Project 2:

FVM Analysis of Mars Entry Capsule

Tate Gill
[redacted]

Nov 20, 2018

Table of Contents:

Introduction:

Finite Volume Methods, or FVMs, are methods of discretizing a spatial domain into separate volumes to facilitate a numerical solution of fluid flow problems. The methods presented here utilize a two dimensional FVM paired with a numerical 'HLLE' flux which calculates the transport of fluid parameters between adjacent volumes. In this report, these numerical tools are applied to resolve the steady state flow solution of a hypersonic martian entry capsule governed by the Euler equations. Additionally, a technique for mesh adaptation is discussed and utilized to obtain successively better flow solutions.

Theory and Methods:

Governing Equations:

In this report we will use the Euler equations to describe the evolution of our flows. The state vector for the Euler equations consists of four quantities averaged over each element or volume in the solution mesh:

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

Respectively, density, x-momentum, y-momentum, and energy. Additionally, the fluxes between adjacent fluid elements is given from the Euler equations as:

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

Where p is pressure, and H is total enthalpy. Pressure can be calculated via the following:

$$|V| = \frac{\sqrt{(\rho u)^2 + (\rho v)^2}}{\rho} \tag{3}$$

$$p = (\gamma - 1)(\rho E - (1/2)(\rho)(|V|^2)) \tag{4}$$

Where $|V|$ is the magnitude of the flow velocity of the flow, and γ is the ratio of specific heats, which we will take as a constant 1.3 for the CO2 rich martian atmosphere.

Additionally, the local mach number will be useful in our analysis, and is calculated by:

$$c = \sqrt{\gamma p/\rho} \tag{5}$$

$$Mach = \frac{|V|}{c} \tag{6}$$

Where c can be defined as the local speed of sound.

HLLE Flux:

Due to the fact that our discrete methods require averaging of the flow state over each fluid element, we lose information about the state on the edges of the elements. This makes it impossible to calculate the flux between elements directly from the euler equations, and therefore a approximation for the flux is needed. For the capsule problem at hand, we choose to use the HLLE flux. The HLLE flux demonstrates both diffusivity and upwinding which are important characteristics for stability in our solver.

HLLE.m is the function used to compute this numerical flux approximation for the FVM solver. This function takes in states on the “Left” and “Right” elements on a boundary, as well as the boundaries normal vector, which by definition points from “Left” to “Right”, and computes the HLLE flux for the boundary as defined in the class notes. An example of this setup is shown in Figure 1. In order to prove the flux function was operating correctly, a script called *testflux.m* was designed to test *HLLE.m* in a variety of circumstances in a simple setup along the first interior edge (shown in Figure 1).

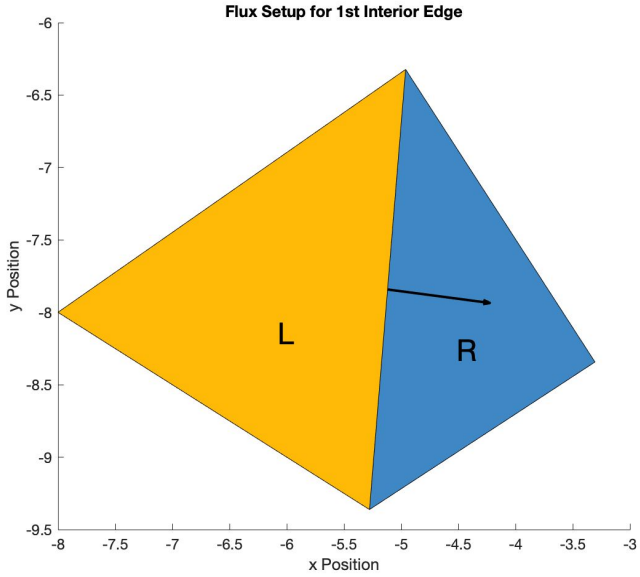

Figure 1: Flux setup for 1st interior edge.

The first test cases used subsonic inputs as shown in Table 1.

Table 1: Subsonic Input Test States.

	Mass [ϱ]	X momentum [ϱu]	Y momentum [ϱv]	Energy [ϱE]
Left	1.0000	0.5000	0.0000	2.6891
Right	1.0000	0.5492	0.0288	2.7154

The results from Case A, which used the regular normal (pointing as shown in Figure 1), are listed in Table 2. These results indicate proper functionality, as we can see that the HLLE flux is a combination of the two Euler fluxes (Calculated by Equation (2) dotted with the normal), and is biased towards the Left and upwind state.

Table 2: Subsonic Fluxes, Case A: Regular Normal

	Mass	X momentum	Y momentum	Energy
HLLE	0.5086	1.0073	-0.0867	1.7525
Euler Left	0.4973	1.0138	-0.0796	1.7199
Euler Right	0.5433	1.0635	-0.0640	1.8932

Case B used a reversed normal vector to calculate the HLLE flux, and the results are listed in Table 3. Here we see that the Euler fluxes change sign, and that the HLLE flux is now biased towards the state on the right as expected.

Table 3: Subsonic Fluxes, Case B: Flipped Normal

	Mass	X momentum	Y momentum	Energy
HLLE	-0.5320	-1.0700	0.0569	-1.8606
Euler Left	-0.4973	-1.0138	0.0796	-1.7199
Euler Right	-0.5433	-1.0635	0.0640	-1.8932

The second test cases used supersonic inputs as shown in Table 4.

Table 4: Supersonic Input Test States.

	Mass [ϱ]	X momentum [ϱu]	Y momentum [ϱv]	Energy [ϱE]
Left	1.0000	2.5000	0.0000	5.6891

Right	1.0000	2.5465	0.1335	5.8154

For the supersonic case with regular normal vector direction (Table 5) we see that the HLLE flux is now identical to the Left state Euler Flux. This shows correct operation, because at speeds larger that the max wave speed there should be no information coming from downstream or in this case the state on the right.

Table 5: Supersonic Fluxes, Case A: Regular Normal

	Mass	X momentum	Y momentum	Energy
HLLE	2.4866	6.9816	-0.0796	16.0592
Euler Left	2.4866	6.9816	-0.0796	16.0592
Euler Right	2.5190	7.1798	0.2566	16.5867

Furthermore, when we flip the normal direction as in Case B (Table 6) we see that now the HLLE flux is now identical to the Right Euler flux indicating that no information is coming from the Left state.

Table 6: Supersonic Fluxes, Case B: Flipped Normal

	Mass	X momentum	Y momentum	Energy
HLLE	-2.5190	-7.1798	-0.2566	-16.5867
Euler Left	-2.4866	-6.9816	0.0796	-16.0592
Euler Right	-2.5190	-7.1798	-0.2566	-16.5867

Finite Volume Method:

FVM.m is the function used in this report to perform the finite volume method calculations. This function takes in the mesh information in the form of a list of vertex positions (V), a list of element nodes (E), and a list of boundary edge nodes (B). From this information, *FVM.m* constructs the edge information for both interior and boundary edges.

The interior edge information comes from a function called *connect.m* which was written for the first homework assignment and modified for use on the project. This function uses a O(N) algorithm to produce a list of elements that share an edge, using sparse matrices. The function *inedgedat.m* takes this information and produces a list of edges with a data structure as follows:

e = [nA nB nx ny dl EL ER flag] (7)

Where e is the edge, nA/nB are the node indices in V, nx/ny are the x and y components of the normal vector from left to right respectfully, dl is the edge length, EL/ER are the indices of the Left and Right element in E, and flag is a boolean to determine if the edge needs to be refined.

The boundary edge information is produced in a similar fashion to the interior edge information, however we already have the list of edges from B. *bedgedat.m* is a function that takes in information from B and computes a similar data structure to that of Equation 7, and is as follows:

b = [nA nB nx ny dl EB flag] (8)

Where the only differences are that there is only one element adjacent to the boundary edges EB, and the normal always points out from EB.

Once the edge data is computed and stored, the main algorithm that *FVM.m* operates on takes over. This algorithm consists of an iteration loop that can be described as follows:

1) Zero-out wave speeds (S) and state residuals (R).
2) Calculate the fluxes for all interior and boundary edges being careful to force flow tangency on the capsule boundaries.
3) Add to the edge length weighted fluxes to the left element residuals, and subtract the edge length weighted fluxes from the right element residuals.
4) Add edge length weighted wave speeds to each element's tally.
5) Check to see if the L2 norm of the residuals is less than the desired tolerance, and If True, end.
6) Calculate the local time step/element area by the following:
$$\frac{\Delta t}{A_j} = 2 * CFL/S_i$$
7) Update states by the following formula:
$$u_j^{n+1} = u_j^n - \frac{\Delta t}{A_j}(R_j)$$
8) Repeat until step 5 is satisfied.

FVM.m also has an advanced feature, that dynamically adjusts the CFL number if the solution becomes unphysical. In short, if a negative pressure is calculated an error is

thrown, the solver returns to the previous physical state, and the CFL number is reduced by 0.5 (nominally 0.9) until the end of the current 100 iteration cycle. This ensures both speed, and stability when dealing with fine meshes.

