Near-Earth Semi-Analytical Uncertainty Propagation Toolkit for Conjunction Analysis

Yashica Khatri
Daniel J. Scheeres
Aerospace Engineering Department, University of Colorado Boulder

ABSTRACT

A semi-analytical uncertainty propagation conjunction analysis toolkit is presented for short-term and long-term conjunctions in the near-Earth space. The primary purpose of this user-accessible toolkit is to provide a fast and reliable method to predict collisions between space objects; and to present an alternative to the existing computationally expensive Monte Carlo methods. The toolkit uses Gaussian Mixture Models, State Transition Tensors, and a Simplified Dynamical System to precisely propagate state and uncertainty information to a conjunction time, and evaluate the probability of collision between two objects. The toolkit is made accessible through GitHub.

1. INTRODUCTION

Due to an increase in the number of resident space objects (RSOs) in the near-Earth space, sensors and observation resources are overloaded. This means that these RSOs are observed intermittently, leading to a need for propagation of information to future times. A major application of this is conjunction prediction. When a Gaussian uncertainty is propagated with highly nonlinear orbit dynamics, it loses its Gaussianity (Fig. 1). Currently prevalent Monte Carlo methods for uncertainty propagation are computationally expensive. To surpass this cost, semi-analytical methods have been developed that allow accurate and efficient propagation of uncertainty.

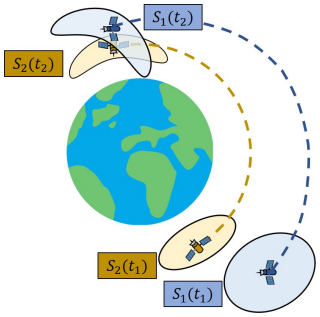


Fig. 1: Uncertainty loses its Gaussianity when propagated with nonlinear orbit dynamics. Even if their trajectories don't intersect, if object uncertainties overlap, there is a possibility of collision. [7]

Khatri and Scheeres developed a semi-analytical method of uncertainty propagation that uses a combination of Gaussian Mixture Models (GMMs) and State Transition Tensors (STTs) to propagate uncertainty accurately. The initial distribution is split into smaller distributions, which can maintain linearity over longer propagation times, when propagated individually. These GMM components are individually propagated using STTs, which are computed using the chosen system dynamics. In near-Earth solutions, the works by Khatri and Scheeres utilise a Simplified Dynamical System (SDS) that combines the speed of mean state propagation with the accuracy of adding in the short-period variations at the ends of the propagation to achieve the osculating object state. This method is implemented for computation of short-term [6, 7] and long-term [5] conjunctions.

This work presents a conjunction toolkit applicable to short-term and long-term conjunctions that takes inputs such as orbital parameters and generates a probability of collision using a semi-analytical uncertainty propagation conjunction

analysis (SAUPCA) toolkit. The MATLAB scripts for this toolkit have been uploaded to GitHub and can be accessed using short-term [4] and long-term [3] repositories. The code implementation and initialization is defined in this paper.

Section 2 defines this method of semi-analytical uncertainty propagation and probability of collision calculations. This is followed by a detailed implementation of the Monte Carlo probability of collision calculations. After the method descriptions, the code implementation parameters are defined and a test case is presented. This is followed by the conclusions and the Appendix.

2. SEMI-ANALYTICAL UNCERTAINTY PROPAGATION AND PROBABILITY OF COLLISION CALCULATION

Fig. 2 summarises the semi-analytical method of uncertainty propagation used for probability of collision calculations in the script 'ComputeGMMSTTMethodPc.m'. The initial distribution is transformed to an Equinoctial element set using a Jacobian matrix ('JacobianCalc.m') and a sub-optimal algorithm from Horwood et al. [2] is used to split this uncertainty into a chosen number of GMM components. These smaller distributions are converted to Delaunay frame for propagation using the STTs. Hamiltonian averaging is used to compute mean dynamics and the analytical time-dependent short-period variation equations. The mean dynamics are used to compute the STTs for the GMM component propagation.

The propagated GMM components are then compared in an all-on-all analysis to calculate the probability of collision. These probabilities are combined using a double weighted sum to achieve a cumulative probability of collision of the system. A benefit of this is that the individual GMM components maintain Gaussianity and thus, can be combined with the Foster method of probability of collision calculation [1].

