Dynamic Soaring: Assignment 8

Izza (AE18B006) &
RV Ramkumar (AE20D025)

Problem

The problem is to find the minimum wind shear β required for supporting the dynamic soaring where the wind velocity at altitude h is given by $V_w = \beta h$ and the solution to be determined using Pseudospectral approach. The following are the parameters for the 3 DOF aircraft model (considering elliptic wing) to be assumed:

Table 1: Aircraft Parameters

AR	15
C_{D_0}	0.01
ρ	1.225 kg/m^3
$C_{L_{max}}$	1.2
g	9.81 m/s^2
m	8 kg
m/S	14 kg/m^2
μ_{max}	$\pi/3$

The guess for the model is given as follows

$$x(t) = 20(\cos(2\pi t/T_f) - 1) \tag{1}$$
$$y(t) = -20\sqrt{2}\sin(2\pi t/T_f) \tag{2}$$
$$z(t) = 20(\cos(2\pi t/T_f) - 1) \tag{3}$$

and the guess for Time of flight T_f is 10 seconds and β is 0.25 second^{-1}.

Formulation of Optimization Problem

The objective is to find the minimum wind shear required such that the energy neutral dynamic soaring trajectory is obtained subject to the 3 degree of freedom point mass dynamics of the aircraft as well as constraints on the states and limits on the controls.

$$\text{Performance Index } J = \min_{C_L,\mu} \ |\beta| \tag{4}$$
$$\text{subject to } \dot{X} = f(X,U) \tag{5}$$
$$U_{min} \leq U \leq U_{max} \tag{6}$$
$$h(X,U) = 0 \tag{7}$$

where X is the state vector representing the dynamics of the 'dynamic soaring' which is given by $X = [V_A \quad \chi \quad \gamma \quad x \quad y \quad z]$ and $U = [C_L \quad \mu]$ is the control control input with control limits on the U. The details of the dynamics, state vector, control, initial and final conditions, upper and lower bounds are explained in the subsequent paragraphs.

The 3 degrees of freedom aircraft dynamics in the wind relative frame is given as follows:

$$\dot{V}_A = -\frac{D}{m} - g\sin\gamma - \cos\gamma\cos\chi\dot{W}_x^E \tag{8}$$

$$(V_A\cos\gamma)\,\dot{\chi} = \frac{1}{m}L\sin\mu + \sin\chi\dot{W}_x^E \tag{9}$$

$$(V_A)\,\dot{\gamma} = \frac{1}{m}L\cos\mu - g\cos\gamma + \sin\gamma\cos\chi\dot{W}_x^E \tag{10}$$

$$\dot{x} = V_A\cos\gamma\cos\chi + W_x^E \tag{11}$$

$$\dot{y} = V_A\cos\gamma\sin\chi \tag{12}$$

$$\dot{z} = -V_A\sin\gamma \tag{13}$$

where L,D represents the Aerodynamic Lift and Drag respectively; γ, χ, μ are respectively flight path angle, heading angle and the bank angle; W_x^E is the wind velocity in the Wind velocity in the earth fixed inertial 'x' direction which varies only in the inertial vertical 'z' direction. The aerodynamic Lift and Drag are expressed as $L = C_L\frac{1}{2}\rho V_A^2 S$ and $D = C_D\frac{1}{2}\rho V_A^2 S = (C_{D_0} + KC_L^2)\frac{1}{2}\rho V_A^2 S$.

In the above sets of equation, the first three equations represents point mass dynamics from the Newton's second law and the next three equations are kinematics of the aircraft.

