CS6380 April 13-15 2020

FAA-NASA vs. LSD Strategic Deconfliction

Problem Statement (A8)

For this problem, we compare the FAA-NASA Strategic Deconfliction (FNSD) approach (pairwise flight deconfliction) to Lane-Based Strategic Deconfliction (LSD). In particular:

- **FNSD**: develop as efficient a method as possible to deconflict aircraft using the grid based method. In this approach a new flight must determine the grid elements it will cross, find any other flights operating in those grid elements at the same time, and make sure it does not violate the safe separation distance.
- **LSD**: use the LSD method to schedule flights.

To do this involves setting up a flight region which allows reasonable comparisons, as well as some way to get flight paths for both methods.

Measures must be developed and scenarios run which clearly demonstrate the advantages and disadvantages of the two methods.

You should handin the source code used in the study. The code should conform to the style requested in the class materials.

Readings

- An Efficient Strategic Deconfliction Algorithm for Lane-Based Large-Scale UAV Flight Planning, Thomas C. Henderson, David Sacharny and Michael Cline, UUCS-19-005, September, 2019. (Henderson2019.pdf)
- UAS Traffic Management (UTM) Project Strategic Deconfliction: System Requirements Final Report, J. Rios, July, 2018. (Rios2018a.pdf)

These can be found at: ~tch, notes, RES, UAM

Lanes

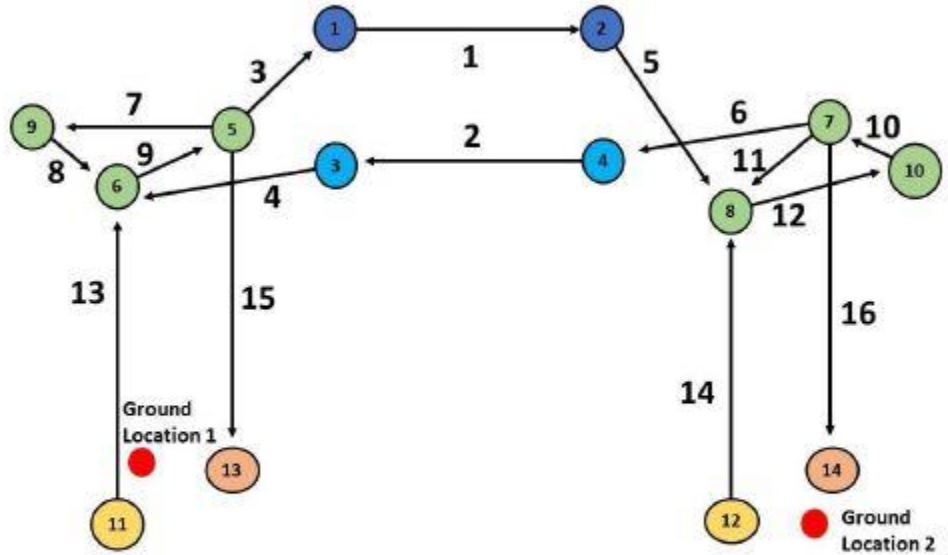


Figure 1: The Lanes (and Vertexes) for the Two Ground Locations Case.

