

A Numerical Solution to the Convergent-Divergent Nozzle Using the McCormack Method

Sidharth Tadeparti
Indian Institute of Technology, Madras

June 14, 2020

Abstract

The problem of the Convergent-Divergent Nozzle is treated as a quasi-one dimensional flow problem. While a closed form solution exists for the problem, the aim will be to generate a sufficiently accurate numerical solution of a suitable form of the Navier-Stokes equations using a finite difference technique.

1 Problem Description

We consider an isentropic flow through a converging-diverging nozzle. The flow originates from a reservoir of large area A which results in a small value of velocity V , hence the pressure P_0 and temperature T_0 at the inlet are the stagnation values, the velocity here is in the subsonic region. The flow passes through the throat, where the flow is choked at the point of minimum cross-sectional area A^* , having pressure P^* and temperature T^* , the flow here is sonic.

The flow now proceeds to the divergent part of the nozzle where the velocity of the flow is supersonic causing a shock. The flow exits with parameters, P_e , T_e , ρ_e . The variation of the parameters with respect to space, at steady state is to be determined.

[Anderson \[1995\]](#) [Anderson \[1982\]](#)

2 Analytical Solution

The Analytical Solution is obtained using the Navier-Stokes equation, the equation of state and relation between enthalpy and temperature for a perfect gas. The area mach number relation (Equation 1) is used to determine the relation between flow speed and cross sectional area. [Anderson \[1982\]](#)

$$\left(\frac{A}{A^*}\right)^2 = \frac{1}{M^2} \left[\frac{2}{\gamma+1} \left(1 + \frac{\gamma-1}{2} M^2 \right) \right]^{\frac{\gamma+1}{\gamma-1}} \quad (1)$$

$$\frac{P}{P_0} = \left(1 + \frac{\gamma-1}{2} M^2 \right)^{\frac{\gamma+1}{\gamma-1}} \quad (2)$$

$$\frac{\rho}{\rho_0} = \left(1 + \frac{\gamma-1}{2} M^2 \right)^{\frac{-1}{\gamma-1}} \quad (3)$$

