Homework 4 & 5

Author: Alexander Jörud

Homework 4 & 5

- Quasi 1D flow
- Converging diverging nozzle

Inviscid 1D flow governing equations:

$$\boxed{\begin{aligned}
&\frac{\partial \rho'}{\partial t'} + V'\frac{\partial \rho'}{\partial x'} + \rho'\frac{\partial V'}{\partial x'} + \rho' V' \frac{\partial(\ln A')}{\partial x'} = 0 \\
&\frac{\partial V'}{\partial t'} + V'\frac{\partial V'}{\partial x'} + \frac{1}{\gamma}\left(\frac{\partial T'}{\partial x'} + \frac{T'}{\rho'}\frac{\partial \rho'}{\partial x'}\right) = 0 \\
&\frac{\partial T'}{\partial t'} + V'\frac{\partial T'}{\partial x'} + (\gamma-1)T'\left(\frac{\partial V'}{\partial x'} + V'\frac{\partial(\ln A')}{\partial x'}\right) = 0
\end{aligned}}$$

$$T' = \frac{T}{T_0} \qquad \rho' = \frac{\rho}{\rho_0} \qquad x' = \frac{x}{L} \qquad V' = \frac{V}{a_0}$$

$$t' = \frac{t a_0}{L} \qquad A' = \frac{A}{A^*} \qquad a_0 = \sqrt{\gamma R T_0}$$

Solve the flow through the nozzle of length $\boxed{3L}$ with the area varying as $\boxed{A' = 1 + 2.2\,(x' - 1.5)^2}$

MacCormack scheme

Continuity

Predictor:

$$\rho_j'^{\,n+1,*} = \rho_j'^{\,n} - V_j'^{\,n}\frac{\Delta t'}{\Delta x'}\left(\rho_{j+1}'^{\,n} - \rho_j'^{\,n}\right) - \rho_j'^{\,n}\frac{\Delta t'}{\Delta x'}\left(V_{j+1}'^{\,n} - V_j'^{\,n}\right)$$

$$- \rho_j'^{\,n} V_j'^{\,n}\frac{\Delta t'}{\Delta x'}\left(\ln A_{j+1}' - \ln A_j'\right)$$

Corrector:

$$\rho_{j-1}'^{\,n+1,*} = \rho_{j-1}'^{\,n} - \ldots$$

$$\rho_j'^{\,n+1} = \frac{\rho_j'^{\,n} + \rho_j'^{\,n+1,*}}{2} - \frac{1}{2}\frac{\Delta t'}{\Delta x'}V_j'^{\,n+1,*}\left(\rho_j'^{\,n+1,*} - \rho_{j-1}'^{\,n+1,*}\right) -$$

$$- \frac{1}{2}\frac{\Delta t}{\Delta x}\rho_j'^{\,n+1,*}\left(V_j'^{\,n+1,*} - V_{j-1}'^{\,n+1,*}\right) - \frac{1}{2}\frac{\Delta t'}{\Delta x'}\rho_j'^{\,n+1,*}V_j'^{\,n+1,*}\left(\ln A_j' - \ln\right.$$

Predictor:

Momentum

$$V_j'^{\,n+1,*} = V_j'^{\,n} - V_j'^{\,n}\frac{\Delta t'}{\Delta x'}\left(V_{j+1}'^{\,n} - V_j'^{\,n}\right) - \frac{1}{\gamma}\frac{\Delta t'}{\Delta x'}\left(\left(T_{j+1}'^{\,n} - T_j'^{\,n}\right) + \right.$$

$$\left. + \frac{T_j'^{\,n}}{\rho_j'^{\,n}}\left(\rho_{j+1}'^{\,n} - \rho_j'^{\,n}\right)\right)$$

$$V_{j-1}'^{\,n+1,*} = V_{j-1}'^{\,n} \ldots$$

Corrector:

Momentum

$$V_j^{\prime\,n+1} = \frac{V_j^{\prime\,n} + V_j^{\prime\,n+1,*}}{2} - \frac{1}{2}\frac{\Delta t'}{\Delta x'} V_j^{\prime\,n+1,*}\left(V_j^{\prime\,n+1,*} - V_{j-1}^{\prime\,n+1,*}\right)$$

$$-\frac{1}{2}\frac{\Delta t'}{\Delta x'}\frac{1}{\gamma}\left(\left(T_j^{\prime\,n+1,*} - T_{j-1}^{\prime\,n+1,*}\right) + \frac{T_j^{\prime\,n+1,*}}{\rho_j^{\prime\,n+1,*}}\left(\rho_j^{\prime\,n+1,*} - \rho_{j-1}^{\prime\,n+1,*}\right)\right)$$

Energy

Predictor:

$$T_j^{\prime\,n+1,*} = T_j^{\prime\,n} - V_j^{\prime\,n}\frac{\Delta t'}{\Delta x'}\left(T_{j+1}^{\prime\,n} - T_j^{\prime\,n}\right)$$

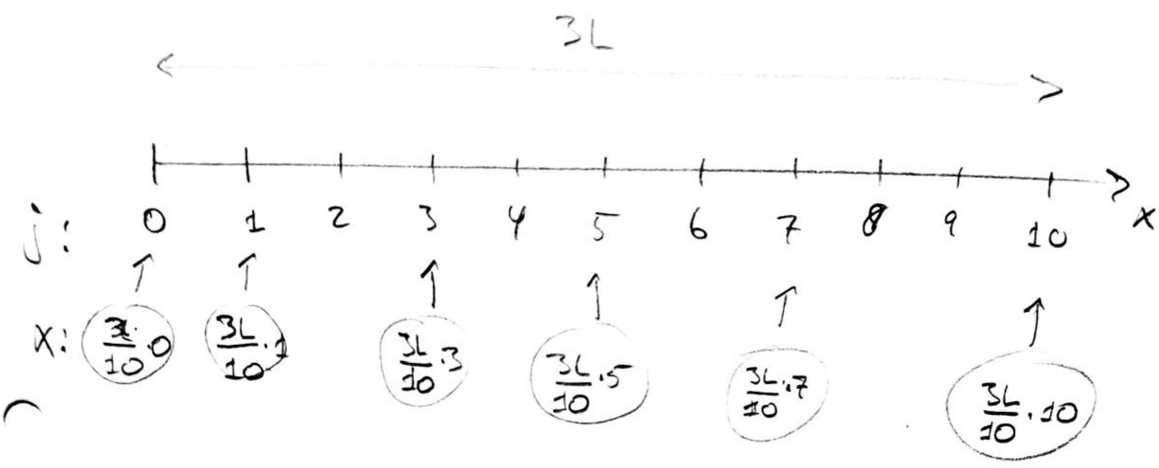
$$-(\gamma-1)\,T_j^{\prime\,n}\frac{\Delta t'}{\Delta x'}\left(\left(V_{j+1}^{\prime\,n} - V_j^{\prime\,n}\right) + V_j^{\prime\,n}\left(\ln A'_{j+1} - \ln A'_j\right)\right)$$

$$T_{j-1}^{\prime\,n+1,*} = T_{j-1}^{\prime\,n} - \dots$$

Corrector:

$$T_j^{\prime\,n+1} = \frac{T_j^{\prime\,n} + T_j^{\prime\,n+1,*}}{2} - \frac{1}{2}\frac{\Delta t'}{\Delta x'} V_j^{\prime\,n+1,*}\left(T_j^{\prime\,n+1,*} - T_{j-1}^{\prime\,n+1,*}\right) -$$

$$-(\gamma-1)\,T_j^{\prime\,n+1,*}\frac{1}{2}\frac{\Delta t'}{\Delta x'}\left(\left(V_j^{\prime\,n+1,*} - V_{j-1}^{\prime\,n+1,*}\right) + V_j^{\prime\,n+1,*}\left(\ln A'_j - \ln A'_{j-1}\right)\right)$$



