

A Framework for construction of and computations on Four Dimensional Aircraft Trajectories

CMSC 663-664

Author:

Jon Dehn

jondehn@umd.edu

Advisor:

Dr. Sergio Torres

Fellow, Leidos Corporation

Sergio.torres@leidos.com

Abstract

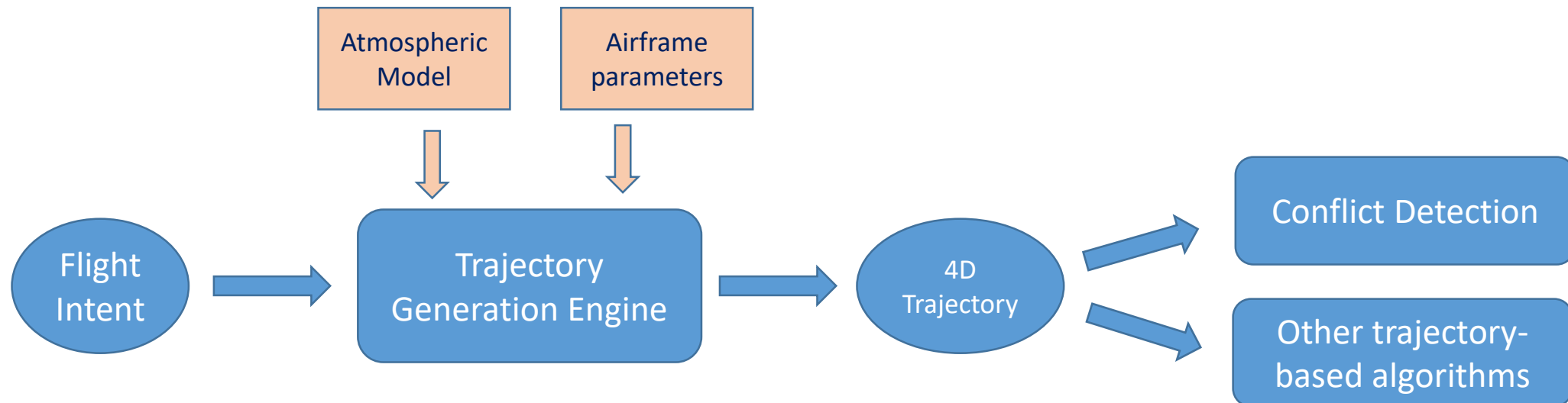
- The proposed project is a framework for testing various compute-intensive algorithms in air traffic management.
- The framework will be modular, in order to test different implementations of various algorithms for both accuracy and performance.
- Various sources of test data can be used with the system, including test data created specifically for proof of concept and test data obtained from real air traffic situations

Background/Introduction

- Global Air Traffic Control is distributed among multiple Flight Information Regions (FIRs). Each country is responsible for safely and efficiently handling traffic in their FIRs. At the core of this is building an estimate of each aircraft's *trajectory* (a four dimensional description of the flight path).
- Most countries have 1 FIR; due to the volume of air traffic and the complexity of its airspace, the US has 20
- Those trajectories are used for many purposes, such as *conflict detection* (which predicts whether two aircraft will be too close to each other, or if a single aircraft will fly into restricted airspace)

Project Goal

- Build a modular system where
 - Different trajectory generation algorithms can be tested
 - Algorithms using those trajectories can be tested for performance and accuracy



Coordinate Systems

- For this project, points on the earth's surface will be transformed to a Cartesian coordinate (x,y,z) on a unit sphere (radius = 1.0)
 - Using the radius of the earth, distances on the surface of the unit sphere can be converted to nautical miles (NMI), for things like speed calculations (Knots)
 - Other calculations such as intersection of great circle arcs are simplified when the unit sphere is used
- All external lat/longs are supplied in WGS 84 coordinates[WGS], which assumes the earth is an ellipsoid with specific major and minor axes (same as is used by the GPS). Translation to the unit sphere is straight forward, and supplied as routines in many programming languages

