Evaluation of the Monotonic Lagrangian Grid and Lat-Long Grid for Air Traffic Management

Carolyn Kaplan [1], Johann Dahm[2*], Elaine Oran[3], Natalia Alexandrov[4†], Jay Boris[5],
Naval Research Laboratory, Washington, DC 20375

[*]*University of Michigan, Ann Arbor, MI 48109*
[†]*NASA Langley Research Center, Hampton, VA 23681*

The Air Traffic Monotonic Lagrangian Grid (ATMLG) is used to simulate a 24 hour period of air traffic flow in the National Airspace System (NAS). During this time period, there are 41,594 flights over the United States, and the flight plan information (departure and arrival airports and times, and waypoints along the way) are obtained from an Federal Aviation Administration (FAA) Enhanced Traffic Management System (ETMS) dataset. Two simulation procedures are tested and compared: one based on the Monotonic Lagrangian Grid (MLG), and the other based on the stationary Latitude-Longitude (Lat-Long) grid. Simulating one full day of air traffic over the United States required the following amounts of CPU time on a single processor of an SGI Altix: 88 s for the MLG method, and 163 s for the Lat-Long grid method. We present a discussion of the amount of CPU time required for each of the simulation processes (updating aircraft trajectories, sorting, conflict detection and resolution, etc.), and show that the main advantage of the MLG method is that it is a general sorting algorithm that can sort on multiple properties. We discuss how many MLG neighbors must be considered in the separation assurance procedure in order to ensure a five-mile separation buffer between aircraft, and we investigate the effect of removing waypoints from aircraft trajectories. When aircraft choose their own trajectory, there are more flights with shorter duration times and fewer CD&R maneuvers, resulting in significant fuel savings.

Nomenclature

N_x:	= number of nodes on the x-axis
N_y:	= number of nodes on the y-axis
N_z:	= number of nodes on the z-axis
N:	= total number of nodes in the MLG, where $N = N_x \cdot N_y \cdot N_z$
N_{active}:	= number of active (airborne) aircraft at any one time
N_{holes}:	= number of blank placeholders at any one time
x, y, z:	= physical location of aircraft, in Cartesian coordinates
t:	= time

I. Introduction

DEVELOPING strategies for effective design and control of complex aspects of air transportation requires a fast simulation tool for modeling the air-traffic system. Such a research tool could be used to evaluate and even suggest new system concepts. For example, it could be used to test possible control strategies involved in separation assurance and traffic flow management, as might occur when the system must adjust to local and global weather

[1] Chemical Engineer, Laboratories for Computational Physics and Fluid Dynamics (LCP&FD), Code 6041, AIAA Associate Fellow.
[2] Graduate Student, Department of Aerospace Engineering, AIAA Student Member.
[3] Senior Scientist for Reactive Flow Physics, LCP&FD, Code 6004, AIAA Fellow.
[4] Senior Research Scientist, Aeronautics Systems Analysis Branch, Mail Stop 442, AIAA Associate Fellow.
[5] Chief Scientist, LCP&FD, Code 6003, AIAA Fellow.

perturbations. Specifically, it could be used to determine the most efficient way to reroute air traffic after local conditions, such as thunderstorms, have propagated the efects of a local disturbance throughout the entire system.

To optimize the air traffic system, we must be able to establish relationships between global dynamic performance (e.g., passenger throughput) and local parameters (e.g., localized congestion) of the complex system. Recent work by Alexandrov et al.[1] quantified the dependencies of airport network-performance measures on two metrics related to the network topology. They ran hundreds of simulations on networks with varying amounts of airport clustering; highly clustered networks more closely resembled the hub-and-spoke topology of the current U.S. air transport network, while less clustered networks had a more evenly-spaced distribution. Their optimization study used simplified flight trajectories, and they showed that maximizing the connectivity of the airport network resulted in the best overall dynamic performance of the simulated air traffic system, where the performance measures included average time in air, holding time, number of direct flights between airports, distance traveled and number of planes in queue over airports.

Our ultimate goal is to develop strategies to design and control complex air transportation systems, within the optimization framework[1] discussed above, using actual flight plans and aircraft trajectories. Computing the relationships between global performance and the local parameters of a complex network, especially in systems that include large numbers of interacting autonomous participants, requires massive computational experiments. That is why, despite the availability of high-fidelity air traffic simulators such as FACET[2] and ACES[3], we have developed a very fast simulator, the ATMLG, that can be more easily incorporated into the overall optimization framework[1] discussed above. Compared to FACET and ACES, ATMLG considers significantly fewer details about the system, as its intended use is for numerical experimentation at the conceptual level. This kind of fast tool is necessary to investigate functional relationships that govern the performance of transport networks and that are required to optimize complex transport systems.

In previous papers[4-6], we used ATMLG to simulate air traffic flow in a 3-D volume to gain a better understanding of the conditions under which a volume of airspace is controllable. The simulations kept track of the primary and subsequent conflict resolution maneuvers necessary to maintain a minimum five mile separation distance among all aircraft, and showed that the number of maneuvers increases exponentially with the number of aircraft in the volume. These earlier studies were intended to be proof-of-concept experiments, and they demonstrated that the ATMLG is a promising platform for investigating airspace complexity in the context of maintaining safe aircraft separation.

In this paper, we use ATMLG to simulate air traffic flow throughout one full day in the NAS, and we compare two approaches for locating near-neighbors to maintain adequate separation assurance: the MLG and the Lat-Long Grid. The MLG is a fast nearest-neighbors interaction algorithm, used for dynamic sorting and tracking many interacting objects. The nodes of the MLG represent the locations of individual aircraft, and therefore, when performing conflict detection and resolution (CD&R) maneuvers, adjacent nodes on the MLG structure are checked for relative proximity. The Lat-long grid, which is the more traditional method used in FACET[2] and ACES[3], partitions the airspace volume into fixed stationary grid cells, and aircraft within each cell are considered for CD&R maneuvers. In Section 2, we present the MLG, including a description of the data structure, sorting methods, and the function of active nodes and blank placeholders. Section 3 contains a brief description of the ATMLG numerical method, including the major processes of the simulation procedure, and the amount of CPU time required for each process. The method presented in Section 3 is intended to be an overview; a detailed description of the ATMLG procedure is contained in the Appendix. Results are presented in Section 4, in which we discuss the process of checking near-neighbors to maintain a five mile separation among aircraft, the effect of removing waypoints on flight duration and number of CD&R maneuvers, and we compare the ATMLG approach with a more traditional Lat-Long method. Finally, Section 5 contains the Discussion, including wrap-up information on the pros and cons of each method, and on the generality of the MLG.

