Post-Maneuver Collision Probability Estimation

Using Sparse Polynomial Chaos Expansions

Brandon A. Jones¹, Nathan Parrish², Alireza Doostan³

University of Colorado, Boulder, CO, 80309, USA

This paper describes the use of polynomial chaos expansions to approximate the probability of a collision between two satellites after at least one performs a translation maneuver. Polynomial chaos provides a computationally efficient means to generate an approximate solution to a stochastic differential equation without introducing any assumptions on the a posteriori distribution. The stochastic solution then allows for orbit state uncertainty propagation. For the maneuvering spacecraft in the presented scenarios, the polynomial chaos expansion is sparse, allowing for the use of compressive sampling methods to improve solution tractability. This paper first demonstrates the use of these techniques for possible intra-formation collisions for the Magnetospheric Multi-Scale mission. The techniques are then applied to a potential collision with debris in low Earth orbit. Results demonstrate that these polynomial chaos-based methods provide a Monte Carlo-like estimate of the collision probability, including adjustments for a spacecraft shape model, with only minutes of computation cost required for scenarios with a probability of collision as low as $10^{-6}$. A graphics processing unit implementation of the polynomial chaos expansion analysis further reduces the computation time for the scenarios presented.

¹ Assistant Research Professor, Colorado Center for Astrodynamics Research, UCB 431, AIAA Senior Member.
² Graduate Research Assistant, Colorado Center for Astrodynamics Research, UCB 431, AIAA Member.
³ Assistant Professor, Department of Aerospace Engineering Sciences, UCB 429.
Nomenclature

\[ A = \text{stochastic ordinary differential operator} \]
\[ c_\alpha = \text{vector of polynomial chaos expansion coefficients} \]
\[ C = \text{matrix comprised of expansion coefficients} \]
\[ C_R = \text{coefficient of reflectivity} \]
\[ d = \text{number of random inputs} \]
\[ H = \text{measurement matrix} \]
\[ J = \text{cost function} \]
\[ i, j, k = \text{indices and counter variables} \]
\[ L = \text{Cholesky decomposition of } \Sigma \]
\[ L_{XY,*}, L_{Z,*} = \text{collision volume boundaries} \]
\[ M_{PC} = \text{number of training samples} \]
\[ M_{MC} = \text{number of comparison samples} \]
\[ m = \text{minimum satellite separation (km)} \]
\[ N_0^d = \text{set of multi-indices of dimension } d \]
\[ n = \text{number of elements in } X \]
\[ \mathcal{N}(\bar{u}, \sigma^2) = \text{Gaussian distribution with mean } \bar{u} \text{ and variance } \sigma^2 \]
\[ P = \text{cardinality of } \Lambda_{p,d} \]
\[ P_c = \text{instantaneous probability of collision} \]
\[ P_s' = \text{complement of survival probability} \]
\[ p = \text{total degree of the basis functions} \]
\[ q = \text{quaternion} \]
\[ R = \text{spherical keep-out radius (km)} \]
\[ R, h = \text{cylinder radius and height (km)} \]
\[ r = \text{tumbling debris keep out radius} \]
\[ r = \text{position vector (km)} \]
\[ s = \text{satellite separation distance (km)} \]
\[ \hat{T}, \hat{W} = \text{unit vectors defined by a given } \Delta \text{v} \]
\[ t = \text{time (s)} \]
\( \bar{u} \) = mean value of variable \( u \)
\( \hat{u} \) = estimated value of variable \( u \)
\( \mathcal{U}(a, b) \) = Uniform distribution in range \([a, b]\)
\( v \) = velocity vector (km/s)
\( X, Y, Z \) = Cartesian coordinates (km)
\( \hat{X}, \hat{Y} \) = conjunction plane basis vectors
\( X \) = vector of Cartesian coordinates
\( Y \) = matrix of propagated Cartesian coordinates
\( y \) = single column of \( Y \)
\( \alpha \) = multi-index in \( \mathbb{N}_0^d \)
\( \Gamma^d \) = \( d \)-dimensional hypercube - the image of random variables \( \xi \)
\( \gamma \) = penalty function scale factor
\( \delta \) = residual of estimation fit
\( \varepsilon \) = truncation error
\( \theta, \phi \) = angles used in maneuver error model (deg)
\( \kappa \) = penalty function
\( \Lambda_{p,d} \) = subset of multi-indices specifying tensor product of maximum degree \( p \) and dimension \( d \)
\( \xi \) = vector of independent random variables \( \xi_i \)
\( \rho(\xi) \) = joint probability measure
\( \sigma \) = standard deviation
\( \tau \) = Sobol index
\( \psi \) = orthogonal polynomial
\( \Sigma \) = variance-covariance matrix

I. Introduction

The Magnetospheric Multi-Scale (MMS) mission consists of four spacecraft in highly elliptic orbits that form a tetrahedron near apogee. Unlike other robotic missions, they not only have to perform conjunction assessment with other satellites in Earth orbit, but between each other [1].
Specifically, there is a potential risk of collision when the orbit planes of the four spacecraft cross at true anomalies of 90° and 270°. To better enable maneuver planning and execution, the ground system must identify any potential collision days in advance of the event. Several tools have been identified for quantifying the risk of collision, including classical Monte Carlo methods and approximations using polynomial chaos. A description of these tools and their use in the MMS ground system may be found in [1]. These methods provide a collision probability estimate as a function of the navigation uncertainty, but updates to the navigation state require a periodic duplication of the analysis, which increases the computation burden.

Another tool developed for conjunction assessment, based on a Wald Sequential Probability Ratio Test (e.g., see [2]), provides a sequential update to an estimated likelihood ratio, which accounts for the temporal changes in collision risk with navigation state updates. Specifically, the Wald epoch-state filter estimates the likelihood ratio of two hypotheses: (1) the miss distance is outside of the combined hard-body radius, and (2) the miss distance is within the hard-body radius. This ratio is then compared to alarm and dismissal thresholds to identify a potential collision, and is designed to better account for temporal changes in the collision probability that may cause false alarms or failed detections. The thresholds used in this procedure are defined by the false alarm and missed detection requirements for the mission, and are also a function of an unknown, aleatoric (random) probability of collision $P_c|\theta$ [3]. These thresholds, which are independent of updates to the navigation state-determined $P_c$, vary at each plane crossing, preventing the selection of fixed values appropriate for the whole mission. Additionally, $P_c|\theta$ is a product of the maneuver execution errors and the navigation uncertainty at the previous formation maintenance maneuver. To aid in the generation of the likelihood thresholds, this paper presents an application of polynomial chaos to estimate the probability of collision following an orbit maintenance maneuver.

Common methods of including maneuver uncertainties in orbit propagation use either Monte Carlo methods or assume a Gaussian distribution for the realized velocity error and navigation uncertainties. The Gates model [4] traditionally provides an estimate of the maneuver execution errors with an assumed Gaussian probability density function (PDF) for the a posteriori solution. However, as demonstrated in recent literature related to uncertainty propagation (e.g., see [5] and the
other references contained within), propagation of the satellite state PDF fails to remain Gaussian under some (common) conditions. Alternatively, Monte Carlo methods may be used to characterize the dispersion of realizations due to maneuver uncertainties. Such methods require the propagation of many realizations with a convergence rate approximately equal to the inverse of the square-root of the number of samples, making such methods computationally inefficient. This work instead considers the use of polynomial chaos expansions to efficiently quantify the solution uncertainty, with such techniques incorporating both the maneuver execution and navigation uncertainties and no Gaussian assumptions in the final probability density functions.

Polynomial chaos (PC) approximates the solution of a stochastic differential equation by projecting it onto a basis of orthogonal polynomials. The resulting approximation, known as a polynomial chaos expansion (PCE), has already been demonstrated for orbit uncertainty propagation [5] and to estimate the probability of collision for non-maneuvering spacecraft [6]. Unlike the previous applications of PC to the problem of conjunction assessment, the post-maneuver solution lends itself to a sparse PCE that allows for approximating a high-degree expansion with a reduced computation cost. Additionally, the random inputs are no longer assumed to be identically distributed, and a mixture of Gaussian and uniform random inputs are considered in the stochastic model. These methods, dubbed generalized Polynomial Chaos (gPC), allow for the use of arbitrary, random inputs to be considered in the generation of the PCE [7, 8]. For the current application, this allows for the definition of the maneuver execution errors via a pointing and force magnitude error.