The state vector and the control vector respectively are the following:

$$\boldsymbol{X} = [V_A \quad \chi \quad \gamma \quad x \quad y \quad z] \tag{14}$$

$$\boldsymbol{U} = [C_L \quad \mu] \tag{15}$$

In view of the 'O' shaped dynamic soaring trajectory (where the aircraft returns back to its original position) simulation, the initial and final conditions considered are as follows: $V_A(0) = V_A(t_f)$; $\chi(t_f) = \chi(0) + 2\pi$; $\gamma(0) = \gamma(t_f)$; $x(0) = x(t_f) = 0$; $y(0) = y(t_f) = 0$; $z(0) = z(t_f) = 0$;

The trajectory optimization problem is solved using Direct method (Discretize then Optimize). The time interval is discretized into a grid of points between $t = t_0$ and $t = t_f$. The Legendre-Gauss-Lobatto nodes (N nodes) are considered for the pseudo spectral approach as the discrete points where the values of state, control are to be evaluated. These discrete points are converted to the domain between -1 to 1 using the transformation $\left(t = \frac{t_f+t_0}{2} + \frac{t_f-t_0}{2}\tau\right)$. Further, the differentiation / derivative matrix is based on using the Lagrange interpolation polynomials using these LGL nodes.

Since the above problem is transcribed to a Non-Linear Programming problem, the 'fmincon' function of the Matlab can be used to obtain the optimal time history of the variables. This requires initial guess for the solution as well as the upper and lower bounds of the state at each of the nodal points. There are totally 8N+2 variables every N points corresponding to the state (6N), control (2N), one variable each for the total time of flight (t_f) and the wind shear (β). The limits used are as follows:

Table 2: Limits of the State Variables

Variable	Velocity	χ	γ	x	y	h
Lower Limit	14	-π	-$\pi/3$	-100	-100	0
Upper Limit	50	π	$\pi/3$	100	100	100

Table 3: Limits of the Control Variables

Variable	C_L	μ
Lower Limit $\boldsymbol{u_{min}}$	0	-$\pi/3$
Upper Limit $\boldsymbol{u_{max}}$	$C_{L_{max}}$=1.2	$\pi/3$

Table 4: Limits of TOF and β

Variable	TOF	β
Lower Limit	5	0.05
Upper Limit	30	0.6

The final Results we got are:

Table 5: Final Results

Variable	Value
Time of Flight (s)	11.5986
min β (s^{-1})	0.1567
Feasibility (end)	4.583 * e^{-13}

The corresponding plots are as follows. We have compared the Pseudospectral Results with that of RK4 method.

