NASA/TM—2012–216308

A Probabilistic, Facility-Centric Approach to Lightning Strike Location

Author Name: Lisa L. Huddleston

Kennedy Space Center, Florida

William P. Roeder
45th Weather Squadron, Patrick Air Force Base

Francis J. Merceret
Kennedy Space Center, Florida

January 2012

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing help desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at 443-757-5803

- Phone the NASA STI Help Desk at 443-757-5802

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM—2012–216308

A Probabilistic, Facility-Centric Approach to Lightning Strike Location

Author Name: Lisa L. Huddleston

Kennedy Space Center, Florida

William P. Roeder
45th Weather Squadron, Patrick Air Force Base

Francis J. Merceret
Kennedy Space Center, Florida

National Aeronautics and
Space Administration

Kennedy Space Center
Kennedy Space Center, FL 32899-0001

January 2012

Acknowledgments

This work would not exist were it not for the generous contributions and suggestions by Dr. Ken Chan of the Aerospace Corp, who is an expert on spacecraft collision probability, on whose work the probability of lightning within a radius of interest was based. Dr. Darrin Leleux of the Johnson Space Center, Houston, TX, and Dr. Walt Gill of Sandia National Laboratory also provided helpful guidance and testing. Mr. Jeremy Hinkley and Mr. Pete Hopman of United Space Alliance, the prime contractor for Space Shuttle operations, provided very valuable efficiency modifications for the visual basic code and also integrated the probability calculations and closest ellipse point algorithm into the 45th Weather Squadron Lightning Report Spreadsheet.

The authors appreciate a helpful review of an earlier draft of this paper by Mr. John Madura of the Kennedy Space Center. This work was done under the Kennedy Space Center Employee Development Program.

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Table of Contents

List of Figures

List of Tables

Abstract

A new probabilistic facility-centric approach to lightning strike location has been developed. This process uses the bivariate Gaussian distribution of probability density provided by the current lightning location error ellipse for the most likely location of a lightning stroke and integrates it to determine the probability that the stroke is inside any specified radius of any location, even if that location is not centered on or even with the location error ellipse. This technique is adapted from a method of calculating the probability of debris collision with spacecraft. Such a technique is important in spaceport processing activities because it allows engineers to quantify the risk of induced current damage to critical electronics due to nearby lightning strokes. This technique was tested extensively and is now in use by space launch organizations at Kennedy Space Center and Cape Canaveral Air Force Station. Future applications could include forensic meteorology.

Nomenclature

θ = angle between the collision plane coordinates and the rotated collision plane coordinates
ρ_{xz} = correlation coefficient of x and z
$\sigma_{K'}$ = standard deviation of x-coordinate of the diagonalized covariance ellipse in the rotated coordinate system
$\sigma_{H'}$ = standard deviation of z-coordinate of the diagonalized covariance ellipse in the rotated coordinate system
σ_x = standard deviation of x
σ_z = standard deviation of z
$\sigma_{x'}$ = standard deviation of x in the rotated coordinate system
$\sigma_{z'}$ = standard deviation of z in the rotated coordinate system
μ_K = x-coordinate of target circle in the (X', Z') coordinate system
μ_H = z-coordinate of target circle in the (X', Z') coordinate system
A = collision cross-sectional area (nautical miles^2, nmi^2)
$d\theta$ = the angle between two points on the target ellipse
dH = integration step
H = intermediate variable in the lightning probability algorithm
P = probability
pdf = probability distribution function
r_A = radius of circle of cross-sectional area, A (nautical miles, nmi)
R = radius of ellipse (nmi)
R1 = distance to first point on target ellipse (nmi)
R2 = distance to second point on target ellipse (nmi)
x = horizontal rectangular coordinate in the collision plane (nautical miles, nmi)
x' = horizontal rectangular coordinate in the rotated collision plane (nmi)
x" = transformation variable that circularizes the x-component of the probability ellipse (nmi)
x_e = nominal distance of closed approach of two colliding objects (nmi)
W = intermediate variable in the lightning probability algorithm
z = vertical rectangular coordinate in the collision plane (nautical miles, nmi)
z' = vertical rectangular coordinate in the collision plane (nautical miles, nmi)
z" = transformation variable that circularizes the z-component of the probability ellipse (nmi)

Introduction

The ability to a Introduction accurately estimate the probability that an individual nearby cloud-to-ground lightning stroke was within a specified distance of any specified spaceport processing facility at Kennedy Space Center (KSC) or Cape Canaveral Air Force Station (CCAFS) is important to processing payloads and space launch vehicles before launch. Such estimates allow engineers to decide if inspection of electronics systems aboard satellite payloads, space launch vehicles, and ground support equipment is warranted due to induced currents from that stroke. If induced current damage has occurred, inspections of the electronics are critical to identify required fixes and avoid degraded performance or failure of the satellite or space launch vehicle. However, inspections are costly both financially and in terms of delayed processing for space launch activities. As such, it is important these inspections be avoided if not needed. At KSC/CCAFS, one of the main purposes of the Four Dimensional Lightning Surveillance System (4DLSS) (Murphy, 2008, Roeder, 2010) is detection of nearby strokes and determination of their peak current to support decisions to inspect electronics (Flinn, 2010a, Flinn, 2010b, Roeder, 2005). The high frequency of lightning occurrence in East Central Florida combined with the large amount of complex sensitive electronics in satellite payloads, space launch vehicles, and associated facilities makes those decisions critically important to space launch processing. The 4DLSS provides the data for 50th percentile location error ellipses for the best location for each stroke, which is then scaled to 95th or 99th percentile ellipses depending on customer requirements. This error ellipse is necessarily centered on the best location of the lightning stroke. The 4DLSS, however, has not been able to provide the probability of the stroke being within a customer specified distance of a point of interest. This paper presents a new method to convert the 4DLSS 50th percentile location error ellipse for best location of any stroke into the probability that the stroke was within any radius of any facility at CCAFS/KSC. This technique could be adapted for use with National Lightning Detection Network (NLDN) data. This new probabilistic facility-centric technique is a significant improvement over the stroke-centric location error ellipses the 45th Weather Squadron (45WS) has provided in the past. This technique is adapted from a method of calculating the probability of debris collision with spacecraft (Chan, 2008, Leleux, 2002, Patera, 2001).

Methodology

Background

In spacecraft collision probability and other applications, at the instant of "nominal" closest approach, the position uncertainty of the collision object relative to the asset being protected is described by a bivariate Gaussian probability density function (pdf) (Chan, 2008, Patera, 2001, Alfano, 2006, Alfano 2007), as shown in the following equation

$$f(x,z)=\frac{1}{2\pi\sigma_x\sigma_z\sqrt{1-\rho_{xz}^2}}e^{-\left[\left(\frac{x}{\sigma_x}\right)^2-2\rho_{xz}\left(\frac{x}{\sigma_x}\right)\left(\frac{z}{\sigma_z}\right)+\left(\frac{z}{\sigma_z}\right)^2\right]/2\left(1-\rho_{xz}^2\right)} \tag{1}$$

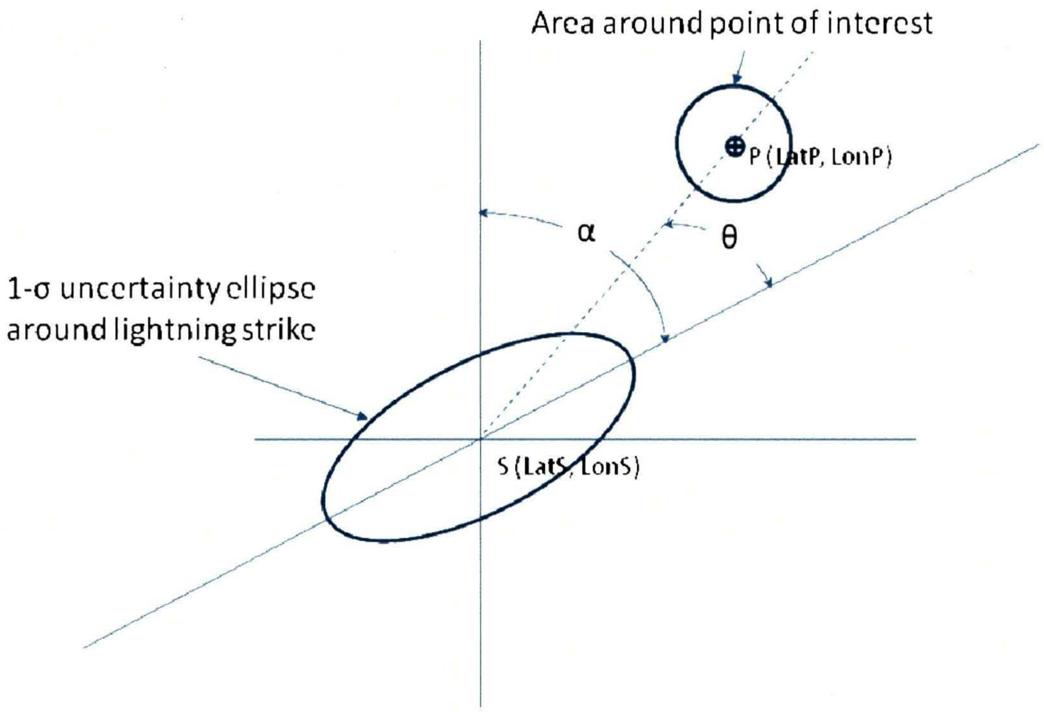
where σ_x and σ_z = the standard deviations of x and z, ρ_{xz} = correlation coefficient of x and z, x and z are the designations for the rectangular coordinates in the collision plane.

The probability of collision (Eq. 2) is given by the two-dimensional integral, where A is the collision cross-sectional area which is a circle with radius, r_A (Chan, 2008).

$$P = \iint_A f(x,z)\,dxdz \tag{2}$$

There is no known analytical solution to the above integral when the two standard deviations σ_x and σ_z are not equal. The solution is found by performing a numerical integration of the two dimensional Gaussian pdf (Chan, 2008, Patera, 2001, Alfano, 2006, Alfano 2007).

