Published in AIAA Journal of Aerospace Information Systems, vol. 10, no. 5, May 2013

Automated Generation of Air Traffic Encounters for Testing Conflict Resolution Software

Russell A. Paielli[*]

NASA Ames Research Center, Moffett Field, CA 94035

A method was developed to automatically generate simulated air traffic encounters for testing conflict resolution software. A trajectory scripting language was developed to generate simulated trajectories that result in conflicts with a specified geometry. A script was then written to automatically generate a set of scripts written in the trajectory language, each of which generates a conflict with a specified geometry involving two flights. A set of level-flight conflicts was generated by taking permutations of three basic encounter parameters: the path crossing angle, the minimum separation, and the aircraft speeds. A set of non-level-flight conflicts was also generated by varying the altitude difference and the vertical velocity difference at minimum separation. The resulting conflicts were then run with conflict resolution in a fast-time, batch-mode simulation of the resulting maneuvers. Plots of the resulting resolution maneuvers allow the algorithm developer to quickly visualize and verify the reasonableness of the maneuvers for a wide range of encounter geometries. The results can also help to find errors in the conflict resolution algorithm and software.

I. Introduction

In current air traffic systems, human controllers maintain safe separation using radar displays and voice communication with pilots. The projected increase in air traffic in future decades is expected to require automation or partial automation of that separation assurance function. Separation assurance is a complex problem, with many variables and uncertainties, and failure could be disastrous. The technical challenge is to develop an automated or partially automated system that can reliably maintain separation standards even as traffic increases by a factor of two or more. A major part of that challenge is to ensure that the automation algorithms and software are extremely reliable.

The reliability requirements for safety-critical air traffic software are unusually demanding. To get an idea of the scope of the problem, consider that until recently the critical software used in the Host Computer at each of the twenty US Air Route Traffic Control Centers (ARTCCs or "Centers") was originally developed in the 1960s. That software started getting updated only a few years ago with the ERAM (En Route Automation Modernization) project, but the new software is not expected to be fully deployed nationally until 2014, and it will not automate conflict resolution. The software tested in this paper is a research prototype of a subsystem that is ultimately intended to be a small but critical part of the new software at each Center.

Safety-critical air traffic software clearly requires very thorough testing. Software testing can be broadly categorized as static or dynamic. In static testing, the software (usually the source code or byte code) is analyzed without actually running it, whereas in dynamic testing the software is

[*]Aerospace Engineer, Aviation Systems Division, mailstop 210-10; Russ.Paielli@nasa.gov, AIAA Associate Fellow.

tested by running it. Dynamic testing of air traffic software can be further categorized as fast-time and real-time, where the latter is required for testing the interaction with and usability by humans. In general, dynamic testing is simpler to do, but it cannot guarantee correctness. As Dijkstra famously said, "Program testing can be used to show the presence of bugs, but never to show their absence!"[1] Static analysis may have the potential to prove correctness but is still largely in the research phase. It can augment dynamic testing and may eventually be required for safety-critical software, but it is currently limited in its practical capabilities.

Formal methods, a form of advanced static analysis, can prove correctness, but the complexity of the problems to which they have been successfully applied thus far is limited. For example, the formal methods developed by Butler et al.[10–12] for air traffic conflict resolution are based on an assumption of instantaneous turns, which is not realistic for tactical (i.e., short-range) conflict resolution. The advanced static analysis methods developed by Bushnell, Giannakpoulou, et al.[13] are also promising but are currently limited to problems with a state space substantially smaller than what is required for an operational air traffic system.

This paper presents a dynamic testing method that augments several other approaches to dynamic testing of air traffic conflict resolution software. It involves the automated generation of an idealized set of simulated conflicts. These idealized conflicts contain no tracking noise or uncertainty in pilot intent, so they are not as realistic as tests based on actual air traffic data, but they provide a more controlled and comprehensive approach to testing algorithms and software for conflict resolution. These tests can be useful for finding algorithm and software errors. They are also useful for visualizing the resulting resolution maneuvers and verifying their reasonableness for a wide range of encounter geometries, where "reasonableness" basically means that an air traffic controller would find them acceptable. The tests can also be used for tuning the parameters of the resolution algorithms and for programming the logic for deciding whether to maneuver one or both flights involved in a conflict, for example.

The rest of the paper is organized as follows. The following section provides some background on the separation assurance problem for air traffic. The section after that outlines the conflict resolution algorithms used in TSAFE. The next section then describes a trajectory scripting language that was developed to generate the trajectories and the specified conflicts. The next section discusses the generation of a set of conflicts by permuting key conflict parameters, followed by the automatic generation of a script in the trajectory language for each conflict case. The results of simulating the conflicts in fast-time batch mode are then presented, and finally conclusions are given.

II. Background

The minimum separation standard in enroute US airspace is 5 nmi (nautical miles) horizontally or 1,000 ft (feet) vertically. If each flight is thought to be at the center of a circular disk with a diameter of 5 nmi and a height of 1000 ft, this separation standard is equivalent to requiring that the disks not overlap in three-dimensional space. A violation of this standard is called a loss of separation (LOS), and a violation that is officially determined to be the result of an error by an air traffic controller is called an "operational error." Operational errors currently occur at the rate of approximately three per day in the US,[14] and approximately one per week is classified by the Federal Aviation Administration (FAA) as "severe." The challenge is to develop an automated or partially automated system that can substantially reduce those errors even as air traffic density increases substantially in future decades.