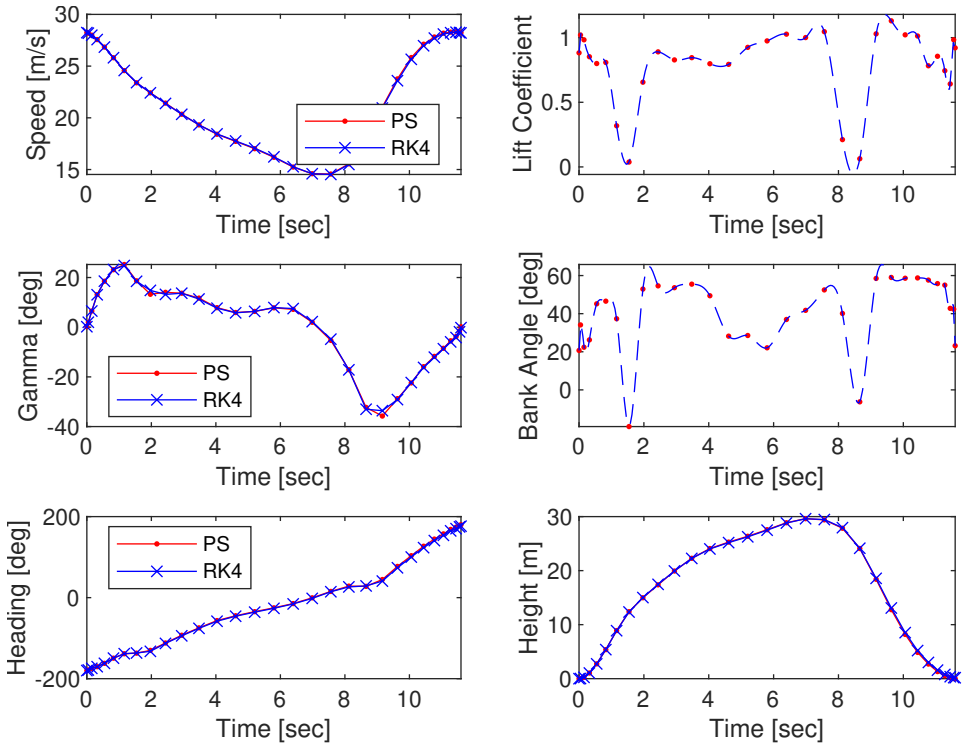

Figure 1

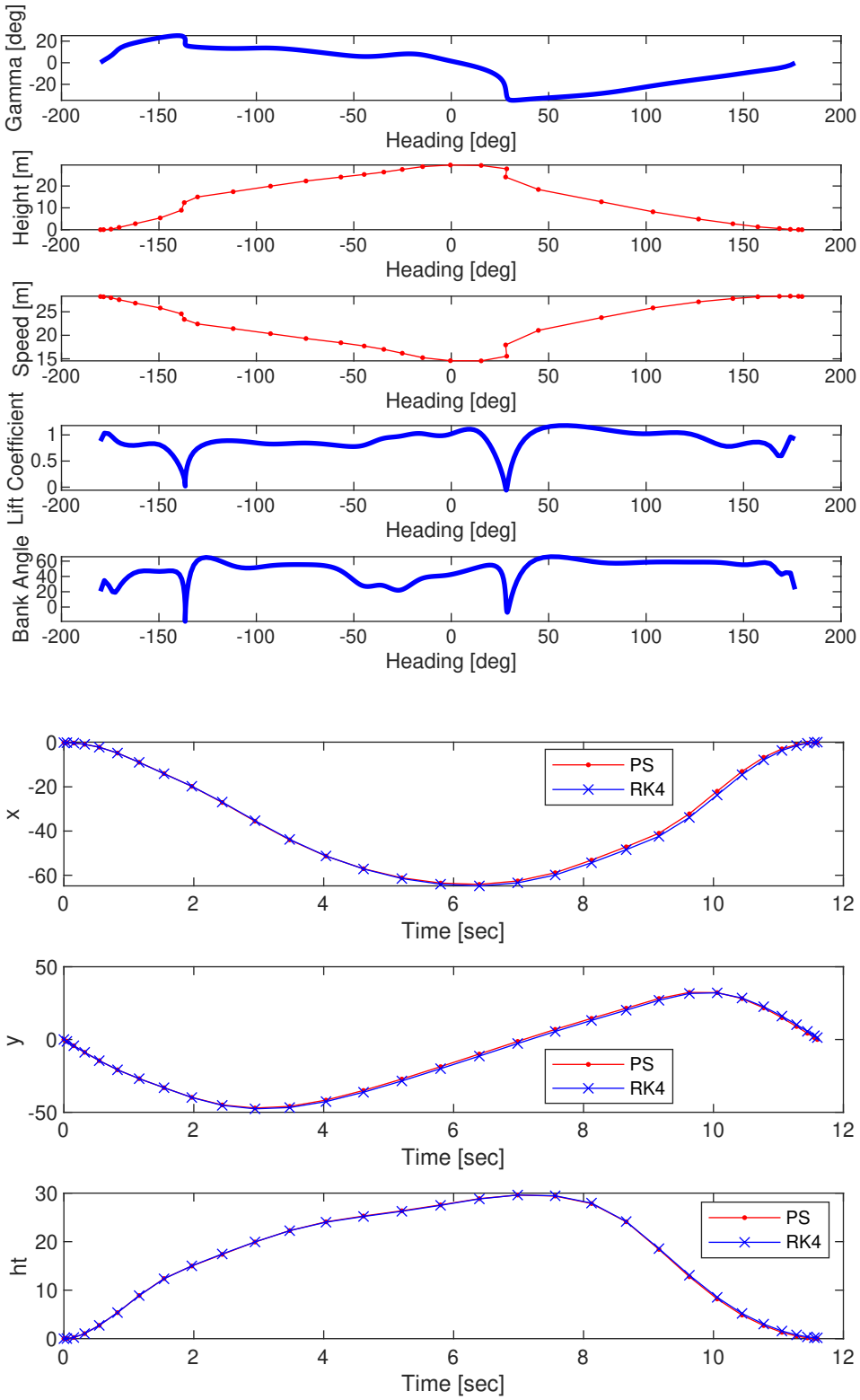

Figure 2: x, y, z vs Time

