Estimation of Total Drag on RPV Vehicle

prepared by: Matthew T. Gates

Advisor: Professor R.C. Nelson

Department of Aerospace and Mechanical Engineering

University of Notre Dame

Notre Dame, Indiana 46556

August 12, 2011

Abstract

Code originally written by Brian Neiswander to calculate the lift and induced drag coefficients based on Prandtl's lifting line theory was supplemented to also calculate form drag. This was done using a method developed by Sivells and Neely for NACA. The form drag coefficient was estimated at numerous span-wise locations, and these were then summed and added to the induced drag coefficient to get the total drag coefficient. This theoretical value is then compared to experimental data obtained using a force balance. The theory aligns relatively well, but its exact accuracy is unknown until further experiments can be run.

1.0 Introduction

There is relatively little data on low Reynolds number airfoil flight, such as for the RPV vehicles used in the Notre Dame senior design class, AME 40452. One particular difficulty is in estimating the total drag on the wings. Induced drag has historically been calculated to sufficient accuracy using Prandtl's lifting line theory.[1] The problem in determining total drag has been an inability to estimate form drag. The purpose of this document, and the attached code, is to resolve this issue. Using the code, total drag should be estimated within enough accuracy to allow for the design of an RPV to suit the requirements of the course.

2.0 Method

Using Prandtl's lifting line theory, the wing is broken into equal span-wise segments about which various wing characteristics are calculated. One of these characteristics is the effective angle of attack. Using the effective angle of attack at each of these span-wise segments, the local lift coefficient can be determined using a plot of the lift coefficient versus angle of attack for that specific airfoil type. Because the effective angle of attack changes across the span of the wing, as shown in figure 1, the local lift coefficient also varies across the span.

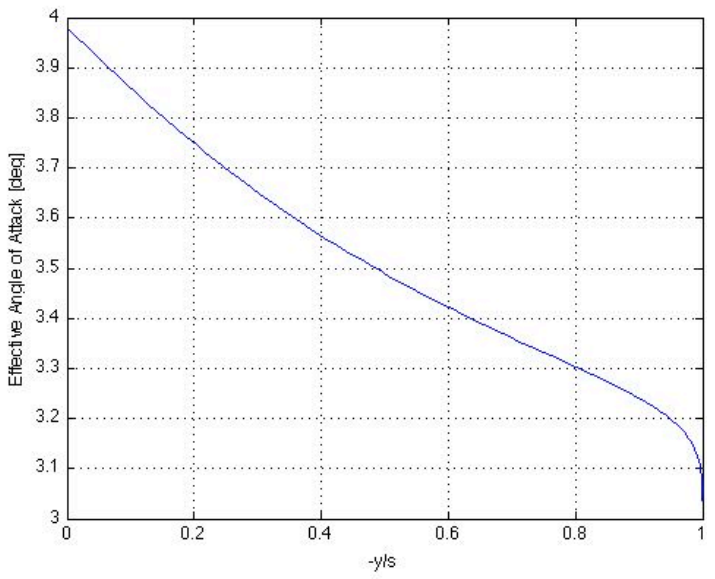

Figure 1. Effective angle of attack across span.

Using the local lift coefficient and a drag polar determined from two-dimensional data of the airfoil, the local form drag is determined. This is done by curve fitting the two-dimensional drag polar and then inputting the local lift coefficient to get the equivalent form drag. This local form drag is then normalized and summed to give an estimate of the total form drag on the airfoil, as shown in equation 1:

$$C_{Df} = \Sigma \frac{C_{df} \Delta S}{S} \quad (1),$$

where C_{Df} is the total form drag coefficient for the wing, C_{df} is the local form drag coefficient of the wing, ΔS is the area of the wing segment for that particular form drag coefficient in square meters, and S is total surface area of the wing in square meters. This method is described in detail in NACA technical note no. 1269.[2]

This form drag coefficient is then combined with the induced drag coefficient to determine the total drag coefficient. The whole process is then iterated through a range of angles of attacks in order to develop three-dimensional estimates for the wing.

3.0 Data Analysis

The original code, before the modifications, was capable of calculating the induced drag on the airfoil. With a slight addition, the induced drag at all of the span stations could be displayed, allowing for the drag across the span, as shown in figure 2. This allows for confirmation that the code is running correctly, and also helps troubleshoot the total drag.

