Presented at the AIAA Guidance, Navigation and Control Conference, August 15 -18, 2005, San Francisco, California.

A Rapid-Prototyping Environment for En Route Air Traffic Management Research

P. K. Menon[*], G. M. Diaz[†] and S. S. Vaddi[‡]
Optimal Synthesis Inc., Palo Alto, CA 94303

and

S. R. Grabbe[§]
NASA Ames Research Center, Moffett Field, CA 94305-1000

An interactive software environment (CARAT#) that allows rapid prototyping of en route air traffic management algorithms is described. This software employs the NASA-FACET software as its computational engine, and allows the user to rapidly develop and evaluate en route air traffic management algorithms. Two versions of CARAT# have been developed. The first is built upon a commercial software platform, while the second version employs a freely available interactive, scriptable environment as its foundation. In addition to permitting direct access to the FACET capabilities, these interactive environments enable the users to readily build additional functionality into FACET and allow the rapid integration with several commercial software packages. The use of CARAT# software is illustrated through the formulation of a few research problems.

I. Introduction

Future air traffic management (ATM) systems are expected to contain several algorithms and software tools for decision-support and airspace automation[1-3]. Continuing research at NASA and other ATM technology centers nationwide has resulted in the development of multiple software tools for investigating future air traffic management system concepts. Specifically, the FACET (Future ATM Concepts Evaluation Tool, Reference 4) software was developed at NASA Ames Research Center to provide research capabilities for the en route portion of the air transportation system. Due to the extensive capabilities offered by FACET, it is expected that this software will continue to play a central role in a variety of air traffic management research initiatives in the future.

While FACET software provides very powerful capabilities, it is often necessary to extend its capabilities during the course of individual research projects. Due to the fact that the FACET is written in C, with a JAVA GUI, modifications require a significant computer programming investment. While such investments can be justified when multi-year investigations are undertaken, it is difficult to find adequate resources to do this on smaller research efforts. In these cases, it is desirable to have a rapid prototyping software environment in which ideas can be rapidly explored without being bogged-down with coding issues such as memory management, precision and variable typing.

Ideally, it should be possible to code algorithms in an interactive scripting environment, allowing them to be executed without the traditional compile-link cycle. This process will allow algorithms to be developed by analysts with modest programming skills. If these algorithms are found to be useful, they can subsequently be coded into languages like C or Java by professional programmers and integrated with the FACET software. An additional benefit of employing an interactive scripting environment is that it permits researchers to share software packages without worrying about inter-operability issues. This fosters collaborative algorithm development without demanding extensive programming expertise from the analysts.

[*] Chief Scientist, 868 San Antonio Road. Associate Fellow, AIAA.
[†] Software Engineer, 868 San Antonio Road.
[‡] Research Scientist, 868 San Antonio Road.
[§] Aerospace Engineer, Mail Stop 210-10. Member, AIAA.

Such an approach has proved highly successful in automatic control and signal processing disciplines, where the availability of scriptable software environments such as MATLAB®[5] and MATRXX®[6] have resulted in the rapid acceptance of advanced system design techniques by the industry. It has been observed by many computer scientists that large, complex applications universally benefit from being scriptable.

The objective of the present research is to develop a scriptable, interactive environment that provides simpler access to the FACET functionality. This software package is termed as CARAT# (pronounced "Carat-Sharp", Configurable Airspace Research and Analysis Tool – Scriptable). CARAT# allows the user to access FACET functionality through a set of scripts, executable interactively, without resorting to the traditional compiling and linking operations. As in the past experience with scriptable environments, such capability can provide immediate benefits by allowing a large number of analysts to employ FACET for research purposes. Moreover, a wide array of commercial software packages can readily be integrated with FACET using the scripting environment to conduct a wide range of research projects. Powerful graphical tools and visualization methodologies available commercially can be used to convey the results with only a modest programming investment.

Two popular scripting environments[5, 7, 8] were used for CARAT# development. Scientists and engineers have long recognized the power of command line interfaces for exploring and manipulating algorithms and data sets. Specifically,

- scripting reduces the amount of code required to perform tasks,
- scripting allows for rapid integration with other software packages,
- the interpreter helps with rapid code development by allowing the user to run code without compilation.

MATLAB[5] interactive scripting environment has been popular in the automatic control and signal processing communities for several years. Another scripting environment that is rapidly gaining acceptance is the Python© [7], and with the advent of the java language, the Jython© [8] scripting environment.

The Section II will discuss the overall architecture of the CARAT# software. Section III will describe a set of research problems that illustrate the use of CARAT# in ATM research. These include trajectory optimization using genetic algorithms, formation flying using Fuzzy logic, Monte-Carlo simulation, data fusion using Kalman filter, advanced data visualization and the investigation of a conflict resolution algorithm. The first two examples use commercial toolboxes, the other three employ MATLAB numerical algorithms and graphics capabilities. The conflict resolution example illustrates the versatility of the CARAT# software environment by implementing the modified potential field method in Java, Jython and MATLAB. Conclusions are given in Section IV.

II. CARAT# Software Architecture

The Configurable Airspace Research and Analysis Tool-Scriptable (CARAT#) builds on the Future ATM Concepts Evaluation Tool (FACET) developed at NASA Ames Research Center, and the CARAT software developed under a recent research project with NASA[9]. CARAT# software allows the user to access the FACET functionality through scriptable interfaces built in MATLAB and Jython. Moreover, the CARAT# architecture is flexible enough to enable access to FACET functions over the internet using web services.

Figure 1 illustrates the overall architecture of the CARAT# software. CARAT# is designed to serve as an application programming interface (API) to FACET, which provides Java classes that can be invoked by the user through MATLAB, Jython or Java. Methods in these classes can be used as primitives for building more complex functions. A total of 8 Java classes with over 200 methods were developed during the present research.

The CARAT# software provides access to the FACET functionality through a collection of Java methods that interface primarily with the FACET GUI. CARAT# uses the FACET Java functions to develop a set of primitives which are provided to the user. This architecture provides multiple approaches for using the CARAT# software. Firstly, the Java interface classes can directly be used as components of a standard Java program. Secondly, it can be used in the MATLAB environment by exploiting MATLAB's native Java connectivity. Thirdly, CARAT# classes can be employed in the Jython environment to develop interactive scripts. In the architecture given in Figure 1, the user can integrate programs written in C/C++, Fortran or Ada with the CARAT# software through the higher level language interface provided in MATLAB. Moreover, by providing appropriate web services, CARAT# functionality can be provided over the internet to remote users. Such capabilities will allow for collaborative computing involving multiple researchers.

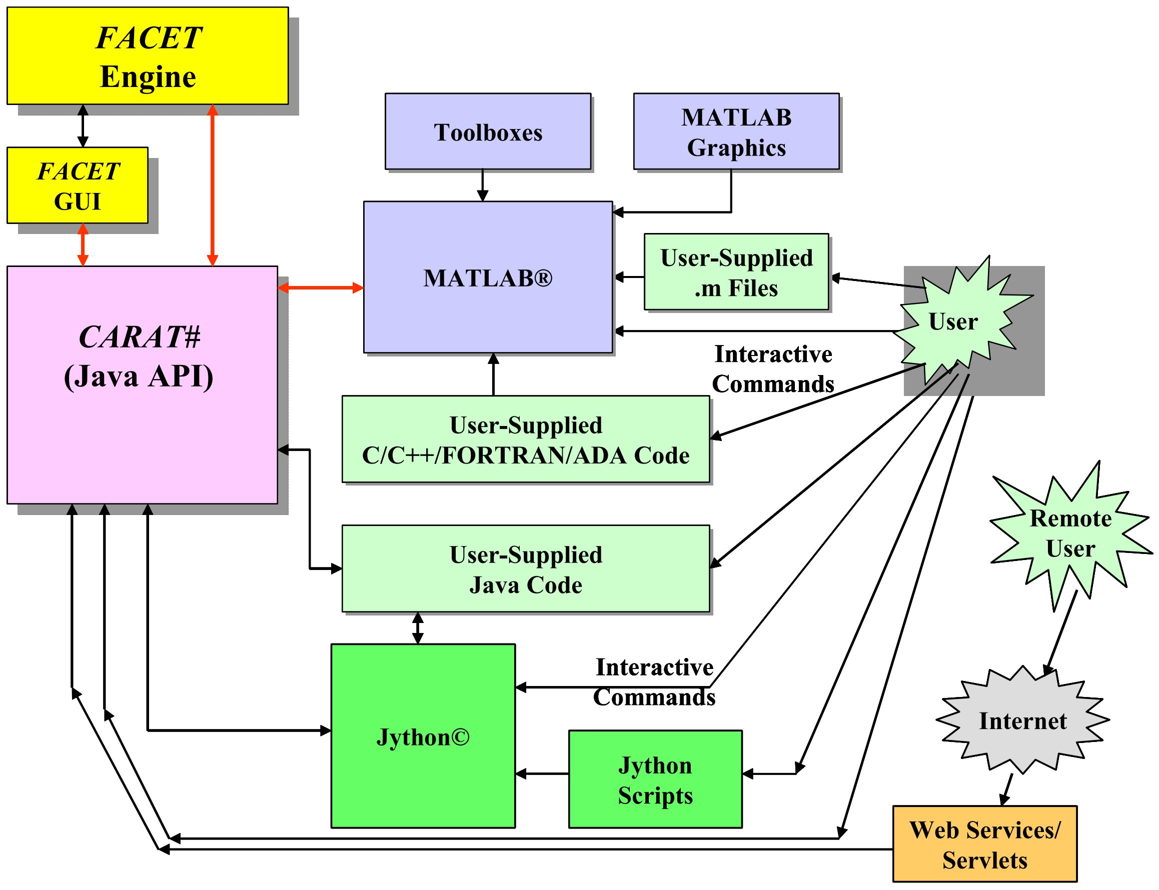


Figure 1. Functional Components of Configurable Airspace Research and Analysis Tool - Scriptable (CARAT#)

The Java classes making up the CARAT# are organized to provide access to the FACET GUI, air traffic simulation controls, states and control variables of aircraft in the airspace, Center/Sector/airport related data, and navigation and atmosphere data. A graphical user interface assists the user in employing the correct syntax for each of the methods during their use. The next section will illustrate how the CARAT# Java classes and methods can be used to formulate and solve a class of research problems.

