Chapter 1

Introduction

1.1 Description

A finite volume method is used to simulate hypersonic flow over a capsule entering Mars. An adaptive first order method is used.
For this work, it is assumed that the Euler equations are valid to describe the flow conditions during entry. Since the atmosphere of Mars consists of mostly CO_2, the value of $\gamma = 1.3$.
The free-stream state is defined to be

$$u_\infty = [\rho, \rho u, \rho v, \rho E]^T = [1, M_\infty cos\alpha, M_\infty sin\alpha, \frac{1}{\gamma(\gamma-1)} + \frac{M_\infty^2}{2}]^T \tag{1.1}$$

where M_∞ is the free stream Mach number and $alpha$ is the angle of attack.
The flow is hypersonic and the free stream Mach Number is $M_1 = 8$ and the angle of attack, α varies from 0°to 10°.

1.2 Boundary Conditions

The computational domain for the current problem consists of the region outside the capsule geometry. The inflow portion of the far-field boundary consists of the left, bottom and top boundaries. The free-stream conditions are applied on these boundaries i.e $M_1 = 8$. The far-field boundaries on the right are assumed to be supersonic and the flux is calculated from the interior state. For the capsule, inviscid wall boundary condition is applied as flow doesn't enter the capsule.
To initialize the state in a new run, all cells are set to the same state. All the states are initialized to $M_1 = 1.3$, a supersonic Mach number to make the solution more robust. For the subsequent runs, $M_1 = 8$ is applied at the inlet boundary.

Chapter 2

Methods

The methods used to execute the given tasks were as follows :

2.1 Task : 1

In order to test our implementation of HLLE flux, we designed a program that computes the HLLE flux at the boundary, given the states on either side of that boundary. The effectiveness of our code was tested by determining the boundary flux for subsonic and supersonic states and varying normal vectors. The working of this function is illustrated by 'Algorithm 1' and 'Algorithm 2'.

2.1.1 Case-1

This case corresponds to the condition where the states on either side of the boundary are supersonic(Mach numbers 2 and 1.5). 'Figure 3.1' shows the X and Y components of the normal vector to the boundary and the four components of the flux across the boundary.

2.1.2 Case-2

This case corresponds to the condition where the states on either side of the boundary are subsonic(Mach numbers 0.5 and 0.8). 'Figure 3.2' shows the X and Y components of the normal vector to the boundary and the four components of the flux across the boundary.

2.2 Task : 2

2.2.1 Code Structure

Our code is structured so as to contain a single 'main()' function which references, passes variables into and receives outputs from functions. The working of the main function is illustrated by 'Algorithm 3'

The functions used are :

geometryinit() : This function reads the given mesh .gri file and creates the E , V , boundaryin , boundaryout and boundarycapsule arrays. The E array is used to compute the elemnormal array. A node-element connectivity hash table was created and this was used to create the globaledgenumber and edgevselement arrays. The calculation of outward normal is illustrated by 'Algorithm 4'. The creation of the hash table is illustrated by 'Algorithm 5' and the creation of the 'globaledgenumber' array is illustrated by 'Algorithm 6'.

Initialize() : This function is used to initialize all the states to free stream conditions initially.The working of this function is illustrated by 'Algorithm 7'.

edgeiteration() : This function is used to loop over the edges and calculate the flux over each of the edges using the HLLE flux. The inlet , outlet and inviscid wall boundary conditions are applied inside this function. This function returns the cell averaged residual and wave speed values.The looping over interior elements is illustrated by 'Algorithm 8,9,10'. The enforcing of the Inlet Boundary condition is illustrated by 'Algorithm 11,12'. The enforcing of the Outlet Boundary condition is illustrated by 'Algorithm 13,14'. The enforcing of the Capsule Boundary condition is illustrated by 'Algorithm 15'.

timestepping() : This function loops over all the elements, determines the local timestep using the cell averaged wave speed and time steps the state vectors. The new flux vectors are calculated from the time stepped state vectors. The working of this function is illustrated by 'Algorithm 17'.

L2errorcalc() : This function uses the cell averaged residues to calculate the L2 error at a particular time step.

forcemoment() : This function calculates the Lift, Drag , Lift Co-efficient , Moment Co-efficient and Pressure Co-efficients. The working of this function is illustrated by 'Algorithm 19'.

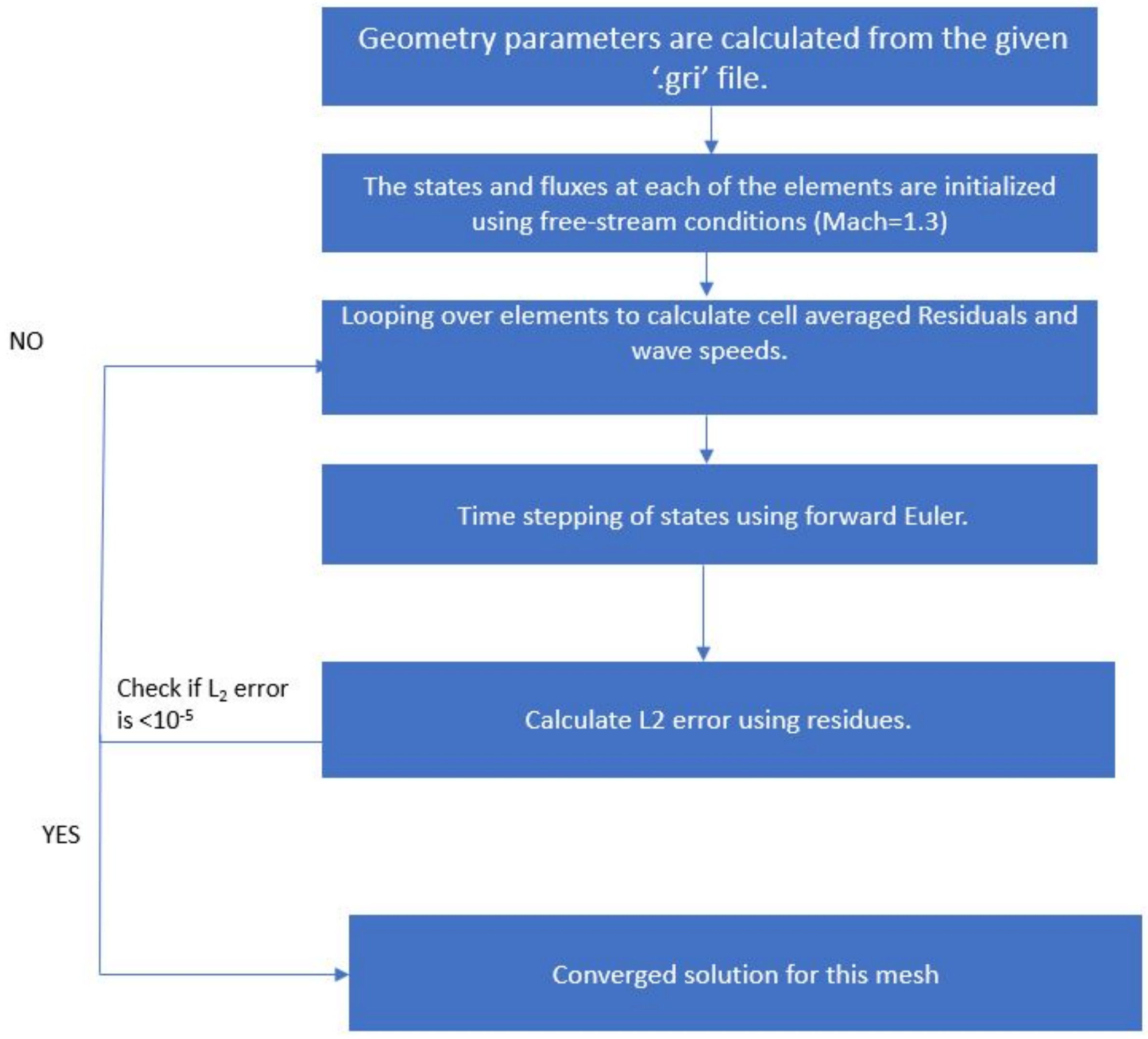

Geometry parameters are calculated from the given '.gri' file.
The states and fluxes at each of the elements are initialized using free-stream conditions (Mach=1.3)
Looping over elements to calculate cell averaged Residuals and wave speeds.
Time stepping of states using forward Euler.
Calculate L2 error using residues.
Check if L_2 error is $<10^{-5}$
NO
YES
Converged solution for this mesh

Figure 2.1: Code Structure

2.2.2 Data Structures

E: A 3-column row array which holds the connectivity information of the mesh, i.e each row of the matrix specifies the nodes comprising an element of the mesh. The row number of an element in this array corresponds to the global element number. The nodes in each row of the matrix are arranges in a counter-clockwise sense.

V : A 2-column row array which holds the X and Y co-ordinates of each node and the row number corresponds to the global node number of a particular node.

globaledgenum : A 2-column row array that references the global node numbers with each edge of the mesh. The row number of an edge in this array corresponds to the global number given to each edge which is used throughout the program.

edgevselement : A 2-column row array which references the global element numbers with the global edge numbers, i.e the ith row of this array holds the global element numbers that are on left and right sides of the ith global edge. If the the second column of any row in this array has a value of 0, the element lies on the boundary.

elemnormal : A 6-column row array which holds the normal vectors of each edge of an element. The first and second columns of the array hold the X and Y components of the outward normal to the local edge-1(edge between node-1 and node-2 of the element) of the element respectively, the third and fourth columns of the array hold the X ad Y components of the local edge-2 of the element and so on.

boundaryin : A 2-column row array which holds the global node numbers comprising each edge on the inlet boundaries(left , top and bottom) of the mesh.

boundaryout : A 2-column row array which holds the global node numbers comprising each edge on the outlet boundary(right) of the mesh.

boundarycapsule : A 2-column row array which holds the global node numbers comprising each edge on the capsule boundary of the mesh.

U : A 4-column row array which holds the averaged states at each cell.

Fx : A 4-column row array which holds the averaged X-component of flux at each cell.

Fy : A 4-column row array which holds the averaged Y-component of flux at each cell.

S : A row matrix that holds the value of wave-speed at each element.

Residulal : A row matrix that holds the value of averages residuals at each element calculated using the HLLE numerical flux.

L2 : A variable which holds the value of the L2 error norm at a particular time step in the iteration.

Drag : A variable which holds the value of the Drag force acting on the capsule.

Lift : A variable which holds the value of the Lift force acting on the capsule.

Cl : A variable which holds the value of the Lift co-efficient.

Cd : A variable which holds the value of the Drag co-efficient.

Cm : A variable which holds the value of the Moment co-efficient.

Cp : A row array which holds the value of the pressure co-efficient of each element on the heat shield of the capsule.

theta : A row array which hold the angle of arc corresponding to each element on the heat shield of the capsule.

2.3 Task : 3

Results and plots are in Later pages.

2.4 Task : 4

2.4.1 Code Structure

The functions used to execute the following task were the same as the ones used for the previous task, in addition ,the following function was used :

adaptation() : This function adds adds new elements to the existing mesh by splitting elements whose edges have a 'Mach Jump error' greater than the threshold value. The working of this function is illustrated by 'Algorithm 20-27'.

