ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

Research on Fault Simulation and Model Prediction Fault Tolerant Control of Electromechanical Systems Based on Digital Twins

Haijun Yang[1,*]

[1] School of Intelligent Manufacturing, Luoyang Institute of Science and Technology, Luoyang, Henan 471023, China

SUMMARY: *This study is committed to fault simulation, prediction and tolerance control in mechatronic systems under digital twins. The goal is to achieve reliable control when the system behavior speeds up and faults are difficult to observe directly. To solve this problem, we propose a unified method that integrates state tables, fault-related event extraction and uncertainty control adjustment in the same framework. Thus fault analysis and control optimization form a more direct connection, and no separate diagnosis stage is carried out. Four experimental results in terms of data show that the proposed method is always better than the baseline model.There is a maximum relative gain on the electromechanical fault simulation dataset, the F1 score rises from 86.11% to 87.89%, which is 1.78 percentage points higher than the strongest baseline. There is the best absolute result on the electromechanical system digital twin dataset, and the model reaches 91.The accuracy rate is 34%. The results show that the proposed framework has a role in handling the joint transactions of fault evolution, state estimation and control.*

KEYWORDS: *Fault simulation; Model prediction; Fault-tolerant control; Electromechanical systems; Digital twins*

1 Introduction

As electromechanical systems become more integrated and operate under more variable conditions, fault-tolerant control has become harder to design with fixed rule-based methods alone [1]. In many industrial settings, faults do not appear as isolated events; they develop through changes in state, control response, and external disturbance over time [2]. For this reason, fault simulation and model prediction are important, since they make it possible to identify potential failures earlier and adjust control actions before system performance degrades further [3]. Digital twins provide a useful setting for this problem because they maintain a virtual representation of the physical system and can be updated continuously from operational data [4]. This makes them suitable not only for monitoring and diagnosis, but also for predictive analysis and control support [5]. When fault simulation and prediction are integrated into a digital twin framework, the result is a more responsive control process that can track system evolution and react to emerging faults with better timing. This is why digital twin based fault-tolerant control has become an important direction in electromechanical system research [6].

Early work on fault-tolerant control relied mainly on predefined rules and relatively fixed simulation frameworks [7]. These methods were effective under known operating conditions because they made fault analysis easier to structure and interpret [8]. Their main limitation was

*yhj3061@163.com
https://doi.org/10.65102/is2026190
Published: 30 April, 2026

the strong dependence on prior assumptions. Once system behavior became more variable, or fault evolution no longer matched the expected pattern, their adaptability declined quickly [9]. This limitation became more apparent in electromechanical systems subject to changing loads, disturbances, and coupled dynamics [10]. Later research turned to data-driven methods that learn fault patterns from historical observations. Statistical models improved fault detection by capturing anomalies and recurring failure signatures in operational data [11]. These methods were generally more flexible than rule-based approaches. They also showed better scalability across different system settings [12]. Their performance, however, still depended heavily on dataset quality and coverage. When operating conditions changed, or the training data failed to represent the target environment well, generalization became a clear weakness [13]. More recent studies have introduced neural network models for fault simulation and prediction. These models learn complex features directly from raw data. This improves detection accuracy and reduces the need for manual feature design [13]. Pre-trained models have also been used to improve transferability across related tasks and operating conditions [14]. These gains come with practical costs. Neural models usually require more labeled data. They also require more computation and more careful deployment than simpler methods [15]. Current research is therefore moving toward frameworks that preserve the predictive strength of data-driven modeling while improving stability and usability in real engineering environments [16].

In response to these limitations, this study proposes a digital twin based framework for fault simulation and fault prediction and fault-tolerant control in electromechanical systems [17]. The framework places state modeling and fault-relevant event analysis and control adjustment in the same process [18]. Fault handling is therefore not restricted to post hoc diagnosis [19]. The framework uses updated system information to support earlier prediction and more stable control decisions under changing operating conditions [20]. This point matters in electromechanical systems because fault development and state evolution and control response are often coupled [21, 22]. The main contributions of this study are as follows:

- A digital twin based framework is proposed to connect fault simulation and fault prediction and fault-tolerant control in one process.
- The method is designed to remain effective under changing system conditions. It does this through structured state representation and event-aware analysis and uncertainty-aware control refinement.
- Experiments on four datasets show consistent improvement over the baseline models. The largest relative gain appears on the Electromechanical Fault Simulation Dataset. The F1 Score increases from 86.11% to 87.89%.

2 Method

2.1 Overview

This section presents the method for fault simulation, fault prediction, and fault-tolerant control in electromechanical systems under a digital twin framework. The goal is to use synchronized system data and predictive modeling to identify potential faults in advance and support control decisions before the physical system enters a critical state. The method is organized into three parts. Section 2.2 defines the problem setting and introduces the main variables, including system states, control inputs, disturbances, fault variables, and the digital twin representation. These definitions provide the basis for modeling fault evolution and control adaptation in a unified form. Section 2.3 presents the Counterfactual Manifold Planner, which serves as the core model of the framework. It contains three coordinated components: Constraint-Driven

Manifold Encoding, Agent-Based Event Segmentation, and Probabilistic Fault Prediction. These components are used to represent system states under operational constraints, identify fault-relevant events from system evolution, and estimate the likelihood of future faults. Section 2.4 then introduces the refinement strategy built on constrained optimization and uncertainty propagation. This part is used to improve the reliability of fault prediction and to adjust control actions under uncertain operating conditions.

2.2 Preliminaries

This subsection defines the fault simulation problem and the model-based fault-tolerant control problem for electromechanical systems under a digital twin framework. It specifies the state variables and the fault variables and the control inputs used in the model. These definitions serve as the mathematical basis of the Counterfactual Manifold Planner.

Let the system state at time t be denoted by $\mathbf{x}(t) \in \mathbb{R}^n$, where n is the dimension of the state space. The system dynamics are written as

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)\big), \tag{1}$$

where $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input, $\mathbf{w}(t) \in \mathbb{R}^p$ denotes external disturbance or noise, and $\mathbf{f}$ describes the nominal system dynamics.

The digital twin is denoted by $\mathbf{X}(t)$, which is updated using sensor observations and model prediction so that it remains consistent with the physical system. Its update process is written as

$$\mathbf{X}(t + \Delta t) = \mathbf{g}\big(\mathbf{X}(t), \mathbf{U}(t), \mathbf{W}(t), \mathbf{d}(t)\big), \tag{2}$$

where $\mathbf{U}(t)$ and $\mathbf{W}(t)$ are the digital counterparts of the control input and disturbance, and $\mathbf{d}(t)$ measures the discrepancy between the physical system and its twin.

To represent fault influence, we introduce a fault vector $\mathbf{v}(t) \in \mathbb{R}^q$. The system dynamics under fault conditions are then written as

$$\dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)\big) + \mathbf{h}\big(\mathbf{x}(t), \mathbf{v}(t)\big), \tag{3}$$

where $\mathbf{h}$ models the effect of the fault on system evolution. This formulation allows fault simulation and control adaptation to be described in the same state-space framework.

