METHOD OF CHARACTERISTICS FOR TWO-DIMENSIONAL ISENTROPIC SUPERSONIC FLOW

MATLAB FUNCTIONS AND APPLICATION SCRIPTS FOR EDUCATIONAL USE

William J. Devenport

Department of Aerospace and Ocean Engineering, Virginia Tech

April 2009

The solution of flow problems using the method of characteristics can be simplified by dividing the flow into regions of uniform flow (with no waves), simple waves (where straight waves belonging to one family of characteristics are present) and complex waves (where curved waves belonging to both families exist), as illustrated in figure 1. The Matlab functions described here provide the means to solve for or plot the flow in each one of these types of regions. The Matlab scripts of this package illustrate how to stitch together these functions to solve more complex problems for a range of applications.

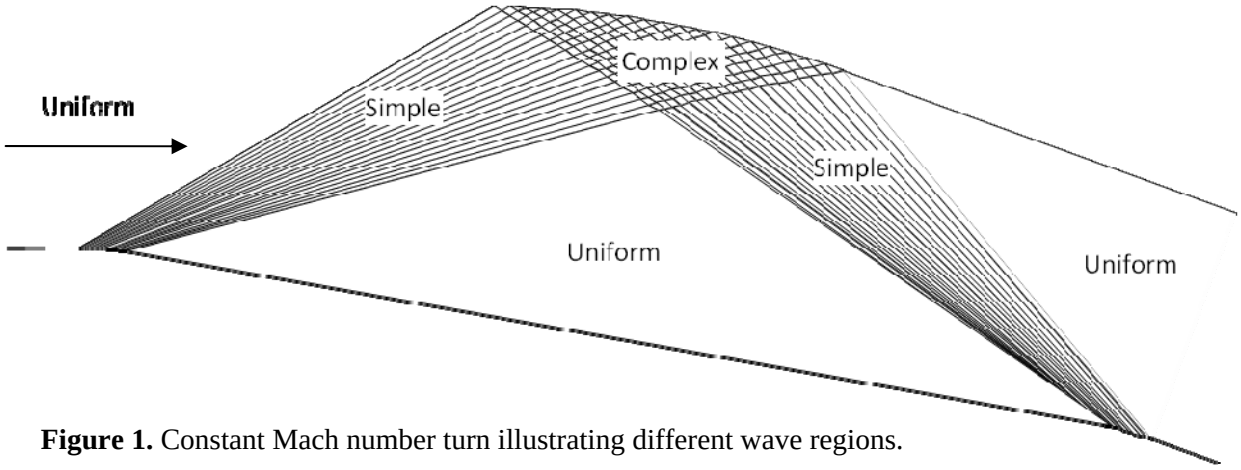

Figure 1. Constant Mach number turn illustrating different wave regions.

1. Function Descriptions

1.1 Simple wave regions

1.1.1 `function [a,n,x,y]=simple(ai,ni,xi,yi,le,g)`

This function solves for the flow and wave geometry in a simple wave region terminated by a characteristic of the opposite sign, as illustrated in figure 2. The figure shows an example of a simple wave lying along C^+ characteristic lines. To calculate this wave the function takes, as input, the flow directions, Prandtl Meyer angles and *x* and *y* coordinates along the 'incoming wave boundary' where the wave is initiated or enters from another region. The specific input arrays for these quantities are:

- `ai` Flow angle α measured from the *x* axis, in radians
- `ni` Prandtl Meyer angle ν, defining the local Mach number, in radians
- `xi` x coordinate
- `yi` y coordinate

These are one dimensional arrays of size 1xN where N is the number of wavelets to be computed. They must be ordered with the upstream wavelet first (index 1). The length of the leading wavelet between this inflow boundary and the point where it is cut by the terminating characteristic is also an input specified through the scalar input variable `le`. Furthermore, the sign of `le` is used to indicate the sign of the characteristics that the simple wave follows. For the C^+ simple wave in figure 2, `le` would be positive. Finally the input variable `g` specifies the ratio of specific heats γ to be used in relating Prandtl Meyer and Mach angles.

