Hidden Markov Inference Framework for Error Propagation Mitigation in Modular Digital Twins

Annice Najafi
Department of Industrial and Manufacturing Engineering
California State Polytechnic University, Pomona
Pomona, CA 91768, USA

Shokoufeh Mirzaei*
Department of Industrial and Manufacturing Engineering
California State Polytechnic University, Pomona
Pomona, CA 91768, USA

Abstract

In this article, we propose a Hidden Markov inference framework for residual modeling and mitigation in modular digital twins. We first fit a Gaussian Hidden Markov Model (HMM) to the residual sequence to infer latent error regimes, estimate regime transition probabilities, and compute smoothed posterior probabilities of regime occupancy over time. We then apply the Hungarian algorithm to establish a one-to-one mapping between the learned HMM states and meaningful ground-truth regimes. To illustrate how the framework works, we model the physical asset using a higher-fidelity physics simulator and the digital twin using lightweight ARX surrogates across a six-module pipeline. The results on this illustrative example show that the HMM reliably detects regime changes, provides interpretable regime posteriors, and enables state-dependent bias corrections and targeted intervention that reduce surrogate prediction error relative to an uncorrected baseline. Although demonstrated on a toy example, the proposed framework provides insight into hidden error dynamics, regime transitions, and targeted error mitigation in modular, black-box digital twins.

Keywords: Digital Twin Error Propagation Stochastic Model Hidden Markov Model

1. Introduction

Digital twins were first popularized in aerospace engineering, but they are now becoming more popular in many other fields where there is a need for linking a digital replica with the physical counterpart (Singh

*Corresponding Author. Email: smirzaei@cpp.edu

et al., 2021). A digital twin is a dynamic virtual representation of a physical entity that uses real-time data synchronization to mirror the status, working condition, and life cycle of the corresponding system (Abdelrahman et al., 2025). Studies predict that by 2027, more than 40% of large corporations will utilize digital twins to boost revenue (Attaran and Celik, 2023). However, the reliability of digital twins is dependent on the ability to accurately track how errors accumulate and spread within the system (Hartwell et al., 2024). As systems age, small issues such as sensor noise or bias drift pile up, and silently hurt performance (Sayghe, 2025; Thai et al., 2025). Therefore, as these models see wider use, we must track how errors propagate to create better intervention strategies (Garcia et al., 2025).

Most traditional monitoring considers raw signals or single error metrics. While these approaches flag anomalies, they rarely identify the source or the solution (Park et al., 2020). Conversely, an increasing body of research is advancing towards probabilistic and uncertainty-aware monitoring. For example, Haslbeck and Braml (2024) have employed Monte Carlo methods to analyze prediction errors in digital twins. Similarly, Cao et al. (2024) performed an uncertainty analysis on a mine ventilation digital twin system, and quantified how errors in key parameters affected the system's outputs. They found out that even a modest (5%) variation in a core parameter caused the twin to lose roughly 34% of correct airflow data points which highlights the digital twin's sensitivity to parameter uncertainty (Cao et al., 2024). These efforts illustrate the value of stochastic, dynamic analysis of digital twin performance where instead of assuming a fixed error margin, they simulate how errors might evolve under different conditions or over time.

Despite these advances, most current research and industrial practice for fault detection still rely on static confidence bounds or fixed error thresholds, rather than the dynamic error forecasts needed to catch error type regime shifts in advance (Grimm et al., 2024; Salehi and Rashidi, 2018). A digital twin operating in the real world may encounter changing regimes. For instance, a machine might enter a new wear-out phase, or an environmental change might invalidate prior calibration (Tang et al., 2025; He et al., 2025). Static error limits might be too coarse or too conservative to detect such shifts in a timely manner. Thus, predictive monitoring techniques that continuously forecast the twin's expected error and flag when that forecast deviates significantly to detect potential emerging fault is needed. In other words, to fully trust and utilize digital twins at scale, we must move from merely applying after-the-fact error bounds to anticipating error propagation in real time and guiding interventions accordingly.

In our prior work, we sought to address these limitations by proposing a preliminary stochastic framework demonstrated on a digital twin of a two-link robotic arm (Najafi et al., 2025). Inspired by embedded control systems (Hiller et al., 2001), we decomposed the twin into discrete modules and utilized Neural Networks and Random Forests to model error evolution. Through the use of Markov chain models, we were able to predict error accumulation and estimate the mean first passage time to failure. We further introduced an importance measure to identify the most influential components in the error propagation chain. However, our previous study relied on simplified assumptions which limited its applicability to more generalized scenarios (Najafi et al., 2025).

