A Predictive Framework for Satellite Re-entry Using Radar Measurements and the Unscented Kalman Filter

Biqing Wang, Jack Roberts, Jai Sagoo, Khadijah Mughrbil, Vijay Dharmaval

May 15, 2025

Abstract

This report presents a predictive framework for estimating the landing site of a de-orbiting satellite by combining a non-linear orbital decay simulation, a radar measurement model that tracks the satellite and provides synthetic observations, and a predictor that uses Unscented Kalman Filter (UKF) to estimate the satellites state (position, velocity and drag coefficient acting on it). Furthermore, it investigates the filter convergence and performance theoretically and under various configurations of the framework.

Keywords: Satellite Re-entry Prediction, Unscented Kalman Filter (UKF), Radar Measurements, State Estimation, Landing Site Prediction

1 Introduction

The use of radar readings and Kalman filter is a common practice for satellite tracking or the tracking of other spacecraft [1]. The methodology and theory of how they work in the aerospace industry has been extensively discussed in a wide range of journey articles and technical reports [1, 2, 3, 4]. To develop further understanding of this framework—particularly how variations in Kalman filter parameters and radar station configurations affect tracking performance—we developed an emulator that simulates the trajectory of a de-orbiting satellite. It also includes a radar measurement model that generates synthetic observational data, which are then processed by a predictor that employs unscented Kalman filter (UKF), a variant of the Kalman filter suited for nonlinear systems [5], to estimate the satellite's trajectory. The emulator finally predicts the satellite's landing position and uncertainty when it is approaching the landing condition.
To gain understanding, we investigate the UKF convergence conditions. By varying the parameters of UKF and the number of radar stations, we analyse how the accuracy and uncertainty of the satellite state estimation, and the prediction on its landing position are affected.

2 Methodology

2.1 Simulation

In order to generate data for the predictor to use, we simulated the satellite's motion in Python to return synthetic radar measurements. This let us take advantage of the `scipy.integrate.solve_ivp` ODE solver and allowed easier integration with the predictor and visualiser.

2.1 Model

We used Newton's second law to formulate a generalised ordinary differential equation (ODE) model describing the motion of a satellite as it de-orbits[1]:

$$\frac{\partial^2 \vec{x}}{\partial t^2} = -\frac{GM_E}{r^2} + \frac{1}{M_{\text{sat}}} f_{\text{drag}}\left(\vec{x}, \vec{v}\right) + \frac{1}{M_{\text{sat}}} \vec{F}_{\text{other}}. \tag{1}$$

F_{other} captures other small forces affecting the satellite's motion, such as radiation pressure from the Sun, tidal forces from the Moon, and electromagnetic forces from the movement of the satellite through the Earth's magnetic field.

Elliptical Earth

We used the non-spherical WGS-84 model [6], which has an absolute accuracy of < 1 metre. Longitude is calculated from Cartesian coordinates as $\lambda = \tan^{-1}\left(\frac{y}{x}\right)$. However, a problem arises when calculating

[1] $\vec{x} = (x, y, z)$ is the position of the satellite, $r = \sqrt{x^2 + y^2 + z^2}$ is the Euclidean distance from the Earth's centre, M_E & M_{sat} are the masses of the Earth & satellite, and $f\left(\vec{x},\vec{v}\right)$ is a function that describes the drag force applied to the satellite by the atmosphere.

latitude as there isn't closed form solution, motivating an iterative method [7]. This was shown to converge to within ≈ 3 nm of the true solution within 3 iterations, which allowed us near-exact accuracy while still remaining computationally efficient.

Atmospheric Density

The Earth's atmospheric density is subject to a large amount of variation that is dependant on a number of factors: temperature, humidity, weather, etc. Calculating the exact density requires measurements and simulations that require a lot of computing power. However, we can make a reasonable approximation of atmospheric density using an exponential model combined with the quadratic drag equation:

$$\rho(\vec{x}) = \rho_0 e^{\frac{r-R_E}{H}} \tag{2}$$

$$\Rightarrow F_{\text{drag}} = \frac{1}{2}\rho(\vec{x}) C_D A \|\vec{v}\|^2, \tag{3}$$

where R_E is the Earth's radius at $\vec{x}$, H is the scale height of the atmosphere, C_D is the drag coefficient, A is cross-sectional area, and ρ_0 is atmospheric density at sea level. This formula works for objects moving in a fluid with high Reynolds number, which is satisfied by the extremely high velocity ($v \sim \mathcal{O}(10^3)$).

2.1 Solving the equations

Now we have a full model, we can formulate the ODE for our system with the assumption that the only forces acting are gravity and atmospheric drag, as they dominate contributions from $\vec{F}_{\text{other}}$. We used the `solve_ivp` function so we had to reformulate (1) as a system of first-order ODEs[2]:

$$\begin{aligned} \frac{\partial \vec{x}}{\partial t} &= \vec{v} \\ \frac{\partial \vec{v}}{\partial t} &= -\frac{GM_E}{r^2} + \frac{1}{2M_{sat}}\rho(\vec{x}) C_D A |v|^2. \end{aligned} \tag{4}$$

This is combined with a śtop conditionźhat stops the solver once the satellite hits the Earth's surface. We used the explicit RK45 method inbuilt in `solve_ivp` as it provides the greatest accuracy while remaining somewhat computationally efficient due to the adaptive time-stepping.