III. Formulating Research Problems in the CARAT# Environment

As motivated in the previous section, the main objective in developing the CARAT# software is to enable easier development of en route ATM algorithms by providing convenient access to the FACET functionality. CARAT# can be used by the researchers to formulate and evaluate algorithms without extensive programming effort. This section will demonstrate the usefulness of the software by formulating a few research problems. While these problems have been considerably simplified for the sake of illustration, they are representative of the types of investigations often undertaken by researchers in the ATM community. The objective is to illustrate the versatility of the software package.

Aircraft Trajectory Optimization Using Genetic Search Techniques

There is currently a strong interest in system-wide optimization of aircraft trajectories to take advantage of ambient winds, while enforcing system constraints[10]. The research problem discussed in this section illustrates use of the CARAT# software to formulate trajectory optimization problems of interest in en route air traffic management.

The present trajectory optimization problem seeks to determine the fuel-optimal trajectory of an aircraft flying to a destination, while avoiding a restricted airspace. Since the airspace restriction introduces a state constraint for the optimization problem, this represents a fairly complex trajectory optimization task. As the first step in this process, the trajectory is parameterized as a series of straight line segments passing through latitude-longitude waypoints, with longitude being chosen as the independent variable. The initial and final latitude-longitude pairs are held constant during the optimization process. The optimization algorithm determines the latitudes of the intermediate waypoints to minimize the fuel consumption while satisfying the path constraint.

Although any one of the several optimization algorithms[11, 12] could have been used for this problem, the Genetic Search Toolbox[13] for MATLAB is employed as the optimizer. The Genetic Search Toolbox provides functions for implementing classical genetic algorithms[14], evolutionary programming[15], or the more modern genetic programming algorithms[16] in the MATLAB environment. These methods are useful for solving optimization problems that may involve complex constraints, multiple objective functions, discontinuities and nonconvex performance indices. Unlike the conventional optimization algorithms, genetic search methods do not require good initial guesses. However, depending on the parameterization of the problem, they may be good only for generating near-optimum results. Conventional methods can subsequently be used to refine the results.

In the genetic search process, each candidate solution is coded as a *chromosome* that can be manipulated using biologically inspired genetic operations such as mutation and crossover. Resulting offspring are then decoded to determine their *fitness* or the performance index. For the present problem, each candidate trajectory is coded as a character string, specifying the flight plan. Candidate trajectories are assumed to consist of eight latitude increments specified at each of the longitude points. For instance, a candidate chromosome may be of the form:

Example_Chromosome = 'A B G H U T I N'

Each character in this string can be mapped as latitude increments in a nominal flight plan consisting of a line joining the initial and final latitude-longitude pairs. The characters in the string are translated into latitude locations as follows:

Latitude_increment = (char – 'K')/10
Flight_Plan_Latitude = Nominal Flight_Plan_Latitude + Latitude_increment

The latitude increments at the 8 way-points for the example chromosome can thus be decoded as:

('A'–'K')/10 ('B'–'K')/10 ('G'–'K')/10 ('H'-'K')/10 ('U'–'K')/10 ('T'–'K')/10 ('I'–'K')/10 ('N'-'K')/10

= [-1 -0.9 -0.4 -0.3 1 0.9 -0.2 0.3]

It should be noted that the subtraction of the characters is based on their ASCII numeric values. The latitude increment is then added to the nominal flight plan latitudes to obtain the new flight plan.

An initial population consisting of 100 members, each an 8-character string composed of random combinations of characters ranging from 'A' to 'U' was used for the genetic search. These characters generate latitude increments in the range of -1deg to 1deg with a resolution of 0.1deg. The restricted airspace in the present example consists of latitude locations that are within 0.5 degrees on either side of the nominal flight plan, between the fourth and fifth way-points. This can be translated into an equivalent constraint on the choice of characters that can be employed as the fourth and fifth characters in the chromosomes.

Figure 2 shows a computational flowchart of the trajectory optimization process using the genetic search technique. In order to demonstrate the power of the genetic search methodology in finding solutions using poor initial guesses, every member of the initial population has been deliberately chosen to violate the constraint. A Fixed Length Crossover operation is conducted on chromosomes that are selected from the initial population based on their fitness values. Chromosomes that violate the restricted region constraint are assigned a very high fitness value to decrease their chances of participating in the crossover operation.

The crossover operation selects a randomly generated point along the string and exchanges the portion of the string to the right of the chosen point. This process creates two new chromosomes. A selection ratio of 5 is chosen for the crossover operation, which implies that a chromosome with the best fitness value is 5 times more likely to chosen for crossover than the chromosome with the poorer fitness value. Each crossover operation corresponds to one generation and the process has to be repeated for a sufficiently large number of generations to find the optimal solution.

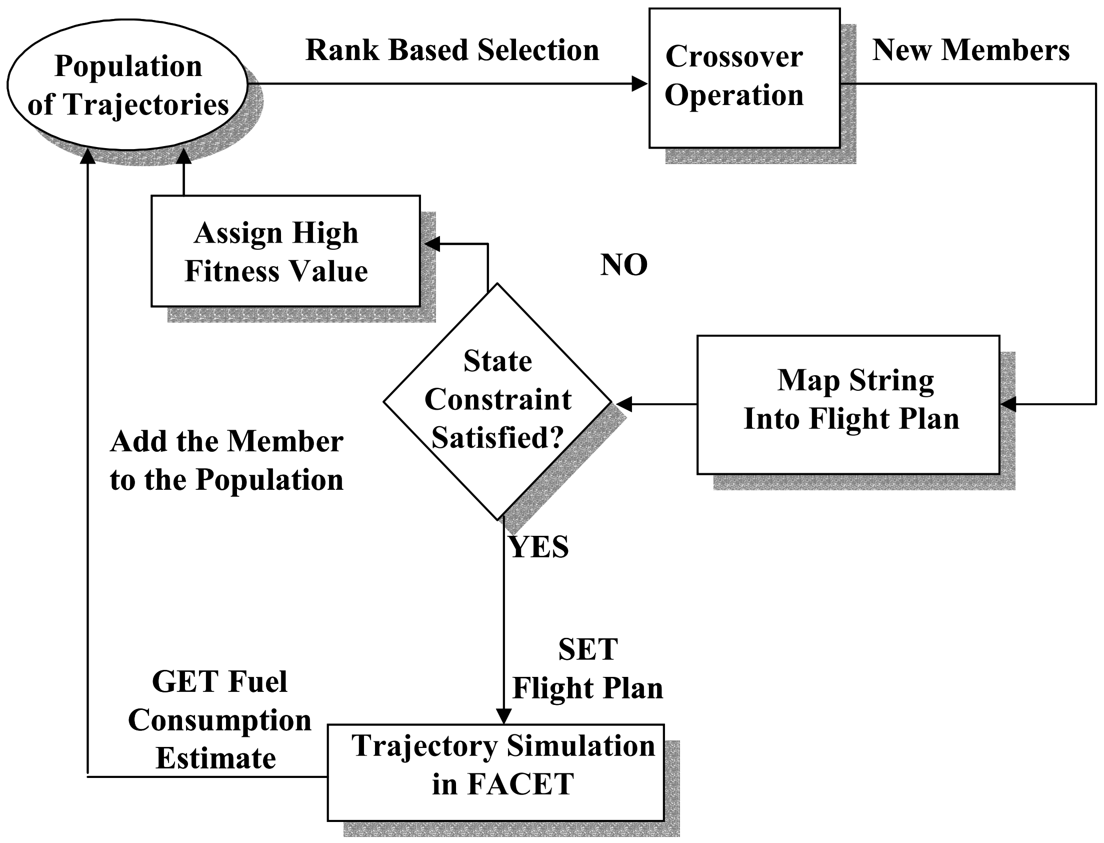

Population of Trajectories
Rank Based Selection
Crossover Operation
New Members
Assign High Fitness Value
NO
State Constraint Satisfied?
Map String Into Flight Plan
Add the Member to the Population
YES
SET Flight Plan
GET Fuel Consumption Estimate
Trajectory Simulation in FACET

Figure 2. Computational Flowchart of the Trajectory Optimization Algorithm Using Genetic Search

The genetic search based trajectory optimization algorithm is used to find a hypothetical optimal flight plan between Seattle and Minneapolis. The result of the optimization run at the end of 1000 generations is shown in Figure 3. This figure shows the best member in the initial population, nominal flight plan, the restricted airspace and the best member in the final population. The left end point of the nominal flight plan represents Seattle and the right end point represents Minneapolis. It can be seen that the best member in the initial population violates the constraint as it passes through the restricted region. The optimal solution obtained at the end of 1000 generations closely matches the nominal flight plan near the departure and arrival airports. However, it takes a detour from the nominal flight plan in the middle to avoid the restricted region by grazing through the boundary of the restricted region.

The evolution of the fitness of the best chromosome in the population with respect to the number of generations is shown in Figure 4. The initial fitness value is very high because all members of the initial population have been chosen to violate the constraint. The fitness value decreases with increasing number of generations. It should be noted that a large number of crossover operations do not produce better offspring thereby causing the fitness value to remain unaltered during those generations.

Monte-Carlo Simulation Evaluation of the Impact of Departure Delays on Arrival Traffic Flow

Monte Carlo simulations are used to investigate statistical relationships between variables of interest in complex dynamical systems. Monte Carlo simulations can provide valuable insights into the dynamics of the national airspace system, thereby helping to formulate sound air traffic management policy. The objective of the present example is to demonstrate the usefulness of the CARAT# software for formulating a Monte-Carlo simulation study.

The formulation of the Monte-Carlo simulation in this section seeks to investigate the effects of departure delays at 15 east coast airports: BOS, ALB, JFK, EWN, LGA, IAD, PHI, PIT, BWI, BDL, DCA, RDU, CHS, ATL, MIA on arrival traffic statistics at the Chicago O'Hare International airport (ORD) over a 24 hour period. The departure delay is modeled as a Poisson random variable with a mean of 3 hours.

