A.1.a File Name: main.m

```
% Optimum Distance Searching for Maximum Induced Drag Reduction Using Vortex Lattice Method
% This program shows the average induced drag reduction of the formation in the distance in lateral and vertical.
% Using method four: lattice averaging

% This one is the initial one which is used to check the program output by comparing Example 7.2 of
% Aerodynamics for Engineers.

% Written by Woon-Ho Cho, woonhocho@yahoo.com
% Mechanical and Aerospace Engineering, San Jose State University, San Jose, California, U.S.A.

% February 14, 2006 - 16 - March 2 - 23 - 24 - April 1 - 2 - 28 - May 4(minor touch)

% Problems
% 1. Drag of airplane is incorrect.
% 2.
% 3.

% Solutions
% 1. Some arbitrary positions in space make the denomiatior of V_AB, V_Ainf or V_Binf. It is shown in the second
% figure. => Linearization at the singularity points.

% Input: Wing Geometry: span, Sweep angle, aspect ratio and taper ratio.
%           Vortex Lattice: number of vortex lattice in spanwise and chordwise.
%           Circulation: free stream velocity and angle of attack.

% Output: down-wash and the change of aerodynamic characteristics.

% The optimum distance is -0.93 ~ -0.94b and -1.1 ~ -1.2b in lateral and longitudinal, respectively. Strangely, the
% vertical position does not affect.

% Note: "clear all" command is needed whenever this program is ran.
%       Be careful for unit.
%         Longitudinal positions are the same in each level.
%         VLM number decides the calculation time.

% fartherone: the far most wing geometry position
% whichoneisfarther: operator to find the far most wing geometry position
%-------------------------------------------------------------------------------------------------------------------

% Sub function files: inducedownwashcpwholeaverage.m:
%                           findingcirculation.m:
%                           trailingcp.m:
%                           inducedownwashbytrailingcp.m:

% Reference: Margason & Lamar NASA TN D-6142 1971 (Margason, R.J., and
% "Vortex-Lattice FORTRAN Program for Estimating Subsonic Aerodynamic
% Characteristics of Complex Planforms,"
% Lamar & Gloss NASA TN D-7921 1975
% Lamar & Herbert NASA TM 83303 1982

clear all;
tic;

%------------------------------
% A. Set the Flight Condition.

rho=0.0186;             % This is density 0.06243 in lb_m/ft^3 at sea level 0.0186 at 36,000ft.
degreetorad=pi/180;
V_inf=959.8747;     % Cruise speed is 200 kts(320ft/s), (1 kts = 1.687810 ft/s) and it should be under Mach 0.3
                        % (334 ft/s) not
angle=1.0;              % to violate the incompressible flow assumption. Boeing 747-400 stall speed is 110kts.
wingsection=2;          % Mach 0.86 is 959.8747ft/s
span=[63 126]; Lambda=[45 45]; AR=[4.5]; taperatio=[0.8 0.5];   % Boeing 747-400
%span=[37.6]; Lambda=[20]; AR=[3.52]; taperatio=[0.42];         % F/A-18
%span=[48.7]; Lambda=[0]; AR=[17.5]; taperatio=[0.45];          % The matrix should match to the section number.
numvortexlatticechordwise=[2 2 2 2 2 2 2 2 2];                 % These value is used to make vortex lattice.
numvortexlatticespanwise=[32 32 32 32 32 32 32 32 32];         % The control point of the test aircraft should be dens
                                                    % than vortex lattice density. The VLM in spanwise should be
                                                    % divided by the
```

```
                                                % than vortex lattice density. The VLM in spanwise should be
                                                % divided by the
                                                % wingsection to use in
                                                % the
                                                % vortexlatticelocation
                                                % subfunction.

%-------------------------------------------------------------------------------------------------------------------
% D. Finding wind blowing at control points in the space.

ss=size(span);                                  % This is the wing section number.
ss=ss(1,2);
SUMofb=span*ones(ss,1);
b=SUMofb;
numberofairplane=2
leadingairplane=1;
firstestedairplane=2;
secondtestedairplane=3;
thirdtestedairplane=4;
fourthtestedairplane=5;
fifthtestedairplane=6;
sixthtestedairplane=7;
distanceinx=[0.00 -3.00 -1.00 -2.00 -2.00 -3.00 -3.00 -4.00 -4.00]*b;
distanceiny=[0.00 -0.95  0.94 -1.84  1.84 -2.76  2.76 -3.68  3.68]*b;
distanceinz=[0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00]*b;
% Second airplane relatived to the first airplane. It should not be behind of the trailing airplances.

% This part compares the root chord lenght with wingspan.
n=1;
[x,y,z,rootLEtest,rootTEtest,tipLEtest,tipTEtest, ...
      yrightend,xrightend,zrightend,yleftend,xleftend,zleftend,leadingedgex, ...
      leadingedgey,trailingedgex,trailingedgey] = ...
           trailingcp(wingsection,span,Lambda,taperatio,AR,distanceinx(n),distanceiny(n),distanceinz(n), ...
           numvortexlatticespanwise(n),numvortexlatticechordwise(n));
       % The output variable name should be matched to input variables name of downwash calculation subroutine.
sizeLEX=size(leadingedgex);
rootxcoordinationlocation=sizeLEX(1,2)/2;
rootchordlength=abs(leadingedgex(rootxcoordinationlocation)-trailingedgex(rootxcoordinationlocation));
if rootchordlength >= b & trailingedgex(1) <= -2*b
    'The case is fault.'
else
    'The case is good.'
end

%-------------------------------------------------------------------------
% D.1 Calculating the drag reduction of each airplan.

distancemaxiny= 0.00*b;
distancemininy=-1.00*b;
distancemaxinx=-abs(max(leadingedgex)-min(trailingedgex));
% abs(max(leadingedgex)-min(trailingedgex)) is the maximum height of wing.
distancemininx=-5.00*b;
distancemaxinz= 0.00*b;
distancemininz=-1.00*b;
lmax=50                                  % It also decides the calculation time.
kmax=50

sectionnumber=32;
optimumVLMlayerchord=2;
nvls=wingsection*sectionnumber              % These set the density of control points.
nvlc=optimumVLMlayerchord                   % It should be denser than VLM distribution.

for l=1:lmax,
    distanceiny(firstestedairplane) = ( (lmax-l)*distancemininy+l*distancemaxiny ) / lmax;
    %distanceiny(secondtestedairplane) = -( (lmax-l)*distancemininy+l*distancemaxiny ) / lmax;
    %longitudinalseparation(l)=distanceinx(firstestedairplane)/b;
    lateralseparation(l)=distanceiny(firstestedairplane)/b;
    for k=1:kmax,
        distanceinz(firstestedairplane) = ( (kmax-k)*distancemininz+k*distancemaxinz ) / kmax;
        verticalseparation(k)=distanceinz(firstestedairplane)/b;
```

```
for n=1:numberofairplane,       % It selects an affected aircraft.

    %-------------------------------------------------------------------
    % D.2 Set the testing airplane control points.

    [x,y,z,rootLEtest,rootTEtest,tipLEtest,tipTEtest, ...
            yrightend,xrightend,zrightend,yleftend,xleftend,zleftend,leadingedgex, ...
            leadingedgey,trailingedgex,trailingedgey] = ...
        trailingcp(wingsection,span,Lambda,taperatio,AR,distanceinx(n),distanceiny(n),distanceinz(n),nvls,nvlc)

    xspace=x;          % x is chordwise.
    yspace=y;          % y is spanwise. The range should be separated since the vector is opposite direction
    % in the result of vector cross product.
    zspace=z;          % z is toward to the center of Earth.
    w(:,:,n)=zeros(nvlc,2*nvls);
    % It initializes the wind velocity.

    % Under loop finds the induced wind in the nth airplane by the other members.
    for m=1:numberofairplane,
        inducedownwash(:,:,m)=zeros(nvlc,2*nvls);
        % It initializes the wind velocity.
        [xvlm yvlm zvlm rootLE Circulation]=vortexlatticelocation...
            (wingsection,span,Lambda,taperatio,AR,distanceinx(m),distanceiny(m),distanceinz(m),...
            numvortexlatticespanwise(m),numvortexlatticechordwise(m),...
            rho,V_inf,angle);
        % It sets the location of test airplane control points location.

        if m==n,
            % It does not count its own effect.
            inducedownwash(:,:,m)=zeros(nvlc,2*nvls);
        elseif mod(n,2)==0,
            % If the test airplane's location is even, it considers very next airplane's trailing vortex.
            if m<=n+1,  % It is leading aircraft effect.
                wbysubroutine=downwashcplatticeaverage ...
                    (x,y,z,yleftend,yrightend,xleftend,xrightend,zleftend,zrightend,xvlm,yvlm,zvlm,rootLE,Circ
                inducedownwash(:,:,m)=wbysubroutine;
            else
                wbysubroutine=downwashbytrailingcp(xspace,yspace,zspace,xvlm,yvlm,zvlm,rootLE,Circulation);
                inducedownwash(:,:,m)=wbysubroutine;
            end
        else
            % If the test airplane's location is odd, it considers the previous member's trailing vortex.
            if m<n,
                wbysubroutine=downwashcplatticeaverage ...
                    (x,y,z,yleftend,yrightend,xleftend,xrightend,zleftend,zrightend,xvlm,yvlm,zvlm,rootLE,Circ
                inducedownwash(:,:,m)=wbysubroutine;
            else
                wbysubroutine=downwashbytrailingcp(xspace,yspace,zspace,xvlm,yvlm,zvlm,rootLE,Circulation);
                inducedownwash(:,:,m)=wbysubroutine;
            end
        end
        w(:,:,n)=w(:,:,n)+inducedownwash(:,:,m);            % It is sum of whole effect of the others.
    end

    %-------------------------------------------------------------------
    % D.3 Figuring out the aerodynamic characteristics.

    % The data should be the tested aircraft's.
    % VLM number should be the same number which is used in the downwash calculation control point.
    [C_L_solo D_v_solo C_Di_solo e S]=inducedragsolo(rho,wingsection,span,Lambda,AR,taperatio,...
        nvls,nvlc,V_inf,angle);
    % The vortex lattice number should be matched to downwash calculating
    % subroutine.

    c_t=5.0e-4;          % TSFC (lb of fuel / lb of thrust x hour) PW 4052-4460: Cruise SFC = 0.537 (1.4917e-4
    W_max=34692;          % Boeing 747 3,893,589N 747-400 964,866lb F/A-18 23,832+10,860lb Predator 1020kg
    W_min=23832;        % Boeing 747 2,191,829N 747-400 524,500lb F/A-18 fuel 10,860lb Predator 431kg fuel 120
    % Boeing 747 169,039N PW4056 thrust 56,750lb
```

```
            C_L=C_L_solo;
            LoverD=10;            % This is the maximum value. Boeing 747 18 F/A-18 10 at Mach 0.8
            C_D=C_L/LoverD;
            C_D_0=C_D-C_Di_solo;
            R_solo=2*sqrt(2/(rho*S))*1/c_t*C_L^0.5/(C_D)*(W_max^0.5-W_min^0.5);
            winds=w(:,:,n);     % The wind from the other airplanes.
            [C_L_formation D_v_formation C_Di_formation e]=inducedragformation(rho,wingsection,span,Lambda,AR,taperati
                nvls,nvlc,V_inf,angle,winds);
            % The vortex lattice number should be matched to downwash calculating
            % subroutine.