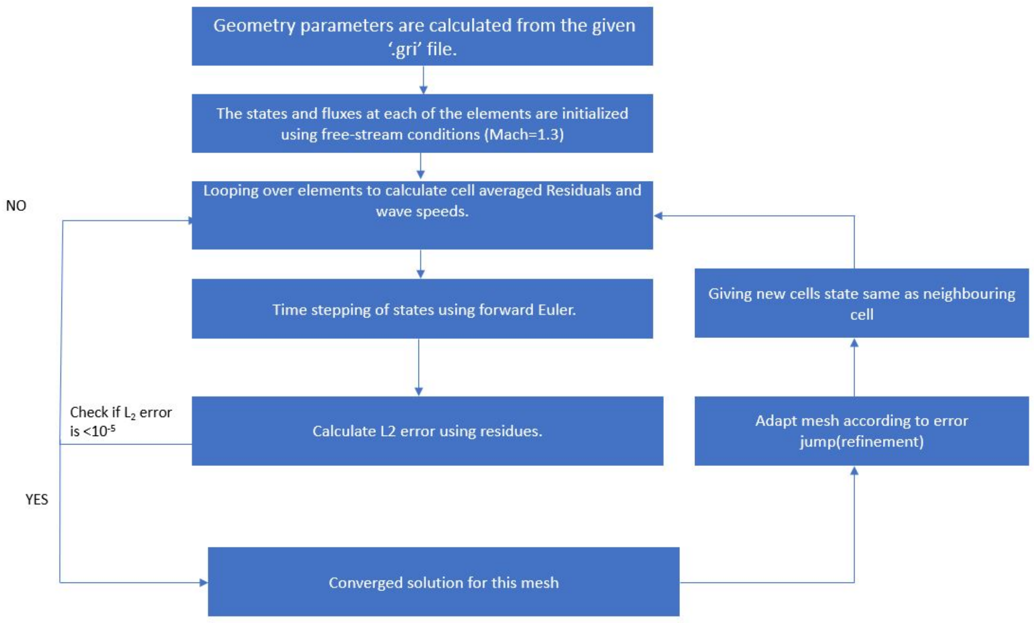

Geometry parameters are calculated from the given '.gri' file.
The states and fluxes at each of the elements are initialized using free-stream conditions (Mach=1.3)
Looping over elements to calculate cell averaged Residuals and wave speeds.
Time stepping of states using forward Euler.
Calculate L2 error using residues.
Check if L_2 error is $<10^{-5}$
NO
YES
Converged solution for this mesh
Adapt mesh according to error jump(refinement)
Giving new cells state same as neighbouring cell

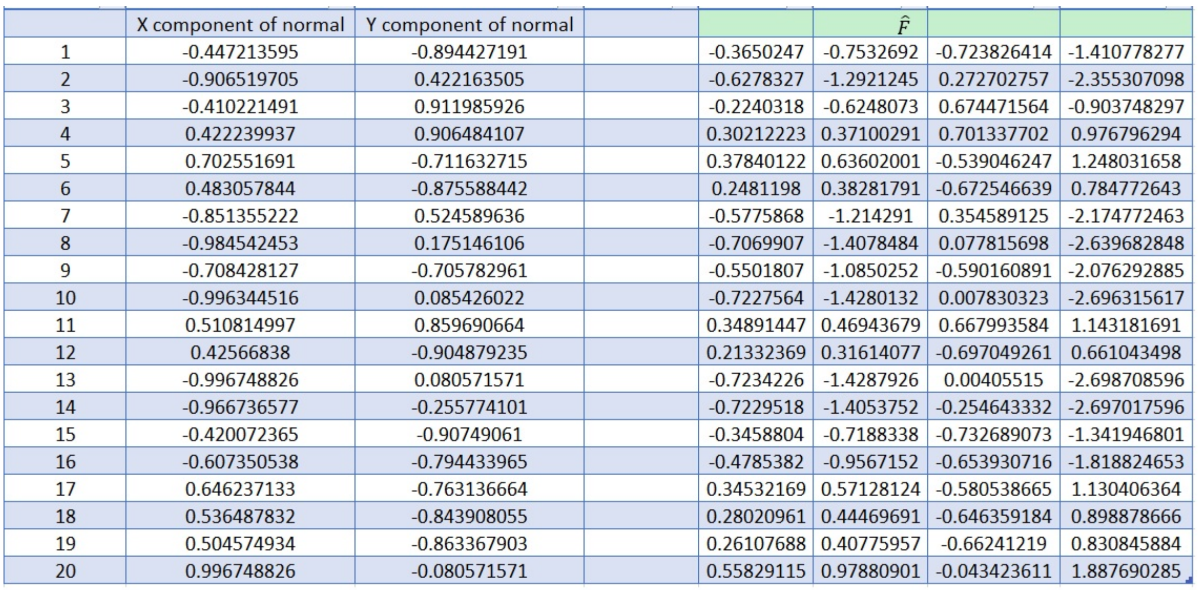
Figure 2.2: Code Structure for adaptation

Chapter 3

Results

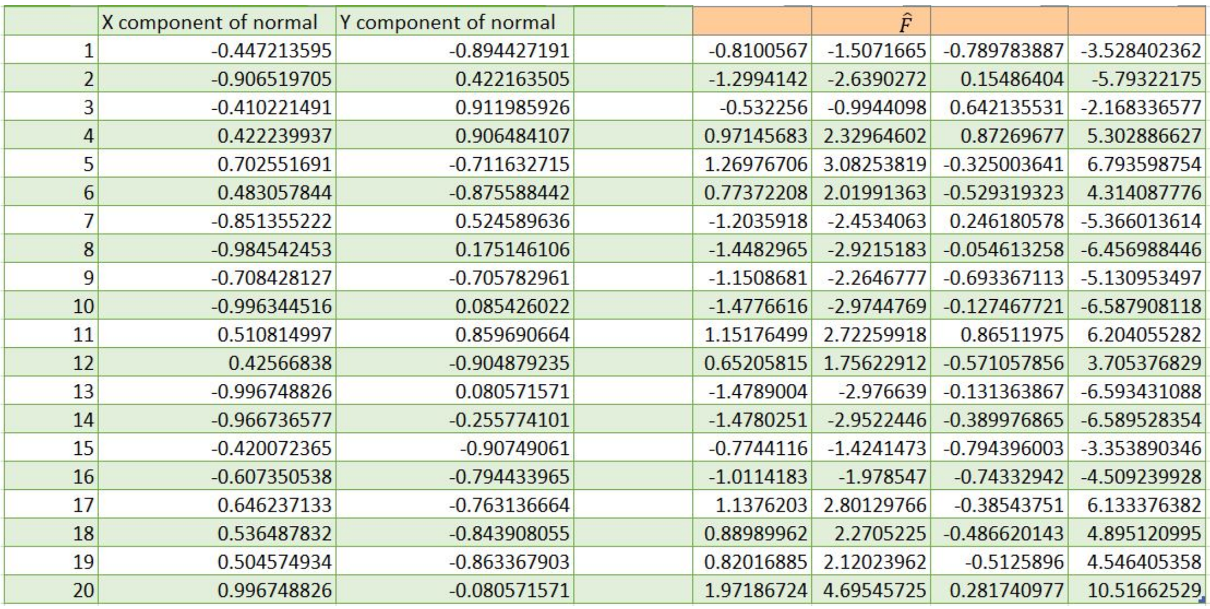
3.1 Task:1

The HLLE flux was tested for M_L=2 and M_R=1.5 and the results are tabulated below.



	X component of normal	Y component of normal			$\hat{F}$		
1	-0.447213595	-0.894427191		-0.3650247	-0.7532692	-0.723826414	-1.410778277
2	-0.906519705	0.422163505		-0.6278327	-1.2921245	0.272702757	-2.355307098
3	-0.410221491	0.911985926		-0.2240318	-0.6248073	0.674471564	-0.903748297
4	0.422239937	0.906484107		0.30212223	0.37100291	0.701337702	0.976796294
5	0.702551691	-0.711632715		0.37840122	0.63602001	-0.539046247	1.248031658
6	0.483057844	-0.875588442		0.2481198	0.38281791	-0.672546639	0.784772643
7	-0.851355222	0.524589636		-0.5775868	-1.214291	0.354589125	-2.174772463
8	-0.984542453	0.175146106		-0.7069907	-1.4078484	0.077815698	-2.639682848
9	-0.708428127	-0.705782961		-0.5501807	-1.0850252	-0.590160891	-2.076292885
10	-0.996344516	0.085426022		-0.7227564	-1.4280132	0.007830323	-2.696315617
11	0.510814997	0.859690664		0.34891447	0.46943679	0.667993584	1.143181691
12	0.42566838	-0.904879235		0.21332369	0.31614077	-0.697049261	0.661043498
13	-0.996748826	0.080571571		-0.7234226	-1.4287926	0.00405515	-2.698708596
14	-0.966736577	-0.255774101		-0.7229518	-1.4053752	-0.254643332	-2.697017596
15	-0.420072365	-0.90749061		-0.3458804	-0.7188338	-0.732689073	-1.341946801
16	-0.607350538	-0.794433965		-0.4785382	-0.9567152	-0.653930716	-1.818824653
17	0.646237133	-0.763136664		0.34532169	0.57128124	-0.580538665	1.130406364
18	0.536487832	-0.843908055		0.28020961	0.44469691	-0.646359184	0.898878666
19	0.504574934	-0.863367903		0.26107688	0.40775957	-0.66241219	0.830845884
20	0.996748826	-0.080571571		0.55829115	0.97880901	-0.043423611	1.887690285

Figure 3.1: Flux calculation for M_L=0.5 and M_R=0.8



	X component of normal	Y component of normal			$\hat{F}$		
1	-0.447213595	-0.894427191		-0.8100567	-1.5071665	-0.789783887	-3.528402362
2	-0.906519705	0.422163505		-1.2994142	-2.6390272	0.15486404	-5.79322175
3	-0.410221491	0.911985926		-0.532256	-0.9944098	0.642135531	-2.168336577
4	0.422239937	0.906484107		0.97145683	2.32964602	0.87269677	5.302886627
5	0.702551691	-0.711632715		1.26976706	3.08253819	-0.325003641	6.793598754
6	0.483057844	-0.875588442		0.77372208	2.01991363	-0.529319323	4.314087776
7	-0.851355222	0.524589636		-1.2035918	-2.4534063	0.246180578	-5.366013614
8	-0.984542453	0.175146106		-1.4482965	-2.9215183	-0.054613258	-6.456988446
9	-0.708428127	-0.705782961		-1.1508681	-2.2646777	-0.693367113	-5.130953497
10	-0.996344516	0.085426022		-1.4776616	-2.9744769	-0.127467721	-6.587908118
11	0.510814997	0.859690664		1.15176499	2.72259918	0.86511975	6.204055282
12	0.42566838	-0.904879235		0.65205815	1.75622912	-0.571057856	3.705376829
13	-0.996748826	0.080571571		-1.4789004	-2.976639	-0.131363867	-6.593431088
14	-0.966736577	-0.255774101		-1.4780251	-2.9522446	-0.389976865	-6.589528354
15	-0.420072365	-0.90749061		-0.7744116	-1.4241473	-0.794396003	-3.353890346
16	-0.607350538	-0.794433965		-1.0114183	-1.978547	-0.74332942	-4.509239928
17	0.646237133	-0.763136664		1.1376203	2.80129766	-0.38543751	6.133376382
18	0.536487832	-0.843908055		0.88989962	2.2705225	-0.486620143	4.895120995
19	0.504574934	-0.863367903		0.82016885	2.12023962	-0.5125896	4.546405358
20	0.996748826	-0.080571571		1.97186724	4.69545725	0.281740977	10.51662529