$x' = \frac{x}{L}$ where $A' = 1 + 2.2(x' - 1.5)^2$

Initial conditions
$$\begin{cases} \rho' = 1 - 0.3146x' \\ T' = 1 - 0.2314x' \\ V' = (0.1 + 1.09x')\sqrt{T'} \end{cases}$$

Matlab result files

10 grid points solution:

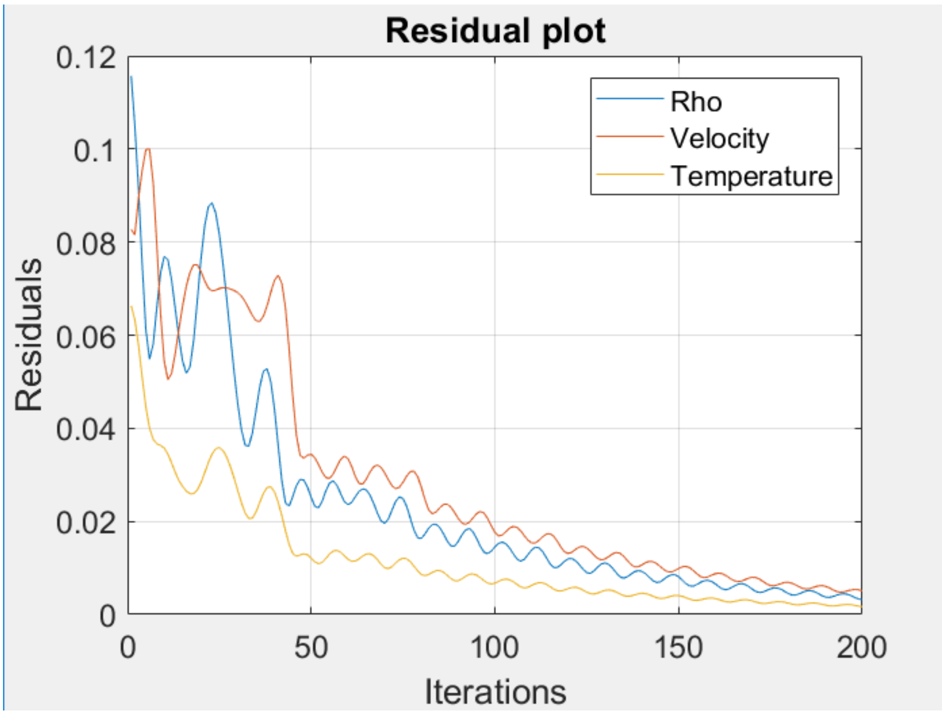
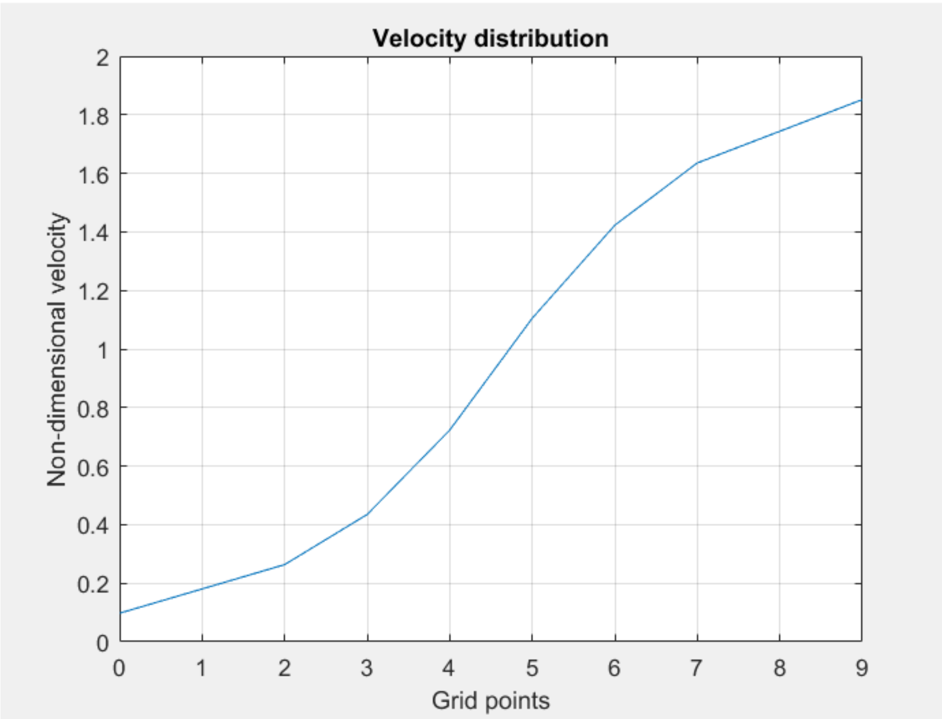


Figure 1, Residual plot of velocity, rho and temperature. Some oscillations occur.



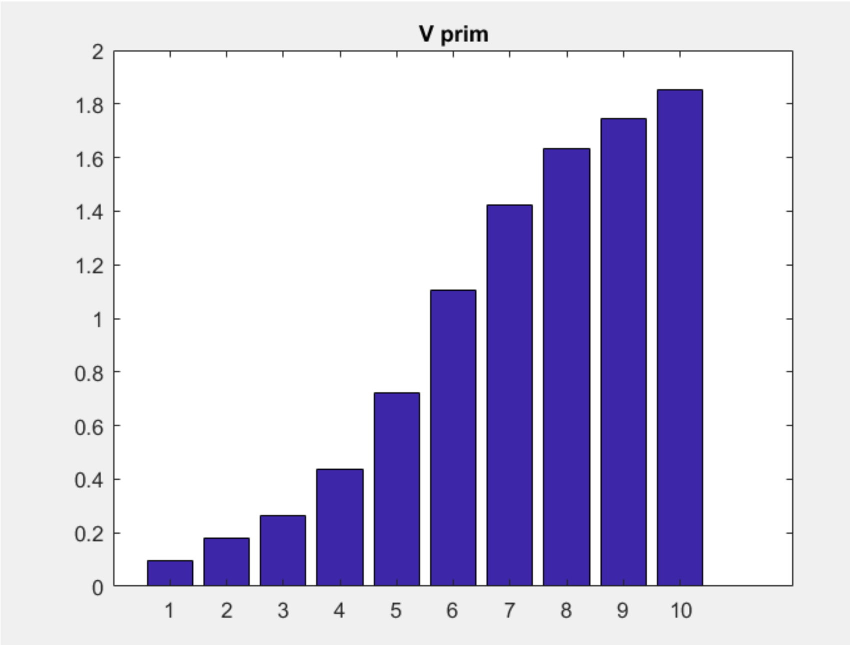


Figure 2, Y-axis: Non-dimensional velocity, X-axis: grid points.

