AE5011 Computational Fluid Dynamics I
Finite Volume Method - Inviscid Burgers
Semester I - 2022/2023

Submitted as the fulfillment of AE5011 Computational Fluid Dynamics I final task

By:

Michael Agung N 23622009
Nayottama Putra S 23622016

Lecturer:

Dr. Ing. Mochammad Agoes Moelyadi, S.T, M.Sc.

GRADUATE PROGRAM IN AEROSPACE ENGINEERING
FACULTY OF MECHANICAL AND AEROSPACE ENGINEERING
INSTITUT TEKNOLOGI BANDUNG

2022

Michael Agung N - 23622009
Nayottama Putra S - 23622016

Contents

Michael Agung N - 23622009
Nayottama Putra S - 23622016

1 Introduction

1.1 Objective

The objective of this project is as follows.

1. To create 2-dimensional grid using Trans-Finite Interpolation technique
2. To derive the finite volume discrete equation of the Inviscid Burgers equation
3. To create finite volume-based numerical program
4. To obtain numerical solution of two-dimensional Inviscid Burgers equation using finite volume method

1.2 Problem Formulation

The inviscid Burgers equation is as follows.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

The domain of the simulation is a square with $0 \leq x, y \leq 1$. The boundary condition is Dirichlet at the left, right, and bottom edge, specifically as follows.

$$u(0, y) = 1.5$$
$$u(1, y) = -0.5$$
$$u(x, 0) = 1.5 - 2x$$

Michael Agung N - 23622009
Nayottama Putra S - 23622016

2 Fundamental Theory

2.1 Trans-Finite Interpolation

In computational fluid dynamics, one of the most critical components is mesh or grid generation. The grid is detrimental to the calculation because it relates the computational domain with the physical domain. Grids provide mathematical support for the numerical solution of governing field equations in a continuum domain. One of the methods for grid generation is the Trans-Finite Interpolation method, categorized as an algebraic method. In doing this task, the TFI method we would use is the linear TFI method. The TFI that would be discussed is also a 2-dimensional TFI.

Suppose we have a physical domain $\mathbf{X}$ and a computational domain of (ξ, η), then the formulation to relate the computational domain to the physical domain is as follows.

$$\mathbf{U}(\xi_I, \eta_J) = (1 - \xi_I)\mathbf{X}(0, \eta_J) + \xi_I \mathbf{X}(1, \eta_J) \tag{2.1}$$

$$\mathbf{V}(\xi_I, \eta_J) = (1 - \eta_J)\mathbf{X}(\xi_I, 0) + \eta_J \mathbf{X}(\xi_I, 1) \tag{2.2}$$

$$\mathbf{UV}(\xi_I, \eta_J) = (1 - \xi_I)(1 - \eta_J)\mathbf{X}(0, 0) + \xi_I(1 - \eta_J)\mathbf{X}(1, 0) + (1 - \xi_I)\eta_J \mathbf{X}(0, 1) + \xi_I \eta_J \mathbf{X}(1, 1) \tag{2.3}$$

$$\mathbf{X}(\xi_I, \eta_J) = \mathbf{U} + \mathbf{V} - \mathbf{UV} \tag{2.4}$$

The computational domain would depend on the number of points generated in the grids. The general equation of the transformation of the computational grid is as follows.

$$0 \leq \xi = \frac{I - 1}{\hat{I} - 1} \leq 1 \qquad\qquad 0 \leq \eta = \frac{J - 1}{\hat{J} - 1} \leq 1$$

2.2 Finite Volume Method

The finite volume method (FVM) is a method to evaluate partial differential equations. In the finite volume method, volume integrals in a partial differential equation are converted to surface integrals using the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. As the flux entering a given volume is identical to that leaving the adjacent volume, these methods are conservative. Unlike the finite difference method, the finite volume method is formulated to allow for unstructured meshes.

Michael Agung N - 23622009
Nayottama Putra S - 23622016

Let we have a 2-dimensional equation as follows.

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0 \tag{2.5}$$

Equation (2.5) can be rewritten by introducing ∇ as follows.

$$\frac{\partial U}{\partial t} + \nabla \cdot H = 0 \tag{2.6}$$

where $H = E\hat{i} + F\hat{j}$.