Figure 3.2: Flux calculation for M_L=2 and M_R=1.5

3.2 Task:2

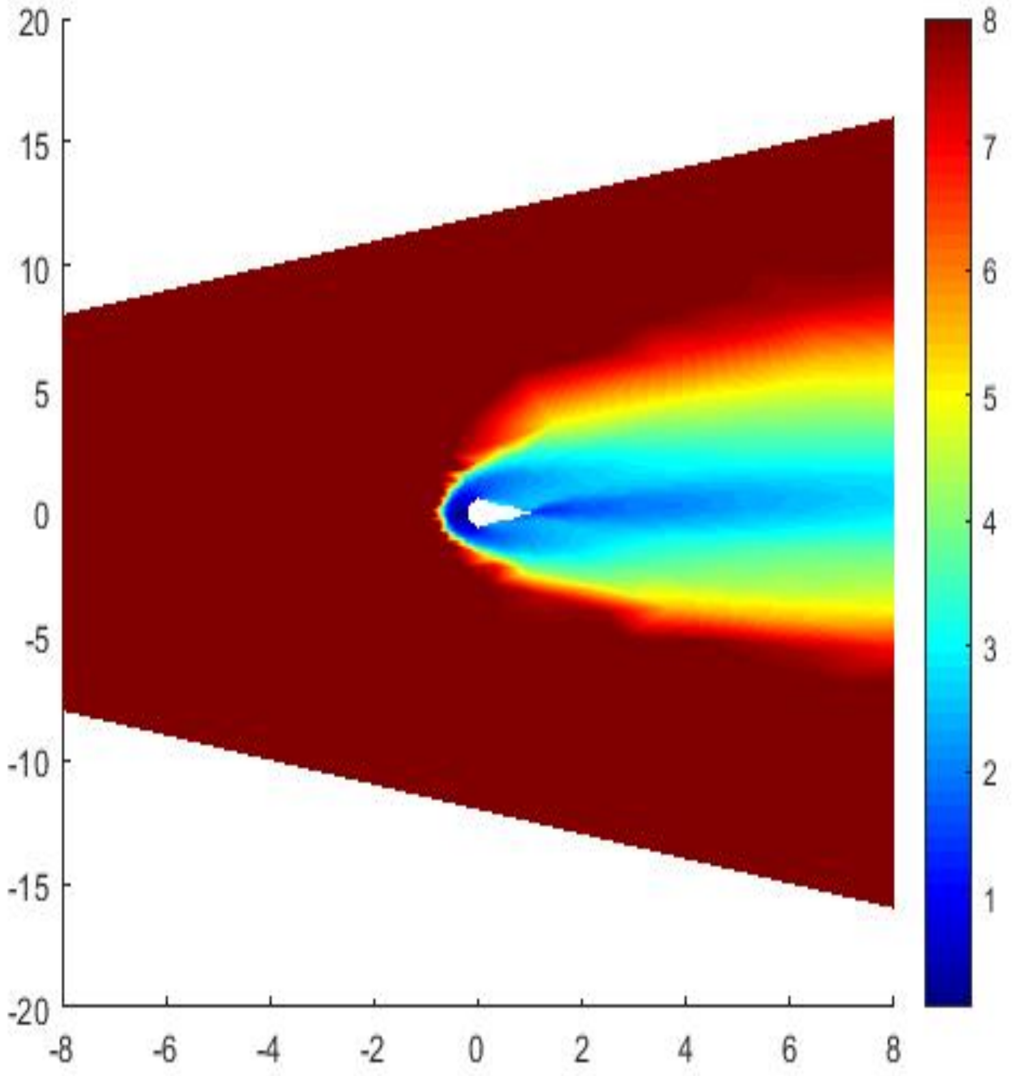


Figure 3.3: Far-field Mach Contour

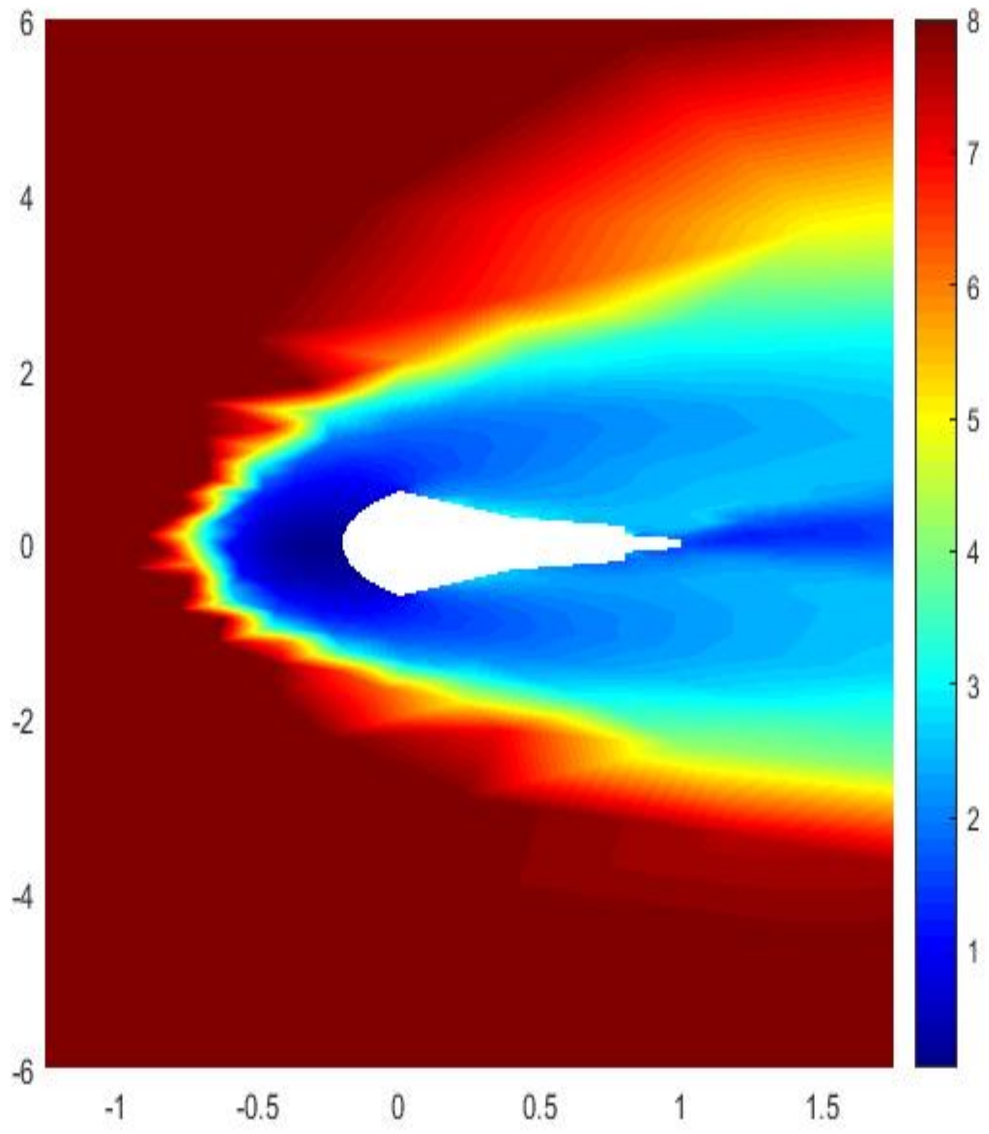


Figure 3.4: Zoomed in Mach Contour

Plotted above are the far-field Mach contour and the zoomed in contour. The discontinuity is clearly visible in the two. The Mach number of the flow near the capsule reduces.

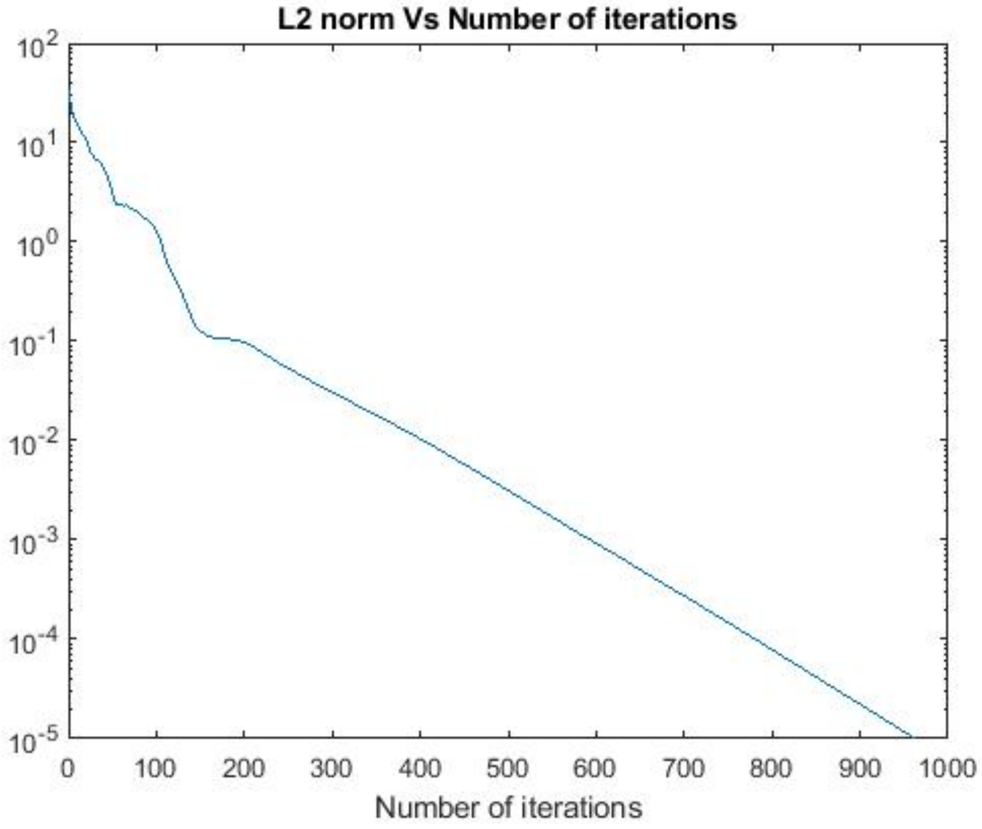


Figure 3.5: L_2 norm v/s Number of Iterations

We can clearly see that the L_2 norm reduces with number of iterations.

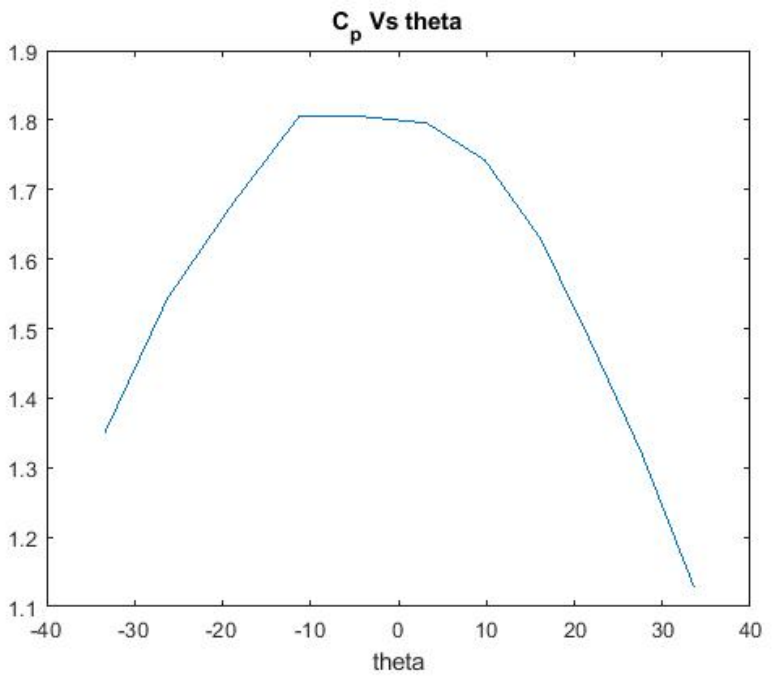


Figure 3.6: C_p v/s θ

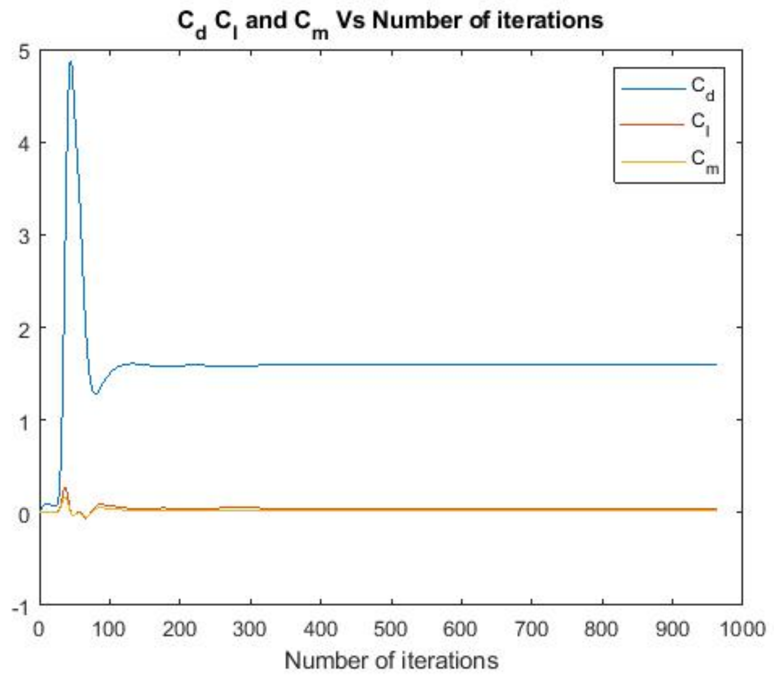


Figure 3.7: C_d, C_l, C_m v/s Number of Iterations

Above is the plot of C_d, C_l, C_m v/s number of iterations. It is clear that these value assume a constant value after a specific number of iterations.