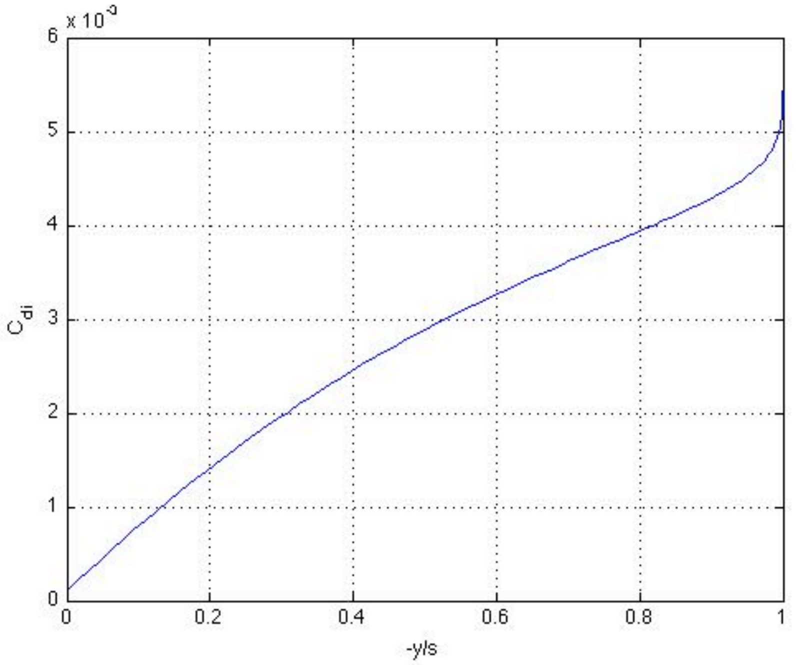

Figure 2. Induced drag across span at a 4^0 angle of attack.

The next addition to the code is the introduction of form drag calculations. Slightly tweaking the code to generate the local lift coefficient allows for calculation of the local induced drag, but also allows for determination of the form drag. Shown in figure 3, these coefficients are the local values of form drag.

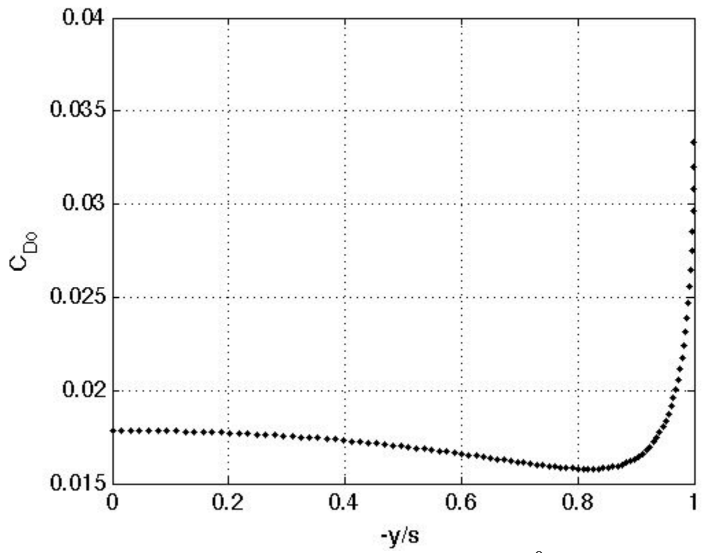

Figure 3. Form drag across span at a 4^0 angle of attack.

Using equation 1, the local form drag is converted to a total form drag coefficient. After running through a range of angles of attacks, a set of drag coefficients are created. Combined with the induced drag, the total drag for each angle of attack is determined. These are plotted against the lift coefficients for each angle of attack, as shown in figure 4.

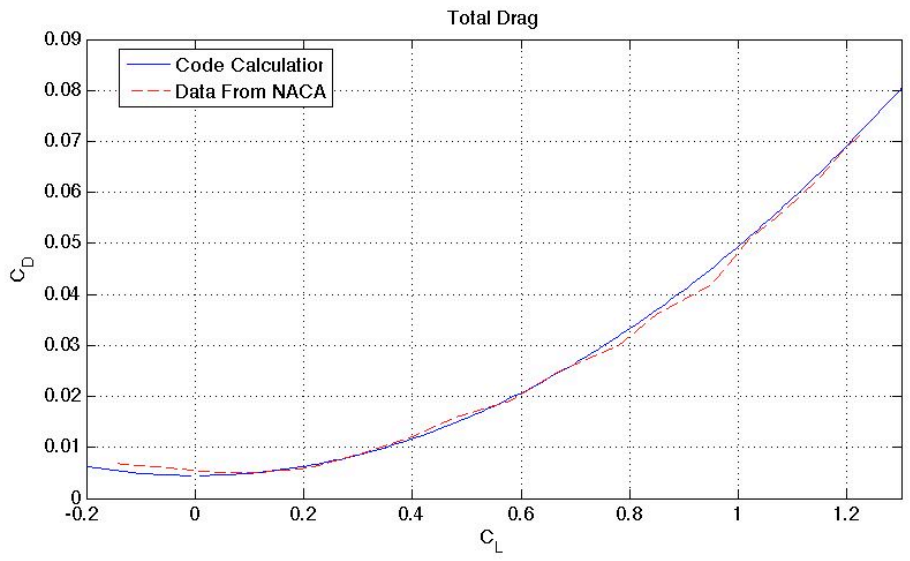

Figure 4. Estimation of drag polar and comparison to NACA data.

