AMSC/CMSC 664 Final Presentation

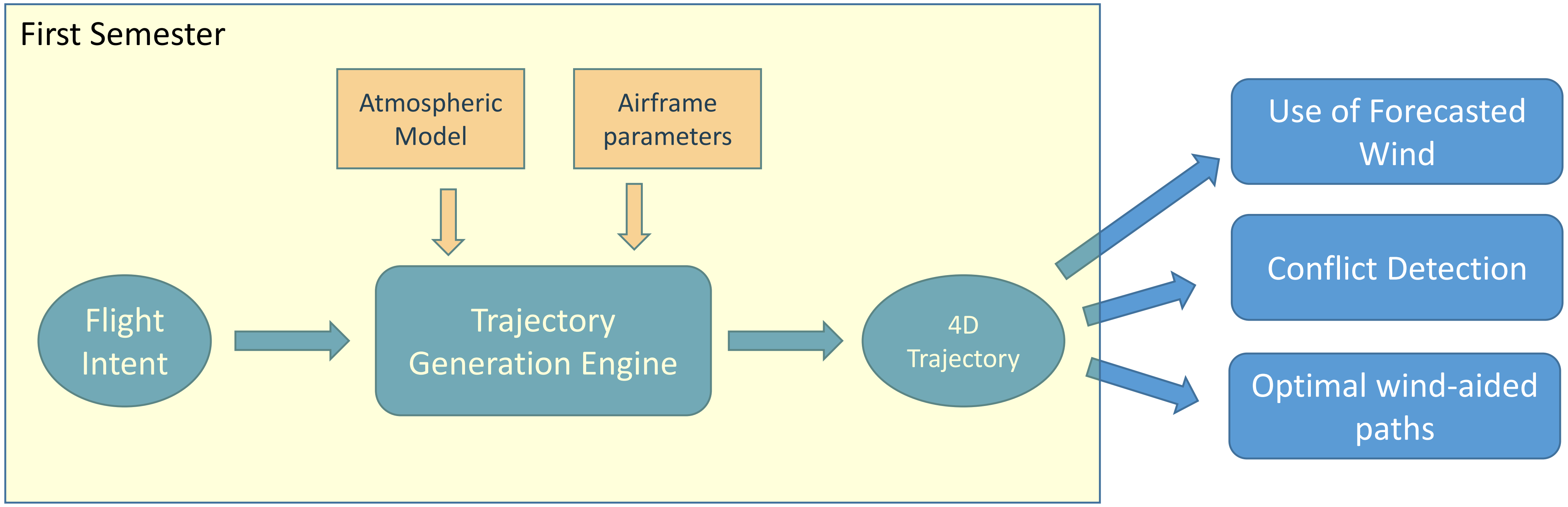
May 9, 2017

Jon Dehn

Advisor: Dr. Sergio Torres, Leidos Corporation

Project Goal

- Build a framework for testing compute-intensive algorithms in air traffic management



Project Parts

1. Build trajectory generation engine (Python, on Windows desktop PC, 4 core 3.4 GHz Intel I7-6700 processors)

2. Analyze use of Forecasted wind and quantify wind gradient characteristics (Python, on desktop)

3. Create algorithm for finding optimal wind-aided paths (using particle swarm optimization techniques) (Python, on desktop)

4. Create faster implementation of conflict detection algorithms by parallelizing algorithm (C, on Nvidia 980 GPU, 1.1 Ghz cores)

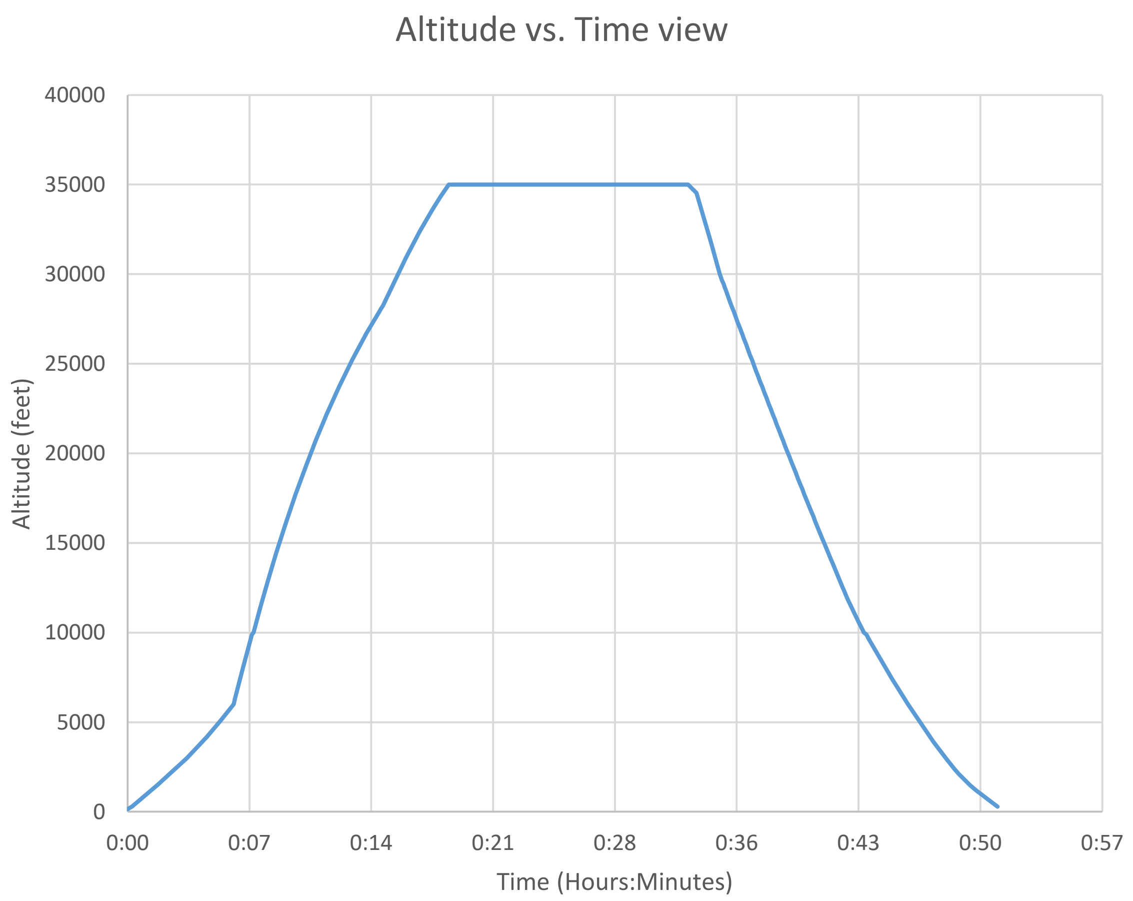
1. Trajectory Generation Engine

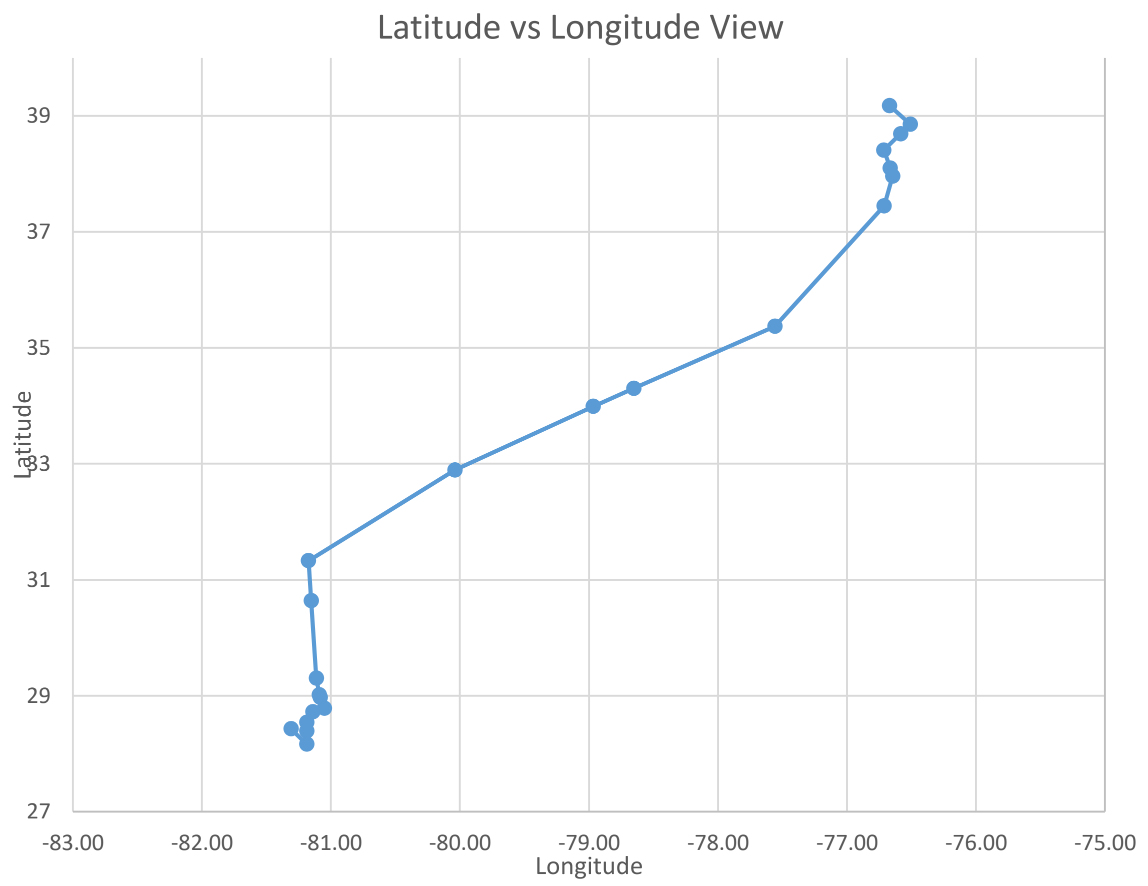
- Completed in first semester; coded in Python
- Solved this equation for climb/descent segments of trajectory:

$$\frac{dH_p}{dt} = \frac{T - \Delta T}{T} \frac{(Thr - D)V_{TAS}}{mg} f(M)$$

- Thr (thrust) supplied by aircraft engines
- D (Drag) from movement through atmosphere
- V_{TAS} - Velocity in True Airspeed; that is, relative to the air mass around the aircraft, which may be moving
- m (mass) of the aircraft, including passengers and fuel, decreases over time
- g – gravitational acceleration
- h – geodetic altitude
- $\frac{d}{dt}$ – time derivative