2.2 Radar Stations

Radar stations, which are generally used to track the satellite trajectory, were assumed to be two-way radars, that is, they could transmit and receive signals. Radars were placed around the equator for a satellite in equatorial orbit. For inclined orbits, the radars were positioned all around the Earth. However, these locations were chosen rather arbitrarily, not reflecting any real-world scenario. Position of the satellite was measured by calculating the range, azimuth and elevation angles of the satellite with respect to radar stations. While most radars in real-life can estimate the range rate using Doppler shift, some sophisticated radars can even measure the 3D velocity components. Our radars were assumed to have the same capabilities. Additionally, restrictions were placed on maximum and minimum range and angles that radars could measure. As a result, a radar could not always detect the satellite unless it was in its defined field of view.

To simulate measurements and to check if a radar can see the satellite to consider its measurement valid, various coordinate conversions were employed. The true positions and velocities of the satellite obtained through the satellite trajectory simulation were in Earth Centred Inertial (ECI) frame. At each time step, these position and velocity components were converted from the ECI frame to the local radar frame called East-North-Up (ENU) coordinates. The rotation and the non-spherical nature of the Earth were taken into account during the conversions. The conversion also required an intermediate step of transforming points on the ECI frame to a rotating coordinate system known as the Earth-Centred Earth-Fixed (ECEF) system that rotates with the Earth. Once the satellite's position relative position was computed in ENU coordinates with respect to the radar, the measurements were obtained using the following formulae.

[2] We pass (4) through `solve_ivp` with the initial conditions as a state vector $\vec{x}_{initial\ state} = (x, y, z, v_x, v_y, v_z)$.

$$\begin{aligned}\rho &= \sqrt{e^2 + n^2 + u^2} \\ \theta &= \arctan\left(\frac{e}{n}\right) \\ \phi &= \arctan\left(\frac{u}{\sqrt{e^2 + n^2}}\right)\end{aligned} \tag{5}$$

where e, n, u are the east, north and up components of the satellite's position, ρ is the range, θ and ϕ are azimuth and elevation angles, respectively. The accuracy of radar measurements is usually affected by various factors, such as atmospheric interference and defects in the measurement devices. All of these factors were introduced in the form of uncertainties in range, velocity, and angle measurements. For simplicity, all radars were assumed to have the same level of measurement uncertainties. The velocity components were processed in the ECI coordinates, and noise was added to the individual components. While some measurement noises are non-Gaussian in nature (known as glint noise) [8], the measurement errors were modelled by additive Gaussian noise with zero mean to use with our implementation of UKF which assumes Gaussian noise for measurement and process models. The noise variance for each parameter was set as follows: Range $= 100m$; Azimuth and elevation angles $= 0.001$rad ; Velocity $= 0.05m/s$. After the measurements were simulated and the noise added, the data was sent to the predictor.

2.3 The Predictor

The prediction problem is inherently non-linear, in which the standard Kalman filter method does not apply well. To address this, a non-linear state estimation technique is required. Among the available methods, two widely-used extensions of the Kalman filter are considered: the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) [9]. The EKF linearises the non-linear model equations around the current state estimate using a Taylor series approximation. While computationally efficient, the EKF can introduce significant errors for a non-linear system due to its first-order approximation. Alternatively, the UKF addresses these issues by employing a deterministic sampling method known as the Unscented Transform. Rather than linearising the equations, the UKF propagates a minimal set of deterministically chosen sample points ("sigma points") through the exact non-linear equations, which provide a second-order accurate approximation of the state's mean and covariance under non-linear dynamics. [10, 5, 2, 4]. In addition, EKF is known to be difficult to implement and tune [10]. For these reasons, we use the UKF for our trajectory prediction.
The UKF algorithm represents the state probability distribution using a minimal set of $2n+1$ deterministically selected sigma points[3]. These points are constructed to capture the true mean and covariance of the prior distribution [3]. Like the standard Kalman filter, UKF operates in two steps: the Prediction step, where the sigma points are propagated through the non-linear system dynamics to estimate the prior state (in which the state uncertainty increases), and the Update step, where the predicted estimate is refined using new measurements (in which the state uncertainty is reduced, providing more confidence in the state value overall).

We implemented the UKF using the Python library `FilterPy`, specifically, to generate the sigma points we utilised the function `MerweScaledSigmaPoints` and we used `UnscentedKalmanFilter` perform the prediction and update steps described above.

