Contents

```
clearvars
close all
clc
```

Prelims

```
model = 'mavsim_chap11';
P.Ts = 0.005;      % autopilot sample rat
P.gravity = 9.8;  % gravity (m/s^2)
s = tf('s');
run_straight_line = 1;
run_straight_line_fillet = 1;
test_dubins = 1;
run_dubins = 1;
```

Params for Aersonade UAV

```
aerosonde
```

Params for autopilot controllers

```
cntrlParams
```

```
Trim converged to 2.01425e-12
```

Wind parameters

```
windParams
```

Commanded trajectory

```
trajectoryParams
```

Sensor parameters

```
sensorParams
```

Guidance model parameters

```
guidanceParams
```

Path follower parameters

```
pathFollowingParams
```

Define the desired path

The following code defines a desired path similar to Figures 11.2, 11.6, and 11.14. The newpath flag is defined such that it is active on the first sample of the simulation, to initialize the path managers.

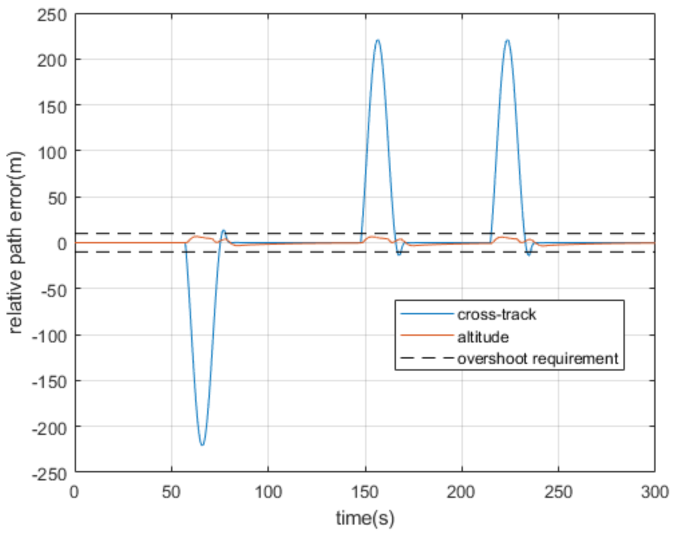
```
d = 2000;
w1 = [P.pn0, P.pe0, P.pd0]';
w2 = [P.pn0 + d, P.pe0, P.pd0]';
```