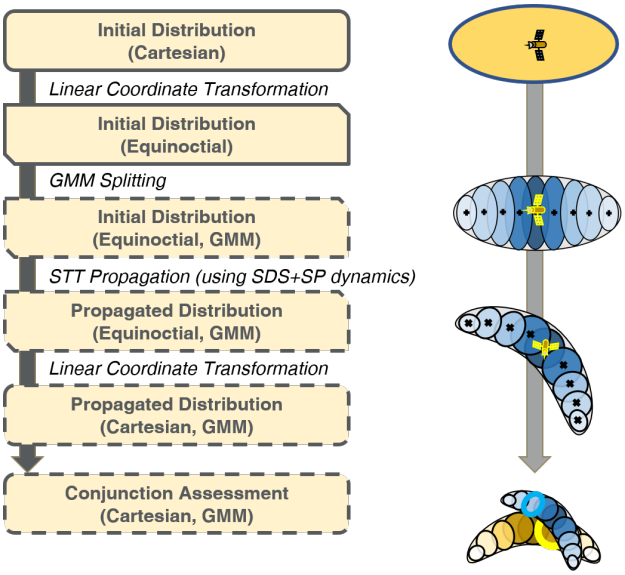


Fig. 2: A summary of the semi-analytical uncertainty propagation and conjunction analysis method. [5]

2.1 Gaussian Mixture Model Splitting

The GMM is split using the Horwood et al. [2] sub-optimal algorithm. This section is set up using the optimization toolbox in MATLAB. A parameter $m = 6$ is defined to initiate the splitting process. Based on this, the following algorithm is set up:

1. $\sigma = \frac{2m}{N-1}$, where N is the number of GMM components to split into.
2. For $\alpha = 1,...,N$, $\mu_\alpha = -m + \sigma(\alpha - 1)$
3. Compute matrix $(M)_{\alpha\beta} = \mathcal{N}(\nu_\alpha - \nu_\beta; 0, 2\sigma^2)$ and vector $(n)_\alpha = \mathcal{N}(\nu_\alpha; 0, \sigma^2 + 1)$ using function normpdf.

4. Define an optimization using quadprog, with 'ConstraintTolerance' of 10^{-25} and 'OptimalityTolerance' of 10^{-25} to minimize: $\frac{1}{2}w^T Mw - w^T n$ over w subject to constraints: $\sum_{\alpha=1}^{N} w_\alpha = 1$ and $w_\alpha \geq 0$, where $\alpha = 1, ..., N$
5. For same α, calculate $\tilde{w}_\alpha = \frac{\sqrt{2\pi}}{\sqrt{1-\sigma^2}} w_\alpha \exp\left[\frac{\mu_\alpha^2}{2(1-\sigma^2)}\right]$, $\tilde{\mu}_\alpha = \frac{\mu_\alpha}{1-\sigma^2}$, and $\tilde{\sigma}^2 = \frac{\sigma^2}{1-\sigma^2}$
6. Compute $\hat{\mu}_\alpha = \nu^1 + \sqrt{Q^1}\tilde{\mu}_\alpha$, $\hat{w}_\alpha = \tilde{w}_\alpha$, and $\hat{\sigma}^2 = \tilde{\sigma}^2 Q^{11}$
7. Based on this, $\bar{Q} = (\hat{\sigma}^{-2} e_1 e_1^T + Q^{-1})^{-1}$, $\bar{\nu}_\alpha = \bar{Q}(\hat{\sigma}^{-2}\hat{\mu}_\alpha e_1 + Q^{-1}\nu)$, and $\bar{w}_\alpha = \hat{w}_\alpha \mathcal{N}(\hat{\mu}_\alpha - e_1^T \nu; 0, \hat{\sigma}^2 + e_1^T Q e_1)$, where $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}$
8. For $\alpha = 1, ..., N$, compute re-normalized weights $\bar{w}_\alpha = \bar{w}_\alpha / \sum_{\beta=1}^{N} \bar{w}_\beta$

This splitting function 'GMM_Component_Calcs.m' uses the number of components, nominal mean, nominal covariance, and a test flag as inputs. It outputs the GMM component weights ($\bar{w}$), means ($\bar{\mu}$), and covariance ($\bar{Q}$).

2.2 Simplified Dynamical System

The MATLAB symbolic computer is used to develop the Hamiltonian averaging equations to compute the mean dynamics equations and the time-based short-period variation equations in the script 'SDSGenerator.m'. As described in previous works [7, 5], the toolkit uses non-Keplerian perturbations from Solar Radiation Pressure (SRP) and J2 gravitational harmonics. The state and state Equations of Motion (EOMs) are represented as follows:

$$\begin{aligned} x_{\text{del}} &= [l,\ g,\ h,\ k_\odot, L,\ G,\ H,\ K_\odot] \\ f_{\text{del}} &= [\dot{l},\ \dot{g},\ \dot{h},\ \dot{k}_\odot,\ \dot{L},\ \dot{G},\ \dot{H},\ \dot{K}_\odot] \end{aligned} \tag{1}$$

The mean Hamiltonian and generating function equations are shown in Appendix A. The mean dynamics are stored in the function: 'MeanDynamicsFunction.m' and the short-period offsets from mean to osculating and vice-versa are stored in 'getInitialDelOffset.m' and 'getFinalDelOffset.m', respectively. These offset functions can be accessed using the script 'getOffset.m' that uses state, time, constants, and a direction flag as inputs. It outputs the offset in the chosen direction.

