Computing Collision Probability Using Linear Covariance and Unscented Transforms

Sun Hur-Diaz[1] and Matthew Ruschmann[2]
Emergent Space Technologies, Inc., Greenbelt, MD 20770

Martin Heyne[3]
Odyssey Space Research, LLC, Houston, TX 77058

and

Michael Phillips[4]
Emergent Space Technologies, Inc., Lakewood, CO 80226

For a cluster of satellites flying in close proximity, the probability of collision (Pc) is of great interest. For a given cluster configuration (geometry), navigation noise, controller, and maneuver execution error, the most reliable way to compute Pc is through Monte Carlo simulations. However, Monte Carlo requires running a large number of cases to accurately determine Pc. This is time-consuming and usually not practical for the low level of Pc that may be desired. An alternative to Monte Carlo that requires much less computational resources involves the use of linear covariance to propagate the position and velocity dispersions of the cluster satellites. This method however is limited by its linear assumptions and is unstable for nonlinear problems that can arise in cluster flight. The use of unscented transforms for covariance propagation is shown to be more stable in this case. A method to incorporate the effects of navigation noise, closed-loop control, and maneuver execution error is developed. A sample cluster scenario is evaluated using the covariance method with a hybrid method of computing Pc that combines the Mahalanobis distance metric and the maximum instantaneous probability. The results are shown to match the Monte Carlo results with a high confidence interval.

Nomenclature

Pc	=	probability of collision
LC	=	linear covariance
MC	=	Monte Carlo
x	=	state
f	=	state time derivative function
$\Phi(t_{i+1}, t_i)$	=	state transition matrix from time t_i to time t_{i+1}
P	=	state covariance
L	=	number of states
λ, κ	=	scaling parameters
α, β	=	tuning parameters
χ	=	sigma points
W	=	weights
u	=	controller output
d_m	=	Mahalanobis distance

[1] Chief Engineer, 6411 Ivy Lane, Suite 303, Greenbelt, MD 20770, AIAA Senior Member.
[2] Aerospace Engineer, 6411 Ivy Lane, Suite 303, Greenbelt, MD 20770, AIAA Member.
[3] Sr. GN&C Systems Engineer, 1120 NASA Parkway, Suite 505, Houston, TX 77058, AIAA Senior Member.
[4] Aerospace Engineer, 355 S. Teller Street, Suite 200 (Room No. 222), Lakewood, CO 80226, AIAA Member.

I. Introduction

FOR missions involving multiple spacecraft flying in close proximity, accurate computation of the collision probability between any two spacecraft is of importance. Mission objectives dictate whether the spacecraft fly in precise formation, for example for synthetic aperture radar, radio interferometry, and distributed imaging, or in a loose cluster where the spacecraft are flying close enough for communications between the spacecraft or possibly for energy transfer. For simplicity in this paper, multiple spacecraft flying in close proximity is referred to as a cluster regardless of the mission objective.

Collision probability is one metric for evaluating the goodness of a cluster configuration design for a given mission. It is also an important consideration in the design of the control system for orbit maintenance, the navigation system, and the propulsion system. The mission will dictate what collision probability is acceptable, and this in turn will affect the design trades among the various subsystems of the cluster.

The most accurate method to compute the probability of collision (Pc) between any two spacecraft is through Monte Carlo (MC) simulations. For a given initial uncertainty of their position and velocity, a given control strategy, if any, and navigation error, the initial conditions are randomly generated, and the trajectory is simulated. This method requires a large computational resource to run a sufficient number of random cases to determine Pc with sufficient level of confidence. For the low level of Pc that might be acceptable for a given mission, the number of cases can be computationally prohibitive. For example, given a mean Pc of 2e-5, 56,000 cases must be simulated to obtain a 95% confidence interval upper bound of 1e-4.

Alternatively, there are numerous methods of computing the collision probability that are based on propagation of the position uncertainty statistics.[1,2,3,6,7] To compute the collision probability using these methods, an accurate determination of the position uncertainty covariance matrix is required for each spacecraft in the cluster. For this purpose, linear covariance analysis can be utilized.[4] Typically, analytical expressions of the system dynamics and the control system are derived in order to apply these techniques. For complicated systems, these analytical expressions are difficult, if not impossible, to derive making this method difficult to implement in a general sense. Furthermore, its use of the state transition matrix limits its accuracy for highly nonlinear systems.