$$\frac{T}{T_0} = \left(1 + \frac{\gamma-1}{2} M^2 \right)^{-1} \quad (4)$$

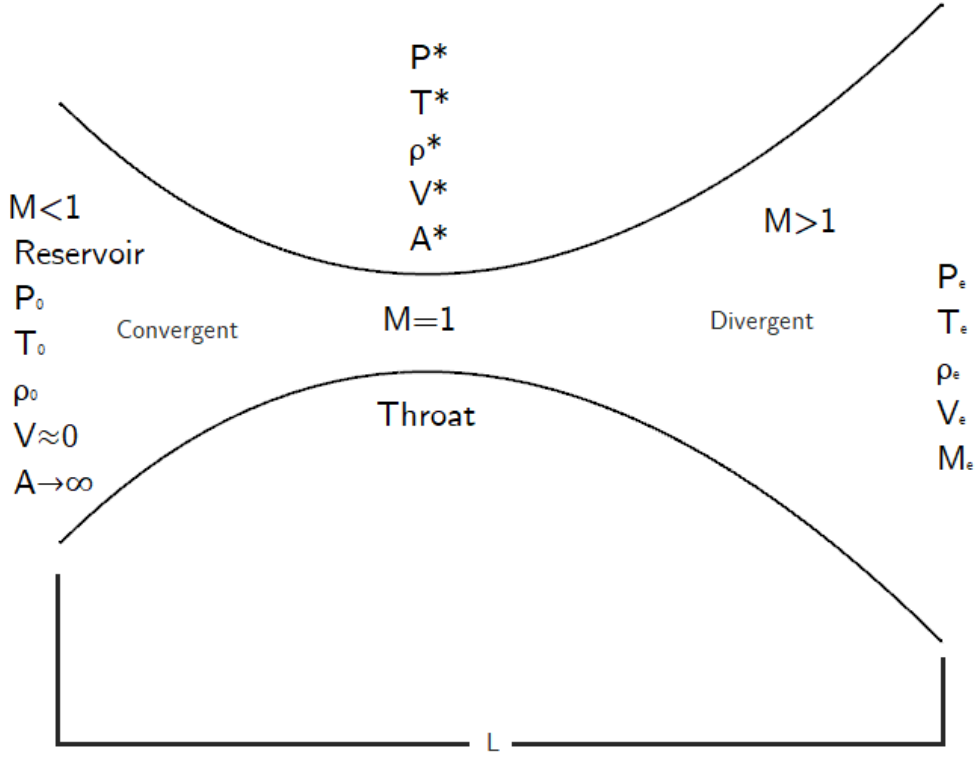


Figure 1: A Converging Diverging Nozzle.

3 Numerical Solution

The differential equations pertaining to the flow are derived using the integral form of the Navier-Stokes equations, the equation of state and the energy relation on a control volume of infinitesimal size. The obtained differential equations are further made dimensionless (Table 1).

Anderson [1995] , Joseph W. Connolly

Note: a represents the speed of sound, a_0 represents the speed of sound in the reservoir.

The Dimensionless Differential Equations:

$$\frac{\partial \rho'}{\partial t'} = -\rho' \frac{\partial V'}{\partial x'} - \rho' V' \frac{\partial \ln A'}{\partial x'} - V' \frac{\partial \rho}{\partial x'} \quad (5)$$

$$\frac{\partial V'}{\partial t'} = -V' \frac{\partial V'}{\partial x'} - \frac{1}{\gamma} \left(\frac{\partial T'}{\partial x'} + \frac{T'}{\rho'} \frac{\partial \rho'}{\partial x'} \right) \quad (6)$$

$$\frac{\partial T'}{\partial t'} = -V' \frac{\partial T'}{\partial x'} - (\gamma - 1) T' \left(\frac{\partial V'}{\partial x'} + V' \frac{\partial \ln(A')}{\partial x'} \right) \quad (7)$$

The McCormack Method: Anderson [1995]

The McCormack method is a conditionally stable explicit method to solve hyperbolic partial differential equations. It is accurate to the second order. Consider a quantity u to be approximated, i represents the spatial coordinate while t represents the time coordinate.

Table 1: Substitution for the Dimensionless Differential Equation

Quantity	Substitution
T'	$\frac{T}{T_0}$
ρ'	$\frac{\rho}{\rho_0}$
x'	$\frac{x}{L}$
a_0	$\sqrt{\gamma RT_0}$
V'	$\frac{V}{a_0}$
t'	$\frac{t}{L/a_0}$
A'	$\frac{A}{A^*}$

$\left(\frac{\partial u}{\partial t}\right)_i^t$ is calculated from its differential differential equation using a **forward difference scheme**

$$\bar{u}_i^{t+\Delta t} = u_i^t + \left(\frac{\partial u}{\partial t}\right)_i^t \Delta t \quad (8)$$

$\frac{\partial \bar{u}_i}{\partial t}^{t+\Delta t}$ is calculated using a **rearward difference scheme**

$$\left[\left(\frac{\partial u}{\partial t}\right)_i^t\right]_{avg} = 0.5 \left(\left(\frac{\partial u}{\partial t}\right)_i^t + \left(\frac{\partial \bar{u}}{\partial t}\right)_i^{t+\Delta t} \right) \quad (9)$$

Note: Consider $\left(\frac{\partial u}{\partial t}\right)_i^t$ and $\frac{\partial u_i^t}{\partial t}$ as equivalent notation

$$u_i^{t+\Delta t} = u_i^t + \left[\left(\frac{\partial u}{\partial t}\right)_i^t\right]_{avg} \Delta t \quad (10)$$

The above quantity is marched in time and the solution is updated until steady state is reached. The above method is applied to the density, the velocity and the temperature of our Nozzle simultaneously.

