```matlab
% Benjamin Stutzke
% ENAE 488P
% Final Project

% Freestream Conditions
M1 = 6;
T1 = 226;           % K
p1 = 1200;          % Pa
rho1 = p1/(287*T1); % kg/m^3
gamma = 1.4;
v1 = M1*sqrt(gamma*287*T1); % m/s

% Waverider Properties
l = 3;
b = l/4:0.01:l;
Tw = 1620;      % K
theta = [5 10];  % deg
theta = deg2rad(theta);
S = (1/2).*b.*l;    % planform area


% Solving for post-shock conditions
beta = zeros(1, length(theta));
p2_p1 = zeros(1, length(theta));
rho2_rho1 = zeros(1, length(theta));
T2_T1 = zeros(1, length(theta));
M2 = zeros(1, length(theta));

for k=1:length(theta)
    beta(k) = tbm(M1, theta(k), true);
    [p2_p1(k), rho2_rho1(k), T2_T1(k), M2(k)] = oshock(M1, beta(k), theta(k),
 gamma);
end

% Calculating lift
p2 = p2_p1 * p1;
L = zeros(length(theta), length(S));

for k=1:length(theta)
    L(k, :) = (p2(k) - p1).*S;
end

% Calculating pressure drag
h = l.*tan(beta);   % total height
t = l.*tan(theta);  % inner height

A_proj = zeros(length(theta), length(b));
for k=1:length(theta)
    A_proj(k, :) = (1/2).*b.*h(k) - (1/2).*b.*(h(k)-t(k));
end

Dr = zeros(length(theta), length(b));
```

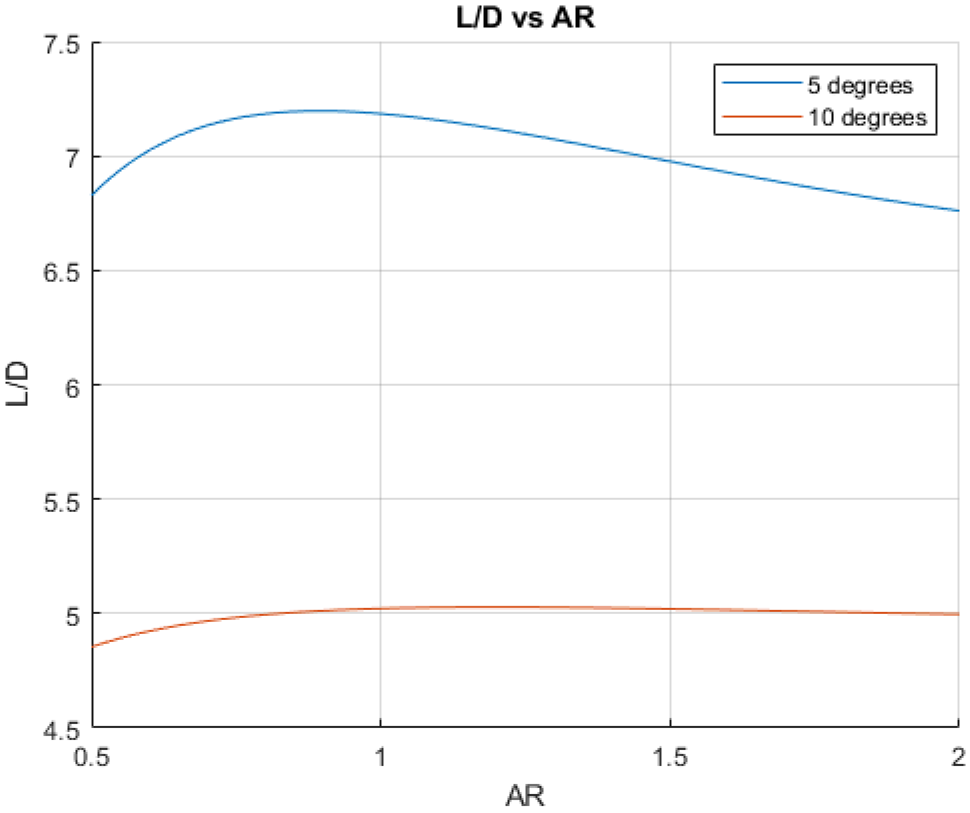
```
for k=1:length(theta)
    Dr(k, :) = (p2(k) - p1).*A_proj(k, :);
end

% Defining waverider shape
A = [0 0 0]; % point at nose
B = [l 0 0]; % point at peak of upper trailing edge
C = zeros(length(theta), 3); % point at peak of lower trailing edge
D = zeros(length(theta), 3, length(b)); % point at wingtip

for k=1:length(theta)
    C(k, :) = [l 0 -l*tan(theta(k))];
    for j=1:length(b)
        D(k, :, j) = [l     b(j)/2      -l*tan(beta(k))];

        % Plotting code for debugging
        %{
        figure;
        hold on
        grid on

        plot3([A(1) B(1)], [A(2) B(2)], [A(3) B(3)]); % top line
        plot3([A(1) D(k, 1, j)], [A(2) D(k, 2, j)], [A(3) D(k, 3, j)]); % LE
