Sensors and Materials, Vol. 37, No. 12 (2025) 5425–5437
MYU Tokyo

S & M 5254

Physics-informed Platform for Flight Dynamics Simulation

Pattiwat Atayagul and Pitikhate Sooraksa*

Department of Robotics and AI, School of Engineering,
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

(Received September 30, 2025; accepted November 17, 2025)

Keywords: flight dynamics model, physics-informed platform, physics-based model

In this study, we present the state-of-the-art development of the flight dynamics model of a rigid body from the theoretical aspect of flight dynamics, design the software architecture, and validate the proposed architecture by comparing the simulation results with the check-cases for the verification of six-degree-of-freedom flight vehicle simulations document issued by the National Aeronautics and Space Administration (NASA). We also design the computational workflow between the ordinary differential equations of the system and other axillary components by weaving those relations in the object-oriented programing style and powered with Python scientific libraries. The simulation outcomes in all cases are well matched with the majority of NASA baseline datasets under the same flight simulation conditions, which reflect the accuracy of the model with considerable confidence.

1. Introduction

In recent years with the advancement of AI, one way of designing the flight control of sophisticated maneuverability is through reinforcement learning.[1–3] An agent is trained to seek the total maximum reward from the path of its interactions with the environment; however, developing such an agent by trial-and-error iterations in the real-world environment is costly and prone to safety-related issues. Thus, the virtual environment of flight dynamics is often used to train an agent before being deployed in the real physical world, and this approach considerably addresses safety concerns and reduces the cost of training.[4]

The flight dynamics model (FDM) is a physics-based tool used to simulate an aircraft's trajectory and orientation under different flight conditions. There is off-the-shelf FDM software, and the main programming languages mostly found among FDM software programs are low-level languages such as C and C++. An example is an open-source Gazebo software program written in C++ and equipped with a distributed architecture of different physics engines and sensors.[5] However, a significant drawback of low-level languages is their complex syntax that requires researchers to dedicate their resources to coding instead of focusing on their problems. Nowadays, there are rich scientific libraries of high-level programming languages, and one of

*Corresponding author: e-mail: pitikhate.so@kmitl.ac.th
https://doi.org/10.18494/SAM5957

ISSN 0914-4935

these languages is Python, the most popular scientific language being used among scientists across different fields. The main benefit of using Python over static coding languages is its intuitively readable manner, which helps researchers become more productive in their projects.[6]

With the trend of AI, the availability of affordable computation and the richness of Python scientific libraries encouraged us to propose the workflow for devising FDM from the theoretical physics aspect of flight dynamics to software architecture design, which can be later used for AI applications such as intelligent flight systems. This paper is structured as follows: the first section considers the mathematical background of 6-DOF flight dynamics such as a geodesy model of Earth's presentation and all state-space equations. This is followed by the simulation software architecture that explains the execution flow of the proposed object-oriented programming (OOP). The third section is the validation part, which contains some relevant check-cases in the NASA verification document of 6-DOF flight vehicle simulations[7] used to verify flight trajectories against our FDM simulation platform. This paper ends with the conclusion and future research.

2. Mathematical Scheme

The mathematics of developing a flight dynamics simulation platform was based on the work of Stevens *et al.*[8] The derivation of governing equations depends on the level of physical fidelity; for most practical purposes, the assumption of a rigid body is good enough for a wide range of flight control designs. The term rigid means that the structure of an aircraft has no flexibility, all points of the aircraft are relatively constant throughout the simulation, and this restriction is used to form the equation of motion (EOM) in Sect. 2.1.

The derivation of EOM starts with the concept of a reference frame, since the vector and its derivative can be taken with respect to any particular frame. There are four essential frames as shown in Fig. 1, which are used for describing the aircraft's maneuver and for deriving both linear and angular EOMs. The first two frames are the rotating Earth-centered, Earth-fixed frame $ECEF = \{X_{ECEF}, Y_{ECEF}, Z_{ECEF}\}$ and the Earth-centered inertial frame $ECI = \{X_{ECI}, Y_{ECI}, Z_{ECI}\}$, which are both used to locate an aircraft's location with respect to the oblate Earth. Their origins are attached to Earth's center; these two frames have the z-axis aligned to the North Pole, the *x*-axis points toward the vernal equinox, and the *y*-axis completes the right-hand rule. As the name suggests, *ECI* acts as an inertial frame, whereas *ECEF* rotates with a constant Earth rotation speed around the *z*-axis. Another frame is the forward-right-down frame $frd = \{X_{frd}, Y_{frd}, Z_{frd}\}$ or the body frame, which is usually attached coincidently to the mass center of an aircraft, where the x-axis points to the nose of the aircraft, the *y*-axis aligns with the right wing, and the *z*-axis obeys the right-hand rule. For aerodynamic effort, the wind frame $wind = \{X_{wind}, Y_{wind}, Z_{wind}\}$ is the frame associated with computing aerodynamics forces. The orientation of the wind frame can be found by rotating the X_{frd} of the *frd* frame to be aligned with the relative velocity vector between the aircraft's velocity and the local velocity of the wind field around the aircraft. The last frame is the north-east-down frame $NED = \{X_{NED}, Y_{NED}, Z_{NED}\}$, which is attached to Earth's ground, where the z-axis points to Earth's center, the y-axis aligns with the East, and the *x*-axis completes the right-hand rule. The *NED* frame serves as an