Speeds

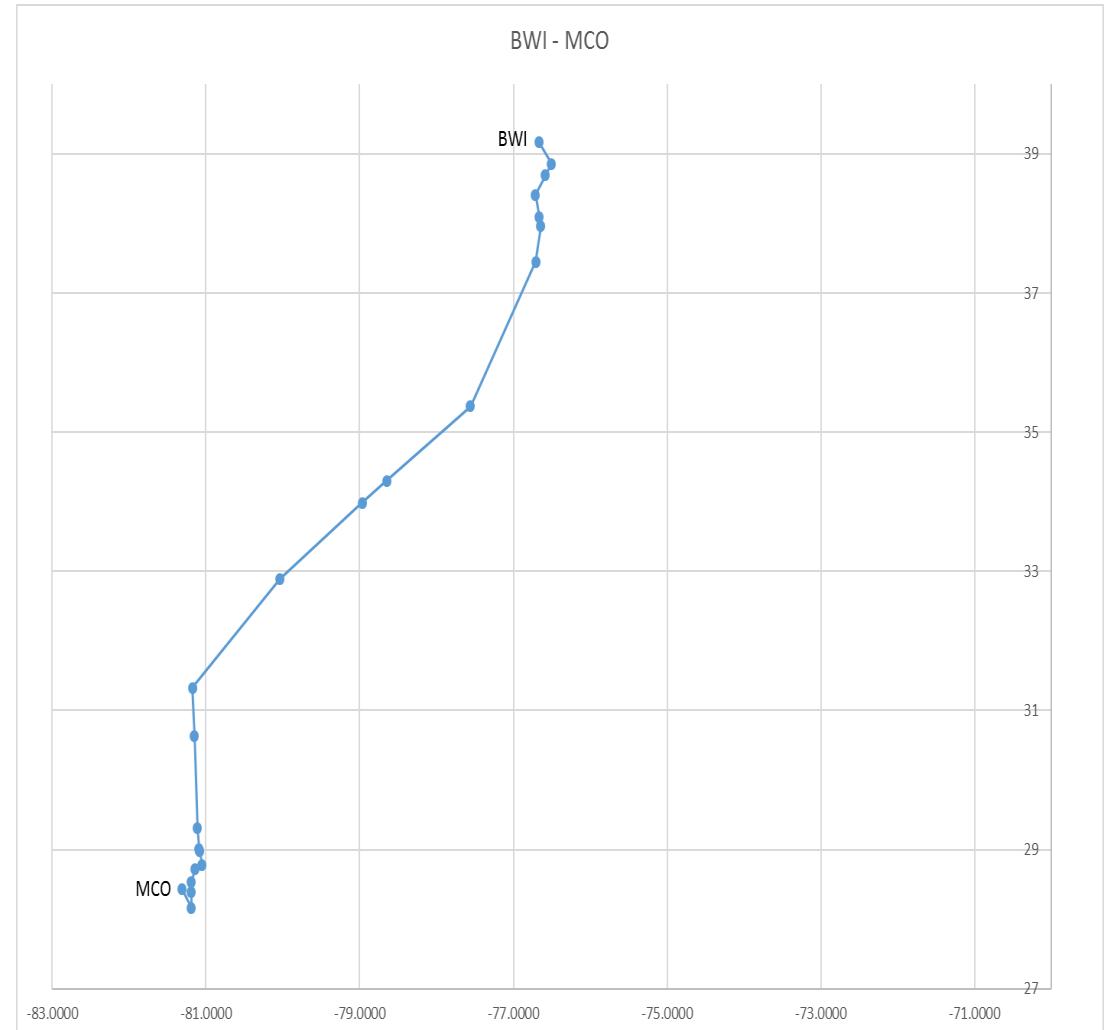
- Several speeds are referred to:
 - **True Airspeed** (TAS) – speed of the aircraft relative to the surrounding air mass (which may be moving with respect to the earth's surface)
 - **Ground speed** – speed of the aircraft over the surface of the earth; this is the true airspeed adjusted by the winds moving the air mass
 - **Calibrated airspeed** (CAS) – airspeed as known to the aircraft's instrumentation; this is based on a “calibration” that takes into account any known bias in the instruments. For a constant CAS, TAS will vary depending on altitude
 - **Mach** – speed of sound; important here because aircraft plan to hold CAS and Mach constant; there is a *crossover altitude* at above which planning is done based on Mach speed, CAS and Mach will be equal at that altitude