Additionally, a free stream stability test was run to demonstrate that *FVM.m* was performing its calculations correctly. In this test every element's state was set to the free stream of Mach 8 at zero angle of attack, and the inviscid wall boundary condition on the capsule was turned off. The solver was left to run for 2000 iterations, and the final mach field is shown in Figure 2. We can conclude that our solver is operating correctly as our solution stayed at free stream, and the residuals stayed at machine precision.

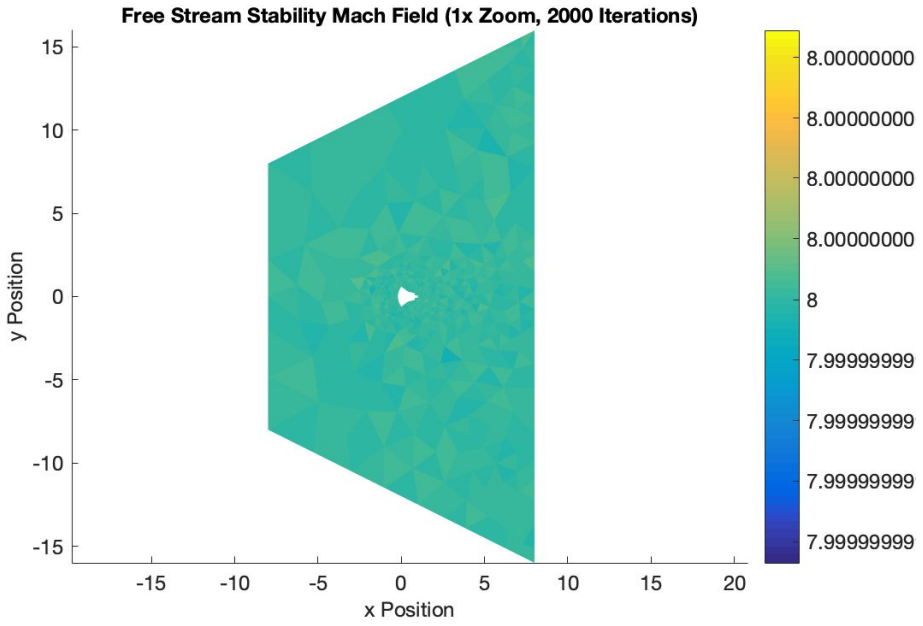

Figure 2: Free Stream Stability Test, Final Mach Field.

Results and Discussion:

FVM Baseline Mesh Results for Alpha = 5°:

The first solution run was on the initial (baseline) mesh at an angle of attack (alpha) of 5 degrees. The results from this solution are presented in Figures 3-7.

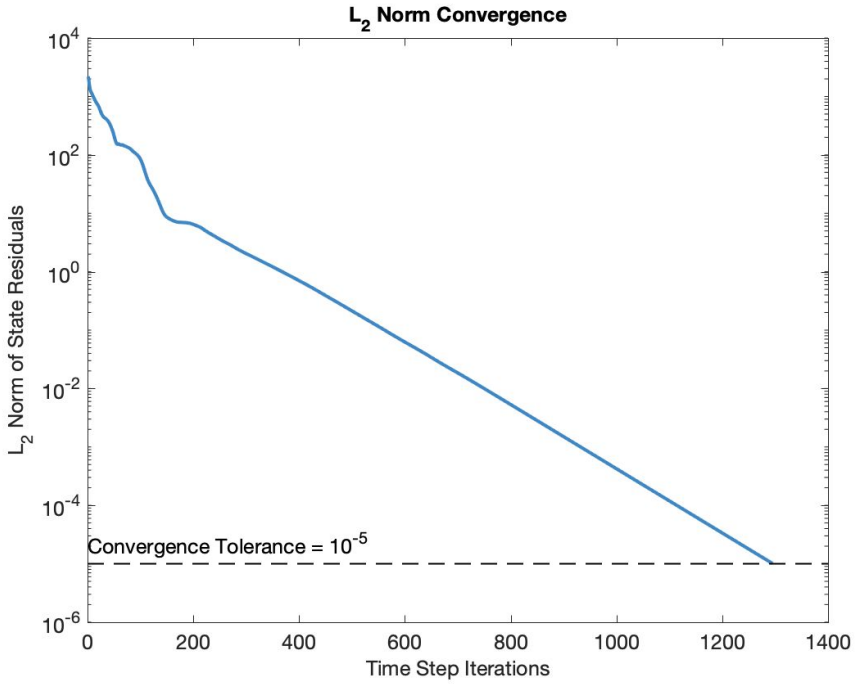

Figure 3: L2 Norm Convergence for Baseline Mesh

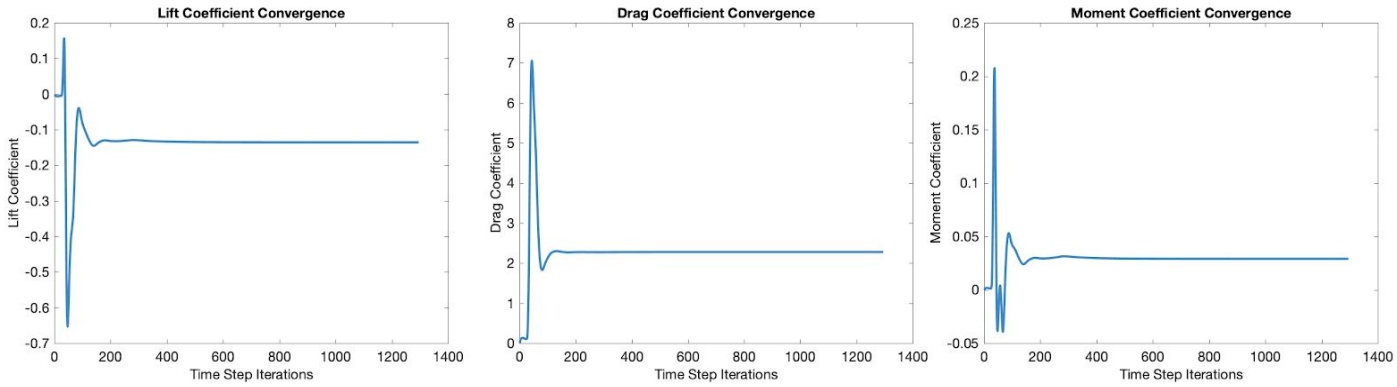

Figure 4: Aerodynamic Coefficients Convergence for Baseline Mesh.

Here in Figures 3 and 4 we see the the convergence of the L2 norm of the state residuals and the aerodynamic coefficients. It is of note that for approximately the first

200 iterations the solution was in a clear startup state, where the flow had to adjust to the capsule's presence, and after this the residual reduces linearly (log scale) as the iterations increase as expected.

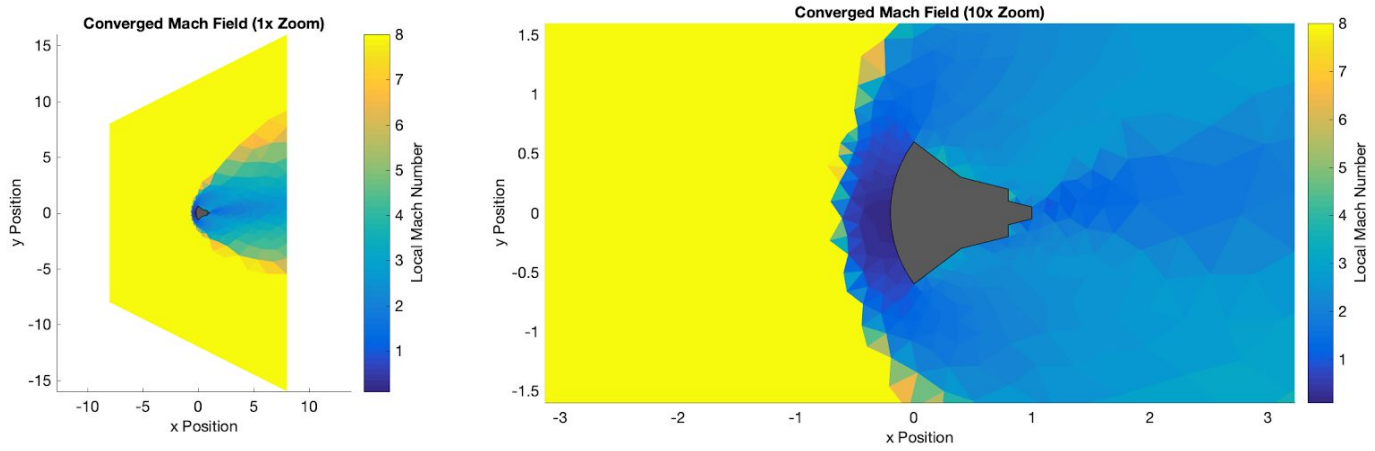

Figure 5: Converged Mach Field for Baseline Mesh.

