Computing the Mach contour of Compression Corner in a supersonic flow by using Method of Characteristic

Ranjithkumar B

I. Problem Definition

The Compression corner has equivalent angle on both sides with inlet Mach number 3.5. The inlet and outlet the flow is turn with respect to the angle of the corner.

SOLUTION:

Given data

$$M_1 = 3.5$$
$$P_1 = 15\ kPa$$
$$\theta = 6^0$$

The solution is done by following procedure

For the test case, The number of nodes consider as 10 on the wall and 15 on the height, therefore the total number of nodes in this problem is

$$N_t = N_w^2 + (N_i - 1)^2$$

The Top Boundary nodes and the Bottom boundary nodes were calculated by using the bellow formulae, here I is denote for the Each inlet node numbers

$$btm - wall = I * (2 \times N(inlet) - 1)$$
$$top - wall = btm_wall[previous] + (N - 1)$$

II. Formula and Procedure

A. Computing Mach number

The Reimann invarients are calculated by using the inlet data for the inlet Nodes. After computing the inlet Node values depend on the inlet values we can continuously computing the succesive Nodes. The initial K_1 *and* K_2 were calculated by intlet angle and the Mach number.

$$K_1 = \nu + \theta$$
$$K_2 = \nu - \theta$$

The Reimann invarients won't change until it get hit and refect by the wall or the boundary. Therefore we can compute Prandtl-Meyer expansion function (ν) and the angles (θ) by using Reimann invarients.

$$\nu = \frac{K_1 + K_2}{2}$$
$$\theta = \frac{K_1 - K_2}{2}$$

The Bottom wall have only K_1 and Top wall have only the K_2 and we know the angle of the top and bottom surface. With these information we can calculate P-M function on that Node.

$$\text{for bottom wall}$$
$$\nu = K_1 - \theta$$
$$\text{for top wall}$$
$$\nu = K_2 + \theta$$

The Mach numbers at each points were calculated by using Prandtl-Meyer expansion function relation

$$\nu(M) = \sqrt{\frac{\gamma + 1}{\gamma - 1}} tan^{-1} \sqrt{\frac{\gamma - 1}{\gamma + 1} \left(M^2 - 1\right)} - tan^{-1} \sqrt{M^2 - 1}$$

Now we know the Angles and Mach number of each Nodes.

B. Computing Location

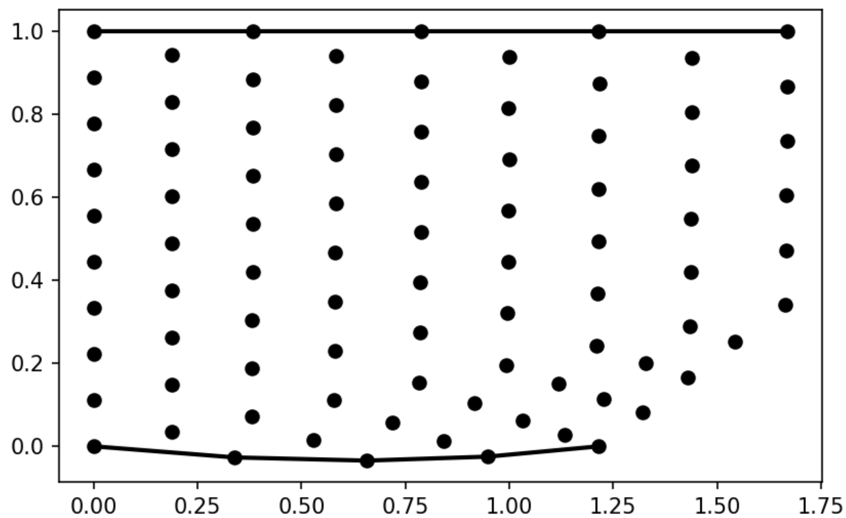
We know the Mach number of each Nodes, with that value we can calculate the Mach angle ($\mu = \arcsin \frac{1}{M}$). Using Mach angle and the flow angle we can compute the X and Y location.

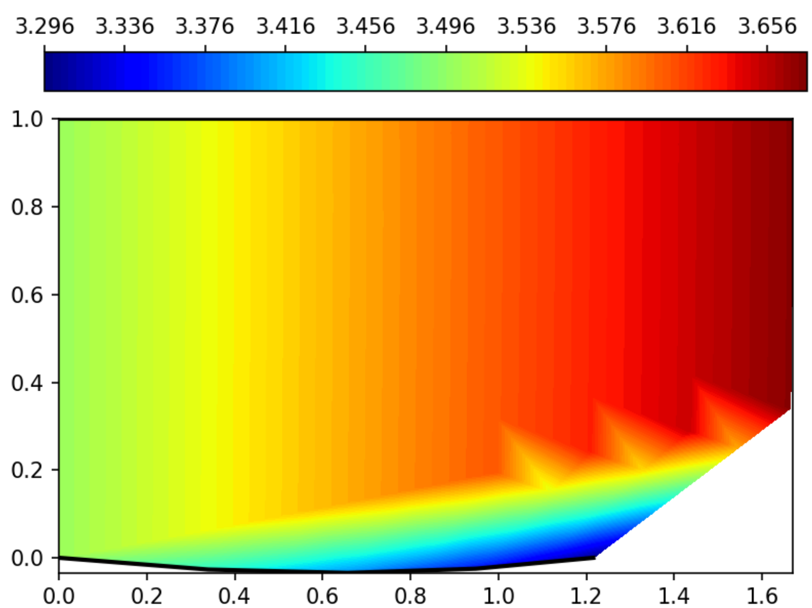
$$S_1 = \frac{tan(\theta - \mu)_A + tan(\theta - \mu)_B}{2}$$
$$S_2 = \frac{tan(\theta + \mu)_A + tan(\theta + \mu)_B}{2}$$
$$y_D = y_A + (x_D - x_A)S_1$$
$$y_D = y_B + (x_D - x_B)S_2$$
$$x_D = \frac{(S_2 x_B - S_1 x_A) + (y_A - y_B)}{S_2 - S_1}$$