To evaluate possible fault scenarios before they occur, the Counterfactual Manifold Planner generates counterfactual states $\mathbf{x}^*(t)$, which represent hypothetical system trajectories under alternative fault conditions:

$$\mathbf{x}^*(t) = \mathbf{f}^*\big(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}^*(t)\big), \tag{4}$$

where $\mathbf{f}^*$ is the counterfactual transition function and $\mathbf{v}^*(t)$ denotes a hypothetical fault input.

Based on the observed state and fault information, Constraint-Driven Manifold Encoding maps the system into a structured manifold representation:

$$\mathcal{M} = \mathcal{E}\big(\mathbf{x}(t), \mathbf{v}(t)\big), \tag{5}$$

where $\mathcal{E}$ is the encoding function. This representation provides a compact but constrained space for later prediction and optimization.

Agent-Based Event Segmentation is used to identify key transitions in system evolution and isolate fault-relevant events. Its segmentation process is written as

$$\mathcal{S}\big(\mathbf{x}(t), \mathbf{u}(t)\big) = \{\text{events}\}. \tag{6}$$

This step separates routine variation from event patterns that are more likely to be associated with fault development.

2.3 Counterfactual Manifold Planner

This subsection introduces the Counterfactual Manifold Planner as the core model of the framework. As shown in Fig. 1, the planner links fault simulation with event identification and fault prediction under the digital twin setting. The model contains three components. They are Constraint-Driven Manifold Encoding and Agent-Based Event Segmentation and Probabilistic Fault Prediction. These components define the state representation. They extract fault-related events from system evolution. They also estimate the probability of future faults.

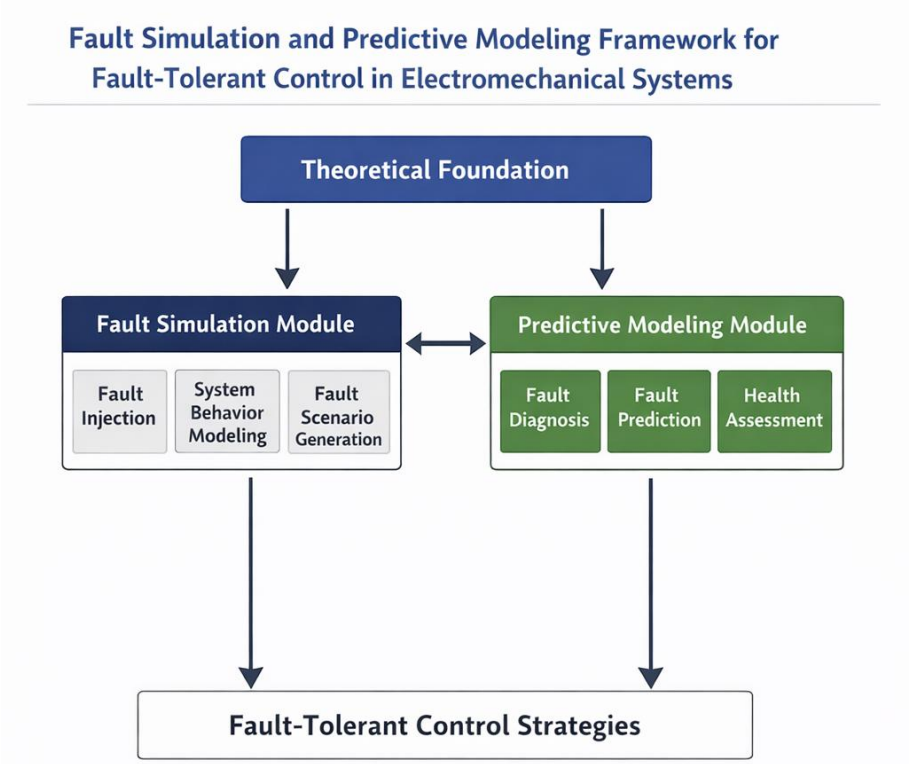

Figure 1: Overall framework for fault simulation and predictive modeling in electromechanical systems. The framework combines a theoretical foundation with fault simulation and predictive modeling, and uses their interaction to support fault-tolerant control.

Constraint-Driven Manifold Encoding. The first step is to project the high-dimensional system state into a lower-dimensional manifold without discarding the operating limits of the physical system. As shown in Figure 2, the encoded state remains connected to system dynamics, control input, fault diagnosis, and fault-tolerant control. In this setting, the manifold is not used only for compression. It also provides a feasible state space in which later prediction and control refinement can be carried out without drifting away from physical or operational constraints.

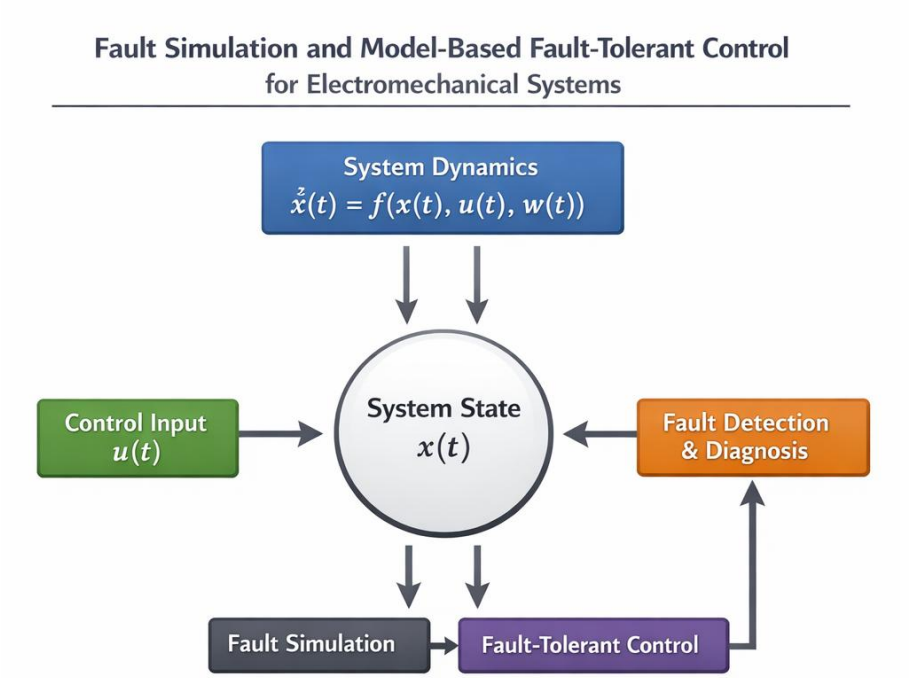

Figure 2: Constraint-Driven Manifold Encoding for electromechanical systems. The figure shows how system dynamics, control input, fault diagnosis, fault simulation, and fault-tolerant control are connected through the encoded system state.

