An Unscented Kalman Filter Based on the Adams-Bashforth Method with Applications to the State Estimation of Osprey-Type Drones Composed of Tiltable Rotor Mechanisms

Keigo Watanabe *, Soma Takeda , Isaku Nagai

Posted Date: 2 March 2026

doi: 10.20944/preprints202603.0141.v1

Keywords: Unscented Kalman Filter; Runge-Kutta method; Adams-Bashforth method; discretization of continuous-time models; UAV state estimation

 doi:10.20944/preprints202603.0141.v1

Article

An Unscented Kalman Filter Based on the Adams-Bashforth Method with Applications to the State Estimation of Osprey-Type Drones Composed of Tiltable Rotor Mechanisms

Keigo Watanabe [1,2,*], Soma Takeda [3] and Isaku Nagai [4]

[1] Okayama University
[2] International Pacific University (IPU)
[3] Sony Group Corporation
[4] National Institute of Technology, Matsue College
* Correspondence: watanabe@sys.okayama-u.ac.jp

Abstract

In the state estimation problem for nonlinear systems, the Unscented Kalman Filter (UKF) has gained attention as an algorithm capable of accurate state estimation based on high-fidelity discretization for strongly nonlinear systems. Furthermore, for applying the UKF to continuous-time state-space models, a method employing the Runge-Kutta method in the time-update equation for sigma points has already been proposed to achieve high-precision state estimation. While this method uses high-order numerical approximations, the associated decrease in computational efficiency due to processing time becomes problematic. It is thus unsuitable for state estimation of relatively fast-moving objects, such as autonomous vehicles and drones, which require high sampling frequencies. In this study, to reduce computational load while achieving relatively high estimation accuracy, we newly apply the Adams-Bashforth method to the UKF algorithm. The effectiveness of the proposed method is demonstrated by first explaining a low-dimensional model's state estimation problem, followed by a comparison of estimation accuracy and computation time in a state estimation simulation for a UAV model of a tandem-rotor drone.

Keywords: Unscented Kalman Filter; Runge-Kutta method; Adams-Bashforth method; discretization of continuous-time models; UAV state estimation

1. Introduction

In recent years, research on autonomous mobile robots, such as self-driving cars and autonomous drones, has been actively pursued. Simultaneous Localization and Mapping (SLAM) is a fundamental technology for the autonomous movement of these robots. SLAM performs self-localization using sequential observation data obtained from multiple different sensors. However, since sensor data contain noise, stable and accurate estimation values cannot be obtained without appropriately integrating them. Therefore, in SLAM, the Kalman filter, a filtering algorithm for sequentially estimating time-varying quantities from discrete, error-contained observation data, is used.

Generally, the motion models of many real-world systems, including autonomous mobile robots, are continuous-time. However, since observation information from sensors is obtained discretely through sampling, the observation model becomes discrete-time. Such a system is called a continuous-discrete (CD) system. Directly applying the Kalman filter to this CD system is called CD-KF. However, sooner or later, during the implementation stage, the continuous-time motion model (i.e., the prediction model) will need to be discretized or numerically integrated, and a discrete-time Kalman filter will be constructed for that discrete-time motion model. This type of Kalman filter is called the discrete-discrete Kalman filter (DD-KF), as both the motion and observation models are discrete-time. Moreover, for nonlinear dynamics or nonlinear observation systems, it is well-known that the CD-EKF or DD-EKF

 doi:10.20944/preprints202603.0141.v1

formally applies the Kalman filter to their linearly approximated systems [1–4]. Furthermore, as a stable algorithm that does not require linear approximation via Jacobians, the Unscented Kalman Filter (UKF) has been proposed for DD nonlinear systems [5–7]. This will be referred to as DD-UKF hereafter.

The UKF employs a nonlinear transformation called the Unscented Transform (UT). For a system of order n, it prepares $(2n+1)$ sample points called sigma points, applies a nonlinear transformation to each, and then calculates the conditional expectation of the state vector and the covariance matrix by taking the sample average of the transformed points. By using this UT, the UKF enables state estimation based on high-fidelity discretization. Sarkka [8] derived continuous-time and continuous-discrete versions of DD-UKF and applied them to nonlinear continuous-time filtering and reentry vehicle tracking problems. Furthermore, in previous research, the Runge-Kutta (RK) method has been applied to the time-update equation for sigma points, demonstrating the possibility of high-precision state estimation for several models [9–11]. Additionally, Takeno & Katayama [12] applied Heun's Method, an improved version of the Euler method, during the prediction step.

However, discussions on the proposed UKFs have been limited to estimation accuracy, with no consideration given to computational efficiency. Due to its characteristic of performing nonlinear transformations on each of the $(2n+1)$ sigma points in the UT algorithm, the UKF has the drawback of increasing computation time as the system order n grows. In practice, considering state estimation for larger-scale models in the real world, both accuracy and real-time performance are required, making the improvement of the UKF algorithm's computational efficiency very important [13].

Recent studies have begun to address this computational challenge from various perspectives. For instance, Wang et al. [14] proposed an adaptive step-size UKF for continuous-discrete systems based on the degree of nonlinearity, achieving a balance between accuracy and computational cost by dynamically adjusting the step size in highly nonlinear regions. Another approach by researchers in the aerospace domain [15] introduced a high-efficiency UKF utilizing parallel computation for sigma point propagation, demonstrating significant speedup for multi-target trajectory estimation. Furthermore, the integration of machine learning techniques with UKF has been explored; a GAN-enhanced UKF framework [16] dynamically predicts and updates filter parameters in real-time, improving estimation accuracy without sacrificing computational performance. In the realm of numerical implementation, Kulikova and Kulikov [17] developed square-root information-type methods for continuous-discrete extended Kalman filtering, enhancing numerical stability which is crucial for efficient computation. Additionally, theoretical advances in sensor scheduling for continuous-discrete systems [18] provide insights into optimizing the trade-off between resource allocation and estimation accuracy. A comparative evaluation of nonlinear filters [19] further contextualizes the performance of UKF against other methods in practical tracking applications. These recent developments underscore the growing recognition of computational efficiency as a critical factor in the practical deployment of nonlinear Kalman filters.

Therefore, in this study, to improve the computational efficiency of the UKF, we propose a UKF that applies the Adams-Bashforth (AB) method, instead of the RK method, to the time-update equation for sigma points. The RK method is a single-step numerical integration method; it calculates the next numerical point $(x(t_{k+1}), t_{k+1})$ based solely on the current point $x(t_k)$, without using past information. Moreover, when the order of accuracy is s $(s < 5)$, s computations are required to generate the next numerical point. On the other hand, the AB method is a multi-step numerical integration method. For an accuracy order s of 4, the next numerical point $x(t_{k+1})$ is calculated based on the last four points $x(t_{k-3}), x(t_{k-2}), x(t_{k-1}), x(t_k)$, and only one computation is needed to generate the next numerical point. Thus, regardless of the accuracy order s of the AB method, the number of computations is always one. Therefore, compared to the fixed-step RK method, computation with the AB method is more efficient.

It should be noted, as a related study focusing on the computational efficiency degradation of the RK method in applying numerical integration to nonlinear Kalman filters, that Renke He et al. [20] have already conducted research applying a multi-step method based on the Adams-Bashforth-Moulton (ABM) method to the Extended Kalman Filter (EKF).

 doi:10.20944/preprints202603.0141.v1

The objective of this study is to clarify that the UKF applying the AB method to the time-update equation for sigma points in the UT can achieve estimation with computational efficiency superior to that of the RK method, while maintaining comparable estimation accuracy. In addition to comparison with the RK method, we also compare the differences in estimation accuracy and computation time due to varying orders (2nd to 6th) of the AB method itself.

The effectiveness of the proposed method is demonstrated through state estimation in a MATLAB simulation environment using two nonlinear models. Specifically, we first verify applicability to a low-dimensional falling object model used in previous research as a preliminary experiment and compare estimation performance under the same conditions. Subsequently, as an application to a more complex model, we conduct a similar comparison of estimation performance using a UAV (Unmanned Aerial Vehicle) model of a tandem-rotor drone.

Section 2 reviews the UKF algorithm for discrete-time state-space models. Section 3 demonstrates the compatibility of UT with two discretization methods, the RK method and the AB method, and presents the time update formulas for the σ point based on each. Section 4 describes the modeling of the Osprey-type drone. After defining the coordinate systems, the rotational and translational motions are explained in detail. Section 5 explains the controller design and control allocation. Specifically, a controller based on the computed torque method is described, and the control input allocation problem is detailed. Section 6 performs state estimation by applying the sigma-point time-update equations presented in Section 2 to the falling object model [21] used in previous research as a preliminary experiment, comparing estimation accuracy and computation time. Section 7 first derives the state equations and sigma-point time-update equations for the resulting tandem-rotor UAV model, then performs comparisons of estimation accuracy and computation time similar to Section 6. Furthermore, we provide discussions based on a wider range of estimation results, including not only comparison with the RK method but also differences in estimation performance due to the order of the AB method and comparisons of estimation performance under different sampling periods. Section 8 summarizes the paper and presents concluding remarks.

2. Review of the Unscented Kalman Filter

Consider the following discrete-time state-space model.

$$x(t_{k+1}) = f(x(t_k), u(t_k)) + w(t_k) \tag{1}$$

$$y(t_k) = h_m(x(t_k)) + v(t_k) \tag{2}$$

where $t_k = k\Delta t$, $k = 0, 1, \ldots$ (Δt: sampling interval). $x(t_k) \in \mathbb{R}^n$, $u(t_k) \in \mathbb{R}^m$, $y(t_k) \in \mathbb{R}^p$ are the state vector, input vector, and output vector, respectively, and $f \in \mathbb{R}^n$, $h_m \in \mathbb{R}^p$ are nonlinear functions. Also, $w(t_k) \in \mathbb{R}^n$, $v(t_k) \in \mathbb{R}^p$ are Gaussian white noises with mean 0 and covariance matrices Q and R, respectively.

Hereafter, the filtering value of the state vector and its covariance matrix are denoted as $\hat{x}(t_k|t_k)$, $P(t_k|t_k)$, and the one-step predicted value and its covariance matrix as $\hat{x}(t_{k+1}|t_k)$, $P(t_{k+1}|t_k)$. It is assumed that the initial filtering values $\hat{x}(0|0)$, $P(0|0)$ are given. The DD-UKF algorithm is then composed of a prediction step and an update step as follows [5–7].

Prediction Step

Step 1: Sigma-point vectors $\mathcal{X}_i (i = 1, \cdots, 2n+1)$ and scalar weights W_i

 doi:10.20944/preprints202603.0141.v1

$$\mathcal{X}_0(t_k|t_k) = \hat{x}(t_k|t_k) \tag{3}$$

$$\mathcal{X}_i(t_k|t_k) = \hat{x}(t_k|t_k) + \left(\sqrt{(n+\lambda)P(t_k|t_k)}\right)_i \tag{4}$$

$$\mathcal{X}_{i+n}(t_k|t_k) = \hat{x}(t_k|t_k) - \left(\sqrt{(n+\lambda)P(t_k|t_k)}\right)_i \tag{5}$$

$$W_0 = \frac{\lambda}{n+\lambda} \tag{6}$$

$$W_i = W_{n+i} = \frac{1}{2(n+\lambda)} \quad (i = 1, \cdots, n) \tag{7}$$

Here, $\left(\sqrt{(n+\lambda)P(t_k|t_k)}\right)_i$ denotes the i-th column vector of the matrix square root of $P(t_k|t_k)$, scaled by $\sqrt{(n+\lambda)}$. λ is a parameter used for computing higher-order moments, such as the covariance.
Step 2: One-step predicted state estimate and its covariance matrix via time update of sigma points

$$\mathcal{X}_i(t_{k+1}|t_k) = f(\mathcal{X}_i(t_k|t_k), u(t_k)) \quad (i = 1, \cdots, 2n+1) \tag{8}$$

$$\hat{x}(t_{k+1}|t_k) = \sum_{i=0}^{2n} W_i \mathcal{X}_i(t_{k+1}|t_k) \tag{9}$$

$$P(t_{k+1}|t_k) = \sum_{i=0}^{2n} W_i \{\mathcal{X}_i(t_{k+1}|t_k) - \hat{x}(t_{k+1}|t_k)\}\{\mathcal{X}_i(t_{k+1}|t_k) - \hat{x}(t_{k+1}|t_k)\}^T + Q \tag{10}$$

Update Step

Step 3: Predicted output estimate and its covariance matrix using updated sigma points

$$\eta_i(t_{k+1}|t_k) = h_m(\mathcal{X}_i(t_{k+1}|t_k)) \quad (i = 1, \cdots, 2n+1) \tag{11}$$

$$\hat{y}(t_{k+1}|t_k) = \sum_{i=0}^{2n} W_i \eta_i(t_{k+1}|t_k) \tag{12}$$

$$P_{yy}(t_{k+1}|t_k) = \sum_{i=0}^{2n} W_i \{\eta_i(t_{k+1}|t_k) - \hat{y}(t_{k+1}|t_k)\}\{\eta_i(t_{k+1}|t_k) - \hat{y}(t_{k+1}|t_k)\}^T \tag{13}$$

Step 4: Output prediction error (innovation)

$$\nu(t_{k+1}|t_k) = y(t_{k+1}) - \hat{y}(t_{k+1}|t_k) \tag{14}$$

Step 5: Auto-covariance matrix of $\nu(t_{k+1})$ and cross-covariance matrix between $x(t_{k+1})$ and $\nu(t_{k+1})$

$$P_{\nu\nu}(t_{k+1}|t_k) = R + P_{yy}(t_{k+1}|t_k) \tag{15}$$

$$P_{x\nu}(t_{k+1}|t_k) = \sum_{i=0}^{2n} W_i \{\mathcal{X}_i(t_{k+1}|t_k) - \hat{x}(t_{k+1}|t_k)\}\{\nu_i(t_{k+1}|t_k) - \hat{y}(t_{k+1}|t_k)\}^T \tag{16}$$

Step 6: Observation update using $\nu(t_{k+1})$ and the Kalman gain $K(t_{k+1})$

$$K(t_{k+1}) = P_{x\nu}(t_{k+1}|t_k) P_{\nu\nu}^{-1}(t_{k+1}|t_k) \tag{17}$$

$$\hat{x}(t_{k+1}|t_{k+1}) = \hat{x}(t_{k+1}|t_k) + K(t_{k+1})\nu(t_{k+1}) \tag{18}$$

$$P(t_{k+1}|t_{k+1}) = P(t_{k+1}|t_k) - K(t_{k+1})P_{\nu\nu}(t_{k+1}|t_k)K^T(t_{k+1}) \tag{19}$$

 doi:10.20944/preprints202603.0141.v1

3. Unscented Transform and Discretization Methods

3.1. Discrete-Time State-Space Model

Consider the following continuous-time nonlinear system.

$$\frac{dx}{dt} = f(x), \qquad x(0) = x_0 \tag{20}$$

Here, $x \in \mathbb{R}^n$ is the state vector and x_0 is the initial value. Let f_d be the approximate function obtained by discretizing Eq. (20). Then, we obtain the time-update equation

$$x(t_{k+1}) = f_d(x(t_k)), \qquad k = 0, 1, \cdots \tag{21}$$

The observation equation is the same as Eq. (2),

$$y(t_k) = h_m(x(t_k)) + v(t_k), \qquad k = 0, 1, \cdots \tag{22}$$

where v is Gaussian white noise with mean 0 and covariance matrix R. Eqs. (21) and (22) constitute the discrete-time state-space model.

The problem is to propose a nonlinear filtering algorithm that sequentially estimates the system's state vector $x(t_k)$ based on the observation data $y(t_k)$. Takeno et al. [9] demonstrated that combining the UKF with the RK method enables high-precision state estimation simulation.

Based on this, this paper shows that by newly combining the AB method with the UT instead of the RK method, relatively high-precision state estimation can be achieved while reducing computational load.

3.2. UT and the Runge-Kutta Method

Here, since the nonlinear function f in Eq. (20) is a continuous-time model before applying discretization, it is redefined as $f_c := f$. First, we derive the time-update equation for the sigma points when applying the UT and the Runge-Kutta method to Eq. (20). For this, let the dimension of x in Eq. (20) be n, the i-th component of f_c be f_i $(i = 1, \cdots, n)$, and redefine the sigma point matrix $\mathcal{X}_k$ at time k with its elements $\mathcal{X}_{i,j,k}$, $(i = 1, \cdots, n, j = 1, \cdots, 2n+1)$ as

$$\mathcal{X}_k = \begin{bmatrix} \mathcal{X}_{1,1,k} & \cdots & \mathcal{X}_{1,2n+1,k} \\ \vdots & \ddots & \vdots \\ \mathcal{X}_{n,1,k} & \cdots & \mathcal{X}_{n,2n+1,k} \end{bmatrix} \tag{23}$$

where i is the subscript indicating the element of the n-dimensional state vector, and j is the subscript indicating the element of the sigma point. Let the difference width be $\Delta t = h$ and $t_k = kh, k = 0, 1, \ldots$. Also, for simplicity, the sigma points in Eqs. (8) and (10) are denoted as $\mathcal{X}^-_{i,j,k+1} = \mathcal{X}_{i,j}(t_{k+1}|t_k)$.

Applying the 4th-order RK method to each sigma point, we define the following [9].

$$\begin{cases} a_{1,ij} = f_i\left(\mathcal{X}_{1,j,k}, \cdots, \mathcal{X}_{n,j,k}, t_k\right) \\ a_{2,ij} = f_i\left(\mathcal{X}_{1,j,k} + \frac{h}{2}a_{1,1j}, \cdots, \mathcal{X}_{n,j,k} + \frac{h}{2}a_{1,nj}, t_k + \frac{h}{2}\right) \\ a_{3,ij} = f_i\left(\mathcal{X}_{1,j,k} + \frac{h}{2}a_{2,1j}, \cdots, \mathcal{X}_{n,j,k} + \frac{h}{2}a_{2,nj}, t_k + \frac{h}{2}\right) \\ a_{4,ij} = f_i\left(\mathcal{X}_{1,j,k} + ha_{3,1j}, \cdots, \mathcal{X}_{n,j,k} + ha_{3,nj}, t_k + h\right) \end{cases} \tag{24}$$

Then, the time-update equation for the sigma points is

 doi:10.20944/preprints202603.0141.v1

$$\mathcal{X}^{-}_{i,j,k+1} = \mathcal{X}_{i,j,k} + \frac{h}{6}\Big(a_{1,ij} + a_{2,ij} + a_{3,ij} + a_{4,ij}\Big) \quad (i = 1, \ldots, n, j = 1, \ldots, 2n+1) \tag{25}$$

Taking the weighted average of these updated sigma points yields the time-updated value (i.e., the predicted value) of the state vector estimate via Eq. (9). The corresponding time-updated covariance matrix is obtained via Eq. (10).