The code produces a result that agrees with the results demonstrated in NACA technical note no. 1422, proving that for at least one particular airfoil, a NACA 65-210, the code works to sufficient accuracy.[3] The next step is to test the correlation with experimental data from a different airfoil. Using an FX 60-160, experimental data obtained using a force balance is compared to the output of the code. The drag polars for two different aspect ratios of 5 and 6.7 are shown in figures 5 and 6 respectively.

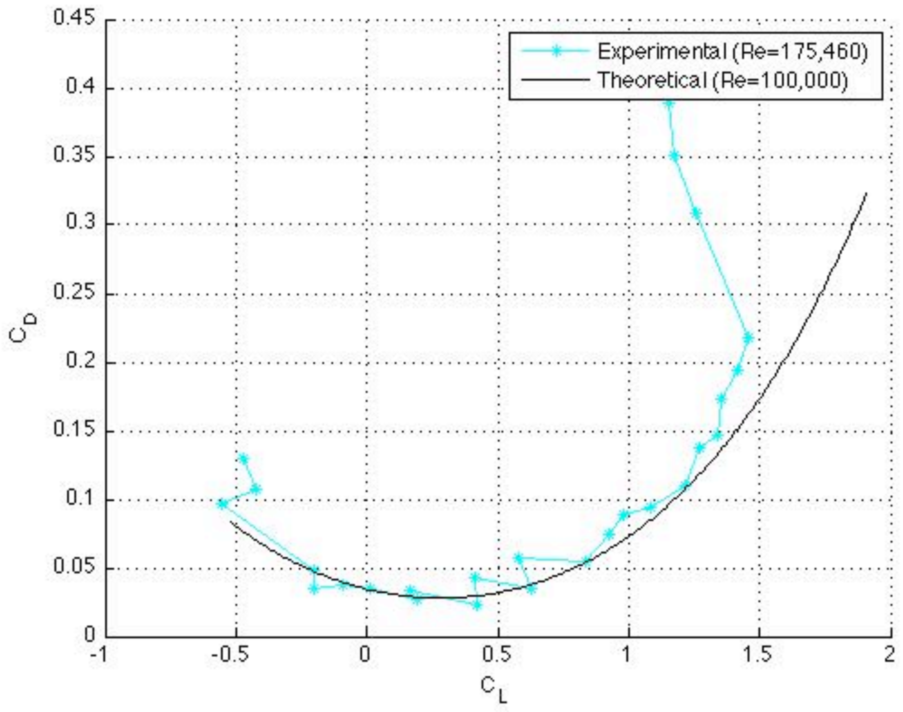

Figure 5. Drag polar comparison for aspect ratio of 5.

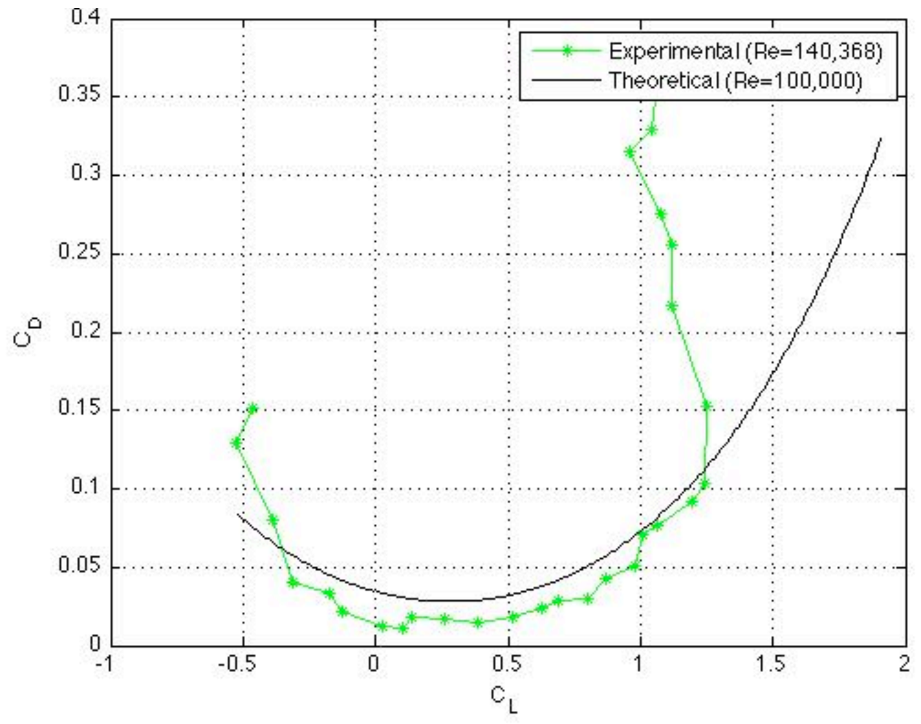

Figure 6. Drag polar comparison for aspect ratio of 6.7.