Trajectory Generation Basics

- Flight Intent is supplied by the pilot (or airlines), indicating:
 - What path over the surface of the earth will be followed
 - What altitude is desired
 - What speed is desired (true airspeed)
 - What kind of airplane they have
 - When they will depart
- Example: Southwest flight 3156 from Baltimore to Orlando:
 - altitude 40,000 feet
 - speed 452 knots
 - Boeing 737-300
 - Depart at 10:35 am
 - CONLE3 COLIN J61 HUBBS J193 HCM ISO J121 CHS J79 MILIE J79 OMN CWRLD4

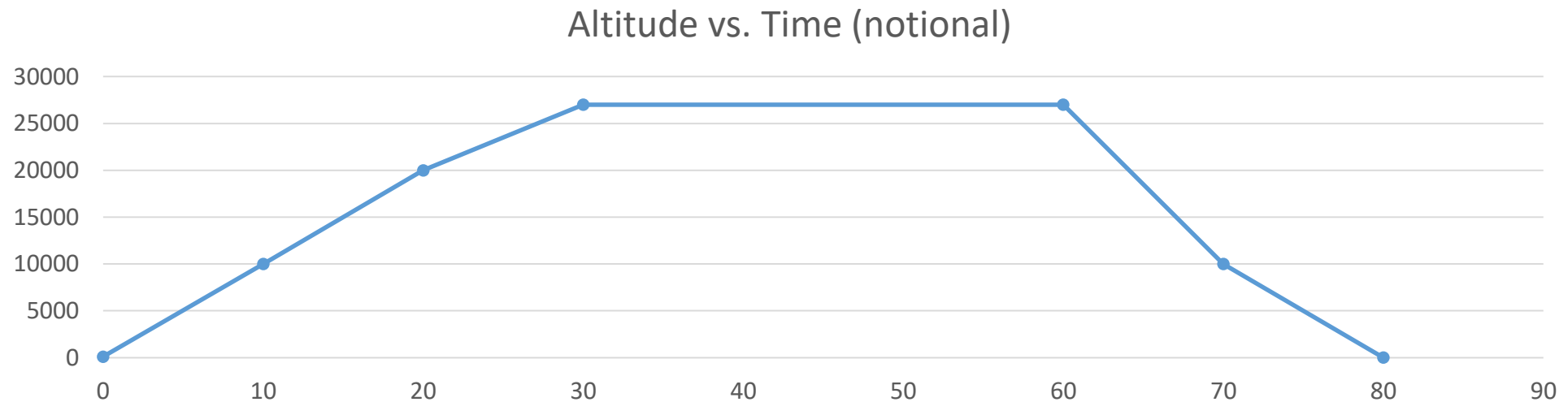
First two dimensions...

- The complex “route string” is translated into a set of *waypoints*, each with a geodetic latitude/longitude
- Waypoints are connected by great circle arcs
- “sharp” turns can be smoothed by adding additional waypoints



Add altitude and time...