$$\frac{\partial \rho_i^t}{\partial t} = -\rho_i^t \frac{V_{i+1}^t - V_i^t}{dx} - \rho_i^t V_i^t \frac{\ln A_{i+1}^t - \ln A_i^t}{dx} - V_i^t \frac{\rho_{i+1}^t - \rho_i^t}{dx} \quad (11)$$

$$\frac{\partial V_i^t}{\partial t} = -V_i^t \frac{V_{i+1}^t - V_i^t}{dx} - \frac{1}{\gamma} \left(\frac{T_{i+1}^t - T_i^t}{dx} + \frac{T_i^t}{\rho_i^t} \frac{\rho_{i+1}^t - \rho_i^t}{dx} \right) \quad (12)$$

$$\frac{\partial T_i^t}{\partial t} = -V_i^t \frac{T_{i+1}^t - T_i^t}{dx} - (\gamma - 1) T_i^t \left(\frac{V_{i+1}^t - V_i^t}{dx} + V_i^t \frac{\ln A_{i+1}^t - \ln A_i^t}{dx} \right) \quad (13)$$

Naive Prediction:

$$\bar{\rho}_i^{t+\Delta t} = \rho_i^t + \left(\frac{\partial \rho}{\partial t}\right)_i^t \Delta t \quad (14)$$

$$\bar{V}_i^{t+\Delta t} = V_i^t + \left(\frac{\partial V}{\partial t}\right)_i^t \Delta t \quad (15)$$

$$\bar{T}_i^{t+\Delta T} = T_i^t + \left(\frac{\partial T}{\partial t}\right)_i^t \Delta t \quad (16)$$

$$\frac{\partial \bar{\rho}_i^t}{\partial t} = -\bar{\rho}_i^t \frac{\bar{V}_{i+1}^t - \bar{V}_i^t}{dx} - \rho_i^t \bar{V}_i^t \frac{\ln A_{i+1}^t - \ln A_i^t}{dx} - \bar{V}_i^t \frac{\bar{\rho}_{i+1}^t - \bar{\rho}_i^t}{dx} \quad (17)$$

$$\frac{\partial \bar{V}_i^t}{\partial t} = -\bar{V}_i^t \frac{\bar{V}_{i+1}^t - \bar{V}_i^t}{dx} - \frac{1}{\gamma} \left(\frac{\bar{T}_{i+1}^t - \bar{T}_i^t}{dx} + \frac{\bar{T}_i^t}{\bar{\rho}_i^t} \frac{\bar{\rho}_{i+1}^t - \bar{\rho}_i^t}{dx} \right) \quad (18)$$

$$\frac{\partial \bar{T}_i^t}{\partial t} = -\bar{V}_i^t \frac{\bar{T}_{i+1}^t - \bar{T}_i^t}{dx} - (\gamma - 1) \bar{T}_i^t \left(\frac{\bar{V}_{i+1}^t - \bar{V}_i^t}{dx} + \bar{V}_i^t \frac{\ln A_{i+1}^t - \ln A_i^t}{dx} \right) \quad (19)$$

Corrector:

$$\left[\left(\frac{\partial \rho}{\partial t} \right)_i^t \right]_{avg} = 0.5 \left(\frac{\partial \rho_i^t}{\partial t} + \frac{\partial \bar{\rho}_i^{t+\Delta t}}{\partial t} \right) \quad (20)$$

$$\left[\left(\frac{\partial V}{\partial t} \right)_i^t \right]_{avg} = 0.5 \left(\frac{\partial V_i^t}{\partial t} + \frac{\partial \bar{V}_i^{t+\Delta t}}{\partial t} \right) \quad (21)$$

$$\left[\left(\frac{\partial T}{\partial t} \right)_i^t \right]_{avg} = 0.5 \left(\frac{\partial T_i^t}{\partial t} + \frac{\partial \bar{T}_i^{t+\Delta t}}{\partial t} \right) \quad (22)$$

Final Update:

$$\rho_i^{t+\Delta t} = \rho_i^t + \left[\left(\frac{\partial \rho}{\partial t} \right)_i^t \right]_{avg} \Delta t \quad (23)$$

$$V_i^{t+\Delta t} = V_i^t + \left[\left(\frac{\partial V}{\partial t} \right)_i^t \right]_{avg} \Delta t \quad (24)$$

$$T_i^{t+\Delta t} = T_i^t + \left[\left(\frac{\partial T}{\partial t} \right)_i^t \right]_{avg} \Delta t \quad (25)$$

Determining The Time Step(Δt): The McCormack Method is conditionally stable and the time-step to be taken for stability is given by the Courant–Friedrichs–Lewy(CFL) condition.

