10th ANKARA INTERNATIONAL AEROSPACE CONFERENCE AIAC-2019-035
18-20 September 2019 - METU, Ankara TURKEY

AN EDUCATIONAL IN-HOUSE PYTHON COMPUTATIONAL FLUID DYNAMICS (CFD) CODE

Fatima Abusbei* and Abdulhaq Emhemmed†
University of Tripoli
Tripoli, Libya

Ashraf A. Omar‡
International University of Rabat
Rabat, Morocco

ABSTRACT

In this work, in-house computational fluid dynamics (CFD) code was developed to solve five popular fluid dynamics and heat transfer problems. The code was written using python programming language and in order to improve the written code object-oriented programming protocol (OOP) was used. The solved problems are Couette flow, steady and unsteady heat conduction, cavity flow and backward facing step. The finite difference method (FDM) was used to reduce the partial differential equations of the governing equation to a set of algebraic equations (Discretization). Iterative procedure for the explicit and implicit methods was used to solve the discrete system. Modern Multi-steps technique such as alternate directional implicit (ADI) method was also used in some cases. The obtained numerical results of the solved problems were compared to published results from the literature. Good agreement was achieved in most cases.

Keywords: In house CFD code, python programming language, object-oriented programming protocol (OOP).

INTRODUCTION

To analyze engineering systems, two methods are applied. First method, a physical model of the system has to be made and put in an experimental. The second method is to drive a mathematical model that represents the system and then solve it. Experimentation has been an extremely important method to advance engineering. However, it has limitations. For example, it is expensive. Systems that have many variables need to be analyzed with all these variables altered, this is difficult to perform in a laboratory. Another example is that experimentation is sometimes impossible or it requires tools that are not available. For example, if the flow of blood in the human body is to be studied, it is impossible to perform such study especially in certain areas of the body.

Solving the mathematical model that represents the system is also a difficult task, however, it has the advantage of not requiring any physical model to be made. Its disadvantage is that for most engineering systems, the mathematical model is a set of partial differential equations (PDEs) that are difficult or impossible to solve analytically. Therefore, numerical techniques need to be used to obtain a solution.

*Researcher, Email: f.abusbei@aerodept.edu.ly
†Teaching/Research Assistant , Email: abdulhaq.emhemmed@aerodept.edu.ly
‡Prof. in a school of Aerospace and Automotive Engineering , Email: ashraf_omar@uir.ac.ma

When using computers to solve PDEs that model fluid flow numerically, this topic is called Computational Fluid Dynamics (CFD). As the power of computing increased, CFD has become an independent field taught on its own. This work summarizes a project carried out to produce a simple in-house CFD python code that will allow students to learn from and adapt to their needs very easily. The purpose of the code is not to simulate real engineering cases such as those used by commercial products or large research codes. It is to demonstrate CFD on basic applications with the ability to extend the code without large editing. The choice of tools made to create the code to satisfy these requirements.

METHODOLOGY

Numerical Method

The general form of a partial differential equation is given by:

$$A\frac{\partial^2\phi}{\partial x^2} + B\frac{\partial^2\phi}{\partial x\partial y} + C\frac{\partial^2\phi}{\partial y^2} + D\frac{\partial\phi}{\partial x} + E\frac{\partial\phi}{\partial y} + F\phi + G = 0 \tag{1}$$

The equation above was taken from ref[VERSTEEG H. K. , and MALALASEKERA W., 1995]
Where A, B, C, D, E, F, G are constants. The values of these constants determine the type of the partial differential equation. The type depends on $B^2 - 4AC$:

