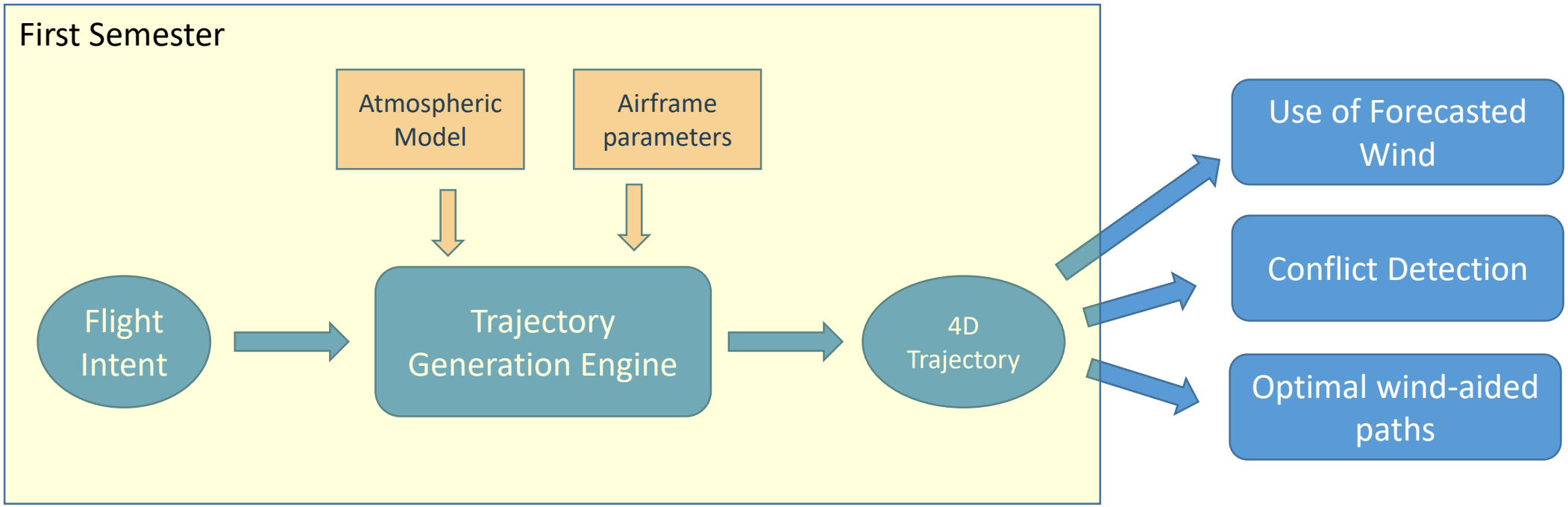
# AMSC/CMSC 664
# Mid Term Report

March 7, 2017

Jon Dehn

# Project Goal

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



# Use of Forecasted Wind/Temperature Data

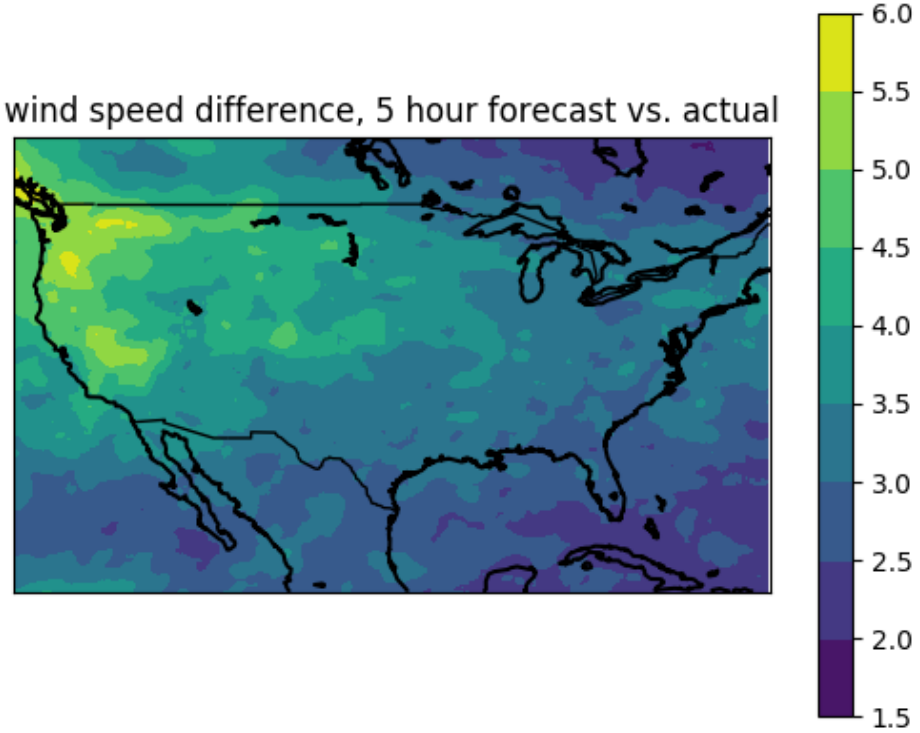
- Current Operational Systems use only the current weather information to build a trajectory
- Difference systems have different time horizons; accuracy of longer flights in systems with longer time horizons may be improved by using NOAA weather forecasts