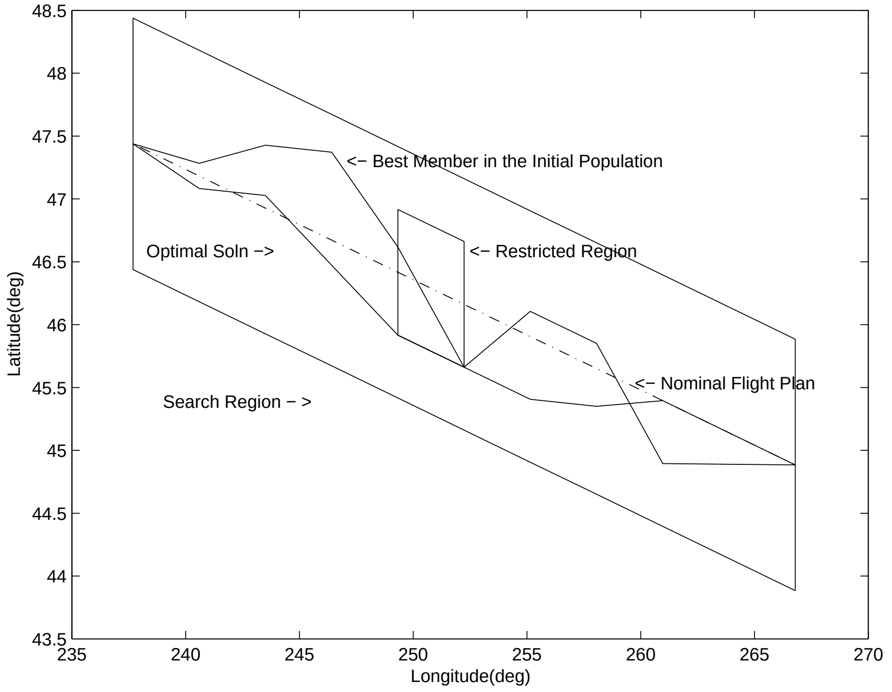


Figure 3. Optimal Trajectory, Together with the Best Member in the Initial Population

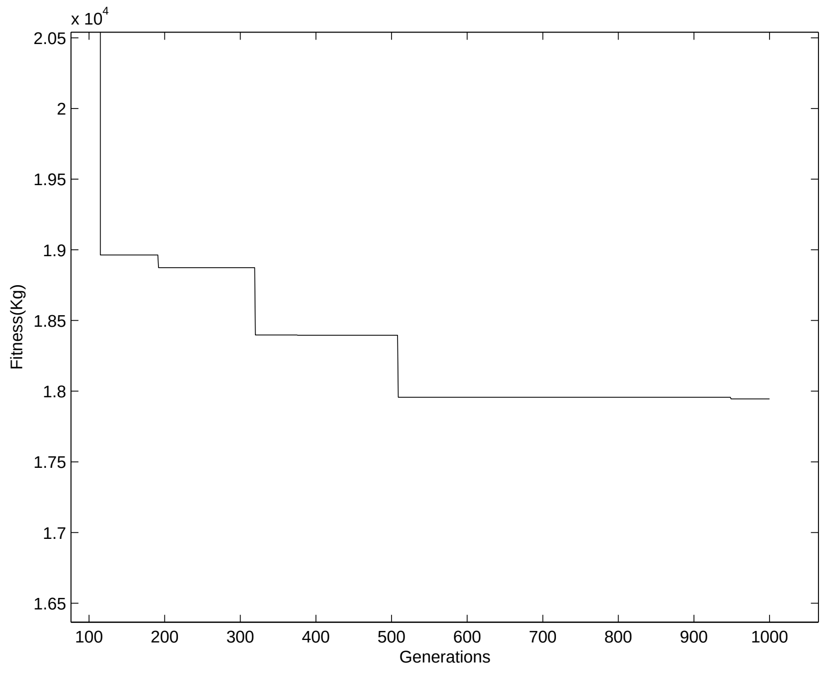


Figure 4. Evolution of the Fitness in the Genetic Trajectory Optimization Process

Upon detection of a departure from any of the 15 airports, a Poisson-distributed random number is generated and the aircraft is delayed by that amount. This process is carried out for traffic evolving over a 24-hour period. At

ORD, the arrival traffic is counted in bins of 2-hour duration, centered at 7 AM, 9 AM, etc. Thus, the 7 AM data represents arrival traffic between 6 AM and 8 AM, 9 AM data represents traffic between 8 AM and 10 AM and so on. At the end of Monte-Carlo simulation runs, the 2-hour aggregate air traffic is averaged.

A computational flowchart of the Monte-Carlo simulation is given in Figure 5. Poisson distributed random numbers were generated using MATLAB random number generator, and CARAT# methods were used to delay aircraft in FACET and to retrieve arrival information. Simulation controls are also implemented using CARAT# methods.

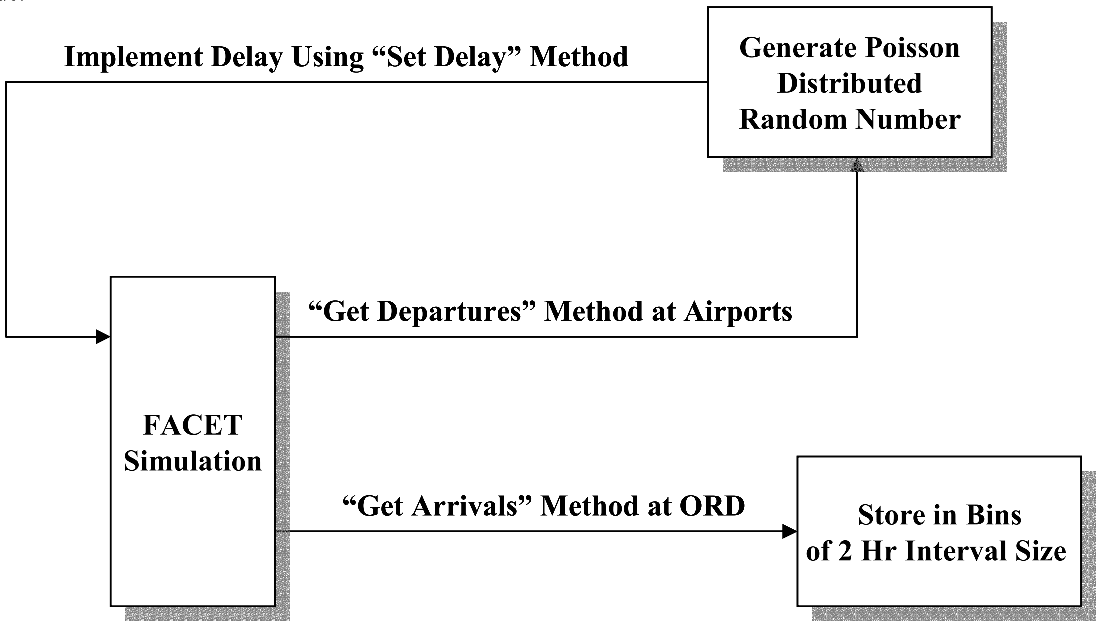

Implement Delay Using "Set Delay" Method
Generate Poisson Distributed Random Number
"Get Departures" Method at Airports
FACET Simulation
"Get Arrivals" Method at ORD
Store in Bins of 2 Hr Interval Size

Figure 5. Computational Flowchart of the Monte-Carlo Simulation using CARAT#

A sample set of results from the Monte-Carlo simulation study are given in Figure 6. The data used in this plot was derived using 125, 24-hour air traffic simulations.

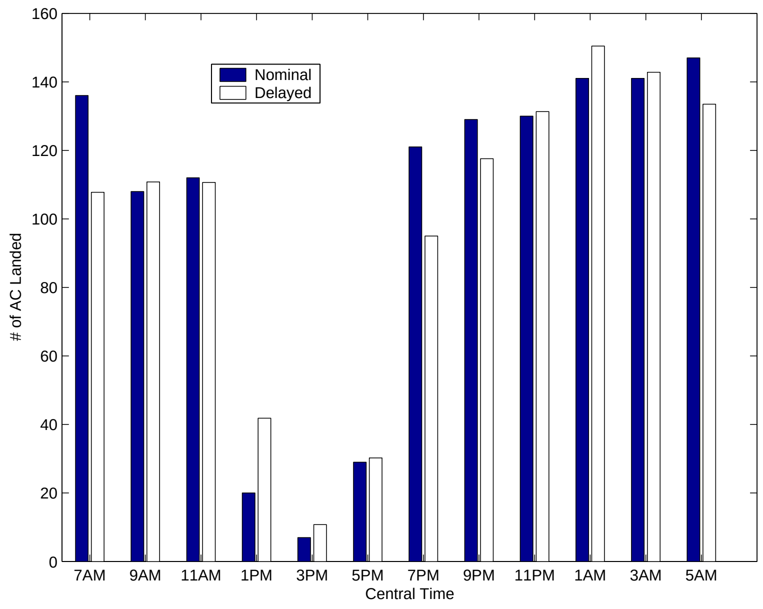


Figure 6. Impact of Departure Delays on Arrival Traffic Flow at ORD, Monte Carlo Simulation Results using 125, 24-hour Traffic Simulations

Figure 6 shows the nominal traffic pattern as well as the delayed traffic pattern. Delayed traffic data from each of the 125 simulations are averaged and presented in this figure. Although the variances were also computed, they are not given here.

It can be observed that a mean delay of 3 hours for aircraft originating from the major east coast airports bound to ORD can significantly reduce the early morning traffic at ORD. Moreover, the peak traffic originally centered on 5 AM is now seen at 1AM. The traffic flows into ORD with departure delays are significantly different from the nominal traffic patterns. Although the present example is rather simplified, it demonstrates how the CARAT# methods can be used to derive useful results. The CARAT# architecture is flexible enough to allow the set up of a wide variety of Monte-Carlo simulations.

A Fuzzy-Logic Controller for Formation Flying

One of the proposals advanced for reducing the complexity of the air traffic management problem in the future is to have the aircraft headed in a particular direction fly in formations at the FAA minimum separation standards. Such operations may make sense to cargo operators and airlines employing hub-spoke type of operations. Moreover, this formalism readily allows the use of robotic aircraft in the controlled airspace. The example discussed in this section illustrates the simulation assessment of a formation flight control system in FACET using the CARAT# software running in MATLAB. As in the previous example, the objective of the present formulation is to illustrate how the FACET functionality may be combined with other software packages using the CARAT# environment. Towards this end, the MATLAB Fuzzy Logic Toolbox[17] is used to derive a fuzzy-logic formation controller.