        plot3([B(1) D(k, 1, j)], [B(2) D(k, 2, j)], [B(3) D(k, 3, j)]); % TE

        plot3([A(1) C(k, 1)], [A(2) C(k, 2)], [A(3) C(k, 3)]); % bottom corner
        plot3([C(k, 1) D(k, 1, j)], [C(k, 2) D(k, 2, j)], [C(k, 3) D(k, 3,
 j)]); % bottom TE

        title_str = "angle: " + num2str(theta(k)) + " span: " + num2str(b(j));
        title(title_str);
        xlabel("x");
        ylabel("y");
        zlabel("z");
        axis([0 3 0 1.5 -1 0]);
        %}
    end
end

% Calculating Cw for top surface (freestream conditions)
rhow = p1/(287*Tw);

% Sutherland's Law
Sref = 110;
Tref = 273;
muref = 1.716e-5;

mu1 = muref*((T1/Tref)^(3/2))*((Tref+Sref)/(T1+Sref));
muw = muref*((Tw/Tref)^(3/2))*((Tref+Sref)/(Tw+Sref));

Cw_top = (rhow*muw)/(rho1*mu1);

% Calculating Cw for bottom surface (post-shock)
rho2 = rho2_rho1.*rho1;
```

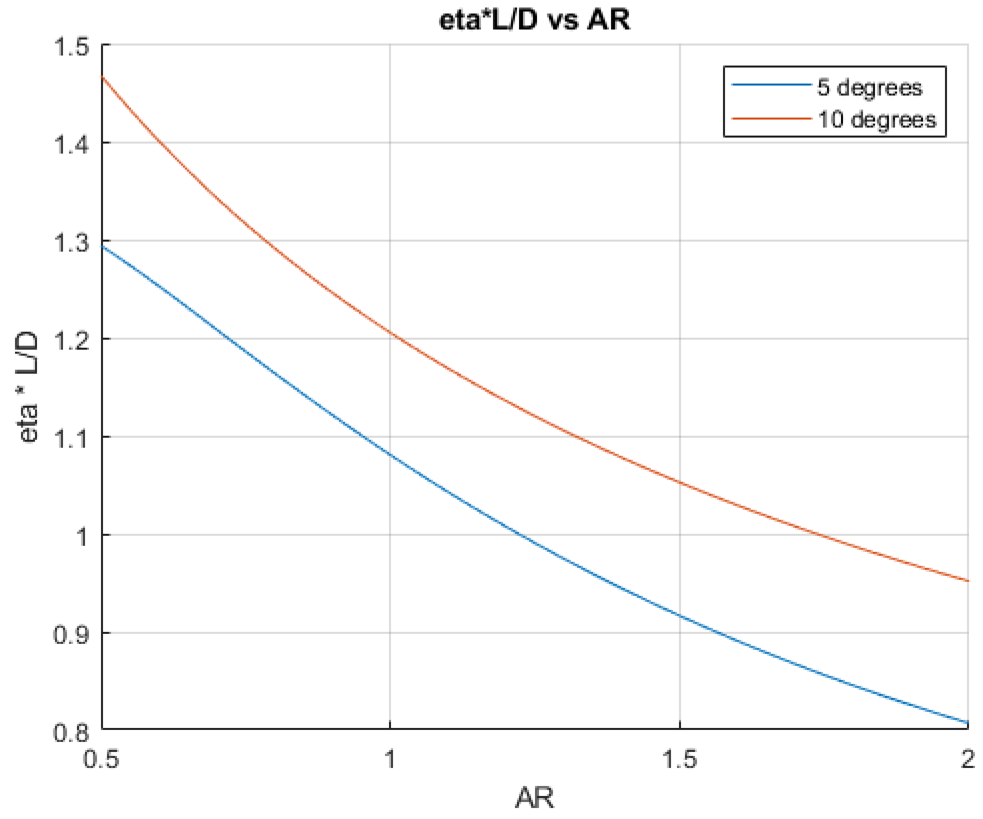
```
T2 = T2_T1.*T1;
mu2 = muref.*((T2./Tref).^(3/2)).*((Tref+Sref)./(T2+Sref));

Cw_bot = (rhow*muw)./(rho2.*mu2);

% Calculating shear stress contributions to lift and drag
v2 = M2.*sqrt(gamma*287.*T2);

Cf_top = zeros(length(theta), length(b));
Cf_bot = zeros(length(theta), length(b));

% Drag due to friction
Df_top = zeros(length(theta), length(b));
Df_bot = zeros(length(theta), length(b));

% Lift due to friction
Lf_top = zeros(length(theta), length(b));
Lf_bot = zeros(length(theta), length(b));

Vol = zeros(length(theta), length(b));
eff = zeros(length(theta), length(b)); % volumetric efficiency

% Surface area
S_top_arr = zeros(length(theta), length(b));
S_bot_arr = zeros(length(theta), length(b));

for k=1:length(theta)

    for j=1:length(b)

        % Using Rodrigues' rotation formula to rotate waverider surfaces
        % Rotating top surface for integration

        psi = atan2(-2*h(k), b(j));