To address that challenge, researchers at NASA Ames Research Center are developing the Advanced Airspace Concept (AAC).[2,3] AAC comprises two stages of separation assurance, and a third stage is provided by standard airborne collision avoidance systems (ACAS, aka TCAS, the Traffic

Alert and Collision Avoidance System). The first stage is the Strategic Auto-resolver,[4] an advanced "expert system" that attempts to detect and efficiently resolve conflicts up to approximately 10 minutes or more in advance. The second stage is a simpler system called the Tactical Separation-Assured Flight Environment (TSAFE),[5–9] which is intended to backup the Strategic Auto-resolver and handle any conflicts left undetected or unresolved with loss of separation (LOS) predicted to occur within approximately two minutes. If TSAFE fails to resolve a conflict, TCAS is available on larger commercial aircraft to prevent a collision by commanding vertical maneuvers. However, TCAS maneuvers are disruptive and are not highly reliable, so they cannot be depended on to guarantee safety.

Because separation assurance is critical for safety, TSAFE is designed to be relatively simple yet still capable of resolving conflicts with high reliability. The Strategic Auto-resolver will be too complex and not reliable enough to be usable for automated conflict resolution without a backup, and TSAFE is designed to provide that backup. Thus, TSAFE generates relatively simple maneuvers consisting mainly of altitude or heading changes. These maneuvers can be used as controller advisories or could eventually be automatically uplinked to the flight deck. For simplicity, TSAFE does not attempt to return maneuvered flights back to their planned routes after the conflict passes (it leaves that less urgent task for another program). Because the conflicts that TSAFE is intended to resolve are relatively close, however, maneuver delays and basic flight dynamics (e.g., non-instantaneous turns) must be accounted for.

Several testing methods have been previously documented for TSAFE, including the use of archived tracking data for approximately 100 actual operational errors.[5,6] That testing captures many "real-world" effects that are difficult to model in simulation, including lack of knowledge of aircraft weight and lack of reliable knowledge of pilot intent (due in part to the fact that controllers sometimes issue voice clearances without entering them into the system). Another promising approach currently being developed to test TSAFE involves random time-shifting of actual recorded trajectories to simulate uncontrolled traffic using real trajectories. The testing methods presented in this paper augment these methods, focusing more on conflict resolution in the idealized case rather than the heuristics of dealing with real-world uncertainties in conflict prediction. These methods were developed specifically for TSAFE, but they could be adapted for testing any similar program designed for tactical conflict detection and resolution.

III. Conflict Resolution Algorithms

TSAFE can provide resolution maneuver advisories to the controller, or it could eventually uplink the resulting maneuvers directly to the flight deck. The resolution algorithms are presented in earlier papers[6–9] and will be outlined only briefly here. TSAFE generates resolution maneuvers when it predicts that loss of separation will occur within 2 minutes. The resolution maneuvers considered are altitude and heading changes. Altitude maneuvers are usually preferred over heading maneuvers for several reasons if they can resolve the conflict, but the choice between the two is often somewhat arbitrary, and this paper does not focus on the logic for choosing between the two. Speed changes are also a possibility for certain types of conflicts (mainly conflicts between arrivals going to the same meter fix). However, speed changes are often ineffective for the short maneuver times available to TSAFE, and they are not considered in this paper.

A. Heading Maneuver Algorithm

The heading maneuver algorithm used in TSAFE is discussed in earlier papers,[8,9] but it has evolved since those papers were published. A heading maneuver, sometimes referred to as a heading "vector," can actually be given in terms of heading angle (direction of flight relative to the air mass or wind) or course angle (direction of flight relative to the ground, also know as track angle), depending on aircraft class. In the absence of winds, heading angle is equal to course angle. In the presence of winds, converting from one to the other is basic, and the term "heading" will be used in this paper to refer to both options. Winds were not simulated in this study, but TSAFE models winds as a uniform wind field in the vicinity of each conflict.

The algorithm generates a set of candidate heading maneuvers as explained below. It models each candidate heading maneuver starting with a pilot delay followed by a circular turn arc of constant radius to the commanded heading, and the maneuver is completed with a straight segment in the direction of the commanded heading. The pilot delay is 15 seconds by default, and the turn radius is calculated for a coordinated turn at a nominal bank angle of 20 deg by default. The straight segment continues until TSAFE explicitly releases the maneuver to allow the flight to return to its original route or a new route. Each candidate heading maneuver is simulated by the resolution algorithm to predict the resulting minimum separation and added pathlength.

For each of the two flights in conflict, the algorithm tries the options of continuing straight, turning right, or turning left. Those three options applied independently to each of the two flights produces nine permutations or maneuver-pair candidates. (The straight/straight combination is a valid candidate because it is different than following the assigned routes and could actually resolve the conflict in some cases.) For each of those nine candidates, the turning flight(s) are stepped simultaneously in small time steps through the constant-radius turn arc mentioned above until headings are reached for which the flights can go straight again and remain above a specified target value of predicted minimum separation. The default target value of minimum separation is 7 nmi, which includes a buffer of 2 nmi added to the minimum legal requirement of 5 nmi to add robustness to variations in communication delay and pilot delay. If the target value of minimum separation cannot be maintained through the turn (because the flights are already too close), then the resulting predicted minimum separation is recorded. For each candidate that is predicted to successfully maintain the target minimum separation, an estimate is also computed for the total added pathlength resulting from the maneuver or maneuver pair. The added pathlength is an approximate metric for comparison of maneuver candidates. It accounts for the predicted length of the maneuver but not for the return to the planned route.

The nine candidates are then ranked, first by whether or not they are predicted to successfully maintain the target value of minimum separation, then by added pathlength. The algorithm then steps through the ranked list of candidates, checks for secondary conflicts, and selects the first candidate that has no secondary conflict. Barring secondary conflicts, if any of the nine candidates are successful in maintaining the target value of minimum separation, then the successful candidate with the smallest added pathlength is selected. If none of the nine candidates are successful in maintaining the target value of minimum separation, then the one with the largest predicted minimum separation is selected.

To favor candidates that turn only one flight, a small penalty in terms of added pathlength is applied to candidates that turn both flights. Heading vectors are issued as absolute course angles (rounded to the nearest degree), rather than relative turn angles, to avoid ambiguity if the flight is already turning when the maneuver is issued. Another algorithm is also used for dealing with secondary conflicts when necessary, but it will not be discussed in this paper.