2.3 Initialisation of the UKF

State vector x, sigma points and state uncertainty P_0

The state vector $\mathbf{x}$ includes seven components representing the satellite's position in the ECI coordinate system (x, y, z), velocity in the ECI coordinate system (v_x, v_y, v_z), and drag coefficient C_d. In addition to tracking the satellite's trajectory, the filter is also capable of estimating unknown parameters within the system model [5]. By augmenting the state vector to include such quantities, the UKF can infer their values over time from the available measurements. In our case, this approach enables the filter to estimate the atmospheric drag coefficient adaptively during flight. According to that, we initialised the state vector to take the position and velocity of the satellite measured by a radar station at time 0, and C_d is set to be 2.0 as an initial guess.
For generating the sigma points, we used the widely adopted Van der Merwes scaled sigma point algorithm with parameters α, β and κ set to be 0.1, 2.0 and -4.0 respectively. α determines the spread of

[3] n is the dimension of the state vector

the sigma points around the mean, and we set it small to limit the distortion caused by non-linearity of the system. β incorporates prior knowledge about the distribution, and the value of $\beta = 2$ is suitable for Gaussian distributions. κ is used alongside α to control the spread of the sigma points usually set to $3 - n$, with n being the dimension of the state vector $\mathbf{x}$ [9, 3].
The matrix $\mathbf{P}$ represents the covariance of the state estimate at each time step. It quantifies the uncertainty about the current state and is used in both the prediction and update steps to balance trust between the model and incoming measurements. At initialisation, $\mathbf{P}_0$ was defined as:

$$\mathbf{P}_0 = \text{diag}(50000^2, 50000^2, 50000^2, 100^2, 100^2, 100^2, 0.5^2) \quad \text{(units: m}^2\text{, (m/s)}^2\text{, (unitless))}$$

which gives high uncertainty in the initial state estimate. As the filter propagates through the predict-update loops, $\mathbf{P}$ gets updated to reflect the uncertainties of the prior and posterior state.

Process model $f(\mathbf{x})$ and process noise Q

We implemented a non-linear process model $f(\mathbf{x})$ to propagate the state $\mathbf{x}$ of the system forward in time. The ODE that governs this model is in line with that implemented in the satellite trajectory simulation, except that the drag coefficient is an assumed constant ($\frac{d(C_d)}{dt} = 0$) to allow UKF to estimate over time, and the ODE solver employed is RK23 to inject some deviation to the true trajectory. At every time step k, $f(\mathbf{x})$ predicts the current state $\hat{\mathbf{x}}_{k|k-1}$ from the previously updated state $\hat{\mathbf{x}}_{k-1|k-1}$ by numerically integrating the ODE over a fixed time step $\Delta t = 50$ (seconds); the `solve_ivp` solver from `scipy.integrate` with `method = 'RK23'` was used in $f(\mathbf{x})$ over the ODE.
The process model is subject to uncertainties arising from unmodelled dynamics, discretisation errors, and external disturbances such as atmospheric variability [9]. To account for these uncertainties, we define a process noise term $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$, where $\mathbf{Q}$ is the process noise covariance matrix. We constructed $\mathbf{Q}$ using the `Q_discrete_white_noise` utility from `FilterPy`, structuring it as a block-diagonal matrix. The diagonal entries for position and velocity components were assigned small values to reflect the filters initial confidence in the accuracy of the process model. For the drag coefficient C_d, we initialised its estimate at 2.0, slightly below the true value of 2.2. A very small process noise variance was assigned to this parameter, indicating the assumption that C_d changes slowly over time. This helps the filter converge smoothly toward the correct value without overreacting to short-term fluctuations in the measurement data. The resulting $\mathbf{Q}$ matrix can be represented as:

$$\mathbf{Q} = \text{diag\_blocks}(Q_{xv_x}, Q_{yv_y}, Q_{zv_z}, Q_{C_d})$$

Where each Q_{xv_x}, Qyv_y, Qzv_z is generated by `Q_discrete_white_noise` with `var` $= 1 \times 10^{-6}$, and $Q_{C_d} = 1 \times 10^{-6}$.

Measurement function $h(\mathbf{x})$ and measurement noise R

The radar measurements received are satellite positions and velocities in spherical and ECI coordinate systems, respectively. These data are incorporated into the UKF update step to correct the state estimate. The measurement model $h(\mathbf{x})$ maps the predicted state vector to the measurement space to facilitate the calculation of the residual between predicted state and measurement received at each time step.
Measurement noise is modelled as zero-mean Gaussian noise with covariance matrix $\mathbf{R}$. This matrix represents the uncertainty associated with radar measurements and is used in the UKF update step to balance trust between the model and observations [9]. The values in $\mathbf{R}$ are chosen based on the expected accuracy of the radars model. This ensures consistency between the filter's internal assumptions and the actual measurement quality. The defined matrix is:

$$\mathbf{R} = \text{diag}(100^2, 0.001^2, 0.001^2, 0.05^2, 0.05^2, 0.05^2) \quad \text{(units: m}^2\text{, rad}^2\text{, rad}^2\text{, (m/s)}^2\text{, (m/s)}^2\text{, (m/s)}^2\text{)}$$

2.3 UKF Execution and Update Strategy

We executed the UKF in a sequential loop consisting of prediction and conditional update steps. At each time step, the filter first performs a prediction of the state using the process model and the process noise configuration.