Typical 4-Dimensional Trajectory





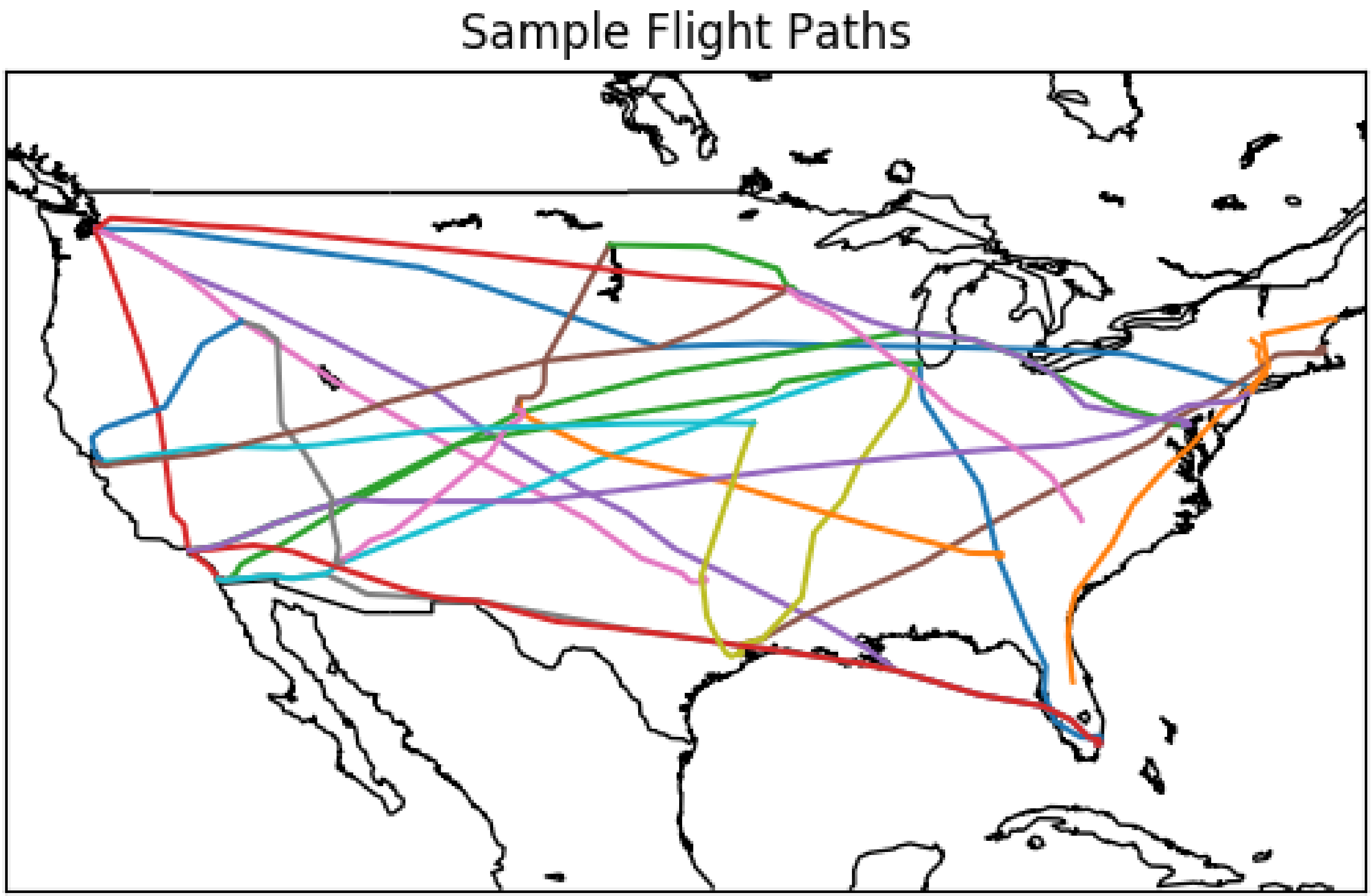
2. Use of Forecasted Wind/Temperature Data

- Current Operational Systems use only the current weather (wind speeds, temperature) information to build a trajectory
- Different systems have different time horizons; accuracy of longer flights in systems with longer time horizons may be improved by using NOAA weather forecasts
- Weather information is supplied in hourly forecast sets; weather at intermediate points is interpolated from surrounding hourly sets

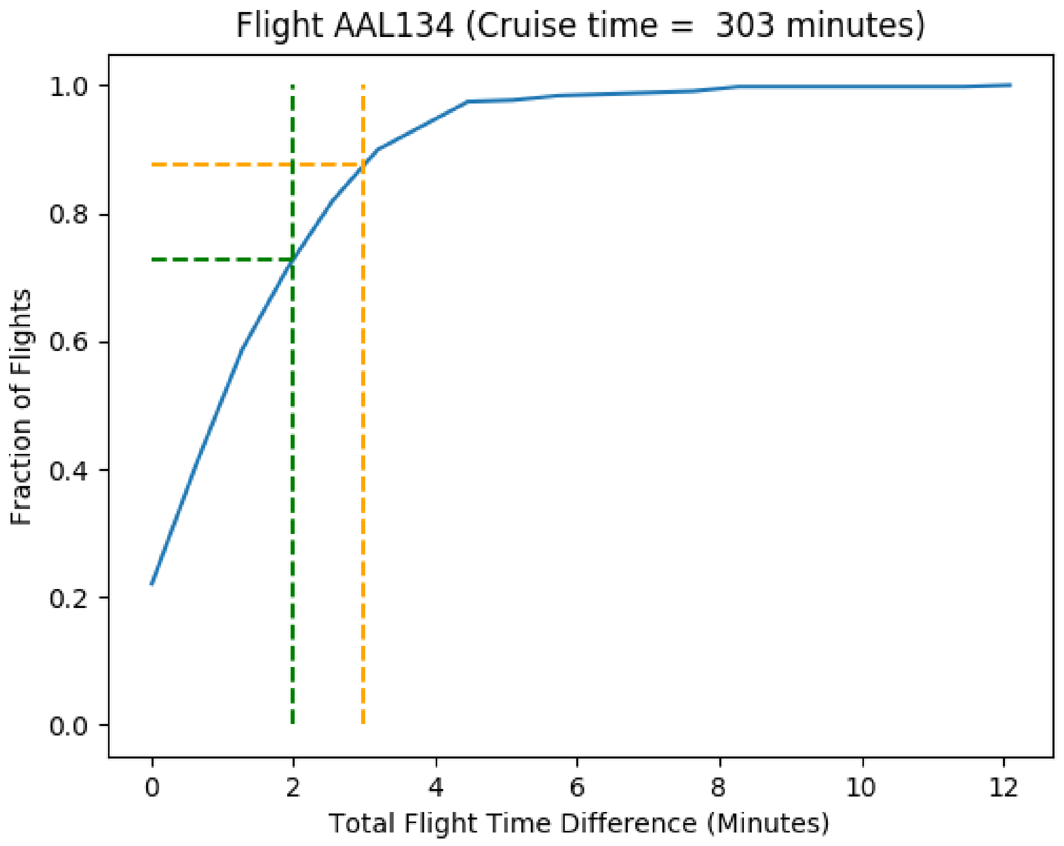
Forecast Weather Experiment:

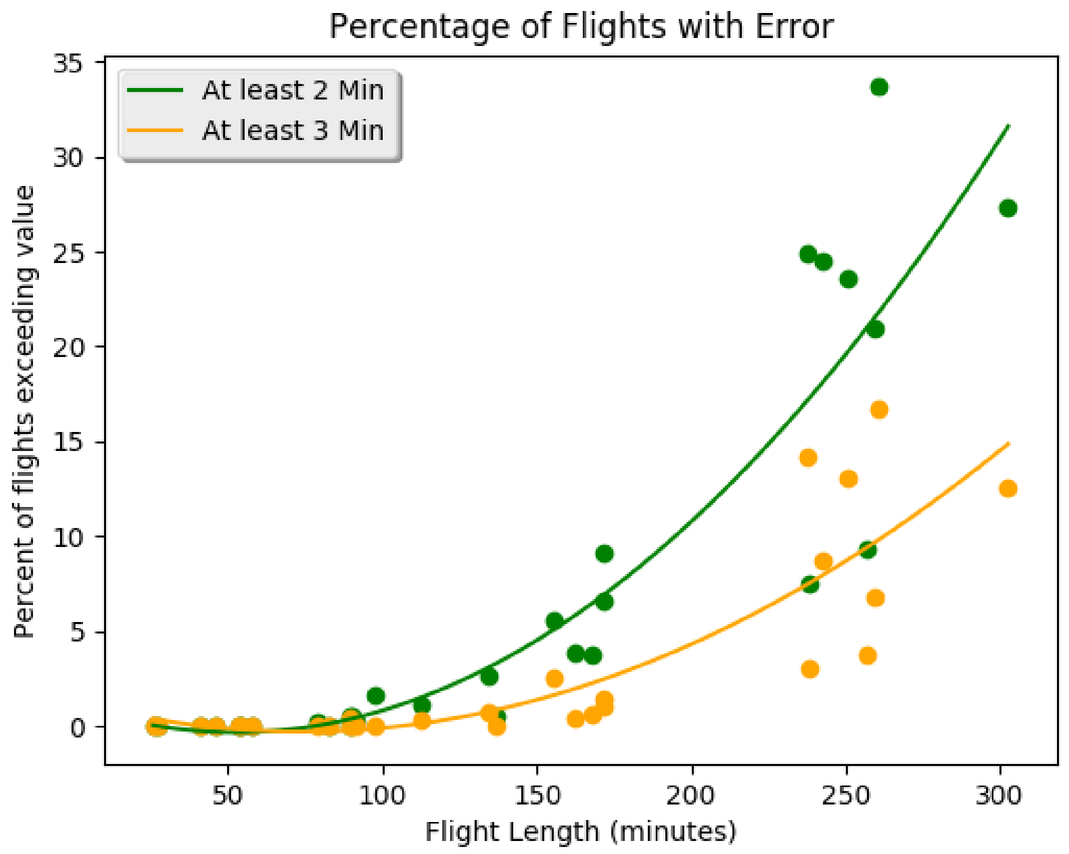
- Select flight paths of varying duration, covering east/west and north/south flight paths
- Determine time duration difference between using just current weather conditions vs. using forecasted weather at appropriate time
- Compare time difference to intended use of the data to see if difference is significant