3.3. UT and the Adams-Bashforth Method

Let the j-th column ($j = 1, \cdots, 2n+1$) of the sigma point matrix $\mathcal{X}_k$ at time k be sequentially substituted into f_c. The resulting sigma point matrix $\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$ can be represented by the following $n \times (2n+1)$ matrix.

$$\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) = \begin{bmatrix} f_1(\mathcal{X}_{1,1,k}) & \cdots & f_1(\mathcal{X}_{1,2n+1,k}) \\ \vdots & \ddots & \vdots \\ f_n(\mathcal{X}_{n,1,k}) & \cdots & f_n(\mathcal{X}_{n,2n+1,k}) \end{bmatrix} \tag{26}$$

In this paper, focusing on the key feature of the AB method—its ability to utilize "previously computed information"—we propose a discretization method that uses the sigma point matrix represented by Eq. (26), computed at times $k-1$ and earlier, as "previously computed information." Below, we present the time-update equations for the sigma points when applying 2nd to 6th order AB methods [22,23].

- Sigma point update equation using the 2nd-order Adams-Bashforth method

$$\mathcal{X}^{-}_{k+1} = \mathcal{X}_k + \frac{h}{2}\Big(3\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) - \boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-1})\Big) \tag{27}$$

- Sigma point update equation using the 3rd-order Adams-Bashforth method

$$\mathcal{X}^{-}_{k+1} = \mathcal{X}_k + \frac{h}{12}\Big(23\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) - 16\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-1}) + 5\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-2})\Big) \tag{28}$$

- Sigma point update equation using the 4th-order Adams-Bashforth method

$$\mathcal{X}^{-}_{k+1} = \mathcal{X}_k + \frac{h}{24}\Big(55\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) - 59\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-1}) + 37\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-2}) - 9\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-3})\Big) \tag{29}$$

- Sigma point update equation using the 5th-order Adams-Bashforth method

$$\begin{aligned}\mathcal{X}^{-}_{k+1} = \mathcal{X}_k + \frac{h}{720}\Big(&1901\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) - 2774\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-1}) + 2616\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-2}) - 1274\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-3}) \\ &+ 251\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-4})\Big)\end{aligned} \tag{30}$$

- Sigma point update equation using the 6th-order Adams-Bashforth method

$$\begin{aligned}\mathcal{X}^{-}_{k+1} = \mathcal{X}_k + \frac{h}{1440}\Big(&4277\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) - 7923\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-1}) + 9982\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-2}) - 7298\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-3}) \\ &+ 2877\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-4}) - 475\boldsymbol{\mathcal{F}}_c(\mathcal{X}_{k-5})\Big)\end{aligned} \tag{31}$$

where $\mathcal{X}_k$ is considered an $n \times (2n+1)$ dimensional matrix.

3.4. Comparison of Computational Complexity: Runge-Kutta Method vs. Adams-Bashforth Method

We compare the computational complexity of the RK and AB methods in the UT from the following two perspectives.

 doi:10.20944/preprints202603.0141.v1

3.4.1. Computational Complexity Comparison Based on "Number of Stages"

The difference width $\Delta t = h$ in discretization is generally called the step size in numerical computation. The RK method is a discretization method that computes the derivative $f_i(\mathcal{X})$ four times (this is called having 4 stages) while advancing one step size, and its global truncation error is proportional to $O(h^4)$, indicating 4th-order accuracy.

On the other hand, in the AB method, since it uses previously computed sigma point matrices $\mathcal{F}_c(\mathcal{X}_{k-1})$, $\mathcal{F}_c(\mathcal{X}_{k-2})$, $\mathcal{F}_c(\mathcal{X}_{k-3})$ to derive the time-update equation for the current sigma points, computing $\mathcal{F}_c(\mathcal{X}_k)$ only once is sufficient to advance one step size. In other words, regardless of the order of the global truncation error, the number of stages is always $1/4$ that of the RK method.

3.4.2. Computational Complexity Comparison Based on "Number of Elements in the Sigma Point Set"

Generally, when simulating large-scale models in two- or three-dimensional real space, the number of state variables becomes very large, with examples where the vector element count n reaches millions or tens of millions. The UKF is an example where this effect is pronounced because the sigma point set size is $(2n+1)$, leading to a vast number of elements. The RK method corresponds to each element of the n-dimensional state vector and each element of the sigma point set, performing derivative computations of 4 stages for each sigma point.

In contrast, the AB method requires only one stage of function computation per element of the sigma point set. Moreover, for past sigma point sets, it only involves multiplying the entire matrix by coefficients according to the order of accuracy. Therefore, it can be seen that the impact of increasing the number of state variables is significantly smaller compared to the RK method.

4. Modeling of the Osprey-Type Drone

4.1. Definition of Coordinate Systems

Figure 1 shows the coordinate systems used in this study for the Osprey-type drone. The world coordinate system F_W is a right-handed coordinate system with origin $\boldsymbol{O}_W$ and axes X_W, Y_W, Z_W, where Z_W is positive vertically downward. The body coordinate system F_B has its origin $\boldsymbol{O}_B$ at the center of gravity of the vehicle. It is also a right-handed coordinate system with axes X_B, Y_B, Z_B and Z_B positive vertically downward. The positive direction of the X_B axis is designated as the forward direction of the vehicle.

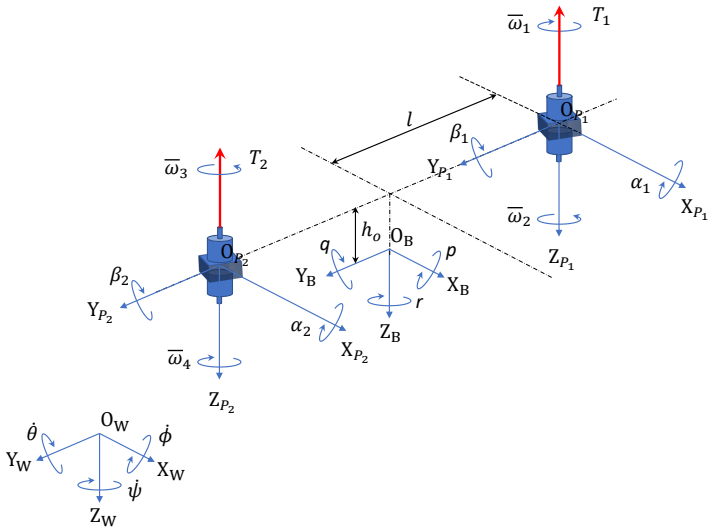


Figure 1. Definition of the coodinate systems of the Osprey-type drone.

The coordinate system for the first coaxial rotor, F_{P_1}, has origin $\boldsymbol{O}_{P_1}$ and axes X_{P_1}, Y_{P_1}, Z_{P_1}. Similarly, the coordinate system for the second coaxial rotor, F_{P_2}, has origin $\boldsymbol{O}_{P_2}$ and axes X_{P_2}, Y_{P_2}, Z_{P_2}. The coordinate system for the i-th ($i = 1, 2$) coaxial rotor is integrated into F_{P_i}. In the body coordinate

 doi:10.20944/preprints202603.0141.v1

system F_B, the angular velocities about the X_B, Y_B, Z_B axes are (p, q, r), respectively. Furthermore, in the rotor coordinate system F_{P_i}, the tilt angles about the X_{P_i}, Y_{P_i} axes are (α_i, β_i). The initial tilt angles α_i and β_i are 0. Due to servo motor characteristics, the range of α_i is $-2/\pi \leq \alpha_i \leq 2/\pi$, while β_i is unlimited.

Let the Euler angles in the world coordinate system be $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^T$. The rotation matrices about the x, y, z axes, $\boldsymbol{R}_x$, $\boldsymbol{R}_y$, $\boldsymbol{R}_z$, can be expressed as follows (S denotes sin, C denotes cos).

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi & C_\phi \end{bmatrix} \tag{32}$$

$$\boldsymbol{R}_y = \begin{bmatrix} C_\theta & 0 & S_\theta \\ 0 & 1 & 0 \\ -S_\theta & 0 & C_\theta \end{bmatrix} \tag{33}$$

$$\boldsymbol{R}_z = \begin{bmatrix} C_\psi & -S_\psi & 0 \\ S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

From these $\boldsymbol{R}_x$, $\boldsymbol{R}_y$, $\boldsymbol{R}_z$, the transformation matrix from the body coordinate system to the world coordinate system, ${}^W\boldsymbol{R}_B$, can be expressed as

$$\begin{aligned} {}^W\boldsymbol{R}_B &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \\ &= \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \end{aligned} \tag{35}$$

Also, let the transformation matrix for angular velocity from the world coordinate system to the body coordinate system be W_η. Then, $\boldsymbol{\omega}_B \triangleq [p\ q\ r]^T$ is

$$\boldsymbol{\omega}_B = W_\eta \dot{\boldsymbol{\eta}} \tag{36}$$

and its inverse is given by

$$\dot{\boldsymbol{\eta}} = W_\eta^{-1} \boldsymbol{\omega}_B \tag{37}$$

Here, it is found in [24,25] that

$$W_\eta = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \tag{38}$$

$$W_\eta^{-1} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \tag{39}$$

where $T_x = \tan(x)$. Note that W_η is invertible unless $\theta = (2k-1)\phi/2, (k \in \mathbb{Z})$. That is, it is invertible as long as it does not take the specific $\phi = \pm\pi/(2k-1)$ where gimbal lock occurs.

From the tilt angles α_i, β_i, $(i = 1, 2)$ in each rotor coordinate system, the transformation matrix from each rotor coordinate system to the body coordinate system, ${}^B\boldsymbol{R}_{P_i}$, can be expressed as

 doi:10.20944/preprints202603.0141.v1

$$
\begin{aligned}
{}^{B}\boldsymbol{R}_{P_1} &= \boldsymbol{R}_z(0)\boldsymbol{R}_y(\beta_1)\boldsymbol{R}_x(\alpha_1) \\
&= \begin{bmatrix} C_{\beta_1} & S_{\alpha_1}S_{\beta_1} & C_{\alpha_1}S_{\beta_1} \\ 0 & C_{\alpha_1} & -S_{\alpha_1} \\ -S_{\beta_1} & S_{\alpha_1}C_{\beta_1} & C_{\alpha_1}C_{\beta_1} \end{bmatrix}
\end{aligned} \tag{40}
$$

$$
\begin{aligned}
{}^{B}\boldsymbol{R}_{P_2} &= \boldsymbol{R}_z(0)\boldsymbol{R}_y(\beta_2)\boldsymbol{R}_x(\alpha_2) \\
&= \begin{bmatrix} C_{\beta_2} & S_{\alpha_2}S_{\beta_2} & C_{\alpha_2}S_{\beta_2} \\ 0 & C_{\alpha_2} & -S_{\alpha_2} \\ -S_{\beta_2} & S_{\alpha_2}C_{\beta_2} & C_{\alpha_2}C_{\beta_2} \end{bmatrix}
\end{aligned} \tag{41}
$$

4.2. Rotational Motion

First, the angular velocity $\boldsymbol{\omega}_{P_i}$ and angular acceleration $\dot{\boldsymbol{\omega}}_{P_i}$ of the i-th ($i = 1, 2$) coaxial rotor can be expressed as

$$
\begin{cases}
\boldsymbol{\omega}_{P_1} = \begin{bmatrix} \dot{\alpha}_1 & \dot{\beta}_1 & \bar{\omega}_2 - \bar{\omega}_1 \end{bmatrix}^T \\
\boldsymbol{\omega}_{P_2} = \begin{bmatrix} \dot{\alpha}_2 & \dot{\beta}_2 & \bar{\omega}_3 - \bar{\omega}_4 \end{bmatrix}^T
\end{cases} \tag{42}
$$

$$
\begin{cases}
\dot{\boldsymbol{\omega}}_{P_1} = \begin{bmatrix} \ddot{\alpha}_1 & \ddot{\beta}_1 & \dot{\bar{\omega}}_2 - \dot{\bar{\omega}}_1 \end{bmatrix}^T \\
\dot{\boldsymbol{\omega}}_{P_2} = \begin{bmatrix} \ddot{\alpha}_2 & \ddot{\beta}_2 & \dot{\bar{\omega}}_3 - \dot{\bar{\omega}}_4 \end{bmatrix}^T
\end{cases} \tag{43}
$$

where $\bar{\omega}_1$, $\bar{\omega}_2$ are the angular velocities of each brushless motor in the first coaxial rotor, and $\bar{\omega}_3$, $\bar{\omega}_4$ are those of the second coaxial rotor.

Next, the reaction torque $\boldsymbol{\tau}_{c,i}$ of the i-th ($i = 1, 2$) coaxial rotor can be expressed as

$$
\begin{cases}
\boldsymbol{\tau}_{c,1} = \begin{bmatrix} 0 & 0 & k_t(\bar{\omega_2}^2 - \bar{\omega}_1^2) \end{bmatrix}^T \\
\boldsymbol{\tau}_{c,2} = \begin{bmatrix} 0 & 0 & k_t(\bar{\omega_3}^2 - \bar{\omega}_4^2) \end{bmatrix}^T
\end{cases} \tag{44}
$$

where $k_t > 0$ represents the torque coefficient.

Also, the thrust $\boldsymbol{T}_i$ of the i-th ($i = 1, 2$) coaxial rotor can be expressed as

$$
\begin{cases}
\boldsymbol{T}_1 = \begin{bmatrix} 0 & 0 & -T_1 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & -k_f(\bar{\omega}_1^2 + \bar{\omega}_2^2) \end{bmatrix}^T \\
\boldsymbol{T}_2 = \begin{bmatrix} 0 & 0 & -T_2 \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & -k_f(\bar{\omega}_3^2 + \bar{\omega}_4^2) \end{bmatrix}^T
\end{cases} \tag{45}
$$

where $k_f > 0$ represents the thrust coefficient of the coaxial rotor.

Using the Newton-Euler method, the torque $\boldsymbol{\tau}_{P_i}$ generated by the coaxial rotor can be expressed as

$$
\boldsymbol{\tau}_{P_i} = \boldsymbol{I}_P\dot{\boldsymbol{\omega}}_{P_i} + \boldsymbol{\omega}_{P_i} \times \boldsymbol{I}_P\boldsymbol{\omega}_{P_i} + \boldsymbol{\tau}_{c,i} \tag{46}
$$

Here, since the angular velocity and angular acceleration caused by the servo motor tilt are instantaneous and minute, they can be expressed as

$$
\begin{gathered}
\dot{\alpha}_i = 0, \quad \ddot{\alpha}_i = 0, \quad \dot{\beta}_i = 0, \quad \ddot{\beta}_i = 0 \quad \dot{\boldsymbol{\omega}}_{P_j} = 0 \\
(i = 1, 2) \quad (j = 1, 2, 3, 4)
\end{gathered} \tag{47}
$$

Therefore, $\boldsymbol{\omega}_{P_i}$ and $\boldsymbol{\tau}_{P_i}$ become as follows:

 doi:10.20944/preprints202603.0141.v1

$$\begin{cases} \boldsymbol{\omega}_{P_1} = \begin{bmatrix} 0 & 0 & \bar{\omega}_2 - \bar{\omega}_1 \end{bmatrix}^T \\ \boldsymbol{\omega}_{P_2} = \begin{bmatrix} 0 & 0 & \bar{\omega}_3 - \bar{\omega}_4 \end{bmatrix}^T \end{cases} \tag{48}$$

$$\begin{cases} \boldsymbol{\tau}_{P_1} = \begin{bmatrix} 0 & 0 & k_t(\bar{\omega}_2^2 - \bar{\omega}_1^2) \end{bmatrix}^T \\ \boldsymbol{\tau}_{P_2} = \begin{bmatrix} 0 & 0 & k_t(\bar{\omega}_3^2 - \bar{\omega}_4^2) \end{bmatrix}^T \end{cases} \tag{49}$$

4.2.1. Rotational Torque $\boldsymbol{\tau}_B$ Generated by Rotor Thrust

The rotational torque $\boldsymbol{\tau}_B$ generated on the vehicle by rotor thrust can be expressed using the position vector ${}^B\boldsymbol{O}_{r_i}$ of each rotor in the body coordinate system as follows:

$$\boldsymbol{\tau}_B = \sum_{i=1}^{2} ({}^B\boldsymbol{O}_{P_i} \times {}^B\boldsymbol{R}_{P_i}\boldsymbol{T}_i) \tag{50}$$

$${}^B\boldsymbol{O}_{P_i} = \begin{bmatrix} 0 & (-1)^i l & -h_o \end{bmatrix}^T \tag{51}$$

The calculation details for $i = 1, 2$ are shown below:

$${}^B\boldsymbol{R}_{P_1}\boldsymbol{T}_1 = \begin{bmatrix} C_{\beta_1} & S_{\alpha_1}S_{\beta_1} & C_{\alpha_1}S_{\beta_1} \\ 0 & C_{\alpha_1} & -S_{\alpha_1} \\ -S_{\beta_1} & S_{\alpha_1}C_{\beta_1} & C_{\alpha_1}C_{\beta_1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -T_1 \end{bmatrix} = \begin{bmatrix} -C_{\alpha_1}S_{\beta_1}T_1 \\ S_{\alpha_1}T_1 \\ -C_{\alpha_1}C_{\beta_1}T_1 \end{bmatrix}$$

$${}^B\boldsymbol{R}_{r_2}\boldsymbol{T}_2 = \begin{bmatrix} C_{\beta_2} & S_{\alpha_2}S_{\beta_2} & C_{\alpha_2}S_{\beta_2} \\ 0 & C_{\alpha_2} & -S_{\alpha_2} \\ -S_{\beta_2} & S_{\alpha_2}C_{\beta_2} & C_{\alpha_2}C_{\beta_2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -T_2 \end{bmatrix} = \begin{bmatrix} -C_{\alpha_2}S_{\beta_2}T_2 \\ S_{\alpha_2}T_2 \\ -C_{\alpha_2}C_{\beta_2}T_2 \end{bmatrix}$$

$${}^B\boldsymbol{O}_{P_1} \times {}^B\boldsymbol{R}_{P_1}\boldsymbol{T}_1 = \begin{bmatrix} 0 \\ -l \\ -h_o \end{bmatrix} \times \begin{bmatrix} -C_{\alpha_1}S_{\beta_1}T_1 \\ S_{\alpha_1}T_1 \\ -C_{\alpha_1}C_{\beta_1}T_1 \end{bmatrix} = \begin{bmatrix} lC_{\alpha_1}C_{\beta_1}T_1 + h_oS_{\alpha_1}T_1 \\ h_oC_{\alpha_1}S_{\beta_1}T_1 \\ -lC_{\alpha_1}S_{\beta_1}T_1 \end{bmatrix}$$

$${}^B\boldsymbol{O}_{P_2} \times {}^B\boldsymbol{R}_{P_2}\boldsymbol{T}_2 = \begin{bmatrix} 0 \\ l \\ -h_o \end{bmatrix} \times \begin{bmatrix} -C_{\alpha_2}S_{\beta_2}T_2 \\ S_{\alpha_2}T_2 \\ -C_{\alpha_2}C_{\beta_2}T_2 \end{bmatrix} = \begin{bmatrix} -lC_{\alpha_2}C_{\beta_2}T_2 + h_oS_{\alpha_2}T_2 \\ h_oC_{\alpha_2}S_{\beta_2}T_2 \\ lC_{\alpha_2}S_{\beta_2}T_2 \end{bmatrix}$$