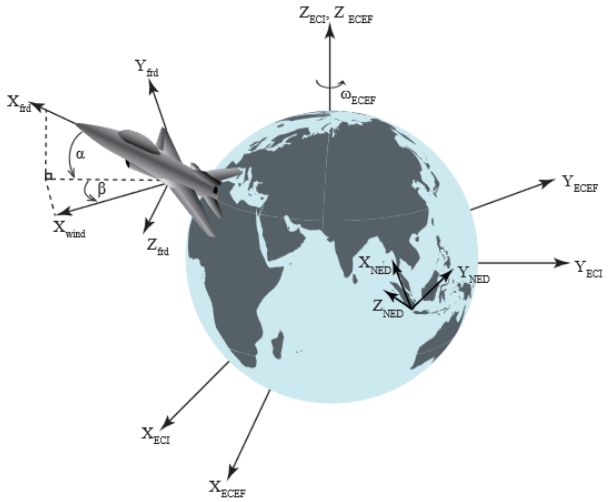

Fig. 1. (Color online) Essential frame of reference.

observation frame for measuring the aircraft's orientation, which is commonly known as Euler's angle.

2.1 6-DOF equation of motion

A vector can be observed with respect to one frame, while its derivative is taken in another frame. To express those relations, one superscript and two subscripts are used to capsulate such information. The right subscript can flexibly designate two points for a positioning vector p or a point with its reference frame for a velocity vector v or two frames for an angular speed vector ω. The right superscript indicates the frame used to project the vector's components onto or expresses only a scalar value if only a specific axis is given. The left superscript indicates the frame that the vector's derivative is being taken. The trajectory and orientation of the aircraft are governed by the rate of change in both linear and angular momenta; the full derivation is described in Ref. 8. For the linear momentum, the velocity derivative $\dot{v}$ with respect to the *ECEF* frame to form the translational EOM with the assumption of the aircraft's mass and inertia is held constant throughout the simulation time and is expressed as Eq. (1). The rate of change in the velocity of an aircraft depends on the current states of the angular velocity ω and the positional vector p, wherein the notations m, cm, and o stand for the aircraft's mass, the mass center of the aircraft, and the Earth's center, respectively. The external force F^{ECEF} shown in Eq. (1) might include the aerodynamic force, the thrust force from the aircraft's engine, or any type of external force. However, these external forces must be projected onto the *ECEF* frame before being eligible to be applied to the equation.

$$
\begin{aligned}
{}^{ECEF}\dot{v}_{cm/ECEF}^{ECEF} = {} & \frac{F^{ECEF}}{m} - \omega_{ECEF/ECI}^{ECEF} \times \left(\omega_{ECEF/ECI}^{ECEF} \times p_{cm/o}^{ECEF} \right) \\
& - 2\omega_{ECEF/ECI}^{ECEF} \times v_{cm/ECEF}^{ECEF} + g^{ECEF}
\end{aligned}
\tag{1}
$$

Here, $\omega_{ECEF/ECI}^{ECEF} \times v_{cm/ECEF}^{ECEF}$ is the Coriolis term, which is the product of Earth's angular speed and the aircraft's velocity. This term is essential as the aircraft's speed approaches the supersonic level, which is demonstrated in case 7 of the simulation verification session. The J2 gravitational model is presented as g^{ECEF} and the full equation is expressed as Eq. (2), where GM is the product of Earth's mass and the universal gravitational constant of the inverse square law, J_2 is a constant variable computed from the EGM96 coefficient, a is the semi-major axis of the World's reference ellipse, and ψ is a geocentric latitude at the instant of time.

$$g^{ECEF} = -\frac{GM}{\left\| p_{cm/o}^{ECEF} \right\|^2} \begin{bmatrix} \left[1 + \frac{3}{2}\left(\frac{a}{r}\right)^2 J_2\left(1 - 5\sin^2\psi\right)\right] p_{cm/o}^{X_{ECEF}} / \left\| p_{cm/o}^{ECEF} \right\| \\ \left[1 + \frac{3}{2}\left(\frac{a}{r}\right)^2 J_2\left(1 - 5\sin^2\psi\right)\right] p_{cm/o}^{Y_{ECEF}} / \left\| p_{cm/o}^{ECEF} \right\| \\ \left[1 + \frac{3}{2}\left(\frac{a}{r}\right)^2 J_2\left(3 - 5\sin^2\psi\right)\right] p_{cm/o}^{Z_{ECEF}} / \left\| p_{cm/o}^{ECEF} \right\| \end{bmatrix} \tag{2}$$

The angular EOM expressed as Eq. (3) is derived through the angular momentum conservation with respect to the *frd* frame. Similarly, any external moment must be expressed in the *frd* frame as M^{frd} for the compatibility of the equation. The cross-product of the inertia matrix J^{frd} is expressed in Eq. (4) with the restrictive assumption of the symmetrical plane being made along the $x - z$ plane of the *frd* frame of the aircraft.