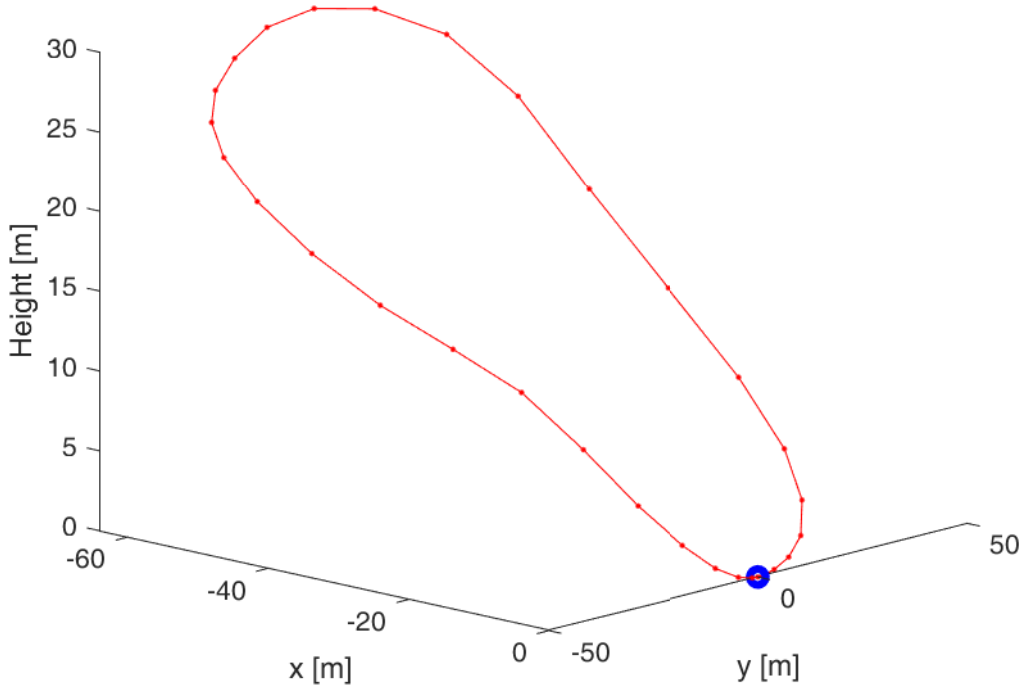

Figure 3: Trajectory

Code Listing

The codes that have been used are listed below:

main.m

```
clear all; close all; clc;
global g rho m S CD0 K Dmat tau N beta1

g = 9.81;
rho = 1.225;
m   = 8;
mbyS = 14;
AR   = 15;
S    = m/mbyS;
K    = 1/(pi*AR);
CD0  = 0.01;
CLmax = 1.2;

N = 31;

[tau, Dmat] = legDc(N-1);

%decision variable Cost = [Va (1 to N) chi(N+1 to 2N) gama(2N+1 to 3N)
%                          x1(3N+1 to 4N) y1(4N+1 to 5N) z1(5N+1 to 6N)
%                          CL (6N+1 to 7N) mu1(7N+1 to 8N) tf(8N+1) beta(8N+2]

% Upper and lower bounds for the state and control;

% states
vstall = sqrt(2*mbyS*g/(rho*CLmax));
% velmin = vstall;     velmax = 50;

velmin = 14;   velmax = 50;
chimin = -pi;       chimax = pi;
gamamin = -pi/3;    gamamax = pi/3;
xmin = -100;   xmax = 100;
ymin = -100; ymax = 100;
htmin = 0.0;        htmax = 100;

%control
clmin = 0.0; clmax = CLmax;
mumin = -60*pi/180; mumax = 60*pi/180;
% flight time;
tfmin = 5;  tfmax = 30;

% wind shear
betamin = 0.05; betamax = 0.6;
```

```
vars = 8*N+2; % variables 6*N states; 2*N control; One Time of flight;
                    % One wind shear

limits =[velmin velmax;...
chimin chimax;...
gamamin gamamax;...
xmin xmax;...
ymin ymax;...
%  zmin zmax;...
        htmin htmax;...
clmin clmax;...
mumin mumax;...
tfmin tfmax;...
betamin betamax];

% Lower bound and Upper bound for the 6N state variables and 2N control variables

for j=1:8
Lb(N*(j-1)+1:j*N) = limits(j,1)*ones(1,N);
Ub(N*(j-1)+1:j*N) = limits(j,2)*ones(1,N);
end
% Lower and Upper bound for flight time
Lb(8*N+1) = limits(9,1);
Ub(8*N+1) = limits(9,2);
% Lower and Upper bound for wind shear
Lb(8*N+2) = limits(10,1);
Ub(8*N+2) = limits(10,2);
Lb = Lb';
Ub = Ub';

% Initial guess for all the variables;

tfg = 10;
tvec = tfg/2 + tfg/2*tau;

% Initial conditions from "Optimal Patterns of Dynamic Soaring with a small
% unmanned aerial vehicle," by Liu D.N., et.al.,

Va_guess = 5*(3+cos(2*pi*tvec/tfg)) + velmin;
chi_guess =  pi*tvec/tfg;
gama_guess = pi/6*sin(2*pi*tvec/tfg);
x1_guess = 20*(-1+cos(2*pi*tvec/tfg));
y1_guess = -20*sin(2*pi*tvec/tfg);
ht_guess = 20*(1-cos(2*pi*tvec/tfg));

CL_guess = 0.75*ones(N,1);
mu1_guess = pi/4*ones(N,1);
```

```
beta_guess = 0.25;

X0 = [Va_guess;chi_guess;gama_guess;x1_guess;y1_guess;ht_guess;...
  CL_guess;mu1_guess;...
  tfg;beta_guess];

%%
A=[]; b=[]; Aeq=[]; beq=[];

opts = optimoptions('fmincon','Display','Iter','OptimalityTolerance',1e-5,...
     'ConstraintTolerance',1e-5,'Algorithm','interior-point',...
'MaxFunEvals',250000,'MaxIter',6000);
[Z,costval,exitflag,output] = fmincon(@costfun,X0,A,b,Aeq,beq,Lb,
Ub,@Constraints,opts);

%%
% Results obtained from the pseudospectral method

Vel = Z(1:N); % Velocity;
hdg = Z(N+1:2*N); % Heading
fpa = Z(2*N+1:3*N); % flight path angle - gama;
xv  = Z(3*N+1:4*N); % x coordinate;
yv  = Z(4*N+1:5*N); % y coordinate;
% zv  = Z(5*N+1:6*N); % height;
ht = Z(5*N+1:6*N);      % height
cl1 = Z(6*N+1:7*N); % Lift coefficient;
phi1 = Z(7*N+1:8*N); % Bank angle;
Tflight = Z(end-1);
beta = Z(end);

disp(['             ****************']);
disp(['             ****Results*****']);
disp(['             ****************']);
disp(['             Time of Flight :   ', num2str(Tflight), ' sec']);
disp(['             Min. Wind Shear:   ', num2str(beta), ' sec^{-1}']);
disp(['             ****************']);


t1 = (Tflight/2) + Tflight/2*tau;