B. Altitude Maneuver Algorithm

The altitude maneuver algorithm used in TSAFE is discussed in an earlier paper,[7] but it has also evolved since that paper was published. The algorithm models an altitude maneuver starting with a pilot delay followed by a vertical acceleration or deceleration segment to the nominal climb or descent rate, and the maneuver is completed with a leveloff at the commanded altitude. The pilot delay is 15 seconds by default as before, and the vertical acceleration or deceleration magnitude is 0.1 g by default (where g is gravitational acceleration). The nominal climb or descent rate is determined by a table lookup of the BADA (Base of Aircraft Data)[16] database of aircraft performance models for the given aircraft type and altitude. As before, each candidate maneuver is simulated by the algorithm to predict the resulting minimum separation.

The algorithm tries to find an altitude for each flight that will resolve the conflict while avoiding unnecessary altitude excursions. If a flight is climbing or descending, the algorithm first tries altitudes that are stops along the way to the current assigned altitude. If those can't resolve the conflict, then the range of altitudes candidates is expanded in both directions. A cost function is computed for each altitude candidate based on a weighted sum of (1) the absolute value of the difference between the altitude candidate and the current assigned altitude, (2) the vertical distance of the altitude candidate outside of the interval from the current altitude to the current assigned altitude, (3) a reversal cost to penalize forcing the flight to reverse its current vertical velocity, (4) a climb cost to favor descent over climb, since descending can be done faster, and also the altitude ceiling for each flight is not known accurately by TSAFE. These costs and their weighting factors were developed in previous testing with real data discussed earlier in the paper. The resulting cost of the candidate altitude maneuver for each of the two flights is added. To favor changing only one altitude assignment rather than both, an additional cost or penalty is added if both altitude assignments are changed. The maneuver-pair candidates are then ranked by cost, and the algorithm steps through the ranked list, checks for secondary conflicts, and selects the first maneuver pair that has no secondary conflicts.

IV. Trajectory Scripting Language

A trajectory scripting language was developed to specify and generate simulated reference trajectories and conflicts. This domain-specific language (DSL) was implemented in the Python programming language. It is an internal DSL because scripts written in the language are actual Python scripts (whereas an external DSL is a new language). The language allows the user to specify a series of trajectory segments of various types, such as straight segments, turns, climbs, etc. It also allows the user to conveniently generate an encounter or conflict with prescribed parameters, such as the path crossing angle and the minimum separation. Only a few of the methods available in the trajectory language were used in this work, and they are explained below. (Because the language itself is not the main subject of this paper, the unused methods are not discussed in detail.)

An example of a trajectory script is shown in Listing 1, which is a Python script. The first line imports from a module called "`Flight`," which implements the trajectory scripting language with a Python class of the same name. The rest of the script is explained in the following paragraphs.

The line, "`timing(dt=1*sec)`," sets the simulated surveillance update period to one second, which corresponds to the ADS-B (Automatic Dependent Surveillance – Broadcast) state update rate. The assignment within the argument list is an example of an optional feature in Python called argument passing by keyword. The expression "`1*sec`" in the "`timing`" line is an example of the use of a Python class that represents physical scalars. The scalar class automates unit conversions and prevents operations with inconsistent units. Some of the units used in the `Flight` module include degrees (`deg`), seconds (`sec`), nautical miles (`nmi`), minutes (`Min`), knots (`kn`), and Flight Level (100 feet of altitude, `FL`).

The next line of the script, "`flight1 = Flight(...)`," creates an object called "`flight1`," an instance of the "`Flight`" class. The "`ID`" argument is the flight identification, which is the same as the object name in this case. The "`pos`" argument sets the initial position of the flight in terms of a list containing the x, y, and altitude coordinates. The initial altitude is FL350 (flight level 350: 35,000 feet pressure altitude). The "`course`" and "`gspeed`" arguments set the initial course (and

```
from Flight import *

timing(dt=1*sec) # set surveillance update period to 1 second

flight1 = Flight(ID="flight1", pos=[350*nmi, 260*nmi, 350*FL],
    course=90*deg, gspeed=450*kn, clearAlt=350*FL)

flight2 = Flight(ID="flight2", clearAlt=350*FL) # create flight 2

# set flight 2 state to realize specified conflict with flight 1:

flight2.setEncounter(flight1, gspeed=450*kn, angle=90*deg, tLOS=4*Min,
    dmin=-2*nmi)

# set flightplan routes consistent with current states:

flight1.straightRoute()
flight2.straightRoute()

flight1.steady(time=10*Min) # fly steady for 10 minutes
flight2.steady(time=10*Min)

writeTSAFEfile() # write a standard TSAFE input file
```

Listing 1. Example trajectory script in Python

heading) angle and groundspeed (speed relative to the ground) to 90 deg and 450 kn, respectively, and the "`clearAlt`" argument sets the initial cleared (assigned) altitude to FL350.

Next, the line "`flight2 = Flight(...)`," creates another instance of the "`Flight`" class, but sets only the flight ID and the cleared altitude. The next line, "`flight2.setEncounter(flight1,...)`," produces an encounter with `flight1` according to the specified parameters, which determine the initial state of `flight2`. The "`gspeed`" argument sets the groundspeed of `flight2` to 450 knots. The "`angle`" argument sets the path crossing angle to 90 deg. The "`tLOS`" argument sets the time to loss of separation (LOS) to four minutes. Finally, the "`dmin`" argument sets the (pre-resolution) "signed" minimum separation to –2 nmi. The signed minimum separation is defined as positive if `flight2` passes in front of `flight1`, or negative if it passes behind. The required initial position of flight 2 is computed analytically using basic kinematic equations. The `setEncounter` function also has optional arguments "`altdif`" and "`altdifrate`" to set the altitude and altitude-rate differences at minimum separation, both of which default to zero if not specified. More flights could also be added, but traffic was limited to two flights only for all conflicts simulated in this study.