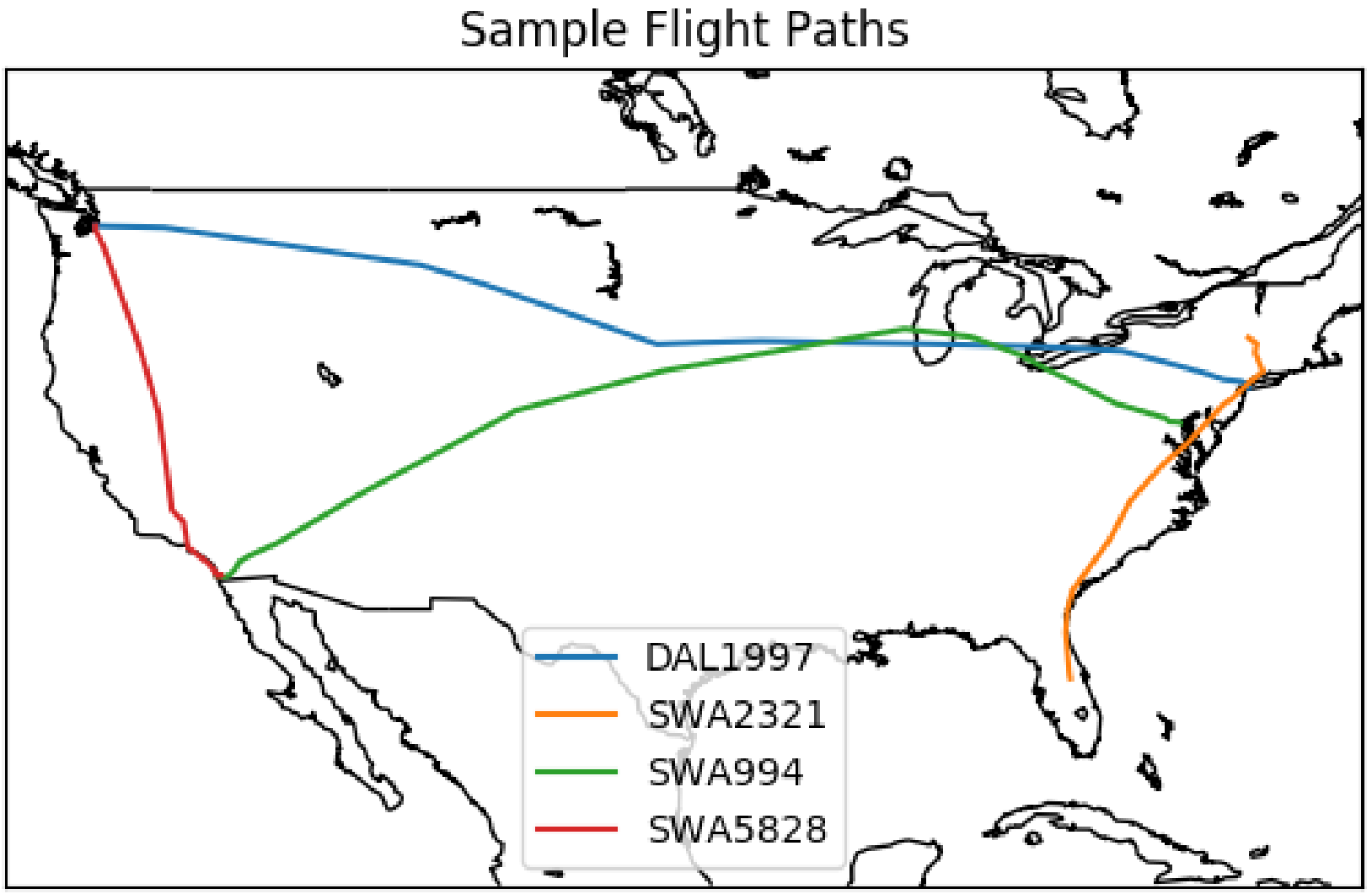
# Experiment:

- 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

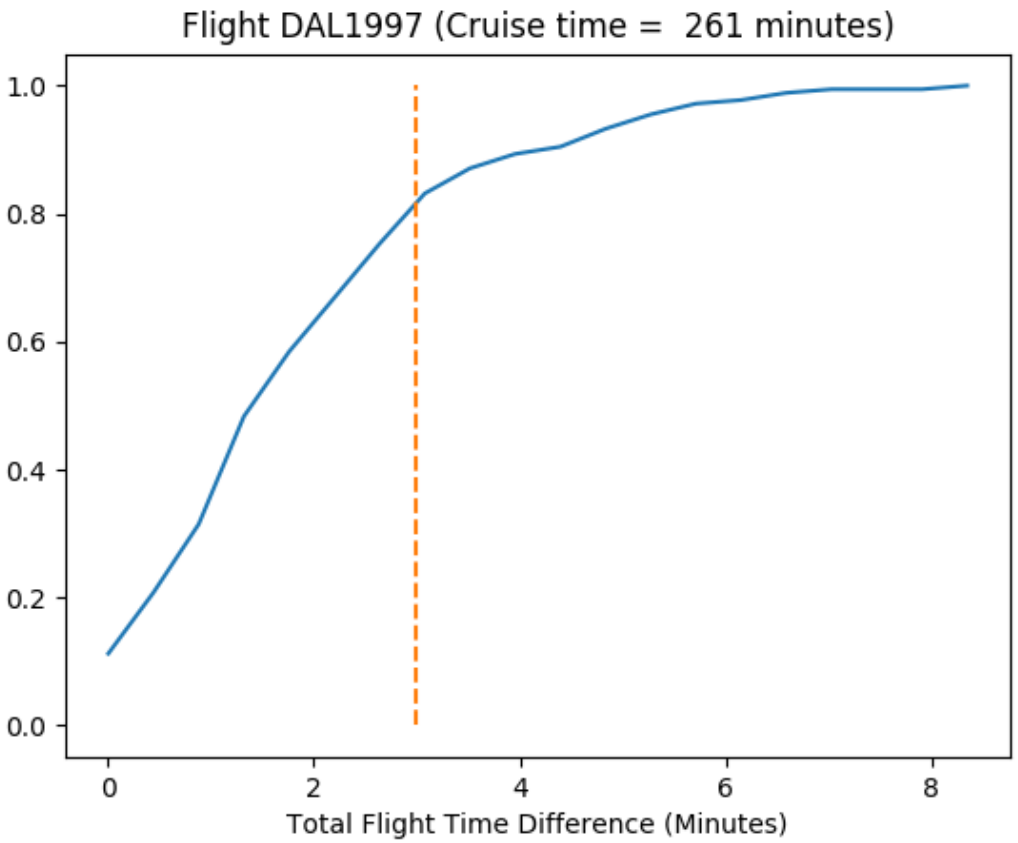
# How good is the forecast?

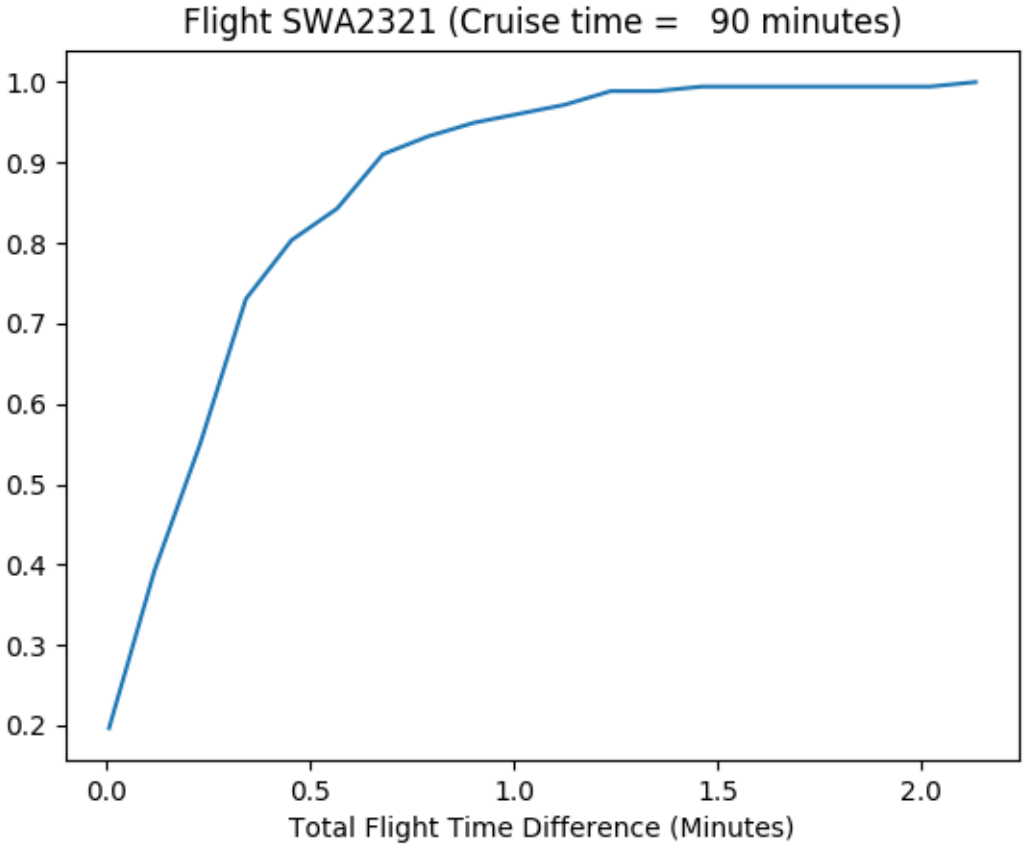
- Data collected for days in 2017 (so far)
- Four samples per day collected
- Difference between wind speed forecasted and time T+5 hours and the actual wind reported at time T+5 hours calculated
- Accuracy varies by geography, east better than west
- A flight traveling across country, with an error of 3 knots in typical cruise speed of 420 knots, would be 2 minutes off in total flight time
- In practice, since flights don't fly directly into the wind and wind varies across the flight, error in forecast accounts for at most 0.2 minutes of flight time