In this paper, we develop a numerical methodology that applies the basic concept of the linear covariance analysis, but is flexible for application to a variety of problems. We further expand the utility of this tool through the use of unscented transforms which improves the accuracy for highly nonlinear systems.

First we develop the linear covariance concept with the state transition matrix and describe how both closed-loop control and navigation errors are incorporated. Then we show how the unscented transforms can be applied, and show an example with validation using the Monte Carlo method.

II. Methodology

The approach described in this paper is to compute the probability of collision from the information about the statistics associated with each module's position. To this end, a simulation framework was developed that computes the covariance matrix required for the Pc calculation for any given scenario.

A. Simulation Framework

Our simulation framework consists of four major blocks: scenario definition, environment and controller models, the covariance generator, and the Pc calculator. The scenario definition, the environment model, and the controller are inputs to the covariance generator. The scenario definition block contains scenario data such as cluster configuration, number of satellites in the cluster, their hard body radii, initial conditions, and navigation uncertainty.

The environment and the controller are user specified functions and executed from within the covariance generator. The environment function specifies the fidelity of the orbit dynamics propagation such as the order and the degree of the Earth's geo-potential model, the atmospheric drag model, solar radiation pressure model, and the third-body gravitation model. The controller function defines how the maneuvers are computed and executed. If the control is open-loop, then the nominal maneuvers are executed irrespective of the state of the satellites. If the control is closed-loop, the maneuvers are computed based on the estimated state of the satellites from the navigation system. In either case, the maneuver execution error can be incorporated.

B. Covariance Generator

The covariance generator is primarily based on the linear covariance (LC) methodology as described in Geller[4] where it is designed to produce the same statistical results as a Monte Carlo simulation without performing thousands of propagations for a given scenario definition. The covariance computed represents the uncertainty of the position and the velocity of a module relative to its nominal trajectory, which is first computed without any of

the random components such as maneuver execution error or navigation error. The state transition matrix is utilized to propagate the initial uncertainty covariance to the maneuver time. At the maneuver time, the uncertainty of the maneuver due to navigation error is added to the propagated covariance as well as any uncertainty associated with the maneuver execution error. Note that this methodology assumes that the system is linear. For small perturbations and for short durations, these assumptions hold up well, and a later section shows that the LC-generated covariances statistically match the MC-generated covariances fairly well. We have found that in some cases, the LC method produces divergent covariances. For these cases, we have implemented unscented transforms for the propagation of the covariance.

C. Covariance Matrix

Most non-MC methods of computing probability of collision require processing of the probability density function (PDF) usually assumed to be normal (Gaussian) and summarized as a mean and a covariance representing the uncertainty of the state with respect to its nominal state. There are at least three types of covariances associated with a given problem: true dispersion covariance, true navigation error covariance, and estimated navigation error covariance.

Figure 1 depicts the covariance (red ellipses) that represents the true dispersion from the nominal trajectory. This is the covariance used in the computation of the true probability of collision. It is a combination of the propagated initial dispersion and the dispersion due to maneuvers based on estimated states containing navigation error. Dispersion due to actuation error can be included as well.