$${}^{frd}\dot{\omega}_{frd/ECI}^{frd} = \left(J^{frd}\right)^{-1}\left[M^{frd} - \tilde{\omega}_{frd/ECI}^{frd} J^{frd} \omega_{frd/ECI}^{frd}\right] \tag{3}$$

$$J^{frd} = \begin{bmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{bmatrix} \tag{4}$$

The aerodynamic effort force and moment in Eqs. (1) and (3) come from the dynamic pressure exerting on the aircraft body, which is being transformed to force or moment as proportional to the reference surface S and the moment reference lengths b and $\overline{c}$. Then, these forces are corrected using the non-dimensional aerodynamic coefficients usually obtained from the empirical or simulation data of that particular aircraft. To be compatible with EOMs, the aerodynamic force and moment must be projected on the *ECEF* and *frd* frames as in Eqs. (5) and (6). The drag, side, and lift coefficients (C_D, C_C, and C_L, respectively) are commonly measured along with the *wind* frame; therefore, these forces must be converted into the *ECEF* frame via the directional cosine matrix (DCM) of $C_{ECEF/wind}$. In contrast, moment correction factors $\left(C_{\mathcal{L}}, C_{\mathcal{M}}, \text{and } C_{\mathcal{N}}\right)$ are usually measured along the *frd* frame, so no conversion is required. Usually, both force and moment coefficients are functions of the attack angle α and the slip angle β.

$$AF^{ECEF} = -C_{ECEF/wind}\frac{1}{2}\rho_{air}\left\|V^{ECEF}\right\|^2 S\begin{bmatrix} C_D \\ C_C \\ C_L \end{bmatrix} \tag{5}$$

$$AM^{frd} = \frac{1}{2}\rho_{air}\left\|V^{ECEF}\right\|^2\begin{bmatrix} SbC_{\mathcal{L}} \\ S\bar{c}C_{\mathcal{M}} \\ SbC_{\mathcal{N}} \end{bmatrix} \tag{6}$$

V^{ECEF} is the relative velocity between the aircraft's velocity $v_{cm/ECEF}^{ECEF}$ and the local wind field w^{ECEF} as shown in Eq. (7).

$$V^{ECEF} = \left(v_{cm/ECEF}^{ECEF} - w^{ECEF}\right) \tag{7}$$

The *wind* frame can be found by rotating the *frd* frame around Y_{frd} for the angle α, which is called the stability frame $S = \{X_S, Y_S, Z_S\}$, and then further rotating it around Z_S for the angle β. Both angles are calculated through the inverse trigonometric functions as shown in Eq. (8).

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(V^{Z_{ECEF}}, V^{X_{ECEF}}\right) \\ \operatorname{asin}\left(V^{Y_{ECEF}} / \left\|V^{ECEF}\right\|\right) \end{bmatrix} \tag{8}$$

2.2 Quaternion

In the platform, the quaternion is used as a means of constructing DCM to convert any vector between those four frames. The advantage of using the quaternion is that the quaternion derivative can be computationally stated along other states in Eqs. (1) and (3) as additional states in the state-space form, making simulation more efficient and compact. The quaternion $q_{frd/ECEF}$ was chosen to be the state variable. The notation $q_{frd/ECEF}$ means that any vector present in the *ECEF* frame will be projected onto the *frd* frame by multiplying that vector with the DCM of $q_{frd/ECEF}$. The quaternion $q_{frd/ECEF}$ is indirectly constructed from the two predefined quaternions $q_{frd/NED}$ and $q_{NED/ECEF}$ as shown in Eq. (9).

$$q_{frd/ECEF} = q_{NED/ECEF} * q_{frd/NED} \tag{9}$$

The quaternion $q_{frd/NED}$ is initialized as Eq. (10) from the set of successive rotations for aligning the *NED* frame to coincide with the *frd* frame, and this set of successive rotations is called Euler's angles, denoted as ψ, θ, and ϕ for such successive rotation order.

$$q_{frd/NED} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{10}$$

The initialization of the quaternion $q_{NED/ECEF}$ as Eq. (11) depends on the geocentric relationship between the geocentric latitude ϕ and the geocentric longitude ℓ as known from the given initial conditions.

$$q_{NED/ECEF} = \begin{bmatrix} \sin(\ell/2)\sin([\phi+\pi]/2) \\ \sin(\ell/2)\sin([\phi+\pi]/2) \\ -\cos(\ell/2)\sin([\phi+\pi]/2) \\ \sin(\ell/2)\cos([\phi+\pi]/2) \end{bmatrix} \tag{11}$$

As the program being executed, the quaternion $q_{frd/ECEF}$ will be tracked and updated through its time-derivative form as Eq. (12). This form is included in the state-space form along with other physical states. Euler's angle is known by manipulating an inversion of Eq. (9) as in Eq. (13).