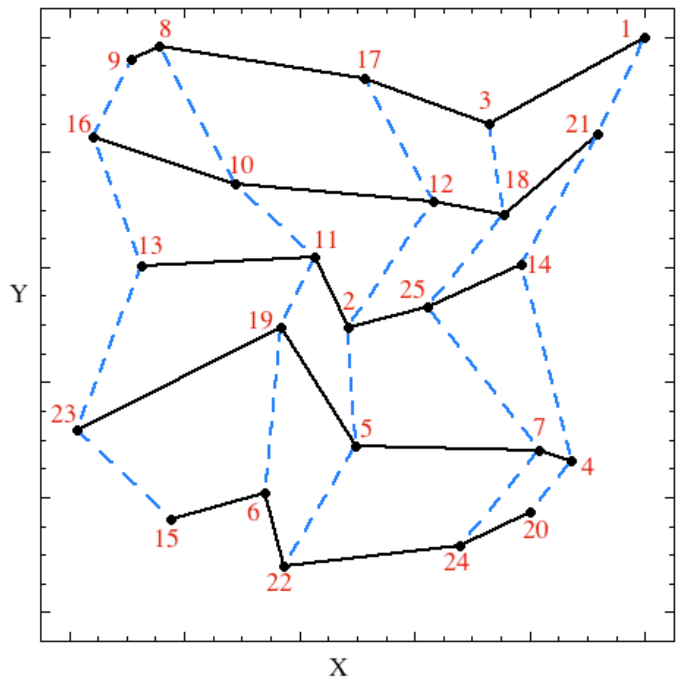
II. Monotonic Lagrangian Grid

The underlying sorting and ordering algorithm and data structure is the MLG,[7-9] a free-Lagrangian data structure for storing the positions and other data needed to describe *N* moving bodies. The MLG has been used for two decades as the underpinning for various particle dynamics simulations, including molecular dynamics,[10] direct simulation Monte Carlo,[11] and exploratory missile defense[12-15] applications. A three-dimensional MLG data structure in Cartesian coordinates is defined by the constraints:

$$\begin{array}{llll} x(i,j,k) \le x(i+1,j,k) & i = 1,....,N_x - 1 & j = 1,....,N_y & k = 1,....,N_z \\ y(i,j,k) \le y(i,j+1,k) & i = 1,....,N_x & j = 1,....,N_y - 1 & k = 1,....,N_z \\ z(i,j,k) \le z(i,j,k+1) & i = 1,....,N_x & j = 1,....,N_y & k = 1,....,N_z - 1 \end{array} \qquad (1)$$

where, N_x, N_y and N_z are the number of objects or nodes in each direction., and $N = N_x \cdot N_y \cdot N_z$ is the total number of objects. Here, the meaning of "object" or "node" depends on the particular application. That is, for molecular dynamics simulations, a node may correspond to an atom; for direct simulation Monte Carlo applications, it corresponds to a group of molecules; for the present application, a node or object corresponds to an aircraft. Moreover, a node may be a dimensionless point or it may correspond to a physical object of any appropriate size and characteristics. The constraints mean that each grid line in each spatial direction is forced to be a monotone index mapping. As an example, Figure 1 depicts a small two-dimensional (5x5) MLG. The MLG model developed here is three-dimensional in space; however we are showing a two-dimensional subset to explain the principles of the MLG. The image on the left shows 25 MLG nodes (points in space) at their irregular spatial physical locations, while the table on the right shows the grid indices (in *i-j* space) of each node. Although the nodes are irregularly spaced (left figure), they are indexed regularly in the MLG by a monotonic mapping between the grid indices and the locations.

A node with three spatial coordinates has three indices in the MLG data arrays. Data relating to each node are stored in computer memory locations identified by these indices. Thus nodes that are close in physical space are always near neighbors in the MLG data arrays. A computer program based on the MLG data structure does not need to check *N-1* possible distances to find which nodes are close to a particular node. Rather, the indices of the neighboring nodes are automatically known because the MLG node indices vary monotonically in all directions with the Lagrangian node coordinates. For example, as shown in Figure 1, we automatically know that near neighbors of node 11 are nodes 16, 10, 12, 13, 2, 23, 19, 5, without having to check the distances of all 24 remaining nodes. The cost of most tracking algorithms using the MLG is dominated by the calculation of the interactions of nodes with their near neighbors, and thus the timing scales as *N*. For applications involving a large number of nodes, such as the air-traffic problem, knowing each node's nearest neighbors automatically, without having to check all *N-1* distances, results in significant computational savings.



5	9	8	17	3	1
4	16	10	12	18	21
3	13	11	2	25	14
2	23	19	5	7	4
1	15	6	22	24	20
J/I	1	2	3	4	5

Figure 1. An example of a two-dimensional MLG. *The figure shows a 2-D MLG (5x5) containing the x- and y-locations of 25 labeled nodes. The solid black (horizontal) lines show the x-links and the dotted blue (vertical) lines show the y-links. The table shows the regular grid indices of the nodes shown in the figure. That is, node-15 is indexed at i = 1, j =1; node-6 is indexed at i =2, j = 1, etc.*

When sorting nodes to MLG order, the simplest MLG algorithm sorts each axis individually. That is, for a 5x5 MLG, it sorts the first five points in the x-direction, then the next five points in the x-direction, etc. After all 25 points have been sorted in the x-direction, it then sorts all 25 points in the y-direction. As each axis becomes sorted, the sorting process may destroy monotonicity in the other axes. Therefore, the sorting process is repeated until all axes are monotonic. The MLG uses two sorting algorithms to put nodes in order. One is a bubble sort, in which each node is compared with the node immediately following it on that axis, and if they are not in monotonic order, their MLG position is switched. This bubble-sorting algorithm is most effective when the nodes are already partially pre-sorted. The other algorithm is a shell sort, in which each node is compared with one that is a half-axis length away, and if the two nodes are not in monotonic order, they are switched. This shell-sorting algorithm is best for completely random data.

