Gaussian Processes for Improving Orbit Prediction Accuracy

Hao Peng[1], Xiaoli Bai[2,*]

Department of Mechanical and Aerospace Engineering,
Rutgers, The State University of New Jersey, NJ, 08854, USA

Abstract

A machine learning (ML) approach has been recently proposed to improve orbit prediction accuracy of resident space objects (RSOs) through learning from historical data. Previous results have shown that the ML approach can successfully improve the point estimation accuracy. This paper extends the ML approach by introducing Gaussian Processes (GPs) which can generate uncertainty information about its point estimate. Both the simulation environment and the publicly available RSO catalogs are used to test the advanced ML approach. Numerical results demonstrate that the trained GP model can effectively improve the orbit prediction accuracy and generate uncertainty boundaries with high performance. Discussions and insights are also presented during the investigation using real data, including suggestions on designing learning variables and the possible causes for some unsatisfying results.

Keywords: Orbit Prediction; Machine Learning; Gaussian Processes; Two-Line Element Catalog; Uncertainty Prediction.

1. Introduction

The amount of resident space objects (RSOs) are escalating and so does the collision possibility between RSOs [1]. There are many proposals for active space debris capturing or removing to control the population of RSOs and reduce the collision risk [2, 3, 4]. However, before any active debris removal project eventually comes true, one significant challenge for Space Situational Awareness (SSA) is to predict each RSO's orbit accurately. Current approaches for orbit prediction mostly use some physics-based analytical models. The accuracy of orbit prediction depends on a good knowledge of the environment information (such as earth gravity, atmospheric drag, and solar radiation pressure), the RSO's state and shape information, and the awareness of maneuvering events. Conventional approaches to improve orbit prediction accuracy include developing more accurate dynamic models and enhancing measurement techniques, but there are many theoretical and practical limitations to achieve high-accuracy physics-based models.

Recently, machine learning (ML) methods have arisen again and drawn great attention from diverse fields, including the aerospace research. Nesvold et al. have developed a model to evaluate the effectiveness of

*Corresponding author
[1]Postdoctoral Associate. Astro.H.Peng@gmail.com
[2]Assistant Professor. xiaoli.bai@rutgers.edu

several approaches to deflect the hazardous object [5]. Pérez et al. have proposed to used neural network model to improve accuracy of the SGP4 propagator [6]. Later, San-Juan et al. have compared a similar hybrid SGP4 propagator with a different statistical smoother [7]. Pérez and Bevilacqua have proposed a neural network-based approach for reducing the density error in empirical models which is critical for orbit prediction of low earth orbits (LEOs) [6]. However, this approach still focuses on providing better system parameter identification results for conventional physics-based analytical models. Since the advanced computational power has made it possible to utilize a large amount of data in an acceptable duration, data-driven models could be built to recover missing information that is hidden in the historical data.

In our previous studies [8, 9, 10], a framework of ML approach to improve orbit prediction accuracy of resident space objects (RSOs) has been developed and demonstrated to be effective. Through designing appropriate learning variables and using true orbit prediction error as target variables, the proposed ML approach can use a supervised learning algorithm to capture the relationship between the learning and target variables. Afterwards, the trained ML model can directly modify the predicted orbital state at future epochs to improve orbit prediction accuracy. As demonstrated in previous studies, both support vector machine (SVM) and artificial neural network (ANN) can achieve good performance. However, the standard versions of these two algorithms only generate the so-called "point estimate" of the target variable, meaning the output is a best estimate of the target value without information about how reliable the estimate is.

As an important enhancement, in this paper, the ML approach is further developed to provide the additional uncertainty information. In essence, the ML approach will generate both a guess of the orbit prediction error and a corresponding uncertainty. Since the overall ML-based orbit prediction has been designed in a modularized fashion [8], the new implementation only requires to replace the ML algorithm block. This modular design also makes it convenient to systematically compare different ML algorithms on the same problem, which is out of the scope of this paper and has been presented in a separate paper [11].

Specifically, Gaussian Processes (GPs) are explored to solve the supervised learning problem to improve orbit prediction accuracy. The GP model is a classical regression algorithm [12, Sec. 2.8] and there are many comprehensive introductions about it [13, p.304][14, p.515]. Generally speaking, GPs are Bayesian inference methods which assume Gaussian Process a prior over the underlying function between the learning and target variables. A Gaussian process is defined as a special stochastic process of a collection of an infinite number of variables such that any subset of the random variables has a multivariate normal distribution. For example, assuming a smooth target function f is given, a GP assumes each sample $\hat{f}_i$ is Gaussian (with respect to its input) and all the n samples $\{\hat{f}_i\}_{i=1}^n$ are multivariate Gaussian. Therefore, a new prediction f_* can be inferred from the new multivariate Gaussian of $\{\{\hat{f}_i\}_{i=1}^n, \hat{f}_*\}$ as a set of samples from f following the joint multivariate normal distribution. The computing complexity of training is $O(n^3)$ which makes large scale and high-dimensional problems challenging; Furthermore, the kernel function (determining the covariance of multivariate normal distribution) has significant effects on predictions but there is no general guidance rule on the optimal choices.

Gaussian Processes have been used in the control problem. The goal is to build a data-driven external dynamic model of the unknown system which leads to an effective control of the system [15, 16]. There are also many other applications in the aerospace field. Shang and Liu [17] have used the GP method to approximate the transfer cost of two-impulsive or Mars-gravity-assist transfers to main belt asteroids. They have found that the GP model offers good accuracy and is more efficient than directly solving the problem. Eerland et al. [18] have used GPs to model the probability distribution of a set of aircraft trajectories from historical measurement data. Yan et al. [19] have introduced GPs to model the time-dependent variance of the target variable and their experiments on real flight data achieved good performance. In order to deal with the input-dependent variance, Almosallam et al. [20, 21, 22] have proposed a new sparse GP algorithm to accommodate the input-dependent variances. Their method has been tested on predicting photometric redshifts and showed great capability to capture the varying noise.

The main contribution of this paper is that the recently developed ML approach has been extended to generate uncertainty information about its orbit prediction errors, which is critical for the real operation. Furthermore, the previously demonstrated capability of the proposed ML approach using SVM or ANN is also validated by the GP approach. The remaining part of the paper is organized as follows. In Sec. 2, the background is briefly reviewed, including the ML approach, the design of the learning and target variables, and the specific GP method used in this study. In Sec. 3, numerical results of the trained GP models are demonstrated in the simulation environment, including detailed analysis and insightful discussions. In Sec. 4, RSO catalogs are used to validate the ML approach and a few practical problems are investigated. In the last section, conclusions and future directions are presented.

2. Machine Learning Approach

In this section, the ML approach is briefly reviewed as well as a simulation environment developed to evaluate the approach. At the end, design of the learning and target variables is summarized, which is the same as our previous studies [9, 10].

2.1. Brief Summary

Detailed descriptions about the ML approach can be found in our previous publications [8, 9, 23, 10]. As a background, the ML approach is briefly reviewed in this subsection. As shown in Fig. 1, at the epoch t_i, the "estimated state" $\hat{\boldsymbol{X}}(t_i)$ does not overlap with the "true state" $\boldsymbol{X}_{\mathrm{T}}(t_i)$, therefore, the "predicted state" $\hat{\boldsymbol{X}}(t_j;t_i)$ at a future epoch t_j deviates from the $\boldsymbol{X}_{\mathrm{T}}(t_j)$ even further. The ML approach is designed to take a series of available inputs at t_i and then generate the "ML-modified state" $\hat{\boldsymbol{X}}_{\mathrm{ML}}(t_j;t_i)$, which is expected to be closer to $\boldsymbol{X}_{\mathrm{T}}(t_j)$ through learning the prediction errors.

Orbit prediction is generated by propagating the estimated state of an RSO using a particular dynamic model to future epochs. Because of the approximated dynamic models and the state estimated from noisy measurements, orbit prediction errors are usually inevitable. One advantage of the proposed ML approach

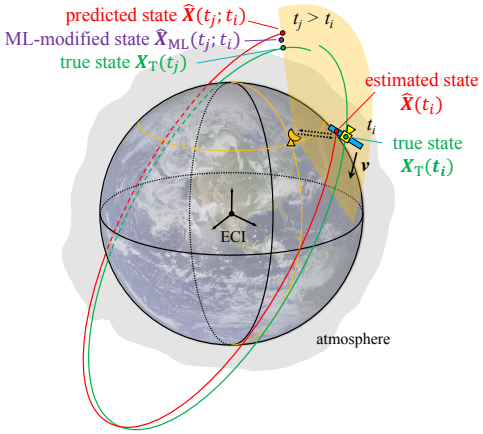


Figure 1: Illustration of the ML approach to modify the orbit prediction at a future epoch [8].

is that since the ML approach directly works on the future orbit predictions, it can be integrated with state-of-art dynamic models, orbit determination/prediction techniques, and conjuncture analysis frameworks.

The key step of the above process is to generate $\hat{\boldsymbol{X}}_{\mathrm{ML}}(t_j;t_i)$. This procedure is modeled as a supervised learning problem in the ML approach. In previous studies, we have explored the support vector machine (SVM) [8] and artificial neural network (ANN) [10], and the studies show good performance on capturing the underlying relationship between the learning and target variables in various circumstances. A not yet addressed question is the uncertainty information about the ML approach, which is required by many space operations, such as the conjuncture analysis. The above modification process is modeled as a supervised learning problem, meaning the ML methods will learn a function or mapping from labeled data. One "labeled data" consists of a set of learning variables and its corresponding target variables.

2.2. Design of Learning and Target Variables

In this subsection, the design of the learning and target variables is first presented before discussing the actual machine learning process. The basic design is based on the common feature of publicly available catalogs which usually have only limited information. For example, in the two-line element (TLE) catalog, one TLE set contains only the epoch, orbital parameters, drag coefficient, but nothing about the measurements and characteristic of RSOs. Other catalog-dependent designs will be presented in the following sections. We note that, according to our experience in previous studies, an advantage of the ML approach is that most algorithms are not sensitive to an irrelevant learning variables. In other words, including irrelevant or random variables will usually not affect the relationship between the learning and target variables too much. From this point of view, the ML approach allows flexibility to design learning variables.

Our design of the structure of one labeled data point is illustrated in Fig. 2. It is important that there exists a relationship between the designed learning and target variables in the dataset such that ML models can be trained to capture it. Due to the lack of an explicit relationship between the learning and target variables which is not explicitly captured by the dynamic model, our design procedure is based on a trial-and-error process.

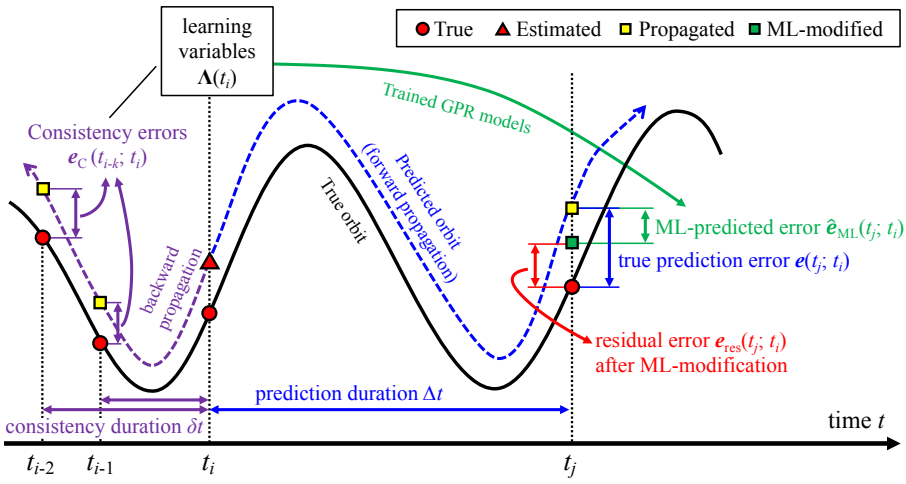


Figure 2: Illustration of the dataset structure for the ML approach to improve orbit prediction accuracy.

