presented in this course, you will be able to judge for yourself which method to use in a specific case. If you choose a method suggested on the web, and try it out, it might happen that it works quite fine in many cases. But sooner or later you will encounter a problem that the computer struggles with, and it is very useful to have some skills within numerical analysis which can be helpful in identifying why the numerical method does not find a solution in some cases. If you know the why, it is much easier to find a cure. Another motivation is that development of most of the different numerical methods are *not that difficult*, they usually follow a very similar pattern, but there are some "tricks". It is extremely useful to learn these tricks, they can be adopted to a range of different problems, many are easily implemented in a spreadsheet. There are some more deeper arguments, and that is that the numerical methods are developed to solve a *general* problem. Most of the time we work with *specific* problems, and we would like to have an algorithm that is optimal for our problem that goes beyond only choosing the right one. Having understood and learned all the cool tricks that was used in the development of the algorithm in the general case, it is an excellent starting point for adopting the algorithm to your specific situation. Secondly development of an algorithm is a concrete case of *Computational Thinking*. Computational thinking is not necessarily related to computers and programming, but it is a way of structuring your work into precise statements that are being executed one at a time in a specific order. By learning about algorithmic development, you will train yourself in the art of computational thinking, which is a useful skill in all kind of problem solving.

July 2020 *Aksel Hiorth*