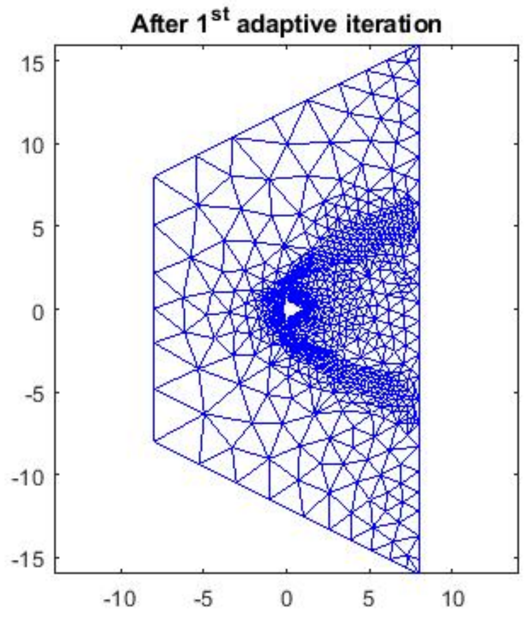


Figure 3.8: Mesh in entirety after 1^{st} adaption
$\alpha = 0$

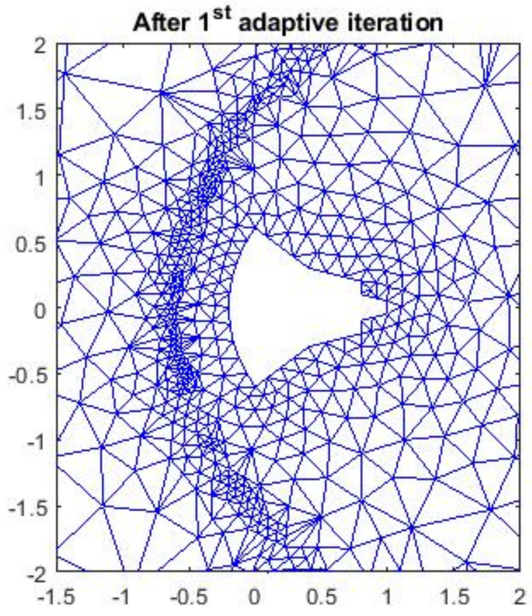


Figure 3.9: Zoomed in mesh after 1^{st} adaption
$\alpha = 0$

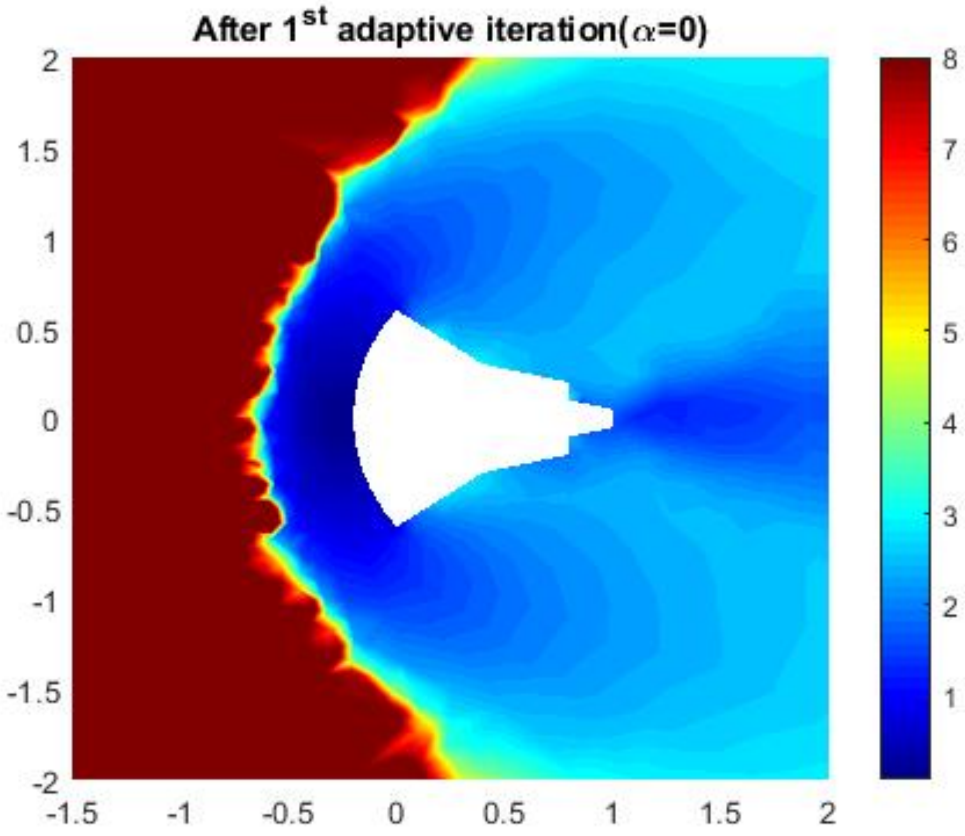


Figure 3.10: Zoomed in mach contour after 1^{st} adaptive iteration

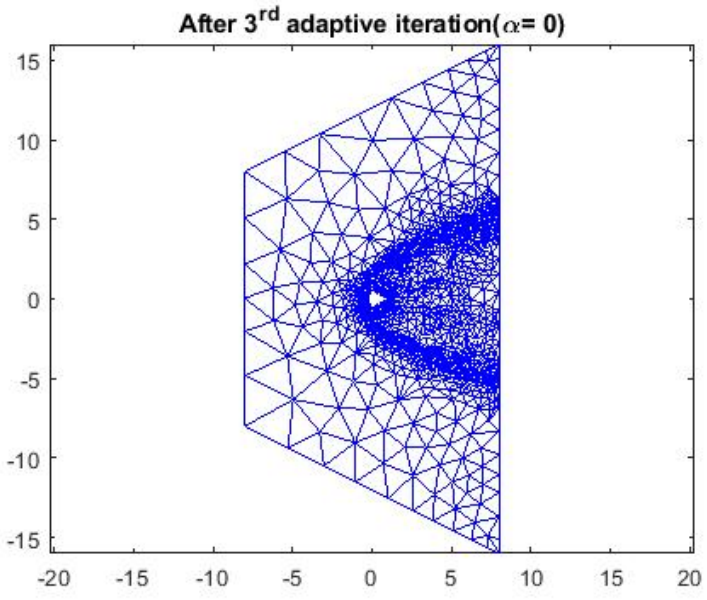


Figure 3.11: Mesh in entirety after 3^{rd} adaptive iteration

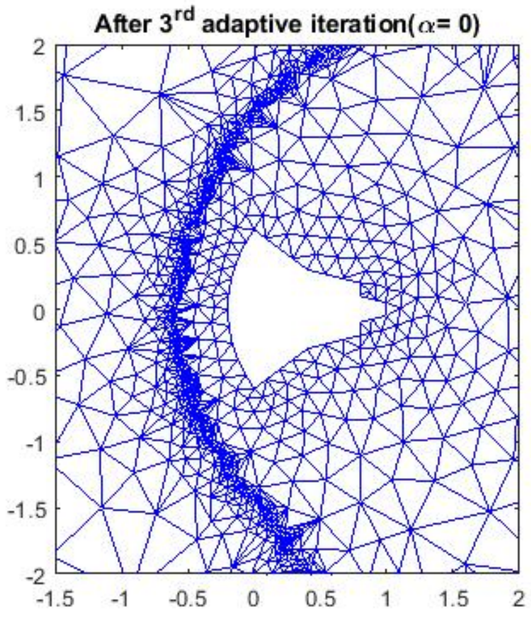


Figure 3.12: Zoomed in mesh after 3^{rd} adaptive iteration

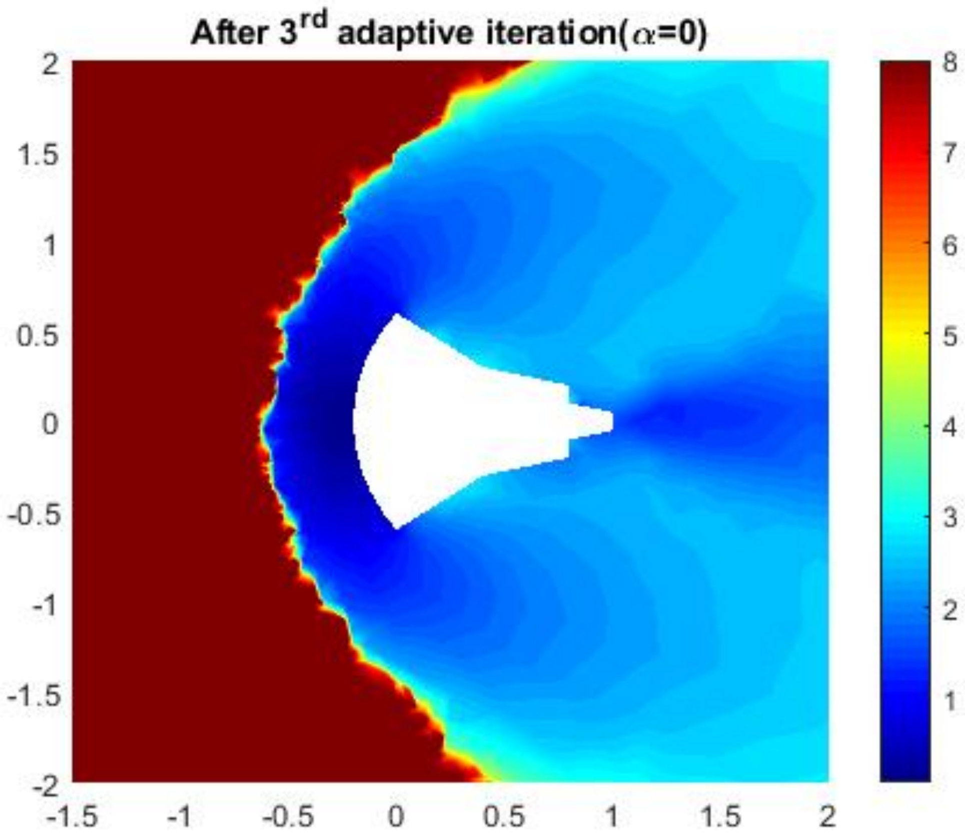


Figure 3.13: Zoomed in mach contour after 3^{rd} adaptive iteration

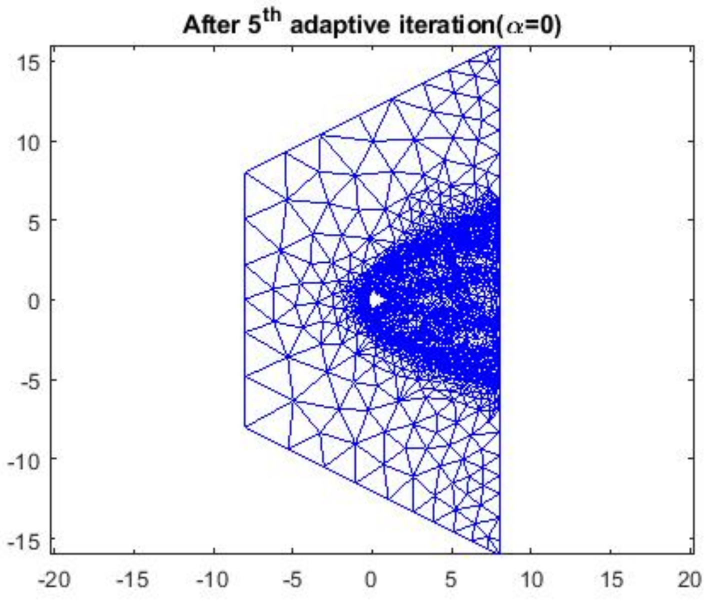


Figure 3.14: Mesh in entirety after 5^{th} adaptive iteration

