Numerical Simulation of Flow in a Converging-Diverging Nozzle to Observe the Shock Formation

Neelotpal Dutta,B16106
B.Tech., 3(rd) Year

I. Introduction

In most of the applications of the fluid, we are concerned with harnessing the energy in a flowing fluid. One of the most basic relations for a fluid, the Bernaulli's equation deals with the conservation of the net energy content of a flowing fluid in absence of any external supply, *non-idealitis* or losses. The relation mentions the energy in three forms: pressure, velocity (kinetic) and potential (Eqn. 1).

$$P + (1/2)\rho V^2 + \rho g H = Constant \tag{1}$$

Hence, energy can be harvested in any of the three forms. If we want to harvest the kinetic energy, then it will be diserable to increase the net content of the kinetic energy in the same total energy content. Thus, we need to convert between the forms of the energy (Reaction and Impulse based devices harvest pressure and kinetic energy respectively). In other applications also we may like to change the velocity of a fluid with changing its net energy content (or without doing any work on it).

A nozzle is a device which incresases the velocity of the fluid. A converging-diverging nozzle is type of it in which the cross sectional area first decreases and then increases. Since, the mass is conserved, we shall expect the product of density, velocity and cross sectional area to be constant. Thus, changing the cross sectional area would change the velocity. This is as simple in case of incompressible flows as we assume the density to remain constant. However, in high speed flows and in most gases, there is a significant change in density as the conditions change which makes it necessary to consider the comnpressible nature of such flows. As we can feel, reducing the pressure at the output side would force more mass to flow or increase the velocity of the fluid. But in a certain range of output pressures, we observe a phenomenon called *shock*. The pressure initially drops and suddenly increases irreversably in the diverging part to match the output pressure at the exit. In the follwing sections, I shall discuss an attempt to observe the shocks using CFD techniques.

II. Assumptions

To simplify the analysis, I make the following assumptions:

- Since we are concerned with the variation of flow properties along the length of the nozzle, I assume the properties to be constant for any cross-section. However, the cross sectional area changes along the length which makes the problem a quasi-one dimensional one.
- I assume the fluid to be a gas and it will obey the ideal gas laws. (Since gases in general have low viscosity, it further establishes the first assumption)
- No presence of any dissipative effects or natural viscosity.

III. Governing Relations

In the formulation of the governing equation, non-dimensional quantities will be used. The non-dimensional density is defined as $\rho' = \frac{\rho}{\rho_o}$. The non-dimensional temperature is defined as $T' = \frac{T}{T_o}$. The non-dimensional pressure is defined as $P' = \frac{P}{P_o}$. The non-dimensional cross-sectional area is defined as $A' = \frac{A}{A^*}$. The non-dimensional time is defined as $t' = \frac{t}{L/a_o}$. The non-dimensional distance is defined as $x' = \frac{x}{L}$. Where, the 'o' subscriped parameters are the values of the respective parameter at the inlet (assumed to be stagnation property). A^* is the area at the throat. The term L can be any reference length. In my case, I assume the L such the the non-dimensional length varies from 0 to 3 (This is done so that the results are comparable to one provided in the text I am reffering to). V' will be equal to V/a_o, where a is the speed of sound.

A. Equations

The following are the governing relations:
Mass conservation-

$$\frac{\partial(\rho' A')}{t'} + \frac{\partial(\rho' A' V')}{x'} = 0 \tag{2}$$

Momentum equation-

$$\frac{\partial(\rho' A')}{t'} + \frac{\partial[(\rho' A' V'^2) + (1/2)p' A']}{x'} = (1/\gamma)p' \frac{\partial(A')}{x'} \tag{3}$$

Energy conservation-

$$\frac{\partial[\rho'(\frac{e'}{\gamma-1} + (\gamma/2)V'^2)A']}{\partial t'} + \frac{\partial[\rho'(\frac{e'}{\gamma-1} + (\gamma/2)V'^2)A'V' + p'A'V']}{\partial x'} = 0 \tag{4}$$