t2 = linspace(min(t1),max(t1),200);
%%

% To verify the trajectory by performing the time marching simulation with initial
% conditions and control variations obtained from the pseudospectral method.

% x_ini = [Vel(N) hdg(N) fpa(N) xv(N) yv(N) ht(N) Tflight beta];
data = [flip(t1) flip(cl1) flip(phi1*180/pi)];
```

```
tspan = data(:,1);
for j=1:length(tspan)
    CL11(j) = data(j,2);         % lift coefficient
    mu11(j)= data(j,3)*pi/180;   % bank angle
end
t22 = tspan; t33 = tspan;
x0 = [Vel(N) hdg(N) fpa(N) xv(N) yv(N) ht(N)];
beta1 = beta;

[t11,x] = ode45(@(t,y) eqnmotion1(t,y,t22,CL11,t33,mu11),tspan,x0);
v11 = x(:,1); chi11 = x(:,2); gama11 = x(:,3);
x11 = x(:,4); y11 = x(:,5); ht11 = x(:,6);

CL111 = interp1(t11,CL11,t2,'spline');
mu111 = interp1(t11,mu11,t2,'spline');
chi111 = interp1(t11,chi11,t2,'spline');
gama111 = interp1(t11,gama11,t2,'spline');

%%
% Plotting of the results

figure(2);
subplot (3,2,1); plot (t1,Vel,'r.-');  hold on;  plot (t11,v11,'bx-');
                 xlabel ('Time [sec]'); ylabel ('Speed [m/s]');
                 legend('PS','RK4','Location','Best');
subplot (3,2,3); plot (t1,fpa*180/pi,'r.-'); hold on; plot (t11,gama11*180/pi,'bx-');
                 xlabel ('Time [sec]'); ylabel ('Gamma [deg]');
                 legend('PS','RK4','Location','Best');
subplot (3,2,5); plot (t1,hdg*180/pi,'r.-'); hold on; plot (t11,chi11*180/pi,'bx-');
                 xlabel ('Time [sec]'); ylabel ('Heading [deg]');
                 legend('PS','RK4','Location','Best');

subplot (3,2,2); plot (t1,cl1,'r.'); hold on; plot(t2,CL111,'b--');
                 xlabel ('Time [sec]'); ylabel ('Lift Coefficient');
subplot (3,2,4); plot (t1,phi1*180/pi,'r.'); hold on; plot(t2,mu111*180/pi,'b--');
                 xlabel ('Time [sec]'); ylabel ('Bank Angle [deg]');
subplot (3,2,6); plot (t1,ht,'r.-'); hold on; plot (t11,ht11,'bx-');
 xlabel ('Time [sec]'); ylabel ('Height [m]');

figure(3);
subplot (5,1,1); plot (chi111*180/pi,gama111*180/pi','b-','LineWidth',2.0);
xlabel ('Heading [deg]'); ylabel ('Gamma [deg]');
subplot (5,1,2); plot (hdg*180/pi,ht,'r.-');
xlabel ('Heading [deg]'); ylabel ('Height [m]');
subplot (5,1,3); plot (hdg*180/pi,Vel,'r.-');
xlabel ('Heading [deg]'); ylabel ('Speed [m]');
subplot (5,1,4); plot (chi111*180/pi,CL111,'b-','LineWidth',2.0);
```

```
xlabel ('Heading [deg]'); ylabel ('Lift Coefficient');
subplot (5,1,5); plot (chi111*180/pi,mu111*180/pi,'b-','LineWidth',2.0);
xlabel ('Heading [deg]'); ylabel ('Bank Angle')


figure(4);
subplot (3,1,1); plot (t1,xv,'r.-'); hold on; plot (t11,x11,'bx-');
            xlabel ('Time [sec]'); ylabel ('x');
            legend('PS','RK4','Location','Best');
subplot (3,1,2); plot (t1,yv,'r.-'); hold on; plot (t11,y11,'bx-');
            xlabel ('Time [sec]'); ylabel ('y');
            legend('PS','RK4','Location','Best');
subplot (3,1,3); plot (t1,ht,'r.-'); hold on; plot (t11,ht11,'bx-');
            xlabel ('Time [sec]'); ylabel ('ht');
            legend('PS','RK4','Location','Best');

figure(5);
for j=1:length(t1)
clf;
xvi = xv(j); yvi = yv(j); zvi= ht(j);
plot3 (xv,yv,ht,'r.-');     view([43,17]); hold on;
plot3 (xvi,yvi,zvi,'bo','LineWidth',3.0); hold on;
pause(0.1);
end

xlabel ('x [m]');
ylabel ('y [m]');
zlabel ('Height [m]');
```