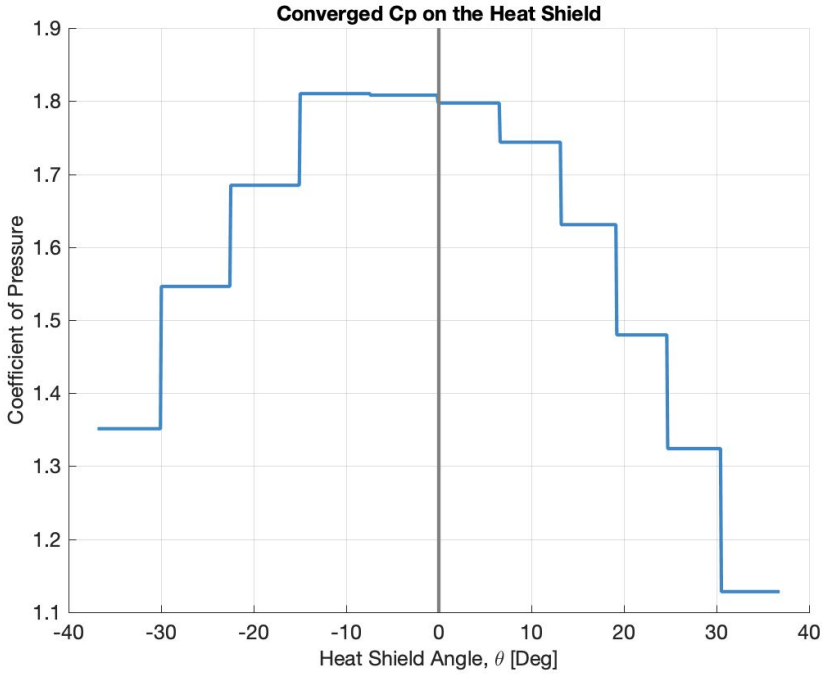

Figure 6: Converged Pressure Coefficient on Heat Shield for Baseline Mesh.

Figures 5 and 6, show the converged mach field and coefficient of pressure plot for the heat shield. We can see from these plots that the baseline mesh is very coarse, and that further refinement is needed for an accurate solution.

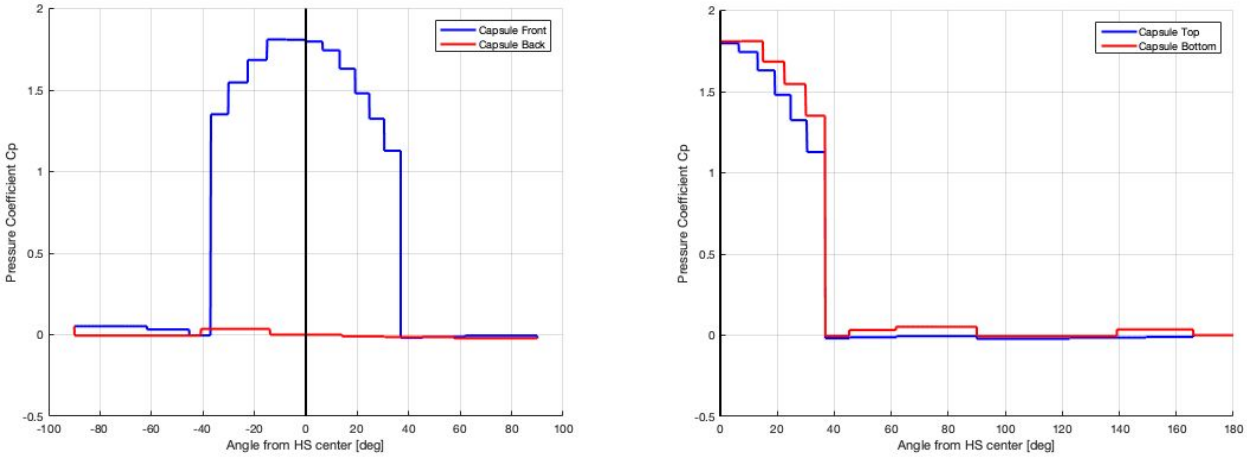

Figure 7: Converged Pressure Coefficient on Capsule for Baseline Mesh.

Figure 7, shown above, serves as a sanity check on our aerodynamic coefficient results from Figure 4. We can see from Figure 4 that the converged lift coefficient is negative for a positive angle of attack, and this would go against intuition. However, we are not dealing with a lifting body, and it is clearly seen in figure 7 that the axial force (The difference between blue and red on the left) clearly dominates the normal force (The difference between blue and red on the right). So, even for a small angle of attack the lift component of the axial force will be larger than the lift component of the normal force, and therefore the lift will be negative.

Adaptation Run: Alpha = 0°

The first adaptive run performed was at a angle of attack of zero degrees, and the results of this run are shown for the 1st, 3rd, and 5th adaptive iteration in Figures 8, 9, and 10. Figures 11 and 12, show the convergence of aerodynamic coefficients, and the converged coefficient of pressure for the heat shield for the 1st, 3rd, and 5th adaptive iteration respectively.

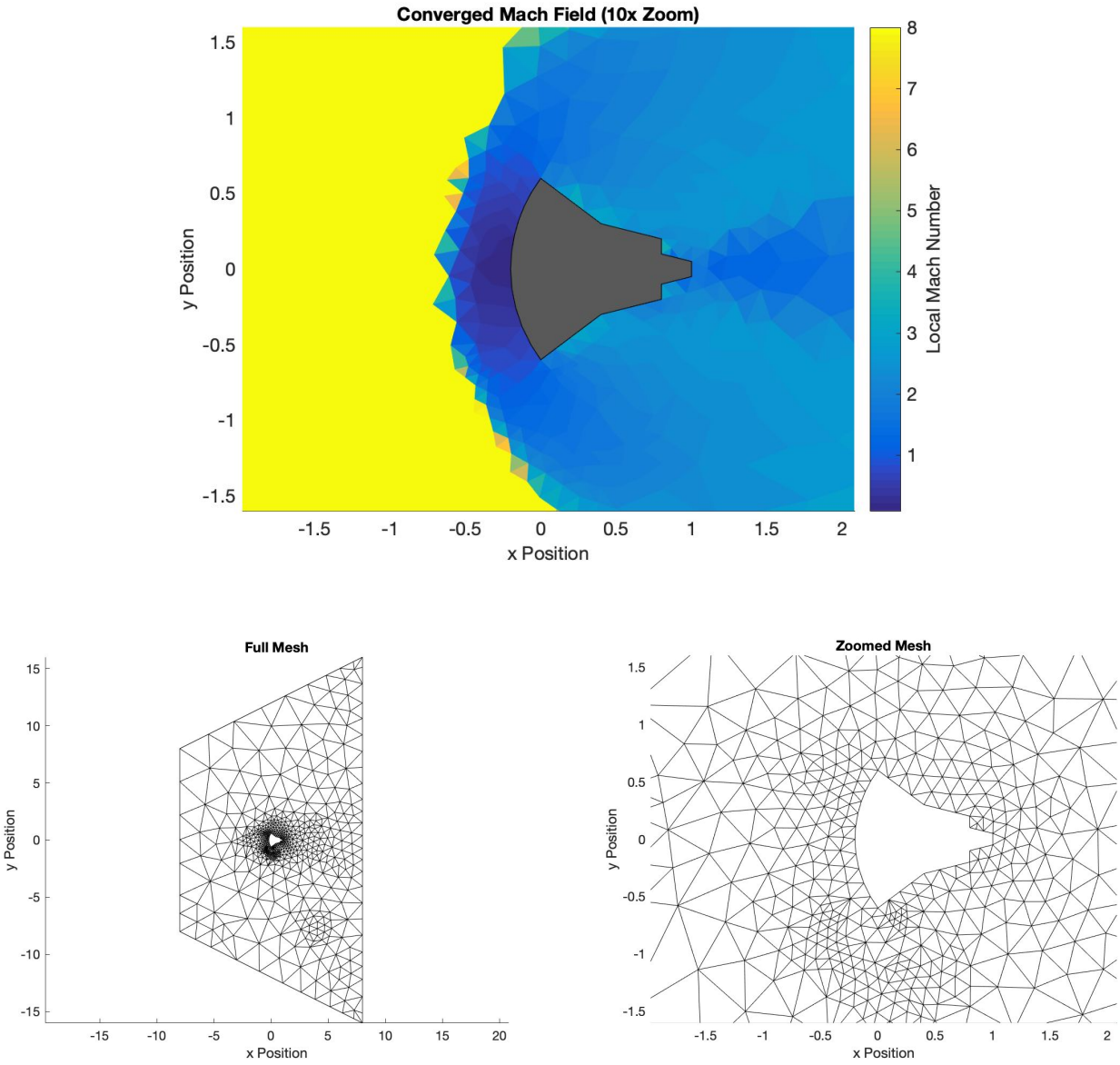

Figure 8: 0 degree AOA, 1st Mesh, Converged Mach Field and Mesh Plots.