2.3 State Transition Tensors

MATLAB symbolic computer is used to compute the STT EOMs. These EOMs are then used to propagate the STTs to the final time. The propagated STTs are used to map the mean and covariance of the GMM components to the final time using the following set of equations:

$$\dot{\Phi}_{i,ab} = \sum_{\alpha=1}^{N} A_{i,\alpha} \Phi_{\alpha,ab} + \sum_{\alpha=1}^{N} \sum_{\beta=1}^{N} A_{i,\alpha\beta} \Phi_{\alpha a} \Phi_{\beta b} \tag{2}$$

Here, $\Phi_{\alpha,ab}$ is a second order STT, $\Phi_{\alpha a}$ and $\Phi_{\alpha b}$ are first order STTs, $A_{i,\alpha}$ is the first order Linear Dynamics Tensor (LDT), and $A_{i,\alpha\beta}$ is the second order LDT. N denotes the state dimension. i, a, and b are the three dimensions of the STT EOMs. The LDTs are computed using partials of the system dynamics with respect to the state about the nominal trajectory (denoted by *):

$$A_{i,k_1 \ldots k_p} = \left. \frac{\partial^p f_{\text{del}i}}{\partial x_{\text{del}k_1} \ldots \partial x_{\text{del}k_p}} \right|_* \tag{3}$$

where, i and k_j are the dimensions of the LDT.

The LDT calculations are performed using the MATLAB symbolic computer by taking symbolic partial derivatives of the Delaunay mean dynamics from the SDS with respect to the state in function: 'SymbolicSDSJ2Computer.m'. 'propagateWithDynamics.m' script is used to propagated the STTs alongside the state using ode113, with 10^{-13} as the relative and absolute tolerances.

2.4 Time of Closest Approach Finder

The function 'FindTCAGMMSTTMethod.m' is used to find the time of closest approach (TCA) between two chosen GMM components. Fig. 3 shows the propagation process for the GMM component means. The means of the GMM components are mapped to the final propagation time using the STTs computed with the mean dynamics using the function 'STTcalcs2BP.m':

$$\delta m_i(t) = \sum_{p=1}^{m} \frac{1}{p!} \Phi_{i,k_1 \ldots k_p} E[\delta x_{k_1}^0 \ldots \delta x_{k_p}^0] \tag{4}$$

where, the moments, E, are defined in functions 'E1calc.m' and 'E2calc.m' using:

$$\begin{aligned} E[\delta x_i] &= \delta m_i \\ E[\delta x_i \delta x_j] &= \delta m_i \delta m_j + P_{ij} \\ E[\delta x_i \delta x_j \delta x_k] &= \delta m_i \delta m_j \delta m_k + (\delta m_i P_{jk} + \delta m_j P_{ik} + \delta m_k P_{ij}) \\ &\ldots \end{aligned} \tag{5}$$

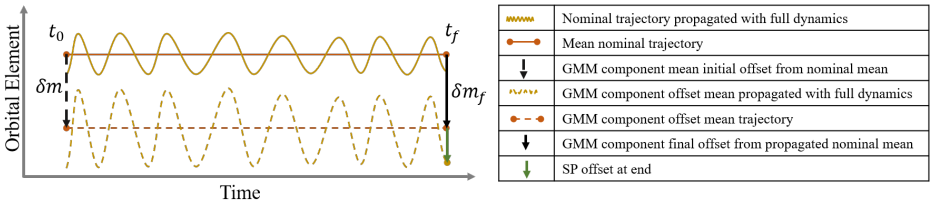


Fig. 3: The initial GMM component mean is a deviation from the nominal mean state. This GMM mean is mapped to the final time using STTs and it is converted back to the osculating state. [5]

The TCA approach between two GMM components is calculated using a stepping method that uses the time update:

$$t = \frac{\Delta r . \Delta v}{|\Delta v|^2} \tag{6}$$

where Δr and Δv are the relative position and velocity vectors respectively. Each time update ensures a smaller difference in the relative distance between the two objects at the next time. The final converged time is the actual TCA. The covariance is mapped ('Covariance_Propagation.m') to this actual TCA using the second order STTs, as described in the equation below:

$$P_{ij}(t) = \left(\sum_{p=1}^{m} \sum_{q=1}^{m} \frac{1}{p!q!} \Phi_{i,k_1 \ldots k_p} \Phi_{j,l_1 \ldots l_q} E[\delta x_{k_1}^0 \ldots \delta x_{k_p}^0 \delta x_{l_1}^0 \ldots \delta x_{l_q}^0] \right) - \delta m_i(t) \delta m_j(t) \tag{7}$$

The means and moments are the same as those described in Eq. 4 and Eq. 5, respectively.