The geometry used for spaceflight collision probability can also be used for estimation of the probability of an individual nearby lightning stroke contacting the surface within a specified distance of a specified point of interest as shown in Fig. 1. In Fig. 1, α is the heading of the semi-major axis of the lightning location uncertainty ellipse from true north and θ is the angle between the semi-major axis of the lightning location uncertainty ellipse and line connecting the center of the lightning uncertainty ellipse and the center of the area of interest. Two methods of integrating the above probability were tested and gave identical results. The first solution method was based on an algorithm by Patera (2001) and the second solution method was based on an algorithm by Chan (2011). Chan's algorithm ran much faster and therefore was selected as the algorithm for the 45WS lightning probability program.



ımple

The Numerical Integration Technique (Chan, 2011)

This numerical integration technique is one in which the miss distance is given by a non-central chi distribution with unequal variances (Chan, 2011). The covariance matrix corresponding to the bivariate Gaussian pdf in Eq. 1 is (Chan, 2008):

$$C = \begin{bmatrix} \sigma_x^2 & \rho_{xz}\sigma_x\sigma_z \\ \rho_{xz}\sigma_x\sigma_z & \sigma_z^2 \end{bmatrix} \tag{3}$$

When the correlation coefficient, ρ_{xz}, is not zero, there are undesirable off-diagonal terms that overly complicate the calculation. In order to eliminate these terms, the coordinate system (x,z) is rotated to a new coordinate system (x',z') such that the major and minor axes of the ellipse associated with the covariance are aligned along the coordinate axes and the new covariance matrix is (Chan, 2008):

$$C' = \begin{bmatrix} \sigma_{x'}^2 & 0 \\ 0 & \sigma_{z'}^2 \end{bmatrix} \tag{4}$$

The angle, θ, between the two coordinate systems is (Chan, 2008):

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\rho_{xz}\sigma_x\sigma_z}{\left(\sigma_x^2 - \sigma_z^2\right)}\right] \tag{5}$$

The KSC/CCAFS 4DLSS system does not provide the covariance matrix, but instead provides the semi-major axis, semi-minor axis, and the orientation of the semi-major axis of the 50% location error ellipse relative to north. Therefore the angle, θ, in Eq. 5 is found using geometry where θ is the angle between the semi-major axis of the lightning location uncertainty ellipse and line connecting the center of the lightning uncertainty ellipse and the center of the area of interest.

In the (x', z') system, the Eq. 1 pdf becomes (Chan, 2008)

$$f(x', z') = \frac{1}{2\pi\sigma_{x'}\sigma_{z'}} e^{-\frac{1}{2}\left[\left(\frac{x'}{\sigma_{x'}}\right)^2 + \left(\frac{z'}{\sigma_{z'}}\right)^2\right]} \tag{6}$$

and Eq. 2, the collision probability becomes (Chan, 2008)

$$P = \frac{1}{2\pi\sigma_{x'}\sigma_{z'}} \iint_{A'} e^{-\frac{1}{2}\left[\left(\frac{x'}{\sigma_{x'}}\right)^2 + \left(\frac{z'}{\sigma_{z'}}\right)^2\right]} dx'\,dz' \tag{7}$$

where

$$A' = A, r_{A'} = r_A, x'_p = x_e \cos\theta, z'_p = x_e \sin\theta \tag{8}$$

For spacecraft collision, x_e is the nominal distance of closest approach of the two colliding objects and (x'_p, z'_p) are the coordinates of the spacecraft relative to the debris. For lightning strike probability, x_e (the distance between the position of the center of the strike location ellipse and the position of the target area) is calculated using the Haversine distance formula.

The standard deviations in the new rotated coordinate system are calculated by dividing the semi-major and semi-minor axes of the 50% lightning positional confidence ellipse by the scaling constant used to scale standard error to the 50% confidence level. The scaling constant is:

$$k = \sqrt{-2 * \ln(1 - 0.50)} \tag{9}$$

The probability is given by (Chan, 2011)

$$P = \frac{1}{2\sqrt{2\pi}\sigma_H} \int_0^{\sqrt{W}} \left[e^{-(H-\mu_H)^2/2\sigma_H^2} + e^{-(H+\mu_H)^2/2\sigma_H^2} \right] \left[erf(Z_1) + erf(Z_2) \right] dH \tag{10}$$

where

$$Z_1 = \left[\sqrt{(W - H^2)} - \mu_K \right] / \sqrt{2}\sigma_K$$

$$Z_2 = \left[\sqrt{(W - H^2)} + \mu_K \right] / \sqrt{2}\sigma_K \tag{11}$$

The parameters μ_K and μ_H are the coordinates of the target circle in the (X', Z') coordinate system; and σ_K and σ_H are the standard deviations of the diagonalized covariance ellipse shown in Eq. 4. The derivation of equations (10) and (11) above is shown in further detail in (Chan, 2011). A detailed example of the calculations using a real-world case is provided in Appendix-A. The Excel Visual Basic code to implement these calculations is shown in Appendix-B.

Evaluation

The probability that any lightning strike is within any radius of any point of interest would be extremely difficult to estimate intuitively. As a result, given the high impact of the decisions on space launch operations, the tool developed for this application was extensively tested. Tests were conducted and are discussed in the following sections: 1) known mathematical solutions, and 2) examination of real-world events. The new technique passed all of the tests. Tests were also conducted to assure the probabilities calculated using the algorithm of Chan (2011) matched probabilities calculated using the algorithm of Patera (2001). The probabilities calculated from the two algorithms were identical and thus are not shown here.

Test Set 1

The first set of testing compared the lightning strike probability calculated using the 45WS lightning strike spreadsheet (which uses an adaptation of the numerical integration algorithm by Chan, 2011, to the corresponding circular probability from the CRC Handbook of Tables for Probability and Statistics (Beyer, 1968). Table 1 shows the probability from the new numerical integration technique for various inputs and the corresponding correct probability from the CRC Handbook. The values matched to within a tenth of a percent. These errors in the final digit may be due to round-off error.

Table 1. Calculated probability vs. CRC Handbook probability for various inputs

Semi-major axis (nmi)	Semi-minor axis (nmi)	Heading of semi-major axis from true North	Point Of Interest latitude	Point Of Interest longitude	Strike Latitude	Strike Longitude	Radius around Point Of Interest (nmi)	Calculated probability	CRC Handbook probability (Beyer, 1968)
3	3	15	28.6082	-80.6041	28.6995	-80.6041	3	0.095	0.095
3	3	15	28.6082	-80.6041	28.631	-80.6041	3	0.453	0.452
3	3	15	28.6082	-80.6041	28.608	-80.6041	3	0.500	0.499
1	1	15	28.6082	-80.6041	28.608	-80.6041	1	0.500	0.499
1	1	15	28.6082	-80.6041	28.631	-80.6041	1	0.200	0.200
1	1	15	28.6082	-80.6041	28.6995	-80.6041	1	0.000	0.000
1	1	15	28.6082	-80.6041	28.608	-80.6041	2	0.937	0.938

Test Set 2

The second type of testing analyzed six real-world lightning strikes near Space Launch Complex 39A on 3 August 2009. Figure 2 shows the spreadsheet used to generate the lightning report for those six strikes. Additional data on three of these six strikes are in Table 2. These strikes were selected because the closest point on the lightning position uncertainty ellipse was within 0.45 nautical miles of Launch Complex 39A, which is the key radius for assessing the need to inspect electronics for induced current damage to the Space Shuttle. Figures 3 through 5 are Google Maps depictions of three of these six strokes. In Figures 3-5, the black ellipse is the 99% lightning location uncertainty ellipse. The white circle is a 0.45 nmi radius around the point of interest. The probabilities for a small area around a facility, even for a nearby stroke, may appear to be surprisingly low. For example, figures 3 and 4 respectively illustrate a 53.8 % and 7.7% probability that the lightning strike occurred within the area of interest while figure 5 shows that a strike just 0.65 nautical miles away had only a 1.1% probability of being within the 0.45 nautical mile radius of Launch Complex 39A. All calculated probabilities are consistent with these real-world events.

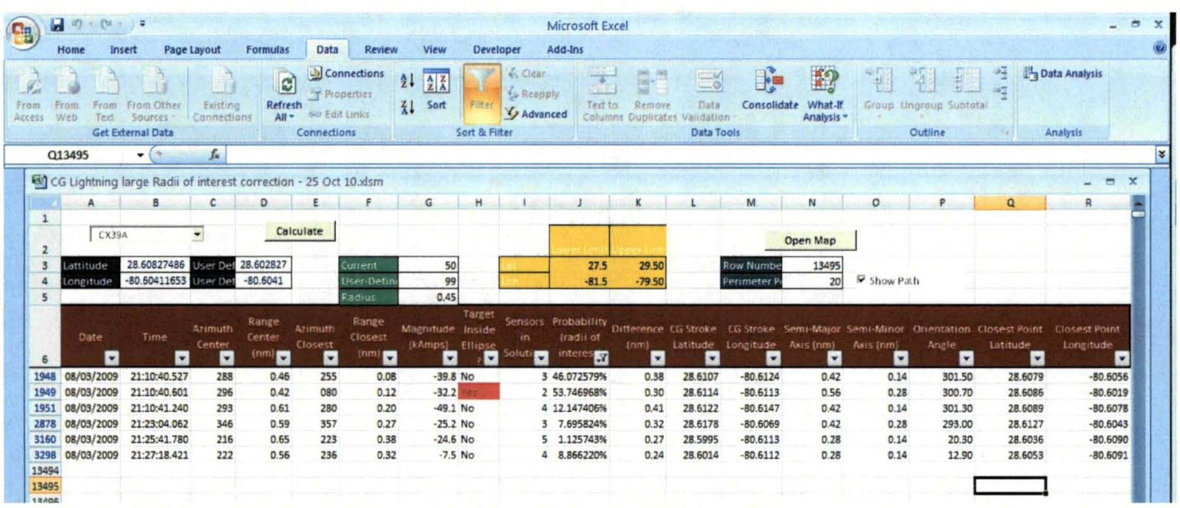


Figure 2. Sample of lightning strikes where the closest point on the lightning position uncertainty ellipse was within 0.45 nmi of Launch Complex 39A on 3 August 2009.