Therefore, $\boldsymbol{\tau}_B = [\tau_x^B \ \tau_y^B \ \tau_z^B]^T$ can be expressed as

$$\begin{aligned} \boldsymbol{\tau}_B = \begin{bmatrix} \tau_x^B \\ \tau_y^B \\ \tau_z^B \end{bmatrix} &= \begin{bmatrix} (lC_{\alpha_1}C_{\beta_1} + h_oS_{\alpha_1})T_1 - (lC_{\alpha_2}C_{\beta_2} - h_oS_{\alpha_2})T_2 \\ h_oC_{\alpha_1}S_{\beta_1}T_1 + h_oC_{\alpha_2}S_{\beta_2}T_2 \\ -lC_{\alpha_1}S_{\beta_1}T_1 + lC_{\alpha_2}S_{\beta_2}T_2 \end{bmatrix} \\ &= k_f \begin{bmatrix} (lC_{\alpha_1}C_{\beta_1} + h_oS_{\alpha_1})(\bar{\omega}_1^2 + \bar{\omega}_2^2) \\ \quad - (lC_{\alpha_2}C_{\beta_2} - h_oS_{\alpha_2})(\bar{\omega}_3^2 + \bar{\omega}_4^2) \\ h_oC_{\alpha_1}S_{\beta_1}(\bar{\omega}_1^2 + \bar{\omega}_2^2) + h_oC_{\alpha_2}S_{\beta_2}(\bar{\omega}_3^2 + \bar{\omega}_4^2) \\ -lC_{\alpha_1}S_{\beta_1}(\bar{\omega}_1^2 + \bar{\omega}_2^2) + lC_{\alpha_2}S_{\beta_2}(\bar{\omega}_3^2 + \bar{\omega}_4^2) \end{bmatrix} \end{aligned} \tag{52}$$

4.2.2. Reaction Torque $\boldsymbol{\tau}_c$ Generated by Rotor Rotation

The reaction torque $\boldsymbol{\tau}_c$ generated by rotor rotation can be expressed using the reaction torque $\boldsymbol{\tau}_{P_i}$ of each rotor as follows:

$$\boldsymbol{\tau}_c = \sum_{i=1}^{2} ({}^B\boldsymbol{R}_{P_i}\boldsymbol{\tau}_{P_i}) \tag{53}$$

 doi:10.20944/preprints202603.0141.v1

The calculation details for $i = 1, 2$ are shown below.

$$
\begin{aligned}
{}^{B}\boldsymbol{R}_{P_1}\boldsymbol{\tau}_{P_1} &= \begin{bmatrix} C_{\beta_1} & S_{\alpha_1}S_{\beta_1} & C_{\alpha_1}S_{\beta_1} \\ 0 & C_{\alpha_1} & -S_{\alpha_1} \\ -S_{\beta_1} & S_{\alpha_1}C_{\beta_1} & C_{\alpha_1}C_{\beta_1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ k_t(\bar{\omega}_2^2 - \bar{\omega}_1^2) \end{bmatrix} \\
&= k_t \begin{bmatrix} C_{\alpha_1}S_{\beta_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) \\ -S_{\alpha_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) \\ C_{\alpha_1}C_{\beta_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
{}^{B}\boldsymbol{R}_{P_2}\boldsymbol{\tau}_{P_2} &= \begin{bmatrix} C_{\beta_2} & S_{\alpha_2}S_{\beta_2} & C_{\alpha_2}S_{\beta_2} \\ 0 & C_{\alpha_2} & -S_{\alpha_2} \\ -S_{\beta_2} & S_{\alpha_2}C_{\beta_2} & C_{\alpha_2}C_{\beta_2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ k_t(\bar{\omega}_3^2 - \bar{\omega}_4^2) \end{bmatrix} \\
&= k_t \begin{bmatrix} C_{\alpha_2}S_{\beta_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \\ -S_{\alpha_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \\ C_{\alpha_2}C_{\beta_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \end{bmatrix}
\end{aligned}
$$

Therefore, $\boldsymbol{\tau}_c = [\tau_x^c \ \tau_y^c \ \tau_z^c]^T$ can be expressed as

$$
\begin{aligned}
\boldsymbol{\tau}_c &= \begin{bmatrix} \tau_x^c \\ \tau_y^c \\ \tau_z^c \end{bmatrix} \\
&= k_t \begin{bmatrix} C_{\alpha_1}S_{\beta_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) + C_{\alpha_2}S_{\beta_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \\ -S_{\alpha_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) - S_{\alpha_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \\ C_{\alpha_1}C_{\beta_1}(\bar{\omega}_2^2 - \bar{\omega}_1^2) + C_{\alpha_2}C_{\beta_2}(\bar{\omega}_3^2 - \bar{\omega}_4^2) \end{bmatrix}
\end{aligned} \tag{54}
$$

4.2.3. Vehicle Coriolis Force $\boldsymbol{\tau}_{cori}$

The vehicle Coriolis force $\boldsymbol{\tau}_{cori}$ can be expressed using the vehicle's angular velocity $\boldsymbol{\omega}_B$ and its inertia matrix $\boldsymbol{I}_B$ as follows:

$$
\boldsymbol{\tau}_{cori} = \boldsymbol{\omega}_B \times \boldsymbol{I}_B\boldsymbol{\omega}_B \tag{55}
$$

Here, since

$$
\boldsymbol{I}_B = \begin{bmatrix} I_{Bxx} & 0 & 0 \\ 0 & I_{Byy} & 0 \\ 0 & 0 & I_{Bzz} \end{bmatrix}, \quad \boldsymbol{\omega}_B = \begin{bmatrix} p \\ q \\ r \end{bmatrix}
$$

we have

$$
\boldsymbol{I}_B\boldsymbol{\omega}_B = \begin{bmatrix} I_{Bxx} & 0 & 0 \\ 0 & I_{Byy} & 0 \\ 0 & 0 & I_{Bzz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} pI_{Bxx} \\ qI_{Byy} \\ rI_{Bzz} \end{bmatrix}.
$$

Therefore, letting $\boldsymbol{\tau}_{cori} = [\tau_x^{cori} \ \tau_y^{cori} \ \tau_z^{cori}]^T$, we obtain

$$
\begin{aligned}
\boldsymbol{\tau}_{cori} = \begin{bmatrix} \tau_x^{cori} \\ \tau_y^{cori} \\ \tau_z^{cori} \end{bmatrix} &= \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} pI_{Bxx} \\ qI_{Byy} \\ rI_{Bzz} \end{bmatrix} \\
&= \begin{bmatrix} qr(I_{Bzz} - I_{Byy}) \\ pr(I_{Bxx} - I_{Bzz}) \\ pq(I_{Byy} - I_{Bxx}) \end{bmatrix}.
\end{aligned} \tag{56}
$$

 doi:10.20944/preprints202603.0141.v1

4.2.4. Vehicle Angular Acceleration $\dot{\boldsymbol{\omega}}_B$

Using the equations above, the vehicle's angular acceleration $\dot{\boldsymbol{\omega}}_B = [\dot{p}\ \dot{q}\ \dot{r}]^T$ can be expressed as

$$\dot{\boldsymbol{\omega}}_B = \boldsymbol{I}_B^{-1}(\boldsymbol{\tau}_B + \boldsymbol{\tau}_c - \boldsymbol{\tau}_{cori} + \boldsymbol{\tau}_{ext}) \tag{57}$$

Furthermore, neglecting external disturbance torques $\boldsymbol{\tau}_{ext}$ acting on the vehicle and gyroscopic effects (i.e., $\boldsymbol{\tau}_{ext} = 0$), we have

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{Bxx}}(\tau_x^B + \tau_x^c) + qr\left(\frac{I_{Byy}-I_{Bzz}}{I_{Bxx}}\right) \\ \frac{1}{I_{Byy}}(\tau_y^B + \tau_y^c) + pr\left(\frac{I_{Bzz}-I_{Bxx}}{I_{Byy}}\right) \\ \frac{1}{I_{Bzz}}(\tau_z^B + \tau_z^c) + pq\left(\frac{I_{Bxx}-I_{Byy}}{I_{Bzz}}\right) \end{bmatrix}. \tag{58}$$

4.3. Translational Motion

The thrust $\boldsymbol{T}_B$ in the body coordinate system can be expressed using the thrust T_i of each rotor as

$$\boldsymbol{T}_B = \sum_{i=1}^{2}({}^B\boldsymbol{R}_{P_i}\boldsymbol{T}_i). \tag{59}$$

Therefore, letting $\boldsymbol{T}_B = [T_x\ T_y\ T_z]^T$, we obtain

$$\begin{aligned} \boldsymbol{T}_B = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} &= \begin{bmatrix} -C_{\alpha_1}S_{\beta_1}T_1 - C_{\alpha_2}S_{\beta_2}T_2 \\ S_{\alpha_1}T_1 + S_{\alpha_2}T_2 \\ -C_{\alpha_1}C_{\beta_1}T_1 - C_{\alpha_2}C_{\beta_2}T_2 \end{bmatrix} \\ &= k_f \begin{bmatrix} -C_{\alpha_1}S_{\beta_1}(\bar{\omega}_1^2 + \bar{\omega}_2^2) - C_{\alpha_2}S_{\beta_2}(\bar{\omega}_3^2 + \bar{\omega}_4^2) \\ S_{\alpha_1}(\bar{\omega}_1^2 + \bar{\omega}_2^2) + S_{\alpha_2}(\bar{\omega}_3^2 + \bar{\omega}_4^2) \\ -C_{\alpha_1}C_{\beta_1}(\bar{\omega}_1^2 + \bar{\omega}_2^2) - C_{\alpha_2}C_{\beta_2}(\bar{\omega}_3^2 + \bar{\omega}_4^2) \end{bmatrix}. \end{aligned} \tag{60}$$

Using the vehicle mass m and gravitational acceleration $\boldsymbol{g} = [0\ 0\ g]^T$, the vehicle's translational acceleration $\ddot{\boldsymbol{\xi}} = [\ddot{x}\ \ddot{y}\ \ddot{z}]^T$ can be expressed as

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g} + \frac{1}{m}({}^W\boldsymbol{R}_B\boldsymbol{T}_B + \boldsymbol{F}_{ext}) \tag{61}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} C_\theta C_\psi T_x + (S_\phi S_\theta C_\psi - C_\phi S_\psi)T_y \\ \quad + (C_\phi S_\theta C_\psi + S_\phi S_\psi)T_z \\ C_\theta S_\psi T_x + (S_\phi S_\theta S_\psi + C_\phi C_\psi)T_y \\ \quad + (C_\phi S_\theta S_\psi - S_\phi C_\psi)T_z \\ -S_\theta T_x + S_\phi C_\theta T_y + C_\phi C_\theta T_z + mg \end{bmatrix}. \tag{62}$$

where $\boldsymbol{F}_{ext}$ is the external disturbance force acting on the vehicle. Neglecting effects such as friction, we set $\boldsymbol{F}_{ext} = 0$.

5. Controller Design and Control Allocation

5.1. Computed Torque Method

The computed torque method is a technique to determine the generalized force (comprising thrust and torque in the body coordinate system) from the desired translational and angular accelerations in the world coordinate system, given that the vehicle's physical information is known. Utilizing the relation $\dot{\boldsymbol{\eta}} = W_\eta^{-1}\boldsymbol{\omega}_B$ in the translational acceleration of Eq. (61) and the rotational angular acceleration of Eq. (57), and defining the generalized coordinates as $X \triangleq [\boldsymbol{\xi}^T\boldsymbol{\eta}^T]^T$, the system can be represented as

 doi:10.20944/preprints202603.0141.v1

$$\ddot{X} = F(\dot{X}) + G(X)\boldsymbol{u} + D(X)\boldsymbol{d} \tag{63}$$

This is because, from Eq. (37),

$$\begin{aligned}\ddot{\boldsymbol{\eta}} &= \frac{d}{dt}(W_\eta^{-1})\boldsymbol{\omega}_B + W_\eta^{-1}\dot{\boldsymbol{\omega}}_B \\ &= \frac{d}{dt}(W_\eta^{-1})W_\eta\dot{\boldsymbol{\eta}} + W_\eta^{-1}\left\{I_B^{-1}[(\tau_B + \tau_c) - \tau_{cori} + \tau_{ext}]\right\} \\ &= \frac{d}{dt}(W_\eta^{-1})W_\eta\dot{\boldsymbol{\eta}} - (I_B W_\eta)^{-1}[(W_\eta\dot{\boldsymbol{\eta}}) \times I_B(W_\eta\dot{\boldsymbol{\eta}})] + (I_B W_\eta)^{-1}(\tau_B + \tau_c) + (I_B W_\eta)^{-1}\tau_{ext}\end{aligned} \tag{64}$$

and thus

$$F(\dot{X}) = \begin{bmatrix} \boldsymbol{g} \\ \frac{d}{dt}(W_\eta^{-1})W_\eta\dot{\boldsymbol{\eta}} - (I_B W_\eta)^{-1}[(W_\eta\dot{\boldsymbol{\eta}}) \times I_B(W_\eta\dot{\boldsymbol{\eta}})] \end{bmatrix}$$

$$G(X) = \begin{bmatrix} \frac{1}{m}{}^W R_B & 0 \\ 0 & (I_B W_\eta)^{-1} \end{bmatrix}, \quad D(X) = \begin{bmatrix} \frac{1}{m} I_{3\times 3} & 0 \\ 0 & (I_B W_\eta)^{-1} \end{bmatrix}$$

$$\boldsymbol{u} = [T_B^T \quad (\tau_B + \tau_c)^T]^T, \quad \boldsymbol{d} = [F_{ext}^T \quad \tau_{ext}^T]^T$$

Here, since the rotation matrix ${}^W R_B$ is orthogonal, ${}^W R_B^{-1} = {}^W R_B^T$. Also, as each principal moment of inertia $I_{B_{ii}} \neq 0$ $(i = x, y, z)$, $G^{-1}(X) = \mathrm{diag}({}^m W R_B^T, W_\eta I_B)$. Then, the inverse system of Eq. (63) is

$$\boldsymbol{u} = G^{-1}(X)[\ddot{X} - F(\dot{X}) - D(X)\boldsymbol{d}] \tag{65}$$

Let the target value for the generalized coordinate vector X be X_d. Constructing its augmented acceleration vector $\ddot{X}^*$ using a PD servo system yields

$$\ddot{X}^* = \ddot{X}_d + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e} \tag{66}$$

where $\boldsymbol{e} = X_d - X$, $K_d > 0$ is the derivative gain, and $K_p > 0$ is the proportional gain. Substituting $\ddot{X}^*$ from Eq. (66) for $\ddot{X}$ in Eq. (65) gives

$$\boldsymbol{u}^* = G^{-1}(X)[\ddot{X}_d + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e} - F(\dot{X}) - D(X)\boldsymbol{d}] \tag{67}$$

Substituting this $\boldsymbol{u}^*$ into the original plant equation (63) yields

$$\ddot{\boldsymbol{e}} + K_d\dot{\boldsymbol{e}} + K_p\boldsymbol{e} = 0 \tag{68}$$

Here, we set $K_p = \mathrm{diag}(K_{p1}, ..., K_{p6})$ and $K_d = \mathrm{diag}(K_{d1}, ..., K_{d6})$. For application to actual vehicles, considering modeling errors, the condition $K_{di} = 2\sqrt{K_{pi}}$, which ideally achieves a damping ratio of 1 (i.e., critical damping), is used as a base, but the damping condition for this error may need slight modification. In the simulations, the proportional gains for translational position and attitude angles are defined as $K_p = \mathrm{diag}([K_{px}, K_{py}, K_{pz}, K_{p\phi}, K_{p\theta}, K_{p\psi})$, and the derivative gains as $K_d = \mathrm{diag}([K_{dx}, K_{dy}, K_{dz}, K_{d\phi}, K_{d\theta}, K_{d\psi})$, with the following settings:

$$\begin{aligned} K_{px} = K_{py} = 3.0; \quad K_{pz} = 5; \quad K_{p\phi} = K_{p\theta} = K_{p\psi} = 2.0 \\ K_{dx} = K_{dy} = 3.46; \quad K_{dz} = 5; \quad K_{d\phi} = K_{d\theta} = K_{d\psi} = 2.83 \end{aligned}$$

 doi:10.20944/preprints202603.0141.v1

5.2. Control Input Allocation Problem

Appropriate rotor speeds and tilt angles are allocated to each coaxial rotor to achieve the generalized force in the body coordinate system, determined earlier by the computed torque method. A single coaxial rotor constructed in this study utilizes two brushless motors. The thrust of coaxial rotor 1 is $T_1 = k_f(\bar{\omega}_1^2 + \bar{\omega}_2^2)$, and that of coaxial rotor 2 is $T_2 = k_f(\bar{\omega}_3^2 + \bar{\omega}_4^2)$. The reaction torque for the first rotor is $\tau_{c,1} = k_t(\bar{\omega}_2^2 - \bar{\omega}_1^2)$, and for the second rotor is $\tau_{c,2} = k_t(\bar{\omega}_3^2 - \bar{\omega}_4^2)$. However, directly solving for the rotation speeds $\bar{\omega}_1^2 \sim \bar{\omega}_4^2$ and tilt angles $\alpha_i, \beta_i, (i = 1, 2)$ results in an underdetermined problem: 6 generalized forces versus 8 unknown decision variables. This would yield an effectiveness matrix of size 6×8, requiring solution via pseudo-inverse or numerical optimization such as constrained QP. Below, we present a simpler method based on experimental results for coaxial rotor thrust.