Constraints.m

```
function [c,ceq] = Constraints(Z)
global Dmat N tau

%indexing as a single column vector for Va, chi,gama, x, y, z, CL, mu1;
% 1 to N corresponds to Va (relative Air Speed)
% N+1 to 2N corresponds to chi   (Heading angle)
% 2N+1 to 3N corresponds to gama (Flight Path angle)
% 3N+1 to 4N corresponds to x
% 4N+1 to 5N corresponds to y
% 5N+1 to 6N corresponds to z  (inertial height)
% 6N+1 to 7N corresponds to CL (Lift Coefficient)
% 7N+1 to 8N corresponds to mu (bank angle)
% 8N+1 corresponds to the time of flight tf
% 8N+2 corresponds to the windshear beta

Va = Z(1:N);
chi = Z(N+1:2*N);
gama1 = Z(2*N+1:3*N);
```

```
x1 = Z(3*N+1:4*N); y1 = Z(4*N+1:5*N); ht = Z(5*N+1:6*N);
CL = Z(6*N+1:7*N);
mu1 = Z(7*N+1:8*N);
tf1 = Z(8*N+1);
beta = Z(8*N+2);

ceq = zeros(6*N,1);

VaRHS = zeros(N,1);
chiRHS  = zeros(N,1);
gamaRHS = zeros(N,1);
x1RHS   = zeros(N,1);
y1RHS   = zeros(N,1);
% z1RHS = zeros(N,1);
htRHS  = zeros(N,1);


for j=1:N
xdot = eqnmotion([Va(j);chi(j);gama1(j);x1(j);y1(j);ht(j)],[CL(j);mu1(j)],beta);
VaRHS(j) = xdot(1);
chiRHS(j) = xdot(2);
gamaRHS(j) = xdot(3);
x1RHS(j) = xdot(4);
y1RHS(j) = xdot(5);
htRHS(j) = xdot(6);
end


ceq(1:N) = Dmat*Va - 0.5*tf1*VaRHS;
ceq(N+1:2*N) = Dmat*chi - 0.5*tf1*chiRHS;
ceq(2*N+1:3*N) = Dmat*gama1 - 0.5*tf1*gamaRHS;
ceq(3*N+1:4*N) = Dmat*x1 - 0.5*tf1*x1RHS;
ceq(4*N+1:5*N) = Dmat*y1 - 0.5*tf1*y1RHS;
% ceq(5*N+1:6*N) = Dmat*z1 - 0.5*tf1*z1RHS;
ceq(5*N+1:6*N) = Dmat*ht - 0.5*tf1*htRHS;

% ceq = [ceq' Va(1)-Va(N) chi(1)-chi(N)-2*pi gama1(1)-gama1(N) ...
%     x1(1)-x1(N) y1(1)-y1(N) ht(1)-ht(N)];

ceq = [ceq' Va(1)-Va(N) chi(1)-chi(N)-2*pi gama1(1)-gama1(N) ht(1) ht(N) ...
      x1(1) x1(N) y1(1) y1(N)];

c = [];

end
```