During a simulation, we keep the overall dimensions (number of nodes) of the 3-D MLG constant, but the composition of the MLG changes with time. That is, the MLG is composed of active aircraft (planes that are airborne at that current time) and of blank placeholders, or "holes," i.e., $N = N_{active} + N_{holes}$. Since the number of airborne planes varies with time (based on flight plan data in the ETMS file), the composition of the MLG (the relative number of active nodes vs. holes) changes with each timestep. That is, as N_{active} decreases, N_{holes} increases, so that N remains constant. The procedure for swapping holes and active nodes, as aircraft depart and land, is discussed in more detail in Section 3.

In addition to enabling us to keep the number of MLG nodes constant, holes serve an additional purpose: to improve the quality of an MLG. As shown in Figure 2, the addition of holes re-structures the MLG so that adjacent nodes are also nearest-neighbors. Determining the optimum location for the placement of holes to ensure a well-structured MLG is a separate area of research that we are pursuing, and is outside the scope of this paper.

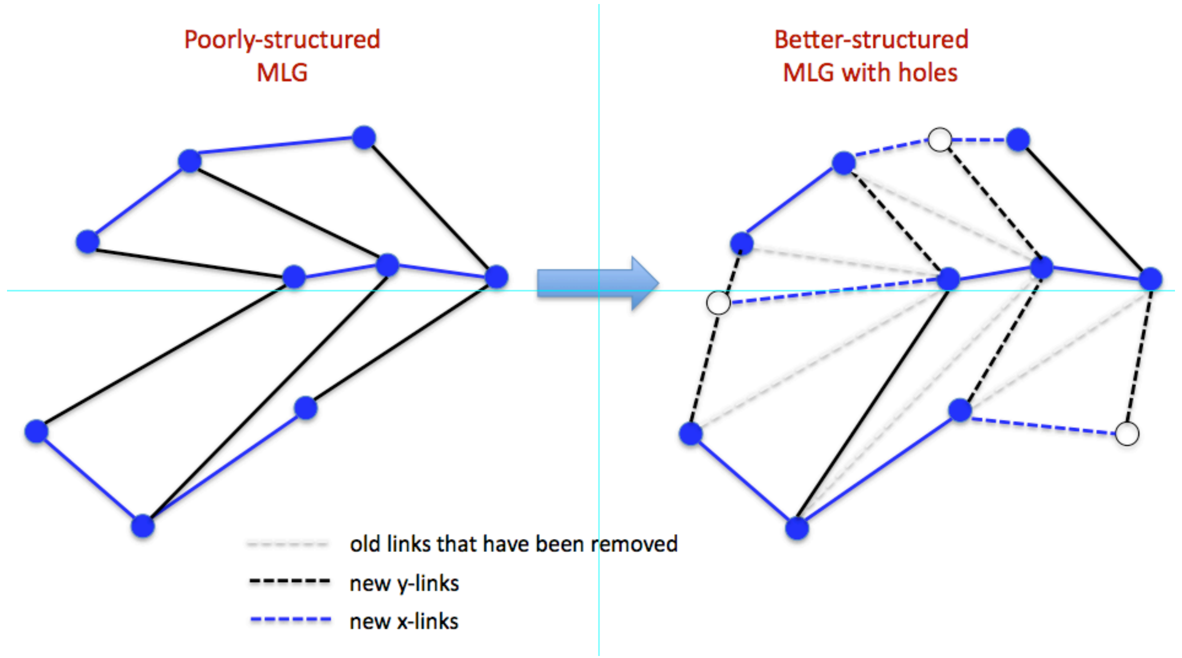

Poorly-structured MLG
Better-structured MLG with holes
old links that have been removed
new y-links
new x-links

Figure 2. The addition of blank placeholders, or "holes," serve to improve the quality of the MLG. *The poorly-structured MLG, on the left, contains some nodes that are adjacent to one another, but that are not nearest-neighbors. By adding holes, as shown on the right, we improve the quality of the MLG, by ensuring that adjacent nodes are also nearest-neighbors.*

III. Numerical Method

In this section, we present a brief description of the ATMLG procedure used to simulate one full day of air traffic flow in the NAS. A detailed description of the procedure is contained in the attached Appendix.

Air traffic data are obtained from an FAA ETMS dataset, containing intended flight plans for 169,016 domestic and international flights, over a 72 hour period of time in September 2006. This dataset is then filtered to contain only the 123,573 domestic flights over three days, as shown by the blue curve in Figure 3. We have chosen to simulate the 24-hour period shown between the red dotted lines in the figure, during which there are a total of 41,594 flights. ETMS data that are used in ATMLG include departure airport and time, arrival airport, cruising altitude, and a set of waypoints for each flight. All waypoints, which are in latitude-longitude format, are converted into Cartesian x- and y-coordinates using a Mercator projection, The z-coordinate of each waypoint is set to the cruising altitude. As the records of the ETMS dataset are in random order, we use the MLG to sort those records in order of flight departure time.

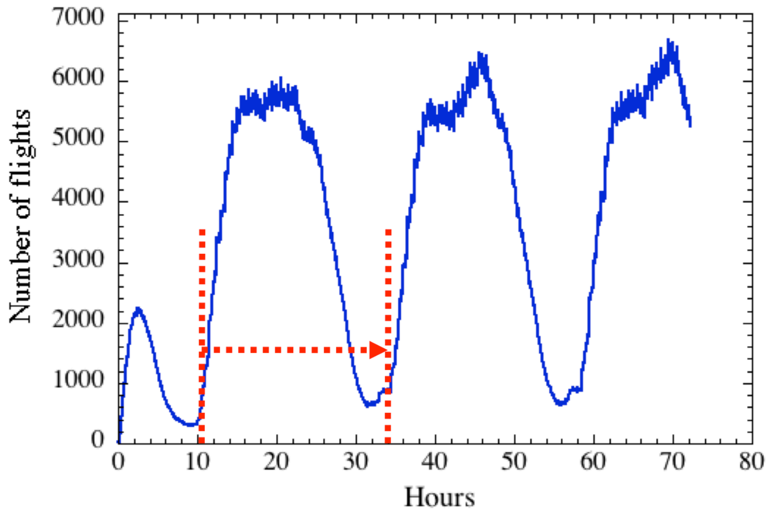


Figure 3. Air traffic volume from an FAA ETMS dataset, over a 72 hour period of time in September 2006. *This dataset was then filtered to contain only domestic flights, resulting in 123,573 flights over three days. The simulations below are based on the 24-hour time period shown between the red dotted lines.*

Although there are total of 41,594 flights during the simulated 24-hr time period, the maximum number of airborne aircraft at any one time is 6045. Therefore, the number of nodes in the MLG is set to 6050 nodes, corresponding to a 55 x 55 x 2 MLG. The total number of nodes, N, equals the number of active aircraft plus the number of holes, that is $N=N_{active}+N_{holes}$. As the number of active aircraft changes at each time step, the composition of the MLG (number of real nodes versus holes) changes accordingly. Therefore, the active nodes represent airborne flights, while all other nodes represent holes. Active nodes move at the airspeed from the ETMS file, while holes remain stationary. Initially, the 6050 node MLG is populated only with holes, which are evenly spaced throughout the computational domain, covering the United States.

