Course Project 8 - Compressible Navier-Stokes Python Coding

AERO 455 - CFD for Aerospace Applications

Instructor: Dr. Ziad Boutanios
Concordia University - MIAE

Given: 28th March 2022
Due: 13th April 2022
No extensions!

1 Introduction

This project is about coding a basic 2D finite volume method (FVM) compressible Navier-Stokes solver using the Python language. Project members are to base their work on the course notes. Two additional GitHub repositories of Navier-Stokes Python solvers are provided to that effect as well [1],[2]. The code consists of three general parts:

1. The preprocessor, which reads/creates the mesh, and reads the simulations parameters.
2. The Navier-Stokes solver per se.
3. The postprocessor, which outputs the results in the desired format, be it solution files or graphical plots and figures.

2 The Preprocessor

Most of the work will focus on reading and creating meshes. Many such possibilities exist, with many that can be consulted at the MeshPro GitHub page [3]. For mesh reading and conversion one can also refer to the meshio Python package [4]. The coders are of course free to implement their own modules from scratch as well. The meshing module can be set up within the code, or separately from it as a standalone module that can be used for other purposes as well.

Simulation parameters are best read from external files instead of setting them within a code module. This separates the preprocessor part from the code, reducing the risk of accidental tampering with the code. The preprocessor should also allow for reading previously saved solution files for restarts.

3 The Navier-Stokes Solver

The two solvers provided as guidance can be modified to write the required solver. One needs to make sure the following features are implemented:

- Transient, implicit, and conservative formulation.
- Godunov 1st order flux upwinding, as well as a stabilized higher order flux method (i.e. 2nd order MUSCL).
- Access to a direct and iterative solvers library, like PETSC [5] and its Python module [6].

4 The Postprocessor

The postprocessor is the part of the code that writes solution files in the desired format, and produces graphical output on the fly as well. A suitable solution format must be chosen and VTK [7] is recommended. This will allow visualizing the solution using Kitware Paraview [8]. One can also plot solution fields using the Python module matplotlib [9].

5 Validation

Validation should be done by simulating a published compressible flow and matching profiles such as velocity, pressure, density, etc. The AGARD database [10] that is used for some of the other projects can be consulted to this effect.

References

[1] Barba Group. *CFDPython*. URL: https://github.com/barbagroup/CFDPython.

[2] P. Mocz. *finitevolume-python*. URL: https://github.com/pmocz/finitevolume-python.

[3] MeshPro. *Mesh tools for professionals*. URL: https://github.com/meshpro.

[4] meshio. *I/O for mesh files*. URL: https://github.com/nschloe/meshio.

[5] PETSc. *Portable, Extensible Toolkit for Scientific Computation*. URL: https://gitlab.com/petsc/petsc.

[6] PETSc. *Python bindings for PETSc*. URL: https://pypi.org/project/petsc4py/.

[7] Kitware. *VTK, The Visualization Toolkit*. URL: https://pypi.org/project/vtk/.

[8] Kitware. *ParaView*. URL: https://www.paraview.org/.

[9] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[10] AGARD Fluid Dynamics Panel WG04. *Experimental Data Base for Computer Program Assessment*. Tech. rep. 138. AGARD, 1979.