Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

AIRSPACE GEOMETRY AND 4D FLIGHT PROXIMITY DETECTION FOR SIMULATION OF THE NATIONAL AIRSPACE SYSTEM

Paul T. R. Wang
Richard E. Snow

The MITRE Corporation
Center for Advanced Aviation System Development
7515 Colshire Drive
McLean, VA 22102, U.S.A.

ABSTRACT

The authors present uncomplicated and well established equations that can be used in simulation or real-world applications to determine key crossing points and aircraft proximity when the trajectory and speed of aircraft pairs are known. These equations, in closed form, were developed for computing the minimum distance between two aircraft within the four-dimensional (4D) space-time domain. The 4D flight proximity information can be used in simulation to evaluate large numbers of scheduled routes over a limited airspace for controller workload assessment. Also, it can be used to detect potential separation violations and impacts of traffic flow management (TFM) strategies. An example of computing the distance between two flights in 4D is presented. Sample aircraft proximity landscape in 4D space-time simulation with MATLAB code is also provided.

1 INTRODUCTION

Airspace geometry is defined typically as polygons or polylines in 3D by latitude (Lat), longitude (Lon), and altitude (Alt). Airspaces can be divided into airspace with fixed altitudes for their floor or ceiling. Flight paths are defined by a set of 3D points. Proximity detection for 4D (time added to 3D) flights is a key element in managing the National Airspace System (NAS). Up to 90K flights may enter the national airspace daily by 2020. When a large number of flights are scheduled to arrive at a given airspace, it is critical that conflicts be detected in a timely manner. The MITRE Corporation's Center for Advanced Aviation System Development (CAASD) has developed a suite of state-of-the-art simulation tools to model the NAS. One of these tools (the mid-level simulator) simulates both international and NAS-wide air traffic control (ATC) events at progressively detailed levels of granularity (Wieland, 2004 and Wang, 2005). This model is written in Simulation Language with eXtensibility (SLX) (Wolverine Software Corporation, 2003). This paper shows how such a simulation tool is used to calculate 4D flight proximity for conflict detection. This work is related to previous work in conflict estimation reported by Richard Irvine (Irvine, 2002 and 2003) for the EUROCONTROL Experimental Center. We have replaced the ratio of speed between two aircraft with vectors in 4D so that both the altitude and time are explicitly incorporated. The concepts presented in this paper are also applicable to any moving objects in 4D.

2 DEFINING 3D POINTS

In this paper, we adopt the convention that Greenwich is 0 Lon and Lon is negative west of Greenwich; the equator is 0 Lat and Lat is negative south of the equator, and sea level 0 Alt above the earth's surface. We define a 3D point in space as a triple (Lat, Lon, Alt) with the following convention that $-\pi/2 \le Lat < \pi/2$, $-\pi \le Lon < \pi$, and $R \le Alt$, where R is the radius of the earth. This triplet is then converted to a vector in space as follows:

$$\vec{P} = (P_x, P_y, P_z) = h_P u_P$$

where h_P = the altitude (from earth center) at point *P*; u_P is a unit vector ($u_P \bullet u_P = 1$) defined as follows:

$$u_p = (x_p, y_p, z_p) = (\cos\alpha_p \cos\beta_p, \cos\alpha_p \sin\beta_p, \sin\alpha_p)$$

with $\alpha_P = Lat$ and $\beta_P = Lon$ where

$$\alpha_P = \tan^{-1}(z_P / \sqrt{x_P^2 + y_P^2}), \beta_P = \tan^{-1}(y_P / x_P)$$
$$\text{where } -\pi < \beta_P = \tan^{-1}(y_P, x_P) \le \pi$$

We note that the cross product of two unit vectors, u_P and u_Q, is a vector $u_P \times u_Q$ that is orthogonal to both u_P and u_Q.