The KSC Electromagnetic Environmental Effects (EEE) Panel requested six more real-world lightning strikes be investigated. These were recently investigated lightning strikes near Launch Complexes 39A or 39B where there was camera verification of the location of the strike. The EEE Panel wanted to compare the results of the new facility-centric probabilistic technique to these cases where the true answers were known unambiguously. The data used for this analysis are in Table 3. Both 4DLSS and National Lightning Data Network (NLDN) cases were examined, depending upon which sensor system recorded the stroke. CGLSS strokes were obtained from 45WS 4DLSS. The NLDN usually provided flash data, so NLDN return stroke data were purchased as special StrikeNet[1] reports from Vaisala Corporation. This was done to match the return strokes routinely provided by 4DLSS. Figures 6 through 8 show the probability results from these cases. In Figures 6-8, the black ellipse is the 99% lightning location uncertainty ellipse. The white circle is a 0.45 nmi radius around the point of interest. As with the previous real-world tests, all calculated probabilities were consistent with these additional real-world events.

Table 2. Input values used for scenarios shown in Figures 3 through 5.

Figure	Semi-major axis of 50% confidence ellipse (km)	Semi-major axis of 50% confidence ellipse (km)	Confidence	Heading (from true North) of semi-major axis	Point of interest latitude (°N)	Point of interest longitude (°W)	Strike latitude (°N)	Strike longitude (°W)	Radius around point of interest (nmi)
3	0.4	0.2	0.99	300.7	28.60827	-80.6041	28.6114	-80.6113	0.45
4	0.3	0.2	0.99	293	28.60827	-80.6041	28.6178	-80.6069	0.45
5	0.2	0.1	0.99	20.3	28.60827	-80.6041	28.5995	-80.6113	0.45

[1] Vaisala, Inc., 2006, "Vaisala StrikeNet Information," URL: http://www.vaisala.com/files/StrikeNet-Brochure.pdf

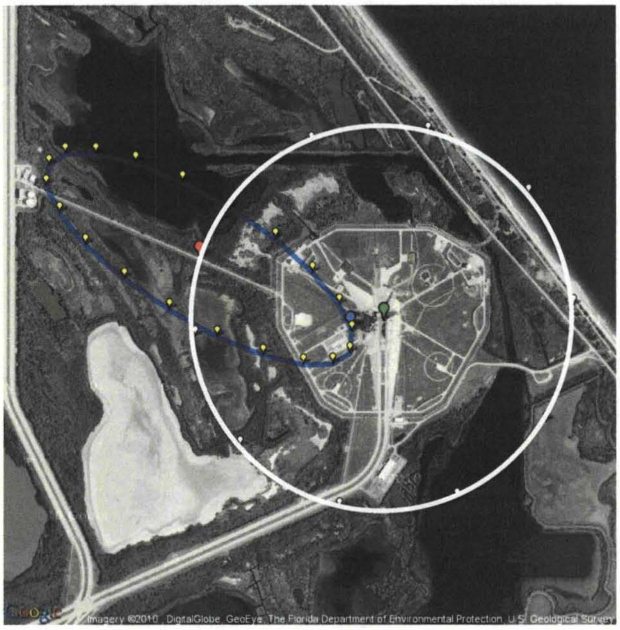


Figure 3. Google Maps visualization of the 99% confidence uncertainty ellipse for one of the closest lightning strikes to Complex 39A on 03 August 2009.

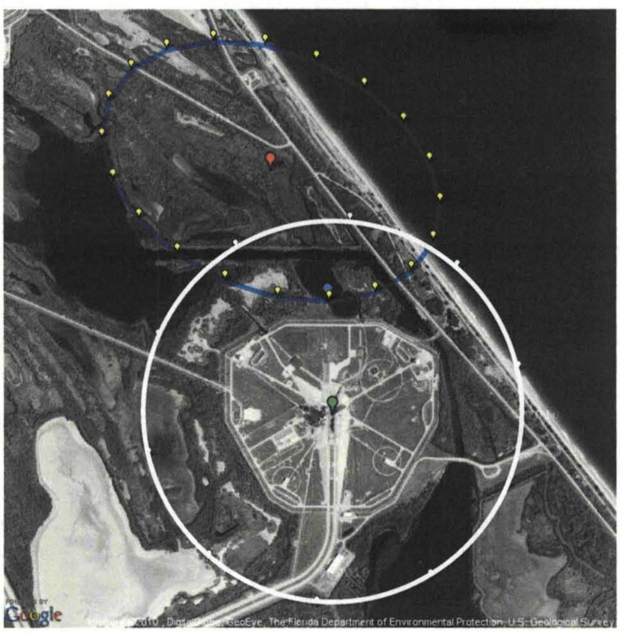


Figure 4. Google Maps visualization of the 99% confidence uncertainty ellipse for a lightning strike near Complex 39A on 03 August 2009.

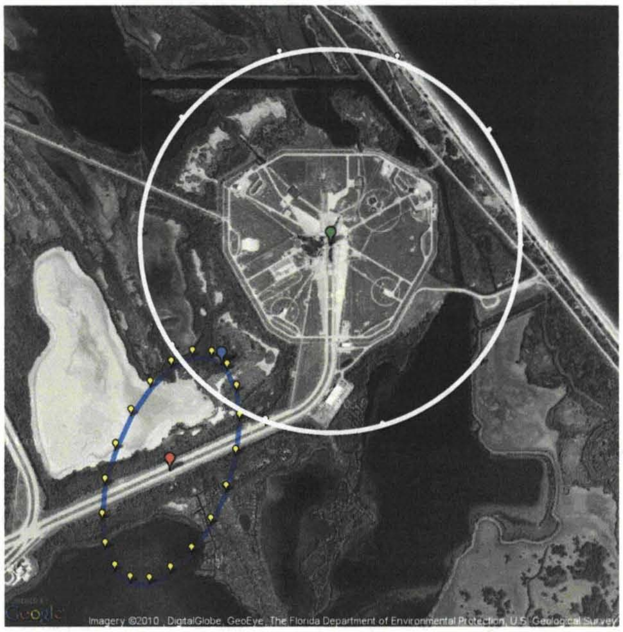


Figure 5. Google Maps visualization of the 99% confidence uncertainty ellipse for nearby lightning strike to Complex 39A on 03 August 2009.

Table 3. Input values used for scenarios shown in Figures 6 through 8.

Figure	Semi-major axis of 50% confidence ellipse (km)	Semi-major axis of 50% confidence ellipse (km)	Confidence	Heading (from true North) of semi-major axis	Point of interest latitude (°N)	Point of interest longitude (°W)	Strike latitude (°N)	Strike longitude (°W)	Radius around point of interest (nmi)
6	0.6	0.4	0.99	82	28.60827	-80.6041	28.6069	-80.6087	0.45
7	0.4	0.4	0.99	95	28.60827	-80.6041	28.6057	-80.6085	0.45
8	0.2	0.1	0.99	72	28.62716	-80.6275	28.6275	-80.6202	0.45

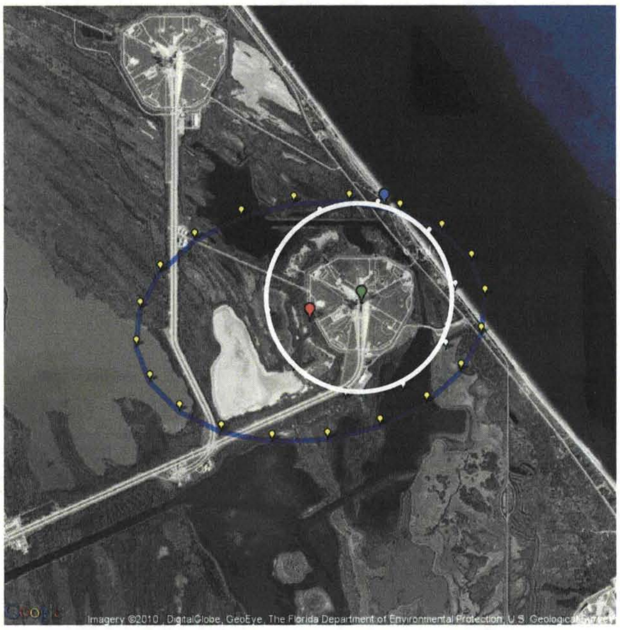


Figure 6. Illustrates a probability of 69.1% of a lightning strike of amplitude -43.0 kA detected by NLDN occurring 0.26 nmi from the center of Launch Complex 39A on 8/16/2009.

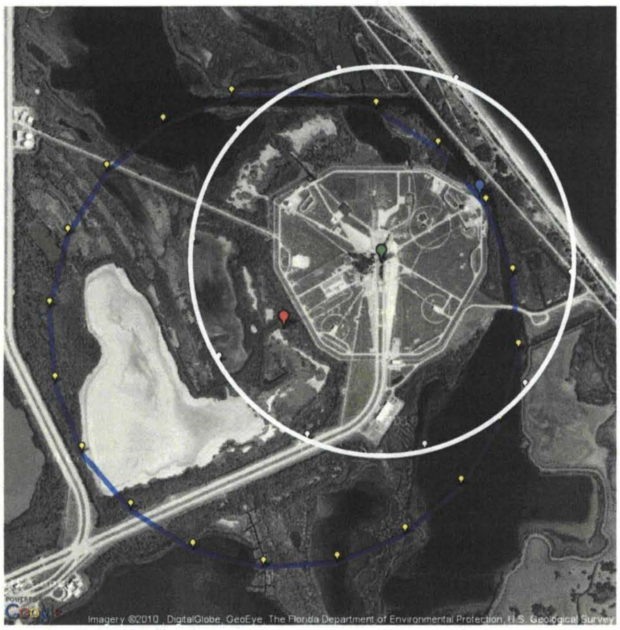


Figure 7. Illustrates a probability of 74.7% of a lightning strike of amplitude -71.4 kA detected by NLDN occurring 0.28 nautical miles from the center of Launch Complex 39A on 10/14/2009.

Figure 8. Illustrates a probability of 99.9996% of a lightning strike of amplitude -21.7 kA detected by CGLSS occurring 0.04 nmi from the center of Launch Complex 39B on 6/27/2009.

Other Applications

The techniques and methods described in this paper clearly have application reaching far beyond the space program uses for which they were designed. The list of potential applications is many and varied and would be of interest to anyone seeking information pertaining to probability of lightning strike locations, such as the power industry, aviation, or any industry sensitive to electrical overloads. This methodology is also applicable to forensic meteorology [13], where the question of whether lightning struck at or near a particular location is an issue in litigation. This technique could be applied to NLDN or any other cloud-to-ground lightning detection system that provides location error ellipses.

