```
clear all
clc
tic
%--------------------------------------------------------------------------%
%------------------ Provide Auxiliary Data for Problem --------------------%
%--------------------------------------------------------------------------%
auxdata.rho  = 1.2;
auxdata.CD0  = 0.00873;
auxdata.K    = 0.045;
auxdata.g    = 9.81;
auxdata.m    = 10;
auxdata.S    = 1;
auxdata.W0   = 0;
auxdata.A    = 1.3;

%--------------------------------------------------------------------%
%----------------------- Boundary Conditions ------------------------%
%--------------------------------------------------------------------%
t0 = 0; x0 = 0; y0 = 0; z0 = 0; v0 = 100;

%--------------------------------------------------------------------%
%----------------------- Limits on Variables ------------------------%
%--------------------------------------------------------------------%
c = pi/180;
tf_min    = 1;          tf_max    = 100;
x_min     = -1000;      x_max     = +1000;
y_min     = -1000;      y_max     = +1000;
z_min     =  0;         z_max     = +1000;
v_min     = +10;        v_max     = +350;
gamma_min = -75*c;      gamma_max = 75*c;
psi_min   = -540*c;     psi_max   = 90*c;
beta_min  = 0.005;      beta_max  = 0.15;
CL_min    = -0.5;       CL_max    = 1.5;
mu_min   = -75*c;       mu_max    = 75*c;

%--------------------------------------------------------------------%
%--------------- Set Up Problem Using Data Provided Above -----------%
%--------------------------------------------------------------------%
bounds.phase.initialtime.lower  = t0;
bounds.phase.initialtime.upper  = t0;
bounds.phase.finaltime.lower    = tf_min;
bounds.phase.finaltime.upper    = tf_max;
bounds.phase.initialstate.lower = [x0, y0, z0, v_min, gamma_min, psi_min];
bounds.phase.initialstate.upper = [x0, y0, z0, v_max, gamma_max, psi_max];
bounds.phase.state.lower        = [x_min, y_min, z_min, v_min, gamma_min, psi_min];
bounds.phase.state.upper        = [x_max, y_max, z_max, v_max, gamma_max, psi_max];
bounds.phase.finalstate.lower   = [x0, y0, z0, v_min, gamma_min, psi_min];
bounds.phase.finalstate.upper   = [x0, y0, z0, v_max, gamma_max, psi_max];
bounds.phase.control.lower      = [CL_min, mu_min];
bounds.phase.control.upper      = [CL_max, mu_max];
bounds.phase.path.lower         = -2;
bounds.phase.path.upper         = 5;
bounds.eventgroup(1).lower      = [0, 0, -2*pi];
bounds.eventgroup(1).upper      = [0, 0, -2*pi];
bounds.parameter.lower          = beta_min;
bounds.parameter.upper          = beta_max;

%-------------------------------------------------------------------------%
%---------------------- Provide Guess of Solution ------------------------%
%-------------------------------------------------------------------------%
N                               = 100;
CL0                             = CL_max;
tGuess                          = linspace(0,24,N).';
xguess                          = 500*cos(2*pi*tGuess/24)-500;
yguess                          = 300*sin(2*pi*tGuess/24);
zguess                          = -400*cos(2*pi*tGuess/24)+400;
vguess                          = 0.8*v0*(1.5+cos(2*pi*tGuess/24));
gammaguess                      = pi/6*sin(2*pi*tGuess/24);
psiguess                        = -1-tGuess/4;
CLguess                         = CL0*ones(N,1)/3;
muguess                         = -ones(N,1);
betaguess                       = 0.08;
guess.phase.time                = tGuess;
guess.phase.state               = [xguess, yguess, zguess, vguess, gammaguess, psiguess];
guess.phase.control             = [CLguess, muguess];
guess.parameter                 = betaguess;

%-------------------------------------------------------------------------%
%----------Provide Mesh Refinement Method and Initial Mesh ---------------%
%-------------------------------------------------------------------------%
mesh.maxiteration               = 10;
mesh.method                     = 'hp-LiuRao';
mesh.tolerance                  = 1e-6;

%-------------------------------------------------------------------%
%---------- Configure Setup Using the information provided ---------%
%-------------------------------------------------------------------%
setup.name                             = 'DS_MAIN';
setup.functions.continuous             = @DSContinuous;
setup.functions.endpoint               = @DSEndpoint;
setup.nlp.solver                       = 'ipopt';
setup.nlp.ipoptoptions.linear_solver   = 'ma57';
setup.displaylevel                     = 2;
setup.auxdata                          = auxdata;
setup.bounds                           = bounds;
setup.guess                            = guess;
setup.mesh                             = mesh;
setup.derivatives.supplier             = 'sparseCD';
setup.derivatives.derivativelevel      = 'second';
setup.scales.method                    = 'automatic-bounds';
setup.method                           = 'RPM-Differentiation';
```

```
%-------------------------------------------------------------------%
%------------------ Solve Problem Using GPOPS-II--------------------%
%-------------------------------------------------------------------%
output = gpops2(setup);
solution = output.result.solution;
multi_axes;
minimum_beta = solution.parameter
z = max(solution.phase.state(:,3));
A = auxdata.A ;
Min_required_windspeed = minimum_beta.*(A.*z + (1 - A)./213.*z.^2)
toc
DS_Stability_logarithmic;
```