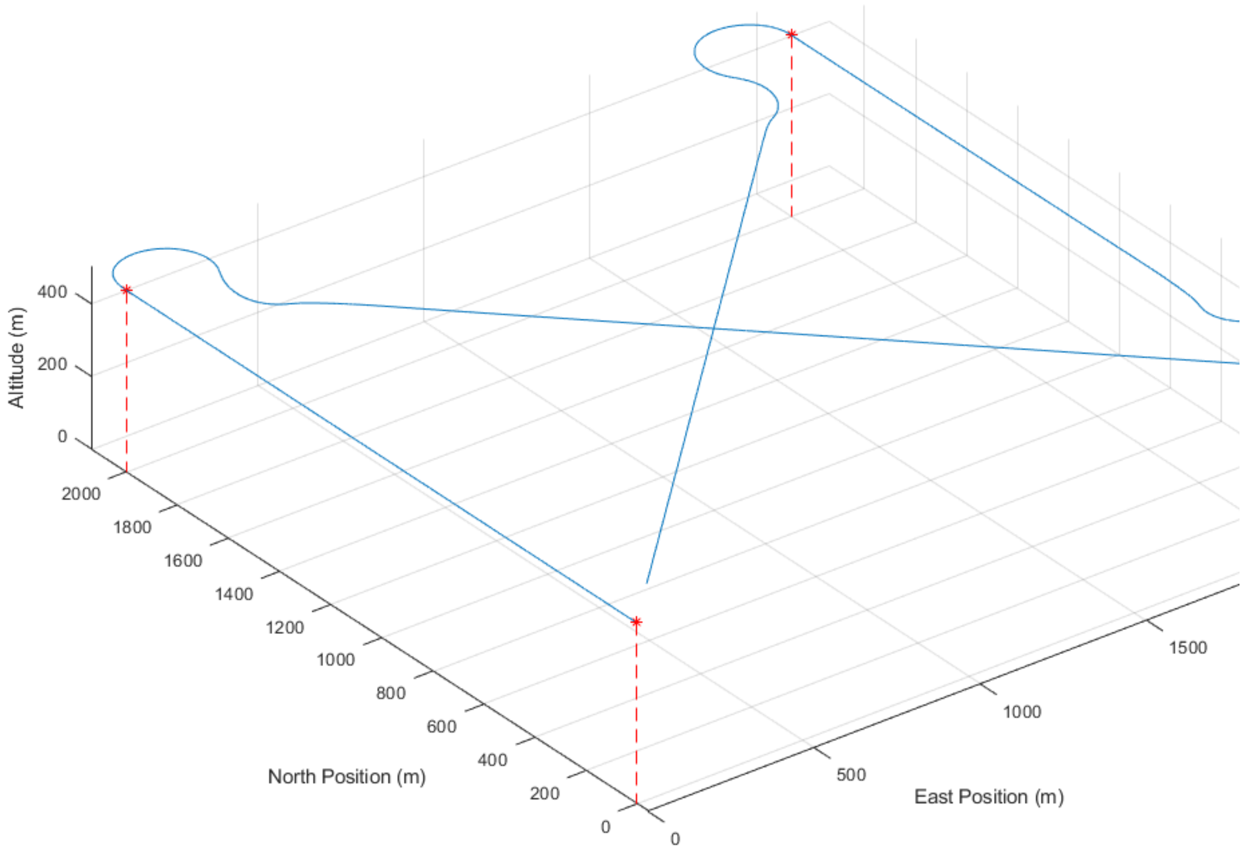
```
w3 = [P.pn0, P.pe0 + d, P.pd0]';
w4 = [P.pn0 + d, P.pe0 + d, P.pd0]';
w5 = [P.pn0, P.pe0, P.pd0]';
w6 = w2;
w7 = w3;
W = [w1, w2, w3, w4, w5, w6, w7];
Chi = [0, 45, 45, 225, 0, 45, 45]*pi/180;
waypoint_path = timeseries(W,0);
course_angles = timeseries(Chi,0);
newpath = timeseries([1,1,0],[0,P.Ts,P.Ts]);
```

Waypoint following with straight-line paths

The following code simulates the path manager using algorithm 5. Implement algorithm 5 in the followWpp() function and demonstrate successful waypoint following.

```
if run_straight_line
    tsim = 60*5;
    skipStatePlots = true;
    path_manager_selection = 5;
    sim(model,tsim);
    generatePlots;
end
```

