Project 2 Solution: Turbine-Blade Passage

AE 523: Computational Fluid Dynamics

Erin Levesque

Fall 2022

1 Introduction

In this project we were tasked with developing a two-dimensional finite volume method solver to simulate compressible flow in a turbine passage. We make use of the Euler equations of gas dynamics to define and update the state on each cell as the solver marches forward in time. This problem is solved under the periodic boundary conditions defined on the upper and lower surface of the blade, the inflow and the outflow.

2 Problem Set-up

2.1 Physical Model

The two-dimensional turbine blade passage domain is depicted below in Figure (1).

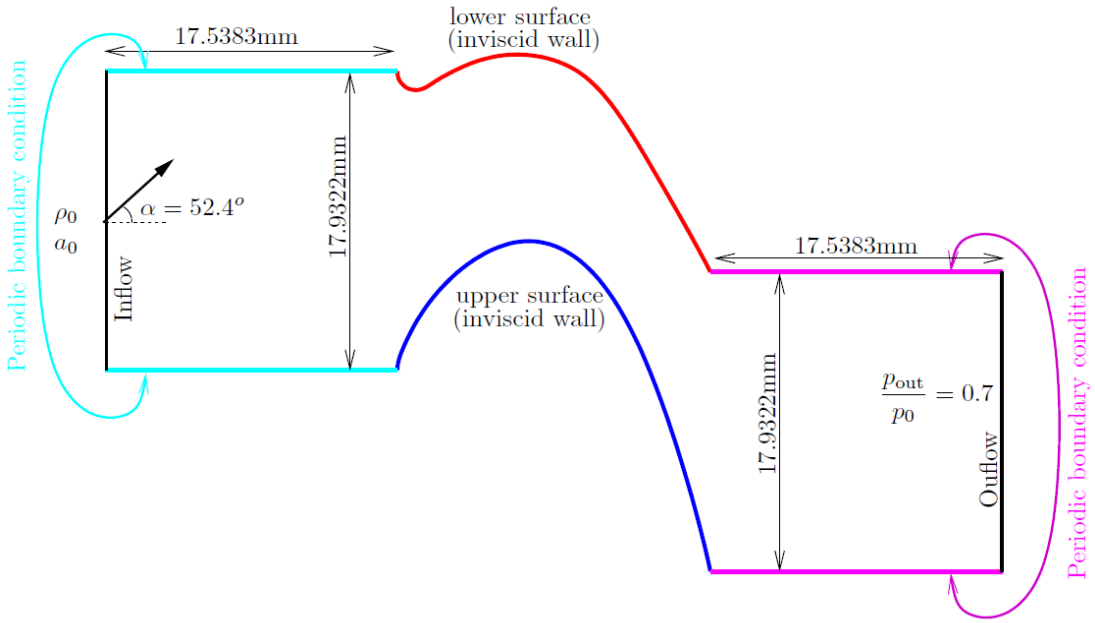

Figure 1: Turbine Blade Passage Set Up

This domain is spatially discretized into non-overlapping cells used to numerical assess the fluid flow. The provided unstructured meshes represent the domain in three different levels of refinement (cell counts of 196, 784, and 3136), and they are defined by .connect, .elem, and .node files. These contain information such as node numbers, node coordinates, and cell connectivity information.

2.2 Governing Equations

To describe the state and flux of the compressible flow on each element of the domain in two spatial dimensions, Euler equations are used in vector form as follows:

$$state : \mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \tag{1}$$

$$flux : \overrightarrow{\mathbf{F}} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix} \hat{x} + \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{bmatrix} \hat{y} \tag{2}$$

where ρ is density, u is the x-component of the velocity vector, v is the y-component of the velocity vector, p, is pressure, E is total energy per unit mass, and H is total enthalpy per unit mass. The Euler equations encode mass, momentum, and energy conservation laws. Additionally, the ratio of specific heats for the domain is $\gamma = 1.4$, and stagnation quantities speed of sound, a_0, and density, ρ, are set to O(1) units for convenience and to avoid ill-conditioning.

The initial condition for this problem is defined by the freestream stagnation density of 1, stagnation speed of sound of 1, an assumed Mach number of 0.1, and the flow angle of 52.4∘into the domain. Boundary conditions (walls, inflow, and outflow) are imposed through fluxes, which will be expanded on in later sections.