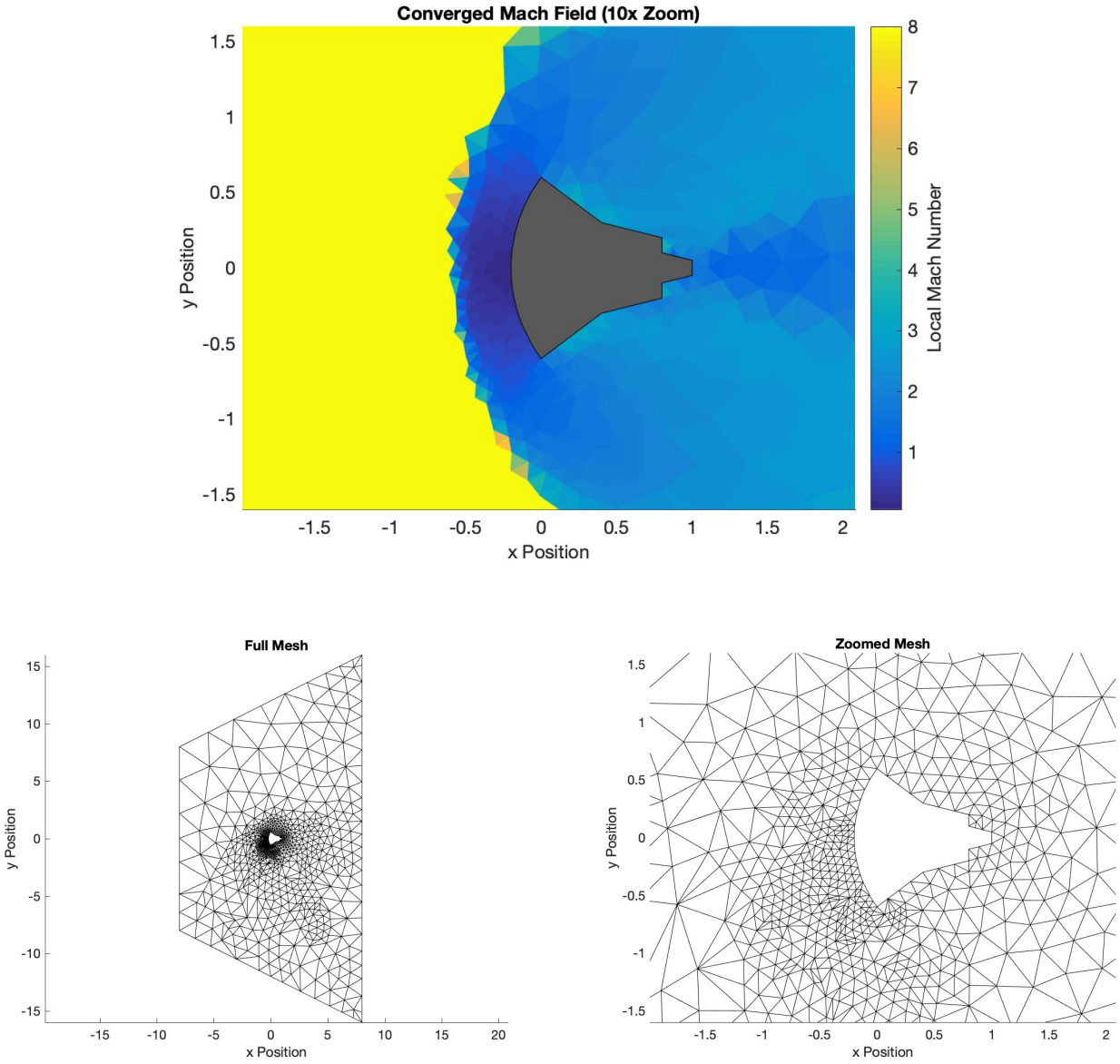

Figure 9: 0 degree AOA, 3rd Mesh, Converged Mach Field and Mesh Plots.

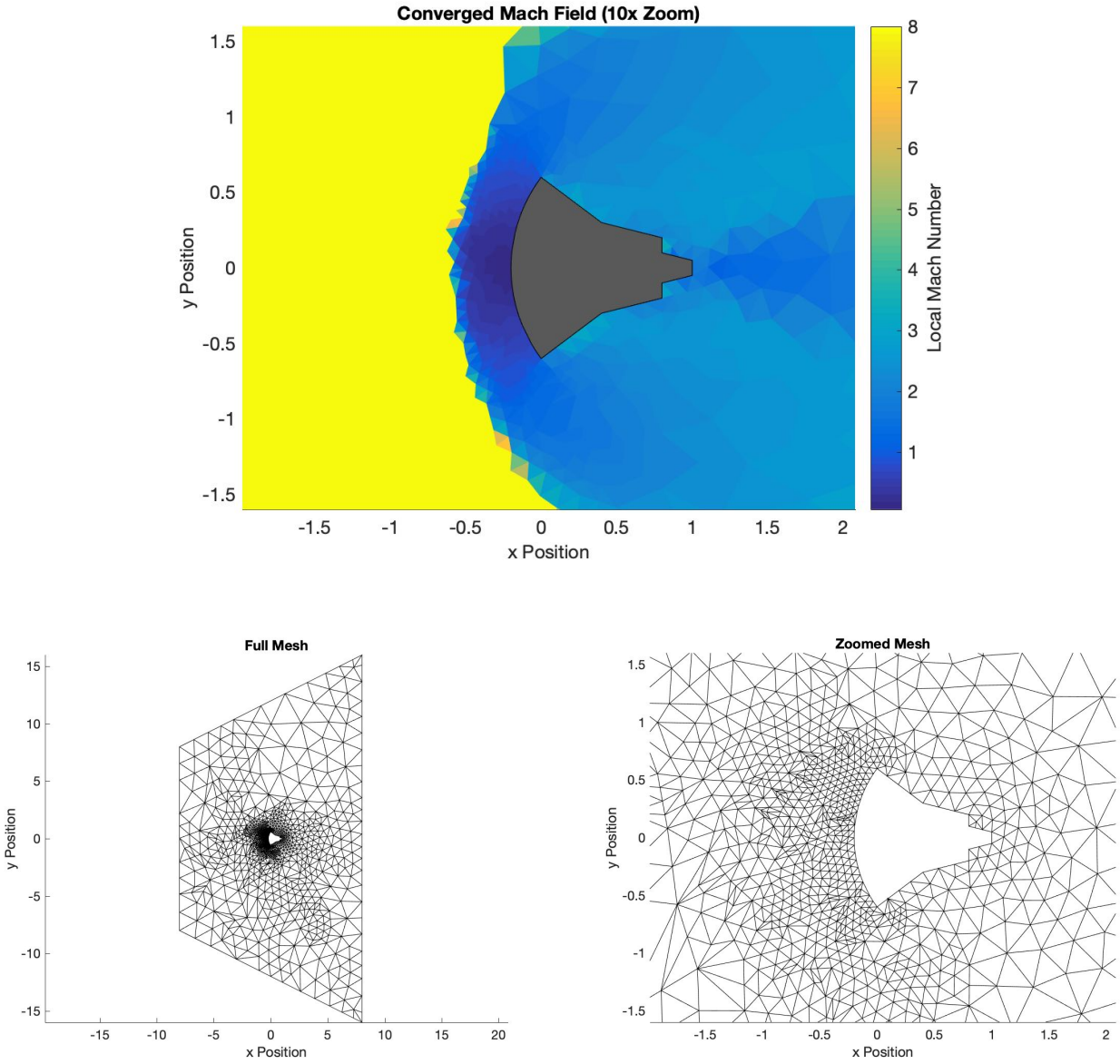

Figure 10: 0 degree AOA, 5th Mesh, Converged Mach Field and Mesh Plots.

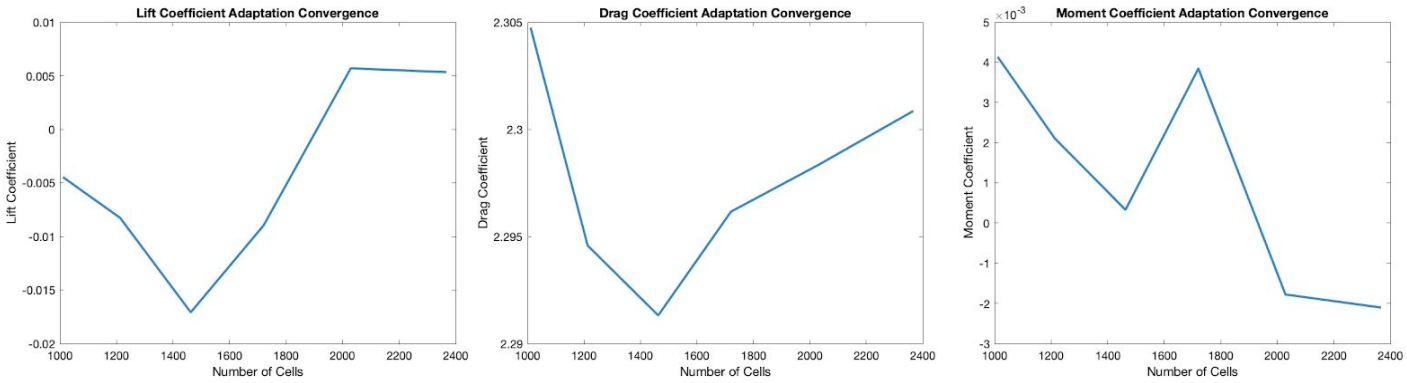

Figure 11: 0 degree AOA, Convergence of Aerodynamic Coefficients vs Number of Cells.