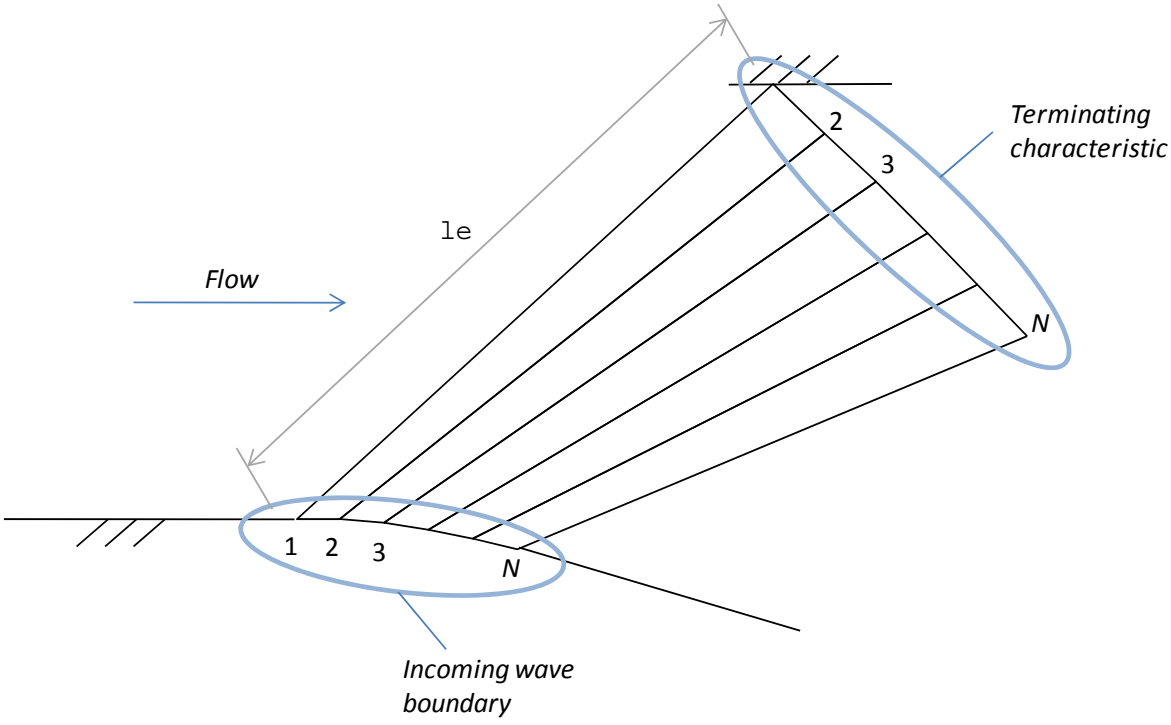

Figure 2. Schematic of a simple wave region as computed by the `simple()` function.

The outputs of this function are `a,n,x,y`. These are arrays giving, respectively, the flow angles, Prandtl Meyer angles and *x* and *y* positions that completely define the simple wave. The arrays are all of size 2xN. The first row of these arrays merely replicates the values and the incoming wave boundary specified as input. The second row provides the values of these quantities computed at the outgoing wave boundary, in this case the terminating characteristic.

1.1.2 `function [a,n,x,y]= simpleCancel(ai,ni,xi,yi,c,x0,y0,a0,g)`

This function solves for the flow and wave geometry in a simple wave region along with the wall shape needed to cancel its reflection. The function inputs and outputs are identical to those of the `simple()` function with the exception that the input argument defining the extent of the wave `le` is replaced by the arguments `c,x0,y0,a0` defined as follows:

c	Sign of characteristic (+1 for C^+, -1 for C^-)
x0	x coordinate of last point on flow boundary preceding cancelation region, see figure 3
y0	y coordinate of last point on flow boundary preceding cancelation region, see figure 3
a0	Flow angle α measured from the *x* axis, in radians, at last point on flow boundary, see figure 3