The line "`flight1.straightRoute()`" sets the flightplan route for `flight1` to a simple, straight path consistent with the current state of the flight. TSAFE determines whether the flight is in conformance with its planned route and, if so, computes a trajectory prediction based on it. The "`straightRoute`" function is simply a convenient way to set the planned route to be exactly consistent with the current state of the flight. Another function called "`setRoute`" allows the user to set the planned route more generally as a series of two-dimensional waypoints. Neither "`straightRoute`" nor "`setRoute`" have any effect on the trajectory itself, which is specified separately and can be inconsistent with the flightplan route (to simulate such an inconsistency in the real world).

The line "`flight1.steady(time=10*Min)`" causes `flight1` to fly steady at its current velocity for 10 minutes, storing the simulated surveillance tracks along the way at the specified rate. This is the first line in the script that actually causes time to elapse. Finally, the line "`writeTSAFEfile()`" writes the simulated surveillance data and controller entries to a standard TSAFE input file[15] for later replay.

The "`Flight`" class has several other methods that are not used in the example script above or in this paper, including methods for setting cleared altitudes, changing course or heading angle, turning by a given angle, climbing and descending, and changing speed. However, they will not be discussed further in this paper since they were not used for this work.

V. Generating Conflicts

A. Level Conflicts

First, a set of steady (constant-velocity) level conflicts was generated in the horizontal plane at an arbitrary altitude of FL350 (35,000 ft pressure altitude). The basic parameters of a steady level conflict are

- time to loss of separation
- path crossing angle
- signed minimum separation
- aircraft speeds

The trajectory script shown in Listing 1 generates a simulated conflict with a time to loss of separation of 4 minutes, a path crossing angle of 90 deg, a signed minimum separation of -2 nmi, and aircraft speeds of 450 kn for both flights. Because no winds are modeled in this study, airspeed is always equal to groundspeed, and the heading angle (direction of flight relative to the air mass) is always equal to the course angle (direction of flight relative to true north, positive clockwise). A Python script was written to automatically generate a script of this form for each of a set of permutations of the basic parameters. Those permutations are discussed in the following paragraphs.

For this study, the time to LOS at which the simulated tracking data starts is arbitrary as long as it exceeds the predicted time to LOS at which TSAFE attempts to resolve a conflict, which is currently two minutes by default. The tracking data was therefore set to start four minutes before LOS for all cases in this paper. Late detection of conflicts can be simulated by simply reducing that time to a smaller value. When TSAFE predicts that a conflict is too close to be successfully resolved (i.e., to maintain the minimum required separation), it tries to maximize the minimum separation. Such cases are interesting and important, but they are not considered in this paper.

The path crossing angle was permuted from 0 to 180 deg in steps of 10 deg, for a total of 19 different values. The signed minimum separation was permuted through five values: 0, ±2, and ±4 nmi. The speed of flight 2 was permuted through three values: 400, 450, and 500 kn, with the speed of flight 1 set to 450 kn, and its course angle was set to an arbitrary value of 90 deg (eastbound) for all cases. The significant point is that the speed of flight 2 was set to values that are slower than, faster than, and equal to the speed of flight 1. The total number of permutations is $19 \times 5 \times 3 = 285$.

However, the cases in which both flights have identical velocities (a path crossing angle of zero, and identical speeds) were automatically eliminated (because separation remains constant, and a conflict cannot develop), leaving a total of 280 cases.

This set of 280 encounters is just an example that will be used for this paper. They provide a representative sampling of all possible encounter geometries in the horizontal plane. The range and step size for each parameter used to generate them is a trade-off between comprehensiveness and manageability. Different values might be appropriate, depending on the objective. If the objective is to aggressively exercise the software and automatically find resolution failures, then a large number of permutations is appropriate, perhaps many thousands. However, if the objective is to visualize the resulting resolution maneuvers and check them for reasonableness, then a huge number of cases may not be appropriate. And if the objective is to run a quick regression test after a software change, then perhaps 20 to 30 cases may be sufficient. This method of generating encounters can be used for any of those objectives. Once a particular set of encounters has been generated, it can be saved and reused many times, of course.

For each of the 280 conflict cases discussed above, a directory is automatically created to store the data and plots for that case. Each directory is of the same standard form, with the same file names (the same form that was used earlier for each operational error case used to test TSAFE). The files include the trajectory script used to generate the conflict, the resulting TSAFE input file, the TSAFE output file (created when the case is run in TSAFE), various files containing reference data, and a sub-directory of more than a dozen plots, samples of which will be presented later in the paper.

The directory for each conflict case was named to show the parameters of the conflict. For example, the directory "`Z:060deg:-2nmi:450:450kn`" is for a conflict with a path crossing angle of 60 deg, a signed minimum separation of –2 nmi, and speeds of 450 kn for both flights. (The "`Z:`" is just a general identifier for all conflict directories, and the leading zeros, as in "060," cause the directories to list in a logical order in Linux.) The directory name for each case appears for reference as the subtitle of the plots shown in the next section.

B. Non-Level Conflicts

In addition to the parameters listed earlier for steady level conflicts, the parameters for steady (constant-velocity) non-level conflicts are

- initial altitude difference between flights
- altitude rates (vertical velocities)

at minimum horizontal separation. A set of steady non-level conflicts was generated by permuting these parameters through a set of values. One flight was maintained level at its cleared altitude of FL350, and the other flight was also cleared to FL350, but it was permuted through a set of altitude rates and altitude differences at minimum horizontal separation. The altitude difference was permuted through nine values: 0, ±200, ±400, ±600, and ±800 ft. The altitude rate was permuted through four values: ±1500 and ±2500 fpm (feet per minute). The total number of permutations was therefore $9 \times 4 = 36$. As before, this set of encounters is just an example that will be used for this paper.