For the finite volume method, equation (2.6) needs to be integrated over the whole area. Then, using the Gauss theorem, equation (2.6) can be written as follows.

$$\begin{aligned} \iint_A \frac{\partial U}{\partial t}\, dA + \iint_A \nabla \cdot H\, dA &= 0 \\ \iint_A \frac{\partial U}{\partial t}\, dA + \oint_S H \cdot \hat{n}\, dS &= 0 \\ \frac{\partial}{\partial t} \iint_A U\, dA + \oint_S (E dy - F dx) &= 0 \end{aligned} \tag{2.7}$$

Equation (2.7) is rearranged as follows.

$$\frac{\partial}{\partial t} \iint_A U\, dA = \oint_S (F dx - E dy) \tag{2.8}$$

Let a cell has n edges $(k_1, k_2, \cdots, k_n)$ and n vertices $(m_1, m_2, \cdots, m_n)$, where edge k_i is formed by point m_i and m_{i+1}, with the assumption of $m_{n+1} = m_1$. Then, equation (2.8) can be written for each cell as follows.

$$\frac{\partial}{\partial t} \left(\langle U_i \rangle A_i \right) = \Sigma_{i=1}^{n} \left[F_{k_i} \left(x_{m_{i+1}} - x_{m_i} \right) \right] - \Sigma_{i=1}^{n} \left[E_{k_i} \left(y_{m_{i+1}} - y_{m_i} \right) \right] \tag{2.9}$$

where $\langle U_i \rangle$ is the average value of property U over the area A_i and E_{k_i}, F_{k_i} are the value of conservative properties E and F on edge k_i.

For complex cell shape, the area A_i can be approximated as follows.

$$A_i \approx \frac{1}{2} \Sigma_{i=1}^{n} \left(x_{m_i} + x_{m_{i+1}} \right) \left(y_{m_{i+1}} - y_{m_i} \right) \tag{2.10}$$

Discretizing the time in equation (2.9) using Euler method yields

$$\left\langle U_i^{t+1} \right\rangle A_i^{t+1} = \left\langle U_i^t \right\rangle A_i^t + \Delta t \Sigma_{i=1}^n \left[F_{k_i}^t \left(x_{m_{i+1}} - x_{m_i} \right) \right] - \Delta t \Sigma_{i=1}^n \left[E_{k_i}^t \left(y_{m_{i+1}} - y_{m_i} \right) \right] \tag{2.11}$$

Michael Agung N - 23622009
Nayottama Putra S - 23622016

3 Methodology

3.1 Grid Generation

The grid is created using the Trans-Finite Interpolation (TFI) method. There are two grids that would be made, which are the grid and the center point grid. In practice, the TFI would be done for the matrix that describes the X position and the matrix that describes the Y position. The code implementation is fairly based on Equation 2.1, 2.2, 2.3, and 2.4. The implementation of the grid would be by creating an object called Grid with an input of number of points along x and y and the extreme positions of x and y.

Th class object called Grid would also provides the center points of each cell by using TFI too. The way the code generate this is by creating points inside with the number of points minus one than the normal grid. Using TFI, it would start generating from the middle point of first cell to the middle point of the last cell. The class object Grid would have a property of distance between points in x and y direction for the purpose of generating the middle point of cells.

3.2 Equation Discretization

The equation to be solved is as follows.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{3.1}$$

Equation (3.1) can be written in conservative form as follows.

$$\frac{\partial u}{\partial t} + \frac{\partial \frac{1}{2}u^2}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{3.2}$$

Let $E = \frac{1}{2}u^2$ and $F = u$. Equation (3.2) can be written as follows.

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} &= 0 \\ \frac{\partial u}{\partial t} + \nabla \cdot H &= 0 \end{aligned} \tag{3.3}$$

where $H = E\hat{i} + F\hat{j}$.