In this paper, both the simulation data and the practical data is used. They share the same design of the learning and target variables, as illustrated in Fig. 2. The design of the target variables is intuitive. Since we want to know the difference between any predicted state and its corresponding true state, the true orbit prediction error $\boldsymbol{e}_{\mathrm{T}}(t_j)$ is chosen as the target variable, defined as

$$\boldsymbol{e}_{\mathrm{T}}(t_j; t_i) = \boldsymbol{X}_{\mathrm{T}}(t_j) - \hat{\boldsymbol{X}}(t_j; t_i), \tag{1}$$

where we denote the true state at t_j as $\boldsymbol{X}_{\mathrm{T}}(t_j)$ and the predicted state at t_j as $\hat{\boldsymbol{X}}(t_j; t_i)$ which is based on the estimate $\hat{\boldsymbol{X}}(t_i)$ $(t_i < t_j)$. As shown in Fig. 2, at t_i, the estimated state $\hat{\boldsymbol{X}}(t_i)$ is forwardly propagated for a duration of Δt, resulting in the predicted state $\hat{\boldsymbol{X}}(t_j; t_i)$.

The goal of the ML approach is to predict the true orbit prediction error based on inputs. Therefore, when an ML model is well-trained, it will produce an approximation $\hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j; t_i)$ of the true orbit prediction error $\boldsymbol{e}_{\mathrm{T}}(t_j; t_i)$. Then, this approximation will be applied to the original prediction $\hat{\boldsymbol{X}}(t_j; t_i)$ to generate the ML-modified predicted state $\hat{\boldsymbol{X}}_{\mathrm{ML}}(t_j; t_i)$ as

$$\begin{aligned} \hat{\boldsymbol{X}}_{\mathrm{ML}}(t_j; t_i) &= \hat{\boldsymbol{X}}(t_j; t_i) + \hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j; t_i) \\ &= \boldsymbol{X}_{\mathrm{T}}(t_j) - \boldsymbol{e}_{\mathrm{T}}(t_j; t_i) + \hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j; t_i). \end{aligned} \tag{2}$$

where we have used Eq. (1). As revealed by Eq. (2), if $\hat{\boldsymbol{e}}_{\mathrm{ML}}$ can approximate $\boldsymbol{e}_{\mathrm{T}}$ precisely, the ML-modified prediction $\hat{\boldsymbol{X}}_{\mathrm{ML}}$ will be identical to the true state $\boldsymbol{X}_{\mathrm{T}}$, meaning that the orbit prediction error is zero after the ML-modification.

In the RSW frame centered at $\hat{\boldsymbol{X}}(t_j; t_i)$, the true error is expressed as ${}^{\mathrm{RSW}}\boldsymbol{e}_{\mathrm{T}}(t_j; t_i) = \{e_x, e_y, e_z, e_{vx}, e_{vy}, e_{vz}\}^{\mathsf{T}}$, where the x-axis is along the radial direction, the y-axis (along-track direction) is perpendicular to the x-axis in the orbital plane and points to the inertial velocity direction, and the z-axis (cross-track direction) is along the angular momentum direction [24]. Totally six GP models will be trained for each component e_ξ of the true error ${}^{\mathrm{RSW}}\boldsymbol{e}_{\mathrm{T}}(t_j; t_i)$, where ξ is taken from $\{x, y, z, vx, vy, vz\}$. We emphasize that the use of the

RSW frame which is a body frame can isolate the orbit prediction error from the inertial state of the RSO, so that the description of the errors only depend on the RSO itself.

The design of the learning variables is more dependent on the available information or data source. In practice, a satellite owner or operator with tracking capabilities will have more information. We note that the different available data only requires to modify the learning variables in the proposed ML approach, which will certainly have an effect on the final performance. If the performance is unacceptable, it would indicate that the currently available data is not adequate to further improve the orbit prediction accuracy through the proposed ML approach.

As shown in Fig. 2, the current estimated state $\hat{\boldsymbol{X}}(t_i)$ is backwardly propagated to its past epochs t_{i-1} and t_{i-2} to generate the states $\hat{\boldsymbol{X}}(t_{i-1};t_i)$ and $\hat{\boldsymbol{X}}(t_{i-2};t_i)$. At t_{i-1}, the propagated states are compared with the corresponding true states to generate the consistency errors $\boldsymbol{e}_{\mathrm{c}}(t_{i-1};t_i) = \hat{\boldsymbol{X}}(t_{i-1}) - \hat{\boldsymbol{X}}(t_{i-1};t_i)$. In the same way, one could get $\boldsymbol{e}_{\mathrm{c}}(t_{i-2},t_i)$ and consistency errors with even earlier estimates. These errors are expressed as Cartesian coordinates in the RSW frame as ${}^{\mathsf{RSW}}\boldsymbol{e}_{\mathrm{c}}$. The drag coefficient is also assumed to be related to the orbit prediction error and included as a learning variable. In summary, suppose k previous TLE sets are used to calculate consistency errors, the set of leaning variables $\boldsymbol{\lambda}$ for the ML approach at the epoch t_i consists of:

- Prediction duration $\Delta t = t_j - t_i$ to the future epoch t_j;
- Consistency errors $\{\dots, \delta t_k, \boldsymbol{e}_{\mathrm{c}}(t_k;t_i), \dots\}$, where $t_k < t_i$, $\delta t_k = t_i - t_k$ is the backward propagation duration;
- Drag coefficients and the consistency deviations among with all the previous estimates.

In the above design, some parameters depend on the feature of the catalog used, including the maximum prediction duration $\Delta t_{\max}$, the number k of pairs of consistent error, and the specific drag coefficients which represent the drag information.

2.3. Heteroscedastic Sparse Gaussian Processes

Based on the designed learning and target variables for the ML approach, the Gaussian processes method [12] is briefly introduced in this section. Then one specific sparse GP method capable of learning input-dependent observation noise in the system is introduced [21, 20].

The GP method can provide uncertainty information about its prediction. However, the information comes with a price. The time complexity to train the GP model is $O(n^3)$ due to the inversion of the $n \times n$ covariance matrix at each iteration [21, p.14], where n is the number of data points in the training data. This can be computationally infeasible when there are tens of thousands of training data. Much research has been conducted to reduce the computational burden of GP method while preserving its performance on generating posterior distribution. In this paper, we tackle the challenge through using a particular sparse GP method, which belongs to a wide range of sparse GP methods [25].

Another shortcoming of GPs for the application to reduce orbit prediction accuracy is that the uncertainty

of the ML-predicted error is expected to vary with respect to the prediction duration Δt, but the standard GPs assume homoscedastic noise that is constant for all Δt. We briefly review a modified sparse GP method that can handle the heteroscedastic noise, which will be used in this paper. More details about this method can be found in the works of Ibrahim A. Almosallam et al. [20, 21].

2.3.1. Modeling

All the learning variables in the training data are collected as the matrix

$$\Lambda = \begin{bmatrix} \boldsymbol{\lambda}_1 & \cdots & \boldsymbol{\lambda}_i & \cdots & \boldsymbol{\lambda}_n \end{bmatrix} \in \mathbb{R}^{d\times n}, \tag{3}$$

where d is the dimension of the learning variable vector $\boldsymbol{\lambda}_i$, n is the number of data points in the training data, and the subscript indicates the i-th data point. Denote one component of the target true error $\boldsymbol{e}_{\mathrm{T},i}$ as τ_i, then, all the target variables are represented by a vector

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \cdots & \tau_i & \cdots & \tau_n \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^n. \tag{4}$$

First, each vector of learning variables $\boldsymbol{\lambda}_i$ is projected onto the m-dimensional feature space as $\boldsymbol{\phi}(\boldsymbol{\lambda}_i)$ through a nonlinear transformation

$$\begin{aligned} &\boldsymbol{\phi} : \mathbb{R}^d \longmapsto \mathbb{R}^m \\ &\boldsymbol{\lambda}_i \longmapsto \boldsymbol{\phi}(\boldsymbol{\lambda}_i) = \begin{bmatrix} \phi_1(\boldsymbol{\lambda}_i) & \cdots & \phi_i(\boldsymbol{\lambda}_i) & \cdots & \phi_m(\boldsymbol{\lambda}_i) \end{bmatrix}^{\mathsf{T}}. \end{aligned} \tag{5}$$

In this paper, ϕ_i is chosen to be the radial basis function

$$\phi_j(\boldsymbol{\lambda}_i) = \exp\left[-\frac{1}{2}(\boldsymbol{\lambda}_i - \boldsymbol{p}_j)^{\mathsf{T}}\Gamma_j^{\mathsf{T}}\Gamma_j(\boldsymbol{\lambda}_i - \boldsymbol{p}_j)\right], \tag{6}$$

where $\boldsymbol{p}_j \in \mathbb{R}^d$ is the j-th center vector and $\Gamma_j = \mathrm{diag}(\gamma_{j1}, \ldots, \gamma_{jm}) \in \mathbb{R}^{m\times m}$ is the to-be designed precision matrix associated with the j-th basis function. In this way, the learning variable matrix Λ in the training set is transformed to

$$\Phi = \boldsymbol{\phi}(\Lambda) = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{\lambda}_1) & \cdots & \boldsymbol{\phi}(\boldsymbol{\lambda}_i) & \cdots & \boldsymbol{\phi}(\boldsymbol{\lambda}_n) \end{bmatrix} = \begin{bmatrix} \phi_1(\boldsymbol{\lambda}_1) & \cdots & \phi_1(\boldsymbol{\lambda}_n) \\ \vdots & \ddots & \vdots \\ \phi_m(\boldsymbol{\lambda}_1) & \cdots & \phi_m(\boldsymbol{\lambda}_n) \end{bmatrix}_{m\times n} \tag{7}$$

Then, the observed target variable τ_i is modeled as a linear combination of ϕ_i in the feature space as

$$\tau_i = \boldsymbol{\phi}(\boldsymbol{\lambda}_i)^{\mathsf{T}}\boldsymbol{\omega} + \varepsilon_i, \tag{8}$$

where $\boldsymbol{\omega} \in \mathbb{R}^m$ is a vector of weights of the linear model and ε_i is the additive noise describing the intrinsic uncertainty of the data.

In the classical GPs, all the noise ε_i is assumed to follow a same independent identically distributed Gaussian distribution with zero mean and a constant variance of β^{-1}. In this paper, the heteroscedastic noise which depends on the learning variables is modeled by introducing a varying β_i for each data point $\boldsymbol{\lambda}_i$, i.e., $p(\varepsilon_i) = \mathcal{N}(0, \beta_i^{-1})$. Specifically, the precision parameter (inversion of the variance) β is designed to be the exponential of a linear combination of the transformed learning variable vector $\boldsymbol{\phi}(\boldsymbol{\lambda}_i)$

$$\beta_i = \exp\left[\boldsymbol{\phi}(\boldsymbol{\lambda}_i)^{\mathsf{T}}\boldsymbol{u} + b)\right], \tag{9}$$

$$\tag{10}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ are the coefficients and the prior of $\boldsymbol{u}$ is assumed to be $p(\boldsymbol{u}) = \mathcal{N}(0,\, N^{-1})$ with $N = \mathrm{diag}(\eta_1, \ldots, \eta_m)$.