If a radar measurement is available, the UKF performs an update step to correct its prediction using

the new observation. Otherwise, if no measurement is available, the filter skips the update and proceeds with prediction only. This allows the filter to continue estimating the state even during periods of limited observability, while maintaining numerical stability.

2.3 Landing prediction

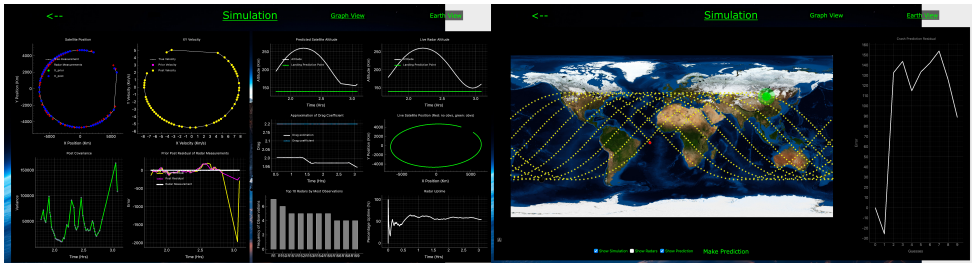
After each update or prediction step, the altitude of the satellite is estimated using the updated position. When the estimated altitude drops below a predefined threshold (140 km), the system triggers a landing prediction phase. This is done by integrating the satellites motion forward in time using `solve_ivp` with the `RK23` method, starting from the last available state and covariance estimate.

To account for the uncertainty in this final estimate, 20 samples are drawn from a multivariate Gaussian defined by the posterior state $\mathbf{x}_k$ and covariance $\mathbf{P}_k$. Each sample is propagated individually to generate a predicted landing point, which is then converted to geographic coordinates (latitude and longitude). The resulting points are aggregated, and their mean is computed to represent the average predicted impact location. Additionally, the covariance of these points is calculated to describe the spread of the predictions and serves as a measure of uncertainty around the estimated landing site.

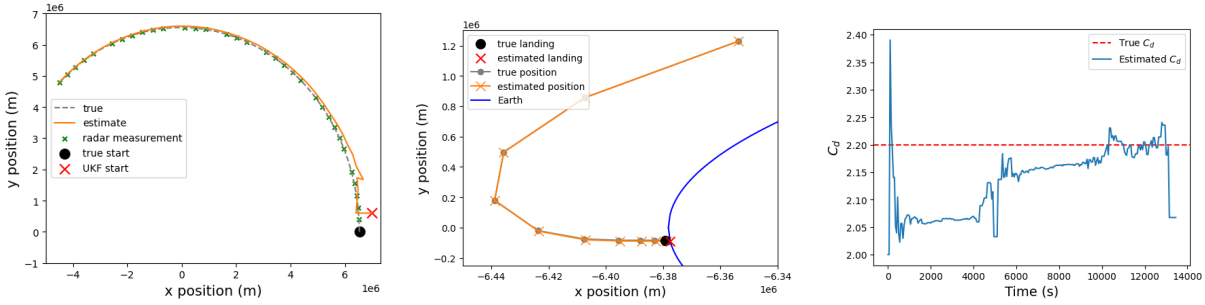
We acknowledge that using 20 samples may not fully capture the true uncertainty, but this choice reflects a practical trade-off between computational cost and estimation accuracy.

2.4 The Visualiser

To integrate the different components of the system, namely the simulation, radar model, and UKF predictor, we developed a custom Graphical User Interface (GUI) using `PyQt` and `PyQtGraph` that serves as an end-to-end software package, controlling each part of the model. The GUI allows the user to modify key parameters such as the initial state of the satellite before running the simulation, and observing how these changes affect the predicted landing location. The visualiser also provides real-time plotting of the filter and simulator's outputs, including trajectory estimates, uncertainty metrics along with true trajectories and a radar uptime percentage, which all update dynamically during execution. An image of the interface is shown in the figure below.



3 Results



(a) Filter estimation from poor initial state

(b) Landing position estimation (distance between true and estimation: 5.3km)

(c) Estimation of drag coefficient

Figure 1: Results validation

To demonstrate that our emulator provides the expected results, we have run simulations to show that (in Figure 1):
(a) Given a suboptimal initial state, UKF has successfully found its way back to be close to the true trajectory within a few steps, showing that the filter is converging well.
(b) The estimated landing position is very close to the true landing position (a distance of 5.6km in this even with 20 samples at each landing prediction step).
(c) The estimation of C_d by UKF has been approaching the true C_d over time, showing a good performance of the filter on the estimation of this augmented state.