$$\dot{q}_{frd/ECEF} = \frac{1}{4}\begin{bmatrix} 0 & -\left(\omega^{frd}_{frd/ECEF}\right)^T \\ \omega^{frd}_{frd/ECEF} & -\tilde{\omega}^{frd}_{frd/ECEF} \end{bmatrix}\begin{bmatrix} \left(q_{frd/ECEF} + q^*_{frd/ECEF}\right) \\ \left(q_{frd/ECEF} - q^*_{frd/ECEF}\right) \end{bmatrix} \tag{12}$$

$$q_{frd/NED} = inv\left(q_{NED/ECEF}\right) * q_{frd/ECEF} \tag{13}$$

2.3 Numerical method

The time evolution of the flight trajectory depends on the set of states being capsulated in y as shown in Eq. (14), whose derivatives are described throughout nonlinear ODEs as mentioned earlier. The platform deployed the high-level SciPy API called solve_ivp, which offers myriad choices of numerical methods to solve the initial-valued problem of ODE. We decided to use the explicit Runge–Kutta method of order 5(4) as a numerical method for solving the proposed FDM.

$$y = \left\{ v^{ECEF}_{cm/ECEF}, \omega^{frd}_{frd/ECI}, p^{ECEF}_{cm/o}, q_{frd/ECEF} \right\} \tag{14}$$

3. Simulation Platform Architecture

OOP is used to abstract attributes and operations to execute the proposed FDM simulation. The class diagram of the unified modeling language (UML) in Fig. 2 visualizes the relations among classes within the simulation model, and only pivotal attributes and operations are shown in the figure for the purpose of being concise. The main class is the StateSpace class, and its primary duty is to symbolically hold the state-space form of ordinary differentiation equations and update them along the simulation time with the help of other auxiliary classes. The StateSpace class has an operation externalForce; under the hood, this operation computes all aerodynamic efforts that affect all six EOMs of a certain vehicle type being stated in the VehicleModel class at the instant of time and updates EOMs in the symbolic form. Another operation in the StateSpace class is callable function __call__ used to convert the easy-manageable SymPy symbolic form into a lambda function. This allows the SciPy library to perform the numerical approximation that is faster than in the symbolic form. The USSA1976 class contains an attribute Air Density in the form of tabular data; this is air density data at various attitudes published in the U.S. Standard Atmosphere 1976. At any time instant, the air density will be extrapolated from a given temporal altitude to be used in aerodynamic effort calculation within the StateSpace class. The transformation of vectors from one frame to another frame is treated by the Quaternion class. Inside the class, there is a quaternion attribute that is later converted to DCM and multiplied with the targeted vector for projecting a vector into the specific frame, and the quaternation gets updated through an update operation. For data manipulation, the DataProcessing class stacks simulation data of both states and time as its attributes, and has a callable function __call__ for collecting simulation data at a specific time step.

4. Simulation Verification

The accuracy of the proposed FDM is verified through the comparison plot against the NASA Check-Cases for Verification of 6-DOF Vehicle Simulations document along with an average root mean square error $RMSE_{ave}$. Only relevant cases are selected as baseline data versus the proposed platform outcomes under the identical initial and flight conditions as shown

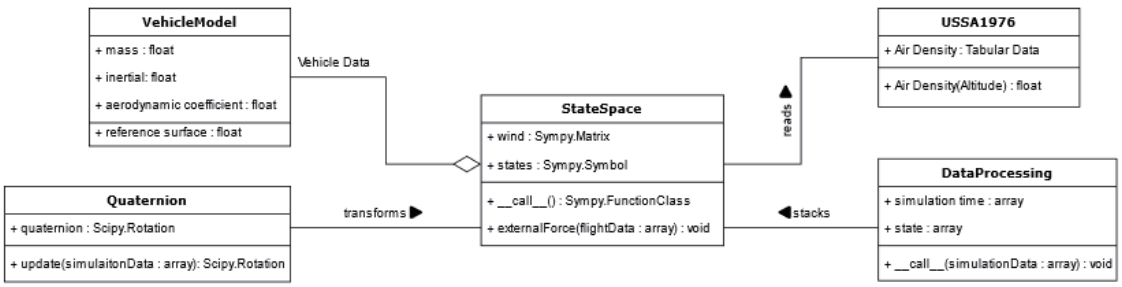

Fig. 2. UML of simulation platform.

in Table 1. The vehicle properties, initial conditions, and flight conditions of each verification case are described in Ref 7. In all cases, the aircraft flies around the model of an oblate Earth WGS-84 under the influence of the J_2 gravitational model and the 1976 U.S. standard atmosphere. In each case, the comparison plot includes Euler's angles, the geodetic position of the aircraft, the flight path projected on the *ECEF* frame, and the aerodynamics force components projected onto the *frd* frame as shown in Figs. 3–9. $RMSE_{ave}$ in Eq. (15) is an accumulated error along the simulation time t from the beginning of the simulation that lasted for n iterations. The error is a squared difference between the simulation state y and the identical baseline state b among the N available NASA baseline datasets. The NASA05 dataset is not included in $RMSE_{ave}$ since its simulation time step is different from ours and even NASA's peers, which is even considered as an outlier dataset.

$$RMSE_{avg} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{30}\sqrt{\frac{\left(y_t - b_t^i\right)^2}{n}} \tag{15}$$

Table 1
Verification case.

