

Article

Physics-Informed Graph Neural Network for Flight Dynamics Modeling

Liang Ma, Zhanwu Li, Juntao Zhang, You Li * and Shijie Deng *

Aviation Engineering School, Air Force Engineering University, Xi'an 710025, China; 13259405258@163.com (L.M.); afeulzw@189.cn (Z.L.); 17710707116@163.com (J.Z.)

* Correspondence: reason891020@126.com (Y.L.); dengshijiesilence@163.com (S.D.)

Abstract

Flight dynamics modeling is a fundamental cornerstone of aircraft design, simulation, and control. Traditional approaches rely on aerodynamic look-up tables for numerical integration, which suffer from high data-acquisition costs, poor extrapolation capability, and difficulty in assimilating flight test data. This paper proposes an architectural integration of physics-informed neural networks (PINNs), graph neural networks (GNNs), and known flight mechanics equations for flight dynamics modeling. Without requiring aerodynamic coefficient labels, the method predicts flight state derivatives using state-transition data. The approach encodes the structural knowledge of flight mechanics equations into graph topology and a physics computation layer (PhysicsLayer), so that the neural network only needs to learn the unknown aerodynamic coefficients while all remaining physical relationships are computed by the governing equations. Using an F-16 fighter six-degree-of-freedom model as the verification platform, an ablation study involving Direct-MLP, PINN, PIGNN, and GNN is conducted. Results show that the PIGNN architecture improves single-step derivative prediction accuracy by 86.6% over Direct-MLP, 60.9% over pure PINN, and 90.8% over GNN. In 499-step (approximately 5 s) rollout state prediction, the PIGNN Core RMSE is 1.1554, with approximately linear error growth within the first 100 steps indicating well-controlled short-range error accumulation. The graph-structural prior enables the network to learn aerodynamic coefficients that closely match the F-16 reference aerodynamic database without aerodynamic coefficient supervision. The results demonstrate that combining graph-based dependency modeling with hard physical constraints is effective for interpretable flight dynamics surrogate modeling.

Keywords: flight dynamics modeling; physics-informed neural network; graph neural network; aerodynamic coefficient prediction; flight simulation



Academic Editor: Karim Abu Salem

Received: 16 April 2026

Revised: 11 May 2026

Accepted: 14 May 2026

Published: 16 May 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid advancement of aviation technology, modern aircraft face increasingly stringent requirements for maneuverability, agility, and safety. Their flight envelopes continue to expand, frequently involving extreme flight conditions such as high angles of attack, high rotation rates, and even spin. In these complex flow fields dominated by nonlinear, unsteady aerodynamic forces, establishing an accurate flight dynamics model is a critical prerequisite for flight control design, performance evaluation, and flight safety analysis [1]. However, aerodynamic forces under extreme flight conditions exhibit strong nonlinearity, time-varying behavior, and severe flow separation, posing formidable challenges to existing modeling methods [2–4].

Traditional aircraft parameter estimation and system identification methods mainly include the output error method, the equation error method, and the filter error method. These methods typically rely on a prescribed mathematical model structure (e.g., Taylor series expansion or polynomial models) and assume that the system operates in a quasi-steady or weakly nonlinear regime [5]. Although these methods have been successful within conventional flight envelopes, their prescribed model structures often fail to capture the complex nonlinear dynamic characteristics encountered during high-angle-of-attack maneuvers, rapid maneuvers, or free-flight conditions, resulting in insufficient model generalization and significantly degraded identification accuracy [6].

In recent years, with the rise of artificial intelligence and data science, data-driven methods—especially neural networks—have provided new perspectives for flight dynamics modeling. Leveraging their powerful nonlinear function approximation capability [7], neural networks have been successfully applied to aerodynamic parameter estimation [8]. Early studies primarily used static neural networks to map state variables and control inputs to aerodynamic force coefficients, and extracted aerodynamic derivatives through post-processing techniques such as the Delta method, the Modified Delta method, or the neural partial differential method [9–11]. However, purely data-driven “black-box” models often lack physical interpretability and are prone to producing predictions that violate physical laws under small-sample or extrapolation conditions.

Physics-informed neural networks (PINNs), proposed by Raissi et al. [12], achieve the fusion of data-driven learning and physical constraints by embedding physical equations into the loss function or network architecture. PINNs have been widely applied in the flight dynamics domain. Sun et al. [13] used the PINN approach for integrated modeling and optimization of airfoil aerodynamic shapes. Michek et al. [14] employed an improved non-deterministic physics-informed neural network to directly utilize position, velocity, and acceleration measurements from F-16 flight data, training a parameter estimation network through a physics-informed loss function to simultaneously predict aerodynamic force and moment coefficients in six-degree-of-freedom nonlinear flight dynamics. Chen et al. [15] replaced the differential loss in the PINN with an integral loss, proposing a new online aerodynamic parameter identification method. Existing PINN methods typically adopt Multi-Layer Perceptrons (MLPs) as the backbone network, which cannot explicitly model the complex coupling relationships among flight state variables.

Graph neural networks (GNNs) model relationships between entities through graph structures and have demonstrated unique advantages in physical system modeling [16]. Sanchez-Gonzalez et al. [17] proposed interaction networks that were successfully applied to multi-body physics simulation. Pfaff et al. [18] applied a GNN to mesh-based physical simulation, achieving results superior to traditional numerical methods.

This paper proposes an architectural integration of the PINN, graph-based aerodynamic dependency modeling, and known flight mechanics equations. The main contributions are as follows:

1. A GNN architecture based on physics-prior graph topology is proposed, which encodes the physical dependency relationships between aerodynamic coefficients and flight states as edge connections in the graph, enabling the network to propagate information precisely along physically meaningful paths.
2. An end-to-end prediction framework incorporating a PhysicsLayer is designed, which hard-codes the known flight mechanics equations into the network structure, ensuring that the output strictly satisfies Newtonian mechanics constraints.
3. The learned aerodynamic coefficients are evaluated against the F-16 reference aerodynamic database without using aerodynamic coefficient labels during training, providing a physically interpretable validation of the latent variables.

2. Materials and Methods

2.1. F-16 Flight Dynamics Model

2.1.1. F-16 Physical Constants and Inertial Parameters

Before establishing the F-16 flight dynamics model, the physical constants and inertial parameters are defined. These parameters are determined by the aircraft geometry, mass distribution, and engine configuration, and are kept constant throughout the simulations. The data are taken from the classical reference by Stevens and Lewis [1]. Table 1 lists all constants and their numerical values used in this paper.

Table 1. F-16 physical constants and inertial parameters.

Symbol	Physical Meaning	Value
g	Gravitational acceleration	32.17 ft/s ²
m	Mass	637.4 slug
S	Reference area	300 ft ²
b	Wingspan	30 ft
$\bar{c}$	Mean aerodynamic chord	11.32 ft
h_x	Engine angular momentum	160 slug·ft ² /s
A_{XX}	Roll moment of inertia	9496 slug·ft ²
A_{YY}	Pitch moment of inertia	55,814 slug·ft ²
A_{ZZ}	Yaw moment of inertia	63,100 slug·ft ²
A_{XZ}	Product of inertia	982 slug·ft ²
Γ	Inertia coupling determinant, $A_{XX}A_{ZZ} - A_{XZ}^2$	598,636,076 slug ² ·ft ⁴
X_{PQ}	Inertia coupling coefficient, $A_{XZ}(A_{XX} - A_{YY} + A_{ZZ})$	16,918,892 slug ² ·ft ⁴
X_{QR}	Inertia coupling coefficient, $A_{ZZ}(A_{ZZ} - A_{YY}) + A_{XZ}^2$	462,847,124 slug ² ·ft ⁴
Z_{PQ}	Inertia coupling coefficient, $(A_{XX} - A_{YY})A_{XZ} + A_{XZ}^2$	−439,156,880 slug ² ·ft ⁴

Here, S , b , and $\bar{c}$ are wing geometric parameters that appear in the computation of aerodynamic forces and moments; A_{XX} , A_{YY} , A_{ZZ} , and A_{XZ} are the aircraft's moments and product of inertia, which determine the angular acceleration response to applied moments; and Γ , X_{PQ} , X_{QR} , and Z_{PQ} are auxiliary inertia combinations used to simplify the coupled roll–yaw moment equations. Additionally, $\bar{x}_{cg,ref} = 0.35$ is the reference center-of-gravity position (in units of mean aerodynamic chord), and x_{cg} is the actual center-of-gravity position. In this study, $x_{cg} = \bar{x}_{cg,ref}$ is used to isolate the modeling effect of the proposed network architecture from center-of-gravity variation. Modeling variable center-of-gravity effects is straightforward by treating x_{cg} as an additional input, but is outside the scope of the present validation.

2.1.2. State Vector and Control Inputs

Based on the above aircraft parameters, the F-16 fighter employs a six-degree-of-freedom rigid-body flight dynamics model [1]. This model treats the aircraft as a rigid body with six degrees of freedom (three translational and three rotational) in three-dimensional space, representing the most classical and complete description in flight dynamics modeling.