Michael Agung N - 23622009
Nayottama Putra S - 23622016

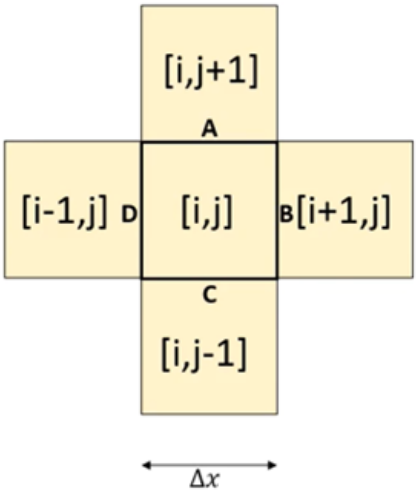


Figure 3.1: Cell Illustration

For finite volume method, equation (3.3) needs to be integrated over the whole area. Then, using Gauss theorem, the inviscid Burgers equation can be written as follows.

$$\begin{aligned} \iint_A \frac{\partial u}{\partial t}\, dA + \iint_A \nabla \cdot H\, dA &= 0 \\ \iint_A \frac{\partial u}{\partial t}\, dA + \oint_A H \cdot \hat{n}\, dA &= 0 \\ \frac{\partial}{\partial t} \iint_A u\, dA + \oint_A (E dy - F dx) &= 0 \end{aligned} \tag{3.4}$$

Equation (3.4) is rearranged as follows.

$$\frac{\partial}{\partial t} \iint_A u\, dA = \oint_A (F dx - E dy) \tag{3.5}$$

In this project, the grid is structured. Every grid is a square with 4 edges and 4 vertices. Each cell can be illustrated as shown in figure 3.1.

The discretization of equation (3.5) for each cell is as follows.

$$\begin{aligned} \frac{\partial}{\partial t}\left(\left\langle u_{i,j}\right\rangle A_{i,j}\right) &= \Sigma_{i=1}^{4}\left[F_{k_i}\left(x_{m_{i+1}} - x_{m_i}\right)\right] - \Sigma_{i=1}^{n}\left[E_{k_i}\left(y_{m_{i+1}} - y_{m_i}\right)\right] \\ &= \left[F_A \Delta x_A + F_B \Delta x_B + F_C \Delta x_C + F_D \Delta x_D\right] - \\ &\quad \left[E_A \Delta y_A + E_B \Delta y_B + E_C \Delta y_C + E_D \Delta y_D\right] \end{aligned} \tag{3.6}$$

From figure 3.1, it is clear that $\Delta x_B = \Delta x_D = \Delta y_A = \Delta y_C = 0$. On the other hand, $\Delta x_A = -\Delta x$, $\Delta y_B = \Delta y$, $\Delta x_C = \Delta x$, and $\Delta y_D = \Delta y$. Also, $A_{i,j} = \Delta x \Delta y$. Thus, equation (3.6) can be simplified as follows.

$$\begin{aligned} \Delta x \Delta y \frac{\partial \left\langle u_{i,j} \right\rangle}{\partial t} &= F_C \Delta x - F_A \Delta x + E_D \Delta y - E_B \Delta y \\ \frac{\partial \left\langle u_{i,j} \right\rangle}{\partial t} &= \frac{1}{\Delta x} \left(E_D - E_B \right) + \frac{1}{\Delta y} \left(F_C - F_A \right) \end{aligned} \tag{3.7}$$

In this project, Euler method is used for time integration. The final form of the discretization of equation for each cell is as follows.

$$\left\langle u_{i,j}^{n+1} \right\rangle = \left\langle u_{i,j}^{n} \right\rangle + \frac{\Delta t}{\Delta x} \left(E_D^n - E_B^n \right) + \frac{\Delta t}{\Delta y} \left(F_C^n - F_A^n \right) \tag{3.8}$$

The value of properties at surface $A, B, C,$ and D is taken as the average value of those corresponding cells.

$$\begin{aligned} E_B &= \frac{1}{2} \left(E_{i,j} + E_{i+1,j} \right) \\ E_D &= \frac{1}{2} \left(E_{i-1,j} + E_{i,j} \right) \\ F_A &= \frac{1}{2} \left(F_{i,j} + E_{i,j+1} \right) \\ F_C &= \frac{1}{2} \left(F_{i-1,j} + E_{i,j} \right) \end{aligned}$$

Substituting these values into equation (3.8) yields

$$\left\langle u_{i,j}^{n+1} \right\rangle = \left\langle u_{i,j}^{n} \right\rangle + \frac{\Delta t}{2\Delta x} \left(E_{i-1,j}^n - E_{i+1,j}^n \right) + \frac{\Delta t}{2\Delta y} \left(F_{i,j-1}^n - F_{i,j+1}^n \right) \tag{3.9}$$

For stability, the value of $u_{i,j}^n$ is taken as the average values of the other neighboring cells.