            C_L=C_L_formation;
            NewC_D = C_D - C_Di_solo + C_Di_formation;
            DragChange(n)=(NewC_D-C_D)/C_D*100;
            R_formation=2*sqrt(2/(rho*S))*1/c_t*C_L^0.5/(NewC_D)*(W_max^0.5-W_min^0.5);
            VortexInducedDrag=[D_v_solo D_v_formation];
            InducedDragCoefficient=[C_Di_solo C_Di_formation];
            JetPlaneRange=[R_solo R_formation]/5280/1.15;
            RangeChange(n)=(R_formation-R_solo)/R_solo*100;
            InduceDragChange(n)=(D_v_formation-D_v_solo)/D_v_solo*100;
        end

        sf=size(InduceDragChange);                                    % This is the wing section number.
        sf=sf(1,2);
        SUMofInduceDragChange=InduceDragChange*ones(sf,1);
        SUMofDragChange=DragChange*ones(sf,1);
        SUMofRangeChange=RangeChange*ones(sf,1);
        AverageInduceDragChange(k,l)=SUMofInduceDragChange/numberofairplane;
        AverageDragChange(k,l)=SUMofDragChange/numberofairplane;
        AverageRangeChange(k,l)=SUMofRangeChange/numberofairplane;
        FirstAirplaneIDC(k,l)=InduceDragChange(leadingairplane);
        SecondAirplaneIDC(k,l)=InduceDragChange(firstestedairplane);
        SecondAirplaneDC(k,l)=DragChange(firstestedairplane);
        SecondAirplaneRC(k,l)=RangeChange(firstestedairplane);
    end
end

%-------------------------------------------------------------------------
% E. Plot the induced drag reduction

%figure(2), plot(lateralseparation,AverageReduction,'bs-', ...
%   lateralseparation,FirstAirplaneReduction,'g',lateralseparation,SecondAirplaneReduction,'r'), grid on;
%legend('Linearing the Average','Reduction of Second Airplane','Reduction of First Airplane');
%xlabel('Distance Between Fuselage Centerlines','FontSize',15), ylabel('Induced Drag Change [%]','FontSize',15)
%figure(3), plot(longitudinalseparation,AverageReduction,'bd-'), grid on;
%legend('Linearing','Actual Value');
%xlabel('Longitudinal Separation between Leading and Trailing Aircraft','FontSize',15), ylabel('Induced Drag Change [%]
% The figure(1) is made in findingcirculation sub-function.
% AverageDragChange
% AverageRangeChange
% SecondAirplaneReduction
% AverageReduction
figure(4),[c,h] = contour(lateralseparation,verticalseparation,SecondAirplaneRC), clabel(c,h), colorbar
% The transpose is used to plot the lateral separation in X-axis.
title('Flight Range Change as a Function of Lateral and Vertical Separation [%]','FontSize',15);
xlabel('Lateral Separation [wingspan]','FontSize',15);
ylabel('Vertical Separtaion [wingspan]','FontSize',15)';
toc         % It measures the program running time.
```

A.1.b File Name: downwashcplatticeaverage.m

```
% This is down-wash calculation subroutine at the specific points defined within x, y and z.
% It is different from the downwashcp sub-function by averaging wind over each lattice. It averages between two
% points of the each end of
% the vortex lattice where the control point are located. This the fifth downwash calculation program because
% the control point one has inifinite, whole average one does not match to the other program, and section averinging
% has incorrect interval (the accuracy does not checked).
% xvlm, yvlm and zvlm is the vortex lattice edge points of an effecting aircraft.
% latticex, latticey and latticez have the coordination of each end point of ~

% Circulation is a circulation strenght of each vortex lattice.

% Note: y gap one over 5,000 take at least two hours half.
%     do not use Circulation, which is Matlab built in function.

% Revised March 10 - 11 - 25 - 26 - 31 - April 20(minor touch)
% Problem: the vortex core still has not feasible value.
% Solution: use average value: 1) whole area 2) downwash and upwash region separately

function [w] = downwashcplatticeaverage ...
   (x,y,z,yleftend,yrightend,xleftend,xrightend,zleftend,zrightend,xvlm,yvlm,zvlm,rootLE,Circulation)

%function [V_AinfTest, V_BinfTest, V_Ainf_portTest, V_Binf_portTest, w] = ...
%   downwashcpwholeaverage(x,y,z,xvlm,yvlm,zvlm,rootLE,Circulation)

xspace=x;            % x is chordwise.
xrange=size(xspace);
xchord=xrange(1,1);
xspan=xrange(1,2);
yspace=y;
% y is spanwise. The range should be separated since the vector is opposite direction
% in the result of vector cross product.
zspace=z;
% Current model is symmetric to the fuselage.
% Current model is symmetric to the fuselage.
% The function input x is only starboard points and z also, but y is whole wing, so the matrix size is different.
% x and z will be mirrored to port section.
% The approximation is not affected by the adjascent value location. It may be due to the symmetric funtion.

for m=1:xspan,
   mirroredx(:,m)=xspace(:,xspan-(m-1));
   mirroredz(:,m)=zspace(:,xspan-(m-1));
end
xspace=[mirroredx xspace];
zspace=[mirroredz zspace];

yvlm_port=2*rootLE(1)-yvlm;

xgapsize=1000;                  % It is used to make the singularity position value.
ygapsize=5000;
zgapsize=1000;
sizeCir=size(Circulation);  % m is spanwise and n is chordwise.
nvlc=sizeCir(1,1);
nvls=sizeCir(1,2);
xrange=size(xspace);
xchord=xrange(1,1);
xspan=xrange(1,2);
problemstarboard=0;
problemport=0;
intervalsize=100;       % It will be used to calculate the average wind speed.
extreamspot=0;
V_inf=959.8747;         % This value should be changed manually.
possibledownwashmax=V_inf*200/(761.2*0.85);
% Boeing 747 may experience 200mph downwash at 0.85M.
%figure(4), title('Calculation Points','Fonts',15);
%axis equal,grid on;

for xn=1:xchord,        % mcp is spanwise. xspace and yspace is arbitray point in the space.
```

```
for xn=1:xchord,        % mcp is spanwise. xspace and yspace is arbitray point in the space.
    for xm=1:xspan,
        wsum=0;
        numofpoint=0;
        for q=1:1:intervalsize,
            y = ( q*yrightend(xn,xm) + (intervalsize-q)*yleftend(xn,xm) ) / intervalsize;
            x = ( q*xrightend(xn,xm) + (intervalsize-q)*xleftend(xn,xm) ) / intervalsize;
            z = ( q*zrightend(xn,xm) + (intervalsize-q)*zleftend(xn,xm) ) / intervalsize;
            % It is the point between the vortex lattice ends.
            %hold on,plot(y,x,'cs');
            % It checks the calculation point locations.
            V_ABsum=0;
            V_Ainfsum=0;
            V_Binfsum=0;
            V_ABsum_port=0;
            V_Ainfsum_port=0;
            V_Binfsum_port=0;
            for n=1:nvlc,       % nvlc is chordwise. n and m is defined within wing area becasue the points are on the
                for m=1:nvls,   % n and m depend on affecting airplane vortex panel number.
                    % V_AB(m,n) is the down-wash at one point at the space.
                    % w(n,m)+V_infsin(alpha)=0
                    % w(n,m)=Circulation(n,m)*(V_AB(n,m)+V_Ainf(n,m)+V_Binf(n,m))
                    EachCirculation=Circulation(n,m);

                    % (x-xvlm(n,m))*(y-yvlm(n,m+1))-(x-xvlm(n,m+1))*(y-yvlm(n,m)) can be zero.
                    % A is x_1n, y_1n and B is x_2n, y_2n.
                    % It is the stardboard part.
                    distancebetweenxandA=x-xvlm(n,m);
                    distancebetweenxandB=x-xvlm(n,m+1);
                    distancebetweenyandA=y-yvlm(n,m);
                    distancebetweenyandB=y-yvlm(n,m+1);
                    distancebetweenzandA=z-zvlm(n,m);
                    distancebetweenzandB=z-zvlm(n,m+1);
                    distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
                    distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
                    diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
                    diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
                    dbxA=distancebetweenxandA;
                    dbxB=distancebetweenxandB;
                    dbyA=distancebetweenyandA;
                    dbyB=distancebetweenyandB;
                    dbzA=distancebetweenzandA;
                    dbzB=distancebetweenzandB;
                    dbABx=distancebetweenAandBx;
                    dbABy=diatancebetweenAandBy;
                    dbABz=diatancebetweenAandBz;
                    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
                    distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
                    vectorcrossABandp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
                    % It can be zero at the long distance.
                    vcABp=vectorcrossABandp;
                    if vcABp==0 | distancebetweenpandA==0 | distancebetweenpandB==0
                        problemstarboard=problemstarboard+1;
                    else
                        problemstarboard=problemstarboard;
                    end

                    % Every downwash are added.
                    V_AB(xn,xm) = V_ABsum + EachCirculation/(4*pi) * -( dbxA*dbyB-dbxB*dbyA )/vcABp * ...
                        ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweenpandA - ...
                        ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweenpandB );
                    V_ABsum=V_AB(xn,xm);

                    % (yvlm(n,m)-y) and zvlm(n,m)-z can be zero, when the other stay side at the same level.
                    if  dbyA==0 & dbzA==0,
                        approaching=abs(xvlm(n,2)-xvlm(n,1))/xgapsize;
                        xprevious=xvlm(n,m)-approaching;
                        xnext=xvlm(n,m)+approaching;
```

```
    xnext=xvlm(n,m)+approaching;
    approaching=abs(yvlm(n,2)-yvlm(n,1))/ygapsize;
    yprevious=yvlm(n,m)-approaching;
    ynext=yvlm(n,m)+approaching;
    approaching=abs(zvlm(n,2)-zvlm(n,1))/zgapsize;
    zprevious=zvlm(n,m)-approaching;
    znext=zvlm(n,m)+approaching;
    distancebetweenxandA=x-xprevious;
    distancebetweenyandA=y-yprevious;
    distancebetweenzandA=z-zprevious;
    dbxA=distancebetweenxandA;
    dbyA=distancebetweenyandA;
    dbzA=distancebetweenzandA;
    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
    V_Ainfprevious = EachCirculation/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ...
        ( -1+dbxA/distancebetweenpandA );
    distancebetweenxandA=x-xnext;
    distancebetweenyandA=y-ynext;
    distancebetweenzandA=z-znext;
    dbxA=distancebetweenxandA;
    dbyA=distancebetweenyandA;
    dbzA=distancebetweenzandA;
    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
    V_Ainfnext = EachCirculation/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ...
        ( -1+dbxA/distancebetweenpandA );
    V_Ainf(xn,xm) = V_Ainfsum + 0.5*(V_Ainfprevious+V_Ainfnext);
else
    V_Ainf(xn,xm) = V_Ainfsum + EachCirculation/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ...
        ( -1+dbxA/distancebetweenpandA );
end
V_Ainfsum=V_Ainf(xn,xm);
V_AinfTest(n,m) = EachCirculation/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ...
    ( -1+dbxA/distancebetweenpandA );


if dbzB==0 & dbyB==0,
    approaching=abs(xvlm(n,2)-xvlm(n,1))/xgapsize;
    xprevious=xvlm(n,m+1)-approaching;
    xnext=xvlm(n,m+1)+approaching;
    approaching=abs(yvlm(n,2)-yvlm(n,1))/ygapsize;
    yprevious=yvlm(n,m+1)-approaching;
    ynext=yvlm(n,m+1)+approaching;
    approaching=abs(zvlm(n,2)-zvlm(n,1))/zgapsize;
    zprevious=zvlm(n,m+1)-approaching;
    znext=zvlm(n,m+1)+approaching;
    distancebetweenxandB=x-xprevious;
    distancebetweenyandB=y-yprevious;
    distancebetweenzandB=z-zprevious;
    dbxB=distancebetweenxandB;
    dbyB=distancebetweenyandB;
    dbzB=distancebetweenzandB;
    distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
    V_Binfprevious = EachCirculation/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ...