B. Simplification

To simplify the relations in the previous subsection, the following relations are defined:

$$U_1 = \rho' A' \tag{5}$$

$$U_2 = \rho' A' V' \tag{6}$$

$$U_3 = \rho'(\frac{e'}{\gamma - 1} + (\gamma/2)V'^2)A' \tag{7}$$

$$F_1 = \rho' A' V' \tag{8}$$

$$F_2 = \rho' A' V'^2 + \frac{1}{\gamma} p' A' \tag{9}$$

$$F_3 = \rho'(\frac{e'}{\gamma - 1} + (\gamma/2)V'^2)V' A' + p' A' V' \tag{10}$$

$$J_2 = \frac{1}{\gamma} p' \frac{\partial A'}{\partial x'} \tag{11}$$

C. Reduced Equations

Using the relations in the previous two subsections, the following relations are deduced:

$$\frac{\partial U_1}{\partial t'} = -\frac{\partial F_1}{\partial x'} \tag{12}$$

$$\frac{\partial U_3}{\partial t'} = -\frac{\partial F_2}{\partial x'} + J_2 \tag{13}$$

$$F_3 = \rho'(\frac{e'}{\gamma - 1} + (\gamma/2)V'^2)V' A' + p' A' V' \tag{14}$$

The the equations (12), (13) and (14) will be discretized. The equations (5) to (14) can be analysed to establish relations between Us and Fs. Though such relations are not explicitly described in this report, I shall use those in the code.

IV. Method

A. The MacCormac Scheme

For solving the equations, I shall use the MacCormac method. It is a two-step predictor-corrector method. In the predictor, the parital time derivative of the parameter is determined using the predicted values of all the variables. Now this derivative is used to find the new values of the parameter. Again using the new parameter value, a new time derivative is determined. Using the average of the new and the old derivative, the corrected value of the parameter is determined. For equation (12):
Predictor step-

$$(\frac{\partial U_i}{\partial t'})_t = -\frac{F^t_{i+1} - F^t_i}{\Delta x'} \tag{15}$$

$$U^*_i = U^t_i + \Delta t'(\frac{\partial U_i}{\partial t'})_t \tag{16}$$

F^*s are calculated from the U^*s
Corrector-

$$(\frac{\partial U_i}{\partial t'})^*_t = -\frac{F^{*t}_i - F^{*t}_{i-1}}{\Delta x'} \tag{17}$$

$$U^{t+1}_i = U^n_i + \Delta t' * 0.5[(\frac{\partial U_i}{\partial t'})_t + (\frac{\partial U_i}{\partial t'})^*_t] \tag{18}$$

The scheme is second accurate in both space and time.

B. Stability and Time Step

Since the method is an explicit one, we can expect it to have a conditional stability. For stability, the absolute Courant number should be less than 1. The Courant number can be represented as:

$$C = \frac{\Delta t(a + V)}{\Delta x} \tag{19}$$

In non-dimensional case, $a = \sqrt{T'}$ and $V = V'$. The book by Anderson suggests using C=0.5 for better results. Since the value of V and a differs at each point, we should use the minimum value of time step among the values for all points.

C. Shock Capturing

At the region of shock, the derivative is very large and sharp changes occur. In order to smooth out the changes and reduce the risk of infinite values, we can add a dissipative term in the following manner [Artificial Viscosity]:

$$S^t_i = \frac{C|(p')^{t'}_{i+1} - 2(p')^{t'}_i + (p')^{t'}_{i-1}|}{(p')^{t'}_{i+1} - 2(p')^{t'}_i + (p')^{t'}_{i-1}}(U^{t'}_{i+1} - U^{t'}_i + U^{t'}_{i-1}) \tag{20}$$

Where C is a constant and its value is 0.2 in our case [from the reference of Anderson's book] This term is computed for each case and added while computing the new value of Us.

D. Shape of the Nozzle

In my case, the cross sectional area is varied parabolically with symmetric distribution. The following distribution is used:

$$A' = 1 + 2.2(x' - 1.5)^2 \tag{21}$$

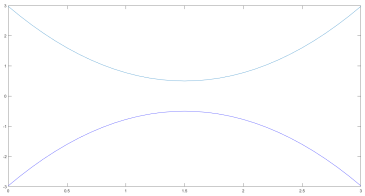

Fig. 1. Representation of Nozzle

E. Boundary Values

At the inlet, the values of density, pressure and temperature (all non-dimensional) are maintained as 1. At the outlet, the value of p' is fixed whose effect we analyse at different values. Thus for subsonic outlet:

$$(U1)_{i=N(end)} = 2U1_{i=N-1} - U1_{i=N-2} \tag{22}$$

$$(U2)_{i=1} = 2U1_{i=2} - U1_{i=3} \tag{23}$$

$$(U2)_{i=N(end)} = 2U2_{i=N-1} - U2_{i=N-2} \tag{24}$$

$$(U3)_{i=1} = U1_{i=1} * (1/(\gamma - 1)) + (\gamma/2)(V)'^2) \tag{25}$$

$$(U3)_{i=N} = \frac{[p'_{(outlet)} * A'_{(outlet)}]}{(\gamma - 1)} U2_{i=N} * V'_{i=N} \tag{26}$$

when i varies from 1 to N

V. OBSERVATIONS

The result plots are as shown from next page:

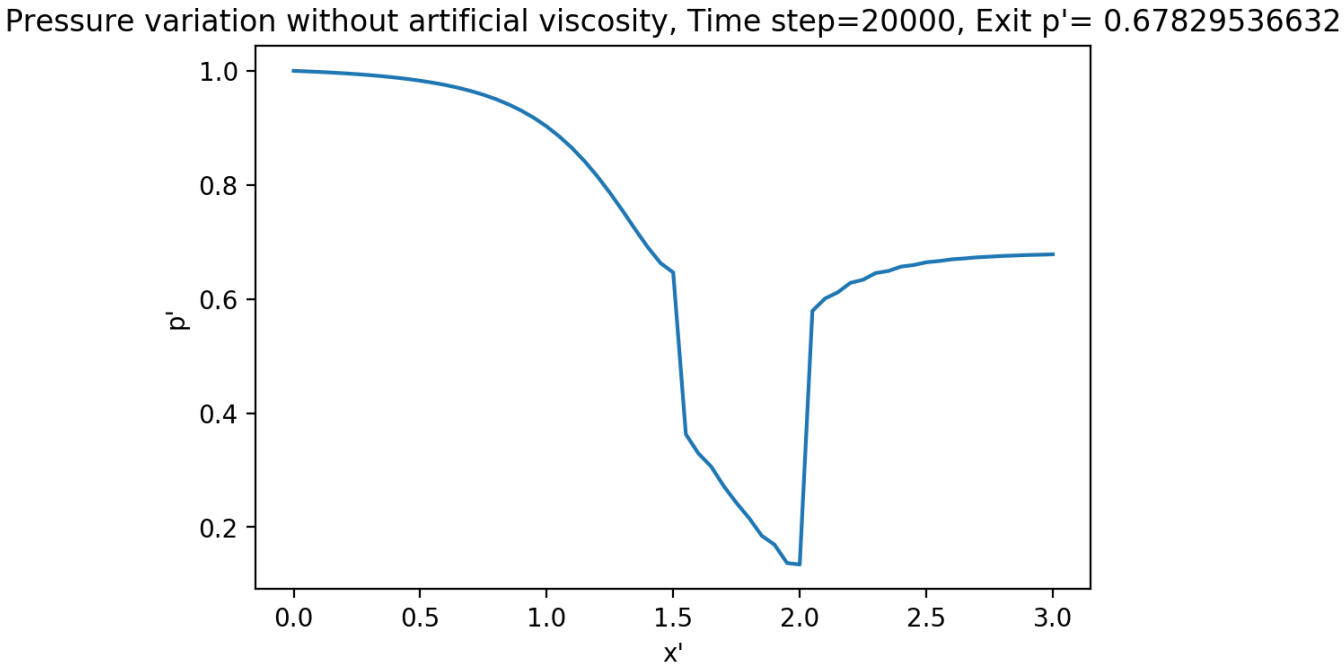

Fig. 2. Pressure variation without artificial viscosity

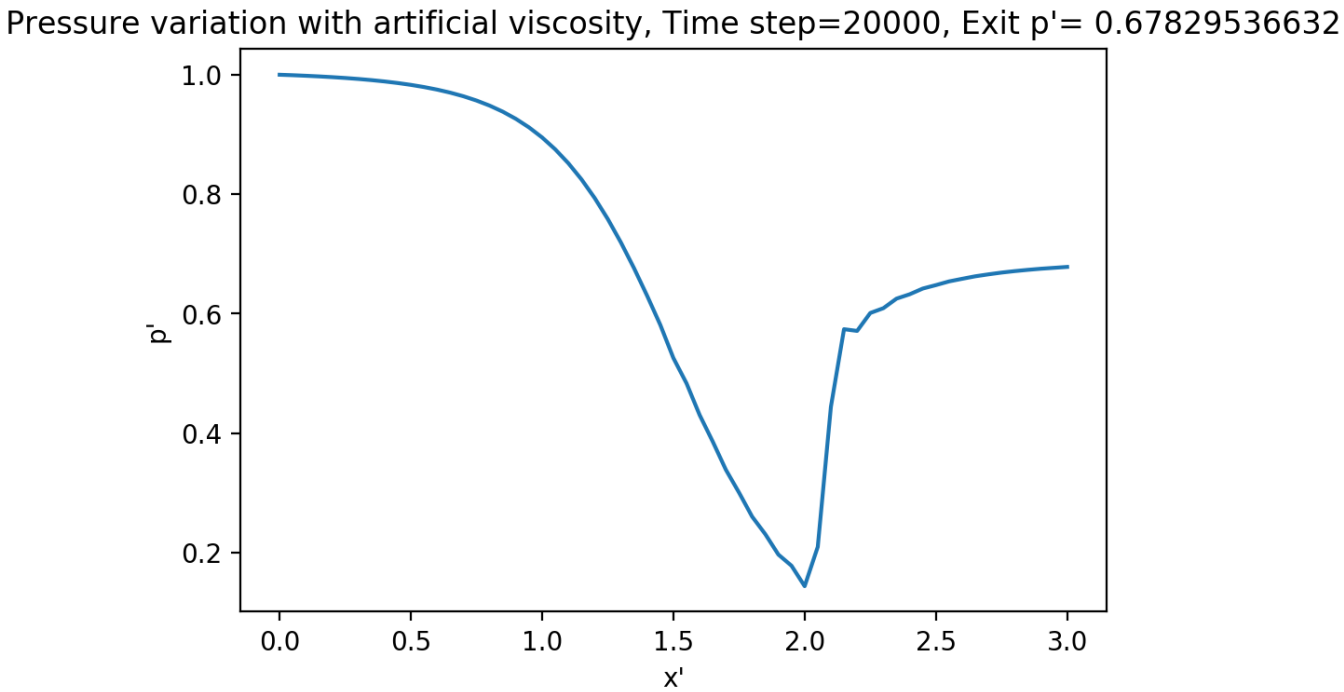