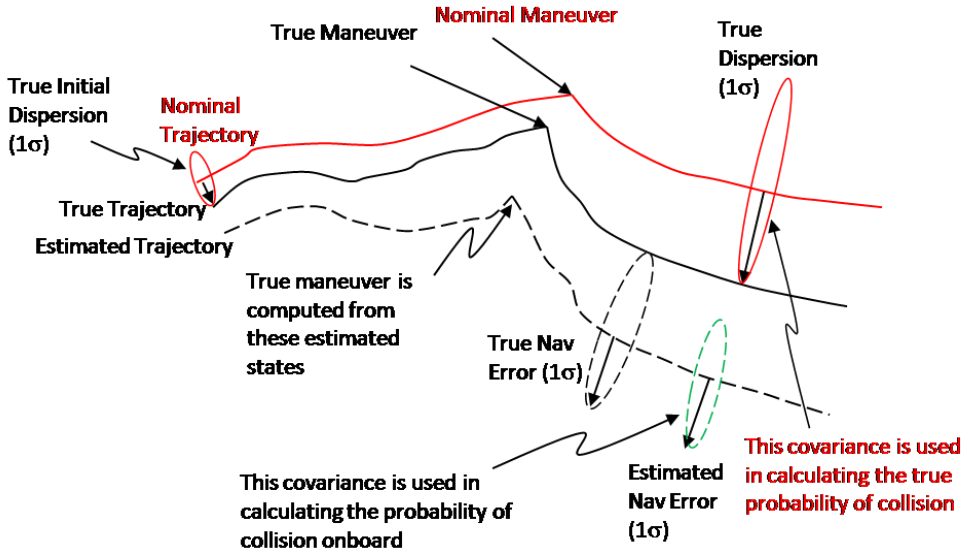

Figure 1. Illustration of different covariances.

The other two covariances of interest are shown in Figure 1, as well. One corresponds to the true navigation error and the other to the estimated navigation error. If the navigation filter is well-tuned, the two can agree well, but in general, neither matches the true dispersion which is required for Pc evaluation. Operationally, however, the estimated navigation error covariance is all the information we have about the uncertainty of the states, and it is used in the onboard prediction of collision probability. Note that this estimated collision probability is different from the true collision probability computed from the true dispersion as described above. For evaluation of the goodness of a given trajectory profile, the true collision probability is of interest.

D. Linear Covariance Propagation

The LC formulation requires linearization of the system models. Accordingly, we define the system matrix as:

$$F \equiv \left.\frac{\partial f}{\partial x}\right|_{x=x_{NOM}} \tag{1}$$

where x is the state, typically consisting of position and velocity, and the derivative is evaluated at some nominal state x_{NOM}. The state and its covariance are propagated using the following equations:

$$\begin{aligned} \dot{x}(t) &= f(x(t),t) \\ \dot{\Phi}(t_{i+1},t_i) &= F(t_i)\Phi(t_{i+1},t_i) \\ P_{i+1} &= \Phi(t_{i+1},t_i)P_i\Phi^T(t_{i+1},t_i) \end{aligned} \tag{2}$$

where $\Phi(t_{i+1},t_i)$ is the state transition matrix from time t_i to time t_{i+1}.

In order to develop a tool that is flexible for application to a variety of dynamical systems, the state transition matrix in the covariance generator is computed by numerical differencing of the system propagation function, which we call the 'blackbox'. Numerical differencing alleviates the need to derive the analytical expression of the Jacobian matrix of the system function. The blackbox includes the truth environment, navigation, controller, and bus models such as maneuver execution error. It takes the state at time t_i and computes the state at a future time t_{i+1}.

The j-th column of the state transition matrix is computed as follows:

$$\Phi(i+1,i)(:,j) = \frac{x(i+1,j) - x(i+1)_{NOM}}{x_j(i) - x_j(i)_{NOM}} \tag{3}$$

where $x(i+1,j)$ corresponds to the output of the blackbox when the j-th element of the input state has been perturbed. The most difficult part of the numerical differencing is determining the appropriate step size

$$dx_j = x_j(i) - x_j(i)_{NOM} \tag{4}$$

for each perturbation of the j-th state. In our implementation, the step sizes were determined from iteration. If the system is linear, the state transition matrix is not as sensitive to the step size. If the system is nonlinear, it can be quite sensitive. The use of unscented transform ameliorates this problem.