There appears to be a fairly accurate correlation between the theory and the experiment for the airfoil with an aspect ratio of five, but the correlation is less accurate for the airfoil with an aspect ratio of 6.7. The theoretical values are consistently higher until the flow separates. In order to try and determine a cause for the difference, the lift coefficient versus angle of attack were plotted for each aspect ratio, as shown in figures 7 and 8.

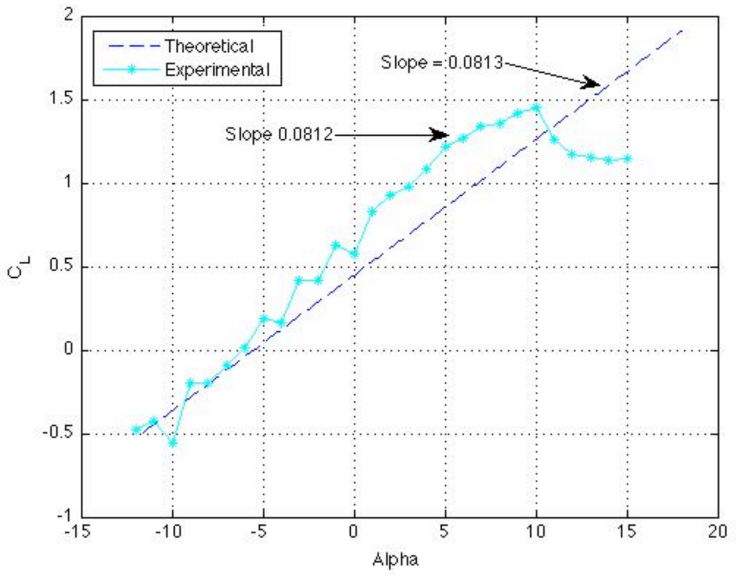

Figure 7. Lift coefficient versus alpha for an aspect ratio of 5.

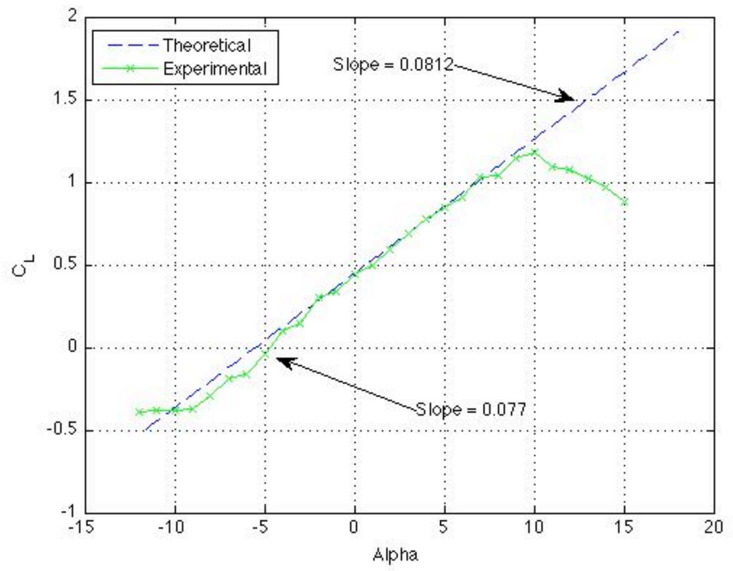

Figure 8. Lift coefficient versus alpha for an aspect ratio of 6.7.

The purpose of this comparison was to ensure that the experimental data was accurate. For the airfoil with an aspect ratio of five, the smoothest part of the experimental curve has nearly the same slope as the theoretical value, but is offset from the theoretical curve. For the airfoil with an aspect ratio of 6.7, the two curves line up fairly well, but the slopes do not agree well because of the bottom section of the experimental data where the two curves diverge. Prandtl's lifting line theory should accurately predict the coefficient of lift, and because this is not the case for the airfoil with an aspect ratio of five, it is likely that the accuracy of the total drag prediction cannot be trusted. To further examine this difference, the theoretical values for different aspect ratios of the same airfoil are plotted in figure 9.