Expanded set of flight paths since mid-term:

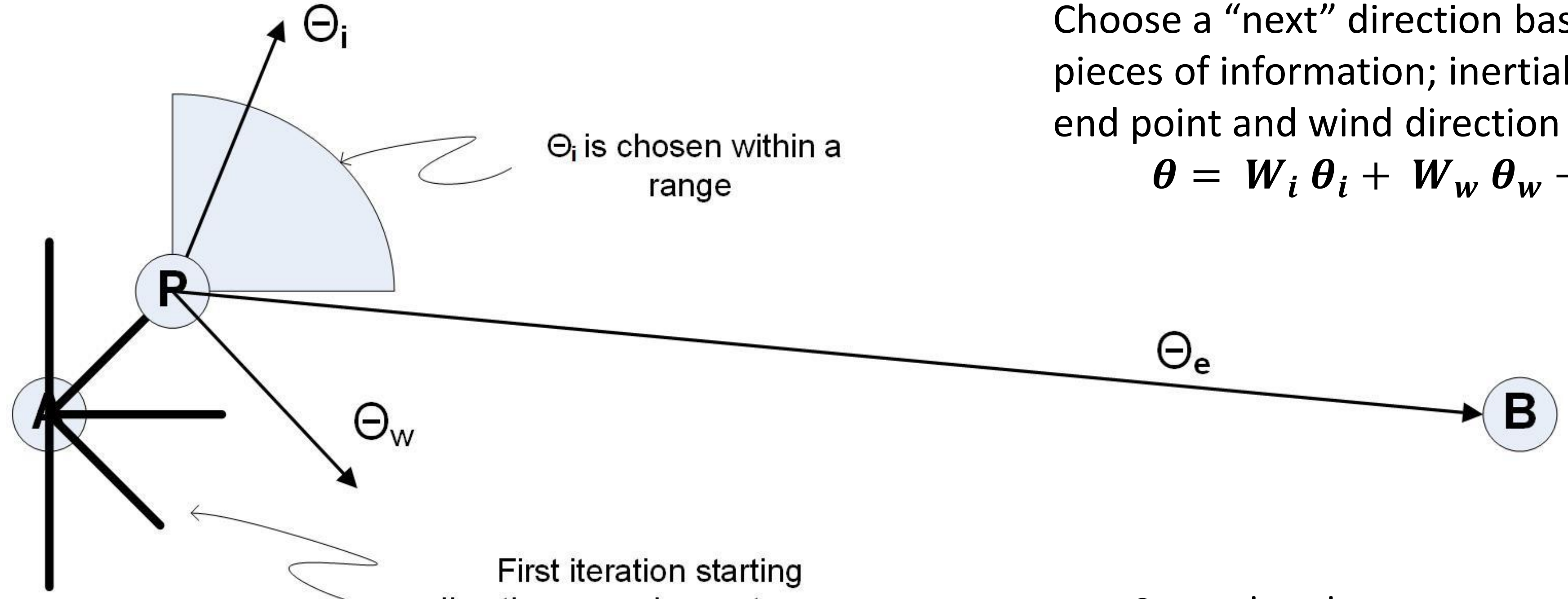


Results – 420 samples of each flight





3. Wind Aided Trajectory – Original plan


Θ_i
Θ_i is chosen within a range
R
A
Θ_w
Θ_e
B
First iteration starting directions are chosen to cover the possible solution space

Choose a “next” direction based on three pieces of information; inertial, direction to end point and wind direction

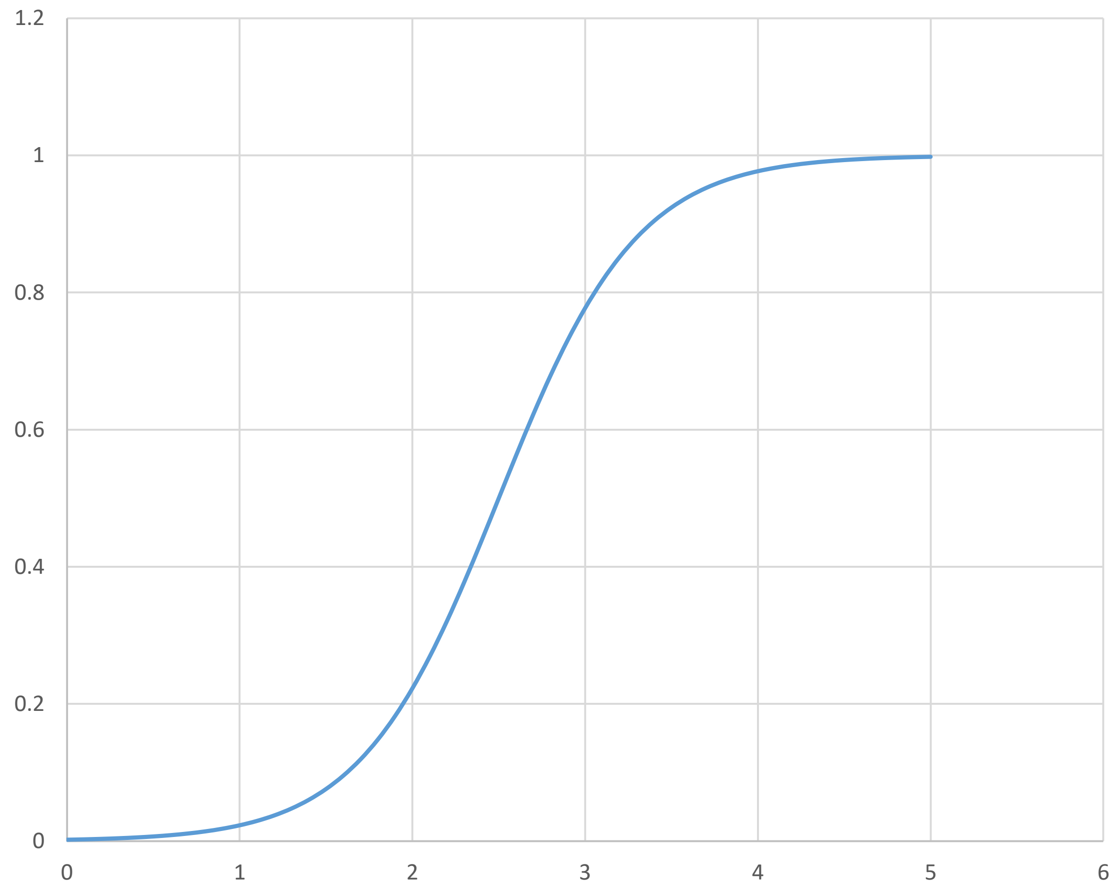
$$\boldsymbol{\theta} = \boldsymbol{W_i\,\theta_i} + \boldsymbol{W_w\,\theta_w} + \boldsymbol{W_e\theta_e}$$

Several paths are constructed, path with minimum cost (fuel burn) is selected, a new set of paths is constructed with starting directions around that previous best case path