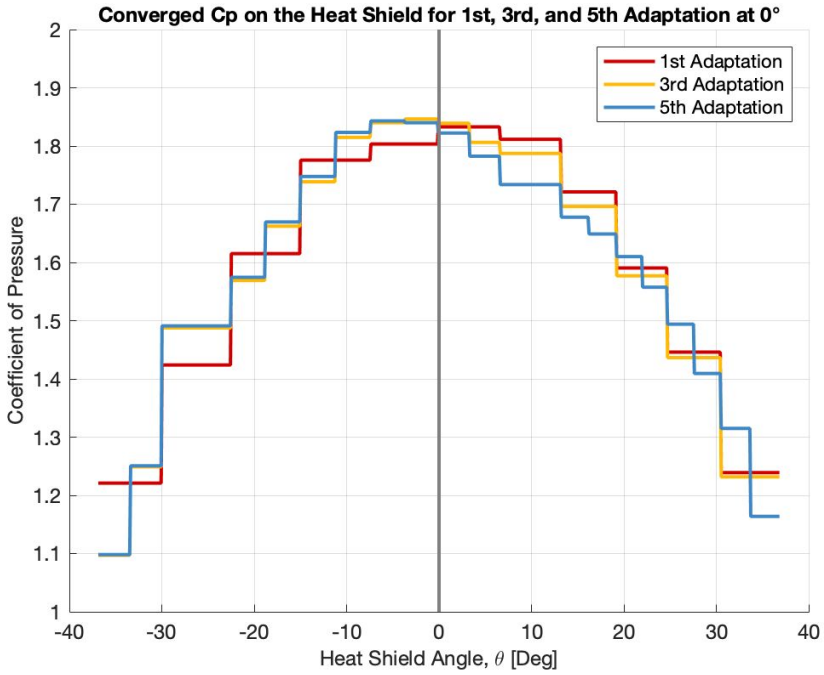

Figure 12: 0 degree AOA, Converged Pressure Coefficient on Heat Shield.

Adaptation Run: Alpha = 5°

The second adaptive run performed was at a angle of attack of five degrees, and the results of this run are shown for the 1st, 3rd, and 5th adaptive iteration in Figures 13, 14, and 15. Figures 16 and 17, show the convergence of aerodynamic coefficients, and the converged coefficient of pressure for the heat shield for the 1st, 3rd, and 5th adaptive iteration respectively.

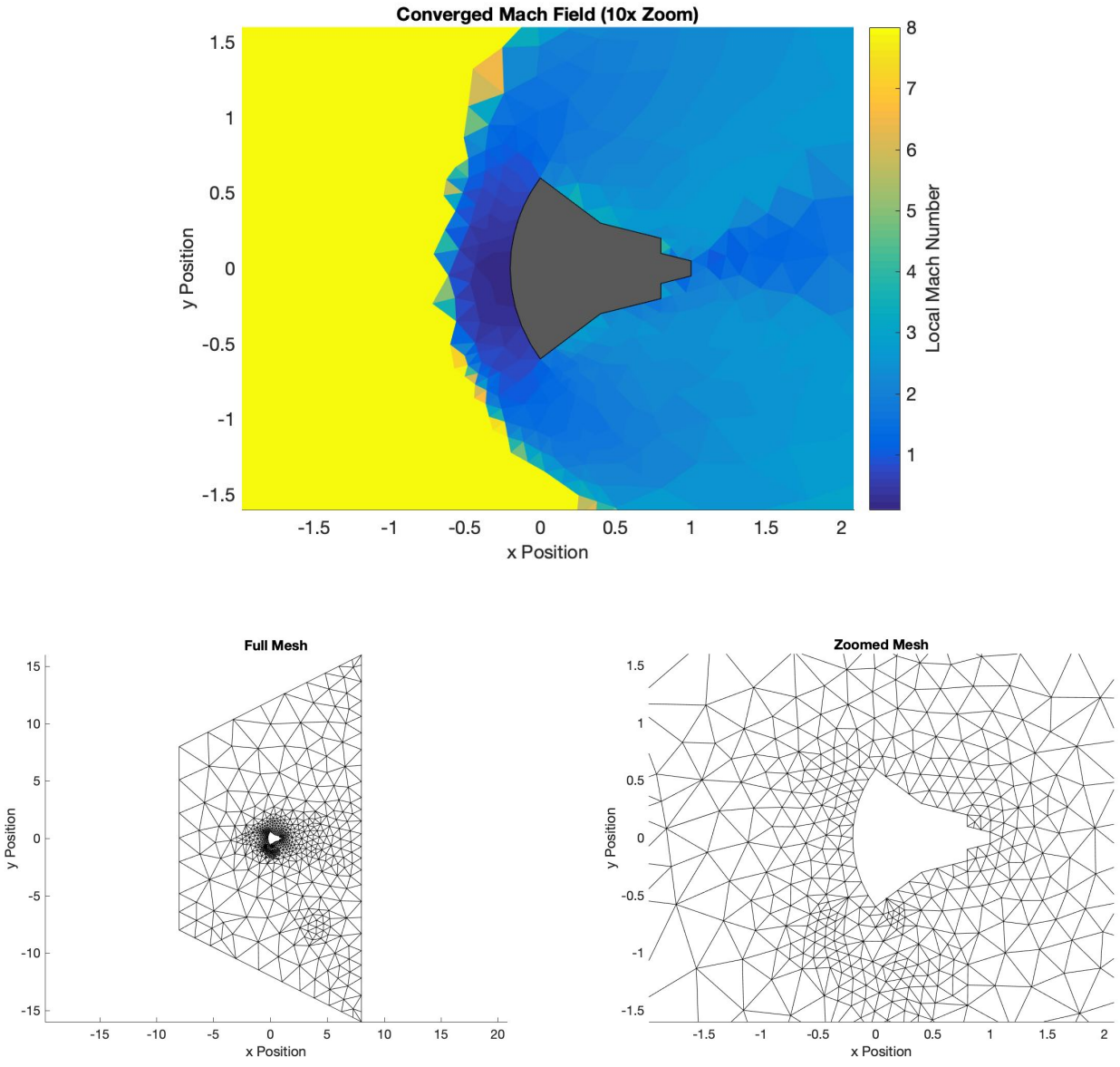

Figure 13: 5 degree AOA, 1st Mesh, Converged Mach Field and Mesh Plots.