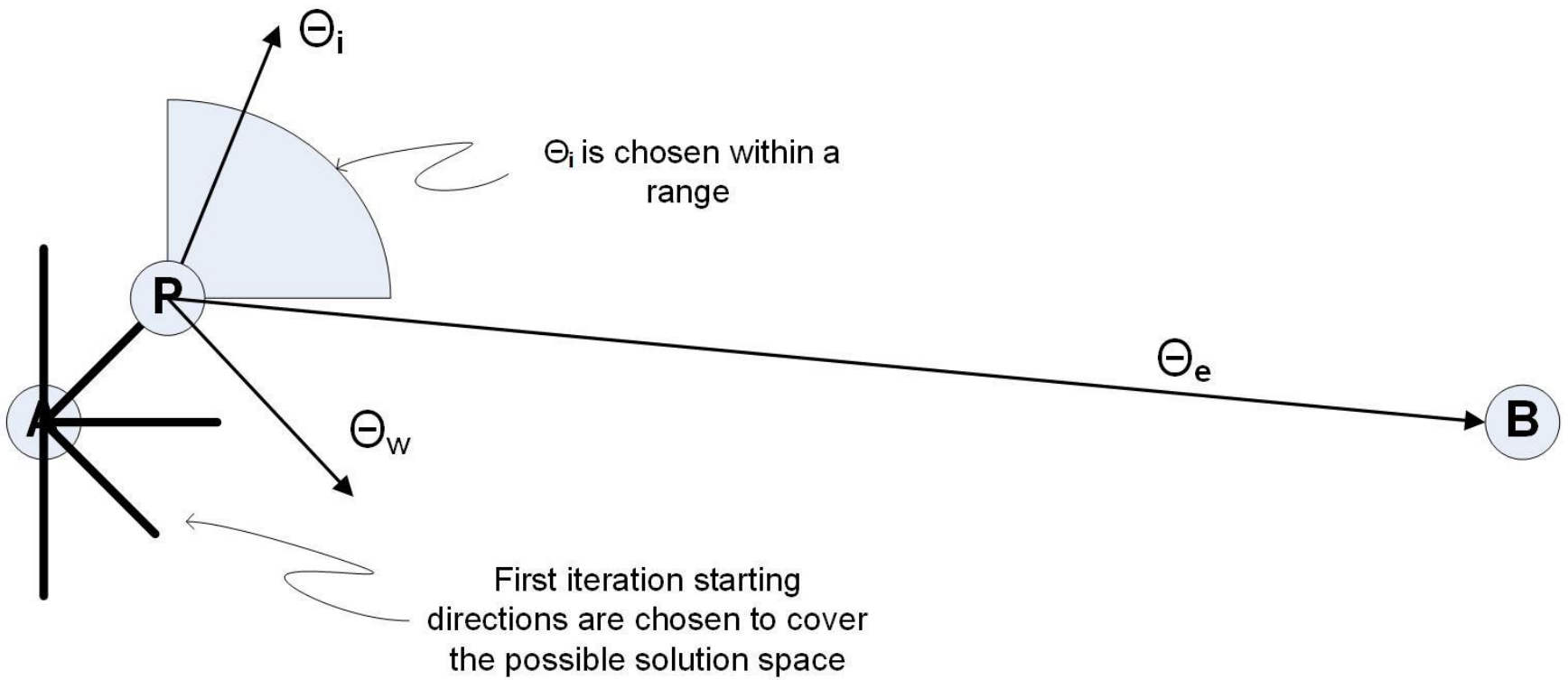

Sample Flight Paths
DAL1997
SWA2321
SWA994
SWA5828

# Results – 178 samples of each flight



# Wind Aided Trajectory - Overview



# Overview description

- First iteration computes 5 paths, each with a difference starting course, 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
- Algorithm stops when some number (3) iterations have not improved on best fuel burn

# Optimal Wind-Aided Paths Refinements

- Algorithm used steps from starting point to ending point, determining a path to take based on weights applied to
  - Inertial course