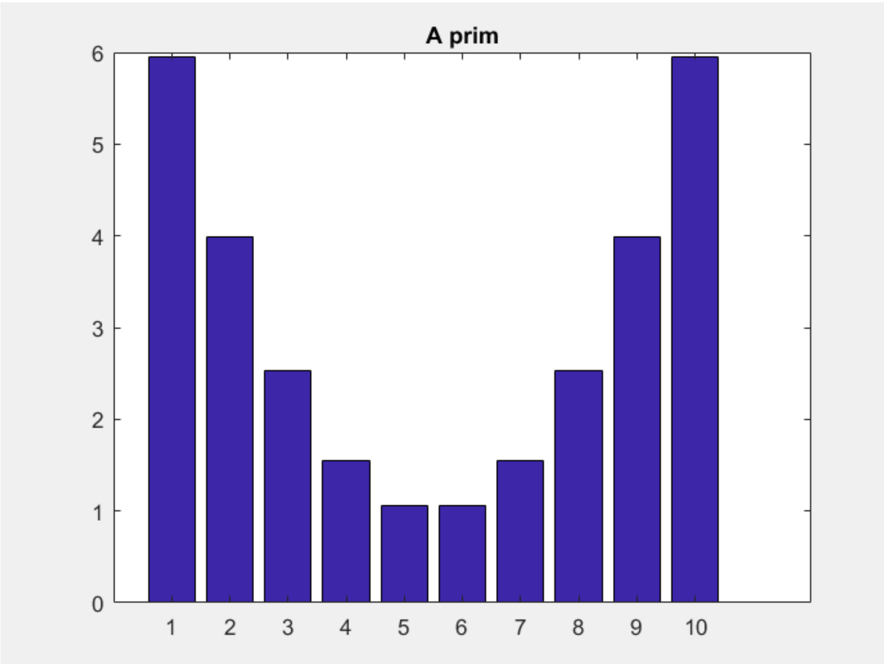


Figure 3, Y-axis: Non-dimensional Area, X-axis: grid points. Clearly depicts the converging-diverging nozzle.

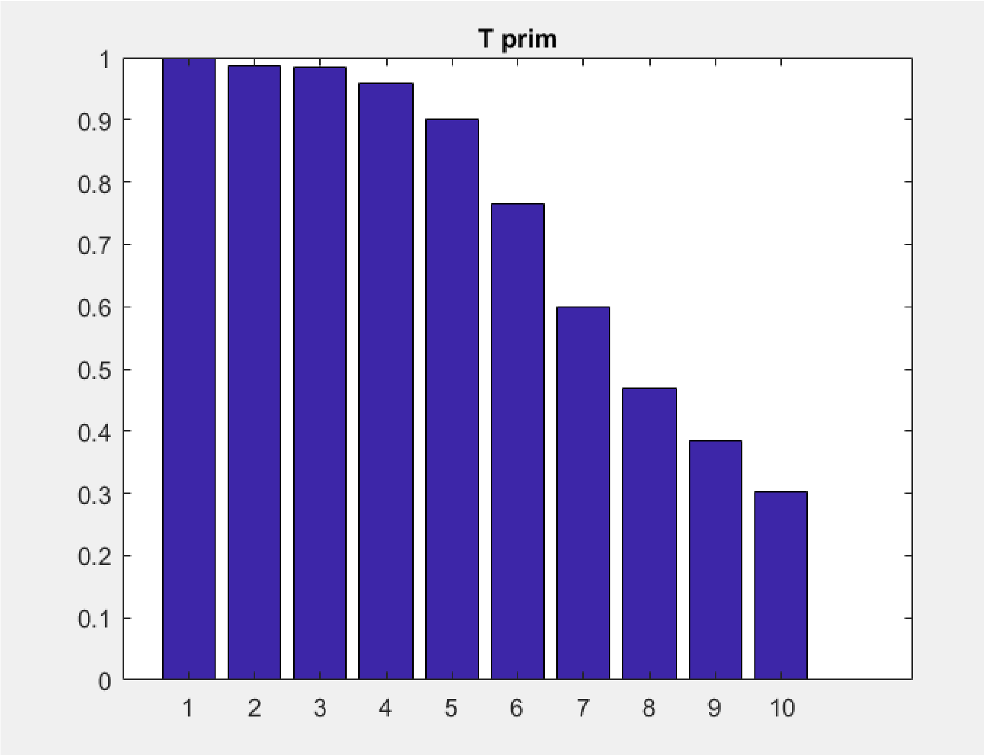


Figure 4, Y-axis: Non-dimensional temperature, X-axis: grid points.

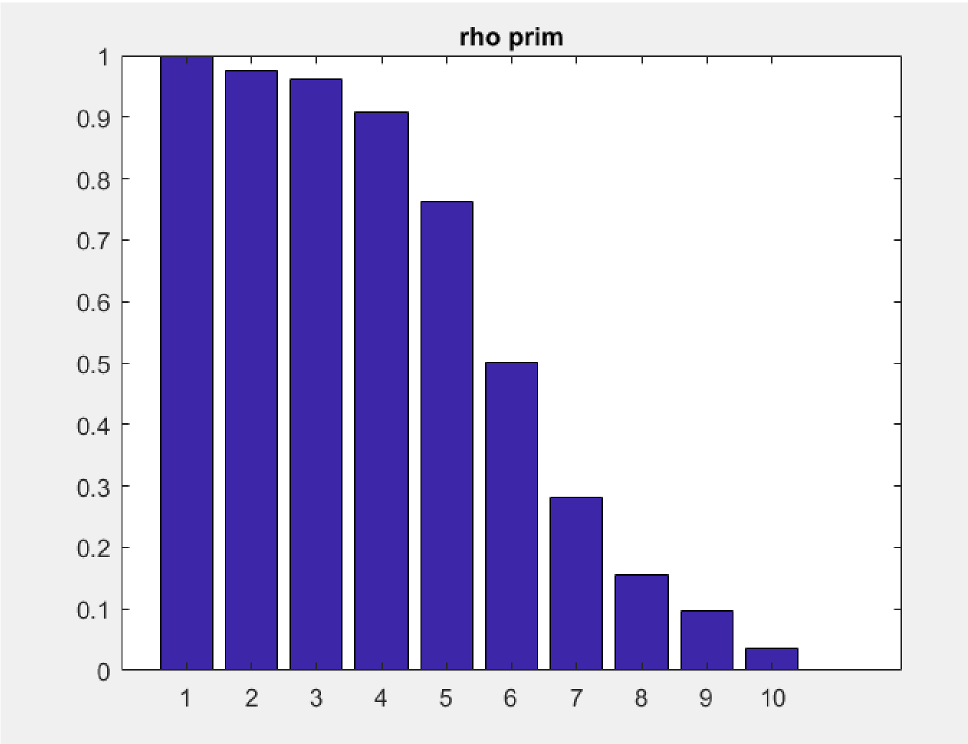


Figure 5, Y-axis: Non-dimensional density, X-axis: grid points.