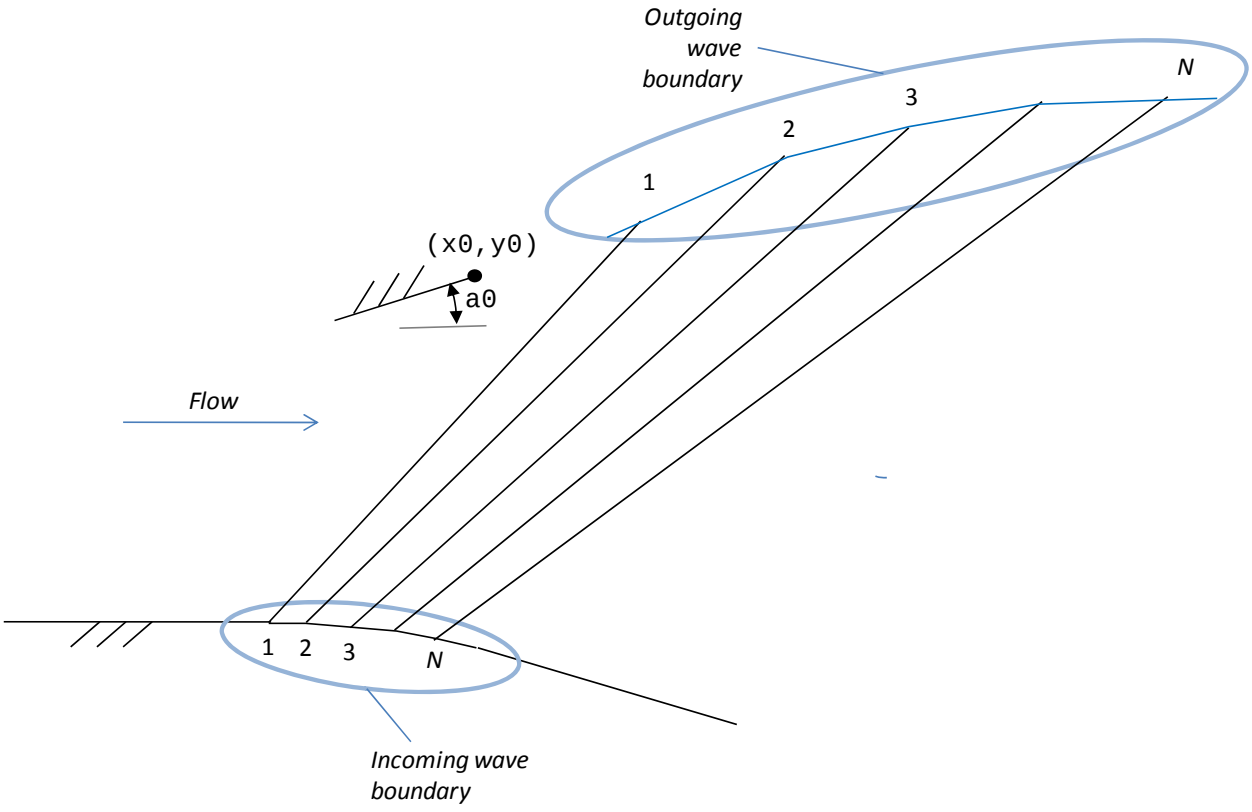

Figure 3. Schematic of a simple wave region as computed by the `simpleCancel()` function.

1.1.3 `function simpleplot(a,n,x,y,g,cl,ch)`

This function plots color contours of Mach number for a simple wave solution on the current figure. Input variables `a,n,x,y` are the output variables provided by the `simple()` or `simpleCancel()` functions, and `g` is the ratio of specific heats γ. The inputs `cl,ch` are the contour limits used to scale the colors representing Mach number.

1.2 Complex wave regions

1.2.1 `function [a,n,x,y]=complex3(ai,ni,xi,yi,bc,g)`

This function solves for the flow and wave geometry in a triangular complex wave region formed by reflection of a simple wave from a plane wall, as illustrated in figure 4. To calculate this complex wave the function takes, as input, the flow directions, Prandtl Meyer angles and *x* and *y* coordinates along the 'incoming wave boundary'. The specific input arrays for these quantities are the same as for the simple wave functions above, i.e.:

- `ai` Flow angle α measured from the *x* axis, in radians
- `ni` Prandtl Meyer angle ν, defining the local Mach number, in radians
- `xi` x coordinate
- `yi` y coordinate

These are one dimensional arrays of size 1xN where N is the number of wavelets to be computed. They must be ordered with the upstream wavelet first (index 1). The function also requires the angle of the wall (in radians) from which the wave is reflecting, provided in the variable `bc`, and the ratio of specific heats γ in the variable `g`. The sign of the characteristic of the incoming wave does not need to be specified since it can be inferred from the rest of the input data.

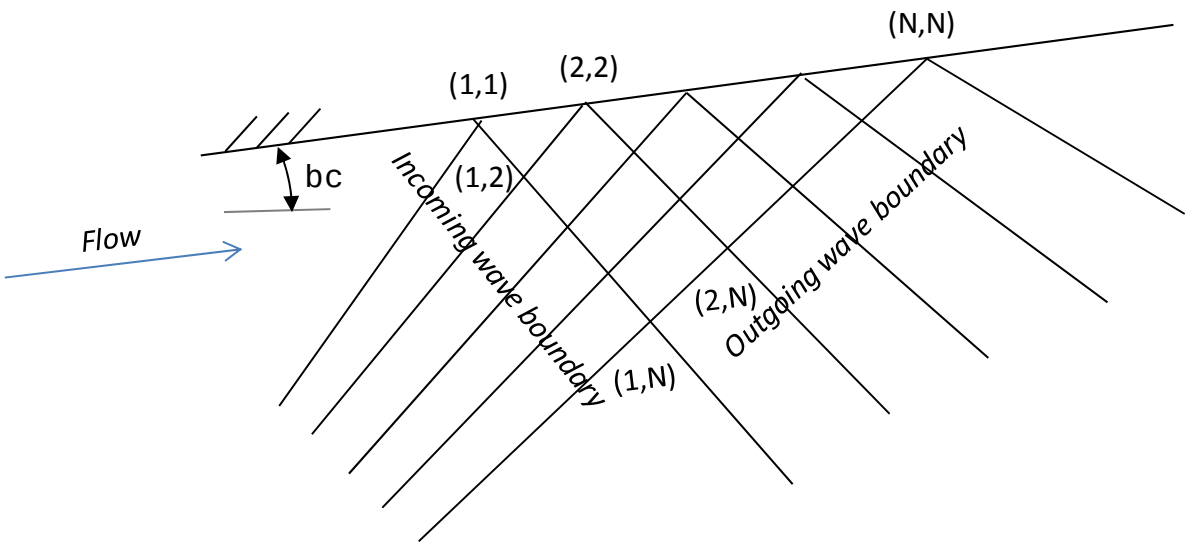

Figure 4. Schematic of a complex wave region as computed by the `complex3()` function.