Roundabout

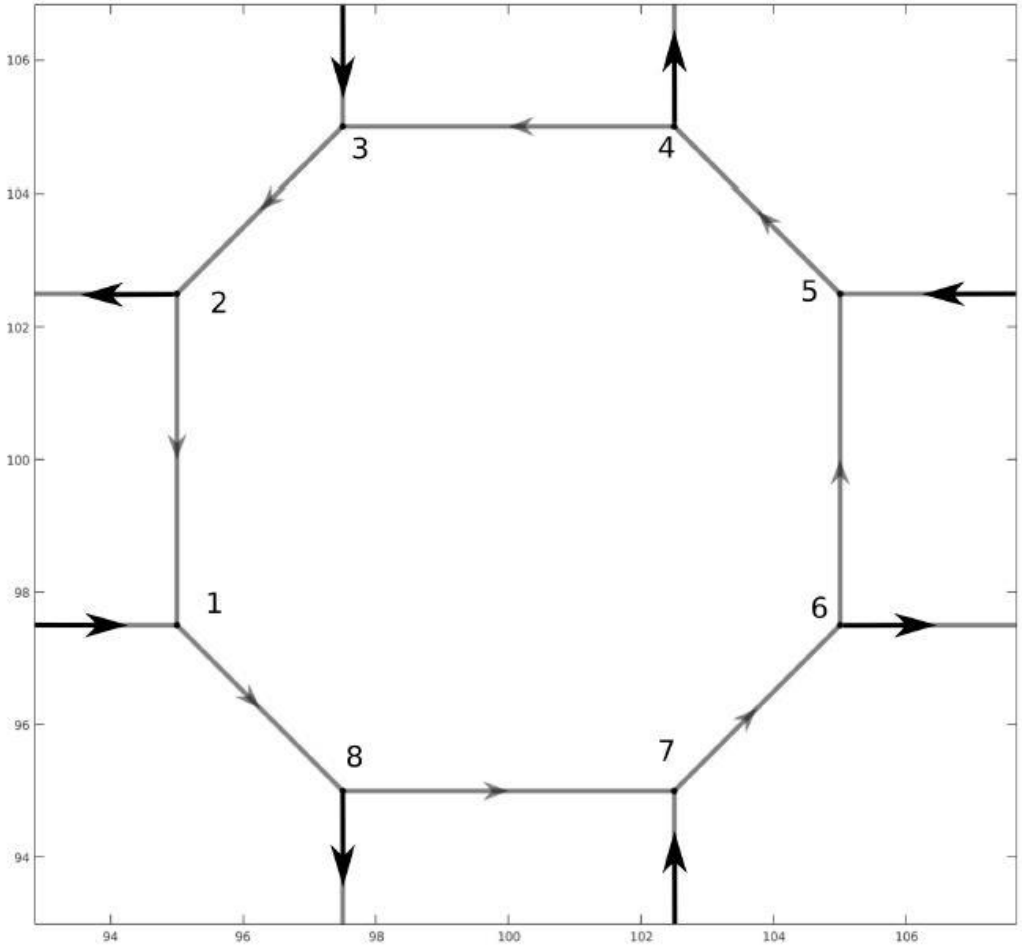
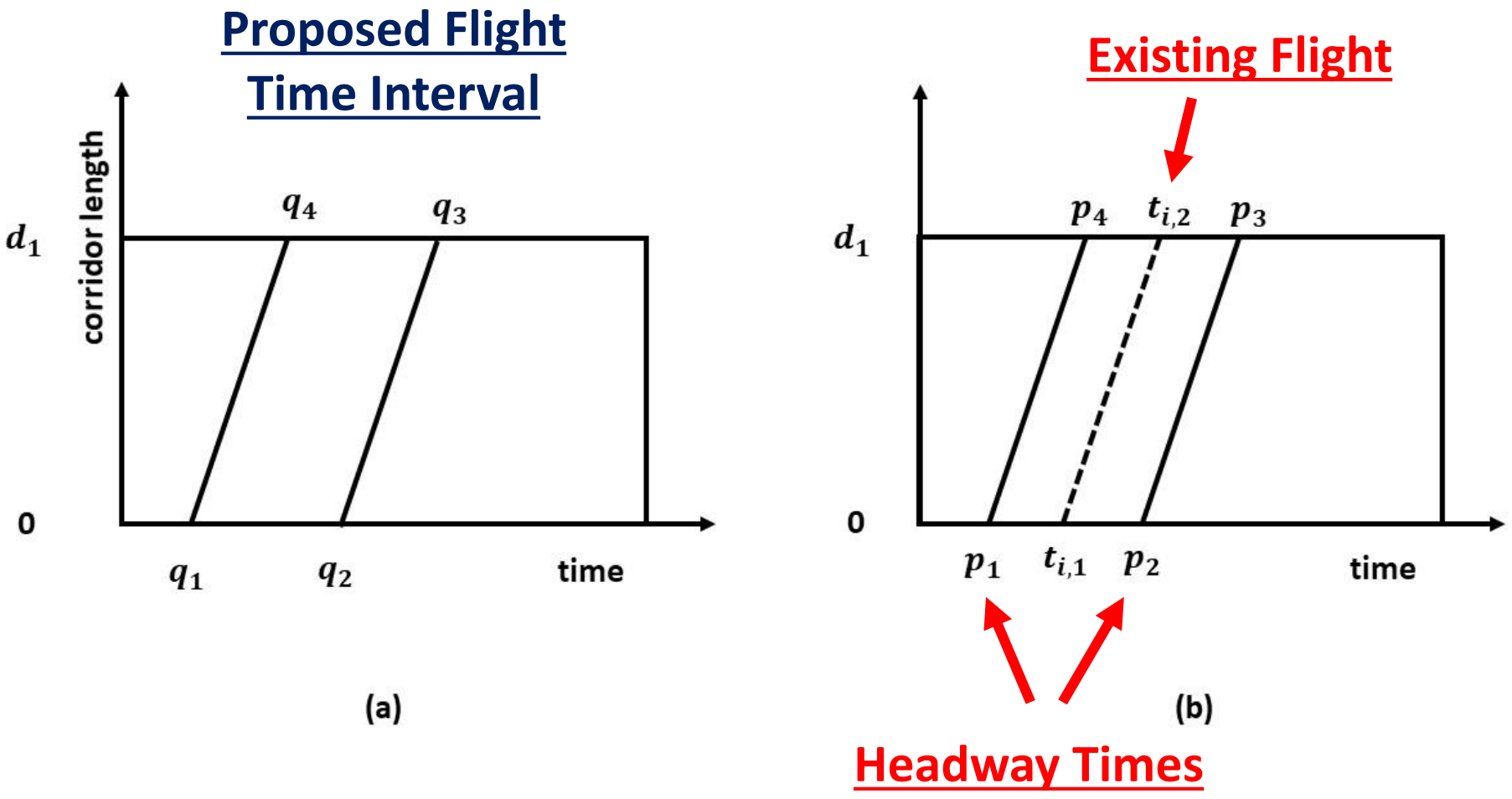


Figure 2: Airway Roundabout.