The remaining part of the procedure consists of a time step loop, in which we perform the following sequence of steps: At each time step, we compare the simulated current time with the departure time of each flight, and if they match, that flight becomes active. Newly active flights are inserted into the MLG structure, by replacing one of the holes. All MLG nodes (active nodes and holes) are then sorted, based on their x-, y-, and z-location. For the active nodes, we then check the locations of nearest-neighbors on the MLG for relative proximity. If two active nodes are within five miles and are approaching one another, we modify the velocity vector of one aircraft by 20 degrees. Then, the positions of active aircraft are updated, using a velocity vector pointing directly to the next waypoint, or a velocity vector modified by a CD&R maneuver. Flights that reach their arrival airport become newly-inactive, and are converted to holes. Time is then incremented by 10 s, and this sequence of steps is repeated until the end of the simulation.

Calculations are conducted on a single processor of a 1.3 GHz SGI Altix. Figure 4 shows that 88 s of CPU time are required to simulate the full day of air traffic flow in the NAS. This figure shows that the most time-consuming process is for CD&R. Within the CD&R step, it is the process of checking near-neighbors for proximity (rather than making the CD&R maneuvers for aircraft within five miles) that requires most of this CPU time. Figure 4 also shows that only 11 s of CPU time is required for the step to determine active vs. inactive aircraft. It is during this step in which we read the ETMS file and compare the simulated current time with the departure time of each flight. Because we have used the MLG to sort the ETMS dataset in order of departure time, we do not need to check all of the records in the dataset. As will be discussed in Section IV-C, this saves a considerable amount of CPU time.

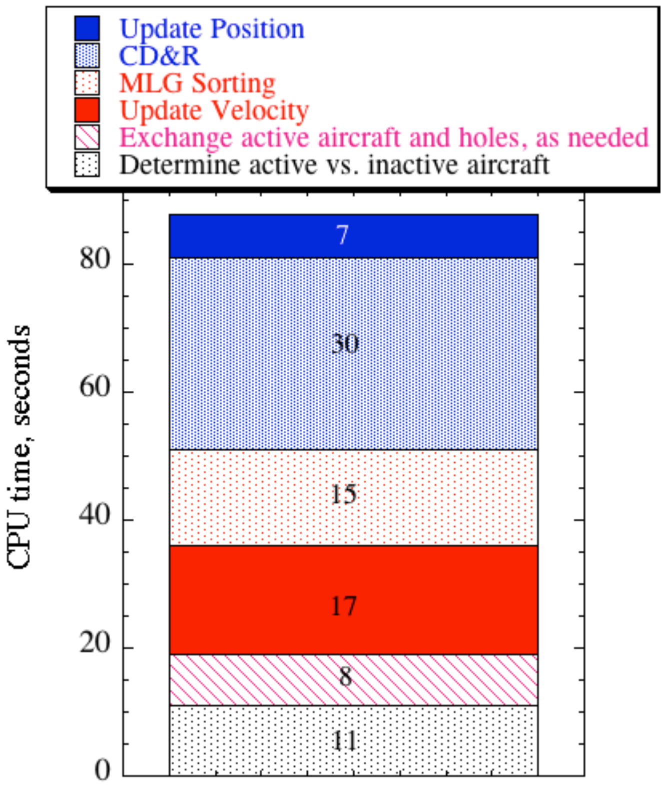


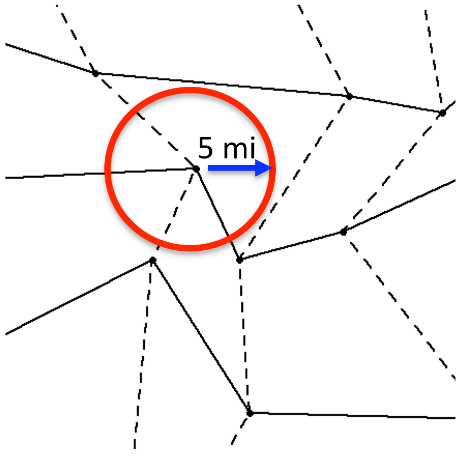
Figure 4. Total CPU time required to simulate one full day in the NAS, using ATMLG, is 88 s. *The CPU times for the individual processes are shown within each section.*

IV. Results

A. Maintaining Five Mile Separation

The current standard FAA practice is to maintain at least a five mile separation between aircraft. The CD&R algorithm, discussed in more detail (in Step 7.6) in the Appendix, involves checking the physical distance between nearby aircraft, and if that distance is less than five miles, and if the aircraft are moving toward each other, then one of the aircraft is re-directed. With the MLG method, the user specifies how many neighbors to check in each direction (rather then specifying a physical distance). Depending on the air-traffic density at any one point in time, however, the distance between MLG nodes may be more or less than five miles. For example, for the sample MLG shown in Figure 5, the five mile radius is shown by the red circle. If we check one node in each direction, we will have checked the full five mile radius. But, if aircraft are spaced more closely, there may be several nodes that are within that five mile radius.

To further investigate this, we tested how many near-neighbor nodes must be checked to measure distances between all aircraft within a five mile radius. Results are shown in Figure 6. To check all neighbors within a five mile radius, 95% of the time we only need to check one node, 4% of the time we need to check two nodes, and less than 1% of the time, we need to check three or four nodes. In the simulations presented in this paper, we have checked two neighboring nodes for each aircraft.