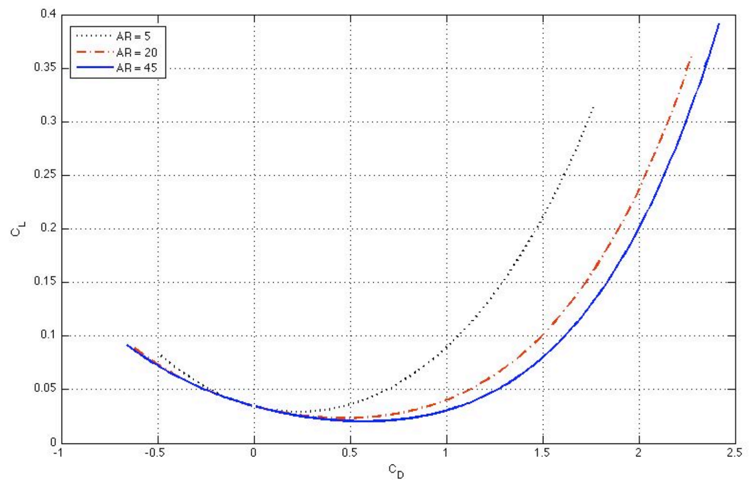

Figure 9. Theoretical drag polar for varied aspect ratios.

These curves do not appear out of the ordinary, indicating that the experimental data from the two tests is inconsistent. One final check between the theory and the experimental results is the comparison of the span efficiency factor. The plots to determine the experimental value are shown in figures 10 and 11 for the aspect ratios of 5 and 6.7 respectively.

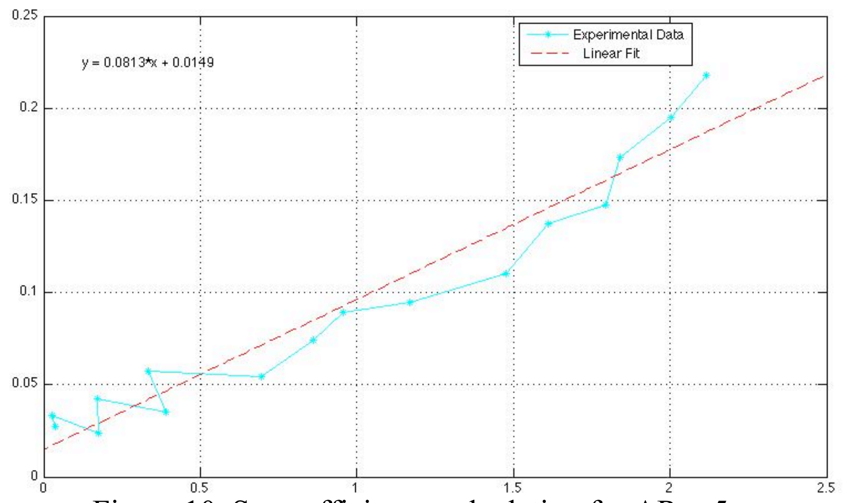

Figure 10. Span efficiency calculation for AR = 5.

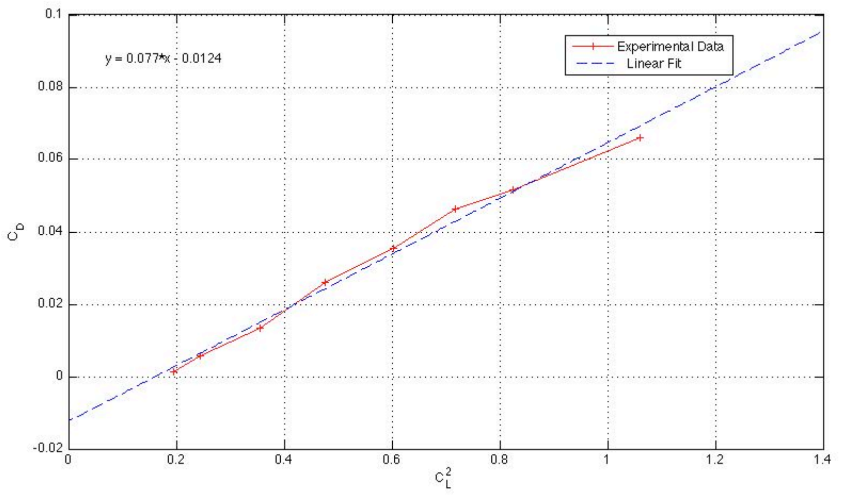

Figure 11. Span efficiency calculation for AR = 6.7.

Using equation 2, the span efficiency is calculated for the experimental data:

$$\frac{C_L^2}{C_D} = \pi A R e \quad (2),$$

where C_L is the lift coefficient, C_D is the total drag coefficient, AR is the aspect ratio, and e is the span efficiency. The span efficiency of the experimental data for the airfoil with an aspect ratio of five is 0.786. This compares fairly well to the theoretical span efficiency of 0.692. A similar result occurs with the airfoil of aspect ratio 6.7. The experimental data has a span efficiency of 0.617, while the theoretical value is 0.546.

Ideally these would be closer, but because of the fluctuations in the experimental data, the span efficiency can vary greatly from run to run.