3 Solution Methods

3.1 Pre-Processing

The finite-volume method solver that is detailed below utilizes an algorithm that loops through the edges of each element, on which fluxes are computed. However, edge information is not included in the provided mesh files. To obtain edge data, the $find\_edges.m$ function was generated. It first loops through each element in the mesh and uses a hash table to track whether an edge has been checked. On the second instance that an edge is visited, the following information is stored for either the interior or boundary edges: the node numbers that define the edge, neighboring element(s), and the boundary type tag if applicable. The function detects if the edge is inside the domain or on the boundary based on the sign of the corresponding index in the mesh.connect array. Once the edges are organized into the two categories, normal vectors and edge lengths are computed and stored into the edge data structure. The meshes and the edges have the following data structure forms:

Table 1: Mesh Data Structure

Field	Values	Description
node	(x,y)	coordinates of all nodes in mesh
elem	(n1,n2,n3)	node numbers for each triangular element in mesh
connect	(e1,e2,e3)	neighboring element across each edge

Table 2: Edge Data Structure

Field	Values	Description
edges, interior	(n1,n2,e1,e2)	defining nodes and adjacent elements of each edge
edges, boundary	(n1,n2,e1,tag)	defining nodes, adjacent element, and boundary type tag
Ne	Ne	number of (interior/boundary) edges in mesh
norm	(nx,ny)	normal vectors
dl	dl	edge lengths

3.2 Flux Implementation and Tests

Because the discrete method used in this project only stores cell averages yet requires knowledge on the flux on cell interfaces, approximate numerical fluxes must be defined. For all interior edges, the given flux.m file takes in states of the two cells adjacent to the boundary as well as the unit normal vector pointing from left to right to compute the Roe Flux. Separate functions were generated to compute fluxes on the walls, inflow, and outflow.

Inviscid Wall: To impose the inviscid wall boundary condition on the upper and lower blade surfaces using $wallflux.m$, the following flux definition is used:

$$\hat{\mathbf{F}}^b = [0, p^b n_x, p^b n_y, 0]^T \tag{3}$$

The unit normal vector, $\vec{n} = n_x\hat{x} + n_y\hat{y}$, is provided from the boundary data structure as calculated in $find\_edges.m$. The boundary pressure, p^b, is dependent on the adjacent interior state:

$$p^b = (\gamma - 1)[\rho E^+ - \frac{1}{2}\rho^+ |\vec{v^b}|^2] \tag{4}$$

The quantity $\vec{v^b}$ represents the tangential boundary velocity (interior velocity with the wall-component is removed):

$$\vec{v^b} = \vec{v^+} - (\vec{v^+} \cdot \vec{n})\vec{n} \tag{5}$$

Inflow: Given the interior state in the adjacent cell, we can first compute the Riemann invariant, J^+. This quantity contains all of the information needed to construct the exterior boundary state. It is defined as:

$$J^+ = u_n^+ + \frac{2c^+}{(\gamma - 1)} \tag{6}$$

Next, we want to solve for the quantity $R * T_t$, or gas constant multiplied by the total temperature. We can get this by equation two expressions for the total speed of sound, shown by Equation (7). Taking the known stagnation quantities to be the same as the total quantities, we can solve for $R * T_t$

$$c = \sqrt{\gamma RT} = \sqrt{\frac{\gamma p}{\rho}} \tag{7}$$

$$RT_t = \frac{p_t}{\rho_t} \tag{8}$$

We can then use that value to solve for Mach number at the boundary using Equation (9). The quantity d_n in the below equation is defined as $d_n = \vec{n_{in}} \cdot \vec{n}$, and the vector $\vec{n_{in}}$ is defined as $\vec{n_{in}} = [cos(\alpha), sin(\alpha)]$, which uses the known flow angle α. We use a numeric root solver in MATLAB to obtain Mach Number at the boundary.

$$(\gamma RT_t d_n^2 - \frac{\gamma - 1}{2}(J^+)^2))(M^b)^2 + (\frac{4\gamma RT_t d_n}{\gamma - 1})M^b + (\frac{4\gamma RT_t}{(\gamma - 1)^2}) - (J^+)^2 = 0 \tag{9}$$

Using isentropic flow relations, we can obtain an expression for temperature at the boundary which can then be input into Equation (7) to get speed of sound at the boundary:

$$T^b = \frac{T_t}{1 + 0.5(\gamma - 1)(M^b)^2} \tag{10}$$

$$(c^b)^2 = \frac{\gamma RT_t}{1 + 0.5(\gamma - 1)(M^b)^2} \tag{11}$$

These values allow us to fully define the state and flux at the boundary using the Governing Euler equations, according to $\hat{\mathbf{F}^b} = \vec{\mathbf{F}}(\mathbf{u}) \cdot \vec{n}$.

Subsonic Outflow: Following the same process detailed for the inflow boundary condition, we start by computing the Riemann invariant based on the neighboring interior element's state, u^+. Using the interior quantities, we can also determine the interior entropy S^+:

$$S^+ = \frac{p^+}{(\rho^+)^\gamma} \tag{12}$$

The boundary density can be found, using the interior entropy and a known exterior pressure.

$$\rho^b = (\frac{p^b}{S^+})^{1/\gamma} \tag{13}$$

We next find the boundary velocity vector using the following expression:

$$\vec{v^b} = \vec{v^+} - (\vec{v^+} \cdot \vec{n})\vec{n} + u_n^b \tag{14}$$

where the boundary normal velocity, u_n^b, is obtained from the Riemann invariant as displayed previously and the boundary tangential velocity is equal to the interior tangential velocity. Finally, the energy term is calculated using:

$$(\rho E)^b = \frac{p^b}{\gamma - 1} + \frac{1}{2}\rho^b |\vec{v^b}|^2 \tag{15}$$

With these quantities, the state and flux at the subsonic outflow boundary can be fully defined, according to $\hat{\mathbf{F}^b} = \vec{\mathbf{F}}(\mathbf{u}) \cdot \vec{n}$.