Then the Top and Bottom wall Nodes location is calculated by using the slope given bellow.

$$\frac{dy}{dx}_A = \tan(\theta - \mu)_A$$
$$\frac{dy}{dx}_B = \tan(\theta - \mu)B$$

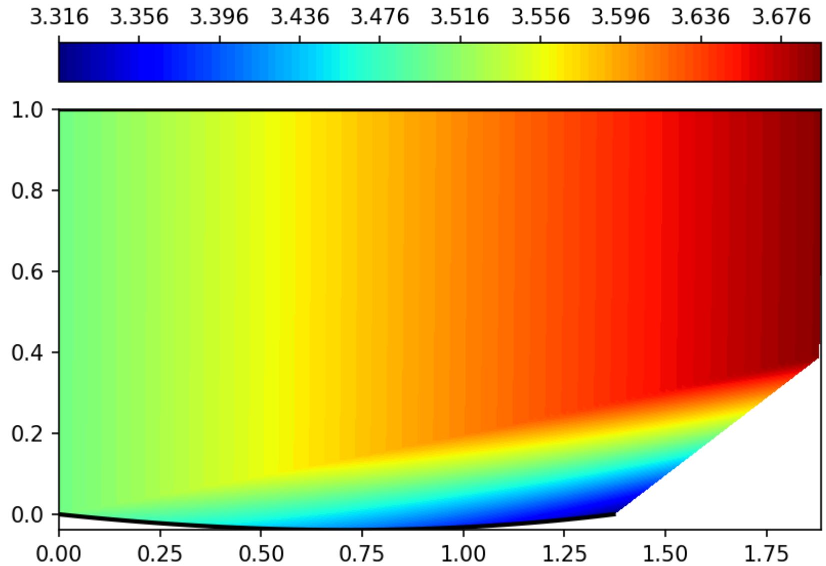
III. Results:


1.0
0.8
0.6
0.4
0.2
0.0
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75


3.296 3.336 3.376 3.416 3.456 3.496 3.536 3.576 3.616 3.656
1.0
0.8
0.6
0.4
0.2
0.0
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6

IV. Concultion

1) The given problem is a gradual compression corner, taken the top boundary to wall distance is slightly higher.
2) The reason of higher inlet , the flow highly trun near the wall after going up the flow almost get straight and smooth. The infuluence of the wall will be less when we go higher distance.
3) Increaing the inlet and wall node number we can get the smooth solution. I attached the solution bellow for inlet Node 100 and wall node 50. and also attached the python code in Appendix-A.