4.0 Conclusions

Code that was originally written to model Prandtl's lifting line theory has been supplemented to model form drag as well as induced drag. The results of these additions are mixed. Although the theoretical model agrees with the experimental data reasonably well for the purposes of the senior design class, there is some uncertainty of accuracy because of inconsistent experimental data. With more experimental data, particularly with different airfoils, accuracy could be improved.

5.0 References

Bertin, John J., and Russell M. Cummings. "Incompressible Flow About Wings of Finite Span." *Aerodynamics for Engineers*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2009. 299-332. Print.

Sivells, James C.: Experimental and Calculated Characteristics of Three Wings of NACA 64-210 and 65-210 Airfoil Sections With and Without 2^0 Washout. NACA TN No. 1422, 1947.

Sivells, James C., and Neely, Robert H.: Method for Calculating Wing Characteristics by Lifting-Line Theory Using Nonlinear Section Lift Data. NACA TN No. 1269, 1947.

6.0 Appendix

```
%-----------------------------------------------------------------------
%
%  Prandtl's Lifting Line Code
%
%  Brian Neiswander
%
%  Form Drag Portions Supplemented by Matthew Gates
%  (Code prefaced by "Form Drag Stuff")
```

```
%
%
%  Calculates PLLT estimates for lift and induced drag coefficients.
%  Assumes symmetric loading (odd Fourier coefficients only).  Solves
%  monoplane equation as described in Bertin and Cummings.
%
%
%
%  Program input:
%
%           Variable      Description                         Units
%           ------------------------------------------------------------
%           N             number of chordwise spanstations
%           ao            lift curve slope                    rad/rad
%           alpha_g       geometric angle of attack           degrees
%           cr            root chord length                   meters
%           AR            wing aspect ratio
%           lambda        taper ratio
%           alpha_tr      alpha_twist at root                 degrees
%           alpha_tt      alpha_twist at tip                  degrees
%           alpha_l0r     Z.L.L angle of attack at root       degrees
%           alpha_l0t     Z.L.L angle of attack at tip        degrees
%           holdonflag    bool flag 0=new figures, 1=hold on  [0 or 1]
%           plotstyle     string containing plot style        string
%
%
%                 .----------------------------------------.
%                 |                   |                    |
%                 |   Port Wing       |  Starboard wing    |
%                 .----------------------------------------.
%     theta :     0                 pi/2                 pi
%     y     :    -s                   0                   s
%                 |--Solution Region--|
%
%
%  Solves lifting line equation for the Fourier coefficients, [An], where
%
%       [A] = [B] * [An]
%
%   so that
%
%       [An]  = [A]\[B].
%
%--------------------------------------------------------------------------

%clear all;



%input parameters (adjust these) -----------------------------------------
%you can use this for loop to vary some of the characteristics of the wing.
%Perhaps the most useful is the way it is set up now, with the angle of
%attack varying.
for j=1:300
N               =   100;
%Form Drag Stuff..........................................................
bucket          =   0; %Set to 1 if there is a drag bucket
db              =   [-2 .007 5 .007];%[-.05 .006 0.45 .006]; %drag bucket
bounds (smaller Cl first) (CL1 CD1 CL2 CD2)
dbz             =   [.2 .0045];%[.2 .004]; %drag bucket center (Cl,Cd)
aft             =   60126; %airfoil polar .dat file name (must be only
numbers,specify prefix later)
%.........................................................................
ao              =   2*pi; % 2D Lift curve slope
alpha_g         =   -12+.1*j; % degrees
```

```
b                =   1.5;% m
AR               =   6.7;%*(aspr^2); % Aspect ratio  AR=b^2/SW
SW               =   .3358; % Wing area m^2
lambda           =   1+0*j;
cr               =   8.4*2.54/100; %2.0*SW/(b*(1+lambda)); %root chord meters
alpha_tr         =   0;
alpha_tt         =   0;
alpha_l0r        =   -5.5;
alpha_l0t        =   -5.5;
holdonflag       =   0;
plotstyle        =   'k-*';

%--------------------------------------------------------------------------



%intermediate calculations ------------------------------------------------

%set up vectors