3.2.1 Unit Testing

In order to verify the boundary flux functions performed as intended, a simple two-element four-node mesh was generated, shown in Figure (2), as an input case for a series of unit tests.

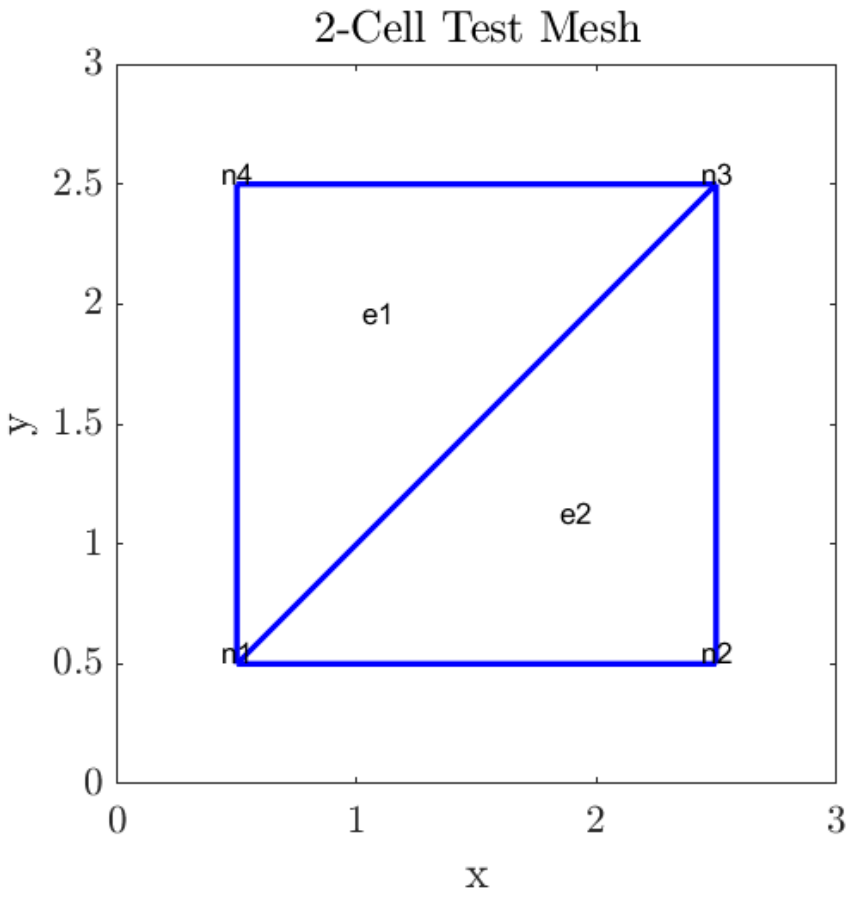

Figure 2: Two-Element Mesh for Unit Testing

Because this test mesh is only two cells, the flux and residuals for each boundary can be easily calculated by hand on a selected an initial state. These expected values are then compared with the actual function outputs using the same parameters. By comparing the hand-calculated interface fluxes with the function outputs, we can not only verify the functions run as intended, but we can also see how much error results from the calculations. The initial state chosen must have a sufficiently high energy term such that it results in a physical solution. For a chosen initial state of $u_{inf} = [1, 1, 2, 4]$, some results of the tests performed tests are detailed in Tables (3), (4), and (5).

Table 3: Upper Wall Flux Unit Test

Expected Flux	**Actual Flux**
0	0
0	0
1.4	1.4
0	0

Table 4: Inflow Flux Unit Test

Expected Flux	Actual Flux
-0.3251	-0.3462
-0.4712	-0.5945
-0.2967	-0.2465
-0.8127	-0.9792

Table 5: Outflow Flux Unit Test

Expected Flux	Actual Flux
0.7538	0.9813
1.7298	1.5969
1.7536	1.9626
3.8713	4.5319

3.3 Finite Volume Method Implementation

The two-dimensional finite volume method (FVM) was used to obtain an approximate solution to the conservation law. Upon starting the calculation, all cells in the domain are initialized to the constant freestream state and Mach number of 0.1. The function loops through all interior edges, computes the fluxes, increments or decrements the residual for the elements neighboring the edge, then updates the wave propagation speed with the values from the neighboring elements. This process is then repeated for all of the boundary edges of the domain. The flux calculations for each boundary type are performed with the functions outlined above: $wallflux.m$, $inflow.m$, and $outflow.m$.