1. Parabolic PDE if $B^2 4AC = 0$.
2. Elliptic PDE if $B^2 4AC < 0$.
3. Hyperbolic PDE if $B^2 4AC > 0$.

Each type has its own behavior and solution hence different numerical methods need to be applied.
Numerical methods have been developed to solve any of the PDEs. Three famous methods are the finite difference method (FDM), the finite volume method (FVM) and finite element method (FEM).
The FDM is the most basic one and understanding it will help to understand others. It can be explained simply by expressing each partial derivative by an algebraic fraction and the resulting equation is applied to a set of points (called nodes) and this set of points make what is called the mesh.
Transforming the PDE into an algebraic equation is called discretization. Many numerical schemes are developed for this process. Each scheme has its advantages and disadvantages. For example, some schemes are slower to converge than others, some schemes are unstable for a certain equation or only stable under certain conditions.
The basis of discretization is using Taylor series:

$$f(x+\Delta x) = f(x) + \Delta x\frac{\partial f}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 f}{\partial x^3} + ...\frac{\Delta x^n}{n!}\frac{\partial^n f}{\partial x^n} \tag{2}$$

Different type of numerical schemes are used to solve problems and this depends on flow models.
Whereas:
Couette Flow was solved by this schemes: FTCS, DuFort-Frankel, Laasonen and Cronk-Nicolson scheme. Unsteady heat conduction problem was solved by Alternating Direction Implicit (ADI) scheme. Steady heat conduction problem was solved by Point Successive Over-Relaxation (PSOR) and Line Successive Over-Relaxation (LSOR) scheme. Cavity Flow problem was solved by: FTCS scheme for vorticity and PSOR scheme for stream function.

Code Development

Python chosen as the programming language, because it is tops all the recent programming language charts [Diakopoulos N. , and Cass S., 2017]. Moreover, Its features and packages has made scientists

and engineers to adopt it. Finally, it is first choice in computational and data science [Python Core Team, 2015].

Object Oriented Programming protocol (OOP) has been used to make structured code easy to use, understand, read and develop. Object oriented programming is a software engineering paradigm used to help software engineers to design software by modeling its components like real life objects. This helps in organizing the codes and to solve problems like code repetition which exists in procedural programming.

The code has been developed by using the agile philosophy. In the agile method, the requirements are not completely gathered, the code is improved by steps. For example, a very simple sample is developed initially, it is tested to ensure it is working correctly. Then the code is *refactored* to add more features. This is repeated until the code satisfies the requirements.

The features of OOP were used extensively, and it has been used to avoid repeat functions. For this purpose, the code has been divided to many classes, the main class that all solvers inherent is solver class. This class has two other classes: a class for steady state solvers inherits from the base solvers class and a class is available for unsteady solvers. All solvers will inherit from these two base classes. For example, a solver for Laplace equation is a steady solver, it will inherit from steady solver and implement the appropriate methods (e.g. solve steps and calculate error).

Moreover, using the code is very easy, no need to even open the code at sometimes, just the required class must be imported and all the information such as mesh, time step and dimensions ..etc must be accessed. This is the main features of the code. What is more, the cod is easy to develop, as a result, just by writing the equations and inheritance the new scheme will be added easily.

Couette flow problem has been solved by four schemes. Couette-Flow-Solver is the main class for Couette flow case. This class inherits from Steady Solve class. and each scheme has its own class, all classes inherit from Couette-Flow-Solver, consequently, add new scheme is very simple. All other cases have been written by the same technique.

The testing code is an important aspect of software engineering, the importance is greater when the code changes frequently. In the development of this code, testing was used extensively to ensure that no mistakes were made as the code changes. All schemes have code to test all the results. Version control software (VCS) is an important tool in software engineering and has been used in this work. From this tool the returning to any code edition easy and simple. Moreover, it can be used and edited.