The outputs of this function are `a, n, x, y`. These are arrays giving, respectively, the flow angles, Prandtl Meyer angles and *x* and *y* positions that completely define the complex wave. The arrays are all of size NxN and contain data only on and above the main diagonal. The first row of each array merely replicates the values at the incoming wave boundary specified as input. The last column provides values computed at the outgoing wave boundary. The remaining intermediate array elements contain values computed at

the grid of points defined by the intersection of the incoming wavelets and their reflection. The wall is represented by the values on the main diagonal. Indexing is illustrated in figure 4.

1.2.2 `function [a,n,x,y]=complex3curve(ai,ni,xi,yi,bc,g)`

This function solves for the flow and wave geometry in a triangular complex wave region formed by reflection of a simple wave from a curved wall with a shape defined by a polynomial. The operation of the function is identical to `complex3()`with the exception of the input variable `bc` which here is a 1D array of polynomial coefficients for the polynomial describing the curved wall shape, as in the Matlab function poly().

1.2.3 `function [a,n,x,y]=complex3free(ai,ni,xi,yi,bc,g)`

This function solves for the flow and wave geometry in a triangular complex wave region formed by reflection of a simple wave from a constant Mach number boundary, such as a jet edge. The operation of the function is identical to `complex3()`with the exception of the input variable `bc` which here is a scalar giving in radians the value of the Prandtl-Meyer angle (and thus implicitly the Mach number) on the free boundary.

1.2.4 `function complex3plot(a,n,x,y,g,cl,ch)`

This function plots color contours of Mach number for a triangular complex wave solution on the current figure. Input variables `a,n,x,y` are the output variables provided by the `complex3(), complex3curve()` or `complex3free()`functions, and g is the ratio of specific heats γ. The inputs `cl, ch` are the contour limits used to scale the colors representing Mach number.

1.2.5 `function [a,n,x,y]=complex4(ap,np,xp,yp,an,nn,xn,yn,g)`

This function solves for the flow and wave geometry in a quadrilateral complex wave region formed by the intersection of two simple waves, as illustrated in figure 5. To calculate this wave the function takes, as input, the flow directions, Prandtl Meyer angles and *x* and *y* coordinates along the two incoming wave boundaries. For the C+ incoming wave boundary the input arrays are:

ap	Flow angle α measured from the *x* axis, in radians
np	Prandtl Meyer angle ν, defining the local Mach number, in radians
xp	x coordinate
yp	y coordinate

For the C- incoming wave boundary they are:

an	Flow angle α measured from the *x* axis, in radians
nn	Prandtl Meyer angle ν, defining the local Mach number, in radians
xn	x coordinate
yn	y coordinate

These are one dimensional arrays of size 1xN where N is the number of wavelets to be computed. They must be ordered with the upstream wavelet first (index 1). As before g is the ratio of specific heats γ.

The outputs of this function are `a, n, x, y`. These are arrays giving, respectively, the flow angles, Prandtl Meyer angles and *x* and *y* positions that completely define the complex wave. The arrays are all of size NxN and are completely defined. The first row and first column of each array merely replicate the values at the incoming C+ and C- wave boundaries, specified as input. The last row and column provide values

computed at the outgoing wave boundaries. The remaining intermediate array elements contain values computed at the grid of points defined by the incoming wavelets and their intersection. Indexing is illustrated in figure 5.