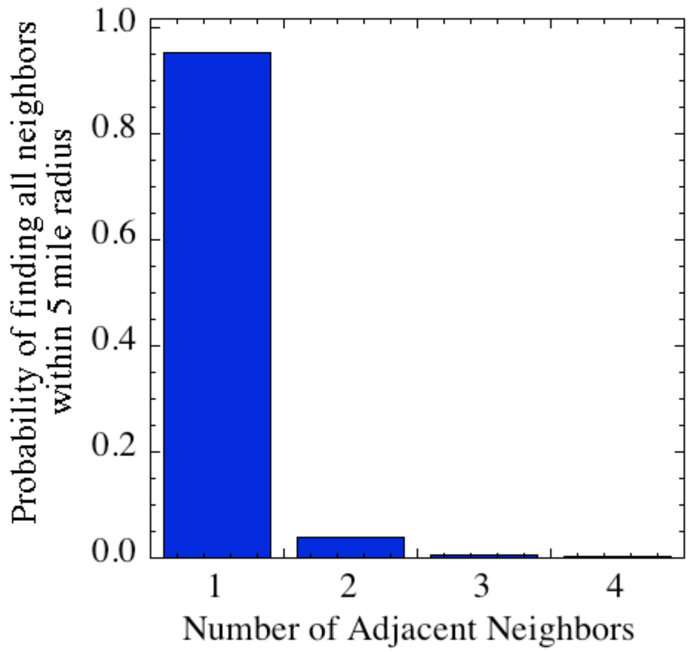


Figure 5 (on left). The current standard FAA practice is to keep aircraft separated by at least five miles. *The distance between MLG nodes may be more or less than five miles, depending on the air-traffic density at any one point in time. Here, the five mile radius is shown by the red circle. If we check one MLG node in each direction, we will have checked the full five mile radius. If aircraft are spaced more closely than this, there may be several MLG nodes that are within that five mile radius.*

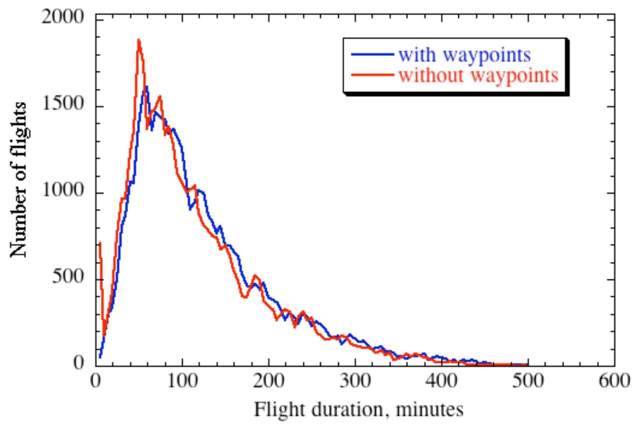
Figure 6 (on right). Checking a sufficient number of MLG nodes to ensure that all neighbors within a certain distance are considered. *To check all neighbors within a five mile radius, 95% of the time we need to check only one node, 4% of the time we need to check two nodes, 0.5% of the time we need to check three nodes, and 0.2% of the time we need to check four nodes.*

B. Effect of Waypoints on Flight Duration and CD&R Maneuvers

Future Air Transportation System may include high-altitude free-flight, with aircraft pilots (rather than Air Traffic Control) choosing their own route and self-separating. In order to quantify how this would affect flight times, we consider a test case in which there are no waypoints; that is, aircraft choose their own trajectories between their departure and arrival airports. In the test case presented here, the aircraft choose a straight path from departure airport to arrival airport, but they may change trajectories for CD&R maneuvers. We simulate this free-flight scenario and compare results with the case in which flights follow the ETMS-specified waypoints.

Figure 7 compares the number of flights as a function of flight duration time, with and without waypoints. At first glance, the two plots appear to be similar, however, the y-axis spans a large range (0 to 2000 flights). Closer examination indicates that when waypoints are removed, there are more flights of a shorter duration, and fewer flights of longer duration. For the case without waypoints, there is a significant reduction in total flight duration, which could result in fuel savings and less passenger delays. This figure also shows that that when waypoints are removed, there are many more flights of *very* short (<5 minutes) duration. As indicated in the ETMS datset, there are approximately 700 flights covering very short distance (e.g., from Orlando Executive airport to Orlando International airport, or from Norfolk airport to Langley AFB). If these flights were flown in a straight line, they would require only a couple of minutes, however, the FAA flight plan, directs them along a circuitous route, so that they take 20 minutes each. The circuitous waypoints could also be due to limitations of airport space.

Figure 8 shows that when waypoints are removed, there are more flights with fewer CD&R maneuvers, and fewer flights with more CD&R maneuvers. It would be intuitive to expect that there are fewer CD&R maneuvers because the flight duration times are shorter. However, our analysis has shown that many flights are directed to the same waypoint, creating more conflicts, which then require more CD&R maneuvers. This is indicated in Figure 9, which shows the number of CD&R maneuvers as a function of physical location over the United States, and indicates that many more CD&R maneuvers are required over commonly-used waypoints. This is by no means a definite conclusion, given the lack of detail in the simulation (e.g., the constraints imposed by the near-airport airspace). However, the result points to an interesting phenomenon that mertis further study, with increased fidelity.



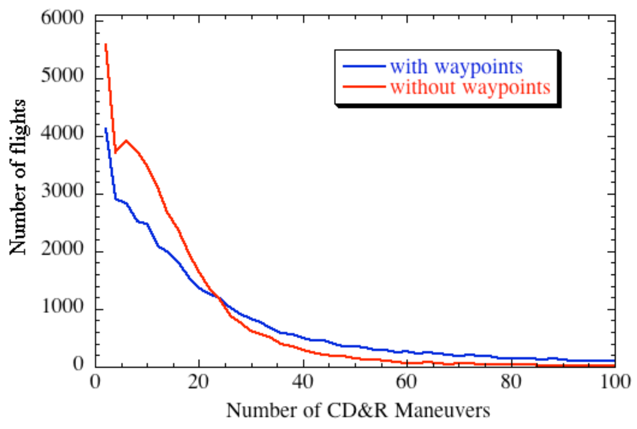


Figure 7 (left). *When waypoints are removed, there are more flights with shorter flight duration times, and fewer flights with longer duration times. The ~700 flights of very short duration time, when waypoints are removed, correspond to short-distance flights, such as from Norfolk to Langley AFB.*