E. Unscented Transform Method

In many instances, the nonlinear effects of the system are large enough to make the above linear covariance method less accurate. To better account for the nonlinear effects in the covariance propagation, the unscented transforms can be used to include the higher-order non-linear dynamic effects.[5]

In this implementation, the state vector and the error covariance are used to create a collection of representative vectors with the same statistical properties as the state and state error covariance. These equivalent state vectors are referred to as sigma points with an assigned weight for each vector. The individual vectors are propagated through the complete nonlinear dynamic model of the system. Based on the assigned weights a new propagated state and covariance are obtained. In order to arrive at the mean and covariance of a variable, a set of weights is required. Given an initial mean, provided by the nominal trajectory of a satellite, and its initial covariance

$$P_0 = E\left[\left(x_0 - (x_0)_{NOM}\right)\left(x_0 - (x_0)_{NOM}\right)^T\right] \tag{5}$$

the set of sigma points is obtained by combining the Cholesky decomposition of the covariance matrix with the mean, which is the nominal state, according to

$$\chi_{i-1} = \left[(x_{i-1})_{NOM} \quad (x_{i-1})_{NOM} \pm \left(\sqrt{(L+\lambda)P_j}\right)_{i-1}\right] \quad \text{for } j = \{1,...,L\} \tag{6}$$

where $\sqrt{P_j}$ is the j-th column of the Cholesky decomposition of the covariance matrix, and λ is a scalingparameter. The total number of sigma points is $2L+1$ where L is the number of states.

For the propagation, each sigma point state is assigned a weight according to

$$W_0^m = \frac{\lambda}{L+\lambda} \tag{7}$$

$$W_0^c = \frac{\lambda}{L+\lambda} + (1-\alpha+\beta) \tag{8}$$

$$W_k^m = W_k^c = \frac{\lambda}{2(L+\lambda)} \tag{9}$$

where

$$\lambda = \alpha^2(L-\kappa) - L \tag{10}$$

with α controlling the spread of the sigma points, β incorporating distribution knowledge and κ as a secondary scaling parameter. The propagation step requires the propagation of the sigma points through the blackbox. Process noise covariance can be included, if desired.

The propagated state and covariance are obtained by combining the set of propagated sigma points χ_i with their appropriate weights yielding

$$\bar{x}_i = \sum_{k=0}^{2L} W_k^m \chi_{i,k} \tag{11}$$

$$P_i = \sum_{k=0}^{2L} W_k^c \left(\chi_{i,k} - \bar{x}_i\right)\left(\chi_{i,k} - \bar{x}_i\right)^T \tag{12}$$

The sigma points are generated and propagated for each time step of the covariance propagation.

F. Accounting for Maneuvers

Maneuvers can affect the state dispersion covariance in two ways: 1) maneuver execution error, and 2) maneuver variation due to navigation noise. As stated previously, the controller is included in the system blackbox. If the control is open loop, then the nominal planned maneuvers are executed without correction, and the state covariance is propagated through the maneuver. The change in the dispersion due to navigation noise is not accounted for. If the control is closed-loop, the maneuvers are computed based on the estimated states from the navigation system, therefore the effect of the navigation noise on the state uncertainty will have to be taken into account in the covariance propagation.

1. Maneuver Execution Error

For both open-loop and closed-loop, maneuver execution error can be included. For calculating probability of collision using linear covariance methods, Schiff found that taking into account the presence of the maneuver in the computation of the state and the corresponding state transition matrix while accounting for the maneuver uncertainty by applying process noise was the most accurate.[10] Without loss of generality, we assume impulsive maneuvers and corresponding update to the covariance is

$$P(i)^+ = P(i)^- + \frac{\partial(x_i)}{\partial(u_i)} P_u(i) \left(\frac{\partial(x_i)}{\partial(u_i)}\right)^T \tag{13}$$

where $\frac{\partial(x_i)}{\partial(u_i)}$ is the impulsive maneuver sensitivity matrix $\begin{bmatrix} 0_{3x3} \\ I_{3x3} \end{bmatrix}$, for blackbox states consisting of position and velocity, and $P_u(i)$ is a 3x3 covariance matrix of the maneuver execution error.