3 DEFINING SECTOR AIRSPACE AND FLIGHT PATHS

In this paper, we have made a simplifying assumption that a sector is a polygon with a fixed altitude for its floor or ceiling, and a flight path is an ordered set of points in 3D. A flight in 4D will follow its predetermined flight path with advancing time at each 3D point. The geometry of sectors and flight paths may be represented by a set of points as vectors in 3D. Lat and Lon are uniquely defined on the unit sphere with the center of the earth as its origin and a radius of 1. Any point in a given airspace is uniquely identifiable as a vector, the product of the altitude (measured from the center of the earth) and a unit vector. Using the dot product and cross product for vectors, one can compute easily all the key points along a flight path entering or exiting a sector through the top, bottom, or sides known as sector crossings and the closest vertex of a sector to a given path in 3D.

4 SECTOR CROSSING WHEN ASCENDING OR DESCENDING

An aircraft may cross a sector in many different ways. A flight may stay at a fixed altitude when leaving the current airspace and entering an adjacent airspace at a side crossing. Flights may enter target airspace or exit the current airspace from the top or bottom of the airspace. Flights may also cross a sector through a side while climbing or descending. Further, flights can cut through an edge, part of an edge, or a corner of a given airspace. Figure 1 illustrates some of the common ways in which a flight may traverse a given airspace with a given route in terms of distinct nodes in 3D.

In Figure 1, a sector is defined as a polygon consisting of a set of points in 3D with fixed ceiling and floor $\{\alpha,\beta,\gamma,\sigma,\mu,\eta,\rho,\tau\}$. The path (trajectory) of a given flight is shown as a sequence of points each with distinct Lat, Lon, and Alt $\{A,B,C,D,E,F,G,H,I,J\}$. For flights entering a sector from the top (ceiling) or bottom (floor), Figure 2 illustrates the relationship among the three points, *P*, *Q* and *X*. From Figure 2, we have the following relationships:

$$\lambda = (h_P - h_X)/(h_P - h_Q)$$

$$u_P = \{x_P, y_P, z_P\},\quad u_Q = \{x_Q, y_Q, z_Q\},$$

$$u_n = u_P \times u_Q / |u_P \times u_Q|.$$

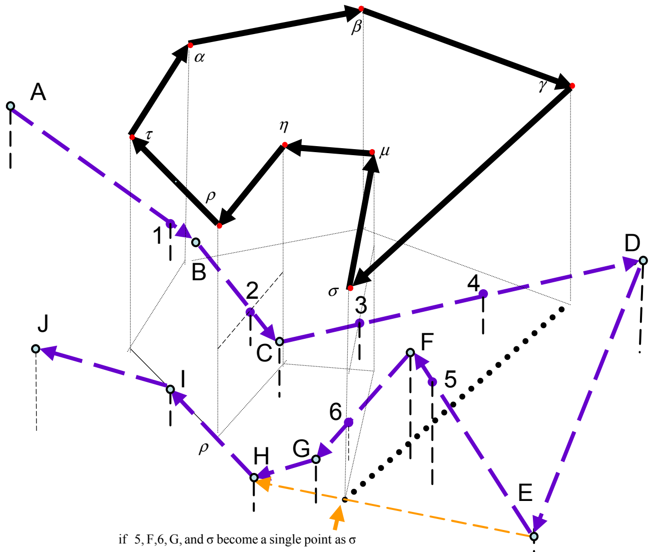

Figure 1: Sample 3D Sector Airspace Crossings

The crossing point X is identified by the vector $\vec{X} = h_X u_X$ where

$$u_X = u_P \cos\lambda\theta + (u_n \times u_P / |u_n \times u_P|)\sin\lambda\theta$$

Arc $\overset{\frown}{PQ}$ is subtended by $\theta = \angle POQ$ where *O* is the center of the earth, so that $\theta = \cos^{-1}(u_P \bullet u_Q)$. Arc $\overset{\frown}{PX}$ is subtended by $\lambda\theta = \angle POX$.

Also, we have $u_X = (x_X, y_X, z_X)$ with α_X and β_X as defined before. Note that the unit vector u_X is completely determined by the unit vectors u_P, u_Q, and altitudes, h_P, h_Q, and h_X.