The state vector contains 13 variables:

$$\mathbf{X} = [V_T, \alpha, \beta, \phi, \theta, \psi, p, q, r, h, P, x_N, x_E]^T \quad (1)$$

The physical meaning and domain of each state variable are shown in Table 2.

Table 2. F-16 state vector definition and value ranges.

Symbol	Physical Meaning	Unit	Valid Range
V_T	True airspeed	ft/s	300–800
α	Angle of attack	rad	$-0.17-0.79$ ($-10-45^\circ$)
β	Sideslip angle	rad	$-0.52-0.44$ ($-30-25^\circ$)
ϕ	Roll angle	rad	$-\pi-\pi$
θ	Pitch angle	rad	$-\pi/2-\pi/2$
ψ	Yaw angle	rad	$0-2\pi$
p	Roll rate	rad/s	$-5.24-5.24$ ($\pm 300^\circ/\text{s}$)
q	Pitch rate	rad/s	$-3.49-3.49$ ($\pm 200^\circ/\text{s}$)
r	Yaw rate	rad/s	$-3.49-3.49$ ($\pm 200^\circ/\text{s}$)
h	Altitude	ft	0–40,000
P	Engine power state	%	0–100
x_N	Northward position	ft	$-\infty-+\infty$
x_E	Eastward position	ft	$-\infty-+\infty$

The selection of these 13 state variables follows the classical flight mechanics modeling paradigm and they can be grouped by physical attributes into four categories:

(1) Aerodynamic state group (V_T, α, β): True airspeed V_T describes the aircraft's velocity relative to the air and is the basis for computing dynamic pressure and aerodynamic forces. Angle of attack α and sideslip angle β describe the direction of the airflow relative to the body longitudinal and lateral axes, respectively, and are the most critical variables determining aerodynamic coefficients. These three variables jointly define the aircraft's motion state in the wind-axis frame and directly determine the magnitude of aerodynamic forces and moments.

(2) Attitude group (ϕ, θ, ψ): Roll angle ϕ , pitch angle θ , and yaw angle ψ form the Euler angle triplet, describing the orientation of the body frame relative to the ground frame. The attitude angles are coupled with angular rates through kinematic equations, and their derivatives are determined exactly by pure kinematic relationships without depending on aerodynamic coefficients.

(3) Angular rate group (p, q, r): Roll rate p , pitch rate q , and yaw rate r describe the aircraft's rotational velocities about the three body axes. Changes in angular rates are driven by aerodynamic moments, while simultaneously feeding back through damping derivative terms to influence aerodynamic coefficients, forming an aerodynamic–kinematic coupling.

(4) Auxiliary state group (h, P, x_N, x_E): Altitude h affects atmospheric density and engine thrust and is an important parameter for aerodynamic force computation. Engine power state P describes the engine response dynamics, and its evolution is computed exactly by a known engine model. Northward position x_N and eastward position x_E describe the aircraft's horizontal position in the ground frame, and their derivatives are determined by pure kinematic relationships.

The valid ranges of α and β are determined by the interpolation range of the aerodynamic data tables; extrapolation results beyond these ranges are unreliable.

The control input contains 4 variables:

$$\mathbf{U} = [\delta_{th}, \delta_e, \delta_a, \delta_r]^T \quad (2)$$

The domain of each control variable is shown in Table 3.

The selection of control inputs covers all major control channels of the F-16 fighter. Throttle δ_{th} controls engine thrust, normalized to the $[0, 1]$ interval. Elevator δ_e is the primary longitudinal control surface, controlling the aircraft's pitching motion by altering the pitching moment. Aileron δ_a produces a rolling moment through differential deflection and is the primary means of lateral control. Rudder δ_r generates a yawing moment for

heading control and coordinated turns. These four control variables influence the aircraft's forces and moments through the aerodynamic coefficient model, thereby driving the state evolution.

Table 3. F-16 control input definition and value ranges.

Symbol	Physical Meaning	Unit	Valid Range
δ_{th}	Throttle setting	—	0–1
δ_e	Elevator deflection	°	–25–25
δ_a	Aileron deflection	°	–20–20
δ_r	Rudder deflection	°	–30–30

2.1.3. Aerodynamic Coefficient Model

The aerodynamic characteristics of the F-16 are described by six total aerodynamic coefficients, each composed of a static aerodynamic coefficient, control surface effects, and damping derivative corrections. The complete formulas for each coefficient are given below, where the geometric parameters such as wingspan b and mean aerodynamic chord $\bar{c}$ are listed in Table 1. These formulas reveal the physical dependency relationships between aerodynamic coefficients and flight states, forming the basis for constructing the graph topology in this paper.

(1) Axial force coefficient C_{XT} :

$$C_{XT} = C_X(\alpha, \delta_e) + \frac{\bar{c}q}{2V_T} \cdot C_{X_q}(\alpha) \quad (3)$$

where $C_X(\alpha, \delta_e)$ is the static axial force coefficient obtained from a two-dimensional α – δ_e look-up table, and $C_{X_q}(\alpha)$ is the pitch damping derivative obtained from a one-dimensional α look-up table. Therefore, C_{XT} depends on α , q , δ_e , and V_T .

(2) Side force coefficient C_{YT} :

$$C_{YT} = -0.02\beta + 0.021\frac{\delta_a}{20} + 0.086\frac{\delta_r}{30} + \frac{b}{2V_T} \left(C_{Y_r}(\alpha) \cdot r + C_{Y_p}(\alpha) \cdot p \right) \quad (4)$$

where the static side force is a linear function of β , the control surface effects are linear functions of δ_a and δ_r , and the damping terms depend on p , r , and α . Therefore, C_{YT} depends on β , p , r , δ_a , δ_r , α , and V_T .

(3) Normal force coefficient C_{ZT} :

$$C_{ZT} = C_Z(\alpha) \left(1 - \left(\frac{\beta}{57.3} \right)^2 \right) - 0.19\frac{\delta_e}{25} + \frac{\bar{c}q}{2V_T} \cdot C_{Z_q}(\alpha) \quad (5)$$

where $C_Z(\alpha)$ is the static normal force coefficient obtained from a one-dimensional α look-up table, with a quadratic correction by β . Therefore, C_{ZT} depends on α , β , q , δ_e , and V_T .

(4) Rolling moment coefficient C_{LT} :

$$C_{LT} = C_L(\alpha, \beta) + \Delta C_{L_{\delta_a}}(\alpha, \beta)\frac{\delta_a}{20} + \Delta C_{L_{\delta_r}}(\alpha, \beta)\frac{\delta_r}{30} + \frac{b}{2V_T} \left(C_{L_r}(\alpha) \cdot r + C_{L_p}(\alpha) \cdot p \right) \quad (6)$$

where $C_L(\alpha, \beta)$ is the static rolling moment, and $\Delta C_{L_{\delta_a}}$ and $\Delta C_{L_{\delta_r}}$ are the rolling moment increments due to aileron and rudder, all obtained from two-dimensional α – β look-up tables. Therefore, C_{LT} depends on α , β , p , r , δ_a , δ_r , and V_T .

(5) Pitching moment coefficient C_{MT} :

$$C_{MT} = C_M(\alpha, \delta_e) + \frac{\bar{c}q}{2V_T} \cdot C_{M_q}(\alpha) + C_{ZT}(\bar{x}_{cg,ref} - x_{cg}) \quad (7)$$

where $C_M(\alpha, \delta_e)$ is the static pitching moment obtained from a two-dimensional α - δ_e look-up table. The last term is the center-of-gravity offset correction; the inclusion of C_{ZT} makes C_{MT} indirectly dependent on β . Therefore, C_{MT} depends on $\alpha, \beta, q, \delta_e$, and V_T .

(6) Yawing moment coefficient C_{NT} :

$$C_{NT} = C_N(\alpha, \beta) + \Delta C_{N_{\delta_a}}(\alpha, \beta) \frac{\delta_a}{20} + \Delta C_{N_{\delta_r}}(\alpha, \beta) \frac{\delta_r}{30} + \frac{b}{2V_T} \left(C_{N_r}(\alpha) \cdot r + C_{N_p}(\alpha) \cdot p \right) - C_{YT} \frac{(\bar{x}_{cg,ref} - x_{cg})\bar{c}}{b} \quad (8)$$

where $C_N(\alpha, \beta)$ is the static yawing moment, and $\Delta C_{N_{\delta_a}}$ and $\Delta C_{N_{\delta_r}}$ are the control surface yawing moment increments. The last term is the center-of-gravity offset correction; the inclusion of C_{YT} makes C_{NT} indirectly dependent on δ_a and δ_r . Therefore, C_{NT} depends on $\alpha, \beta, p, r, \delta_a, \delta_r$, and V_T .

Summarizing the above analysis, Table 4 presents the physical dependency relationships of each aerodynamic coefficient. These dependencies directly determine the edge connections in the graph topology of the PIGNN architecture.

Table 4. Physical dependencies between aerodynamic coefficients and flight states.