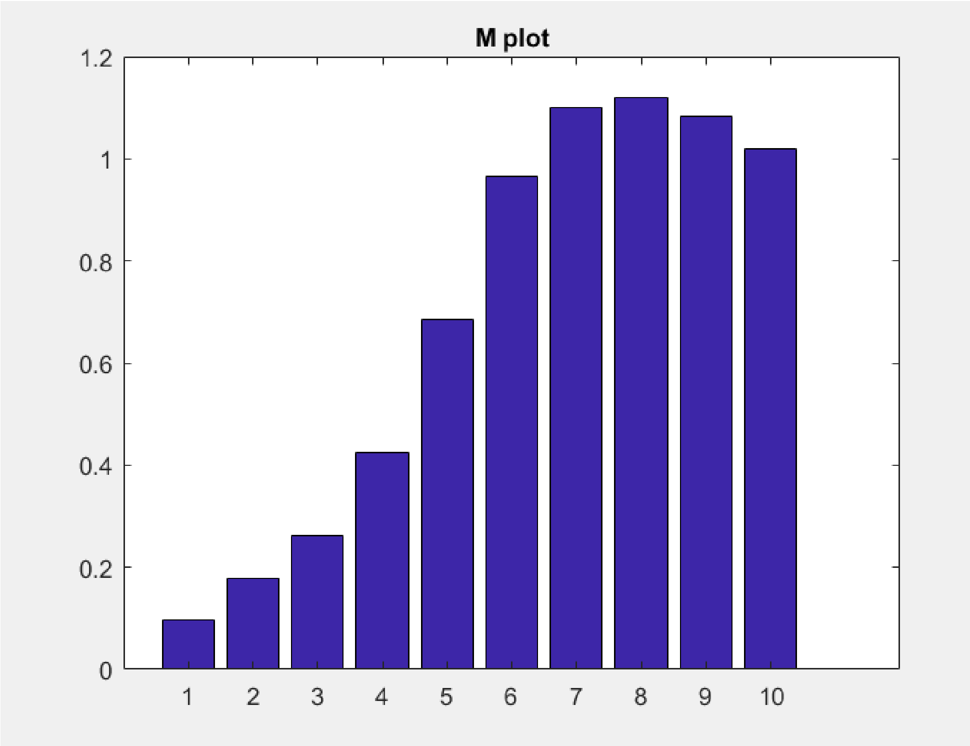


Figure 6, Y-axis: Non-dimensional mach number, X-axis: grid points.

Non-dimensional Mach number is defined as M = V * sqrt(T), Conclusion:

Several attempts at different stability solutions was carried out. With alternating Courant number or different methods of stability condition. 2 different methods are provided in the attached MATLAB code. Note that the stability time step should be calculated in every iteration.

Although when observing the results from figure 6, it provides a satisfying result in what was expected from a subsonic inlet and a supersonic outlet. Note this is non-dimensional Mach number. But in theory with a supersonic outlet, Mach = 1 should be achieved in at the throat.

19 grid point solution:

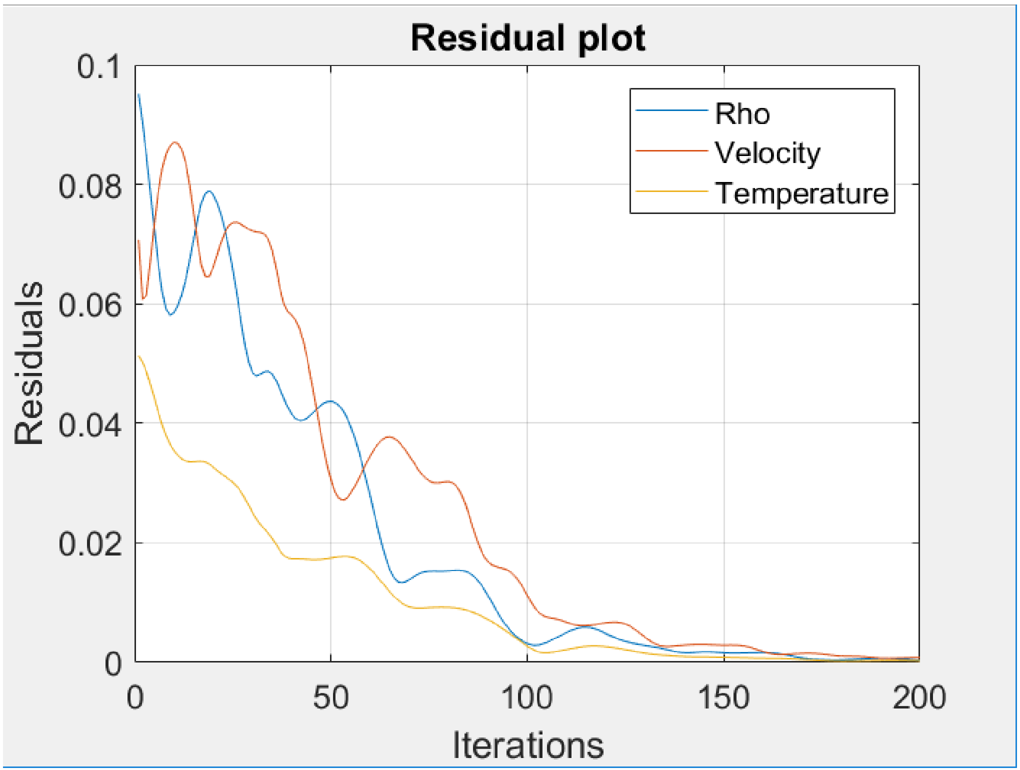


Figure 7, Residual plot of velocity, rho and temperature.