The priori of the weight $\boldsymbol{\omega}$ is assumed to follow another multivariate normal distribution

$$p(\boldsymbol{\omega}) = \mathcal{N}(0,\, A^{-1}), \tag{11}$$

with $A = \mathrm{diag}(\alpha_1, \ldots, \alpha_m)$. For the basis function model in Eq. (8), the likelihood function of $\boldsymbol{\tau}$ is given by

$$p(\boldsymbol{\tau}|\Phi, \boldsymbol{\omega}) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\phi}(\lambda_i)^{\mathsf{T}}\boldsymbol{\omega}, \beta_i^{-1}) = \mathcal{N}(\Phi^{\mathsf{T}}\boldsymbol{\omega},\, B^{-1}). \tag{12}$$

with $B = \mathrm{diag}(\beta_1, \ldots, \beta_n)$. Then, applying the Bayes rule, the posterior distribution of the weights $\boldsymbol{\omega}$ is obtained as

$$p(\boldsymbol{\omega}|\Phi, \boldsymbol{\tau}) = \mathcal{N}\left(\bar{\boldsymbol{\omega}},\, \Sigma^{-1}\right), \tag{13}$$

where $\bar{\boldsymbol{\omega}} = \Sigma^{-1}\Phi B\boldsymbol{\tau} \in \mathbb{R}^{m\times 1}$ and $\Sigma = \Phi B\Phi^{\mathsf{T}} + A \in \mathbb{R}^{m\times m}$. More detailed derivations and discussions are referred to Almosallam's thesis [21], publication [20, p.3] and Bishop's book [13].

2.3.2. Training

The GP model is trained by maximizing the log marginal likelihood

$$\begin{aligned}\log p(\boldsymbol{\tau}|\Phi, \theta) = &-\frac{1}{2}(\Phi\bar{\boldsymbol{\omega}} - \boldsymbol{\tau})^{\mathsf{T}}B(\Phi\bar{\boldsymbol{\omega}} - \boldsymbol{\tau}) + \frac{1}{2}\log|B| - \frac{n}{2}\log 2\pi \\ &-\frac{1}{2}\bar{\omega}^{\mathsf{T}}A\bar{\omega} + \frac{1}{2}\log|A| - \frac{1}{2}\log|\Sigma| \\ &-\frac{1}{2}\boldsymbol{u}^{\mathsf{T}}N\boldsymbol{u} + \frac{1}{2}\log|N| - \frac{m}{2}\log 2\pi,\end{aligned} \tag{14}$$

where all the hyperparameters have been collected as the vector $\boldsymbol{\theta}$. The hyperparameter includes (assuming $i = 1, \ldots, n$ and $j = 1, \ldots, m$):

- the center vectors $\boldsymbol{p}_j$ of each basis function $\boldsymbol{\phi}_j$;

- the diagonal elements γ_{ji} of the precision metrics of each basis function $\boldsymbol{\phi}_j$;
- the parameters α_j of the a priori distribution of the weights $\boldsymbol{\omega}$;
- the coefficients u_j and b of the heteroscedastic precision β_i;
- the parameters η_j of the a priori distribution of the coefficients u_j.

Using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm, the hyperparameter $\boldsymbol{\theta}$ is iteratively solved until a local optimum is reached or the performance on the validation data cannot be further reduced. We note that $\boldsymbol{p}_j$ are initialized through a principal components analysis (PCA) process by choosing the first m characteristic vectors with a uniformly distributed perturbation. This introduces randomness to the optimization process, leading to different solutions. Different random seeds will be tested in the next section.

2.3.3. Prediction

After the GP model is well trained, given a new set of learning variables $\boldsymbol{\lambda}_*$, the output τ_*, which is one of the six components of the ML-predicted orbit prediction error $\hat{\boldsymbol{e}}_{\mathrm{ML}}$, follows the distribution

$$p(\tau_*|\boldsymbol{\phi}(\boldsymbol{\lambda}_*), \Lambda, \boldsymbol{\tau}, \boldsymbol{\theta}) = \mathcal{N}(\mu_*, \sigma_*^2), \tag{15}$$

with

$$\mu_* = \boldsymbol{\phi}(\boldsymbol{\lambda}_*)^{\mathsf{T}} \Sigma^{-1} \Phi^{\mathsf{T}} B \boldsymbol{\tau}, \tag{16}$$

$$\sigma_*^2 = \nu_* + \beta_*^{-1}, \tag{17}$$

where $\nu_* = \boldsymbol{\phi}(\boldsymbol{\lambda}_*)^{\mathsf{T}} \Sigma^{-1} \boldsymbol{\phi}(\boldsymbol{\lambda}_*)$ and $\beta_* = \exp\left[\boldsymbol{\phi}(\boldsymbol{\lambda}_*)^{\mathsf{T}} \boldsymbol{u} + b\right]$.

Eq. (15) reveals that, instead of a single best-guessed value obtained from other regression algorithms (such as SVM [8] or ANN [10]), the ML-predicted τ_* solved by GP follows a Gaussian posterior distribution with a mean of μ_* and variance of σ_*^2. An intuitive interpretation of the predicted distribution Eq. (15) is that the trained GP model characterizes the confidence of the trained GP model on its mean output μ_*, based on the information hidden in the training data. For example, if the testing data $\boldsymbol{\lambda}_*$ is very different from any training data, the output τ_* could be meaningless and the GP model's confidence on τ_* should be low, i.e., the uncertainty σ_* should be large.

Additionally, as shown in Eq. (17), the uncertainty σ_*^2 of the ML-predicted orbit prediction error τ_* consists of two parts. Theoretically, ν_* represents the model uncertainty and should reduce as the training data size increases because the model becomes more confident about ν_* with a larger number of nearby samples; β_*^{-1} represents the noise uncertainty depending on $\boldsymbol{\phi}(\boldsymbol{x}_i)$ in the feature space and should reduce as the training data contain fewer observation errors. However, in practice, the effects of inadequate data and large noise are not easy to distinguish because they are learned from data together through training algorithms.

3. Performance in Simulation Environment

In our previous studies, a simulation environment has been developed to study the capability [8, 9] and some limitations [10] of the proposed ML approach. A validation study using the TLE catalog [23] shows that the results in the simulation environment can also be observed using real data. The advantage of the simulation environment is that it gives us full control over all the models, which is critical to theoretical studies. The results in the simulation environment are presented in this section, including: the performance of the means and uncertainties of the trained GP models on all the components of the orbit prediction error; the effects of the number of basis functions; and the capability of the ML approach to handle different magnitudes of the measurement noise. Along with the demonstration of numerical results, analysis and insightful discussions about the ML approach are presented.

3.1. Simulation Setups

For self-completeness, the simulation environment is briefly reviewed here first, which is consistent to previous publications. The specific setups of the experiments are summarized for reproduction purpose.

Fundamentally, two dynamic models have been used, as summarized in Table 1. The first model, with all the force modeled as accurately as possible, is used as the truth dynamic model in the simulation environment. The RSO is propagated using the truth model to provide the true orbit for the measurement process which uses ground radar stations. Three ground radar stations [8] are simulated using a topocentric frame with limited field of view, generating discrete azimuth α, elevation η, and range ρ when the target RSO is visible. Then, being a part of the simulation, random measurement noise σ_α, σ_η, and σ_ρ will be added to the radar station outputs. At last, consecutive measurements are organized as observation tracks for the next procedure.

The second model, with deliberately reduced-accuracy force models, is used as the "assumed model" for the orbit determination (OD) and orbit prediction (OP) processes. A batch least square estimator is used during the OD process, which takes all available tracks in the past 12 hours as input and generates the estimated orbit state $\hat{\boldsymbol{X}}$ and drag coefficient parameter $\hat{C}_d$. Then the OP is carried out using the same assumed dynamic model. The prediction error is the difference between the predicted state and the recorded true state at the same epoch, represented in a proper coordinate frame. The maximum prediction duration $\Delta t_{\max}$ is set as 7 days (one week) as a compromise between practical requirements and computational requirements.

Same as the previous studies [8, 10], an RSO in a Sun-synchronous orbit (SSO) is simulated based on ENVISAT, whose parameters are summarized in Table 2. During the generation of the true orbit, a spherical RSO with a constant area-to-mass ratio of 0.00195 is assumed, and the drag coefficient $C_d = 2.2$ and a single-parameter reflection coefficient $C_r = 1.25$ are assumed to be constant. The orbit propagation, measurement, estimation and prediction in the simulation environment are all implemented with the verified open-source Java library Orekit [26].

Table 1: Setups of the truth and assumed models [10].

Parameters	Truth model	Assumed model
Earth Shape	WGS84	WGS84
Harmonics Gravity Field	40×40	10×10
Third-Body Purterbation	Sun + Solar Planets + Pluto + the Moon	Sun + Jupiter + the Moon
Atomosphere Model	DTM2000	NRLMSISE-00
Solar Activity	MSAFE	$(F_{10.7}, K_p) = (150.0, 3.0)$

Table 2: The simulated RSO based on ENVISAT [10].

Parameters	Values
NORAD ID	27386
Orbit	SSO
Launch Date	1 March 2002
Altitude	∼796 km
Period	∼100 minutes
Weight	∼8211 kg
Area-to-mass Ratio	∼0.00195
Inclination	∼98.54 deg
Eccentricity	∼0.001165

Additionally, we design to use five previous estimates to generate all the consistency errors $\boldsymbol{e}_{\mathrm{c}}$ in the vector $\boldsymbol{\lambda}(t_i)$ of the learning variables (see Sec. 2.2), with the following constraint,

$$\begin{aligned} t_0 - t_{-1} &> 0 \\ t_{-1} - t_{-2} &> 0.5 \text{ day} \\ t_{-2} - t_{-3} &> 1 \text{ day} \\ t_{-l+1} - t_{-l} &> 2 \text{ days} \qquad l = 4, \ldots, 5, \end{aligned} \tag{18}$$

where t_0 denotes the current epoch and t_{-l} $(l = 1, \ldots, 5)$ denotes the epoch of the l-th previous estimate. This choice of $\boldsymbol{e}_{\mathrm{c}}$ will provide a backward coverage of roughly 7 days.

After applying the constraints and collecting data from the simulation environment, the training data is collected from the data in days 1–27; the validation data in days 28; and the testing data in days 29–35. For each case, five random seeds $\{42, 401, 953, 744, 616\}$ are tested to train GP models (set by `rng` in MATLAB). Training of the GP model is terminated when the optimization solver converges with a default tolerance of 10^{-6} or when the performance on the validation data stops from improving after 15 attempts.

The number of basis functions m is determined by varying m and observing the performance of the trained GP models on the validation data, seeking for a better performance with an acceptable computational burden. We note that the random initialization will also affect the performance. We emphasize that the performance on the testing data should not be used to tune this parameter, because the testing data should remain unknown in the design so that it can be used to evaluate the generalization capability of the trained models on new data. In this paper, m has been increased from 5 to 60 with a step of 5 and tested on the

validation data using the five random seeds. At last, it is determined to use 40 basis functions, because it can provide the best performance on the validation data of the along-track error e_y that has the largest magnitude among the position components, whilst keeping the computational burden at an acceptable level. The performance of the chosen m on the validation data of other components also belongs to the optimal ones. We note that the same m has been used for all the six components of $\boldsymbol{e}_{\mathrm{T}}$ for simplicity because the goal of this paper is to demonstrate the capability, but certainly different parameters can be used for each component if one seeks the best performance.