Automatic formation flight control systems have been of interest in recent literature, motivated primarily by the use of unmanned air vehicles (UAVs) in military operations. The controller development in this paper employs a compact set of fuzzy logic rules to form and maintain formations.

The present example consists of three aircraft departing from San Francisco (SFO), San Jose (SJC) and Oakland (OAK), all bound for the Washington Dulles International airport (IAD). The objective of the controller is to place the three aircraft in an equilateral triangle geometry with a 5000 feet separation between each of them. The aircraft departing from SFO (UAL208) is treated as the leader while the aircraft departing from OAK(UAL209) and SJC(UAL210) are treated as followers. The leader aircraft-UAL208 follows a flight plan available in FACET and the other two aircraft follow the commands generated by a fuzzy logic controller to form and maintain the desired formation. The control commands to the follower aircraft are in the form of airspeed and heading angle settings.

The controllers compute the relative distance between the leader and the followers and transform these distances into the local horizontal plane. The errors between desired and actual values of the relative distances are then used by the fuzzy-logic controller to generate control settings. Using a fuzzifier, a fuzzy inference system, and a defuzzifier, the controllers generate corrections in the longitudinal and lateral directions. These are then transformed into airspeed and heading angle settings for each aircraft. Figure 7 shows a schematic of the closed-loop simulation of the fuzzy formation controller.

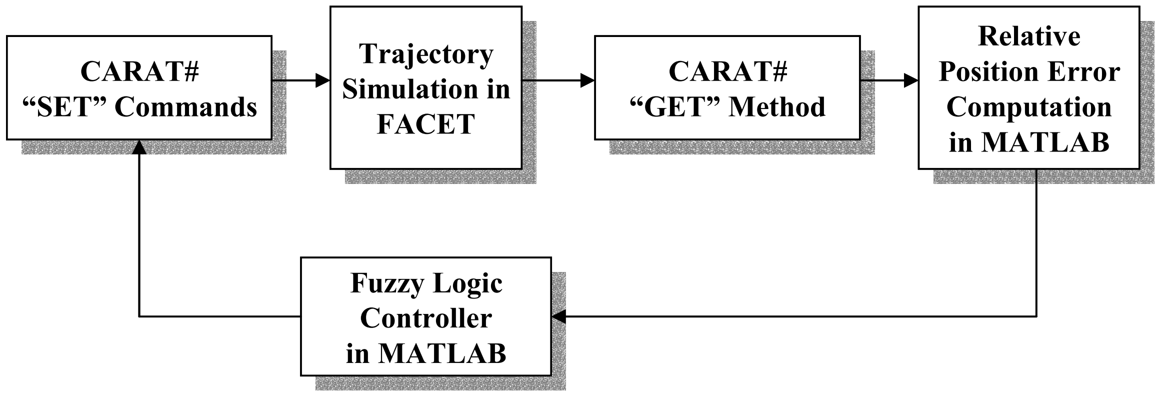

CARAT# "SET" Commands
Trajectory Simulation in FACET
CARAT# "GET" Method
Relative Position Error Computation in MATLAB
Fuzzy Logic Controller in MATLAB

Figure 7. Schematic of the Formation Flying Controller Implementation

The fuzzifier and the defuzzifier employ two fuzzy sets {'SMALL', 'BIG'}, to classify the errors and the corrections respectively. Trapezoidal membership functions are used to characterize the membership values in these two sets. Figure 8 and Figure 9 illustrate the membership functions for the error. The membership functions are chosen to be the same for both longitudinal and lateral errors.

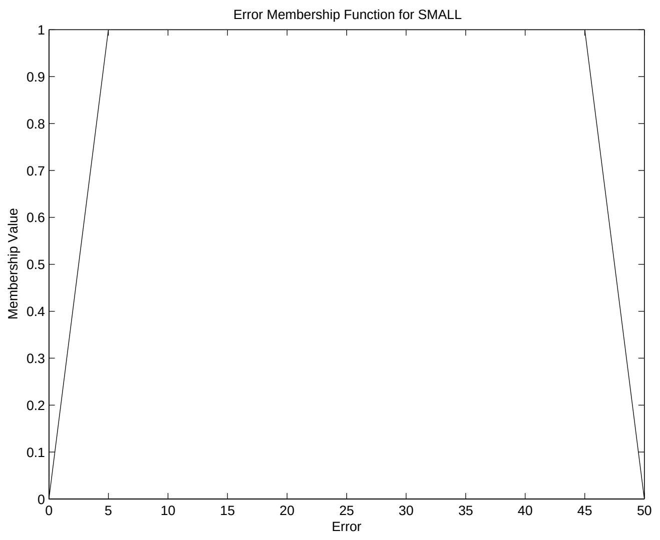


Figure 8. Membership for the Error Set 'SMALL'

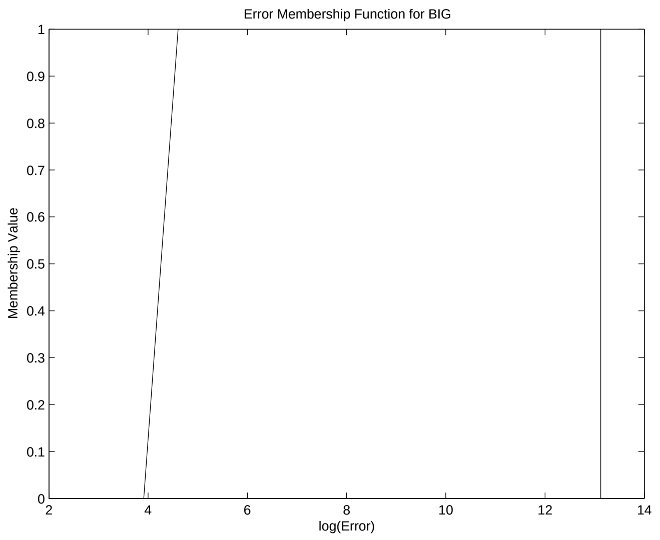


Figure 9. Membership Function for the Error Set 'BIG'

The fuzzy inference system uses the fuzzified tracking errors, together with a set of rules to determine the control settings. The rules used in the present example are:

IF MAGNITUDE OF ERROR ALONG Z DIRECTION IS SMALL THEN MAGNITUDE OF VELOCITY CORRECTION IS SMALL

IF MAGNITUDE OF ERROR ALONG Z DIRECTION IS BIG THEN MAGNITUDE OF VELOCITY CORRECTION IS BIG

IF MAGNITUDE OF ERROR ALONG Y DIRECTION IS SMALL THEN MAGNITUDE OF HEADING ANGLE CORRECTION IS SMALL

IF MAGNITUDE OF ERROR ALONG Y DIRECTION IS BIG THEN MAGNITUDE OF HEADING ANGLE CORRECTION IS BIG

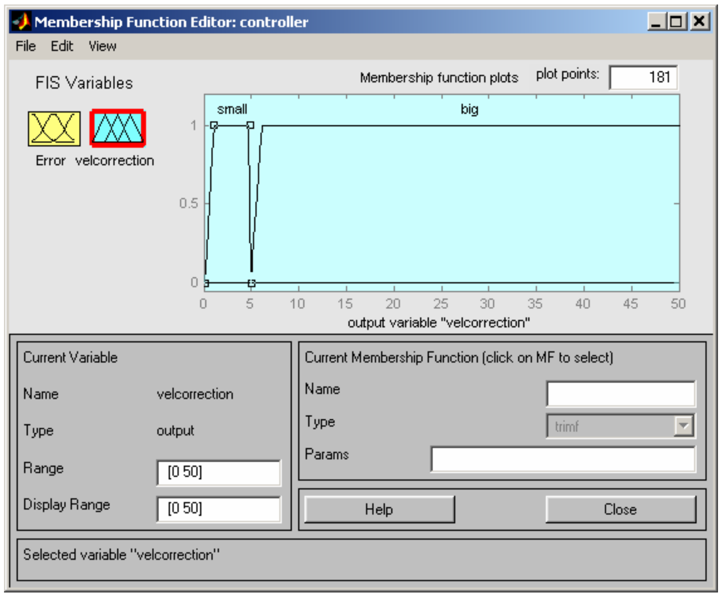
Note that rather naïve set of rules are used in the present example. The Z direction is treated as the longitudinal direction and the Y direction is the lateral direction.

The membership functions used for the control variables are given in Figure 10 and Figure 11. The defuzzification process is based on the centeroid of the area under the membership functions. The sign of the actual control is determined by the sign of the error along the corresponding direction. The airspeed and heading angle corrections generated by the fuzzy-logic controller for different values of the error inputs along the longitudinal and lateral directions are illustrated in Figures Figure 12 and Figure 13.

The Fuzzy Logic Toolbox generates a MATLAB function for implementing the fuzzy logic controller. This function, together with a MATLAB script employing CARAT# methods are used to generate closed-loop simulation results.

A sample simulation result is presented in Figure 14. This figure shows the aircraft trajectories in a leader relative coordinate system. In this frame, the leader aircraft position is always zero. The follower aircraft in the formation, UAL209 and UAL210 takeoff from their respect airports and reach the formation after about 30 minutes. From that point on, they fly in very tight formation to their destination.

The fuzzy logic formation control system presented in this section demonstrated how software packages can be conveniently integrated with FACET functionality in the CARAT# environment. The MATLAB-CARAT# script implementing the closed-loop simulation is very compact and required very little programming expertise to assemble.