4 Analysis and Discussion

In this section, we investigate how the parameters of UKF affect the performance of the satellite trajectory estimation, and how the number of radars impact the accuracy of the landing position and its uncertainty.

4.1 Proof of Convergence

The UKF ideally gives us predictions of the state of the satellite, but it's not guaranteed to converge [11]. Hence we must find conditions under which the UKF converges so that we can tune our model accordingly. We start with a dynamical system and measurement function of the form:

$$x_{n+1} = f(x_n) + \delta_n \tag{6}$$
$$y_n = h(x_n) + \varepsilon_n, \tag{7}$$

where $f : \mathbb{R}^M \to \mathbb{R}^M$ is a smooth function and $\delta_i, \varepsilon_i > 0 \;\; \forall i \in \{1, 2, \dots N\}, \;\; n \in \{1, 2, \dots N\}$.

The update step for the prediction $\hat{x}_{n+1}$ is:

$$K_n = P_{x_n y_n}(P_{y_n y_n})^{-1} \tag{8}$$
$$\hat{x}_{n+1} = \hat{x}_{n+1|n} + K_{n+1}(y_{n+1} - \hat{y}_{n+1|n}), \tag{9}$$

where $P_{x_n y_n}$ is the prior covariance matrix of x_n & y_n and K_n is the Kalman gain at the n^{th} instant.

We define the error as: $e_n = x_n - \hat{x}_n$. Looking at the long-term evolution of e_n we can find conditions under which the UKF converges[4]. By linearising the model and measurement functions in (6) & (7) we can construct an expression for e_{n+1} in terms of e_n and the Jacobians J_n^f & J_n^h (with noise terms):

$$e_{n+1} = (I - K_{n+1}J_n^h)J_n^f e_n + \underbrace{(I - K_{n+1}J_n^h)\delta_n - K_{n+1}\varepsilon_{n+1}}_{\text{noise}} \tag{10}$$

From (10) we can define an error transition matrix $\Phi_{n+1} = (I - K_{n+1})\, J_n^f$ so that:

$$e_{n+1} = \Phi_{n+1}e_0 + \text{noise} \tag{11}$$

We can establish conditions on the convergence of the UKF by analysing the Lyapunov exponents of (11), as if our maximum exponent $\lambda_{\max} < 0$, then the error will converge over time:

$$\begin{aligned}
\lambda_{max} &= \lim_{n\to\infty} \frac{1}{n} \log \left\| \prod_{i=0}^{n-1} \Phi_i \right\| = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log \|\Phi_i\| \\
&\leq \lim_{n\to\infty} \sup \log \left[\|I - K_{n+1}\| \cdot \left(\|J_{21}\|^2 + \|J_{22}\|^2 + 1 \right)^{\frac{1}{2}} \right] \\
&\leq \lim_{n\to\infty} \sup \log \left[(1 + \|K_{n+1}\|) \cdot \left(\|J_{21}\|^2 + \|J_{22}\|^2 + 1 \right)^{\frac{1}{2}} \right]
\end{aligned} \tag{12}$$

In order for $\lambda_{max} < 0$, we must have $(\|I - K_{n+1}\|) \cdot \left(\|J_{21}\|^2 + \|J_{22}\|^2 + 1 \right)^{\frac{1}{2}} < 1$, which we can rearrange

[4]For the generalised derivation/proof of the convergence conditions, refer to the documentation

to get our convergence bounds:

$$\|I - K_{n+1}\| < \frac{1}{\sqrt{\|J_{21}\|^2 + \|J_{22}\|^2 + 1}}$$
$$= \frac{1}{\sqrt{\left(\frac{2GM_E}{r^3} + \frac{\rho_0 C_D A}{2M_{sat}H}\|\vec{v}\|^2\right)^2 + \left(\frac{\rho_0 C_D A}{M_{sat}H} \cdot \|\vec{v}\|\right)^2 + 1}}, \tag{13}$$

$$\Rightarrow K_{n+1} \in \left(1 - \frac{1}{\sqrt{\|J_{21}\|^2 + \|J_{22}\|^2 + 1}}, 1 + \frac{1}{\sqrt{\|J_{21}\|^2 + \|J_{22}\|^2 + 1}}\right). \tag{14}$$

Physically this means that if our process model noise is too high or too low, then our Kalman gain (8) will be out of the convergence range (14):

1. If K_{n+1} is too low, the UKF becomes overconfident and will not converge to the correct solution (in this case the landing location of the satellite).
2. If K_{n+1} is too high then the estimates $\hat{x}_{n+1}$ become too noisy themselves and cause the filter to diverge.