3.2. Performance on Means of Target Variables

After the six GP models for all the six components have been trained, the ML-predicted error $\hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j;t_i)$ can be generated. As explained before, in practice, $\hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j;t_i)$ will never equal to $\boldsymbol{e}_{\mathrm{T}}(t_j;t_i)$, resulting the residual error $\hat{\boldsymbol{e}}_{\mathrm{res}}(t_j;t_i) = \boldsymbol{e}_{\mathrm{T}}(t_j;t_i) - \hat{\boldsymbol{e}}_{\mathrm{ML}}(t_j;t_i)$ to be nonzero. The statistical properties of $\hat{\boldsymbol{e}}_{\mathrm{res}}(t_j;t_i)$ will be analyzed to evaluate the performance of the trained GP models.

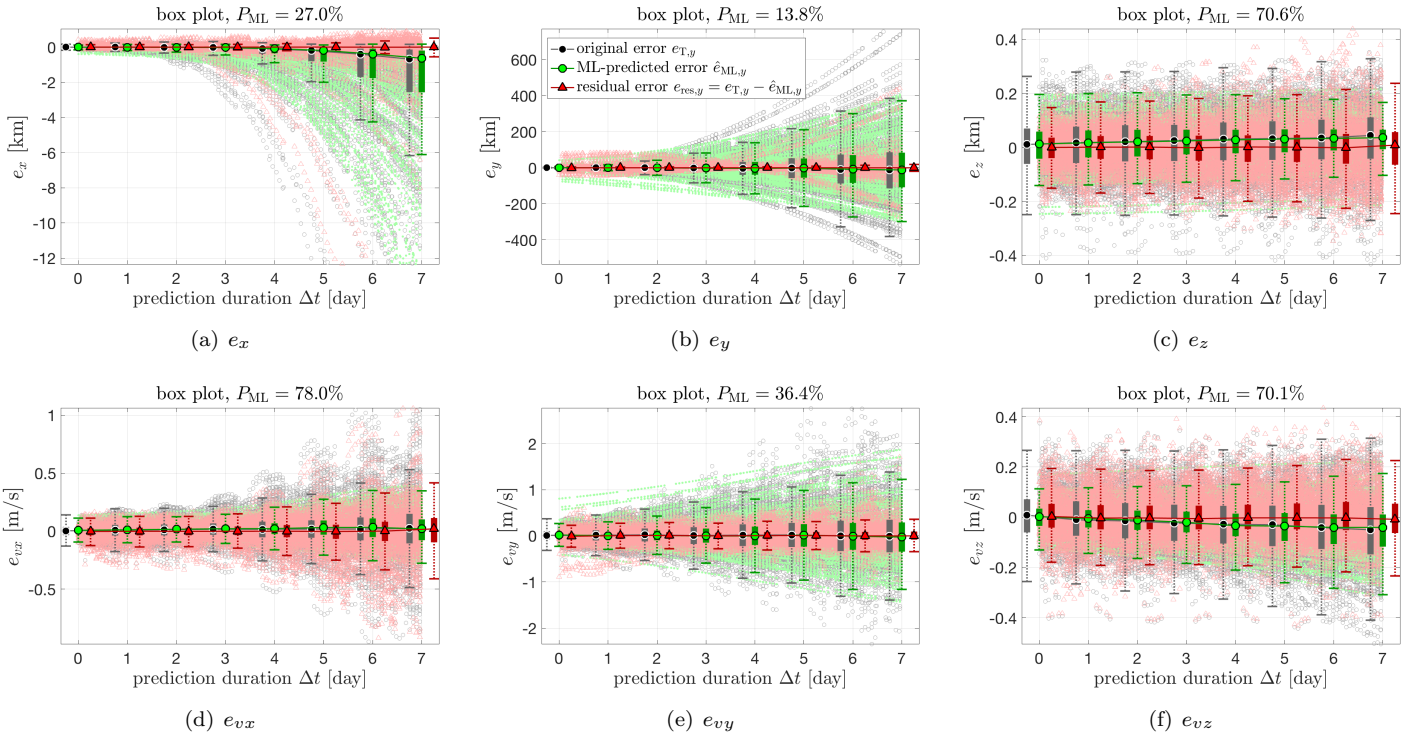
The metric P_{ML} has been defined to quantify the performance of the trained SVM and ANN models [8, 10], which is formulated as

$$P_{\mathrm{ML}}(e_\xi) = 100\% \cdot \frac{\sum |e_{\mathrm{T},\xi} - \hat{e}_{\mathrm{ML},\xi}|}{\sum |e_{\mathrm{T},\xi}|}, \tag{19}$$

where the summation is taken over the whole input data and ξ is taken from $\{x, y, z, vx, vy, vz\}$ for each component. This metric represents the percentage of the error that remains after the ML-modification. In this paper, P_{ML} is also used to evaluate the performance of trained GP models, which actually measures the mean μ_* (or expectation) in Eq. (16). The evaluation of the uncertainty σ_*, which has not been dealt with in previous studies, will be presented in the next subsection. Before discussing the uncertainty, it is helpful to first investigate the performance on the means of the ML-predicted error.

The boxplot in Fig. 3 demonstrates the performance of the trained GP on the testing data, for which the results having the smallest metrics P_{ML} among five random experiments are shown. In each panel, the horizontal axis represents the prediction duration Δt; the vertical axis shows true error $e_{\mathrm{T},\xi}$, ML-predicted error $\hat{e}_{\mathrm{ML},\xi}$, and residual errors $\hat{e}_{\mathrm{res},\xi}$ for the component ξ. The data is binned per day for each boxplot. The boxplots are standard: the center point represents the median (second quartile q_2) of each group of binned data; the bottom of the solid box represents the first quartile q_1; the top of the solid box represents the third quartile q_3; the top and botton bars of the dashed line represent the minimum and maximum of the binned data without outliers beyond $[q_2 - 1.5 \cdot \delta q, q_2 + 1.5 \cdot \delta q]$, where $\delta q = q_3 - q_1$ is the interquartile. The performance metrics $P_{\mathrm{ML}}(e_\xi)$ of the whole training data are shown above each plot. For clarity, the three sets of boxplot for $e_{\mathrm{T},\xi}$, $\hat{e}_{\mathrm{ML},\xi}$, and $\hat{e}_{\mathrm{res},\xi}$ have been slightly displaced along the horizontal axis to avoid overlapping. Additionally, at the background, scattering plots of the three kinds of errors are provided.

In Fig. 3, first we focus on the along-track position error e_y, since it has the largest magnitude compared to e_x and e_z. The metrics $P_{\mathrm{ML}}(e_y)$ is 13.8%, indicating that averagely 86.2% of errors have been compensated for. As observed, for most bins, especially the last several ones, the residual error has been significantly



(a) e_x (b) e_y (c) e_z

(d) e_{vx} (e) e_{vy} (f) e_{vz}

Figure 3: Performance on the testing data (days 29–42) using 40 basis functions.

condensed around zero. Considering that the learning variables do not directly relate to whether e_y increases or decreases with Δt, the trained GP model has automatically learned to detect the pattern from the training data and applied it for new predictions successfully. This achievement cannot be obtained by simply applying a fitting model of Δt. Although it is possible to utilize all learning variables in a simple fitting model, one needs to carefully design the fitting functions, which is an empirical and time-consuming procedure.

As can be seen, the residual errors of other components are also most concentrated around zero, closer than the original true errors. For e_x, the background scattering plot shows that occasionally $\hat{e}_{\mathrm{ML},x}$ could be largely deviated from $e_{\mathrm{T},x}$, but the boxplot is not severely effected, which means these can be treated as outliers. Detailed observations of e_z, e_{vx}, e_{vy}, and e_{vz} reveal that the ML-predicted errors $\hat{e}_{\mathrm{ML}}$ (green dots) appear to be almost linear functions of Δt, while the original errors (black circles) have obviously more fluctuations. Considering that their metrics P_{ML} are all smaller than 100% and that the boxplots of residual errors show both the means and interquartiles are reduced, it could be concluded that the trained GP models have successfully captured the general patterns and can compensate for a large amount of the errors.

We note that in the previous studies using SVM [9] and ANN [10], better performance on x- and y-axis components has been reported, due to a different design of learning variables which have more information. But the performance on z-axis is severely overfitted due to the more informative learning variables. As mentioned earlier, we use this set of learning variables in this paper with the goal to be consistent with the real data that will be presented in the next section.

Together with results obtained using SVM and ANN, we have validated that the ML approach can indeed recover the relationship between the learning and target variables, by purely learning from the historical

data, although such relationship may be only statistically correct.

3.3. Performance on Uncertainties of Target Variables

Compared with SVM and ANN methods, a critical advantage of GP method is that it can generate uncertainty information about its outputs $\hat{\boldsymbol{e}}_{\mathrm{ML}}$. As revealed by Eq. (15), for each input vector of learning variables $\boldsymbol{\lambda}$, the trained GP model will generate a predicted $\hat{e}_{\mathrm{ML}}$ as well as the uncertainty σ_{ML} about the prediction. If the target true error $\boldsymbol{e}_{\mathrm{T}}$ satisfies GP method's assumptions, it should follow a Gaussian distribution with the mean of $\hat{e}_{\mathrm{ML}}$ and the variance of σ_{ML}. We note that in practice it is difficult to verify this condition a priori. However, the following results will show that the trained GP models are effective. Here, we consider three uncertainty boundaries, 1σ, 2σ, and 3σ-boundaries (for simplicity, the subscription "$_{\mathrm{ML}}$" will be dropped when possible).

Since each testing data point $(\boldsymbol{\lambda}_i, e_\xi)$ has its own corresponding uncertainty σ_i, all results cannot be shown in a single plot. Figure 4 shows four examples of the variation of the uncertainty for each error component with respect to the prediction duration Δt, which are extracted from the testing data corresponding to those in Fig. 3. For each example, the true orbit prediction errors $e_{\mathrm{T},\xi}(t_j;t_i)$ and the ML-predicted errors $\hat{e}_{\mathrm{ML},\xi}(t_j;t_i)$ are based on the same estimate $\hat{\boldsymbol{X}}(t_i)$, so we can connect all the uncertainties in each example to observe the variation of the uncertainty with respect to Δt. We note that this is only feasible for a set of predictions based on the same estimate. In each subplot, the black circles represent $e_{\mathrm{T},\xi}$; the green dots represent $\hat{e}_{\mathrm{ML},\xi}$; and the blue boundaries from dark to light represent the $1\sigma_\xi$, $2\sigma_\xi$, $3\sigma_\xi$-boundaries which are centered at and symmetric to $\hat{e}_{\mathrm{ML},\xi}$.

As can be observed in Fig. 4, most of the true errors (black circles) fall inside of the uncertainty boundaries; the widths of the boundaries increase slightly with Δt; a few true errors fall outside of 3σ boundary, mostly when Δt is large. For all the components, the trained GP models not only accurately captured the trend of the true error, but also captured the growth of the fluctuation and reflected it in the uncertainty of $\hat{e}_{\mathrm{ML},\xi}$. For e_y, the trained GP model appears overconfident sometimes when Δt is large, although the performance is acceptable considering the large magnitude of $e_{\mathrm{T},y}$. We note that the four examples show different variation patterns with respect to Δt and the trained GP models have automatically learned the various patterns, which cannot be achieved by a simple nonlinear fitting of Δt.