NASA verification case	Verification purpose	Winds
I. Dropped sphere with no drag	Gravitation, Translation EOM	Still air
II. Tumbling brick with no damping, no drag	Rotational EOM	Still air
III. Dropped sphere with constant C_D, no wind	Ellipsoidal Earth	Still air
IV. Dropped sphere with constant C_D, wind	Wind effects	Steady wind
V. Dropped sphere with constant C_D, wind shear	2D wind	Depending on altitude
VI. Sphere launched eastward along equator	Translation EOM	Still air
VII. Sphere launched northward along prime meridian	Coriolis	Still air

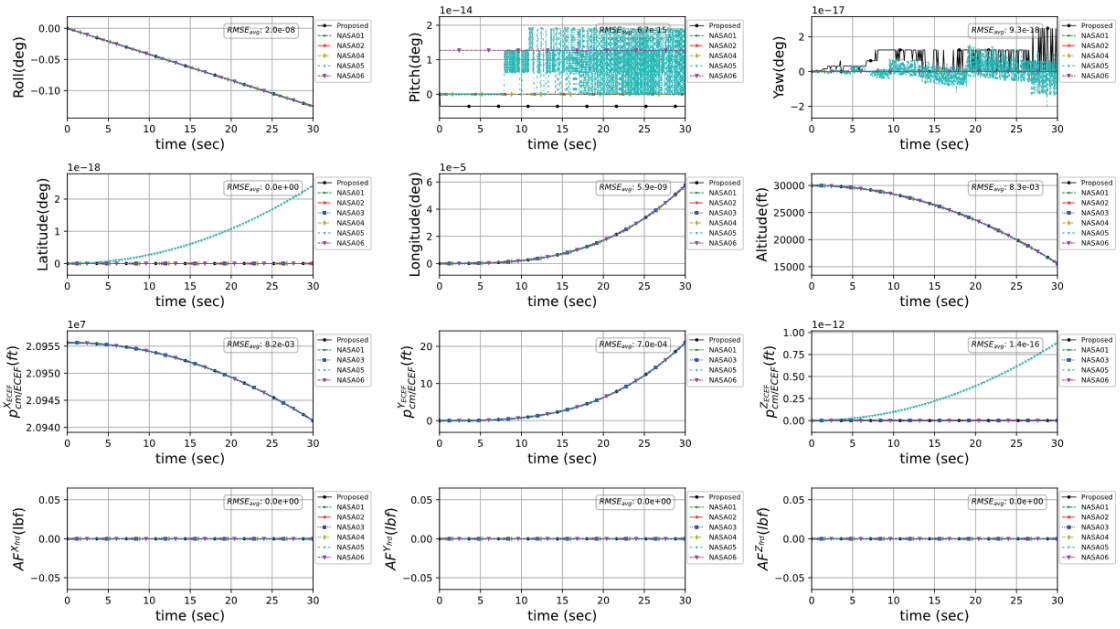

Fig. 3. (Color online) Dropped sphere with no drag.

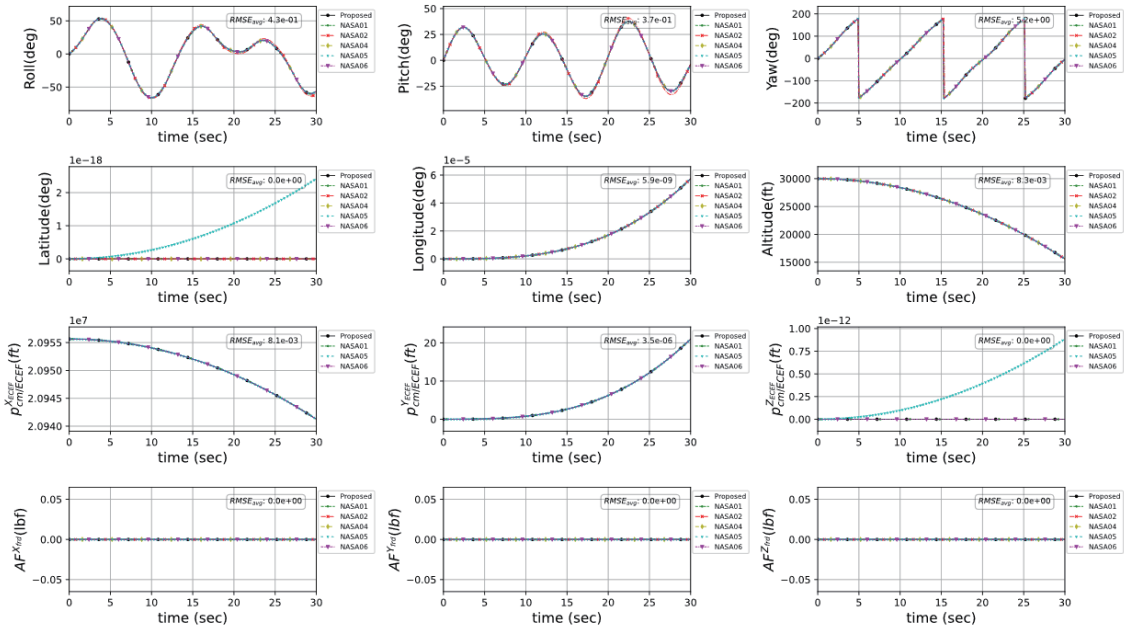

Fig. 4. (Color online) Tumbling brick with no damping, no drag.