  - Wind direction
  - Direction to end point
- Original plan determined end-point-direction weight as

$$W_e = (1 - \frac{dist(P,B)}{dist(A,B)})$$

- $W_e$ is limited to >= 0.0; then apportioned remaining weight between wind direction and inertial, according to some pre-determined ratio (70/30 was initial guess)
- Once path is found to end point, full trajectory is built and fuel consumption computed

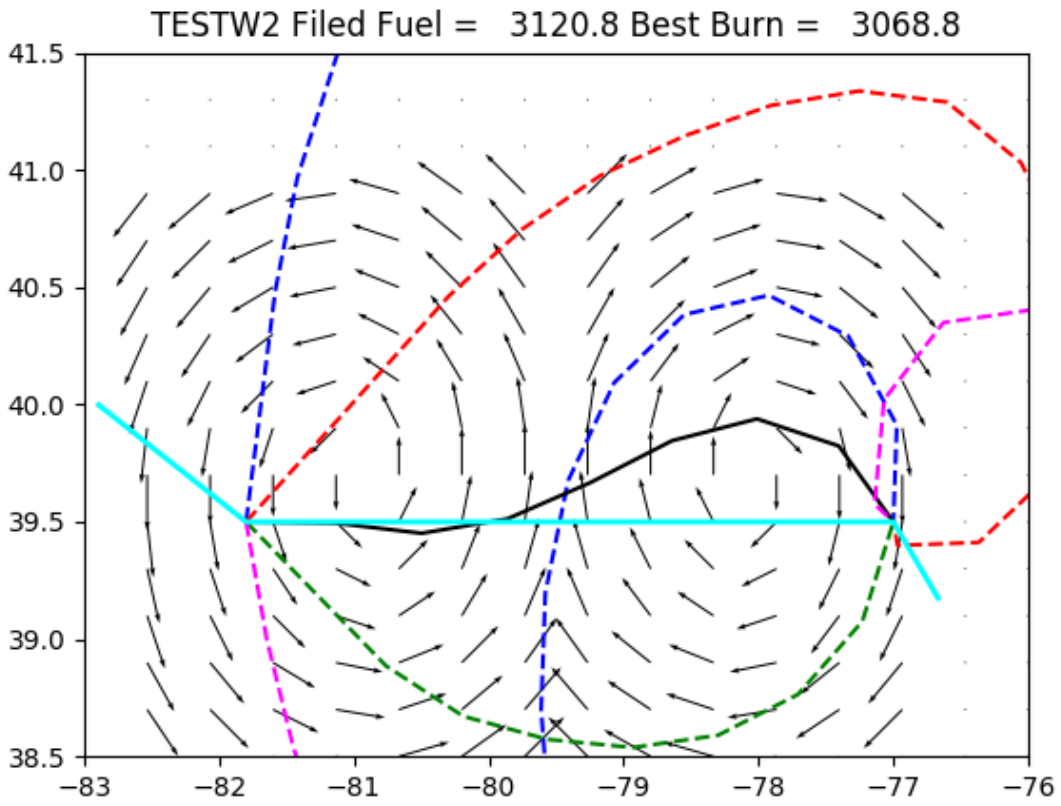
# Updated weights

- Initial end-point weight changed to reduce its contribution when far away from end point (typical Q is 2.0):