[Anderson \[1995\]](#)

On a given iteration at time t , the time-step to be taken is given by:

In order to ensure stability at all spatial locations the common time-step taken is the minimum of the time-steps calculated at all the points.

$$\Delta t = \min(\Delta t_1^t, \Delta t_2^t, \dots, \Delta t_n^t) \quad (26)$$

Where:

$$\Delta t_i^t = C \frac{\Delta x}{a_i^t + V_i^t}$$

C is the Courant Number, for a stable convergence, it is less than one. **The non-dimensional speed of sound** is given by \sqrt{T}

Boundary Conditions: [Joseph W. Connolly](#)

The area of the inlet of the nozzle is considered to be finite in our solution, this means that out inlet is not a reservoir but rather a subsonic inlet. At the subsonic inlet two flow variables are kept constant while the velocity kept floating and is extrapolated along a characteristic line.

At the exit of the nozzle the flow reaches supersonic conditions. Here all three flow variables are made to float and the exit values are extrapolated along the characteristic line.

4 Implementation

The above Numerical Method is implemented in MATLAB, with appropriate initialization to ensure a smoother convergence. The three flow variables under iteration are initialized as follows: [Anderson \[1995\]](#)

$$\rho = 1 - 0.3146x$$

$$T = 1 - 0.2314x$$

$$V = (0.1 + 1.09x)T^{\frac{1}{2}}$$

The area of the Nozzle:

$$A = 1 + 2.2(x - 1.5)^2$$

This function initializes the flow variables according to the above relations.

```
1 function[density,temperature,velocity] = initialize_flow_variables(spacesteps,
    timesteps,delta_x)
2 %Create Placeholder
3 density = zeros(timesteps,spacesteps);
4 temperature = zeros(timesteps,spacesteps);
5 velocity = zeros(timesteps,spacesteps);
6
7 %Create intial values which help with better convergence
8
9 for i = 1:spacesteps
10     x = (i-1)*delta_x;
11     density(1,i) = 1-(0.3146*x);
12     temperature(1,i) = 1-(0.2314*x);
13     velocity(1,i) = (0.1+(1.09*x)).*(temperature(1,i)).^0.5;
14
15 end
16
17 end
```

This calculates the time-step to be taken on each iteration

```
1 %We try to calculate the timestep by which each iteration should march by
2 function[delta_t] = timestep(temperature_at_t,velocity_at_t,delta_x,C)
3
4     speed = sqrt(temperature_at_t);
5
6     delta_T = C*(delta_x)./(velocity_at_t + speed);
7
8     delta_t = min(delta_T);
9
10
11
12 end
```

This function implements the floating boundary conditions of the flow

```

1 function[density,temperature,velocity] = boundary_condition(density,temperature,
    velocity)
2     %The reservoir has only one floating variable , the inlet density
3     %and temperature are fixed
4     density(:,1) = 1;
5     temperature(:,1) = 1;
6     velocity(:,1) = 2.*velocity(:,2) - velocity(:,3);
7
8     %All the flow variables are floating at the exit
9     density(:,end) = 2.*density(:,end-1) - density(:,end-2);
10    temperature(:,end) = 2.*temperature(:,end-1) - temperature(:,end-2);
11    velocity(:,end) = 2.*velocity(:,end-1) - velocity(:,end-2);
12
13
14 end