Conclusion

A probabilistic facility-centric approach to lightning location has been developed to calculate the probability that any nearby cloud-to-ground lightning stroke occurred within any radius of any point of interest. In practice, this provides the probability that a nearby lightning stroke was within a key distance of a facility, rather than within the error ellipses centered on the stroke. This process uses the bivariate Gaussian distribution of probability density provided by the current lightning location error ellipse for the most likely location of a lightning stroke and integrates it to determine the probability that the stroke is inside any specified radius. This new facility-centric technique was tested extensively and is much more useful to the space launch customers.

Appendix A – Example Lightning Probability Calculation

This appendix is an example of calculating the probability of any lightning stroke with a known error ellipse being within a circle of any radius around any point. It is provided to clarify the calculation process. An example calculation is shown in Table 4.

This example is a real-world event from a lightning strike near the Space Shuttle launch pad 39A at 02:35 GMT on 16 August 2009 (ref. Figure 6). Although the lightning data usually are from the cloud-to-ground component of the Four Dimensional Lightning Surveillance System (CG-4DLSS) (Murphy et al 2008, Roeder 2010) in this example a lightning stroke from the NLDN is used. We sometimes use StrikeNet reports that provide stroke data from the NLDN to double check the CG-4DLSS report. The lightning stroke input values are shown in row 1 of Table 3.

The following assumptions are applicable to the example calculation in Table 4. The location of Launch Pad 39A is 28.60827486 north latitude (or 0.499309 radians) and 80.60411653 west longitude (or -1.406807 radians). This is also the center of the circle in which the lightning probability will be calculated. The desired radius for probability of lightning around Launch Pad 39A is 0.45 nautical miles (nmi). This lightning stroke occurred on 16 August 2009 at 0235 GMT.

Table 4. Lightning strike probability calculation process.

Step	Action	Equation and other information	Calculation and Result
1	Convert semi-major and semi-minor axes from km to nmi	1 nmi = 1.852 km	0.6 km = 0.324 nmi semi-major axis 0.4 km = 0.216 nmi semi-minor axis
2	Calculate distance from lightning stroke to center of circle	Haversine Distance Formula: Distance = Earth Radius * C • Earth Radius = 3443.920086 nmi • $C = 2 * Atn2[\sqrt{1-A}, \sqrt{A})$ Atn2 is a two parameter arc tangent function which returns values in all four quadrants. • A = sin(dlat/2)*sin(dlat/2) + cos((target lat))* cos((stroke lat)) *sin(dlon/2)*sin(dlon/2) • dlat = latitude difference from target to stroke = 28.60827° - 28.6069° = 2.39959×10^{-5} (radians) • dlon = longitude difference from circle to stroke = -80.60411°- 80.6085° = $7.99967 \text{X } 10^{-5}$ (radians)	Distance = 3443.920086 * 7.4217×10^{-5} = 0.2556 nmi C = 2*Atn2{sqr(1 - 1.377×10^{-9}), sqr(1.377×10^{-9})} = 7.4217×10^{-5} A = sin(1.200×10^{-5}/2)* sin(1.200×10^{-5}/2)* + cos(0.4993)*cos(0.4993)* sin(4.000×10^{-5}/2) *sin(4.000×10^{-5}/2) = 1.3770×10^{-9}
3	Perform coordinate system rotation	• X = (Longitude of Target – Longitude of Stroke) * Cos (Latitude of Strike)	X = (-1.4068– (-1.4069) * Cos (0.4993)

	to eliminate the off-diagonal term in the covariance matrix.	• Z = Latitude of Target – Latitude of Stroke • $\theta = \alpha - ((\pi/2) - \mathrm{Atn2}(X,Z))$ • α is the orientation angle of the 50% lightning positional confidence ellipse • X' = miss distance*Cos(θ (coordinate system rotation angle)) • Z' = miss distance*Sin(θ)	= 7.0231 X 10^{-5} Z = 0.49931 – 0.49929 = 2.400 X 10^{-5} θ = 1.431 – 1.571 – Atn2(7.023 X 10^{-5}, 2.400 X 10^{-5}) = 0.1896 X' = 0.2556*Cos(0.1896) = 0.2510 Z' = 0.2556*Sin(0.1896) = 0.0482
4	Calculate the standard deviations in the new rotated coordinate system.	• $\sigma_{X'}$ = Semi-major axis of the 50% lightning positional confidence ellipse /elliptical scaling constant used to scale standard error to the 50% confidence level • $\sigma_{Z'}$ = Semi-minor axis of the 50% lightning positional confidence ellipse /elliptical scaling constant used to scale standard error to the 50% confidence level • Elliptical scaling constant, k, is: $\sqrt{-2 * \ln(1 - probability)}$	$k = \sqrt{-2 * \ln(1-0.50)}$ = 1.1774 $\sigma_{X'}$ = 0.3240/1.1774 = 0.2752 $\sigma_{Z'}$ = 0.2160/1.1774 = 0.1834
5	Calculate the probability that lightning stroke was within the target area of interest by performing a numerical integration using Simpson's rule of the lightning uncertainty ellipse over the area of the circle around the target of interest.	$P = \frac{1}{2\sqrt{2\pi}\sigma_H} \int_0^{\sqrt{W}} \left[e^{-(H-\mu_H)^2/2\sigma_H^2} + e^{-(H+\mu_H)^2/2\sigma_H^2} \right] \left[erf(Z_1) + erf(Z_2) \right] dH$ $Z_1 = \left[\sqrt{(W-H^2)} - \mu_K \right] / \sqrt{2}\sigma_K$ $Z_2 = \left[\sqrt{(W-H^2)} + \mu_K \right] / \sqrt{2}\sigma_K$ • μ_K and μ_H are the coordinates of the target circle in the (X', Z') coordinate system • σ_K and σ_H are equal to $\sigma_{X'}$ and $\sigma_{Z'}$ which are the standard deviations of the diagonalized covariance matrix. • W = Radius around target^2 • N = the number of iterations to perform in the integration (for this example, N is set to 200). • DH = $\sqrt{W}$ / N = integration step • B, C, and D are intermediate variables in	W = Radius around target^2 W = 0.45^2 = 0.2025 DH = $\sqrt{W}$ / N DH = $\sqrt{0.2025}$ /200 = 0.00225 B = $\sqrt{2}$ * $\sigma_{X'}$ B = $\sqrt{2}$ * 0.2752 = 0.3891 C = X'/ B C = 0.2510/0.3891 = 0.6451 D = 1 / (2 * $\sqrt{2\pi}$ * $\sigma_{Z'}$) D = 1/(2 * $\sqrt{2\pi}$ * 0.1834) = 1.0874 H = iteration no. * DH H = 199*0.00225 H = 0.4478

the algorithm corresponding to various parts of the probability equation shown above

- $B = \sqrt{2} * \sigma_{X'}$
- $C = X'/B$
- $D = 1/(2 * \sqrt{2\pi} * \sigma_{Z'})$

- A, H, z1, z2, E, F, Erf(z1), Erf(z2), Q and sum are intermediate variables in the algorithm corresponding to various parts of the probability equation shown above
- A loop is performed for j = 1 to 199. This example is shown for j = 199.
  - Sum = 0
  - Begin Loop here: H = j* DH
  - $A = \sqrt{W - H^2}$
  - z1 = A/B – C
  - z2 = A/B + C
  - Erf(x) = error function = $\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$
  - $E = (H - Z')^2 / (2 * \sigma_{Z}'^2)$
  - $F = (H + Z')^2 / (2 * \sigma_{Z}'^2)$
  - $Q(j) = (e^{-E} + e^{-F}) * (\mathrm{Erf}(Z1) + \mathrm{Erf}(Z2))$
  - $sum = sum + (3 - (-1)^j) * Q(j)$
- sum = sum + Q(0) + Q(N)
- Probability = D*sum*DH/3

$A = \sqrt{W - H^2}$
$A = \sqrt{0.2025 - 0.44775^2}$
$A = 4.494 \times 10^{-2}$

z1 = A / B – C
$z1 = 4.494 \times 10^{-2} / 0.3891 - 0.6451$
z1 = -0.5296
z2 = A / B + C
$z2 = 4.494 \times 10^{-2} / 0.3891 + 0.6451$
z2 = 0.7606

Erf(Z1) = ErrorFunction(z1)
Erf(-0.5296) = -0.5461

Erf(Z2) = ErrorFunction(z2)
Erf(0.7606) = 0.7179
$E = (H - Z')^2 / (2 * \sigma_{Z}'^2)$
$E = (0.4478 - 0.0482)^2/(2*0.1834^2)$
E = 2.372
$F = (H + Z')^2 / (2 * \sigma_{Z}'^2)$
$F = (0.4478 + 0.0482)^2/(2*0.1834^2)$
F = 3.654

$Q(j) = (e^{-E} + e^{-F}) * (\mathrm{Erf}(Z1) + \mathrm{Erf}(Z2))$
$Q(199) = (e^{-2.372} + e^{-3.654}) * (-0.5461 + 0.7179) = 2.0467 \times 10^{-2}$

$sum = sum + (3 - (-1)^j) * Q(j)$
$sum = sum + (3 - (-1)^{199}) * 2.047 \times 10^{-2}$
sum = 844.8952
End Loop

sum = sum + Q(0) + Q(N)
sum = 844.8952 + 2.9361 + 0 = 847.8317

Probability = D * sum * DH / 3
Probability = 1.0874 * 847.8317 * 0.00225 / 3 =

0.6914

$E = (H - Z')^2 / (2 * \sigma_Z'^2)$
$E = (0.4478 - 0.0482)^2/(2*0.1834^2)$
$E = 2.372$
$F = (H + Z')^2 / (2 * \sigma_Z'^2)$
$F = (0.4478 + 0.0482)^2/(2*0.1834^2)$
$F = 3.654$

$Q(j) = (e^{-E} + e^{-F}) * (Erf(Z1) + Erf(Z2))$
$Q(199) = (e^{-2.372} + e^{-3.654})* (-0.5461+ 0.7179) = 2.0467 X 10^{-2}$