Similarly, to determine side crossings, Figure 3 illustrates the relationship between the path segment $\overline{PQ}$, which intersects the sector segment $\overline{VW}$. We have the following equality:

$$u_X = n_{\overline{PQ}} \times n_{\overline{VW}} / |n_{\overline{PQ}} \times n_{\overline{VW}}|$$

$$= (u_P \times u_Q) \times (u_V \times u_W) / |(u_P \times u_Q) \times (u_V \times u_W)|$$

where $n_{\overline{PQ}} = u_P \times u_Q$ and $n_{\overline{VW}} = u_V \times u_W$.

$$h_{X_S} = h_p + (h_Q - h_P)\cos^{-1}(u_p \bullet u_X)/\cos^{-1}(u_P \bullet u_Q)$$

In Figure 3 both u_X and h_{X_S} are completely determined by u_P, u_Q, u_V, u_W, h_P, and h_Q.

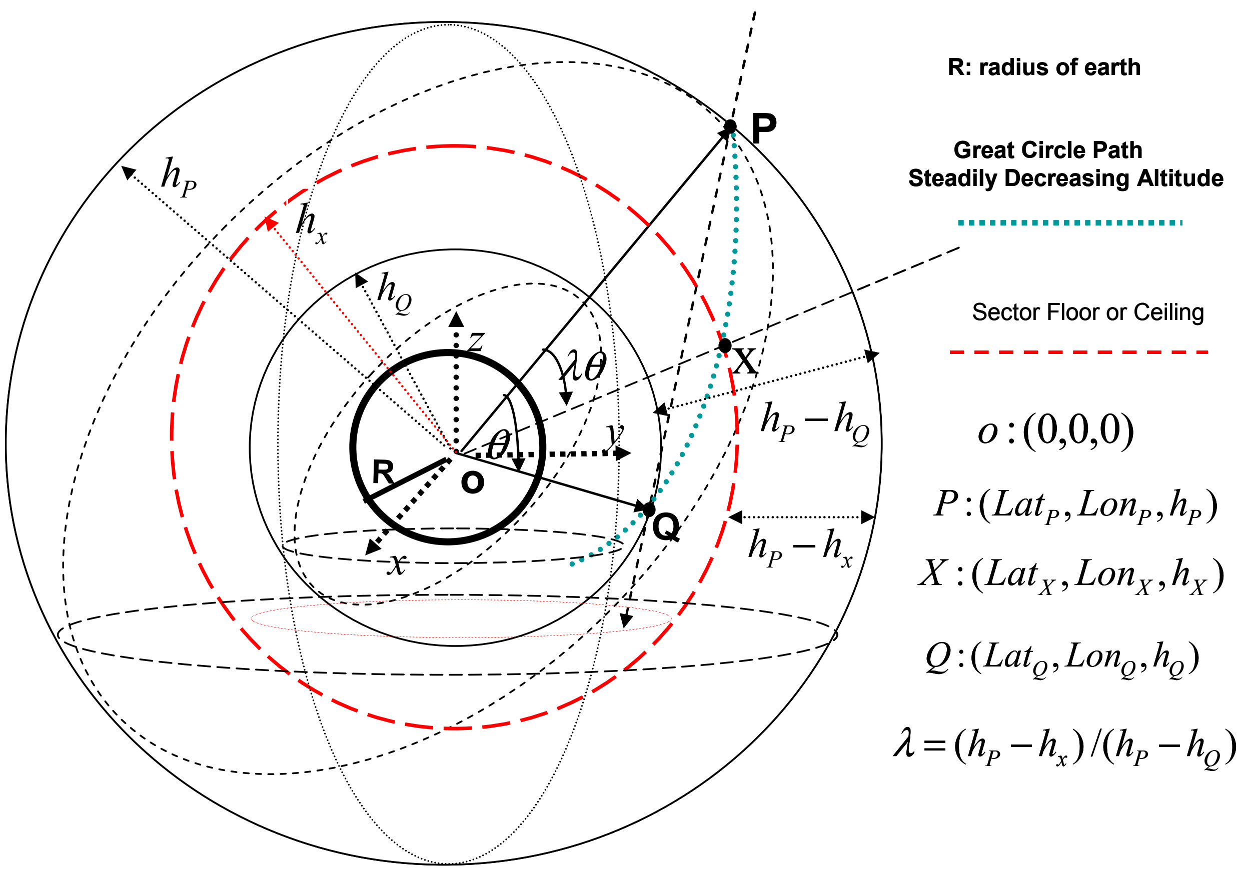

Figure 2: Top or Bottom Sector Airspace Crossing

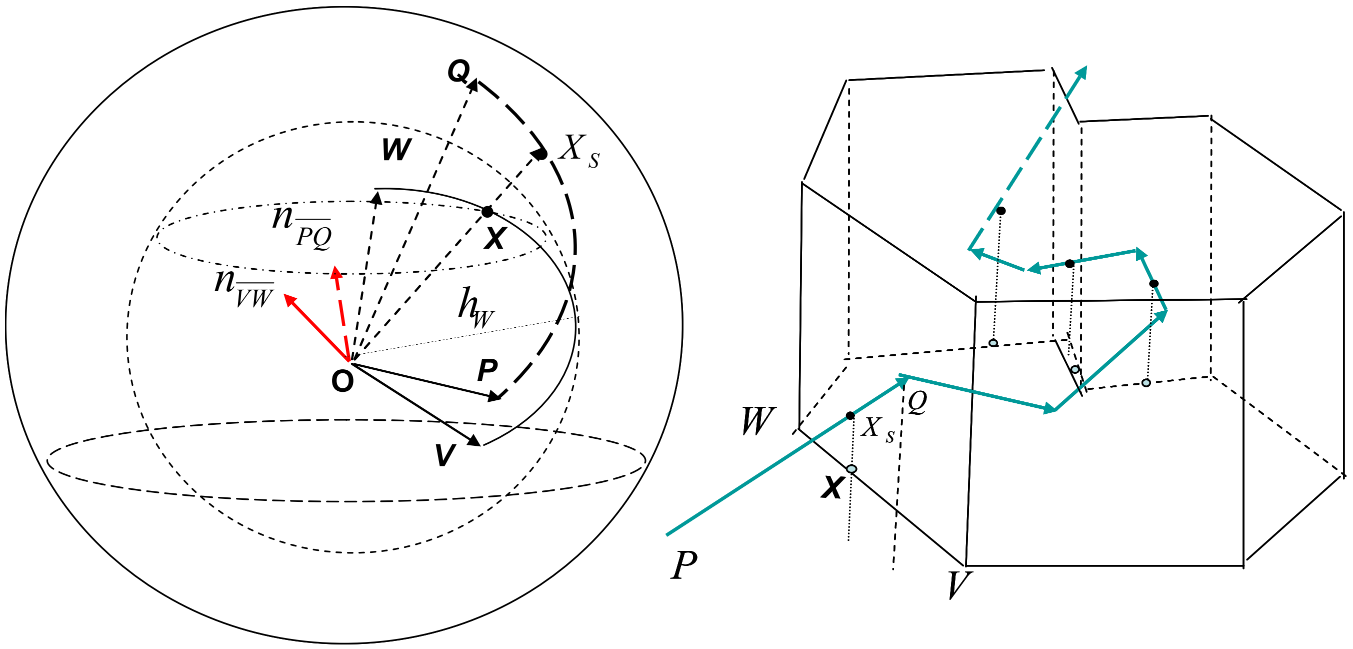

Figure 3: Computing Sector Airspace Side Crossing

5 THE NEAREST POINT ON A PATH TO A VERTEX

If the path of a given flight does not intersect or cross a given sector, it is possible to determine a point on the flight path that is closest to the defined sector. This can be achieved by finding the closest point on a segment of the path to the vertices defining the airspace. Such information is very useful in determining the distance between a flight path and a given sector.

The equations for locating the nearest point on a path segment to a vertex as shown in Figure 4 are as follows:

$$\text{Let } W = \vec{P} + |\overline{PV}|(u_{\overline{PV}} \bullet u_{\overline{PQ}})\, u_{\overline{PQ}}, \quad \text{where}$$

$$\overrightarrow{PQ} = h_P u_P - h_Q u_Q, \; \overrightarrow{PV} = h_P u_P - h_V u_V,.$$

and where

$$u_{\overline{PV}} = \overrightarrow{PV} / |\overrightarrow{PV}|, \;\; u_{\overline{PQ}} = \overrightarrow{PQ} / |\overrightarrow{PQ}|,$$

$$\vec{P} = h_P u_P, \;\; \text{and} \;\; h_W = |W|.$$

Thus, we have the desired vector $h_W u_W$ and closest distance $|\overline{VW}|$ as follows:

$$|\overline{PV}| = |h_P u_P - h_V u_V| = \sqrt{h_P^2 + h_V^2 - 2h_P h_V (u_P \bullet u_V)}$$

$$|\overline{VW}| = |h_V u_V - h_W u_W| = \sqrt{h_V^2 + h_W^2 - 2h_V h_W (u_V \bullet u_W)}$$

$$u_W = W / |W| = (x_W, y_W, z_W)$$

$$\alpha_W = \tan^{-1}(z_W / \sqrt{x_W^2 + y_W^2})$$

$$\beta_W = \tan^{-1}(y_W / x_W)$$

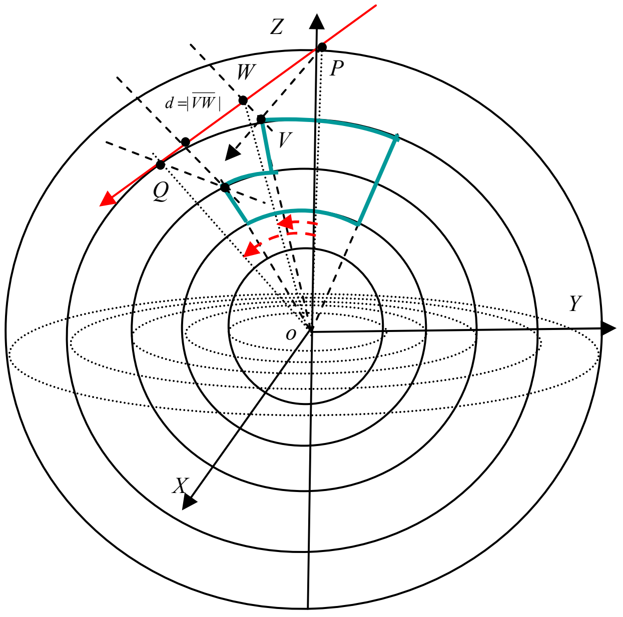

Figure 4: The Nearest Point on a Path Segment to a Vertex

6 4D PROXIMITY DETECTION

A flight traversing its path forms a 4D track in the space-time domain. To determine whether or not two flights are safely separated, it is necessary first to determine the shortest distance between the flights during their airborne trips. When the position and speed of two aircraft are known, one can determine the shortest distance between the two aircraft in 4D for proximity detection, collision prediction, or conflict avoidance. Such information can be used for Traffic Flow Management (TFM) during congested periods or poor weather conditions.

We have diagrammed the shortest distance between two airborne aircraft in 4D as shown in Figure 5.

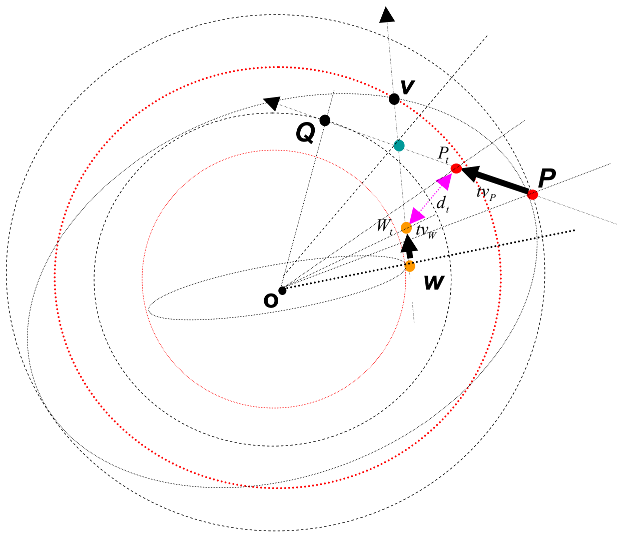

Figure 5: Shortest Distance Between Two Flights

In Figure 5, we have $d_t = |\overline{P_t W_t}| = \sqrt{\overrightarrow{(P_t - W_t)} \bullet \overrightarrow{(P_t - W_t)}}$

where $\overrightarrow{P_t - W_t} = (\vec{P} + tv_P u_{\overrightarrow{PQ}}) - (\vec{W} + tv_W u_{\overrightarrow{WV}})$