4.2 The effect of UKF parameters on its performance

The analysis is conducted using an equatorial satellite trajectory around a non-spherical Earth with 50 radar stations evenly spaced along the equator, each randomly changing the direction it faces. [5]

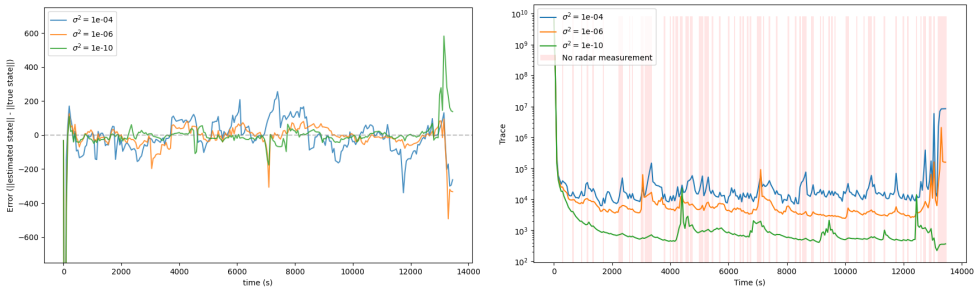
4.2 σ^2 in process noise Q

`filterpy.common.Q_discrete_white_noise` is used to generate sub-blocks within the process noise $\mathbf{Q}$, in which the parameter `var` (i.e. σ^2) controls the 'size' of $\mathbf{Q}$ — larger σ^2 leads to larger $\mathbf{Q}$, hence lower confidence in the process model $f(\mathbf{x})$, and vice versa. To see how σ^2 affects the performance of UKF, we fix all other parameters [6] , then generate three state estimations with σ^2 being 1×10^{-4}, 1×10^{-6} and 1×10^{-10} and examine the error in state estimation and the trace in the estimated covariance $\mathbf{P}$.
The outcome (Figure 2) shows that for all three cases, the estimation errors (2a) are hovering around 0 throughout most of the trajectory, which is what we hope to see for a non-diverging filter ([9]). However, the fluctuation for $\sigma^2 = 1 \times 10^{-4}$ is clearly the highest among all, which is likely due to the filter being overly biased towards the noisy radar measurements. Trace of the estimation covariance demonstrates a clearer distinction among the three scenarios (2b). In general, trace decreases after the update step, reflecting more confidence in the state of the satellite, and increases when a measurement is missing (predict but no update). However, with relatively large σ^2 ($= 1 \times 10^{-4}$), the trace trajectory is relatively volatile and highly sensitive to missing radar measurements, whereas for minute σ^2 ($= 1 \times 10^{-10}$) the filter is much less influenced by missing measurements. Even though minute σ^2 demonstrates a more stable trace of $\mathbf{P}$ for a longer period, at times (e.g. at 4500s) the trace jumps high. This suggests that our process model may be overly confident that the non-linear propagation of the sigma points selected under a 'small' $\mathbf{P}_{k-1}$ leads to the inflation of the estimation uncertainty, i.e. a large jump in $\hat{\mathbf{P}}_k$. [7]
The Normalised Estimation Error Squared (NEES), a metric that measures how well the estimated state uncertainty matches the actual variability in the estimation error [9] (Figure 3, support the error and trace analysis by showing that both large and minute σ^2 values cause NEES to deviate from the 95%CI of its expected distribution.
In all cases, the estimation error jumps to a large value close to the end of the satellite trajectory. This is because as the satellite lands its speed decreases rapidly, change large changes in state values. As the satellite is close to Earth, radar measurements are mostly missing for this period of time, which leads to an increase in the estimation error and uncertainty.

[5] Our emulator can simulate a non-equatorial satellite trajectory with radars placed anywhere on Earth. Our analysis only uses simple scenarios to demonstrate key results

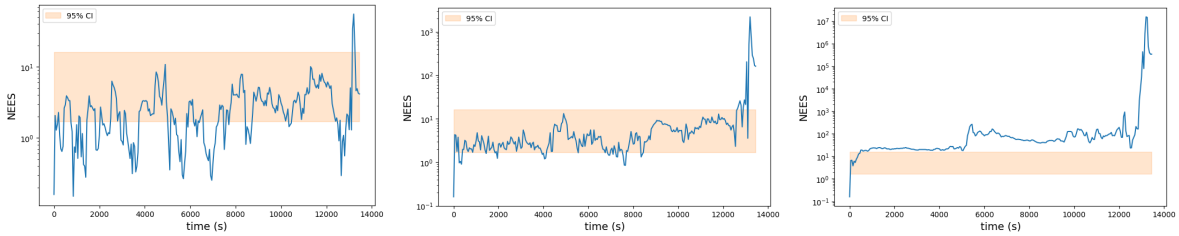
[6] sigma points parameters: $\alpha = 0.1$, $\beta = 2$, $\kappa = -4$, see the initialisation section 2.3.1 for $\mathbf{x}$, $\mathbf{P}$ and $\mathbf{R}$.