Referring to the experimental results on coaxial rotors by Itakura et al. [26], for a total thrust of $T_{total} = 10.6$ N of the coaxial rotor, the thrust of the upstream propeller alone was $T_{upstream} = 7.3$ N, and that of the downstream propeller alone was $T_{downstream} = 8.0$ N. Therefore, the thrust efficiency is

$$\begin{aligned} \eta_{thrust} &= \frac{T_{total}}{T_{upstream} + T_{downstream}} \\ &= \frac{10.6}{7.3 + 8.0} = 0.693. \end{aligned} \tag{69}$$

This serves as an indicator of deviation from theoretical ideal conditions. On the other hand, the interference efficiency, which is the increase rate of total thrust relative to the upstream propeller's thrust, is

$$\begin{aligned} \eta_{if} &= \frac{T_{total}}{T_{upstream}} - 1 \\ &= \frac{10.6}{7.3} - 1 = 0.452. \end{aligned} \tag{70}$$

Introducing this interference efficiency allows substitution of $\bar{\omega}_2^2$ with $\eta_{if}\bar{\omega}_1^2$ and $\bar{\omega}_4^2$ with $\eta_{if}\bar{\omega}_3^2$, thereby eliminating $\bar{\omega}_2^2$ and $\bar{\omega}_4^2$ from the unknown variables.

With this interference efficiency, the reaction torque $\boldsymbol{\tau}_c$ from Eq. (54) becomes

$$\boldsymbol{\tau}_c = (\eta_{if} - 1)k_t \begin{bmatrix} C_{\alpha_1}S_{\beta_1}\bar{\omega}_1^2 - C_{\alpha_2}S_{\beta_2}\bar{\omega}_3^2 \\ -S_{\alpha_1}\bar{\omega}_1^2 + S_{\alpha_2}\bar{\omega}_3^2 \\ C_{\alpha_1}C_{\beta_1}\bar{\omega}_1^2 - C_{\alpha_2}C_{\beta_2}\bar{\omega}_3^2 \end{bmatrix}. \tag{71}$$

Meanwhile, the rotational torque due to thrust $\boldsymbol{\tau}_B$ from Eq. (52) becomes

$$\boldsymbol{\tau}_B = (\eta_{if} + 1)k_f \begin{bmatrix} (lC_{\alpha_1}C_{\beta_1} + h_oS_{\alpha_1})\bar{\omega}_1^2 \\ \quad - (lC_{\alpha_2}C_{\beta_2} - h_oS_{\alpha_2})\bar{\omega}_3^2 \\ h_oC_{\alpha_1}S_{\beta_1}\bar{\omega}_1^2 + h_oC_{\alpha_2}S_{\beta_2}\bar{\omega}_3^2 \\ -lC_{\alpha_1}S_{\beta_1}\bar{\omega}_1^2 + lC_{\alpha_2}S_{\beta_2}\bar{\omega}_3^2 \end{bmatrix}. \tag{72}$$

Similarly, the thrust $\boldsymbol{T}_B$ from Eq. (60) is rewritten as

$$\boldsymbol{T}_B = (\eta_{if} + 1)k_f \begin{bmatrix} -C_{\alpha_1}S_{\beta_1}\bar{\omega}_1^2 - C_{\alpha_2}S_{\beta_2}\bar{\omega}_3^2 \\ S_{\alpha_1}\bar{\omega}_1^2 + S_{\alpha_2}\bar{\omega}_3^2 \\ -C_{\alpha_1}C_{\beta_1}\bar{\omega}_1^2 - C_{\alpha_2}C_{\beta_2}\bar{\omega}_3^2 \end{bmatrix}. \tag{73}$$

Therefore, by introducing an intermediate 6-dimensional vector $\boldsymbol{n} = [n_{1a}\ n_{1b}\ n_{1c}\ n_{2a}\ n_{2b}\ n_{2c}]^T$, defining $\Omega_1 \triangleq \bar{\omega}_1^2$, $\Omega_2 \triangleq \bar{\omega}_3^2$, and for $i = 1, 2$,

 doi:10.20944/preprints202603.0141.v1

$$\begin{cases} n_{ia} = \Omega_i C_{\alpha_i} S_{\beta_i} \\ n_{ib} = \Omega_i S_{\alpha_i} \\ n_{ic} = \Omega_i C_{\alpha_i} C_{\beta_i} \end{cases}, \tag{74}$$

the relation $[T_B^T \ (\boldsymbol{\tau}_B + \boldsymbol{\tau}_c)^T]^T$ finally reduces to a linear equation:

$$\begin{bmatrix} \boldsymbol{T}_B \\ \boldsymbol{\tau}_B + \boldsymbol{\tau}_c \end{bmatrix} = A\boldsymbol{n}(\Omega_i, \alpha_i, \beta_i). \tag{75}$$

Here, the effectiveness matrix A is

$$\boldsymbol{A} = \begin{bmatrix} -(1+\eta_{if})k_f & 0 & 0 & -(1+\eta_{if})k_f & 0 & 0 \\ 0 & (1+\eta_{if})k_f & 0 & 0 & (1+\eta_{if})k_f & 0 \\ 0 & 0 & -(\eta_{if}+1)k_f & 0 & 0 & -(1+\eta_{if})k_f \\ (\eta_{if}-1)k_t & (1+\eta_{if})h_o k_f & (1+\eta_{if})lk_f & -(1-\eta_{if})k_t & (1+\eta_{if})h_o k_t & -(1+\eta_{if})lk_f \\ (1+\eta_{if})h_o k_f & (1-\eta_{if})k_t & 0 & (1+\eta_{if})h_o k_f & (\eta_{if}-1)k_t & 0 \\ -(1+\eta_{if})lk_f & 0 & (\eta_{if}-1)k_t & (1+\eta_{if})k_f & 0 & (1-\eta_{if})k_t \end{bmatrix}. \tag{76}$$

To generate control inputs for the actual vehicle, the intermediate variable vector $\boldsymbol{n}$ is obtained using A^{-1}, and then the squared rotation speeds $\Omega_1 = \bar{\omega}_1^2$, $\Omega_2 = \bar{\omega}_3^2$ and tilt angles (α_1, β_1), (α_2, β_2) for each rotor are generated via the following relations:

$$\begin{cases} \Omega_i = \sqrt{n_{ia}^2 + n_{ib}^2 + n_{ic}^2} \\ \alpha_i = \arcsin(n_{ib}/\Omega_i) \\ \beta_i = \arctan(n_{ia}/n_{ic}) \end{cases} \tag{77}$$

Figure 2 illustrates the sequential calculation flow for this control allocation problem.

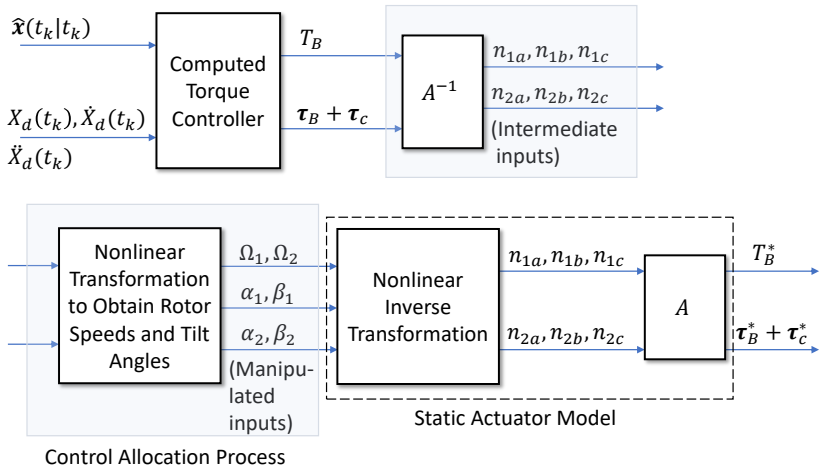


Figure 2. Control allocation process.

6. Preliminary Simulation: Application to the Falling Body Model

This section describes state estimation for the falling body model [21] as a preliminary experiment. The falling body model, often used as a nonlinear model, was also employed as a simulation model by Takeno and Katayama [9]. The aim here is to compare with previous research by performing state estimation under similar conditions. Since the RK method by Takeno and Katayama is a 4th-order discretization method, this section discusses the 4th-order AB method for comparison.

 doi:10.20944/preprints202603.0141.v1

6.1. Model Overview

The differential equations for the falling body model are as follows [21].

$$\dot{x}_1 = x_2 \tag{78}$$

$$\dot{x}_2 = \frac{\rho_0 e^{-x_1/k_\rho} x_2^2}{2\beta} - g \tag{79}$$

Here, x_1: altitude of the falling object, g: gravitational acceleration, ρ_0: atmospheric density at sea level 0 m, k_ρ: decay coefficient, β: ballistic coefficient.

Defining the extended state vector as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \beta \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \text{(3-dimensional state vector)}$$

and letting $\boldsymbol{f} = [f_1\ f_2\ f_3]^T$, the state-space model becomes

$$\frac{dx_1}{dt} \triangleq f_1 = x_2 \tag{80}$$

$$\frac{dx_2}{dt} \triangleq f_2 = \frac{\rho_0}{2x_3} e^{-x_1/k_\rho} x_2^2 - g \tag{81}$$

$$\frac{dx_3}{dt} \triangleq f_3 = 0 \tag{82}$$

The observation equation is assumed to be nonlinear as shown below:

$$y = \sqrt{M_1^2 + (x_1 - M_2)^2} + v \tag{83}$$

Here, M_1 is the horizontal distance between the falling object and the radar, M_2 is the radar height. Also, v is observation noise, which is Gaussian white noise with mean 0 and variance R.

6.2. Estimation Algorithms

We present the time-update equations for the sigma point matrix $\mathcal{X}$ when applying UKF with the Euler, RK, and AB methods. Hereafter, the UKF using the Euler method is called "Euler-UKF", that using the RK method "RK-UKF", and that using the AB method "AB-UKF".

6.2.1. Euler-UKF

Applying the Euler method to Eqs. (80)–(82) yields the following sigma point time-update equations:

$$\mathcal{X}^-_{1,j,k+1} = \mathcal{X}_{1,j,k} + h\mathcal{X}_{2,j,k} \tag{84}$$

$$\mathcal{X}^-_{2,j,k+1} = \mathcal{X}_{2,j,k} + h\left(\frac{\rho_0}{2\mathcal{X}_{3,j,k}} e^{-\mathcal{X}_{1,j,k}/k_\rho} \mathcal{X}^2_{2,j,k} - g\right) \tag{85}$$

$$\mathcal{X}^-_{3,j,k+1} = \mathcal{X}_{3,j,k} \tag{86}$$

Then, for the time-updated sigma points from Eqs. (84)–(86), the predicted state estimate is obtained by taking the weighted average via Eq. (9), and its covariance matrix via Eq. (10). The observation update proceeds similarly.

6.2.2. RK-UKF

Applying the RK method to Eqs. (80)–(82) and defining

 doi:10.20944/preprints202603.0141.v1

$$c_1 = \mathcal{X}_{2,j,k} \tag{87}$$
$$q_1 = \frac{\rho_0}{2\mathcal{X}_{3,j,k}} e^{-\mathcal{X}_{1,j,k}/k_\rho} \mathcal{X}_{2,j,k}^2 - g \tag{88}$$
$$c_2 = \mathcal{X}_{2,j,k} + \frac{h}{2} q_1 \tag{89}$$
$$q_2 = \frac{\rho_0}{2\mathcal{X}_{3,j,k}} e^{-(\mathcal{X}_{1,j,k}+hc_1/2)/k_\rho} (\mathcal{X}_{2,j,k} + \frac{h}{2} q_1)^2 - g \tag{90}$$
$$c_3 = \mathcal{X}_{2,j,k} + \frac{h}{2} q_2 \tag{91}$$
$$q_3 = \frac{\rho_0}{2\mathcal{X}_{3,j,k}} e^{-(\mathcal{X}_{1,j,k}+hc_2/2)/k_\rho} (\mathcal{X}_{2,j,k} + \frac{h}{2} q_2)^2 - g \tag{92}$$
$$c_4 = \mathcal{X}_{2,j,k} + h q_3 \tag{93}$$
$$q_4 = \frac{\rho_0}{2\mathcal{X}_{3,j,k}} e^{-(\mathcal{X}_{1,j,k}+hc_3/2)/k_\rho} (\mathcal{X}_{2,j,k} + \frac{h}{2} q_3)^2 - g \tag{94}$$

yields the sigma point time-update equations as follows [9]:

$$\mathcal{X}^-_{1,j,k+1} = \mathcal{X}_{1,j,k} + \frac{h}{6}(c_1 + 2c_2 + 2c_3 + c_4) \tag{95}$$
$$\mathcal{X}^-_{2,j,k+1} = \mathcal{X}_{2,j,k} + \frac{h}{6}(q_1 + 2q_2 + 2q_3 + q_4) \tag{96}$$
$$\mathcal{X}^-_{3,j,k+1} = \mathcal{X}_{3,j,k} \tag{97}$$
$$(j = 1, \ldots, 7)$$

Similar to the Euler method, the predicted state estimate and its covariance matrix are obtained from the time-updated sigma points in Eqs. (95)–(97) via Eqs. (9) and (10). The observation update proceeds similarly.

6.2.3. AB-UKF

Combining Eqs. (80)–(82), we redefine the nonlinear dynamics $f_c(x)$ of the continuous-time state equation as

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{\rho_0}{2x_3} e^{-x_1/k_\rho} x_2^2 - g \\ 0 \end{bmatrix} \triangleq f_c(x). \tag{98}$$

Then, the sigma point matrix $\mathcal{X}_k$ at time k and the matrix $\mathcal{F}_c(\mathcal{X}_k)$ obtained by substituting $\mathcal{X}_k$ into $f_c(x)$ are represented by the following 3×7 matrices:

$$\mathcal{X}_k = \begin{bmatrix} \mathcal{X}_{1,1,k} & \cdots & \mathcal{X}_{1,7,k} \\ \mathcal{X}_{2,1,k} & \cdots & \mathcal{X}_{2,7,k} \\ \mathcal{X}_{3,1,k} & \cdots & \mathcal{X}_{3,7,k} \end{bmatrix} \tag{99}$$

$$\begin{aligned} \mathcal{F}_c(\mathcal{X}_k) &= \begin{bmatrix} f_1(\mathcal{X}_{1,1,k}) & \cdots & f_1(\mathcal{X}_{1,7,k}) \\ f_2(\mathcal{X}_{2,1,k}) & \cdots & f_2(\mathcal{X}_{2,7,k}) \\ f_3(\mathcal{X}_{3,1,k}) & \cdots & f_3(\mathcal{X}_{3,7,k}) \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{X}_{2,1,k} & \cdots & \mathcal{X}_{2,7,k} \\ \frac{\rho_0}{2\mathcal{X}_{3,1,k}} e^{-\mathcal{X}_{1,1,k}/k_\rho} \mathcal{X}_{2,1,k}^2 - g & \cdots & \frac{\rho_0}{2\mathcal{X}_{3,7,k}} e^{-\mathcal{X}_{1,7,k}/k_\rho} \mathcal{X}_{2,7,k}^2 - g \\ 0 & \cdots & 0 \end{bmatrix}. \end{aligned} \tag{100}$$

 doi:10.20944/preprints202603.0141.v1

To distinguish this $\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$ specific to the falling body model from the general one in Eq. (26), we redefine the above as $\boldsymbol{\mathcal{F}}_c^{fb}(\mathcal{X}_k) := \boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$. Then, the 4th-order AB method yields the sigma point update equation:

$$\mathcal{X}_{k+1}^{-} = \mathcal{X}_k + \frac{h}{24}\left(55\boldsymbol{\mathcal{F}}_c^{fb}(\mathcal{X}_k) - 59\boldsymbol{\mathcal{F}}_c^{fb}(\mathcal{X}_{k-1})\right. \tag{101}$$

where the superscript fb denotes the falling body model.

6.3. Estimation Conditions

State estimation simulations are performed using the three methods described above. Physical parameters are: $g = 9.8$ [m/s^2], $\rho_0 = 2.202$ [kg·s^2/m^4], k_ρ=1000/0.1558 [m]. The true ballistic coefficient is $\beta = 2000$ [kg/m^2]. The horizontal distance between the falling object and radar is $M_1 = 10000$ [m], and the radar height is $M_2 = 0$ [m]. The step size is $h = 0.1$ [s], total number of sampling points $N = 300$, UT parameter $\lambda = 2.0$. The initial state vector, system noise covariance, and observation noise covariance are set as

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 40000 \text{ m} \\ -3000 \text{ m/s} \\ 2000 \text{ kg/m}^2 \end{bmatrix} \tag{102}$$

$$Q = \text{diag}(0, 0, 10 \text{ kg}^2/\text{m}^4) \tag{103}$$

$$R = 3600 \text{ m}^2 \tag{104}$$

The initial state estimate and its covariance matrix are

$$\hat{\boldsymbol{x}}(0|0) = \begin{bmatrix} 42000 \text{ m} \\ -3100 \text{ m/s} \\ 3000 \text{ kg/m}^2 \end{bmatrix} \tag{105}$$

$$P(0|0) = \text{diag}(10000 \text{ m}^2, 10000 \text{ m}^2/\text{s}^2, 10000 \text{ kg}^2/\text{m}^4) \tag{106}$$

as used in [9].

6.4. Simulation Results

6.4.1. State Estimation Accuracy

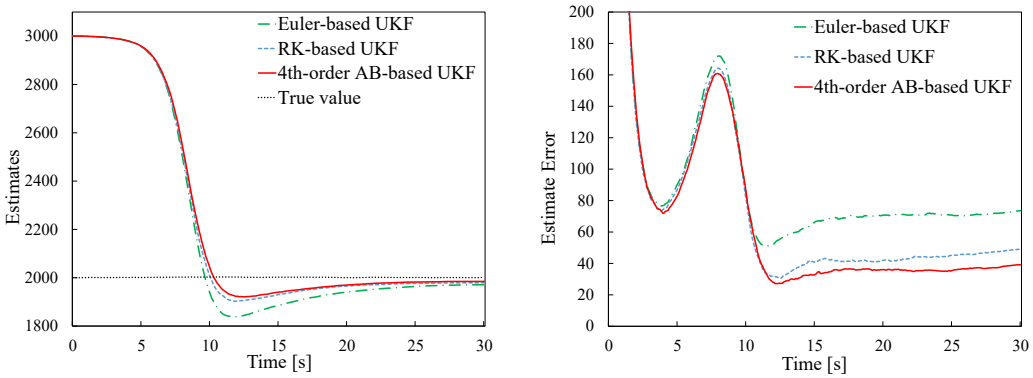
Figure 3 (a) shows the estimation results for the ballistic coefficient β, and (b) shows the time evolution of the state estimation error calculated using the root mean squared error (RMSE) formula,

$$E(t_k) = \sqrt{\sum_{i=1}^{2}(x_i(t_k) - \hat{x}_i(t_k|t_k))^2}, \tag{107}$$

for the three methods: Euler-UKF, RK-UKF, and AB-UKF. The results are averages from 100 simulation runs for each method.