To drive towards a steady-state solution, Forward Euler method of numerical integration is used to march the solution forward in time by updating both the state and the residual based on the flux imbalance. Local time stepping, or a time step calculation for each cell, is implemented. This integration scheme is defined below in Equation (16), with i denoting the cell number, n denoting the current iteration step, (A_i) denoting the cell area, and (Δt_i) denoting the local time step.

$$u_i^{n+1} = u_i^n - \frac{\Delta t_i^n}{A_i} R_i \tag{16}$$

The quantity $\frac{\Delta t_i^n}{A_i}$ is calculated by rearranging the Courant number (CFL) definition:

$$CFL_i = \frac{\Delta t_i}{2A_i} \sum_{e=1}^{3} |s|_{i,e} \Delta l_{i,e} \tag{17}$$

$$\frac{\Delta t_i}{A_i} = \frac{2CFL_i}{\sum_{e=1}^{3} |s|_{i,e} \Delta l_{i,e}} \tag{18}$$

A CFL number of 1 was used for all simulations in this project to prevent instabilities and convergence issues.

Finally, the L1 residual norm is logged before advancing to the next iteration. Figure (3), displays the convergence of the this quantity as the solver progresses through time. The FVM solver deems the steady-state solution accurate when the (L1) norm is less than the specified tolerance of 10^{-5}. The solver reached the steady-state in approximately 1365 iterations for Blade 0, 3508 iterations for Blade 1 and 9014 iterations for Blade 2.

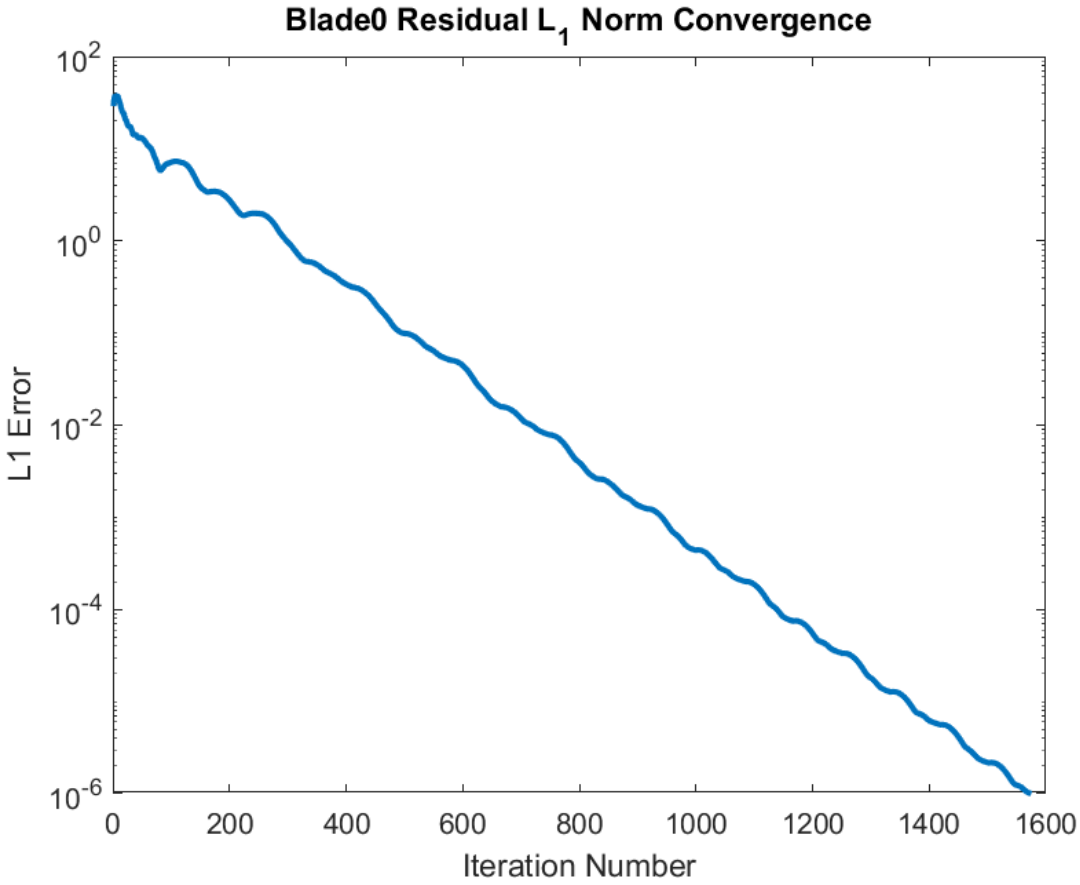

Figure 3: Convergence plot of L1 Residual Norm, Blade 0 Mesh

3.3.1 Free-stream Condition

Before the given initial conditions were passed into the FVM solver to get the steady-state solutions, as represented above in Figure (3), freestream stability tests were performed to demonstrate that the FVM solver performed as intended. In these tests, all elements in the domain were initialized to the constant free-steam state. Next, the solver calculated Roe fluxes on all boundaries by replacing the $wallflux.m$, $inflow.m$, and $outflow.m$ functions with $flux.m$ and passing in the freestream state as the right element adjacent to the interface. We first checked to verify that the residual norm is near machine-precision zero after a single iteration. Once this checkpoint was passed, the freestream test ran for 1000 iterations. By

plotting the residual norm against iterations, I confirmed the solution stayed at freestream and the residuals hovered at machine precision. Both of these tests were successfully passed and the L1 residual convergence plot for the freestream test performed on the Blade 0 mesh is shown in Figure (4).