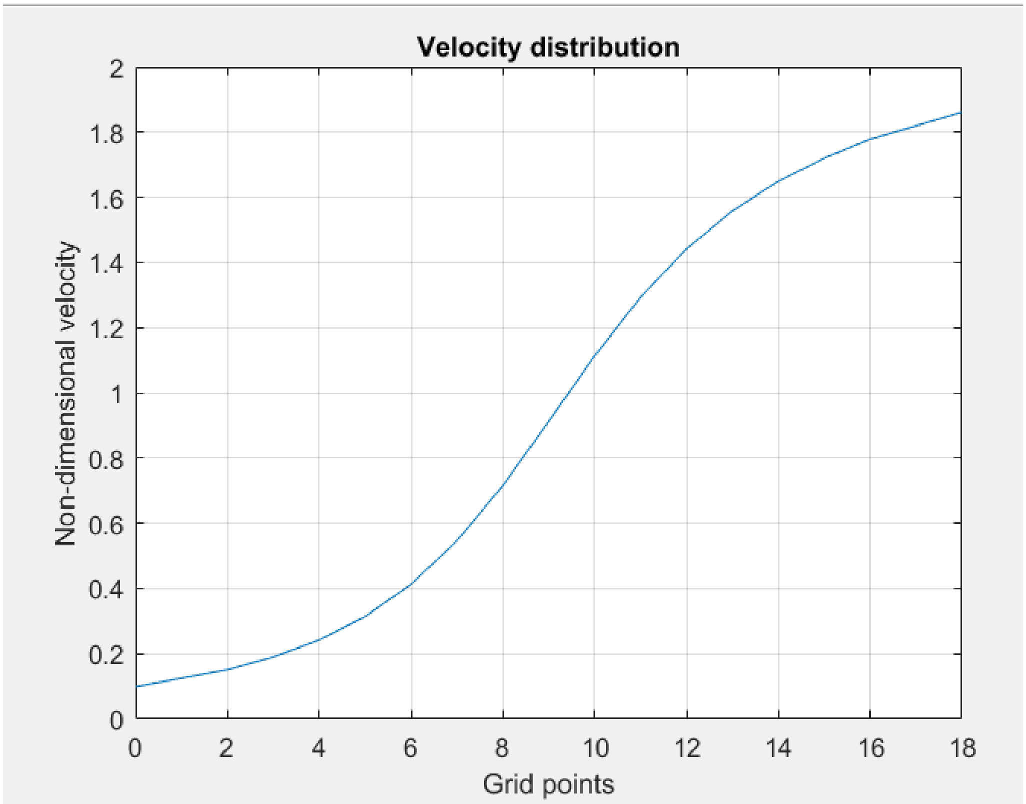


Figure 8, Velocity distribution over the different grid points.

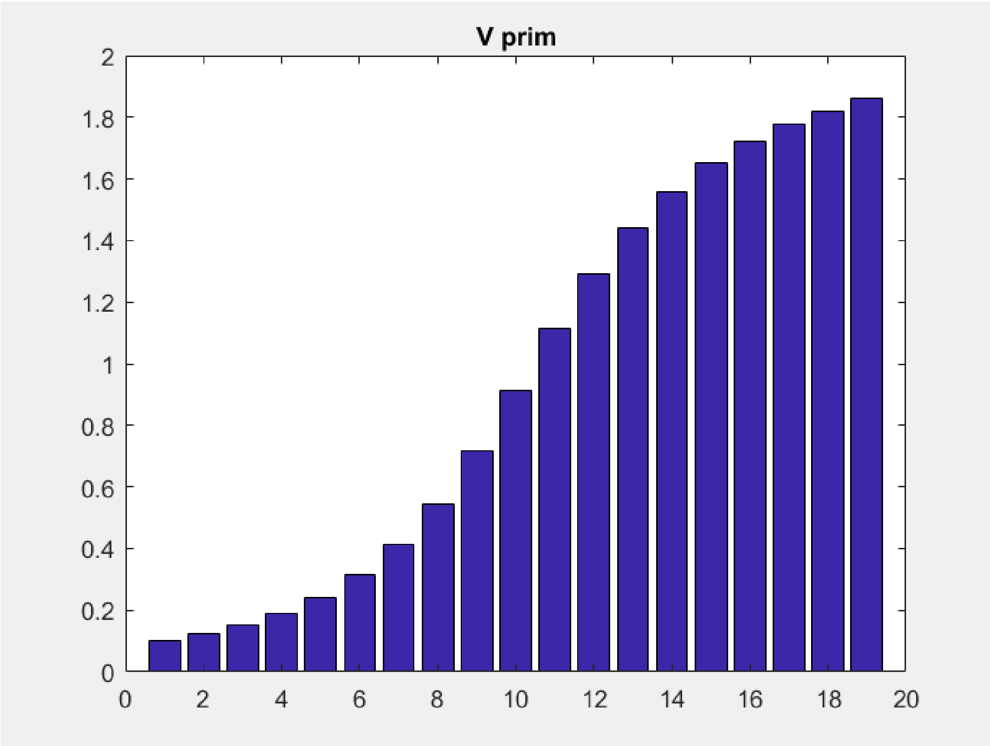


Figure 9, Y-axis: Non-dimensional velocity, X-axis: grid points. Actually, the very same data plotted as in figure 8.

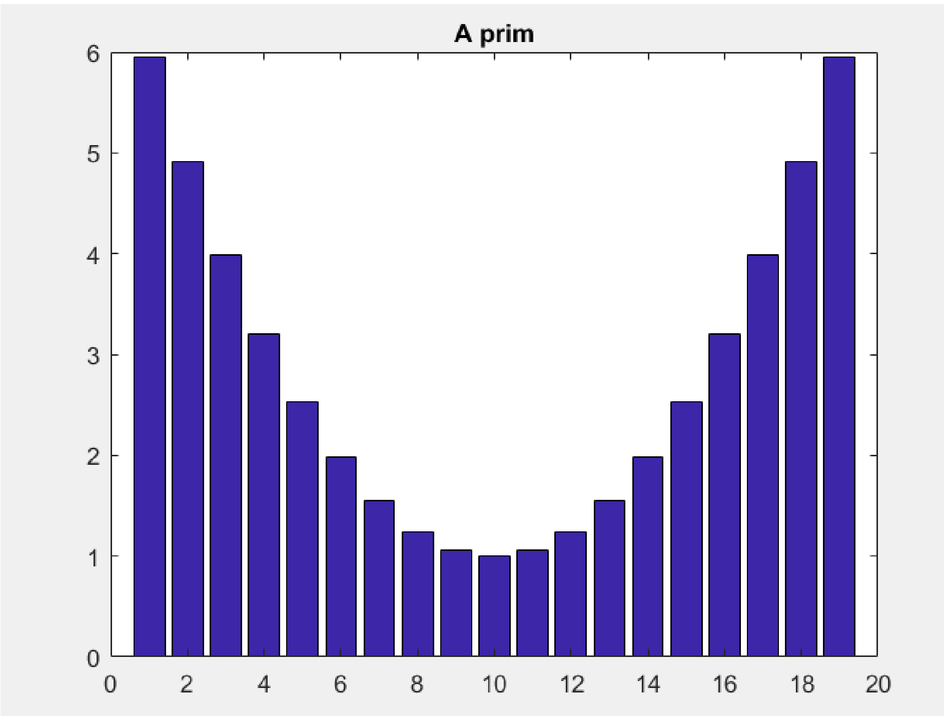


Figure 10, Y-axis: Non-dimensional Area, X-axis: grid points. Clearly depicts the converging-diverging nozzle.