- From this a vertical profile can be built, adding **altitude** and **time** to each waypoint
- Each *segment* endpoint is identified by (latitude, longitude, altitude, time) or equivalently (Cartesian x/y/z on unit sphere, altitude, time).
- Each segment also has a constant vertical speed (feet/minute), starting speed and horizontal acceleration, so that the position of the aircraft at any time within the segment can be easily found
- There may be more segments than waypoints



Climb and descent rate – from total energy model

$$(Thr - D)V_{TAS} = mg \frac{dh}{dt} + mV_{TAS} \frac{dV_{TAS}}{dt}$$

- 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

Rate of Climb/Descent

- Typical aircraft mode during climb and descent holds speed (CAS or Mach) and throttle constant, yielding a change in altitude; so above can be solved for dh/dt :

$$\frac{dh}{dt} = \frac{(Thr - D)V_{TAS}}{mg} \left[1 + \left(\frac{dV_{TAS}}{dh} \right) \left(\frac{dh}{dt} \right) \right]^{-1}$$

- The last term can be replaced by an “energy share factor”[AMS]; the ratio of energy allocated to climb vs. acceleration for a constant velocity; a function of Mach:

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

Energy Share Factor

- When holding speed constant, it depends on altitude (above or below tropopause), and whether we are holding CAS or Mach constant (that is, above or below the *crossover altitude* where CAS == Mach)
- When accelerating (e.g., after takeoff when trying to reach at specific climb speed), a constant value is used (e.g., 0.3 when accelerating during climb)
- For example...

Example $f(M)$ below crossover

$$f(M) = \left\{ 1 + \frac{\kappa R \beta}{2g} M^2 \frac{T - \Delta T}{T} + \left(1 + \frac{\kappa - 1}{2} M^2 \right)^{\frac{-1}{\kappa - 1}} \left\{ \left(1 + \frac{\kappa - 1}{2} M^2 \right)^{\frac{\kappa}{\kappa - 1}} - 1 \right\} \right\}^{-1}$$

- κ : Adiabatic index of air = 1.4
- R : real gas constant for air = 287.05287
- β : International Standard Atmosphere (ISA) temperature gradient, altitude below tropopause = -0.0065 (degrees Kelvin/meter)
- T : air temperature
- ΔT : temperature differential at mean sea level from ISA temperature

Rate of Climb/Descent

- ROCD is typically expressed as change in pressure altitude (H_p), instead of geodetic altitude (h); pressure altitude is known to the pilot via the plane's altimeter:

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

- What about Thrust and Drag?

Thrust

- Thrust depends on engine type (Jet, Turboprop or Piston), pressure altitude, airspeed (in some cases) and temperature differential
- For example, for a jet engine, max climb thrust would be:

$$Thr = C_{TC1} * \left(1 - \frac{H_p}{C_{TC2}} + C_{TC3} * H_p^2 \right) (1 - C_{TC5}(\Delta T - C_{TC4}))$$

- Where the different coefficients C_x are given for the aircraft type from the airframe parameters[BADA]
 - In the case of a jet engine, there is no term for airspeed

Drag

- Drag is a function of the drag coefficient, which in turn is a function of the lift coefficient:

$$C_L = \frac{2 * m * g}{\rho * V_{TAS}^2 * S * \cos(\Phi)}$$

- Drag Coefficient:

$$C_D = C_{D0,CR} + C_{D2,CR} * (C_L)^2$$

- Drag:

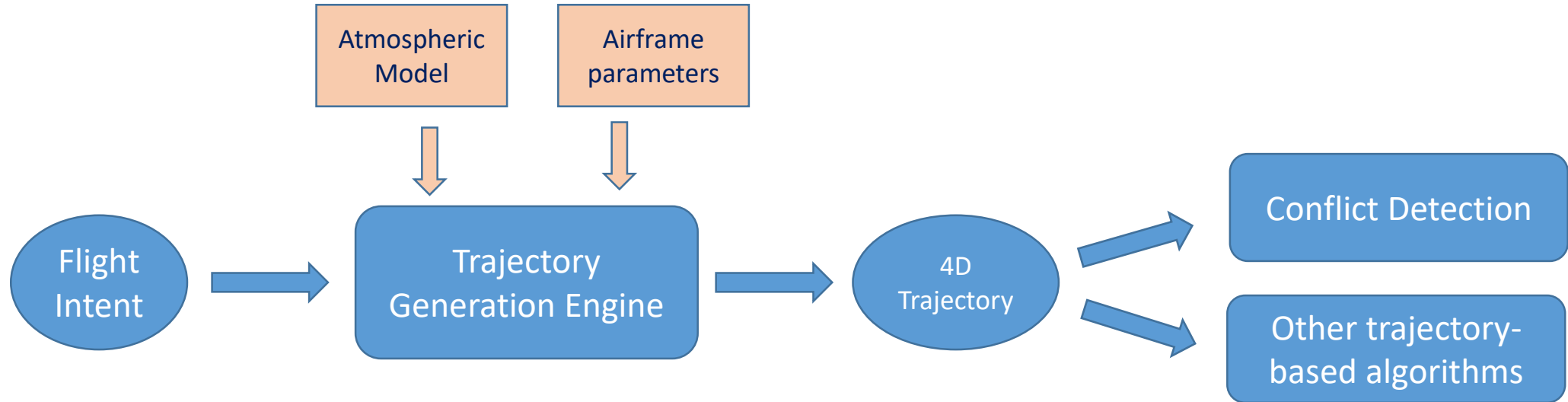
$$D = \frac{C_D * \rho * V_{TAS}^2 * S}{2}$$

- ρ : air density = $\frac{p}{RT}$
- S : wing surface area
- Φ : bank angle, typically zero

Summary of trajectory generation

- Given a starting point (departure airport) and time, solve the ordinary differential equation for dh/dt using a Runge-Kutta technique
 - (perhaps experiment with different order Runge-Kutta techniques, or adaptive step size)
- Descent segments end at a certain point (destination airport), but the starting point (top of descent) cannot be directly computed since mass decreases over the descent; an iterative approach is used here

Review



Approach

- Use Eurocontrol's Base of Aircraft Data (BADA) model for the airframe parameters and equations for trajectory generation[BADA]
 - There are two versions of this model in use; version 3 (which has a rich set of airframe parameters) and version 4 (which is more detailed but covers fewer airframes currently). Both are available to research institutions
- The Atmospheric model, including hourly forecasts, is available from NOAA.
 - Initial testing can be done with a "standard" static model; no winds aloft
 - This data is supplied in "GRIB2" format, and uses a Lambert Conical projection; a corresponding spherical grid is built with uniform increments of latitude and longitude, for efficient lookup of wind, temperature and pressure data[GRIB][ERAM2]
- Flight Intent can be either synthetic for testing or derived from real-world data (via the publicly available "Aircraft Situation Display to Industry" (ASDI) data stream, or websites such as "FlightAware")

Approach – Analysis

1. Look at the increased accuracy of BADA version 3 vs. version 4 for identical aircraft types, balanced against the increase in complexity and computational load
2. Current US system uses current-hour weather model for all computations, even though a flight can last several hours. Implement a version that uses future weather forecasts and compare changes in long-duration trajectories
3. Find an optimal wind-aided trajectory
4. Parallelize conflict detection algorithms, measure complexity of implementation vs. speed tradeoffs for different parallel techniques

Computing Algorithms – use forecast weather

- This is a simple application of the weather model (winds, pressures, temperatures) at the time of the trajectory segment.
- Interpolation between hourly forecasts may be needed
- The key metric measured will be the change in total flight time, and potential movement in space of key trajectory points such as the top-of-descent point.

Algorithms – Wind Aided Trajectories

- The goal is to produce an optimal (as defined by a user supplied criteria; initially Fuel Consumption is used) trajectory given the presence of winds aloft
- The algorithm is based on the Particle Swarm Optimization (PSO) and it incorporates hard and soft constraints that makes it useful for the trajectory problem at hand[PSO1][PSO2]
- PSO relies on the cost quantification of diverse paths and identification of most efficient routes to the 'food' (emulating swarm behavior).
- Both hard constraints (depart at fixed point A and arrive a fixed point B) and soft constraints (deviate along the transverse direction by no more than x miles) are incorporated by the use of ' $1/x^q$ attractors'.
 - These attractors act as relative weights to the random search paths so as to guide the solution towards compliance with the constraints.