Fig. 3. Pressure variation with artificial viscosity

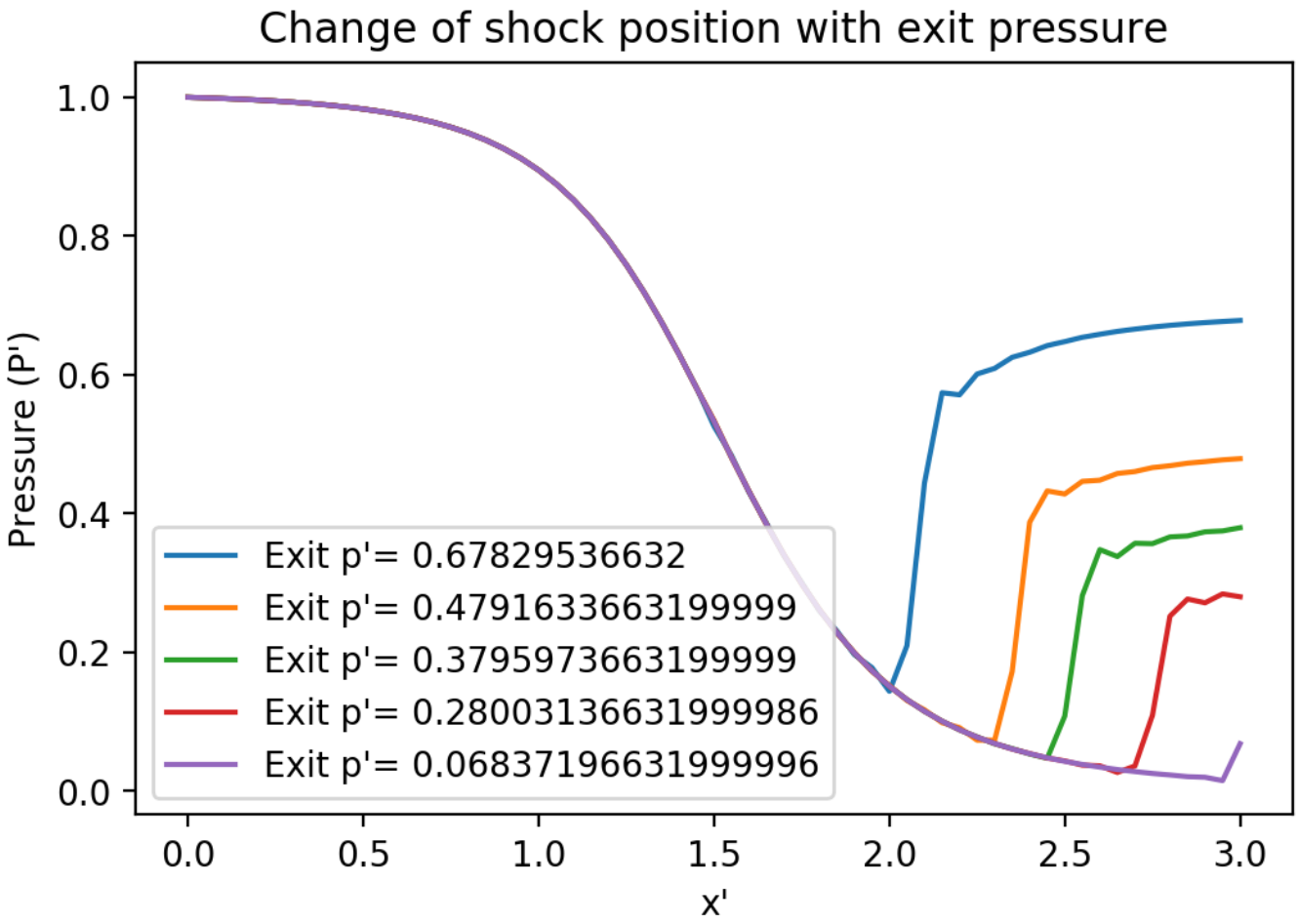

Fig. 4. Movement of Shock