$$= (h_P u_P + tv_P \overrightarrow{PQ}/|\overline{PQ}|) - (h_W u_W + tv_W \overrightarrow{WV}/|\overline{WV}|),$$

and where

$$\overrightarrow{PQ} = h_Q u_Q - h_P u_P,\ |\overline{PQ}| = \sqrt{h_P^2 + h_Q^2 - 2h_P h_Q (u_P \bullet u_Q)}$$

$$\overrightarrow{WV} = h_V u_V - h_W u_W,\ |\overline{WV}| = \sqrt{h_W^2 + h_V^2 - 2h_W h_V (u_W \bullet u_V)}$$

Therefore,

$$d_t^2 = ((h_P u_P - h_W u_W) + t(v_P u_{\overrightarrow{PQ}} - v_W u_{\overrightarrow{WV}})) \bullet$$

$$((h_P u_P - h_W u_W) + t(v_P u_{\overrightarrow{PQ}} - v_W u_{\overrightarrow{WV}}))$$

$$= At^2 + Bt + C = f(t),$$

where $A = v_P^2 + v_W^2 - 2v_P v_W (u_{\overrightarrow{PQ}} \bullet u_{\overrightarrow{WV}});$

$$B = 2h_P v_P (u_P \bullet u_{\overrightarrow{PQ}}) + 2h_W v_W (u_W \bullet u_{\overrightarrow{WV}})$$

$$- 2h_W v_P (u_W \bullet u_{\overrightarrow{PQ}}) - 2h_P v_W (u_P \bullet u_{\overrightarrow{WV}});$$

$$C = h_P^2 + h_W^2 - 2h_P h_W (u_P \bullet u_W);$$

Finally, the minimizing time and distance are given by $t^* = -B/2A$ with $d^* = \sqrt{f(t^*)}$, the result of setting $f'(t^*) = 0$ and ensuring that $f''(t^*) > 0$. Note that the shortest distance d^* between two flights is completely determined by the current positions, the directions of the paths, and the speeds of the flights.

7 AN EXAMPLE

As an example, we plot the distance function between two flights with the following data to illustrate the 4D landscape of the distance function $f(t) = At^2 + Bt + C$.

Aircraft 1 at *P*: Lat = 42.0, Lon = -86.0, Alt = 14k ft, speed=500 knots flying to Lat = 45.6, Lon = -81.0, Alt = 12k ft.

Aircraft 2 at *W:* Lat = 43.4, Lon = -85.0 Alt = 11k ft. speed = r*500 knots with r = 0.1, 0.2, -----, 1.0. flying to Lat = 40.0, Lon = -83.0, Alt = 15k ft.

Figure 6 plots the relevant 4D landscape determined by the function $f(t)$. Note that the worst case flight separation is identified as t = 10.5 (minutes), r = 0.3 (v2 = 150 knots), and d = 3.6 nautical miles (nmi).