costfun.m

```
function cost = costfun(Z)
```

```
cost = Z(end);
end
```

eqnmotion.m

```
function [xdot] = eqnmotion(x,u,beta)

global K CD0 rho S m g

va = x(1);
chi = x(2);
gama1 = x(3);
x1 = x(4); y1 = x(5);  ht1 = x(6);
CL = u(1);
mu1 = u(2);


xdot = zeros(6,1);

Lbym = ( CL*0.5*rho*va^2*S )/m;
Dbym = ( (CD0 + K*CL^2)*0.5*rho*va^2*S )/m;

dhdt = va*sin(gama1);
Wdot = beta*dhdt;

% dwdz = -beta;
% dzdt = -va*sin(gama1);
% xdot(1) = -Dbym - g*sin(gama1) - beta*cos(gama1)*cos(chi)*va*sin(gama1);
% xdot(2) = 1.0/(va*cos(gama1))*( Lbym*sin(mu1) + sin(chi)*beta*va*sin(gama1));
% xdot(3) = (1.0/va)* ( Lbym*cos(mu1) - g*cos(gama1) + beta*cos(chi)*
sin(gama1)*va*sin(gama1) );

xdot(1) = -Dbym - g*sin(gama1) - Wdot*cos(gama1)*sin(chi);
xdot(2) = 1.0/(va*cos(gama1)) * ( Lbym*sin(mu1) - Wdot*cos(chi) );
xdot(3) = 1.0/va * ( Lbym*cos(mu1) - g*cos(gama1) + Wdot*sin(gama1)*sin(chi) );

xdot(4) = va*cos(gama1)*sin(chi) + beta*ht1;
xdot(5) = va*cos(gama1)*cos(chi);
xdot(6) = dhdt;

end
```

eqnmotion1.m

```
function [xdot] = eqnmotion1(t,x,t2,cl,t3,phi1)

global K CD0 rho S m g beta1

va = x(1);
```

```
chi = x(2);
gama1 = x(3);
x1 = x(4); y1 = x(5);  ht1 = x(6);

CL = interp1(t2,cl,t,'spline');
mu1 = interp1(t3,phi1,t,'spline');

xdot = zeros(6,1);

Lbym = ( CL*0.5*rho*va^2*S )/m;
Dbym = ( (CD0 + K*CL^2)*0.5*rho*va^2*S )/m;

dhdt = va*sin(gama1);
Wdot = beta1*dhdt;

xdot(1) = -Dbym - g*sin(gama1) - Wdot*cos(gama1)*sin(chi);
xdot(2) = 1.0/(va*cos(gama1)) * ( Lbym*sin(mu1) - Wdot*cos(chi) );
xdot(3) = 1.0/va * ( Lbym*cos(mu1) - g*cos(gama1) + Wdot*sin(gama1)*sin(chi) );

xdot(4) = va*cos(gama1)*sin(chi) + beta1*ht1;
xdot(5) = va*cos(gama1)*cos(chi);
xdot(6) = dhdt;

end
```

legDc.m

```
function [x,D]=legDc(N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% legDc.m
%
% Computes the Legendre differentiation matrix with collocation at the
% Legendre-Gauss-Lobatto nodes.
%
% Reference:
%   C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Tang, "Spectral Methods
%   in Fluid Dynamics," Section 2.3. Springer-Verlag 1987
%
% Written by Greg von Winckel - 05/26/2004
% Contact: gregvw@chtm.unm.edu
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Truncation + 1
N1=N+1;
```

```
% CGL nodes
xc=cos(pi*(0:N)/N)';

% Uniform nodes
xu=linspace(-1,1,N1)';

% Make a close first guess to reduce iterations
if N<3
    x=xc;
else
    x=xc+sin(pi*xu)./(4*N);
end

% The Legendre Vandermonde Matrix
P=zeros(N1,N1);

% Compute P_(N) using the recursion relation
% Compute its first and second derivatives and
% update x using the Newton-Raphson method.

xold=2;
while max(abs(x-xold))>eps

    xold=x;

    P(:,1)=1;    P(:,2)=x;

    for k=2:N
        P(:,k+1)=( (2*k-1)*x.*P(:,k)-(k-1)*P(:,k-1) )/k;
    end

    x=xold-( x.*P(:,N1)-P(:,N) )./( N1*P(:,N1) );
end

X=repmat(x,1,N1);
Xdiff=X-X'+eye(N1);

L=repmat(P(:,N1),1,N1);
L(1:(N1+1):N1*N1)=1;
D=(L./(Xdiff.*L'));
D(1:(N1+1):N1*N1)=0;
D(1)=(N1*N)/4;
D(N1*N1)=-(N1*N)/4;
```