Supersonic Right Circular Cone at Zero Angle of Attack

Dr. J. V. Lassaline

March 13, 2009

For supersonic flow past a cone at zero angle of attack (see Fig. 1), we can apply the governing equations and the assumption of irrotationality to develop the non-dimensional Taylor-Maccoll ODE [1]

$$\frac{\gamma-1}{2}\left(1-v_r^2-{v_r'}^2\right)\left(2v_r+v_r'\cot\theta+v_r''\right)-v_r'\left(v_rv_r'+v_r'v_r''\right)=0 \tag{1}$$

where v_r is the radial velocity component and $v_\theta = v_r' = \frac{dv_r}{d\theta}$ is the polar velocity component. Note that this equation applies from the shock θ_s to the surface of the cone θ_c. Solving for $v_r(\theta)$, the remaining flow properties can be determined using the isentropic relations. The velocity components are related to the local Mach number through

$$V^2 = v_r^2 + v_\theta^2 \tag{2}$$

$$V = \frac{1}{\sqrt{1+\frac{2}{(\gamma-1)M^2}}} \tag{3}$$

We can employ a numerical solver, such as the `ode15s` function available in MATLAB, to solve the Taylor-Maccoll ODE for $v_r(\theta)$. In this case, one can determine v_r and $v_\theta = v_r'$ immediately behind the shock as a function of M_∞ using the oblique shock relations. Note that $\beta = \theta_s$ and $\theta = \delta$ in the β-θ-M oblique shock relation. The values immediately behind the shock form one boundary condition for our ODE, while flow tangency at the cone surface provides another boundary condition. We can numerically solve for the solution to $v_r(\theta)$ by marching the solution from the initial conditions at θ_s until we reach θ_c where $v_\theta = v_r' = 0$. Using this

[1] See Anderson, *Modern Compressible Flow*, 2003.

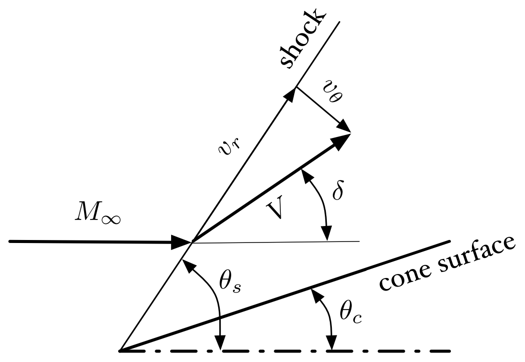

Figure 1: Geometry of the flow past a supersonic cone at zero angle of attack.

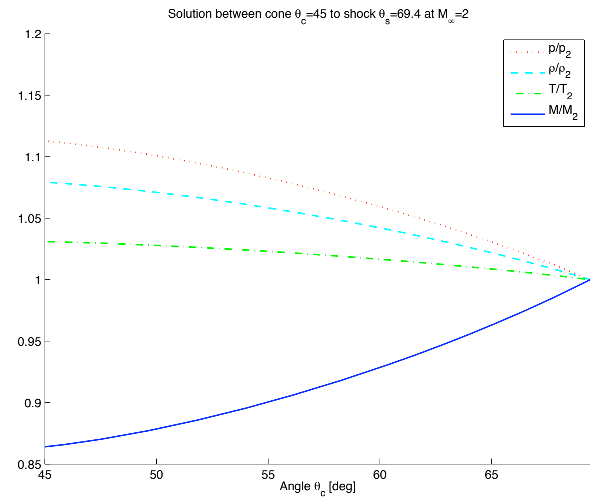

Figure 2: Flow solution for cone normalized relative to flow properties immediately behind the shock.

technique one can indirectly determine the half cone angle θ_c that produces a given shock angle θ_s.

Matlab's family of ODE solvers (eg. `ode15s`) can determine the solution $y(t)$ that satisfies the ODE $y' = f(t, y)$ by marching forward in t from the initial value $y(t_0)$. The ODE $f(t, y)$ can be a scalar ordinary differential equation or a system of ordinary differential equations. In the case of the Taylor-Maccoll equation it is convenient to represent the ODE as a system of ODE with solution vector

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} v_r \\ v_r' \end{bmatrix} \tag{4}$$

such that the ODE can be represented as the following system of differential equations

$$y' = f(\theta, y) = \begin{bmatrix} y_2 \\ \frac{y_2^2 y_1 - \frac{\gamma-1}{2}\left(1-y_1^2-y_2^2\right)(2y_1+y_2\cot(\theta))}{\frac{\gamma-1}{2}\left(1-y_1^2-y_2^2\right)-y_2^2} \end{bmatrix} \tag{5}$$

Note that the new ODE, y', can be expressed entirely in terms of a given y and θ where the second vector component is determined by rearranging Eq. 1 for v_r''.