We emphasize that although the increasing of σ_ξ coincides with the fact that the orbit estimation uncertainty grows during the propagation, here the uncertainty generated by the GP model purely depends on the training data and relies on the density and noise of sampling. So the GP model's uncertainty output is different from the conventional estimation uncertainty. Further research will study the relationship and the mechanism to integrate them.

In order to evaluate the overall performance on the uncertainty prediction, another metric $Q_k(e_\xi)$ is defined to quantify the percentage of the testing data points whose true error $e_{\mathrm{T},\xi}$ falls within the boundary

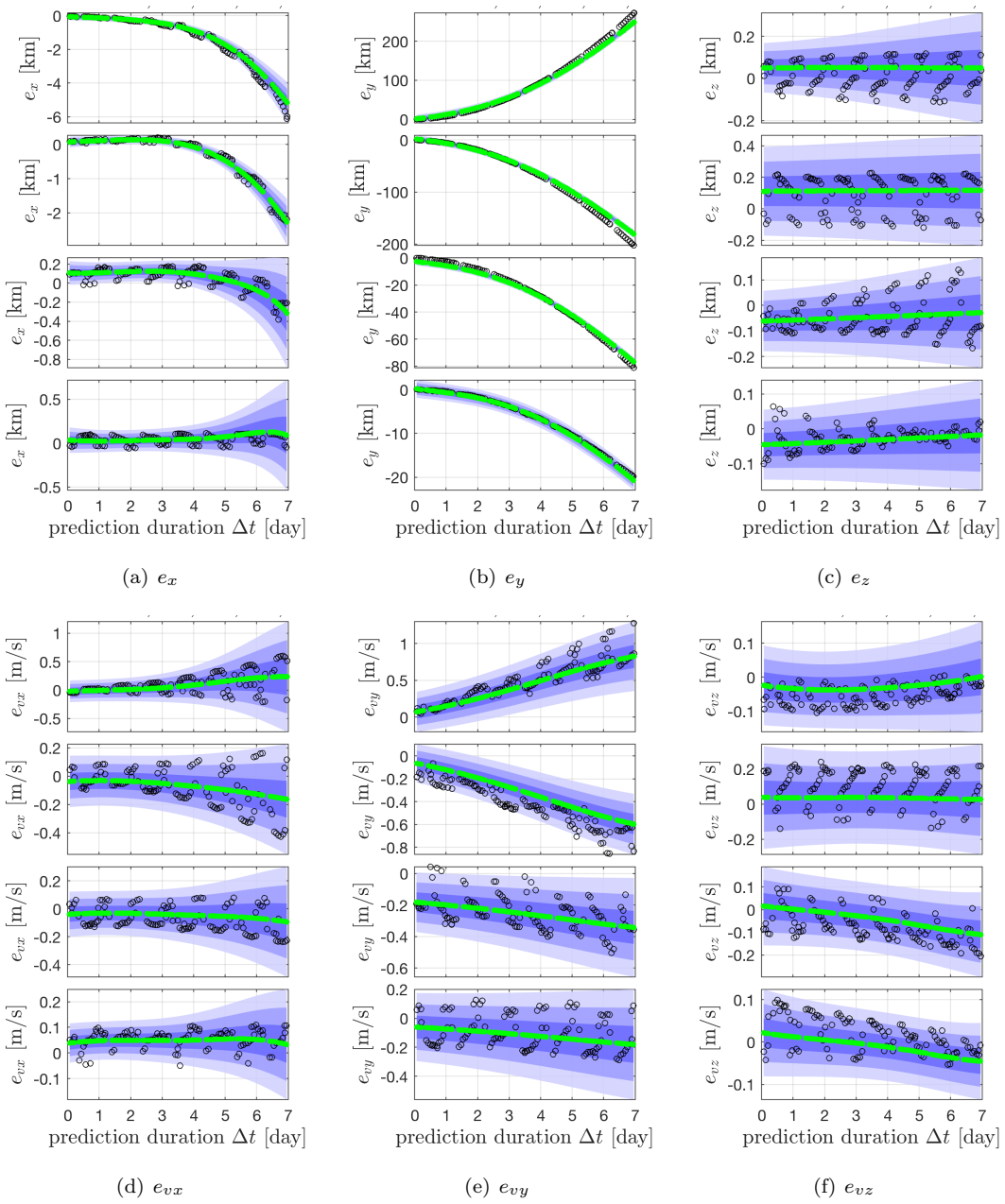


Figure 4: Four examples of the ML-predicted uncertainties $k\sigma$-boundaries ($k = 1, 2, 3$) around the mean of each error component $\hat{e}_{\mathrm{ML},\xi}$. The green dots represent ML-predicted value, the black circles represent the true value, and the blue ribbons represent the uncertainty boundaries.

of $\hat{e}_{\mathrm{ML},\xi} \pm k\sigma_\xi$, expresses as

$$Q_k(e_\xi) = 100\% \cdot \frac{1}{2n} \sum_{i=1}^{n} \Big(1 + \mathrm{sgn}\left(k\sigma_\xi - |e_{\mathrm{T},i,\xi} - \hat{e}_{\mathrm{ML},i,\xi}|\right)\Big), \tag{20}$$

where $k > 0$ is the multiple of the uncertainty σ_ξ, n is the size of input dataset, and $\mathrm{sgn}(\cdot)$ is the sign function. The minimum value of $Q_k(e_\xi)$ is zero. If all the errors are captured within the $k\sigma$-boundary, $Q_k(e_\xi) = 1$.

The results for all components of testing data are demonstrated in Fig. 5. The horizontal axis represents k, the multiples of σ_ξ. The vertical axis represents $Q_k(e_\xi)$, and the circles represent Q_k using different random seeds. For all the components except e_y, almost all the testing data (>99%) fall in the $3\sigma_\xi$-boundary and the

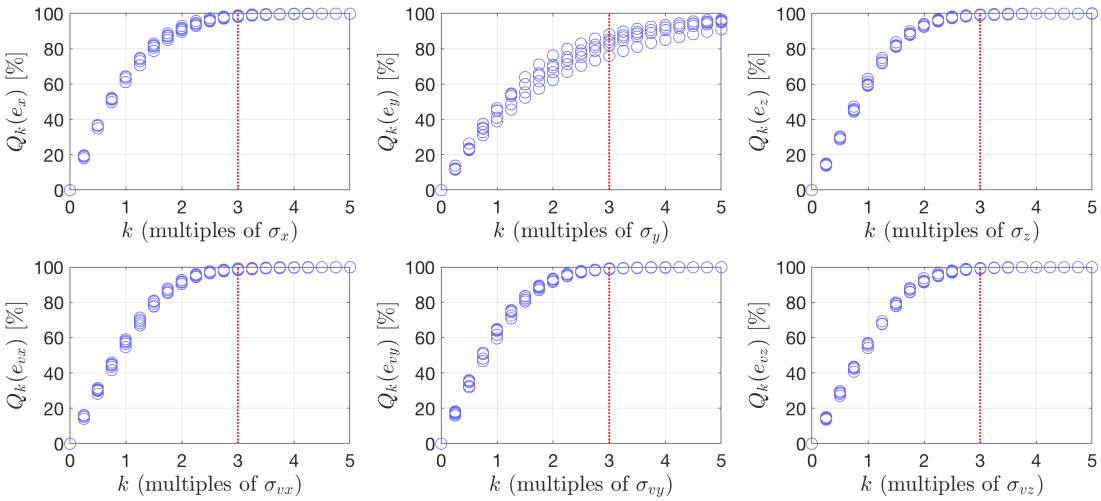


Figure 5: Curves of the metrics $Q_k(e_\xi)$ for all the six components with respect to k.

random seed has little effect on the coverage. For e_y, the randomness has greater effect. Averagely, about 85% of the testing data fall in the $3\sigma_y$-boundary and the best performance among the five random seeds is 88.3%.

4. Results on Satellite Catalog Data

In this section, two publicly available RSO catalogs, the TLE and the ILRS catalogs, are used to further demonstrate the capability of the ML approach using GPs. Under this situation, the orbit measurement and determination processes are unknown to the users, and the orbit prediction errors are expected to be much more complicated than those in the simulation environment. As presented later, the trained GP models can achieve good performance for many cases.

4.1. RSO Catalogs and Setups

The TLE catalog provides orbit information of RSOs larger than 10 centimeter diameter, but the orbital information has limited accuracy. The International Laser Ranging Service (ILRS) provides several data services, including precise satellite ephemerides for the tracked RSOs [27]. The precise ephemeris of a tracked RSO is published in the Consolidated Prediction Format (CPF) file [28]. The TLE data used in this paper are retrieved from Space-Track website, which are then parsed and propagated using the SGP4 model in Orekit. The CPF data are accessed from Crustal Dynamics Data Information System (CDDIS). The data are parsed and interpolated using a routine based on the sample code from CDDIS[3]. Further descriptions including the detailed use of the TLE and CPF data are referred to our previous publication [23]. Since the CPF data has much higher accuracy then the TLE data, the CPF data is used as the "true" state of the TLE catalog, whereas the TLE data corresponds to the "estimate" in previous discussions.

[3] https://ilrs.cddis.eosdis.nasa.gov/docs/2017/cpf_sample_code_v1.01d.tgz, retrieved at 2018/05/29.

The ILRS catalog has data of totally 217 RSOs, among which 68 RSOs are being tracked now[4] and 149 RSOs have been tracked before[5]. In this paper, we have used the same seven RSOs as in the previous study [29]. Their parameters are summarized in Table 3. In the table, all the RSOs are in SSO and ENVISAT which has been used in the simulation is included as RSO-6. The inclination, eccentricity, RAAN, and B^* are extracted from the TLE set at a common epoch which is the time to start of the simulation. The CPF agent indicates the source of the CPF data, where "HTS" refers to "Honeywell Technology Solutions Inc." and "GFZ" refers to "GeoForschungsZentrum, Germany"[6]. The RSOs are divided into two categories and are sorted by altitude. The first three RSOs are passive objects, whereas the last four are operational and capable of orbit maneuvers. We note that the coverage and data quality can differ for RSOs in different orbit regimes other than SSO, therefore, the problem studied in this paper is a smaller scale of the real applications.

Table 3: Information of RSOs studied in this paper, extracted from TLE sets in year 2010.