Each of these permutations could have been applied to each of the 280 level permutations discussed earlier, but that would have produced $36 \times 280 = 10,080$ permutations, which was considered excessive. Most of these conflicts will be resolved with a "temporary" or "interim" altitude (a brief leveloff before the destination altitude is reached), so the large number of underlying horizontal encounter permutations is unnecessary. Instead, the underlying horizontal permutations were reduced to three path crossing angles, 0, 90, and 180 deg, with a minimum horizontal separation of

0 nmi, and arbitrary speeds of 400 kn and 500 kn. The total number of permutations was thus $36 \times 3 = 108$.

As explained earlier, a directory was created to store the data and plots for each case. The directory names were the same as those discussed earlier but with an addendum such as "`:-0400ft:+2500fpm`" to identify the altitude and altitude rate parameters.

VI. Maneuver Simulation

When TSAFE predicts a loss of separation (LOS) to occur within two minutes, it attempts to compute a maneuver that will resolve the conflict. In this study, maneuvers generated by TSAFE are simulated by taking control of the maneuvered flight and making it execute the prescribed maneuver. For most simulation tests, the maneuvers are propagated for only approximately four minutes since TSAFE is only designed for tactical conflict resolution.

The simulation of resolution maneuvers in this study is relatively simple. A basic maneuver simulator takes control of the flight by intercepting and altering its surveillance track updates. The simulator imposes a default delay of 15 seconds to model the communication delay and the reaction time of the pilot, then it executes the maneuver as explained below. This delay parameter is based on the assumption that resolution maneuvers are automatically uplinked to the flight deck. For voice clearances, the delay should be slightly larger.

For altitude maneuvers, the simulator changes the vertical speed to a target value at a default acceleration or deceleration rate of 0.1 g until it reaches the target (or cleared) altitude and levels off. The target value of vertical speed is determined by a table lookup of the BADA aircraft performance database developed and provided by Eurocontrol.

For heading maneuvers, the simulator changes the course angle at the turn rate of a coordinated turn with a default bank angle of 20 deg until it reaches the target course angle and flies straight. The simulated turns are arcs of constant radius between straight segments. The turn radius of a coordinated turn is $r = v^2/(g \tan \phi)$, where v is the groundspeed, ϕ is the bank angle, and g is gravitational acceleration.

The maneuver delay, vertical acceleration, and bank angle parameters are based on engineering judgment and observation of traffic data. They are approximations intended to add basic realism to the simulation and are not intended for high-fidelity simulation. Higher-order roll and pitch dynamics are not modeled because in the real world they are insignificant compared to unpredictable variations in pilot delay, bank angle, and climb and descent rates.

VII. Sample Simulation Results

The conflicts described above were run in TSAFE in fast-time batch mode with the maneuver simulator described above. These are merely a sampling of the conflicts that could be generated for testing if TSAFE were being prepared for actual operational use. Resolution performance results and plots of maneuvers for sample cases are presented below, first for level conflicts then for non-level conflicts.

Altitude maneuvers are usually preferred over heading maneuvers for several reasons if they can resolve the conflict, but the choice between the two is often somewhat arbitrary, and this paper does not focus on the logic for choosing between the two types of maneuvers. Instead, TSAFE is run with heading resolution only, then with altitude resolution only. Note that other traffic in the vicinity of a conflict could prevent a vertical resolution, thereby forcing a horizontal resolution, or vice versa.

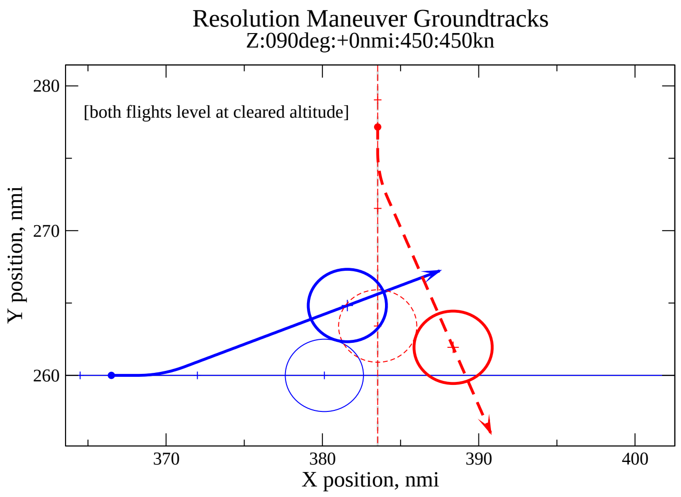

Figure 1. Sample conflict resolved with heading maneuvers

A. Level Conflicts

The 280 simulated level reference conflict cases discussed above were run in TSAFE in batch mode, first with heading maneuvers only, then with altitude maneuvers.

1. Heading Maneuvers Only

All of the 280 generated conflicts discussed earlier were resolved successfully by TSAFE using heading maneuvers, but for 17 of those cases the target separation of 7 nmi was not maintained. All of those 17 cases were for the two smallest path crossing angles of 0 and 10 deg. The smallest minimum separation for all cases was slightly greater than 6 nmi at minimum separation. Heading maneuvers tend to be less effective when the path crossing angle is small, and altitude or speed maneuvers are usually preferable in those cases, but only heading maneuvers were allowed in this particular set of tests. Note that conflict detection is much easier for these simulated cases with idealized, straight trajectories than it is for real air traffic. Successful resolution of all 280 cases is a considered a minimal test of the software, which is useful for regression testing. More importantly, a visual check of the resulting maneuver plots for reasonable maneuvers is very useful.