$$\left\langle u_{i,j}^{n+1} \right\rangle = \frac{\left\langle u_{i-1,j}^{n} \right\rangle + \left\langle u_{i+1,j}^{n} \right\rangle + \left\langle u_{i,j-1}^{n} \right\rangle + \left\langle u_{i,j+1}^{n} \right\rangle}{4} + \frac{\Delta t}{2\Delta x} \left(E_{i-1,j}^n - E_{i+1,j}^n \right) + \frac{\Delta t}{2\Delta y} \left(F_{i,j-1}^n - F_{i,j+1}^n \right) \tag{3.10}$$

3.3 Boundary Conditions Enforcement

To enforce the boundary conditions, a layer of ghost points is placed on the outside of the domain. For the left, right, and bottom edges, the boundary conditions are Dirichlet. Therefore, the value are specified on those ghost points. As for the tof edge, the boundary condition is assumed to be Neumann. Therefore, its values are set to be the same as those of the top layer of the domain after every timestep.

3.4 Timestep

As the grid is square, CFL stability criterion in used.

$$CFL = \left| \frac{\Delta t}{\Delta x} u_{max} \right| \leq 1 \tag{3.11}$$

Assuming that maximum u is 1.5, the maximum timestep can be calculated.

$$\Delta t \leq \frac{\Delta x}{u_{max}} = \frac{\Delta x}{1.5} \tag{3.12}$$

In this project, the timestep is taken at

$$\Delta t = \frac{\Delta x}{2.5} \tag{3.13}$$

which corresponds to a constant CFL number of 0.6.

3.5 Error Calculation

In this project, L_2 norm relative error calculation is used.

$$\frac{||x - \hat{x}||_2}{||x||_2} = \sqrt{\frac{\sum_i ||x_i - \hat{x}_i||^2}{\sum_i ||x_i||^2}} \tag{3.14}$$

3.6 Code Implementation

The solver program is developed using Python programming language. It uses cloud-based Google Colab notebook. First, the grid is generated using Trans-Finite Interpolation. There are two grids that would be made, which are the nodes grid and the center point grid. The center point grid stores the value of variables. Then, the calculation is conducted based on the discrete equation derived earlier for each cell. This calculation is used to update the property value stored in each cell at every timestep. The simulation ended when the change of the stored values at each cell is deemed insignificant, in particular less than the maximum threshold of 1E-5.

The complete solver program can be found on
https : *//github.com/Magnoobz/CFD*1*_Inviscid_Burgers_FVM*

4 Numerical Results and Analysis

4.1 Analytical Solution

The analytic solution of the inviscid Burgers equation is written in equation (4.1), and visualized as shown in figure 4.1.

$$u(x,y) = \begin{cases} 1.5 & y \leq 0.5 \text{ and } x \leq 1.5y \\ \frac{1.5-2x}{1-2y} & y \leq 0.5 \text{ and } 1.5y \leq x \leq 1-0.5y \\ -0.5 & y \leq 0.5 \text{ and } x \geq 1-0.5y \\ 1.5 & y > 0.5 \text{ and } x \leq 0.5+0.5y \\ -0.5 & y > 0.5 \text{ and } x > 0.5+0.5y \end{cases} \tag{4.1}$$

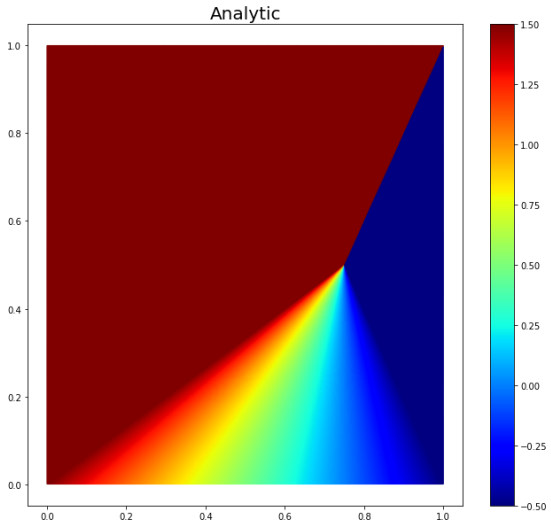
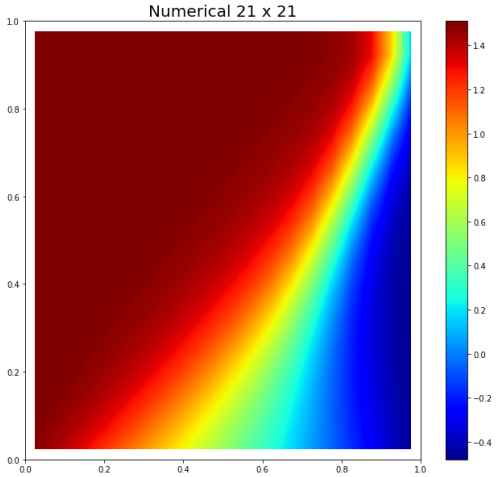