Figure 8 (right). *Removing waypoints results in fewer CD&R maneuvers.*

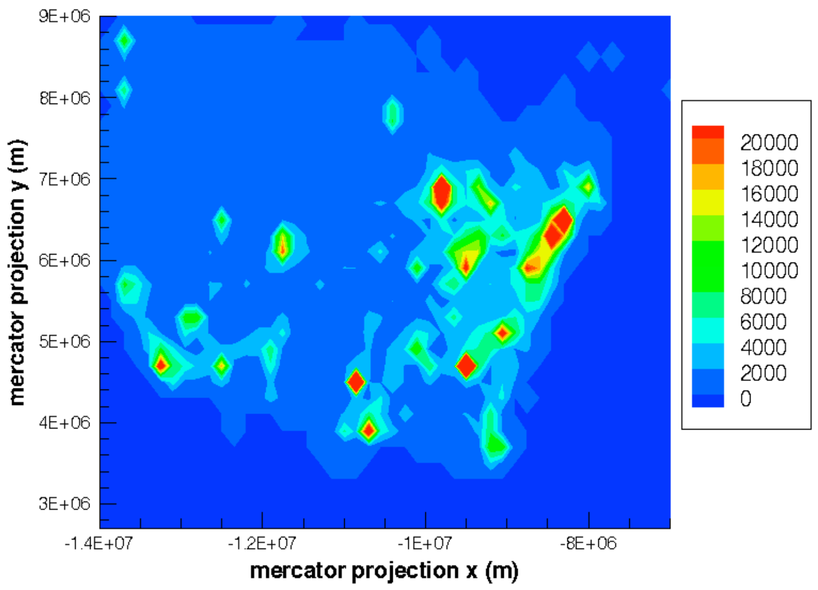


Figure 9. *Number of CD&R maneuvers, as a function of physical location over the United States. Areas where there are many maneuvers correspond to waypoints to which many flights are directed. This suggests that an additional benefit of removing waypoints is a reduction in CD&R maneuvers.*

C. Comparison of Two Approaches: MLG and Lat-Long Grid

In this Section, we compare the MLG with the more traditional approach based on the Lat-Long grid. In ATMLG, the nodes of the MLG represent the locations of individual aircraft. The MLG grid moves, and the physical space covered by the moving grid changes at each timestep. And, in ATMLG, when performing CD&R maneuvers, it is adjacent nodes on the MLG structure that are checked for relative proximity.

In contrast, the Lat-Long grid approach uses a stationary grid which covers the entire airspace volume. This approach partitions the airspace volume into fixed stationary grid cells, and when performing CD&R maneuvers, it is aircraft within each fixed cell that are checked for relative proximity.

To compare the two approaches, we have implemented the Lat-Long grid method in the ATMLG code. The number of lat-long grid cells is set to 55 x 55 x 2, and the grid cell sizes are $\Delta x = 150{,}000$ m, $\Delta y = 200{,}000$ m, and $\Delta z = 9{,}000$ m. This corresponds to a stationary grid of size: $-1.4\text{x}10^7 < x < -5.75\text{x}10^6$ m, $2.7\text{x}10^6 < y < 1.37\text{x}10^7$ m, and $0 < z < 18{,}000$ m. This 3-D box covers the airspace volume over the United States. A more detailed explanation of the Lat-Long procedure can be found in the attached Appendix.

As aircraft follow their trajectories among waypoints, they move among the cells in the stationary grid. Figure 10 shows a snapshot of one z-plane of the lat-long grid at 2000 s. At this time (early in the simulation), there are only ~400 airborne aircraft; therefore, only a small portion of the lat-long grid is populated with aircraft. For those grid cells containing more than one aircraft, the user checks distances among all aircraft within that same grid cell, for potential conflicts.

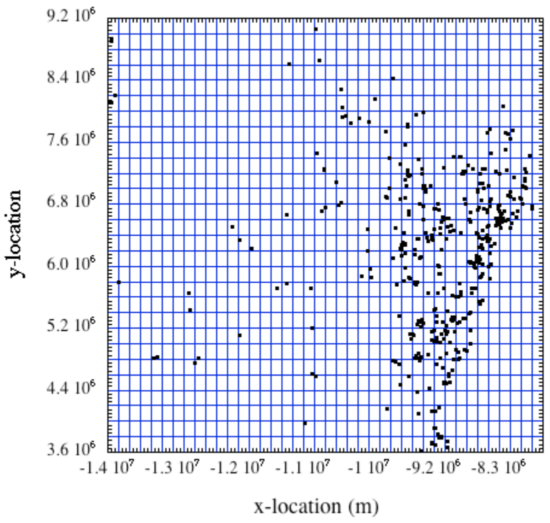


Figure 10. An example Lat-Long grid. *The volume of airspace is divided into bins, and aircraft (represented by dots here) within the same bin are checked for proximity. This snapshot was taken at 2000, when there were only 400 active aircraft. At peak times, there are 6050 aircraft.*

The amount of CPU time required to simulate one full day of air traffic flow in the NAS, for both methods, is shown in Table 1. As indicated, both methods use similar amounts of CPU time for most of the processes simulated. The most significant difference between the two methods involves the process to determine active vs. inactive aircraft. For this process, both the ATMLG and Lat-Long methods need to go through each flight in the ETMS file, every timestep, and determine if the departure time for that flight is during the current simulator timestep. The flights in the ETMS file are in random order; some flights with early departure times are at the end of the file. We have used the MLG to sort the ETMS file in order of departure time; therefore, we do not need to go through the entire ETMS file to find active flights. Once we find flights with later departure times, we can stop searching the ETMS file. Since the Lat-Long grid method only sorts physical location, we do need to search through the entire ETMS file (containaing 123,573 records) to determine flight status, and therefore that process takes considerably more CPU time.

The MLG method includes an additional step (that is not applicable to the Lat-Long method) in which active aircraft are converted to holes when they land, and in which holes are converted to active aircraft when taking off. As shown in Table 1, this additional step in the MLG method adds only 8 s of additional CPU time to the entire simulation.

Process	MLG (s)	Lat-Long (s)
Determine active vs inactive aircraft	11	90
Exchange active aircraft nodes and holes, as needed	8	(n/a)
Update Velocity	17	17
Sorting physical locations of aircraft	15	18
CD&R	30	29
Update Position	7	9
TOTAL	88	163

Table 1. *CPU time to simulate 24-hrs of air traffic flow, corresponding to 41,594 flights, over the United States.*

V. Discussion

We have presented the ATMLG, that can simulate one full day of air traffic in the NAS (41,594 flights), in less than 100 s of CPU time. We have developed this simulator using two approaches to locate aircraft near-neighbors for potential conflicts. One approach is based on the traditionally used Lat-Long grid, which partitions the volume of airspace into stationary grid cells, and checks for potential conflicts among the aircraft within each grid cell. The other approach is based on the Monotonic Lagrangian Grid, in which the moving grid is formed from the locations of aircraft, and potential conflicts are checked by measuring the distance between adjacent nodes on the grid.