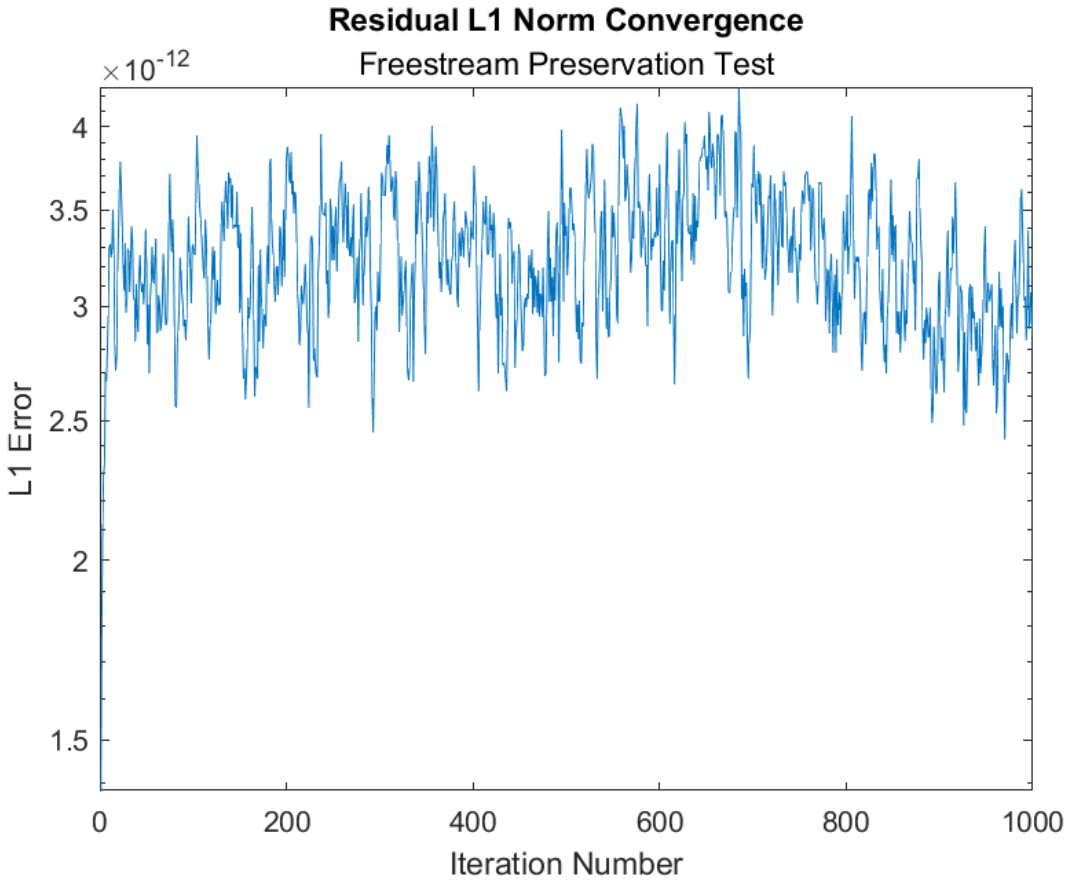

Figure 4: Freestream Preservation Test of Blade 0 Mesh, L1 Residual Norm Convergence

3.4 Post-Processing

After the solution has reached a converged state, quantities can be extracted to better visualize the flow data through the domain.

3.4.1 Mach Number

The Mach number at the converged state can be computed using the following expression:

$$M = \frac{|v|}{c} \tag{19}$$

where is the flow's velocity and is the speed of sound. The speed of sound calculation is shown below in Equation (20). The x and y components of velocity, pressure, and density values are all extracted from the converged state vector.

$$c = \sqrt{\frac{\gamma p}{\rho}} \tag{20}$$

The resulting contours of the Mach number on the converged state are shown below for each mesh, in Figures (5), (6), and (7). From these plots, we can see that the flow moves faster as it reaches the outlet. As the mesh increases in refinement, the solution is more accurately depicted and we can see smoother gradients of how the final state changes in space.