Let $\mathbf{x}(t) \in \mathbb{R}^n$ denote the original system state and let $\mathbf{z}(t) \in \mathbb{R}^d$ denote its encoded representation, where $d \ll n$. The encoding process is defined as

$$\mathbf{z}(t) = \mathcal{E}\big(\mathbf{x}(t), \mathbf{v}(t)\big), \tag{7}$$

where $\mathcal{E}$ is the encoding function and $\mathbf{v}(t)$ denotes the fault variable. In this form, the latent state depends not only on the nominal system condition but also on the fault condition observed at time t.

The constrained manifold is defined as

$$\mathcal{M} = \{\mathbf{z} \in \mathbb{R}^d \mid \mathbf{C}(\mathbf{z}) \leq \mathbf{b}\}, \tag{8}$$

where $\mathbf{C}(\mathbf{z})$ denotes the constraint function and $\mathbf{b}$ is the vector of admissible bounds. This means that only encoded states satisfying physical and operational conditions are retained in the feasible manifold.

To make the constraint structure explicit, we write

$$\mathbf{C}(\mathbf{z}) = \begin{bmatrix} c_1(\mathbf{z}) \\ c_2(\mathbf{z}) \\ \vdots \\ c_k(\mathbf{z}) \end{bmatrix}, \tag{9}$$

where each $c_i(\mathbf{z})$ corresponds to one constraint, such as thermal limits, stress bounds, actuator range, or energy constraints. This decomposition makes it easier to analyze which operational conditions dominate the encoded representation.

The encoded state is further required to preserve the main system dynamics. A reconstruction relation is introduced as

$$\hat{\mathbf{x}}(t) = \mathcal{D}\big(\mathbf{z}(t)\big), \tag{10}$$

where $\mathcal{D}$ is the decoding function and $\hat{\mathbf{x}}(t)$ is the reconstructed system state. This step ensures that the manifold does not lose the main information needed for later fault analysis.

Based on the encoder-decoder pair, the representation loss is written as

$$\mathcal{L}_{\text{enc}} = \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|_2^2, \tag{11}$$

which penalizes distortion between the original state and its reconstruction. A smaller value of $\mathcal{L}_{\text{enc}}$ indicates that the encoded manifold preserves more of the original system behavior.

To enforce feasibility during encoding, the constraint penalty is defined as

$$\mathcal{L}_{\text{con}} = \sum_{i=1}^{k} \max\,(0,\, c_i(\mathbf{z}) - b_i)^2, \tag{12}$$

where b_i is the bound associated with the i-th constraint.

Agent-Based Event Segmentation. This component divides the system trajectory into segments that are more relevant to fault development. Let the event set be denoted by

$$\mathcal{E} = \{e_i \mid e_i = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{y}_t)\}, \tag{13}$$

where e_i is an event, $\mathbf{x}_t$ is the system state at time t, $\mathbf{u}_t$ is the control input, and $\mathbf{y}_t$ is the observed output. This formulation means that event generation depends jointly on internal dynamics, control response, and observed behavior.

To describe agent-level interaction more explicitly, we assign each agent a_j a local event score:

$$s_j(t) = \phi_j(\mathbf{x}_t, \mathbf{u}_t, \mathbf{y}_t), \tag{14}$$

where $\phi_j(\cdot)$ is the scoring function associated with agent a_j. A higher score indicates that the current state is more likely to correspond to a meaningful event boundary or fault-related transition.

The event decision is then written as

$$e_i(t) = \begin{cases} 1, & s_i(t) > \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where τ is the event threshold. In this way, the segmentation process converts continuous system evolution into discrete event indicators.

The event detection function is further decomposed as

$$f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{y}_t) = g(\mathbf{x}_t) + h(\mathbf{u}_t) + k(\mathbf{y}_t), \tag{16}$$

where $g(\mathbf{x}_t)$, $h(\mathbf{u}_t)$, and $k(\mathbf{y}_t)$ represent the contributions of system state, control input, and observed output, respectively. This decomposition makes it possible to examine which signal source is mainly responsible for event formation.

To preserve temporal continuity, adjacent event indicators are grouped into segments:

$$\mathcal{B}_r = \{t \mid e_i(t) = 1,\ t \in \left[t_r^{\text{start}}, t_r^{\text{end}}\right]\}, \tag{17}$$

where $\mathcal{B}_r$ denotes the r-th event segment. Each segment corresponds to a contiguous time interval with coherent fault-relevant behavior.

The resulting segmented representation can be summarized as

$$\tilde{\mathcal{E}} = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_R\}, \tag{18}$$

where R is the number of retained event segments. This representation reduces irrelevant variation and allows the next stage to focus on transitions that carry the most diagnostic value.

Probabilistic Fault Prediction. This component estimates fault probability from the encoded manifold and the segmented event structure. The prediction is conditioned on the latent state and the retained event evidence. This setting is suitable when the observed signal is noisy. It also applies when fault evolution is only partially observable.

The posterior fault probability over the manifold is defined as

$$P\big(\text{Fault} \mid \tilde{\mathcal{E}}, \mathcal{M}\big) = \int P\big(\text{Fault} \mid \mathbf{z}, \tilde{\mathcal{E}}\big) P(\mathbf{z} \mid \mathcal{M})\, d\mathbf{z}, \tag{19}$$

where $\mathbf{z}$ denotes the encoded latent state. The term $\tilde{\mathcal{E}}$ denotes the retained event segments.

For a single event segment $\mathcal{B}_r$, the segment-level fault probability is written as

$$P(\text{Fault} \mid \mathcal{B}_r, \mathbf{z}) = \psi(\mathbf{z}, \mathcal{B}_r), \tag{20}$$

where $\psi(\cdot)$ is the segment-level prediction function.

The contribution of one event evidence term e_i is expressed by

$$P(\text{Fault} \mid \mathbf{z}, e_i) = \frac{P(e_i \mid \text{Fault}, \mathbf{z})\, P(\text{Fault} \mid \mathbf{z})}{P(e_i \mid \mathbf{z})}. \tag{21}$$

The latent-state-dependent prior is defined as

$$P(\text{Fault} \mid \mathbf{z}) = \pi(\mathbf{z}), \tag{22}$$

where $\pi(\mathbf{z})$ denotes the prior fault probability under latent state $\mathbf{z}$.