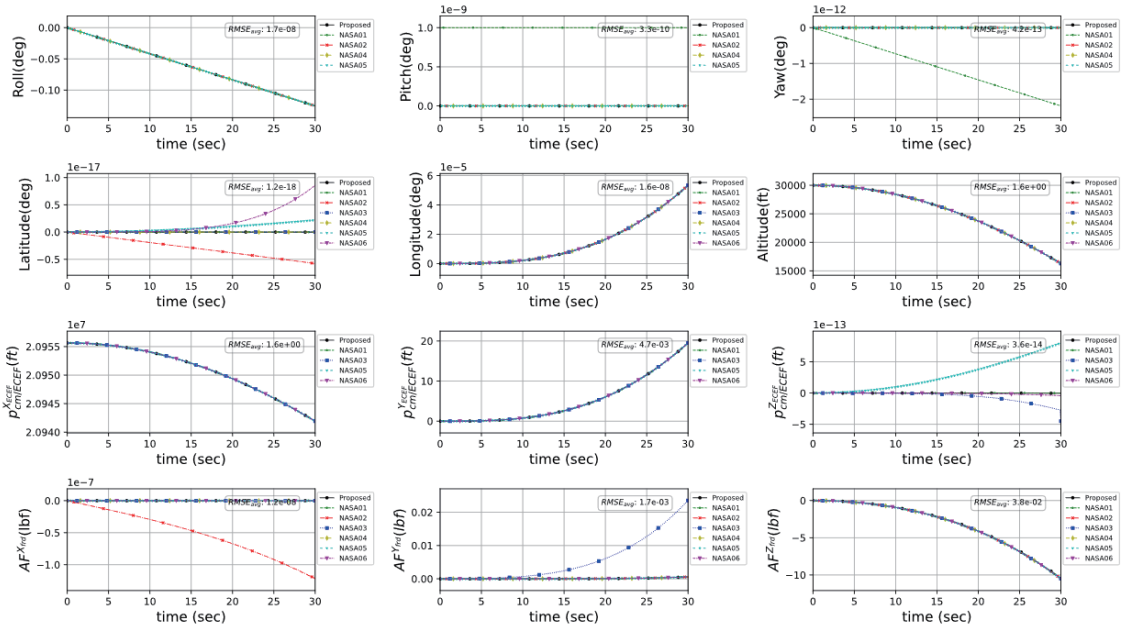

Fig. 5. (Color online) Dropped sphere with constant C_D, no wind.

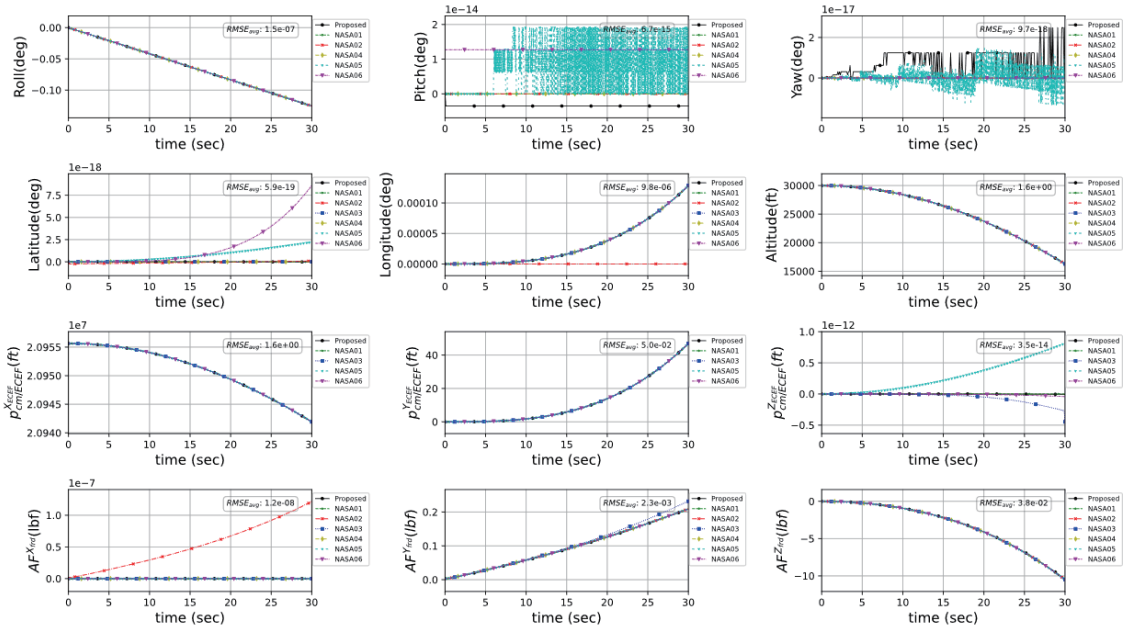

Fig. 6. (Color online) Dropped sphere with constant C_D, wind.