2. Navigation Noise

At maneuver times, the contribution of the navigation noise is added to the propagated covariance matrix. The navigation noise affects the states through the controller, where $\hat{x}_i$ is the estimated state with navigation noise, and u_i is the controller output. For our implementation, we assume that the control is in the form of an impulsive maneuver command. The covariance is updated using the following equation:

$$P(i)^+ = P(i)^- + \frac{\partial(x_i)}{\partial(u_i)} \frac{\partial(u_i)}{\partial(\hat{x}_i)} P_{nav}(i) \left(\frac{\partial(u_i)}{\partial(\hat{x}_i)} \right)^T \left(\frac{\partial(x_i)}{\partial(u_i)} \right)^T \tag{14}$$

where P_{nav} is the covariance associated with the navigation noise, and $\frac{\partial(u_i)}{\partial(\hat{x}_i)}$ is the sensitivity of the controller to the navigation state. As in the system state transition matrix, this can be computed by numerical differencing of the controller function:

$$\frac{\partial(u_i)}{\partial(\hat{x}_i)}(:, j) = \frac{u(i, j) - u(i)_{NOM}}{\hat{x}_j(i) - \hat{x}_j(i)_{NOM}} \tag{15}$$

where $\hat{x}_j(i)$ is the *j*-th element of the perturbed navigation state and $u(i, j)$ corresponds to the output of the controller for that particular input state with the *j*-th element perturbed. Alternatively as in the covariance propagation, unscented transforms can be applied for this sensitivity matrix as well.

A more general application of the navigation noise is given by Geller[4] where an augmented system consisting of both navigation and control is considered. In our implementation, such a coupled system can be represented as the blackbox.

III. Monte Carlo Validation of Covariance Computation

The linear covariance approach to computing the true dispersion covariance matrix was validated through Monte Carlo analysis. For a given spacecraft and for a given Monte Carlo run, the simulation proceeds as shown in Figure 2.

The nominal trajectory is first computed without navigation noise where dxo and dx are zero in the above block diagram. Then for each Monte Carlo run, the initial condition is sampled from the initial uncertainty matrix, and the navigation noise is sampled for the estimated state used in the controller. The states are saved at a fixed interval, and the covariance over *N* number of runs at a given time *i* is computed as follows:

$$P(i) = \frac{\sum_{j=1}^{N} (x(i) - x_{NOM}(i))(x(i) - x_{NOM}(i))^T}{N - 1} \tag{16}$$

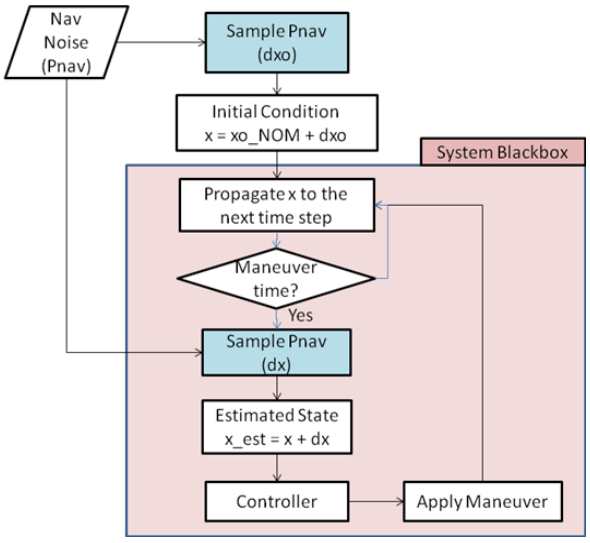

Figure 2. Monte Carlo simulation for covariance.

A. Example

We consider a station-keeping example of four satellites flying at 300 km altitude in circular reference orbit of 42 degree inclination. The cluster configuration was designed such that the satellites are passively safe with a minimum inter-satellite distance of 100 meters as shown in Figure 3.

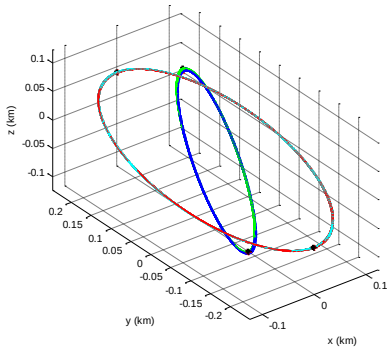

Figure 3. A sample four-satellite cluster.