        ( -1+dbxB/distancebetweenpandB );
    distancebetweenxandB=x-xnext;
    distancebetweenyandB=y-ynext;
    distancebetweenzandB=z-znext;
    dbxB=distancebetweenxandB;
    dbyB=distancebetweenyandB;
    dbzB=distancebetweenzandB;
    distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
    V_Binfnext = EachCirculation/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ...
        ( -1+dbxB/distancebetweenpandB );
    V_Binf(xn,xm) = V_Binfsum - 0.5*(V_Binfprevious+V_Binfnext);
else
    V_Binf(xn,xm) = V_Binfsum - EachCirculation/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ...
        ( -1+dbxB/distancebetweenpandB );
end
```

```
end
V_Binfsum=V_Binf(xn,xm);
V_BinfTest(n,m) = - EachCirculation/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ...
    ( -1+dbxB/distancebetweenpandB );   % This is need to check the calculation.

% It is the port part.
distancebetweenxandA=x-xvlm(n,m);
distancebetweenxandB=x-xvlm(n,m+1);
distancebetweenyandA=y-yvlm_port(n,m);
distancebetweenyandB=y-yvlm_port(n,m+1);
distancebetweenzandA=z-zvlm(n,m);
distancebetweenzandB=z-zvlm(n,m+1);
distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
diatancebetweenAandBy=yvlm_port(n,m+1)-yvlm_port(n,m);
diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
dbxA=distancebetweenxandA;
dbxB=distancebetweenxandB;
dbyA=distancebetweenyandA;
dbyB=distancebetweenyandB;
dbzA=distancebetweenzandA;
dbzB=distancebetweenzandB;
dbABx=distancebetweenAandBx;
dbABy=diatancebetweenAandBy;
dbABz=diatancebetweenAandBz;
distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
vectorcrossABandp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
% It can be zero at the long distance.
vcABp=vectorcrossABandp;
if vcABp==0
    problemport=problemport+1;
else
    problemport=problemport;
end
end

V_AB_port(xn,xm) = V_ABsum_port + EachCirculation/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABp * ...
    ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweenpandA - ...
    ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweenpandB );
V_ABsum_port=V_AB_port(xn,xm);

% (yvlm(n,m)-y) and zvlm(n,m)-z can be zero, when the other stay side at the same level.
if  dbyA==0 & dbzA==0,
    approaching=abs(xvlm(n,2)-xvlm(n,1))/xgapsize;
    xprevious=xvlm(n,m)-approaching;
    xnext=xvlm(n,m)+approaching;
    approaching=abs(yvlm_port(n,2)-yvlm_port(n,1))/ygapsize;
    yprevious=yvlm_port(n,m)-approaching;
    ynext=yvlm_port(n,m)+approaching;
    approaching=abs(zvlm(n,2)-zvlm(n,1))/zgapsize;
    zprevious=zvlm(n,m)-approaching;
    znext=zvlm(n,m)+approaching;
    distancebetweenxandA=x-xprevious;
    distancebetweenyandA=y-yprevious;
    distancebetweenzandA=z-zprevious;
    dbxA=distancebetweenxandA;
    dbyA=distancebetweenyandA;
    dbzA=distancebetweenzandA;
    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
    V_Ainf_portprevious = EachCirculation/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ...
        ( -1+dbxA/distancebetweenpandA );
    distancebetweenxandA=x-xnext;
    distancebetweenyandA=y-ynext;
    distancebetweenzandA=z-znext;
    dbxA=distancebetweenxandA;
    dbyA=distancebetweenyandA;
    dbzA=distancebetweenzandA;
    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
```

```
                    distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
                    V_Ainf_portnext = EachCirculation/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ...
                        ( -1+dbxA/distancebetweenpandA );
                    V_Ainf_port(xn,xm) = V_Ainfsum + 0.5*(V_Ainf_portprevious+V_Ainf_portnext);
                else
                    V_Ainf_port(xn,xm) = V_Ainfsum_port + EachCirculation/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ...
                        ( -1+dbxA/distancebetweenpandA );
                end
                V_Ainfsum_port=V_Ainf_port(xn,xm);
                V_Ainf_portTest(n,m) = + EachCirculation/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ...
                    ( -1+dbxA/distancebetweenpandA );

                if dbzB==0 & dbyB==0,
                    approaching=abs(xvlm(n,2)-xvlm(n,1))/xgapsize;
                    xprevious=xvlm(n,m+1)-approaching;
                    xnext=xvlm(n,m+1)+approaching;
                    approaching=abs(yvlm_port(n,2)-yvlm_port(n,1))/ygapsize;
                    yprevious=yvlm_port(n,m+1)-approaching;
                    ynext=yvlm_port(n,m+1)+approaching;
                    approaching=abs(zvlm(n,2)-zvlm(n,1))/zgapsize;
                    zprevious=zvlm(n,m+1)-approaching;
                    znext=zvlm(n,m+1)+approaching;
                    distancebetweenxandB=x-xprevious;
                    distancebetweenyandB=y-yprevious;
                    distancebetweenzandB=z-zprevious;
                    dbxB=distancebetweenxandB;
                    dbyB=distancebetweenyandB;
                    dbzB=distancebetweenzandB;
                    distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
                    V_Binf_portprevious = EachCirculation/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ...
                        ( -1+dbxB/distancebetweenpandB );
                    distancebetweenxandB=x-xnext;
                    distancebetweenyandB=y-ynext;
                    distancebetweenzandB=z-znext;
                    distancebetweenzandB=z-znext;
                    dbxB=distancebetweenxandB;
                    dbyB=distancebetweenyandB;
                    dbzB=distancebetweenzandB;
                    distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
                    V_Binf_portnext = EachCirculation/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ...
                        ( -1+dbxB/distancebetweenpandB );
                    V_Binf_port(xn,xm) = V_Binfsum_port - 0.5*(V_Binf_portprevious+V_Binf_portnext);
                else
                    V_Binf_port(xn,xm) = V_Binfsum_port - EachCirculation/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ...
                        ( -1+dbxB/distancebetweenpandB );
                end
                V_Binfsum_port=V_Binf_port(xn,xm);
                V_Binf_portTest(n,m) = - EachCirculation/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ...
                    ( -1+dbxB/distancebetweenpandB );

            end
        end
        w_starboard=V_ABsum+V_Ainfsum+V_Binfsum;
        w_port=V_ABsum_port+V_Ainfsum_port+V_Binfsum_port;
        % It is the effect of one leading airplane vortex lattice.
        numofpoint=numofpoint+1;
        wsum=wsum+w_starboard+w_port;
        % It is the summation of wind mangnitude on a point between wing tip to another tip.
    end
    w(xn,xm)=wsum/numofpoint;

    if abs(w(xn,xm))>=possibledownwashmax       % 200mph(292ft/s) is known the maximum downwash in reality.
        extreamspot(xn,xm)=1;
        if w(xn,xm)>=0
            w(xn,xm)=possibledownwashmax;
        else
            w(xn,xm)=-possibledownwashmax;
        end
```

```
            w_starboard=V_ABsum+V_Ainfsum+V_Binfsum;
            w_port=V_ABsum_port+V_Ainfsum_port+V_Binfsum_port;
            % It is the effect of one leading airplane vortex lattice.
            numofpoint=numofpoint+1;
            wsum=wsum+w_starboard+w_port;
            % It is the summation of wind mangnitude on a point between wing tip to another tip.
        end
        w(xn,xm)=wsum/numofpoint;

        if abs(w(xn,xm))>=possibledownwashmax       % 200mph(292ft/s) is known the maximum downwash in reality.
            extreamspot(xn,xm)=1;
            if w(xn,xm)>=0
                w(xn,xm)=possibledownwashmax;
            else
                w(xn,xm)=-possibledownwashmax;
            end
        else
            extreamspot(xn,xm)=0;
        end

    end
end
% w(n,m) is down-wash at a point in space, and n is chordwise and m is
% spanwise direction. It can be confused with eq. 7.44.

if problemstarboard ~= 0 | problemport ~=0
   problemstarboard
   problemport
else
   problemstarboard;
   problemport;
end
```

A.1.c File Name: downwashbytrailingcp.m

```
% Written November 25, 2005
% Revised January 31, 2006 - February 21, 2006
% This is down-wash calculation subroutine at the specific points defined within x, y and z.
% xvlm, yvlm and zvlm is the vortex lattice edge points of an effecting aircraft.
% Circulation is a circulation strenght of each vortex lattice.
% Note: do not use Circulation, which is Matlab built in function.

function [w]=downwashbytrailingcp(x,y,z,xvlm,yvlm,zvlm,rootLE,Circulation)

xspace=x;            % x is chordwise.
xrange=size(xspace);
xchord=xrange(1,1); % It is
xspan=xrange(1,2);
yspace=y;
% y is spanwise. The range should be separated since the vector is opposite direction
% in the result of vector cross product.
zspace=z;
% The function input x is only starboard points and z also, but y is whole wing, so the matrix size is different.
% x and z will be mirrored to port section.
for m=1:xspan,
   mirroredx(:,m)=xspace(:,xspan-(m-1));
   mirroredz(:,m)=zspace(:,xspan-(m-1));
end
xspace=[mirroredx xspace];
zspace=[mirroredz zspace];

yvlm_port=2*rootLE(1)-yvlm;

sizeCir=size(Circulation);  % m is spanwise and n is chordwise.
nvlc=sizeCir(1,1);
nvls=sizeCir(1,2);
xrange=size(xspace);
xchord=xrange(1,1);
xspan=xrange(1,2);
xspan=xrange(1,2);
problemstarboard=0;
problemport=0;
for xn=1:xchord,        % mcp is spanwise. xspace and yspace is arbitray point in the space.
   for xm=1:xspan,
      x=xspace(xn,xm);
      y=yspace(xn,xm);
      z=zspace(xn,xm);
        V_ABsum=0;
      V_Ainfsum=0;
      V_Binfsum=0;
      V_ABsum_port=0;
      V_Ainfsum_port=0;
      V_Binfsum_port=0;
      for n=1:nvlc,     % nvlc is chordwise. n and m is defined within wing area becasue the points are on the vortex
         for m=1:nvls,  % If there is only one layer, maximum n is one and sum of down-wash (eq. 7.45) is summation of
            % same n: it means only m need to be changed. yvlm is spanwise
            % and xvlm is chordwise.
            % V_AB(m,n) is the down-wash at one point at the space.
            % w(n,m)+V_infsin(alpha)=0
            % w(n,m)=Circulation(n,m)*(V_AB(n,m)+V_Ainf(n,m)+V_Binf(n,m))
            EachCirculation=Circulation(n,m);

            % (x-xvlm(n,m))*(y-yvlm(n,m+1))-(x-xvlm(n,m+1))*(y-yvlm(n,m)) can be zero.
            % A is x_1n, y_1n and B is x_2n, y_2n.
            % It is the stardboard part.
            distancebetweenxandA=x-xvlm(n,m);
            distancebetweenxandB=x-xvlm(n,m+1);
            distancebetweenyandA=y-yvlm(n,m);
            distancebetweenyandB=y-yvlm(n,m+1);
            distancebetweenzandA=z-zvlm(n,m);
            distancebetweenzandB=z-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
```

```
            diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            % It can be zero at the long distance.
            vcABp=vectorcrossABandp;
            if vcABp==0 | distancebetweenpandA==0 | distancebetweenpandB==0
               problemstarboard=problemstarboard+1;
            else
               problemstarboard=problemstarboard;
            end

            V_AB(xn,xm) = V_ABsum + EachCirculation/(4*pi) * -( dbxA*dbyB-dbxB*dbyA )/vcABp * ...
               ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweenpandA - ...
               ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweenpandB );
            V_ABsum=V_AB(xn,xm);

             % It is the port part.
            distancebetweenxandA=x-xvlm(n,m);
            distancebetweenxandB=x-xvlm(n,m+1);
            distancebetweenyandA=y-yvlm_port(n,m);
            distancebetweenyandB=y-yvlm_port(n,m+1);
            distancebetweenzandA=z-zvlm(n,m);
            distancebetweenzandB=z-zvlm(n,m+1);
            distancebetweenzandB=z-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm_port(n,m+1)-yvlm_port(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            % It can be zero at the long distance.
            vcABp=vectorcrossABandp;
                if vcABp==0
               problemport=problemport+1;
            else
               problemport=problemport;
            end

            V_AB_port(xn,xm) = V_ABsum_port + EachCirculation/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABp * ...
               ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweenpandA - ...
               ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweenpandB );
            V_ABsum_port=V_AB_port(xn,xm);

         end
      end
   end
end
```

```
            distancebetweenpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweenpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            % It can be zero at the long distance.
            vcABp=vectorcrossABandp;
                if vcABp==0
               problemport=problemport+1;
            else
               problemport=problemport;
            end

            V_AB_port(xn,xm) = V_ABsum_port + EachCirculation/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABp * ...
               ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweenpandA - ...
               ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweenpandB );
            V_ABsum_port=V_AB_port(xn,xm);

        end
      end
   end
end

w_starboard=V_AB;
w_port=V_AB_port;
w=w_starboard+w_port;
% w(n,m) is down-wash at a point in space, and n is chordwise and m is
% spanwise direction. It can be confused with eq. 7.44.
if problemstarboard ~= 0 | problemport ~=0
   problemstarboard
   problemport
else
   problemstarboard;
   problemport;
end
```

A.1.d File Name: inducedragsolo.m

```
% The code calculate the vortex drag, vortex drag coefficient and Oswald efficiency.
% This is the same findingcirculation.m file.
% Revised January 28, 2006 - 30 - February 13
function [C_L,D_v,C_Di,e,S]=inducedragsolo(rho,wingsection,span,Lambda,AR,taperatio,...
   numvortexlatticespanwise,numvortexlatticechordwise,V_inf,angle)


%-------------------------------
% A. Drawing The Wing Geometry.

degreetorad=pi/180;
Lambda=Lambda*degreetorad;

% AR = (SUM of b)^2/S => S = SUM of 2( (b_n/2(b_n/2tan(Lambda_n)+taperatio_n x c_r_n) -
% (b_n/2)^2tan(Lambda_n)/2 -
% b_n/2(b_n/2tan(Lambda_n)+(taper ratio_n-1)c_r_n)/2 )
% c_r_n = c_r_0 x taper ratio_0 x taper ratio_1 x ~ taper ratio_n
% c_r_0 = SUM of taper ratio_x (-2b_n/2taperatio_n + 2b_n/2(taper ratio_n-1)/2) = SUM of 2(b_n/2 b_n/2tan(Lambda_n)
% - (b_n/2)^2tan(Lambda_n)/2 - b_n/2 b_n/2tan(Lambda_n)/2) - S
% If there is only one section (n=0.)
%             2(b/2 b/2tan(Lambda) - (b/2)^2tan(Lambda)/2 - (b/2)^2tan(Lambda)/2) -S            b^2
% c_r = ------------------------------------------------------------------------------ <= S = -------
%                      -2b/2taperatio + 2b/2(taper ratio-1)/2                                    AR
ss=size(span);                                  % This is the wing section number.
ss=ss(1,2);
SUMofb=span*ones(ss,1);

% Possible maximum c_r_0 = 2b/AR <- S = b/2 x c_r_0 in a delta wing
% It finds the appropriate root chord length.
maxc_r_0=2*SUMofb/AR;
for c_r_0=0:maxc_r_0/1000:maxc_r_0,
   for n=1:ss,
      S=0;                                      % If the surface area is not initialized, it will keep increasing
      if n==1                                   % root chord length trial.
      if n==1                                   % root chord length trial.
         c(n)=c_r_0*taperatio(n);
         S=S+2*(span(n)/2*(c_r_0+c(n)));
      else
         c(n)=c(n-1)*taperatio(n);
         S=S+2*(span(n)/2*(c(n-1)+c(n)));
      end
   end

   if abs( (S-SUMofb^2/AR)/(SUMofb^2/AR) )<=0.01        % There will be limit to be found by trial and error.
      answerc_r=c_r_0;
      answerS=S;
      c_r_0=SUMofb/AR;
   else
      c_r_0=c_r_0;
   end
end

% Dividing the wing sectons.

S=answerS;
c_r_0=answerc_r;
rootLE=[0 c_r_0];                               % [y_1 x_1] format
rootTE=[rootLE(1)+0 rootLE(2)-c_r_0];       % [y_4 x_4]
intermediateLE=rootLE;
intermediateTE=rootTE;
for n=1:ss,
   if n==1
      c(n)=c_r_0*taperatio(n);
   else
      c(n)=c(n-1)*taperatio(n);
   end
end
for n=1:ss,                                     % It finds the leading edge and trailing edge position.
```

```
for n=1:ss,                                          % It finds the leading edge and trailing edge position.
   nexty_n=intermediateLE(2*n-1)+span(n)/2;
   nextx_n=0.5*abs(intermediateLE(2*n)-intermediateTE(2*n))+intermediateTE(2*n)-span(n)/2*tan(Lambda(n));
   % The sweep angle is at the mid-chord line.
   intermediateLE=[intermediateLE nexty_n nextx_n+0.5*c(n)];
   intermediateTE=[intermediateTE nexty_n nextx_n-0.5*c(n)];
end
tipLE=[intermediateLE(2*ss+1) intermediateLE(2*(ss+1))];                    % [y_2 x_2]
tipTE=[intermediateTE(2*ss+1) intermediateTE(2*(ss+1))];                  % [y_3 x_3]
for n=1:ss+1,
   LEx(n)=intermediateLE(2*n);
   LEy(n)=intermediateLE(2*n-1);
   TEx(n)=intermediateTE(2*n);
   TEy(n)=intermediateTE(2*n-1);
end

%-----------------------------------------------------------------------------------------------------------------------
% B. Selecting Vortex Lattice Location.

nvls=numvortexlatticespanwise;
nvlc=numvortexlatticechordwise;
sectionnumber=nvls/wingsection;
nmax=nvlc;
mmax=nvls;
for n=1:nmax,
   for m=1:mmax+1,                                                  % y is spanwise + direction, x is chordwise - dire
                                                                       % The equation is set to use section parameter
      section=ceil(m/sectionnumber);                              % has problem at the wing tip point, so there is if co
      if    m==mmax+1                                                 % part.
         y_TE(n,m)=TEy(section);                                   % y_LE, x_LE, z_LE is the boundary point of each
         y_LE(n,m)=LEy(section);                                   % lattice.
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         x_LE(n,m) = xatsectionend;
         xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
        x_TE(n,m) = xatsectionend;
      else
         y_TE(n,m) = (m-1-(section-1)*sectionnumber) * (TEy(section+1)-TEy(section))  / sectionnumber + TEy(section);
         y_LE(n,m) = (m-1-(section-1)*sectionnumber) * (LEy(section+1)-LEy(section))  / sectionnumber + LEy(section);
         xatsectionstart = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_LE(n,m) = (xatsectionend-xatsectionstart) / (LEy(section+1)-LEy(section)) * (y_LE(n,m)-LEy(section)) + xatse
         xatsectionstart = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-n)/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_TE(n,m) = (xatsectionend-xatsectionstart) / (TEy(section+1)-TEy(section)) * (y_TE(n,m)-TEy(section)) + xatse
      end
      % y_5 and x_5 are between intermediateLE and intermediateTE, and y_6 and x_6 are between intermediateLE and inter
      %                                                                                  nmax-(n-1)
      % y_5 = y_1 (y positions are not affected by the n), x_5 = ------------(x_1-x_4) + x_4
      %                                                                                    nmax
      % [y_5 x_5] ... [y_6 x_6]
      % [y_8 x_8] ... [y_7 x_7]
      % The points between the edge:
      % y_6-y_5    y-y_5                  x_6-x_5
      %-------- = --------- => x = (y-y_5)---------- + x_5 : This is a general equation.
      % x_6-x_5    x-x_5                  y_6-y_5
      midchordpoint(n,m) = 0.25*(x_LE(n,m)-x_TE(n,m))+x_TE(n,m);
      yvlm(n,m) = y_LE(n,m);                                                  % vlm is the vortex starting point of e
      xvlm(n,m) = x_TE(n,m)+0.75*(x_LE(n,m)-x_TE(n,m));
      zvlm(n,m) = 0;
   end
end

%-----------------------------------------------------------------------------------------------------------------------
% C. Finding The Strenght of Circulation.

k=1;
for n=1:nvlc,
```

```
for n=1:nvlc,
   for m=1:nvls,
      xcontrolpoint(n,m)=0.5*(midchordpoint(n,m+1)+midchordpoint(n,m));
      ycontrolpoint(n,m)=0.5*(yvlm(n,m+1)+yvlm(n,m));
      zcontrolpoint(n,m)=0;
   end
end

xcp=xcontrolpoint;
ycp=ycontrolpoint;
zcp=zcontrolpoint;

% C.1 Calculating the each vortex circulation.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise if there is only one layer.
      for m=1:nvls,     % nvlc is chordwise.
         for n=1:nvlc,  % If there is only one layer, maximum n is one and sum of down-wash (eq. 7.45) is summation of
            % same n: it means only m need to be changed.
            % V_AB(m,n) is the down-wash portion at the mth control point by the nth vortex lattice.
            % The sigularity should be considered.

            % C.1.a) It is starboard part.

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

            cpxmx1n(mcp,m)=dbxA;        % This part is for the varification with the text book example at p.273.
            cpxmx2n(mcp,m)=dbxB;
            cpymy1n(mcp,m)=dbyA;
            cpymy2n(mcp,m)=dbyB;
            cpzmz1n(mcp,m)=dbzA;
            cpzmz2n(mcp,m)=dbzB;
            x2nmx1n(mcp,m)=dbABx;
            y2nmy1n(mcp,m)=dbABy;
            z2nmz1n(mcp,m)=dbABz;
            cpm1n(mcp,m)=distancebetweencpandA;
            cpm2n(mcp,m)=distancebetweencpandB;
            cpvectorAB(mcp,m)=vcABcp;

                % This velocity is only z-direction.
            V_AB_wing(n,m,therow,mcp) = 1/(4*pi) * -( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );
            V_AB_wing_previous=V_AB_wing(n,m,therow,mcp);

            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing(n,m,therow,mcp) = 1/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
```

```
            V_Ainf_wing(n,m,therow,mcp) = 1/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
            V_Binf_wing(n,m,therow,mcp) = - 1/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB );

            % C.1.b) It is port part.

            yvlm_port(n,m)=-yvlm(n,m);
            yvlm_port(n,m+1)=-yvlm(n,m+1);

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm_port(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm_port(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm_port(n,m+1)-yvlm_port(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

                % This velocity is only z-direction.
            V_AB_wing_port(n,m,therow,mcp) = 1/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );

            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing_port(n,m,therow,mcp) = 1/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
            V_Binf_wing_port(n,m,therow,mcp) = - 1/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB
        end
      end
   end
end

% w(n,m,therow,mcp) is down-wash at the mcp-th control point on therow-th row in chordwise
% by the mth vortex lattice to spanwise on nth row in chordwise.
% m is spanwise and n is chordwise.
w_wing=V_AB_wing+V_Ainf_wing+V_Binf_wing;       % It includes 1/(4 pi b)
w_wing_starboard=w_wing;
w_wing_port=V_AB_wing_port+V_Ainf_wing_port+V_Binf_wing_port;
w_wing=w_wing_starboard+w_wing_port;

% C.1.c) Merging the downwash matrix at each control point to one matrix.