Aerodynamic Coefficient	Dependent State/Control Variables
C_{XT}	V_T, α, q, δ_e
C_{YT}	$V_T, \alpha, \beta, p, r, \delta_a, \delta_r$
C_{ZT}	$V_T, \alpha, \beta, q, \delta_e$
C_{LT}	$V_T, \alpha, \beta, p, r, \delta_a, \delta_r$
C_{MT}	$V_T, \alpha, \beta, q, \delta_e$
C_{NT}	$V_T, \alpha, \beta, p, r, \delta_a, \delta_r$

It should be noted that in the neural network framework of this paper, control surface deflections ($\delta_e, \delta_a, \delta_r$) are injected into the features of all nodes through the control input vector $\mathbf{U}$, rather than as independent graph nodes. Therefore, the edge connections of aerodynamic coefficient nodes in the graph topology only involve state variables ($V_T, \alpha, \beta, p, q, r$), and the influence of control surfaces is implicitly conveyed through node features.

2.1.4. Equations of Motion

Based on the above state variables, control inputs, and aerodynamic coefficient model, the evolution of flight states is described by the six-degree-of-freedom equations of motion. The state derivatives $\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U}, \mathbf{C})$ are computed by the following equation set, where the physical constants and inertial parameters are given in Table 1. The adopted formulation follows the standard F-16 body-axis convention used in the flight dynamics literature [1]: the body-fixed x -axis points forward along the fuselage, the y -axis points toward the right wing, and the z -axis points downward. The aerodynamic force coefficients used in this paper are therefore body-axis components rather than direct lift and drag coefficients; the aerodynamic force is resolved as $X_A = QSC_X$, $Y_A = QSC_Y$, and $Z_A = QSC_Z$ before entering the translational dynamics. The aerodynamic moments L_m , M_m , and N_m act about the corresponding body axes, thrust is assumed to act along the body x -axis, and gravity is introduced through the standard Euler-angle transformation in the body-axis velocity equations.

In the following equations, the newly introduced symbols are defined as follows: T_K is the atmospheric temperature at the current altitude (K); $T_0 = 288.15$ K is the sea-level standard temperature; p_0 is the sea-level standard pressure; ρ is the atmospheric density;

R is the specific gas constant for air; $\gamma = 1.4$ is the ratio of specific heats; $f_{t2m} = 0.3048$ is the feet-to-meters conversion factor; f_{pa2lbf} is the Pascals-to-pounds-per-square-foot conversion factor; Ma is the Mach number; Q is the dynamic pressure (lbf/ft²); u, v, w are the body-axis velocity components; a_x, a_y, a_z are the body-axis acceleration components; T is the engine thrust; L_m, M_m, N_m are the rolling, pitching, and yawing aerodynamic moments, respectively; and f_{thrust} and f_{pdot} are empirical model functions for thrust and power derivative. The atmospheric calculation is performed in SI units to obtain temperature, density, and Mach number, and the dynamic pressure is then converted back to lbf/ft² for consistency with the English-unit F-16 equations. All other symbols retain their previously defined meanings.

(1) Atmospheric model. Temperature, density, Mach number, and dynamic pressure are computed according to the International Standard Atmosphere model:

$$T_K = \max(288.15 - 0.0065 \cdot h \cdot f_{t2m}, 216.65), \quad \rho = \frac{p_0(T_K/T_0)^{5.256}}{RT_K} \quad (9)$$

$$Ma = \frac{V_T \cdot f_{t2m}}{\sqrt{\gamma RT_K}}, \quad Q = \frac{1}{2} \rho (V_T \cdot f_{t2m})^2 \cdot f_{pa2lbf} \quad (10)$$

(2) Body-axis velocity components.

$$u = V_T \cos \alpha \cos \beta, \quad v = V_T \sin \beta, \quad w = V_T \sin \alpha \cos \beta \quad (11)$$

(3) Force equations. Aerodynamic coefficients and thrust are converted to body-axis accelerations:

$$a_x = \frac{QSC_X + T}{m}, \quad a_y = \frac{QSC_Y}{m}, \quad a_z = \frac{QSC_Z}{m} \quad (12)$$

(4) Body-axis velocity derivatives.

$$\dot{u} = rv - qw - g \sin \theta + a_x \quad (13)$$

$$\dot{v} = pw - ru + g \cos \theta \sin \phi + a_y \quad (14)$$

$$\dot{w} = qu - pv + g \cos \theta \cos \phi + a_z \quad (15)$$

(5) Wind-axis derivatives.

$$\dot{V}_T = \frac{u\dot{u} + v\dot{v} + w\dot{w}}{V_T} \quad (16)$$

$$\dot{\alpha} = \frac{u\dot{w} - w\dot{u}}{u^2 + w^2} \quad (17)$$

$$\dot{\beta} = \frac{V_T \dot{v} - v \dot{V}_T}{u^2 + w^2} \cos \beta \quad (18)$$

Equations (16)–(18) are obtained by differentiating the wind-axis definitions of V_T , α , and β and substituting the body-axis velocity derivatives in Equations (13)–(15). They therefore represent transformed translational dynamics rather than independent empirical equations.

(6) Attitude kinematic equations (independent of aerodynamic coefficients; exactly known).

$$\dot{\phi} = p + \frac{\sin \theta}{\cos \theta} (q \sin \phi + r \cos \phi) \quad (19)$$

$$\dot{\theta} = q \cos \phi - r \sin \phi \quad (20)$$

$$\dot{\psi} = \frac{q \sin \phi + r \cos \phi}{\cos \theta} \quad (21)$$

(7) Moment equations. The aerodynamic moments are:

$$L_m = Q S b C_L, \quad M_m = Q S \bar{c} C_M, \quad N_m = Q S b C_N \quad (22)$$

The angular accelerations are:

$$\dot{p} = \frac{X_{PQ} p q - X_{QR} q r + A_{ZZ} L_m + A_{XZ} (N_m + q h_x)}{\Gamma} \quad (23)$$

$$\dot{q} = \frac{(A_{ZZ} - A_{XX}) p r - A_{XZ} (p^2 - r^2) + M_m - r h_x}{A_{YY}} \quad (24)$$

$$\dot{r} = \frac{Z_{PQ} p q - X_{PQ} q r + A_{XZ} L_m + A_{XX} (N_m + q h_x)}{\Gamma} \quad (25)$$

(8) Position derivatives (independent of aerodynamic coefficients; exactly known).

$$\dot{h} = u \sin \theta - v \sin \phi \cos \theta - w \cos \phi \cos \theta \quad (26)$$

$$\dot{x}_N = u \cos \theta \cos \psi + v (\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi) + w (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \quad (27)$$

$$\dot{x}_E = u \cos \theta \sin \psi + v (\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi) + w (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \quad (28)$$

(9) Thrust and engine model (known model; computed exactly).

$$T = f_{thrust}(P, h, Ma), \quad \dot{P} = f_{pdot}(P, \delta_{th}) \quad (29)$$

The thrust model is based on three 6×6 empirical look-up tables (idle, military, and maximum thrust), obtained through bilinear interpolation and power-state weighting.

2.1.5. Problem Definition

Based on the above flight dynamics model, the flight state prediction problem can be formalized as follows: given the current state $\mathbf{X}_t$ and control input $\mathbf{U}_t$, predict the state derivatives $\dot{\mathbf{X}}_t$, and then obtain the next-step state through Euler integration:

$$\mathbf{X}_{t+1} = \mathbf{X}_t + \dot{\mathbf{X}}_t \cdot \Delta t \quad (30)$$

The core challenge is that the aerodynamic coefficients $\mathbf{C}$ cannot be directly measured during actual flight and must be implicitly learned from state-transition data.

2.2. Method Design

2.2.1. Overall Framework

The core idea of this paper is to incorporate the governing equations of flight mechanics into the neural network training process, following the philosophy of physics-informed neural networks (PINNs) [12]. For the F-16 flight dynamics system, the state derivatives are determined by the governing equations:

$$\dot{\mathbf{X}} = \mathcal{F}(\mathbf{X}, \mathbf{U}; \mathbf{C}) \quad (31)$$

where the aerodynamic coefficients $\mathbf{C}$ are unknown parameters. PINN incorporates physical knowledge in two ways: (1) **soft constraints**—including the physical equation residuals as part of the loss function; (2) **hard constraints**—embedding the physical equations di-

rectly into the network computation graph (PhysicsLayer). This paper employs both approaches simultaneously.

Meanwhile, this paper introduces graph neural networks (GNNs) to model the coupling relationships among flight state variables. GNN updates node representations on graph structures through a message-passing mechanism:

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}), \quad \mathbf{h}_i^{(l+1)} = \phi_v\left(\mathbf{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\right) \quad (32)$$

where $\mathbf{h}_i^{(l)}$ is the hidden vector of node i at layer l , $\mathbf{e}_{ij}$ is the edge feature, and ϕ_e and ϕ_v are learnable message and update functions. The key advantage of GNN is that the graph topology can encode the physical dependency relationships between aerodynamic coefficients and flight states, allowing information to propagate only along physically meaningful paths.