Figure 3(a) shows that the parameter β converges well for all three methods: Euler-UKF, RK-UKF, and AB-UKF. On the other hand, Figure 3(b) indicates that after the parameter β converges as shown in Figure 3(a), the state estimation error is suppressed more effectively in the order of Euler-UKF, RK-UKF, and AB-UKF. Table 1 presents the average RMSE values of the state estimation error from 100 simulation runs for each of the three methods. These computations were performed on a laptop with 16 GB RAM, an AMD Ryzen7 4800H with Radeon Graphics CPU, under Windows 10, using MATLAB 2022a.

 doi:10.20944/preprints202603.0141.v1



(a) Estimation of parameter β (b) Estimation error of states

Figure 3. Comparison of estimation results.

Table 1. Comparison of RMSE among three methods.

	Method		
	Euler-based	RK-based	AB4-based
RMSE	125.585	116.826	115.537

As seen in Table 1, the Euler method has the largest RMSE, while the RK and AB methods are smaller than Euler's. This result is attributed to the difference in discretization order: Euler method is 1st-order accurate, while RK and AB methods are 4th-order accurate.

6.4.2. Computation Time

To more clearly demonstrate the differences between methods, the estimation conditions from Section 6.3 were modified: $h = 0.01$ and total sampling points $N = 5000$. Table 2 shows the comparison results for the total UKF algorithm computation time (from Eq. (3) to Eq. (19)) and the prediction-only computation time (Eq. (8)).

Table 2. Comparison of computational time among three methods.

	Method		
	Euler-based	RK-based	AB4-based
Alg. total time ($\times 10^{-2}$) [s]	8.4612	9.2542	8.7108
Pred. time only ($\times 10^{-6}$) [s]	58.3842	77.8698	70.6750

Table 2 shows that computation time is shortest for the Euler method, followed by the AB method, then the RK method. The prediction time for Euler's method is particularly short, a result stemming from the difference in discretization order.

6.5. Discussion

The results shown in Section 6.4 can be attributed to the following two potential issues.

1. The falling body model has a small number of state variables.

This is considered the most significant issue. Although the AB method was applied to the falling body model for comparison with prior work, this model has only three state variables, and x_3 represents a parameter estimation problem. Consequently, as evident from the sigma point time-update equations, the discretization formulas were actually applied only to state variables x_1 and x_2, making it a model where differences between the RK and AB methods are less likely to manifest.

2. The UT occupies a small portion of the overall UKF algorithm.

This issue varies with each model, but for this falling body model, the following can be stated. As mentioned, the falling body model had relatively few state variables and simple discretization equations, making it a problem where computational load differences are less pronounced. Therefore, within the

 doi:10.20944/preprints202603.0141.v1

overall UKF estimation algorithm, the time spent on UT for this model constituted a small proportion, resulting in less noticeable differences in computation time due to the discretization method.

Considering the above two points, Section 7 addresses a state estimation problem for a UAV model, which is higher-dimensional than the falling body model.

7. Application to the Osprey-Type Drone

7.1. Model Overview

The UAV vehicle conditions are based on the Osprey-type drone with 2-DOF tiltable coaxial rotors by Itakura et al. [26]. The vehicle's general appearance and the derivation of its dynamic model have already been detailed in Section 4.

The vehicle's position in the world coordinate system, $\boldsymbol{\xi} = [x\ y\ z]^T$, follows the translational motion equation expressed by Eq. (61) or Eq. (62).

On the other hand, the vehicle's angular acceleration $\dot{\boldsymbol{\omega}}_B = [\dot{p}\ \dot{q}\ \dot{r}]^T$ is given by Eq. (57) or Eq. (58). Assuming the Euler angle variations are relatively small, i.e., $W_\eta \simeq I$, then $p \simeq \dot{\phi}, q \simeq \dot{\theta}, r \simeq \dot{\psi}$. Therefore, Eq. (58) can be rewritten as:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{Bxx}}(\tau_x^B + \tau_x^c) + \dot{\theta}\dot{\psi}\left(\frac{I_{Byy}-I_{Bzz}}{I_{Bxx}}\right) \\ \frac{1}{I_{Byy}}(\tau_y^B + \tau_y^c) + \dot{\phi}\dot{\psi}\left(\frac{I_{Bzz}-I_{Bxx}}{I_{Byy}}\right) \\ \frac{1}{I_{Bzz}}(\tau_z^B + \tau_z^c) + \dot{\phi}\dot{\theta}\left(\frac{I_{Bxx}-I_{Byy}}{I_{Bzz}}\right) \end{bmatrix} \tag{108}$$

Now, defining a 12-dimensional state vector as $\boldsymbol{x} = [x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}]^T = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}]^T$, the continuous-time state equation for the UAV model can be derived from Eqs. (62) and (108) as follows:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} x_2 \\ A_1(x)T_x + A_2(x)T_y + A_3(x)T_z \\ x_4 \\ A_4(x)T_x + A_5(x)T_y + A_6(x)T_z \\ x_6 \\ A_7(x)T_x + A_8(x)T_y + A_9(x)T_z + g \\ x_8 \\ \frac{1}{I_{Bxx}}(\tau_x^B + \tau_x^c) + x_{10}x_{12}\left(\frac{I_{Byy}-I_{Bzz}}{I_{Bxx}}\right) \\ x_{10} \\ \frac{1}{I_{Byy}}(\tau_y^B + \tau_y^c) + x_8x_{12}\left(\frac{I_{Bzz}-I_{Bxx}}{I_{Byy}}\right) \\ x_{12} \\ \frac{1}{I_{Bzz}}(\tau_z^B + \tau_z^c) + x_8x_{10}\left(\frac{I_{Bxx}-I_{Byy}}{I_{Bzz}}\right) \end{bmatrix} \tag{109}$$

Here, $A_1(x), A_2(x), \cdots, A_9(x)$ in Eq. (109) are defined as follows:

$$\begin{cases} A_1(x) = C_{x_9}C_{x_{11}}/m \\ A_2(x) = (S_{x_7}S_{x_9}C_{x_{11}} - C_{x_7}S_{x_{11}})/m \\ A_3(x) = (C_{x_7}S_{x_9}C_{x_{11}} + S_{x_7}S_{x_{11}})/m \\ A_4(x) = C_{x_9}S_{x_{11}}/m \\ A_5(x) = (S_{x_7}S_{x_9}S_{x_{11}} + C_{x_7}C_{x_{11}})/m \\ A_6(x) = (C_{x_7}S_{x_9}C_{x_{11}} - S_{x_7}C_{x_{11}})/m \\ A_7(x) = -S_{x_9}/m \\ A_8(x) = S_{x_7}C_{x_9}/m \\ A_9(x) = C_{x_7}C_{x_9}/m \end{cases} \tag{110}$$

 doi:10.20944/preprints202603.0141.v1

7.2. Estimation Algorithms

Sections 7.2.1 to 7.2.3 present the time-update equations for sigma points when applying the Euler method, RK method, and AB method, respectively, to the UT within the UAV model, similar to the falling body model.

7.2.1. Euler-Type UKF

Applying the Euler method to Eq. (109) yields the following discrete-time state equation:

$$\begin{aligned}
&\begin{bmatrix} x_{1,k+1} & x_{2,k+1} & \cdots & x_{11,k+1} & x_{12,k+1} \end{bmatrix}^T \\
&= \begin{bmatrix}
x_1 + hx_2 \\
x_2 + h\Big(A_1(x)T_x + A_2(x)T_y + A_3(x)T_z\Big) \\
x_3 + hx_4 \\
x_4 + h\Big(A_4(x)T_x + A_5(x)T_y + A_6(x)T_z\Big) \\
x_5 + hx_6 \\
x_6 + h\Big(A_7(x)T_x + A_8(x)T_y + A_9(x)T_z + g\Big) \\
x_7 + hx_8 \\
x_8 + h\left(\frac{1}{I_{Bxx}}(\tau_x^B + \tau_x^c) + x_{10}x_{12}\left(\frac{I_{Byy} - I_{Bzz}}{I_{Bxx}}\right)\right) \\
x_9 + hx_{10} \\
x_{10} + h\left(\frac{1}{I_{Byy}}(\tau_y^B + \tau_y^c) + x_8x_{12}\left(\frac{I_{Bzz} - I_{Bxx}}{I_{Byy}}\right)\right) \\
x_{11} + hx_{12} \\
x_{12} + h\left(\frac{1}{I_{Bzz}}(\tau_z^B + \tau_z^c) + x_8x_{10}\left(\frac{I_{Bxx} - I_{Byy}}{I_{Bzz}}\right)\right)
\end{bmatrix}
\end{aligned} \tag{111}$$

Therefore, by substituting $\mathcal{X}_k$ into Eq. (111), the time-update equation for the sigma points is derived as:

$$\mathcal{X}^-_{1,j,k+1} = \mathcal{X}_{1,j,k} + h\mathcal{X}_{2,j,k} \tag{112}$$

$$\mathcal{X}^-_{2,j,k+1} = \mathcal{X}_{2,j,k} + h\Big(A_1(\mathcal{X}_{(9,11),j,k})T_x + A_2(\mathcal{X}_{(7,9,11),j,k})T_y + A_3(\mathcal{X}_{(7,9,11),j,k})T_z\Big) \tag{113}$$

$$\mathcal{X}^-_{3,j,k+1} = \mathcal{X}_{3,j,k} + h\mathcal{X}_{4,j,k} \tag{114}$$

$$\mathcal{X}^-_{4,j,k+1} = \mathcal{X}_{4,j,k} + h\Big(A_4(\mathcal{X}_{(9,11),j,k})T_x + A_5(\mathcal{X}_{(7,9,11),j,k})T_y + A_6(\mathcal{X}_{(7,9,11),j,k})T_z\Big) \tag{115}$$

$$\mathcal{X}^-_{5,j,k+1} = \mathcal{X}_{5,j,k} + h\mathcal{X}_{6,j,k} \tag{116}$$

$$\mathcal{X}^-_{6,j,k+1} = \mathcal{X}_{6,j,k} + h\Big(A_7(\mathcal{X}_{9,j,k})T_x + A_8(\mathcal{X}_{(7,9),j,k})T_y + A_9(\mathcal{X}_{(7,9),j,k})T_z + g\Big) \tag{117}$$

$$\mathcal{X}^-_{7,j,k+1} = \mathcal{X}_{7,j,k} + h\mathcal{X}_{8,j,k} \tag{118}$$

$$\mathcal{X}^-_{8,j,k+1} = \mathcal{X}_{8,j,k} + h\left(\frac{1}{I_{Bxx}}(\tau_x^B + \tau_x^c) + \mathcal{X}_{10,j,k}\mathcal{X}_{12,j,k}\left(\frac{I_{Byy} - I_{Bzz}}{I_{Bxx}}\right)\right) \tag{119}$$

$$\mathcal{X}^-_{9,j,k+1} = \mathcal{X}_{9,j,k} + h\mathcal{X}_{10,j,k} \tag{120}$$

$$\mathcal{X}^-_{10,j,k+1} = \mathcal{X}_{10,j,k} + h\left(\frac{1}{I_{Byy}}(\tau_y^B + \tau_y^c) + \mathcal{X}_{8,j,k}\mathcal{X}_{12,j,k}\left(\frac{I_{Bzz} - I_{Bxx}}{I_{Byy}}\right)\right) \tag{121}$$

$$\mathcal{X}^-_{11,j,k+1} = \mathcal{X}_{11,j,k} + h\mathcal{X}_{12,j,k} \tag{122}$$

$$\mathcal{X}^-_{12,j,k+1} = \mathcal{X}_{12,j,k} + h\left(\frac{1}{I_{Bzz}}(\tau_z^B + \tau_z^c) + \mathcal{X}_{8,j,k}\mathcal{X}_{10,j,k}\left(\frac{I_{Bxx} - I_{Byy}}{I_{Bzz}}\right)\right) \tag{123}$$

 doi:10.20944/preprints202603.0141.v1

7.2.2. RK-Type UKF

First, regarding Eq. (109), define $\boldsymbol{f}_c = [f_1(x)\ f_2(x)\cdots f_{12}(x)]^T$. Then, define the derivatives $a_1(x)$–$a_4(x)$,$\cdots$,$l_1(x)$–$l_4(x)$ for each state variable as follows. Note that some equations are omitted due to their considerable number.

$$a_1(x) = f_1(x_2) \tag{124}$$
$$a_2(x) = f_1(x_2 + \frac{b_1}{2}) \tag{125}$$
$$a_3(x) = f_1(x_2 + \frac{b_2}{2}) \tag{126}$$
$$a_4(x) = f_1(x_2 + b_3) \tag{127}$$
$$\vdots$$
$$l_1(x) = f_{12}(x_8, x_{10}) \tag{128}$$
$$l_2(x) = f_{12}(x_8 + \frac{h_1}{2}, x_{10} + \frac{j_1}{2}) \tag{129}$$
$$l_3(x) = f_{12}(x_8 + \frac{h_2}{2}, x_{10} + \frac{j_2}{2}) \tag{130}$$
$$l_4(x) = f_{12}(x_8 + h_3, x_{10} + j_3) \tag{131}$$

Thus, applying the RK method to Eq. (109) yields the discrete-time state equation using $a_1(x)$–$a_4(x)$, $\cdots$,$l_1(x)$–$l_4(x)$:

$$\begin{aligned}
&\begin{bmatrix} x_{1,k+1}\ x_{2,k+1}\cdots x_{11,k+1}\ x_{12,k+1} \end{bmatrix}^T \\
&= \begin{bmatrix}
x_{1,k} + \frac{h}{6}\Big(a_1(x) + 2a_2(x) + 2a_3(x) + a_4(x)\Big) \\
x_{2,k} + \frac{h}{6}\Big(b_1(x) + 2b_2(x) + 2b_3(x) + b_4(x)\Big) \\
x_{3,k} + \frac{h}{6}\Big(c_1(x) + 2c_2(x) + 2c_3(x) + c_4(x)\Big) \\
x_{4,k} + \frac{h}{6}\Big(d_1(x) + 2d_2(x) + 2d_3(x) + d_4(x)\Big) \\
x_{5,k} + \frac{h}{6}\Big(e_1(x) + 2e_2(x) + 2e_3(x) + e_4(x)\Big) \\
x_{6,k} + \frac{h}{6}\Big(f_1(x) + 2f_2(x) + 2f_3(x) + f_4(x)\Big) \\
x_{7,k} + \frac{h}{6}\Big(g_1(x) + 2g_2(x) + 2g_3(x) + g_4(x)\Big) \\
x_{8,k} + \frac{h}{6}\Big(h_1(x) + 2h_2(x) + 2h_3(x) + h_4(x)\Big) \\
x_{9,k} + \frac{h}{6}\Big(i_1(x) + 2i_2(x) + 2i_3(x) + i_4(x)\Big) \\
x_{10,k} + \frac{h}{6}\Big(j_1(x) + 2j_2(x) + 2j_3(x) + j_4(x)\Big) \\
x_{11,k} + \frac{h}{6}\Big(k_1(x) + 2k_2(x) + 2k_3(x) + k_4(x)\Big) \\
x_{12,k} + \frac{h}{6}\Big(l_1(x) + 2l_2(x) + 2l_3(x) + l_4(x)\Big)
\end{bmatrix}
\end{aligned} \tag{132}$$

Therefore, by substituting $\mathcal{X}_k$ into Eq. (111), the time-update equation for the sigma points is derived as:

$$\mathcal{X}^-_{1,j,k+1} = \mathcal{X}_{1,j,k} + \frac{h}{6}\Big(a_1(\mathcal{X}_{2,j,k}) + 2a_2(\mathcal{X}_{2,j,k}) + 2a_3(\mathcal{X}_{2,j,k}) + a_4(\mathcal{X}_{2,j,k})\Big) \tag{133}$$
$$\mathcal{X}^-_{2,j,k+1} = \mathcal{X}_{2,j,k} + \frac{h}{6}\Big(b_1(\mathcal{X}_{(7,9,11),j,k}) + 2b_2(\mathcal{X}_{(7,9,11),j,k}) + 2b_3(\mathcal{X}_{(7,9,11),j,k}) + b_4(\mathcal{X}_{(7,9,11),j,k})\Big) \tag{134}$$
$$\mathcal{X}^-_{3,j,k+1} = \mathcal{X}_{3,j,k} + \frac{h}{6}\Big(c_1(\mathcal{X}_{4,j,k}) + 2c_2(\mathcal{X}_{4,j,k}) + 2c_3(\mathcal{X}_{4,j,k}) + c_4(\mathcal{X}_{2,j,k})\Big) \tag{135}$$
$$\mathcal{X}^-_{4,j,k+1} = \mathcal{X}_{4,j,k} + \frac{h}{6}\Big(d_1(\mathcal{X}_{(7,9,11),j,k}) + 2d_2(\mathcal{X}_{(7,9,11),j,k}) + 2d_3(\mathcal{X}_{(7,9,11),j,k}) + d_4(\mathcal{X}_{(7,9,11),j,k})\Big) \tag{136}$$

 doi:10.20944/preprints202603.0141.v1

$$\mathcal{X}^{-}_{5,j,k+1} = \mathcal{X}_{5,j,k} + \frac{h}{6}\Big(e_1(\mathcal{X}_{6,j,k}) + 2e_2(\mathcal{X}_{6,j,k}) + 2e_3(\mathcal{X}_{6,j,k}) + e_4(\mathcal{X}_{6,j,k})\Big) \tag{137}$$

$$\mathcal{X}^{-}_{6,j,k+1} = \mathcal{X}_{6,j,k} + \frac{h}{6}\Big(f_1(\mathcal{X}_{(7,9,11),j,k}) + 2f_2(\mathcal{X}_{(7,9,11),j,k}) + 2f_3(\mathcal{X}_{(7,9,11),j,k}) + f_4(\mathcal{X}_{(7,9,11),j,k})\Big) \tag{138}$$

$$\mathcal{X}^{-}_{7,j,k+1} = \mathcal{X}_{7,j,k} + \frac{h}{6}\Big(g_1(\mathcal{X}_{8,j,k}) + 2g_2(\mathcal{X}_{8,j,k}) + 2g_3(\mathcal{X}_{8,j,k}) + g_4(\mathcal{X}_{8,j,k})\Big) \tag{139}$$

$$\mathcal{X}^{-}_{8,j,k+1} = \mathcal{X}_{8,j,k} + \frac{h}{6}\Big(l_1(\mathcal{X}_{(10,12),j,k}) + 2l_2(\mathcal{X}_{(10,12),j,k}) + 2l_3(\mathcal{X}_{(10,12),j,k}) + l_4(\mathcal{X}_{(10,12),j,k})\Big) \tag{140}$$

$$\mathcal{X}^{-}_{9,j,k+1} = \mathcal{X}_{9,j,k} + \frac{h}{6}\Big(i_1(\mathcal{X}_{10,j,k}) + 2i_2(\mathcal{X}_{10,j,k}) + 2i_3(\mathcal{X}_{10,j,k}) + i_4(\mathcal{X}_{10,j,k})\Big) \tag{141}$$

$$\mathcal{X}^{-}_{10,j,k+1} = \mathcal{X}_{10,j,k} + \frac{h}{6}\Big(j_1(\mathcal{X}_{(8,12),j,k}) + 2j_2(\mathcal{X}_{(8,12),j,k}) + 2j_3(\mathcal{X}_{(8,12),j,k}) + j_4(\mathcal{X}_{(8,12),j,k})\Big) \tag{142}$$

$$\mathcal{X}^{-}_{11,j,k+1} = \mathcal{X}_{11,j,k} + \frac{h}{6}\Big(k_1(\mathcal{X}_{12,j,k}) + 2k_2(\mathcal{X}_{12,j,k}) + 2k_3(\mathcal{X}_{12,j,k}) + k_4(\mathcal{X}_{12,j,k})\Big) \tag{143}$$

$$\mathcal{X}^{-}_{12,j,k+1} = \mathcal{X}_{12,j,k} + \frac{h}{6}\Big(l_1(\mathcal{X}_{(8,10),j,k}) + 2l_2(\mathcal{X}_{(8,10),j,k}) + 2l_3(\mathcal{X}_{(8,10),j,k}) + l_4(\mathcal{X}_{(8,10),j,k})\Big) \tag{144}$$