% The downwash matrix is not down-wash, but the component of each circulation (p.273).
% Each control point has m by n matrix.

nthcontrolpoint=0;      % It initialize a control point number.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise.
      nthcontrolpoint=nthcontrolpoint+1;    % It is a row and means each point.
      nthvortexlattice=0;                     % It initialize the vortex lattice number.
      for n=1:nvlc,     % nvlc is chordwise.
         for m=1:nvls,  % The vortex arrangement will be the order of row.
            nthvortexlattice=nthvortexlattice+1;        % It is a coulumn and means each circulation.
            w_wingforcalculation(nthcontrolpoint,nthvortexlattice)=w_wing(n,m,therow,mcp);
            % In this martix, the row indicates each control point, and the column means each vortex lattice.
         end
```

```
        end
      end
   end
end

angle=angle*degreetorad;
Circulation=inv(w_wingforcalculation)*-V_inf*angle*ones(nvls*nvlc,1);
% Circulation(x) is vortex strenght of mth vortex lattice on nth layer. For example, if there is six vortex lattices
% on each four layer, that x equals seven shows the strenght of the first vortex lattice on the second layer.
Circulationfromcalculation=Circulation;
% This will be used in inducedragformation.m file.

%---------------------------------------------
% C.1.c) Relocating the circulation strenght.

% Although the vortex position and control point is used n by m matrix, the circulation strength n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=Circulation;
Circulation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      Circulation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end

%-------------------------------------------------
% It verify the result of subroutine.
localspan=span/sectionnumber;                                    % The span is different in each sectio
Lift=0;
Lift=0;
for n=1:nmax,
   for m=1:mmax,
      k=ceil(m/sectionnumber);
      Lift=Lift+rho*V_inf*Circulation(n,m)*localspan(k);    % It is total lift of a wing.
   end
end
Lift;
C_L=Lift/(0.5*rho*V_inf^2*S);
% down-wash at point s = sum of 1/(4pi) A(n,s)xCirculation(n)
% 1/(4pi)A(n,s)=w_wing(s,n)
% s=mcp(mcp th control point) n=m (n th vortex lattice)
C_LTheory=1*pi*angle;
Lift=C_LTheory*(0.5*rho*V_inf^2*S);
downwashCirculation=w_wingforcalculation*Circulationfromcalculation;
% downwashCirculation(x) is down-wash velocity of mth control point on nth layer. For example, if there is
% six vortex lattices
% on each four layer, that x equals seven shows the velocity of the first control point on the second layer.
% It is only down-wash at the control points. Assumption is down-wash on the vortex panel is the same at control point.
% (eq. 7.7)
% Although the vortex position and control point is used n by m matrix, the velocity magnitude n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=downwashCirculation;
downwashCirculation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      downwashCirculation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end
```

```
% downwashCirculation(x) is down-wash velocity of mth control point on nth layer. For example, if there is
% six vortex lattices
% on each four layer, that x equals seven shows the velocity of the first control point on the second layer.
% It is only down-wash at the control points. Assumption is down-wash on the vortex panel is the same at control point
% (eq. 7.7)
% Although the vortex position and control point is used n by m matrix, the velocity magnitude n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=downwashCirculation;
downwashCirculation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      downwashCirculation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end
D_v=0;
sizedC=size(downwashCirculation);
for n=1:sizedC(1,1),
   for m=1:sizedC(1,2),
      k=ceil(m/sectionumber);
      D_v=D_v-rho*downwashCirculation(n,m)*Circulation(n,m)*localspan(k);
   end
end
% (eq. 7.14)
C_Di=D_v/(0.5*rho*V_inf^2*S);
C_DiTheory=C_LTheory^2/(pi*AR*0.99);
e=C_L^2/(pi*AR)/C_Di;
```

A.1.e File Name: inducedragformation.m

```
% The code calculate the vortex drag, vortex drag coefficient and Oswald efficiency.
% This has the some part from the findingcirculation.m file.
% Revised November 21, 2005 - 30 - February 13

function [C_L,D_v,C_Di,e]=inducedragformation(rho,wingsection,span,Lambda,AR,taperatio,...
   numvortexlatticespanwise,numvortexlatticechordwise,V_inf,angle,inducedownwash)

%------------------------------
% A. Drawing The Wing Geometry.

degreetorad=pi/180;
Lambda=Lambda*degreetorad;

% AR = (SUM of b)^2/S => S = SUM of 2( (b_n/2(b_n/2tan(Lambda_n)+taperatio_n x c_r_n) -
% (b_n/2)^2tan(Lambda_n)/2 -
% b_n/2(b_n/2tan(Lambda_n)+(taper ratio_n-1)c_r_n)/2 )
% c_r_n = c_r_0 x taper ratio_0 x taper ratio_1 x ~ taper ratio_n
% c_r_0 = SUM of taper ratio_x (-2b_n/2taperatio_n + 2b_n/2(taper ratio_n-1)/2) = SUM of 2(b_n/2 b_n/2tan(Lambda_n)
% - (b_n/2)^2tan(Lambda_n)/2 - b_n/2 b_n/2tan(Lambda_n)/2) - S
% If there is only one section (n=0.)
%             2(b/2 b/2tan(Lambda) - (b/2)^2tan(Lambda)/2 - (b/2)^2tan(Lambda)/2) -S            b^2
% c_r = ------------------------------------------------------------------------------ <= S = -------
%                      -2b/2taperatio + 2b/2(taper ratio-1)/2                                    AR
ss=size(span);                                  % This is the wing section number.
ss=ss(1,2);
SUMofb=span*ones(ss,1);

% Possible maximum c_r_0 = 2b/AR <- S = b/2 x c_r_0 in a delta wing
% It finds the appropriate root chord length.
maxc_r_0=2*SUMofb/AR;
for c_r_0=0:maxc_r_0/1000:maxc_r_0,
   for n=1:ss,
      S=0;                                      % If the surface area is not initialized, it will keep increasing :
      if n==1                                   % root chord length trial.
      if n==1                                   % root chord length trial.
         c(n)=c_r_0*taperatio(n);
         S=S+2*(span(n)/2*(c_r_0+c(n)));
      else
         c(n)=c(n-1)*taperatio(n);
         S=S+2*(span(n)/2*(c(n-1)+c(n)));
      end
   end

   if abs( (S-SUMofb^2/AR)/(SUMofb^2/AR) )<=0.01        % There will be limit to be found by trial and error.
      answerc_r=c_r_0;
      answerS=S;
      c_r_0=SUMofb/AR;
   else
      c_r_0=c_r_0;
   end
end

% Dividing the wing sectons.

S=answerS;
c_r_0=answerc_r;
rootLE=[0 c_r_0];                               % [y_1 x_1] format
rootTE=[rootLE(1)+0 rootLE(2)-c_r_0];       % [y_4 x_4]
intermediateLE=rootLE;
intermediateTE=rootTE;
for n=1:ss,
   if n==1
      c(n)=c_r_0*taperatio(n);
   else
      c(n)=c(n-1)*taperatio(n);
   end
end
for n=1:ss,                                     % It finds the leading edge and trailing edge position.
```

```
for n=1:ss,                                          % It finds the leading edge and trailing edge position.
   nexty_n=intermediateLE(2*n-1)+span(n)/2;
   nextx_n=0.5*abs(intermediateLE(2*n)-intermediateTE(2*n))+intermediateTE(2*n)-span(n)/2*tan(Lambda(n));
   % The sweep angle is at the mid-chord line.
   intermediateLE=[intermediateLE nexty_n nextx_n+0.5*c(n)];
   intermediateTE=[intermediateTE nexty_n nextx_n-0.5*c(n)];
end
tipLE=[intermediateLE(2*ss+1) intermediateLE(2*(ss+1))];                    % [y_2 x_2]
tipTE=[intermediateTE(2*ss+1) intermediateTE(2*(ss+1))];                  % [y_3 x_3]
for n=1:ss+1,
   LEx(n)=intermediateLE(2*n);
   LEy(n)=intermediateLE(2*n-1);
   TEx(n)=intermediateTE(2*n);
   TEy(n)=intermediateTE(2*n-1);
end

%-----------------------------------------------------------------------------------------------------------------------
% B. Selecting Vortex Lattice Location.

nvls=numvortexlatticespanwise;
nvlc=numvortexlatticechordwise;
sectionumber=nvls/wingsection;
nmax=nvlc;
mmax=nvls;
for n=1:nmax,
   for m=1:mmax+1,                                                  % y is spanwise + direction, x is chordwise - direc
                                                                       % The equation is set to use section parameter,
      section=ceil(m/sectionumber);                               % has problem at the wing tip point, so there is if cor
      if    m==mmax+1                                                 % part.
         y_TE(n,m)=TEy(section);                                   % y_LE, x_LE, z_LE is the boundary point of each
         y_LE(n,m)=LEy(section);                                   % lattice.
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         x_LE(n,m) = xatsectionend;
         xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);

         xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
        x_TE(n,m) = xatsectionend;
      else
         y_TE(n,m) = (m-1-(section-1)*sectionumber) * (TEy(section+1)-TEy(section))  / sectionumber + TEy(section);
         y_LE(n,m) = (m-1-(section-1)*sectionumber) * (LEy(section+1)-LEy(section))  / sectionumber + LEy(section);
         xatsectionstart = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_LE(n,m) = (xatsectionend-xatsectionstart) / (LEy(section+1)-LEy(section)) * (y_LE(n,m)-LEy(section)) + xats
         xatsectionstart = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-n)/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_TE(n,m) = (xatsectionend-xatsectionstart) / (TEy(section+1)-TEy(section)) * (y_TE(n,m)-TEy(section)) + xats
      end
      % y_5 and x_5 are between intermediateLE and intermediateTE, and y_6 and x_6 are between intermediateLE and inte
      %                                                                                  nmax-(n-1)
      % y_5 = y_1 (y positions are not affected by the n), x_5 = ------------(x_1-x_4) + x_4
      %                                                                    nmax
      % [y_5 x_5] ... [y_6 x_6]
      % [y_8 x_8] ... [y_7 x_7]
      % The points between the edge:
      % y_6-y_5    y-y_5                  x_6-x_5
      %-------- = --------- => x = (y-y_5)---------- + x_5 : This is a general equation.
      % x_6-x_5    x-x_5                  y_6-y_5
      midchordpoint(n,m) = 0.25*(x_LE(n,m)-x_TE(n,m))+x_TE(n,m);
      yvlm(n,m) = y_LE(n,m);                                                  % vlm is the vortex starting point of
      xvlm(n,m) = x_TE(n,m)+0.75*(x_LE(n,m)-x_TE(n,m));
      zvlm(n,m) = 0;
   end
end

%-----------------------------------------------------------------------------------------------------------------------
% C. Finding The Strenght of Circulation.

k=1;
for n=1:nvlc,
```

```
for n=1:nvlc,
   for m=1:nvls,
      xcontrolpoint(n,m)=0.5*(midchordpoint(n,m+1)+midchordpoint(n,m));
      ycontrolpoint(n,m)=0.5*(yvlm(n,m+1)+yvlm(n,m));
      zcontrolpoint(n,m)=0;
   end
end

xcp=xcontrolpoint;
ycp=ycontrolpoint;
zcp=zcontrolpoint;

% C.1 Calculating the each vortex circulation.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise if there is only one layer.
      for m=1:nvls,     % nvlc is chordwise.
         for n=1:nvlc,  % If there is only one layer, maximum n is one and sum of down-wash (eq. 7.45) is summation of
            % same n: it means only m need to be changed.
            % V_AB(m,n) is the down-wash portion at the mth control point by the nth vortex lattice.
            % The sigularity should be considered.

            % C.1.a) It is starboard part.