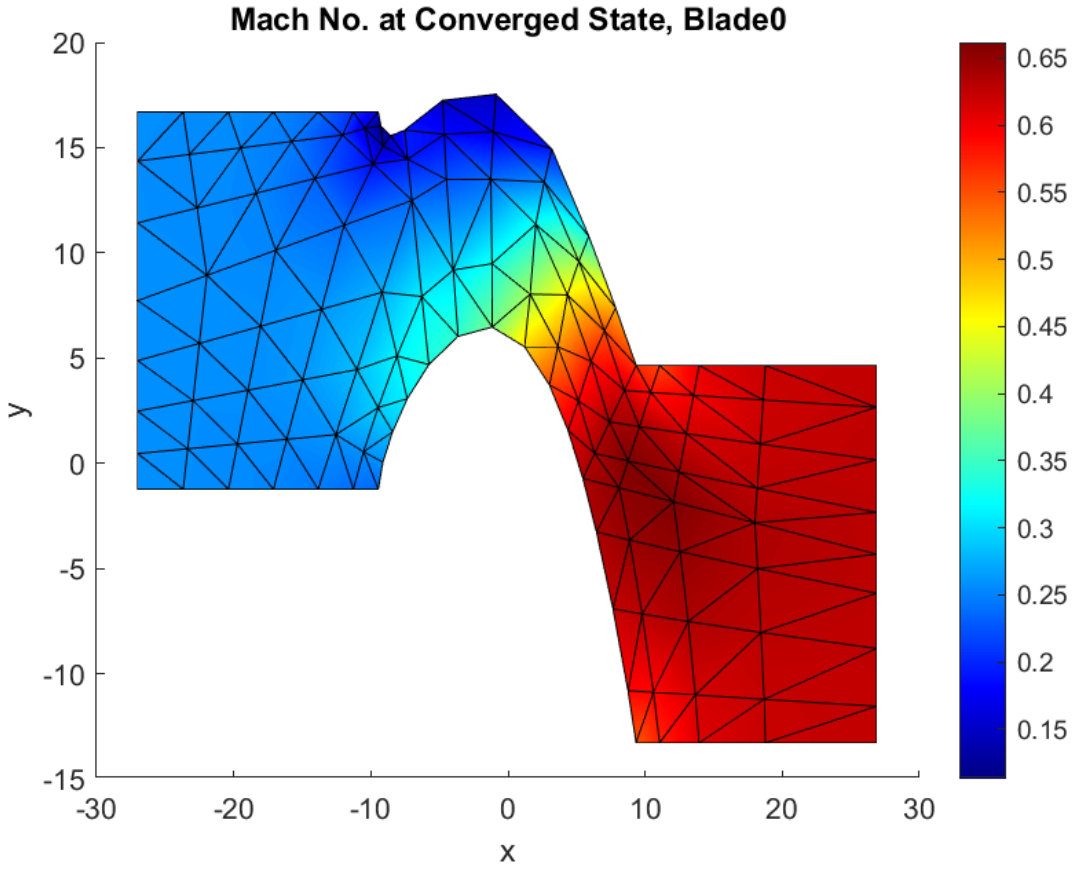

Figure 5: Mach Number contour, Blade0 Mesh

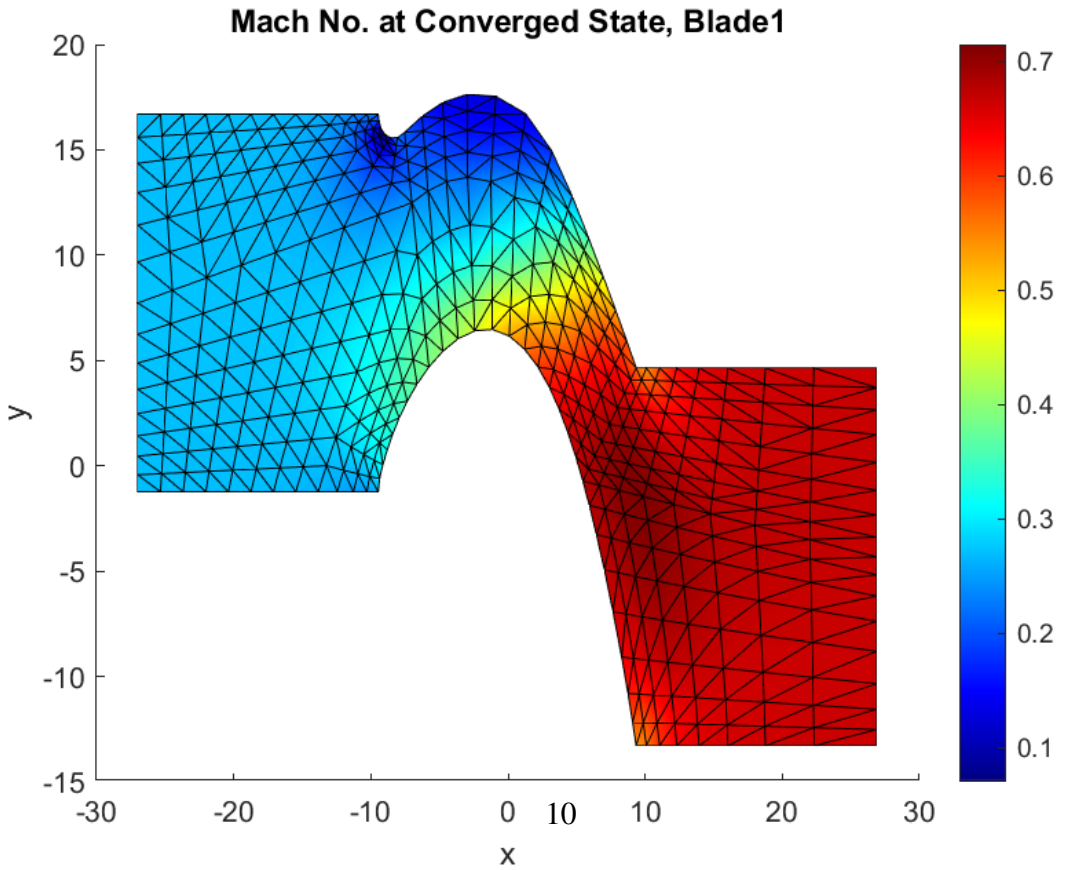

Figure 6: Mach Number contour, Blade1 Mesh

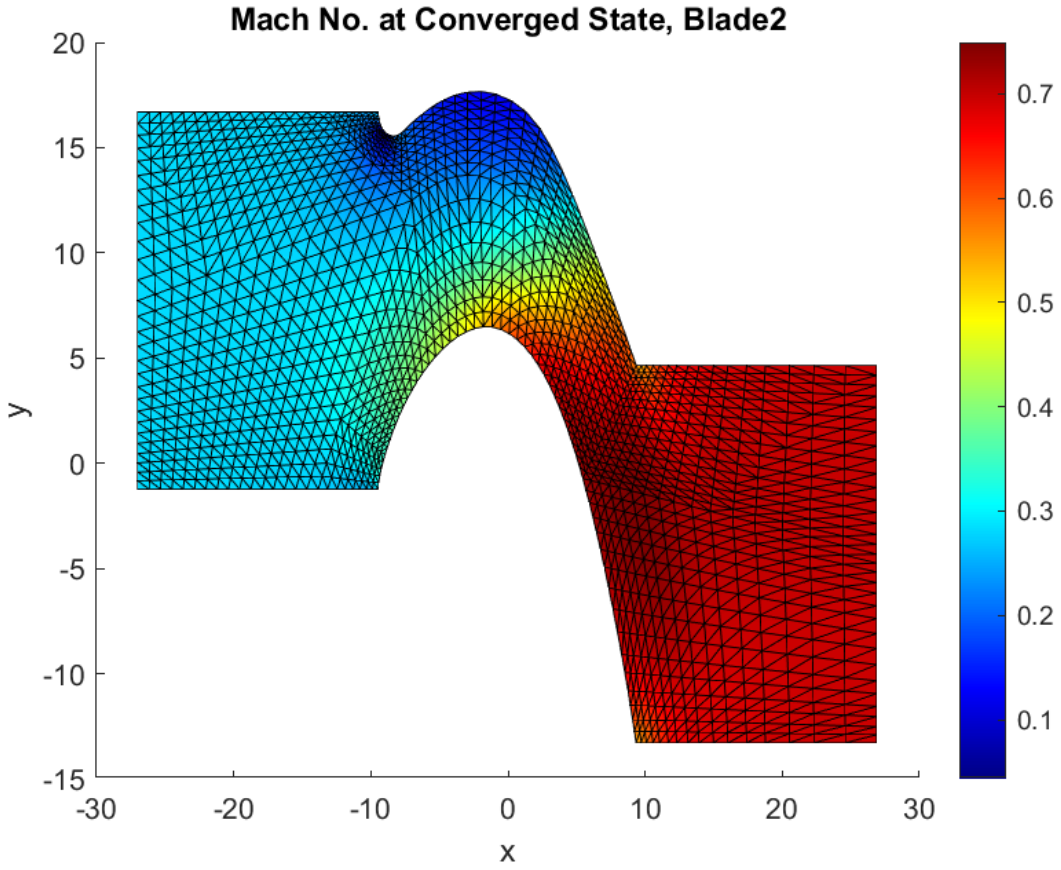

Figure 7: Mach Number contour, Blade2 Mesh

3.4.2 Force and Pressure Coefficients

The two-dimensional force vector $\vec{F'}$ was computed by iterating across the upper and lower walls of the domain and obtaining the following summation:

$$\vec{F'} = \sum p\vec{n}dl \tag{21}$$

The non-dimensional force coefficients are then obtained by:

$$C_x = \frac{F'_x}{q_{out}c_{ref}} C_y = \frac{F'_y}{q_{out}c_{ref}} \tag{22}$$

where $c_{ref} = 18.804mm$ is the reference chord length and q_{out} is the dynamic pressure outside of the domain, defined as follows:

$$q_{out} = \frac{1}{2}\gamma p_{out} M^2_{out} \tag{23}$$

The quantity M^2_{out} is the square of the Mach number outside of the domain, and it can be computed using the expression below.

$$M^2_{out} = \frac{2}{\gamma - 1}\left[\left(\frac{p_0}{p_{out}}\right)^{(\gamma-1)/\gamma} - 1\right] \tag{24}$$

Using the above equations, the force coefficients on the blade's surface for each mesh are determined in *post_process.m* using the converged, steady state. The resulting values are shown in Table (6).

