The Journal of Open Source Education

A course on the implicit finite volume method for CFD using Python

Christopher T. DeGroot[1]

1 Department of Mechanical and Materials Engineering, Western University, London, Ontario, Canada

DOI: 10.21105/jose.00066

Software

- Review
- Repository
- Archive

Submitted: 29 July 2019
Published: 01 August 2019

Introduction

This paper presents an open source learning module suitable for a semester-long graduate course in Computational Fluid Dynamics (CFD). This learning module is focused specifically on the implicit finite volume method (FVM) and follows generally from the approach of Patankar (Patankar, 1980), updated to reflect more recent developments to the field. In addition to segregated approaches for pressure-velocity coupling, this course introduces a fully-coupled method. The code is primarily presented for 1-dimensional problems to keep code manageable and runtimes low, although the extension to higher dimensions and unstructured grids are presented conceptually.

Statement of Need

Most engineering graduate programs at universities offer at least one course in CFD, many of which focus on FVM, since this is the basis of the most commonly used general-purpose CFD codes (e.g. OpenFOAM, ANSYS Fluent, etc.). There are many textbooks available on the subject that outline the details of the finite volume formulation (Patankar, 1980,Versteeg & Malalasekera (2007),Ferziger & Perić (2002)). However, there is a gap between reading about the discretization theory and actually implementing it into a computational code. This learning module emphasizes these implementation details which will help students to be able to (i) write their own codes, (ii) modify existing open source codes, and (iii) better understand the underlying implementation of CFD codes they may be using to solve problems. The purpose of this learning module is similar to that of the "CFD Python" module (Barba & Forsyth, 2018), except that it uses different discretization methods. While CFD Python uses finite differences in space and explicit time integration, this module uses the FVM for spatial discretization and implicit time integration.

Learning Objectives

An overall learning objective, besides the technical CFD content of the learning module, is to increase students' competencies in coding and management of source codes. Specific learning objectives for each lesson in the learning module is summarized as:

- Lesson 1 – Introduction to the Finite Volume Method: Introduction to the finite volume discretization method. Application of FVM to generic scalar equation,

mass/momentum/energy conservation. Concept of linearization. Uncertainty and Error. No assignment included.

- Lesson 2 – Steady-State Diffusion: Solution of the steady, one-dimensional heat diffusion equation. Discretization on one-dimensional grid. Linearization and formation of linear system of equations. Implementation of boundary conditions. Assignment included.
- Lesson 3 – Transient Diffusion: Solution of the transient one-dimensional heat diffusion equation. Introducing different time integration schemes. Stability criteria. Linearization of transient term. Assignment included.
- Lesson 4 – Convection of a Scalar: Solution for convection of a scalar in one-dimension with known velocity field. Introducing different advection schemes. Stability criteria. Linearization of advection term. False diffusion. Assignment included.
- Lesson 5 – Solution of Mass and Momentum Equations: Solution of coupled mass and momentum equations using segregated and fully-coupled methods. The issue of pressure velocity coupling. Staggered and colocated grid methods. Rhie-Chow interpolation. Linearization of mass and momentum equations. Assignment included.
- Lesson 6 – Two and Three Dimensional Grids: Structured and unstructured grids. Two dimensional structured grid discretization. False diffusion in two dimensions. Non-orthogonal grids. Gradient reconstruction. No assignment included.

Instructional Design

To achieve the objective of increasing students' competence with coding, all lessons have code and coding exercises embedded within them. At the end of most lessons, there is a summative assignment. Students are required to use Git to version control their assignment submission, which takes the form of a Jupyter notebook. All assignments are submitted by sharing the repository with the course instructor for evaluation. The assignments are meant to be challenging and require students to write their own code that extends the code presented in class. Normally students are given around two weeks to work on each assignment. Prior to each class, students are encouraged to read through the lesson for that week and attempt any exercises that are given. In class, students are encouraged to ask questions and engage in an open discussion about the lesson. They are also given additional time to complete exercises and troubleshoot issues they may be having with their assignment codes.

Teaching Experience

This course was taught as a graduate-level course at Western University in the Fall semester of 2018. Classes were held once a week for 12 weeks, with the duration of each class being 2 hours. No students reported any significant prior experience coding with Python or the use of Jupyter notebooks. The majority of students had experience using Matlab, so were not entirely new to programming. During the first class, the instructor helped students get Python and Git installed on their systems, demonstrated basic usage of Git in order to obtain the course notes (i.e. the Jupyter notebooks), and demonstrated basic usage of Jupyter notebooks. In order to teach the students to use Python, a link to the Software Carpentry lessons on "Programming with Python" (Bostroem, Bekolay, & Staneva, 2016) was provided and the very basics were explained with the time remaining in the first class. It was requested that the students work through the lessons in the week between the first and second classes. During the second class, students were encouraged to ask questions about the Software Carpentry lessons that they had worked through, before moving on to the main content of the course. While students clearly could not

master Python in such a short time, they were able to learn enough to follow the content and further develop their skills through working with the code embedded in the lessons. Anecdotal observations indicated that the use of Python, an unfamiliar programming language, was not a significant barrier to student learning.

The official enrollment in the course was 7 students, although several additional students attended the course on a non-for-credit basis. An anonymous online feedback survey, administered by the University, was given to students to complete before the last day of class in the semester. A total of 6 of the 7 students officially enrolled in the course completed the survey. Most of the questions are related to feedback about the instructor, however, one question relates to the course itself. Students were asked "Overall, this course was a good learning experience" and were asked to rank on a 7-point scale where: 7=Strongly Agree, 6=Agree, 5=Somewhat Agree, 4=Neither Agree Nor Disagree, 3=Somewhat Disagree, 2=Disagree, 1=Strongly Disagree. In response to this question, all students responded with a score of 7, indicating that they strongly agree that the course was a good learning experience. Despite the well-known biases of such standardized feedback tools, this result does provide a good indication that the course is at minimum providing an experience where students believe they are learning effectively.

References

Barba, L. A., & Forsyth, G. F. (2018). CFD Python: The 12 steps to Navier–Stokes equations. *Journal of Open Source Education*, *1*(9), 21. doi:10.21105/jose.00021

Bostroem, A., Bekolay, T., & Staneva, V. (2016). *Software carpentry: Programming with python, version 2016.06.* doi:10.5281/zenodo.57492

Ferziger, J. H., & Perić, M. (2002). *Computational methods for fluid dynamics* (3rd ed.). Berlin: Springer.

Patankar, S. V. (1980). *Numerical heat transfer and fluid flow.* Boca Raton: CRC Press.

Versteeg, H. K., & Malalasekera, W. (2007). *An introduction to computational fluid dynamics – the finite volume method* (2nd ed.). Essex: Pearson.