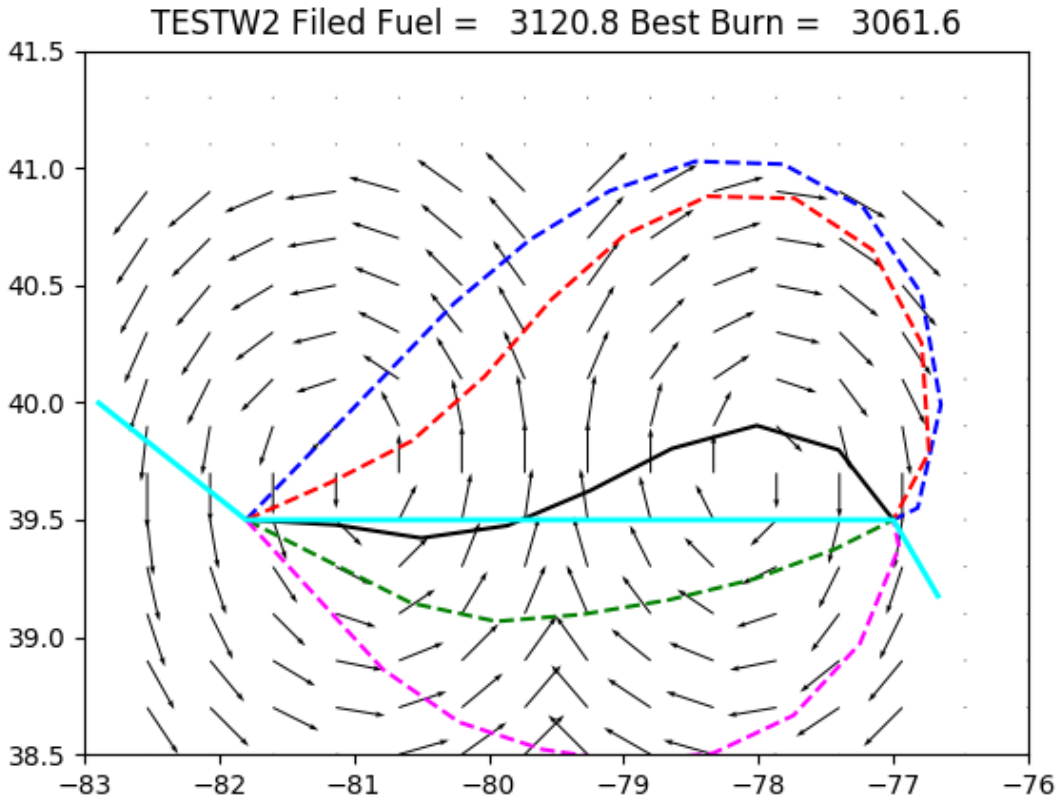
$$W_e = (1 - \frac{dist(P,B)}{dist(A,B)})^Q$$

- Again, (1 - distance_ratio) is limited to range 0.0 .. 1.0
- Wind/inertial split being used is 20/80.
- If wind direction is away from end point, wind weight is set to zero
- Path found by PSO is compared to brute force path varying initial course and wind/inertial split with a range of values to arrive at these values