2.5 Probability of Collision Calculations

Each propagated GMM components associated with the objects in conjunction are compared in an all-on-all analysis at their respective actual TCAs to achieve the cumulative probability of collision, as demonstrated in Fig. 4. This is done using the function script: 'GMM_Pc_Calcs_Fun.m'. The individual probability of collision between the i^{th} and j^{th} components of objects 1 and 2 respectively, can be computed using:

$$P_{c,ij} = \frac{1}{2\pi\sqrt{|P^*|}} \int_{-R}^{R} \int_{-\sqrt{R^2-x^2}}^{\sqrt{R^2-x^2}} \exp(-A^*)\, dz\, dx \tag{8}$$

P^* is the combined covariance of the two objects and A^* is defined as follows:

$$s = xi + zk,\ s_0 = x_0 i + z_0 k \tag{9a}$$

$$A^* = (s - s_0)^T P^{*-1} (s - s_0)/2 \tag{9b}$$

where, s and s_0 are the relative positions in the encounter plane at the actual and nominal TCA, respectively.

The individual probabilities are combined using a double weighted sum:

$$P_c = \sum_{i=1}^{L_a} \sum_{j=1}^{L_b} w_a^{(i)} w_b^{(j)} P_{c,ij} \tag{10}$$

where, $w_a^{(i)}$ and $w_b^{(j)}$ are the weights associated with the i^{th} and j^{th} GMM components of objects 1 and 2, respectively. The double integral to compute the total probability of collision is implemented using 'integral2' in MATLAB.

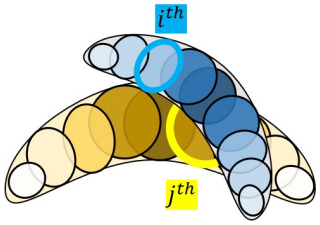


Fig. 4: Each GMM component from object 1 is compared to that from object 2 to compute the cumulative probability of collision. [6]

3. MONTE CARLO METHOD

In the Monte Carlo analysis, a cloud of points is propagated numerically to evaluate the probability of collision in the function script: 'Compute_MC_Pc_Cart_1on1.m'.

A random point is generated for each object in conjunction from the nominal states and epoch covariances using 'mvnrnd'. These points are propagated using 'propagateWithDynamics.m' to find the actual TCA, as shown in Fig. 5. The actual TCA is found using 'FindTCAMCMethod.m' with 'while' loops, using the nominal TCA as an initial guess. Same as before, the time update is computed as described in Eq. 6 using the relative position and velocity of the two objects.

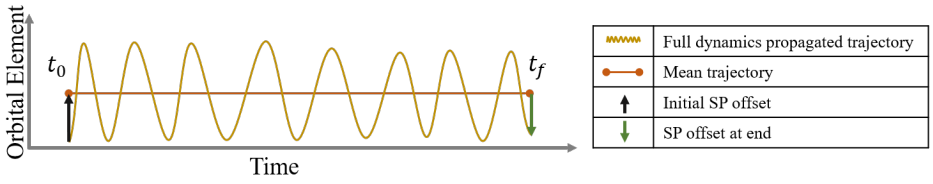


Fig. 5: The initial state is converted to a mean state and is mapped to the final time using STTs, where it is converted back to the osculating state. [5]

The probability of collision in the Monte Carlo analysis is calculated by comparing the number of objects closer than the combined hard body radii of the two objects, to the total number of points in the analysis.:

$$\text{MC } P_c = \frac{\text{Conjunctions where DCA} < R}{\text{Total number of conjunctions tested}} \tag{11}$$

The binomial confidence bounds are found using the normal approximation [8]:

$$\mathrm{CI} = \mathrm{MC}\ P_c \pm z\sqrt{\frac{\mathrm{MC}\ P_c(1-\mathrm{MC}\ P_c)}{n}} \tag{12}$$

$z = 1.96$ for a 95% confidence interval and n denotes the number of points in the Monte Carlo analysis.