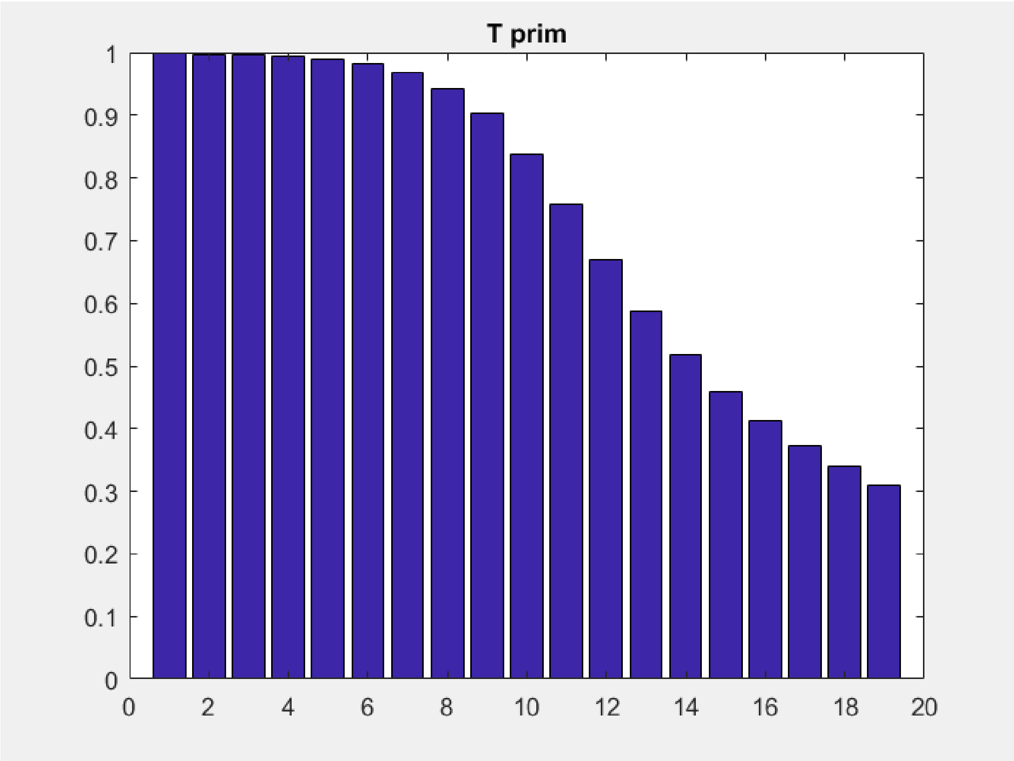


Figure 11, Y-axis: Non-dimensional temperature, X-axis: grid points.

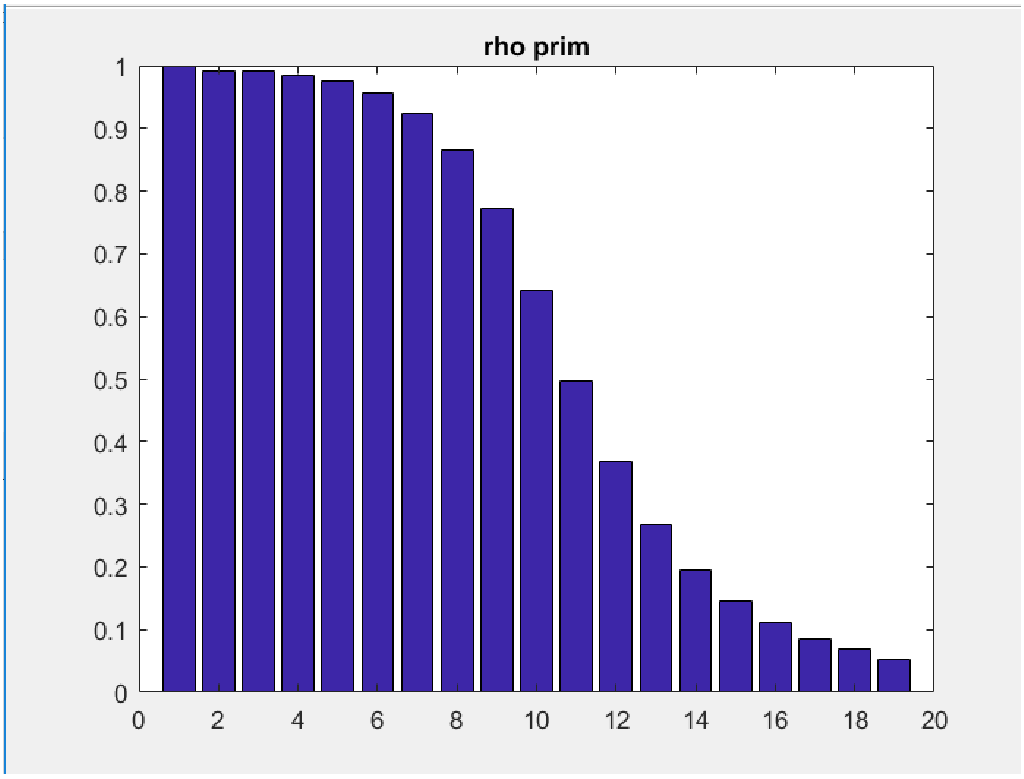


Figure 12, Y-axis: Non-dimensional density, X-axis: grid points.