Each case involves two flights in conflict, each flying level at its cleared altitude and following its planned route exactly. Flight 1 always starts out flying due east at 450 kn, and is represented as the solid line in each of the plots to follow, for example Fig. 1. Flight 2 is represented by the dashed line in each plot. The light circles are 5 nmi in diameter and represent the first point of loss of separation (LOS) had no resolution been attempted. The small "+" symbols (which appear as small ticks on the trajectories) indicate approximate minute markers going back from the point of LOS. Light gray lines represent the flightplan routes for the flight with the corresponding line type (solid or dashed), but they are not visible in these particular plots because they are exactly under the tracks for these idealized simulations. The thick lines represent the resulting simulated maneuver, and the thick circles represent the positions at the resulting time of minimum separation. A more detailed discussion of a few sample conflict cases follows.

Figure 1 shows a conflict with a path crossing angle of 90 deg (one flight heading south and the other heading east), a minimum separation of 0 nmi, and speeds of 450 kn for both flights.

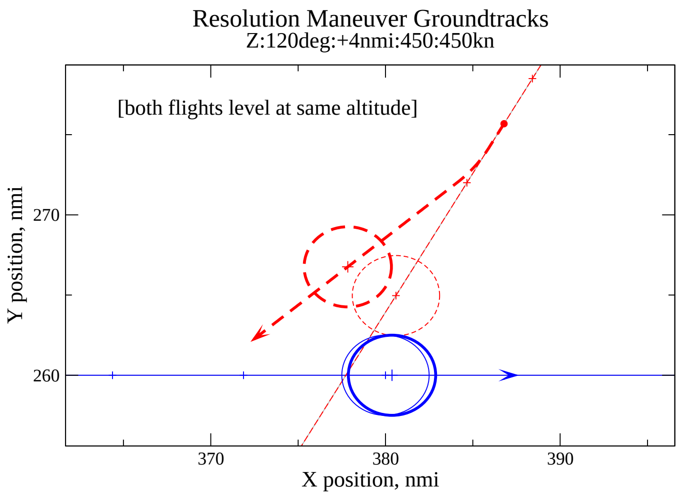

Figure 2. Sample conflict resolved with a single heading maneuver

The conventions explained earlier apply. At –1:51, TSAFE issued a heading vector of 68 deg (a left turn of 22 deg) to turn flight 1 (solid line) behind flight 2 (dashed line), and it simultaneously issued a heading vector of 158 deg to flight 2 (also a left turn of 22 deg). The thick lines represent the resulting simulated maneuver, and the thick circles represent the resulting point of minimum separation. The target minimum separation of 7 nmi was exceeded slightly, as desired.

Figure 2 shows a conflict with a path crossing angle of 120 deg (one flight heading east and the other south-southwest), a minimum separation of 4 nmi, and speeds of 450 kn for both flights. The conventions explained earlier apply. The signed minimum separation is +4 nmi from the perspective of flight 1 (solid line) because it arrives at the path intersection before flight 2 (dashed line). At –1:39, TSAFE issued a heading vector to 229 deg (a right turn of 19 deg) to turn flight 2 behind flight 1, and it issued no maneuver for flight 1. The target minimum separation of 7 nmi was exceeded.

Figure 3 shows another conflict with a path crossing angle of 120 deg, but this time with a signed minimum separation of –4 nmi from the perspective of flight 1, which means that, without resolution, flight 1 would have passed behind flight 2 with a minimum separation of 4 nmi. At –1:50, TSAFE issued a heading vector to 198 deg (a left turn of 12 deg) to flight 2 (dashed line) and a heading vector of 78 deg (also a left turn of 12 deg) to flight 1 (solid line). Again, the target minimum separation of 7 nmi was exceeded.

Figure 4 shows a head-on conflict with a minimum separation of 0 nmi, and speeds of 500 and 450 kn. Flight 1, represented by the solid line, was eastbound, while flight 2, represented by the dashed line, was westbound. The same conventions used for the previous plots apply. As before, the flights are following their planned routes exactly. At –1:51, TSAFE issued a heading vector of 73 deg (a left turn of 17 deg) to flight 1 and a heading vector of 255 deg (a left turn of 15 deg) to flight 2 to successfully resolve the conflict.

Figure 5 shows a shallow-angle conflict with a path crossing angle of 10 deg (one flight heading east and the other east-southeast), a minimum separation of 0 nmi, and speeds of 500 and 450 kn. The same conventions used for the previous plots apply. At –1:51, TSAFE issued a heading of 96 deg (a right turn of 6 deg) to the eastbound flight and a heading of 94 deg (a left turn of 6 deg) to the other flight to essentially split the difference between the two flights. Such non-crossing