```

This function carries out the time-marching of the computation. The function returns the values of the flow variables at steady state and also the variation of parameters at the throat, with time.

```

1
2 function [density,velocity,temperature,pressure,mach,mass_flow_rate_memory,
    pressure_throat,density_throat,velocity_throat,temperature_throat,Mach_throat,
    iter] = McCormack(Nx,x,delta_x,A,gamma,throat_index,courant_number)
3
4 %We initialize the data
5 [density,temperature,velocity] = initialize_flow_variables(Nx,1,delta_x);
6
7
8 change = inf;
9 tolerance = 1e-11;
10
11
12 iter = 1;
13
14 % We march in time until our error metric is smaller than our tolerance
15
16 while iter<50000
17
18     if change<tolerance
19         break
20     end
21
22     density_old = density;
23     velocity_old = velocity;
24     temperature_old = temperature;
25     mass_old = density_old.*A.*velocity_old;
26
27     % Using the CFL criterion we predict the timestep to be taken
28     [delta_t] = timestep(temperature,velocity,delta_x,courant_number);
29
30     % Naive prediction
31
32     for i = 2 : Nx-1
33
34         diff1 = (density(i+1)-density(i))/delta_x;
35         diff2 = (velocity(i+1)-velocity(i))/delta_x;
36         diff3 = (log(A(i+1))-log(A(i)))/delta_x;
37         diff4 = (temperature(i+1)-temperature(i))/delta_x;

```

```

38
39
40
41     d_density(i) = (-density(i)*diff2) - ((density(i)*velocity(i))*diff3) - (
velocity(i)*diff1);
42
43
44     d_velocity(i) = (-velocity(i)*diff2) - ((1/gamma)*((diff4)+((temperature(i)
)/density(i))*diff1)));
45
46
47     d_temperature(i) = (-velocity(i)*diff4) - ((gamma-1)*temperature(i))*(
diff2 + (velocity(i)*diff3));
48
49     density(i) = density(i) + (d_density(i)*delta_t);
50     velocity(i) = velocity(i) + (d_velocity(i)*delta_t);
51     temperature(i) = temperature(i) + (d_temperature(i)*delta_t);
52
53 end
54
55 % We correct our naive prediction
56 for i = 2 : Nx-1
57
58     diff1 = (density(i)-density(i-1))/delta_x;
59     diff2 = (velocity(i)-velocity(i-1))/delta_x;
60     diff3 = (temperature(i)-temperature(i-1))/delta_x;
61     diff4 = (log(A(i))-log(A(i-1)))/delta_x;
62
63
64     d_density_p(i) = (-density(i)*diff2) - ((density(i)*velocity(i))*diff4) -
(velocity(i)*diff1);
65
66
67     d_velocity_p(i) = (-velocity(i)*diff2) - ((1/gamma)*((diff3)+((temperature
(i)/density(i))*diff1)));
68
69
70     d_temperature_p(i) = (-velocity(i)*diff3) - ((gamma-1)*temperature(i))*(
diff2 + (velocity(i)*diff4));
71
72 end
73
74 % We take the average
75
76 d_density_avg = 0.5*(d_density+d_density_p);
77 d_velocity_avg = 0.5*(d_velocity+d_velocity_p);
78 d_temperature_avg = 0.5*(d_temperature+d_temperature_p);
79
80 % Note the average arrays have size Nx-1 not Nx
81
82
83 density = density_old + [d_density_avg*delta_t,0];
84 velocity = velocity_old + [d_velocity_avg*delta_t,0];
85 temperature = temperature_old + [d_temperature_avg*delta_t,0];
86 % We ensure the floating conditions are implemented
87
88 [density,temperature,velocity] = boundary_condition(density,temperature,
velocity);
89

```

```

90
91     mach = velocity./sqrt(temperature);
92     mass_flow = density.*A.*velocity;
93     pressure = density.*temperature;
94
95     % We monitor what happens in the throat
96
97     density_throat(iter) = density(throat_index);
98     temperature_throat(iter) = temperature(throat_index);
99     Mach_throat(iter) = mach(throat_index);
100    velocity_throat(iter) = velocity(throat_index);
101    pressure_throat(iter) = pressure(throat_index);
102
103    change = max(abs(mass_flow-mass_old));
104
105    %We put our flow variables into memory
106
107    mass_flow_rate_memory(iter,:) = mass_flow;
108
109    iter = iter+1;
110
111
112 end
113
114
115 end