with change in outlet pressure

.

.

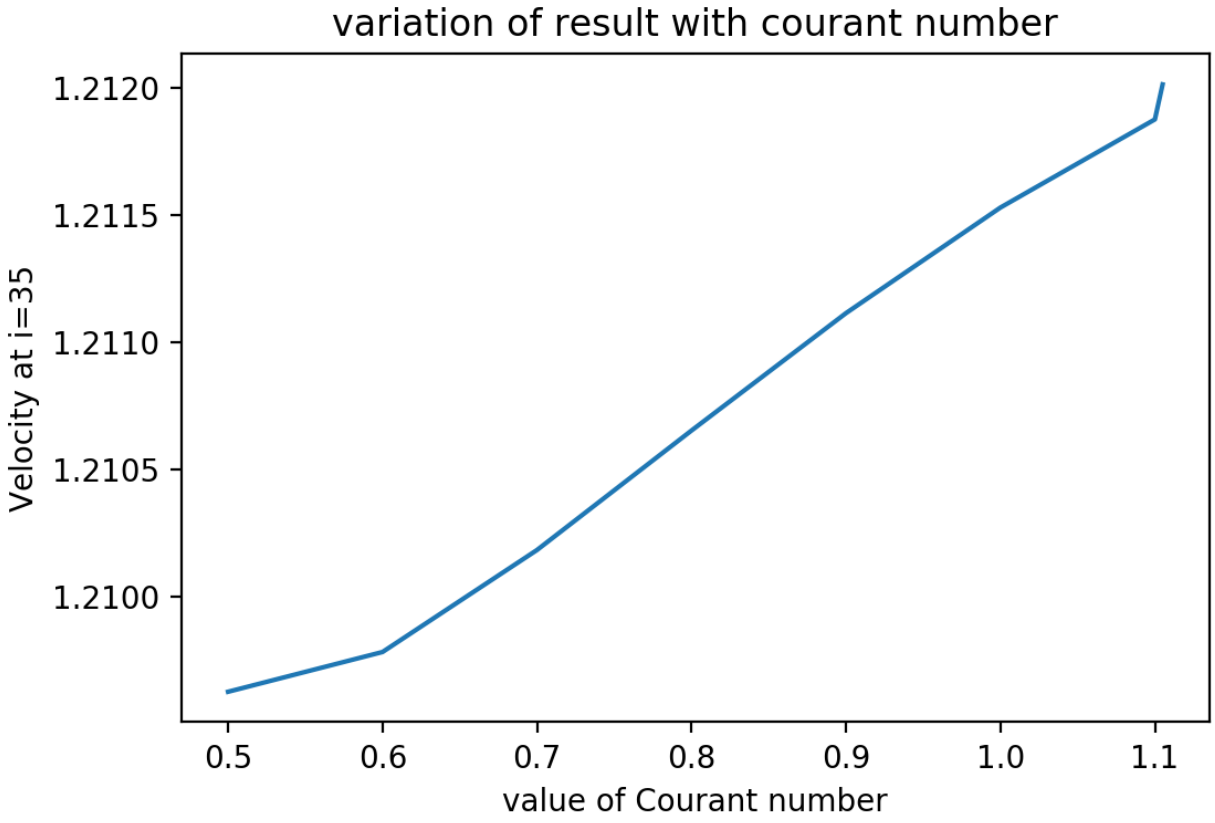

Fig. 5. Variation of result with Courant number

.