Based on the fusion of these two methods, three physics-informed architectures are designed for comparative study. A conventional Direct-MLP surrogate is also used as an external neural baseline:

- **Direct-MLP:** MLP directly predicts the 13-dimensional state derivatives from the current state, control input, Mach number, and dynamic pressure. It has neither a graph-structural prior nor a PhysicsLayer hard constraint.
- **PINN:** MLP predicts aerodynamic coefficients $\rightarrow$ PhysicsLayer computes derivatives. Physical knowledge is only reflected at the output end (PhysicsLayer hard constraint), with no graph-structural prior at the input end.
- **PIGNN:** GNN predicts aerodynamic coefficients on a physics-prior graph $\rightarrow$ PhysicsLayer computes derivatives. Physical knowledge is reflected at both the input end (graph topology) and the output end (PhysicsLayer).
- **GNN:** GNN directly predicts derivatives on an augmented physics-prior graph, without PhysicsLayer. Physical knowledge is only reflected at the input end (graph topology), while the output end is purely data-driven.

2.2.2. Direct-MLP Baseline

The Direct-MLP baseline is a conventional data-driven surrogate model. It uses the same 19-dimensional input as PINN, including all 13 state variables, 4 control inputs, Mach number, and dynamic pressure, but directly outputs the 13 state derivatives:

$$\hat{\mathbf{X}} = \text{MLP}(\mathbf{X}, \mathbf{U}, Ma, Q) \quad (33)$$

The network structure is $19 \rightarrow 512 \rightarrow 512 \rightarrow 256 \rightarrow 13$, with LayerNorm and SiLU activation functions. This baseline is included to distinguish the proposed physics-informed architecture from a standard neural surrogate without graph message passing or hard physical constraints.

2.2.3. PINN Architecture

The PINN architecture employs a 4-layer MLP to predict 6 aerodynamic coefficients:

$$\mathbf{C} = \text{MLP}([V_T, \alpha, \beta, p, q, r, P, \phi, \theta, \psi, h, x_N, x_E, \delta_e, \delta_a, \delta_r, \delta_{th}, Ma, Q]; \boldsymbol{\theta}) \quad (34)$$

The network structure is $19 \rightarrow 256 \rightarrow 256 \rightarrow 128 \rightarrow 6$, with LayerNorm and SiLU activation functions. The MLP receives all 13 state variables, 4 control inputs, Mach number, and dynamic pressure (19 dimensions total), fully connecting them to 6 outputs without distin-

guishing which inputs affect which aerodynamic coefficient. The predicted aerodynamic coefficients are fed into the PhysicsLayer, where the flight mechanics equations exactly compute the 13-dimensional state derivatives.

The PhysicsLayer is the core component of the proposed method. It hard-codes the flight mechanics equations from Section 2.1.4 into the network's computation graph. The PhysicsLayer itself has no trainable weights; it is a deterministic differentiable computation module. The aerodynamic coefficients supplied to it are predicted by the neural network, and gradients are backpropagated through the PhysicsLayer to train that network.

Given the current state $\mathbf{X}$, control input $\mathbf{U}$, and the network-predicted 6 aerodynamic coefficients $\hat{\mathbf{C}}$, the PhysicsLayer computation proceeds as follows:

1. Compute atmospheric parameters Ma and Q from Equations (9) and (10);
2. Compute thrust T and power derivative $\dot{P}$ exactly using the known thrust model from Equation (29);
3. Compute body-axis velocity components u, v, w from Equation (11);
4. Convert $\hat{\mathbf{C}}$ and T to body-axis accelerations a_x, a_y, a_z from Equation (12);
5. Compute body-axis velocity derivatives $\dot{u}, \dot{v}, \dot{w}$ from Equations (13)–(15);
6. Compute wind-axis derivatives $\dot{V}_T, \dot{\alpha}, \dot{\beta}$ from Equations (16)–(18);
7. Compute attitude kinematics $\dot{\phi}, \dot{\theta}, \dot{\psi}$ exactly from Equations (19)–(21);
8. Compute angular accelerations $\dot{p}, \dot{q}, \dot{r}$ from Equations (22)–(25);
9. Compute position derivatives $\dot{h}, \dot{x}_N, \dot{x}_E$ exactly from Equations (26)–(28).

The key property of the PhysicsLayer is that among the 13-dimensional output derivatives, the attitude kinematics in step 7 ($\dot{\phi}, \dot{\theta}, \dot{\psi}$), the position derivatives in step 9 ($\dot{h}, \dot{x}_N, \dot{x}_E$), and the engine derivative ($\dot{P}$) are computed exactly from the current state and known models, independent of the network-predicted aerodynamic coefficients $\hat{\mathbf{C}}$. Therefore, the prediction error for these 7 dimensions is zero. The network only needs to accurately predict 6 aerodynamic coefficients, while the derivatives for the remaining dimensions are given exactly by the physical equations. This design substantially reduces the learning difficulty for the network while ensuring the physical self-consistency of the output.

2.2.4. PIGNN Architecture

The PIGNN architecture introduces graph neural networks on top of PINN, explicitly encoding the physical dependency relationships between aerodynamic coefficients and flight states as graph topology. This is the core method proposed in this paper.

Physics-Prior Graph Structure

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with 19 nodes is constructed: state nodes (13) correspond to each component of the 13-dimensional state vector, and aerodynamic coefficient nodes (6) correspond to the 6 aerodynamic coefficients as intermediate latent variables. Nodes are grouped by physical attributes into 5 categories: aerodynamic state group (V_T, α, β), attitude group (ϕ, θ, ψ), angular rate group (p, q, r), auxiliary group (h, P, x_N, x_E), and aerodynamic coefficient group ($C_X \sim C_N$).

The edge set $\mathcal{E}$ contains three types: (1) **Intra-group edges**: Full connections among state variables within the same group, reflecting the strong coupling of co-grouped variables. (2) **Inter-group edges**: State-to-state couplings based on the physical equations. (3) **Physical dependency edges**: Connections between aerodynamic coefficient nodes and their physically dependent state nodes, established strictly according to the dependency relationships in Table 4. All edges are bidirectional.

Node Feature Construction

The initial feature of each node is a 14-dimensional vector:

$$\mathbf{h}_i^{(0)} = [s_i, \mathbf{n}_i, \mathbf{u}, Ma, Q, \alpha, \beta] \quad (35)$$

where s_i is the state value corresponding to the node (initialized to 0 for aerodynamic coefficient nodes), $\mathbf{n}_i$ is a 5-dimensional node-type encoding, $\mathbf{u}$ is the 4-dimensional control input, and Ma and Q are atmospheric parameters.

Message Passing and Decoding

After the node features are mapped to 128-dimensional hidden vectors by an encoder, 3 rounds of message passing are performed:

$$\mathbf{m}_{ij} = \text{MLP}_e([\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}]) \quad (36)$$

$$\mathbf{h}_i^{(l+1)} = \text{LayerNorm}\left(\mathbf{h}_i^{(l)} + \text{MLP}_v\left([\mathbf{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}]\right)\right) \quad (37)$$

After 3 rounds of message passing, an attention mechanism is used to decode 6 aerodynamic coefficient values from the aerodynamic coefficient nodes:

$$\hat{\mathbf{C}}_k = \text{MLP}_k([\mathbf{h}_{C_k}^{(L)}, \text{Attn}(\mathbf{h}_{C_k}^{(L)}, \{\mathbf{h}_j^{(L)}\}_{j \in \text{deps}(k)})]) \quad (38)$$

The decoded aerodynamic coefficients are then fed into the PhysicsLayer to compute derivatives.

2.2.5. GNN Architecture

The GNN architecture serves as an ablation control group for the PhysicsLayer. It uses the same 19-node graph structure as PIGNN and adds dependency edges from aerodynamic coefficient nodes to state-derivative nodes according to the equations of motion. For example, C_X and C_Z are connected to the translational derivatives, while C_L and C_N are connected to the roll and yaw acceleration channels. After message passing, the 13-dimensional derivatives are decoded directly from the hidden vectors of the 13 state nodes. No PhysicsLayer is used in this architecture, so the known kinematic and dynamic relationships must be learned from data.

A schematic illustration of the three architectures is shown in Figure 1.

2.2.6. Training Strategy

Loss Function

For architectures with PhysicsLayer (PINN, PIGNN), the loss function includes data loss and physical residual losses:

$$\mathcal{L} = \mathcal{L}_{data} + \lambda_{kin}\mathcal{L}_{kin} + \lambda_{alt}\mathcal{L}_{alt} + \lambda_{eng}\mathcal{L}_{eng} + \lambda_{nav}\mathcal{L}_{nav} + \lambda_{mom}\mathcal{L}_{mom} + \lambda_{vel}\mathcal{L}_{vel} \quad (39)$$

where $\mathcal{L}_{data}$ is the normalized derivative MSE, and the 6 physical residual terms respectively constrain the self-consistency of the kinematic, altitude, engine, navigation, moment, and velocity equations. For GNN, only the data loss and the kinematic residuals that do not depend on aerodynamic coefficients are used. A conventional Direct-MLP baseline, when used for comparison, is trained only with $\mathcal{L}_{data}$ because it does not contain either graph-structured message passing or a PhysicsLayer.