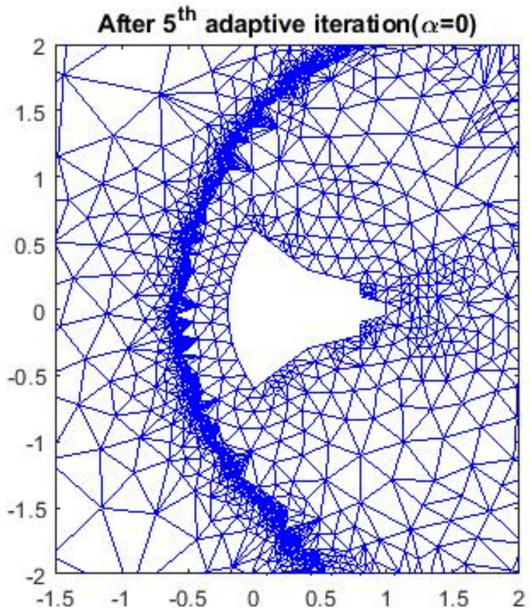


Figure 3.15: Zoomed in mesh after 5^{th} adaptive iteration

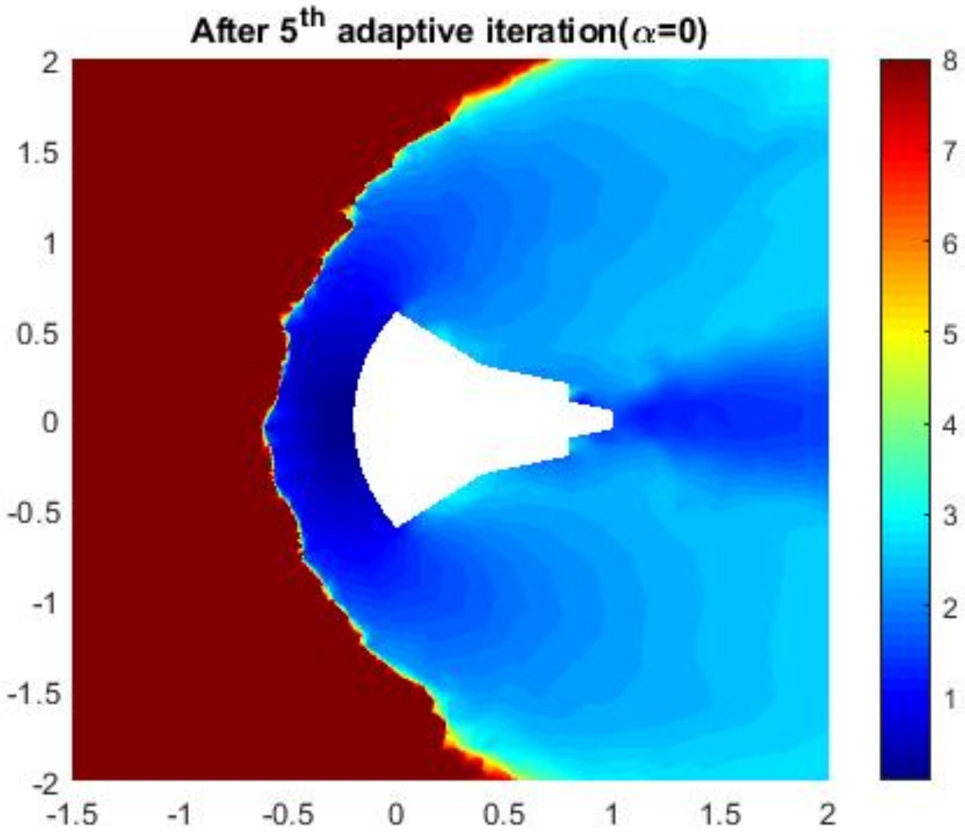


Figure 3.16: Zoomed in mach contour after 5^{th} adaptive iteration

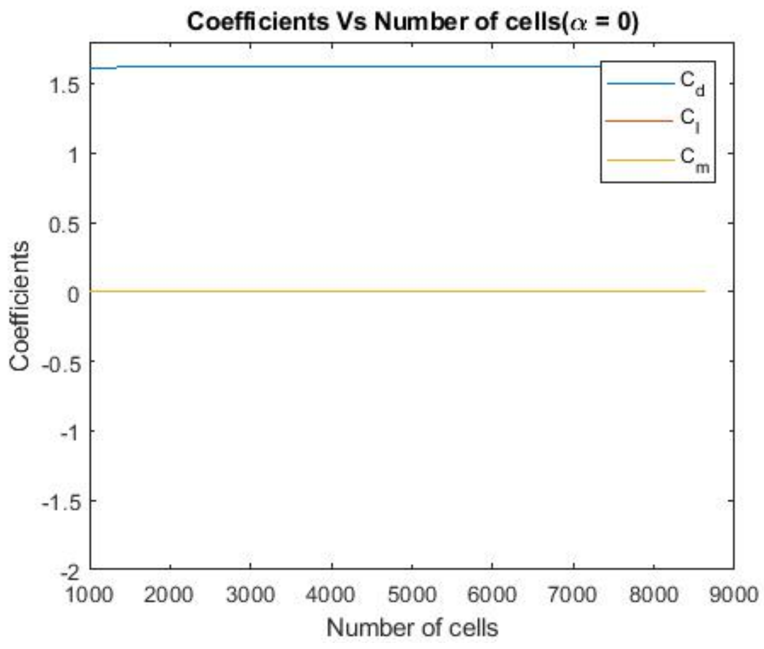


Figure 3.17: C_d, C_l, C_m v/s number of cells

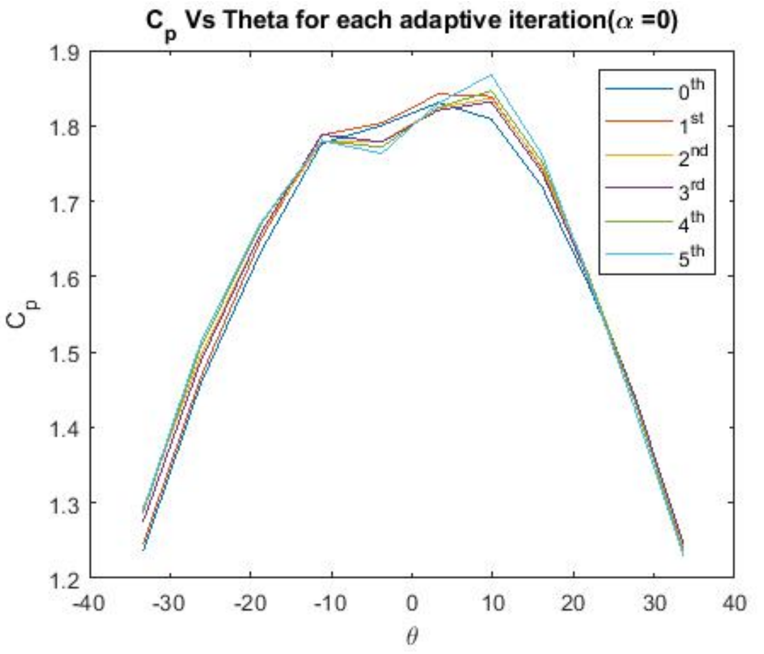


Figure 3.18: C_p v/s θ for α=0

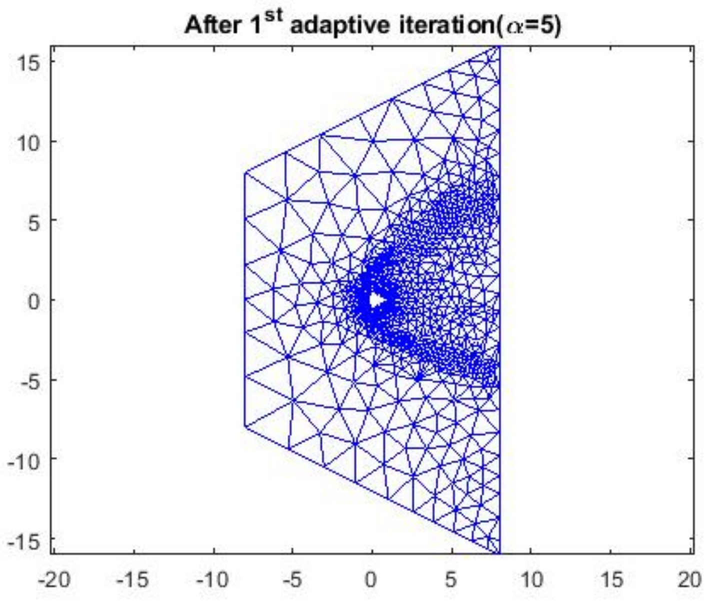


Figure 3.19: Mesh in entirety after 1^{st} adaptive iteration

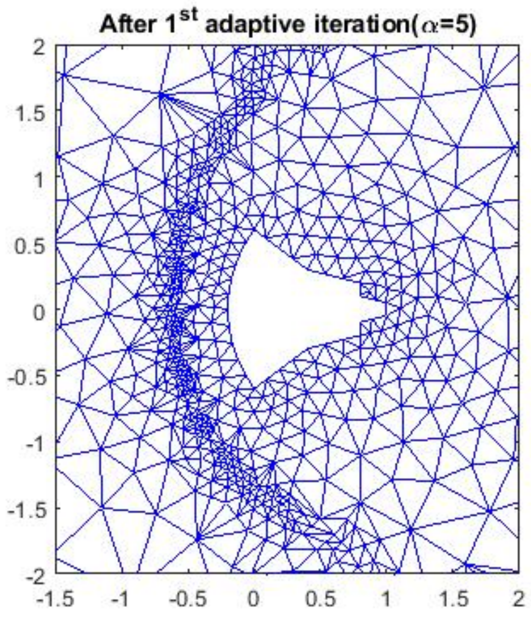


Figure 3.20: Zoomed in mesh after 1^{st} adaptive iteration

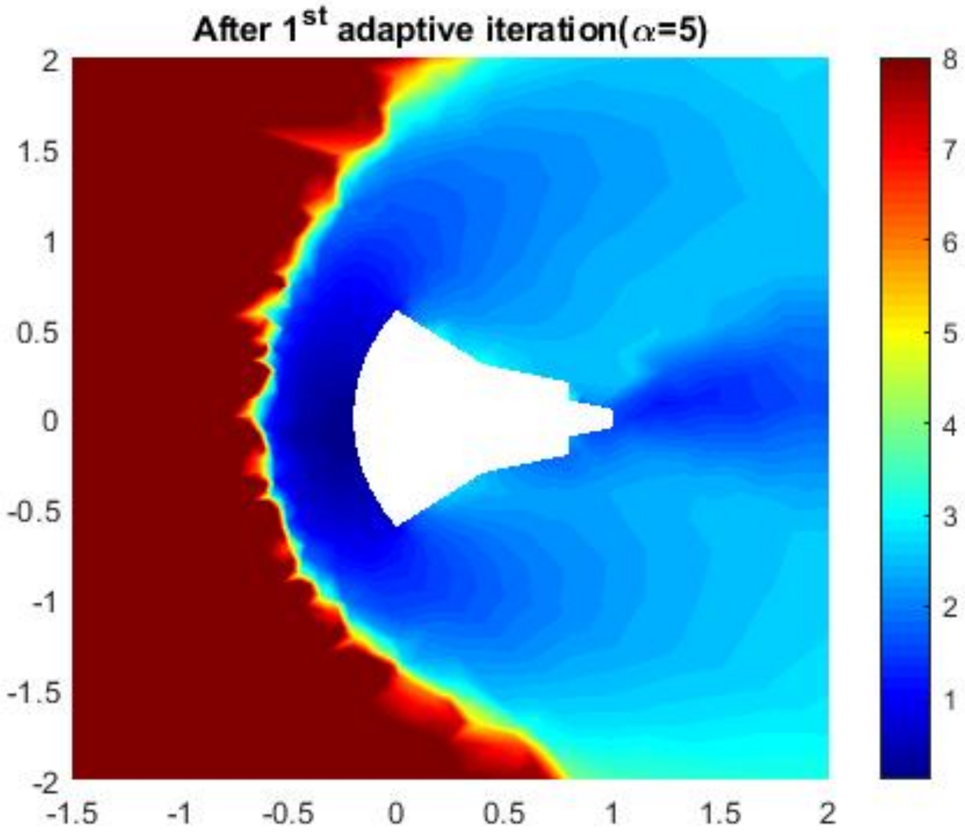


Figure 3.21: Zoomed in mach contour after 1^{st} adaptive iteration

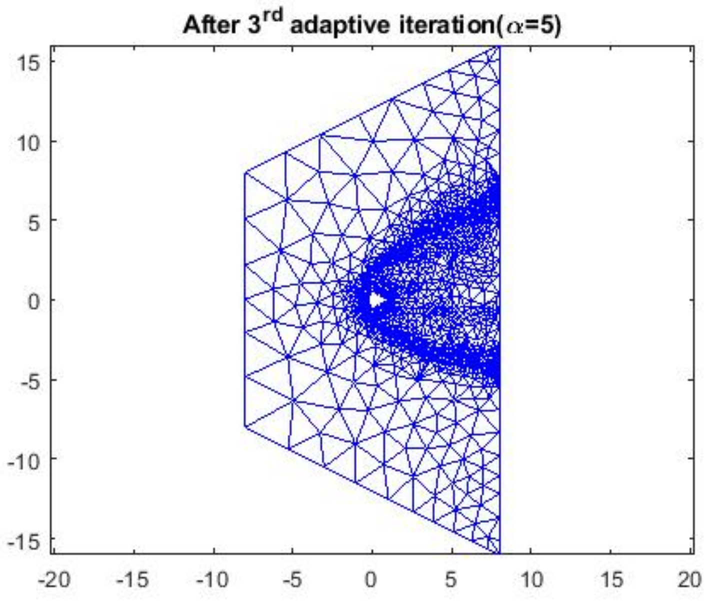