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

            cpxmx1n(mcp,m)=dbxA;        % This part is for the varification with the text book example at p.273.
            cpxmx2n(mcp,m)=dbxB;
            cpymy1n(mcp,m)=dbyA;
            cpymy2n(mcp,m)=dbyB;
            cpzmz1n(mcp,m)=dbzA;
            cpzmz2n(mcp,m)=dbzB;
            x2nmx1n(mcp,m)=dbABx;
            y2nmy1n(mcp,m)=dbABy;
            z2nmz1n(mcp,m)=dbABz;
            cpm1n(mcp,m)=distancebetweencpandA;
            cpm2n(mcp,m)=distancebetweencpandB;
            cpvectorAB(mcp,m)=vcABcp;

                % This velocity is only z-direction.
            V_AB_wing(n,m,therow,mcp) = 1/(4*pi) * -( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                   ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                   ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );
            V_AB_wing_previous=V_AB_wing(n,m,therow,mcp);

            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing(n,m,therow,mcp) = 1/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
```

```
            V_Ainf_wing(n,m,therow,mcp) = 1/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
            V_Binf_wing(n,m,therow,mcp) = - 1/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB );

            % C.1.b) It is port part.

            yvlm_port(n,m)=-yvlm(n,m);
            yvlm_port(n,m+1)=-yvlm(n,m+1);

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm_port(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm_port(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm_port(n,m+1)-yvlm_port(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

                % This velocity is only z-direction.
            V_AB_wing_port(n,m,therow,mcp) = 1/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );

            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing_port(n,m,therow,mcp) = 1/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA ).
            V_Binf_wing_port(n,m,therow,mcp) = - 1/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB
        end
      end
   end
end

% w(n,m,therow,mcp) is down-wash at the mcp-th control point on therow-th row in chordwise
% by the mth vortex lattice to spanwise on nth row in chordwise.
% m is spanwise and n is chordwise.
w_wing=V_AB_wing+V_Ainf_wing+V_Binf_wing;       % It includes 1/(4 pi b)
w_wing_starboard=w_wing;
w_wing_port=V_AB_wing_port+V_Ainf_wing_port+V_Binf_wing_port;
w_wing=w_wing_starboard+w_wing_port;

% C.1.c) Merging the downwash matrix at each control point to one matrix.
% The downwash matrix is not down-wash, but the component of each circulation (p.273).
% Each control point has m by n matrix.

nthcontrolpoint=0;      % It initialize a control point number.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise.
      nthcontrolpoint=nthcontrolpoint+1;    % It is a row and means each point.
      nthvortexlattice=0;                       % It initialize the vortex lattice number.
      for n=1:nvlc,     % nvlc is chordwise.
         for m=1:nvls,  % The vortex arrangement will be the order of row.
            nthvortexlattice=nthvortexlattice+1;        % It is a coulumn and means each circulation.
            w_wingforcalculation(nthcontrolpoint,nthvortexlattice)=w_wing(n,m,therow,mcp);
            % In this martix, the row indicates each control point, and the column means each vortex lattice.
         end
```

```
        end
      end
   end
end

angle=angle*degreetorad;
Circulation=inv(w_wingforcalculation)*-V_inf*angle*ones(nvls*nvlc,1);
% Circulation(x) is vortex strenght of mth vortex lattice on nth layer. For example, if there is six vortex lattices
% on each four layer, that x equals seven shows the strenght of the first vortex lattice on the second layer.
Circulationfromcalculation=Circulation;
% This will be used in inducedragformation.m file.

%-----------------------------------------------
% C.1.c) Relocating the circulation strenght.

% Although the vortex position and control point is used n by m matrix, the circulation strength n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=Circulation;
Circulation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      Circulation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end

%-------------------------------------------------
% It verify the result of subroutine.
localspan=span/sectionnumber;                                          % The span is different in each sectio
Lift=0;
Lift=0;
for n=1:nmax,
   for m=1:mmax,
      k=ceil(m/sectionnumber);
      Lift=Lift+rho*V_inf*Circulation(n,m)*localspan(k);               % It is total lift of a wing. The local span
   end                                                                          % does not divided by two since the
end                                                                                % strenght is symmetric.
Lift;
C_L=Lift/(0.5*rho*V_inf^2*S);
% down-wash at point s = sum of 1/(4pi) A(n,s)xCirculation(n)
% 1/(4pi)A(n,s)=w_wing(s,n)
% s=mcp(mcp th control point) n=m (n th vortex lattice)
C_LTheory=1*pi*angle;
Lift=C_LTheory*(0.5*rho*V_inf^2*S);
downwashCirculation=w_wingforcalculation*Circulationfromcalculation;
% downwashCirculation(x) is down-wash velocity of mth control point on nth layer. For example, if there is
% six vortex lattices
% on each four layer, that x equals seven shows the velocity of the first control point on the second layer.
% It is only down-wash at the control points. Assumption is down-wash on the vortex panel is the same at control point.
% (eq. 7.7)
% Although the vortex position and control point is used n by m matrix, the velocity magnitude n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=downwashCirculation;
downwashCirculation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      downwashCirculation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end
```

```
end

%-------------------------------------------------
% C.2 Considering the effect from other airplanes.

sizedC=size(downwashCirculation);
sizeid=size(inducedownwash);
% The downwashCirculation is from root to tip in the both side, but
% inducedownwash is from port tip to starport tip. For this reason, the rearranging the inducedownwash matrix
% is excuted.
for n=1:sizedC(1,1),            % n is the mth control panel in span wise.
   for m=1:sizedC(1,2),     % m is the mth control panel in chord wise.
      upwash_starport(n,m)=inducedownwash(n,sizeid(1,2)/2+m);       % This process is due to the arrangement of contro
      upwash_port(n,m)=inducedownwash(n,sizeid(1,2)/2-(m-1));           % points.
   end
end
UpwashinStarport=upwash_starport;
UpwashPort=upwash_port;

D_v=0;
sizedC=size(downwashCirculation);
for n=1:sizedC(1,1),
   for m=1:sizedC(1,2),
      k=ceil(m/sectionumber);
      D_v=D_v-rho*(downwashCirculation(n,m)+upwash_starport(n,m))*Circulation(n,m)*localspan(k)/2 ...
          -rho*(downwashCirculation(n,m)+upwash_port(n,m))*Circulation(n,m)*localspan(k)/2;
   end                          % The wind is not symmetric, so the local spans are divided by two.
end
% (eq. 7.14)
C_Di=D_v/(0.5*rho*V_inf^2*S);
C_DiTheory=C_LTheory^2/(pi*AR*0.99);
e=C_L^2/(pi*AR)/C_Di;
```

A.1.f File Name: trailingcp.m

```
% This program find the coordination of control points, its wing portion and wing boundary of an airplane.
% Revised November 21, 2005 - January 27, 2006 - February 9 - March 11 - 16
% - 20 - 21 - 22 - 23 - 24 - 28 - 31 - April 15, 2006 (minor touch)
% Most of part is from the findcirculation.m file.
% x, z, tipLE and tipTE show only the cooridnaton of a starboard, but y has port as well.

% note:

function [x,y,z,rootLE,rootTE,tipLE,tipTE, ...
      rightendy,rightendx,rightendz,leftendy,leftendx,leftendz,leadingedgex, ...
      leadingedgey,trailingedgex,trailingedgey] = ...
   trailingcp(wingsection,span,Lambda,taperatio,AR,distanceinx,distanceiny,distanceinz,nvls,nvlc)

%--------------------------------
% A.

degreetorad=pi/180;
Lambda=Lambda*degreetorad;

% AR = (SUM of b)^2/S => S = SUM of 2( (b_n/2(b_n/2tan(Lambda_n)+taperatio_n x c_r_n) -
% (b_n/2)^2tan(Lambda_n)/2 -
% b_n/2(b_n/2tan(Lambda_n)+(taper ratio_n-1)c_r_n)/2 )
% c_r_n = c_r_0 x taper ratio_0 x taper ratio_1 x ~ taper ratio_n
% c_r_0 = SUM of taper ratio_x (-2b_n/2taperatio_n + 2b_n/2(taper ratio_n-1)/2) = SUM of 2(b_n/2 b_n/2tan(Lambda_n)
% - (b_n/2)^2tan(Lambda_n)/2 - b_n/2 b_n/2tan(Lambda_n)/2) - S
% If there is only one section (n=0.)
%             2(b/2 b/2tan(Lambda) - (b/2)^2tan(Lambda)/2 - (b/2)^2tan(Lambda)/2) -S            b^2
% c_r = ------------------------------------------------------------------------------- <= S = -------
%                      -2b/2taperatio + 2b/2(taper ratio-1)/2                                    AR
ss=size(span);                                  % This is the wing section number.
ss=ss(1,2);
SUMofb=span*ones(ss,1);

% Possible maximum c_r_0 = 2b/AR <- S = b/2 x c_r_0 in a delta wing
% Possible maximum c_r_0 = 2b/AR <- S = b/2 x c_r_0 in a delta wing
% It finds the appropriate root chord length.
maxc_r_0=2*SUMofb/AR;
for c_r_0=0:maxc_r_0/1000:maxc_r_0,
   for n=1:ss,
      S=0;                                      % If the surface area is not initialized, it will keep increasing
      if n==1                                   % root chord length trial.
         c(n)=c_r_0*taperatio(n);
         S=S+2*(span(n)/2*(c_r_0+c(n)));
      else
         c(n)=c(n-1)*taperatio(n);
         S=S+2*(span(n)/2*(c(n-1)+c(n)));
      end
   end

   if abs( (S-SUMofb^2/AR)/(SUMofb^2/AR) )<=0.01        % There will be limit to be found by trial and error.
      answerc_r=c_r_0;
      answerS=S;
      c_r_0=SUMofb/AR;
   else
      c_r_0=c_r_0;
   end
end

% Dividing the wing sectons.

S=answerS;
c_r_0=answerc_r;
rootLE=[0 c_r_0];
rootLE=rootLE+[distanceiny distanceinx];    % [y_1 x_1] format
rootTE=[rootLE(1)+0 rootLE(2)-c_r_0];       % [y_4 x_4]
intermediateLE=rootLE;
intermediateTE=rootTE;
for n=1:ss,
```

```
for n=1:ss,
   if n==1
      c(n)=c_r_0*taperatio(n);
   else
      c(n)=c(n-1)*taperatio(n);
   end
end
for n=1:ss,                                          % It finds the leading edge and trailing edge position.
   nexty_n=intermediateLE(2*n-1)+span(n)/2;
   nextx_n=0.5*abs(intermediateLE(2*n)-intermediateTE(2*n))+intermediateTE(2*n)-span(n)/2*tan(Lambda(n));
   % The sweep angle is at the mid-chord line.
   intermediateLE=[intermediateLE nexty_n nextx_n+0.5*c(n)];
   intermediateTE=[intermediateTE nexty_n nextx_n-0.5*c(n)];
end
tipLE=[intermediateLE(2*ss+1) intermediateLE(2*(ss+1))];                    % [y_2 x_2]
tipTE=[intermediateTE(2*ss+1) intermediateTE(2*(ss+1))];                % [y_3 x_3]
for n=1:ss+1,
   LEx(n)=intermediateLE(2*n);
   LEy(n)=intermediateLE(2*n-1);
   TEx(n)=intermediateTE(2*n);
   TEy(n)=intermediateTE(2*n-1);
end

% This loop build the denser leading and trailing edge coordinations.
% When n is one, the position is not the centerline.

intervalgap=10*nvls;
% The intervalgap should be more than the sectionnumber(nvls) in each section.