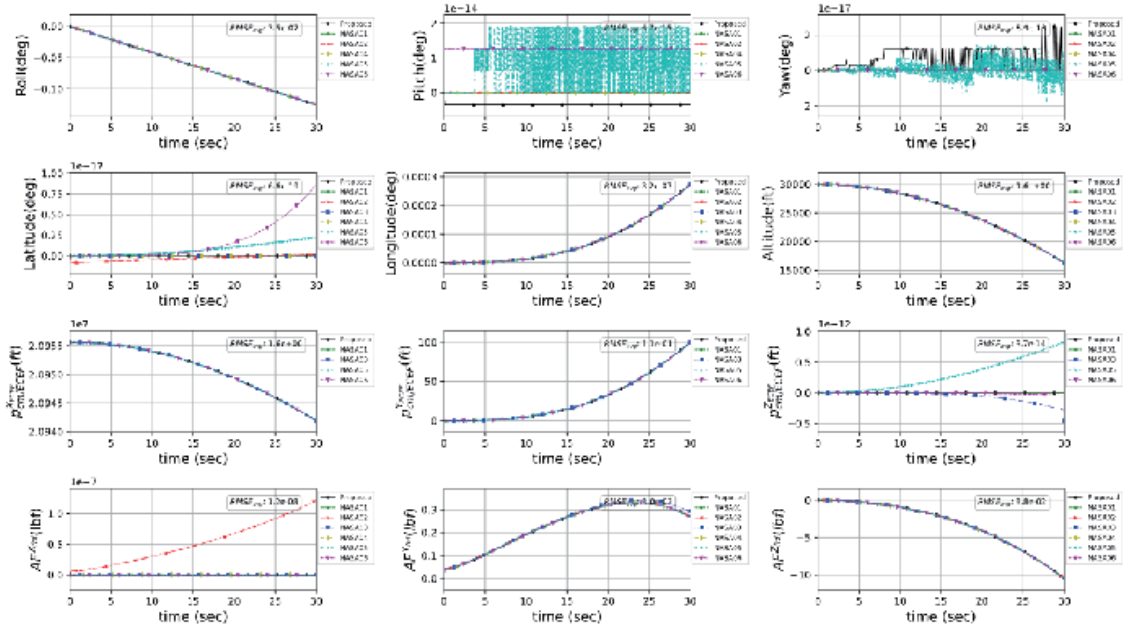

Fig. 7. (Color online) Dropped sphere with constant C_D, wind shear.

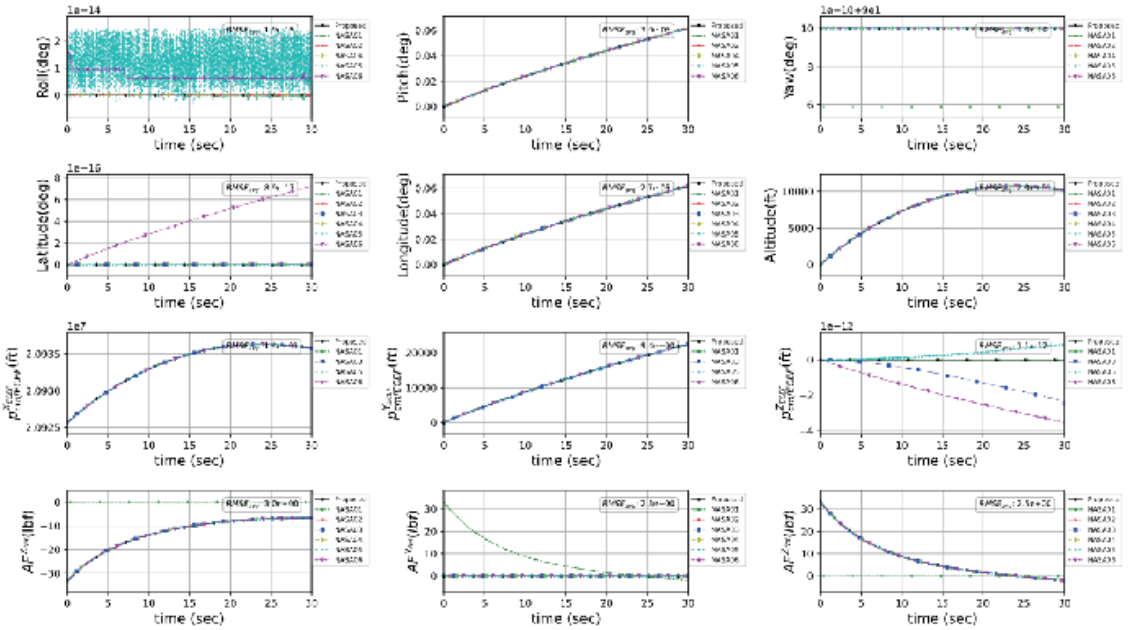

Fig. 8. (Color online) Sphere launched eastward along equator.