Figure 3.22: Mesh in entirety after 3^{rd} adaptive iteration

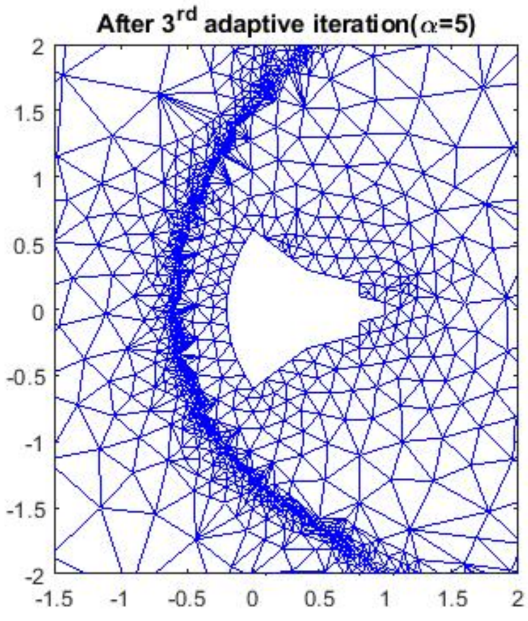


Figure 3.23: Zoomed in mesh after 3^{rd} adaptive iteration

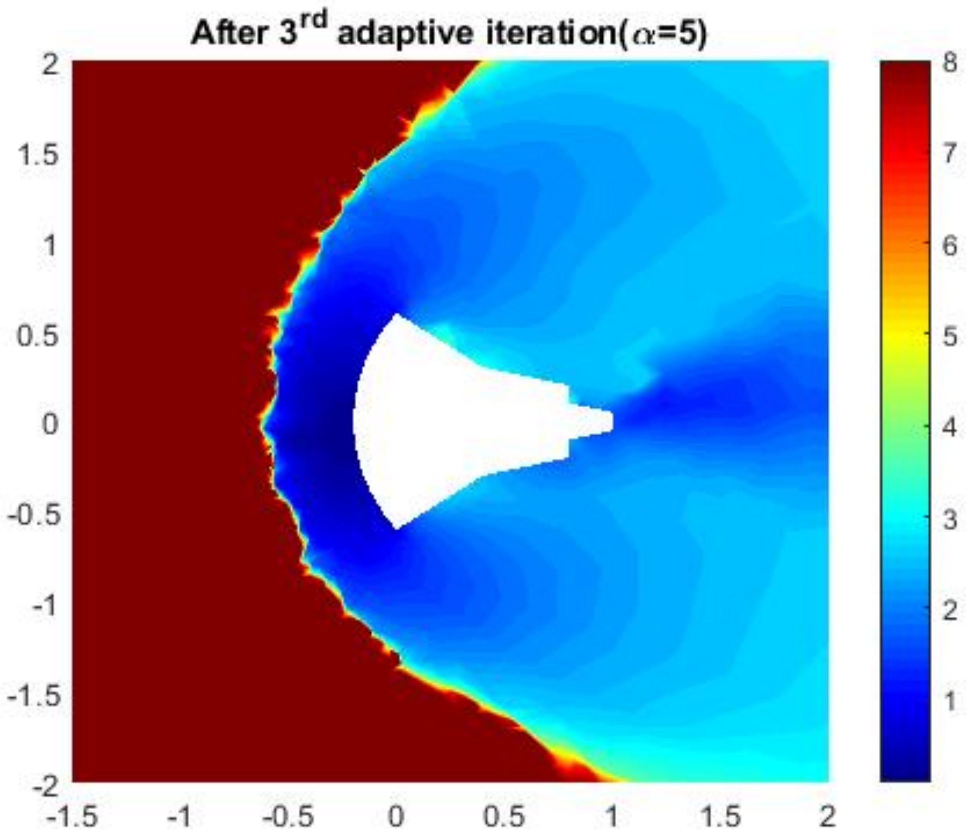


Figure 3.24: Zoomed in mach contour after 3^{rd} adaptive iteration

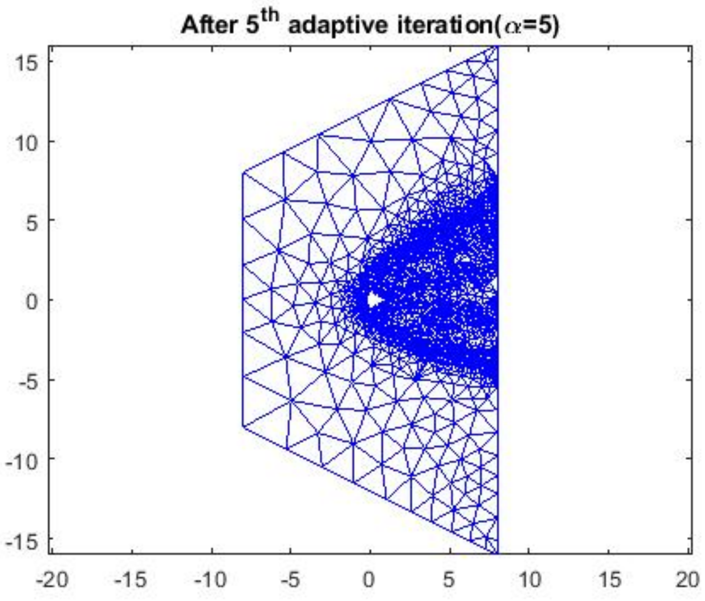


Figure 3.25: Mesh in entirety after $5th^{th}$ adaptive iteration

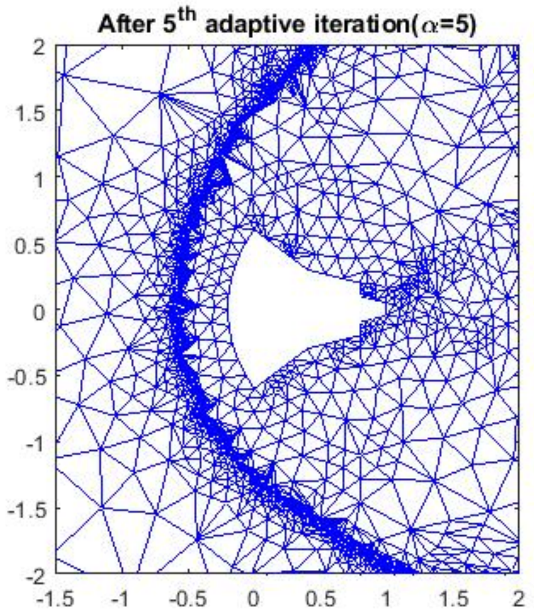


Figure 3.26: Zoomed in mesh after 5^{th} adaptive iteration

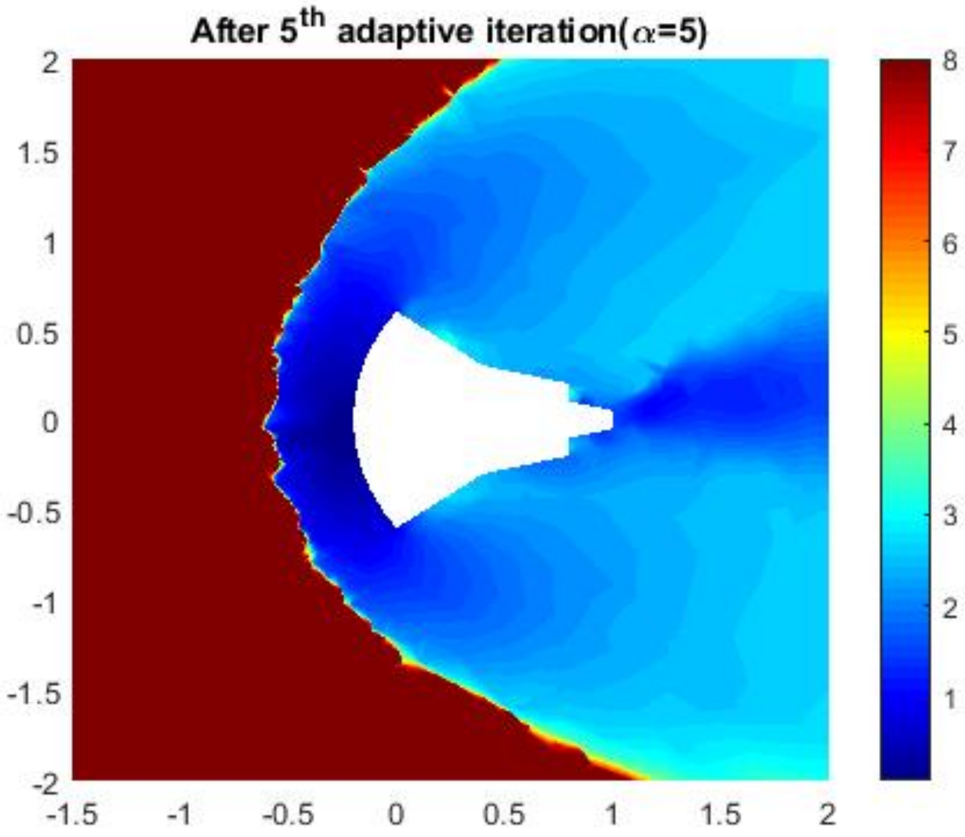


Figure 3.27: Zoomed in mach contour after 5th adaptive iteration

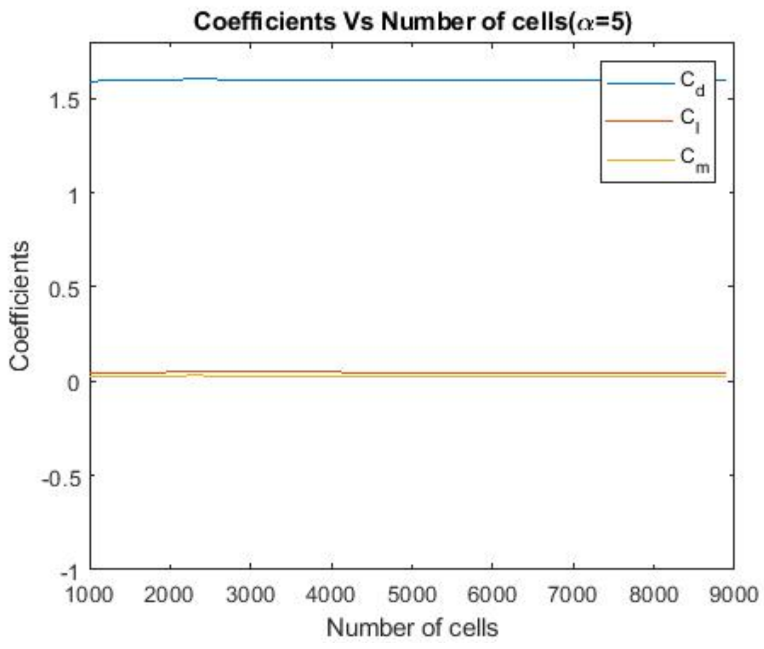


Figure 3.28: C_d, C_l, C_m v/s number of cells

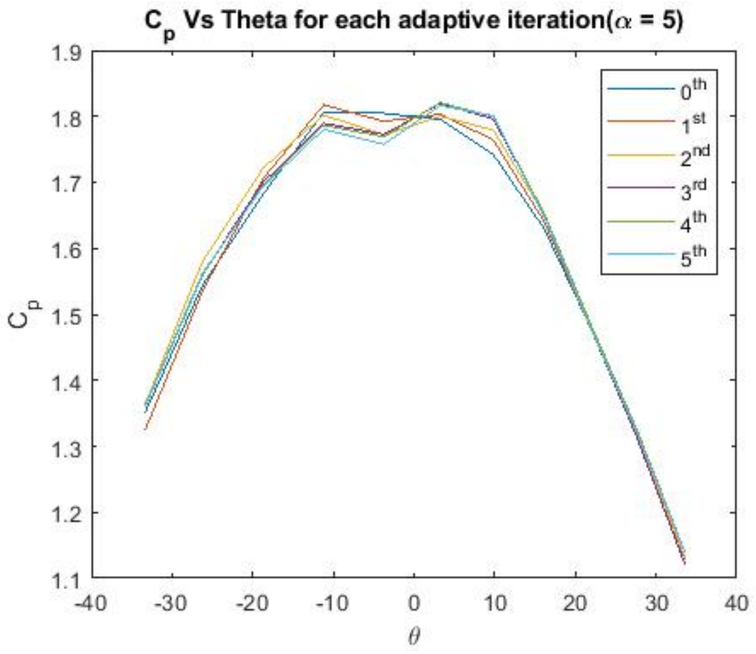


Figure 3.29: C_p v/s θ for α=5

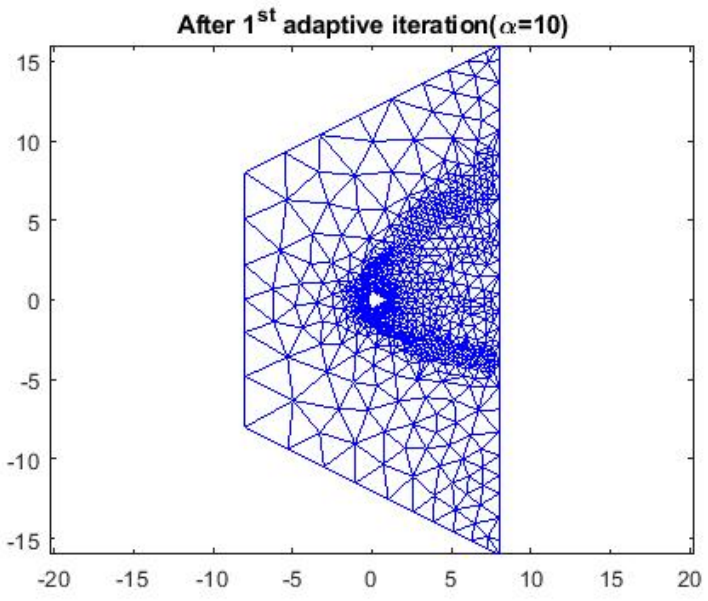