No.	RSO Name	NORAD ID	Orbit Type	Maneuver Capability	Year	TLE sets	CPF Agent	Altitude [km]	Semimajor axis [km]	Eccentricity	Incli. [deg]	RAAN [deg]	B*	Period [min]
1	LARETS	27944	SSO	No	2010	931	HTS	679	7050	0.0020	97.9	-129.6	5.33E-06	100.2
2	STELLA	22824	SSO	No	2010	564	HTS	811	7182	0.0015	98.5	-51.4	7.29E-06	103.0
3	BLITS	35871	SSO	No	2010	796	HTS	818	7189	0.0026	98.8	55.7	1.00E-04	103.1
4	TERRASAR	31698	SSO	Yes	2010	1250	GFZ	505	6876	0.0008	97.4	10.6	7.78E-06	96.6
5	TANDEM	36605	SSO	Yes	2010	691	GFZ	505	6876	0.0031	97.5	10.1	1.45E-05	96.6
6	ENVISAT	27386	SSO	Yes	2010	1195	HTS	780	7151	0.0028	98.6	70.5	3.85E-06	102.3
7	ERS-2	23560	SSO	Yes	2010	1191	HTS	794	7165	0.0014	98.5	78.4	1.49E-05	102.6

Before conducting the experiments, a critical process is to examine and clean up the dataset. For the TLE and CPF data, we have designed two procedures. First, each TLE state $\hat{\boldsymbol{X}}_{\mathrm{TLE}}$ converted from the TLE set through the SGP4 model is compared with its corresponding "true state" $\boldsymbol{X}_{\mathrm{T}}$ (interpolated from CPF data). Too much deviated TLE sets are removed from the database. Second, the cleaned TLE sets are used to generate the designed dataset similar as in the previous section. The learning and target variables in the dataset are further examined to manually remove possible outliers. These remaining outliers are possibly due to unknown maneuvers or inconsistent TLE and CPF pairs. The number of these manually removed data points is very small with respect to the whole dataset. We note that the goal of the data cleaning is not to manipulate the data for good performance.

The learning and target variables have been slightly modified since the TLE catalog has much sparser data than the simulation environment. The maximum prediction duration $\Delta t_{\max}$ is extended to 14 days to include more data points. Therefore, for the TLE data, eight previous estimates are used to generate the

[4] https://ilrs.cddis.eosdis.nasa.gov/missions/satellite_missions/current_missions/index.html, retrieved on 2018/05/30.
[5] https://ilrs.cddis.eosdis.nasa.gov/missions/satellite_missions/past_missions/index.html, retrieved on 2018/05/30.
[6] https://ilrs.cddis.eosdis.nasa.gov/data_and_products/predictions/prediction_centers.html, retrieved on 2018/04/17.

consistency errors $\boldsymbol{e}_{\rm c}$, satisfying the following constraints,

$$\begin{aligned} t_0 - t_{-1} &> 0 \\ t_{-1} - t_{-2} &> 0.5 \text{ day} \\ t_{-2} - t_{-3} &> 1 \text{ day} \\ t_{-l+1} - t_{-l} &> 2 \text{ days} \qquad l = 4, \ldots, 8, \end{aligned} \tag{21}$$

where t_{-l} denotes the epoch of the l-th previous estimate. This choice of $\boldsymbol{e}_{\rm c}$ provides a backward coverage of roughly 14 days. We note again that this design of learning variables is flexible without manipulating for an optimal performance. In practice, optimal learning variables could be determined through a systematical survey of combinations of available information.

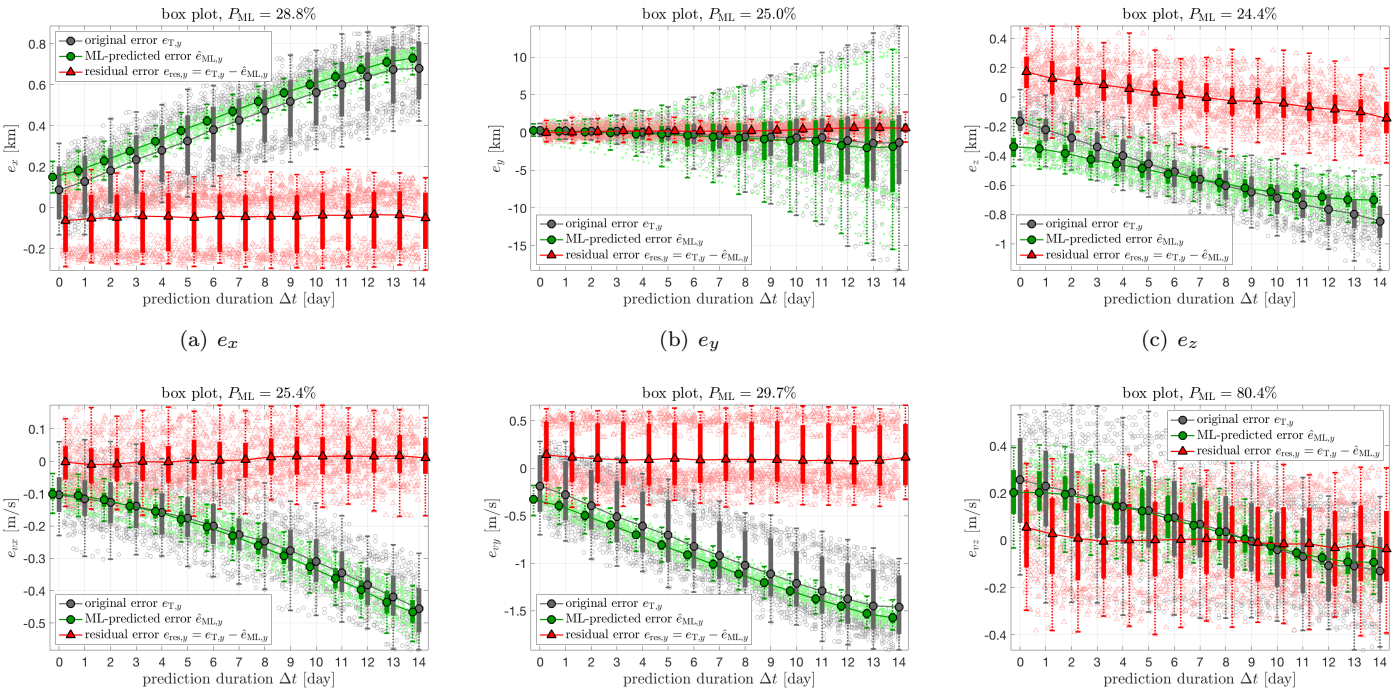
The training data is chosen as the data in days 1–252, the validation data in days 253–266, and the testing data in days 267–280. Similarly, each experiment has been carried out for five times using the five random seeds. Although these RSOs are very different from each other, we will next demonstrate that the ML approach with GP models can be applied to all of them and achieve satisfying performance. Therefore, the hidden relationship between learning and target variables that is intractable to conventional methods can potentially be recovered from historical data automatically by the ML model.

4.2. Performance on LARETS

The performance on RSO-1, LARETS, is analyzed in details first. The performance of others will be presented later.

First, the performance of the trained GP models (best among five random cases) for each component is demonstrated in Figs 6 and 7. In Fig. 6, the metrics $P_{\rm ML}$ are all much smaller than 100% for all components except e_{vz}; all the residual errors (red boxes) have been reduced to around or close to zero; the interquartile of the residual errors of e_y has been significantly reduced compared with the original errors (black boxes); and other components' residual errors' interquartiles have been preserved and similar to the original ones. Figure 7 demonstrates that almost all the testing data of all the components fall inside the 3σ-boundaries, indicating that the trained GP models have generated very reliably approximations of the true orbit prediction errors.

Second, in Fig. 8, we have drawn four examples randomly for each components, in order to demonstrate the variation of the uncertainty boundary and the coverage performance. In each subplot, the sets of true errors $e_{\rm T}$ (black circles) and ML-predicted errors $\hat{e}_{\rm ML}$ (green dots) are based on the same TLE sets, such that the σ of each ML-prediction can be connected to show the boundary. As explained before, the ribbons from dark to light blue centered at $\hat{e}_{\rm ML}$ represent the 1, 2, 3σ-boundaries. For e_y, the trained GP model could detect the correct direction the error grows, either to the negative or to the positive direction. For other components, the ML-predicted error $\hat{e}_{\rm ML}$ captures the general trend but cannot recover the small fluctuation around the mean locations. Further detailed observations tell that the results are not a simple nonlinear



(a) e_x (b) e_y (c) e_z

(d) e_{vx} (e) e_{vy} (f) e_{vz}

Figure 6: Performance metrics $P_{\mathrm{ML}}(e_\xi)$ on RSO-1, LARETS.

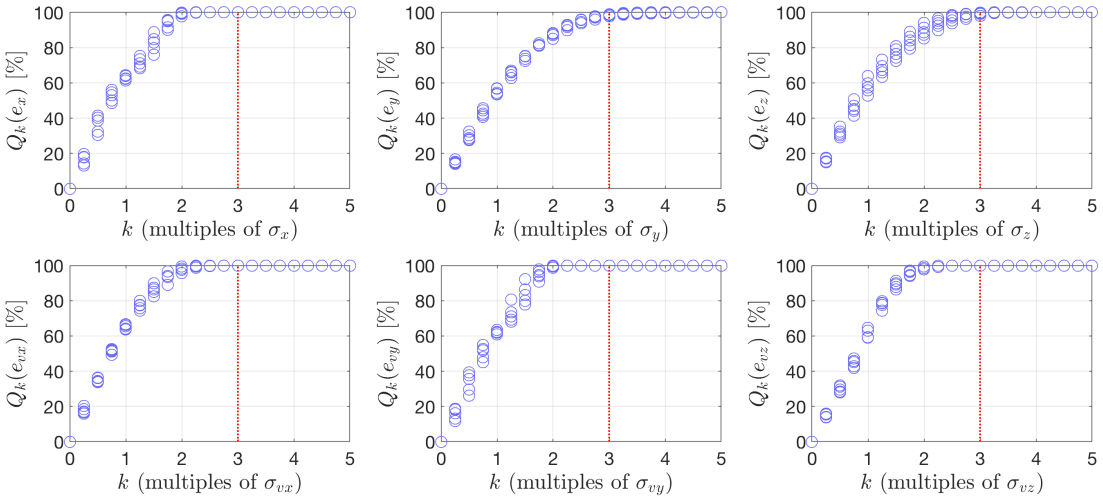
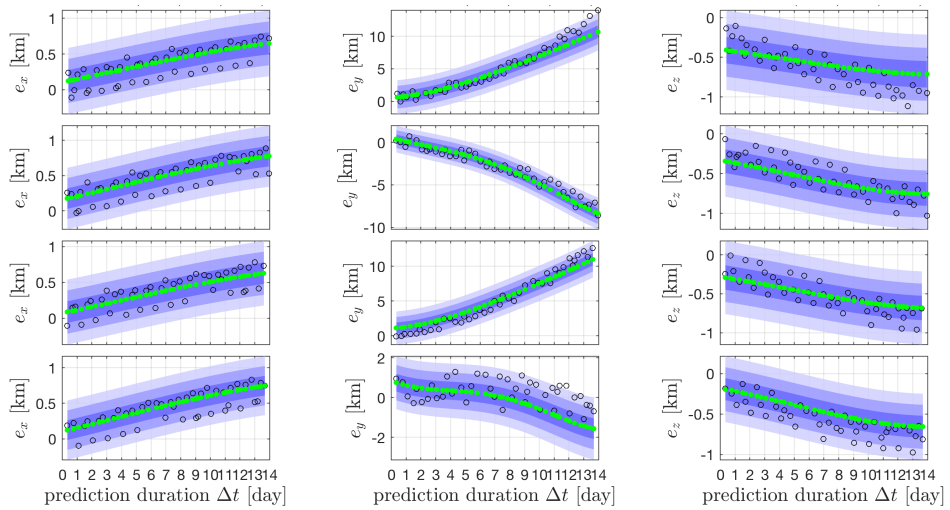


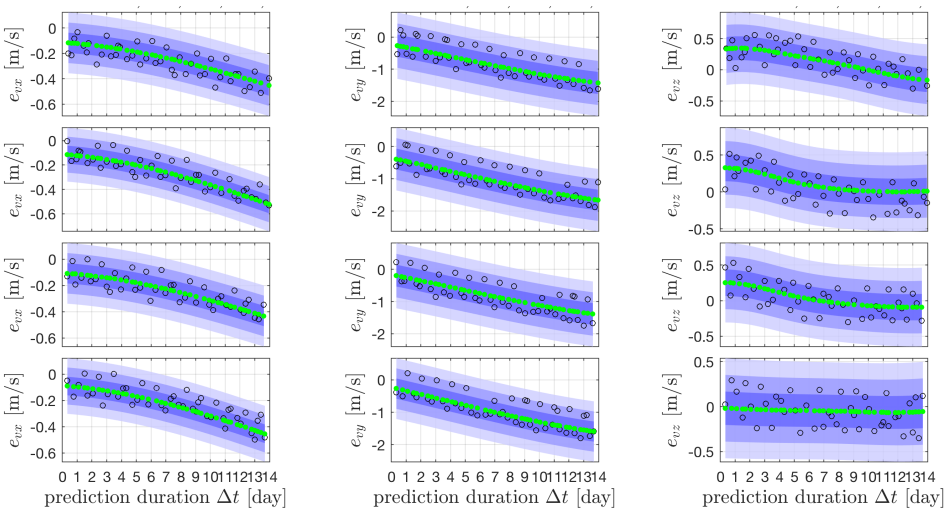
Figure 7: Performance metrics $Q_k(e_\xi)$ on RSO-1, LARETS.

fitting with Δt.