.

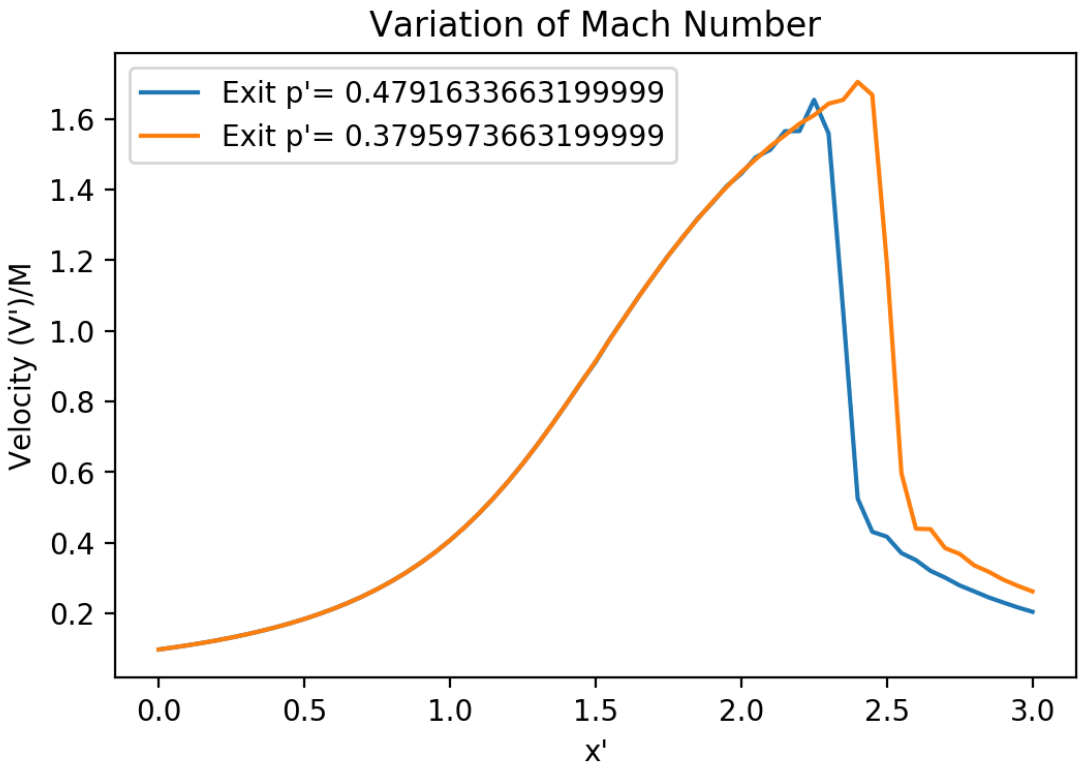

Fig. 6. Variation of Mach Number

VI. Results and Conclusion

From the observations, we can infer the following conclusions:

- The addition of the dissipative term makes the variations smoother. While running the code, I found that after adding the term, the scheme becomes stable at those back pressures at which it was blowing up without the dissipative terms. [Fig. 2 and Fig. 3]
- The Courant number has a significant impact on the scheme stability. As can be seen in the observation section, the value of velocity starts to increase as the Courant number increases and becomes infinity (unstable) after 1.105. [Fig. 5]
- As expected, with the decrease in the outlet (back) pressure, the point of shock shifts toward the end. At very low outlet pressure, the whole flow become supersonic and shock occurs outside the nozzle [Fig. 4]
- The Mach number=1 occurs only at the throat [Fig.6]

VII. Future Work

- Try with different schemes
- Observe 2-Dimensional variation
- Include viscous nature of fluid

VIII. Reference Material

John Anderson, Computational Fluid Dynamics- The Basics with Applications [Chapter-7]