The isotropic one-sigma relative navigation noise is 1 meter in position and 5 cm/sec in velocity. The motion of the satellites is affected by a 12x12 Earth gravitational field and atmospheric drag. Because of the low altitude and the high drag environment, luni-solar gravity and solar radiation pressure are ignored. The ballistic coefficients among the satellites vary widely with values of 20, 80, 140, 200 kg/m^2, and the hard body radius of all the satellites is 4 meters. The control strategy is to maintain the cluster altitude and the relative orbit geometry. A 500 second control cycle was implemented to correct for perturbations and bring the satellites to the nominal relative orbit geometry. To compute the probability of the collision over 50 orbits, we apply the linear covariance methodology described in this paper to first compute the dispersion covariance history of each satellite over 50 orbits.

For validation purposes, the covariances were also computed using the Monte Carlo method as described in Figure 2. Figure 4 shows a plot of the uncertainties from both covariances in the Radial-Intrack-Crosstrack frame, and Figure 5 shows the difference between them. In Figure 4, the blue curve is from a Monte Carlo simulation of 240 runs, and green is from LC. Note that because of the tight control, the covariances are maintained almost constant. Except for the noisiness of the Monte Carlo result, the two are almost indistinguishable. The Monte Carlo generated covariances should become smoother as more runs are used in the calculation. The standard deviation of the difference between the two sigmas is less than one meter.

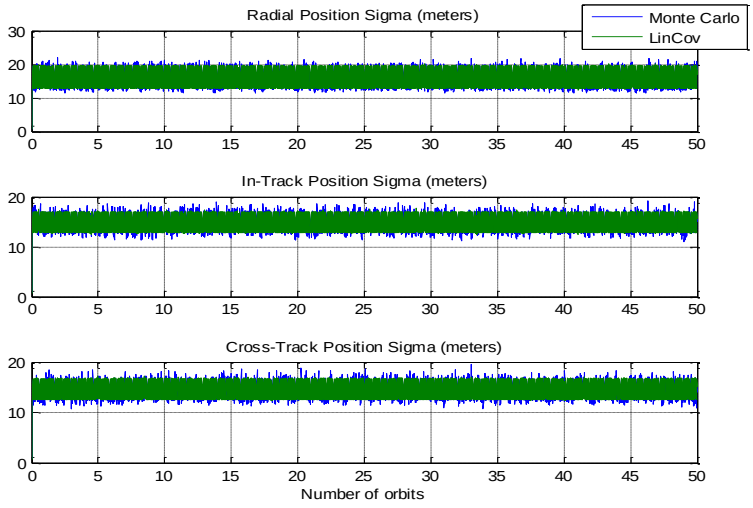

Figure 4. Comparison of covariances computed using LC and Monte Carlo.

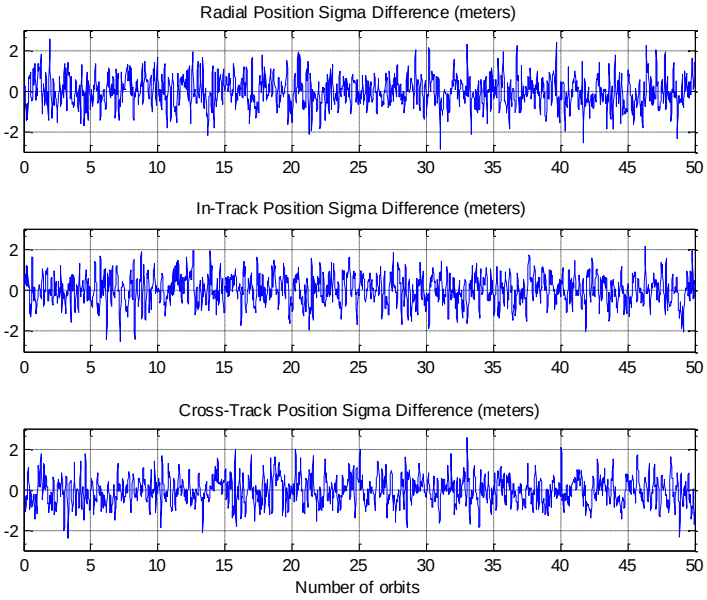

Figure 5: Difference between LC and Monte Carlo computed covariances.

