Preface

I have been planning for this book for a long time now. However, when I started to write this book using document editor, it seemed a bit out of place. As and when I moved to write the code for this book, it led to a disconnect between the book and the code. Moving to GitHub, or for that matter any git hosting website, seems like a good marriage between the code and the accompanying text. This will also preserve the timeline and therefore reflect the reasons for which the code or the text was modified. Writing this book as an open source project will also provide with an opportunity for collaboration and hence iron out any errors or provide improvements and additional features.

This book is a story of a journey through writing CFD software. Hence, it will present things which you may not find in a conventional book on CFD. For example, if some change is made in an existing API at any point in time, then you can see the reasons as to why they were made from a theoretical point of view. Therefore, scanning through the commits and its messages will tell a story on its own.

The upcoming text is just a speculation and will be modified from time to time. So read without any real expectations ...

The book begins with an introduction to CFD and its applications. The second chapter will take a simple example and show what it means to solve a problem numerically. The initial programs will be written using Python as a programming language. The next chapter will be on basics of finite volume method from the stand point of solving hyperbolic problems. The basic data structures for CFD and the basic algorithm will be introduced after this. The code and chapters of the book will evolve into high order schemes and other concepts like TVD and time integration. Various high order schemes will be introduced and added into the solver at this point.

Next, the concept of hyperbolicity and Riemann solvers will be introduced. Various Riemann solvers like Rusanov, HLL, HLLC, Roe solver will be briefly explained and implemented in the code. Since we are focusing on generic software for solving problems and PDEs from various fields, we will restrict implementations to a few solvers only.

After this we will move to multiple equations and extend our finite volume method to tackle system of partial differential equations. Also, here I will move to unstructured mesh in multiple dimensions. The idea will be to write an abstract code capable of solving 1D, 2D and 3D problems efficiently without changing any thing inside the code. Therefore, at this point I will start using Java or C++ as a programming language. This code will incorporate most of the features required for solving various types of coupled physics problems. The input mesh and solution output will be an abstract specification and hence anyone can write their format reader and writer for their favourite mesh generation or post processing software. I will however write concrete implementations for few formats commonly used.

Probably, at this point we will start looking at parallel computing for using the ever growing compute cores in our computers. Porting the code for GPGPU, MPI and multi thread computing will be focused on.

Specifying the inputs for a solver of such complexity (with many options) is never fun. Therefore, we will build a graphical user interface for interacting with the CFD code. We will learn about the MVC framework and how to incorporate it into an existing code base. Another reason for the GUI is to eliminate incompatible parameters right at the beginning and provide with default options so that we don't have to think about trivial things every time we try to solve the problem at hand. The graphics will also provide a visual feedback on the mesh geometry and intermediate evolution of the solution. Providing a full fledged pre/post processor is not a goal, so you will obviously have to rely on other software for generation of the mesh and post processing of the results.

At this point it may seem like the end. But really this will be the beginning. The code will provide with infinite experiment opportunities (numerical), enhancements for performance etc. I cannot even speculate at this time. So let's leave it to the future to decide as to how this story continues...