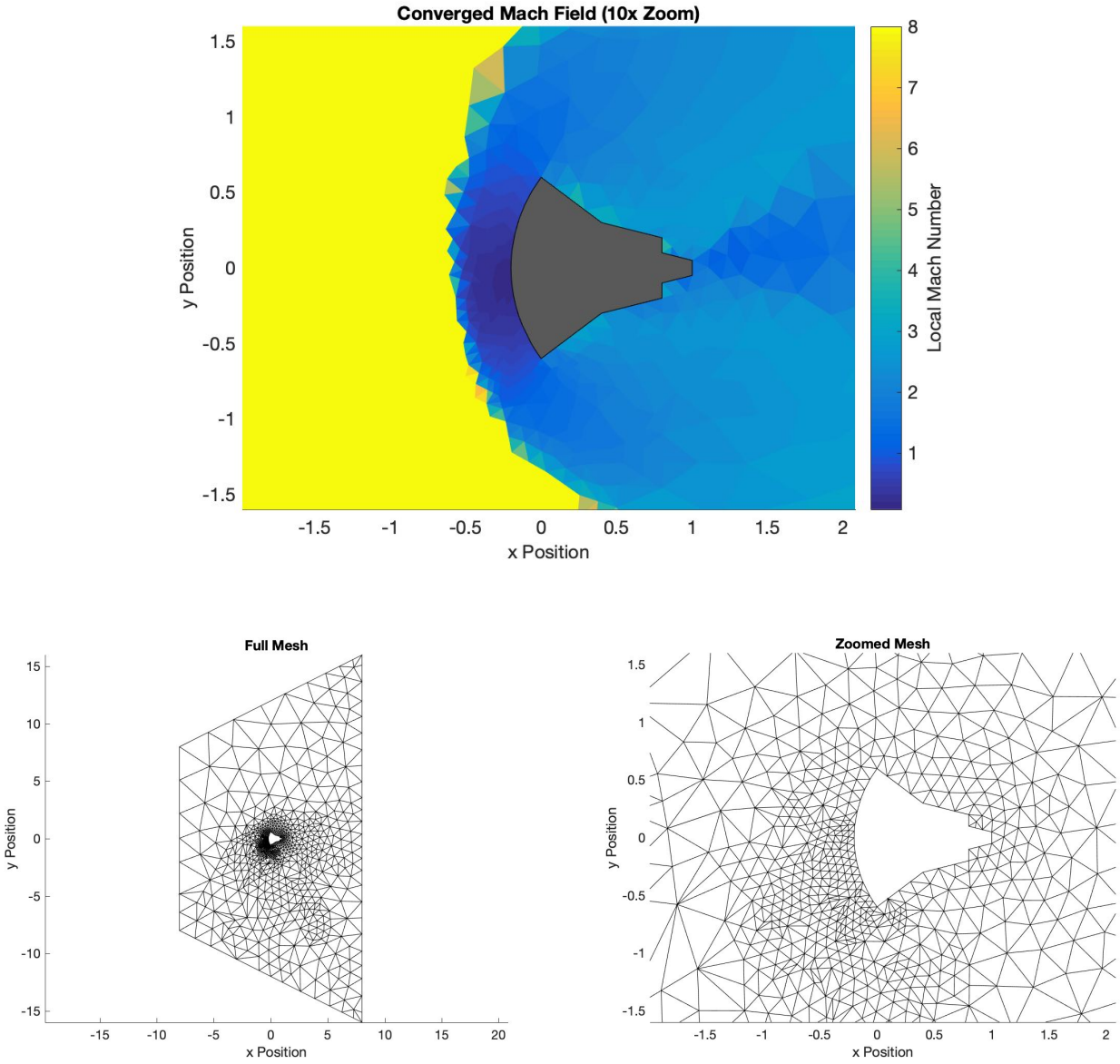

Figure 14: 5 degree AOA, 3rd Mesh, Converged Mach Field and Mesh Plots.

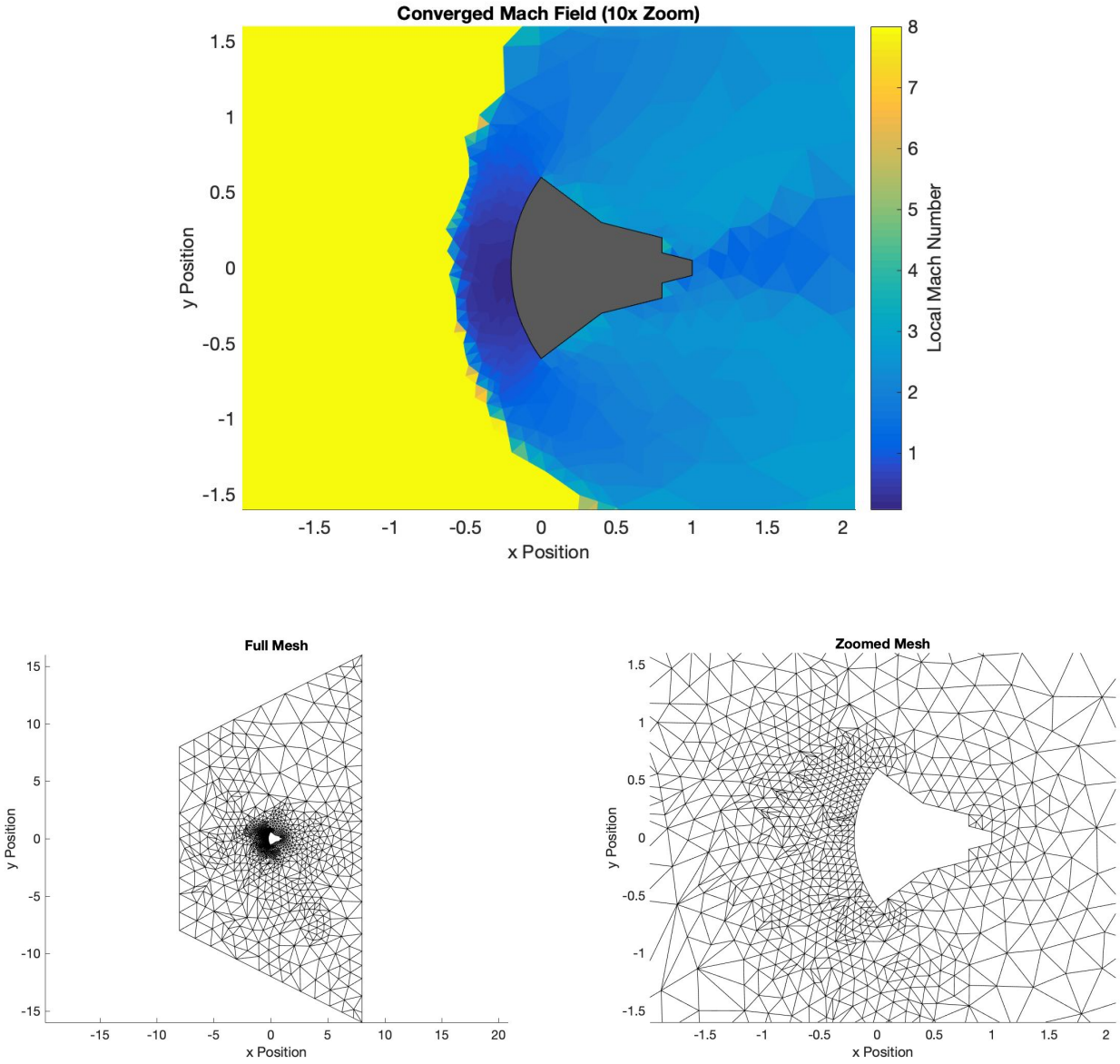

Figure 15: 5 degree AOA, 5th Mesh, Converged Mach Field and Mesh Plots.

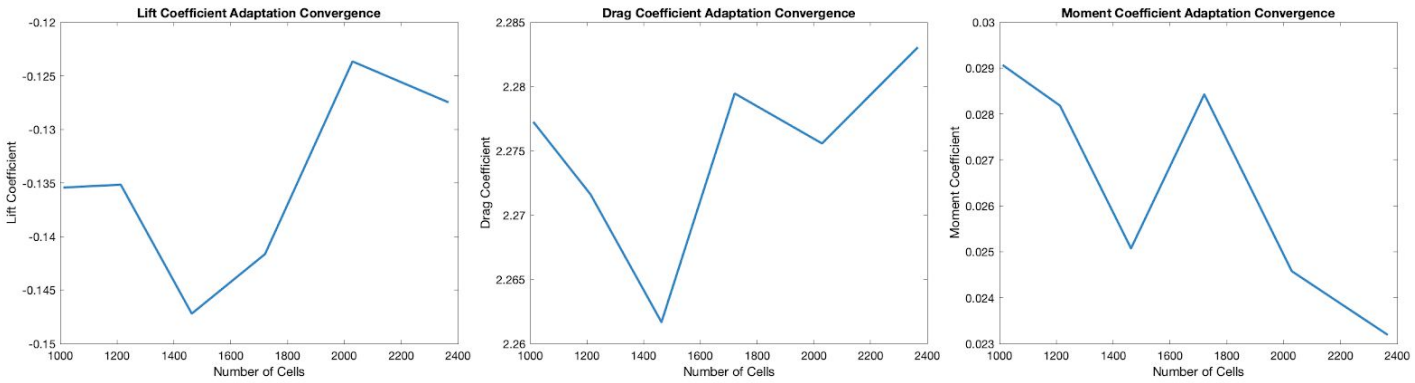

Figure 16: 5 degree AOA, Convergence of Aerodynamic Coefficients vs Number of Cells.