4. CODE INITIALIZATION

To initiate the code, the only scripts that need to be edited are: 'Main_Code.m' and 'constantsAndInitialState.m'. The states and uncertainties of the two objects in conjunction and some constant parameters need to be defined. The input parameters that are required to run the code and that can be changed without needing code updates include:

Main_Code.m	———————————
runMonteCarlo:	When toggled to 1, this flag allows the Monte Carlo probability of collision calculation process to run, using a chosen number of points.
runGMMSTTMethod:	When toggled to 1, this flag allows the semi-analytical GMM-STT probability of collision calculation process to run, using a chosen number of points.
plotThings:	Flag to plot probability of collision results.
saveResultsToText:	Save results to an output file.
constantsAndInitialState.m	———————————
[a e i O w M]:	[km and rad] Classical orbital elements (COEs) defining the epoch states of the two objects in conjunction.
P and P2:	[km and km/s] Cartesian covariance of both objects in conjunction.
constants.case_flag:	Pre-coded sample test cases can be accessed using this variables.
constants.points:	Number of Monte Carlo points currently being evaluated for each object.
constants.JMAX:	List of number of GMM components to split each object into. The only limitation is that this script cannot split the distribution into less than 15 components.
constants.testFig2:	When toggled to 1, this test flag plots the GMM mean spread for object 1 at the nominal TCA using 15 components. 'runMonteCarlo' and 'runGMMSTTMethod' must be toggled to 1 to run this.
constants.plot_GMM_ell:	When toggled to 1, this test flag plots the GMM covariances, in addition to their means. 'runMonteCarlo' and 'runGMMSTTMethod' must be toggled to 1 to run this.
constants.testFig4:	When toggled to 1, this test flag plots the weights against the means of the GMM distribution. 'runGMMSTTMethod' must be toggled to 1 to run this.
constants.par:	Allow parallel runs in MATLAB.
constants.nodes:	Number of nodes allowed for the parallel MATLAB runs.
usePredefinedCases:	When toggled to 1, this flag allows the predefined test cases to be accessed, using constants.case_flag.
constants.P_prop:	Define Nominal TCA.
ICState.obj2.COE:	Define the COE for object 2.
r1:	Hard body radius of object 1.
r2:	Hard body radius of object 2.
constants.rho:	Reflectivity of the objects.
constants.Aom:	Area over mass ratio of objects.

5. TEST CASE

A case for the short-term conjunction case is setup below and the code is initiated for a test run.

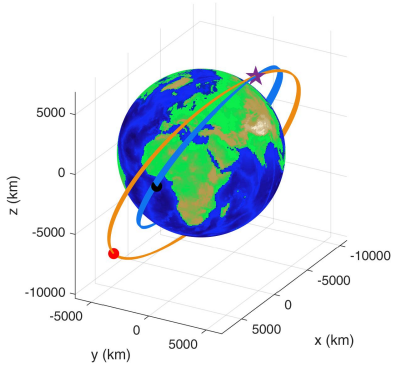
5.1 Example Setup

The inputs and constants to run this example test case are listed in the initialization table below. The initial states and epochs are introduced. The flags: 'runMonteCarlo' and 'runGMMSTTMethods' are toggled to 1 to run the probability

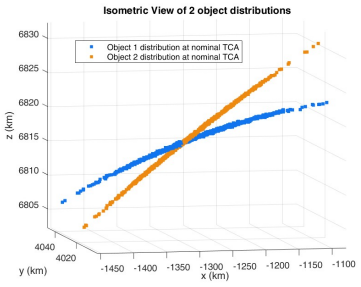
of collision calculations using both methods. 'plotThings' is 1 to allow plotting of the results. 10^7 points are generated associated with object and tested for probability of collision calculations. No test figures are plotted and parallelization is turned off. Nominal TCA is set to 1.5 days. The hard body radii of the two objects and other parameters are also defined in the table.

Main_Code.m	———————————————
runMonteCarlo:	1
runGMMSTTMethod:	1
plotThings:	1
saveResultsToText:	0
constantsAndInitialState.m	———————————————
[a e i O w M]:	[km and deg] [8000; 0.15; 60; 0; 0; 0]
P and P2:	[km and km/s] [0.0067664 -0.0029183 0.0027112 -9.9816E-7 -1.7636E-7 2.1797E-6;... -0.0029183 0.005348 -0.0011671 -1.5861E-6 -3.5203E-7 3.3414E-6;... 0.0027112 -0.0011671 0.001087 -3.9883E-7 -7.5945E-8 8.6148E-7;... -9.9816E-7 -1.5861E-6 -3.9883E-7 9.4587E-9 -1.1375E-10 1.1511E-9;... -1.7636E-7 -3.5203E-7 -7.5945E-8 -1.1375E-10 9.8844E-9 8.5671E-11;... 2.1797E-6 3.3414E-6 8.6148E-7 1.1511E-9 8.5671E-11 7.2859E-9]
constants.case_flag:	1
constants.points:	1E7
constants.JMAX:	[37;101;301;501]
constants.testFig2:	0
constants.plot_GMM_ell:	0
constants.testFig4:	0
constants.par:	0
constants.nodes:	4
usePredefinedCases:	0
constants.P_prop:	1.5 days
ICState.obj2.COE:	[km and deg] [9843.3; 0.241038; 61.8685; -10.9875; 55.5795; -87.5619]
r1:	20 m
r2:	20 m
constants.rho:	0.2
constants.Aom:	2E-6

Fig 6a shows the nominal trajectories of the two objects and their conjunction point. Fig. 6b shows the uncertainty distribution of the two objects at the nominal TCA.