#APPENDIX#

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 22 18:39:36 2019

@author: Neelotpal
"""

import numpy as np
import matplotlib.pyplot as plt




######################## initialisations ########################################
L=3
dx=0.05
N=(int(L/dx)+1)
gamma=1.4
R=8.314



denp=np.zeros(N)
denp_temp=np.zeros(N)
pp=np.zeros(N)
pp_temp=np.zeros(N)
vp=np.zeros(N)
Tp=np.zeros(N)
vp_temp=np.zeros(N)
Tp_temp=np.zeros(N)
ep=Tp[:]

U1=np.zeros(N)
U1_temp=np.zeros(N)
U2=np.zeros(N)
U2_temp=np.zeros(N)
U3=np.zeros(N)
U3_temp=np.zeros(N)
F1=np.zeros(N)
F2=np.zeros(N)
F3=np.zeros(N)
J2=x=np.zeros(N)
S1=np.zeros(N)
S2=np.zeros(N)
S3=np.zeros(N)

diff_n_U1=np.zeros(N)
diff_np1_U1=np.zeros(N)
diff_n_U2=np.zeros(N)
diff_np1_U2=np.zeros(N)
diff_n_U3=np.zeros(N)
diff_np1_U3=np.zeros(N)
Ap=np.zeros(N)
diff_Apf=np.zeros(N)
diff_Apb=np.zeros(N)

temp=[]

#initial distribution of the density and temperature [Non dimensional]
for i in range(N):
    if (i*dx<=0.5):
        denp[i]=1.
        Tp[i]=1.
    elif (i*dx<=1.5):
       denp[i]=1.0-0.366*(i*dx-0.5)
       Tp[i]=1.0-0.167*(i*dx-0.5)
    elif (i*dx<=2.1):
        denp[i]=0.634-0.702*(i*dx-1.5)
        Tp[i]=0.833-0.4908*(i*dx-1.5)
    else:
        denp[i]=0.5892+(0.10228)*(i*dx-2.1)
        Tp[i]=0.93968+0.0622*(i*dx-2.1)
```

```
#Area variation with axial distance
for i in range(N):
    Ap[i]=1+2.2*((i*dx-1.5)**2)


#Area forward and and backward derivative
for i in range(N):
    diff_Apf[i]=((1+2.2*(((i+1)*dx-1.5)**2))-(Ap[i]))/dx

for i in range(N):
    diff_Apb[i]=((Ap[i])-(1+2.2*(((i-1)*dx-1.5)**2)))/dx


pend=denp[N-1]*Tp[N-1]   #pressure assuming the fluid obeys ideal gas law


vp=0.59/(denp*Ap) ##assume initial mass flow rate and determine the initial velocity

#Solution vectors
U1=U1_temp=denp*Ap
U2=U2_temp=U1*vp
U3=U1_temp=denp*((Tp/(gamma-1))+(gamma/2)*vp*vp)*Ap

denp=U1/Ap
pp=denp*Tp


#iterate over 'time' time steps
for time in range(20000):

#######################Predictor Steps#########################################
    F1=U2
    F2=((U2*U2/U1)+((gamma-1)/gamma)*(U3-(gamma/2)*(U2*U2/U1)))
    F3=(gamma*U2*U3/U1)-((gamma*(gamma-1)/2)*(U2*U2*U2/(U1*U1)))
    J2=((gamma-1)/gamma)*(U3-(gamma/2)*(U2*U2/U1))*diff_Apf/Ap

    #determine the predicted derivative of Us:
    for i in range(1,N-1):
        diff_n_U1[i]=-(F1[i+1]-F1[i])/dx
        diff_n_U2[i]=(-(F2[i+1]-F2[i])/dx)+J2[i]
        diff_n_U3[i]=-(F3[i+1]-F3[i])/dx
    dt_all=np.zeros(N-2)

    for i in range(1,N-1): #optimise time step
        dt_all[i-1]=0.5*dx/(vp[i]+(Tp[i]**0.5))
        dt=min(dt_all)

    #Dissipative terms to smoothen out the sharp changes
    for i in range(1,N-1):
        S1[i]=0.2*(abs(pp[i-1]-2*pp[i]+pp[i+1])/(pp[i-1]+2*pp[i]+pp[i+1]))*(U1[i+1]-2*U1[i]+U1[i-1])
        S2[i]=0.2*(abs(pp[i-1]-2*pp[i]+pp[i+1])/(pp[i-1]+2*pp[i]+pp[i+1]))*(U2[i+1]-2*U2[i]+U2[i-1])
        S3[i]=0.2*(abs(pp[i-1]-2*pp[i]+pp[i+1])/(pp[i-1]+2*pp[i]+pp[i+1]))*(U3[i+1]-2*U3[i]+U3[i-1])