Problems with initial approach

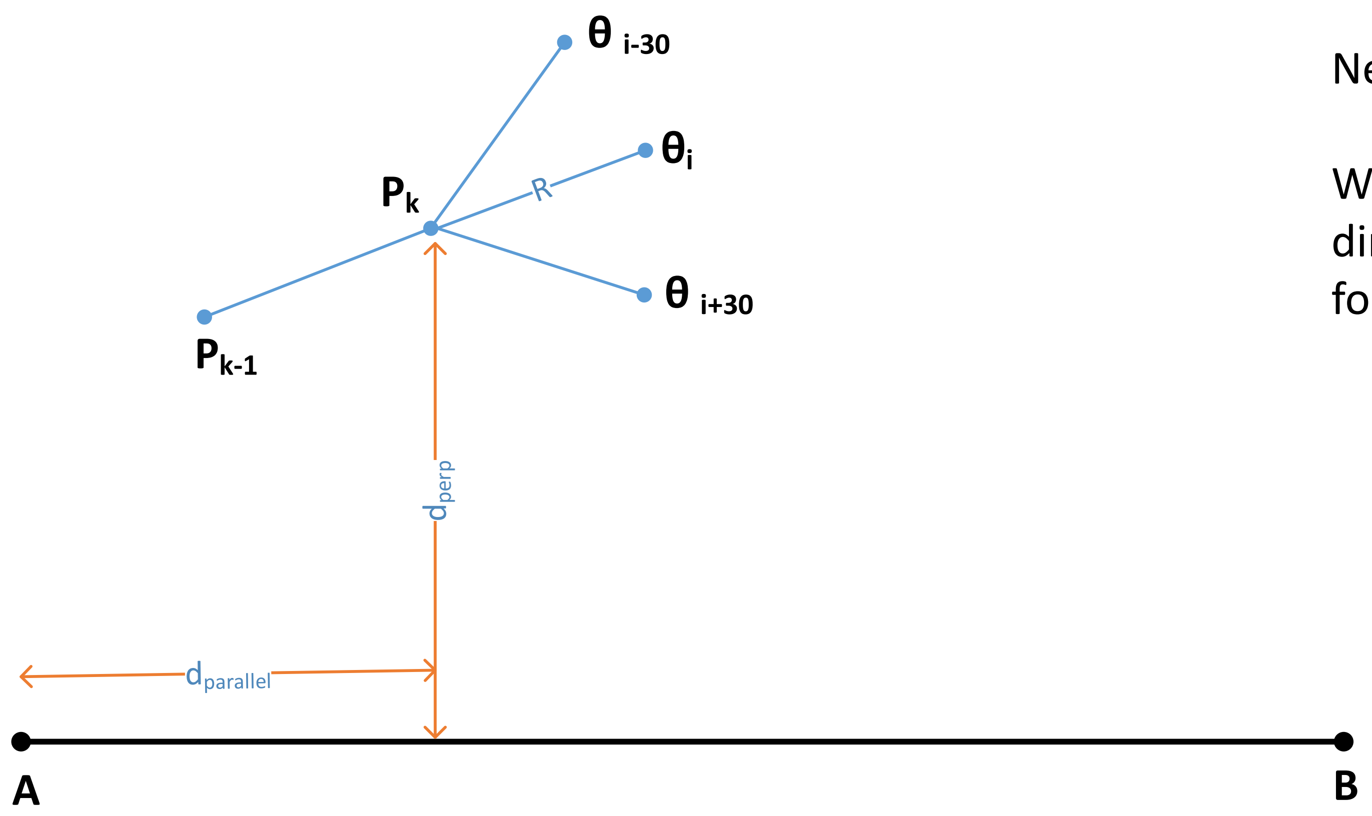
- Head wind situation – first approach zero-ed out wind weight when flying into head wind. This ignores some valuable information, as forward ground speed can still be optimized
- Magnitude of wind – first approach did not factor magnitude of the wind, just the direction
- Large perpendicular deviations – first approach did not penalize paths that were far off the direct A-B path
- Smooth Behavior – first approach had some step functions (such as zeroing wind for head winds) that gives discontinuous results. Sigmoid functions are used instead of step functions to give smooth results

Sigmoid Function - Review



$$Sig(u) = \frac{1}{1 + \exp\left(\frac{midpoint - u}{slope}\right)}$$

Revised Implementation



Next direction now defined as

$$\boldsymbol{\Theta = W_i\, \Theta_f + W_e\, \Theta_e}$$

Where Θ_f is defined from inertial direction and wind fields (see following slide)

$$\boldsymbol{\theta = W_i\,\theta_f + W_e\theta_e}$$

$$W_e = q^2 + s * Sig(q)$$

- $q = \frac{d_{parallel}}{d_{AB}}$
- $s = \frac{d_{perp}}{D}$, where D = constant 20 nautical miles
- Normalize: $W_e = \frac{W_e}{1+s}$

$$\theta_f = \theta_i + (+/-30) * Sig(u)$$

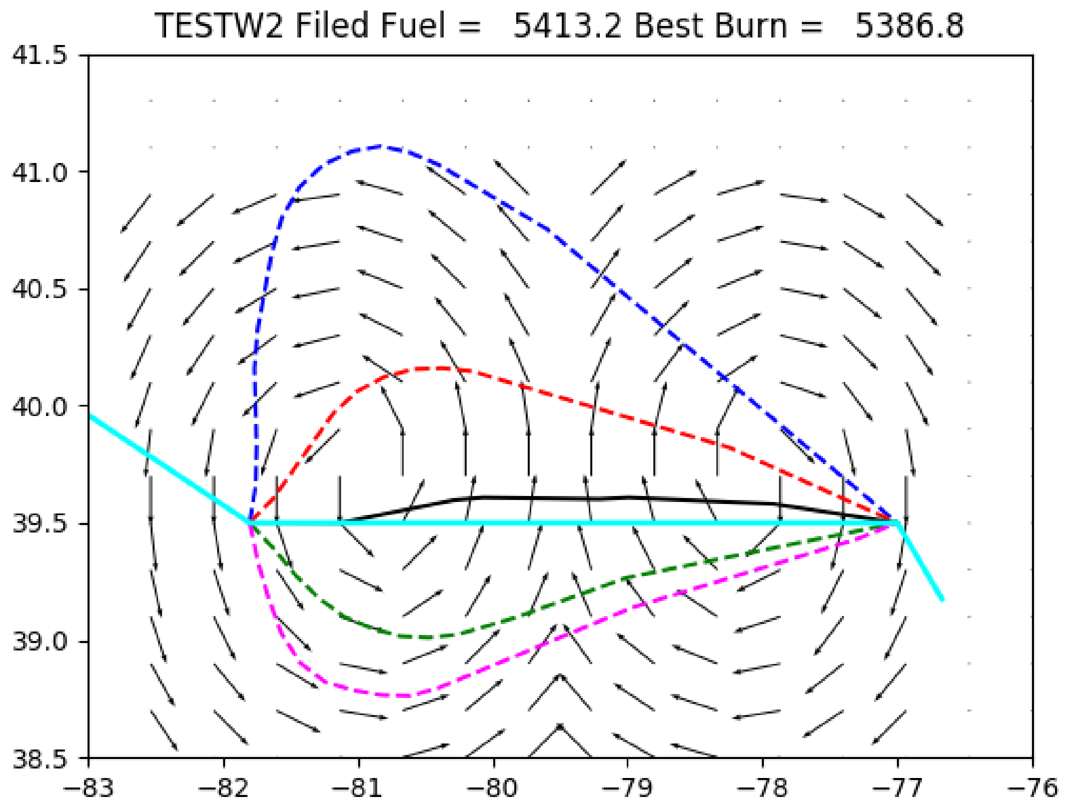
- $u = \frac{Wind\ Speed\ Along\ Path}{V_{TAS}}$

$$1 = W_i + W_e$$

Overview description

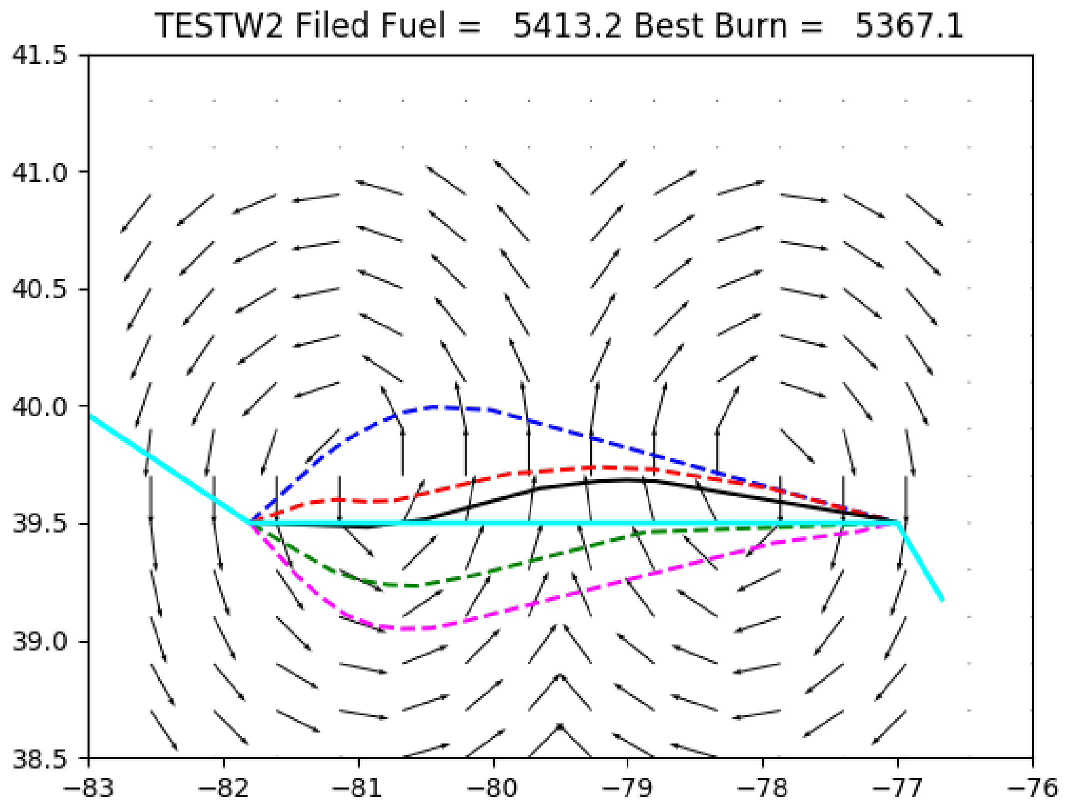
- First iteration computes 5 paths, each with a different starting course (within +/- 40º of A-B), then computes the best path based on fuel burn
- Each path consists of fixed length segments (30 NMI in this case)
- Subsequent iterations choose 5 paths, centered on the previous best path, with a smaller fanout (within +/- 20º of previous best)
- Algorithm stops when some number (3) iterations have not improved on best fuel burn