Here, we develop a controlled toy digital twin setting in which a simplified AutoRegressive eXogenous (ARX) surrogate is compared against a higher-fidelity physics simulator, and we show that a stochastic model trained on surrogate residuals can recover injected operating regimes and enable regime-conditioned bias correction to reduce surrogate prediction error. We propose this model to provide a preliminary framework for understanding hidden error dynamics and targeted intervention strategies in modular digital twins where individual components are often treated as black-box surrogates and direct access to internal states is limited. In such settings, the most reliable information available for monitoring is frequently the residual structure because residuals naturally encode local modeling deficiencies and how those deficiencies propagate downstream through inter-module coupling. Modeling different types of errors as latent states of a Hidden Markov Model (HMM) allows us to quantify when the twin is likely operating different error regime

types such as nominal, noisy, drift, or dynamics-mismatch regimes.

2. Methods

As a general framework, consider a modular digital twin comprising of n interconnected modules $M_1, M_2, \ldots, M_n$ arranged in a processing pipeline. Each module M_i receives input x_i and produces output y_i, where the output of one module serves as input to downstream modules. The physical asset generates ground-truth outputs y_i^{phys}, while the digital twin employs surrogate models $\hat{f}_i$ to approximate each module's behavior, and produces predictions y_i^{surr}. The surrogate models may take any form appropriate to the application:

1. Data-driven models (ARX, neural networks, Gaussian processes)
2. Reduced-order physics models
3. Hybrid physics-informed models

2.1. Hidden Markov Model

We model the system as operating in one of K latent regimes $S_t \in 1, \ldots, K$, where each regime represents a distinct error pattern (e.g., nominal operation, sensor degradation, model mismatch, drift). We then consider regime transitions to follow a first-order Markov process. Because we cannot directly observe which regime the system is in and instead we are able to only observe the residuals it produces, we employ a HMM to infer the latent error regimes from the observed residual sequence. The HMM learns the characteristic residual pattern associated with each regime, estimates the probabilities of transitioning between regimes, and computes the posterior probability of each regime at each time point. The generic framework would allow us to perform both real-time regime monitoring and targeted interventions based on the inferred system state. In the sections below, we consider a simplistic example for demonstration purposes.

Due to the propagation of error through modules, we are unable to observe the state of the system directly. Instead, for each module m, we observe an e_t vector from module residuals. Where y_{ARX} and y_{phys} correspond to the surrogate and ground-truth values respectively.

$$e_t^{(m)} = y_{surr}^{(m)}(t) - y_{phys}^{(m)}(t) \tag{1}$$

These residuals demonstrate where and how the surrogate diverges from physics which we would then use as our key signal for regime inference. We assume that the system alternates among a small set of latent error regimes or states ($S_t \in \{1, \ldots, K\}$).

Regime switches follow a first-order Markov process where within a regime, residuals are approximately Gaussian (We adopt Gaussian emissions as a practical baseline. We note that although marginal residuals can be non-Gaussian due to regime mixing, Gaussian state-conditional emissions provide stable inference and correction in the toy example we use later). We then model $\{S_t\}_{t=1}^T$ as a Markov chain with initial distribution $\pi \in \Delta^{K-1}$ and transition matrix $A \in \mathbb{R}^{K \times K}$ such that $A_{ij} = \Pr(S_t = j \mid S_{t-1} = i)$, $\sum_j A_{ij} = 1$. Conditioned on $S_t = k$, the residual vector e_t follows a Gaussian emission:

$$e_t \mid (S_t = k) \sim \mathcal{N}(\mu_k, C_k), \qquad k = 1, \ldots, K. \tag{2}$$

Where μ_k is the mean vector of residuals for regime k and C_k is the covariance matrix of residuals for regime k. In practice we z-score each residual channel to balance scales prior to fitting.

We model the emission distribution using conditionally independent Gaussian residual channels,

$$p(e_t \mid S_t = k) = \prod_{j \in \{1,\dots,D\}} \mathcal{N}\Big(e_t^{(j)} | \mu_{k,j}, \sigma_{k,j}^2\Big), \tag{3}$$

which is equivalent to a multivariate Gaussian with diagonal covariance (D is the number of modules).