Each MMS spacecraft includes four magnetic booms that extend 60 m from the center of mass, yielding a spherical hard body radius of 120 m for intra-formation conjunction assessment [1]. Each of these magnetic booms occupies a relatively small percentage of the total keep out sphere, resulting in a likely increase in the number of false alarms. Some methods of computing collision probabilities have been demonstrated to instead allow for non-spherical shapes (e.g., see [9]), and the method based on PCEs discussed here better quantifies the collision probability using a simple MMS shape model. When combined with the proposed PCE-based methods that account for possibly non-Gaussian maneuver uncertainties, this produces a smaller collision probability and reduces the number of false alarms.
This paper presents the application of polynomial chaos to estimating post-manuever collision probabilities for an MMS-like mission. To improve the tractability of the problem and leverage the sparse expansion, compressive sampling (CS) based estimation of the PCE, introduced in [10–12], reduces the number of orbit propagations required to generate the solution. This is demonstrated for: (1) an MMS intra-formation conjunction, and (2) a potential collision between an MMS spacecraft and debris in low-Earth orbit (LEO).

The paper begins with a brief description of the stochastic system and the maneuver execution errors in Section II, followed by an overview of PC and the methods of generating a PCE in Section III. The PCE-based method of computing the collision probability is discussed in Section IV, which also includes a formulation to account for a simple spacecraft shape model. Section V then presents a demonstration of the PCE-based solutions and a comparison to a Monte Carlo baseline to demonstrate their efficacy. This includes an implementation of the algorithm on a graphics processing unit (GPU), which reduces the computation time of the method. Finally, conclusions are discussed.

II. Problem Setup

Let $\xi \in \prod_{i=1}^{d} \Gamma_i \subseteq \mathbb{R}^d$ be a vector of $d$ independent random variables, with joint density function $\rho(\xi)$, representing the orbit state uncertainty. In this paper, $d$ is referred to as the stochastic dimension. For the purposes of this work, the elements $\xi_i$ of $\xi$ are assumed independent and not identically distributed, hence $\rho(\xi) = \prod_{i=1}^{d} \rho_i(\xi_i)$. The set of ordinary differential equations (ODEs) considered in this work for describing the temporal evolution of the satellite state are

$$\mathcal{A}(t, \xi; X(t_0), C_R, \Delta v(t_m)) = 0, \quad (t, \xi) \in [t_0, t] \times \prod_{i=1}^{d} \Gamma_i, \quad (1)$$

where $\mathcal{A}$ is a stochastic ODE operator, $t \in [t_0, t]$ is time, $X(t_0)$ is the initial position and velocity state of the spacecraft at $t_0$, $C_R$ is the coefficient of reflectivity, and $\Delta v(t_m)$ is the maneuver executed at $t_m \in [t_0, t]$. For the purposes of this work, a priori knowledge of the PDF for the input parameters $X$, $C_R$, and $\Delta v$ is assumed known or reasonably approximated. Estimation of satellite collision probability requires a solution to the stochastic operator $\mathcal{A}$ that describes the space of possible solutions $X(t, \xi)$. Additional input parameters may be employed, e.g., the coefficient of
drag, but they are not considered in this paper. See [6] for example applications.

![Diagram](image)

Fig. 1 Maneuver execution error represented by magnitude and pointing errors

The generation of a stochastic solution requires a model defining the relationship between $\xi$ and the initial state $X(t_0, \xi)$. For the cases considered in this paper, $\xi \in \mathbb{R}^{10}$ for propagation with a maneuver, and $\xi \in \mathbb{R}^{7}$ for a non-maneuvering object. The initial translation state PDF is assumed Gaussian at $t_0$ with mean $\bar{X}(t_0)$ and covariance $\Sigma_0$, i.e., $X \sim \mathcal{N}(\bar{X}(t_0), \Sigma_0)$. Similarly, $C_R \sim \mathcal{N}(\bar{C}_R, \sigma_{C_R}^2)$. Samples of the initial state given realizations of the random inputs $\xi \in \mathbb{R}^{7}$, $\xi \sim \mathcal{N}(0, \mathbb{I}_7)$ (where $\mathbb{I}_7 \in \mathbb{R}^{7 \times 7}$ is the identity matrix) are then generated via

$$
\begin{bmatrix}
X(t_0, \xi_1, \ldots, \xi_6) \\
C_R(\xi_7)
\end{bmatrix} =
\begin{bmatrix}
\bar{X}(t_0) \\
\bar{C}_R
\end{bmatrix} +
\begin{bmatrix}
L & 0 \\
0 & \sigma_{C_R}
\end{bmatrix}
\xi
$$

(2)

where $L$ is the lower Cholesky decomposition of $\Sigma_0$. In regards to implementation, $\xi$ may be generated using any pseudo-random number generator for Gaussian variables. The maneuver model considers a hybrid of Gaussian and uniform $\xi_t$ values, which then yields a deviation in the thrust direction and magnitude with respect to the nominal maneuver $\Delta \bar{v}$. The subscript on the following maneuver random inputs begin at eight since the first seven values are associated with the other stochastic variables previously described. Given $\Delta \bar{v}$ in Cartesian inertial coordinates, the realized vector is

$$
\Delta v(\xi_8, \xi_9, \xi_{10}) = \Delta \bar{v} + \delta \Delta v(\xi_8, \xi_9, \xi_{10}),
$$

(3)

$$
\delta \Delta v(\xi_8, \xi_9, \xi_{10}) = \bar{q}(\xi_9, \xi_{10}) \otimes (\xi_8 \sigma_{\text{mag}} \Delta \bar{v}) \otimes \bar{q}^*(\xi_9, \xi_{10}),
$$

(4)

where $\bar{q}$ is the quaternion defining the transformation from a maneuver error frame to the inertial
frame and is a function of $\xi_9$ and $\xi_{10}$, $\otimes$ is the quaternion multiplication operator, $\overline{q}$ indicates the conjugate quaternion, and $\sigma_{mag}$ is the maneuver magnitude error standard deviation as a percentage of the nominal. In this model, $\xi_8, \xi_9 \sim \mathcal{N}(0,1)$ and $\xi_{10} \sim \mathcal{U}(-1,1)$, i.e., one of the random inputs is uniformly distributed while the other two are Gaussian. Figure 1 illustrates the geometry of the maneuver execution error relative to $\Delta \mathbf{v}$ with $\overline{q}$ represented by two angles $\theta(\xi_9)$ and $\phi(\xi_{10})$. Two axes of the maneuver error frame are defined by the unit vectors

$$
\hat{T} = \frac{\Delta \mathbf{v}}{\|\Delta \mathbf{v}\|_2},
$$

$$
\hat{W} = \frac{(r(t_m) \times \Delta \mathbf{v}) / \|r(t_m) \times \Delta \mathbf{v}\|_2},
$$

where $r(t_m)$ is the position at the maneuver time. This yields one axis parallel to the maneuver and another normal to the maneuver and the absolute position vector. The angle $\theta(\xi_9)$ defines the realized maneuver error direction relative to the nominal maneuver, while $\phi(\xi_{10})$ is a rotation about the nominal maneuver direction. These rotations are represented mathematically by the quaternions

$$
\overline{q}_1(\xi_9) = \begin{bmatrix}
\sin(\theta(\xi_9)/2) \\
\cos(\theta(\xi_9)/2)
\end{bmatrix}, \quad \overline{q}_2(\xi_{10}) = \begin{bmatrix}
\sin(\phi(\xi_{10})/2) \\
\cos(\phi(\xi_{10})/2)
\end{bmatrix},
$$

where $\theta(\xi_9)$ is Gaussian distributed and defined by a given error PDF, e.g., $\mathcal{N}(0, \sigma^2_\theta)$, and $\phi(\xi_{10}) \sim \mathcal{U}(-\pi, \pi)$. Hence, given realizations of the random inputs $\xi_9$ and $\xi_{10},$

$$
\theta(\xi_9) = \sigma_\theta \xi_9, \quad \phi(\xi_{10}) = \pi \xi_{10},
$$

and, with Eq. (7),

$$
\overline{q}(\xi_9, \xi_{10}) = \overline{q}_2(\xi_9) \otimes \overline{q}_1(\xi_{10}).
$$

In this work, the approximation of the stochastic solution of the satellite state as a function of the previously described random inputs is accomplished through the use of a PCE. The next section provides a mathematical overview of such techniques as needed for the current problem.

### III. Polynomial Chaos

Methods based on polynomial chaos expansions (PCEs) provide a means for generating an approximation to the solution of a stochastic system by projecting it onto a basis of (multivariate)
polynomials orthogonal with respect to the probability measure of input variables, here denoted by $\xi$. The work in [13] first proposed this type of approximation, with methods based on Hermite polynomial chaos more recently established in [14, 15], among other works, and generalized to other types of orthogonal polynomials [7]. Unlike techniques that seek to propagate a PDF based on, for instance, an approximation of the Fokker-Planck equation, PCE methods provide a state solution at $t$ as an *explicit* function of $\xi$, i.e., a polynomial representation. The resulting expansion provides a computationally efficient means to represent any finite-variance, possibly non-Gaussian solution, and has already been demonstrated for orbit propagation [5] and conjunction assessment for non-maneuvering spacecraft [6].