Space Time Lane Diagram



Space Time Lane Diagram

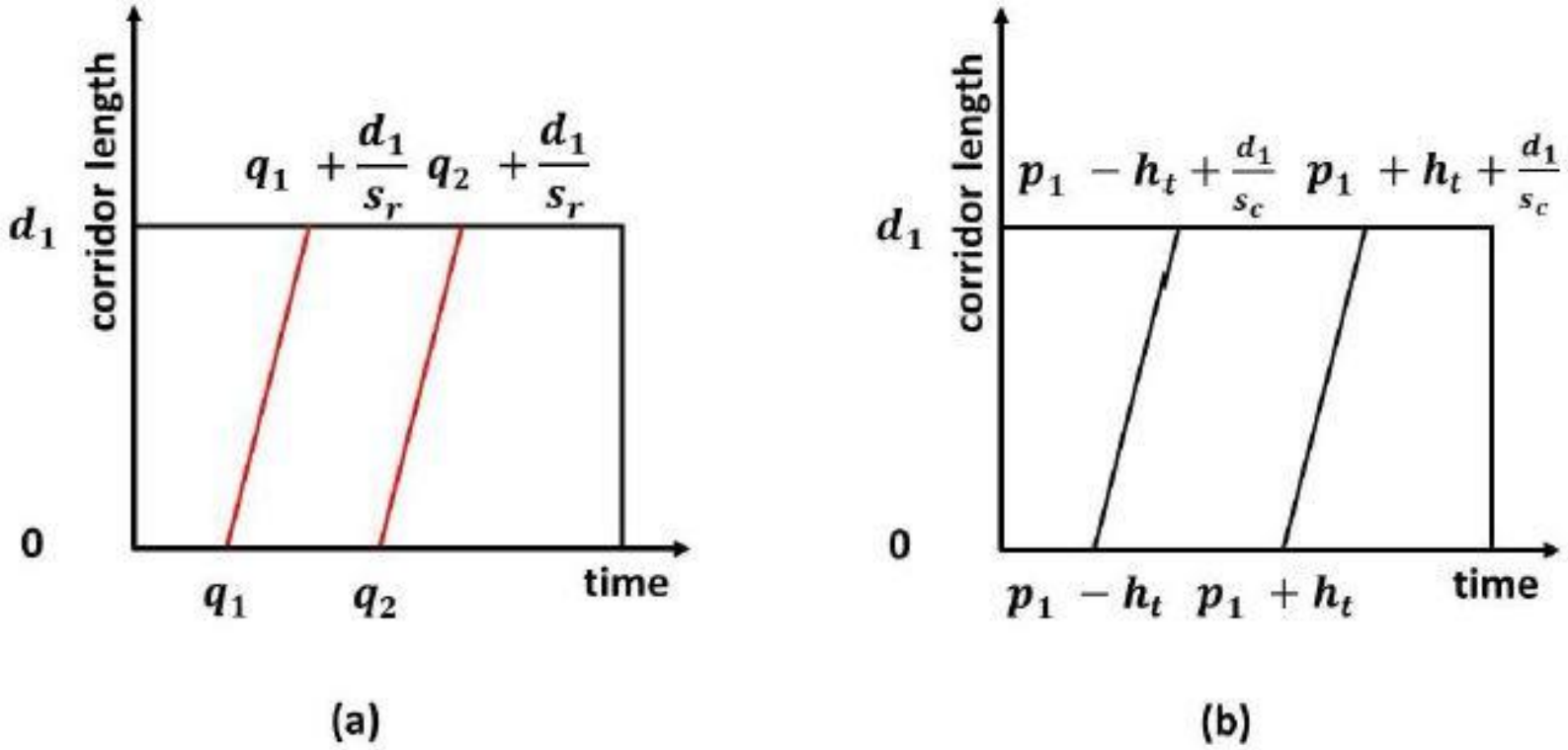
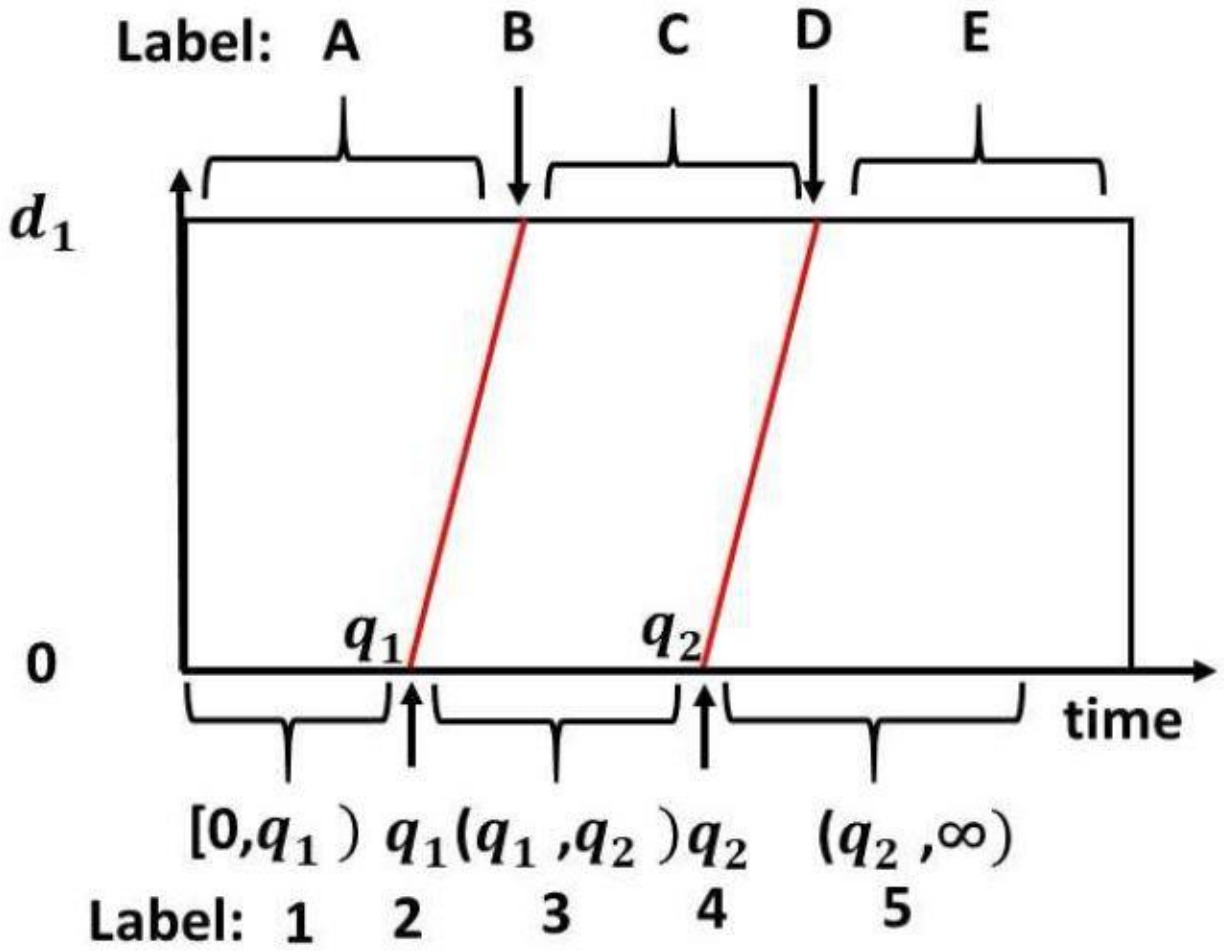


Figure 4: Space-Time Lane Diagrams: (a) trajectory boundaries for requested launch time interval $[q_1, q_2]$. (b) The headway boundary trajectories for a scheduled flight which enters the lane at time p and exits at time p'.