```

5 Verification and Validation

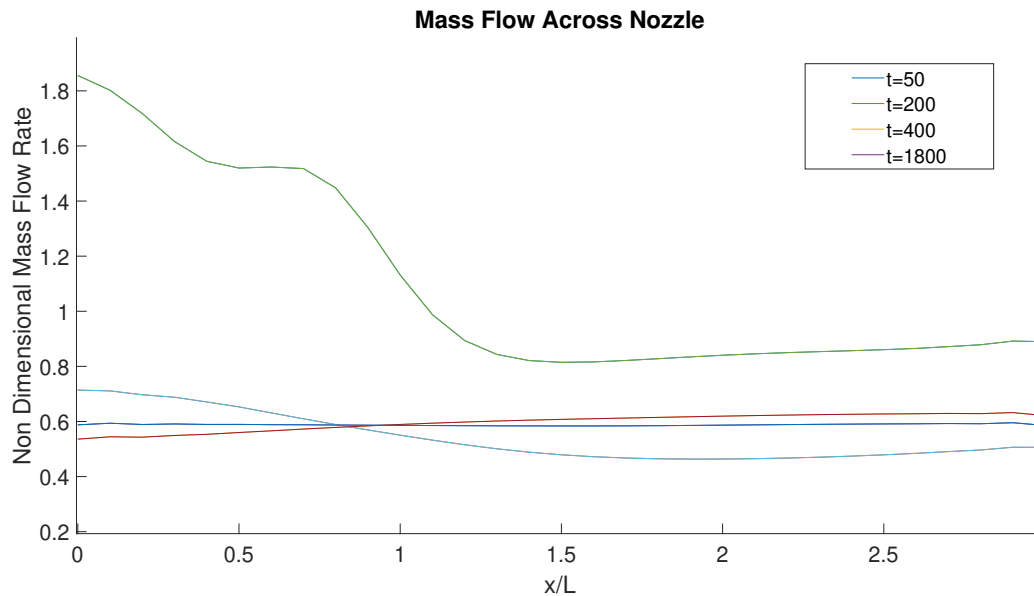


Figure 2: Conservation of mass.

As seen, the mass flow rate across the nozzle stabilizes with time to a constant value ensuring mass is conserved. The continuity equation is a direct consequence of mass conservation, hence it serves as a verification for the final solution.

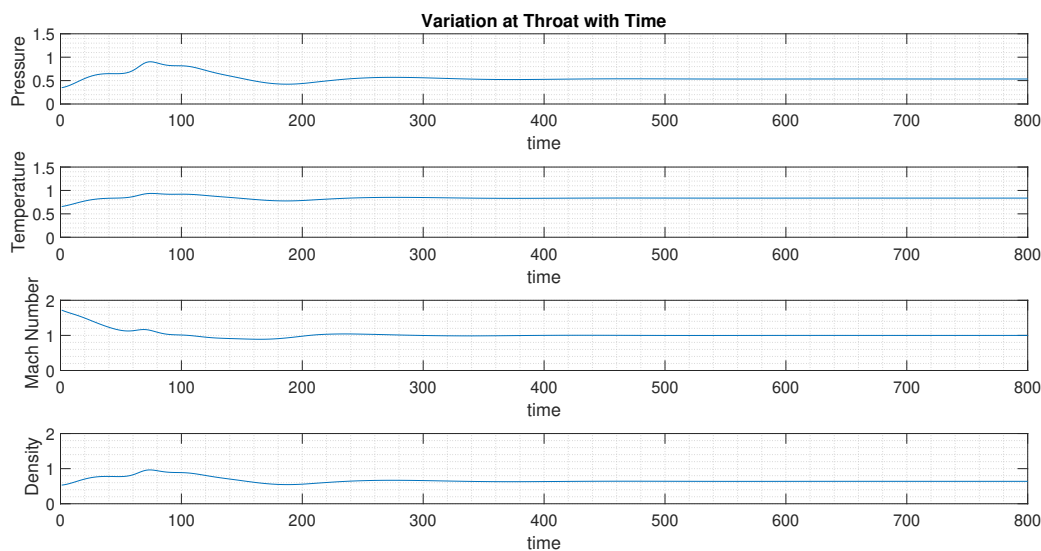


Figure 3: Convergence.