In the context of PCEs, the orbit solution $X(t, \xi) \in \mathbb{R}^n$, assumed to have entries with finite variance, may be represented by the mean-squared convergent series

$$X(t, \xi) = \sum_{\alpha \in \mathbb{N}_0^d} c_\alpha(t) \psi_\alpha(\xi),$$

where $\mathbb{N}_0^d := \{(\alpha_1, \cdots, \alpha_d) : \alpha_i \in \mathbb{N} \cup \{0\}\}$ is the set of multi-indices of size $d$ defined on non-negative integers. The basis functions $\psi_\alpha(\xi)$, referred to as PC basis functions, are multi-dimensional polynomials of the form

$$\psi_\alpha(\xi) = \psi_{\alpha_1}(\xi_1)\psi_{\alpha_2}(\xi_2)\cdots\psi_{\alpha_d}(\xi_d),$$

where $\psi_{\alpha_i}(\xi_i)$ are univariate polynomials of degree $\alpha_i$ and are orthogonal with respect to the measure of $\xi_i$. For example, when $\xi_i$ has a Gaussian or uniform distribution, as in the present study, $\psi_{\alpha_i}(\xi_i)$ are Hermite or Legendre polynomials, respectively. This discussion assumes $\psi_{\alpha_i}(\xi_i)$ are normalized to have unit variance. Together with the tensor-product construction of Eq. (11), this implies the orthonormality of the PC basis functions $\psi_\alpha(\xi)$ with respect to the measure $\rho(\xi)$ of $\xi$.

In Eq. (10), the vector-valued coefficients $c_\alpha(t)$, referred to as PC coefficients, are given by the projection of $X(t, \xi)$ onto each basis function $\psi_\alpha(\xi)$,

$$c_\alpha(t) = \int X(t, \xi) \psi_\alpha(\xi) \rho(\xi) d\xi.$$ 

In practice, the infinite series in Eq. (10) is truncated, for instance, by limiting the total order of
$\psi_\alpha(\xi)$ to some finite value $p$. This leads to an approximate orbit solution $\hat{X}(t, \xi)$ given by

$$
\hat{X}(t, \xi) = \sum_{\alpha \in \Lambda_{p,d}} c_\alpha(t) \psi_\alpha(\xi),
$$

where the set of multi-indices $\Lambda_{p,d}$ is given by

$$
\Lambda_{p,d} := \{ \alpha \in \mathbb{N}_0^d : \| \alpha \|_1 \leq p, \| \alpha \|_0 \leq d \}, \tag{14}
$$

and has the cardinality (i.e., number of terms)

$$
P := \frac{(p + d)!}{p!d!}. \tag{15}
$$

In Eq. (14), $\| \alpha \|_1 = \sum_{i=1}^d \alpha_i$ and $\| \alpha \|_0 = \# \{ i : \alpha_i > 0 \}$ are the total order (degree) and dimensionality of the basis function $\psi_\alpha(\xi)$, respectively. The approximation is then refined by increasing $p$ to achieve a given target accuracy. An algorithm for generating the set $\Lambda_{p,d}$ may be found in [16].

As implied by Eq. (15), the number of terms $P$ increases exponentially in (asymptotically large) $p$ and $d$, leading to the issue of curse-of-dimensionality. Such effects may be mitigated through various means, which constitute an active area of research, see, e.g., [10, 17–22] and the references therein.

## A. PCE Solution Methods

This study only considers non-intrusive methods to estimate the coefficients $c_\alpha(t)$ to generate the PCE solution to the propagated orbit at time $t$. Such methods are dubbed non-intrusive since they treat an existing ODE solver, in this case the orbit propagation software, as a black box. The algorithm behind the non-intrusive methods used in this paper may be summarized by:

1. Generate $M_{PC}$ realizations $\xi_i$ distributed according to $\rho(\xi)$.

2. For each of the $M_{PC}$ random vectors, use the initial uncertainty in $X(t_0, \xi_i), C_R,$ and $\Delta v$ to generate a realization based on the random input $\xi_i$ (e.g., Eqs. (2) and (3)).

3. Using the existing ODE solver with each of the $M_{PC}$ realizations, solve for $X(t, \xi_i)$.

4. Generate the PCE coefficients $c_\alpha(t)$ based on $X(t, \xi_i)$ and the method of choice.
This procedure outlines a method for estimating a PCE at a fixed, but arbitrary, time \( t \). To reduce the complexity of notation, the presentation of the PCE estimation methods discussed in the remainder of this section omit the time variable \( t \). While there are several non-intrusive methods of computing PCE coefficients \( c_{\alpha} \), the current work uses procedures based on the least-squares regression [23] and compressive sampling [10], both of which are described further in the following sections.

1. Solutions via Lease-Squares Regression

The method to solve for \( c_{\alpha} \) based on least-squares regression uses random samples \( \xi \) from the density function \( \rho(\xi) \). Given the \( M_{PC} \) propagated states, one solution for the coefficients \( c_{\alpha} \) minimizes the least-squares cost function

\[
J(c_{\alpha}) = \sum_{i=1}^{M_{PC}} (\bar{X}(\xi_i; c_{\alpha}) - X(\xi_i))^T (\bar{X}(\xi_i; c_{\alpha}) - X(\xi_i)).
\]  

(16)

Together with Eq. (13), \( J(c_{\alpha}) \) in Eq. (16) may be rewritten as

\[
J(c_{\alpha}) = \|HC - Y\|_F^2,
\]  

(17)

where \( \| \cdot \|_F \) denotes the Frobenius norm, and

\[
H := \begin{bmatrix}
\psi_{\alpha_1}(\xi_1) & \cdots & \psi_{\alpha_P}(\xi_1) \\
\psi_{\alpha_1}(\xi_2) & \cdots & \psi_{\alpha_P}(\xi_2) \\
\vdots & \ddots & \vdots \\
\psi_{\alpha_1}(\xi_{M_{PC}}) & \cdots & \psi_{\alpha_P}(\xi_{M_{PC}})
\end{bmatrix}, \quad C := \begin{bmatrix}
c_{\alpha_1}^T \\
c_{\alpha_2}^T \\
\vdots \\
c_{\alpha_P}^T
\end{bmatrix}, \quad \text{and} \quad Y := \begin{bmatrix}
X^T(\xi_1) \\
X^T(\xi_2) \\
\vdots \\
X^T(\xi_{M_{PC}})
\end{bmatrix}.
\]  

(18)

Here, \( H \in \mathbb{R}^{M_{PC} \times P} \) is the measurement matrix, \( C \in \mathbb{R}^{P \times n} \) is the matrix of PCE coefficients, and \( Y \in \mathbb{R}^{M_{PC} \times n} \) is comprised of the propagated states at \( t \). Given this formulation and \( M_{PC} \geq P \) measurements, the minimum of Eq. (17) is attained by

\[
\hat{C} = (H^T H)^{-1} H^T Y.
\]  

(19)

Assuming the same samples are used at all times, \( H \) does not change as a function of \( t \). Hence, \( (H^T H)^{-1} H^T \) remains constant, and solving for \( c_{\alpha} \) at different \( t \) only requires assembling \( Y \) and evaluating a single matrix multiplication. Details on this procedure, along with similar formulations
to reduce the computation time in evaluating Eq. (13), may be found in [6]. The same paper also outlines a method for adaptive generation of the PCE solution via cross-validation.

2. Solutions via Compressive Sampling Methods

The nascent field of compressive sampling (CS), sometimes called compressive sensing, leverages the sparsity of the solution in a given basis – when it exists – for an accurate solution reconstruction with a number of measurements that is smaller than the cardinality of the basis [24, 25]. In this case, sparsity is defined by a large percentage of expansion coefficients that are small, i.e., most information on the function is provided by a small number of terms. The non-zero components are not known \textit{a priori}, and are identified in the solution method by minimizing a sparsity-promoting norm of the coefficients. For sparse PCEs, i.e., when a large fraction of coefficients in any of the columns of $C$ in Eq. (18) is negligible, compressive sampling methods may be employed as an alternative to least-squares regression. Such methods require $M_{PC} \ll P$ propagated states, thereby improving the efficiency of estimating the PCE for such cases [10, 12].

A key assumption in standard PCE methods is that the solution of interest depends \textit{smoothly} on the random inputs; that is, the solution and its derivatives (with respect to the inputs) are bounded, and do not exhibit discontinuities or sharp gradients. In such cases, the expansion coefficients decay to zero quickly [26] thus implying a low degree expansion is sufficient to approximate the solution. In addition to such sparsity, random inputs may contribute differently to the variability of the solution, e.g., the solution may be nearly constant along certain inputs. Such an anisotropic dependence results in faster decay of a subset of PCE coefficients, e.g., coefficients of higher order basis functions corresponding to near-constant directions decay to zero quickly, which yields sparsity of the PCE. Except for certain classes of problems with random inputs, see, e.g., [27–29], it is not generally possible to know \textit{a priori} how sparse a solution of interest is in a given PC basis. Nevertheless, it should be noted that a complete lack of sparsity in a low-order PCE, specially for high-dimensional random inputs, indicates an inefficient choice of PCE basis for the problem at hand. The remainder of this section outlines one CS technique for approximating sparse PCEs.