[7] The predicted prior covariance $\hat{\mathbf{P}}_k$ is determined as $\hat{\mathbf{P}}_k \approx \sum_{i=0}^{2L} W_i^{(c)} (\mathbf{X}_i - \bar{\mathbf{x}})(\mathbf{X}_i - \bar{\mathbf{x}})^T$ where $W_i^{(c)}$ is the covariance weight of the ith sigma point, $\mathbf{X}_i$ is the ith propagated sigma point, and $\bar{\mathbf{x}}$ is the mean of the sigma points [5].



(a) Error comparison (b) Trace comparison

Figure 2: Error and trace comparison by varying σ^2 of 1×10^{-4}, 1×10^{-6} and 1×10^{-10}



(a) $\sigma^2 = 1 \times 10^{-4}$ (b) $\sigma^2 = 1 \times 10^{-6}$ (c) $\sigma^2 = 1 \times 10^{-10}$

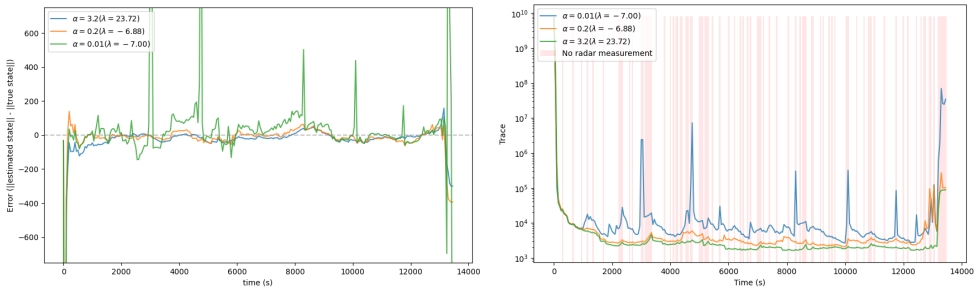
Figure 3: NEES comparison by varying σ^2 of 1×10^{-4}, 1×10^{-6} and 1×10^{-10}

4.2 α in Merwe sigma point algorithm

The spread of the sigma points is controlled by $\lambda = \alpha^2(n + \kappa) - n$, where α, κ are spread-scaling factors, and n is the dimension of the state $\mathbf{x}$. Small λ results in the sigma points being tightly packed and close to the mean. Large λ results in the sigma points being widespread. By fixing the other parameters [8] and letting $\kappa = 3 - n = -4$, we examine how varying α to be 0.01, 0.2, 3.2 (which leads to $\lambda = -7.00$, -6.88 and 23.72 respectively) affects the performance of the UKF.

Figure 4 shows that when α is very small (0.01), the filter is behaving erratically with both the state estimation error and the trace of the state uncertainty producing large jumps often at times when a small $\mathbf{P}_{k-1}$ is followed by a missing update. Similar to the $\sigma^2 = 1 \times 10^{-10}$ case in 4.2.1, the highly non-linear propagation of the tightly clustered sigma points leads to inflated uncertainty in the prediction. $\alpha = 0.2$ still leads to a very small λ. However, we see significant improvement in the filter performance in terms of error and trace trajectories. This is in line with the suggested sigma point spread for systems with strong nonlinearity (since our state dimension is 7, an α larger than the suggested 1e-3 should be better) [5]. We tuned α up gradually to see its effect on the filter. Since our system is highly non-linear, we expected the filter to perform significantly worse when $\alpha > 1$, especially when it causes λ to be positive [9]. In contrary, there was not much of a performance deterioration or variation until α reaches beyond 3.2, for which the filter struggled to propagate. It is possible that the weights of the sigma points played a significant role in preventing bad performances [9]. Such behaviour could also be specific to the setup of our analysis.

[8] $\sigma^2 = 1 \times 10^{-6}$, see initialisation section 2.3.1 for $\mathbf{x}$, $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$



(a) Error comparison (b) Trace comparison

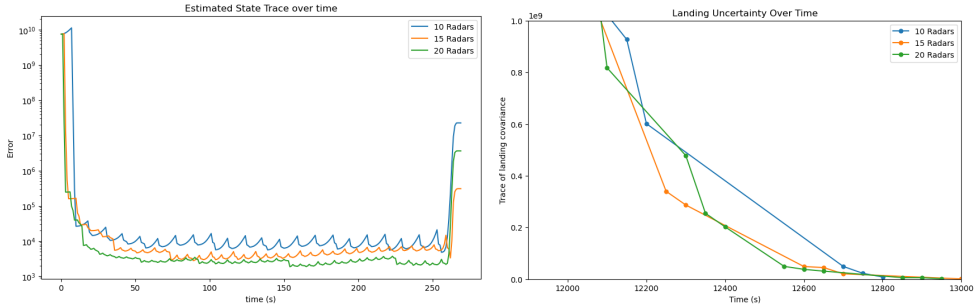
Figure 4: Error and trace comparison by varying α of 0.01, 0.2 and 3.2

4.3 How the number of radars affect the filter performance and landing prediction

We evaluate the filters performance under different simulation scenarios using 10, 15, and 20 radar stations. All radars are evenly distributed along the equator and have symmetric, fixed observation directions pointing vertically upwards without any random changes in orientation. Apart from their geographical locations, the radars are identical in specifications.