STLD Labels



- *Label 1*: The interval $[0, q_1)$
- *Label 2*: The point q_1
- *Label 3*: The interval (q_1, q_2)
- *Label 4*: The point q_2
- *Label 5*: The interval (q_2, ∞)

- *Label A*: The interval $[0, q_1 + \frac{d_1}{s^r})$
- *Label B*: The point $q_1 + \frac{d_1}{s^r}$
- *Label C*: The interval $(q_1 + \frac{d_1}{s^r}, q_2 + \frac{d_1}{s^r})$
- *Label D*: The point $q_2 + \frac{d_1}{s^r}$
- *Label E*: The interval $(q_2 + \frac{d_1}{s^r}, \infty)$

Figure 5: Space-Time Lane Diagram Labels. 1,2,3,4,5 indicate intervals and times at the entry to the lane, and A,B,C,D,E indicate times at the lane exit.

STLD Configurations

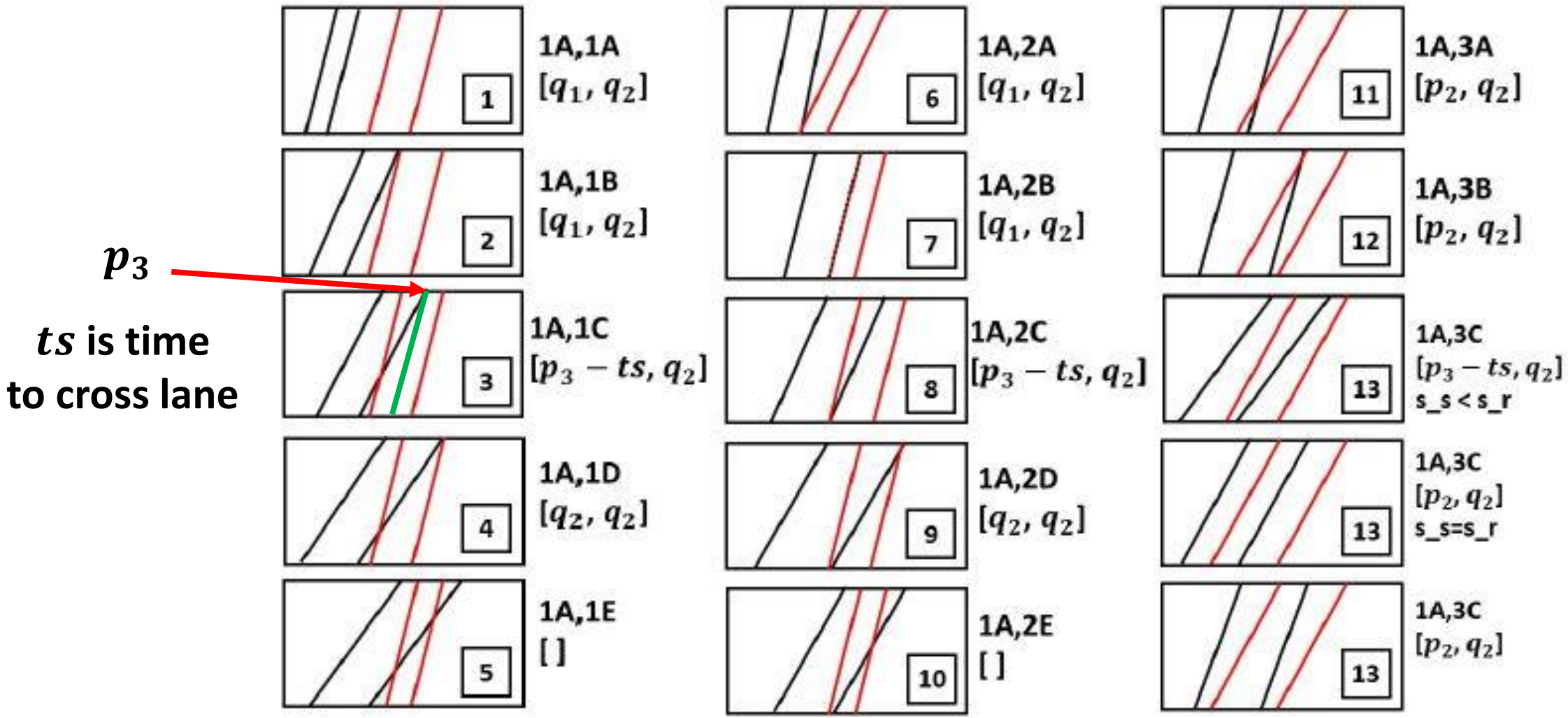


Figure 6: Space-Time Lane Diagrams for the First 13 Possible Label Combinations.