$sum = sum + (3 - (-1)^{j}) * Q(j)$
$sum = sum + (3 - (-1)^{199}) * 2.047 X 10^{-2}$
sum = 844.8952
End Loop

sum = sum + Q(0) + Q(N)
sum = 844.8952 + 2.9361 + 0 = 847.8317

Probability = D * sum * DH / 3
Probability = 1.0874 * 847.8317 * 0.00225 / 3 = 0.6914

Appendix B – Excel VBA Macro for 45th Weather Squadron Lightning Spreadsheet that Calculates the Probability of any Nearby Lightning Stroke being Inside any Radius of Any Point of Interest

Conversions Module

```
Const pie As Double = 3.14159265358979
Const ecc As Double = 0.081819190842622 'eccentricity (e^)
Const kmPerMile As Double = 1.852
----------------------------------------------------------
Function toRad(Degrees As Double) As Double
   'Converts degrees to radians
   toRad = Degrees * pie / 180
End Function
----------------------------------------------------------
Function toDeg(Radians As Double) As Double
   'Converts radians to degrees
   toDeg = Radians * 180 / pie
End Function
----------------------------------------------------------

Function applyConfidence(length As Double, newSigma As Double, currentSigma As Double) As
Double
   'convert length based on confidence interval
   applyConfidence = length * newSigma / currentSigma
End Function
Function revertConfidence(length As Double, newSigma As Double, currentSigma As Double) As
Double
   'convert length based on confidence interval
   revertConfidence = length * currentSigma / newSigma
End Function
----------------------------------------------------------

Function kmToNmi(kilometers As Double) As Double
   'Converts kilometers to nautical miles
   kmToNmi = kilometers / kmPerMile
End Function
----------------------------------------------------------

Function geodToECEF(Lat As Double, Lon As Double) As Variant
   'convert lat-lon to EFG
   Dim ret(2) As Double
   ret(0) = getE(Lat, Lon)
   ret(1) = getF(Lat, Lon)
   ret(2) = getG(Lat)
   geodToECEF = ret
End Function
----------------------------------------------------------
```

```
Function EcefToGeod(E As Double, F As Double, g As Double) As Variant
   'convert EFG coordinate to lat-lon
   Dim ret(1) As Double 'array of (lat,lon)
   ret(0) = Atn(g / ((1 - ecc ^ 2) * Sqr(E ^ 2 + F ^ 2))) * 180 / pie
   ret(1) = Atn2(E, F) * 180 / pie
   EcefToGeod = ret
End Function
-----------------------------------------------------------
```

Functions Module

```
Const pie As Double = 3.14159265358979
Const eRad As Double = 3443.920086      'radius of earth in nautical miles
Const ecc As Double = 0.081819190842622 'eccentricity (e^)
-----------------------------------------------------------

Public Function Atn2(ByVal X As Double, ByVal Y As Double) As Double
   Dim at As Double

   If X = 0 Then
      If Y < 0 Then
         Atn2 = -pie / 2
      ElseIf Y > 0 Then
         Atn2 = pie / 2
      Else
         Atn2 = 0
      End If
   ElseIf Y = 0 Then
      If X > 0 Then
         Atn2 = 0
      ElseIf X < 0 Then
         Atn2 = pie
      End If
   Else
      at = Atn(Y / X)
      If X < 0 And Y < 0 Then
         Atn2 = at - pie
      ElseIf X < 0 And Y > 0 Then
         Atn2 = at + pie
      Else
         Atn2 = at
      End If
   End If
End Function
-----------------------------------------------------------

Function arcCos(X As Double)
   arcCos = Atn((1 - X ^ 2) ^ 0.5 / X)
End Function
-----------------------------------------------------------
```

```
Function addPoints(p1() As Double, p2() As Double) As Variant
   'add two EFG coordinates together
   Dim ret(2) As Double
   ret(0) = p1(0) + p2(0)
   ret(1) = p1(1) + p2(1)
   ret(2) = p1(2) + p2(2)
   addPoints = ret
End Function
----------------------------------------------------------

Function getEPointDiffs(eLat As Double, eLon As Double, A As Double, B As Double, phi As Double, t
As Double) As Variant
   'get EFG differentials for a point on ellipse at t radians
   Dim ret(2) As Double
   Dim x1 As Double, y1 As Double
   x1 = getX(A, B, 0, 90 - phi, t)
   y1 = getY(A, B, 0, 90 - phi, t)
   ret(0) = -x1 * Sin(eLon * pie / 180) - y1 * Sin(eLat * pie / 180) * Cos(eLon * pie / 180)
   ret(1) = x1 * Cos(eLon * pie / 180) - y1 * Sin(eLon * pie / 180) * Sin(eLat * pie / 180)
   ret(2) = y1 * Cos(eLat * pie / 180)
   getEPointDiffs = ret
End Function
----------------------------------------------------------

Function getFocalPointDiffs(eLat As Double, eLon As Double, A As Double, B As Double, phi As
Double) As Variant
   'get EFG differentials for focal point of ellipse
   Dim ret(2) As Double
   Dim fociDist As Double, x1 As Double, y1 As Double
   fociDist = Sqr(A ^ 2 - B ^ 2)

   x1 = fociDist * Cos(toRad(90 - phi))
   y1 = fociDist * Sin(toRad(90 - phi))
   ret(0) = -x1 * Sin(eLon * pie / 180) - y1 * Sin(eLat * pie / 180) * Cos(eLon * pie / 180)
   ret(1) = x1 * Cos(eLon * pie / 180) - y1 * Sin(eLon * pie / 180) * Sin(eLat * pie / 180)
   ret(2) = y1 * Cos(eLat * pie / 180)

   getFocalPointDiffs = ret
End Function
----------------------------------------------------------
Function getDistGeod(lat1 As Double, lon1 As Double, lat2 As Double, lon2 As Double) As Double
   'spherical
   Dim rLat1 As Double, rLat2 As Double, rLon1 As Double, rLon2 As Double

   rLat1 = toRad(lat1)
   rLat2 = toRad(lat2)
   rLon1 = toRad(lon1)
   rLon2 = toRad(lon2)
```

```
getDistGeod = getN((lat1 + lat2) / 2) * arcCos(Cos(rLat1) * Cos(rLon1) * Cos(rLat2) * Cos(rLon2) +
Cos(rLat1) * Sin(rLon1) * Cos(rLat2) * Sin(rLon2) + Sin(rLat1) * Sin(rLat2)) / 180 * pie

End Function
------------------------------------------------------------

Function greatCircle(lat1 As Double, lon1 As Double, lat2 As Double, lon2 As Double) As Double
    'great circle distance in NM
    Dim rLat1 As Double, rLat2 As Double, rLon1 As Double, rLon2 As Double

    rLat1 = toRad(lat1)
    rLat2 = toRad(lat2)
    rLon1 = toRad(lon1)
    rLon2 = toRad(lon2)
    greatCircle = (eRad) * arcCos(Cos(rLat1) * Cos(rLat2) * Cos(rLon2 - rLon1) + Sin(rLat1) *
Sin(rLat2))

End Function
------------------------------------------------------------

Function Haversine(lat1 As Double, lon1 As Double, lat2 As Double, lon2 As Double) As Double
    'Calculates distance between two latitude and longitude points by the haversine formula in NM
    Dim dlat As Double, dlon As Double, C As Double, A As Double


    dlat = toRad(lat1) - toRad(lat2)
    dlon = toRad(lon1) - toRad(lon2)
    A = Sin(dlat / 2) * Sin(dlat / 2) + Cos(toRad(lat1)) * Cos(toRad(lat2)) * Sin(dlon / 2) * Sin(dlon / 2)
    C = 2 * Atn2(Sqr(1 - A), Sqr(A))
    Haversine = eRad * C

End Function
Function getX(A As Double, B As Double, H As Double, phi As Double, t As Double) As Double
    'get the X cartesian coordinate of a point on the ellipse at t radians
    getX = H + A * Cos(t) * Cos(phi * pie / 180) - B * Sin(t) * Sin(phi * pie / 180)
End Function
------------------------------------------------------------

Function getY(A As Double, B As Double, k As Double, phi As Double, t As Double) As Double
    'get the Y cartesian coordinate of a point on the ellipse at t radians
    getY = k + B * Sin(t) * Cos(phi * pie / 180) + A * Cos(t) * Sin(phi * pie / 180)
End Function
------------------------------------------------------------

Function getN(Lat As Double) As Double
    'get radius of the earth as a given latitude, adjusted for eccentricity
    getN = eRad / (1 - ((ecc ^ 2) * Sin(Lat * pie / 180) ^ 2)) ^ 0.5
End Function
------------------------------------------------------------

Function getE(Lat As Double, Lon As Double) As Double
```

```
  'get E coordinate from lat-lon
  getE = getN(Lat) * Cos(Lat * pie / 180) * Cos(Lon * pie / 180)
End Function
-----------------------------------------------------------

Function getF(Lat As Double, Lon As Double) As Double
  'get F coordinate from lat-lon
  getF = getN(Lat) * Cos(Lat * pie / 180) * Sin(Lon * pie / 180)
End Function
-----------------------------------------------------------

Function getG(Lat As Double) As Double
  'get G coordinate from lat
  getG = getN(Lat) * (1 - ecc ^ 2) * Sin(Lat * pie / 180)
End Function
-----------------------------------------------------------
Function getEPoint(ecLat As Double, ecLon As Double, A As Double, B As Double, phi As Double, t As
Double) As Variant
  'get lat-lon of a point on ellipse at t radians
  Dim ret(1) As Double
  Dim eC() As Double
  Dim point() As Double
  Dim diffs() As Double
  eC = geodToECEF(ecLat, ecLon)
  diffs = getEPointDiffs(ecLat, ecLon, A, B, phi, t)
  point = addPoints(eC, diffs)
  getEPoint = EcefToGeod(point(0), point(1), point(2))
End Function
-----------------------------------------------------------

Function ePointLat(ecLat As Double, ecLon As Double, A As Double, B As Double, phi As Double, t As
Double) As Double
  'get lat of a point on ellipse at t radians
  Dim geod() As Double
  geod = getEPoint(ecLat, ecLon, A, B, phi, t)
  ePointLat = geod(0)
End Function

Function ePointLon(ecLat As Double, ecLon As Double, A As Double, B As Double, phi As Double, t
As Double) As Double
  'get lon of a point on ellipse at t radians
  Dim geod() As Double
  geod = getEPoint(ecLat, ecLon, A, B, phi, t)
  ePointLon = geod(1)
End Function
-----------------------------------------------------------

Function getAzimuth(lat1 As Double, lon1 As Double, lat2 As Double, lon2 As Double) As Double
  'get azimuth between two points
  Dim angle As Double
```

```
  angle = (Atn2(Cos(toRad(lat1)) * Sin(toRad(lat2)) - Sin(toRad(lat1)) * Cos(toRad(lat2)) *
Cos(toRad(lon2) - toRad(lon1)), Sin(toRad(lon2) - toRad(lon1)) * Cos(toRad(lat2))) * 180 / pie)
  If (angle = 0) Then
    getAzimuth = 360
  Else
    getAzimuth = (angle + 360) Mod 360
  End If
End Function
------------------------------------------------------------
```