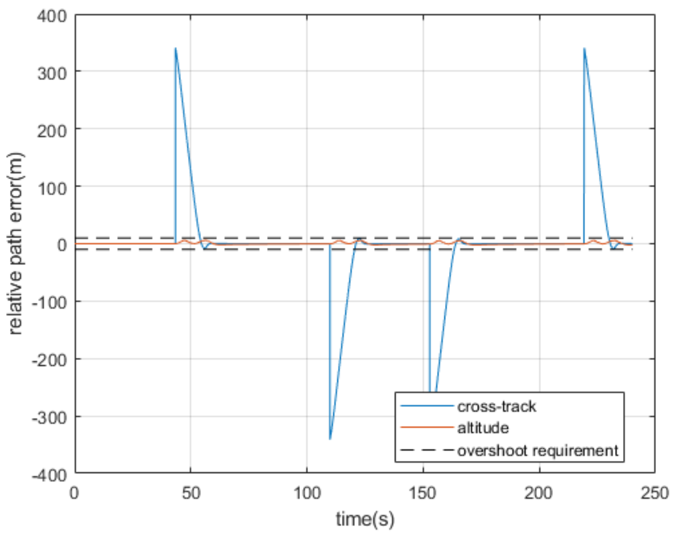


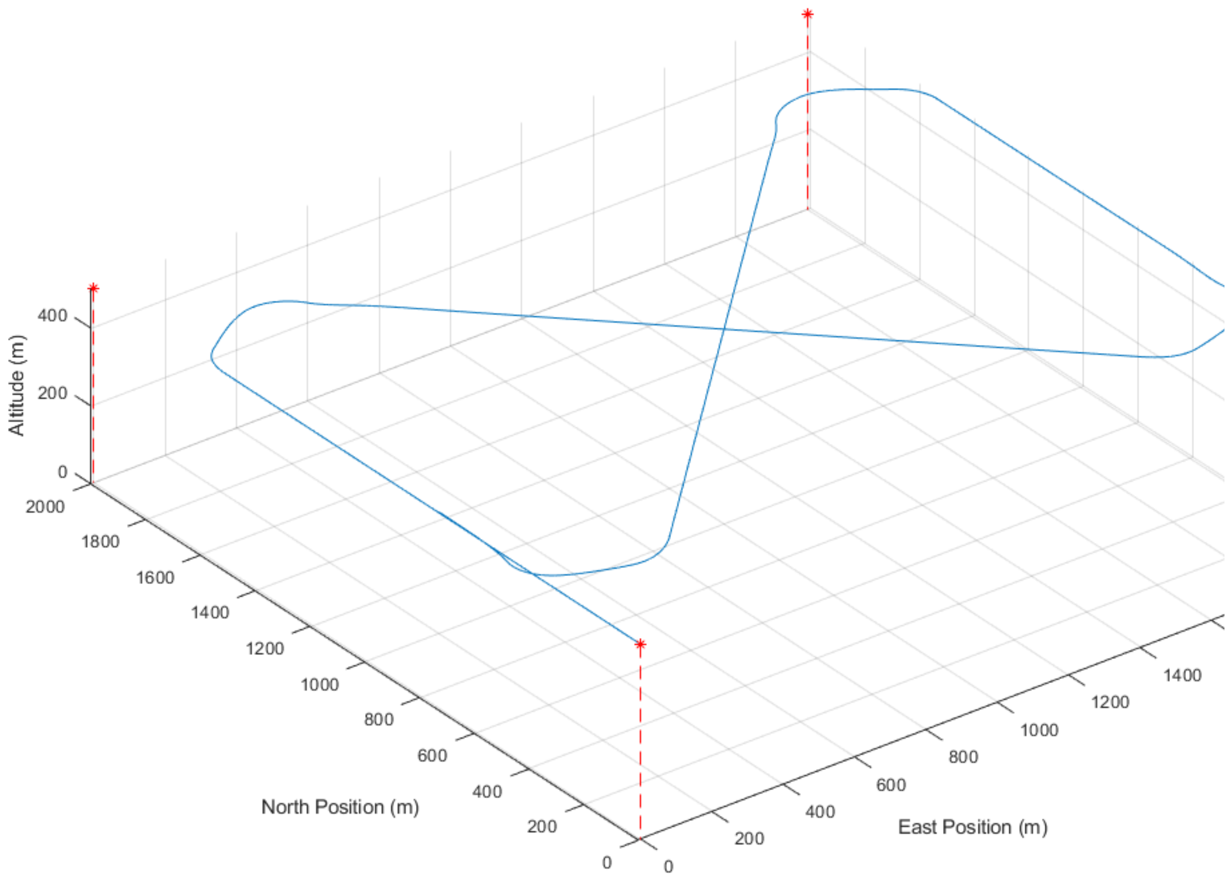


Waypoint following with straight-lines paths connected by fillets

The following code simulates the path manager using algorihtm 6. Implement algorithm 6 in followWppFillet() function and demonstrate successful waypoint following.

```
if run_straight_line_fillet
    tsim = 60*4;
    skipStatePlots = true;
    path_manager_selection = 6;
    sim(model,tsim);
    generatePlots;
end
```