Lane Strategic Deconfliction (LSD)

Algorithm SD (Strategic Deconfliction)
On input:
lanes: lane sequence for requested flight
$[q_1, q_2]$: requested launch interval
n_c: number of lanes
flights: flights per lane
h_t: maximum required headway time
On output:
Safe time intervals to launch
begin
possible_intervals $\leftarrow [q_1, q_2]$
for each lane $c \in$ lanes
time_offset $\leftarrow$ time to get to lane c
possible_intervals $\leftarrow$ possible_intervals + time_offset
for each flight, f, in lane c
new_intervals $\leftarrow \emptyset$
for each interval in possible_intervals
$[t_1, t_2] \leftarrow$ interval i
label $\leftarrow$ get_label($t_{f,1}, t_{f,2}, s_f, t_1, t_2, s, h_t$)
f_int $\leftarrow$ get_interval(label,$t_{f,1}, t_{f,2}, s_f, t_1, t_2, s, h_t$)
new_intervals $\leftarrow$ merge(new_intervals,f_int)
end
end
possible_intervals $\leftarrow$ new_intervals
end
possible_intervals $\leftarrow$ possible_intervals - time to last lane

LSD Computational Complexity

The key computational cost of this algorithm is the determination of f_int; each instance of this can be done in constant time; call it operation $\mathcal{I}$. Then given n lanes, f_k flights in lane k, and f is the total number of flights in the lane sequence, then the total number of $\mathcal{I}$ operations is less than or equal to:

$$\sum_{k=1}^{n} f_k + \sum_{i \neq j} f_i f_j$$

The second sum dominates the complexity, and assuming f_k is on average $\frac{f}{n}$, and since there are $\binom{n}{2}$ terms, then the big O complexity is $O(f^2)$.

$$\frac{n(n+1)}{2} * \frac{f^2}{n^2}$$

STLD Label Enumeration

Labels	Intervals
1A,1A	$[q_1, q_2]$
1A,1B	$[q_1, q_2]$
1A,1C	$[p_3 - t_s, q_2]$
1A,1D	$[q_2, q_2]$
1A,1E	$\emptyset$
1A,2A	$[q_1, q_2]$
1A,2B	$[q_1, q_2]$
1A,2C	$[p_3 - t_s, q_2]$
1A,2D	$[q_2, q_2]$
1A,2E	$\emptyset$
1A,3A	$[p_2, q_2]$
1A,3B	$[p_2, q_2]$
1A,3C	$[p_3 - t_s, q_2, <]$
1A,3C	$[p_2, q_2, \geq]$
1A,3D	$[q_2, q_2]$
1A,3E	$\emptyset$
1A,4A	$[q_2, q_2]$
1A,4B	$[q_2, q_2]$
1A,4C	$[q_2, q_2]$
1A,4D	$[q_2, q_2]$
1A,4E	$\emptyset$
1A,5A	$\emptyset$
1A,5B	$\emptyset$
1A,5C	$\emptyset$

Labels	*Intervals*
1C,5E	$\emptyset$
1D,1E	$\emptyset$
1D,2E	$\emptyset$
1D,3E	$\emptyset$
1D,4E	$\emptyset$
1D,5E	$\emptyset$
1E,1E	$\emptyset$
1E,2E	$\emptyset$
1E,3E	$\emptyset$
1E,4E	$\emptyset$
1E,5E	$\emptyset$
2A,3A	$[p_2, q_2]$
2A,3B	$[p_2, q_2]$
2A,3C	$[p_2, q_2]$
2A,4A	$[q_2, q_2]$
2A,4B	$[q_2, q_2]$
2A,4C	$[q_2, q_2]$
2A,5A	$\emptyset$
2A,5B	$\emptyset$
2A,5C	$\emptyset$
2A,5D	$\emptyset$
2A,5E	$\emptyset$
2B,3C	$[p_1, q1;$
2B,3C	$p_2, q_2]$

Labels	*Intervals*
3B,4C	$[q_1, q_1;$
3B,4C	$q_2, q_2]$
3B,5C	$[q_1, q_1]$
3B,5D	$[q_1, q_1]$
3B,5E	$[q_1, q_1]$
3C,3C	$[q_1, p_1, <;$
3C,3C	$p_3 - t_s, q_2, <]$
3C,3C	$[q_1, p_1, =;$
3C,3C	$p_2, q_2, =]$
3C,3C	$[q_1, p_4 - t_s, >;$
3C,3C	$p_2, q_2, >]$
3C,3D	$[q_1, p_1;$
3C,3D	$q_2, q_2]$
3C,3E	$[q_1, p_1]$
3C,4C	$[p_1, p_4 - t_s;$
3C,4C	$q_2, q_2]$
3C,4D	$[q_1, p_1;$
3C,4D	$q_2, q_2]$
3C,4E	$[q_1, p_1]$
3C,5C	$[q_1, p_4 - t_s]$
3C,5D	$[q_1, p_4 - t_s]$
3C,5E	$[q_1, p_1, \leq;$
3C,5E	$q_1, p_4 - t_s, >];$
3D,3E	$[q_1, p_1]$
3D,4E	$[q_1, p_1]$
3D,5E	$[q_1, p_1]$
3E,3E	$[q_1, p_1]$
3E,4E	$[q_1, p_1]$
3E,5E	$[q_1, p_1]$
4A,5A	$\emptyset$
4A,5B	$\emptyset$
4A,5C	$\emptyset$
4A,5D	$\emptyset$

1A,5D	$\emptyset$
1A,5E	$\emptyset$
1B,1C	$[p_3 - t_s, q_2]$
1B,1D	$[q_2, q_2]$
1B,1E	$\emptyset$
1B,2C	$[p_3 - t_s, q_2]$
1B,2D	$[q_2, q_2]$
1B,2E	$\emptyset$
1B,3C	$[p_3 - t_s, q_2]$
1B,3D	$[q_2, q_2]$
1B,3E	$\emptyset$
1B,4E	$\emptyset$
1B,5E	$\emptyset$
1C,1C	$[p_3 - t_s, q_2]$
1C,1D	$[q_2, q_2]$
1C,1E	$\emptyset$
1C,2C	$[p_3 - t_s, q_2]$
1C,2D	$[q_2, q_2]$
1C,2E	$\emptyset$
1C,3C	$[p_3 - t_s, q_2]$
1C,3D	$[q_2, q_2]$
1C,3E	$\emptyset$
1C,4E	$\emptyset$