Code samples

The following sample is for solving Couette flow using FTCS discretization:

```
class CouetteFlowFTCSSolver(CouetteFlowSolver):

def discretisation(self):
u_n = self.solution['temp_solution']['u_n']
return u_n[1:-1] + self.CFL * (
u_n[2:] - 2 * u_n[1:-1] + u_n[0:-2])

def solve_time_steps(self):
for n in range(1, self.NM + 1):
self.set_boundary_conditions()
self.time_step = n
self.solution['temp_solution']['u_n']=
self.solution['u'].copy()
self.solution['u'][1:-1] = self.discretisation()
```

A sample test for the above code is given below:

```
class TestCouetteFlowFTCS:
```

```
config = {
'Scheme': 'FTCS',
'Time Step': 0.002,
'Time': 1.08,
'mu': 0.000217,
'Mesh': mesh,
'Boundary Conditions': BC
}
solver = CouetteFlowFTCSSolver(config=config)
solver.solve()

def test_couette_flow_boundary_conditions(self):
# bottom plate
assert np.all(np.isclose(self.solver.solution
['u'][0], 40))
# top plate
assert np.all(np.isclose(self.solver.
solution['u'][-1], 0))
#def test_coutte_flow_initial_conditions():
# assert np.all(np.isclose(solver.solution
['u'][1:-1], 0))

def test_couette_flow_results(self):
a = self.solver.solution['u']
assert np.all(np.isclose([a[10]],[25.739],
rtol=1.e-4))
```

RESULTS AND DISCUSSION

In this section, some obtained results are showed and discussed.

Couette Flow

Figure 1 shows the variation of the velocity was function of time 1, with moving bottom plate, using the FTCS scheme.

This scheme is conditionally stable, with the condition that $\mathrm{CFL} \leq 0.5$. When the time step is equal to $\Delta t = 0.002$, the CFL is $0.434 \leq 0.5$ therefore this time step will give a stable solution as shown in figure 1.

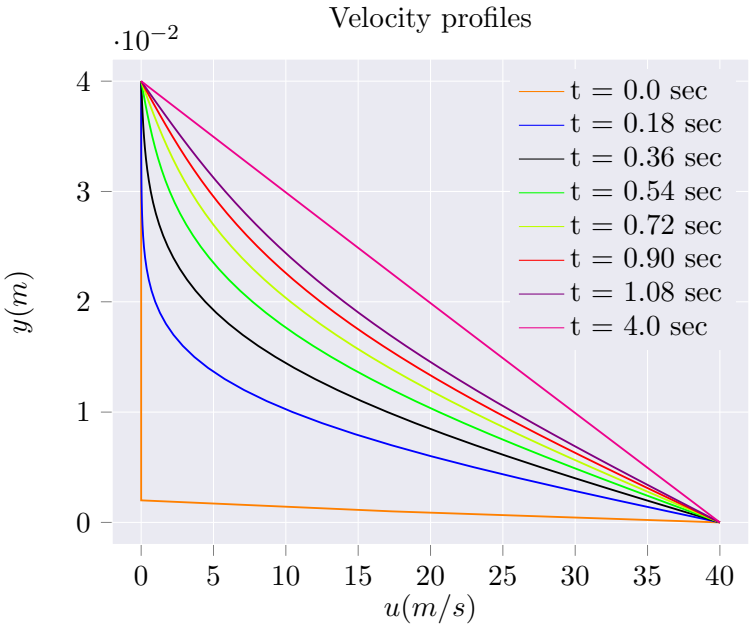

Figure 1: Velocity profiles obtained by FTCS explicit method

The figure 1 shows that, the velocity increases at a linear rate as rise to the top, because the bottom plate velocity is zero, ie, when the $y = 0$, the air particles gains the velocity of the toucher plate. This is called no slip condition. And because the effect of the viscosity each layer of fluid working to reduce the velocity of fluid above it. This happens in the upper plate also, ie, when the $y = 40$, the velocity is the maximum, because it gained the velocity of the plate 40, ie, the velocity increases from zero on the bottom plate to 40 on top plate, and the relation is linear, because no pressure gradient. Vice versa in the figure 1, because in this case the bottom plate moves whereas the top plate is fixed, ie, the velocity decreases as rise to the top and at a linear rate also.

Unsteady heat conduction

Figure 2 shows contours of temperature for a rectangular plate, by using ADI scheme.