Unlike the method presented in Eq. (19), this presentation of the CS-based methods describes
the PCE of the components of the state vector $X$, independently from each other. Let $c \in \mathbb{R}^P$ and $y \in \mathbb{R}^{M_{PC}}$ denote the $i$-th column of $C$ and $Y$ in Eq. (18), respectively. Stated differently, $c$ and $y$ are, respectively, the PCE coefficient vector and realizations of the $i$-th component of the state vector $X(\xi)$. The most basic compressive sampling formulation aims at computing a sparsest $c$ from the optimization problem

$$
\min_c \|c\|_0 \quad \text{subject to} \quad \|Hc - y\|_2 \leq \varepsilon, \tag{20}
$$

where $\varepsilon \geq 0$ accounts for the finite order PCE truncation, thus it is referred to as the truncation error. The global minimum solution of Eq. (20) may not be unique and is generally NP-hard to compute: the cost of a global search is (asymptotically) exponential in $P$. To overcome the complexity of solving Eq. (20), several relaxation techniques primarily based on $\ell_1$-minimization [25, 30–32] and greedy pursuit [33–38] have been developed. Methods based on $\ell_1$-minimization search for solutions $c$ with minimum $\ell_1$ norm (a sparsity promoting norm), instead of the $\ell_0$ norm, in Eq. (20). This allows for the use of convex optimization tools with tractable computation cost. As opposed to performing an exhaustive search for finding the non-zero components of $c$ in Eq. (20), greedy pursuit methods successively find one or more components of $c$ that lead to largest improvement in the approximation.

The present work employs the greedy pursuit algorithm Orthogonal Matching Pursuit (OMP), [33, 34, 37], to estimate the sparse PCE coefficients $c$. The algorithm is initialized with

$$
\hat{c} = 0, \quad \delta = y, \quad \Lambda = \emptyset, \tag{21}
$$

where $\Lambda \subseteq \Lambda_{p,d}$ is the active set of multi-indices indicating non-zero entries of the approximate solution $\hat{c} \in \mathbb{R}^P$, and $\delta = y - H\hat{c}$ denotes the corresponding residual vector. At any iteration, OMP identifies an index $j$ corresponding to a column in $H$ to be added to $\Lambda$. This column is chosen such that the $\ell_2$-norm of the residual, $\|y - H\hat{c}\|_2$, is maximally reduced. It is straightforward to verify that $j$ is given by

$$
j = \arg \max_{i \notin \Lambda} \frac{H_i^T \delta}{\|H_i\|_2}, \tag{22}
$$

where $H_i$ is the $i$-th column of $H$. The active set $\Lambda$ and the non-zero entries of $\hat{c}$ supported on $\Lambda$,
here denoted by \( \tilde{c}_A \), are then updated by

\[
\Lambda \leftarrow \Lambda \cup \{j\},
\]

\[
\tilde{c}_A = (H_A^T H_A)^{-1} H_A^T \delta,
\]

(23)

in which \( H_A \) is the sub-matrix of \( H \) whose columns are those in \( H \) corresponding to the active set \( \Lambda \). Although the normal solution to the least squares estimator is given in Eq. (23), practical application should use, for example, QR decomposition to improve numerical stability [39]. The residual \( \delta \) is then updated and the process defined by Eqs. (22) and (23) is repeated until \( \| \delta \|_2 \leq \varepsilon \) or a maximum number of iterations is reached. For the results of this paper, \( \varepsilon \) is estimated via cross-validation as described in [10]. Each iteration of OMP, i.e., updating \( \Lambda \) and solving for \( \tilde{c}_A \), may be implemented with a cost of \( O(N \cdot P) \) [40].

It is worthwhile to note that the CS procedure outlined above does not require a time-independent sparsity of solution. Stated differently, the set of active PC basis \( \Lambda \) changes as a function of time, thus requiring the execution of the OMP algorithm for each time instance the solution is sought for. In the case of the MMS-like scenario presented later in the paper, the sparsity in the solution is a result of the maneuver, i.e., the random inputs associated with the maneuver execution errors dominate the stochastic solution. However, as illustrated in the results section, other inputs are not negligible and coupled interactions yield high-degree terms in the PCE. Additionally, the relative contribution of the inputs varies in time, hence, different solutions must be generated at different times to account for variations in \( \Lambda \).

The software used in this work is based on the OMP implementation in SparseLab 2.1\(^1\), but was converted to Python and customized for the current application. Although the generation of the PCE via CS in the current implementation does not use an automated procedure for the selection of \( M_{PC} \) or \( p \), it does compare to a smaller number of independent samples for cross-validation. For cases where the root-mean square error, when compared to the independent samples, fails to agree with the required precision, a new PCE is generated with new samples. As discussed later, the tests considered in this paper seldom require the regeneration of the PCE in this manner.

\(^1\) Available at [http://sparselab.stanford.edu/](http://sparselab.stanford.edu/)
Estimating a satellite collision probability requires estimates $\hat{C}$ at multiple times, but the iterative procedure previously outlined may yield additional computation time when compared to the least-squares method in [6]. It should be noted that such a computation cost is negligible as compared to the computation time required for generating random realizations of the states. As may be observed from the results of Section V, the latter cost will make the least-squares construction of the PCE significantly more expensive than the CS-based counterpart. To further reduce the number of iterations in the OMP algorithm, the implementation employs an estimate at a previous time to “warm start” the procedure. In this case, the initialization in Eq. (21) is removed in favor of the values from the previous use of the algorithm. This can reduce the number of OMP algorithm iterations if the PCE at time $t_k$ is similar to that of $t_{k-1}$, which is the case if $t_k - t_{k-1}$ is sufficiently small. The algorithm will correct any differences in the solution and add elements to $\Lambda$ if necessary. For the case where an element is included in the solution in $t_{k-1}$ but not needed at $t_k$, its magnitude is reduced by the OMP algorithm. To prevent the accumulation of such sufficiently small elements in the PCE, a check should be added to dismiss these elements of $\hat{C}$.

IV. Collision Probability Estimation

Given their similarities to such methods, the PCE-based methods are framed in the context of Monte Carlo analyses. However, the PCE algorithm trades the computationally expensive ODE solver for simple polynomial evaluations. This section first outlines methods in the context of Monte Carlo, and then discusses alterations required when using PCEs.

A. Monte Carlo Estimation of $P_c$ with a Spherical Keep-Out Radius

For the translation state vector $X_l(t_0) \in \mathbb{R}^6$ for satellite $l$, let $\overline{X}_l(t_0)$ be the mean at time $t_0$ with covariance matrix $\Sigma_{0,l} \in \mathbb{R}^{6 \times 6}$. Monte Carlo methods employ, for example, Eq. (2) to generate $M_{MC}$ trials from the random inputs $\xi_i$. Each trial $X_l(t_0, \xi_i)$ is then propagated to get $X_l(t_k, \xi_i)$. For two satellites ($l = 1, 2$), the square of their separation at $t_k$ is

$$s^2(t_k, \xi_i, \xi'_i) = (r_1(t_k, \xi_i) - r_2(t_k, \xi'_i)) \cdot (r_1(t_k, \xi_i) - r_2(t_k, \xi'_i)),$$  

(24)
where the \( \cdot \) operator is the normal vector dot product, \( r_i \) is the position portion of \( X_i \), and the prime on \( \xi_i \) indicates a set of samples with \( \xi_i \neq \xi_i' \). The resulting \textit{instantaneous} probability of collision is

\[
P_c = \frac{\text{count}(s^2(t_k, \xi_i, \xi_i') \leq R^2)}{M_{MC}},
\]

(25)

where \( R \) is a given spherical hard-body radius, and the \text{count}() operator indicates the number of true results of the argument over \( i = 1, \ldots, M_{MC} \).

Using the realizations at multiple points in time, i.e., \( X(t_k, \xi_i) \), a probability of survival \( P_s \) may be computed. The complement of the survival probability, \( P'_s \), describes the probability of collision over a period of time as opposed to a single epoch. To enable the computation of \( P_s \) or \( P'_s \), let

\[
m(\xi_i, \xi_i') = \min_{k=1, \ldots, K} s^2(t_k, \xi_i, \xi_i'),
\]

(26)

which produces the minimum separation distance for \( t_k \in [t_1, t_2, \ldots, t_K] \). The complement of the survival probability is then

\[
P'_s = 1 - P_s = \frac{\text{count}(m(\xi_i, \xi_i') \leq R^2)}{M_{MC}}.
\]

(27)

The method defined in Eqs. (26) and (27) assumes the times \( t_k \) are dense enough to provide adequate accuracy when computing \( P'_s \). A refinement for insufficient temporal density is proposed below with another alteration for non-spherical bodies.