2.1.1. HMM Learning (EM/Baum-Welch)

Given residuals $\{e_t\}_{t=1}^T$, our goal is to maximize the log-likelihood of observing the residuals vector given the sequence of states or system regimes via Expectation Maximization (EM). We let $b_k(e_t) = \mathcal{N}(e_t|\mu_k, C_k)$ denote the emission likelihood. The forward-backward (E-step) recursions are then as follows:

$$\alpha_t(j) = \Big(\sum_i \alpha_{t-1}(i) A_{ij}\Big)\, b_j(e_t), \quad \alpha_1(j) = \pi_j\, b_j(e_1), \tag{4}$$

$$\beta_t(i) = \sum_j A_{ij}\, b_j(e_{t+1})\, \beta_{t+1}(j), \quad \beta_T(i) = 1, \tag{5}$$

Where $\pi = (\pi_1, \pi_2, \dots, \pi_k)$ is the probability vector representing the initial state distribution of the system with $\pi_j = \mathsf{Pr}(S_1 = j), \sum_j \pi_j = 1$ and per-time normalization to avoid underflow. The smoothed posteriors and expected transitions are as follows:

$$\gamma_t(k) = \mathsf{Pr}(S_t = k \mid e_{1:T}) = \frac{\alpha_t(k)\beta_t(k)}{\sum_\ell \alpha_t(\ell)\beta_t(\ell)}, \tag{6}$$

$$\xi_t(i,j) = \mathsf{Pr}(S_{t-1} = i, S_t = j \mid e_{1:T}). \tag{7}$$

Where $\gamma_t(k)$ is the smoothed posterior for regime (k) at timepoint t (the probability that the hidden state at time t is k) and $\xi_t(i,j)$ is the probability that the model was in state i at $t-1$ and in state j at t given the sequence of residuals $e_{1:T}$. The M-step updates are then found as:

$$\pi_k \leftarrow \gamma_1(k), \qquad A_{ij} \leftarrow \frac{\sum_{t=2}^T \xi_t(i,j)}{\sum_{t=2}^T \sum_{j'} \xi_t(i,j')}, \tag{8}$$

$$\mu_k \leftarrow \frac{\sum_{t=1}^T \gamma_t(k)\, e_t}{\sum_{t=1}^T \gamma_t(k)}, \qquad C_k \leftarrow \frac{\sum_{t=1}^T \gamma_t(k)\, (e_t - \mu_k)(e_t - \mu_k)^\top}{\sum_{t=1}^T \gamma_t(k)}. \tag{9}$$

We iterate E/M steps until the (normalized) log-likelihood converges.

2.1.2. Decoding

For a hard segmentation we use Viterbi:

$$\delta_t(j) = \max_i\ \delta_{t-1}(i) A_{ij}\, b_j(e_t), \quad \psi_t(j) = \arg\max_i\ \delta_{t-1}(i) A_{ij}, \tag{10}$$

with backtracking to recover $\hat{S}_{1:T}$ ($\delta_t(j)$ is the probability score of the best path ending in state j at time t and $\psi_t(j)$ is an index indicating which previous state i resulted in the best path into j). For monitoring, we report smoothed probabilities $\gamma_t(k)$ (rows sum to 1). In this paper, we used the `depmixS4` package in R for HMM learning and all of the steps explained above.