The retained event segments are combined as

$$P\big(\text{Fault} \mid \tilde{\mathcal{E}}, \mathbf{z}\big) = 1 - \prod_{r=1}^{R} \big(1 - P(\text{Fault} \mid \mathcal{B}_r, \mathbf{z})\big). \tag{23}$$

The manifold distribution satisfies

$$\int P(\mathbf{z} \mid \mathcal{M})\, d\mathbf{z} = 1. \tag{24}$$

The expected fault risk is then written as

$$\mathcal{R}_{\text{fault}} = \int P\big(\text{Fault} \mid \tilde{\mathcal{E}}, \mathbf{z}\big) P(\mathbf{z} \mid \mathcal{M})\, d\mathbf{z}. \tag{25}$$

2.4 Constrained Optimization Refinement and Uncertainty Propagation

This subsection describes how fault prediction is converted into feasible control adjustment. The strategy is built on the Counterfactual Manifold Planner. It is used when the system operates under uncertainty. Figure 3 shows the structure of this stage.

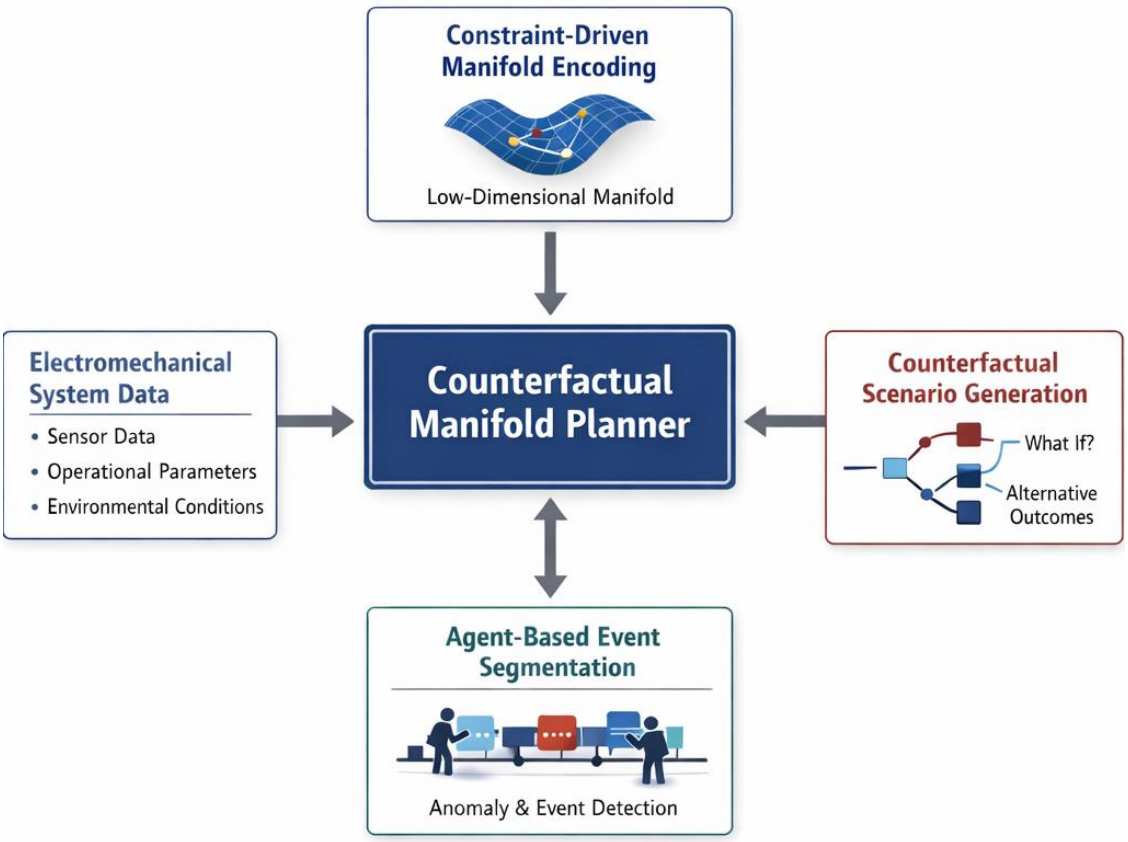

Figure 3: Control refinement strategy based on the Counterfactual Manifold Planner. The figure shows how manifold-based representation is linked to fault analysis and control adjustment for fault-tolerant operation in electromechanical systems.

Constrained Optimization Refinement. This part refines control actions in the manifold space under physical and operational constraints. As shown in Fig. 4, the refinement stage connects predicted fault scenarios with safe control adjustment. The optimization target is defined together with the feasibility boundary.

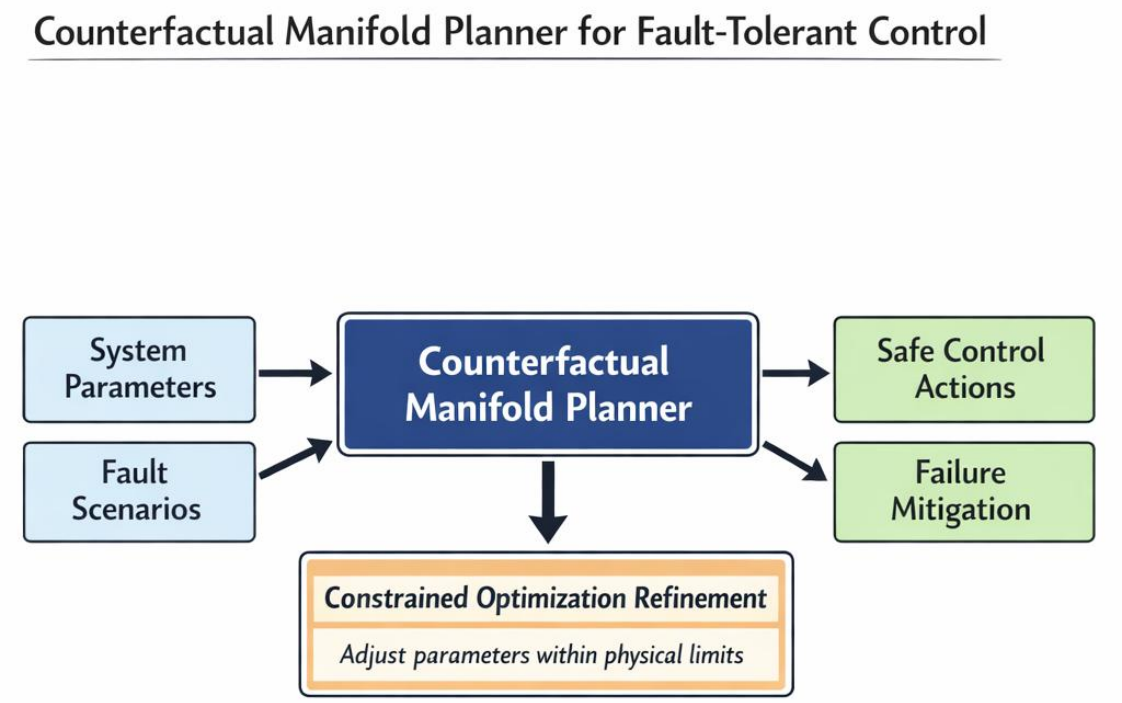

Figure 4: Constrained Optimization Refinement for fault-tolerant control. The refinement stage uses system states, predicted fault information, and operational constraints to generate feasible control actions and mitigation strategies.