Figure 4.1: Analytic Solution

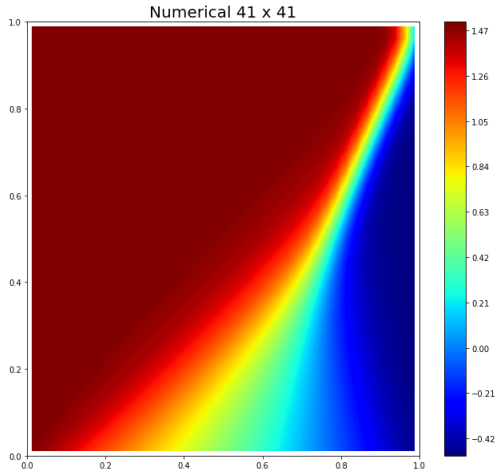
4.2 Numerical Result

In this project, several grid distributions are used, which are 21x21, 41x41, 61x61, 101x101, 151x151, and 201x201. The numerical simulation results are shown in figure 4.2. In this project, the simulation is steady. In this case, the steadiness is defined when the absolute flux in every cell is less than the maximum threshold of 1E-5. The absolute flux for each distribution are presented in figure 4.3, which show that the simulation has been steady.

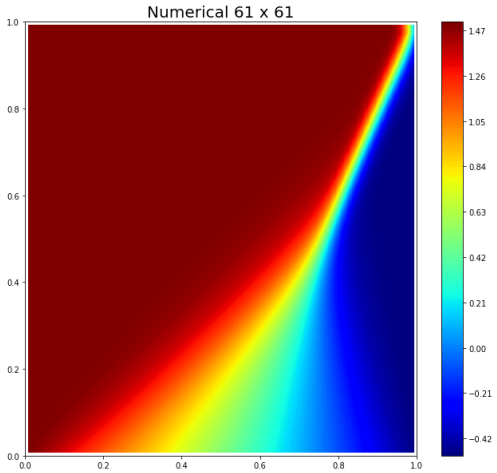
Michael Agung N - 23622009
Nayottama Putra S - 23622016