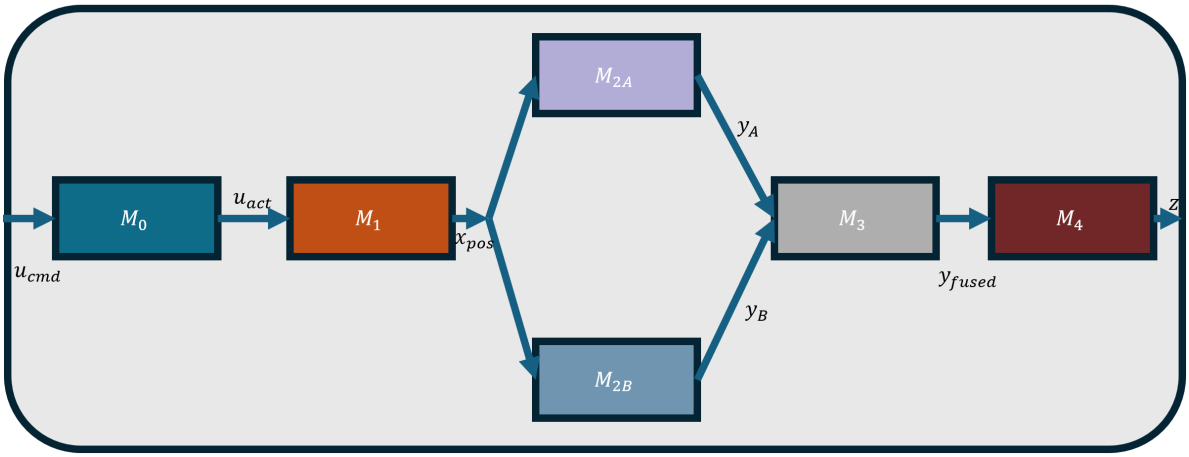

Figure 1: Six module digital twin model used for testing.

2.2. Hungarian Algorithm for State-to-Regime Matching

Now suppose that we have followed the steps explained above and obtained K HMM states and are aware of K ground-truth regimes. We now face a problem of matching the inferred regimes with the meaningful regimes existing within the system. To address this issue, we construct a benefit matrix $B \in \mathbb{R}^{K\times K}$ where entry B_{ij} represents the number of time points where HMM state i occurred during ground-truth regime j:

$$B_{ij} = \sum_{t=1}^{T} \mathbf{1}\{\hat{S}_t = i \text{ and } S_t^{\text{true}} = j\} \tag{11}$$

$\hat{S}_t$ is the Viterbi-decoded HMM state at time t and S_t^{true} is the ground-truth regime. We need to find a permutation $\pi : \{1,\dots,K\} \to \{1,\dots,K\}$ that assigns each HMM state to exactly one regime such that total overlap is maximized:

$$\pi^* = \arg\max_{\pi} \sum_{i=1}^{K} B_{i,\pi(i)} \tag{12}$$

Alternatively, we can seek a permutation matrix $P \in \{0,1\}^{K\times K}$ that solves:

$$\max_P \sum_{i=1}^{K}\sum_{j=1}^{K} P_{ij}B_{ij} = \max_P \operatorname{tr}(P^\top B) \tag{13}$$

subject to the following constraints:

$$\sum_{j=1}^{K} P_{ij} = 1 \quad \forall\, i \in \{1,\dots,K\} \tag{14}$$

$$\sum_{i=1}^{K} P_{ij} = 1 \quad \forall\, j \in \{1,\dots,K\} \tag{15}$$

$$P_{ij} \in \{0,1\} \tag{16}$$

These constraints would ensure that each HMM state is assigned to exactly one regime and each regime is assigned to exactly one state. The Hungarian algorithm traditionally solves a minimization problem. To

convert our maximization problem, we can define a cost matrix C:

$$C_{ij} = \max_{k,\ell}(B_{k\ell}) - B_{ij} \tag{17}$$

Minimizing the total cost under this transformation is equivalent to maximizing total benefit:

$$\arg\min_P \mathrm{tr}(P^\top C) = \arg\max_P \mathrm{tr}(P^\top B) \tag{18}$$

The Hungarian algorithm then proceeds as follows (we have included an Rmd notebook on our Github page that explains how the algorithm works):

1. For each row i, compute $C'_{ij} = C_{ij} - \min_j C_{ij}$
2. For each column j, compute $C''_{ij} = C'_{ij} - \min_i C'_{ij}$
3. Find a maximum matching using only zero-cost entries
4. If the matching is incomplete, adjust the reduced cost matrix to introduce new zeros while preserving existing zeros in the current matching
5. Repeat steps 3-4 until a complete assignment is found

3. Illustrative Example

Here, we consider a six module pipeline starting with an actuator low-pass block ($M0$) that converts the commanded input which we refer to as u_{cmd} into a corresponding actuation u_{act}; a mass-spring-damper plant ($M1$) that maps actuation to motion; two distinct position sensors ($M2a$/$M2b$) that measure x with different lags, biases, and noise; a fusion stage ($M3$) that combines the sensor readings into a single estimate y_{fused}; and a performance mapper ($M4$) that produces the operational Key Performance Indicator (KPI). In our setup, we consider a physical system that generates the ground-truth signals (including process and measurement noise), while the digital twin is a data-driven replicate that approximates each module's input-output behavior (via lightweight ARX surrogates). Comparing the digital twin's outputs to the physical system's outputs would then provide us with module-level residuals that localize where uncertainty enters and how it propagates. With this representation, our goal is to understand the full causal chain and provide a basis for monitoring digital twin health and error type regimes. This would then allow us to devise targeted error recovery interventions. Below, we summarize the physical asset modules and their functions in greater detail (we have also shown the structure of the model in Figure 1) we then continue by explaining how we use ARX surrogates to model the digital replica:

3.1. Physical Asset

Input Command Generator Synthesizes an excitation signal u_{cmd} through mixing sinusoids, a square-wave, and random step offsets.

$$u_k^{(\text{base})} = 0.6\,\sin(2\pi \cdot 0.25\,t_k) + 0.4\ \text{sign}\left(\sin(2\pi \cdot 0.05\,t_k)\right)$$
$$u_{\text{cmd},k} = \text{amp} \cdot \left(u_k^{(\text{base})} + \sum_{i\in\mathcal{I}} \delta_i\,\mathbf{1}_{\{k\geq i\}}\right)$$

where step times are defined $\mathcal{I} = \{\,1,\ 1+s,\ 1+2s,\ \ldots,\ 1+Ms\,\}, M = \left\lfloor \frac{n-1}{s} \right\rfloor$ and δ_i is the random step offset.

M0 Actuator low-pass filter (LPF) Smooths commands to realized actuation.

$$u_{\text{act},k} = a\, u_{\text{act},k-1} + (1-a)\, u_{\text{cmd},k-1} + w_k$$

Parameters:

- $u_{\text{cmd},k}$: Commanded input (pre-actuator) at discrete time k.
- $u_{\text{act},k}$: Realized actuation (post-actuator) at time k.
- Δt: Sample period (seconds).
- τ_{cmd}: Actuator time constant (seconds). Larger τ_{cmd} implied stronger smoothing or slower response.
- $a = \exp(-\Delta t/\tau_{\text{cmd}})$: Exact-discrete LPF coefficient ($0 < a < 1$).
- $w_k \sim \mathcal{N}(0, \sigma^2_{\text{proc}})$: Actuator/process noise with variance σ^2_{proc}.

We note that in the implementation, the recursion is applied for $k = 2, \ldots, n$ with initial condition $u_{\text{act},1} = 0$. A one-sample actuator delay is assumed and the update at time k uses the previous command $u_{\text{cmd},k-1}$ which is consistent with the implementation using `u[k-1]`.

M1 Plant (mass-spring-damper exemplar) Simulates physical dynamics using explicit Euler integration.

$$\begin{aligned} a_{k-1} &= \frac{u_{\text{act},k-1} - c\, v_{k-1} - k_{\text{stiff}}\, x_{k-1}}{m} + \xi_k \\ v_k &= v_{k-1} + \Delta t\, a_{k-1} \\ x_k &= x_{k-1} + \Delta t\, v_{k-1} \end{aligned}$$

Parameters:

- m: Mass (kg).
- c: Viscous damping coefficient (N·s/m).
- k_{stiff}: Spring stiffness (N/m).
- x_k, v_k: Position (m) and velocity (m/s) at time k.
- a_k: Discrete-time acceleration (m/s^2).
- $\xi_k \sim \mathcal{N}(0, \sigma^2_{\text{proc}})$: Plant process noise added to acceleration.

M2a/M2b Sensors A/B Models first-order sensing dynamics with bias and noise. We implement the sensor as a noise-free first-order lag on the true position, followed by additive bias and measurement noise:

$$\begin{aligned} \tilde{y}_{s,k} &= a_s\, \tilde{y}_{s,k-1} + (1-a_s)\, x_{k-1}, \qquad \tilde{y}_{s,1} = 0 \\ y_{s,k} &= \tilde{y}_{s,k} + b_s + \varepsilon_{s,k} \end{aligned}$$

Parameters:

- x_k: True position (plant output).

- $\tilde{y}_{s,k}$: Noise-free lagged sensor state for sensor $s \in \{A, B\}$ at time k.
- $y_{s,k}$: Sensor reading for sensor $s \in \{A, B\}$ at time k.
- τ_s: Time constant for sensor s (seconds).
- $a_s = \exp(-\Delta t/\tau_s)$: Discrete sensor lag coefficient.
- b_s: Additive bias (sensor offset).
- $\varepsilon_{s,k} \sim \mathcal{N}(0, \sigma^2_{\text{meas},s})$: Measurement noise (can differ per sensor).