Let $\mathbf{x}$ denote the system state. Let $\mathbf{u}$ denote the control action. The constrained optimization problem is written as

$$\min_{\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) \quad \text{subject to} \quad C(\mathbf{x},\mathbf{u}) \leq 0, \tag{26}$$

where $f(\mathbf{x},\mathbf{u})$ is the control objective. The term $C(\mathbf{x},\mathbf{u})$ defines the feasibility constraints.

The constraint structure is written as

$$C(\mathbf{x},\mathbf{u}) = \begin{bmatrix} c_1(\mathbf{x},\mathbf{u}) \\ c_2(\mathbf{x},\mathbf{u}) \\ \vdots \\ c_K(\mathbf{x},\mathbf{u}) \end{bmatrix}, \tag{27}$$

where each $c_k(\mathbf{x},\mathbf{u})$ corresponds to one requirement of the operating region.

The optimization objective is decomposed as

$$f(\mathbf{x},\mathbf{u}) = f_{\text{perf}}(\mathbf{x},\mathbf{u}) + \lambda_1 f_{\text{ctrl}}(\mathbf{u}), \tag{28}$$

where f_{perf} measures performance loss. The term f_{ctrl} penalizes excessive control effort.

The smoothness term is defined as

$$f_{\text{ctrl}}(\mathbf{u}) = \| \mathbf{u}_t - \mathbf{u}_{t-1} \|_2^2, \tag{29}$$

which penalizes abrupt step-to-step variation in the control input.

The constrained problem is rewritten in penalty form as

$$\mathcal{J}(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u}) + \lambda_2 \sum_{k=1}^{K} \max(0,\, c_k(\mathbf{x}, \mathbf{u}))^2, \tag{30}$$

where λ_2 controls the penalty on constraint violation.

The refined control action is obtained as

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \mathcal{J}(\mathbf{x}, \mathbf{u}), \tag{31}$$

which selects the feasible control input with the lowest penalized objective value.

The state update after control adjustment is written as

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t^*, \mathbf{w}_t) + \mathbf{h}(\mathbf{x}_t, \mathbf{v}_t), \tag{32}$$

where $\mathbf{u}_t^*$ is the refined control input at time t.

Uncertainty Propagation Modeling. This stage models the uncertainty that remains in system observation and state estimation and fault prediction. The planner incorporates this uncertainty into the prediction process. The resulting fault risk is evaluated under a distribution rather than a fixed estimate.

Let $\mathbf{x}$ denote the current system state. Let $\boldsymbol{\theta}$ denote the uncertain parameters in the prediction model. The fault probability under uncertainty is written as

$$P(\text{Fault} \mid \mathbf{x}) = \int P(\text{Fault} \mid \mathbf{x}, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \mid \mathbf{x})\, d\boldsymbol{\theta}, \tag{33}$$

where $P(\boldsymbol{\theta} \mid \mathbf{x})$ denotes the conditional distribution of the uncertain parameters under state $\mathbf{x}$.

When new observations are available, the parameter distribution is updated by Bayesian inference:

$$P(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y}) = \frac{P(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \mid \mathbf{x})}{P(\mathbf{y} \mid \mathbf{x})}, \tag{34}$$

where $\mathbf{y}$ denotes the observed output.

The expected fault probability is then defined as

$$\bar{P}_{\text{fault}}(\mathbf{x}) = \int P(\text{Fault} \mid \mathbf{x}, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})\, d\boldsymbol{\theta}, \tag{35}$$

which gives the posterior mean fault risk under the latest observation.

The uncertainty level of the prediction is measured by

$$\text{Var}_{\text{fault}}(\mathbf{x}) = \int \left(P(\text{Fault} \mid \mathbf{x}, \boldsymbol{\theta}) - \bar{P}_{\text{fault}}(\mathbf{x})\right)^2 P(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})\, d\boldsymbol{\theta}, \tag{36}$$

which measures the variation of the fault estimate under posterior parameter uncertainty.

To propagate uncertainty into the future state, we define

$$P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = \int P(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \mid \mathbf{x}_t, \mathbf{y}_t)\, d\boldsymbol{\theta}, \tag{37}$$

which links parameter uncertainty to the next-state distribution under control action $\mathbf{u}_t$.

The uncertainty-aware fault score is then written as

$$\mathcal{R}_u(\mathbf{x}) = \bar{P}_{\text{fault}}(\mathbf{x}) + \eta\, \text{Var}_{\text{fault}}(\mathbf{x}), \tag{38}$$

where η controls the contribution of prediction variance to the final risk score.

Integrated Fault Tolerance. This stage combines control refinement with uncertainty propagation in one decision process. The control action is selected under both feasibility and predicted fault risk.

Let **x** denote the system state. Let **u** denote the control input. The integrated objective is written as

$$\min_{\mathbf{u}} \quad f(\mathbf{x},\mathbf{u}) + \lambda \int P(\text{Fault} \mid \mathbf{x},\boldsymbol{\theta},\mathbf{u})\, P(\boldsymbol{\theta} \mid \mathbf{x})\, d\boldsymbol{\theta} \quad \text{subject to} \quad C(\mathbf{x},\mathbf{u}) \le 0, \tag{39}$$

where λ controls the balance between control performance and fault risk.

Using the uncertainty-aware risk score defined above, the objective is rewritten as

$$\mathcal{J}_{\text{ift}}(\mathbf{x},\mathbf{u}) = f(\mathbf{x},\mathbf{u}) + \lambda \mathcal{R}_u(\mathbf{x},\mathbf{u}), \tag{40}$$

where $\mathcal{R}_u(\mathbf{x},\mathbf{u})$ denotes the predicted fault risk after action **u**.

The admissible control set is defined as

$$\mathcal{U}_{\text{safe}} = \{\mathbf{u} \mid C(\mathbf{x},\mathbf{u}) \le 0\}, \tag{41}$$

which restricts the search space to feasible actions.

The resulting control policy is written as

$$\mathbf{u}^* = \arg\min_{\mathbf{u}\in\mathcal{U}_{\text{safe}}} \mathcal{J}_{\text{ift}}(\mathbf{x},\mathbf{u}), \tag{42}$$

After the refined action is applied, the closed-loop system evolves as

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t,\mathbf{u}_t^*,\mathbf{w}_t) + \mathbf{h}(\mathbf{x}_t,\mathbf{v}_t), \tag{43}$$

The cumulative control objective is defined as

$$\mathcal{J}_{\text{total}} = \sum_{t=0}^{T} \gamma^t\, \mathcal{J}_{\text{ift}}(\mathbf{x}_t,\mathbf{u}_t), \tag{44}$$

where $\gamma \in (0,1)$ is the discount factor.