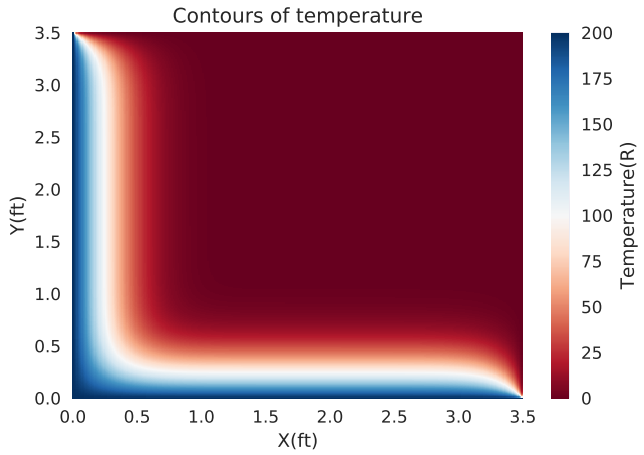

Figure 2: Contours of temperature distribution for rectangular plate

From figure2, it shows that the temperature gradually changes from the sides which are highest temperature to the sides which have a lower temperature. So that the temperature distribution at the points which are touching the sides which have the highest temperature are higher than the other points.

This distribution is compared with results obtained by Ref. [Hoffmann K. A. , and Chiang S. T., 2000]. The comparison is shown in figure 3.

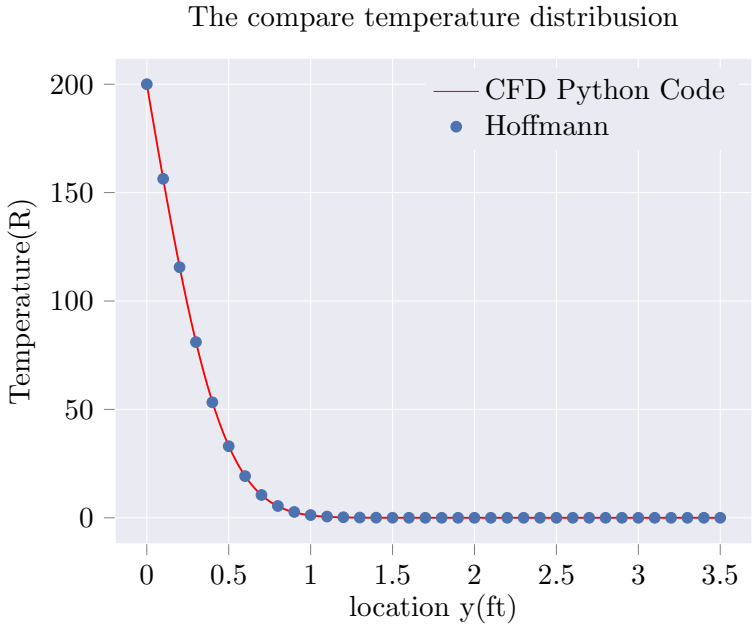

Figure 3: The compare temperature distribution of CFD code with Hoffmann

The result have been taken at location $(x = 2ft)$ for all (y) location. The identical between the results is clear.

Steady heat conduction

Figure 4 shows contours of temperature for a rectangular plate, by using PSOR scheme.