        r = [1 0 0]; % rotating around x-axis

        K = [  0   -r(3)   r(2);
                r(3)   0    -r(1);
                -r(2)  r(1)    0;  ];

        R = expm(-psi.*K); % rotating by angle psi

        % updating vertices
        D_unrot = D(k, :, j)';
        D_rot = R*D_unrot;

        C_unrot = C(k, :)';
        C_rot = R*C_unrot;

        %{
        figure;
        hold on;
        grid on;
        plot3([A(1) D_rot(1)], [A(2) D_rot(2)], [A(3) D_rot(3)], 'color',
'blue');
        plot3([B(1) D_rot(1)], [B(2) D_rot(2)], [B(3) D_rot(3)], 'color',
'blue');
        plot3([A(1) D_unrot(1)], [A(2) D_unrot(2)], [A(3) D_unrot(3)],
'color', 'green');
        plot3([B(1) D_unrot(1)], [B(2) D_unrot(2)], [B(3) D_unrot(3)],
'color', 'green');
        title_str = "angle: " + num2str(theta(k)) + " span: " + num2str(b(j));
        title(title_str);
        xlabel("x");
        ylabel("y");
        zlabel("z");
        %}

        % integral2 function requires swapping x and y
        Cf_top_plate = @(x,y) 0.664.*((Cw_top./((rho1.*v1.*y)./mu1)).^(1/2));

        ymin = @(x) x*D_rot(2)/l;
        Cf_top_area = integral2(Cf_top_plate, 0, D_rot(2), ymin, l);

        Cf_top(k, j) = Cf_top_area;

        % Calculating total drag force on top surface
        S_top = (1/2)*l*D_rot(2);
        S_top_arr(k, j) = S_top;
        F_top = Cf_top_area*((1/2)*rho1*(v1^2)*S_top);
        Df_top(k, j) = F_top;
        Lf_top(k, j) = F_top.*0;

        % Rotating bottom surface for integration

        r = [0 1 0]; % rotating around y axis
        K = [  0   -r(3)   r(2);
                r(3)   0    -r(1);
                -r(2)  r(1)    0;  ];

        R2 = expm(-theta(k).*K); % rotating by angle theta

        D_rot2 = R2*D_unrot;
        C_rot2 = R2*C_unrot;

        r = [1 0 0]; % rotating around x axis