Sample Results – iteration 1



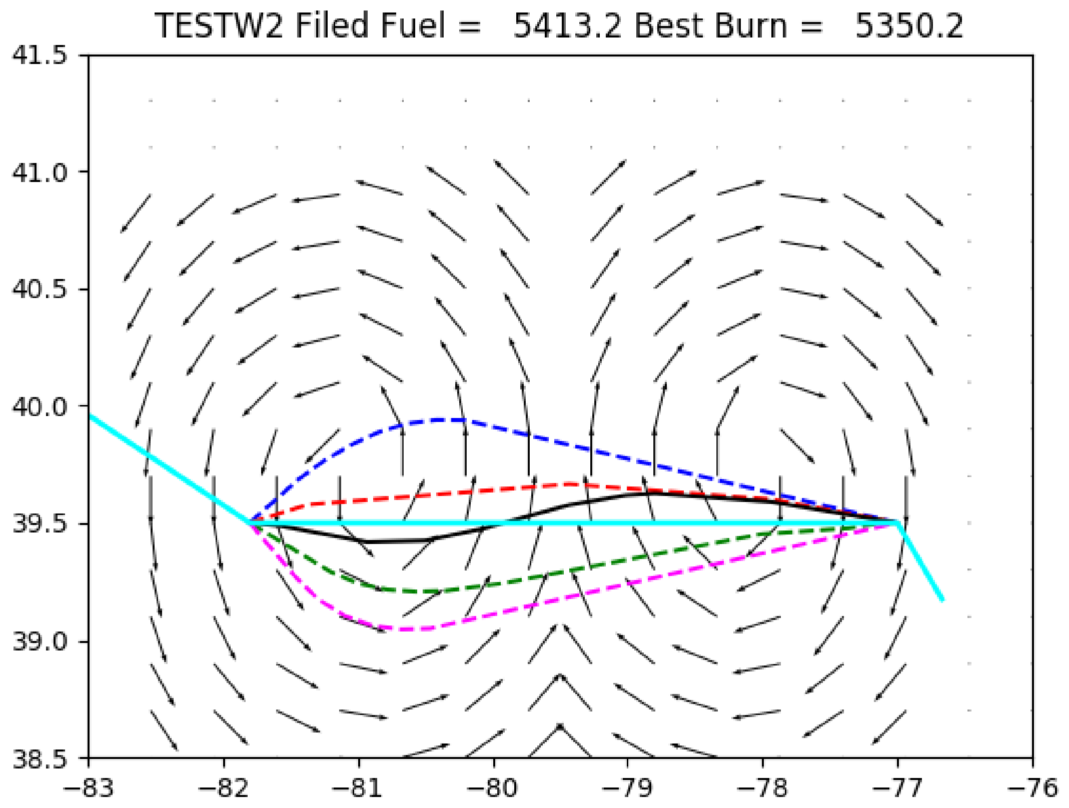
Black solid line is best path from iteration 1, 26.4KG of fuel saved over A-B path

Sample Results – iteration 2



Black solid line is best path from iteration 2. 10.7 KG less fuel than iteration 1

Sample Results – iteration 3 (1.6 minute better)



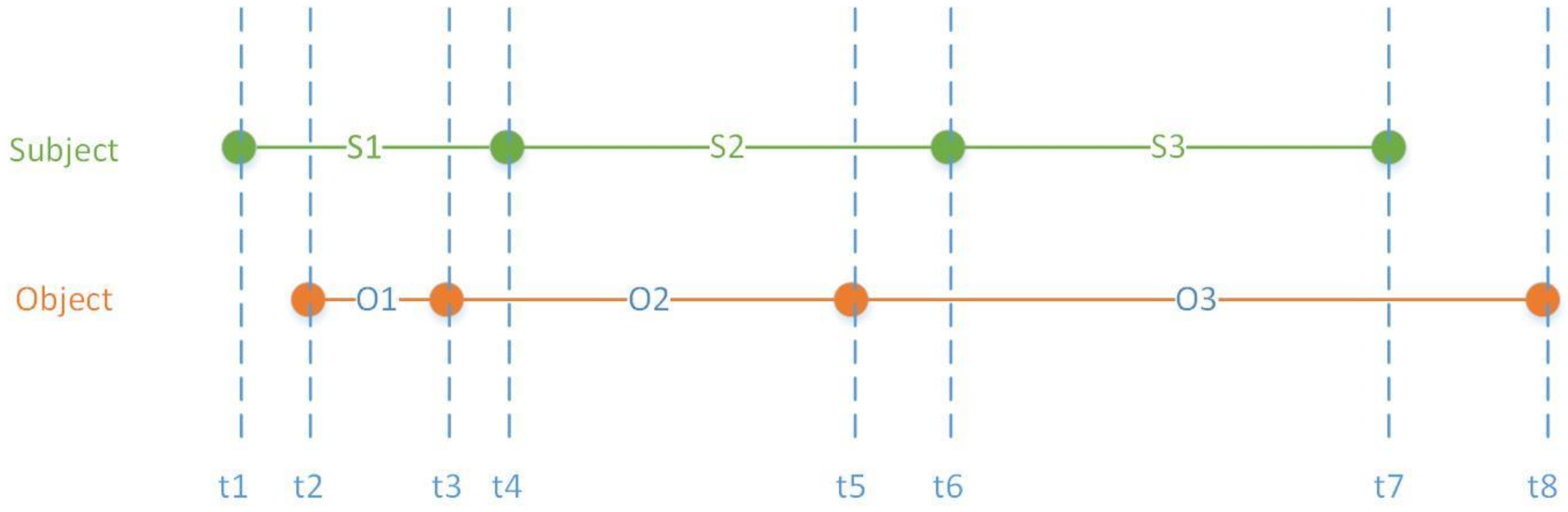
Black solid line is best path from iteration 3. 16.9 KG less fuel than iteration 2, 63.0 KG better than A-B path

No further improvement with subsequent iterations

4. Conflict Detection

- Each flight’s trajectory is composed of several (N) segments (as built by trajectory generation engine)
- Aircraft-to-aircraft conflict detection checks one flight’s segments vs. another flight’s segments (N*N compares). For clarity of presentation, call this “B” subject segments and “C” object segments
- A full conflict detection scheme compares any changed/new flight (the *subject*) to all existing (A) flights (the *objects*) (A*B*C compares)
- In real time, a system must process this at the rate of changed/new flights (in US busy systems, approximately 7/second)

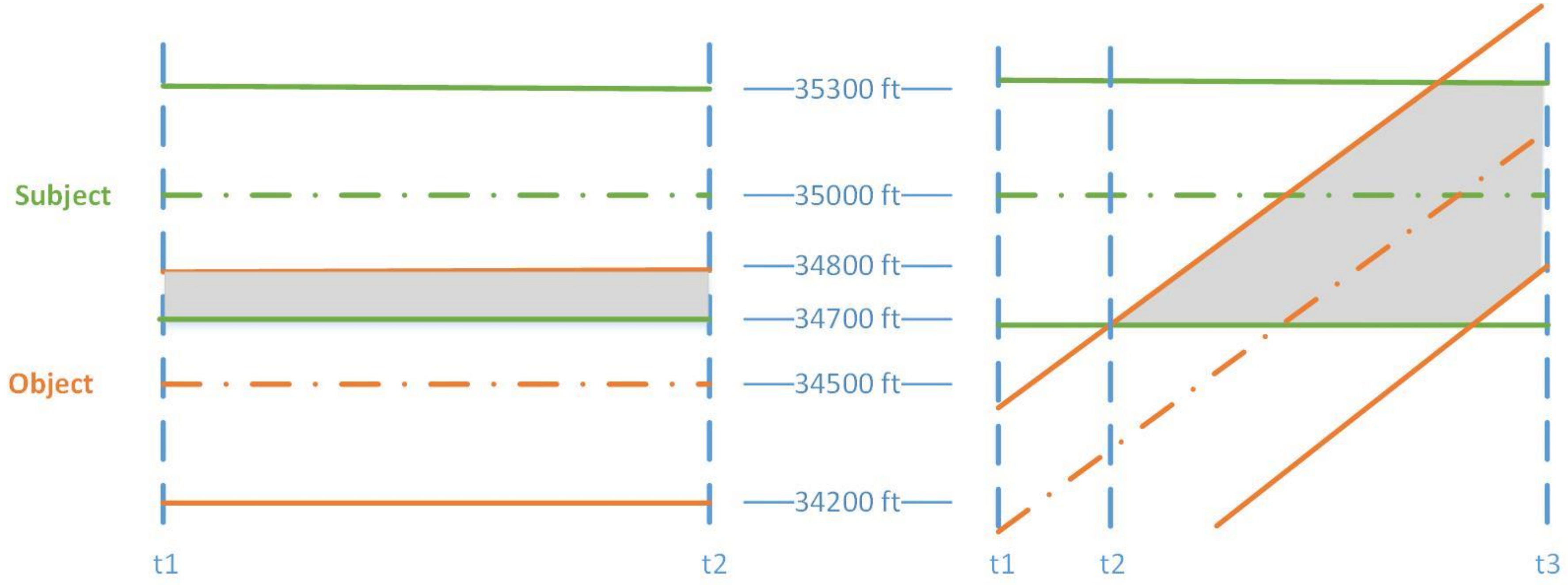
Conflict Detection – Time Filter



S1 overlaps O1 from t2 to t3
S1 overlaps O2 from t3 to t4
S2 overlaps O2 from t4 to t5