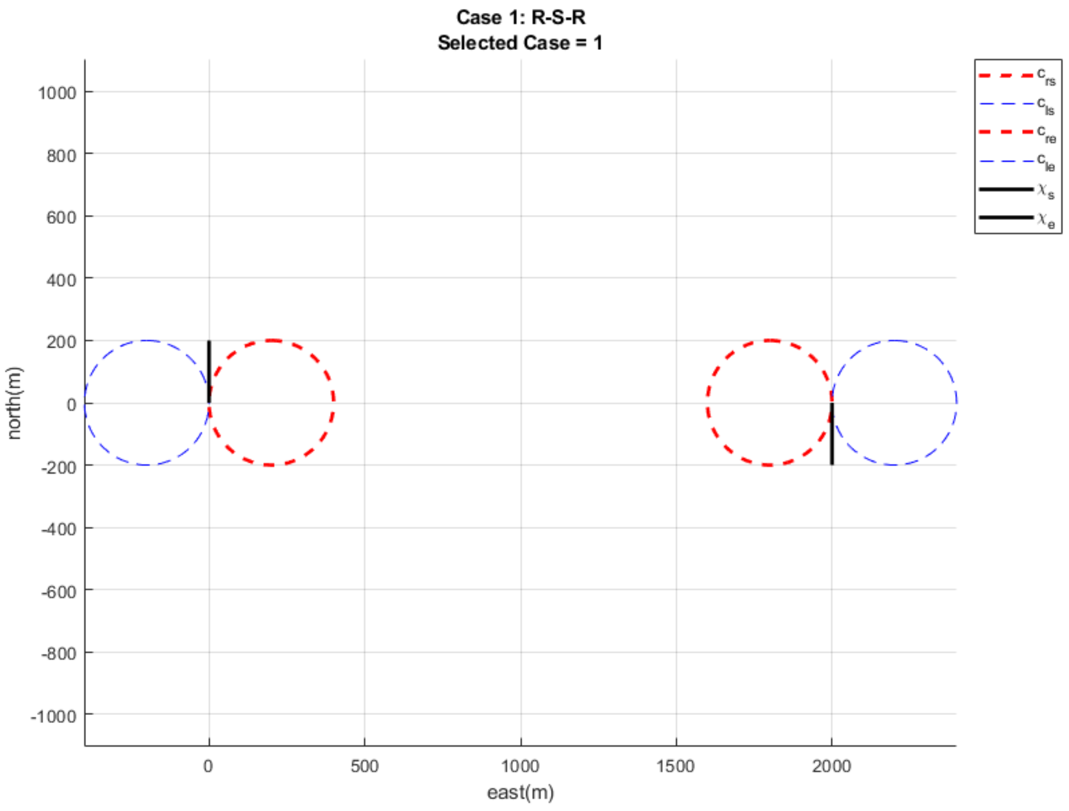


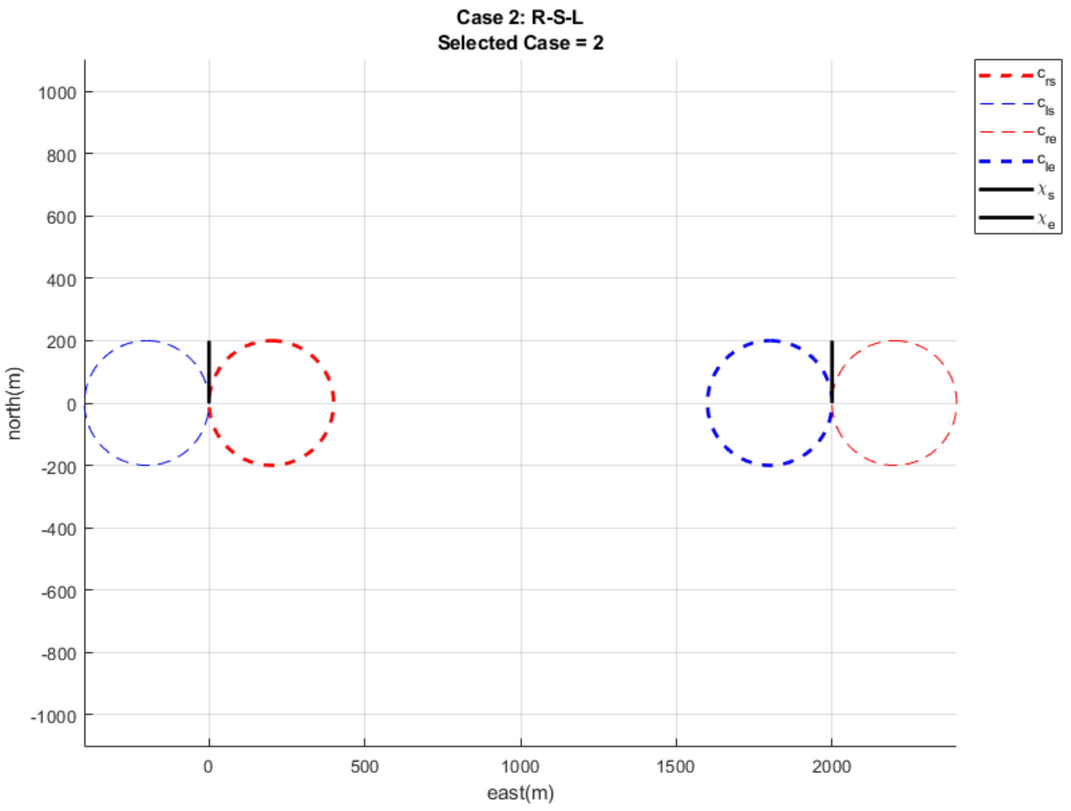


Determining the Dubins path

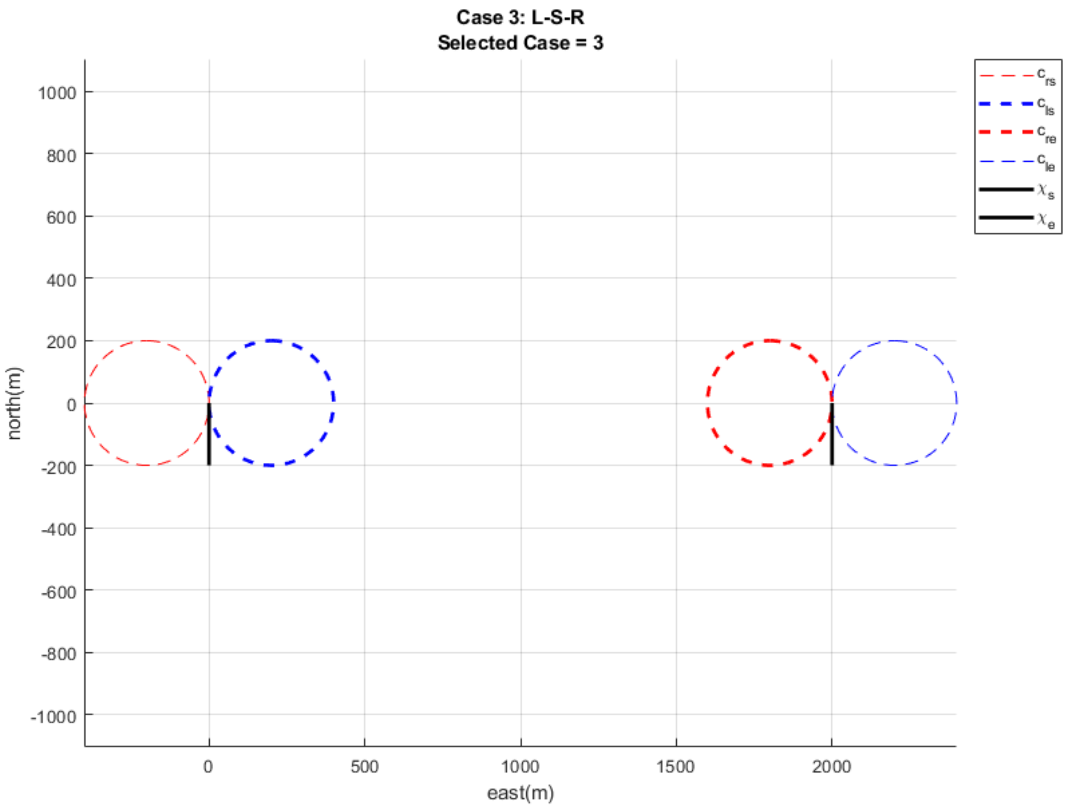
The following code tests your implementation of algorithm 7. Implement algorithm 7 in findDubinsParameters(), and demonstrate the correct determination of the Dubins path for the provide start and end vehicle configurations. The figures shows the candidate orbits of the Dubins path, and the beginning and end configurations. The selected orbits are denote using bold dashed lines.

```
if test_dubins
    figpos = [461, 222, 1011, 630]';

    % The following path should result in the selection of Case 1: R-S-R
    p_s = [0,0,0]';
    chi_s = 0;
    p_e = [0, 2000, 0]';
    chi_e = pi;
    dp = findDubinsParameters(p_s, chi_s, p_e, chi_e, P.min_radius);
    h_case_1 = figure;
    plotDubinsPath(p_s, chi_s, p_e, chi_e, P.min_radius, h_case_1);
    axis equal;
    xlabel('east(m)');