2B,4D	$[p_1, q_1;$
2B,4D	$q_2, q_2]$
2B,5E	$[p_1, q_1]$
2C,3C	$[p_1, q_1;$
2C,3C	$p_3 - t_s, q_2]$
2C,3D	$[p_1, q_1;$
2C,3D	$q_2, q_2]$
2C,3E	$[p_1, q_1]$
2C,4E	$[p_1, q_1]$
2C,5E	$[p_1, q_1]$
2D,3E	$[p_1, q_1]$
2D,4E	$[p_1, q_1]$
2D,5E	$[p_1, q_1]$
2E,3E	$[p_1, q_1]$
2E,4E	$[p_1, q_1]$
2E,5E	$[p_1, q_1]$
3A,3A	$[p_2, q_2]$
3A,3B	$[p_2, q_2]$
3A,3C	$[p_2, q_2]$
3A,4A	$[q_2, q_2]$
3A,4B	$[q_2, q_2]$
3A,4C	$[q_2, q_2]$
3A,5A	$\emptyset$
3A,5B	$\emptyset$
3A,5C	$\emptyset$
3B,3C	$[q_1, q_1;$
3B,3C	$p_2, q_2]$

4A,5E	$\emptyset$
4B,5C	$[q_1, q_1]$
4B,5D	$[q_1, q_1]$
4B,5E	$[q_1, q_1]$
4C,5C	$[q_1, p_4 - t_s]$
4C,5D	$[q_1, p_4 - t_s]$
4C,5E	$[q_1, p_4 - t_s]$
4D,5E	$[q_1, q_2]$
4E,5E	$[q_1, q_2]$
5A,5A	$\emptyset$
5A,5B	$\emptyset$
5A,5C	$\emptyset$
5A,5D	$\emptyset$
5A,5E	$\emptyset$
5B,5C	$[q_1, q_1]$
5B,5D	$[q_1, q_1]$
5B,5E	$[q_1, q_1]$
5C,5C	$[q_1, p_4 - t_s]$
5C,5D	$[q_1, p_4 - t_s]$
5C,5E	$[q_1, p_4 - t_s]$
5D,5E	$[q_1, q_2]$
5E,5E	$[q_1, q_2]$