The above graphs shows the status of the flow variables at the nozzle with time, as can be seen after 300 time steps the solution has reached steady state. Hence the convergence of the solution is observed.

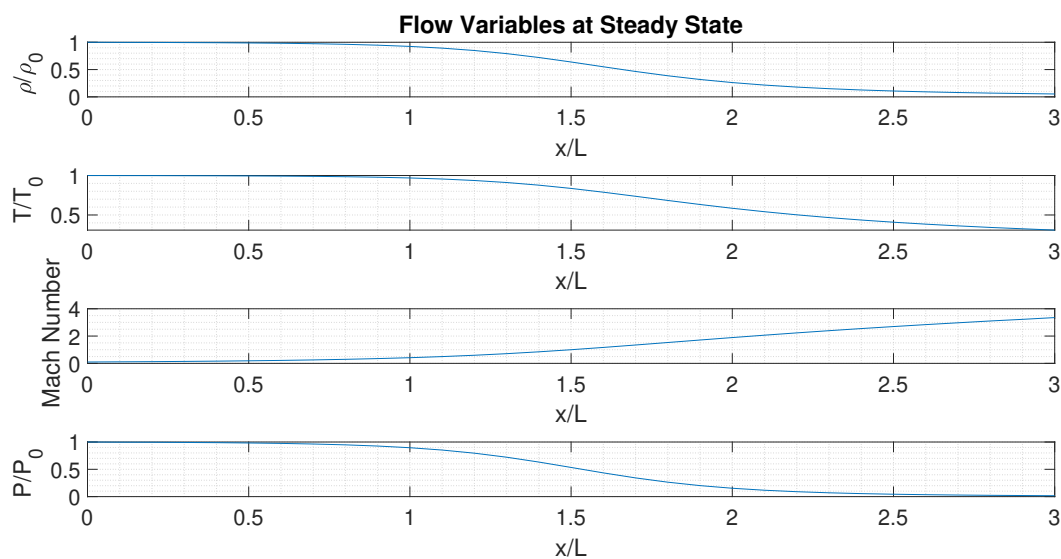


Figure 4: Steady State.

The above graph shows the profile of the Nozzle at Steady State, as can be seen the Nozzle Mach Number is 1.

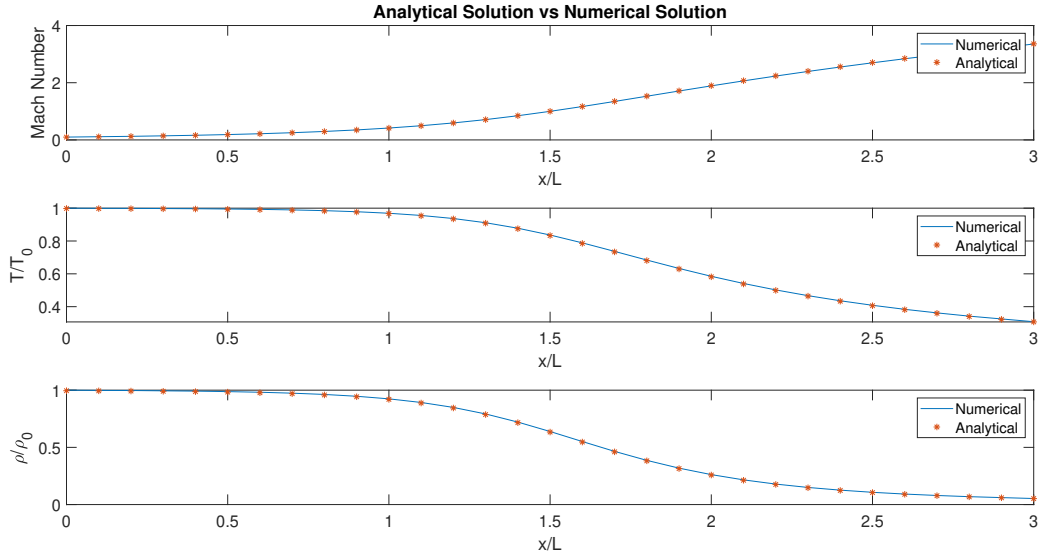


Figure 5: Comparing the solutions.