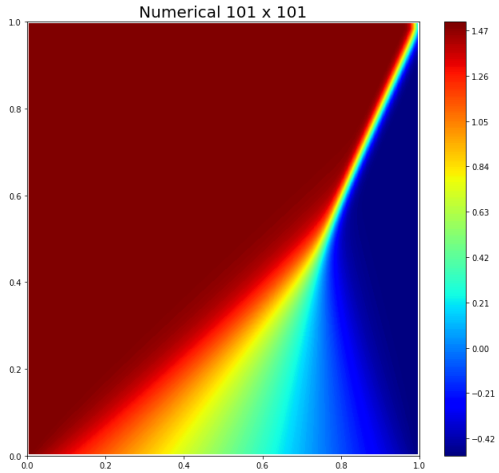
(a) 21 x 21



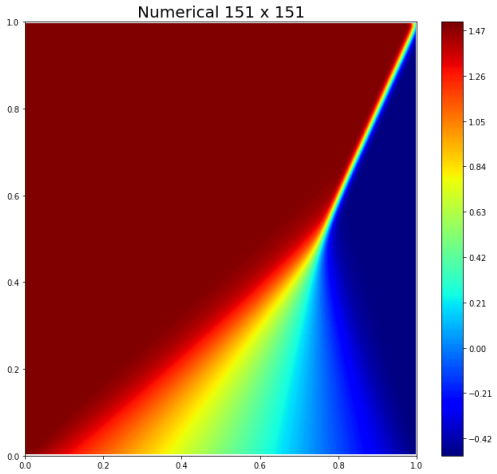
(b) 41 x 41



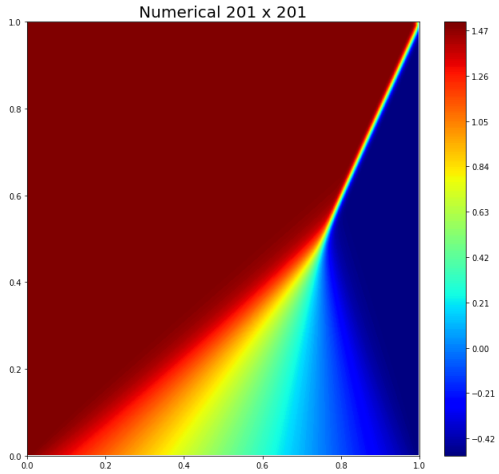
(c) 61 x 61



(d) 101 x 101

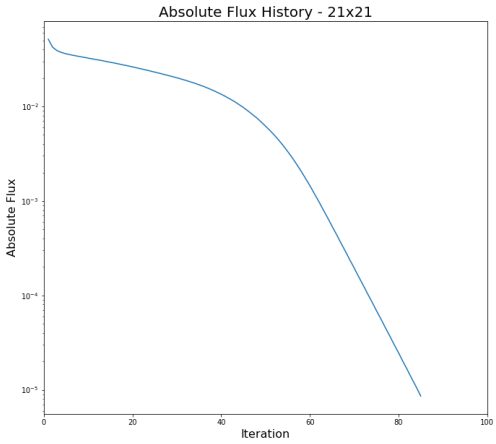


(e) 151 x 151

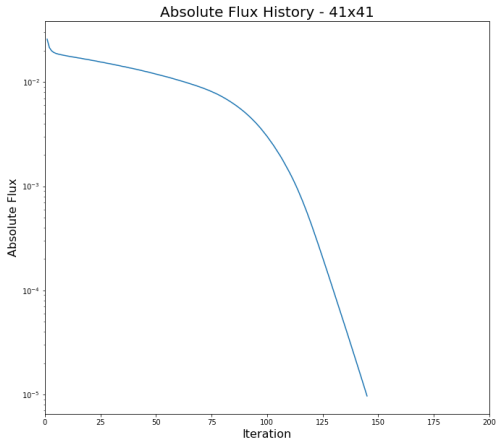


(f) 201 x 201

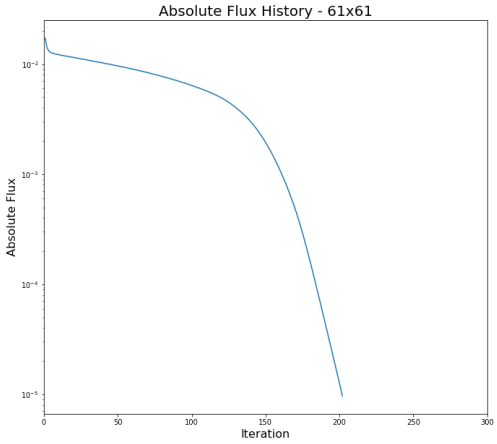
Figure 4.2: Numerical Simulation Results