The loss weights used in all reported experiments are $\lambda_{kin} = 0.5$, $\lambda_{alt} = 0.3$, $\lambda_{eng} = 0.3$, $\lambda_{nav} = 0.3$, $\lambda_{mom} = 0.5$, and $\lambda_{vel} = 0.5$. All trainable layers are initialized with Xavier normal initialization and zero bias. The optimizer is AdamW with an initial learning rate

of 10^{-3} and a weight decay of 10^{-5} . A cosine annealing learning-rate schedule is used with $T_{\max} = 200$ epochs and a minimum learning rate of 10^{-6} . The batch size is 1024, the maximum number of epochs is 200, and gradient clipping with a maximum norm of 10 is applied. The checkpoint with the lowest training data loss is used for evaluation.

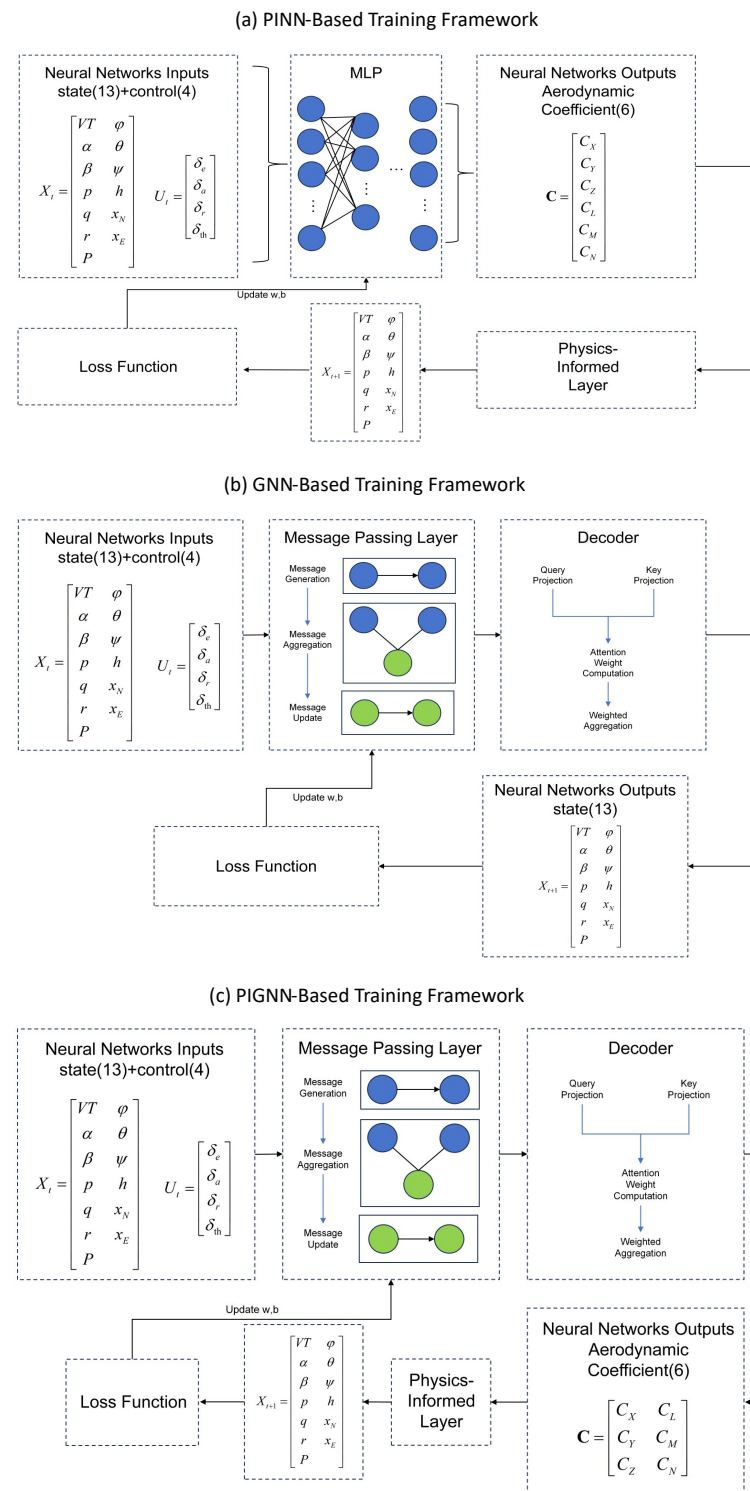


Figure 1. Schematic illustration of the three training frameworks. Arrows indicate the direction of data flow and information propagation, and different colors distinguish the neural network modules, physics-based computation layers, and output quantities.

3. Results

3.1. Experimental Setup

3.1.1. Dataset

Training data were generated by an F-16 six-degree-of-freedom simulator, comprising four categories of flight data: Dataset A (single-variable independent variation), Dataset B (multi-variable coupled variation), Dataset C (typical maneuver actions), and Dataset D (extreme operating conditions). The training set contains 374,880 samples and the test set contains 129,306 samples, covering a flight envelope of 300–800 ft/s in airspeed and 0–40,000 ft in altitude. Each sample includes the current state (13-dimensional), control input (4-dimensional), and state derivatives (13-dimensional), but does not include aerodynamic coefficients.

3.1.2. Ablation Study Design

To systematically evaluate the contributions of the physics-prior graph structure and the PhysicsLayer hard constraint—the two key design elements—to prediction accuracy, an ablation study with three main architectures is designed, as shown in Table 5. The three architectures are: PINN (PhysicsLayer only, no graph structure), PIGNN (both graph structure and PhysicsLayer, the complete method proposed in this paper), and GNN (graph structure only, no PhysicsLayer). Comparing PINN with PIGNN evaluates the contribution of the graph-structural prior, while comparing PIGNN with GNN evaluates the contribution of the PhysicsLayer hard constraint. In addition, a conventional Direct-MLP surrogate is included as an external neural baseline that directly maps the 19-dimensional input to 13 state derivatives without graph message passing or hard physical constraints.

The trainable parameter counts differ because the graph-based models contain node encoders, edge/message functions, and decoders in addition to ordinary feed-forward layers. Therefore, the comparison should be interpreted primarily as an architectural ablation rather than a strictly parameter-matched capacity study. The fact that the GNN baseline has the largest parameter count but the largest prediction error also indicates that parameter count alone does not explain the performance difference; the hard physical constraint introduced by the PhysicsLayer is essential for stable flight dynamics prediction.

Table 5. Ablation study design. A check mark (✓) indicates that the corresponding component is included, whereas a cross (×) indicates that it is not included.

Architecture	Physics-Prior Graph	PhysicsLayer	Trainable Parameters
Direct-MLP	×	×	410,125
PINN	×	✓	105,862
PIGNN	✓	✓	856,454
GNN	✓	×	1,455,885

3.1.3. Evaluation Metrics

This paper adopts Root Mean Square Error (RMSE) as the primary evaluation metric.

Single-step derivative RMSE: For the j -th state dimension, the single-step derivative prediction RMSE is defined as

$$\text{RMSE}_j = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{X}_{i,j} - \dot{X}_{i,j})^2} \quad (40)$$

where $\hat{X}_{i,j}$ is the predicted derivative of the i -th sample for the j -th dimension, $\dot{X}_{i,j}$ is the ground-truth derivative, and N is the number of test samples.

Core state RMSE (Core RMSE): To comprehensively evaluate the prediction accuracy of the core flight dynamics states, the Core(9) RMSE is defined as the joint RMSE over nine core state dimensions:

$$\text{Core RMSE} = \sqrt{\frac{1}{N \times 9} \sum_{i=1}^N \sum_{j \in \mathcal{S}_{\text{core}}} (\hat{X}_{i,j} - \dot{X}_{i,j})^2} \quad (41)$$

where $\mathcal{S}_{\text{core}} = \{V_T, \alpha, \beta, \phi, \theta, \psi, p, q, r\}$ is the core state set. These nine dimensions are selected because they directly reflect the aircraft's aerodynamic state, attitude, and angular motion characteristics, which are the primary concerns of flight dynamics modeling. The position state derivatives (h, x_N, x_E) are computed exactly by pure kinematic equations, and the power state derivative (P) is computed exactly by the known engine model; for architectures with PhysicsLayer, these errors are zero and are therefore excluded from the Core RMSE.

Multi-step rollout RMSE: For K -step rollout prediction, autoregressive prediction is performed on each test segment. Let there be M test segments, and let $e_{m,k}$ be the core state RMSE of the m -th segment at step k . The mean and standard deviation are reported:

$$\text{Rollout Core RMSE}(k) = \frac{1}{M} \sum_{m=1}^M e_{m,k} \pm \sqrt{\frac{1}{M} \sum_{m=1}^M (e_{m,k} - \bar{e}_k)^2} \quad (42)$$

Aerodynamic coefficient RMSE: Only for architectures with PhysicsLayer (PINN, PIGNN), the six latent variables predicted by the network are compared post hoc with the ground-truth aerodynamic coefficients in the test set. It must be emphasized that aerodynamic coefficient labels are never used during training; this metric is used to verify whether the network has learned physically correct aerodynamic coefficient decompositions from state-transition supervision alone.