S2 overlaps O3 from t5 to t6
S3 overlaps O3 from t6 to t7

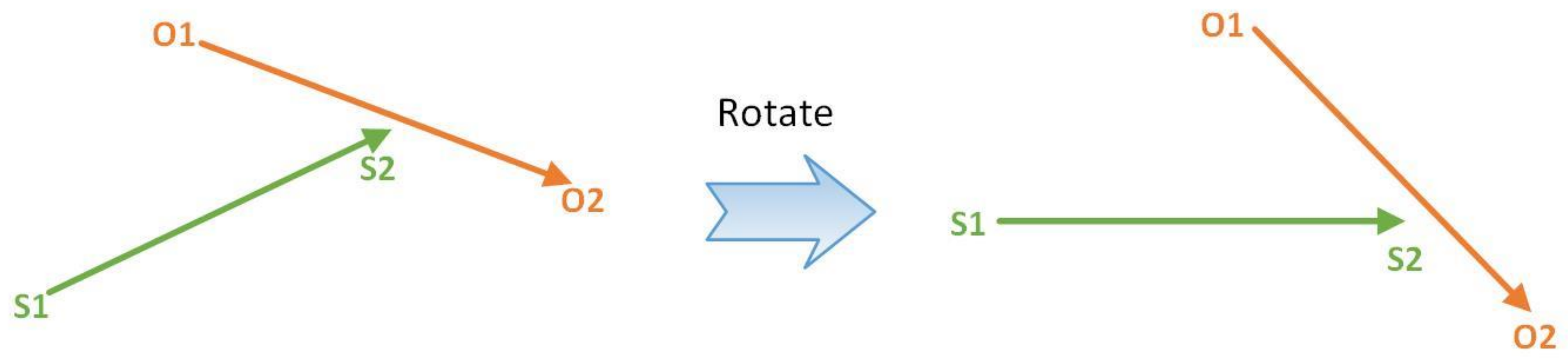
Conflict Detection – Altitude Filter



Subject and Object aircraft, with separation distance of 600 feet, are in conflict for entire interval

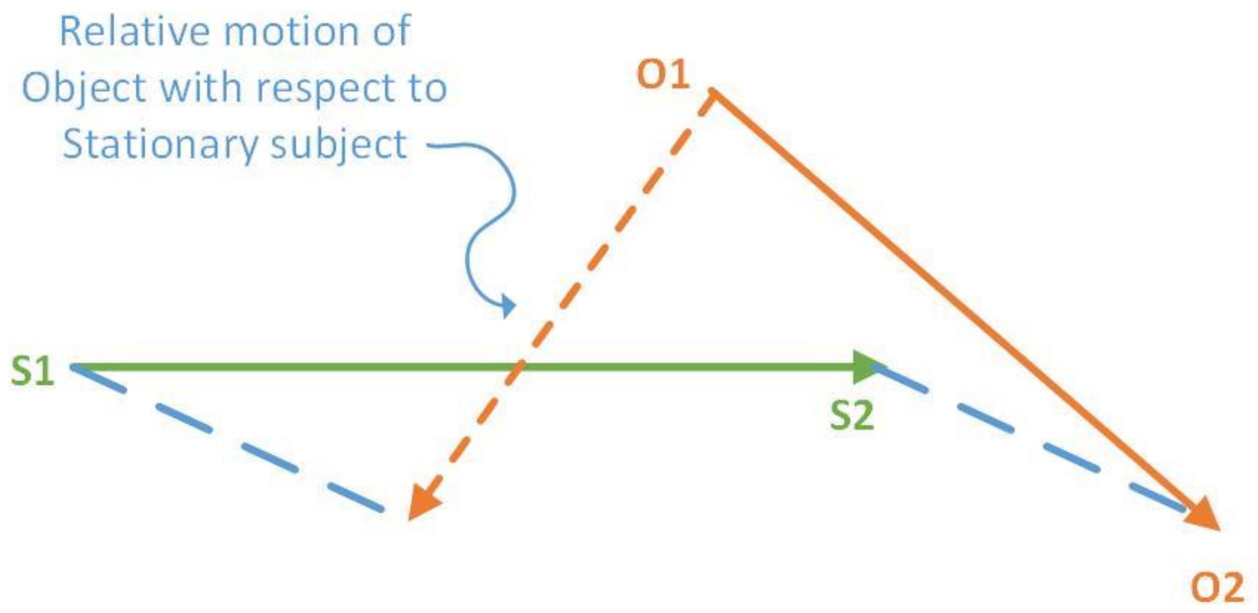
Object aircraft is climbing; becomes in conflict at time t2 until end of interval (t3)

Conflict Detection – Horizontal Filter 1



Segment endpoints are translated from geodetic coordinates to a stereographic plane with a tangent point at S1.
Rotate both segments so that S1 -> S2 is along the X axis; S1 is placed at the origin of the cartesian plane

Conflict Detection – Horizontal Filter 2

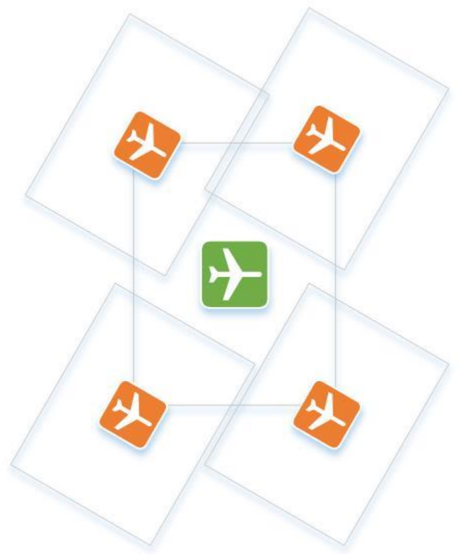

Relative motion of Object with respect to Stationary subject
O1
S1
S2
O2

The problem is approached from the view of a stationary subject aircraft. Motion of object aircraft relative to that stationary subject is calculated

Conflict Detection – Horizontal Filter 3

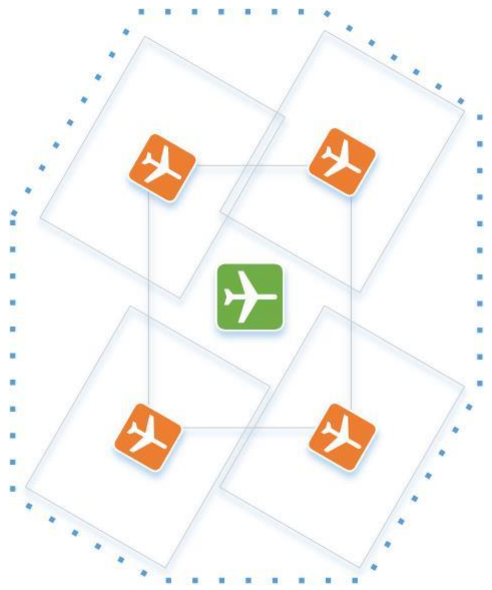
There is some uncertainty in the location of the subject and object, both in the forward and sideways direction. This is represented as a rectangle around the predicted position.

To account for uncertainty of both subject and object, a rectangle is drawn around the subject, and four rectangles, centered at the four corners of that rectangle, are drawn



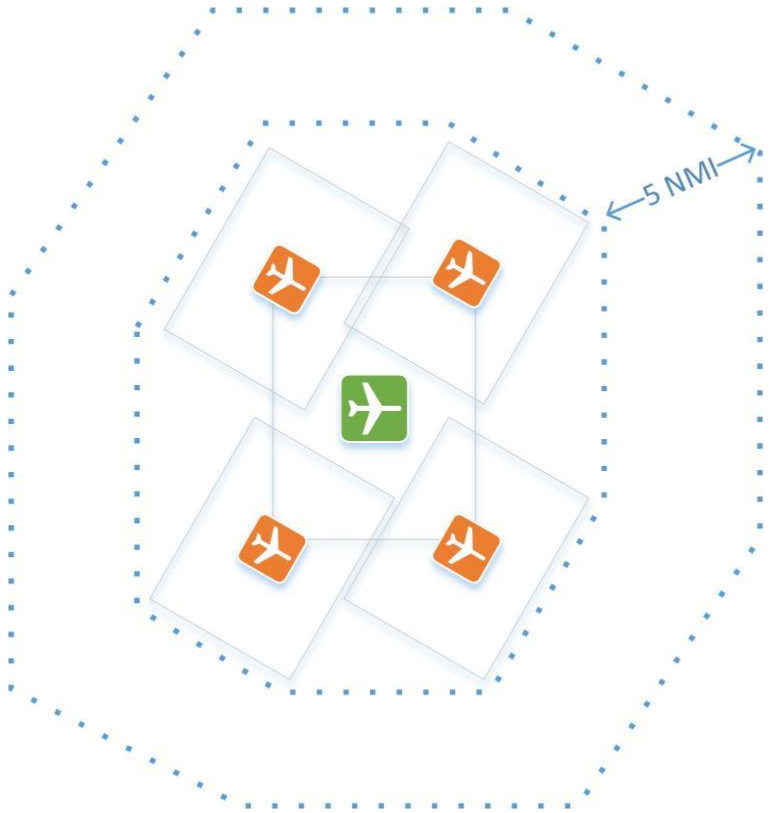
Conflict Detection- Horizontal Filter 4

A bounding octagon is drawn around the outside corners of these object rectangles