(a) Nominal trajectories. [6]



(b) Uncertainty distribution at the nominal TCA.

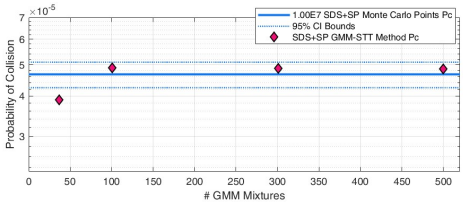
Fig. 6: (a) Blue and yellow trajectories show the nominal trajectories of objects 1 and 2, respectively. The black dot and red dot denote the initial locations of objects 1 and 2, respectively. Purple star denotes the conjunction point. (b) The blue points and yellow points show the uncertainty distributions at the nominal TCA of objects 1 and 2, respectively.

The probability of collision for this test case is first computed with the Monte Carlo analysis, followed by the semi-analytical approach. This allows a comparison between the two results.

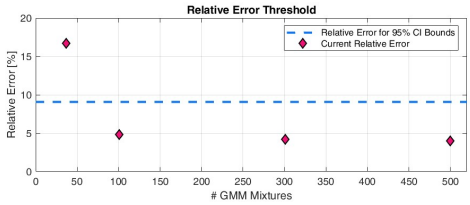
5.2 Results

The Monte Carlo scripts outputs a probability of collision of 4.67×10^{-5} in 5.3 hours [6] (when run with 24 parallel nodes on the CU Boulder supercomputer), when using the SDS dynamics. The semi-analytical GMM-STT script outputs a probability of collision of: 3.8894×10^{-5}, 4.8952×10^{-5}, 4.8659×10^{-5}, 4.8560×10^{-5} with 37, 101, 301, and 501 components, respectively. The semi-analytical computation with 101 components takes 4.43 minutes [6] (when run with 24 parallel nodes on the CU Boulder supercomputer).

Fig 7a shows the comparison between the Monte Carlo results and those from the semi-analytical method. Fig. 7b shows the relative error between these two results. It is clear that above 101 components, the results have converged with the Monte Carlo probability of collision. This confirms the validity of the method and provides a demonstration of the toolkit.



(a) Probability of collision results. [6]



(b) Relative error in results.

Fig. 7: (a) The solid blue line refers to the Monte Carlo probability of collision and the dashed blue lines are the confidence intervals around this result. The pink diamonds show the semi-analytical approach probability of collision at different number of GMM components. (b) The dashed line shows the binomial normal approximation 95% confidence interval line. The pink diamonds show the semi-analytical approach results for the probability of collision.

6. CONCLUSION

This work provides a semi-analytical uncertainty propagation conjunction analysis (SAUPCA) toolkit that can propagate the uncertainty and evaluate the probability of collision results in an accurate and efficient manner. The paper describes the different components of the coded toolkit and the parameters that need to be defined to initiate the code. A test case is presented that is set up using defined constants and initial states and covariances of the two objects in conjunction. The probability of collision is calculated using the Monte Carlo analysis and the SAUPCA toolkit to test the output and compare the final results. The plotted results show a convergence of the SAUPCA probability of collision with those from the Monte Carlo analysis, hence demonstrating a successful run. The toolkit can be accessed through GitHub repositories [4, 3].

REFERENCES

[1] Salvatore Alfano. A parametric analysis of orbital debris collision probability and maneuver rate for space vehicles. *NASA JSC-25898*, 1992.

[2] Joshua T. Horwood, Nathan D. Aragon, and Aubrey B. Poore. Gaussian Sum Filters for Space Surveillance: Theory and Simulations. *Journal of Guidance, Control, and Dynamics*, 34(6):1839–1851, 2011.

[3] Yashica Khatri. Long-term-saupca-toolkit. Available at https://github.com/yashicakhatri2/Long-Term-SAUPCA-Toolkit.git.

[4] Yashica Khatri. Short-term-saupca-toolkit. Available at https://github.com/yashicakhatri2/Short-Term-SAUPCA-Toolkit.git.

[5] Yashica Khatri and Daniel J Scheeres. HYBRID METHOD OF UNCERTAINTY PROPAGATION FOR LONG-TERM CONJUNCTION ANALYSIS. In *33rd AAS/AIAA Space Flight Mechanics Meeting, Austin, Texas*. AAS 23-376.