3.2. Training Process and Analysis

Figure 2 shows the training behavior of the three architectures. Both the total training loss (left panel) and the data fitting loss (right panel) indicate that PIGNN (red) performs best, with its loss decreasing fastest and ultimately converging to the lowest level (approximately 10^{-5}), far superior to PINN (blue) and GNN (green). PINN converges at a moderate rate, stabilizing around 10^{-3} , while GNN performs worst, with its loss only decreasing to approximately 10^0 and converging slowly. Notably, PIGNN exhibits pronounced oscillations during training, which may be related to gradient variations introduced by the physics-information embedding mechanism, but the overall downward trend is clear. Overall, the PIGNN algorithm with embedded physics information significantly outperforms both traditional PINN and purely data-driven GNN in terms of convergence speed and final accuracy, demonstrating that the fusion of physical constraints can effectively improve training efficiency and fitting capability.

3.3. Prediction Results and Ablation Analysis

3.3.1. Single-Step Derivative Prediction and Multi-Step Rollout Prediction Results

The model's predictive capability is evaluated from two complementary perspectives: (1) **Single-step derivative prediction**—Given the true state $\mathbf{X}_t$ and control input $\mathbf{U}_t$, the RMSE between the model-predicted state derivatives $\hat{\dot{\mathbf{X}}}_t$ and the ground-truth derivatives $\dot{\mathbf{X}}_t$ is evaluated, reflecting the model's instantaneous dynamics fitting accuracy. (2) **Multi-step rollout state prediction**—Starting from an initial state, the model autoregressively predicts future state values $\hat{\mathbf{X}}_{t+k}$, which are compared against the true states $\mathbf{X}_{t+k}$, reflecting

the error accumulation characteristics in actual simulation. The two metrics are measured in derivative space and state space, respectively, with different physical meanings but complementary roles.



Figure 2. Training loss curves for the three architectures.

Table 6 presents the single-step derivative prediction Core(9) RMSE for the four architectures over all test samples.

Table 6. Single-step derivative prediction Core(9) RMSE.

Metric	Direct-MLP	PINN	PIGNN	GNN
Single-step derivative Core RMSE	1.6213	0.5541	0.2166	2.3518

PIGNN achieves a single-step derivative Core RMSE of 0.2166, representing an 86.6% improvement over Direct-MLP (1.6213), a 60.9% improvement over PINN (0.5541), and a 90.8% improvement over GNN (2.3518).

Table 7 and Figure 3 further present the per-dimension single-step derivative RMSE for the four architectures.

PINN and PIGNN achieve zero RMSE on the attitude angle derivatives ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$), position derivatives ($\dot{h}$, $\dot{x}_N$, $\dot{x}_E$), and engine derivative ($\dot{P}$), because the PhysicsLayer computes these dimensions exactly through kinematic equations and known models, independent of the network-predicted aerodynamic coefficients. Direct-MLP and GNN must learn these known relationships from data, resulting in nonzero attitude derivative errors. The RMSE of PIGNN on $\dot{V}_T$ (0.567) is only 37.6% of that of PINN (1.510), demonstrating that the graph-structural prior significantly improves prediction accuracy for aerodynamic-force-related states.

Table 7. Per-dimension single-step derivative RMSE.

State	Direct-MLP	PINN	PIGNN	GNN
V_T	4.7633	1.5098	0.5673	6.9804
α	0.0569	0.0175	0.0059	0.0633
β	0.0645	0.0062	0.0018	0.0762
ϕ	0.1653	0.0000	0.0000	0.1840
θ	0.1064	0.0000	0.0000	0.1454
ψ	0.1022	0.0000	0.0000	0.1297
p	0.9098	0.6702	0.3092	0.9611
q	0.1274	0.0621	0.0481	0.1265
r	0.2600	0.1733	0.0484	0.1738

Table 8 and Figure 4 present the multi-step rollout state prediction Core(9) RMSE for the four architectures over 200 test trajectories. The rollout prediction starts from the true

initial state and advances autoregressively for 499 steps ($\Delta t = 0.01$ s, approximately 5 s total) via Euler integration, evaluating the error between predicted and true states.

Table 8. Multi-step rollout state prediction Core(9) RMSE.

Rollout Step	Direct-MLP	PINN	PIGNN	GNN
1	0.0017	0.0010	0.0003	0.0069
10	0.0139	0.0091	0.0024	0.0605
50	0.0612	0.0357	0.0091	0.2820
100	0.1540	0.0685	0.0210	0.5497
200	0.6123	0.2362	0.1053	1.6360
499	5.2722	2.8814	1.1554	10.1141

PIGNN achieves the best performance at all rollout steps. At 100 steps (1 s), the state RMSE of PIGNN is 0.0210, representing an 86.4% improvement over Direct-MLP (0.1540), a 69.3% improvement over PINN (0.0685), and a 96.2% improvement over GNN (0.5497). At 499 steps (approximately 5 s), the state RMSE of PIGNN is 1.1554, still significantly better than Direct-MLP (5.2722), PINN (2.8814), and GNN (10.1141). Within the first 100 steps, error growth is approximately linear, indicating that the physical constraints of the PhysicsLayer effectively suppress error divergence in short-range prediction; beyond 200 steps, error growth accelerates, primarily limited by the accuracy of the Euler integration scheme. GNN, lacking the PhysicsLayer hard constraint, reaches a 499-step error of 10.1141, approximately 8.8 times that of PIGNN.

Comparing PINN and PIGNN, the graph-structural prior yields a 60.9% improvement in single-step derivative accuracy. This improvement stems from the GNN graph topology precisely restricting each aerodynamic coefficient to receive information only from its physically dependent state variables, avoiding the information redundancy caused by MLP full connectivity. Both architectures receive identical input information (all 13 state variables, 4 control inputs, Mach number, and dynamic pressure), but the MLP processes them through fully connected layers without distinguishing which inputs affect which output, while the graph structure encodes the physical dependency relationships as edge connections. Comparing PIGNN and GNN, the PhysicsLayer yields a 90.8% improvement in single-step derivative accuracy. The disadvantage of GNN is mainly reflected in: (1) the need to learn known exact kinematic relationships from data, and (2) the inability to guarantee the physical self-consistency of the output.

As shown in Figure 4, the GNN architecture exhibits significantly larger rollout errors than the two PhysicsLayer-based architectures, making the difference between PINN and PIGNN difficult to discern at the same scale. Figure 5 therefore presents a focused comparison between PINN and PIGNN only, showing the per-step median state prediction error for each of the 12 state dimensions (excluding engine power). This focused view reveals that PIGNN consistently outperforms PINN across all aerodynamic-force-related states (V_T , α , β) and angular rate states (p , q , r) throughout the rollout horizon, with the advantage being most pronounced in V_T and α where the force coefficient prediction accuracy plays a dominant role. The errors of all architectures accumulate approximately linearly within the first 100 steps, with PIGNN exhibiting the lowest accumulation rate.

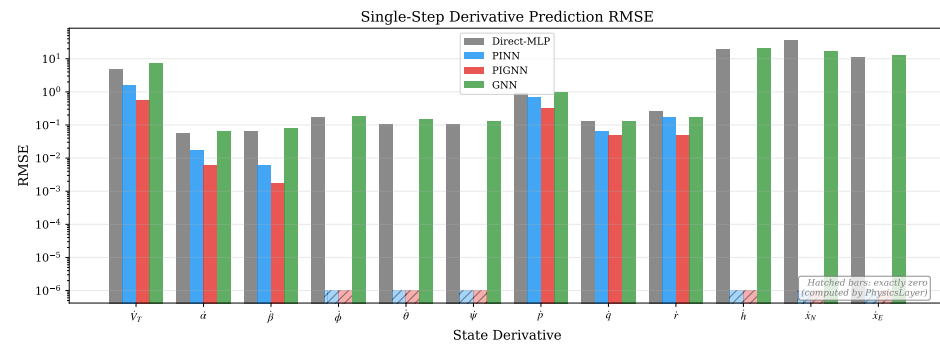


Figure 3. Single-step derivative RMSE comparison across four architectures. Colors and diagonal hatching are used to distinguish the compared architectures.

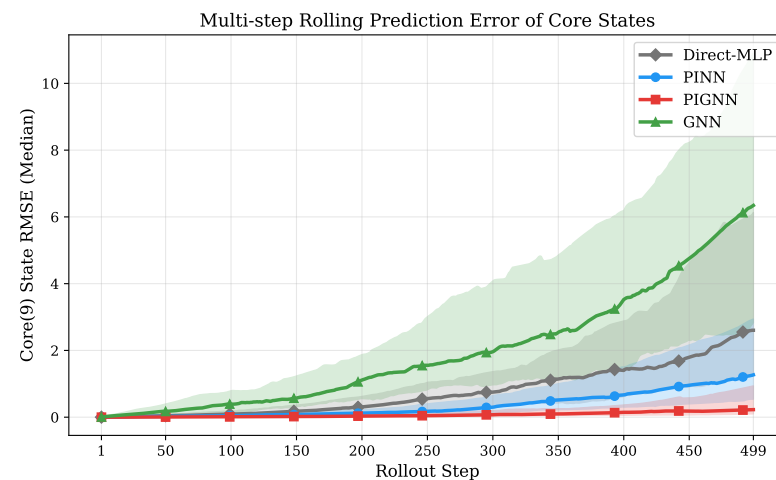


Figure 4. Median Core(9) state RMSE during multi-step rollout prediction for four architectures, with interquartile range shading.