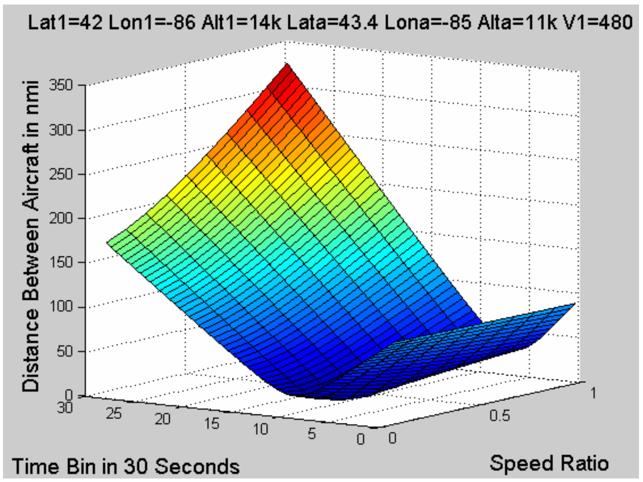

Figure 6: Flight Distance in 4D Landscape

A program coded in MATLAB that produces Figure 6 is provided as follows:

```
% code in MATLAB
ER = 3437.7468*6076.115;
lat1 = 42.0*pi/180.0;
lon1 = -86.0*pi/180.0;
lat2 = 45.6*pi/180.0;
lon2 = -81.0*pi/180.0;
lata = 43.4*pi/180.0;
lona = -85.0*pi/180.0;
latb = 40.0*pi/180.0;
lonb = -83.0*pi/180.0;
alt1 = ER+100.0*140.0;
alt2 = ER+100.0*120.0;
alta = ER+100.0*110.0;
altb = ER+100.0*150.0;
vp = 500.0*6076.115/60.0;
for w = 1:10
vw=vp*w*0.1;
u2{1} = [];
ua{1} = [];
ub{1} = [];
u1{1}(1) = cos(lat1)*cos(lon1);
u1{1}(2) = cos(lat1)*sin(lon1);
u1{1}(3) = sin(lat1);
u2{1}(1) = cos(lat2)*cos(lon2);
u2{1}(2) = cos(lat2)*sin(lon2);
u2{1}(3) = sin(lat2);
ua{1}(1) = cos(lata)*cos(lona);
ua{1}(2) = cos(lata)*sin(lona);
ua{1}(3) = sin(lata);
ub{1}(1) = cos(latb)*cos(lonb);
ub{1}(2) = cos(latb)*sin(lonb);
ub{1}(3) = sin(latb);
hp = alt1;
hq = alt2;
hw = alta;
hv = altb;
Lpq = sqrt(hp*hp+hq*hq-
    2.0*hp*hq*dotproduct(u1{1},u2{1}));
```

```
Lwv = sqrt(hw*hw+hv*hv-
    2.0*hw*hv*dotproduct(ua{1},ub{1}));
upq{1}(1) = (hq*u2{1}(1)-hp*u1{1}(1))/Lpq;
upq{1}(2) = (hq*u2{1}(2)-hp*u1{1}(2))/Lpq;
upq{1}(3) = (hq*u2{1}(3)-hp*u1{1}(3))/Lpq;
uwv{1}(1) = (hv*ub{1}(1)-hw*ua{1}(1))/Lwv;
uwv{1}(2) = (hv*ub{1}(2)-hw*ua{1}(2))/Lwv;
uwv{1}(3) = (hv*ub{1}(3)-hw*ua{1}(3))/Lwv;
C  =hp*hp+hw*hw-
   2.0*hp*hw*dotproduct(u1{1},ua{1});
B = 2.0*(hp*vp*dotproduct(u1{1},upq{1})
    +hw*vw*dotproduct(ua{1},uwv{1})
    - hw*vp*dotproduct(ua{1},upq{1})
     -hp*vw*dotproduct(u1{1},uwv{1}));
A=vp*vp+vw*vw-
    2.0*vp*vw*dotproduct(upq{1},uwv{1});
t = -0.5*B/A;
mdist = sqrt(C+B*t+A*t*t)/6076.115;
%  graphic portion
 Data{w} = [];
 for i = 1:60
 ti = 0.5*i;
 dist = sqrt(C+B*ti+A*ti*ti)/6076.115;
 Data{w}(i) = dist;
 Dmatrix(i,w) = dist;
 end %
end % w
[dm,dn]=size(Dmatrix);
 x = (0.1:0.1:1.0);
 y = (0.5:0.5:30);
for i = 1:60
 for k = 1:10
    f = Dmatrix(i,k);
    Z(i,k) = f;
 end
end
[X,Y] = meshgrid(x,y);
vpint = 60.0*round(vp/6076.11);
surf(X,Y,Z);
xlabel(' Speed Ratio        ','FontSize',14);
zlabel(' Distance Between Aircraft in
             nmi','FontSize',14);
ylabel(' Time Bin in 30 Seconds
','FontSize',14);
txt = sprintf('Lat1=42 Lon1=-86 Alt1=14k
Lata=43.4
         Lona=-85 Alta=11k V1=%d',vpint);
text (70,1.2,0.01,txt);
        title(txt,'FontSize',13);
end
```