The main advantage of the MLG is that it is a general sorting algorithm, and can be used to sort on multiple properties. In the calculations presented here, the MLG is used to sort not only the physical locations of aircraft, but also to sort the ETMS records in order of departure time. The Lat-Long approach only sorts the physical locations of aircraft. The additional sorting by the MLG method resulted in significant savings in computational time. It should be noted that if the Lat-Long method used a pre-sorted ETMS dataset, then it would require similar amounts of CPU time as the MLG method.

Whether we use the Lat-Long grid or the MLG approach, we have developed a very fast simulation tool that can be used to investigate functional relationships that govern the performance of transport networks and that are required to optimize complex transport systems. ATMLG is significantly faster than the traditionally used, high-fidelity air traffic simulators, such as FACET[2] and ACES[3]; this is because ATMLG considers significantly fewer details about the system, and is intended for numerical experimentation at the conceptual level. More details can be added to ATMLG, as needed.

In the discussion on CD&R, we showed that to check all neighboring nodes within a five mile radius, 99% of the time we needed to check only two nodes. A near-term improvement planned for ATMLG is to vary the number of nodes checked, depending on the local air-traffic density. In other words, rather than check a fixed number of nodes, we would check a fixed distance. To check that fixed distance, sometimes we may only need to check one node, while other times we might need to check two or three nodes. Another possible approach is to check on distances based on the velocities of the aircraft in a neighborhood. Checking near-neighbors based on distance (rather than number of nodes) will reduce CPU time (some times we may only need to check one node), and will also ensure that there are no possible near-neighbor misses.

The main objective for creating ATMLG was to develop a fast simulation tool that could be easily incorporated into an optimization framework for design and control of complex aspects of the air transportation system, and these optimization studies will be our next immediate steps.

Acknowledgments

The authors gratefully acknowledge support from NASA Langley Research Center (LaRC) under Interagency Agreement IA1-910 between LaRC and LCP&FD.

Appendix

A detailed explanation of the ATMLG procedure is presented in the sequence of steps below:

1. The inital step is to read in the data from the ETMS file, for each flight. This includes flight ID, departure airport and time, arrival airport and time, cruising altitude and airspeed, physical class, user class, weight class, number of intended waypoints, and a list of those waypoints (latitude and longitude).

2. Use the MLG sorting routines to sort the ETMS records in order of flight departure time.

3. Create 4-D trajectories (*x, y, z, t* at each waypoint) for each flight:

The latitude and longitude waypoints are converted into Cartesian x- and y- coordinates using a Mercator projection, while the Cartesian z-coordinate is assumed to be the specified cruising altitude. The arrival time at each waypoint is calculated from the distance between each waypoint and the cruising airspeed.

4. Set the simulation start time to be the departure time of the earliest flight in the ETMS dataset.

5. Set the MLG size:

Although there are a total of 41, 594 flights during the simulated 24-hour time period, the maximum number of aircraft that are airborne at any one time is 6045 aircraft. Therefore, the size of the 3-D MLG is set to 55 x 55 x 2 = 6050 nodes. The total number of nodes in the MLG, N, equals the number of active aircraft plus the number of holes; that is, $N = N_{active} + N_{holes}$. As the number of active aircraft changes at each time step, the composition of the MLG (the number of real nodes versus holes) changes accordingly.

6. Populate the MLG with blank placeholders over the US:

Fill the entire MLG (6050 nodes) with holes, evenly spaced throughout the computational domain covering the United States. This corresponds to an area defined by the following latitude and longitude, and Mercator projection coordinates, respectively:

$27° < latitude < 48°$; $-124° < longitude < -70°$, corresponding to
$-1.38\text{x}10^7$ m $< x < -7.78\text{x}10^6$ m ; $3.50 \text{ x } 10^6$ m $< y < 9.12 \text{ x } 10^6$ m.

The blank placeholders are also evenly spaced vertically, with the first row at an altitude of $z = 6400$ m (4 mi) and the second row at $z = 12{,}800$ m (8 mi).

7. Begin Timestep Loop:

7.1 Set current time:

If (time_step = 1): set the current time equal to the departure time of the earliest flight in the ETMS file)
If (time_step $\neq$1), increment the time by 10 seconds, and set as the current time

7.2 Determine the flight status at this current time, by comparing the departure time in the ETMS file with the current time. Since the ETMS file was sorted by departure time using the MLG (in Step 2), we do not need to search through the entire ETMS file. Possible values of flight status are: Active, Inactive, Newly_Active, and Newly-Inactive.

7.3 Modify the MLG, depending on the flight status determined above:

Newly_Active Flights take the place of a blank placeholder: If the departure time is between the current time and the next time step (in 10 seconds), the aircraft is Newly Active. Newly active flights are inserted into the MLG structure, replacing one of the blank placeholders.

Newly_Inactive Flights are converted to blank placeholders: An airborne aircraft that has just reached its final waypoint (in the current timestep) becomes a blank placeholder in the MLG grid.

Active Flights: Any airborne aircraft that has not reached its final waypoint

Inactive Flights: All inactive aircraft (aircraft that are not airborne at the current time) are not part of the MLG structure

7.4 Sort, using the MLG:

All nodes (representing active aircraft and holes) are sorted, based on their x-, y- and z-location, using a bubble sort process.

7.5 Update velocity vector from current position towards next waypoint:

For each node on the MLG that corresponds to an active aircraft, calculate a velocity vector during this timestep. This velocity vector is calculated based on the distance from the current position to the next waypoint, using the airspeed from the ETMS file. All of the other nodes on the MLG, corresponding to holes, remain stationary.

7.6 Conflict Detection and Resolution:

Once all of the nodes are sorted, aircraft that are closest to each other are also nearest neighbors on the MLG. When checking the distances between nearest neighbors, the user can specify how many neighbors in each direction to check.

If the distance between the two checked nodes is within five miles, we project the two flights forward in time, and check the distance again. If the projected distance is smaller, the aircraft are assumed to be moving toward each other. The "resolution" is to modify the velocity vector of one of the aircraft by 20 degrees. The 20 degree modification is made in the x-y plane only, so that the aircraft maintains the same altitude. This conflict resolution algorithm is illustrated in Figure A-1.

We have recently incorporated more complex CD&R algorithms, such as the NASA Stratway algorithm[16], and this will be described in a future paper. The goal of this presentation, however, is to compare the results of using the MLG to the Lat-Long grid, and therefore we use the same simplified CD&R algorithm for all simulations shown.