A. Appendix - Python code

```python
#!/bin/python3

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#--------------------------------------------------------------------------------
# initial conditions

# specific heat constant of air
g=1.4

# inlet Mach number
Mi= 3.5

# Inlet Node points
Ni= 10

# Total wall points
Nw= 5

# Theta values for Bottom , top and inlet Nodes
bottom_t=np.radians(np.linspace(-6,0,Nw))
top_t=np.radians(0)
inlet_t=np.radians(np.linspace(-6,0,Ni))

# Height of the inlet
H=1

#--------------------------------------------------------------------------------
# Prandtl meyer function

def fun_Nu(M):
    a=(g+1)/(g-1)
    b=M**2-1
    nu=np.sqrt(a)*np.arctan(np.sqrt(b/a))-np.arctan(np.sqrt(b))

    return nu

#--------------------------------------------------------------------------------
# Mach number getting from P-M function
    # using Bi-section Method

def fun_M(nu):
    a=0.1
    b=10

    while True:
        c=(a+b)/2
        res=fun_Nu(c)-nu
        if res < 0:
            a=c
        else:
            b=c

        c=(a+b)/2
        res=fun_Nu(c)-nu
        # print(c,res)
        if np.abs(res)<1e-6:
            break

    return c

#--------------------------------------------------------------------------------
# Total number of points
Nt=(Ni*Nw)+((Ni-1)*(Nw-1))
```

```
# Needed array for Node
Node=np.zeros(Nt,dtype=int)
btm=np.zeros(Nw,dtype=int)
top=np.zeros(Nw,dtype=int)
inlet=np.zeros(Ni,dtype=int)
internal=np.zeros(((Nt-1)-(2*Nw)),dtype=int)

# Total Node
for i in range(Nt):
    Node[i]=i

# Inlet Node
for i in range(Ni):
    inlet[i]=i

# Top and Bottom Node
for i in range(Nw):
    btm[i]=i*(2*Ni-1)
    top[i]=btm[i]+(Ni-1)

# internal Nodes
count=0
for i in range(Nt):
    if i in inlet or i in btm or i in top:
        continue
    else:
        internal[count]=i
        count=count+1

#-------------------------------------------------------------------------------
# necessary Arrays
M=np.zeros(Nt)
K1=np.zeros(Nt)
K2=np.zeros(Nt)
nu=np.zeros(Nt)
theta=np.zeros(Nt)
Mu=np.zeros(Nt)

count=1
# Computing Mach number
for i in range(Nt):
    if i in inlet:
        M[i]=Mi
        theta[i]=inlet_t[i]
        nu[i]=fun_Nu(M[i])
        K1[i]=nu[i]+theta[i]
        K2[i]=nu[i]-theta[i]
        Mu[i]=np.arcsin(1/M[i])
    elif i in btm:
        theta[i]=bottom_t[count]
        K1[i]=K1[i-(Ni-1)]
        K2[i]=K1[i]-(2*theta[i])
        nu[i]=(K1[i-(Ni-1)]+K2[i])/2
        M[i]=fun_M(nu[i])
        Mu[i]=np.arcsin(1/M[i])
        count=count+1
    elif i in top:
        theta[i]=top_t
        K2[i]=K2[i-Ni]
        K1[i]=(2*theta[i])+K2[i]
        nu[i]=(K1[i]+K2[i-Ni])/2
        M[i]=fun_M(nu[i])
        Mu[i]=np.arcsin(1/M[i])
    else:
        theta[i]=(K1[i-(Ni-1)]-K2[i-Ni])/2
        nu[i]=(K1[i-(Ni-1)]+K2[i-Ni])/2
        K1[i]=nu[i]+theta[i]
```

```python
        K2[i]=nu[i]-theta[i]
        M[i]=fun_M(nu[i])
        Mu[i]=np.arcsin(1/M[i])

#------------------------------------------------------------------------------
# Computing the location
x=np.zeros(Nt)
y=np.zeros(Nt)

# height between the two consicutive points in the inlet
h = H/(Ni-1)
for i in range(Nt):
    if i in inlet:
        x[i]=0
        y[i]=i*h
    elif i in btm:
        # only right running curve present
        S1=np.tan(theta[i-(Ni-1)]-Mu[i-(Ni-1)])
        x[i]=(y[i-(Ni-1)]-x[i-(Ni-1)]*S1)/(np.tan(theta[i])-S1)
        y[i]=x[i]*np.tan(theta[i])
    elif i in top:
        S2=np.tan(theta[i]+Mu[i])
        x[i]=(y[i-Ni]-x[i-Ni]*S2-H)/(np.tan(theta[i])-S2)
        y[i]=(x[i]*np.tan(theta[i]))+H
    else:
        S1=(np.tan(theta[i]+Mu[i])+np.tan(theta[i-Ni]+Mu[i-Ni]))/2
        S2=(np.tan(theta[i]-Mu[i])+np.tan(theta[i-(Ni-1)]-Mu[i-(Ni-1)]))/2
        x[i]=((S2*x[i-(Ni-1)]-S1*x[i-Ni])+(y[i-Ni]-y[i-(Ni-1)]))/(S2-S1)
        y[i]=y[i-(Ni)]+((x[i]-x[i-(Ni)])*S1)
#------------------------------------------------------------------------------
# DataFrame

df=pd.DataFrame(np.transpose([M,K1,K2,theta,nu,Mu,x,y]),
        columns=["M","K1","K2","Theta","Nu","Mu","X","Y"])
print(df)

#------------------------------------------------------------------------------
# plotting section
Yt=np.zeros(Nw)
Xt=np.zeros(Nw)
Xb=np.zeros(Nw) ; Yb=np.zeros(Nw)
for i in range(Nw):
    Xt[i]=x[top[i]]
    Yt[i]=y[top[i]]

for i in range(Nw):
    Xb[i]=x[btm[i]]
    Yb[i]=y[btm[i]]

plt.figure()
plt.plot(df["X"],df["Y"],'ko')
plt.plot(Xt,Yt,'-k',linewidth=2)
plt.plot(Xb,Yb,'-k',linewidth=2)
plt.axis("image")
plt.savefig("Grid.png",dpi=150)
plt.show()

plt.figure()
plt.tricontourf(df["X"],df["Y"],df["M"],100,cmap="jet")
plt.plot(Xt,Yt,'-k',linewidth=2)
plt.plot(Xb,Yb,'-k',linewidth=2)
plt.axis("image")
plt.colorbar(location='top')
plt.savefig("Contour.png",dpi=150)
plt.show()
```