        K = [  0   -r(3)   r(2);
                r(3)   0    -r(1);
                -r(2)  r(1)    0;  ];

        psi2 = atan2(D_rot2(3), D_rot2(2));
        R3 = expm(-psi2.*K);

        D_rot3 = R3*D_rot2;
        C_rot3 = R3*C_rot2;

        %{
```

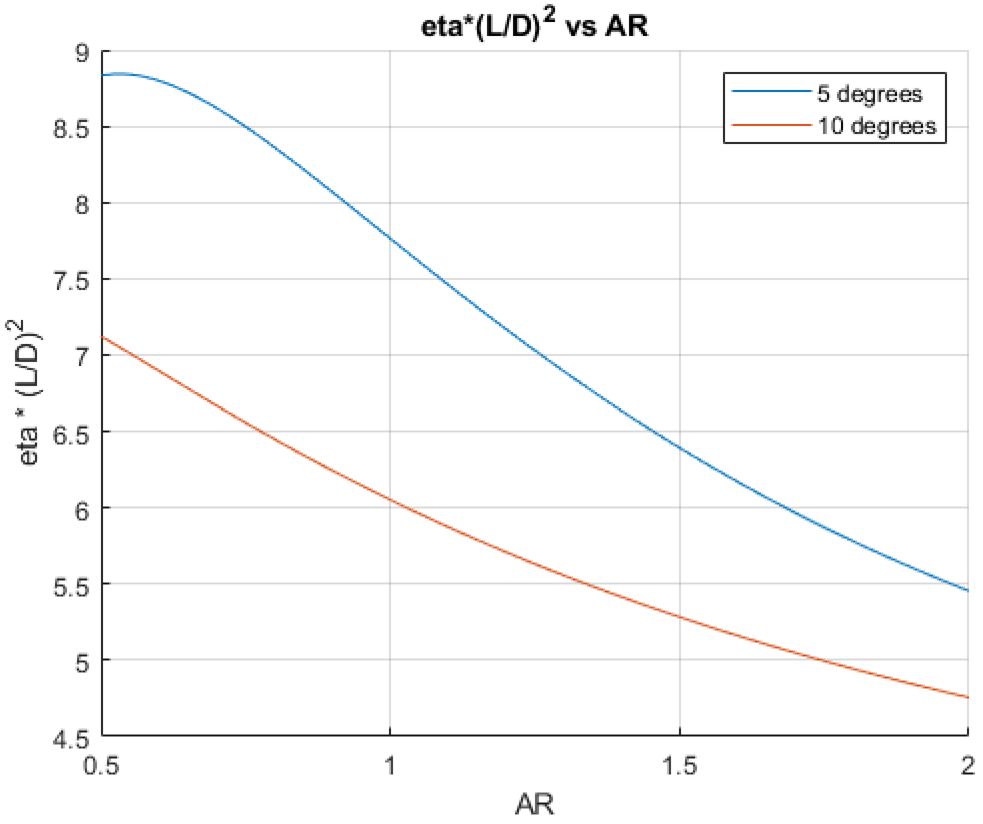
```
        figure;
        hold on;
        grid on;


        plot3([A(1) D_unrot(1)], [A(2) D_unrot(2)], [A(3) D_unrot(3)],
 'color', 'green');
        plot3([C_unrot(1) D_unrot(1)], [C_unrot(2) D_unrot(2)], [C_unrot(3)
 D_unrot(3)], 'color', 'green');
        plot3([A(1) C_unrot(1)], [A(2) C_unrot(2)], [A(3) C_unrot(3)],
 'color', 'green');

        plot3([A(1) D_rot2(1)], [A(2) D_rot2(2)], [A(3) D_rot2(3)], 'color',
 'red');
        plot3([C_rot2(1) D_rot2(1)], [C_rot2(2) D_rot2(2)], [C_rot2(3)
 D_rot2(3)], 'color', 'red');
        plot3([A(1) C_rot2(1)], [A(2) C_rot2(2)], [A(3) C_rot2(3)], 'color',
 'red');

        plot3([A(1) D_rot3(1)], [A(2) D_rot3(2)], [A(3) D_rot3(3)], 'color',
 'blue');
        plot3([C_rot3(1) D_rot3(1)], [C_rot3(2) D_rot3(2)], [C_rot3(3)
 D_rot3(3)], 'color', 'blue');
        plot3([A(1) C_rot3(1)], [A(2) C_rot3(2)], [A(3) C_rot3(3)], 'color',
 'blue');

        title_str = "angle: " + num2str(theta(k)) + " span: " + num2str(b(j));
        title(title_str);
        xlabel("x");
        ylabel("y");
        zlabel("z");
        %}

        % integral2 function requires swapping x and y
        Cf_bot_plate = @(x,y) 0.664.*((Cw_bot(k)./((rho2(k).*v2(k).*y)./
mu2(k))).^(1/2));

        ymin = @(x) x*D_rot3(2)/l;
        Cf_bot_area = integral2(Cf_bot_plate, 0, D_rot3(2), ymin, l);

        Cf_bot(k, j) = Cf_bot_area;

        % Calculating total drag force on bottom surface
        S_bot = (1/2)*l*D_rot3(2);
        S_bot_arr(k, j) = S_bot;
        F_bot = Cf_bot_area*((1/2)*rho2(k)*(v2(k)^2)*S_bot);