The derivatives are as follows (similarly, some are omitted):

$$a_1(\mathcal{X}_{2,j,k}) = f_1(\mathcal{X}_{2,j,k}) \tag{145}$$

$$a_2(\mathcal{X}_{2,j,k}) = f_1(\mathcal{X}_{2,j,k} + \frac{b_1}{2}) \tag{146}$$

$$a_3(\mathcal{X}_{2,j,k}) = f_1(\mathcal{X}_{2,j,k} + \frac{b_2}{2}) \tag{147}$$

$$a_4(\mathcal{X}_{2,j,k}) = f_1(\mathcal{X}_{2,j,k} + b_3) \tag{148}$$

$$\vdots$$

$$l_1(\mathcal{X}_{(8,10),j,k}) = f_{12}(\mathcal{X}_{8,j,k}, \mathcal{X}_{10,j,k},) \tag{149}$$

$$l_2(\mathcal{X}_{(8,10),j,k}) = f_{12}(\mathcal{X}_{8,j,k}, + \frac{h_1}{2}, \mathcal{X}_{10,j,k} + \frac{j_1}{2}), \tag{150}$$

$$l_3(\mathcal{X}_{(8,10),j,k}) = f_{12}(\mathcal{X}_{8,j,k}, + \frac{h_2}{2}, \mathcal{X}_{10,j,k} + \frac{j_2}{2}), \tag{151}$$

$$l_4(\mathcal{X}_{(8,10),j,k}) = f_{12}(\mathcal{X}_{8,j,k}, + h_3, \mathcal{X}_{10,j,k} + j_3) \tag{152}$$

7.2.3. AB-Type UKF

First, the sigma point matrix $\mathcal{X}_k$ at time k and the matrix $\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$ obtained by substituting $\mathcal{X}_k$ into Eq. (109) are represented by the following 12×25 matrix.

$$\mathcal{X}_k = \begin{bmatrix} \mathcal{X}_{1,1,k} & \cdots & \mathcal{X}_{1,25,k} \\ \vdots & \ddots & \vdots \\ \mathcal{X}_{12,1,k} & \cdots & \mathcal{X}_{12,25,k} \end{bmatrix} \tag{153}$$

$$\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k) = \begin{bmatrix} f_1(\mathcal{X}_{1,1,k}) & \cdots & f_1(\mathcal{X}_{1,25,k}) \\ \vdots & \ddots & \vdots \\ f_{12}(\mathcal{X}_{12,1,k}) & \cdots & f_{12}(\mathcal{X}_{12,25,k}) \end{bmatrix} \tag{154}$$

 doi:10.20944/preprints202603.0141.v1

Here, similar to Eq. (100), to distinguish this from the general $\boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$ in Eq. (26), we redefine the above as $\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) := \boldsymbol{\mathcal{F}}_c(\mathcal{X}_k)$. Then, the sigma point update equations using the 2nd to 6th-order AB methods for the UAV model are expressed as follows.

- Sigma point update equation using the 2nd-order Adams-Bashforth method:

$$\mathcal{X}_{k+1}^- = \mathcal{X}_k + \frac{h}{2}\left(3\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) - \boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-1})\right) \tag{155}$$

- Sigma point update equation using the 3rd-order Adams-Bashforth method:

$$\mathcal{X}_{k+1}^- = \mathcal{X}_k + \frac{h}{12}\left(23\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) - 16\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-1}) + 5\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-2})\right) \tag{156}$$

- Sigma point update equation using the 4th-order Adams-Bashforth method:

$$\mathcal{X}_{k+1}^- = \mathcal{X}_k + \frac{h}{24}\left(55\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) - 59\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-1}) + 37\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-2}) - 9\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-3})\right) \tag{157}$$

- Sigma point update equation using the 5th-order Adams-Bashforth method:

$$\mathcal{X}_{k+1}^- = \mathcal{X}_k + \frac{h}{720}\left(1901\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) - 2774\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-1}) + 2616\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-2}) - 1274\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-3}) + 251\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-4})\right) \tag{158}$$

- Sigma point update equation using the 6th-order Adams-Bashforth method:

$$\begin{aligned}\mathcal{X}_{k+1}^- = \mathcal{X}_k + \frac{h}{1440}\Big(&4277\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_k) - 7923\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-1}) + 9982\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-2}) - 7298\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-3}) + 2877\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-4}) \\ &- 475\boldsymbol{\mathcal{F}}_c^U(\mathcal{X}_{k-5})\Big)\end{aligned} \tag{159}$$

Here, the superscript U denotes the UAV model.

7.3. Description of Estimation Conditions

This section describes the UAV's physical parameters and the covariance of system and observation noise used in the estimation simulation, based on the UAV model and each estimation algorithm presented in Sections 7.1 and 7.2.

7.3.1. Parameter Settings for the UAV Model

Similar to the falling body model, state estimation simulations are performed using the three methods described above. First, Table 3 summarizes the definitions and values of each parameter.

Table 3. Simulation parameters of the UAV.

Variable	Definition	Value
m [kg]	Mass of UAV	1.5
g [m/s^2]	Acceleration of gravity	9.81
I_{Bxx} [kg·m^2]	Moment of inertia around X_B axis	0.01
I_{Byy} [kg·m^2]	Moment of inertia around Y_B axis	0.01
I_{Bzz} [kg·m^2]	Moment of inertia around Z_B axis	0.006
l [m]	Y_B axis distance between O_B and O_{P_i}	0.24
h_o [m]	Z_B axis distance between O_B and O_{P_i}	0.045
k_f [Ns^2/rad^2]	Thrust coefficient of the propeller	1.784×10^{-5}
k_t [Nms^2/rad^2]	Drag coefficient of the propeller	4.379×10^{-7}

All initial states are set to $\boldsymbol{x}(0) = [0...0]^T$, and the initial estimation information is set to $\hat{\boldsymbol{x}}(0|0) = [0...0]^T$ and $P(0|0) = \mathrm{diag}(1, ..., 1)$.

 doi:10.20944/preprints202603.0141.v1

7.3.2. Determination of System Noise

In this simulation experiment, the system noise is set based on the global truncation error arising from discretization by the Euler, RK, and AB methods. When the step size is denoted by h, the global truncation error for the Euler method is proportional to h^1, and for the RK and 4th-order AB methods, it is proportional to h^4. Therefore, letting the variance values of the respective system noises be Q_{Euler} and $Q_{RK,AB4}$, they can be expressed as:

$$Q_{Euler} = h \times I_{12\times 12} \tag{160}$$

$$Q_{RK,AB4} = h^4 \times I_{12\times 12} \tag{161}$$

Thus, for comparison purposes in this simulation experiment, the value Q is set as follows:

$$Q = h^{5/2} \times I_{12\times 12} \tag{162}$$

Here, $5/2$ in the above equation signifies taking an intermediate order of magnitude between Q_{Euler} and $Q_{RK,AB4}$.

7.3.3. Determination of Observation Noise

Typically, UAVs are equipped with GPS sensors for position observation and gyro sensors for attitude observation. Determining a precise noise value for GPS sensor error is difficult due to various influencing factors; thus, for simplicity, the noise value is set to 2 m here. For the gyro sensor, we assume the use of the MPU-9250, a 9-axis sensor module from InvenSense.

The MPU-9250 datasheet states a gyro sensor noise value of 0.01 [deg/$\sqrt{\text{Hz}}$]. Using this noise value and the step size h [s], the observation noise variance R_1 [rad^2] for attitude angles ϕ, θ, ψ and R_2 [rad^2/s^2] for attitude angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are determined.

First, convert the dimension 0.01 [deg/$\sqrt{\text{Hz}}$] to [rad/$\sqrt{\text{Hz}}$].

$$1.0 \times 10^{-2}\left(\frac{\pi}{180}\right) \quad [\text{rad}/\sqrt{\text{Hz}}] \tag{163}$$

Squaring the value obtained in Eq. (163) yields:

$$1.0 \times 10^{-4}\left(\frac{\pi}{180}\right)^2 \quad [\text{rad}^2/\text{Hz}] \tag{164}$$

Here, the dimension [Hz] can also be expressed as [1/s]; thus, the dimension [rad^2/Hz] can also be expressed as [rad$^2\cdot$ s]. Therefore, the observation noise variance R_1 [rad^2] for attitude angles ϕ, θ, ψ is obtained by dividing Eq. (164) by h [s]:

$$R_1 = 1.0 \times 10^{-4}\left(\frac{\pi}{180}\right)^2 / h \quad [\text{rad}^2] \tag{165}$$

Then, the observation noise variance R_2 [rad^2/s^2] for attitude angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$ is obtained by dividing Eq. (165) by h^2 [s]:

$$R_2 = 1.0 \times 10^{-4}\left(\frac{\pi}{180}\right)^2 / h^3 \quad [\text{rad}^2/\text{s}^2] \tag{166}$$

Consequently, the observation equation is

$$y(t_k) = Hx(t_k) + v(t_k) \tag{167}$$

$$H = \begin{bmatrix} H_{pos} & \mathbf{0} \\ \mathbf{0} & I_{6\times 6} \end{bmatrix} \tag{168}$$

 doi:10.20944/preprints202603.0141.v1

where H_{pos} is the observation matrix for position (x, y, z):

$$H_{pos} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{169}$$

and the covariance $\boldsymbol{R}$ of the observation noise $v(t_k)$ is represented by the following diagonal matrix:

$$\boldsymbol{R} = \text{diag}(\ 2\ \ 2\ \ 2\ \ R_1\ \ R_2\ \ R_1\ \ R_2\ \ R_1\ \ R_2\ ). \tag{170}$$

7.4. Target Trajectory

The simulation starts from the initial position $[x\ y\ z] = [0\ 0\ 0]^T$ and initial attitude angles $[\phi\ \theta\ \psi] = [0\ 0\ 0]^T$. First, over 10 s from the start time, the vehicle moves to the target position $[x_d\ y_d\ z_d] = [0\ 0\ 1]^T$, while all target attitude angles remain zero. Subsequently, from 10 s to 70 s, the vehicle is commanded to follow the target position, velocity, and acceleration given below [27].

Target position $\boldsymbol{X}_d$:

$$\begin{aligned} \boldsymbol{X}_d &= \begin{bmatrix} x_d(t)\ y_d(t)\ z_d(t)\ \phi_d(t)\ \theta_d(t)\ \psi_d(t) \end{bmatrix}^T \\ &= \begin{bmatrix} \sin(\frac{\pi}{10} \cdot (t-10)) \\ \cos(\frac{\pi}{4}) \Big(-1 + \cos(\frac{\pi}{10} \cdot (t-10)) \Big) \\ -1 + \sin(\frac{\pi}{4}) \Big(-1 + \cos(\frac{\pi}{10} \cdot (t-10)) \Big) \\ \frac{\pi}{4} \cdot \sin(\frac{\pi}{40} \cdot (t-10)) \\ -\frac{\pi}{4} \cdot \sin(\frac{\pi}{40} \cdot (t-10)) \\ 0 \end{bmatrix} \end{aligned} \tag{171}$$

Target velocity $\dot{\boldsymbol{X}}_d$:

$$\dot{\boldsymbol{X}}_d = \begin{bmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \\ \dot{z}_d(t) \\ \dot{\phi}_d(t) \\ \dot{\theta}_d(t) \\ \dot{\psi}_d(t) \end{bmatrix} = \begin{bmatrix} \frac{\pi}{10} \cos(\frac{\pi}{10} \cdot (t-10)) \\ -\frac{\pi}{10} \cos(\frac{\pi}{4}) \sin(\frac{\pi}{10} \cdot (t-10)) \\ -\frac{\pi}{10} \sin(\frac{\pi}{4}) \sin(\frac{\pi}{10} \cdot (t-10)) \\ \frac{\pi^2}{160} \cdot \cos(\frac{\pi}{40} \cdot (t-10)) \\ -\frac{\pi^2}{160} \cdot \cos(\frac{\pi}{40} \cdot (t-10)) \\ 0 \end{bmatrix} \tag{172}$$

Target acceleration $\ddot{\boldsymbol{X}}_d$:

$$\ddot{\boldsymbol{X}}_d = \begin{bmatrix} \ddot{x}_d(t) \\ \ddot{y}_d(t) \\ \ddot{z}_d(t) \\ \ddot{\phi}_d(t) \\ \ddot{\theta}_d(t) \\ \ddot{\psi}_d(t) \end{bmatrix} = \begin{bmatrix} -\frac{\pi^2}{100} \sin(\frac{\pi}{10} \cdot (t-10)) \\ -\frac{\pi^2}{100} \cos(\frac{\pi}{4}) \cos(\frac{\pi}{10} \cdot (t-10)) \\ -\frac{\pi^2}{100} \sin(\frac{\pi}{4}) \cos(\frac{\pi}{10} \cdot (t-10)) \\ -\frac{\pi^3}{6400} \cdot \sin(\frac{\pi}{40} \cdot (t-10)) \\ \frac{\pi^3}{6400} \cdot \sin(\frac{\pi}{40} \cdot (t-10)) \\ 0 \end{bmatrix} \tag{173}$$

This target trajectory is a circular path with a radius of 1 m and center $(x, y) = (0, -1)$ in the xy-plane, rotated 45 deg about the x-axis, and translated -1 m along the z-axis. Since this vehicle is left-right symmetric, independent control in three degrees of freedom (x, y, z directions) can be verified by following this circular path. The vehicle follows this trajectory with a period of 20 s per revolution, while the target roll angle changes every revolution: from 0 deg to 45 deg, 45 deg to 0 deg, and 0 deg to -45 deg. The target pitch angle changes every revolution: from 0 deg to -45 deg, -45 deg to 0

 doi:10.20944/preprints202603.0141.v1

deg, and 0 deg to 45 deg. The target yaw angle is always 0 deg. Following this trajectory verifies the possibility of independent control in all six degrees of freedom.

7.5. Description of Comparative Experiments

This comparative experiment focuses on two aspects of the UKF based on each discretization method: "estimation accuracy" and "computational efficiency." Details of the comparison methods are described below. Comparisons are performed in two patterns: "comparison of three methods: Euler, RK, and 4th-order AB" and "comparison of 2nd to 6th-order AB methods." The results of this comparative experiment are averages from 50 Monte Carlo simulations.

7.5.1. Estimation Accuracy Comparison Experiment

The estimation accuracy comparison is performed by calculating the RMSE values between the estimates from each discretization-based UKF and the true values.

The RMSE value at estimation time k $(k = 1, \cdots, N)$ is calculated using the following equations. Here, N is the total number of sampling points, determined from the simulation time of 70 s and the discretization (i.e., sampling) period h [s] as $N = 70/h$. For example, if $h = 0.01$ [s], then $N = 7000$.

$$MSE_x(k) = \frac{1}{k}\sum_{i=1}^{k}\Big(x(t_i) - \hat{x}(t_i|t_i)\Big)^2 \tag{174}$$

$$RMSE_x(k) = \sqrt{MSE_x(k)} \tag{175}$$

Here, $x(t_i)$ and $\hat{x}(t_i|t_i)$ represent the true value and the estimate of position x at time i, respectively. Equations (174) and (175) are similarly applied to other variables: positions y, z, velocities $\dot{x}, \dot{y}, \dot{z}$, attitude angles ϕ, θ, ψ, and attitude angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$.

Furthermore, when comparing estimation accuracy under different discretization periods ($h = 0.01, 0.02, \cdots, 0.1$), the sum of the individual RMSE values $RMSE_x(N), RMSE_y(N), \cdots, RMSE_{\dot{\psi}}(N)$ for the 12 state variables at the final discrete time $k = N$ is denoted as $RMSE(N)$, and this value is used for comparison.

Here, we discuss the theoretical meaning of $RMSE(N)$. In the UAV model, when the a priori error covariance matrix $P(t_k|t_k)$ at time t_k is expressed as

$$P(t_k|t_k) = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,12} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,12} \\ \vdots & \vdots & \ddots & \vdots \\ p_{12,1} & p_{12,2} & \cdots & p_{12,12} \end{bmatrix}, \tag{176}$$

the trace of this error covariance matrix is denoted as $\mathrm{tr}(P(t_k|t_k))$. The aforementioned $RMSE(N)$ is the sum of RMSE values based on actual sampled estimates, averaged over time up to $k = N$ and averaged over the ensemble of 50 simulation results. Therefore, while transient values differ from the actual $\mathrm{tr}(P(t_k|t_k))$, they can be treated as equivalent values in the steady state.

7.5.2. Computational Efficiency Comparison Experiment

The computational efficiency comparison experiment measures and compares computation times at the following three points:

- *Measurement Time I*: Computation time for sigma point time update (computation of Eq. (8))
- *Measurement Time II*: Computation time for the UKF algorithm (computation from Eq. (3) to Eq. (19))
- *Measurement Time III*: Total computation time of the state estimation program for the UAV model

 doi:10.20944/preprints202603.0141.v1

Figure 4 shows the overall diagram of the state estimation program for the UAV model and the locations of "Measurement Time I," "Measurement Time II," and "Measurement Time III."