The estimation errors during the simulation were broadly similar across the three radar configurations, fluctuating around zero with comparable magnitude. However, when it comes to the landing site prediction, we notice that the estimated landing positions varied significantly depending on the number of radars used. With 10 radars, the final landing error reached 29,401 meters, whereas using 15 radars reduced the error to 10,461 meters, and to only 1,855 meters with 20 radars. This highlights that, although real-time estimation may appear stable across different setups, the accuracy of long-term predictions such is highly sensitive to the radar configuration.

This observation is further supported by the trace of the state covariance and the landing covariance over time. In figure 5 we observe that the 20-radar setup consistently maintained the lowest trace values throughout the simulation, both in the overall state estimate and the predicted landing site. The 15-radar case also demonstrated relatively stable and low uncertainty, whereas the 10-radar configuration showed higher fluctuations and consistently larger trace values. This confirms that increased radar coverage enhances both filter confidence and landing prediction precision.



(a) Estimated State Trace Over Time (b) Landing Covariance Trace

Figure 5: Comparison of filter uncertainty evolution using different radar coverage configurations

5 Conclusion

We have demonstrated that the emulator developed for simulating and predicting the satellite's trajectory, velocity and drag coefficient is one that produces good state estimates and can predict the location of landing fairly accurately. Theoretical analysis shows that if the filter will fail to converge if the process noise $\mathbf{Q}$ is too high or too low. Using the tool to vary the process noise variance parameter σ^2 supports this finding — high σ^2 leads to UKF being overly influenced by any missing radar measurement and the state estimation is relatively poor, whereas low σ^2 leads to the process model being overconfident and giving inflated uncertainties at times.

The spread scaling factor α used in the Van der Merwe sigma point algorithm also plays a significant

role in the filter performance. When κ is fixed at -4, and with our state dimension being 7, small α (e.g. 0.01) results in a highly negative λ, leading to a tightly clustered set of sigma points. This causes the filter estimation to become volatile and exhibit erratic uncertainty behaviour. A moderate α yields stable and accurate estimations. Interestingly, in our analysis, turning up α— and thus the spread of the sigma points—does not noticeably affect the filter's performance. The reasonable filter performance may be due to the weighting scheme: sigma points that are farther away from the mean are assigned much lower weights. However, when α reaches a threshold, the filter fails to propagate.
Furthermore, the number of equally spaced radar stations around the equator affects the estimation uncertainty of the state of a satellite with an equatorial trajectory, as well as the accuracy and uncertainty of the landing position prediction. With more radars, the measurement downtime decreases, helping to keep the filter's estimation uncertainty low. This also improve the accuracy of the landing prediction and leads to a faster reduction in prediction uncertainty shortly before landing.

References

[1] L. A. McGee and S. F. Schmidt, "Discovery of the kalman filter as a practical tool for aerospace and industry," NASA Ames Research Center, Moffett Field, CA, NASA Technical Memorandum NASA-TM-86847, Nov. 1985, unclassified. [Online]. Available: https://ntrs.nasa.gov/api/citations/19860003843/downloads/19860003843.pdf

[2] J. A. P. d. Abreu, R. C. L. d. Oliveira, and J. V. d. Fonseca Neto, "Rocket tracking impact point prediction using α-β, standard kalman, extended kalman, and unscented kalman filters: A comparative analysis," *Research, Society and Development*, vol. 9, no. 3, p. e42932022, 2020. [Online]. Available: https://doi.org/10.33448/rsd-v9i3.2022

[3] R. van der Merwe and E. Wan, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," Oregon Health and Science University, Portland, Tech. Rep., 2004.

[4] M. C. VanDyke, J. L. Schwartz, and C. D. Hall, "Unscented kalman filtering for spacecraft attitude state and parameter estimation," in *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, 2004, paper AAS-04-115. [Online]. Available: https://www.researchgate.net/publication/228751083_Unscented_Kalman_Filtering_for_spacecraft_attitude_state_and_parameter_estimation

[5] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*. IEEE, 2000, pp. 153–158.

[6] U. S. D. of Defense, "Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems," National Imagery and Mapping Agency, Technical Report TR8350.2, 2000.

[7] P. W. Yuanxin Wu and X. Hu, "Algorithm of earth-centered earth-fixed coordinates to geodetic coordinates," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1457–1461, 2003.

[8] W.-R. Wu, "Target racking with glint noise," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 174–185, 1993.

[9] R. R. L. Jr, *Kalman and Bayesian Filters in Python*. GitHub, 2020.

[10] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[11] J. Feng and C. K. Tse, "An unscented-transform-based filtering algorithm for noisy nonlinear systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2007, pp. 1349–1352. [Online]. Available: https://www.eie.polyu.edu.hk/~cktse/pdf-paper/ISCAS07-Feng1.pdf