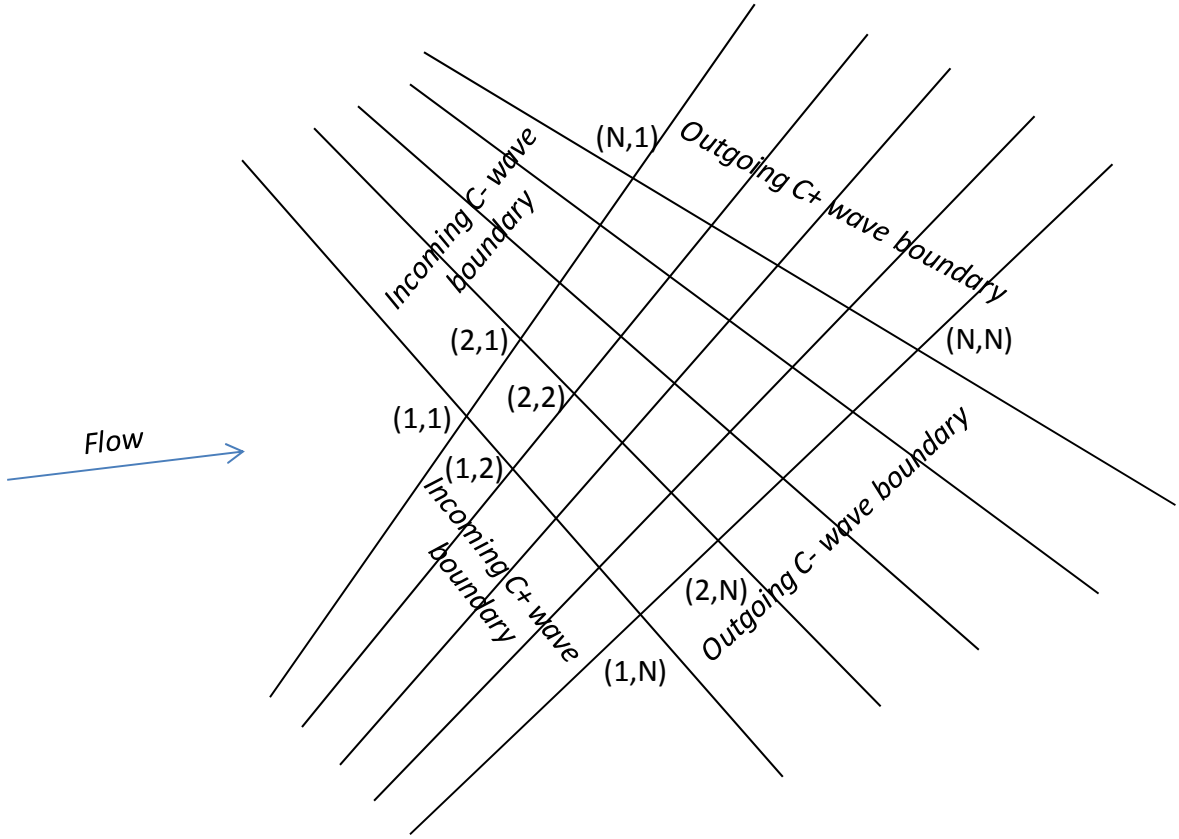

Figure 5. Schematic of a complex wave region as computed by the `complex4()` function.

1.2.6 `function complex4plot(a,n,x,y,g,cl,ch)`

This function plots color contours of Mach number for a quadrilateral complex wave solution on the current figure. Input variables `a,n,x,y` are the output variables provided by the `complex4()` function, and `g` is the ratio of specific heats γ. The inputs `cl,ch` are the contour limits used to scale the colors representing Mach number.

1.3 Utility and low level functions

1.3.1 `function uniformplot(a,n,x,y,g,cl,ch)`

This function plots a polygon on the current figure with a single color scaled according to a given Mach number. This function provides a means to add regions of uniform flow to figures containing simple and/or complex waves plotted using `simpleplot()`, `complex3plot()` and `complex4plot()`. The input variables are the same as those functions (for consistency) and specified as follows. The variables `x` and `y` are one-dimensional arrays that specify in clockwise or counter-clockwise order the coordinates of the vertices of the polygon region. Variables `a` and `n` are arrays of the same size as `x` and `y` that specify respectively the flow angle and Prandtl Meyer angle at these points. Note that only the average Mach number implied by these Prandtl Meyer angles is used to set the color and that the flow angles are not used. As before `g` is the ratio of specific heats γ and the inputs `cl,ch` are the contour limits used to scale the colors representing Mach number.

1.3.2 `function n=nu(m,g)`

Computes the Prandtl Meyer angles `n` implied by a given set of Mach numbers `m` and a ratio of specific heats `g`. Note that `m` may be scalar or an array of any shape and that the shape of `n` will match that of `m`. Note angles are in radians and that `g` must be a scalar.

1.3.3 `function m=m_nu(n,g)`

Computes the Mach numbers `m` implied by a given set of Prandtl Meyer angles `n` and a ratio of specific heats `g`. Note that `n` may be scalar or an array of any shape and that the shape of `m` will match that of `n`. Note angles are in radians and that `g` must be a scalar.

1.3.4 `function [x,y]=intercept(x1,y1,t1,x2,y2,t2)`

Computes the intercept `x,y` of a line of angle `t1` (in radians), starting at `x1,y1`, and a line of angle `t2` starting at `x2,y2`. All variables are scalars.

1.3.5 `function [x,y,a]=interceptCurve(x1,y1,t1,cf)`

Computes the intercept `x,y` of a line of angle `t1` (in radians), starting at `x1,y1`, and a polynomial y=f(x) with coefficients `cf` (organized as in the Matlab function `poly()`). Also provides the angle `a` (in radians) of a tangent to the polynomial at the intercept. All variables are scalars.