for n=1:intervalgap*ss,
   section=ceil(n/intervalgap);
   leadingedgex(n)=(LEx(section+1)-LEx(section)) * (n-(section-1)*intervalgap)/intervalgap + LEx(section);
   leadingedgey(n)=(LEy(section+1)-LEy(section)) * (n-(section-1)*intervalgap)/intervalgap + LEy(section);
   trailingedgex(n)=(TEx(section+1)-TEx(section)) * (n-(section-1)*intervalgap)/intervalgap + TEx(section);
   trailingedgex(n)=(TEx(section+1)-TEx(section)) * (n-(section-1)*intervalgap)/intervalgap + TEx(section);
   trailingedgey(n)=(TEy(section+1)-TEy(section)) * (n-(section-1)*intervalgap)/intervalgap + TEy(section);
end

% This plot is for checking.
% figure(1), title('Wing geometry','Fonts',15);
% for n=1:ss,
%    line([intermediateLE(2*n-1) intermediateLE(2*n+1)],[intermediateLE(2*n) intermediateLE(2*(n+1))]);
%    line([intermediateTE(2*n-1) intermediateTE(2*n+1)],[intermediateTE(2*n) intermediateTE(2*(n+1))]);
%    % Starboard
%    line([2*LEy(1)-intermediateLE(2*n-1) 2*LEy(1)-intermediateLE(2*n+1)],[intermediateLE(2*n) intermediateLE(2*(n+1))]
%    line([2*TEy(1)-intermediateTE(2*n-1) 2*TEy(1)-intermediateTE(2*n+1)],[intermediateTE(2*n) intermediateTE(2*(n+1))]
%    % Port
% end
% line([tipLE(1) tipTE(1)],[tipLE(2) tipTE(2)]);
% line([2*LEy(1)-tipLE(1) 2*TEy(1)-tipTE(1)],[tipLE(2) tipTE(2)]);
% axis equal,grid on;
%figure(2), title('Leading and Trailing Edge Points','Fonts',15);
%plot(LEy,LEx,'cs',TEy,TEx,'cd',leadingedgey,leadingedgex,'m+',trailingedgey,trailingedgex,'m<');
%axis equal,grid on;
%-------------------------------------------------------------------------------------------------------------------
% B.

sectionnumber=nvls/wingsection;
nmax=nvlc;
mmax=nvls;
for n=1:nmax,
   for m=1:mmax+1,                                                  % y is spanwise + direction, x is chordwise - direc
                                                                       % The equation is set to use section parameter,
      section=ceil(m/sectionnumber);                             % has problem at the wing tip point, so there is if cor
      if    m==mmax+1                                                 % part.
         y_TE(n,m)=TEy(section);                                    % y_LE, x_LE, z_LE is the boundary point of each
         y_LE(n,m)=LEy(section);                                    % lattice.
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
```

```
            xatsectionend = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
            x_LE(n,m) = xatsectionend;
            xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
          x_TE(n,m) = xatsectionend;
        else
            y_TE(n,m) = (m-1-(section-1)*sectionnumber) * (TEy(section+1)-TEy(section))  / sectionnumber + TEy(section);
            y_LE(n,m) = (m-1-(section-1)*sectionnumber) * (LEy(section+1)-LEy(section))  / sectionnumber + LEy(section);
            xatsectionstart = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
            xatsectionend = (nmax-(n-1))/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
            x_LE(n,m) = (xatsectionend-xatsectionstart) / (LEy(section+1)-LEy(section)) * (y_LE(n,m)-LEy(section)) + xatse
            xatsectionstart = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
            xatsectionend = (nmax-n)/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
            x_TE(n,m) = (xatsectionend-xatsectionstart) / (TEy(section+1)-TEy(section)) * (y_TE(n,m)-TEy(section)) + xatse
        end
        % y_5 and x_5 are between intermediateLE and intermediateTE, and y_6 and x_6 are between intermediateLE and inter
        %                                                                                  nmax-(n-1)
        % y_5 = y_1 (y positions are not affected by the n), x_5 = ------------(x_1-x_4) + x_4
        %                                                                    nmax
        % [y_5 x_5] ... [y_6 x_6]
        % [y_8 x_8] ... [y_7 x_7]
        % The points between the edge:
        % y_6-y_5    y-y_5                  x_6-x_5
        %-------- = --------- => x = (y-y_5)---------- + x_5 : This is a general equation.
        % x_6-x_5    x-x_5                  y_6-y_5
        midchordpoint(n,m) = 0.25*(x_LE(n,m)-x_TE(n,m))+x_TE(n,m);
        yvlm(n,m) = y_LE(n,m);                                                  % vlm is the vortex starting point of e
        xvlm(n,m) = x_TE(n,m)+0.75*(x_LE(n,m)-x_TE(n,m));
        zvlm(n,m) = distanceinz;                                                % It is assumed the no dihedral angle o
    end
end

%-------------------------------------------------------------------------------------------------------------------------
% C.


k=1;
for n=1:nvlc,
    for m=1:nvls,
        xcontrolpoint(n,m)=0.5*(midchordpoint(n,m+1)+midchordpoint(n,m));
        ycontrolpoint(n,m)=0.5*(yvlm(n,m+1)+yvlm(n,m));
        ycontrolpoint_port(n,m)=2*rootLE(1)-ycontrolpoint(n,m);   % It is the port part of the wing.
        ycontrolpoint_port_range(n,nvls+1-m)=ycontrolpoint_port(n,m);
        zcontrolpoint(n,m)=distanceinz;                           % It is assumed the no dihedral angle of the wing.
    end
end

x=xcontrolpoint;
z=zcontrolpoint;
y=[ycontrolpoint_port_range ycontrolpoint];

% This routine finds the wing range at the specific point. leftend and rightend are three colum matrix
% and each row is related
% to the each control point. The leftend matrix is the cooridnation of vortex lattice left end where the line meet
% the lattice left side. The line should have the control point of the lattice.

maxn=intervalgap*ss;

%------------------------
% D. 1 Starboard portion
% This has x,y and z coordinations.
for n=1:nvlc,
    for m=1:nvls,
        rightendy(n,m)=yvlm(n,m+1);
        rightendx(n,m)=midchordpoint(n,m+1);
        rightendz(n,m)=zvlm(n,m+1);                               % y x z format
        leftendy(n,m)=yvlm(n,m);
        leftendx(n,m)=midchordpoint(n,m);
        leftendz(n,m)=zvlm(n,m);
```

A.1.g File Name: vortexlatticelocation.m

```
% Revised February 9, 2006
% Most of part is from the findcirculation.m file.

function [xvlm,yvlm,zvlm,rootLE,Circulation]=vortexlatticelocation...
   (wingsection,span,Lambda,taperatio,AR,distanceinx,distanceiny,distanceinz,nvls,nvlc,rho,V_inf,angle);

%------------------------------
% A. Drawing The Wing Geometry.

degreetorad=pi/180;
Lambda=Lambda*degreetorad;

% AR = (SUM of b)^2/S => S = SUM of 2( (b_n/2(b_n/2tan(Lambda_n)+taperatio_n x c_r_n) -
% (b_n/2)^2tan(Lambda_n)/2 -
% b_n/2(b_n/2tan(Lambda_n)+(taper ratio_n-1)c_r_n)/2 )
% c_r_n = c_r_0 x taper ratio_0 x taper ratio_1 x ~ taper ratio_n
% c_r_0 = SUM of taper ratio_x (-2b_n/2taperatio_n + 2b_n/2(taper ratio_n-1)/2) = SUM of 2(b_n/2 b_n/2tan(Lambda_n)
% - (b_n/2)^2tan(Lambda_n)/2 - b_n/2 b_n/2tan(Lambda_n)/2) - S
% If there is only one section (n=0.)
%             2(b/2 b/2tan(Lambda) - (b/2)^2tan(Lambda)/2 - (b/2)^2tan(Lambda)/2) -S            b^2
% c_r = ------------------------------------------------------------------------------- <= S = -------
%                      -2b/2taperatio + 2b/2(taper ratio-1)/2                                    AR
ss=size(span);                                  % This is the wing section number.
ss=ss(1,2);
SUMofb=span*ones(ss,1);

% Possible maximum c_r_0 = 2b/AR <- S = b/2 x c_r_0 in a delta wing
% It finds the appropriate root chord length.
maxc_r_0=2*SUMofb/AR;
for c_r_0=0:maxc_r_0/1000:maxc_r_0,
   for n=1:ss,
      S=0;                                      % If the surface area is not initialized, it will keep increasing :
      if n==1                                   % root chord length trial.
      if n==1                                   % root chord length trial.
         c(n)=c_r_0*taperatio(n);
         S=S+2*(span(n)/2*(c_r_0+c(n)));
      else
         c(n)=c(n-1)*taperatio(n);
         S=S+2*(span(n)/2*(c(n-1)+c(n)));
      end
   end

   if abs( (S-SUMofb^2/AR)/(SUMofb^2/AR) )<=0.01        % There will be limit to be found by trial and error.
      answerc_r=c_r_0;
      answerS=S;
      c_r_0=SUMofb/AR;
   else
      c_r_0=c_r_0;
   end
end

% Dividing the wing sectons.

S=answerS;
c_r_0=answerc_r;
rootLE=[0 c_r_0];
rootLE=rootLE+[distanceiny distanceinx];    % [y_1 x_1] format
rootTE=[rootLE(1)+0 rootLE(2)-c_r_0];       % [y_4 x_4]
intermediateLE=rootLE;
intermediateTE=rootTE;
for n=1:ss,
   if n==1
      c(n)=c_r_0*taperatio(n);
   else
      c(n)=c(n-1)*taperatio(n);
   end
end
```

```
end
for n=1:ss,                                          % It finds the leading edge and trailing edge position.
   nexty_n=intermediateLE(2*n-1)+span(n)/2;
   nextx_n=0.5*abs(intermediateLE(2*n)-intermediateTE(2*n))+intermediateTE(2*n)-span(n)/2*tan(Lambda(n));
   % The sweep angle is at the mid-chord line.
   intermediateLE=[intermediateLE nexty_n nextx_n+0.5*c(n)];
   intermediateTE=[intermediateTE nexty_n nextx_n-0.5*c(n)];
end
tipLE=[intermediateLE(2*ss+1) intermediateLE(2*(ss+1))];                    % [y_2 x_2]
tipTE=[intermediateTE(2*ss+1) intermediateTE(2*(ss+1))];                 % [y_3 x_3]
for n=1:ss+1,
   LEx(n)=intermediateLE(2*n);
   LEy(n)=intermediateLE(2*n-1);
   TEx(n)=intermediateTE(2*n);
   TEy(n)=intermediateTE(2*n-1);
end

%-------------------------------------------------------------------------------------------------------------------------------
% B. Selecting Vortex Lattice Location.

sectionnumber=nvls/wingsection;
nmax=nvlc;
mmax=nvls;
for n=1:nmax,
   for m=1:mmax+1,                                                  % y is spanwise + direction, x is chordwise - direc
                                                                       % The equation is set to use section parameter,
      section=ceil(m/sectionnumber);                              % has problem at the wing tip point, so there is if cor
      if    m==mmax+1                                                 % part.
         y_TE(n,m)=TEy(section);                                   % y_LE, x_LE, z_LE is the boundary point of each
         y_LE(n,m)=LEy(section);                                   % lattice.