n                =   1:2:2*N;                        %index
phi              =   linspace(0,pi/2,N+1);           %spanstations [radians]
phi1             =   phi;                            %spanstions + wingtip
phi              =   phi(2:end);                     %ignore wingtip
s                =   AR*cr*(1+lambda)/4;             %wing halfspan [meters]
y                =   -s*cos(phi);                    %spanstations [meters]
y1               =   phi1/(pi/2)*7.5;                %spanstations + wingtip
c                =   cr*(1+(lambda-1)*cos(phi));     %chord lengths [meters]
c1               =   cr*(1+(lambda-1)*cos(phi1+(phi1(2)-phi1(1))./2));

%Form Drag Stuff............................................................
%calculate area of each component
sp               =   zeros(length(c)+1,1);
spa              =   zeros(length(c),1);
sp(1)            =   y1(2)./4*(c1(length(c))+cr);
spa(1)           =   sp(1);
for i=1:length(c)-1
    spa(i+1)     =   (y1(2)/2+y1(i+1))./2*(c1(length(c)-i)+cr);
    sp(i+1)      =   spa(i+1)-spa(i);
end
sp(length(c)+1) =   y1(length(y1))./2*(lambda*cr+cr)-spa(length(c));
%...........................................................................

%calculate angle of attack
alpha_t          =   ((alpha_tr-alpha_tt)/s*y+alpha_tr);
alpha_l0         =   ((alpha_l0r-alpha_l0t)/s*y+alpha_l0r);
alpha            =   (alpha_g+alpha_t-alpha_l0)';

%set up matrices
n                =   meshgrid(n,n);                  %size [NxN]
phi              =   meshgrid(phi,phi)';             %size [NxN]
c                =   meshgrid(c,c);                  %size [NxN]

%monoplane coefficient matrix
mu               =   ao*(1+(lambda-1).*cos(phi))./(2*(1+lambda)*AR);

%create [A] matrix
A                =   sin(n.*phi).*(mu.*n+sin(phi));

%create [B] matrix
B                =   mu(:,1).*alpha*pi/180.*sin(phi(:,1));

%calculate [B] solution matrix
An               =   A\B;

%--------------------------------------------------------------------------
```

```
%post calculations -------------------------------------------------------

CL               =   An(1)*pi*AR; % Wing Lift Coeficient
CD               =   CL^2/(pi*AR)*sum(n(1,:)*(An./An(1)).^2); % Induced Drag
Coefficient
e                =   CL^2/(pi*AR*CD); % Wing Span Efficiency factor
GammaST          =   zeros(1,size(phi,1));
for ai  = 1:length(GammaST)
    GammaST(ai) =   4*s*sum(An'.*sin(n(ai,:).*phi(ai,:)));
end
Cl               =   2*GammaST./c(1,:);

%-----------------------------------------------------------------------

%Calculation of effective AOA---------------------------------------------

%Form drag stuff.........................................................
%Local effective angle of attack
for i=1:length(Cl)
aeff(i)            =    alpha(i) - Cl(length(Cl)-i+1)./pi/AR*180/pi;
end
%Local induced drag calculation
cdi                =    CL*(transpose(alpha)-aeff)*pi/180;
%.......................................................................

%-----------------------------------------------------------------------

%Form drag stuff.........................................................
%Calculations of form drag, approximating with 4th degree polynomial------
% It is a good idea to check that this polynomial is indeed the best fit
% for drag polar. If the file is not a naca airfoil, change the prefix in
% the line below.
aft1             =    load(strcat('naca',num2str(aft),'.dat'));
x                =    polyfit(aft1(:,1),aft1(:,2),4);

%Please note that if you changed the order of the polynomial in the above
%line, you will need to change the following lines to accomodate them.
cdo = zeros(length(Cl),1);
for i=1:length(Cl)
    mcd          =    (db(4)-dbz(2))/(db(3)-dbz(1)).^4;
    if bucket == 1;
        if Cl(i)> db(3) || Cl(i)< db(1)
            cdo(i)=
x(1)*Cl(i).^4+x(2)*Cl(i).^3+x(3)*Cl(i).^2+x(4)*Cl(i)+x(5);
        %Drag bucket
        else
           cdo(i)=    mcd*(Cl(i)-dbz(1)).^4+dbz(2);
        end
    %For no drag bucket
    else
         cdo(i)=    x(1)*Cl(i).^4+x(2)*Cl(i).^3+x(3)*Cl(i).^2+x(4)*Cl(i)+x(5);
    end
end


%Summation of form drag along span
for i = 1:length(Cl)
    cdo1(i)    =    (cdo(i)*sp(i))./sum(sp);
end

cdof(j)        =    sum(cdo1);
%.......................................................................
```

```
%--------------------------------------------------------------------------

%display results ----------------------------------------------------------

fprintf('\n');
fprintf('User Input\n');
fprintf('----------------------------\n');