Using the initial conditions behind the shock $y(\theta_s)$, we can use `ode15s`[2] to march the initial solution from θ_s to 0, stopping the solution early if we detect $y_2 = v_r' = v_\theta = 0$. Using the isentropic relations, one can determine the solution to the flow properties behind the shock, as illustrated for the case where $\theta_c = 45°$ and $M_\infty = 2.0$ in Fig. 2. The solution for shock angle and cone surface Mach number as a function of half cone angle appears in Fig. 3 and 4, respectively, for a range of M_∞. The source code used to produce these figures follows.

[2]There are several ODE solvers with differing accuracy and speed available. In this case, `ode15s` quickly solves a relatively stiff problem.

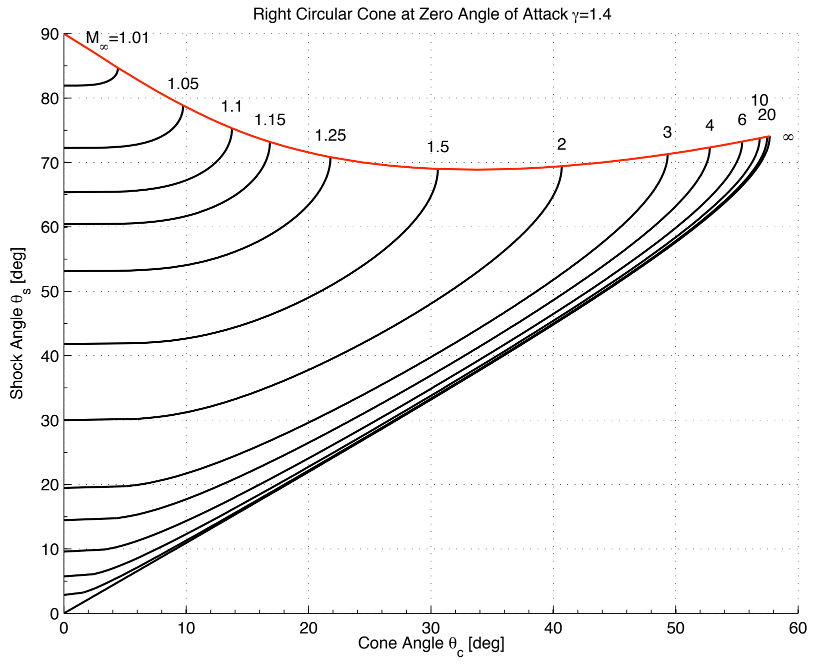

Figure 3: Shock angle θ_s versus half cone angle θ_c.