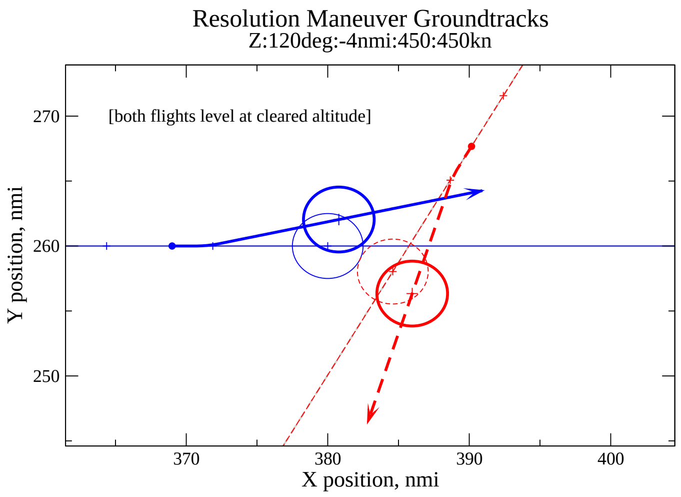

Figure 3. Sample conflict resolved with heading maneuvers

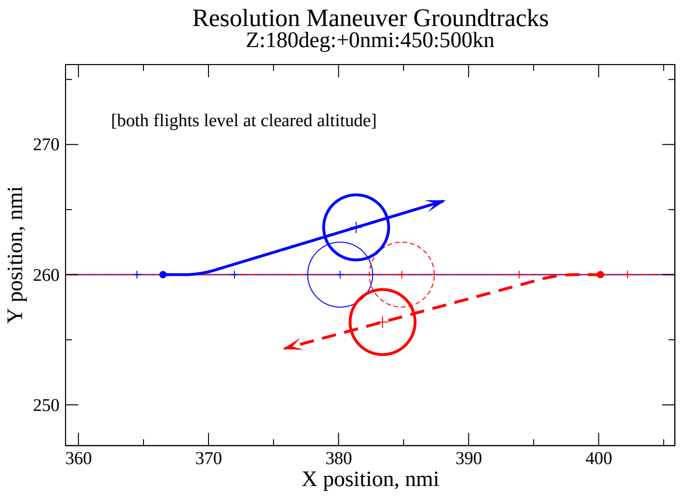

Figure 4. Sample head-on conflict resolved with heading maneuvers

maneuvers are typical for conflicts at shallow crossing angles. They tend to be inefficient, because the encounter is prolonged, and the flights sometimes cannot return to their planned route for an extended period of time. In practice, an altitude maneuver would often be preferred, if possible (i.e., if other traffic does not prevent it), or perhaps even a speed maneuver, but only heading maneuvers were allowed in this particular test.

The preceding tests uncovered some subtle errors in the TSAFE software that might not have been discovered until much later by other tests. For example, one error involved the internal representation of a candidate turn maneuver in the resolution algorithm. A candidate turn maneuver is represented as a list of discrete 4D points (3D position as a function of time) until the end of the

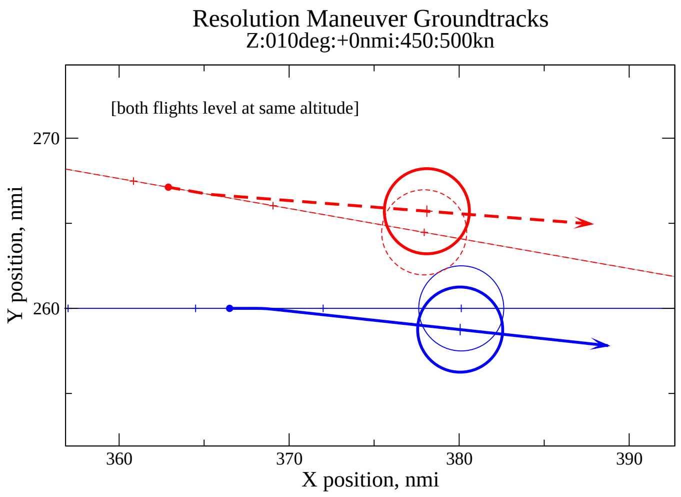

Figure 5. Sample small-angle conflict resolved with a non-crossing heading maneuver

turn arc, but the straight segment following the turn is represented as an initial 4D point and a constant velocity vector (for better computational efficiency). Detailed investigation of unexpected results in a few cases revealed that the initial point of the straight segment was a repeat of the last point of the turn arc, and that repeated point was causing subtle problems with the resolution algorithm in certain cases. After this error was corrected, the results were more reasonable.

2. Altitude Maneuvers Only

The 280 simulated level reference conflict cases discussed above were run in TSAFE with resolution maneuvers restricted to altitude changes only. In all cases, both flights were flying level at FL350 (35,000 ft pressure altitude), as mentioned earlier. All conflicts were resolved successfully by simply descending one flight from FL350 to FL340. The altitude profiles for one example are shown in Fig. 6. The zero reference time is the time of loss of separation without resolution. The horizontal path crossing angle in this case is 90 deg, and the minimum separation is 0 nmi without resolution. At −1:40, flight 2 (dashed line) was descended to FL340, and the thick gray dashed line at FL340 represents the altitude amendment. The thick dashed line descending from FL350 to FL340 represents the simulated maneuver, which resolved the conflict.

These tests revealed a serious error in the TSAFE conflict resolution software. In several cases, the altitude clearances switched within a few seconds of being issued, before any descent even started. The flight that was descended to FL340 was cleared back to FL350, and the other flight was descended to FL340 instead. This unacceptable switching repeated several times. Detailed investigation traced the problem to a $<$ operator that should have been $\leq$ in the algorithm for selecting among candidate altitude maneuvers.

B. Non-Level Conflicts

The 108 simulated non-level reference conflict cases discussed above were run in TSAFE, which resolved all cases successfully with temporary altitudes. A temporary altitude is simply a temporary leveloff at an intermediate altitude on the way to the ultimate cleared altitude. Temporary altitudes are always preferred if they can resolve a conflict, because they are simpler and more efficient than

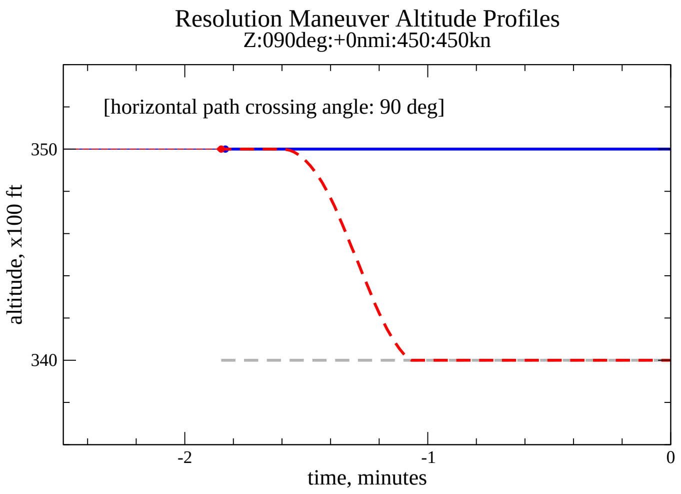

Figure 6. A level conflict resolved with an altitude maneuver

heading maneuvers. Had the allowed maneuvers been restricted to heading maneuvers only (with only two flights involved), they should be identical to what they would be for level flight, the results of which were discussed earlier in the paper.