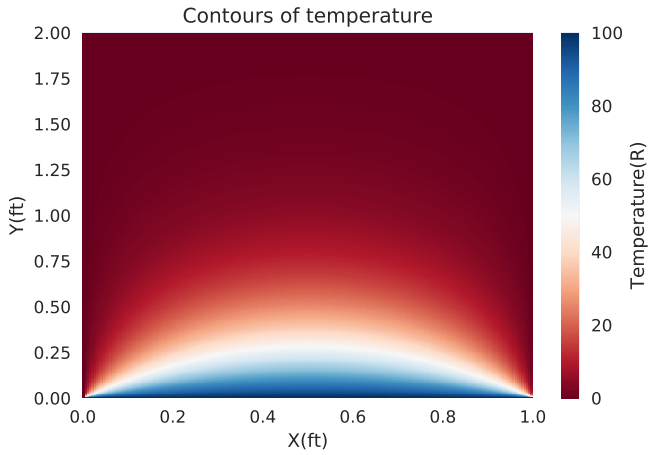

Figure 4: Contours of temperature distribution for rectangular plate

The temperature distribution has characteristics similar to the unsteady case. This distribution is compared with results obtained by Ref. [Hoffmann K. A. , and Chiang S. T., 2000]. The comparison is shown in figure 5.

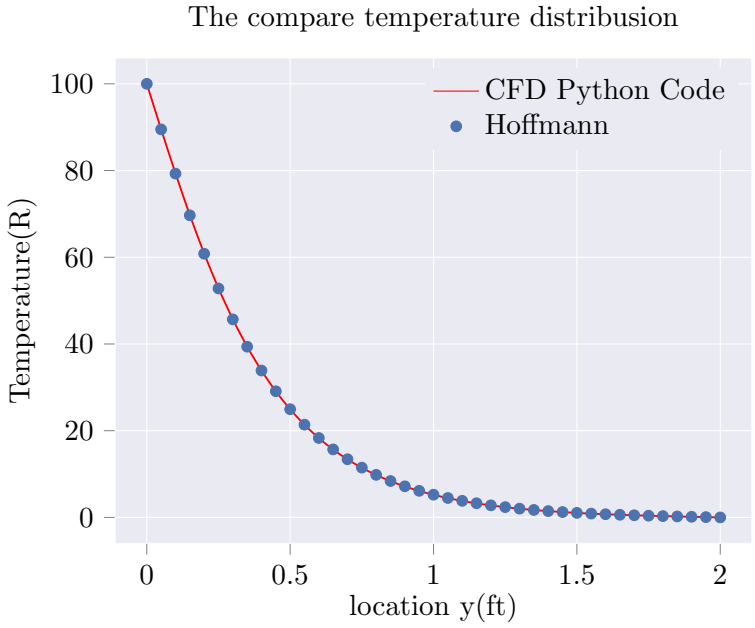

Figure 5: The compare temperature distribution of CFD code with Hoffmann

The result have been taken at location $(x = 0.4ft)$ for all (y) location. The identical between the results is clear.

Cavity Flow

Figures 6 show the stream line for cavity flow without inlet and the outlet, at Reynolds Number ($Re = 100$), by using FTCS scheme.

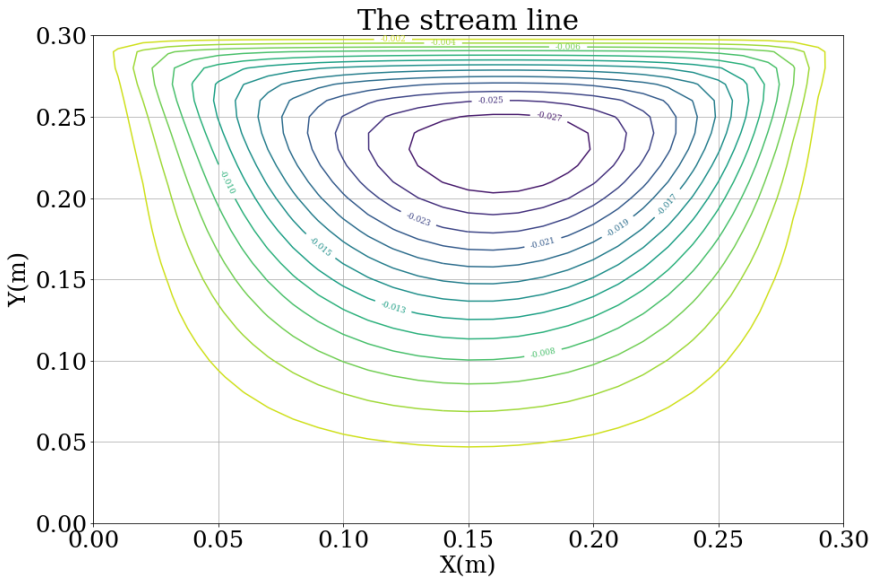

Figure 6: The stream line plot for the rectangular cavity without inlet & outlet

A comparison between the velocity components u and v with results obtained by Ref. [Ghia U. , Ghia K. N., and Shin Wi C. T., 1982] is shown in figures 7. The results are taken through the geometric center of the cavity.