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         x_LE(n,m) = xatsectionend;
         xatsectionend = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
        x_TE(n,m) = xatsectionend;
         x_TE(n,m) = xatsectionend;
      else
         y_TE(n,m) = (m-1-(section-1)*sectionnumber) * (TEy(section+1)-TEy(section))  / sectionnumber + TEy(section);
         y_LE(n,m) = (m-1-(section-1)*sectionnumber) * (LEy(section+1)-LEy(section))  / sectionnumber + LEy(section);
         xatsectionstart = (nmax-(n-1))/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-(n-1))/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_LE(n,m) = (xatsectionend-xatsectionstart) / (LEy(section+1)-LEy(section)) * (y_LE(n,m)-LEy(section)) + xatse
         xatsectionstart = (nmax-n)/nmax*(LEx(section)-TEx(section)) + TEx(section);
         xatsectionend = (nmax-n)/nmax*(LEx(section+1)-TEx(section+1)) + TEx(section+1);
         x_TE(n,m) = (xatsectionend-xatsectionstart) / (TEy(section+1)-TEy(section)) * (y_TE(n,m)-TEy(section)) + xatse
      end
      % y_5 and x_5 are between intermediateLE and intermediateTE, and y_6 and x_6 are between intermediateLE and inter
      %                                                                                 nmax-(n-1)
      % y_5 = y_1 (y positions are not affected by the n), x_5 = ------------(x_1-x_4) + x_4
      %                                                                    nmax
      % [y_5 x_5] ... [y_6 x_6]
      % [y_8 x_8] ... [y_7 x_7]
      % The points between the edge:
      % y_6-y_5    y-y_5                  x_6-x_5
      %-------- = --------- => x = (y-y_5)---------- + x_5 : This is a general equation.
      % x_6-x_5    x-x_5                  y_6-y_5
      midchordpoint(n,m) = 0.25*(x_LE(n,m)-x_TE(n,m))+x_TE(n,m);
      yvlm(n,m) = y_LE(n,m);                                                    % vlm is the vortex starting point of e
      xvlm(n,m) = x_TE(n,m)+0.75*(x_LE(n,m)-x_TE(n,m));
      zvlm(n,m) = 0;
   end
end

%-------------------------------------------------------------------------------------------------------------------------------
% C. Finding The Strenght of Circulation.

k=1;
for n=1:nvlc,
   for m=1:nvls,
```

```
   for m=1:nvls,
      xcontrolpoint(n,m)=0.5*(midchordpoint(n,m+1)+midchordpoint(n,m));
      ycontrolpoint(n,m)=0.5*(yvlm(n,m+1)+yvlm(n,m));
      ycontrolpoint_port(n,m)=2*rootLE(1)-ycontrolpoint(n,m);   % It is the port part of the wing.
      ycontrolpoint_port_range(n,nvls+1-m)=ycontrolpoint_port(n,m);
        zcontrolpoint(n,m)=0;
   end
end

xcp=xcontrolpoint;
ycp=ycontrolpoint;
zcp=zcontrolpoint;

% C.1 Calculating the each vortex circulation.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise if there is only one layer.
      for m=1:nvls,     % nvlc is chordwise.
         for n=1:nvlc,  % If there is only one layer, maximum n is one and sum of down-wash (eq. 7.45) is summation of
            % same n: it means only m need to be changed.
            % V_AB(m,n) is the down-wash portion at the mth control point by the nth vortex lattice.
            % The sigularity should be considered.

            % C.1.a) It is starboard part.

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm(n,m+1)-yvlm(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

            cpxmx1n(mcp,m)=dbxA;        % This part is for the varification with the text book example at p.273.
            cpxmx2n(mcp,m)=dbxB;
            cpymy1n(mcp,m)=dbyA;
            cpymy2n(mcp,m)=dbyB;
            cpzmz1n(mcp,m)=dbzA;
            cpzmz2n(mcp,m)=dbzB;
            x2nmx1n(mcp,m)=dbABx;
            y2nmy1n(mcp,m)=dbABy;
            z2nmz1n(mcp,m)=dbABz;
            cpm1n(mcp,m)=distancebetweencpandA;
            cpm2n(mcp,m)=distancebetweencpandB;
            cpvectorAB(mcp,m)=vcABcp;

                % This velocity is only z-direction.
            V_AB_wing(n,m,therow,mcp) = 1/(4*pi) * -( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );
            V_AB_wing_previous=V_AB_wing(n,m,therow,mcp);

            % (yvlm(n,m)-y) can be zero.
```

```
            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing(n,m,therow,mcp) = 1/(4*pi) * dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
            V_Binf_wing(n,m,therow,mcp) = - 1/(4*pi) * dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB );

            % C.1.b) It is port part.

            yvlm_port(n,m)=-yvlm(n,m);
            yvlm_port(n,m+1)=-yvlm(n,m+1);

            distancebetweenxandA=xcp(therow,mcp)-xvlm(n,m);
            distancebetweenxandB=xcp(therow,mcp)-xvlm(n,m+1);
            distancebetweenyandA=ycp(therow,mcp)-yvlm_port(n,m);
            distancebetweenyandB=ycp(therow,mcp)-yvlm_port(n,m+1);
            distancebetweenzandA=zcp(therow,mcp)-zvlm(n,m);
            distancebetweenzandB=zcp(therow,mcp)-zvlm(n,m+1);
            distancebetweenAandBx=xvlm(n,m+1)-xvlm(n,m);
            diatancebetweenAandBy=yvlm_port(n,m+1)-yvlm_port(n,m);
            diatancebetweenAandBz=zvlm(n,m+1)-zvlm(n,m);
            dbxA=distancebetweenxandA;
            dbxB=distancebetweenxandB;
            dbyA=distancebetweenyandA;
            dbyB=distancebetweenyandB;
            dbzA=distancebetweenzandA;
            dbzB=distancebetweenzandB;
            dbABx=distancebetweenAandBx;
            dbABy=diatancebetweenAandBy;
            dbABz=diatancebetweenAandBz;
            distancebetweencpandA=sqrt( dbxA^2+dbyA^2+dbzA^2 );
            distancebetweencpandB=sqrt( dbxB^2+dbyB^2+dbzB^2 );
            vectorcrossABandcp=(dbyA*dbzB-dbyB*dbzA)^2+(dbxA*dbzB-dbxB*dbzA)^2+(dbxA*dbyB-dbxB*dbyA)^2;
            vcABcp=vectorcrossABandcp;

                % This velocity is only z-direction.
            V_AB_wing_port(n,m,therow,mcp) = 1/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
            V_AB_wing_port(n,m,therow,mcp) = 1/(4*pi) * ( dbxA*dbyB-dbxB*dbyA )/vcABcp * ...
                  ( ( dbABx*dbxA+dbABy*dbyA+dbABz*dbzA )/distancebetweencpandA - ...
                  ( dbABx*dbxB+dbABy*dbyB+dbABz*dbzB )/distancebetweencpandB );

            % (yvlm(n,m)-y) can be zero.
            V_Ainf_wing_port(n,m,therow,mcp) = 1/(4*pi) * -dbyA/(dbzA^2+(-dbyA)^2) * ( -1+dbxA/distancebetweencpandA );
            V_Binf_wing_port(n,m,therow,mcp) = - 1/(4*pi) * -dbyB/(dbzB^2+(-dbyB)^2) * ( -1+dbxB/distancebetweencpandB
         end
      end
   end
end

% w(n,m,therow,mcp) is down-wash at the mcp-th control point on therow-th row in chordwise
% by the mth vortex lattice to spanwise on nth row in chordwise.
% m is spanwise and n is chordwise.
w_wing=V_AB_wing+V_Ainf_wing+V_Binf_wing;       % It includes 1/(4 pi b)
w_wing_starboard=w_wing;
w_wing_port=V_AB_wing_port+V_Ainf_wing_port+V_Binf_wing_port;
w_wing=w_wing_starboard+w_wing_port;

% C.1.c) Merging the downwash matrix at each control point to one matrix.
% The downwash matrix is not down-wash, but the component of each circulation (p.273).
% Each control point has m by n matrix.

nthcontrolpoint=0;      % It initialize a control point number.
for therow=1:nvlc,
   for mcp=1:nvls,      % mcp is spanwise.
      nthcontrolpoint=nthcontrolpoint+1;    % It is a row and means each point.
      nthvortexlattice=0;                   % It initialize the vortex lattice number.
      for n=1:nvlc,     % nvlc is chordwise.
         for m=1:nvls,  % The vortex arrangement will be the order of row.
            nthvortexlattice=nthvortexlattice+1;        % It is a coulumn and means each circulation.
            w_wingforcalculation(nthcontrolpoint,nthvortexlattice)=w_wing(n,m,therow,mcp);
            % In this martix, the row indicates each control point, and the column means each vortex lattice.
```

```
            % In this martix, the row indicates each control point, and the column means each vortex lattice.
         end
      end
   end
end

angle=angle*degreetorad;
Circulation=inv(w_wingforcalculation)*-V_inf*angle*ones(nvls*nvlc,1);
% Circulation(x) is vortex strenght of mth vortex lattice on nth layer. For example, if there is six vortex lattices
% on each four layer, that x equals seven shows the strenght of the first vortex lattice on the second layer.
Circulationfromcalculation=Circulation;
% This will be used in inducedragformation.m file.

%---------------------------------------------
% C.1.c) Relocating the circulation strenght.

% Although the vortex position and control point is used n by m matrix, the circulation strength n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=Circulation;
Circulation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      Circulation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end

%-------------------------------------------------
% It verify the result of subroutine.
localspan=span/sectionnumber;                                          % The span is different in each sectio
localspan=span/sectionnumber;                                          % The span is different in each sectio
Lift=0;
for n=1:nmax,
   for m=1:mmax,
      k=ceil(m/sectionnumber);
      Lift=Lift+rho*V_inf*Circulation(n,m)*localspan(k);    % It is total lift of a wing.
   end
end
Lift;
C_L=Lift/(0.5*rho*V_inf^2*S);
% down-wash at point s = sum of 1/(4pi) A(n,s)xCirculation(n)
% 1/(4pi)A(n,s)=w_wing(s,n)
% s=mcp(mcp th control point) n=m (n th vortex lattice)
C_LTheory=1*pi*angle;
Lift=C_LTheory*(0.5*rho*V_inf^2*S);
downwashCirculation=w_wingforcalculation*Circulationfromcalculation;
% downwashCirculation(x) is down-wash velocity of mth control point on nth layer. For example, if there is
% six vortex lattices
% on each four layer, that x equals seven shows the velocity of the first control point on the second layer.
% It is only down-wash at the control points. Assumption is down-wash on the vortex panel is the same at control point.
% (eq. 7.7)
% Although the vortex position and control point is used n by m matrix, the velocity magnitude n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=downwashCirculation;
downwashCirculation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      downwashCirculation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
```

```
downwashCirculation=w_wingforcalculation*Circulationfromcalculation;
% downwashCirculation(x) is down-wash velocity of mth control point on nth layer. For example, if there is
% six vortex lattices
% on each four layer, that x equals seven shows the velocity of the first control point on the second layer.
% It is only down-wash at the control points. Assumption is down-wash on the vortex panel is the same at control point
% (eq. 7.7)
% Although the vortex position and control point is used n by m matrix, the velocity magnitude n x m by one matrix
% for the convenient calculation of aerodynamic characteristics.
% For further calculation, the circulation also becomes n by m matrix.

TemporaryMatrix=downwashCirculation;
downwashCirculation=0;
nthvortexlattice=0; % It initialize the element order of circulation.
for n=1:nvlc,       % nvlc is chordwise.
   for m=1:nvls,
      nthvortexlattice=nthvortexlattice+1;
      downwashCirculation(n,m)=TemporaryMatrix(nthvortexlattice);
      % In this martix, the row indicates each layer, and the column means each vortex lattice.
   end
end
D_v=0;
sizedC=size(downwashCirculation);
for n=1:sizedC(1,1),
   for m=1:sizedC(1,2),
      k=ceil(m/sectionumber);
      D_v=D_v-rho*downwashCirculation(n,m)*Circulation(n,m)*localspan(k);
   end
end
% (eq. 7.14)
C_Di=D_v/(0.5*rho*V_inf^2*S);
C_DiTheory=C_LTheory^2/(pi*AR*0.99);
e=C_L^2/(pi*AR)/C_Di;
```