Third, we analyze the overall position or velocity accuracy, which is more helpful for some practical applications. For example, taking optical measurement of a passing target RSO requires to point the view center at the predicted position state of the RSO, so it is more possible to observe the target if the predicted state is close to the actual state, especially when the field of view is limited for the steering sensor [30]. Under this situation, the performance on an individual component is not the most important, while the overall position error $|\boldsymbol{e}_{\mathrm{pos}}|$ is more critical to a successful detection of the RSO. In other cases where tracking of the entire pass is desired, RSO's velocity prediction is important, therefore, the improvement of the overall



(a) e_x (b) e_y (c) e_z



(d) e_{vx} (e) e_{vy} (f) e_{vz}

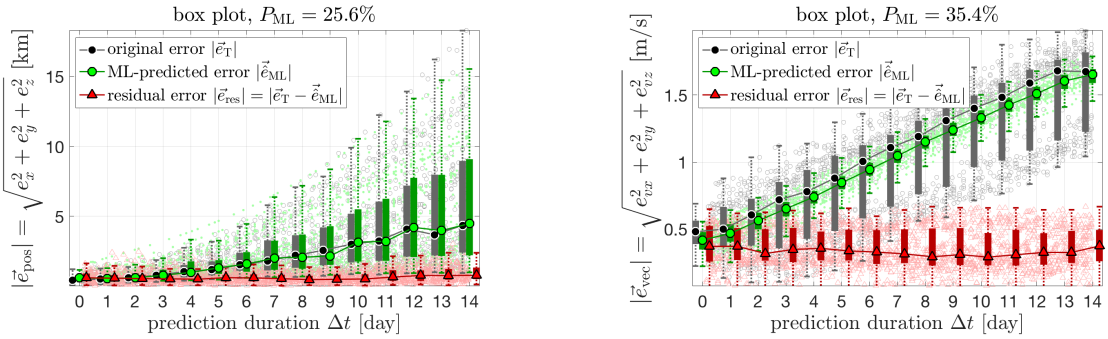
Figure 8: Four examples of the ML-predicted uncertainties $k\sigma$-boundaries ($k = 1, 2, 3$) of RSO-1, LARETS. (The green dots represent ML-predicted value, the black circles represent the true value, and the blue ribbons represent the uncertainty boundaries.)

velocity prediction accuracy is worth to be analyzed.

In Fig. 9, the best performance on the overall position and velocity error is demonstrated, using similar annotations as previously. The overall position error $|\boldsymbol{e}_{\rm pos}|$ is calculated by combining the performance for each component, i.e. $|\boldsymbol{e}_{\rm pos}| = \sqrt{e_x^2 + e_y^2 + e_z^2}$, and the best performance is obtained by using the best trained GP model for each component. Similarly, we define $|\boldsymbol{e}_{\rm vel}| = \sqrt{e_{vx}^2 + e_{vy}^2 + e_{vz}^2}$. In both panels of Fig. 9, the accuracy has been significantly improved after the ML-modification. The effect is especially appealing when Δt is large. We note that even for short predictions during which the TLE set is usually believed to be accurate, for example when $\Delta t < 2$ days, the accuracy can still be slightly improved.

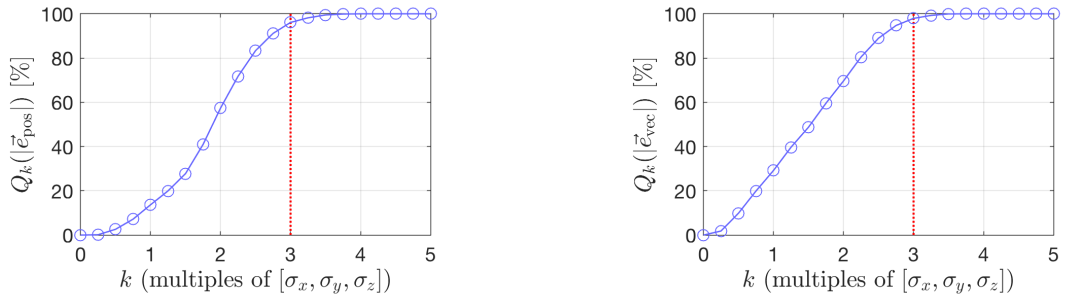
The curve Q_k of overall position vector $\hat{e}_{\rm ML,pos}$ is demonstrated in Fig. 10. The uncertainty boundary of the position vector $\boldsymbol{e}_{\rm pos}$ is an ellipsoid with principal axes lengths of $2\sigma_x$, $2\sigma_y$, and $2\sigma_z$. Similar ellipsoid boundary can be defined for the overall velocity. As shown by the curve, more than 90% of the testing data

have fall inside the 3σ-ellipsoid, meaning that the trained GP models are reliable to predict the uncertainties.



(a) position accuracy (b) velocity accuracy

Figure 9: Performance metrics P_{ML} on the overall position and velocity of RSO-1, LARETS.



(a) position coverage (b) velocity coverage

Figure 10: Performance metrics Q_k on the overall position and velocity of RSO-1, LARETS.

4.3. Performance on All RSOs

The RSOs studied in this paper are chosen based on available data. Since the space environment is complicated and each RSO has its unique features, the performance of the trained GP models on other RSOs will be different from those of LARETS. Both the best and worst (among the five random seeds) performance of the trained GP models on all the RSOs are summarized in Table 4.

First, we study the P_{ML} and Q_3 of the best cases. Remind that the first three RSOs are passive and do not have maneuvering capabilities. They show good performance on reducing the orbit prediction error, small P_{ML}, except BLITS' e_z component. However, since e_z usually has much smaller magnitude than e_y, the overall position and velocity performance of all the three passive RSOs are excellent.

For the other four maneuverable RSOs, the best overall position and velocity performance (the last two columns) reveal that the position and velocity error have been reduced by about 30% for the worst case (RSO-6's $P_{\mathrm{ML}}(|\boldsymbol{e}_{\mathrm{pos}}|) = 70.1\%$; RSO-5's $P_{\mathrm{ML}}(|\boldsymbol{e}_{\mathrm{vel}}|) = 70.8\%$). There is always one component whose metric P_{ML} is large ($> 90\%$), indicating that the orbit prediction errors have not been efficiently compensated for. Considering that these four RSOs are still operational during 2010, the results here indeed reveal that the trained GP models have captured some intrinsic relationships not affected by operations, at least to some

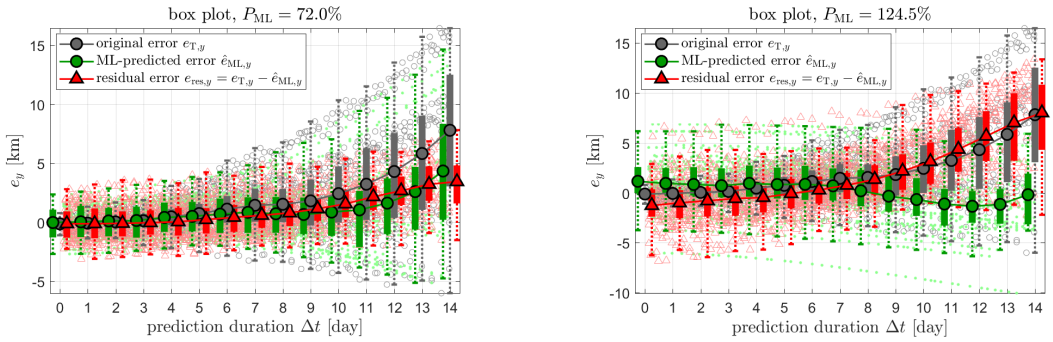
extent. The coverage metric Q_3 show that the true errors mostly fall inside the 3σ-boundaries, meaning that the trained GP models are statistically reliable. For $|\boldsymbol{e}_{\text{pos}}|$ and $|\boldsymbol{e}_{\text{vec}}|$, Q_3 is slightly smaller than that of individual components because the uncertainty boundary is an ellipsoid in the three-dimensional case.

Table 4: Performance metrics P_{ML} and Q_3 [%] of all the components and the overall position/velocity of each RSO, with the results of both the best and worst cases using different random seeds.

No.	RSO Name	Metric	Case	e_x	e_y	e_x	e_{vx}	e_{vy}	e_{vz}	$\lvert\boldsymbol{e}_{\text{pos}}\rvert$	$\lvert\boldsymbol{e}_{\text{vel}}\rvert$
1	LARETS	P_{ML}	best	28.8	25.0	24.4	25.4	29.7	80.4	25.6	35.4
		P_{ML}	worst	30.2	31.4	27.5	26.0	32.1	86.9	31.5	37.3
		Q_3	best	100.0	99.0	99.8	100.0	100.0	100.0	96.1	98.0
		Q_3	worst	100.0	98.4	99.6	100.0	100.0	100.0	92.9	97.8
2	STELLA	P_{ML}	best	34.5	15.1	20.4	33.0	35.1	46.3	16.4	41.1
		P_{ML}	worst	36.2	16.8	26.0	38.6	35.9	51.5	18.4	44.0
		Q_3	best	100.0	99.0	88.0	99.8	100.0	98.8	78.0	75.8
		Q_3	worst	100.0	97.8	89.8	96.7	100.0	95.1	75.8	69.7
3	BLITS	P_{ML}	best	16.0	5.6	122.9	35.5	15.3	45.1	6.3	24.7
		P_{ML}	worst	18.2	8.1	131.7	38.1	18.1	49.3	8.7	27.4
		Q_3	best	100.0	98.6	100.0	100.0	100.0	100.0	95.7	94.2
		Q_3	worst	100.0	99.0	99.2	100.0	100.0	100.0	84.1	97.6
4	TERRASARX	P_{ML}	best	54.3	24.6	92.5	46.7	66.6	62.2	24.7	62.3
		P_{ML}	worst	57.4	58.8	94.5	46.8	67.2	65.1	58.9	63.1
		Q_3	best	96.1	81.0	100.0	99.0	95.9	99.1	71.6	98.2
		Q_3	worst	99.4	92.1	100.0	98.7	98.8	98.9	84.7	96.3
5	TANDEMX	P_{ML}	best	56.1	66.2	92.5	49.7	72.6	78.1	66.2	70.8
		P_{ML}	worst	59.0	81.2	93.5	50.5	73.5	80.9	81.2	72.4
		Q_3	best	94.1	76.6	97.9	99.7	93.4	86.6	68.9	79.7
		Q_3	worst	93.3	87.2	98.9	99.7	92.7	86.1	77.7	79.4
6	ENVISAT	P_{ML}	best	30.8	72.0	68.2	41.3	32.8	110.4	70.1	49.5
		P_{ML}	worst	31.5	124.5	87.1	43.5	33.3	113.2	120.3	50.3
		Q_3	best	100.0	92.4	98.4	100.0	100.0	99.9	82.7	93.2
		Q_3	worst	100.0	98.4	96.8	99.9	100.0	99.4	85.5	88.6
7	ERS2	P_{ML}	best	26.2	39.0	39.4	44.9	27.9	93.1	39.0	35.4
		P_{ML}	worst	26.7	76.4	46.4	45.8	28.4	95.9	74.7	35.8
		Q_3	best	100.0	100.0	99.1	100.0	100.0	100.0	98.1	95.6
		Q_3	worst	100.0	100.0	97.2	100.0	100.0	100.0	95.7	95.3