3 Experimental Setup

3.1 Dataset and Data Preprocessing

3.1.1 Datasets

We evaluated the proposed framework on four datasets under the digital twin setting. These datasets cover fault simulation and state prediction and fault-tolerant control. The Electromechanical Fault Simulation Dataset was generated from high-fidelity electromechanical system simulations. It contains 15,360 samples. It includes 6 fault categories. It uses multivariate time-series signals from 64 sensor channels. Fault labels were generated automatically by the simulation engine under predefined fault injection rules. Sequences with fewer than 200 time steps were removed. Saturated signals were removed. Repeated simulation seeds were also removed. The data were split by time with a ratio of 7:1:2 to preserve temporal order. The Digital Twin Model Prediction Dataset combines simulation records with real operating data. It contains 18,240 samples. It includes 52 system variables. It also includes one regression target. The labels correspond to future system states obtained from synchronized

logging records. Samples with missing timestamps were excluded. Misaligned sensor readings were excluded. Inconsistent state transitions were also excluded. We used a chronological 7:1:2 split to reduce the risk of information leakage. The Fault-Tolerant Control System Dataset was collected from controlled testbeds. It contains 12,800 samples. It includes 48 control-related variables. It also includes 5 system states. Labels were assigned by control engineers based on system stability and recovery performance under fault conditions. Records with incomplete control logs were removed. Records with unstable initialization were removed. Records with recovery times above a fixed threshold were also removed. This dataset was divided with a stratified 8:1:1 split. The Electromechanical System Digital Twin Dataset is a large-scale monitoring dataset. It contains 20,480 samples. It includes 60 variables that describe system operation and environmental conditions. It also includes one regression target. Labels were generated from synchronized digital twin outputs through deterministic mapping rules. Samples with missing sensor channels were discarded. Samples with abnormal spikes were discarded. Samples with duplicated timestamps were also discarded. A time-aware 7:1:2 split with strict sequence isolation was adopted.

3.1.2 Data Preprocessing

The same preprocessing pipeline was applied to all four datasets, including the Electromechanical Fault Simulation Dataset, the Digital Twin Model Prediction Dataset, the Fault-Tolerant Control System Dataset, and the Electromechanical System Digital Twin Dataset. We first removed duplicate sequences according to timestamps and sensor values. Samples with missing values in key channels were then discarded, and abnormal records were filtered using fixed rules. More specifically, we removed sequences with over 10% missing values, sensor readings beyond physical limits, or sudden jumps larger than five times the standard deviation. All numerical variables were standardized to zero mean and unit variance using statistics computed only from the training set. To keep the input shape consistent across datasets, all sequences were resampled to 256 time steps by linear interpolation. Sensor channels were synchronized with unified timestamps, and identity alignment was preserved across system instances. For sequence modeling, training samples were generated with a sliding window of length 256 and stride 64. To improve robustness, three augmentation operations were applied only to the training data: Gaussian noise injection with variance 0.01, random temporal masking over 10% of time steps, and channel dropout on 5% of sensor channels. No external feature extractor was introduced. All inputs came directly from raw sensor sequences and aligned system variables. The preprocessing parameters estimated on the training set were reused for validation and test data without recalculation, so the evaluation procedure remained reproducible and consistent across methods.

3.2 Implementation Details

All experiments were implemented in PyTorch and run on a unified server platform equipped with NVIDIA Tesla V100 GPUs. The training process used stochastic gradient descent with momentum, cosine annealing learning rate decay, mixed-precision training, and a fixed random seed to keep optimization stable and reproducible. Each model was trained for 100 epochs with a batch size of 64. Both early stopping and checkpoint selection were determined by validation performance. Since all four datasets are organized as multivariate time-series signals rather than images, the model was implemented as a sequence encoder for sensor-based fault analysis and prediction. Each sample was represented as a tensor of size $256 \times C$, where C is the number of channels in the corresponding dataset. A temporal feature extractor was used to model both local patterns and longer-range dependencies in the sensor sequence, and its output was passed to a task-specific prediction head for either classification or regression. Dropout was applied

before the final output layer to reduce overfitting. For a fair comparison, the proposed method and all baseline methods used the same data splits, preprocessing pipeline, optimizer settings, and stopping criteria. Hyperparameters were tuned only on the validation set, and final results were reported on the same held-out test set under identical evaluation protocols. Detailed implementation settings are listed in Table 1.

Table 1: Summary of implementation and model configuration details.

Category	Configuration
Hardware	Intel Xeon Gold 6226R CPU, 4×NVIDIA Tesla V100 GPUs (32 GB), 256 GB RAM, Ubuntu 20.04 LTS
Software	PyTorch 2.1.0, CUDA 11.8, cuDNN 8.9.2, numpy 1.24.4, scipy 1.10.1, scikit-learn 1.3.2, pandas 2.0.3
Epochs	100
Batch size	64
Optimizer	SGD with momentum 0.9
Learning rate	0.01
Weight decay	5×10^{-4}
Scheduler	Cosine annealing
Random seed	42
Mixed precision	Enabled
Early stopping	Patience = 10
Checkpoint selection	Best validation F1 score
Input format	Multivariate time series
Sequence length	256
Augmentation	Gaussian noise, temporal masking, channel dropout
Backbone	Temporal sequence encoder
Input dimension	$256 \times C$
Feature dimension	256
Pooling	Temporal average pooling
Classifier / Regressor	Task-specific prediction head
Dropout	0.5
Loss	Cross-entropy for classification / MSE for regression
Training strategy	End-to-end training

3.3 Comparison with SOTA Methods

Tables 2 and 3 show that the proposed method outperforms all baseline models on the four datasets. The margin is not identical across tasks. The improvement is smaller on the pure fault simulation setting. The improvement is clearer on tasks that are closer to digital twin prediction and system-level monitoring. In Table 2, the strongest baseline on the Electromechanical Fault Simulation Dataset is Longformer. Its F1 Score is 86.11%. Our method reaches 87.89%. The gain is 1.78 percentage points. On the Digital Twin Model Prediction Dataset, the F1 Score rises from 88.22% to 89.70%. The Accuracy also increases from 89.54% to 91.02%. Table 3 shows the same trend. On the Fault Tolerant Control System Dataset, the F1 Score increases from 87.12% to 88.34%. On the Electromechanical System Digital Twin Dataset, the gain is larger. The F1 Score rises from 89.23% to 90.12%. The Accuracy rises from 90.56% to 91.34%. These results indicate that the framework works better when fault evolution and state estimation and control response must be modeled in the same process. The advantage is less obvious when the task is limited to static fault discrimination. The improvement is not explained only by

generic feature extraction. It is more closely related to constrained state representation and fault-relevant event isolation and uncertainty-aware fault evaluation.