The flow variables vary in a manner consistent with the Analytical Solution. Thus validating the numerical solution.

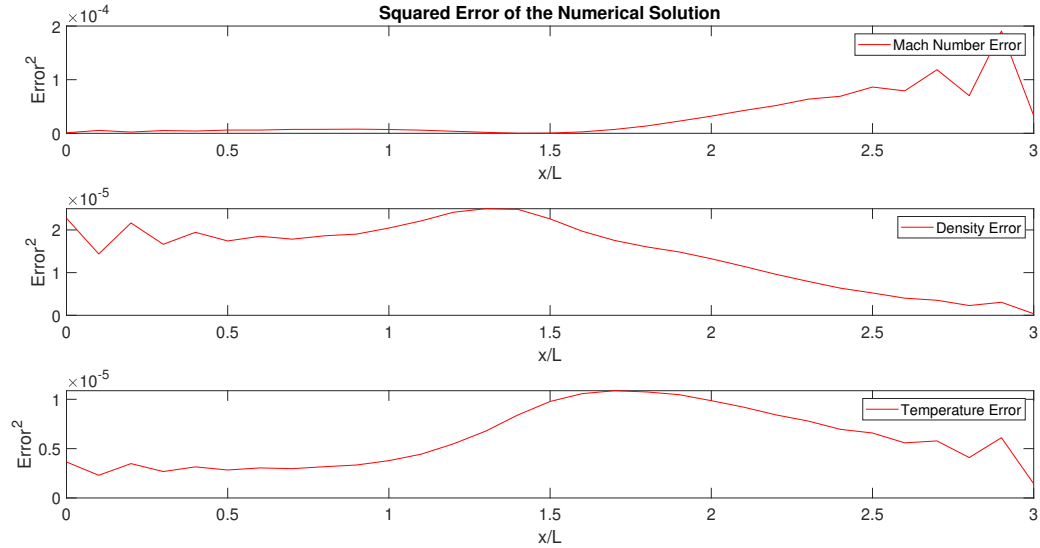


Figure 6: Evaluating the Error.

Overall the error is very small (order $10^{-4} - 10^{-5}$), however there is marginal variation along

the nozzle, the errors seem to peak near the throat of the nozzle. A possible explanation for this could be the fact that the flow experience a sharp change in velocity across the throat which results in the propagation of a larger error.

The mean squared errors:

Mach Number: $3.0799E-05$

Density: $1.4859E-05$

Temperature: $5.9295E-06$

Table 2: Comparing Values from Both the Solutions at The Throat

	M at throat	ρ at throat	T at throat	Time Taken by solver
Numerical	0.9994	0.6387	0.8365	0.3561
Analytical	1	0.6339	0.8333	0.2012

At the throat the solution is accurate to at least the second decimal place, the Mach number is accurate when rounded to the second decimal place.

6 Conclusion

As seen an accurate numerical solution solution has been developed, the errors which appear are caused due to discretization and truncation .The solution can be made further accurate by using a finer mesh.

A complete version of the implementation is available in the following [repository](#).

References

- J.D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill International Editions: Mechanical Engineering. McGraw-Hill, 1995. ISBN 9780071132107. URL https://books.google.co.in/books?id=phG_QgAACAAJ.
- J.D. Anderson. *Modern compressible flow, with historical perspective*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1982. URL <https://books.google.co.in/books?id=cPNQAAAAMAAJ>.
- George Kopasakis Joseph W. Connolly, David Friedlander. *Computational Fluid Dynamics Modeling of a Supersonic Nozzle and Integration into a Variable Cycle Engine Model*. URL <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140016531.pdf>.