[6] Yashica Khatri and Daniel J Scheeres. Hybrid nonlinear semi-analytical uncertainty propagation for conjunction analysis. In *Proceedings of 73rd International Astronautical Congress 2022 IAC-22,C1,3,1,x68776*, Paris, France, 2022.
[7] Yashica Khatri and Daniel J Scheeres. Nonlinear Semi-Analytical Uncertainty Propagation for Conjunction Analysis. *Acta Astronautica*, 203:568–576, 2023.
[8] P.S. Laplace. Théorie analytique des probabilités, 1812.

APPENDIX A

The mean equations from the SDS are shown below [6]:

$$\widehat{\mathscr{K}} = \mathscr{K}_0 + \varepsilon\mathscr{K}_1 + \frac{\varepsilon^2}{2!}\mathscr{K}_2 + \frac{\varepsilon^3}{3!}\mathscr{K}_3 + \frac{\varepsilon^4}{4!}\mathscr{K}_4 + \frac{\varepsilon^5}{5!}\mathscr{K}_5 + \frac{\varepsilon^6}{6!}\mathscr{K}_6 \tag{13}$$

Each computed component of the mean Hamiltonian is as follows [6]:

$$\mathscr{K}_0 = -\frac{\mu}{2a} \tag{14a}$$
$$\mathscr{K}_2 = \frac{2!}{\varepsilon^2}(\nu K_\odot) \tag{14b}$$
$$\mathscr{K}_3 = \frac{3\alpha^2 J_2 \mu (3s^2-2)}{2a^3\eta^3\varepsilon^3} \tag{14c}$$
$$\mathscr{K}_5 = -\frac{180a\beta e T_c}{\varepsilon^5} \tag{14d}$$
$$\begin{aligned}\mathscr{K}_6 = -\frac{45\alpha^4 J_2^2 \mu}{8a^5\eta^9\varepsilon^6}\big[&6e^2\eta^2 s^2(15s^2-14)\cos(2g) + e^2\{-(3\eta^2(5s^4+8s^2-8)+4(2-3s^2)^2)\} \\ &+4\{3\eta^3(2-3s^2)^2 + \eta^2(21s^4-42s^2+20) + (2-3s^2)^2\}\big]\end{aligned} \tag{14e}$$

where,

$$\begin{aligned}T_c = \frac{1}{4}\big[&(c+1)(c_\odot+1)\cos(g+h-h_\odot-k_\odot) + (c-1)(c_\odot-1)\cos(g-h+h_\odot-k_\odot) \\ &-(c+1)(c_\odot-1)\cos(g+h-h_\odot+k_\odot) - (c-1)(c_\odot+1)\cos(g-h+h_\odot+k_\odot) + 4ss_\odot\sin(g)\sin(k_\odot)\big]\end{aligned} \tag{15a}$$
$$\begin{aligned}T_s = \frac{1}{4}\big[&(-c-1)(c_\odot+1)\sin(g+h-h_\odot-k_\odot) - (c-1)(c_\odot-1)\sin(g-h+h_\odot-k_\odot) \\ &+(c+1)(c_\odot-1)\sin(g+h-h_\odot+k_\odot) + (c-1)(c_\odot+1)\sin(g-h+h_\odot+k_\odot) + 4ss_\odot\cos(g)\sin(k_\odot)\big]\end{aligned} \tag{15b}$$

The mean Hamiltonian can be used to compute the mean dynamics [6]:

$$\begin{aligned}\dot{l} &= \frac{\partial\widehat{\mathscr{K}}}{\partial L}, \quad \dot{g} = \frac{\partial\widehat{\mathscr{K}}}{\partial G}, \quad \dot{h} = \frac{\partial\widehat{\mathscr{K}}}{\partial H}, \quad \dot{k}_\odot = \frac{\partial\widehat{\mathscr{K}}}{\partial K_\odot} \\ \dot{L} &= -\frac{\partial\widehat{\mathscr{K}}}{\partial l}, \quad \dot{G} = -\frac{\partial\widehat{\mathscr{K}}}{\partial g}, \quad \dot{H} = -\frac{\partial\widehat{\mathscr{K}}}{\partial h}, \quad \dot{K}_\odot = -\frac{\partial\widehat{\mathscr{K}}}{\partial k_\odot}\end{aligned} \tag{16}$$

Once the mean Hamiltonian is known, the Lie operator is used to back-solve for the generating functions ($\mathscr{W}_n$) [6].

$$\mathscr{L}_0(\mathscr{W}_n) = \widetilde{\mathscr{H}_{0,n}} - \mathscr{K}_n \tag{17}$$