Aircraft Conflicts

- A FIR has a set of existing aircraft, built up over time during the day
 - For this project, a set of aircraft will be created
 - Each time a new flight is added, or an existing flight is modified, conflicts must be checked
- Two types of conflicts are recognized:
 - Two aircraft are too close to each other; typical safe margins are 1000 feet vertically and 5 NMI horizontally
 - An aircraft enters restricted airspace, such as the airspace around downtown Washington DC
- This project will concentrate on aircraft – to – aircraft conflicts
- Any given FIR may have hundreds of aircraft active at one time, each trajectory may have hundreds of 4D segments

Computing Algorithms – Conflict Detection

- Brute force conflict detection compares each line segment in a “new” trajectory against all segments in existing trajectories, looking for the point of closest approach
- State of the practice sequentially applies several “filters” at a level higher than the segment level to eliminate as many checks as can be safely removed, in order to avoid the costly segment-to-segment comparison[ERAM1]
- A sequential conflict detection algorithm will be compared against (possibly multiple) parallel implementations, comparing change in computation time
 - Initially, the “intersection” algorithms can be very simple
 - Time permitting, the publicly available algorithms used in the US can be measured against parallel implementations
 - The technique for parallelization will be the focus here, possibilities include:
 - Multiple GPU processes
 - Multiple CPU cores on a single computer
 - Multiple general purpose computers with network interconnectivity

Implementation

- Implement in Python:
 - Portable to many operating systems
 - Has support for parallel processing
 - Has the concepts of exception handling, classes, objects, modules, and packages
 - But, is not strongly typed or compiled before execution, leading to more errors found at run time
- PyDev Eclipse plug-in used for an IDE; GIT used for configuration management
- Platform will be a personal computer (laptop or desktop).
 - For GPU experiments, desktop with a high end graphics card (specific card to-be-determined)
 - If multiple general purpose CPUs networked together are needed, inexpensive Raspberry Pi computers can be used

Validation Methods

- Eurocontrol conveniently provides a tool that uses their version 3 equations to produce a 4D trajectory. This can be used to verify the basic trajectory generation engine.
- Adding a non-zero-wind model can be verified through spreadsheet analysis of the produced trajectories
- A BADA version 4 implementation can be verified against the version 3 result
- Wind aided trajectories will be tested by using synthetic weather model to force a better solution to exist
- Conflict cases are constructed between flights (where closest approach is within 5 NMI, slight above 5 NMI and greatly above 5 NMI) for testing

Expected Results

For these main experiments:

1. BADA version 3 vs version 4, it is expected that version 4 offers greater accuracy, especially in low altitude situations, but may not be worth the extra complexity and computation cost
2. Using future weather forecasts as opposed to only the current conditions should yield improvements in flight duration times at little computation or complexity expense; memory used would increase
3. Optimal wind-aided trajectories should find “less expensive” trajectories than nominal case
4. Parallel solutions for the conflict detection algorithms should reduce computation time, perhaps meaning that simpler algorithms can be used in place of the existing state of the practice

Conclusion

- The resulting framework can be used to carry out the experiments listed here, and can be used for future work
- By building the system in components with clearly defined interfaces, different implementations can be plugged in without affecting the overall framework

Proposed Project Schedule

- Thanksgiving:
 - implement BADA version 3.
 - Detail algorithm for wind aided trajectories
- December – implement use of forecast weather
- January –implement BADA 4, compare the two
- February – initial parallel conflict detection algorithm, wind aided trajectories
- March – additional parallel algorithm tests
- May – final algorithms and analysis

Deliverables

- Python Source Code
- Design documentation, including interface definitions
- Results
- Class presentations and reports

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