2. Example Application Scripts

Five Matlab scripts are provided to illustrate application of the above functions to the analysis of some compressible flow problems. The Matlab script names all start with the letter A to distinguish them from the functions. The scripts and examples of the plots they produced are given below. Detailed description of the script is provided for with the constant Mach number turn example.

2.1 `AconstantMachNumberTurn.m`

Computes and plots a 20 degree constant Mach number turn of a uniform flow at Mach 2.

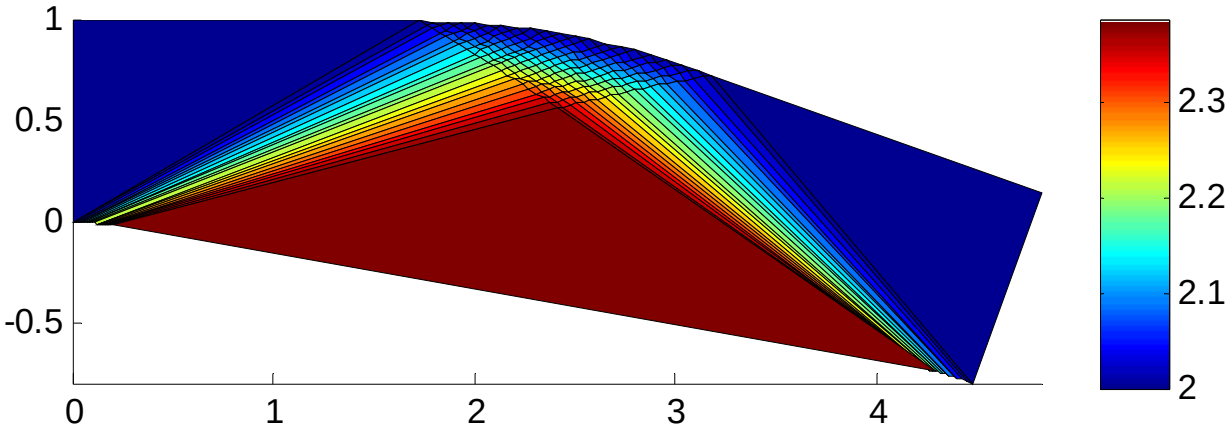

Figure 6. Output plot from `AconstantMachNumberTurn.m`.

Detailed description (see listing in figure 7 on following page):

Line 2	Specifies the ratio of specific heats approach Mach number, and duct height
Line 3	Specifies flow angles on the lower wall associated with initial (10-degree) turn in the lower wall. Number of turn angles defines number of wavelets to be used in computing flow. In this case 21 angles are used with a resolution of 0.5 degrees.
Lines 4-6	Specifies shape of initial turn, which begins at (0,0) and has a total length of 0.2 duct heights in the x-direction. Note that y-coordinates must be chosen so that local slope matches the local flow angles specified in line 3. Here this is done using a parabolic shape.
Line 7	Determines Prandtl Meyer angles on the wall in the initial turn.
Line 8	Determines the length of the leading wavelet of the first simple wave (=ductHeight/sin(arcsin(1/M)).
Lines 9-10	Opens figure window, determines Mach number limits of solution for plotting
Lines 11-12	Computes and plots initial simple wave. Note that the values of the input variables `ai,ni,xi,yi` at the 'incoming wave boundary' are those specified on the lower wall for the initial turn.
Lines 13-14	Computes and plots the complex wave reflection from the upper duct wall. Note that the input variables for the incoming wave are simply extracted from the second (i.e. last) row of the output variables from the simple wave solution as `a1(end,:), n1(end,:), x1(end,:), y1(end,:).`

```
1.  clear all
2.  g=1.4;minf=2;ductHeight=1;
3.  ai=-[0:.5:10]*pi/180;
4.  x0=0;y0=0;L=0.2*ductHeight %L is length of turn
5.  xi=linspace(x0,x0+L,length(ai));
6.  yi=y0+0.5*tan(ai).*(xi-x0);
7.  ni=nu(minf,g)-ai;
8.  le=ductHeight*minf;

9.  figure(1);clf(1);
10. cl=minf;ch=m_nu(nu(minf,g)-min(ai),g);

11. [a1,n1,x1,y1]=simple(ai,ni,xi,yi,le,g);
12. simpleplot(a1,n1,x1,y1,g,cl,ch);

13. [a2,n2,x2,y2]=complex3free(a1(end,:),n1(end,:),x1(end,:),y1(end,:),nu(minf,g),g);
14. complex3plot(a2,n2,x2,y2,g,cl,ch);

15. [a3,n3,x3,y3]=simpleCancel(a2(:,end),n2(:,end),x2(:,end),y2(:,end),-
    1,xi(end),yi(end),ai(end),g);
16. simpleplot(a3,n3,x3,y3,g,cl,ch);

17. uniformplot([a1(1,1) a1(2,1) a1(1,1)],[n1(1,1) n1(2,1) n1(1,1)],[x1(1,1) x1(2,1)
    x1(1,1)],[y1(1,1) y1(2,1) y1(2,1)],g,cl,ch);
18. uniformplot([a1(1,end) a1(2,end) a3(2,1)],[n1(1,end) n1(2,end) n3(2,1)],[x1(1,end)
    x1(2,end) x3(2,1)],[y1(1,end) y1(2,end) y3(2,1)],g,cl,ch);
19. [xe,ye]=intercept(x3(1,end),y3(1,end),a3(1,end),x3(2,end),y3(2,end),a3(2,end)+pi/2);
20. uniformplot([a3(1,end) a3(2,end) a3(1,end)],[n3(1,end) n3(2,end)
    n3(1,end)],[x3(1,end) x3(2,end) xe],[y3(1,end) y3(2,end) ye],g,cl,ch);

21. hold off
22. caxis([cl ch]);colorbar;
23. axis image
```