3.3.2. Aerodynamic Coefficient Prediction Accuracy

Table 9 and Figure 6 present the comparison between the aerodynamic coefficients predicted by PINN and PIGNN and the ground-truth values (with no aerodynamic coefficient supervision during training). Direct-MLP and GNN are not included in this aerodynamic coefficient comparison because they directly output state derivatives and do not contain aerodynamic coefficients as physically interpretable latent variables. To further examine whether the learned coefficients preserve physically meaningful trends, Figure 7 plots sectional coefficient trends of PINN and PIGNN against the dominant aerodynamic variables. The longitudinal coefficients C_X , C_Z , and C_M are shown as functions of angle of attack α , while the lateral-directional coefficients C_Y , C_L , and C_N are shown as functions of sideslip angle β .

Table 9. Aerodynamic coefficient prediction RMSE (post hoc verification without aerodynamic coefficient supervision).

Coefficient	PINN	PIGNN
C_X	0.0096	0.0042
C_Y	0.0279	0.0080
C_Z	0.1029	0.0429
C_L	0.0028	0.0013
C_M	0.0060	0.0045
C_N	0.0043	0.0012

Multi-step Rolling Prediction Error for Each State Dimension

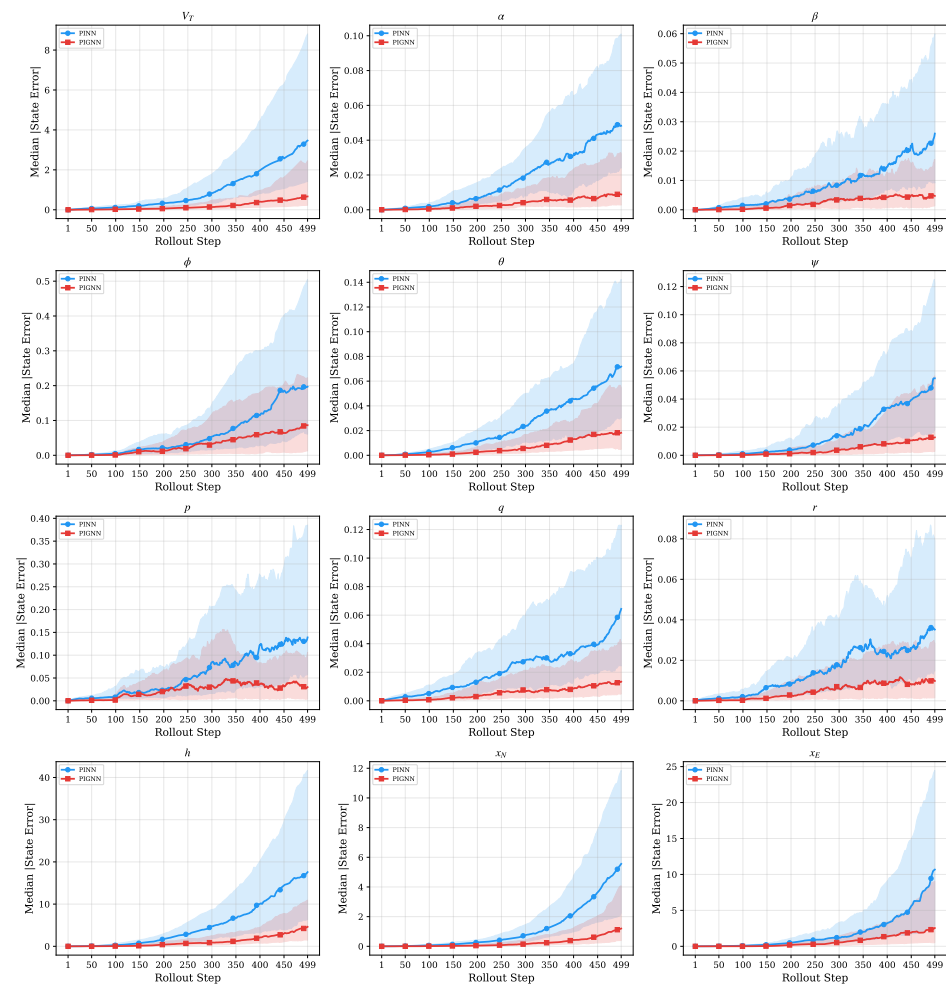


Figure 5. Per-step median state prediction error for 12 state dimensions: PINN vs. PIGNN comparison, with interquartile range shading.

Aerodynamic Coefficient Prediction

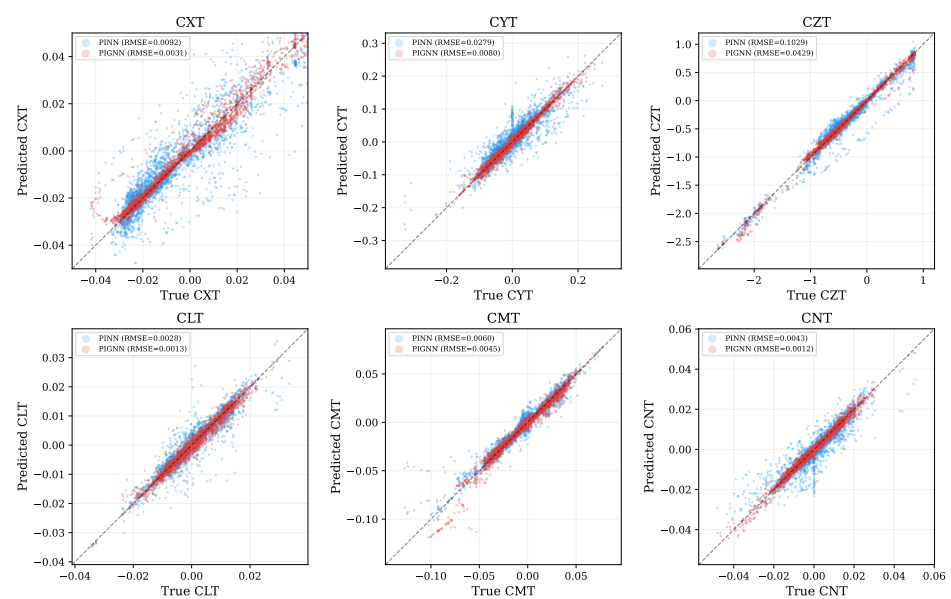


Figure 6. Scatter plot of predicted versus ground-truth aerodynamic coefficients.

PIGNN significantly outperforms PINN across all six aerodynamic coefficients: C_X (56.3% improvement), C_Y (71.3%), C_Z (58.3%), C_L (53.6%), C_M (25.0%), and C_N (72.1%). These results demonstrate that, without aerodynamic coefficient labels during training, the physics-prior graph structure effectively guides the network to discover a physically meaningful aerodynamic coefficient decomposition.

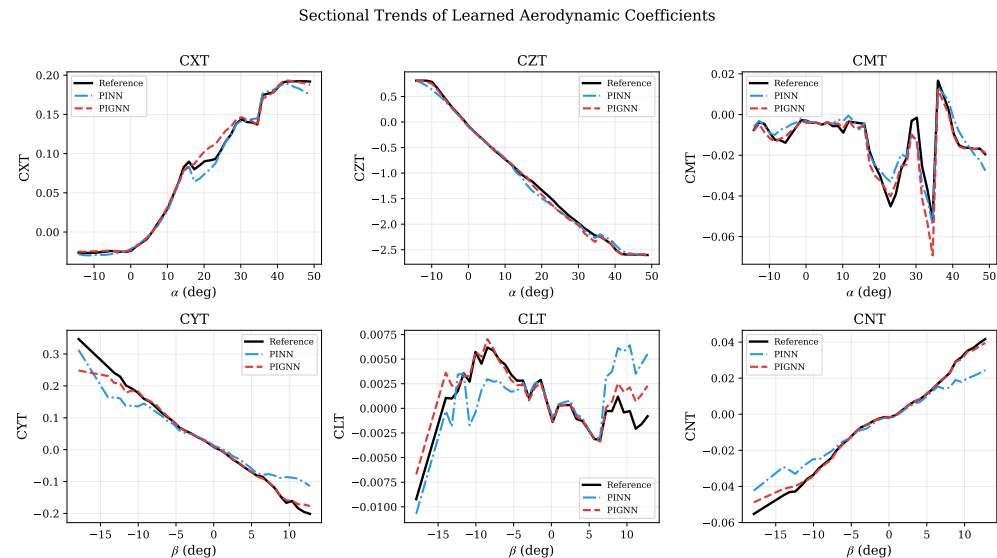


Figure 7. Sectional trends in aerodynamic coefficients learned by PINN and PIGNN compared with the F-16 reference aerodynamic database. Longitudinal coefficients are binned by angle of attack, and lateral-directional coefficients are binned by sideslip angle.