Figure 3.30: Mesh in entirety after 1st adaptive iteration

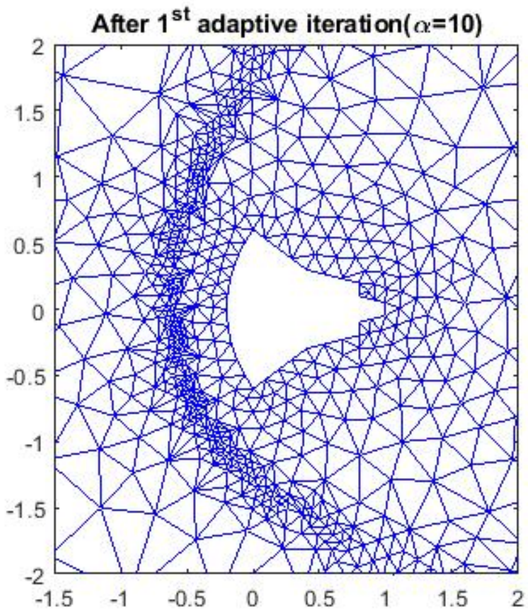


Figure 3.31: Zoomed in mesh after 1st adaptive iteration

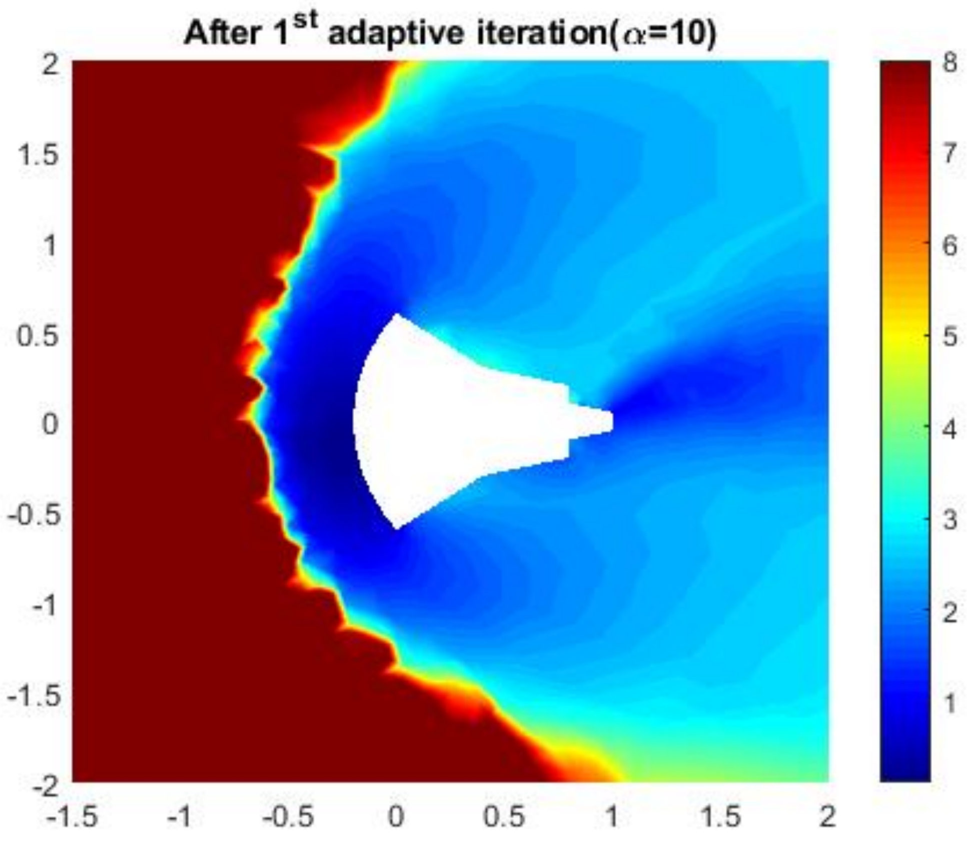


Figure 3.32: Zoomed in mach contour after 1^{st} adaptive iteration

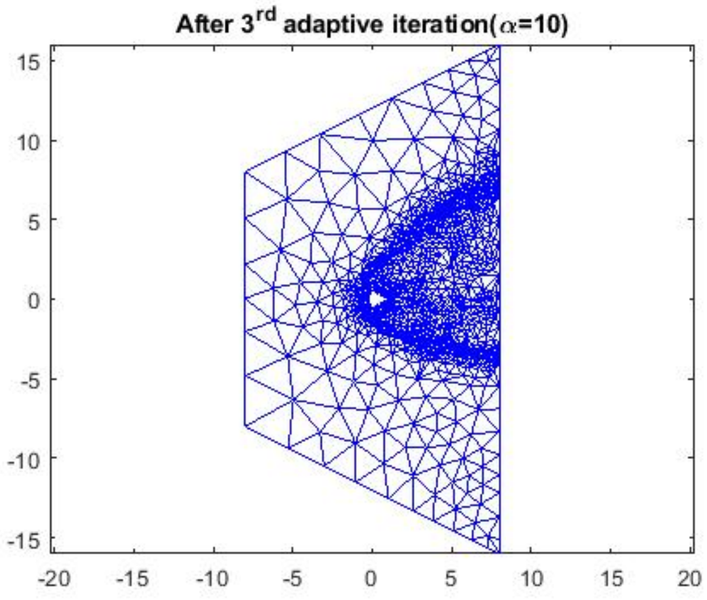


Figure 3.33: Mesh in entirety after 3^{rd} adaptive iteration

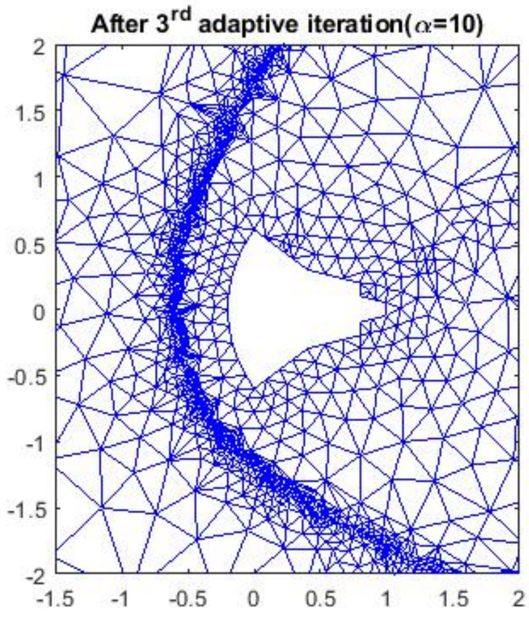


Figure 3.34: Zoomed in mesh after 3^{rd} adaptive iteration

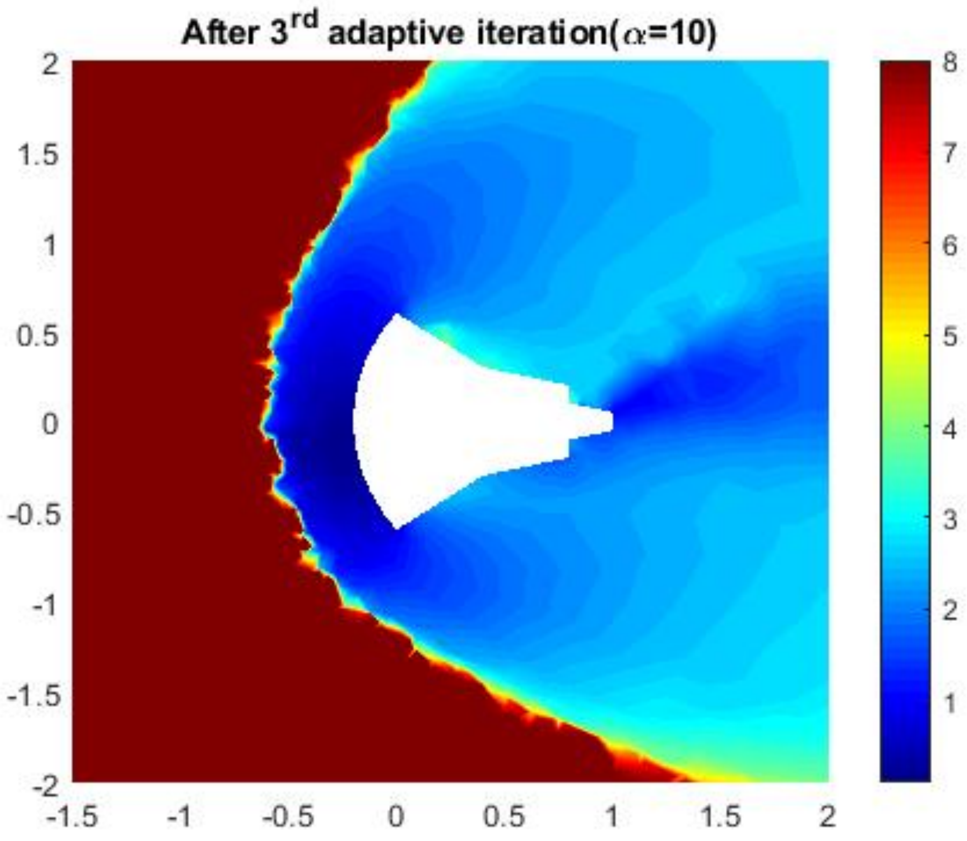


Figure 3.35: Zoomed in mach contour after 3^{rd} adaptive iteration

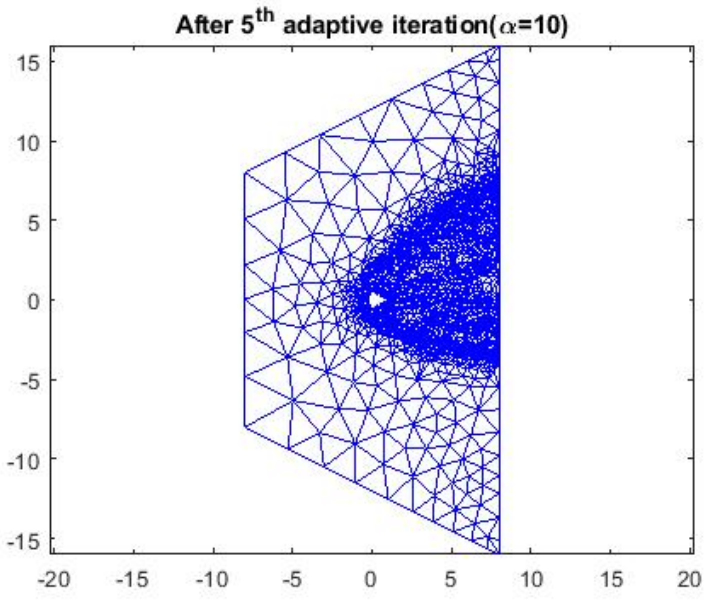


Figure 3.36: Mesh in entirety after 5^{th} adaptive iteration

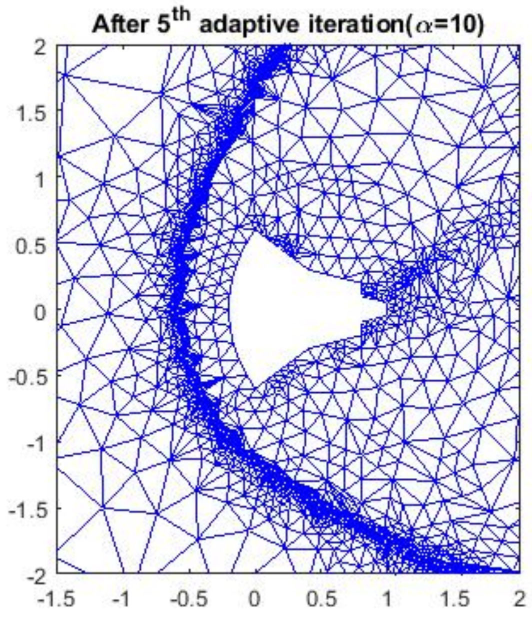