Figures 7 show the distribution of streamline, respectively. The figures show clearly the smooth distribution of the flow. The results were compared Ghia [Ghia U. , Ghia K. N., and Shin Wi C. T., 1982] in figures 7.A good agreement was observed, which prove the accuracy of the developed CFD in-House code in predicting the cavity flow problem.

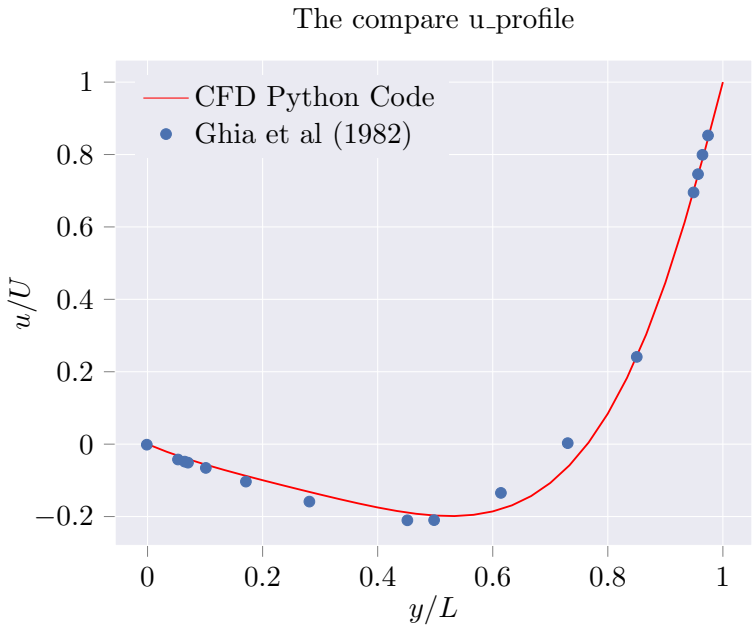

Figure 7: Comparison of profile for velocity component u

CONCLUSIONS

In conclusion, an educational in-house CFD code has been developed. It is educational because it is easy to understand expand and a useful teaching tool. The paper has used modern software engineering principles with a modern programming language. Its structure allows for the easy extending and simplicity. Different mathematical models for fluid flow and heat transfer have been implemented and verified. This includes Couette flow, Laplace equation and vorticity-stream function formulation of the Navier-Stokes equations. Furthermore, these models were solved using different techniques and numerical schemes.

References

Diakopoulos N. , and Cass S. , (2017) , *IEEE Spectrum.*, URL: `https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017` (visited on 02/27/2019), 2017

Ghia U. , Ghia K. N., and Shin Wi C. T. , (1982) *High-Re Solutions for Incompressible Flow Using the Navier- Stokes Equations and a Multigrid Method**, JOURNAL OF COMPUTATIONAL PHYSICS 48 (1982), pp: 387411, 1982

Hoffmann K. A. , and Chiang S. T. , (2000) *Computational Fluid Dynamics*, Volume I. 4th Ed. , Engineering Education System, 2000

Python Core Team, ed. Python: A dynamics, open source programming language. *Python Software Foundation. 2015*, URL: `https://www.python.org/` (visited on 01/27/2019), 2015

VERSTEEG H. K. , and MALALASEKERA W. , (1995) *An Introduction to Computational Fluid Dynamics The Finite Volume Method*, 1995