B. PCE-Based Estimation of \( P_c \) and \( P'_s \)

Estimating \( P_c \) or \( P'_s \) via a PCE differs slightly from the above procedure in that using a polynomial surrogate requires the propagation of significantly fewer samples. Instead, a relatively small number of propagations are used to generate the coefficients \( c_\alpha \) in the PCEs for each spacecraft. The set of inputs \( \{\xi_i\}_{i=0}^{M_{MC}} \) and Eq. (13) then directly produce the needed Monte Carlo trials \( \widehat{X}(t_k, \xi_i) \).

In other words, most evaluations of the orbit propagator required for Monte Carlo analysis are substituted with the more computationally efficient polynomial evaluations. Eqs. (24)-(25) are then used with the PCE output to estimate \( P_c \). To estimate \( P'_s \), the evaluation of Eq. (13) using \( \hat{C} \) for a PCE representing the state at \( t_k \) may be employed to generate \( \widehat{X}(t_k, \xi_i) \) for \( k = 1, \ldots, K \).
Depending on the scenario, orbit propagation may take seconds of computer processing time per evaluation, but these polynomial evaluations are on the scale of thousands per second on a single computer core. Hence, the PCE-based methods reduce the computation time required to generate the realizations required to compute $P_c$ and $P_s'$. Of course, the probability estimate is affected by the accuracy of the PCE realizations. Methods exist to account for errors in the PCE when calculating a rare failure probability by combining PCE evaluations with additional Monte Carlo orbit propagations [41], but such augmentations are not required for this current work.

C. Shape-Dependent Estimation of Collision Probability

The method of computing the collision probabilities given in Eqs. (24)-(27) assumes a spherical collision volume (CV), but such limitations are not required for Monte Carlo-based analyses. Most satellites are more rectangular in shape, and a spherical definition of the CV increases the number of false alarms. In the results of the current paper, a non-spherical CV is used to demonstrate a more shape-dependent assessment of collision risk. To limit additional computation time, the spherical definition of Eqs. (24)-(27) identifies cases that require further analysis., i.e., if $s^2 \geq R^2$, then the satellite-dependent shape models are used to make the final identification of a collision. This work considers simple shapes with primary axes aligned with the inertial coordinate axes, which allows for easy application of the hyperplane separation theorem. Higher-fidelity methods of collision detection may be employed, but require additional assumptions or shape and attitude knowledge. See [42] for such an example.
Figure 2 illustrates the shape model for the MMS-like spacecraft considered in this work. It is based on the current spacecraft design, which is comprised of a central octagon nominally spinning about the axis of symmetry. There are two 12.5 m booms in opposition along the axial direction, with four 60 m booms in the plane perpendicular to the angular velocity vector. Instead of accounting for the instantaneous attitude in the estimation of collision probability, this model employs a cylinder with radius and height given in the figure. A second, longer cylinder in the axial direction defines the CV associated with the vertical booms, which, in this work, is assumed to be parallel to the inertial \( z \) axis. For a potential collision between two MMS-like spacecraft, if both of the conditions

\[
\Delta Z^2 \leq L_{Z,i,j} = \left( \frac{h_{1,i} + h_{2,j}}{2} \right)^2, \tag{28}
\]

\[
\Delta X^2 + \Delta Y^2 \leq L_{XY,i,j} = (R_{1,i} + R_{2,j})^2, \tag{29}
\]

are true for any combination of \( i, j \in \{1, 2\} \), then the test is considered a positive collision. In Eqs. (28) and (29), \( \Delta X, \Delta Y, \) and \( \Delta Z \) indicate the differences in the spacecraft inertial position. For the case of a conjunction with (possibly tumbling) space debris, a positive case satisfies both of the conditions

\[
\Delta Z^2 \leq L_{Z,i,l} = \left( \frac{h_{l,i}}{2} + r \right)^2, \tag{30}
\]

\[
\Delta X^2 + \Delta Y^2 \leq L_{XY,i,l} = \begin{cases} 
(R_i + r)^2 & |\Delta Z| \leq h_{l,i}/2 \\
(R_i + \sqrt{r^2 - (|\Delta Z| - h_{l,i}/2)^2})^2 & |\Delta Z| > h_{l,i}/2,
\end{cases} \tag{31}
\]

for \( i \) equal to one or two, and \( r \) is the radius of the spherical volume containing the debris. The second condition in Eq. (31) for \( |\Delta Z| > h_{l,i}/2 \) is derived from the boundary case where the two volumes touch and accounts for the curvature of the sphere when near the corner of the cylinder. An imaginary number only results from the square root operation when the first condition (Eq. (30)) fails.

D. High-Velocity Collisions

One of the test cases considered in this paper requires special treatment of collision detection for high relative velocities. In such cases, the resolution in the propagator output may not be sufficient to detect a collision, which possibly leads to failed detections (i.e., tunneling) and a low collision
probability estimate. Decreasing the integrator stepsize to account for these high relative velocities is not always feasible for computation reasons.

Instead, the algorithm used here combines a set of samples with reduced density (stepsize equal to 30 s) with interpolation and Brent’s method [43] to minimize the square of the separation distance (instead of the discrete problem in Eq. (26)). Brent’s algorithm was used to identify possible collisions in the Monte Carlo-based estimation of \( P_s' \) in [44], and this work uses the PCE-based realizations to generate the interpolating polynomials. If \( m(\xi_i, \xi_i') \) from Eq. (26) is less than some gating parameter, in this case 4 km, then univariate interpolating polynomials (centered at the minimizing index \( k \)) are generated to approximate \( r_1(t, \xi_i) \) and \( r_2(t, \xi_i') \). For a spherical CV, the condition \( s^2(t, \xi_i, \xi_i') < R^2 \) identifies a positive test case and Brent’s method provides the time \( t \) that minimizes Eq. (24). For scenarios with a shape-dependent model, Brent’s method is used to find the time that minimizes the cost function

\[
J(t) = \frac{\Delta X^2 + \Delta Y^2}{L_{XY,*}} + \frac{\Delta Z^2}{L_{Z,*}} + \gamma (\kappa_1 + \kappa_2),
\]

where

\[
\kappa_1 = \begin{cases} 
0 & \text{if } (\Delta X^2 + \Delta Y^2) / L_{XY,*} < 1.0 \\
\frac{\Delta X^2 + \Delta Y^2}{L_{XY,*}} - 1 & \text{otherwise,}
\end{cases}
\]

\[
\kappa_2 = \begin{cases} 
0 & \text{if } \Delta Z^2 / L_{Z,*} < 1.0 \\
\frac{\Delta Z^2}{L_{Z,*}} - 1 & \text{otherwise,}
\end{cases}
\]

\( L_{XY,*} \) and \( L_{Z,*} \) are the appropriate boundaries for the given shape model in Eqs. (28)-(31), and \( \gamma \) is a design parameter. The final term in Eq. (32) imposes a penalty for times where neither of the collision criteria are satisfied. This work considers the case where \( \gamma = 100 \). If both of the first two terms in Eq. (32) are less than 1.0 at the minimum time \( t \), then a positive test is identified. While Brent’s method may converge on a discontinuity in the cost function instead of its minimum when it is not smooth, both are equal for the purposes of this work.
Table 1 Mean initial conditions (km and km/s) at 23-Feb-2015 15:37:48.78 UTC

<table>
<thead>
<tr>
<th>Satellite</th>
<th>( X )</th>
<th>( Y )</th>
<th>( Z )</th>
<th>( \dot{X} )</th>
<th>( \dot{Y} )</th>
<th>( \dot{Z} )</th>
<th>( C_R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMS1</td>
<td>30,671.521</td>
<td>12,606.602</td>
<td>1,702.861</td>
<td>2.001580</td>
<td>3.004536</td>
<td>1.248378</td>
<td>1.8</td>
</tr>
<tr>
<td>MMS2</td>
<td>30,759.756</td>
<td>12,996.649</td>
<td>1,757.885</td>
<td>1.969278</td>
<td>3.000950</td>
<td>1.243629</td>
<td>1.8</td>
</tr>
<tr>
<td>DEB</td>
<td>3,790.410</td>
<td>-6,397.090</td>
<td>-2,854.436</td>
<td>5.295308</td>
<td>4.125529</td>
<td>-2.237770</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 2 Test case conjunction properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Case 1 Value</th>
<th>Case 2 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Since Epoch ( (t_{CA,1}, t_{CA,2}) )</td>
<td>1290669.99246 s</td>
<td>1292769.99233 s</td>
</tr>
<tr>
<td>Distance of Propagated Mean (m)</td>
<td>120</td>
<td>3.3\times10^{-3}</td>
</tr>
<tr>
<td>Relative Speed of Propagated Mean (m/s)</td>
<td>0.1414</td>
<td>2,447.77</td>
</tr>
<tr>
<td>MMS1 True Anomaly at ( t_{CA} )</td>
<td>268.756⁰</td>
<td>358.559⁰</td>
</tr>
<tr>
<td>MMS2 True Anomaly at ( t_{CA} )</td>
<td>268.759⁰</td>
<td>-</td>
</tr>
<tr>
<td>DEB True Anomaly at ( t_{CA} )</td>
<td>-</td>
<td>0.0⁰</td>
</tr>
<tr>
<td>Spherical Keep-Out Radius (( R, ) m)</td>
<td>120</td>
<td>70</td>
</tr>
</tbody>
</table>

V. Numerical Examples

This section presents two sample conjunctions based on the MMS mission that demonstrate the reduced computation time for post-maneuver collision probability estimation when compared to Monte Carlo methods. The description begins with an overview of the test cases and their simulation methods, followed by the results for each scenario in separate sections.