Table 2: Comparison of Our Model with SOTA methods on Electromechanical Fault Simulation and Digital Twin Model Prediction Datasets

Model	Electromechanical Fault Simulation Dataset				Digital Twin Model Prediction Dataset			
2-9	Accuracy	Recall	F1 Score	AUC	Accuracy	Recall	F1 Score	AUC
ALBERT	84.56 ± 0.52	83.97 ± 0.63	83.21 ± 0.58	83.45 ± 0.47	86.78 ± 0.49	86.23 ± 0.55	85.67 ± 0.61	85.92 ± 0.53
ERNIE	85.34 ± 0.47	84.79 ± 0.54	84.02 ± 0.60	84.28 ± 0.50	87.45 ± 0.44	86.89 ± 0.52	86.12 ± 0.57	86.37 ± 0.48
DeBERTa	86.12 ± 0.43	85.56 ± 0.50	84.78 ± 0.55	85.03 ± 0.46	88.23 ± 0.42	87.67 ± 0.49	86.91 ± 0.54	87.16 ± 0.45
MobileBERT	85.89 ± 0.48	85.32 ± 0.57	84.54 ± 0.62	84.79 ± 0.51	87.98 ± 0.46	87.42 ± 0.53	86.66 ± 0.59	86.91 ± 0.50
ELECTRA	86.67 ± 0.41	86.11 ± 0.48	85.33 ± 0.53	85.58 ± 0.44	88.76 ± 0.40	88.20 ± 0.47	87.44 ± 0.52	87.69 ± 0.43
Longformer	87.45 ± 0.39	86.89 ± 0.46	86.11 ± 0.51	86.36 ± 0.42	89.54 ± 0.38	88.98 ± 0.45	88.22 ± 0.50	88.47 ± 0.41
Ours	**89.23 ± 0.37**	**88.67 ± 0.44**	**87.89 ± 0.49**	**88.14 ± 0.40**	**91.02 ± 0.35**	**90.46 ± 0.42**	**89.70 ± 0.47**	**89.95 ± 0.39**

Table 3: Comparison of our method with SOTA methods on Fault Tolerant Control System and Electromechanical System Digital Twin datasets

Model	Fault Tolerant Control System Dataset				Electromechanical System Digital Twin Dataset			
2-9	Accuracy	Recall	F1 Score	AUC	Accuracy	Recall	F1 Score	AUC
ALBERT	84.56 ± 0.47	83.92 ± 0.58	83.15 ± 0.62	83.67 ± 0.51	86.78 ± 0.49	86.21 ± 0.63	85.45 ± 0.50	85.89 ± 0.54
ERNIE	85.89 ± 0.39	85.34 ± 0.52	84.67 ± 0.59	85.12 ± 0.43	88.02 ± 0.42	87.56 ± 0.55	86.89 ± 0.61	87.34 ± 0.46
DeBERTa	86.45 ± 0.36	85.89 ± 0.49	85.12 ± 0.57	85.56 ± 0.41	88.67 ± 0.44	88.12 ± 0.51	87.34 ± 0.58	87.78 ± 0.48
MobileBERT	87.12 ± 0.34	86.56 ± 0.46	85.89 ± 0.54	86.23 ± 0.39	89.34 ± 0.40	88.78 ± 0.53	88.01 ± 0.60	88.45 ± 0.42
ELECTRA	87.78 ± 0.32	87.23 ± 0.44	86.45 ± 0.52	86.89 ± 0.37	90.01 ± 0.38	89.45 ± 0.49	88.67 ± 0.56	89.12 ± 0.40
Longformer	88.34 ± 0.30	87.89 ± 0.42	87.12 ± 0.50	87.56 ± 0.35	90.56 ± 0.36	90.01 ± 0.47	89.23 ± 0.54	89.67 ± 0.38
Ours	**89.67 ± 0.31**	**89.12 ± 0.40**	**88.34 ± 0.48**	**88.78 ± 0.36**	**91.34 ± 0.35**	**90.89 ± 0.45**	**90.12 ± 0.52**	**90.56 ± 0.37**

3.4 Ablation Study

Tables 4 and 5 show that all three components contribute to the final performance, but their effects are not identical. The largest drop usually appears when Constraint-Driven Manifold Encoding is removed. On the Electromechanical Fault Simulation Dataset, the F1 Score falls from 87.89% to 86.23%, a decrease of 1.66 percentage points, which is larger than the drop caused by removing Agent-Based Event Segmentation or Probabilistic Fault Prediction. A similar pattern appears on the Fault-Tolerant Control System Dataset, where removing manifold encoding reduces F1 Score from 88.34% to 86.89%. This suggests that the constrained manifold is doing more than dimensionality reduction; it is preserving the feasible operating structure that later stages depend on. The contribution of Agent-Based Event Segmentation is more moderate but still consistent across datasets. Its removal causes smaller declines than removing manifold encoding, yet the effect remains visible on both fault simulation and digital twin settings. This pattern suggests that event segmentation is most useful when fault development is distributed over time and cannot be captured well from state snapshots alone. Probabilistic Fault Prediction also has a stable effect, especially on the two digital twin related datasets, where its removal lowers F1 Score from 89.70% to 89.24% on the Digital Twin Model Prediction Dataset and from 90.12% to 89.45% on the Electromechanical System Digital Twin Dataset. Taken together, these results indicate that the full model works best because the three components play different roles: manifold encoding stabilizes the representation space, event segmentation isolates fault-relevant transitions, and probabilistic prediction improves decision quality under uncertainty.

Table 4: Ablation Study on Electromechanical Fault Simulation and Digital Twin Model Prediction Datasets

Model	Electromechanical Fault Simulation Dataset				Digital Twin Model Prediction Dataset			
2-9	Accuracy	Recall	F1 Score	AUC	Accuracy	Recall	F1 Score	AUC
w./o. Constraint Driven Manifold Encoding	87.56 ± 0.42	87.01 ± 0.49	86.23 ± 0.54	86.48 ± 0.45	89.34 ± 0.40	88.78 ± 0.47	88.02 ± 0.52	88.27 ± 0.44
w./o. Agent Based Event Segmentation	88.12 ± 0.39	87.56 ± 0.46	86.78 ± 0.51	87.03 ± 0.42	90.01 ± 0.37	89.45 ± 0.44	88.69 ± 0.49	88.94 ± 0.41
w./o. Probabilistic Fault Prediction	88.67 ± 0.41	88.11 ± 0.48	87.33 ± 0.53	87.58 ± 0.44	90.56 ± 0.39	90.00 ± 0.46	89.24 ± 0.51	89.49 ± 0.43
Ours	**89.23 ± 0.37**	**88.67 ± 0.44**	**87.89 ± 0.49**	**88.14 ± 0.40**	**91.02 ± 0.35**	**90.46 ± 0.42**	**89.70 ± 0.47**	**89.95 ± 0.39**

Table 5: Ablation Study on Fault Tolerant Control System and Electromechanical System Digital Twin Datasets