M3 Sensor fusion Static weighted average.

$$y_f = \alpha\, y_{A,k} + (1 - \alpha)\, y_{B,k}, k = 1, \ldots, n$$

Parameters:

- y_A, y_B: Outputs of sensors A and B.
- $\alpha \in [0, 1]$: Fusion weight on sensor A (implies $1 - \alpha$ on sensor B).
- y_f: Fused measurement.

M4 Performance output Linear performance index (KPI).

$$z = w_1 x + w_2 \dot{x}$$

In the physics model we use $\dot{x} = v$, while in the ARX surrogate rollout we approximate $\dot{x}$ by a backward difference:

$$\dot{x}_k \approx \frac{x_k - x_{k-1}}{\Delta t}, \qquad \dot{x}_1 = 0.$$

Parameters:

- x: Position (m).
- $\dot{x}$ (or v): Velocity (m/s).
- w_1, w_2: Performance weights.
- z: Scalar performance index.

We consider this physics path that we explained above as the high-fidelity reference and consider it the truth (or the physical model representation). It explicitly propagates uncertainty forward through the chain.

We define the latent states of the Markov chain as follows:

1. **Nominal**: the system is healthy and there is no need for intervention.
2. **Sensor noisy**: one or both sensing channels exhibit elevated variances and/or intermittent spikes which could be due to environment or degradation. Residuals are large mainly in $e^{(2a)}$/$e^{(2b)}$ (errors corresponding to $M2a$ and $M2b$). while plant or actuation residuals remain small. For mitigation, we can reweight or temporarily down-select the noisy sensor, increase filtering, or trigger recalibration.

3. **Dynamics-off**: the plant no longer matches its nominal dynamics (for example, we may have parameter drift in m, c, k, unmodeled friction, or load changes). Structured errors appear primarily in $e^{(1)}$ and propagate to $e^{(3)}$/$e^{(4)}$. For mitigation, we can re-identify plant parameters (on-line ARX update), adapt controller gains, or switch to a robust fallback model.

4. **Drift**: bias-like, slowly varying offsets accumulate in one or more channels (sensors or actuation), producing nonzero-mean residuals over extended windows. For mitigation, we should estimate and subtract bias (zero-point/scale correction), schedule calibration, or apply bias-aware fusion.

3.2. Digital Twin

We utilize ARX surrogates for predicting module outputs. ARX uses modules' past outputs and inputs for prediction. We use the least squares methodology from Ljung's chapter 1 (Ljung, 1999). The general ARX model assumes a time invariant and linear difference equation with an exogenous input u_t and output y_t at time t where:

$$A(q^{-1})y_t = B(q^{-1})u_t + e_t \tag{19}$$

We refer to q^{-1} as the backshift operator, and define A and B as follows:

$$A(q^{-1}) = 1 - a_1 q^{-1} - \cdots - a_{n_a} q^{-n_a} \tag{20}$$

$$B(q^{-1}) = b_1 q^{-1-d} + ... + b_{n_b} q^{-n_b - d} \tag{21}$$

Where $\{a_i, b_j\}$ are coefficients learned from data through least squares, $d \geq 0$ is the optional input delay, and n_a, n_b indicate lag orders. Rearranging the equations, we get:

$$y_t = \sum_{i=1}^{n_a} a_i y_{t-i} + \sum_{j=1}^{n_b} b_j u_{t-d-j} + e_t \tag{22}$$

We build a regressor ϕ_t as follows:

$$\phi_t = [y_{t-1}, ..., y_{t-n_a}, u_{t-d-1}, ..., u_{t-d-n_b}] \tag{23}$$

Then we stack the rows over time and solve the below equation to get the $\{a_i, b_j\}$ coefficients:

$$\hat{\theta} = (\phi^\top \phi)^{-1} \phi^\top Y \tag{24}$$

Then we roll out the one-step model recursively over the full horizon.

We note that in our simulations we fix $d = 0$ and use the convention that the input polynomial begins at q^{-1}, so the smallest input lag in the regressor is u_{t-1}.

4. Results

We began our analysis by performing simulations of the process to understand whether the lightweight ARX surrogates could sufficiently capture the behavior of the physics model or not. Figure 2 displays the comparison between the ground-truth signals (solid lines) and the ARX approximations (dashed lines) across the six modules.