IV. Probability of Collision Calculator

The fourth box in our simulation framework is the Pc calculator. A full discourse on the various methods of computing Pc is outside the scope of this paper. A separate paper which has been recently published discusses the various methods as well as a newly developed method that has been shown to be accurate for cluster station-keeping scenarios.[8] The new method is a hybrid of two metrics: Mahalanobis distance and the maximum instantaneous probability.[9]

The Mahalanobis distance is a simple metric to compute an upper bound to collision probability. Once both the state and state uncertainty have been propagated forward for the time period of interest, the computation of the Mahalanobis distance requires little additional computational effort. The Mahalanobis distance, d_M, and its associated upper bound to collision probability, p_M, are given by the following equations.

$$d_M = \left(1 - \frac{r_{HB}}{\|r\|}\right)\sqrt{\mathbf{r}^{\mathrm{T}} P_{rr}^{-1}\mathbf{r}} \tag{17}$$

$$p_M = \operatorname{erfc}\left(\frac{\min(d_M)}{\sqrt{2}}\right) \tag{18}$$

For many cases the probability of collision obtained using the Mahalanobis distance gives an upper bound that is far too conservative. For this reason other metrics are utilized to further reduce the incurred error in those cases.

The instantaneous probability of collision, p_I, is given by the following equation

$$p_I = \frac{\frac{4}{3}\pi r_{HB}^3}{\sqrt{(2\pi)^3 |P_{rr}|}} \exp\left(-\frac{1}{2}\mathbf{r}^{\mathrm{T}} P_{rr}^{-1}\mathbf{r}\right) \tag{19}$$

This metric is the probability that a collision will occur at an instant in time. Unlike the Mahalanobis distance, which is a conservative estimate, the maximum instantaneous probability of collision is a low estimate. This metric

generally agrees more closely with Monte Carlo than the Mahalanobis distance. Together with the Mahalanobis distance these two metrics provide a range in which the true probability of collision will fall.

A hybrid Pc method that combines the above two metrics is a weighted log average of the two metrics. A least squares fit was performed on a data set representing a large range of relative trajectories to find the weights that would best represent the probability of collision obtained from Monte Carlo. The result is

$$p_H = \exp\left(0.16\ln(p_M) + 0.8\ln(p_I)\right) \tag{20}$$

B. Computing Pc for Multiple Module Pairs

The probability of collision, p_i, is first computed for each possible module pair combination over a control cycle. Each of these probabilities must then be combined to obtain the probability, p_j, that at least one collision will occur within the cluster. Simply summing the probabilities for each of the possible combinations is inaccurate and can lead to probabilities greater than 1. The correct method is to calculate the probability that none of the modules will collide and subtract it from 1.

$$p_{\mathrm{j}} = 1 - \prod_{i=1}^{m}(1 - p_i) \tag{21}$$

where m is the number of module pairs.

C. Computing Pc Over Multiple Control Cycles

The probability of a collision, p_j, within the cluster is computed over each control cycle. Each of these probabilities must then be combined to obtain the probability, p_{total}, that at least one collision will occur during at least one of the control cycles. Simply summing the probabilities for each of the control cycles is inaccurate and can lead to probabilities greater than 1. The correct method is to calculate the probability that a collision will not occur during any of the control cycles and subtract the result from 1.

$$p_{\mathrm{total}} = 1 - \prod_{j=1}^{n}\left(1 - p_j\right) \tag{22}$$

where n is the number of control cycles.

This method can also be used when the probability of collision, p_1, is given for a specified length of time, T_1, and it is desirable to estimate the probability of collision, p_2, for a longer time period, T_2.

$$p_2 = 1 - (1 - p_1)^{\frac{T_2}{T_1}} \tag{23}$$

D. End-to-End Validation

To verify the accuracy of the Pc estimate, the previous example was modified with a larger hard body diameter of 20 meters instead of 4 meters. The LC analysis and the hybrid Pc method was compared with Monte Carlo simulations. Computing collision probability using Monte Carlo involves randomly perturbing a nominal initial state based upon the state uncertainty. The state is propagated forward in time, and the position of the spacecraft is checked at each instant in time to see if a collision occurred. This process is repeated, and the probability of collision is computed as follows

$$P_c = \frac{k}{N} \tag{24}$$

where N is the total number of runs and k is the number of runs resulting in a collision.