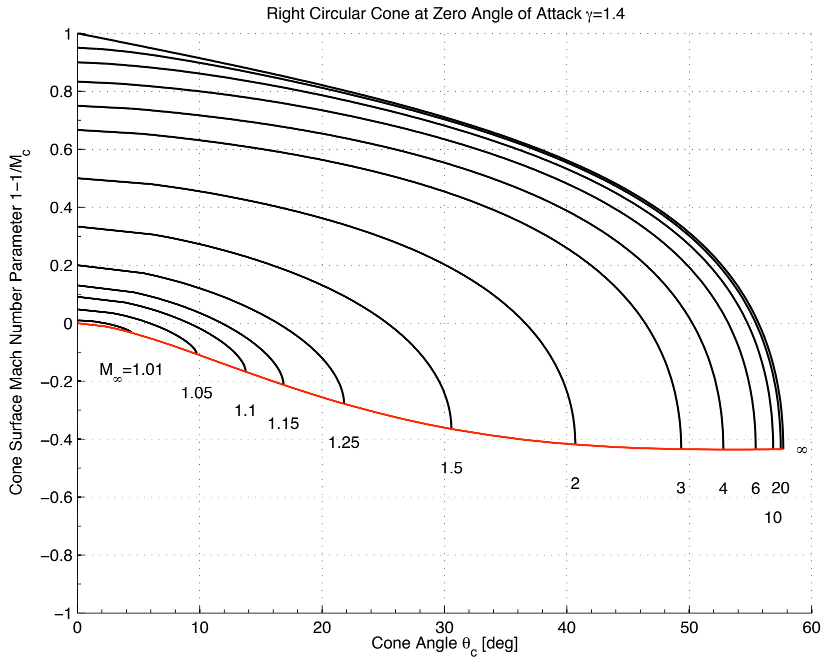

Figure 4: Cone surface Mach number M_c versus half cone angle θ_c.

```
function [thetac,Mc,sol]=solvecone(thetas,Minf,gamma)
% Solves the right circular cone at zero angle of attack in supersonic flow
% thetas - shock angle [degrees]
% Minf   - upstream Mach number
% gamma  - ratio of specific heats

% Convert to radians
thetasr=thetas*pi/180.0;

if (thetasr<=asin(1/Minf))
   thetac=0.0; Mc=Minf;
   return;
end

% Calculate initial flow deflection (delta) just after shock...
delta=thetabetam(thetasr,Minf,gamma);
Mn1=Minf*sin(thetasr);
Mn2=sqrt((Mn1^2+(2/(gamma-1)))/(2*gamma/(gamma-1)*Mn1^2-1));
M2=Mn2/(sin(thetasr-delta));

% All values are non-dimensionalized!
% Calculate the non-dimensional velocity just after the oblique shock
V0=(1+2/((gamma-1)*M2^2))^(-0.5);
% Calculate velocity components in spherical coordinates
Vr0=V0*cos(thetasr-delta);
Vtheta0=-V0*sin(thetasr-delta);
% Calculate initial values for derivatives
dVr0=Vtheta0;
% Assemble initial value for ODE solver
y0=[Vr0;dVr0];

% Set up ODE solver
% Quit integration when vtheta=0
% See: coneevent.m
% Set relative error tolerance small enough to handle low M
options=odeset('Events',@coneevent,'RelTol',1e-5);
% Solve by marching solution away from shock until either 0 degrees or flow
% flow tangency reached as indicated by y(2)==0.
% See cone.m
[sol]=ode15s(@cone,[thetasr 1e-10],y0,options,gamma);
% Check if we have a solution, as ode15s may not converge for some values.
[n,m]=size(sol.ye);
thetac=0.0;
Mc=Minf;
% If ODE solver worked, calculate the angle and Mach number at the cone.
if (n>0 & m>0 & abs(sol.ye(2))<1e-10)
  thetac=sol.xe*180.0/pi;
  Vc2=sol.ye(1)^2+sol.ye(2)^2;
  Mc=((1.0/Vc2-1)*(gamma-1)/2)^-0.5;
end
```

Figure 5: Matlab source code for `solvecone.m`

```
function [dy]=cone2(theta,y,gamma)
% y is a vector containing vr, vr'
% Governing equations are continuity, irrotationality, & Euler's equation.
dy=zeros(2,1);

dy(1)=y(2);
dy(2)=(y(2)^2*y(1)-(gamma-1)/2*(1-y(1)^2-y(2)^2)*(2*y(1)+y(2)*cot(theta)))...
        /((gamma-1)/2*(1-y(1)^2-y(2)^2)-y(2)^2);
```

Figure 6: Matlab source code for `cone.m`

```
function [value,isterminal,direction]=coneevent(theta,y,gamma)
% Check cone solution for point where vtheta=0
% theta - current angle
% y     - current solution vector
% gamma - ratio of specific heats

value=zeros(2,1);
isterminal=zeros(2,1);
direction=zeros(2,1);

%Quit if Vr goes negative (which shouldn't happen!)
value(1)=1.0;
if (y(1)<0.0)
  value(1)=0.0;
end
isterminal(1)=1;
direction(1)=0;

%Quit if Vtheta goes positive (which occurs at the wall)
value(2)=1.0;
if (y(2)>0.0)
  value(2)=0.0;
end
isterminal(2)=1;
direction(2)=0;
```

Figure 7: Matlab source code for `coneevent.m`

```
function [theta]=thetabetam(beta,M,gamma)
% Return theta for beta-theta-M relationship for oblique shocks
% beta  - shock angle in radians
% M     - upstream Mach number
% gamma - ratio of specific heat

%Cut off at Mach wave angle
if (beta<=asin(1/M)) theta=0; return; end

theta=atan(2*cot(beta)*((M*sin(beta))^2-1)/(M^2*(gamma+cos(2*beta))+2));
```

Figure 8: Matlab source code for `thetabetam.m`