fprintf(' N            =   %i\n',N);
fprintf(' ao           =   %f\n',ao);
fprintf(' cr           =   %f meters\n',cr)
fprintf(' AR           =   %f meters\n',AR)
fprintf(' lambda       =   %f meters\n',lambda)
fprintf(' alpha_g      =   %f degrees\n',alpha_g);
fprintf(' alpha_tr     =   %f degrees\n',alpha_tr(1));
fprintf(' alpha_tt     =   %f degrees\n',alpha_tt(1));
fprintf(' alpha_l0r    =   %f degrees\n',alpha_l0r(1));
fprintf(' alpha_l0t    =   %f degrees\n',alpha_l0t(end));
fprintf('\n');
fprintf('Fourier Coefficients\n');
fprintf('----------------------------\n');
for ai=1:length(An)
    fprintf(' An(%i)     = %4.6e\n',ai,An(ai));
end
fprintf('\n');
fprintf('Post Calculations\n');
fprintf('----------------------------\n');
fprintf('  CL     = %4.6f\n',CL);
fprintf('  CDi    = %4.6f\n',CD);
fprintf('\n');

%--------------------------------------------------------------------------



%make plots ---------------------------------------------------------------

% These plots are for a specific angle of attack. Uncomment them if you
% want to view individual plots, otherwise leave them commented so you
% don't have an uncontrollable number of plots

% if holdonflag
%     figure(1);hold on;
% else
%     figure('Color',[1 1 1],'Name','Spanwise Chord Length');
% end
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on');
% plot(-y/s,c(1,:),plotstyle);
% xlabel('-y/s','FontSize',16);
% ylabel('Chord length, m','FontSize',16);
% title('Spanwise Chord Length','FontSize',16);
% grid on;
%
% if holdonflag
%     figure(2);hold on;
% else
%     figure('Color',[1 1 1],'Name','Angles of Attack');
% end
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on');
% plot(-y/s,alpha_t,plotstyle); hold on;
% plot(-y/s,alpha_l0,'k-d');
% plot(-y/s,alpha,'k-o');
% xlabel('-y/s','FontSize',16);
% ylabel('Degrees','FontSize',16);
% title('Angles of Attack','FontSize',16);
% legend({'\alpha_t','\alpha_{l0}','\alpha_{eff}'},'FontSize',14, ...
%     'Location','EastOutside');
% grid on;
%
% if holdonflag
%     figure(3);hold on;
% else
%     figure('Color',[1 1 1],'Name','Scaled Spanwise Circulation');
% end
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on');
% plot(-y/s,GammaST,plotstyle);
% xlabel('-y/s','FontSize',16);
% ylabel('\Gamma / U\infty','FontSize',16);
% title('Scaled Spanwise Circulation','FontSize',16);
% grid on;
%
% if holdonflag
%     figure(4);hold on;
% else
%     figure('Color',[1 1 1],'Name','Local Lift Coefficient Spanwise
Distribution');
% end
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on');
% plot(-y/s,Cl/CL,plotstyle);
% xlabel('-y/s','FontSize',16);
% ylabel('C_l / C_L','FontSize',16);
% title('Local Lift Coefficient Spanwise Distribution','FontSize',16);
% grid on;
CLL(j)=CL;
CDi(j)=CD;
delta(j)=(pi*AR*CD/CL^2)-1;
Spaneff(j)=1/(1+delta(j));
Lambda(j)=lambda;
CDt(j)=CDi(j)+cdof(j);
XDX = [transpose(CLL),transpose(CDt)];
% %pause
% figure('Color',[1 1 1],'Name','Local Effective Angle of Attack')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(-y/s,aeff,'k.')
% grid on
% xlabel('-y/s')
% ylabel('Alpha Effective')
% title('Local Effective Angle of Attack')
%
% figure('Color',[1 1 1],'Name','Local Lift Coefficient')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(-y/s,Cl,'k.')
% grid on
% xlabel('-y/s')
% ylabel('Cl')
% title('Local Lift Coefficient')

% figure('Color',[1 1 1],'Name','Form Drag')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(cdo,Cl,'k.')
% grid on
% xlabel('C_D_o')
% ylabel('C_L')
% title('Form Drag')
%
% figure('Color',[1 1 1],'Name','Induced Drag')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(cdi,Cl,'k.')
% grid on
% xlabel('C_D_i')
% ylabel('C_L')
% title('Induced Drag')
%
% figure('Color',[1 1 1],'Name','Induced Drag')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(-y/s,cdi,'k.')
% grid on
% xlabel('-y/s')
% ylabel('C_D_i')
% title('Induced Drag')
%
% figure('Color',[1 1 1],'Name','Form Drag')
% set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
% plot(-y/s,cdo,'k.')
% grid on
% xlabel('-y/s')
% ylabel('C_D_o')
% title('Form Drag')


end

% figure
% plot(Lambda,delta)
% xlabel('Lambda')
% ylabel('Delta')
% title('delta versus Lambda')
% %pause
%
% plot(Lambda,Spaneff)
% xlabel('Lambda')
% ylabel('e')
% title('Span efficiency -e- versus Lambda')

figure('Color',[1 1 1],'Name','Total Drag')
set(gca,'Color',[1 1 1],'FontSize',14,'box','on')
plot(CLL,CDt,'.')
grid on
xlabel('C_L')
ylabel('C_D')
title('Total Drag')
%------------------------------------------------------------------------
```