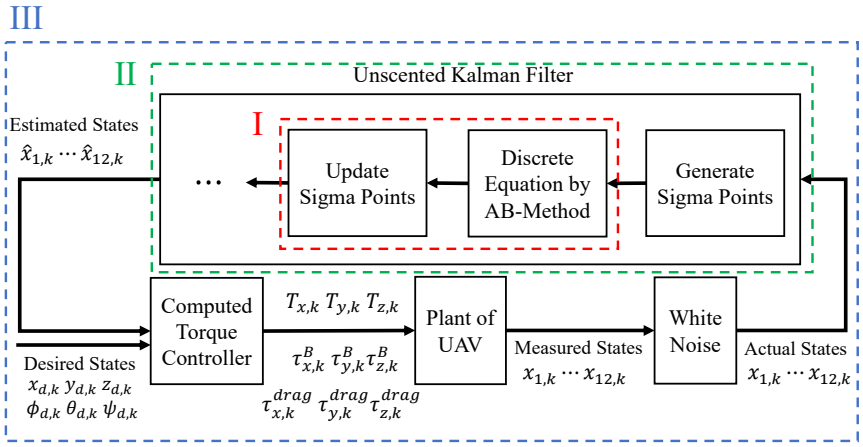


Figure 4. Overall diagram of the state estimation program for the UAV model and the location of measurement times I to III.

The controller design for generating generalized forces follows the computed torque method described in Section 5. The target trajectory involves a circular path from 10 s to 70 s, with yaw angle always at 0 deg, and roll and pitch angles changing to specified angles each revolution. However, the control inputs are first transformed from the generalized forces (thrust and torque for the motion system) provided by the computed torque method into actuator commands (rotor speeds and tilt angles) via the control allocation law. These are then multiplied by the propeller thrust coefficient k_f and torque coefficient k_t to reproduce the thrust and torque (including reaction torque) from the actuators, which are applied as inputs to the plant.

Below, we detail the data collection methods for Measurement Times I, II, and III.

- Measurement Time I
  Since one simulation yields N measurement data points due to the total number of sampling points N, the average of all these is taken to calculate the average computation time per loop. However, due to the nature of the AB method, the average is taken over $N-1$ data points for 2nd order, $N-2$ for 3rd order, $N-3$ for 4th order, $N-4$ for 5th order, and $N-5$ for 6th order.
- Measurement Time II
  Similar to Measurement Time I, the average computation time per loop is calculated, and the total computation time for N loops is also calculated.
- Measurement Time III
  This is the total program computation time, so one simulation yields a single measurement data point.

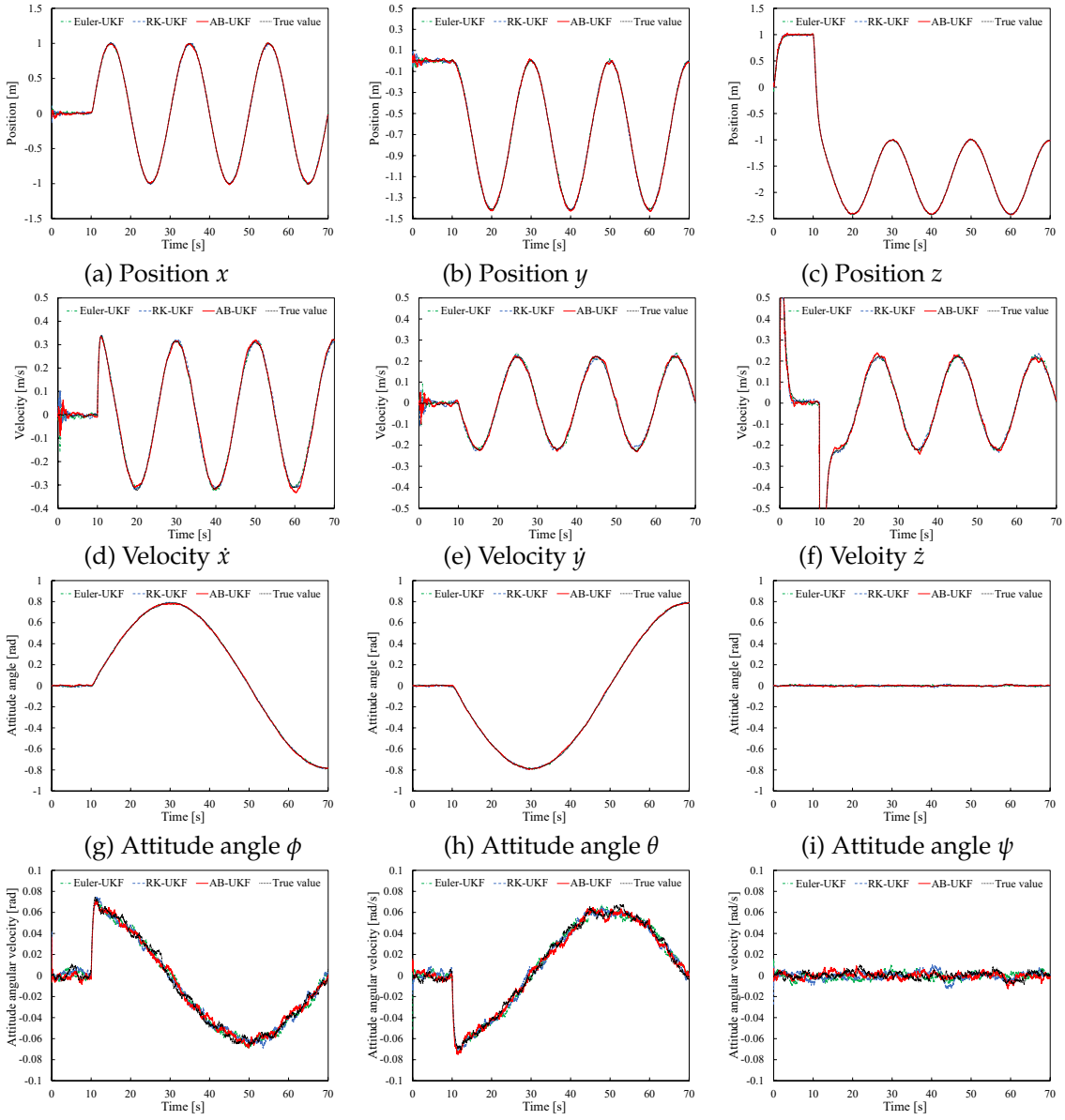
Using these measurement methods, 50 simulations are repeatedly executed, and their average computation times are calculated for each case.

7.6. Results and Discussion of Estimation Accuracy

7.6.1. Comparison of Three Methods: Euler, RK, and AB (4th Order)

First, to show the estimates for the UAV model and to verify whether the UKF based on the multi-step AB method can perform estimation as stably as the single-step Euler and RK methods, Figure 5 shows the time evolution of state estimates by UKF based on the three methods for $h = 0.01$.

 doi:10.20944/preprints202603.0141.v1



(a) Position x (b) Position y (c) Position z

(d) Velocity $\dot{x}$ (e) Velocity $\dot{y}$ (f) Veloity $\dot{z}$

(g) Attitude angle ϕ (h) Attitude angle θ (i) Attitude angle ψ

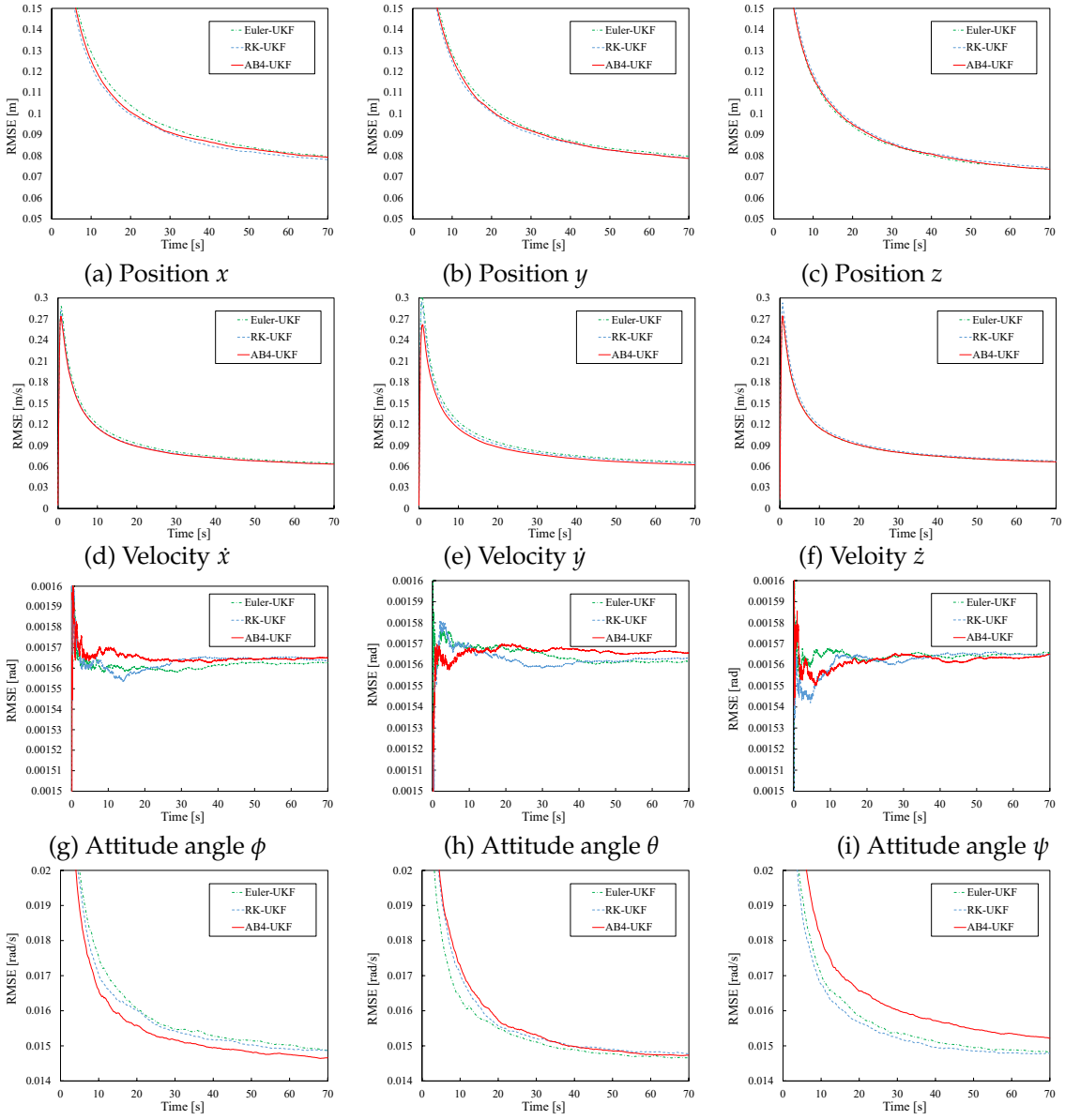
(j) Attitude angular velocity $\dot{\phi}$ (k) Attitude angular velocity $\dot{\theta}$ (l) Attitude angular velocity $\dot{\psi}$

Figure 5. UKF estimates based on three discretization methods ($h = 0.01$).

As seen from Figure 5, the UKF estimates based on the AB method converge stably and achieve high-precision estimation. Also, no significant deviation from the RK method estimates is observed.

Next, Figure 6 shows the time evolution of the RMSE values for the state estimates by UKF based on the three methods for $h = 0.01$, over the 70 s simulation period. From Figure 6, under the condition $h = 0.01$, although the RMSE differences among the methods are very small for all state variables, relatively larger differences are observed in the RMSE values for positions x, y, attitude angles ϕ, θ, ψ, and attitude angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$. Also, for all 12 state variables, the time variation of RMSE values becomes smaller after around 50 s of simulation time.

 doi:10.20944/preprints202603.0141.v1



(a) Position x (b) Position y (c) Position z

(d) Velocity $\dot{x}$ (e) Velocity $\dot{y}$ (f) Veloity $\dot{z}$

(g) Attitude angle ϕ (h) Attitude angle θ (i) Attitude angle ψ

(j) Attitude angular velocity $\dot{\phi}$ (k) Attitude angular velocity $\dot{\theta}$ (l) Attitude angular velocity $\dot{\psi}$

Figure 6. RMSE of UKF estimates based on three discretization methods ($h = 0.01$).

Next, estimation accuracy is compared under different discretization periods: $h = 0.01, 0.02, \cdots, 0.1$. Figure 7 shows the change in $RMSE(N)$ with varying discretization period h. Figure 7 shows that as the discretization period h increases, the differences among the methods become larger, and ultimately the $RMSE(N)$ value for the RK method is the smallest. Furthermore, for the 4th-order AB method, estimation values could not be obtained (diverged) for h values greater than 0.08. Details regarding this are discussed later.

 doi:10.20944/preprints202603.0141.v1

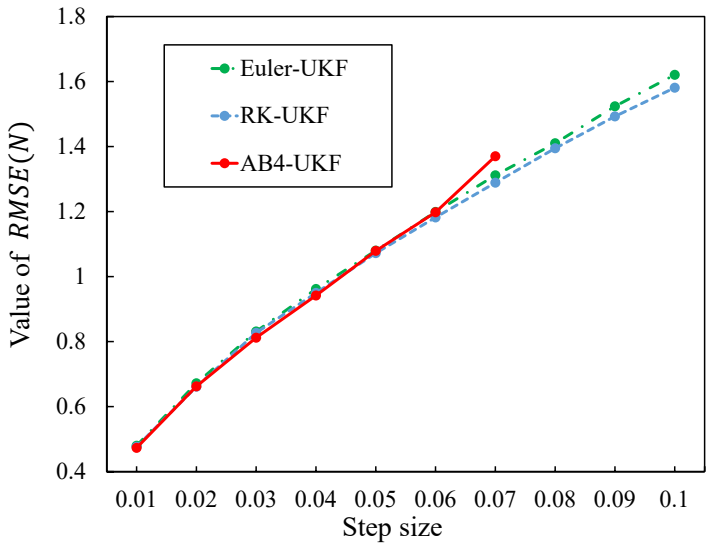


Figure 7. RMSE value change with step size change for three methods.

The specific values for each point in Figure 7 are shown in Table 4. The "N/A" in Table 4 indicates that estimation values could not be obtained due to divergence.

Table 4. Comparison of $RMSE(N)$ with different step size for three methods.

	Method		
	Euler	RK4	AB4
$h = 0.01$	0.47961	0.47646	0.47304
$h = 0.02$	0.67167	0.66121	0.66235
$h = 0.03$	0.83102	0.82695	0.81186
$h = 0.04$	0.96179	0.94971	0.94171
$h = 0.05$	1.07992	1.07212	1.07913
$h = 0.06$	1.19891	1.18197	1.19785
$h = 0.07$	1.31153	1.28889	1.37018
$h = 0.08$	1.41016	1.39465	N/A
$h = 0.09$	1.52378	1.49280	N/A
$h = 0.10$	1.62067	1.58091	N/A

7.6.2. Comparison of AB Methods with Different Orders of Accuracy

The estimation values are omitted as their differences from Figure 6 were very small. Figure 8 shows the time evolution of the RMSE values for UKF estimates based on the 2nd to 6th-order AB methods for $h = 0.01$. From Figure 8, under the condition $h = 0.01$, all AB methods from 2nd to 6th order converge to values comparable to the RMSE values for UKF estimates based on the Euler and RK methods shown in Figure 6, without significant deviation.

Next, estimation accuracy is compared under different discretization periods: $h = 0.01, 0.02, \cdots, 0.1$. Figure 9 shows the change in $RMSE(N)$ with varying discretization period h.

 doi:10.20944/preprints202603.0141.v1

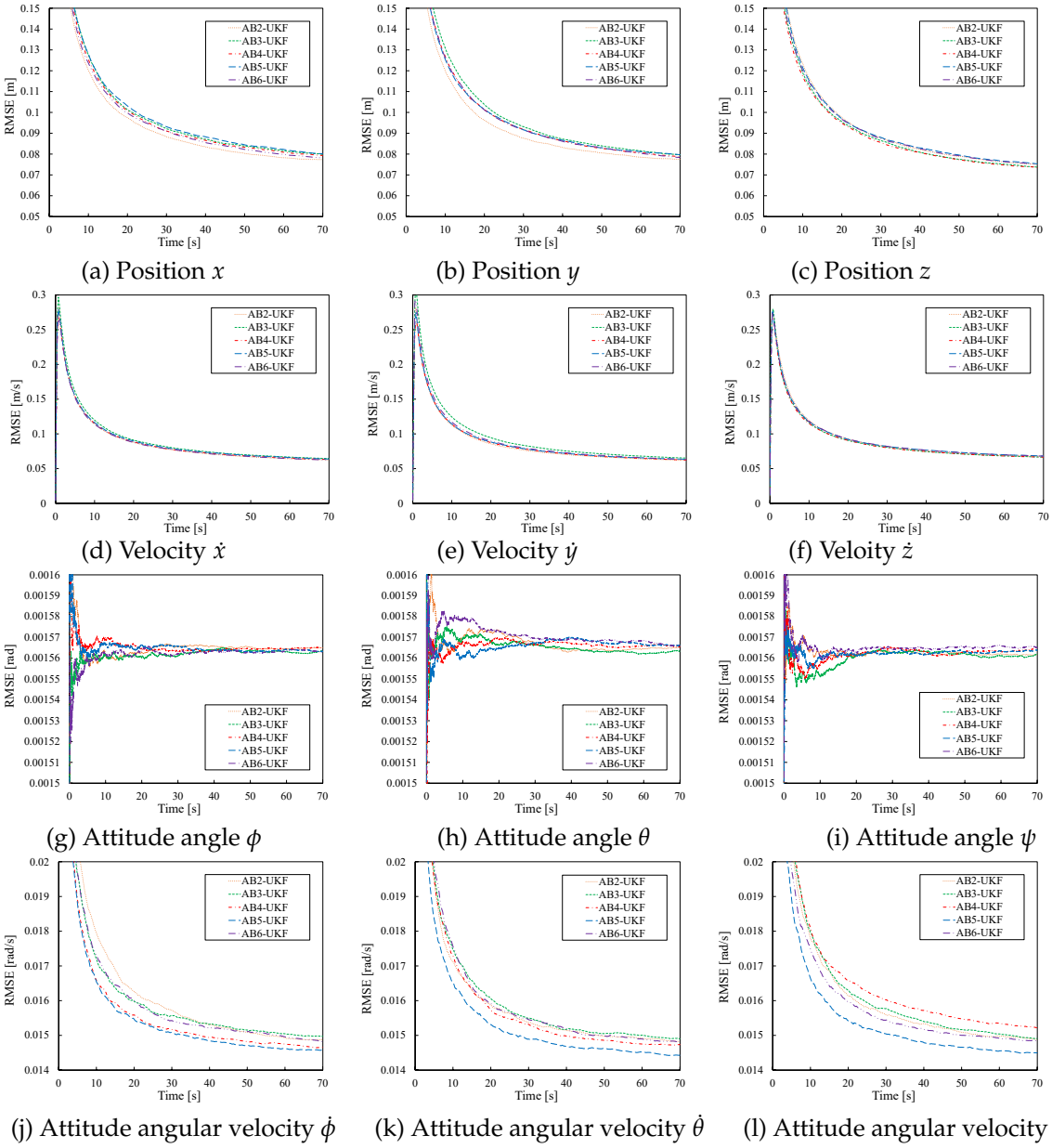


Figure 8. RMSE of UKF estimates for 2nd-order to 6th-order AB-based UKFs ($h = 0.01$).