Module 2 Code

```
Dim point As New Strike
Const passWord As String = "PAEc0d3m0nk3y$"
------------------------------------------------------------

Sub POI_Change()
  Calculate
End Sub
------------------------------------------------------------

Sub calculateRow()

  Application.EnableCancelKey = xlDisabled
  Worksheets("Results").protect Contents:=False, passWord:=passWord
  Dim ecef() As Double
  Dim interestPoint(1) As Double ' (lat,lon)
  Dim rowNumber_data As Double
  Dim rowNumber_result As Double
  Dim rowData As Variant
  Dim AOI(3) As Double
  Dim lastRow As Double
  If Worksheets("Results").FilterMode = True Then
    Worksheets("Results").ShowAllData
  End If
  lastRow = Worksheets("Results").Cells(Rows.Count, "A").End(xlUp).row
  If (lastRow > 6) Then
    Worksheets("Results").Range("A7:R" & lastRow).Select
    Selection.Delete
  End If

  AOI(0) = Range("latLower")
  AOI(1) = Range("latUpper")
  AOI(2) = Range("lonLower")
  AOI(3) = Range("lonUpper")

  interestPoint(0) = Range("poiLat")
  interestPoint(1) = Range("poiLon")
  ecef = geodToECEF(interestPoint(0), interestPoint(1))
```

```
    Call point.setStatics(interestPoint, ecef, Range("newSigma"), Range("currentSigma"),
Range("radius"))
    Dim xlCalc As XlCalculation
    xlCalc = Application.Calculation
    Application.Calculation = xlCalculationManual
  ' On Error GoTo CalcBack

    lastRow = Worksheets("Data").Cells(Rows.Count, "A").End(xlUp).row
    rowNumber_result = 7
    For rowNumber_data = 2 To lastRow
        rowData = Worksheets("Data").Range("A" & rowNumber_data & ":I" & rowNumber_data)
        If (rowData(1, 3) > AOI(0) And rowData(1, 3) < AOI(1) And rowData(1, 4) > AOI(2) And
rowData(1, 4) < AOI(3)) Then
            Call point.init(rowData(1, 3), rowData(1, 4), rowData(1, 6), rowData(1, 7), rowData(1, 8), True)

            Worksheets("Results").Cells(rowNumber_result, 1) = rowData(1, 1)
            Worksheets("Results").Cells(rowNumber_result, 2) = rowData(1, 2)
            Worksheets("Results").Cells(rowNumber_result, 3) = point.centerAzimuth
            Worksheets("Results").Cells(rowNumber_result, 4) = point.centerRange
            Worksheets("Results").Cells(rowNumber_result, 5) = point.criticalAzimuth
            Worksheets("Results").Cells(rowNumber_result, 6) = point.criticalRange
            Worksheets("Results").Cells(rowNumber_result, 7) = rowData(1, 5)
            Worksheets("Results").Cells(rowNumber_result, 8) = point.isInside
            Worksheets("Results").Cells(rowNumber_result, 9) = rowData(1, 9)
            Worksheets("Results").Cells(rowNumber_result, 10) = point.Prob_simpson
            Worksheets("Results").Cells(rowNumber_result, 11) = point.rangeDifference
            Worksheets("Results").Cells(rowNumber_result, 12) = point.Lat
            Worksheets("Results").Cells(rowNumber_result, 13) = point.Lon
            Worksheets("Results").Cells(rowNumber_result, 14) = point.A
            Worksheets("Results").Cells(rowNumber_result, 15) = point.B
            Worksheets("Results").Cells(rowNumber_result, 16) = point.phi
            Worksheets("Results").Cells(rowNumber_result, 17) = point.criticalLat
            Worksheets("Results").Cells(rowNumber_result, 18) = point.criticalLon
            'Worksheets("Results").Cells(rowNumber_result, 19) = point.Prob_rician
            'Worksheets("Results").Cells(rowNumber_result, 20) = point.Prob_simpson

            rowNumber_result = rowNumber_result + 1
        End If
    Next rowNumber_data
    lastRow = Worksheets("Results").Cells(Rows.Count, "A").End(xlUp).row
    Worksheets("Results").Range("A7:A" & lastRow).NumberFormat = "MM/DD/YYYY"
    Worksheets("Results").Range("B7:B" & lastRow).NumberFormat = "HH:mm:ss.000"
    Worksheets("Results").Range("C7:C" & lastRow).NumberFormat = "000"
    Worksheets("Results").Range("D7:D" & lastRow).NumberFormat = "0.00"
    Worksheets("Results").Range("E7:E" & lastRow).NumberFormat = "000"
    Worksheets("Results").Range("F7:F" & lastRow).NumberFormat = "0.00"
    Worksheets("Results").Range("J7:J" & lastRow).NumberFormat = "0.000000%"
    Worksheets("Results").Range("K7:K" & lastRow).NumberFormat = "0.00"
    Worksheets("Results").Range("L7:M" & lastRow).NumberFormat = "0.0000"
    Worksheets("Results").Range("N7:P" & lastRow).NumberFormat = "0.00"
    Worksheets("Results").Range("Q7:R" & lastRow).NumberFormat = "0.0000"
```

```
  'Worksheets("Results").Range("S7:T" & lastRow).NumberFormat = "0.000000%"

  If (lastRow > 6) Then

  With Worksheets("Results").Range("H7:H" & lastRow).Select
    Selection.FormatConditions.Add Type:=xlTextString, String:="Yes", _
    TextOperator:=xlContains
  Selection.FormatConditions(Selection.FormatConditions.Count).SetFirstPriority
  With Selection.FormatConditions(1).Font
    .Color = -16777024
    .TintAndShade = 0
  End With
  With Selection.FormatConditions(1).Interior
    .PatternColorIndex = xlAutomatic
    .Color = 5263615
    .TintAndShade = 0
  End With
  End With

  End If

  Application.Calculation = xlCalc
  Worksheets("Results").EnableAutoFilter = True
  Worksheets("Results").protect Contents:=True, passWord:=passWord, AllowFiltering:=True,
AllowSorting:=True, userInterfaceOnly:=True

  MsgBox ("Data Calculation Complete")

  Exit Sub
--------------------------------------------------------------

CalcBack:
  Worksheets("Results").protect Contents:=True, passWord:=passWord, AllowFiltering:=True,
AllowSorting:=True, userInterfaceOnly:=True
  Application.Calculation = xlCalc

End Sub
--------------------------------------------------------------
Sub openBrowser()
  Dim rowNumber As Double
  Dim N As Integer
  Dim ecef() As Double
  Dim interestPoint(1) As Double ' (lat,lon)
  Dim showPath As Boolean
  showPath = ActiveSheet.showPath.value

  interestPoint(0) = Range("poiLat")
  interestPoint(1) = Range("poiLon")

  ecef = geodToECEF(interestPoint(0), interestPoint(1))
```

```
  Call point.setStatics(interestPoint, ecef, Range("newSigma"), Range("currentSigma"),
Range("radius"))
  rowNumber = Range("nRow")
  N = Range("nPoints")
  Call point.init(Cells(rowNumber, 12), Cells(rowNumber, 13), Cells(rowNumber, 14),
Cells(rowNumber, 15), Cells(rowNumber, 16), False)

  Call point.setURL(N, showPath)

  'Range("urlString") = point.URL

  Set browser = CreateObject("InternetExplorer.Application")

  If (Len(point.URL) > 1950) Then
    MsgBox ("Your request contains too many characters.  Please select a smaller number of perimeter
points or turn on off the path attribute.")
  Else
    browser.Navigate (point.URL)
    browser.Visible = True
  End If

End Sub
-----------------------------------------------------------
```