Next, we analyze some bad performance. As shown in Table 4, for majority cases, the worst metric is no more than 10% larger than the best metric for the passive RSOs and no more than 20% for the maneuvable RSOs. However, there is an exception for e_y, e_z, and e_{vz} of RSO-6, ENVISAT, whose performance appears very sensitive to the choice of random seed. Its metrics $P_{\text{ML}}(e_y)$ is 124.5% in the worst case, but only 72.0% in the best case. The performance of these two cases is further analyzed in Fig. 11. In the best case, the trained GP model has captured the general trend of the true error, but the interquartile of the residual error (red boxes) starts to be smaller than the interquartile of the original error (black boxes) when $\Delta t \geq 6$ days. Meanwhile, in the worst case, the ML-prediction errors (green boxes) fit the original error (black boxes) badly: when Δt is small the ML-prediction errors deviate from the true values greatly; when Δt is large, the general trend is not captured. However, we note that the corresponding coverage metrics Q_3 of the best and worst cases are both good, 92.4% and 98.4% respectively, indicating that the true errors are captured within the uncertainty boundaries. So, although the prediction of means can be bad, the trained GP models can

generate reliable uncertainties.



(a) best case (b) worst case

Figure 11: Comparison of the best and the worst performance of e_y for RSO-6, ENVISAT.

Similar phenomenons have been observed for other bad cases. For passive RSOs, the bad cases could imply that the learning variables have little relation to the target variables, for example, e_z of RSO-3. In these cases, it is necessary to introduce more available information in the learning variables. Alternatively, it is also possible to modify only the good components if the overall accuracy can be significantly improved, for example, $|\boldsymbol{e}_{\mathrm{pos}}|$. For maneuverable RSOs, either regular or occasional maneuvers could introduce inconsistent information in the dataset, which will lead to bad generalization capability on the testing data. Moreover, the geometry and shape of the RSO will also effect the performance of the trained GP models. The current design of learning variables is more capable to capture the averaged variation of the error, rather than the detailed fluctuations. Unfortunately, due to the limitation of the TLE set, especially its publish rate and format, it may be impossible to obtain such details.

5. Conclusions

The recently proposed machine learning (ML) approach to improve the orbit prediction accuracy of resident space objects (RSOs) is further enhanced by generating uncertainty information about the resulting ML-modified orbit prediction error. Specifically, the heteroscedastic Gaussian Processes (GPs) are exploited, which can provide an estimation of the distribution of the orbit prediction errors, including both means and uncertainties. A simulation environment is first used to investigate the performance and capability of the trained GP models. Next, the data from publicly available RSO catalogs are used to further explore the trained GP models. Totally seven RSOs have been examined and the obtained results are mostly positive. The accuracy on the individual components and the overall position and velocity can all be improved. Discussions are presented about a few cases with bad performance, which may be difficult to improve with limited available information in the dataset.

The integration of the ML model and the astrodynamics problem is a relatively new area. Based on the results in this paper as well as our previous publications, the ML approach has been demonstrated to be promising for practical applications. We emphasize that the resulting models need to be checked and

updated regularly, due to possible changes of the orbit determination process as well as the space environment. One research direction is to develop a monitoring system, which will help determine the benefit using the trained ML models and trigger a retraining process when necessary. Another research direction is to fuse the ML-predicted uncertainty information with the conventional uncertainty propagation process.

Acknowledgments

The authors would acknowledge the research support from the Air Force Office of Scientific Research (AFOSR) FA9550-16-1-0184 and the Office of Naval Research (ONR) N00014-16-1-2729. H. Peng would like to thanks Dr. Kuo Chen for inspiring discussions about details of Gaussian Processes method and Mr. Gaurav Misra for his helps on various mathematical derivations.

References

[1] J. C. Liou, A statistical analysis of the future debris environment, Acta Astronautica 62 (2) (2008) 264–271. doi:10.1016/j.actaastro.2006.12.030.

[2] J. C. Liou, N. L. Johnson, N. M. Hill, Controlling the growth of future LEO debris populations with active debris removal, Acta Astronautica 66 (5–6) (2010) 648–653. doi:10.1016/j.actaastro.2009.08.005.

[3] J. T. Olympio, N. Frouvelle, Space debris selection and optimal guidance for removal in the SSO with low-thrust propulsion, Acta Astronautica 99 (2014) 263–275. doi:10.1016/j.actaastro.2014.03.005.

[4] M. Shan, J. Guo, E. Gill, Review and comparison of active space debris capturing and removal methods, Progress in Aerospace Sciences 80 (2016) 18–32. doi:10.1016/j.paerosci.2015.11.001.

[5] E. R. Nesvold, A. Greenberg, N. Erasmus, E. van Heerden, J. L. Galache, E. Dahlstrom, F. Marchis, The Deflector Selector: A machine learning framework for prioritizing hazardous object deflection technology development, Acta Astronautica 146 (2018) 33–45. doi:10.1016/j.actaastro.2018.01.049.

[6] D. Pérez, R. Bevilacqua, Neural Network based calibration of atmospheric density models, Acta Astronautica 110 (2015) 58–76. doi:10.1016/j.actaastro.2014.12.018.

[7] J. F. San-Juan, I. Pérez, M. San-Martín, E. P. Vergara, Hybrid SGP4 orbit propagator, Acta Astronautica 137 (2017) 254–260. doi:10.1016/j.actaastro.2017.04.015.

[8] H. Peng, X. Bai, Improving orbit prediction accuracy through supervised machine learning, Advances in Space Research 61 (10) (2018) 2628–2646. doi:10.1016/j.asr.2018.03.001.

[9] H. Peng, X. Bai, Exploring Capability of Support Vector Machine for Improving Satellite Orbit Prediction Accuracy, Journal of Aerospace Information Systems 15 (6) (2018) 366–381. doi:10.2514/1.I010616.

[10] H. Peng, X. Bai, Artificial Neural Network–Based Machine Learning Approach to Improve Orbit Prediction Accuracy, Journal of Spacecraft and Rockets 55 (5) (2018) 1248–1260. doi:10.2514/1.A34171.

[11] H. Peng, X. Bai, Comparison of Effective Machine Learning Algorithms on Improving Orbit Prediction Accuracy, in: 69th International Astronautical Congress (IAC), Bremen, Germany, 2018, pp. 1–12.

[12] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, 2006, oCLC: ocm61285753.

[13] C. M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer, New York, 2006.

[14] K. P. Murphy, Machine Learning: A Probabilistic Perspective, Adaptive Computation and Machine Learning Series, MIT Press, Cambridge, MA, 2012.

[15] K. Chen, J. Yi, T. Liu, Learning-based modeling and control of underactuated balance robotic systems, in: 2017 13th IEEE Conference on Automation Science and Engineering (CASE), 2017, pp. 1118–1123. doi:10.1109/COASE.2017.8256254.

[16] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, P. Hennig, Gaussian Process-Based Predictive Control for Periodic Error Correction, IEEE Transactions on Control Systems Technology 24 (1) (2016) 110–121. doi:10.1109/TCST.2015.2420629.

[17] H. Shang, Y. Liu, Assessing Accessibility of Main-Belt Asteroids Based on Gaussian Process Regression, Journal of Guidance, Control, and Dynamics 40 (5) (2017) 1144–1154. doi:10.2514/1.G000576.

[18] W. J. Eerland, S. Box, A. Sóbester, Modeling the Dispersion of Aircraft Trajectories Using Gaussian Processes, Journal of Guidance, Control, and Dynamics 39 (12) (2016) 2661–2672. doi:10.2514/1.G000537.

[19] H. Yan, B. Yang, H. Yang, R. Wang, Probabilistic Approach to Conformance Monitoring Using Gaussian Processes, Journal of Guidance, Control, and Dynamics 40 (6) (2017) 1403–1414. doi:10.2514/1.G002383.

[20] I. A. Almosallam, M. J. Jarvis, S. J. Roberts, GPz: Non-stationary sparse Gaussian processes for heteroscedastic uncertainty estimation in photometric redshifts, Monthly Notices of the Royal Astronomical Society 462 (1) (2016) 726–739. arXiv:1604.03593, doi:10.1093/mnras/stw1618.

[21] I. Almosallam, Heteroscedastic Gaussian processes for uncertain and incomplete data, Phd, University of Oxford (Jan. 2017).

[22] Z. Gomes, M. J. Jarvis, I. A. Almosallam, S. J. Roberts, Improving photometric redshift estimation using GPz: Size information, post processing, and improved photometry, Monthly Notices of the Royal Astronomical Society 475 (1) (2018) 331–342. doi:10.1093/mnras/stx3187.

[23] H. Peng, X. Bai, Machine Learning Approach to Improve Satellite Orbit Prediction Accuracy: Validation Using Publicly Available Data (accepted), The Journal of the Astronautical Sciences.

[24] D. A. Vallado, Fundamentals of Astrodynamics and Applications, 1st Edition, The McGraw-Hill Companies, Inc., New York, NY, 1997.

[25] J. Quiñonero-Candela, C. E. Rasmussen, A Unifying View of Sparse Approximate Gaussian Process Regression, Journal of Machine Learning Research 6 (Dec) (2005) 1939–1959.

[26] L. Maisonobe, V. Pommier, P. Parraud, Orekit: An Open Source Library for Operational Flight Dynamics Applications, in: 4th International Conference on Astrodynamics Tools and Techniques, 2010.

[27] M. R. Pearlman, J. J. Degnan, J. M. Bosworth, The International Laser Ranging Service, Advances in Space Research 30 (2) (2002) 135–143. doi:10.1016/S0273-1177(02)00277-6.

[28] R. L. Ricklefs, Consolidated Laser Ranging Prediction Format v1.01, ILRS Data Format and Procedures Working Group.

[29] H. Peng, X. Bai, Generalization Capability of Machine Learning Approach Among Different Satellites: Validated Using TLE Data, in: 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird, UT, 2018, pp. 1–17.

[30] T. A. Hobson, Sensor Management for Enhanced Catalogue Maintenance of Resident Space Objects, Ph.D. thesis, The University of Queensland (2014).