7.7 Update positions of active aircraft:

Calculate the new positions of all active aircraft, using either the velocity computed in Step 7.5, or that computed in Step 7.6 (when aircraft are rerouted for CD&R). Also, for the purposes of the simulation, when an aircraft is within 10 s of reaching a waypoint, it is considered to have reached it. The aircraft is then considered to be on its next leg of the flight.

7.8 Return to Step 7.1 to increment the time and repeat.

The procedure for the Lat-Long grid follows the same basic steps as those presented above for the MLG , with some exceptions, as discussed here:

a) In Step 2 of the MLG procedure, the ETMS records are sorted in order of departure time. Without the MLG, the ETMS records are *not* sorted by departure time, since the Lat-Long method only sorts by physical location.

b) In place of Steps 5 and 6 (in which the size of the MLG is set and it is initially populated with blank placeholders), set the size and dimensions of the airspace volume for the lat-long grid method. The number of lat-long grid cells is set to 55 x 55 x 2, and the grid cell sizes are Δx = 150,000 m, Δy = 200,000 m, and Δz = 9,000 m. This corresponds to a stationary Lat-Long grid of size:

$$-1.4\times10^7 < x < -5.75\times10^6 \text{ m},$$
$$2.7\times10^6 < y < 1.37\times10^7 \text{ m}$$
$$0 < z < 18{,}000 \text{ m}$$

This 3-D box covers the airspace volume over the United States.

c) In step 7.2, to determine the flight status, the user compares the departure time for each flight with the current time. Since, with the Lat-Long procedure, the ETMS records are not sorted by departure time, the user must search through all of the ETMS records (123,5743 records) to determine if the departure time matches the current time. This adds a considerable amount of CPU time to this step for the Lat-Long grid procedure. Possible values of flight status are Active or Inactive. There are no Newly-Active or Newly-Inactive flights, since there is no exchange of holes and active aircraft.

d) In place of steps 7.3 and 7.4, active aircraft are sorted into appropriate Lat-Long grid cells, based on their current position.

e) In step 7.6, we use the same CD&R algorithm (shown in Figure 5) that was used with the MLG procedure. However, rather than checking the neighboring MLG nodes, we instead check distances between aircraft within each lat-long grid cell. Each aircraft pair is checked once; i.e., for m aircraft in one cell, there are $m(m-1)/2$ neighbor checks.

References

[1]Alexandrov, N., Kincaid, R.K., Vargo, E.P., "Toward Optimal Transport Networks", AIAA Paper 2008-5814, American Institute of Aeronautics and Astronautics, Reston, VA, 2008.

[2]Bilimoria, K., Sridhar, B., Chatterji, G., Sheth, K., Grabbe, S.: FACET: Future ATM Concepts Evaluation Tool, 3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, Italy, June 2000.

[3]Meyn, L., Romer, T., Roth, K., Bjarke, L., and Hinton, S., "Preliminary Assessment of Future Operational Concepts Using the Airspace Concept Evaluation System," AIAA-2004-6508, AIAA Aviation Technology, Integration, and Operations (ATIO) Forum, Chicago, IL, Sep. 2004.

[4]Kaplan, C.R., Oran, E.S., Alexandrov, N., Boris, J.P., "The Monotonic Lagrangian Grid Particle Grid: A Fast Tracking Methodology for Air-Traffic Modeling,", AIAA Paper 2009-1635, American Institute of Aeronautics and Astronautics, Reston, VA, 2009.

[5]Kaplan, C.R., Oran, E.S., Alexandrov, N., Boris, J.P., "The Monotonic Lagrangian Grid for Fast Air-Traffic Evaluation," AIAA Paper 2010-597, American Institute of Aeronautics and Astronautics, Reston, VA, 2010.

[6]Kaplan, C.R., Dahm, J.P.S., Oran, E.S., Alexandrov, N., and Boris, J.P., "The Monotonic Lagrangian Grid for Rapid Air-Traffic Evaluation, AIAA Paper 2010-9336, American Institute of Aeronautics and Astronautics, Reston, VA 2010.

[7]Boris, J., "A Vectorized "Near Neighbors" Algorithm of Order N Using a Monotonic Logical Grid," *J. of Computational Physics*, **66**, 1 (1986).

[8]Picone, J.M., Lambrakos, S.G., Boris, J.P. and Jajodia, S., "Initial Comparison of Monotonic Logical Grid and Alternative Data Base Structures," NRL Memorandum Report 5860, Sept 30, 1986.

[9]Lambrakos, S.G. and Boris, J.P., "Geometric Properties of the Monotonic Logical Grid Algorithm for Near Neighbor Calculations," J. of Computational Physics, **73**, 183 (1987).

[10]Lambrakos, S.G., Nagumo, M., Boris, J.P., Oran, E.S., and Gaber, B., "Molecular Dynamics Simulation of a Lipid Bilayer Using Novel Monotonic Logical Grid (MLG) and Multiple Constraint Relaxation (MCR) Algorithms," Biophysical Journal, **51**, A440 (1987).

[11]Cybyk, B., Oran, E.S., Boris, J.P., and Anderson, J.D., "Combining the Monotonic Lagrangian Grid with a direct simulation Monte Carlo Model," J. of Computational Physics, **122**, 323 (1995).

[12]Boris, J.P., Picone, J.M., and Lambrakos, S.G., "BEAST: A High-Performance Battle Engagement Area Simulator/Tracker," NRL Memorandum Report 5908, December 31, 1986.

[13]Jajodia, S., Meadows, C.A., Boris, J.P., Picone, J.M., and Lambrakos, S.G., "NRL Research in Database Management for the SDI Battle Management System," NRL Memorandum Report 5921, January 30, 1987. Distribution limited.

[14]Picone, J.M., Boris, J.P., Lambrakos, S.G., Uhlmann, J. and Zuniga, M., "Near-Neighbor Algorithms for Processing Bearing Data," NRL Memorandum Report 6456, May 10, 1989.

[15]Kolbe, R.L., Boris, J.P. and Picone, J.M., "Battle Engagement Area Simulator/Tracker," NRL Memorandum Report 6705, October 8, 1990.

[16]Hagen, G., Butler, R.W. and Maddalon, J.M., Stratway v0.5.1 User Manual, DRAFT 8-5-2010, NASA/TM-2009-000000, August, 2010.