    ylabel('north(m)');
    grid on;
    legend('location','northeastoutside');
    h_case_1.Position = figpos;
    titlestr = sprintf('Case 1: R-S-R\nSelected Case = %d',dp.case);
    title(titlestr);

    % The following path should result in the selection of Case 2: R-S-L
    p_s = [0,0,0]';
    chi_s = 0;
    p_e = [0, 2000, 0]';
    chi_e = 0;
    dp = findDubinsParameters(p_s, chi_s, p_e, chi_e, P.min_radius);
    h_case_1 = figure;
    plotDubinsPath(p_s, chi_s, p_e, chi_e, P.min_radius, h_case_1);
    axis equal;
    xlabel('east(m)');
    ylabel('north(m)');
    grid on;
    legend('location','northeastoutside');
    h_case_1.Position = figpos;
    titlestr = sprintf('Case 2: R-S-L\nSelected Case = %d',dp.case);
    title(titlestr);
    % The following path should result in the selection of Case 3: L-S-R
    p_s = [0,0,0]';
    chi_s = pi;
    p_e = [0, 2000, 0]';
    chi_e = pi;
    dp = findDubinsParameters(p_s, chi_s, p_e, chi_e, P.min_radius);
    h_case_1 = figure;
    plotDubinsPath(p_s, chi_s, p_e, chi_e, P.min_radius, h_case_1);
    axis equal;
    xlabel('east(m)');
    ylabel('north(m)');
    grid on;
    legend('location','northeastoutside');
    h_case_1.Position = figpos;
    titlestr = sprintf('Case 3: L-S-R\nSelected Case = %d',dp.case);
    title(titlestr);

    % The following path should result in the selection of Case 4: L-S-L
    p_s = [0,2000,0]';
    chi_s = 0;
    p_e = [0, 0, 0]';
    chi_e = pi;
    dp = findDubinsParameters(p_s, chi_s, p_e, chi_e, P.min_radius);
    h_case_1 = figure;
    plotDubinsPath(p_s, chi_s, p_e, chi_e, P.min_radius, h_case_1);
    axis equal;
    xlabel('east(m)');
    ylabel('north(m)');
    grid on;
    legend('location','northeastoutside');
    h_case_1.Position = figpos;
    titlestr = sprintf('Case 4: L-S-L\nSelected Case = %d',dp.case);
    title(titlestr);

end
```