A. Test Case Description and Computation Methods

Table 1 provides the initial mean Cartesian states for the three satellites considered in the following tests. The first two states, MMS1 and MMS2, designate two spacecraft in an MMS-like orbit with a potential collision approximately 15 days from the epoch. However, although the trajectory for MMS2 shares similarities with the spacecraft in the nominal MMS formation, this state is designed to yield a possible collision at \( t_{CA,1} \) in the cross-track direction without any other mission design constraints. The initial root sum square 1\( \sigma \) state uncertainty for these spacecraft is approximately 4.29 m and 5.7\times10^{-4} m/s in position and velocity, respectively. The
third initial condition represents a piece of debris in low-Earth orbit, dubbed DEB, that poses a risk to MMS1 shortly after the previous case. The orbit state for this object has initial Gaussian standard deviations of 100 m and 0.1 m/s. All spacecraft have a $C_R$ of 1.8 with a 5% standard deviation.

The MMS1 and MMS2 spacecraft perform a maneuver at $t_m = 2910.46$ s past the epoch time. In inertial Cartesian coordinates, the nominal impulse is

$$\overrightarrow{\Delta v} = \begin{bmatrix} 0.48 \\ -0.81 \\ 5.1 \end{bmatrix} \text{ m/s}, \tag{35}$$

with a maneuver magnitude $3\sigma$ uncertainty of 1% and a $1\sigma$ pointing error ($\theta$) of 1°. This emulates the formation resize maneuver to transition between the 160 km and the 60 km formation sizes during Phase 1a of the MMS mission. More information on the MMS mission phases may be found in [1].

Table 2 describes the potential collisions considered in the following sections. For this discussion, the time of closest approach $t_{CA}$ is defined as the time of minimum separation between the propagated mean trajectories, and does not necessarily indicate the time of maximum collision probability. As indicated by the true anomaly values, the MMS1-MMS2 conjunction occurs at approximately 270°, which coincides with one of the two regions of higher risk for the mission [1]. The second scenario considers a conjunction between the MMS1 satellite and the DEB object near perigee. This work assumes that a potential collision and the period of risk have already been identified, which is consistent with tools designated for use in the current MMS ground system [1].

All propagations used the CU-TurboProp orbit propagation toolbox, which provides an interface between Python and more computationally efficient tools written in C [45]. Numerical integration of the MMS-like satellites employs a Dormand-Prince 8(7) [46] integrator until $t_{CA,1}=1,500$ s, and switches to an eighth-order Gauss-Jackson [47] (GJ8) algorithm with a 30 s stepsize until $t_{CA,2}=300$ s. The variable-step method is used for the initial phase to efficiently propagate the highly eccentric ($\sim0.81$) MMS orbit, while the fixed-step integrator provides a more computationally efficient solution when requiring a solution every 30 s. The initial time of interest, $t_{CA,1}=1,500$ s, is selected to occur before the initial risk of collision for the MMS1-MMS2 case. Simulation of the DEB orbit only used the GJ8 method with a 30 s stepsize throughout. Force models include gravity,
Table 3 Properties of PCEs used for collision probability estimation

<table>
<thead>
<tr>
<th>Property</th>
<th>MMS1/MMS2</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Dimension ( (d) )</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Maximum Degree ( (p) )</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Number of Training Samples ( (M_{PC}) )</td>
<td>240</td>
<td>156</td>
</tr>
<tr>
<td>RMS Position Design Tolerance ( (\varepsilon, \text{m}) )</td>
<td>5\sqrt{3}</td>
<td>0.1</td>
</tr>
</tbody>
</table>

third-body, and solar radiation pressure (SRP) perturbations. The \( 21 \times 21 \) EGM96 gravity model [48] was used for the central body gravity field with the Earth fixed coordinate system realized with the IAU2000A\( _{R06/2006} \) [49] orientation model. The Jet Propulsion Laboratory DE 421 [50] provided the positions of the Sun and Moon over time, both of which are included as perturbing bodies in the ODE solver. The SRP area-to-mass ratio for all spacecraft is 0.0017 \( \text{m}^2/\text{kg} \) with a solar flux of 1358 \( \text{W/m}^2 \). Atmospheric drag forces are included using the standard 1976 atmosphere model (e.g., see [51]), a coefficient of drag equal to 2.8, and an area-to-mass ratio of 0.006 \( \text{m}^2/\text{kg} \). These drag force parameters are the same for each of the three simulated spacecraft.

Both cases compare the PCE-based results to a Monte Carlo baseline. The data were generated using the Janus supercomputer\(^2\), with, except where otherwise noted, \( 10^7 \) samples for comparison. Unlike the samples used for PCE generation (described previously), the Monte Carlo propagations used step sizes of 1 s during the MMS1-MMS2 conjunction, and 0.1 s for the MMS1-DEB case. Hence, the Monte Carlo samples provide a higher resolution when compared to those used in the PCE solution. Otherwise, the propagation methods are the same.

Generation of the PCEs employed a Python-based implementation of the least squares and CS algorithm introduced in Section III. Table 3 summarizes the parameters used to generate the PCEs and the required number of training samples. Details related to their selection follow in the discussion specific to a given test case. For the MMS-like conjunction, PCE solutions computed via least squares and CS algorithms are compared in the next section, but only the CS-based solutions

\(^2\) Information at https://www.rc.colorado.edu/services/compute/janus
are used when computing collision probabilities. PCE solutions for the DEB satellite used least squares with the cross-validation procedure described in [6], which allows for autonomous generation of the solution. Evaluation of the PCEs and computation of $P_c$ and $P'_c$ are performed using all eight cores of an Intel Core i7 3.4 GHz processor running RedHat Enterprise Linux 6.4. Faster estimation of the probabilities employs the GPU implementation described below.

The collision probability estimation procedure is also implemented in the Compute Unified Device Architecture (CUDA) parallel programming platform, which makes use of NVIDIA GPUs as general-purpose parallel processors. Modern GPUs have hundreds of processing cores, which makes them ideal for parallel computations. The GPU software runs on the previously mentioned RedHat desktop, employs an NVIDIA GeForce GTX 550 Ti with 1GB memory, and is compiled using the gcc 4.4.7 compiler and CUDA toolkit version 5.0. It is noted that the GTX 550 Ti is a standard GPU included in many desktop computers and is not optimized for high-performance computing. As illustrated later, it still yields reduced computation times for computing $P'_c$.

B. Case 1: MMS Intra-Formation Conjunction

This section describes the performance of the PCE-based methods for the MMS1-MMS2 test case. It begins with a description of the PCE accuracy and justifies the use of compressive sampling for generating a stochastic solution after an MMS formation maintenance maneuver. The collision probabilities are then computed and compared to those generated via Monte Carlo. Analysis with a spherical CV uses $\mathcal{R} = 120$ m, and the shape-dependent models use the definition of Figure 2 and Eqs. (28) and (29). Preliminary tests to compare PCEs generated via least squares and CS methods use $p = 2, \ldots, 7$ and $d = 10$. Subsequent tests to compute a collision probability use the $p = 7$ PCEs generated via compressive sampling for both of the MMS spacecraft. PCE solutions are generated at every integration step (30 s) for $t_k \in [t_{CA,1} - 1500, t_{CA,1} + 1500]$ s.

Figure 3 depicts the Sobol indices (e.g., see [16]) $\tau$ for the Cartesian position $X$, $Y$, and $Z$ solutions for the MMS1 case. These indices indicate the relative contribution of each random input to the solution variance, and the figure provides the evolution of $\tau$ over time. The dominant input is $\xi_8$, which is associated with the magnitude of the maneuver execution error. However, this can
change in short periods, and the solution becomes more sensitive to other inputs at such times. Since mission operations cannot dictate the time and location of a possible collision, dimension truncation, i.e., ignoring some inputs, is not recommended for conjunction assessment. For this reason, the stochastic solution accounts for all 10 inputs to the system. As illustrated later, this does not impede the tractability of estimating $P_s$ or $P_c$.