The LC analysis and the hybrid Pc method predicted a Pc of 39.5% for this modified example. The large predicted Pc means that we can run a reasonably small number of Monte Carlo runs to obtain a reasonable confidence interval. A total of 240 Monte Carlo simulation runs was performed. Out of these 240 runs, 79 collisions were detected corresponding to a 32.9% probability of collision with a 99% confidence interval of

$$Pc = 0.2530 < 0.3292 < 0.4122 \quad (25)$$

The predicted Pc from LC and the hybrid Pc method is within the 99% confidence interval, well within an order of magnitude of the true answer.

V. Conclusion

An alternative method to Monte Carlo for computing the probability of collision for a cluster of satellites was developed. Propagation of the position and velocity dispersion statistics through the use of linear covariance analysis based on numerical computation of the state transition matrix was shown. Extension of this method using unscented transform was also shown. The effects of navigation noise, closed-loop controller, and maneuver execution error were also included. A sample cluster scenario was evaluated using the covariance method with a hybrid method of computing Pc that combines the Mahalanobis distance metric and the maximum instantaneous probability. The results were shown to match the Monte Carlo results with a high confidence interval.

In the current implementation, the control of each spacecraft was assumed independent of the others. Future work will consider a multi-input and multi-output system blackbox for the generation of the covariance matrices where the system consists of multiple spacecraft in the cluster. Further enhancement will also include augmenting the system blackbox with the navigation filter to more accurately represent the coupling between the navigation states and the controller states.

Acknowledgments

The work presented in this paper was funded by the Defense Advanced Research Projects Agency in support of the System F6 Program.

The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[1]Alfano, S., "Addressing Nonlinear Relative Motion for Spacecraft Collision Probability," Proceedings of 2006 AAS/AIAA Astrodynamics Specialist Conference, August 2006.

[2]Carpenter, J. R., "Non-Parametric Collision Probability for Low-Velocity Encounters," 17th AAS/AIAA Space Flight Mechanics Meeting, No. AAS 07-201, Sedona, Arizona, January-February 2007.

[3]Chan, K. F., *Spacecraft Collision Probability*. Aerospace Press Series, 2350 E. El Segundo Boulevard, El Segundo, California 90245-4691: The Aerospace Corporation, 2008.

[4]Geller, D., "Linear Covariance Techniques for Orbital Rendezvous Analysis and Autonomous Onboard Mission Planning," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 6, Nov-Dec 2006, p. 1404-1414.

[5] Julier, S.J. and J.K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," Proc. IEEE, vol. 92, pp. 401 – 402, Mar. 2004.

[6] McKinley, D. P., "Development of a Nonlinear Probability of Collision Tool for the Earth Observing System," AIAA/AAS Astrodynamics Specialist Conference and Exhibit, No. AIAA-2006-6295, Lanham, MD, 20706, American Institute of Aeronautics and Astronautics, 2006.

[7] Patera , R. P., "Satellite Collision Probability for Nonlinear Relative Motion," Journal of Guidance, Control, and Dynamics, Vol. 26, September-October 2003, pp. 728--733. 0731-5090.

[8]Phillips, M. R. and S. Hur-Diaz, "On-Board Estimation of Collision Probability for Cluster Flight," AAS/AIAA Spaceflight Mechanics Winter Meeting, Kauai, HI, 10-14 Feb 2013.

[9]Phillips, M. R., Geller, D. K. and F. Chavez, "Procedure for Determining Spacecraft Collision Probability During Orbital Rendezvous and Proximity Operations," AAS/AIAA Spaceflight Mechanics Meeting, New Orleans, LA, 13-17 Feb 2011, AAS 11-144.

[10]Schiff, C., "Adapting Covariance Propagation to Account for the Presence of Modeled and Unmodeled Maneuvers," Paper AIAA-2006-6294, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 21-24 August 2006.