Current Variable

Name velcorrection

Type output

Range [0 50]

Display Range [0 50]

Current Membership Function (click on MF to select)

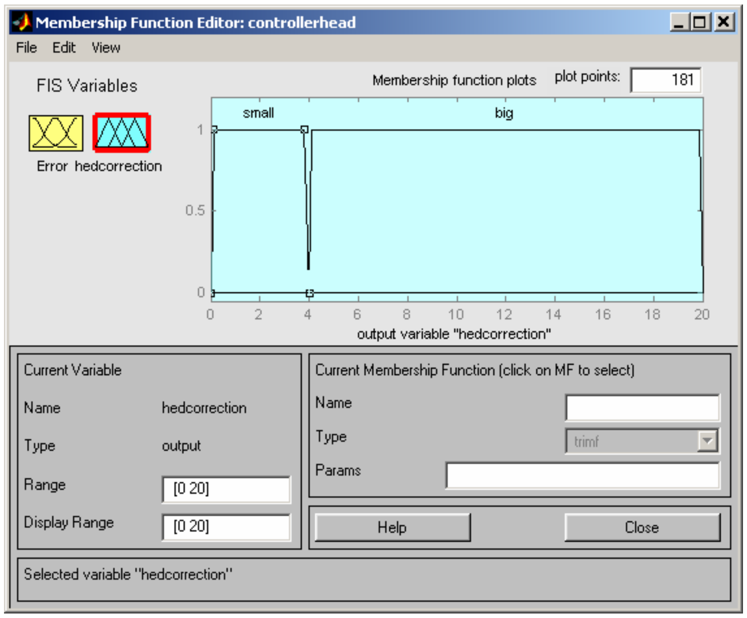
Name

Type trimf

Params

Selected variable "velcorrection"

Figure 10. Membership Functions for Airspeed Correction

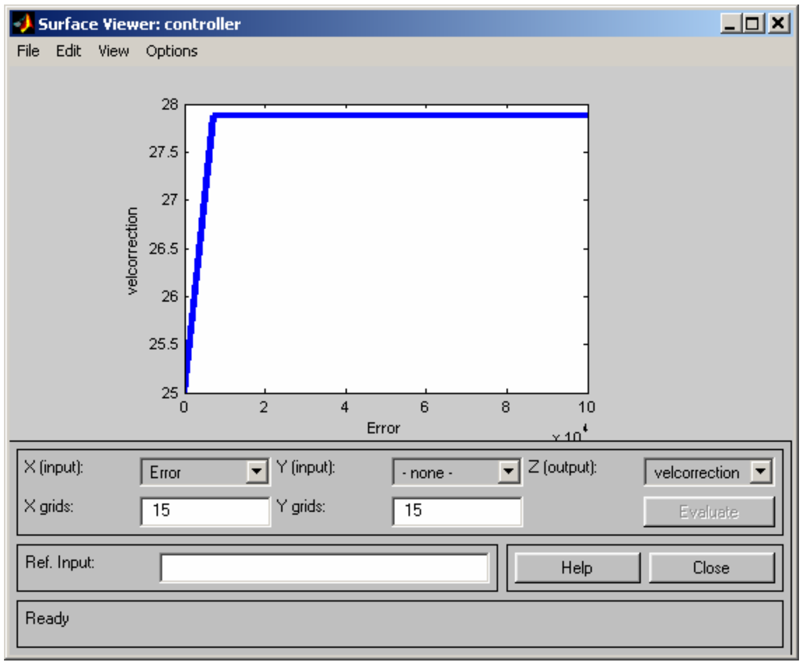

Membership Function Editor: controllerhead
File Edit View
FIS Variables
Error hedcorrection
Membership function plots
plot points: 181
small
big
output variable "hedcorrection"
Help
Close

Current Variable
Name hedcorrection
Type output
Range [0 20]
Display Range [0 20]

Current Membership Function (click on MF to select)
Name
Type trimf
Params

Selected variable "hedcorrection"

Figure 11. Membership Functions for Heading Angle Correction



X (input): Error
Y (input): - none -
Z (output): velcorrection
X grids: 15
Y grids: 15
Ref. Input:
Ready

Figure 12. Airspeed Corrections Generated by the Fuzzy Logic Formation Controller for Different Values of Error

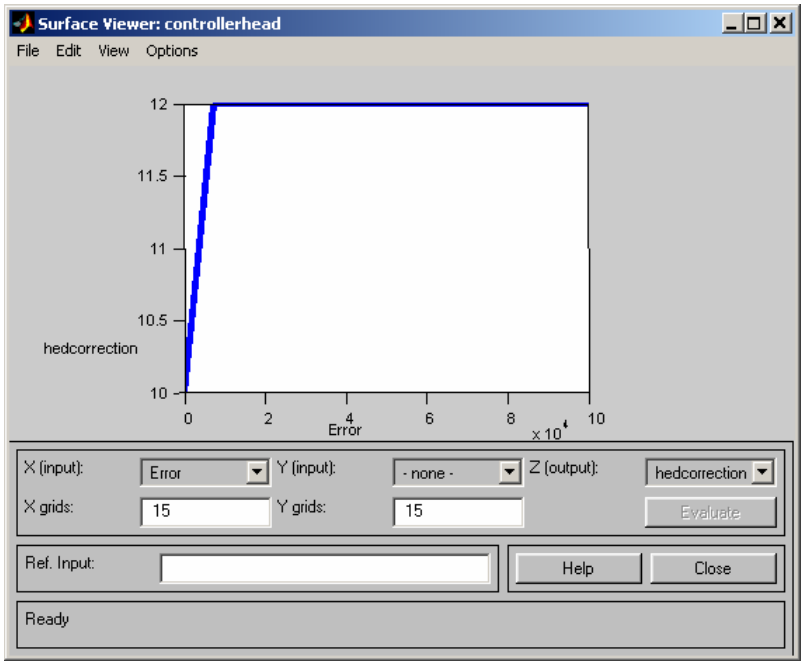


Figure 13. Heading Angle Corrections Generated by the Fuzzy Logic Formation Controller for Different Values of Error

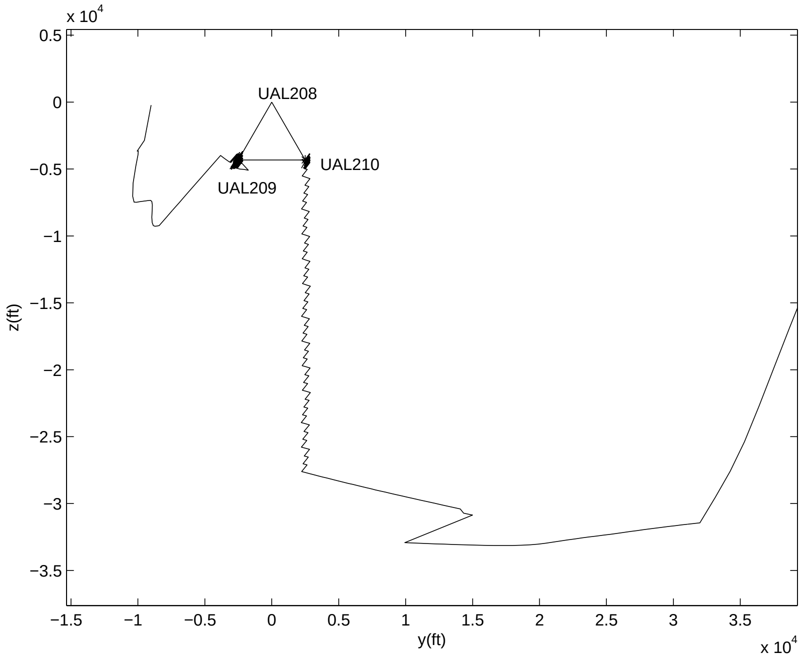


Figure 14. Formation Flying Trajectory in the Leader-Relative Coordinate System

Navigation Data Fusion Using a Kalman Filter

Some of the navigational sensors available onboard aircraft are the inertial navigation system(INS), global positioning system(GPS), VHF omni directional range and distance measuring equipment[18, 19] (VOR-DME). Additionally, locations of aircraft flying in controlled airspace are continuously being monitored by the FAA using tracking radars. These radars determine the two horizontal components of the aircraft position vector, the altitude being provided by the transponder onboard the aircraft. Availability of the Automatic Dependent Surveillance – Broadcast (ADSB) system in the future will enable individual aircraft position data to be made available to all the aircraft. Moreover, future ATM environments may provide additional navigational data to the aircraft.

In this data-rich environment, a central requirement will be the generation of aircraft state estimates that are consistent with all the measurements. It is essential that these state estimates be constructed in a logical manner, because several of the future air traffic management subsystems may use the data for automating aircraft separation and flow control.

This example demonstrates how the numerical linear algebraic capabilities of MATLAB can be used in conjunction with FACET to formulate a data fusion system using the well-known Kalman filtering algorithm[20, 21]. The variables of interest in aircraft navigation are the latitude, longitude, altitude, airspeed, flight-path angle and the heading angle. The present study employs INS, GPS and tracking radar data in the Kalman filter. In order to maintain realism, these sensors are assumed to have different data rates and error characteristics.

The aircraft dynamics used for the development of the Kalman filter consists of the following differential equations:

$$\dot{\lambda} = \frac{V \cos\gamma \cos\chi}{(R+h)\cos\phi} \qquad \dot{\phi} = \frac{V \cos\gamma \sin\chi}{(R+h)} \qquad \dot{h} = V \sin\gamma$$

$$\dot{V} = 0 \qquad \dot{\gamma} = 0 \qquad \dot{\chi} = 0$$

In these equations, λ is the longitude, ϕ is the latitude and h is the altitude. The variable V denotes the airspeed, γ the flight path angle, χ is the heading angle and R is the radius of earth. Note that this model assumes that the aircraft flies at constant speed, constant heading and constant climb rates.

The measurement model consists of:

1. INS Latitude Measurement with bias
2. INS Longitude Measurement with bias
3. INS Altitude Measurement with bias
4. INS Velocity Measurement
5. INS Flight Path Angle Measurement
6. INS Heading Angle Measurement
7. GPS Latitude Measurement
8. GPS Longitude Measurement
9. GPS Altitude Measurement
10. RADAR Latitude Measurement
11. RADAR Longitude Measurement

Assumed error characteristics of these measurements and their data rates are given in Table 1.