The PCE coefficients for the solution at $t_{CA,1}$ for MMS1 are provided in Figure 4. Each plot provides the absolute value of the coefficients for the three expansions (one for each position component). The least squares solution is generated with $M_{PC} = 10,000$ and $p = 6$, while the compressive sampling result uses $M_{PC} = 200$ and $p = 7$. A small number of terms dominate the solution, with most of the coefficients below $10^{-2}$ km, which is smaller than the least-squares approximation accuracy (described later). Accurately estimating PCE coefficients greater than 10 m is desired to limit position solution errors to an order of magnitude less than $R$. This sparsity implies that CS methods may reduce the number of samples required, with such a solution provided in the same image. To the scale of the figure, the CS-based solution agrees with the dominant terms of the least-squares solution and allows for a higher degree expansion. For the $X$-coordinate PCE, the CS-based solution
Fig. 4 Absolute value of PCE coefficients for MMS1 at $t_{CA,1}$ via different methods

Fig. 5 MMS1 PCE at $t_{CA,1}$ realization accuracy as a function of $M_{PC}$ and $p$

requires only 15 terms, which provides an additional benefit when later evaluating the polynomial surrogate.

Figure 5 illustrates the PCE realization accuracy as a function of $M_{PC}$ and $p$ for MMS1 at $t_{CA,1}$. All PCE solutions are generated using the same ordered set (or subset) of $10^4$ training samples. The three-dimensional root-mean-square (RMS) position error is computed using the same set of $10^5$
test samples, which is independent of the training data. The RMS position error is computed via
\[ \varepsilon_{\text{RMS}} = \sqrt{\frac{\sum_{i=1}^{M} ||\tilde{r}(t, \xi_i) - r(t, \xi_i)||^2}{M}}, \] (36)
where \(r\) is a position vector generated via the ODE solver, \(\tilde{r}\) is its PCE approximation via Eq. (13) with coefficients \(c\), and both are computed using the same inputs \(\xi_i\). To complement Fig. 5, Table 4 provides: (i) the number of PCE terms estimated as a function of \(p\) and, (ii) the OMP convergence tolerance employed for the CS-based approximation. The column for the CS-based solution indicates the number of non-zero terms identified using the greedy selection criteria in Eq. (22) with 200 samples. Different convergence tolerances \(\varepsilon\) are required with variations in \(p\) due to the maximum accuracy achievable. The grouping of the different CS-based solutions along the abscissa of the figure results from the similarities in the number of terms in the expansion. The only case that achieves the desired accuracy of 10 m is the CS-based solution with \(p = 7\) and \(M_{\text{PC}} \geq 200\) training samples. Generation of this solution uses an OMP convergence tolerance \(\varepsilon\) of 5 m, which yields an RMS position error cross-validation tolerance of \(5\sqrt{3}\) m. No least squares solution for \(p = 7\) is provided since it requires over \(10^4\) training samples. Since solving for the PCE coefficients via least squares requires \(M_{\text{PC}} \geq P\), and \(P\) is exponential in \(p\), the number of samples needed grows rapidly to achieve a given accuracy. Since the PCE is sparse, the compressive sampling method of generating a PCE allows for \(M_{\text{PC}} < P\) and a reduced dependence on \(p\). Since Fig. 5 demonstrates that a relatively high-degree expansion is required to achieve the required accuracy in this scenario, the CS-based methods provide the most efficient means for approximating the PCE.
when considering the number of training samples required.

In theory, reducing the OMP algorithm tolerance will also improve accuracy, but the sparsity of the PCE solution is reduced in such solutions. The OMP algorithm then requires an increased number of samples to improve the solution accuracy. A value of ε too small when given a set of training samples can lead to an overfitting of the solution, which is identified using cross-validation after completion of the OMP algorithm. Forty independent samples are used to compute the cross-validation-based error, and are not included in the abscissa of Fig. 5. However, statements of computation time for estimating collision probabilities includes the generation of all 240 samples used in estimating the PCE via compressive sampling.

Figure 6 describes the variations in runtime for the compressive sampling and least squares methods. This runtime includes the effort required to generate all PCEs for the MMS1 case at 30 s intervals in the range \([t_{\text{CA,1}} - 1500, t_{\text{CA,1}} + 1500]\) s with 200 and \(P\) samples provided to the OMP and least squares algorithms, respectively. For this case, the OMP algorithm execution time is less than that of the least-squares algorithm with \(p > 3\). Generation of the \(p = 7\) PCEs takes approximately 3 s. In addition to providing improved accuracy and needing fewer training samples, the OMP algorithm requires almost two orders of magnitude less computation time than least squares.

Figure 7 describes the relative position of the MMS1 and MMS2 spacecraft near \(t_{\text{CA,1}}\). The
position is represented in the conjunction plane with basis vectors

\[
\hat{X} = \frac{r_1(t_{CA,1}) - r_2(t_{CA,1})}{\|r_1(t_{CA,1}) - r_2(t_{CA,1})\|_2}, \tag{37}
\]

\[
\hat{Y} = \frac{(r_1(t_{CA,1}) - r_2(t_{CA,1})) \times (\dot{r}_1(t_{CA,1}) - \dot{r}_2(t_{CA,1}))}{\| (r_1(t_{CA,1}) - r_2(t_{CA,1})) \times (\dot{r}_1(t_{CA,1}) - \dot{r}_2(t_{CA,1})) \|_2}, \tag{38}
\]

i.e., the plane perpendicular to the relative velocity vector at \(t_{CA,1}\). The dashed lines represent the expected evolution of the relative position in the conjunction frame under an assumption of linear motion. Although further analysis is required to quantify the effects, this figure, especially the change in the \(X\) component, indicates that the true relative motion is nonlinear during the period of conjunction, which would violate one of the common simplifying assumptions in the classic methods of collision probability estimation (e.g., see [52]).

Figure 8 provides the collision probabilities, both \(P_c\) and \(P_s'\), for Case 1 and a spherical CV. In the top figure, the dependence on time implied by the abscissa indicates the final time considered when computing \(P_s'\). The horizontal trend at the beginning and end of the time interval and \(P_c\) indicate that this analysis encompasses the full period of collision risk. The dashed lines depict the 3\(\sigma\) range of Monte Carlo solutions with variations due only to convergence, i.e., these bounds do
not account for modeling errors. These standard deviations are computed via

$$
\sigma_{P} = \sqrt{\frac{P (1 - P)}{M_{MC}}}.
$$

which is the expected uncertainty for a random value described by a binomial distribution as a function of $M_{MC}$. $M_{MC} = 10^7$ samples are used in the computation of these probabilities, and results demonstrate agreement between the Monte Carlo- and PCE-based solutions within the uncertainty of the baseline. The insets zoom in on the period of highest probability rate, demonstrating that the PCE-based solution agrees with MC for this time span. Based on Eq. (39) and a $3\sigma$ accuracy requirement of approximately one digit, i.e., $M_{MC}$ large enough such that $3\sigma/P < 0.1$, the estimation of $P'_s$ for Case 1 and a spherical CV requires almost $1.5 \times 10^6$ samples. The results presented here include more samples to increase confidence in the results. As discussed later in this section, the generation of the larger number using a PCE-based solution may still be accomplished in tens of seconds on a central processing unit (CPU) or GPU, but some speed-up may be achieved by reducing the number.

Table 5 provides the Monte Carlo- and PCE-determined values of $P'_s$ when using the shape-dependent model of Eqs. (28) and (29). These values indicate a decrease in $P'_s$ by two orders of
Table 5 Shape dependent $P'_s$ for test case 1

<table>
<thead>
<tr>
<th>Method</th>
<th>$P'_s$</th>
<th>$M_{MC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>$1.47 \times 10^{-6}$</td>
<td>821,166,336</td>
</tr>
<tr>
<td>PCE</td>
<td>$1.8 \times 10^{-6}$</td>
<td>$10^8$</td>
</tr>
</tbody>
</table>

Fig. 9 Distribution of $P'_s$ with different pRNG states

magnitude when compared to the spherical CV, but require an increase in $M_{MC}$ for their computation. The generation of the PCE-determined value uses $10^8$ independent samples for probability estimation. However, due to the excessive computation cost, the Monte Carlo-based value is computed using 28,656 samples per spacecraft with an all-on-all analysis, i.e., the comparison of each sample with all of those generated for the other vehicle. This yields approximately $8.2 \times 10^8$ comparisons. Unfortunately, these comparisons are not independent, which may yield estimation bias and Eq. (39) does not necessarily yield an accurate assessment of solution confidence. However, in spite of the uncertainty in the baseline solution, this result does indicate that the PCE-based solution reflects the reduction in collision probability when using the MMS-based shape model for both spacecraft.