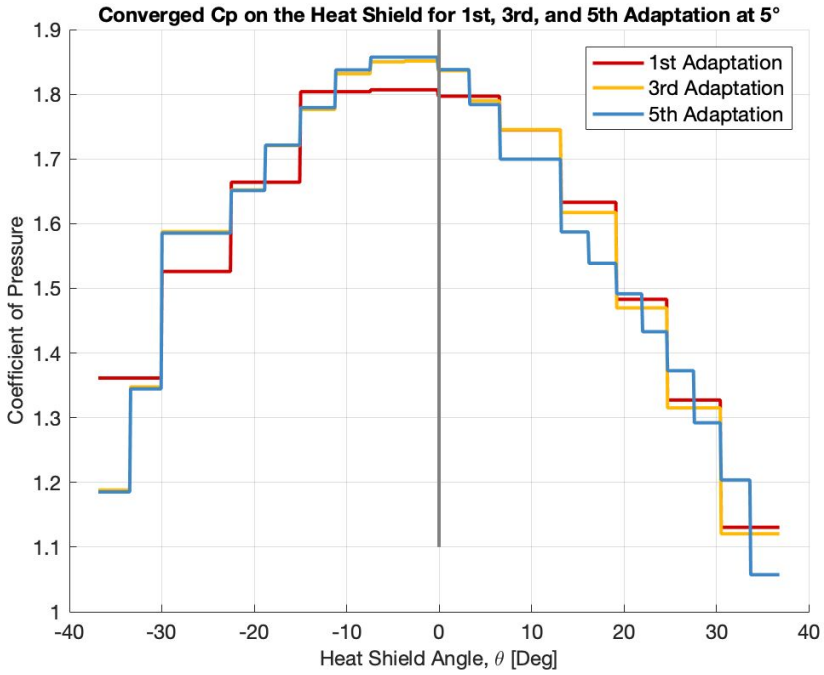

Figure 17: 5 degree AOA, Converged Pressure Coefficient on Heat Shield.

Adaptation Run: Alpha = 10°

The third adaptive run performed was at a angle of attack of ten degrees, and the results of this run are shown for the 1st, 3rd, and 5th adaptive iteration in Figures 18, 19, and 20. Figures 21 and 22, show the convergence of aerodynamic coefficients, and the converged coefficient of pressure for the heat shield for the 1st, 3rd, and 5th adaptive iteration respectively.

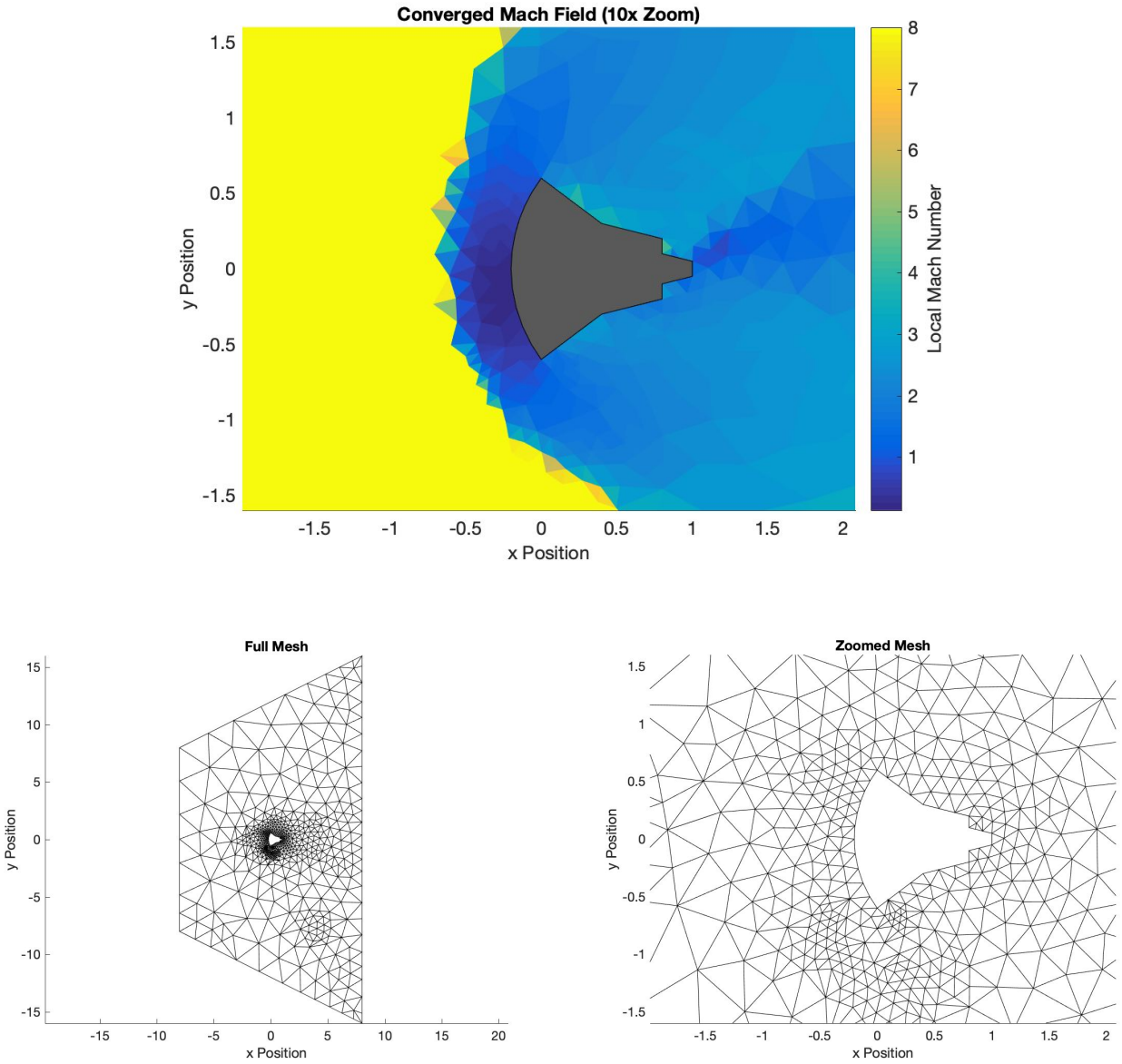

Figure 18: 10 degree AOA, 1st Mesh, Converged Mach Field and Mesh Plots.

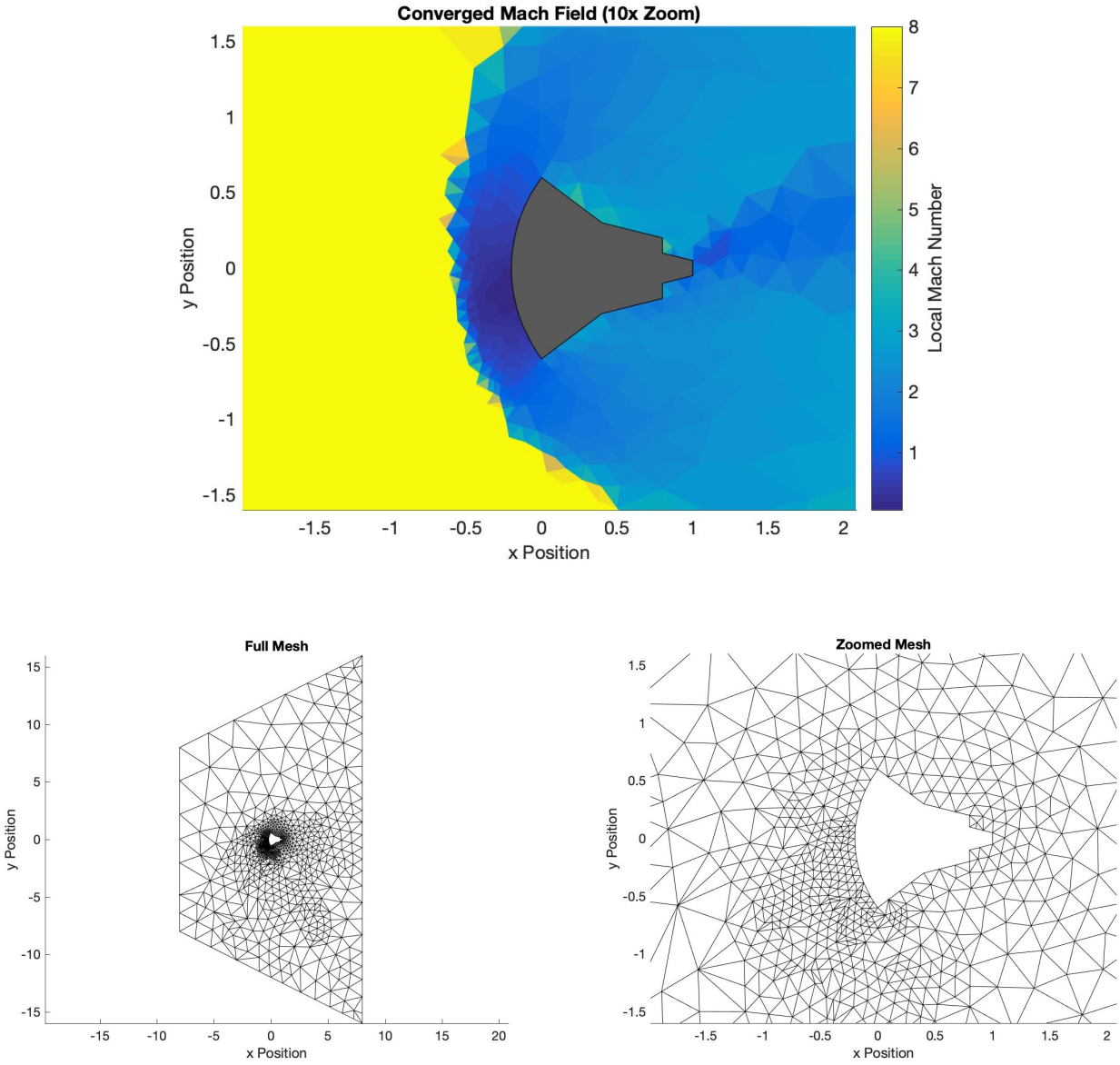

Figure 19: 10 degree AOA, 3rd Mesh, Converged Mach Field and Mesh Plots.