	INS	**GPS**	**RADAR**
Data Rate	1sec	2sec	20sec
Latitude Bias	0.01rad		
Longitude Bias	0.01rad		
Altitude Bias	10ft		
Latitude Noise	1e-4rad	2e-4rad	1e-4rad
Longitude Noise	1e-4rad	2e-4rad	1e-4rad
Altitude Noise	3ft	3ft	
Velocity Noise	10ft/s		
Flight Path Angle Noise	1e-4rad		
Heading Angle Noise	1e-4rad		

Table 1. Measurement Data Rates and Error Characteristics

The data fusion algorithm is responsible for estimating the six states of the aircraft, together with biases in the INS provided latitude, longitude and altitude measurements. The sensor measurements are simulated by retrieving the aircraft state vector from FACET using CARAT# methods, and then adding pseudo-random noise generated in MATLAB.

The state transition matrix $\Phi = e^{Fdt}$ is determined using the Jacobian *F* of the nonlinear system dynamics and the sample interval *dt*. At each sample, the Jacobian is calculated at the current estimates of the system states using the equations:

$$F = \begin{bmatrix} 0 & 0 & -\dfrac{Vc\gamma c\chi}{(R+h)^2} & \dfrac{c\gamma c\chi}{(R+h)} & -\dfrac{Vs\gamma c\chi}{(R+h)} & -\dfrac{Vc\gamma s\chi}{(R+h)} & 0 & 0 & 0 \\ \dfrac{Vc\gamma s\chi s\phi}{(R+h)c\phi^2} & 0 & -\dfrac{Vc\gamma s\chi}{(R+h)^2 c\phi} & \dfrac{c\gamma c\chi}{(R+h)c\chi} & -\dfrac{Vs\gamma s\chi}{(R+h)c\chi} & \dfrac{Vc\gamma c\chi}{(R+h)c\chi} & 0 & 0 & 0 \\ 0 & 0 & 0 & s\gamma & Vc\gamma & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The last three rows in this equation correspond to the biases in INS latitude, longitude and altitude measurements. The state transition matrix is used to propagate the state covariance matrix *P(.)* as:

$$P[k+1] = \Phi[k]P[k]\Phi[k] + Q[k]dt$$

Here, *P(k)* is the *9x9* covariance matrix and *Q(k)* is the *9x9* process noise matrix. The initial covariance matrix is chosen as the diagonal matrix:

P(0) = diag([1e-2 1e-2 100 1 1e-4 1e-2 10 1e-4 1e-4 1e-4 1e-4 3]).

The initial condition vector for propagating the model is chosen as follows:

$$\phi(0) = \phi(0)_{actual} + 0.01\text{rad}, \qquad \lambda(0) = \lambda(0)_{actual} + 0.01\text{rad}$$
$$h(0) = h(0)_{actual} + 500\text{ft}, \quad \dot{V}(0) = 0, \quad \dot{\gamma}(0) = 0, \qquad \dot{\chi}(0) = 0$$
$$V(0) = V(0)_{actual} + 10\text{ft/s}, \quad \gamma(0) = \gamma(0)_{actual} + 0.01rad$$
$$\chi(0) = \chi(0)_{actual} + 0.01rad, \qquad \phi_b(0) = 0 \text{ (actual value 0.001rad)}$$
$$\lambda_b(0) = 0, \quad h_b(0) = 0$$

The process noise matrix *Q* is also chosen to be a diagonal matrix with non-zero elements chosen for the velocity, flight-path angle and heading angle equations,

Q = diag([0 0 0 0.0165 1e-5 1.5e-6 1e-2 1e-2 1e-2 0 0 0]).

The state, covariance update equations and the gain computation for the Kalman filter are:

$$\hat{x}(+) = \hat{x}(-) + K[z - h(\hat{x}(-))]$$
$$P(+) = [I - KH(\hat{x}(-))]P(-)$$
$$K = P(-)H^T[HP(-)H^T + R]^{-1}$$

Here, *h* is *11x1* vector measurement model $[\phi \; \lambda \; h \; V \; \gamma \; \chi \; \phi \; \lambda \; h \; \phi \; \lambda]$, *H* is the *11x9* Jacobian matrix of the measurement model, *K* is the Kalman gain and *R* is a *11x11* diagonal matrix consisting of measurement noise given in table 1. The entries of the H matrix can be computed for the INS, GPS, and RADAR measurements as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It should be noted that although the overall measurement model consists of 11 measurements, their data rates are different. Consequently, the update equations would be different at each stage. At most sample instants, only a sub-set of the measurement vector are available and the corresponding H matrix is picked as a sub-matrix of the overall H matrix. A propagation time of 1sec is used for the model, and every 1 sec an INS measurement is conducted, GPS measurement update every 2 sec and RADAR measurement update once in every 20sec. A block diagram for evaluating the Kalman filter data fusion algorithm is given in Figure 15.

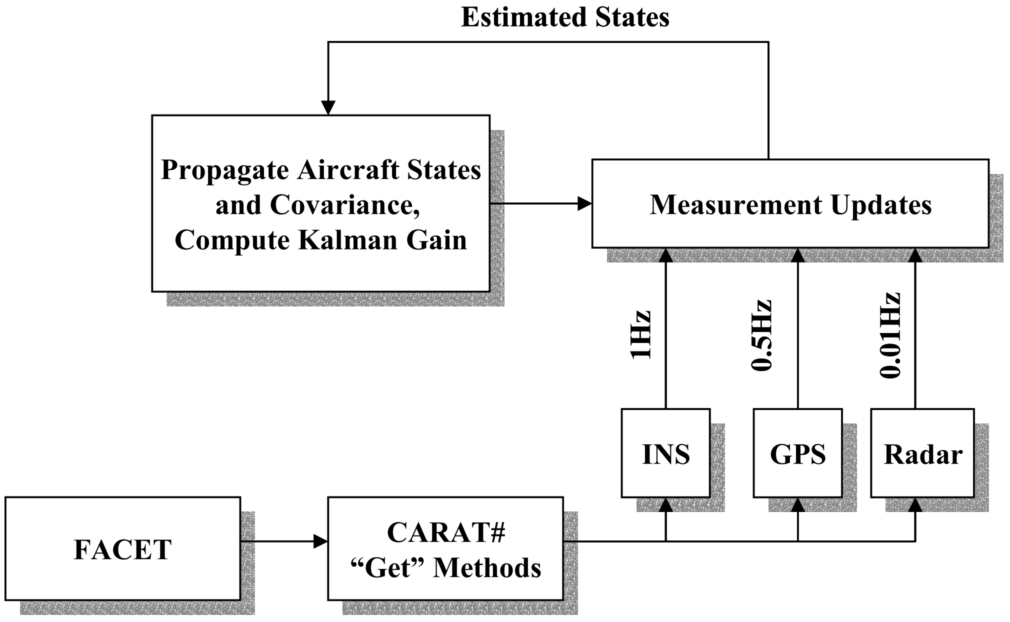


Figure 15. Kalman Filter Data Fusion Algorithm Evaluation

The state vector is sub-sampled to simulate INS, GPS and radar data streams. Each of the data sets are then corrupted using random noise. Both these steps are carried out in MATLAB. The Kalman filter data fusion algorithm is implemented as a MATLAB function.

Figure 16 through Figure 20 show the performance of the data fusion algorithm in estimating the aircraft states and the INS measurement biases. The estimates of latitude and longitude start with large initial condition errors but very quickly settle down to their true values. It can be seen that not only does the estimator overcome initial condition errors but also asymptotically reaches the actual value rejecting the noise in the measurements.