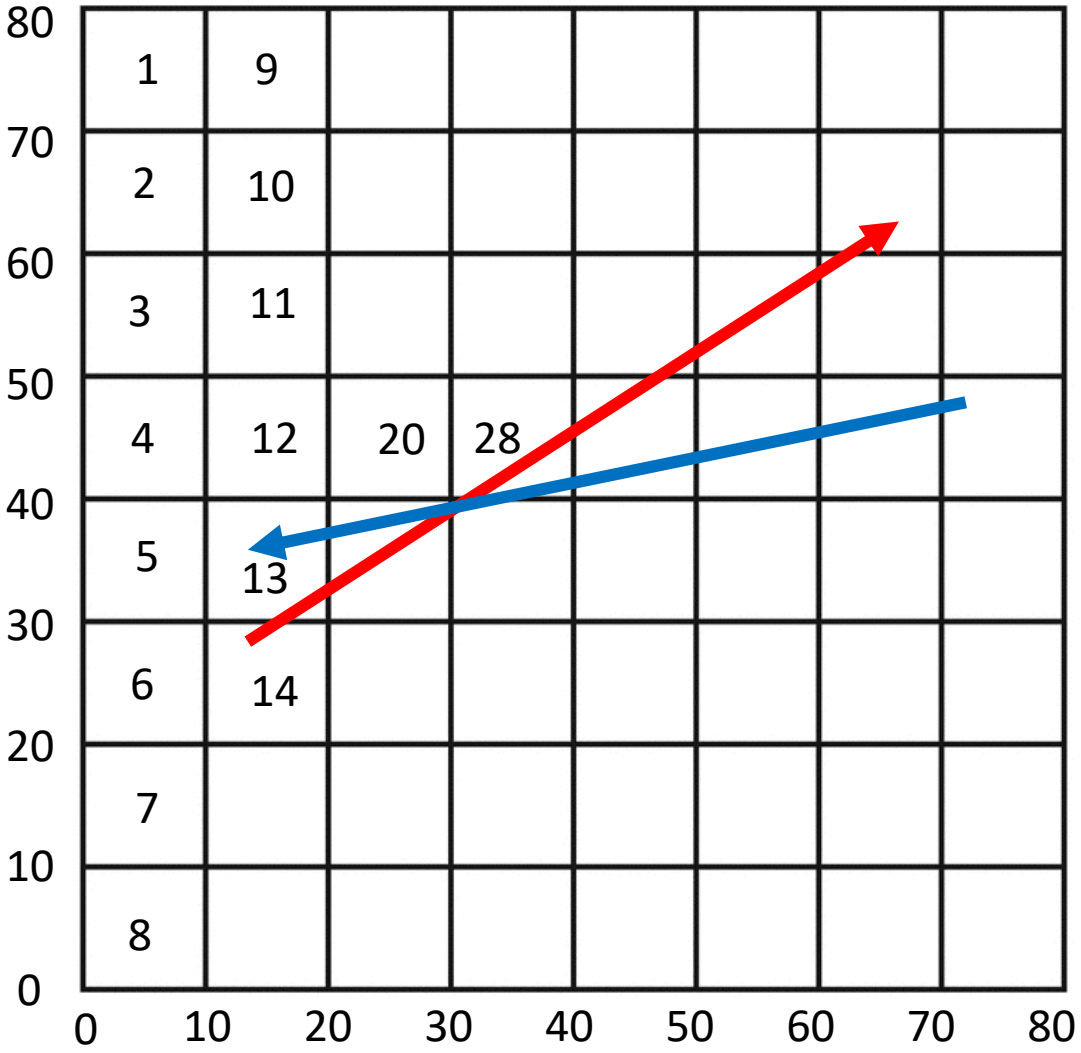
FAA-NASA Strategic Deconfliction

- Starts with set of flights requests (launch,land vertexes,time interval)
- .speed: Random speeds for flights, constant through all segments
- .start_time: launch time
- .traj: Trajectories: 3-element polyline (up, over, down)
  - 3x6 array (xi1,yi1,zi1,xi2,yi2,zi2)
- .flight_path: 3x4 array (entry_time, exit_time, speed, lane #)
- .d_count: deconfliction count (number of segments tested)
- .end_time: landing time for flight
- .grid_els: grid element indexes for flight (fly over cells)

Deconfliction Algorithm

- Produces list of flights with common grid elements
- Finds set of flights with segments that are within headway distance (called pinch points)
  - Mx5 array: (f1,f2,s1,s2,min_d)
- Algorithm continues until new flight does not fail headway distance at any time during flight
  - Checks every pinch segment pair as to whether two flights are that at times which produce a conflict

Deconfliction Algorithm



pt1 = [13,29,0]
pt2 = [66,62,0]
x_min = 0
y_min = 0
x_max = 80
y_max = 80
x_grid = 10
M = 80
N = 80

grid elements

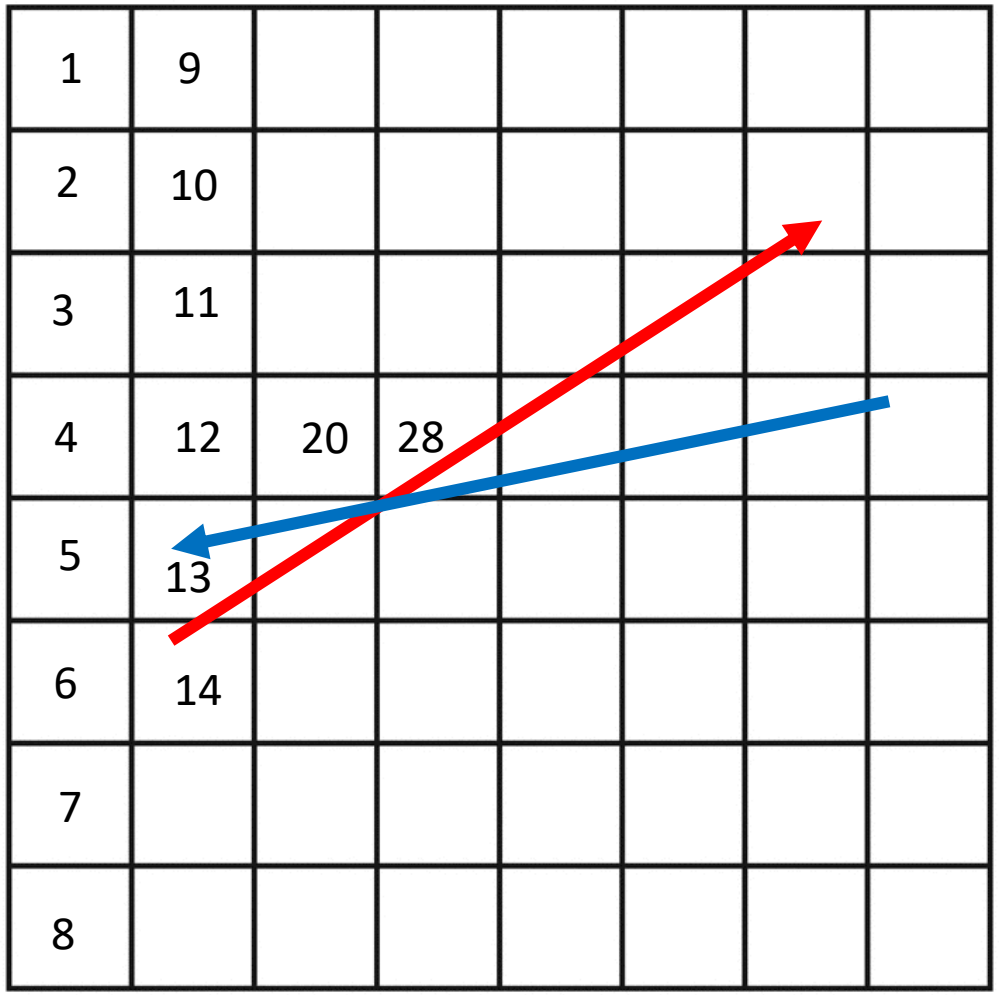
13
14
20
21
28
29
35
36
43
50
51

pt1 = [72,48,0]
pt2 = [13,37,0]
x_min = 0
y_min = 0
x_max = 80
y_max = 80
x_grid = 10
M = 80
N = 80

grid elements

13
20
21
28
29
36
44
52
60

Deconfliction Algorithm



These two segments may be at different heights, even though they share common grid elements.