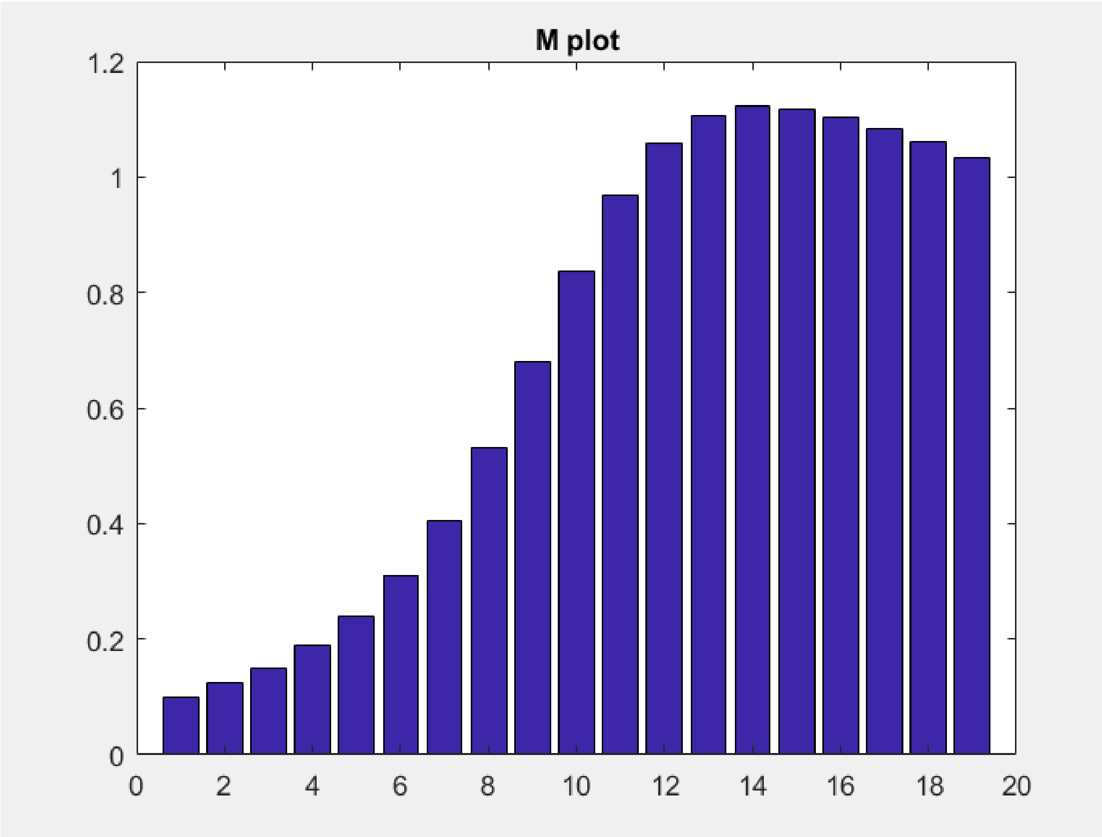


Figure 13, Y-axis: Non-dimensional Mach number, X-axis: grid points.

Conclusions:

Different grid resolutions provided different residual convergence, where the more coarse grid gave some oscillations of the residual but still a slope which entails convergence.

The fine grid provided a more satisfying result regarding residual plot.