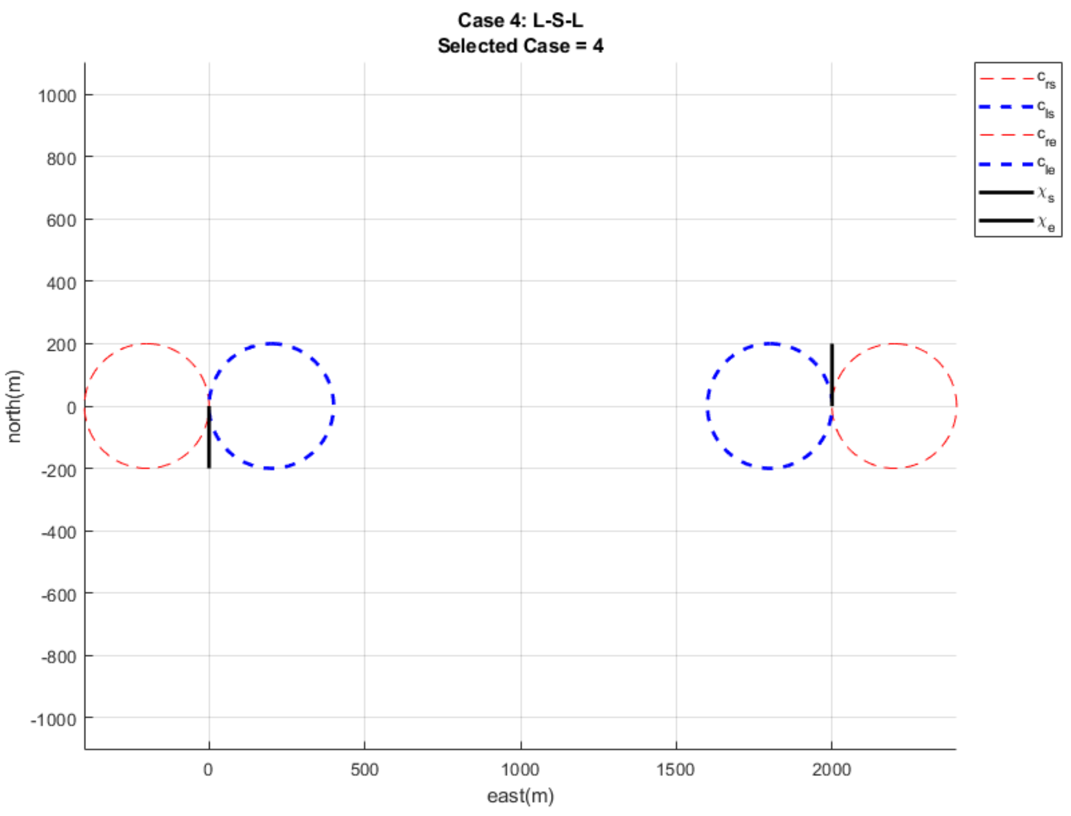




Case 2: R-S-L
Selected Case = 2
c_{rs}
c_{ls}
c_{re}
c_{le}
χ_s
χ_e
north(m)
east(m)



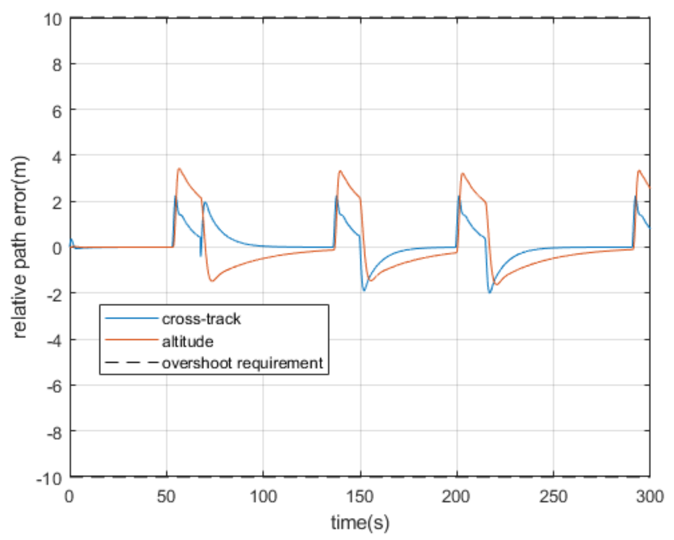
Case 3: L-S-R
Selected Case = 3
c_{rs}
c_{ls}
c_{re}
c_{le}
χ_s
χ_e
north(m)
east(m)