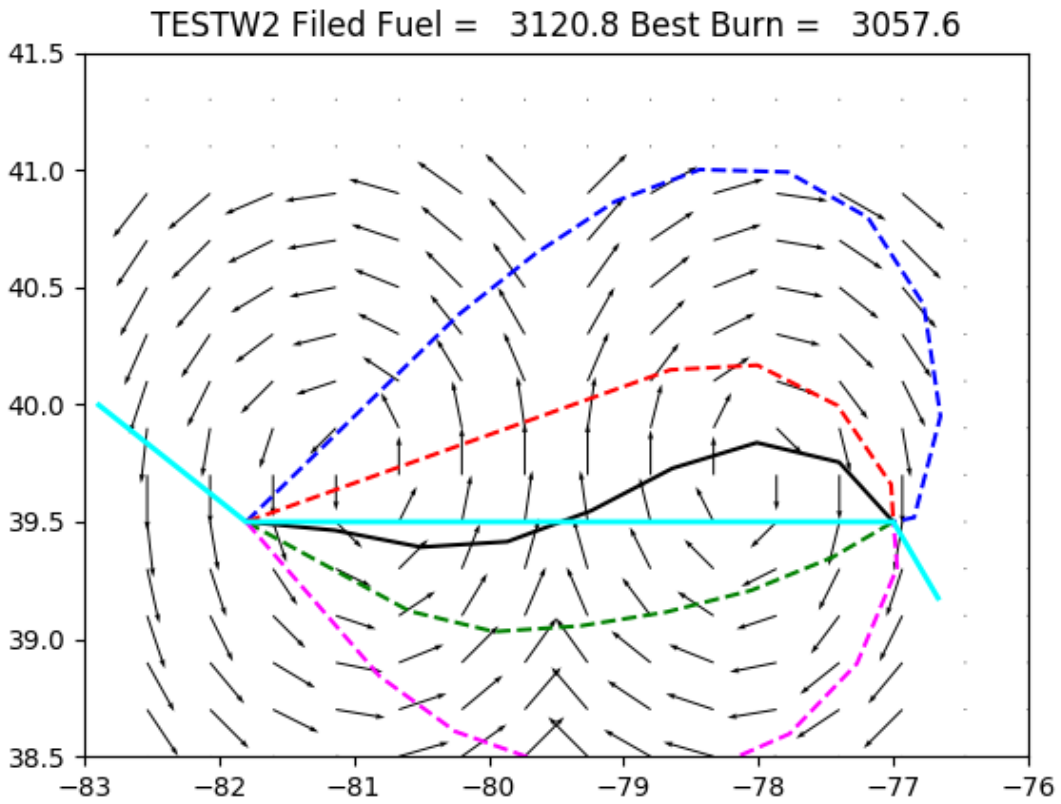
# Sample Results – iteration 1



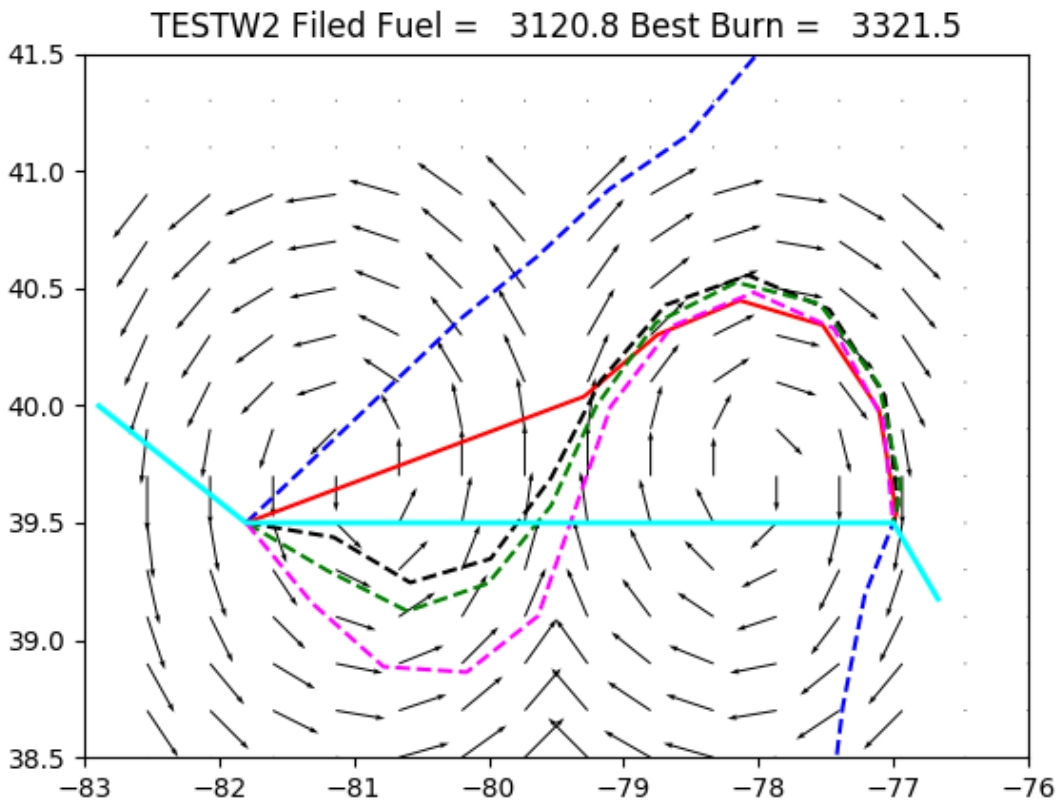
# Sample Results – iteration 2



# Sample Results – iteration 3 (1.6 minute better)



# Weight at 70/30: too much wind influence



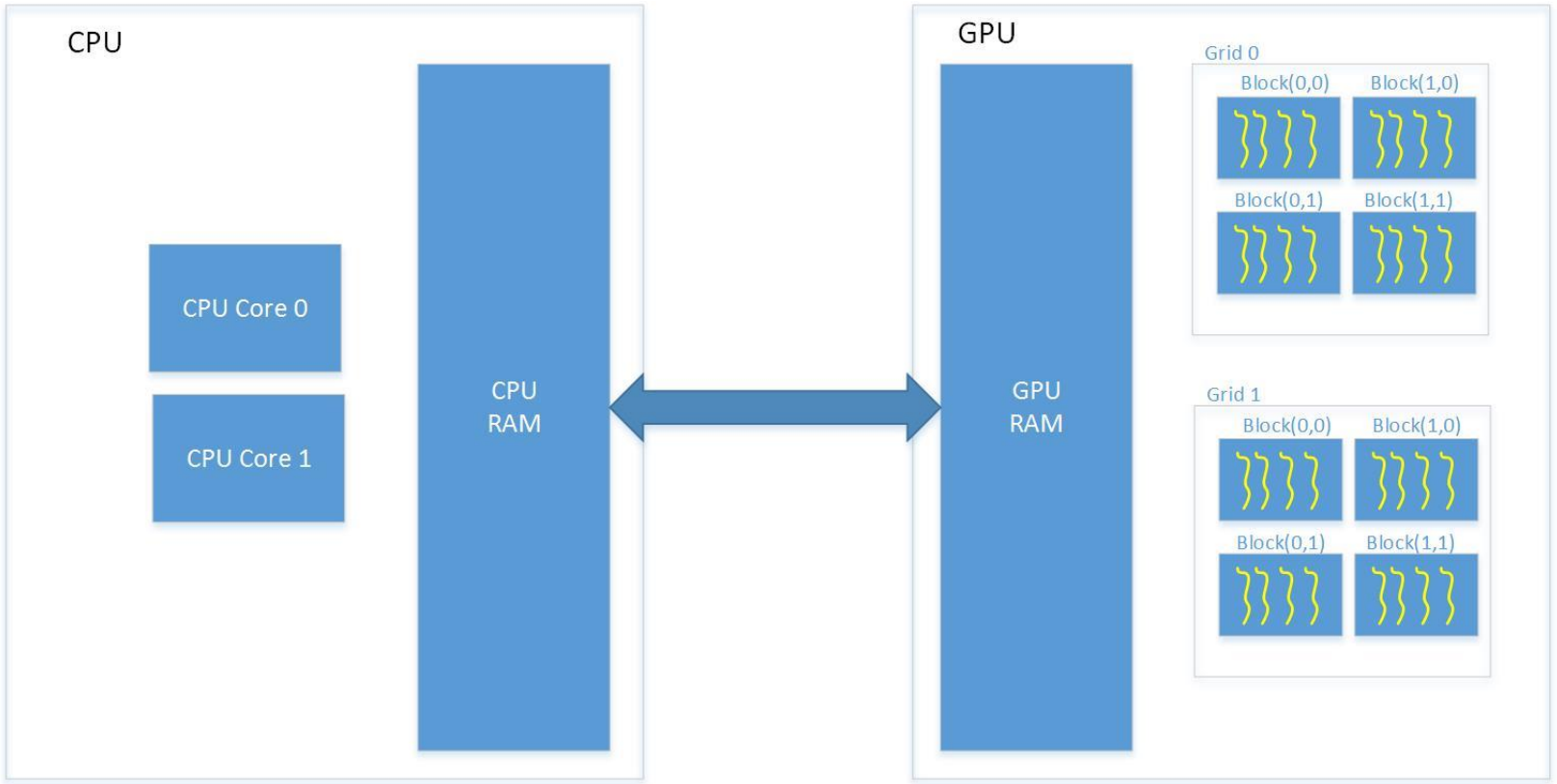
# Conflict Detection

- Each flight's trajectory is composed of several (N) segments
- Aircraft-to-aircraft conflict detection checks one flight's segments vs. another flight's segments (N*N compares)
- A full conflict detection scheme compares any changed/new flight (the *subject*) to all existing (M) flights (the *objects*) (M*N*N 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

- In practice, some compares can be skipped based on high level checks:
  - Do the flights overlap in time at all?
  - Do the flights overlap in X/Y space at all?
- Using the parallel processing constructs in GPUs, several segment-to-segment compares can be done simultaneously
  - M*N*N becomes M*N*1
- High level checks can still be applied if necessary

# 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 (GeForce 960 has 1024 cores)

# Design

- Initial design will store each existing flight’s data in GPU memory
- One Grid, one Block, one Thread per segment in the object trajectory
- In parallel, one segment in the subject’s trajectory will be compared against all segments in the object trajectory
- All coding is done is C; subject and object trajectories are loaded from json files generated from CMSC 663 work

# 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) |
| February | ☐ | Initial implementation of conflict detection |
| April | | Final conflict detection, with speed measurements; |
| Mat | | Final presentation/documentation complete |

# Project Status

- Forecasted weather computations complete; paper abstract of this work submitted to the Digital Avionics System Conference (http://ieee-aess.org/conference/2017-ieeeaiaa-36th-digital-avionics-systems-conference), September 16, 2017
- Wind Optimal trajectories algorithm created; further tuning of parameters may occur. Paper abstract of this work submitted to DASC
- 80% of conflict detection coded in C. Some parallel processing has been tested. Nvidia's CUDA toolkit, along with Microsoft Visual Studio, is used for this work. On track to complete by May.