Model	Fault Tolerant Control System Dataset				Electromechanical System Digital Twin Dataset			
2-9	Accuracy	Recall	F1 Score	AUC	Accuracy	Recall	F1 Score	AUC
w./o. Constraint Driven Manifold Encoding	88.12 ± 0.35	87.56 ± 0.46	86.89 ± 0.54	87.23 ± 0.39	90.12 ± 0.38	89.67 ± 0.50	88.89 ± 0.58	89.34 ± 0.42
w./o. Agent Based Event Segmentation	88.45 ± 0.33	87.89 ± 0.44	87.12 ± 0.52	87.56 ± 0.37	90.45 ± 0.36	89.89 ± 0.48	89.12 ± 0.56	89.67 ± 0.40
w./o. Probabilistic Fault Prediction	88.78 ± 0.32	88.23 ± 0.42	87.56 ± 0.50	87.89 ± 0.35	90.78 ± 0.34	90.23 ± 0.46	89.45 ± 0.54	90.01 ± 0.38
Ours	**89.67 ± 0.31**	**89.12 ± 0.40**	**88.34 ± 0.48**	**88.78 ± 0.36**	**91.34 ± 0.35**	**90.89 ± 0.45**	**90.12 ± 0.52**	**90.56 ± 0.37**

4 Conclusions and Future Work

This study examined how digital twins can be used for fault simulation, fault prediction, and fault-tolerant control in electromechanical systems. The main point is not only that a virtual model can mirror system behavior, but that it can also support earlier and more reliable control adjustment when faults begin to develop. From the experimental results, the proposed framework is particularly useful in settings where state evolution, fault propagation, and control response are tightly coupled. Its advantage is more evident on digital twin related tasks than on simpler fault discrimination settings, which suggests that the method is better suited to dynamic system management than to isolated classification alone.

At the same time, several limitations remain. The current framework is still relatively heavy in both modeling and computation, which may limit its use in real-time applications or in systems with restricted hardware resources. Its performance also depends on the quality of system observations. If sensor readings are noisy, delayed, or incomplete, the benefit of the digital twin may weaken because the prediction and control stages rely on the same input stream. Another issue is deployment. Even when the model performs well offline, practical fault-tolerant control requires stable interaction between prediction, state update, and control execution, which is harder to guarantee in complex operating environments. Future work should therefore move in two directions. One is to simplify the framework so that it can be deployed with lower computational cost and shorter response time. The other is to improve uncertainty handling and system interpretability, especially in cases where sensor quality and operating conditions vary over time. These points matter if digital twin based fault-tolerant control is to move from experimental validation to broader industrial use.

Conflict of Interest Statement

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author Contributions

Haijun Yang contributed to conceptualization, methodology, software, validation, formal analysis, investigation, data curation, original draft preparation, review and editing, visualization, supervision, and funding acquisition. The author has read and agreed to the published version of the manuscript.

Funding

2024 Henan Provincial Science and Technology Research Project: "Wireless Remote Monitoring and Fault Identification of Multi source Information for Electromechanical hydraulic Systems Based on Industrial Internet of Things" (No. 242102221015)

References

[1] Björnsson, B., Borrebaeck, C., Elander, N., Gasslander, T., Gawel, D. R., Gustafsson, M., et al. (2019). Digital twins to personalize medicine. *Genome Medicine*, 12, 4.

[2] Bruynseels, K., Santoni de Sio, F., & Van den Hoven, J. (2018). Digital twins in health care: ethical implications of an emerging engineering paradigm. *Frontiers in Genetics*, 9, 31.

[3] Croatti, A., Gabellini, M., Montagna, S., & Ricci, A. (2020). On the integration of agents and digital twins in healthcare. *Journal of Medical Systems*, 44, 161.

[4] El Saddik, A. (2018). Digital twins: The convergence of multimedia technologies. *IEEE Multimedia*, 25, 87–92.

[5] Farghaly, H. M., & El-Hafeez, T. A. (2023). A high-quality feature selection method based on frequent and correlated items for text classification. *Soft Computing – A Fusion of Foundations, Methodologies and Applications.*

[6] Ferrari, A., & Willcox, K. (2024). Digital twins in mechanical and aerospace engineering. *Nature Computational Science*, 4, 178–183.

[7] Fiok, K., Karwowski, W., Gutierrez-Franco, E., Davahli, M. R., Wilamowski, M., Ahram, T., et al. (2021). Text guide: Improving the quality of long text classification by a text selection method based on feature importance. *IEEE Access.*

[8] Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8, 108952–108971.

[9] Graziosi, P., Li, Z., & Neophytou, N. (2023). Electra code: Full-band electronic transport properties of materials. *Computer Physics Communications*, 287, 108670.

[10] Huan, H., Guo, Z., Cai, T., & He, Z. (2022). A text classification method based on a convolutional and bidirectional long short-term memory model. *Connection Science.*

[11] Iranshahi, K., Brun, J., Arnold, T., Sergi, T., & Müller, U. C. (2025). Digital twins: Recent advances and future directions in engineering fields. *Intelligent Systems with Applications*, 26, 200516.

[12] Ketzler, B., Naserentin, V., Latino, F., Zangelidis, C., Thuvander, L., & Logg, A. (2020). Digital twins for cities: A state-of-the-art review. *Built Environment*, 46, 547–573.

[13] Mohammadi, N., & Taylor, J. E. (2017). Smart city digital twins. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1–5). IEEE.

[14] Rosen, R., Von Wichert, G., Lo, G., & Bettenhausen, K. D. (2015). About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48, 567–572.

[15] Roy, S. S., & Nilizadeh, S. (2024). Phishlang: A real-time, fully client-side phishing detection framework using MobileBERT. *arXiv preprint arXiv:2408.05667*.

[16] Stark, R., Fresemann, C., & Lindow, K. (2019). Development and operation of digital twins for technical systems and services. *CIRP Annals*, 68, 129–132.

[17] Sun, D., He, J., Zhang, H., Qi, Z., Zheng, H., & Wang, X. (2025). A Longformer-based framework for accurate and efficient medical text summarization. In *2025 8th International Conference on Advanced Algorithms and Control Engineering (ICAACE)* (pp. 1527–1531). IEEE.

[18] Tao, F., & Qi, Q. (2019). Make more digital twins. *Nature*, 573, 490–491.

[19] Tao, F., Xiao, B., Qi, Q., Cheng, J., & Ji, P. (2022). Digital twin modeling. *Journal of Manufacturing Systems*, 64, 372–389.

[20] Van der Valk, H., Haße, H., Möller, F., Arbter, M., Henning, J.-L., & Otto, B. (2020). A taxonomy of digital twins.

[21] Van Dinter, R., Tekinerdogan, B., & Catal, C. (2022). Predictive maintenance using digital twins: A systematic literature review. *Information and Software Technology*, 151, 107008.

[22] Yaqoob, I., Salah, K., Uddin, M., Jayaraman, R., Omar, M., & Imran, M. (2020). Blockchain for digital twins: Recent advances and future research challenges. *IEEE Network*, 34, 290–298.