The function UAM_quad(P0,P1,Q0,Q1,del_x) returns the closest points (within del_x) of the two segments, as well as their distance. E.g., suppose segment 1 is at z1 = 10.5, and z2 = 11; then:

```
>> [md,p1,p2] = UAM_quad(P0,P1,Q0,Q1,0.001)
md = 0.5001
p1 = 31.3380   40.4180   10.5000
p2 = 31.3490   40.4210   11.0000
```

Your Goals

- Basic Goal:
  - Get FNSD-LSD comparison running
  - Develop reasonable measures for comparison
  - Develop statistical framework to make measurements
  - Produce statistics
- Advanced Goal:
  - Develop improved FNSD method
  - Demonstrate gathering of measurements on a variety of scenarios

FNSD vs LSD Code

```
function [airways,lane_flights,flights,flights_FN] =   UAM_FNSD_LSD_scenario( min_start, max_start, min_speed, max_speed, num_flights, del_t, h_t)


% UAM_FNSD_LSD_scenario - compare FAA-NASA SD with Lane SD
% On input:
%     min_start (float): earliest start time
%     max_start (float): latest start time
%     min_speed (float): minimum speed (0.1 corresponds to 3mph)
%     max_speed (float): maximum speed (0.31 corresponds to 10 mph)
%     num_flights (int): number of flights to schedule
%     del_t (float): time step for simulated motion
%     h_t (float): minimum headway time
```

```
% On output:
%    airways (airway data structure): airways info
%      .vertexes (nx2 array): x,y locations of road intersections
%      .edges (ex2 array): edges on roads (i.e., between intersections)
%      .r_len (float): minimum lane length in roundabout
%      .launch_vertexes (1xk vector): indexes of launch vertexes (ground)
%      .land_vertexes (1xm vector): indexes of land vertexes (ground)
%      .vertexes3D (px3 array): 3D lane vertexes
%      .lanes (qx10 array): x1,y1,z1,x2,y2,z2,v1_g,v2_g,v1_3D,v2_3D
%      .lane_lengths (qx1 vector): lengths of lanes
%    lane_flights (lane flight data structure): lane-based flight data
%      .flights (kx5 vector): time in,time out,speed,lane,ID
%    flights (flight struct) per flight info
%      (k).start_time (float): start time of flight    % (k).end_time (float): end time of flight
% (k).lanes    % (k).speed)
%    flights_FN (flight struct): FAA-NASA flight data
%      (k).start_time (float): start time of flight      %  (k).end_time (float): end time of flight
%      (k).lanes     % (k).speed)
```

- LOWER_ALTITUDE = 10;
- UPPER_ALTITUDE = 12;
- M = 100;
- N = 100;
-
- grid_x = 10;
-
- h_x = max_speed*h_t;
-
- airways = UAM_create_airway_demo;
- num_ground_vertexes = length(airways.vertexes(:,1));

```
lane_flights = [];   flights = [];
for f = 1:num_flights
  % launch_index = randi(num_ground_vertexes);   land_index = randi(num_ground_vertexes);
   launch_index = 1;  land_index = 25;
   speed = min_speed+rand*(max_speed-min_speed);
   lanes = UAM_flight_path(airways,launch_index,land_index);
   [flight_path,d_count,lane_flights] = UAM_reserve_flight(airways,...
      lane_flights,min_start,max_start,speed,lanes,f,h_t);
   if ~isempty(flight_path)
      flights(f).lanes = lanes;                flights(f).flight_path = flight_path;
      flights(f).d_count = d_count;        flights(f).start_time = flight_path(1,1);
      flights(f).end_time = flight_path(end,2);
   else
      flights(f).lanes = lanes;               flights(f).flight_path = flight_path;
      flights(f).d_count = d_count;       flights(f).start_time = -1;
      flights(f).end_time = -1;
   end
end
```

lane_flights: flights by lane

flights: individual flight data

Create flights

Create 3-polyline trajectory

```
for f = 1:num_flights
    if flights(f).start_time>=0
        flights_FN(f).speed = flights(f).flight_path(1,3);
        flights_FN(f).start_time = flights(f).start_time;
        launch_lane = flights(f).flight_path(1,4);
        land_lane = flights(f).flight_path(end,4);
        launch_pt = lanes(launch_lane,1:3);
        land_pt = lanes(land_lane,4:6);
        altitude = LOWER_ALTITUDE + rand*(UPPER_ALTITUDE-
LOWER_ALTITUDE);
        pt1 = launch_pt;     pt4 = land_pt;     pt2 = [pt1(1:2),altitude];
        pt3 = [pt4(1:2),altitude];     traj = [pt1,pt2; pt2,pt3; pt3,pt4];
        flights_FN(f).traj = traj;
```

```
len1 = norm(traj(1,4:6)-traj(1,1:3));
    len2 = norm(traj(2,4:6)-traj(2,1:3));
    len3 = norm(traj(3,4:6)-traj(3,1:3));
    total_len = len1 + len2 + len3;
    flight_path = zeros(3,4);
    flight_path(1,1) = flights_FN(f).start_time;
    flight_path(1,2) = flight_path(1,1) + len1/flights_FN(f).speed;
    flight_path(1,3) = flights_FN(f).speed;
    flight_path(1,4) = 1;
    flight_path(2,1) = flight_path(1,2);
    flight_path(2,2) = flight_path(2,1) + len2/flights_FN(f).speed;
    flight_path(2,3) = flights_FN(f).speed;
    flight_path(2,4) = 2;
    flight_path(3,1) = flight_path(2,2);
    flight_path(3,2) = flight_path(3,1) + len3/flights_FN(f).speed;
    flight_path(3,3) = flights_FN(f).speed;
    flight_path(3,4) = 3;
```

A

Set flight path & grid elements

A

```
flights_FN(f).flight_path = flight_path;
    flights_FN(f).d_count = 0;
    flights_FN(f).end_time = flights(f).start_time...
      + total_len/flights_FN(f).speed;
    flights_FN(f).grid_els = UAM_grid_els(traj(1,1:3),traj(end,4:6),...
      x_min,y_min,x_max,y_max,grid_x,M,N);
```