Since a direct comparison to a reliable baseline result is not tractable for this shape-dependent case, Fig. 9 illustrates the variability of 1,000 PCE-based solutions for $P'_s$. The line on each histogram represents the normal distribution with the given mean and standard deviation. Using the same procedure previously outlined for each estimate, this analysis considers different pseudo-random
number generator (pRNG) states in: (1) the creation of the $M_{MC}$ comparison samples, (2) generation of the $M_{PC} = 240$ PCE training samples, and (3) a combination of these two elements in the solution. Results indicate that the PCE solution may yield a bias in the collision probability estimate. However, this bias is not statistically significant when compared to the variance in the solution caused by the pRNG state when generating the $M_{MC}$ comparison samples. For a mean $P_0$ of $1.7 \times 10^{-6}$, Eq. 39 implies a standard deviation of $1.3 \times 10^{-7}$. The solution variance demonstrated in Fig. 9 agrees with this value to one digit.

Unlike the pure Monte Carlo methods used to generate a baseline solution, generation of the results presented in Fig. 9 is tractable given ten computers each with the same model GPU card considered in other parts of this study. Given these ten computers, runtimes for each of these three tests is less than five hours. It is also noted that, in some cases, the selection of the training samples failed to converge on a solution meeting the CS cross-validation requirement. For these cases, the PCE solution is discarded and replaced with one generated using 240 new training and cross-validation samples, regardless of potential accuracy of $P_0'$. This occurred for approximately 0.8% of the PCEs generated. As indicated by the discussion of runtimes in the next paragraph, cost of PCE regeneration ($\sim 20$ s) in these rare cases is small when compared to the comparison of state realizations.

Table 6 summarizes the estimated computation time for the CPU and GPU-based implement-
tation of the PCE-based techniques. Propagation of the training samples uses all eight cores of the
desktop computer used in this study. Although solving for the PCEs via CS may be parallelized
(e.g., the $X$ coordinate PCE is independent of the $Y$ coordinate PCE), such an implementation
is not reflected in the table. Additionally, the realization comparison times for the CPU imple-
mentation are also based on the use of all eight CPU cores. The time required to perform the
comparisons via a PCE includes the evaluation time and the computation of $m(\xi, \xi')$. Including
the shape-based estimation of the collision probability yields an increase in computation time for
both PCE implementations due to the change in $M_{MC}$. The GPU implementation yields a halving
of the computation time. The GPU used in this analysis is not state of the art at the time of
this study, and further reductions will likely result from updated hardware. With eight cores for
propagation, the GPU evaluation implementation, and CS-based generation of the PCE, the time
required to compute $P_s$ is less than one minute for the spherical CV. The shape-dependent CV
analysis requires less than three minutes.

C. Case 2: MMS1 and Debris Conjunction

The second case quantifies the collision probability for an MMS-like spacecraft with a piece of
orbital debris, i.e., an uncontrolled object. The spherical keep-out radius for this DEB object is 10
m, which yields $R = 70$ m. The shape-dependent collision probability estimation uses the conditions
in Eqs. (30)-(31) with the MMS-like shape model and $r = 10$ m. The generation of the PCE for
MMS1 for this case uses the same solution configuration of the previous case, but with solutions
generated in the interval $t_{CA,2} = \pm 300$ s (still with 30 s steps). Unlike the MMS-like trajectories,
there is no maneuver performed by the DEB object. Hence, only a least-squares-based estimation is
performed with $d = 7$. Generation of the least-squares based PCEs with a design accuracy of 0.1 m
in position requires 156 training samples and $p = 3$. As discussed previously, this analysis uses the
Brent-based optimization method to prevent tunneling.

Figure 10 describes the distribution of 100,000 propagated realizations of the relative position
states at $t_{CA,2}$ for test case 2. As indicated in the skewed distribution of relative $Y$ component, the
PDF describing the relative position is non-Gaussian, which indicates that the classic methods may
yield inaccurate results for this case as well. See the analysis in [6] for a comparison of solutions generated via a PCE and the analytic techniques for a similar non-Gaussian case. Additionally, as described in the next paragraph, this scenario includes a revisit between the two spacecraft during the time span of potential collision, i.e., the relative motion is nonlinear.

Figure 11 illustrates $P_s'$ when considering both the spherical and shape-dependent CVs. The period of no additional collision risk from approximately 10 to 35 s results from an increased separation in the PDFs as the MMS1 spacecraft passes through perigee and temporarily drops below the
<table>
<thead>
<tr>
<th>Task</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMS1 Training Sample Generation ((M_{PC} = 200))</td>
<td>16.98</td>
<td></td>
</tr>
<tr>
<td>DEB Training Sample Generation ((M_{PC} = 156))</td>
<td>53.16</td>
<td></td>
</tr>
<tr>
<td>MMS1 PCE Generation</td>
<td>2.65</td>
<td></td>
</tr>
<tr>
<td>DEB PCE Generation</td>
<td>1.11</td>
<td></td>
</tr>
<tr>
<td>Spherical CV Comparisons ((M_{MC} = 10^7))</td>
<td>31.36</td>
<td>11.75</td>
</tr>
<tr>
<td>Shape Model CV Comparisons ((M_{MC} = 10^7))</td>
<td>46.74</td>
<td>20.84</td>
</tr>
<tr>
<td>Total Computation Time (Spherical)</td>
<td>104.21</td>
<td>84.60</td>
</tr>
<tr>
<td>Total Computation Time (Shape)</td>
<td>119.59</td>
<td>94.74</td>
</tr>
</tbody>
</table>

The 3σ Monte Carlo range appears larger for the shape-dependent result because more samples are required to accurately estimate the smaller probability. For the shape model CV test, \(1.1 \times 10^6\) samples are required to yield one digit of accuracy based on Eq. (39). Like test case 1, more samples are generated to reduce uncertainty in the presented results. For a spherical CV, \(P_s' = 2 \times 10^{-3}\), which would raise an alarm for systems with a \(10^{-3}\) maneuver requirement. However, with the shape-dependent approximation, the estimated \(P_s'\) decreases by almost an order of magnitude to \(8.3 \times 10^{-4}\) and would pass such tests. However, this would not eliminate it from any watch lists with a \(10^{-5}\) or larger threshold.

Table 7 summarizes the estimated computation time for this test case, with the same assumption and time estimation methods from Table 6 used in its generation. The PCE implementation on the eight-core CPU requires approximately two minutes to perform the case 2 analysis with a shape-defined CV. The GPU implementation reduces the computation time and is dominated by the generation of the 156 training samples for the DEB object. When comparing the runtimes of the \(M_{MC}\) comparisons for this case versus the MMS1-MMS2 case with a spherical CV, there is a nominal increase in computation time. This can result from both the increased number of terms in the PCE for the DEB object or the need for Brent’s method to prevent tunneling. However, the increased computation time is small compared to the total computation time.
VI. Conclusions

Propagation of orbit state uncertainty via Polynomial Chaos provides a tractable means to estimate the probability of collision between two spacecraft after at least one executes a translation maneuver. These methods use a polynomial chaos expansion (PCE) as a surrogate to generate a state realization at a future point in time. A Monte Carlo-like quantification of collision probability is then performed using a sufficiently large set of these realizations. For the case of a maneuvered spacecraft with a probabilistic description of the execution errors, the high-degree PCE describing the position of the maneuvered spacecraft may be sparse. Such cases allow for the use of compressive sampling to generate the surrogate model with only hundreds of training samples, which yields a reduced computation time when compared to generating the same PCE via least-squares regression.

The PCE-based methods also provide a means for increased fidelity in the estimation of collision probability. The accuracy of the PCE-determined result equals that of a Monte Carlo-based solution with the same number of comparison samples. Implementation of the polynomial surrogate evaluations on a graphics processing unit reduces runtimes to on the order of minutes for a Monte Carlo-like analysis with at least $10^7$ comparison samples. The improved tractability allows for the use of collision volumes defined by a spacecraft shape model, which yields a smaller collision probability when compared to a solution based on a more conservative spherical keep-out radius.

VII. Acknowledgements

This research was partially funded by the NASA Goddard Space Flight Center and a.i. solutions. Some material is based upon work of the third author supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, under award number DE-SC0006402.

A sample MMS trajectory that served as the basis for MMS1 in the presented tests was provided by Geoffrey Wawrzyniak of a.i. solutions under contract to the National Aeronautics and Spacecraft Administration (NASA). The OMP solver included in SparseLab$^3$ served as the prototype used

---

$^3$ Available at [http://sparselab.stanford.edu/](http://sparselab.stanford.edu/)
in this work, and our Python-based tool was largely based on its implementation. The authors thank the developers of SparseLab for making their code available to the research community. The authors also thank Michael S. Werner (undergraduate student at the Colorado School of Mines) for his assistance in developing the first prototype of the GPU tool.

This work utilized the Janus supercomputer, which is supported by the National Science Foundation (award number CNS-0821794) and the University of Colorado Boulder. The Janus supercomputer is a joint effort of the University of Colorado Boulder, the University of Colorado Denver and the National Center for Atmospheric Research.

References


doi:10.2514/1.57599.


doi:10.1109/TIT.2006.871582.


doi:10.1109/MSP.2007.914731.