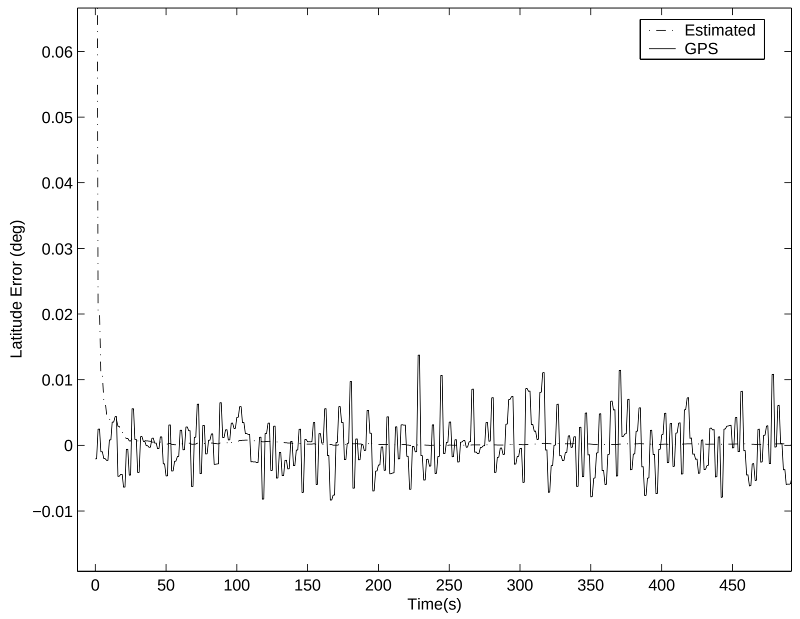


Figure 16. Error in Latitude Estimation

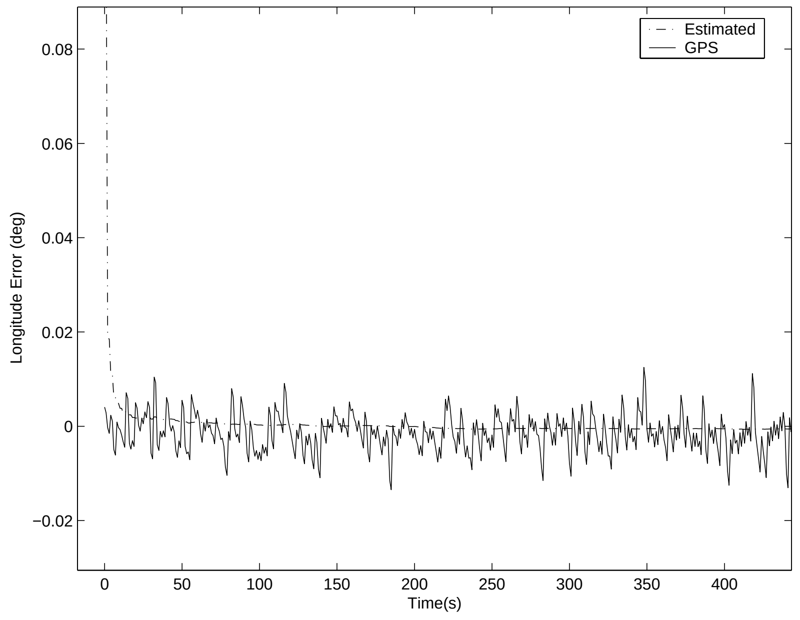


Figure 17. Error in Longitude Estimation

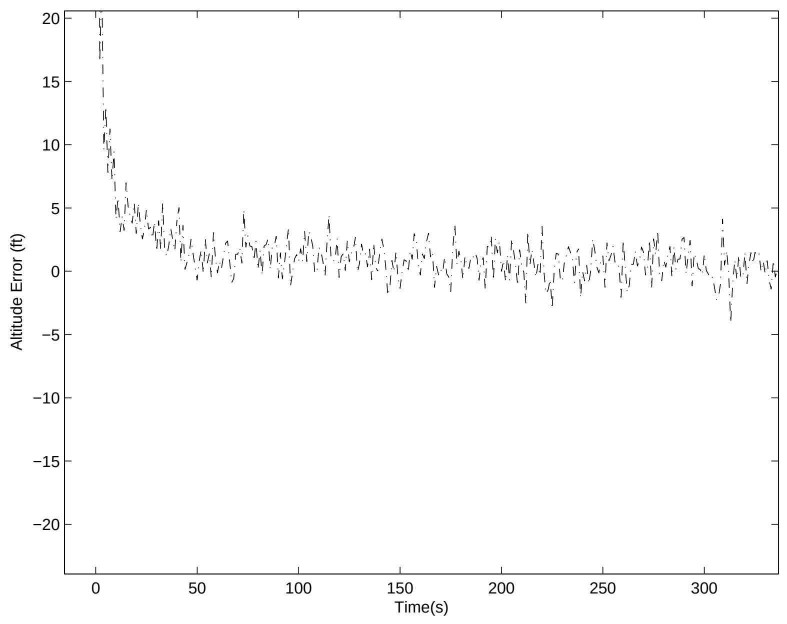


Figure 18. Error in Altitude Estimation

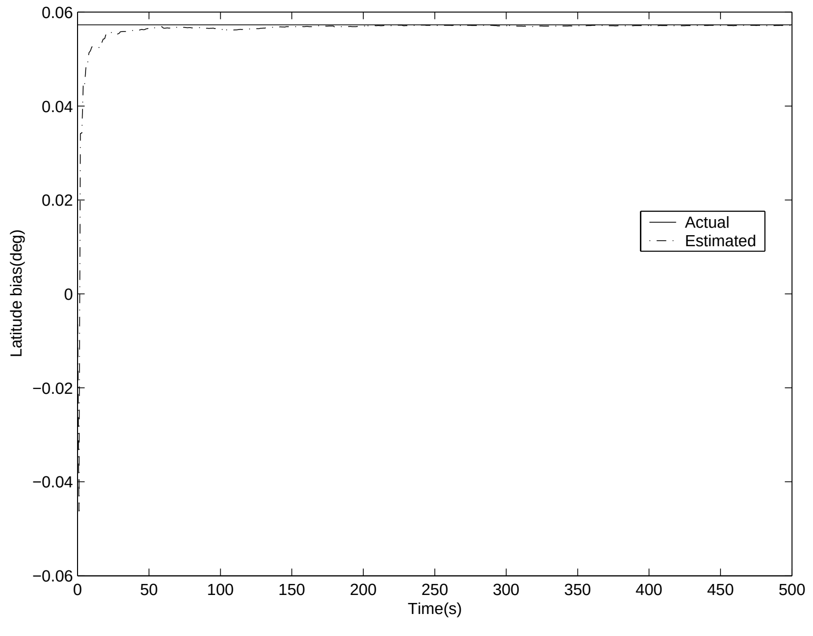


Figure 19. INS Latitude Bias Estimation

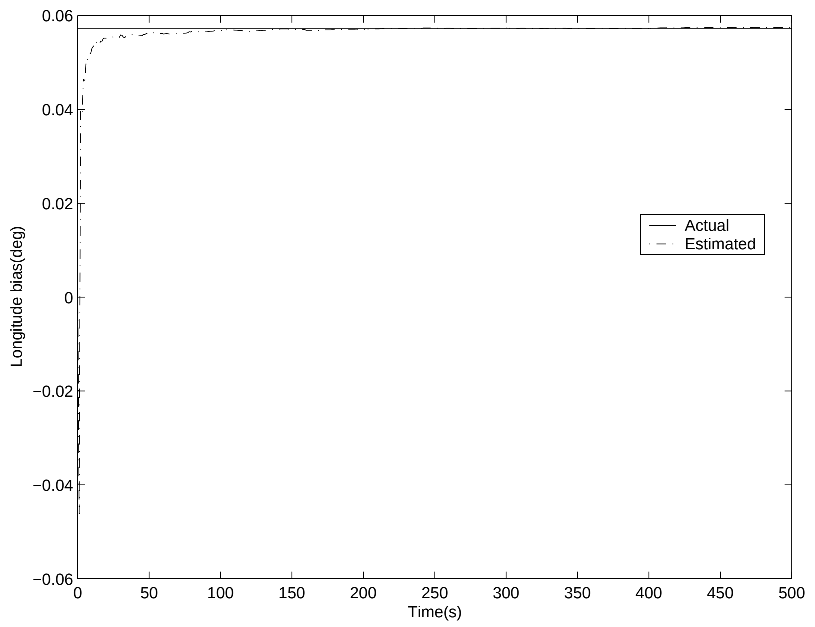


Figure 20. INS Longitude Bias Estimation

The results presented in this section illustrate how the CARAT# software can be used to investigate advanced air traffic management concepts. In the present formulation, FACET was used only for generating measurement data. Note that instead of integrating the equations of motion, a copy of FACET could have been used to propagate the aircraft state vector. Such an approach may find future application in reducing the uncertainty estimates in aircraft departures and en route trajectories by fusing the data available in the NAS. Since the aircraft trajectory data forms the main input for strategic control of the airspace, such data fusion algorithms will be important in future ATM operations. Moreover, since these algorithms produce the state estimates, together with the uncertainty in the predictions, the effect of measurements errors in any strategic decisions can be immediately assessed.

Implementing Conflict Detection and Resolution Algorithms in Java, Jython and MATLAB

In order to illustrate the flexibility of the CARAT#, the Modified Potential Field method[22] for conflict resolution provided in FACET has been re-coded in Java, Jython and MATLAB. A secondary objective of the work discussed in this section is to demonstrate that the CARAT# can be used extend the FACET functionality for formulating future research problems.

The conflict detection and resolution algorithm is ported first to Java. This is accomplished by a simple translation of the FACET module to a Java source file invoking the CARAT# API. Minor changes are made to simplify the original code through decomposition and the addition of a few data structures. After confirming that the new Java conflict resolution algorithm implementation gives identical results to the original FACET implementation, the algorithm is ported from the Java source to both the Jython and MATLAB scripting environments.

The conflict simulation file consists of eight equidistant aircraft converging upon a single point over the Dallas-Fort Worth center. The simulation is run asynchronously with an integration time-step of five seconds. A close visual inspection revealed that in all four implementations, the eight aircraft exhibited identical behavior when resolving conflicts. The FACET "Range Rings" visualization capability[23] was used to confirm that all aircraft were able to successfully avoid coming within the required range of one another. A view of the conflict resolution process is given in Figure 21.

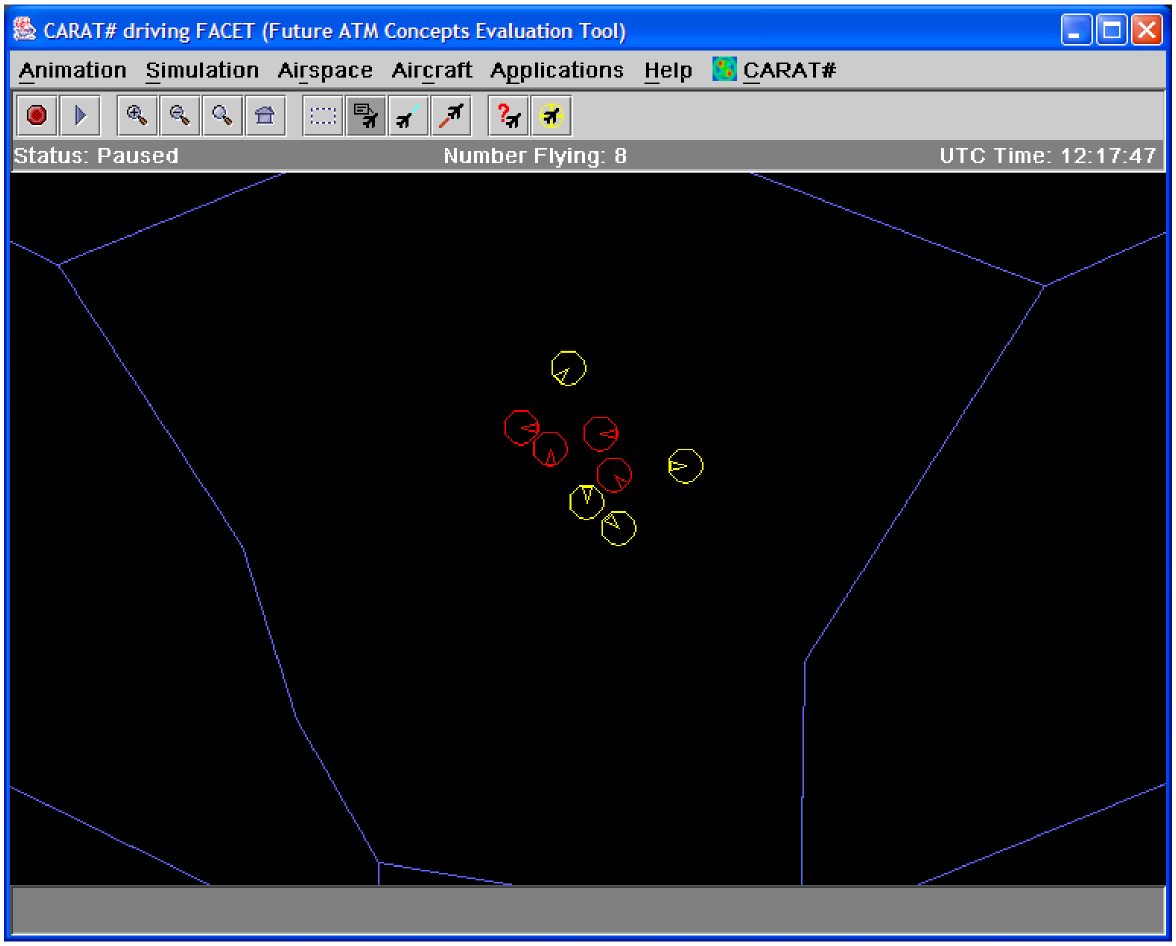


Figure 21. A View of the Conflict Resolution Process, with Aircraft involved in Potential Conflicts Colored Red

Air Traffic Data Visualization Using CARAT#

CARAT# enables access to FACET functionality from scripting environments such as MATLAB and Jython. Powerful graphics and graphing algorithms are available in both of these environments. For instance, MATLAB provides a set of high-level graphing functions. These can be used to display data as polar or rectangular line plots, bar and histogram graphs, contour plots, mesh and surface plots, and animations. The color and shading, axis labeling can all be controlled without requiring the user to manipulate low-level graphics routines.