Probability Module

```
Const pie As Double = 3.14159265358979

Function RicianIntegral(U As Double, V As Double, m As Integer)
  'This subroutine computes the Rician Integral Pm when u, v and m are given.
  'It is preferred because both the exponentials are outside the summation loops.
  'This formulation gives essentially the same numerical value as the other one.
  Dim tm As Double, rm As Double, sm As Double, sumTm As Double, sumTmSm As Double
  On Error GoTo overflow
  tm = 1
  rm = 1
  sm = 1
  sumTm = 1
  sumTmSm = 1

  For i = 1 To m
    tm = tm * V / (2 * i)
    rm = rm * U / (2 * i)
    sm = sm + rm
    sumTm = sumTm + tm
    sumTmSm = sumTmSm + tm * sm
  Next i
  RicianIntegral = Exp(-V / 2) * sumTm - Exp(-(U + V) / 2) * sumTmSm
  Exit Function
overflow:
```

```
  RicianIntegral = 0
End Function
------------------------------------------------------------

Function numericIntegral(Xprime As Double, Zprime As Double, SigmaXp As Double, SigmaZp As
Double, Xe As Double, Ra As Double, N As Integer)
  Dim x1 As Double, z1 As Double, r1 As Double, x2 As Double, z2 As Double, r2 As Double
  Dim R As Double, sum As Double, dTheta As Double, thisPhi As Double, Q As Double

  DPhi = 2 * pie / N

  sum = 0
  thisPhi = 0
  x1 = (Xprime + Ra) * SigmaZp / SigmaXp
  z1 = Zprime

  For i = 1 To N
    thisPhi = thisPhi + DPhi
    x2 = (Xprime + Ra * Cos(thisPhi)) * SigmaZp / SigmaXp
    z2 = Zprime + Ra * Sin(thisPhi)
    r1 = Sqr(x1 * x1 + z1 * z1)
    r2 = Sqr(x2 * x2 + z2 * z2)
    dTheta = Application.WorksheetFunction.Asin((x1 * z2 - x2 * z1) / (r1 * r2))
    R = (r1 + r2) / 2
    sum = sum + Exp(-(R / SigmaZp) ^ 2 / 2) * dTheta
    x1 = x2
    z1 = z2
  Next i

  Q = -sum / (2 * pie)
  If Xe < Ra Then
    numericIntegral = 1 + Q
  ElseIf Xe = Ra Then
    numericIntegral = 1 / 2 + Q
  Else
    numericIntegral = Q
  End If

End Function
------------------------------------------------------------

Function simpsonIntegral(Xprime As Double, Zprime As Double, SigmaXprime As Double,
SigmaZprime As Double, Ra As Double, N As Integer)

  'This program computes the collision probability between two objects.
  'The integration is performed using Simpson's Rule.
  Dim Q(1000) As Double, P As Double
  Dim A As Double, B As Double, C As Double, D As Double, E As Double, F As Double, W As
Double
  Dim H As Double, DH As Double
```

```
pi = 3.14159265358979

B = Sqr(2) * SigmaXprime
C = Xprime / B
D = 1 / (2 * Sqr(2 * pi) * SigmaZprime)
W = Ra ^ 2
DH = Sqr(W) / N
'Compute Q(I) array from 0 to (N-1)
For j = 0 To (N - 1)
   H = j * DH

   A = Sqr(W - H ^ 2)

   z1 = A / B - C
   z2 = A / B + C

   ErfZ1 = ErrorFunction(z1)
   ErfZ2 = ErrorFunction(z2)

   E = (H - Zprime) ^ 2 / (2 * SigmaZprime ^ 2)

   F = (H + Zprime) ^ 2 / (2 * SigmaZprime ^ 2)


   Q(j) = (Exp(-E) + Exp(-F)) * (ErfZ1 + ErfZ2)

Next j

Q(N) = 0

'Compute the integral by Simpson's Rule
sum = 0
For j = 1 To (N - 1)
   sum = sum + (3 - (-1) ^ j) * Q(j)

Next j

sum = sum + Q(0) + Q(N)

P = D * sum * DH / 3

simpsonIntegral = P

End Function
```

```
Function ErrorFunction(X)
   'This subroutine computes the Error Function when X is given.
   Dim Z, SqrtPi, tm, sum, Max, m, ErfX, ErfcX
   SqrtPi = 1.77245385090552
```

```
  Z = Abs(X)
  If Z <= 4 Then
    tm = Z
    sum = tm
    Max = Int(20 * Z) + 1
    For m = 1 To Max
      tm = -tm * Z * Z * (2 * m - 1) / (m * (2 * m + 1))
      sum = sum + tm
    Next m
    ErfX = sum * 2 / SqrtPi
  ElseIf Z <= 5.736 Then
    tm = 1
    sum = 1
    For m = 1 To 1
      tm = -tm * (2 * m - 1) / (2 * Z ^ 2)
      sum = sum + tm
    Next m
    ErfcX = sum * Exp(-Z ^ 2) / (SqrtPi * Z)
    ErfX = 1 - ErfcX
  Else
    ErfX = 1
  End If
  ErrorFunction = Sgn(X) * ErfX
End Function
------------------------------------------------------------
```