Table 6: Calculated Force Coefficients, (Cx, Cy)

Blade0	**Blade1**	**Blade2**
(0.8924, 0.5899)	(0.8724, 0.6592)	(0.8580, 0.7092)

Using the same dynamic pressure calculated above, the known static pressure outside of the domain, and the static pressure on the surface pointing into the blade, we can calculate the non-dimensional pressure coefficient, as shown in Equation (25). Figure (8) displays the resulting pressure coefficients plotted against positions on the upper and lower surfaces of the blade.

$$c_p = \frac{p - p_{out}}{q_{out}} \tag{25}$$

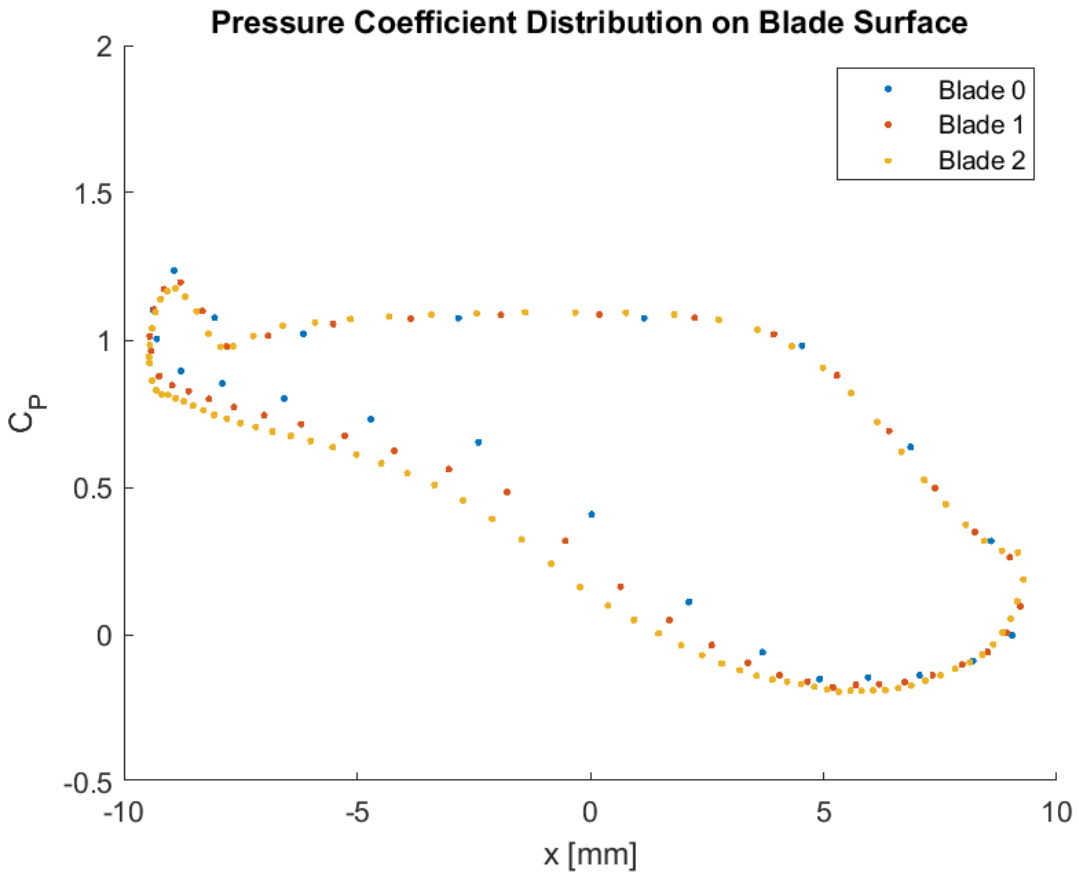

Figure 8: Pressure Coefficient on Blade Surface

4 Conclusion

A first-order finite volume solver was used to solve for the conservative states governed by the Euler equations. All of the three tested mesh-sizes we able to converge to the same criterion of $L_1 norm \leq 10^{-5}$. As expected, the mesh with the highest element count had the most accurate results but required the most iterations to reach convergence, and the smallest mesh required the least. The L1 residual norm displays some oscillatory behavior, potentially representing how the flow data propagates through cells in the domain. The x-component force coefficients decrease as the increases in element count, and the y-component force coefficients increase as mesh becomes more refined. Also as expected, the plot of Mach number on the mesh was visually smoother for the more refined mesh than for the coarser mesh. From this we can assume that information is lost within the interior of the domain for coarser meshes, and more refined meshes result in more accurate solutions.