The altitude profiles for a sample case with a path crossing angle of 90 deg are shown in Fig. 7. Flight 1 (solid line) was flying level at its cleared altitude of FL350. Flight 2 (dashed line) was climbing to its cleared altitude, which was also FL350. Without resolution, flight 2 would have passed 800 ft under flight 1 at the minimum horizontal separation of 0 nmi, with an altitude rate of 1500 fpm. The zero reference time is the time at which separation was lost without resolution. At −1:41, flight 2 was cleared to a temporary altitude of FL340, as represented by the thick dashed gray line, to successfully resolve the conflict.

VIII. Conclusions

A method has been developed to automatically generate a wide variety of simulated air traffic conflicts for testing the TSAFE tactical air traffic conflict resolution software. This testing method complements previous testing methods based on archived tracking data from actual operational errors (losses of separation due to controller error). Although not as realistic as testing with real air traffic data, the methods presented in this paper provide a more controlled and systematic approach to dynamic testing of conflict resolution software. These methods can help to find errors in the conflict resolution algorithms and software, and they are also useful for visualizing and checking reasonableness of resolution maneuvers for a wide variety of conflict geometries. The tests presented in this paper are only a small sampling of the potentially useful tests that could be done with the basic testing approach presented in the paper.

References

[1]Dijkstra, E.W.: "Notes on Structured Programming," T.H.-Report 70-WSK-03, Second Edition, April 1970.

[2]Erzberger, H.: "The Automated Airspace Concept," 4th USA/Europe Air Traffic Management R&D

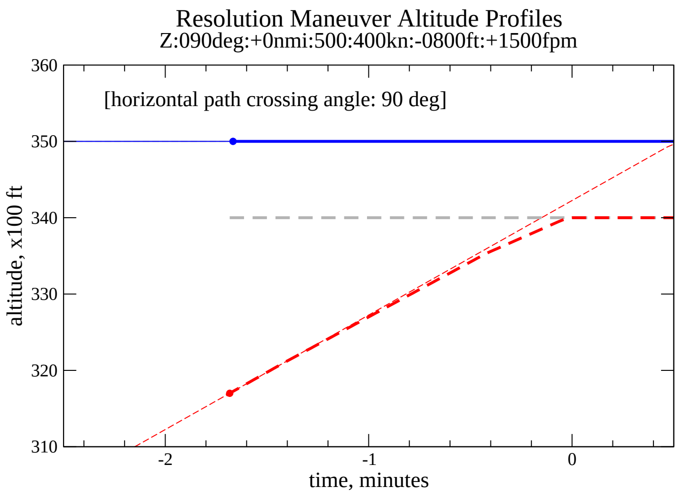

Figure 7. A non-level conflict resolved with a temporary altitude maneuver

Seminar, Santa Fe, NM, USA, 3–7 Dec. 2001.

[3]Erzberger, H.; Paielli, R.A.: "Concept for Next Generation Air Traffic Control System," *Air Traffic Control Quarterly*, Vol. 10(4)(2002), pp 355-378.

[4]Erzberger, H.: "Automated Conflict Resolution for Air Traffic Control," ICAS 2006-8.2.1, 25th International Congress of the Aeronautical Sciences (ICAS), Hamburg, Germany, 3–8 Sep. 2006.

[5]Paielli, R.A.; Erzberger, H.: "Tactical Conflict Detection Methods for Reducing Operational Errors," *Air Traffic Control Quarterly*, Vol. 13(1)(2005).

[6]Paielli, R.A.; Erzberger, H.; Chiu, D.; and Heere, K.R: "Tactical Conflict Alerting Aid for Air Traffic Controllers," AIAA *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 1, Jan-Feb 2009.

[7]Paielli, R.A.: "Tactical Conflict Resolution using Vertical Maneuvers in Enroute Airspace," AIAA *Journal of Aircraft* vol. 45, no. 6, Nov-Dec 2008.

[8]Paielli, R.A.: "Evaluation of Tactical Conflict Resolution Algorithms for Enroute Airspace," AIAA *Journal of Aircraft* vol. 48, no. 1, 2011.

[9]Erzberger, H. and Heere, K.R: "Algorithm and Operational Concept for Resolving Short Range Conflicts," ICAS 2008-8.7.5, International Council of the Aeronautical Sciences (ICAS), Anchorage Alaska, 14-19 Sep. 2008.

[10]Butler, R.; Hagen, G.; Maddalon, J.; Muoz, C.; Narkawicz, A.; and Dowek, G.: "How Formal Methods Impels Discovery: A Short History of an Air Traffic Management Project," Technical Memorandum, NASA/TM-2010-216215, April 2010.

[11]Maddalon, J.; Butler, R.; Muoz, C.; and Dowek, G.: "A Mathematical Analysis of Conflict Prevention Information," AIAA 2009-6907, 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), September 2009. [extended version: NASA TM-2009-215768, June 2009.]

[12]Butler, R.; Muoz, C.: "Formally Verified Practical Algorithms For Recovery From Loss of Separation," Technical Memorandum, NASA TM-2009-215726, June 2009.

[13]Bushnell, D.; Giannakpoulou, D.; Mehlitz, P.; Paielli, R.; Pasareanu, C: "Verification and Validation of Air Traffic Systems: Tactical Separation Assurance," IEEE/AIAA Aerospace Conference, Big Sky, Montana, March 7-14, 2009.

[14]*Administrator's Fact Book*, Federal Aviation Administration, March 2010 (updated periodically and available online at http://www.faa.gov)

[15]Paielli, R.A.: "TSAFE Interface Control Document," NASA/TM–216034, Sept 2012?.

[16]Eurocontrol: *User Manual for the Base of Aircraft Data (BADA)*, Revision 3.6, EEC Note No. 10/04, ACE-C-E2, Eurocontrol Experimental Centre, July 2004.