        Df_bot(k, j) = F_bot.*cos(-theta(k));
        Lf_bot(k, j) = F_bot.*sin(-theta(k));

        % Finding volume of vehicle
        % Outer pyramid
        A_out = (1/2)*b(j)*abs(D(k, 3, j));
        Vout = (1/3)*l*A_out;
        % Inner pyramid
        A_in = (1/2)*b(j)*(abs(D(k, 3, j))-abs(C(k, 3)));
        Vin = (1/3)*l*A_in;

        Vtot = Vout - Vin;
        Vol(k, j) = Vtot;
        eff(k, j) = (Vtot^(2/3))./S(j);
    end
end

% Calculating total L/D
Dtot = Dr + (2.*Df_top) + (2.*Df_bot);
Ltot = L + 2.*Lf_top + 2.*Lf_bot;
L_D = Ltot./Dtot;

AR = (b.^2)./S;
etaL_D = eff.*L_D;

figure;
hold on
grid on
plot(AR, L_D(1, :));
plot(AR, L_D(2, :));
xlabel("AR");
ylabel("L/D");
legend("5 degrees", "10 degrees");
title("L/D vs AR");

figure;
hold on
grid on
plot(AR, etaL_D(1, :));
plot(AR, etaL_D(2, :));
xlabel("AR");
ylabel("eta * L/D");
legend("5 degrees", "10 degrees");
title("eta*L/D vs AR");

% Maximum L/D and eta*L/D occurs at the peak of eta*(L/D)^2
opt = L_D.*etaL_D;

figure;
hold on
grid on
plot(AR, opt(1, :));
plot(AR, opt(2, :));
xlabel("AR");
ylabel("eta * (L/D)^2");
legend("5 degrees", "10 degrees");
title("eta*(L/D)^2 vs AR");

index1 = find(opt(1, :) == max(opt(1, :)));
index2 = find(opt(2, :) == max(opt(2, :)));
```

```
fprintf("Optimal span for 5 degree wedge (meters): ");
disp(b(index1));

fprintf("Optimal span for 10 degree wedge (meters): ");
disp(b(index2));
```

```
Optimal span for 5 degree wedge (meters):     0.8000

Optimal span for 10 degree wedge (meters):     0.7500
```




eta*L/D vs AR
eta * L/D
AR
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.5
1
1.5
2
5 degrees
10 degrees


eta*(L/D)^2 vs AR
eta * (L/D)^2
AR
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
0.5
1
1.5
2
5 degrees
10 degrees

Published with MATLAB® R2022b