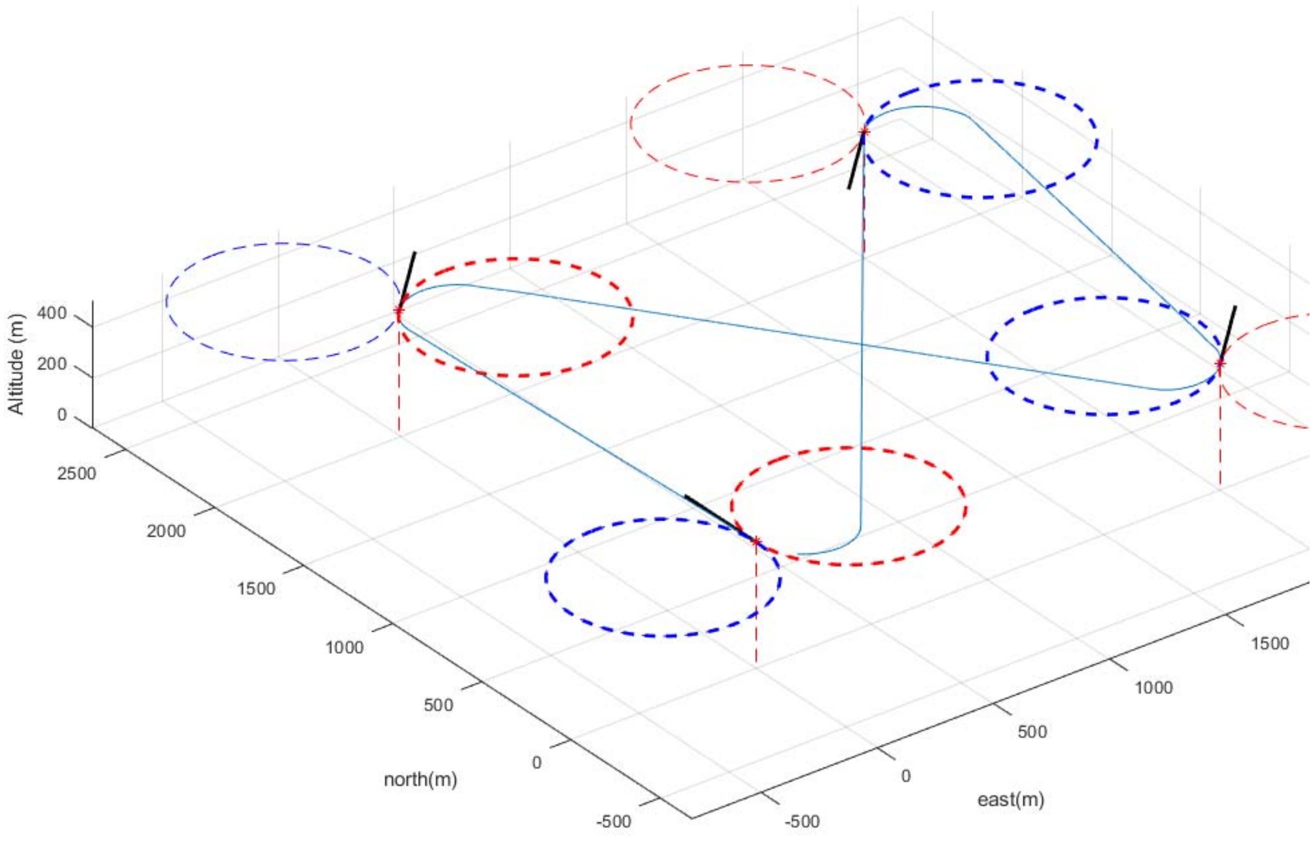


Waypoint following with Dubins paths

The following code simulates the path manager using algorihtm 8. Implement algorithm 8 in followWppDubins() function and demonstrate successful waypoint following.

```
if run_dubins
    tsim = 60*5;
    skipStatePlots = true;
    path_manager_selection = 8;
    sim(model,tsim);
    generatePlots;
end
```




Altitude (m)
400
200
0
2500
2000
1500
1000
500
0
-500
north(m)
-500
0
500
1000
1500
east(m)