Matlab code:

```
clc
clear all
close all

%% Manually set variables
%Number of grid points manually entered by user
nbr_of_gridpoints = 9;
%Number of iterations manually entered by user
nbr_iter = 200;

%%
%10 node point mesh
Grid = 0:nbr_of_gridpoints;

%Length of nozzle is 3L
L = 0.05;

x = (3*L/(length(Grid)-1))*Grid;

x_prim = x./L;

A_prim =  1+2.2*((x_prim-1.5).^2);



%Initial conditions
rho_prim = 1-0.3146*x_prim;
T_prim = 1-0.2314*x_prim;
V_prim = (0.1+1.09*x_prim).*sqrt(T_prim);


%Local speed of sound
gamma = 1.4; %Standard air conditions
R = 287.058; %Universal gas constant for dry air
% a = sqrt(gamma*R*T_prim);
a = sqrt(T_prim);

%Inviscid stability condition

%Conservative approach, set Mach = 3 for stability condition
regarding
%Maximum possible velocities in the system for all iterations.


% M__stability = 2; %Mach number
% T_stability = 500;  %Temperature
% a_stability = sqrt(gamma*R*T_stability);
% V_stability = M__stability*a_stability;
% delta_x = x_prim(2)-x_prim(1);
```

```
% delta_t =
((V_stability/delta_x)+a_stability*sqrt((1/((delta_x)^2))))^-
1;
% dtdx = delta_t/delta_x;

%Courant-Friedrichs-Lewy condition
a_prim = sqrt(T_prim);
delta_x = x_prim(2)-x_prim(1);
C = 0.8;
delta_t = min(C*(delta_x./(a+V_prim)));

dtdx = delta_t./delta_x;


%"The residual is the difference between the previous result
and the
%current result. As these errors are decreasing the equation
results
%are reaching values that are changing less and less. This is
what is
%known as convergence. That is the solutions are converging."
%Reference: https://www.cfd-online.com/Forums/fluent/28009-
residuals.html

%I decide to use a for loop for manual control of number of
iterations,
%while loop may also be used.

%% Boundary conditions version 1

%Subsonic inlet
% M < 0.8
% 1 Mach = Velocity / local speed of sound

%In Isentropic flow properties table
%Mach = 0.6 provides:
%                    T0 / T = 1.072
%                    p0 / p = 1.276
%                    rho0 / rho = 1.19
%                    A / A_star = 1.18


% M_inlet = 0.6; %Inlet Mach number
% T_inlet = 400;  %Inflow temp 400 K
% a_inlet = sqrt(gamma*R*T_inlet);
%
% T0_inlet = T_inlet*1.072;
% a0_inlet = sqrt(gamma*R*T0_inlet);
%
% V_inlet = M_inlet*a_inlet;
% V_prim_inlet = V_inlet/a0_inlet;
% T_prim_inlet = T_inlet/T0_inlet;
```

```
% rho_prim_inlet = 1/1.19;
%
%
% %Set inlet boundary conditions
%     V_prim(1) = V_prim_inlet;
%     T_prim(1) = T_prim_inlet;
%     rho_prim(1) = rho_prim_inlet;


%% Boundary conditions version 2

        %Extrapolation at boundaries if needed
        %Inlet: p1 = 2*p2 - p3
        %Outlet: pN = 2*pNmin1 - pNmin2

        %Subsonic inflow
        %Extrapolate 1 boundary condition from interior

        V_prim_inlet_extrapo =  2*V_prim(2)-V_prim(3);


        %Supersonic outflow
        %Extrapolate all boundary conditions from interior


        rho_prim_outlet_extrapo =
2*rho_prim(length(rho_prim)-1)-...
                                  rho_prim(length(rho_prim)-
2);

        V_prim_outlet_extrapo =  2*V_prim(length(V_prim)-1)-
...
                                  V_prim(length(V_prim)-2);

        T_prim_outlet_extrapo =  2*T_prim(length(T_prim)-1)-
...
                                  T_prim(length(T_prim)-2);

rho_prim_inlet = 1;
T_prim_inlet = 1;

rho_prim(1) = rho_prim_inlet;
rho_prim(length(rho_prim)) = rho_prim_outlet_extrapo;

T_prim(1) = T_prim_inlet;
T_prim(length(T_prim)) = T_prim_outlet_extrapo;

V_prim(1) = V_prim_inlet_extrapo;
V_prim(length(V_prim)) = V_prim_outlet_extrapo;
```

```
%% Start of iteration loop
for iter = 1:nbr_iter



for j = 2:(length(Grid)-1)

%A' values does not alternate between timestep so it is
possible to compute
% outside the for loop
dlnA_dx_ver1 = log(A_prim(j+1))-log(A_prim(j));
dlnA_dx_ver2 = log(A_prim(j))-log(A_prim(j-1));


%% Predictor
%Continuity

rho_nplus1_j_s(j-1) = rho_prim(j)-V_prim(j)*dtdx*...
                    (rho_prim(j+1)-rho_prim(j))-
rho_prim(j)*dtdx*...
                    (V_prim(j+1)-V_prim(j))-
rho_prim(j)*V_prim(j)*...
                    dtdx*dlnA_dx_ver1;



rho_nplus1_jmin1_s(j-1) = rho_prim(j-1)-V_prim(j-1)*dtdx*...
                    (rho_prim(j)-rho_prim(j-1))-rho_prim(j-
1)*dtdx*...
                    (V_prim(j)-V_prim(j-1))-rho_prim(j-
1)*V_prim(j-1)*...
                    dtdx*dlnA_dx_ver2;

%Momentum

V_nplus1_j_s(j-1) = V_prim(j)- V_prim(j)*dtdx*(V_prim(j+1)-
V_prim(j))-...
                  (1/gamma)*dtdx*((T_prim(j+1)-T_prim(j))+...
                  (T_prim(j)/rho_prim(j))*(rho_prim(j+1)-
rho_prim(j)));

V_nplus1_jmin1_s(j-1) = V_prim(j-1)- V_prim(j-
1)*dtdx*(V_prim(j)-...
                V_prim(j-1))-(1/gamma)*dtdx*((T_prim(j)-
T_prim(j-1))+...
                  (T_prim(j-1)/rho_prim(j-1))*(rho_prim(j)-
rho_prim(j-1)));

%Energy
T_nplus1_j_s(j-1) = T_prim(j)-V_prim(j)*dtdx*(T_prim(j+1)-
T_prim(j))-...
```

```
                    (gamma-1)*T_prim(j)*dtdx*((V_prim(j+1)-
V_prim(j))+...
                    V_prim(j)*dlnA_dx_ver1);

T_nplus1_jmin1_s(j-1) = T_prim(j-1)-V_prim(j-
1)*dtdx*(T_prim(j)-...
                    T_prim(j-1))-(gamma-1)*T_prim(j-1)*dtdx*...
                    ((V_prim(j)-V_prim(j-1))+...
                    V_prim(j-1)*dlnA_dx_ver2);

%% Corrector
%Continuity

rho_nplus1_j(j-1) = 0.5*((rho_prim(j)+rho_nplus1_j_s(j-1)) -
...
                      dtdx*V_nplus1_j_s(j-1)*(rho_nplus1_j_s(j-
1)-...
                      rho_nplus1_jmin1_s(j-1))-
dtdx*rho_nplus1_j_s(j-1)*...
                      (V_nplus1_j_s(j-1)-V_nplus1_jmin1_s(j-1))-
dtdx*...
                      rho_nplus1_j_s(j-1)*V_nplus1_j_s(j-
1)*dlnA_dx_ver2);


V_nplus1_j(j-1) = 0.5*((V_prim(j)+V_nplus1_j_s(j-1))-
dtdx*V_nplus1_j_s(j-1)*...
                 (V_nplus1_j_s(j-1)-V_nplus1_jmin1_s(j-1))-...
               dtdx*(1/gamma)*((T_nplus1_j_s(j-1)-
T_nplus1_jmin1_s(j-1))...
                 +(T_nplus1_j_s(j-1)/rho_nplus1_j_s(j-
1))*(rho_nplus1_j_s(j-1)...
                 -rho_nplus1_jmin1_s(j-1))));


T_nplus1_j(j-1) = 0.5*((T_prim(j)+T_nplus1_j_s(j-1))-
dtdx*V_nplus1_j_s(j-1)*...
                 (T_nplus1_j_s(j-1)-T_nplus1_jmin1_s(j-1))-...
                 (gamma-1)*T_nplus1_j_s(j-
1)*dtdx*((V_nplus1_j_s(j-1)-...
                 V_nplus1_jmin1_s(j-1))+V_nplus1_j_s(j-
1)*dlnA_dx_ver2));


end


%Check residuals

rho_res = norm(rho_prim(2:(length(Grid)-1))-rho_nplus1_j);
V_res = norm(V_prim(2:(length(Grid)-1))-V_nplus1_j);
```

```
T_res = norm(T_prim(2:(length(Grid)-1))-T_nplus1_j);
disp('----------------------------------------------')
disp([num2str(iter),' Rho Residual ',num2str(rho_res)])
disp([num2str(iter),' Velocity Residual ',num2str(V_res)])
disp([num2str(iter),' Temperature Residual ',num2str(T_res)])


rho_prim = rho_nplus1_j;
T_prim = T_nplus1_j;
V_prim = V_nplus1_j;

        %Extrapolation at boundaries if needed
        %Inlet: p1 = 2*p2 - p3
        %Outlet: pN = 2*pNmin1 - pNmin2

        %Subsonic inflow
        %Extrapolate 1 boundary condition from interior

        V_prim_inlet_extrapo =  2*V_prim(1)-V_prim(2);


        %Supersonic outflow
        %Extrapolate all boundary conditions from interior


        rho_prim_outlet_extrapo =
2*rho_prim(length(rho_prim))-...
                                  rho_prim(length(rho_prim)-
1);

        V_prim_outlet_extrapo =  2*V_prim(length(V_prim))-...
                                  V_prim(length(V_prim)-1);

        T_prim_outlet_extrapo =  2*T_prim(length(T_prim))-...
                                  T_prim(length(T_prim)-1);


rho_prim = [rho_prim_inlet rho_prim rho_prim_outlet_extrapo];
T_prim = [T_prim_inlet T_prim T_prim_outlet_extrapo];
V_prim = [V_prim_inlet_extrapo V_prim V_prim_outlet_extrapo];

%Note the rest of the zero values will obtain a value in the
next
%iteration from extrapolation

%Arrays for plot
if iter == 1
    iter_arr = iter;
    rho_res_arr = rho_res;
    V_res_arr = V_res;
    T_res_arr = T_res;
else
```

```
    iter_arr = [iter_arr iter];
    rho_res_arr = [rho_res_arr rho_res];
    V_res_arr = [V_res_arr V_res];
    T_res_arr = [T_res_arr T_res];
end


clf
%Plot of residuals
plot(iter_arr,rho_res_arr);
grid on
hold on
plot(iter_arr,V_res_arr)
hold on
plot(iter_arr,T_res_arr)
% title(['Load step:', num2str(n)],'FontSize', 20)
title('Residual plot','FontSize', 20)
xlabel('Iterations','FontSize', 14)
ylabel('Residuals','FontSize', 14)
set(gca,'FontSize',14)
legend('Rho','Velocity','Temperature')
drawnow



end


%% Plot

%Nondimensional Mach number
M_plot = V_prim.*sqrt(abs(T_prim));

figure
plot(Grid,V_prim)
grid on
title('Velocity distribution')
xlabel('Grid points')
ylabel('Non-dimensional velocity')

figure
bar(V_prim)
title('V prim')

figure
bar(A_prim)
title('A prim')

figure
bar(T_prim)
title('T prim')
```

```
figure
bar(rho_prim)
title('rho prim')

figure
bar(M_plot)
title('M plot')
```