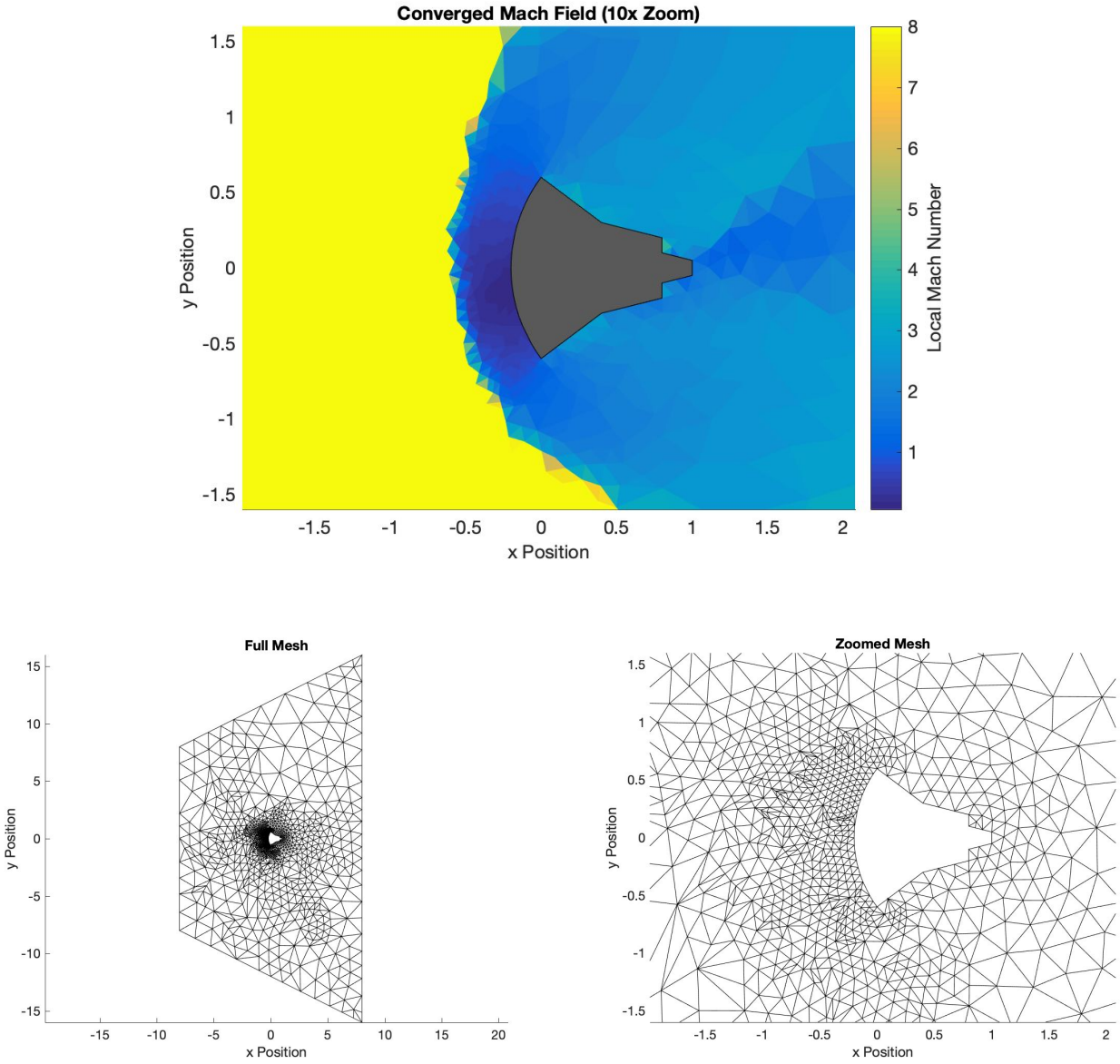

Figure 20: 10 degree AOA, 5th Mesh, Converged Mach Field and Mesh Plots.

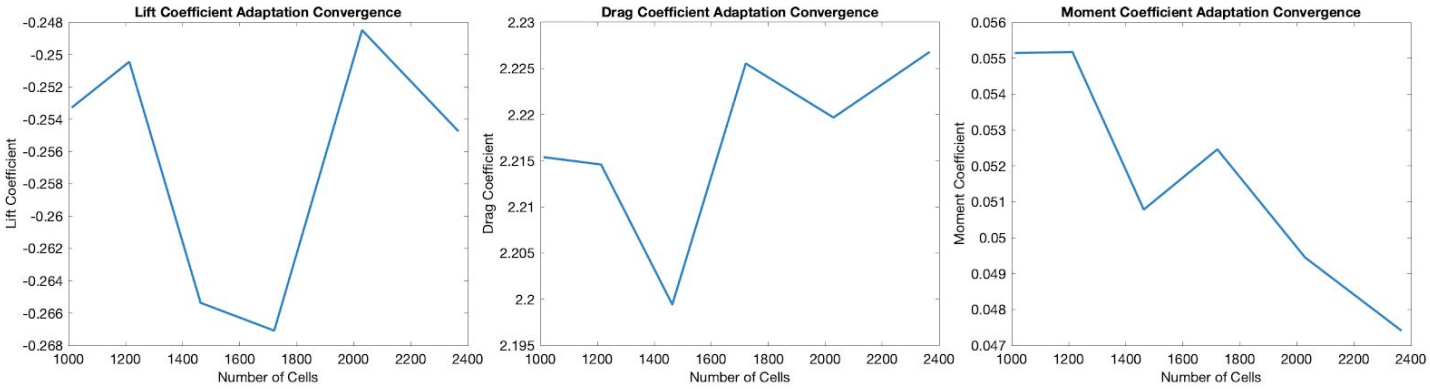

Figure 21: 5 degree AOA, Convergence of Aerodynamic Coefficients vs Number of Cells.

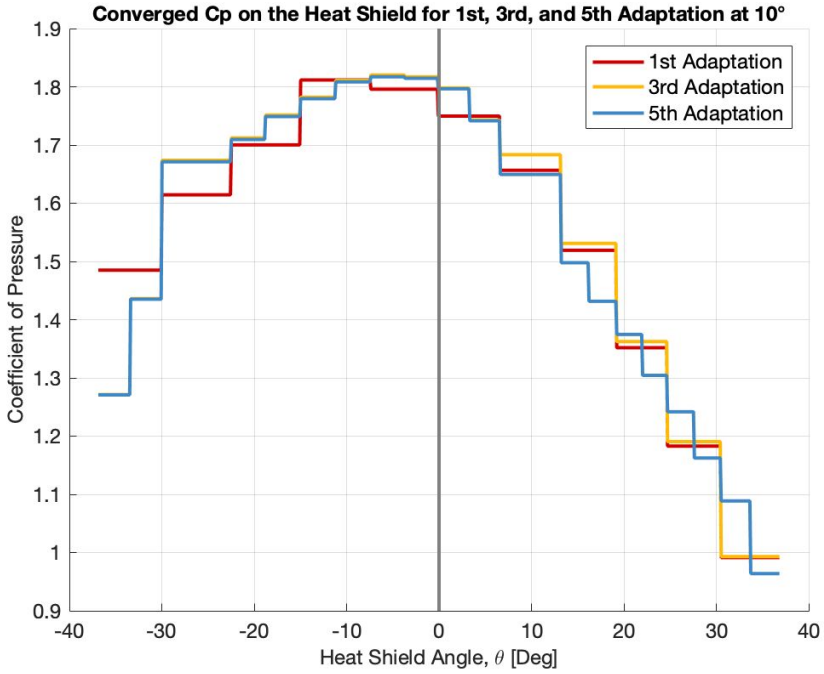

Figure 22: 5 degree AOA, Converged Pressure Coefficient on Heat Shield.

The results of the adaptive iteration runs are shown above in Figures 8-22. From the mach field and mesh plots, we can see that most of the refinement is occurring near the capsule and at the shock location, with the refinement happening more towards the bottom for the higher angles of attack. This is not surprising, as we defined the error for refinement as large mach jumps. The aerodynamic coefficient convergences shown in Figures 11, 16, and 21, display an interesting result, in that as the number of cells is increased there seems to be no real correlation in the direction of convergence for the coefficients. However it is of note, that for all runs they stay at relatively the same level, and that jump appear ‘qualitatively’ to have more to do with refinement along the capsule than with the number of cells in the mesh total. Intuition would lead to the conclusion that when there is more degrees of freedom for forces (pressures) on the capsule, that the converged solution would take larger jumps.