8 APPLICATIONS OF FLIGHT PROXIMITY DETECTION

The equations for computing the shortest distance between two flights may be used in simulation or real-world applications to determine the number of flights that are projected to fall within a given minimum separation distance. We can define the set of all flights with their pairwise shortest distance in 4D less than or equal to *d* as *d*-neighbors. The quantity of *d*-neighbors of a given flight or a group of flights scheduled to arrive at a sector is an ideal performance metric for evaluating different conflict resolution (e.g. path, altitude, or speed changes) or TFM strategies. The 4D flight proximity information may also be used in future automation to provide viable options for detecting potential conflicts or collisions in congested airspace. It may prove useful in helping to gauge the complexity of traffic in a region of airspace that a flight is scheduled to enter. The authors are implementing the flight proximity detection algorithm for scheduled flights (Bhadra 2003) with the mid-level NAS simulation tool developed at CAASD.

Future challenges include the computation of a flight proximity distance function involving acceleration, deceleration, and/or path changes maneuvers, and the sensitivity analysis of the size of *d*-neighbors with respect to minimum separation requirements. It is also possible to animate both the flight proximity function and the size of *d*-neighbors in 4D for the entire NAS or regional airspace with SLX and Proof (Wolverine, 2003 and 2004).

ACKNOWLEDGMENTS

The authors acknowledge Mr. Peter C. Kuzminski for leading the team that enhanced the MITRE's NAS-wide ATC simulator in SLX. We also acknowledge our colleagues, Thomas C. Holden and Noam Tene for their efforts in providing queries that prepared input data for this study. Peer review of the paper by Dr. Leone C. Monticone and Dr. Mark J. Rockman are also greatly appreciated. The authors are also indebted to Ms. Debra A. Pool, Ms. Barbara B. Wise, and Dr. Leonard A. Wojcik, for their editorial advice that significantly improved the readability of this paper.

REFERENCES

Bhadra, D., 2003. Demand for Air Travel in the United States: Bottom-Up Econometric Estimation and Implications for Forecasts by Origin and Destination Pairs, *Journal of Air Transportation*, Vol. 8 No. 2 pp. 19-56.

Irvine, R., 2002. EUROCONTROL Experimental Center, A Simplified Approach to Conflict Probability Estimation, *Air Traffic Control Quarterly* 10, pp. 85-113.

Irvine, R. 2003. EUROCONTROL Experimental Center, Target Miss Distance to Achieve a Required Probability of Conflict. ATM 2003 Proceedings.

Volpe Center, 2000. Aircraft Situation Display To Industry: Functional Description and Interface Control Document, Version 4.0.

Wang, P. T.R., C. R. Wanke, and F. P. Wieland, 2004. Modeling Time and Space Metering of Flights in the National Airspace System. *Proceedings of the 2004*

Winter Simulation Conference, pp. 1299-1304. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wieland, F. P. 2004. Next-Generation of the National Airspace System, CAASD. The MITRE Corporation. Available online via <http://www.mitre.org/news/events/tech04/briefing/1492.pdf>.

Williams, Ed. 2006, Aviation Formulary V1.42. Available online via <http://williams.best.vwh.net>.

Wolverine Software Corporation. 2003. Simulation with SLX, by Daniel T. Brunner and James O. Henriksen.

Wolverine Software Corporation. 2004. Using Proof Animation. Third Edition.

AUTHOR BIOGRAPHIES

PAUL T. R. WANG has been with The MITRE Corporation's Center for Advanced Aviation System Development (CAASD) since 1984. He graduated from The Ohio State University in 1973 with M.S. and Ph.D. degrees in Computer Sciences. His interests include optimization, modeling, simulation, and queuing theory. Dr. Wang's work on simulation and optimization were published by WSC, ORSA, DASC, *Air Traffic Control Quarterly*, AIAA, and the *Journal of Computer Networks and ISDN Systems*. His e-mail address is <pwang@mitre.org>.

RICHARD E. SNOW has been a Senior Software Applications Development Engineer at The MITRE Corporation's Center for Advanced Aviation System Development (CAASD) since 1988. He received a B. A. degree from the University of Minnesota, an M. A. from the University of Michigan, and a Ph. D. from Northwestern University (1973), all three degrees in mathematics. Dr. Snow's present work involves implementation and analysis of algorithms for a variety of simulations used for research studies related to air traffic control. His e-mail address is <rsnow@mitre.org>.