The generating functions corresponding to the system Hamiltonian are given in Eq. 18 [6]:

$$\mathscr{W}_3 = \frac{3!}{\varepsilon^3}\frac{L^3\alpha^2 J_2}{4\mu a^3\eta^3}\Big[-3s^2\left(\frac{1}{2}\sin 2(f+g)+\frac{1}{2}e\sin(f+2g)+\frac{1}{6}e\sin(3f+2g)\right)+(3s^2-2)(f+e\sin f-l)\Big] \tag{18a}$$

$$\mathscr{W}_5 = \frac{30a\beta}{n\varepsilon^5}\left(4e^2T_c\sin E+e(\eta T_s\cos 2E-T_c(6E+\sin 2E-6l))-4\eta T_s\cos E+4T_c\sin E\right) \tag{18b}$$

$$\begin{aligned}\mathscr{W}_6 = \frac{45\alpha^4 J_2^2 n}{8a^2e\eta^9\varepsilon^6}\Big[&-6e^3\eta^2s^2(15s^2-14)(f-l)\cos 2g+8(2-3s^2)^2(e^2+\eta^3-1)\sin f\\ &+e\{2(2-3s^2)^2(e^2+\eta^3-1)\sin 2f+(f-l)(e^2(3\eta^2(5s^4+8s^2-8)+4(2-3s^2)^2)\\ &-4(\eta^2(21s^4-42s^2+20)+(2-3s^2)^2))\}\Big]\end{aligned} \tag{18c}$$

These generating functions can be used to compute the mean to osculating, and reverse, offset equations that are further described in previous works by Khatri and Scheeres [6].

APPENDIX B: LIST OF FUNCTIONS

The following functions and files are included in the **short-term SAUPCA toolkit**:

Covariance_Propagation.m

Initial_Ecalc.m

SDSDynamicsStateOnly.m

getFinalDelOffset.m

propagateWithDynamics.m

E1calc.m

JacobianCalc.m

SDSGenerator.m

getInitialDelOffset.m

saveResultsToTextFun.m

COE_to_Cartesian.m

E2calc.m

MAtoEccTA.m

STTcalcs2BP.m

getJOrder1.m

savedResults.m

COE_to_Delaunay.m

Equinoctial_to_Cartesian.m

Main_Code.m

SymbolicSDSJ2Computer.m

getJOrder2.m

substituteFunctions.m

COE_to_Equinoctial.m

FindTCAGMMSTTMethod.m

Main_Plotting.m

computeEqDelPartials.m

getOffset.m

substitutePartials.m

Compute2ndOrderSTTdot.m

FindTCAMCMethod.m

MeanDynamicsFunction.m

constantsAndInitialState.m

getSun.m

ComputeGMMSTTMethodPc.m

GMM_Component_Calcs.m

PQW_to_IJK_transform.m

dSFull2.m

normalize.m

Compute_MC_Pc_Cart_1on1.m

GMM_Pc_Cacls_Fun.m

SDSDynamics.m

define_cases.m

phi_calc.m

testResults_Case1_Using_SDS+SP.txt

The following functions and files are included in the **long-term SAUPCA toolkit**: Covariance_Propagation.m

JacobianCalc.m

SDSDynamicsForLongTCA.m

dSFull2.m

normalize.m

E1calc.m

MAtoEccTA.m

SDSDynamicsForPcCalcs.m

define_cases.m

phi_calc.m

COE_to_Cartesian.m

E2calc.m

Main_Code.asv

SDSDynamicsStateOnly.m

getFinalDelOffset.m

propagateWithDynamics.m

COE_to_Delaunay.m

Equinoctial_to_Cartesian.m

Main_Code.m

SDSDynamicsStateOnlyForLongTCA.m

getICForObj2SDS.m

propagateWithDynamicsForLongTCA.m

COE_to_Equinoctial.m

FindTCAGMMSTTMethod.m

Main_Plotting.m

SDSGenerator.m

getInitialDelOffset.m

propagateWithDynamicsForPcCalcs.m

CatchTCA.m

FindTCAMCMethod.m

MeanDynamicsFunction.m

SDSGeneratorTestDynamicsStateOnly.m

getJOrder1.m

saveResultsToTextFun.m

CatchTCAGMMSTT.m

GMM_Component_Calcs.m

PQW_to_IJK_transform.m

STTcalcs2BP.m

getJOrder2.m

savedResults.m

Compute2ndOrderSTTdot.m

GMM_Pc_Cacls_Fun.m

Partials_PQW_COE.m

SymbolicSDSJ2Computer.m

getMinDistEvent.m

substituteFunctions.m

ComputeGMMSTTMethodPc.m

Initial_Ecalc.m

Plot_things.m

computeEqDelPartials.m

getOffset.m

substitutePartials.m

Compute_MC_Pc_Cart_1on1.m

J2AccelerationCalculator.m

SDSDynamics.m

constantsAndInitialState.m

getSun.m