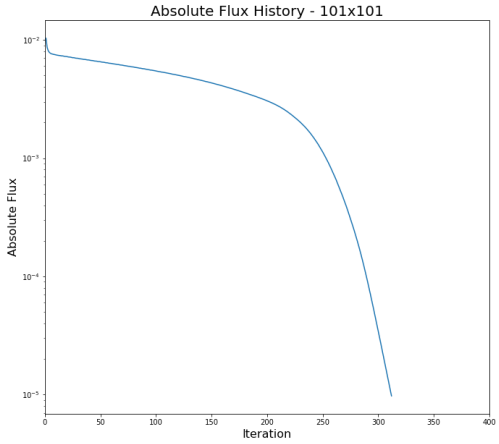
(a) 21 x 21



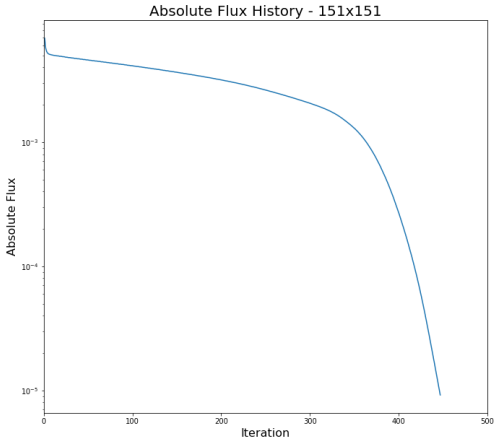
(b) 41 x 41



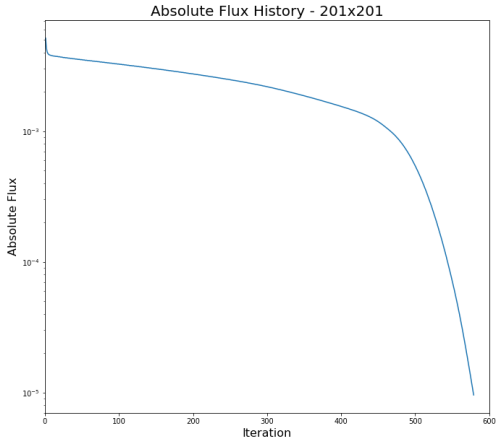
(c) 61 x 61



(d) 101 x 101



(e) 151 x 151



(f) 201 x 201

Figure 4.3: Absolute Flux History Results

The L_2 relative norm error for each distribution are presented in table 4.1 and visualized in figure 4.4. It can be concluded that the higher the resolution, the more accurate the numerical result is. For 201x201 grid, as in figure 4.2(f), the numerical result resembles that of analytical. This is in contrast with the numerical result of 21x21 grid in figure 4.2(a). As shown in figure 4.4, the L_2 relative norm error decreases as the number of grid increases. Figure 4.4 shows that between gridsize of 21x21 and 41x41, the number of cells has not yet been sufficient as it is still not convergent in terms of error, gridsize-wise.

However, higher resolution requires much higher computational cost. First, higher resolution means more cells to be solved. Other than that, higher resolution requires higher number of iteration due to the smaller timestep, as shown in figure 4.3, as the effect of the maximum CFL number.

Grid Size	Number of Cells	Number of Iteration	L_2 Relative Norm Error
21x21	400	85	0.173
41x41	1600	145	0.123
61x61	3600	202	0.099
101x101	10000	312	0.076
151x151	22500	447	0.061
201x201	40000	579	0.052

Table 4.1: L_2 Relative Norm Error

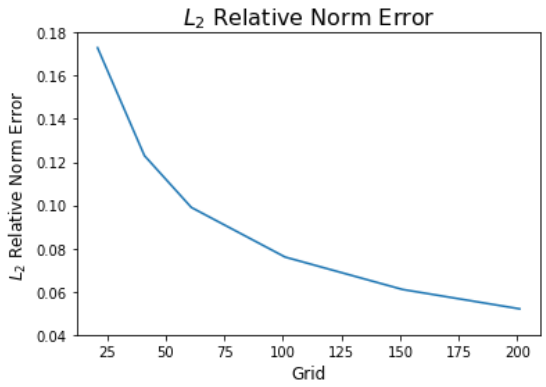


Figure 4.4: L_2 Relative Norm Error Plot

The results in figure 4.2 show that finite volume method has difficulties in modelling shockwave, which is the location when there is sudden change in properties value. Numerical simulation is not able to illustrate this sudden jump in properties value. Instead, this sudden change in values is modelled as gradual change. One of the reasons is due to the use of average value of properties E and F in the cell surface between 2 cells.

In the high resolution simulation, this gradual change does not really affect the simulation result, as the gradual layer is quite thin. This is not the case, however, in the low resolution simulation. The lower the resolution, the larger the gradual layer is, thus resulted in higher relative norm error. This phenomena is shown on figure 4.2.

5 Conclusion

A finite volume solver program for inviscid Burgers equation has been developed in this project. Before developing the program, the finite volume discrete equation of Inviscid Burgers equation has been derived. The developed program implemented Trans-Finite Interpolation technique to generate 2-dimensional grid. The solver program has been able to accurately solve Inviscid Burgers equation, especially for those with high resolution. In general, the solver still has difficulties in modelling shockwave interface, where there is sudden change of properties value. However, this problem has been able to be resolved by having higher resolution, with one of the drawbacks being its longer computation time.