    #predicted values of the Us
    U1_temp=U1+dt*diff_n_U1+S1
    U2_temp=U2+dt*diff_n_U2+S2
    U3_temp=U3+dt*diff_n_U3+S3


    U1_temp[N-1]=2*U1_temp[N-2]-U1_temp[N-3]
    U2_temp[0]=2*U2[1]-U2_temp[2]
    U2_temp[N-1]=2*U2_temp[N-2]-U2_temp[N-3]
    vp_temp=U2_temp/U1_temp
    U3_temp[0]=U1_temp[0]*((1/(gamma-1))+(gamma/2)*vp_temp[0]*vp_temp[0])
    U3_temp[N-1]=(pend*Ap[N-1]/(gamma-1))+((gamma/2)*U2_temp[N-1]*vp_temp[N-1])

    Tp_temp=(gamma-1)*((U3_temp/U1_temp)-(gamma/2)*vp_temp*vp_temp)
    denp_temp=U1_temp/Ap
    pp_temp=denp_temp*Tp

    F1=U2_temp
```

```
    F2=((U2_temp*U2_temp/U1_temp)+((gamma-1)/gamma)*(U3_temp-(gamma/2)*(U2_temp*U2_temp/U1_temp)))
    F3=(gamma*U2_temp*U3_temp/U1_temp)-((gamma*(gamma-1)/2)*(U2_temp*U2_temp*U2_temp/(U1_temp*U1_temp)))
    J2=((gamma-1)/gamma)*(U3_temp-(gamma/2)*(U2_temp*U2_temp/U1_temp))*diff_Apb/Ap



################################Corrector steps#################################
    for i in range(1,N-1):
        diff_np1_U1[i]=-(F1[i]-F1[i-1])/dx
        diff_np1_U2[i]=(-(F2[i]-F2[i-1])/dx)+J2[i]
        diff_np1_U3[i]=-(F3[i]-F3[i-1])/dx

    for i in range(1,N-1):

S1[i]=0.2*(abs(pp_temp[i-1]-2*pp_temp[i]+pp_temp[i+1])/(pp_temp[i-1]+2*pp_temp[i]+pp_temp[i+1]))*(U1_temp[i+1]-2*U1_temp[i]+U1_temp[i-1])


S2[i]=0.2*(abs(pp_temp[i-1]-2*pp_temp[i]+pp_temp[i+1])/(pp_temp[i-1]+2*pp_temp[i]+pp_temp[i+1]))*(U2_temp[i+1]-2*U2_temp[i]+U2_temp[i-1])


S3[i]=0.2*(abs(pp_temp[i-1]-2*pp_temp[i]+pp_temp[i+1])/(pp_temp[i-1]+2*pp[i]+pp_temp[i+1]))*(U3_temp[i+1]-2*U3_temp[i]+U3_temp[i-1])

    for i in range(1,N-1):
        U1[i]=U1[i]+0.5*dt*(diff_n_U1[i]+diff_np1_U1[i])+S1[i]
        U2[i]=U2[i]+0.5*dt*(diff_n_U2[i]+diff_np1_U2[i])+S2[i]
        U3[i]=U3[i]+0.5*dt*(diff_n_U3[i]+diff_np1_U3[i])+S3[i]

    U1[N-1]=2*U1[N-2]-U1[N-3]
    U2[0]=2*U2[1]-U2[2]
    U2[N-1]=2*U2[N-2]-U2[N-3]
    vp=U2/U1
    U3[0]=U1[0]*((1/(gamma-1))+(gamma/2)*vp[0]*vp[0])
    U3[N-1]=(pend*Ap[N-1]/(gamma-1))+((gamma/2)*U2[N-1]*vp[N-1])
    Tp=(gamma-1)*((U3/U1)-(gamma/2)*vp*vp)
    temp.append(vp[39])

denp=U1/Ap
pp=denp*Tp
vect=np.arange(0,N*dx,dx)
plt.figure(1,figsize=(40,20),dpi=72)

plt.plot(vect,pp) #pressure distribution
plt.xlabel("x'")
plt.ylabel("p'")
plt.title("Pressure variation with artificial viscosity, Time step=20000, Exit p'= "+str(pend))


plt.show()
```