Conflict Detection – Horizontal Filter 5

To this, the separation standard of 5 NMI is added, giving an expanded octagon

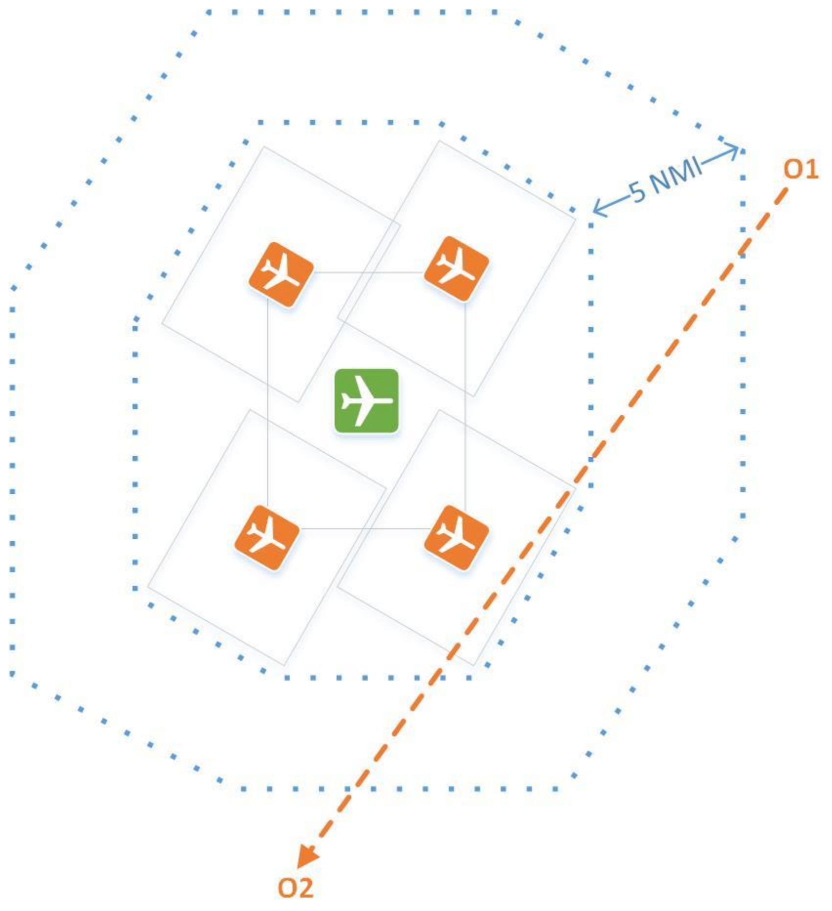


Conflict Detection – Horizontal Filter 6

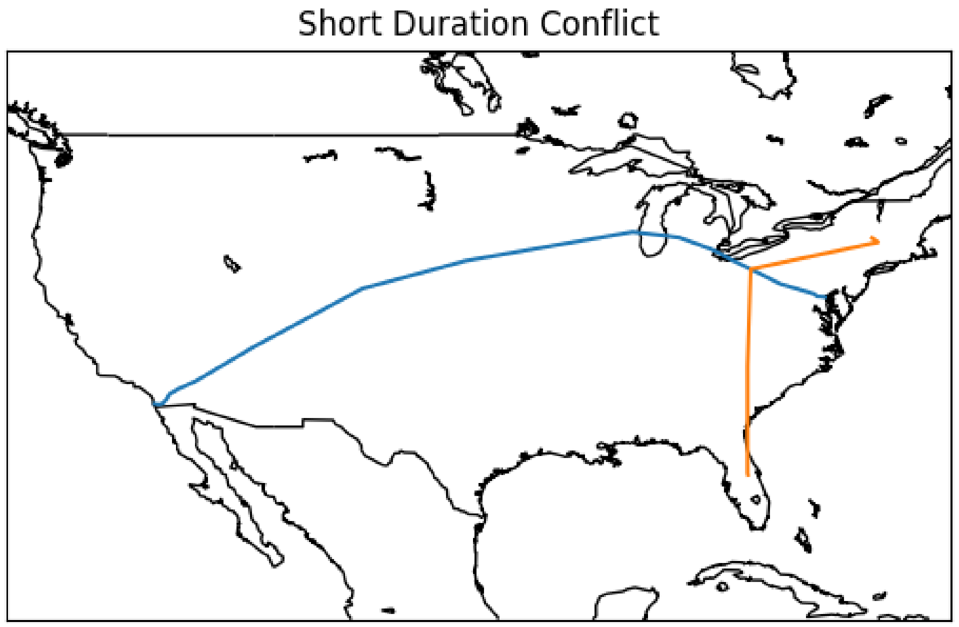
Finally, the object segment, relative to the stationary subject, is checked for intersection with this larger octagon.

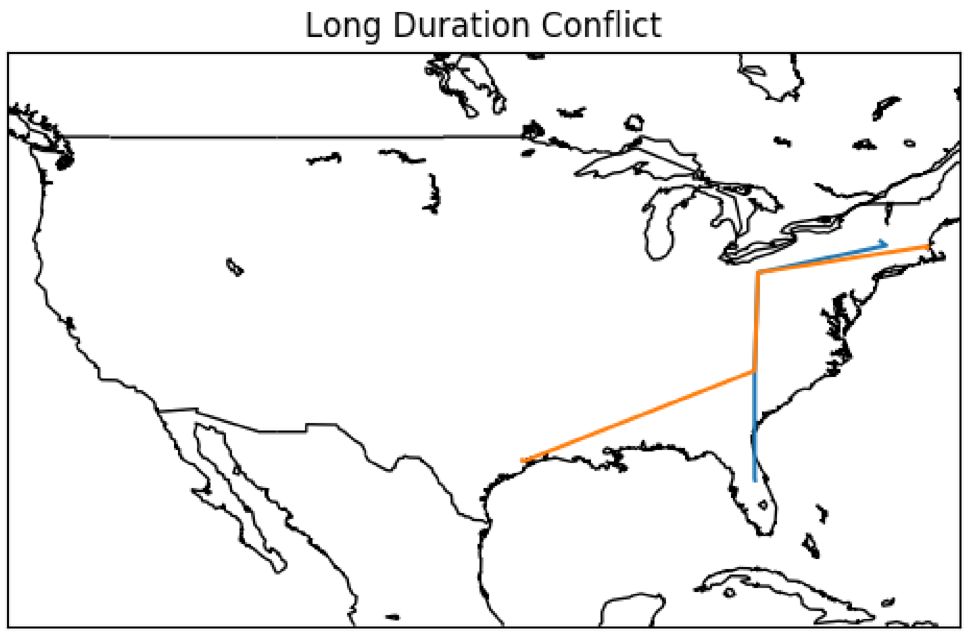
If there is an intersection, there is a conflict.

The start and end times of the conflict can be determined given the relative velocity of object-to-subject.

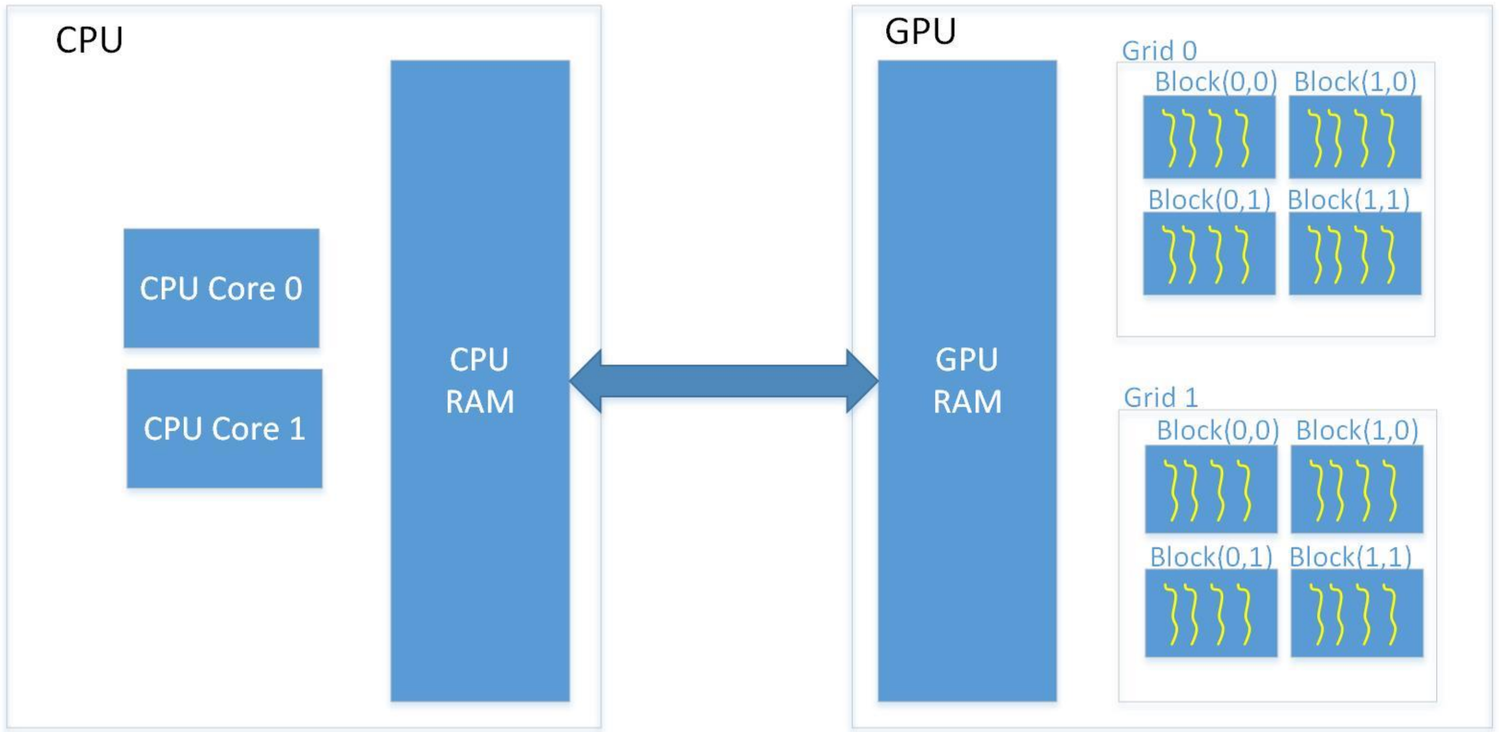


Sample flights (with conflicts):