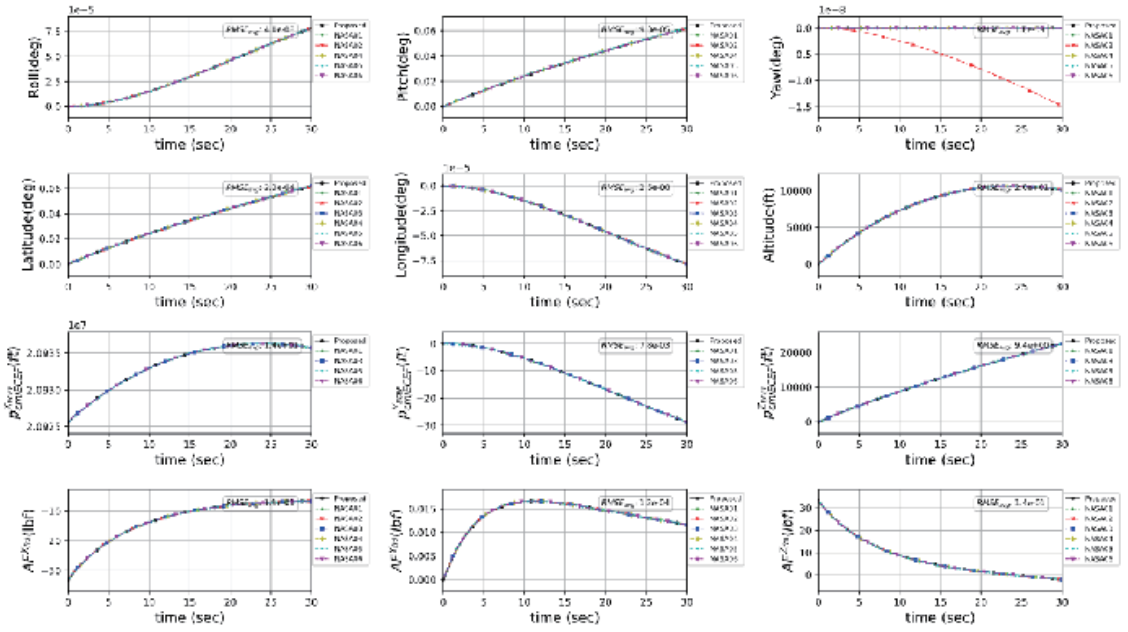

Fig. 9. (Color online) Sphere launched northward along prime meridian.

5. Conclusion and Future Research

We developed the simulation platform of 6-DOF flight dynamics from mathematical background and leveraged the computational power of Python open-source scientific libraries such as SciPy and SymPy, which were crafted under the OOP architecture design. To gain confidence in the purposed platform, the check-case simulation data from NASA were used to verify the simulation response in comparison with the proposed FDM. The majority of the simulation results well matched with the NASA baseline datasets in all cases with the highest $RMSE_{ave}$ of the altitude around 20.0 ft found in cases 6 and 7, which is seemingly negligible for a high cruising altitude at supersonic speed. This direct comparison also underscores the high accuracy of the proposed FDM under various flight conditions from subsonic to supersonic cruising speeds. In future research, we intend to use the proposed FDM as a physics-based model for developing intelligent flight controls under the supervision of a reinforcement learning agent.

Acknowledgments

This work was partially supported by the School of Engineering and was supported by King Mongkut's Institute of Technology Ladkrabang [Grant number 2566-02-01-017].

References

1 L. Shi, X. Wang, and Y. Cheng: IEEE Trans. Veh. Technol. **72** (2023) 9. https://doi.org/10.1109/TVT.2023.3264243
2 A. De Marco, P. M. D'Onza, and S. Manfredi: Nonlinear Dyn. **111** (2023) 17037. https://doi.org/10.1007/s11071-023-08725-y
3 E. Bøhn, J. Bjørnebekk, K. G. Bratli, J. H. Nordbø, and O. B. Fosso: IEEE Trans. Neural Netw. Learn. Syst. **35** (2024) 3. https://doi.org/10.1109/TNNLS.2023.3263430
4 S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll: IEEE Trans. Pattern Anal. Mach. Intell. **46** (2024) 12. https://doi.org/10.1109/TPAMI.2024.3457538
5 E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. A. Schultz: Microprocess. Microsyst. **61** (2018) 11. https://doi.org/10.1016/j.micpro.2018.05.002
6 T. E. Oliphant: Comput. Sci. Eng. **9** (2007) 3. https://doi.org/10.1109/MCSE.2007.58
7 D. G. Murri, E. B. Jackson, and R. O. Shelton: Langley Research Center (2015). https://ntrs.nasa.gov/citations/20150001264
8 L. Stevens, F. L. Lewis, and E. N. Johnson: Aircraft Control and Simulation: Dynamics, Control Design, and Autonomous Systems (Wiley, New Jersey, 2015).

About the Authors

Pattiwat Atayagul received his B.Eng and M.Eng degrees in mechanical engineering from King Mongkut's University of Technology Thonburi (KMUTT), Thailand, in 2015 and 2020, respectively. He is currently an application engineer of commercial finite element and system simulation software at ADT Systems (Asia Pacific) Co., Ltd. His scientific interests include intelligent computational systems and the applications of co-simulation.

Pitikhate Sooraksa received his B.Ed. (Hons.) and M.Sc. degrees in physics from Srinakharinwirot University, Thailand, his M.S. degree in electrical engineering from George Washington University, USA, in 1992, and his Ph.D. degree in electrical engineering from the University of Houston, USA, in 1996. He is currently a professor with the School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Thailand. His research interests include cyber-physical applications and rapid prototypes in robotics and AI.