4. Discussion

An important property of the proposed method is its physical interpretability. The intermediate outputs of PIGNN—the six aerodynamic coefficients—have clear physical meanings and can be compared and validated against traditional aerodynamic databases. Experimental results demonstrate that, without aerodynamic coefficient supervision, the network can learn latent variable representations that closely match the reference aerodynamic coefficients, relying on state-transition supervision, the structural constraints of the physical equations, and the graph topology prior. This means that the method is not merely a “black-box” predictor but can also provide physical insights into the aerodynamic characteristics of the aircraft.

The core assumption of this method—“the equations of motion are structurally known, while the aerodynamic coefficients are unknown”—holds for any rigid-body aircraft, including eVTOL, blended-wing-body, and distributed-propulsion aircraft and other novel configurations. Extending this method to new aircraft types requires only redefining the dependency relationships in the graph topology according to their aerodynamic characteristics and modifying the computation logic of the PhysicsLayer according to the equations of motion. The advantage of using the F-16 as a verification platform lies in its publicly available complete aerodynamic database, which can serve as ground truth for validating the accuracy of network-predicted aerodynamic coefficients. For novel aircraft lacking complete aerodynamic data, this method can learn aerodynamic characteristics from limited flight test data, compensating for the insufficiency of wind tunnel data.

Furthermore, a high-fidelity flight dynamics model is the foundation of flight simulation verification. The model generated by this method possesses physical self-consistency (the PhysicsLayer guarantees that the output satisfies Newtonian mechanics), interpretability (aerodynamic coefficients as intermediate variables are available for expert review), and

potential computational efficiency after training, making it suitable for flight simulation verification and flight safety assessment scenarios.

The present study also has limitations. First, the experiments are based on F-16 simulation data and have not yet been validated on real flight data. Sensor noise and unmodeled effects in real flight data may affect the effectiveness of the method. Second, the ablation study is intended to isolate the roles of graph-based dependency modeling and the PhysicsLayer within the proposed architecture family; it should not be interpreted as a comprehensive superiority claim over all existing flight dynamics system identification methods. Third, the parameter counts of the compared architectures are not identical because graph message passing and attention decoding introduce additional trainable modules. Although the GNN baseline has more parameters than PIGNN and still performs worse, a fully parameter-matched comparison is left for future work. Finally, PhysicsLayer relies on accurate physical equations; for systems where the equations are not fully known, it may be necessary to explore compromise schemes between soft and hard constraints.

5. Conclusions

This paper proposes an architectural integration of physics-informed neural networks, graph-based dependency modeling, and known flight mechanics equations for flight dynamics modeling. The effectiveness of the method is verified through systematic ablation experiments. The main conclusions are as follows:

1. The physics-prior graph structure is the most critical factor for improving prediction accuracy. PIGNN achieves an 86.6% improvement over the conventional Direct-MLP surrogate and a 60.9% improvement over pure PINN in single-step derivative prediction. In 100-step rollout state prediction, PIGNN improves over Direct-MLP by 86.4% and over PINN by 69.3%. PINN and PIGNN receive identical input information, but the graph topology encodes the physical dependency relationships between aerodynamic coefficients and flight states into the network structure, enabling information to propagate along physically meaningful paths rather than through fully connected layers.
2. The PhysicsLayer hard constraint is indispensable for flight dynamics modeling. PIGNN achieves a 90.8% improvement over GNN (without PhysicsLayer) in single-step derivative prediction and a 96.2% improvement in 100-step rollout state prediction. The PhysicsLayer guarantees the physical self-consistency of the output and allows the network to learn only six unknown aerodynamic coefficients, substantially reducing the learning difficulty. In 200 long-trajectory (5 s) rollout tests, PIGNN maintains approximately linear error growth within the first 100 steps (1 s) with a Core RMSE of only 0.0210; at 499 steps (approximately 5 s), the Core RMSE is 1.1554, still significantly outperforming Direct-MLP, PINN, and GNN.
3. Without aerodynamic coefficient labels during training, PIGNN can learn aerodynamic coefficients that closely match the F-16 reference aerodynamic database, demonstrating the physical interpretability of the method.

The performance differences among the compared architectures are consistent with their structural differences: Direct-MLP has neither graph message passing nor hard physical constraints, PINN benefits from the PhysicsLayer but lacks graph-structured dependency modeling, GNN uses graph message passing but must learn known kinematic and dynamic relationships directly from data, and PIGNN combines both graph-based dependency modeling and hard physical constraints.

The results indicate that combining graph-based dependency modeling with known flight mechanics equations is a promising approach for building interpretable flight dynamics surrogate models. Future work will validate the effectiveness of the method on

real flight data, conduct broader parameter-matched comparisons with additional system identification baselines, and explore extensions to novel aircraft such as eVTOL.

Author Contributions: Conceptualization, L.M., S.D. and Z.L.; methodology, L.M. and Z.L.; software, L.M.; validation, L.M. and S.D.; formal analysis, Y.L.; investigation, L.M.; resources, S.D.; data curation, Y.L. and J.Z.; writing—original draft preparation, L.M.; writing—review and editing, S.D. and Z.L.; visualization, J.Z.; supervision, Z.L.; project administration, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The simulation data and code used in this study are available from the authors upon reasonable request.

Acknowledgments: The authors acknowledge the use of the open-source F-16 simulation model and aerodynamic database.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PIGNN	Physics-Informed Graph Neural Network
PINN	Physics-Informed Neural Network
GNN	Graph Neural Network
MLP	Multi-Layer Perceptron
RMSE	Root Mean Square Error
DOF	Degree of Freedom

References

1. Stevens, B.L.; Lewis, F.L.; Johnson, E.N. *Aircraft Control and Simulation: Dynamics, Control Design, and Autonomous Systems*, 3rd ed.; Wiley: Hoboken, NJ, USA, 2015.
2. Iliff, K.W. *Extraction of Aerodynamic Parameters for Aircraft at Extreme Flight Conditions*; NASA TM-86730; NASA: Edwards, CA, USA, 1985.
3. Jones, A.R.; Cetiner, O.; Smith, M.J. Physics and Modeling of Large Flow Disturbances: Discrete Gust Encounters for Modern Air Vehicles. *Annu. Rev. Fluid Mech.* **2022**, *54*, 469–493. [[CrossRef](#)]
4. Klein, V. Estimation of Aircraft Aerodynamic Parameters from Flight Data. *Prog. Aerosp. Sci.* **1989**, *26*, 1–77. [[CrossRef](#)]
5. Morelli, E.A. Global Nonlinear Aerodynamic Modeling Using Multivariate Orthogonal Functions. *J. Aircr.* **1995**, *32*, 270–277. [[CrossRef](#)] [[PubMed](#)]
6. Taylor, L.W.; Klein, V. Analytical Techniques for the Analysis of Stall/Spin Flight Test Data. *SAE Trans.* **1981**, *90*, 2095–2105.
7. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, *4*, 251–257. [[CrossRef](#)]
8. Linse, D.J.; Stengel, R.F. Identification of Aerodynamic Coefficients Using Computational Neural Networks. *J. Guid. Control Dyn.* **1993**, *16*, 1018–1025. [[CrossRef](#)] [[PubMed](#)]
9. Raisinghani, S.; Ghosh, A.; Kalra, P. Two New Techniques for Aircraft Parameter Estimation Using Neural Networks. *Aeronaut. J.* **1998**, *102*, 25–30. [[CrossRef](#)]
10. Singh, S.; Ghosh, A.K. Modified Delta Method for Estimation of Parameters from Flight Data of Stable and Unstable Aircraft. In *Proceedings of the 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, India, 22–23 February 2013*; IEEE: Piscataway, NJ, USA, 2013; pp. 775–781.
11. Das, S.; Kuttieri, R.A.; Sinha, M.; Bhatt, R. Neural Partial Differential Method for Extracting Aerodynamic Derivatives from Flight Data. *J. Guid. Control Dyn.* **2010**, *33*, 376–384. [[CrossRef](#)]
12. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [[CrossRef](#)]

13. Sun, Y.; Sengupta, U.; Juniper, M. Physics-Informed Deep Learning for Simultaneous Surrogate Modeling and PDE-Constrained Optimization of an Airfoil Geometry. *Comput. Methods Appl. Mech. Eng.* **2023**, *411*, 116042. [[CrossRef](#)]
14. Michek, N.E.; Mehta, P.M.; Huebsch, W.W. Flight Dynamics Modeling Using Physics-Informed Neural Networks. *AIAA J.* **2026**, *64*, 241–257. [[CrossRef](#)]
15. Chen, J.; Liu, J.; Shan, J.; Wang, J.; Meng, X. Identification of Aerodynamic Parameters Using Improved Physics-Informed Neural Network Framework. In *Proceedings of the 2024 32nd Mediterranean Conference on Control and Automation (MED)*, Chania, Greece, 25–28 June 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 424–429.
16. Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv* **2018**, arXiv:1806.01261. [[CrossRef](#)]
17. Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P. Learning to Simulate Complex Physics with Graph Networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Virtual, 13–18 July 2020; pp. 8459–8468.
18. Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; Battaglia, P. Learning Mesh-Based Simulation with Graph Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual, 3–7 May 2021.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.