CPU/GPU architecture



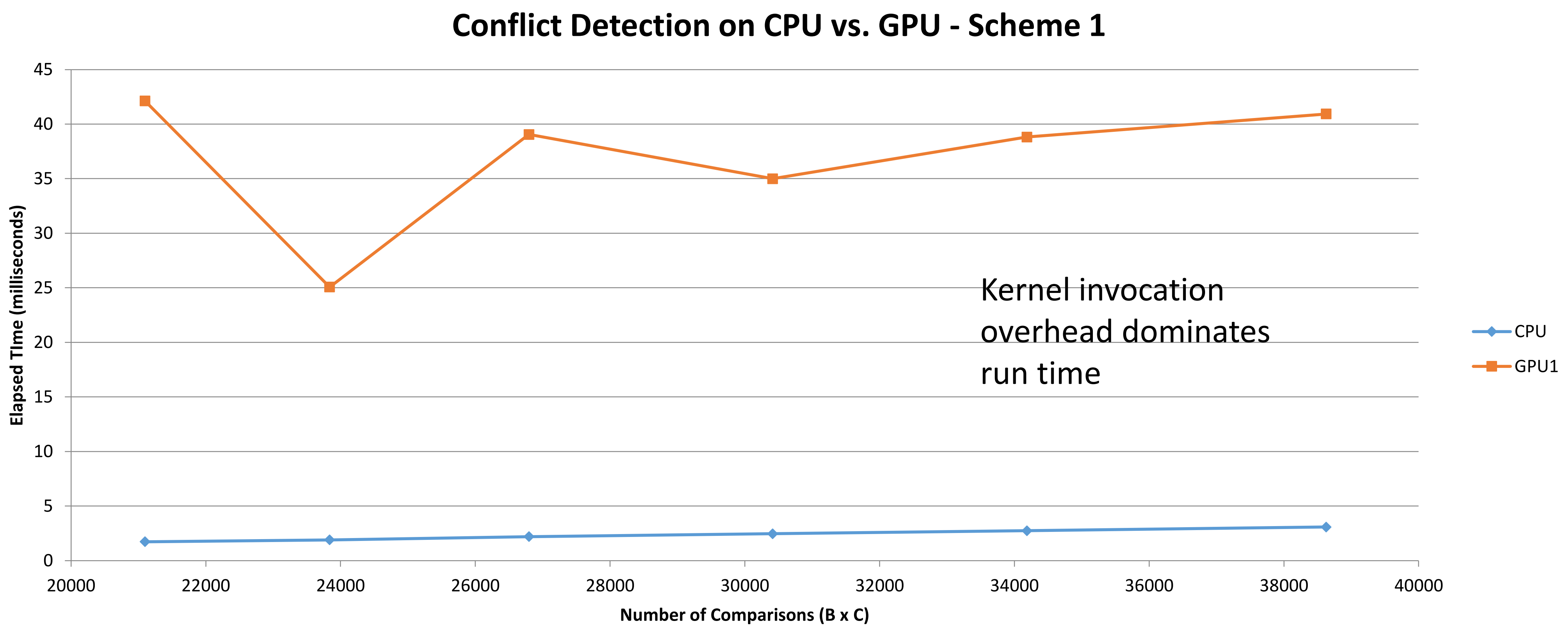
Programs running on CPU copy memory back and forth, and send "kernels" to the GPU to be run.

GPU has potentially many grids, running many blocks of threads, all running on GPU cores

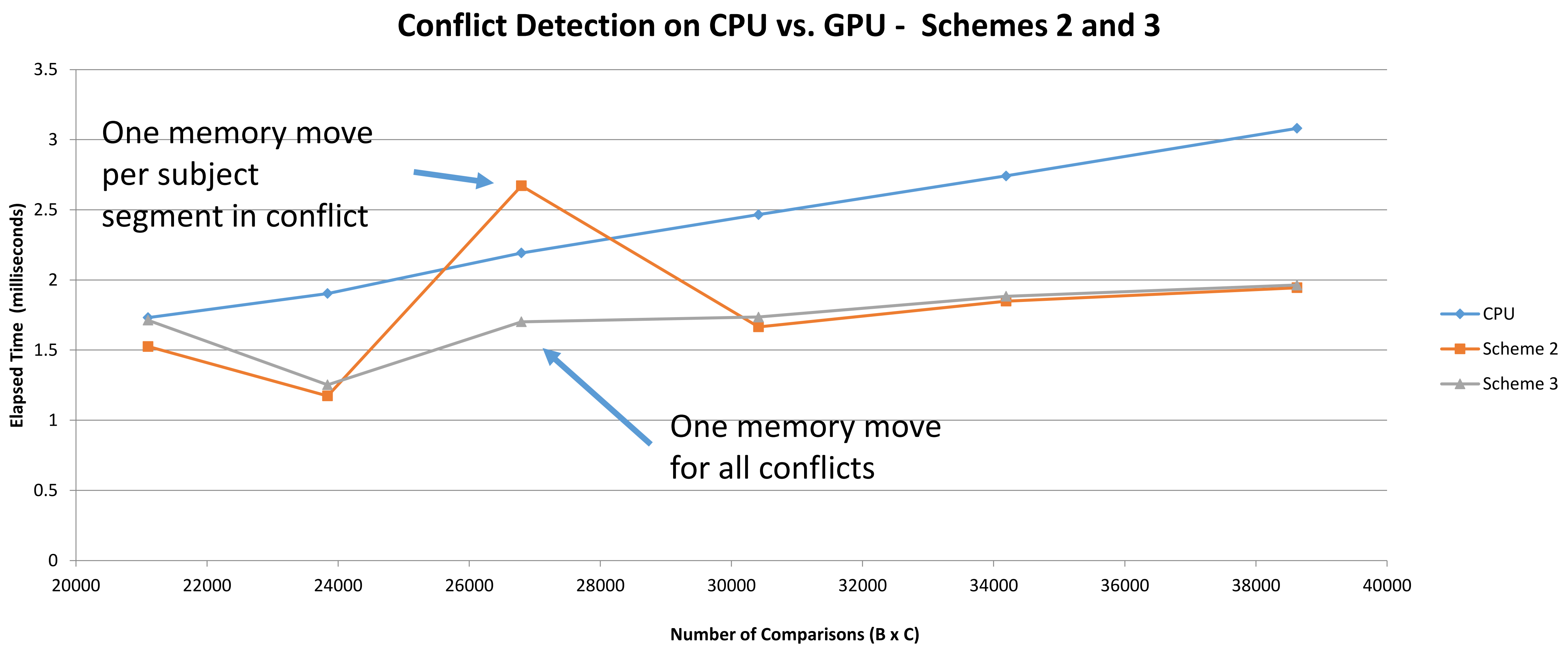
Three different designs:

1. Each subject segment is compared vs. all object segments with one call, hence ***B*** calls for one flight-to-flight compare
    1. One Grid, one Block, ***C*** Threads per segment in the subject trajectory
2. All subject segments compared vs. all object segments with one call, hence one call for a flight-to-flight compare
    1. One Grid, ***B*** Blocks, ***C*** Threads utilized
    2. Results are communicated back to CPU with memory moves for each subject segment with conflicts
3. Same Grid/Block/Thread as in option 2, results consolidated on the GPU then communicated back to CPU with one memory move

Run time comparisons – time per compare



Run comparisons - continued



Future Work

- Try to do all A*B*C compares with one call to GPU
- Current design uses General GPU Memory and Shared GPU Memory (faster, smaller than General Memory, shared by threads within a grid). See if there is a way to use the even more limited Constant Memory on the GPU
- Try In-lining function calls made by kernel
- Explain the decrease time execution time for the 24,000 BxC case

Technical Summary

1. Trajectory Build Engine
    - Framework to build trajectories for use in other experiments was a success; used extensively in other parts
2. Use of Forecasted Weather
    - For flights over two hours, use of forecasted weather is fairly simple (interpolation between hour sets) and is justified given errors in current flight times
3. Optimal Wind Aided Trajectory
    - Through experimentation, an efficient algorithm that quickly finds a close-to-optimal flight path was developed
    - UAS operators (high wind-speed to aircraft speed ratio) would benefit from such an algorithm

Technical Summary (continued)

4. Parallel Conflict Probe
    - Using CUDA and GPU, time can be improved over single-core CPU execution
    - Speedup was not as dramatic as I had hoped (more work to be done)
    - CUDA primitives for synchronization of work make GPU program “simple” (in my opinion, simpler than schemes in languages like Java that have thread synchronization primitives)

Project Timeline

Date		Milestone
November	✓	Complete basic capability of building a trajectory
December	✓	Analyze the use of forecasted weather
January	✓	Wind-aided optimal trajectories (using Particle Swarm Optimzation) [This became version 1; improvements made in April/May]
January	✗	Implement BADA 4.0 trajectory generation, compare to BADA 3.0
February	✓	Initial implementation of conflict detection
April	✓	Final conflict detection, with speed measurements;
May	✓	Final presentation/documentation complete

Deliverables

- Python Source Code
  - Trajectory Generation
  - Use of Forecasted Weather
  - Wind Optimal Paths
- C Source Code
  - Conflict Detection
- Design documentation
- Class presentations and reports

Final Thoughts

- Goals achieved:
  - Built the Framework that is usable for general experiments
  - Learned Python
  - Learned basics of GPU programming
- Was more work than I might have liked
- Two papers co-authored with Dr. Torres accepted by the Digital Avionics System Conference (http://ieee-aess.org/conference/2017-ieeeaiaa-36th-digital-avionics-systems-conference), September 16, 2017
  - "Wind Gradients and their Impact on Trajectory Prediction"
  - "Wind Optimal Trajectories for UAS and Light Aircraft"

References

BADA	Eurocontrol Base of Aircraft Data (BADA), http://www.eurocontrol.int/services/bada
WGS	Eurocontrol Base of Aircraft Data (BADA), http://www.eurocontrol.int/services/bada
AMS	Aircraft Modelling Standards for Future ATC Systems; EUROCONTROL Division E1, Document No. 872003, July 1987.
PSO1	Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proc. IEEE int'l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.
PSO2	http://www.swarmintelligence.org/tutorials.php
GRIB	GRid in Binary (GRIB), the World Meteorological Organization (WMO) Standard for Gridded Data, http://dao.gsfc.nasa.gov/data_stuff/formatPages/GRIB.html
ERAM1	"ERAM Conflict Management, Off-Line Problem Determination, and Utility Algorithms", FAA document FAA-ERAM-2008-0423
ERAM2	"ERAM Flight Data Processing (FDP) and Weather Data Processing (WDP) Algorithms", FAA document FAA-ERAM-2006-0045
CUDA1	"CUDA By Example", Sanders and Kandrot
CUDA2	"CUDA C Programming Guide", Nvidia Corporation