Strike Class Module

```
Const pie As Double = 3.14159265358979
Const eRad As Double = 3443.920086      'radius of earth in nautical miles
Const ecc As Double = 0.081819190842622 'eccentricity (e^)

Private p_poi_geo() As Double           'lat lon point of interest [lat,lon]
Private p_poi_ecef() As Double          'EFG point of interest  [E,F,G]
Private p_newSigma As Double            'user defined confidence level
Private p_currentSigma As Double        '50% confidence

Private p_strike_geo(1) As Double       'lat lon of strike [lat,lon]
Private p_major As Double               'major axis in NM
Private p_minor As Double               'minor axis in NM
Private p_phi As Double                 'stroke orientation in degrees

Private p_a As Double                   'scaled major axis in NM
Private p_b As Double                   'scaled minor axis in NM
Private p_strike_ecef() As Double       'EFG strike [E,F,G]
Private p_criticalT As Double           'angle of closest point in radians
Private p_critical_geo() As Double      'lat lon of closest point [lat,lon]
Private p_critical_ecef(2) As Double    'EFG of closest point [E,F,G]
Private p_aziCenter As Double           'azimuth to strike center
Private p_rangeCenter As Double         'distance to strike center
```

```
Private p_aziCritical As Double        'azimuth to closest point
Private p_rangeCritical As Double      'distance to closest point
Private p_rangeDifference As Double
Private p_isInside As String
Private p_URL As String
Private p_foci1() As Double
Private p_foci2() As Double

Private p_prob_rician As Double
Private p_prob_numeric As Double
Private p_prob_simpson As Double

Private p_radius As Double             'radius around point of interest
------------------------------------------------------------

Public Sub setStatics(poi_geo As Variant, poi_ecef As Variant, newSigma As Double, currentSigma As
Double, radius As Double)
  p_poi_geo = poi_geo
  p_poi_ecef = poi_ecef
  p_newSigma = Sqr(-2 * Log(1 - newSigma / 100))
  p_currentSigma = Sqr(-2 * Log(1 - currentSigma / 100))
  p_radius = radius
End Sub
------------------------------------------------------------

Public Sub init(Lat, Lon, major, minor, phi, adjustAxis As Boolean)
  On Error GoTo initErr


  p_strike_geo(0) = Lat
  p_strike_geo(1) = Lon
  p_phi = phi
  If (adjustAxis) Then
    'input major and minor are in KM, and unscaled
    p_major = kmToNmi(CDbl(major))
    p_minor = kmToNmi(CDbl(minor))


    If p_major = 0 Then
      p_major = 0.05
    End If
    If p_minor = 0 Then
      p_minor = 0.05
    End If
    p_a = applyConfidence(p_major, p_newSigma, p_currentSigma)
    p_b = applyConfidence(p_minor, p_newSigma, p_currentSigma)


    Else
    'input major and minor are in NM and scaled
    p_a = CDbl(major)
```

```
        p_b = CDbl(minor)
        If p_a = 0 Then
          p_a = 0.1
        End If
        If p_b = 0 Then
          p_b = 0.1
        End If
        p_major = revertConfidence(p_a, p_newSigma, p_currentSigma)
        p_minor = revertConfidence(p_b, p_newSigma, p_currentSigma)
    End If

    p_strike_ecef = geodToECEF(p_strike_geo(0), p_strike_geo(1))
    Call setT
    p_critical_geo = getEPoint(p_criticalT)
    p_aziCenter = getAzimuth(p_poi_geo(0), p_poi_geo(1), p_strike_geo(0), p_strike_geo(1))
    p_aziCritical = getAzimuth(p_poi_geo(0), p_poi_geo(1), p_critical_geo(0), p_critical_geo(1))
    p_rangeCenter = Haversine(p_poi_geo(0), p_poi_geo(1), p_strike_geo(0), p_strike_geo(1))
    p_rangeCritical = Haversine(p_poi_geo(0), p_poi_geo(1), p_critical_geo(0), p_critical_geo(1))
    p_rangeDifference = p_rangeCenter - p_rangeCritical
    Dim diffs() As Double
    Dim point() As Double

    diffs = getFocalPointDiffs(p_strike_geo(0), p_strike_geo(1), p_a, p_b, p_phi)
    point = addPoints(p_strike_ecef, diffs)
    p_foci1 = EcefToGeod(point(0), point(1), point(2))
    diffs(0) = -diffs(0)
    diffs(1) = -diffs(1)
    diffs(2) = -diffs(2)
    point = addPoints(p_strike_ecef, diffs)
    p_foci2 = EcefToGeod(point(0), point(1), point(2))
    Call setProbability
    Exit Sub
initErr:
    asd = 0
End Sub
-----------------------------------------------------------

Public Property Get A()
    A = p_a
End Property
-----------------------------------------------------------

Public Property Get B()
    B = p_b
End Property
-----------------------------------------------------------

Public Property Get criticalT()
    criticalT = p_criticalT
End Property
```

```
------------------------------------------------------------

Public Property Get criticalLat()
   criticalLat = p_critical_geo(0)
End Property
------------------------------------------------------------

Public Property Get criticalLon()
   criticalLon = p_critical_geo(1)
End Property
------------------------------------------------------------

Public Property Get criticalAzimuth()
   criticalAzimuth = p_aziCritical
End Property
------------------------------------------------------------

Public Property Get centerAzimuth()
   centerAzimuth = p_aziCenter
End Property
------------------------------------------------------------

Public Property Get criticalRange()
   criticalRange = p_rangeCritical
End Property
------------------------------------------------------------

Public Property Get centerRange()
   centerRange = p_rangeCenter
End Property
------------------------------------------------------------

Public Property Get rangeDifference()
   rangeDifference = p_rangeDifference
End Property
------------------------------------------------------------

Public Property Get URL()
   URL = p_URL
End Property
------------------------------------------------------------

Public Property Get Lat()
   Lat = p_strike_geo(0)
End Property
------------------------------------------------------------

Public Property Get Lon()
   Lon = p_strike_geo(1)
End Property
------------------------------------------------------------
```

```
Public Property Get phi()
   phi = p_phi
End Property
-----------------------------------------------------------

Public Property Get Prob_rician()
   Prob_rician = p_prob_rician
End Property
-----------------------------------------------------------

Public Property Get Prob_numeric()
   Prob_numeric = p_prob_numeric
End Property
-----------------------------------------------------------

Public Property Get Prob_simpson()
   Prob_simpson = p_prob_simpson
End Property
-----------------------------------------------------------

Public Property Get isInside()
   Dim d1 As Double, d2 As Double, t0() As Double, d3 As Double, t1() As Double
   d1 = Haversine(p_foci1(0), p_foci1(1), p_poi_geo(0), p_poi_geo(1))
   d2 = Haversine(p_foci2(0), p_foci2(1), p_poi_geo(0), p_poi_geo(1))
   t0 = getEPoint(0)
   t1 = getEPoint(pie)
   da = Haversine(t1(0), t1(1), t0(0), t0(1))
   If ((d1 + d2) < (da)) Then
      isInside = "Yes"
   Else
      isInside = "No"
   End If
End Property
-----------------------------------------------------------

Private Sub setT()
   Dim thisT As Double, minDist As Double, thisDist As Double, negDist As Double, posDist As
Double, returnT As Double, step As Double
   Dim pointLat As Double, pointLon As Double
   Dim point() As Double

   returnT = -pie
   point = getEPoint(returnT)
   minDist = Haversine(p_poi_geo(0), p_poi_geo(1), point(0), point(1))

   For thisT = -pie To pie Step pie / 4
      point = getEPoint(thisT)
      thisDist = Haversine(p_poi_geo(0), p_poi_geo(1), point(0), point(1))
      If (thisDist < minDist) Then
         minDist = thisDist
```

```
            returnT = thisT
        End If
    Next thisT
    step = pie / 4
    While step > pie / (2 ^ 16)
        point = getEPoint(returnT - step)
        negDist = Haversine(p_poi_geo(0), p_poi_geo(1), point(0), point(1))
        point = getEPoint(returnT + step)
        posDist = Haversine(p_poi_geo(0), p_poi_geo(1), point(0), point(1))
        If (negDist < minDist) Then
            returnT = returnT - step
            minDist = negDist
        ElseIf posDist < minDist Then
            returnT = returnT + step
            minDist = posDist
        End If
        step = step / 2
    Wend
    p_criticalT = returnT
End Sub
-----------------------------------------------------------

Private Function getEPoint(t As Double) As Variant
    Dim ret(1) As Double
    Dim point() As Double
    Dim diffs() As Double
    diffs = getEPointDiffs(p_strike_geo(0), p_strike_geo(1), p_a, p_b, p_phi, t)
    point = addPoints(p_strike_ecef, diffs)
    getEPoint = EcefToGeod(point(0), point(1), point(2))
End Function
-----------------------------------------------------------

Private Function getCPoint(t As Double) As Variant
    Dim ret(1) As Double
    Dim point() As Double
    Dim diffs() As Double
    diffs = getEPointDiffs(p_strike_geo(0), p_strike_geo(1), p_radius, p_radius, p_phi, t)
    point = addPoints(p_poi_ecef, diffs)
    getCPoint = EcefToGeod(point(0), point(1), point(2))
End Function
-----------------------------------------------------------

Public Sub setURL(nPoints As Integer, showPath As Boolean)
    Dim thisT As Double, tStep As Double
    Dim point() As Double

    p_URL = "http://maps.google.com/staticmap?size=640x640&maptype=satellite&markers="
    tStep = 2 * pie / nPoints
    For i = 1 To nPoints
        thisT = -pie + i * tStep
        point = getEPoint(thisT)
```

```
      p_URL = p_URL & Round(point(0), 5) & "," & Round(point(1), 5) & "," & "tinyyellow|"
      'p_URL = p_URL & point(0) & "," & point(1) & "," & "tinyyellow|"
   Next i
   tStep = 2 * pie / 10
   For i = 1 To 10
      thisT = -pie + i * tStep
      point = getCPoint(thisT)
      p_URL = p_URL & Round(point(0), 5) & "," & Round(point(1), 5) & "," & "tinywhite|"
      'p_URL = p_URL & point(0) & "," & point(1) & "," & "tinywhite|"
   Next i
   p_URL = p_URL & p_critical_geo(0) & "," & p_critical_geo(1) & "," & "smallpurple|"
   'p_URL = p_URL & p_foci1(0) & "," & p_foci1(1) & "," & "smallblue|"
   'p_URL = p_URL & p_foci2(0) & "," & p_foci2(1) & "," & "smallblue|"
   p_URL = p_URL & p_poi_geo(0) & "," & p_poi_geo(1) & "," & "smallgreen|"
   p_URL = p_URL & p_strike_geo(0) & "," & p_strike_geo(1) & "," & "smallred"
   If (showPath) Then
      p_URL = p_URL & "&path="
      tStep = 2 * pie / nPoints
      For i = 1 To nPoints + 1
         thisT = -pie + i * tStep
         point = getEPoint(thisT)
         p_URL = p_URL & Round(point(0), 5) & "," & Round(point(1), 5) & "|"
      Next i
   End If
   p_URL = p_URL &
"&sensor=false&key=ABQIAAAAdwjRqRR8FuOdpA0oimTJCBSOxDKO5lwx0GB6dfDkgLOxwdqZC
hSForDLWhvadXUn6EEI6WZWYt853w"
End Sub
-------------------------------------------------------------

Public Sub setProbability()
   On Error GoTo probErr
   'This program computes the probability of of a lightning strike occurring within a
   'specified distance of an asset of interest.
   'The cross-section is a circle.

   Dim alpha As Double, LonP As Double, LatP As Double, LonS As Double, LatS As Double
   Dim Xprime As Double, Zprime As Double, SigmaXp As Double, SigmaZp As Double, sigma As
Double
   Dim X As Double, Z As Double, theta As Double
   Dim U As Double, V As Double
   Dim m As Integer, AspectRatio As Double

   p_prob_rician = 0
   p_prob_numeric = 0

   If (p_major = 0 Or p_minor = 0) Then
      Exit Sub
   End If

   LatP = toRad(p_poi_geo(0))      'Latitude of POI (radians)
```

```
  LonP = toRad(p_poi_geo(1))     'Longitude of POI (radians)
  LatS = toRad(p_strike_geo(0))  'Latitude of Strike Spot (radians)
  LonS = toRad(p_strike_geo(1))  'Longitude of Strike Spot (radians)
  alpha = toRad(p_phi)           'Heading of Semi-Major Axis of 50% Confidence Ellipse (radians)


  AspectRatio = p_major / p_minor

  X = (LonP - LonS) * Cos(LatS)

  Z = LatP - LatS


  theta = alpha - ((pie / 2) - Atn2(X, Z))


  Xprime = p_rangeCenter * Cos(theta)
  Zprime = p_rangeCenter * Sin(theta)

  SigmaXp = p_major / 1.177410023

  SigmaZp = p_minor / 1.177410023

  sigma = Sqr(SigmaXp * SigmaZp)

  'Compute Rician Integral
  'U = (p_radius / sigma) ^ 2
  'V = Xprime ^ 2 / SigmaXp ^ 2 + Zprime ^ 2 / SigmaZp ^ 2
  'm = Int(Sqr(U * V)) + 1
  'p_prob_rician = RicianIntegral(U, V, m)

  'skip numeric method if aspect ratio is low
  'If (m < 25 And AspectRatio < 10) Then
    'Exit Sub
  'End If
  'Use numerical method for computing lightning strike probability:
  'p_prob_numeric = numericIntegral(Xprime, Zprime, SigmaXp, SigmaZp, p_rangeCenter, p_radius,
1000)

  p_prob_simpson = simpsonIntegral(Xprime, Zprime, SigmaXp, SigmaZp, p_radius, 200)
  Exit Sub
probErr:

End Sub
------------------------------------------------------------
```

References

Alfano, S., "Satellite Collision Probability Enhancements," J. Guidance, Control, and Dynamics, Vol. 29, No. 3, May-June 2006, pp. 588-592.

Alfano, S., "Review of Conjunction Probability Methods for Short-Term Encounters," *AAS/AIAA Space Flight Mechanics Meeting*, Paper AAS 07-148, Sedona, AZ, 2007, 44 pp.

Austin, E., "Forensic Meteorology and Climatology: A Rapidly Emerging Niche," *Third Annual CCM Forum, 90th Annual American Meteorological Society Meeting*, Paper 1.2, Atlanta, GA, 2010.

Beyer, W. H. (ed.), *CRC Handbook of Tables for Probability and Statistics*, The Chemical Rubber Co., Cleveland, OH, 1968, pp. 151-157.

Chan, F. Kenneth, *Spacecraft Collision Probability*, The Aerospace Press, El Segundo, CA, 2008, Chaps. 3, 4, 5, 6

Chan, F. Kenneth, "Miss Distance – Generalized Variance Non-Central Chi Distribution," *AAS/AIAA Space Flight Mechanics Meeting*, Paper AAS 11-175, New Orleans, LA, 2011, 12 pp.

Flinn, F. C., Roeder, W. P., Buchanan, M.D., and McNamara, T. M., "Lightning Reporting at 45th Weather Squadron: Recent Improvements," 21st International Lightning Detection Conference, Orlando, FL, 2010a, 18 pp.

Flinn, F. C., Roeder, W. P., Pinter, D. F., Holmquist, S. M., Buchanan, M. D., McNamara, T. M., McAleenan, M., Winters, K.A., Gemmer, P. S., Fitzpatrick, M. E., and Gonzalez, R. D., "Recent Improvements in Lightning Reporting at 45th Weather Squadron," *14th Conference on Aviation, Range, and Aerospace Meteorology*, Paper 7.3, Atlanta, GA, 2010b, 14 pp.

Leleux, D., Spencer, R., Zimmerman, P., Propst, C., Heilman, W., Frisbee, J., and Wortham, M., "Probability-Based Space Shuttle Collision Avoidance," *Space Ops 2002 Conference*, Houston, TX, 2002, 20 pp.

Murphy, M. J., Cummins, K.L., Demetriades, N. W. S., and Roeder, W. P., "Performance of the New Four-Dimensional Lightning Surveillance System (4DLSS) at the Kennedy Space Center/Cape Canaveral Air Force Station Complex," *13th Conference on Aviation, Range, and Aerospace Meteorology*, Paper 7.6, New Orleans, LA, 2008, 18 pp.

Patera, Russell, P., "General Method for Calculating Satellite Collision Probability," *J. Guidance, Control, and Dynamics*, Vol. 24, No. 4, Jul.-Aug. 2001, pp. 716-722.

Roeder, W. P., Weems, J. W., and Wahner, P. B., "Applications of the Cloud-to-Ground Lightning Surveillance System Database," *1st Conference on the Meteorological Applications of Lightning Data*, San Diego, CA, 2005, 5 pp.

Roeder, W.P., "The Four Dimensional Lightning Surveillance System," 21st International Lightning Detection Conference, Orlando, FL, 2010, 15 pp.