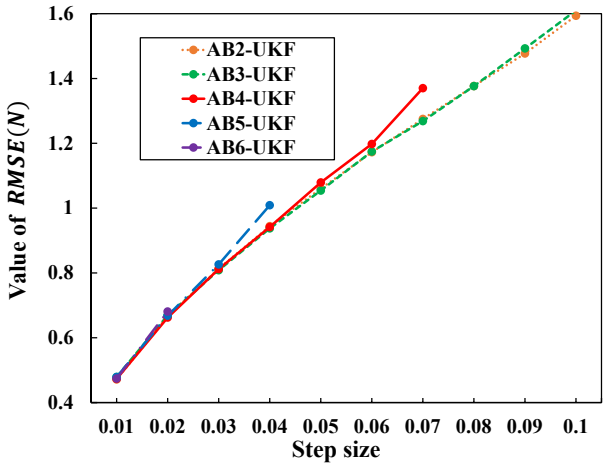


Figure 9. RMSE value change with step size change for different order AB-based UKFs.

 doi:10.20944/preprints202603.0141.v1

Similar to Figure 7, Figure 9 shows that as the discretization period h increases, the differences among the methods become larger. Also, the 6th-order AB method diverged for h greater than 0.03, the 5th-order for h greater than 0.05, and the 4th-order for h greater than 0.08, preventing estimation value acquisition.

The specific values for each point in Figure 9 are shown in Table 5.

Table 5. Comparison of $RMSE(N)$ with different step size for 2nd-order to 6th-order AB-based UKFs.

	AB method with different order				
	AB2	AB3	AB4	AB5	AB6
$h = 0.01$	0.47140	0.47891	0.47304	0.47878	0.47409
$h = 0.02$	0.66619	0.66920	0.66235	0.66816	0.68082
$h = 0.03$	0.81327	0.80840	0.81186	0.82628	N/A
$h = 0.04$	0.94432	0.93755	0.94171	1.00880	N/A
$h = 0.05$	1.06020	1.05421	1.07913	N/A	N/A
$h = 0.06$	1.17243	1.17460	1.19785	N/A	N/A
$h = 0.07$	1.27515	1.26857	1.37018	N/A	N/A
$h = 0.08$	1.37661	1.37650	N/A	N/A	N/A
$h = 0.09$	1.47709	1.49280	N/A	N/A	N/A
$h = 0.10$	1.59363	1.60998	N/A	N/A	N/A

Additionally, for the 2nd and 3rd-order AB methods, further verification confirmed that they similarly diverge for values greater than $h = 0.25$ and $h = 0.15$, respectively.

7.7. Results and Discussion of Computational Efficiency

7.7.1. Comparison of Three Methods: Euler, RK, and AB (4th Order)

Table 6 shows the computation times for Time I, Time II, and Time III for the three methods (Euler, RK, and 4th-order AB) with step size $h = 0.01$.

Table 6. Comparison of computational time among Euler-based UKF, RK-based UKF and AB4-based UKF ($h = 0.01$).

	Method		
	Euler	RK4	AB4
Time I ($\times 10^{-4}$) [s]	3.7593	5.6078	3.2978
Time II ($\times 10^{-4}$) [s]	6.0411	8.0064	5.6144
Time III [s]	6.3785	7.6520	6.0550

From Table 6, the reduction rate in computation time for the 4th-order AB method is 12.276% compared to Euler and 41.119% compared to RK for Time I; 7.063% compared to Euler and 29.876% compared to RK for Time II; and 5.072% compared to Euler and 20.870% compared to RK for Time III.

Next, Figure 10 shows the change in computation time with varying discretization period h. Figure 10 shows that as h becomes smaller, the difference between the two methods increases. This is considered to be due to the influence of the increasing number of program loops as the total number of sampling points N becomes larger for smaller h.

 doi:10.20944/preprints202603.0141.v1

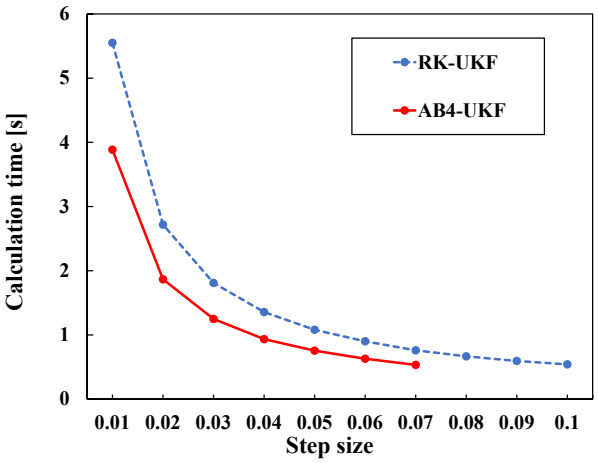


Figure 10. Calculation time change with step size change for RK-based UKF and AB4-based UKF.

7.7.2. Comparison of AB Methods with Different Orders of Accuracy

Next, computation times for Time I, Time II, and Time III are compared under the same conditions for AB methods from 2nd to 6th order for $h = 0.01$, not just the 4th-order AB method. The result is given in Table 7. The computational difference among AB methods from 2nd to 6th order lies only in the number of past sigma point matrices used as "previous information." Therefore, the differences among methods are smaller than the results shown in Table 6.

Table 7. Comparison of computational time for 2nd-order to 6th-order AB-based UKFs.

	AB method with different order				
	AB2	AB3	AB4	AB5	AB6
Time I ($\times 10^{-4}$) [s]	3.2125	3.2839	3.3212	3.3478	3.4263
Time II ($\times 10^{-4}$) [s]	5.5107	5.6166	5.6353	5.6081	5.7562
Time III [s]	6.1799	6.2990	6.1856	6.1427	6.3341

Next, Figure 11 shows the change in computation time with varying discretization period h. From Figure 11, unlike the comparison between RK and AB methods, almost no difference is seen among the methods. The reason for this result is considered to be that the difference in computational load due to the order of the AB method stems only from the difference in the number of sigma point matrices handled, and the number of derivative calculations per step is the same for any order. Also, from this result, it can be understood that while estimation with the AB method diverges if the step size is too large, it does not diverge under conditions with h values smaller than 0.01, allowing estimation with higher order accuracy and shorter computation time compared to the RK method under those conditions.

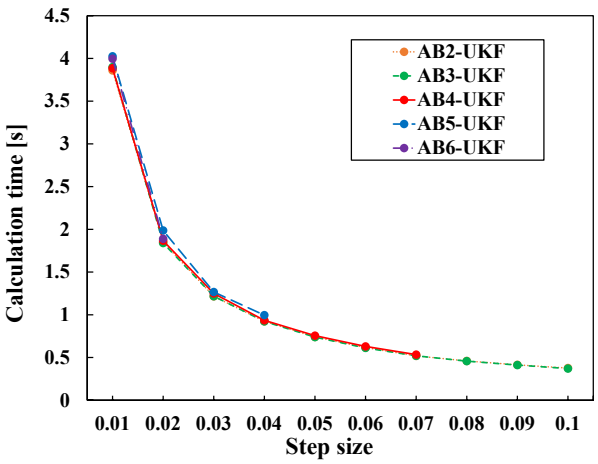


Figure 11. Calculation time change with step size change for different order AB-based UKFs.

 doi:10.20944/preprints202603.0141.v1

8. Conclusion

This paper proposed a UKF that newly incorporates the AB method instead of the RK method into the sigma point time-update equation, maintaining estimation accuracy comparable to the RK method while enabling estimation with more efficient computation time. This is an alternative to the state estimation method using UKF with the RK method adapted to the sigma point time-update equation, as proposed by Takeno et al. [9]. First, we reviewed the UKF, a type of nonlinear filter, and presented its algorithm. Next, we explained that both the RK and AB methods can be applied to the UT and presented the sigma point time-update equations based on these two methods plus the Euler method. For the actual verification of estimation performance, we focused on the falling body model used by Takeno et al. as a preliminary experiment and the UAV model as the main subject. We detailed the derivation of sigma point time-update equations when applying UKF with three discretization methods (Euler, RK, AB) to each model. Numerical simulation results demonstrated that for both models, the proposed method maintains estimation accuracy comparable to the RK method while enabling more efficient computation. Notably, the improvement in computational efficiency was more pronounced for the UAV model with more complex state equations.

Future work includes implementing the proposed algorithm on a microcontroller in a real vehicle environment, starting with the UAV model used in this state estimation simulation, to verify the actual degree of improvement in estimation accuracy and computational efficiency. Also, as mentioned earlier, while the standard UKF prepares $(2n+1)$ sigma points to compute estimates, methods using fewer sigma points have already been proposed. For example, by using the Spherical Simplex Unscented Transformation [28,29,31,32] or the Simplex Unscented Transformation [30], the number of sigma points can be reduced to $n+2$ or $n+1$, constituting a CD-UKF that can reduce computational load. Furthermore, applying the proposed method to the CD Derivative-Free EKF [4,33,35], which uses n sample points, is also an interesting challenge. Combining these methods with the AB method is expected to lead to the development of UKF or EKF methods in continuous-discrete environments that offer improved computational efficiency with reasonable processing speed while maintaining a certain level of computational accuracy.

References

1. Paul Frogerais, Jean-Jacques Bellanger, Lotfi Senhadji. "Various Ways to Compute the Continuous-Discrete Extended Kalman Filter," *IEEE Transactions on Automatic Control*, 2012, 57 (4), pp. 1000–1004.
2. Kulikova, M.V. "Accurate Numerical Implementation of the Continuous-Discrete Extended Kalman Filter," *IEEE Transactions on Automatic Control*, 2014, 59 (1), pp. 273–279.
3. Kulikov, G.Y.; Kulikova, M.V. "The Accurate Continuous-Discrete Extended Kalman Filter for Radar Tracking," *IEEE Transactions on Signal Processing*, 2016, 64 (4), pp. 948–958.
4. Kulikova, M.V.; Kulikov, G.Y. "On Derivative-Free Extended Kalman Filtering and Its MATLAB-Oriented Square-Root Implementations for State Estimation in Continuous-Discrete Nonlinear Stochastic Systems," *European Journal of Control*, 2023, 73, 100885.
5. Julier, S.J.; Uhlmann, J.K. "A New Extension of the Kalman Filter to Nonlinear Systems," In *Proc. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI*, 1997, pp. 182–193.
6. Julier, S.J.; Uhlmann, J.K. "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, 2004, 92 (3), pp. 401–421.
7. Julier, S.J.; Uhlmann, J.K.; Durrant-Whyte, H.F. "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control*, 2000, 45 (3), pp. 477–482.
8. Särkkä, S. "On Unscented Kalman Filtering for State Estimation of Continuous-Time Nonlinear Systems," *IEEE Transactions on Automatic Control*, 2007, 52 (9), pp. 1631–1641.
9. Takeno, M.; Katayama, T. "State and Parameter Estimation for Dynamical Systems by Using Unscented Kalman Filter," *Transactions of the Institute of Systems, Control and Information Engineers*, 2011, 24 (9), pp. 231–239. (In Japanese)
10. Kulikov, G.Y.; Kulikova, M.V. "Accurate Continuous–Discrete Unscented Kalman Filtering for Estimation of Nonlinear Continuous-Time Stochastic Models in Radar Tracking," *Signal Processing*, 2017, 139, pp. 25–35.

 doi:10.20944/preprints202603.0141.v1

11. Kulikova, M.V.; Kulikov, G.Y. "Continuous–Discrete Unscented Kalman Filtering Framework by MATLAB ODE Solvers and Square-Root Methods," *Automatica*, 2022, 142, 110396.
12. Takeno, M.; Katayama, T. "A Numerical Method for Continuous Discrete Unscented Kalman Filter," *International Journal of Innovative Computing, Information and Control*, 2012, 8 (3), pp. 2261–2274.
13. Knudsen, T.; Leth, J. "A New Continuous Discrete Unscented Kalman Filter," *IEEE Transactions on Automatic Control*, 2019, 64 (5), pp. 2198–2205.
14. Wang, A.; Zhang, H.; Huang, X.; Yang, Z. "Adaptive unscented Kalman filter for nonlinear continuous-discrete systems based on the degree of nonlinearity," *Physica Scripta*, 2025, 100 (8), 087001. DOI: 10.1088/1402-4896/adee58
15. Wang, C.; Dai, H.; Yang, W.; Yue, X. "High-efficiency unscented Kalman filter for multi-target trajectory estimation," *Aerospace Science and Technology*, 2025, 159, 109962. DOI: 10.1016/j.ast.2025.109962
16. Tobaly, L; Yaniv, E; Zalevsky, Z. "Integrating GAN-based machine learning with nonlinear Kalman filtering for enhanced state estimation," *Scientific Reports*, 2025, 15, 42361. DOI: 10.1038/s41598-025-26339-9
17. Kulikova, M.V.; Kulikov, G.Y. "Square-root information-type methods for continuous–discrete extended Kalman filtering," *European Journal of Control*, 2025, 85, 101358. DOI: 10.1016/j.ejcon.2025.101358
18. Al Ahdab, M.; et al. "Optimal Sensor Scheduling and Selection for Continuous-Discrete Kalman Filtering with Auxiliary Dynamics," *arXiv preprint*, 2025, arXiv:2507.11240. URL: https://arxiv.org/abs/2507.11240
19. Thieu, T.; Melnik, R. "Estimation of 3D facial dynamics with nonlinear filters for position tracking," *Applied Mathematics in Science and Engineering*, 2025, 32 (1), 2546793. DOI: 10.1080/27690911.2025.2546793
20. He, R.; Chen, S.; Wu, H.; Zhang, F.; Chen, K. "Efficient Extended Cubature Kalman Filtering for Nonlinear Target Tracking," *International Journal of Systems Science*, 2021, 52 (2), pp. 392–406.
21. Gelb, A. *Applied Optimal Estimation*; MIT Press: Cambridge, MA, USA, 1974.
22. Cellier, F.E.; Kofman, E. *Continuous System Simulation*; Springer: New York, NY, USA, 2006; ISBN 0-387-26102-8.
23. Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2016; ISBN 9781119121503.
24. Alderete, T.S. *Simulator Aero Model Implementation*; NASA Ames Research Center: Moffett Field, CA, USA, 1997. Available online: https://api.semanticscholar.org/CorpusID:6652331 (accessed on ...).
25. Luukkonen, T. *Modelling and Control of Quadrotor*; Independent Research Project in Applied Mathematics, Mat-2.4108, School of Science, Aalto University: Espoo, Finland, 2011.
26. Itakura, T.; Watanabe, K.; Nagai, I.; Shibanoki, T. "Design and Production of an Osprey-Type Drone with 2-DOF Tiltable Mechanisms," In *Proceedings of the 2022 JSME Conference on Robotics and Mechatronics*, Sapporo, Japan, 1–4 June 2022; pp. 1A1-J03(1)–1A1-J03(4). (In Japanese)
27. Yoshiwaki, N.; Watanabe, K.; Shibanoki, T.; Nagai, I. "Research on a Fully Actuated UAV with Two 2-DOF Tiltable Rotors," In *Proceedings of the 2022 JSME Conference on Robotics and Mechatronics*, Sapporo, Japan, 1–4 June 2022; pp. 1A1-J02(1)–1A1-J02(4). (In Japanese)
28. Julier, S.J.; Uhlmann, J.K. "Reduced Sigma Point Filters for the Propagation of Means and Covariances Through Nonlinear Transformations," In *Proceedings of the 2002 American Control Conference*, Anchorage, AK, USA, 8–10 May 2002; pp. 887–892.
29. Julier, S.J. "The Spherical Simplex Unscented Transformation," In *Proceedings of the 2003 American Control Conference*, Denver, CO, USA, 4–6 June 2003; pp. 2430–2434.
30. Li, W.-C.; Wei, P.; Xiao, X.-C. "A Novel Simplex Unscented Transform and Filter," In *Proceedings of the 2007 International Symposium on Communications and Information Technologies (ISCIT 2007)*, Sydney, Australia, 17–19 October 2007; pp. 926–931.
31. Lozano, J.G.C.; Carrillo, L.R.G.; Dzul, A.; Lozano, R. "Spherical Simplex Sigma-Point Kalman Filters: A Comparison in the Inertial Navigation of a Terrestrial Vehicle," In *Proceedings of the 2008 American Control Conference*, Seattle, WA, USA, 11–13 June 2008; pp. 4857–4862.
32. Fu, K.; Zhao, G.; Li, X.; Tang, Z.-L.; He, W. "Iterative Spherical Simplex Unscented Particle Filter for CNS/Redshift Integrated Navigation System," *Science China Information Sciences*, 2017, 60 (4), 042201.
33. Quine, B.M. "A Derivative-Free Implementation of the Extended Kalman Filter," *Automatica*, 2006, 42 (11), pp. 1927–1934.

 doi:10.20944/preprints202603.0141.v1

34. Kulikova, M.V.; Kulikov, G.Y. "On Derivative-Free Extended Kalman Filtering and Its Matlab-Oriented Square-Root Implementations for State Estimation in Continuous-Discrete Nonlinear Stochastic Systems," *European Journal of Control*, 2023, 73, 100886.
35. Kulikova, M.V.; Kulikov, G.Y. "Continuous-Discrete Derivative-Free Extended Kalman Filter Based on Euler-Maruyama and Itô-Taylor Discretizations: Conventional and Square-Root Implementations," *European Journal of Control*, 2024, 76, 100960.