Figure 7. Listing of `AconstantMachNumberTurn.m`.

Lines 15-16 Computes and plots second simple wave. The values of the input variables on the incoming wave boundary are here the values computed on the outgoing wave boundary of the complex region, and are thus specified as `a2(:,end)`, `n2(:,end)`, `x2(:,end)`, `y2(:,end)`.

Lines 17-19 Plots the various uniform regions. Note that the top right hand corner of most downstream uniform region is chosen (in lines 19 and 20) as the intercept of the upper wall after the turn and a perpendicular to the lower wall, calculated using the `intercept` function.

Lines 21-22 Finalizes the plot and adds a color scale.

2.2 AminimumLengthNozzle.m

Repeatedly computes the shape of, and flow through, a minimum length nozzle given the nozzle exit Mach number specified by the user.

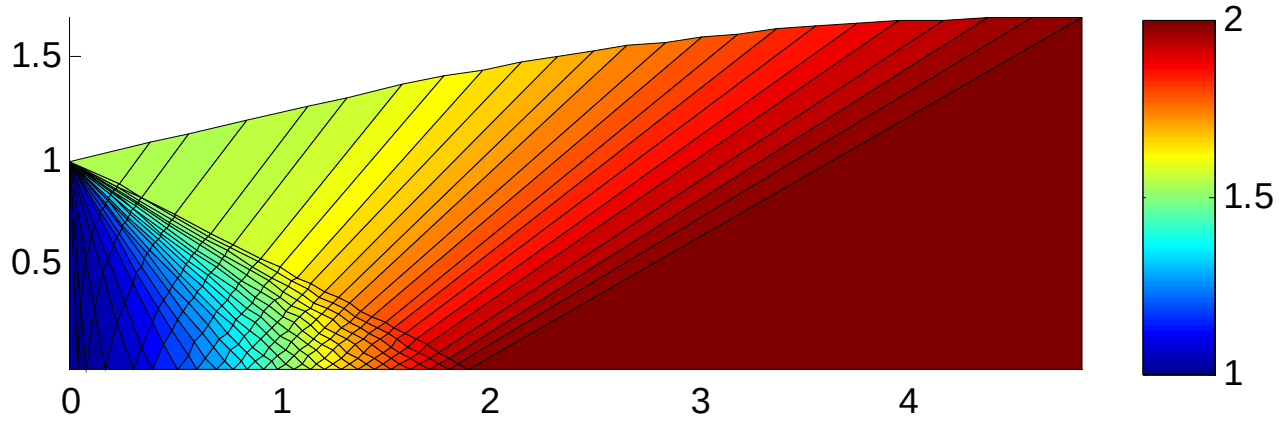

Figure 8. Output plot from `AminimumLengthNozzle.m` for an exit Mach number of 2.

Notes: This calculation requires an uneven distribution of flow angles in the wave to accurately compute the nozzle shape, hence the extra terms in line 6 specifying the vector of flow angles `ai`. The script `ArocketEngineNozzle.m` contains a simplified version of this program that runs for an exit Mach number of 2.

2.3 AwaveInteraction.m

Computes the flow produced by the crossing and then reflection of two centered expansion fans produced by sudden convex turns in the upper and lower wall of a duct. The initial duct width is 2 and the initial Mach number is 2. The upper wall turn angle is 7.5 degrees. The lower wall turn angle is 5 degrees.