As an example of the use of MATLAB graphics capabilities for visualizing air traffic data, Figure 22 shows the traffic density at the air route traffic control centers in a 3-D bar graph form. The MATLAB script file for accessing data from FACET using the CARAT# API and to generate this graphic is only 133 lines long. This is a very compact code, considering the complexity of the graphic in Figure 22. The MATLAB file generates a movie of the traffic flows, indicating dynamic evolution of air traffic over the continental United States. Moreover, the viewpoint of this graphic can be dynamically changed during the FACET simulation.

The usefulness of the CARAT# software for formulating interesting research problems based on the FACET engine is apparent from the foregoing sections.

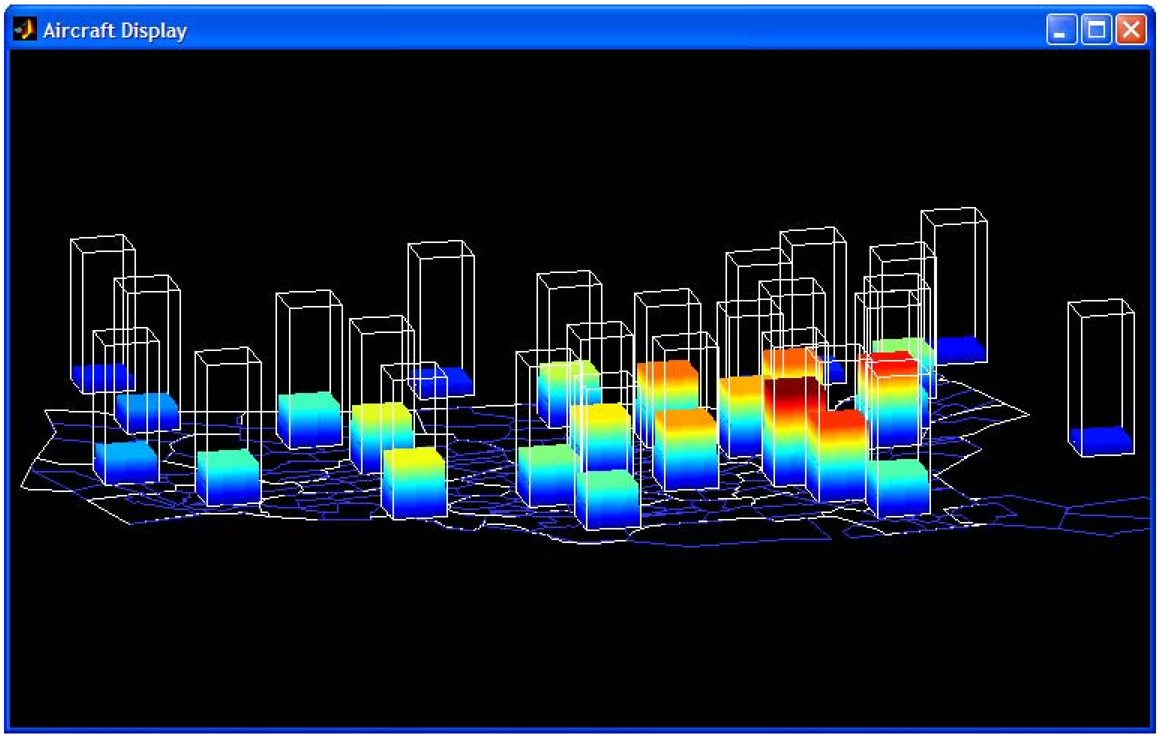


Figure 22. Visualizing Air Traffic Density at Centers Using CARAT# and MATLAB

IV. Conclusions

This paper discussed the development of the CARAT# (Configurable Airspace Research and Analysis Tool – Scriptable) software for accessing the functionality of FACET software through the two different scripting environments. The software provides an extensive set of modeling, simulation and analysis capabilities for studying the National Airspace System. The chief motivation for the present development was to provide simpler access to FACET functionality through interactive scripting languages popular within the research community. CARAT# can help the user formulate and analyze complex research problems with only modest programming skills.

The use of the CARAT# software in air traffic management research was demonstrated through a few research problems. The first considered the aircraft trajectory optimization using genetic search methods and the second illustrated the use of CARAT# software for carrying out Monte Carlo simulation studies. The third example showed the development of a fuzzy logic controller for formation flying, and a fourth example illustrated the development of a data fusion concept for aircraft navigation using the Kalman filtering technique. The aircraft trajectory optimization and the fuzzy formation control problems illustrated the use of commercial toolboxes to expand the types of investigations that can be conducted using FACET along with the CARAT# software. The Monte Carlo simulation study and the data fusion problem exploit the robust numerical algorithms for random number generation and linear algebraic manipulations available in commercial software. The use of the CARAT# software in the Java and Jython environments were illustrated by re-coding one of the conflict resolution algorithms available in FACET and comparing their performance against the original C implementation. Finally, the exploitation of advanced graphics capabilities to visualize air traffic data from FACET using the CARAT# software was demonstrated. These research problems illustrate the versatility of the CARAT# software for conducting future research in the air traffic management area.

Acknowledgments

This research was supported under Contract No. NNA04BA52C from NASA Ames Research Center.

Presented at the AIAA Guidance, Navigation and Control Conference, August 15 -18, 2005, San Francisco, California.

References

[1]Erzberger, H., “CTAS: Integrated Automation Tools for the Center and TRACON”, *The Journal of Air Traffic Control*, Oct. - Dec. 1990, Vol. 32, No. 4, pp. 90 - 91.
[2]Final Report of the RTCA Task Force 3: Free Flight Implementation, RTCA Inc., October 26, 1995.
[3]*Proceedings of the AvSTAR Workshop*, NASA Ames Research Center, March 13-15, 2001.
[4]Bilimoria, K. D., Sridhar, B., Chatterji, G. B., Sheth, G., and Grabbe, S., “FACET: Future ATM Concepts Evaluation Tool,” *3rd USA/Europe Air Traffic Management R&D Seminar*, Naples, Italy, June 2000.
[5]Anon, “MATLAB® User’s Manual,” The MathWorks, Inc., Natick, MA, 2003.
[6]Anon, “The MATRIXX® Product Family,” Integrated Systems Inc, Sunnyvale, CA, 1999.
[7] Martelli, A., *Python in a Nutshell*, O’Reilley, Sebastopol, CA, 2002.
[8] Pedroni, S., and Rappin, N., *Jython Essentials*, O’Reilley, Sebastopol, CA, 2002.
[9] Cheng, V. H. L., Diaz, G. M., Lam, T., and Sweriduk, G. D., “Air and Space Interaction Research,” OSI-NASA-01051, Final Report Prepared Under NASA Contract No. NAS2-01069, July 2003.
[10]Jardin, M., “System-Wide Optimization of the NAS: Phase II Concept Self Assessment,” Proceedings of the VAMS Technical Interchange Meeting #4, NASA Ames Research Center, February 10-11, 2004.
[11]Branch, M. A., and Grace, A., *Optimization Toolbox User’s Guide*, The MathWorks, Inc., Natick, MA, 2003.
[12]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, New York, NY, 1981.
[13]Menon, P. K., Cheng, V. H. L., Lam T., Crawford, L. S., Sweriduk, G. D., and Dewell, L. D., *Genetic Search Toolbox*™, Optimal Synthesis Inc, Palo Alto, CA, 2003.
[14]Goldberg, D. E., *Genetic Algorithms*, Reading, MA: Addison-Wesley, 1989.
[15]Fogel, D. B., *Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1995.
[16]Koza, J. R., *Genetic Programming*, Cambridge, MA: The MIT Press, 1992.
[17]Jang, J. S. R., and Gulley, N., *Fuzzy Logic Toolbox*, The MathWorks, Inc., Natick, MA, 2003.
[18]Nolan, M. S., *Fundamentals of Air Traffic Control*, Brooks/Cole, Pacific Grove, CA, 1999.
[19]Kayton, M. and Fried, W. R., *Avionics Navigation Systems*, John Wiley, New York, NY, 1997.
[20]Jazwinski, A., *Stochastic Processes and Filtering Theory*, Academic Press, New York, NY, 1970.
[21]Gelb, A (Editor), *Applied Optimal Estimation*, The M.I.T Press, Cambridge, MA, 1989.
[22]Eby, M. S., “Self-Organizational Approach for Resolving Air Traffic Conflicts,” *Lincoln Lab Journal*, Vol. 7, No. 2, 1992, Boston, MA.
[23]Bilimoria, K. D., Sheth, K., Lee, H. and Grabbe, S., “Performance Evaluation of Airborne Separation Assurance for Free Flight,”*AIAA Guidance, Navigation and Control Conference*, Denver, CO, August 2000. Paper No. 2000-4269.