Figure 3.37: Zoomed in mesh after 5^{th} adaptive iteration

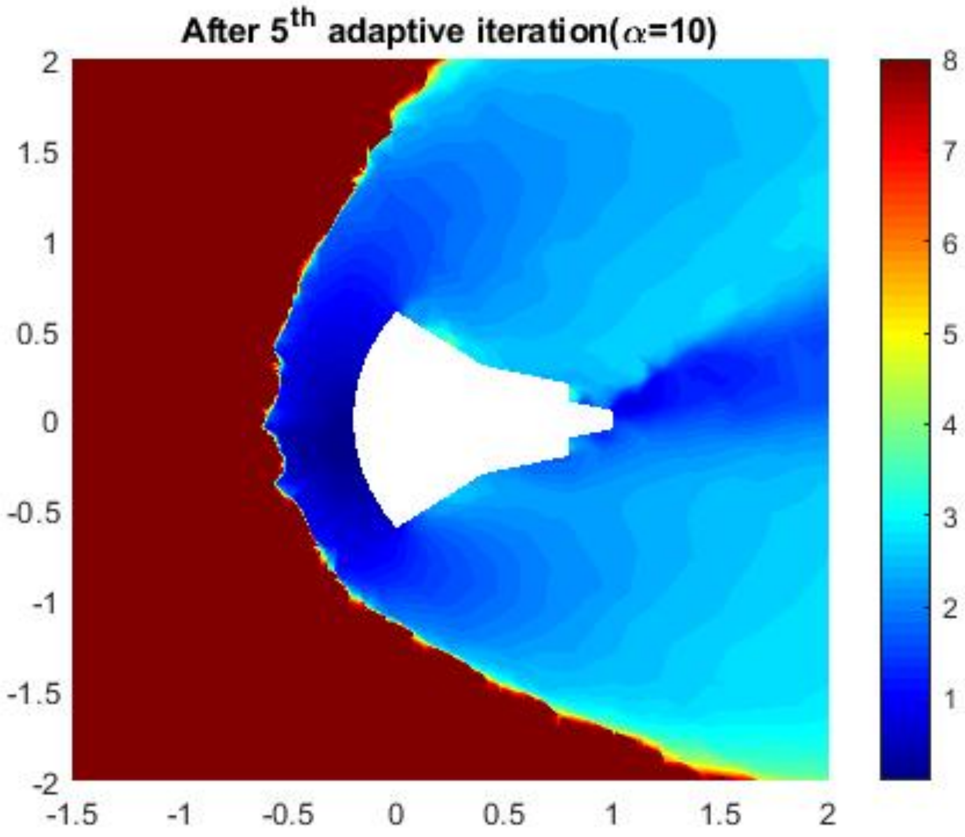


Figure 3.38: Zoomed in mach contour after 5^{th} adaptive iteration

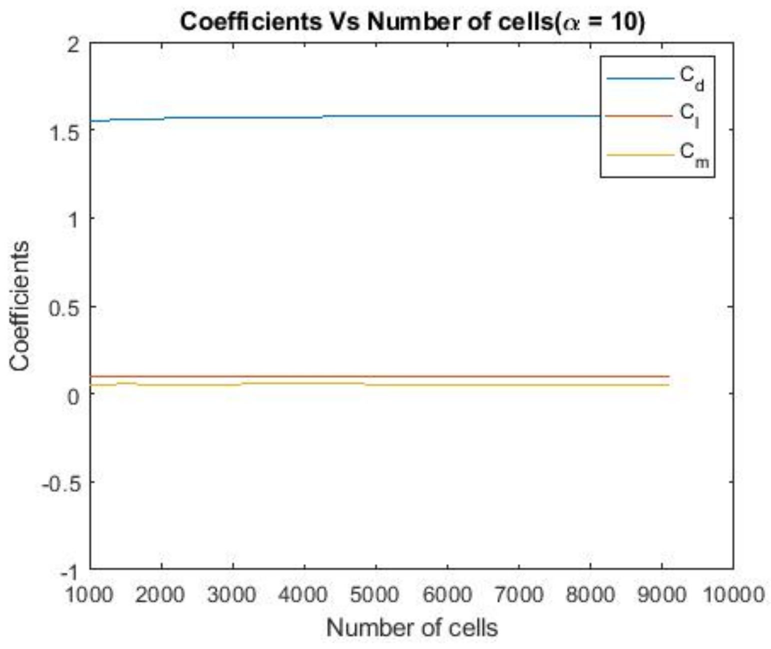


Figure 3.39: C_d, C_l, C_m v/s number of cells

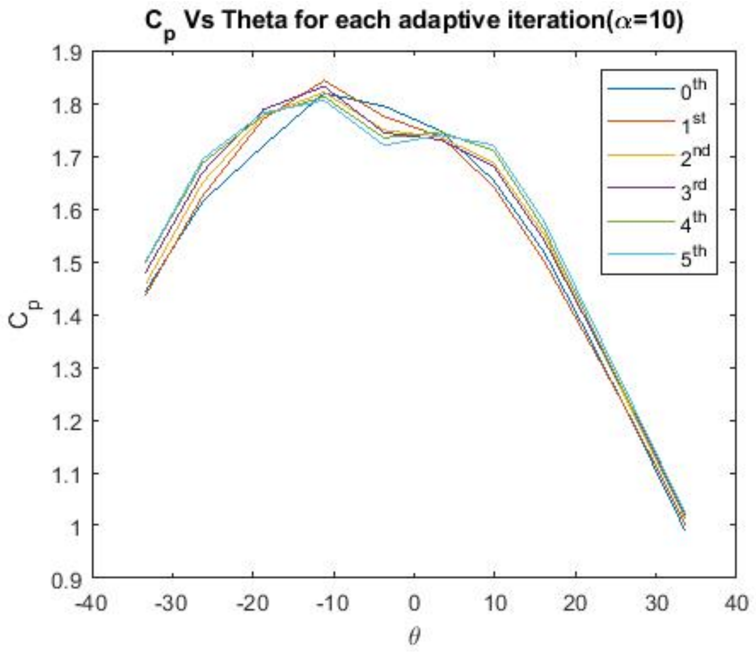


Figure 3.40: C_p v/s θ for α=10

Chapter 4

Conclusion

1)With an increasing number of iterations the L2 error decreases.

2)The C_p peaks at θ=0 , i.e towards the centre of the heat shield.

3)As we run more adaptive iterations, the mesh becomes finer in the region of discontinuity and the region of Mach discontinuity becomes thinner.

4)Value of C_d is maximum at α=0° and reduces as α increases.

5)The value of C_l and C_m increases with the increase in α.