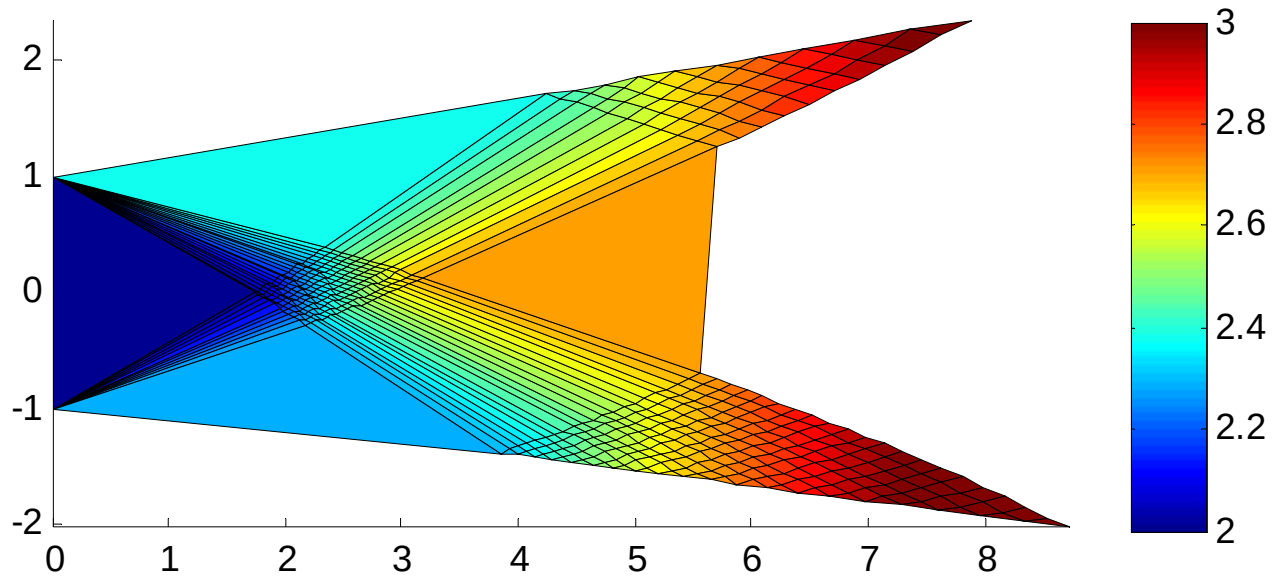

Figure 9. Output plot from `AwaveInteraction.m`.

2.4 `AunderExpandedJet.m`

Computes the underexpanded jet flow for a nozzle exit Mach number of 2 and a back pressure to chamber pressure ratio for which the flow turns by 10 degrees at it leaves the nozzle exit. Note that the extrapolation of the waves leads to some of them crossing as they focus back on the wave edge. The exact solution has them focusing to a single point here.

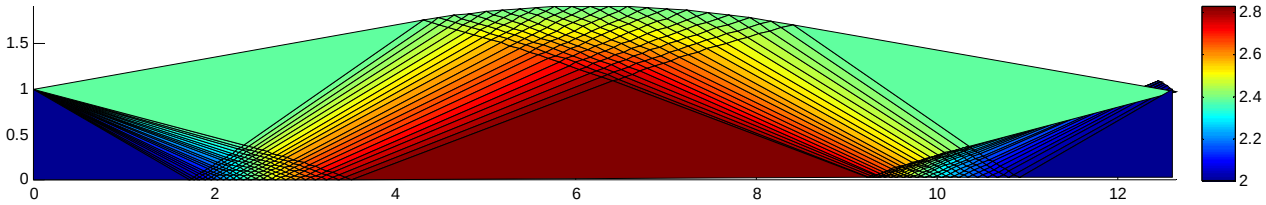

Figure 10. Output plot from `AunderExpandedJet.m`.

2.2 `AwindTunnelNozzle.m`

Computes the shape and flow through a wind tunnel type nozzle with an initial region in which the nozzle contour is parabolic. This is done by computing the flow as a succession of reflections of a single simple wave. Note that to be done successfully requires a consistent selection of the wall shape, length and the initial wave angles (i.e. changing one of these will likely require changing the others.

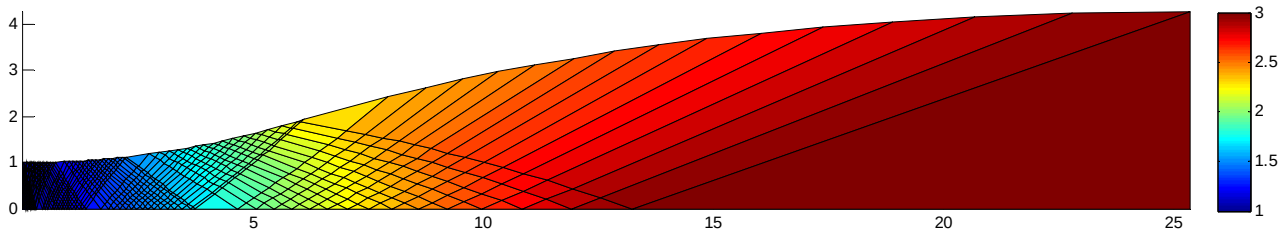

Figure 11. Output plot from `AwindTunnelNozzle.m`.