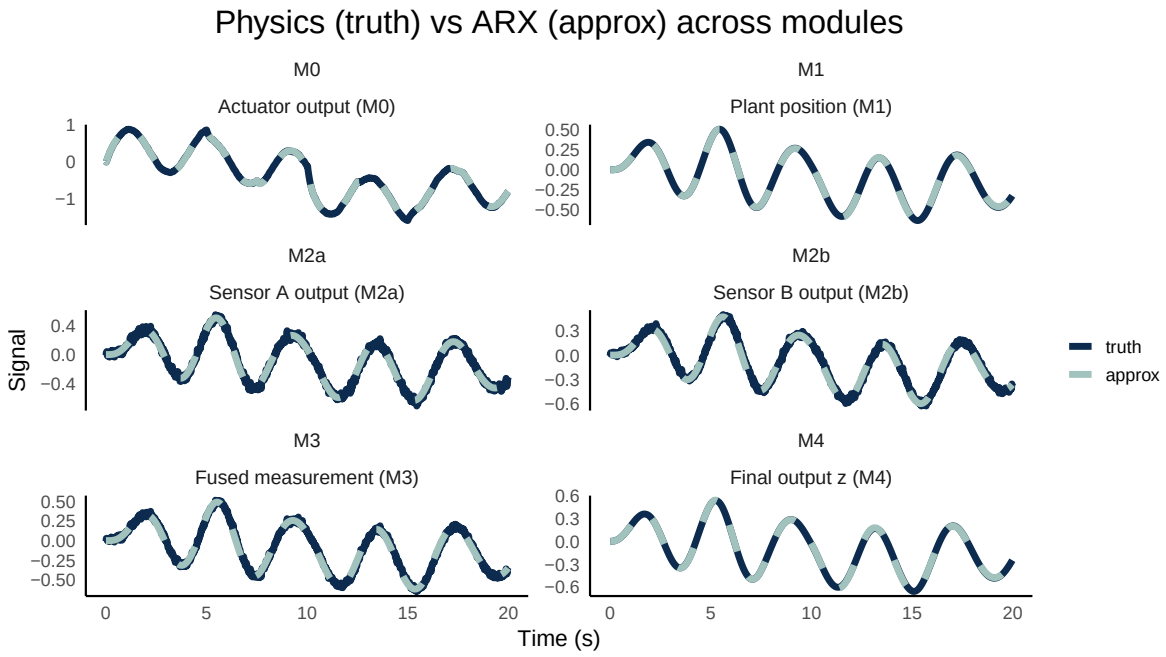

Figure 2: Comparison of ARX surrogate and physics outputs.

4.1. Existence of Structured Noise

As shown in the figure, physics and simulation curves show general agreement with some noticeable discrepancies (Figure 2). To determine if these discrepancies were due to random measurement noise or indicative of deeper ARX modeling deficits, we analyzed the statistical properties of the residual vector e_t. Our primary hypothesis was that if the ARX models were sufficient, the residuals would approximate white noise uncorrelated over time and normally distributed around zero (similar to what we show in Figure 3A for demonstration purposes - to create Figure 3A, we generated random white noise and calculated the autocorrelation analysis which we show in Figure 3B). However, instead our diagnostic plots showed that the errors are highly structured. As shown in the autocorrelation analysis plots in Figure 3D, the residuals for the plant ($e^{(1)}$) and output ($e^{(4)}$) exhibit a slow and linear decay rather than a sharp drop-off which is characteristic of white noise. The long memory in the error signal indicated that the ARX model was systematically undershooting or overshooting the true dynamics for extended periods, rather than jittering randomly. The time-series showed smooth, oscillatory error waves rather than high-frequency fuzz. Therefore, the observed structured noise justified the use of the HMM which we continue with in the next section. We note that a simple threshold-based detector would struggle with these drifting errors, and would likely trigger false positives or miss the onset of a regime change. While an HMM would explicitly model the transition probabilities between latent states, and would capture the persistence of these error regimes.

4.2. Regime Characterization Simulations

After we obtained the residuals and confirmed the existence of structured noise, we fitted a Gaussian HMM with $K = 4$ latent regimes to the standardized residual vector e_t. We trained the HMM via Expectation-Maximization (using `depmixS4`). In the E-step we computed smoothed state posteriors $\gamma_t(k) = P(S_t = k \mid e_{1:T})$, and in the M-step we updated the initial distribution, transition matrix, and state-conditional Gaussian parameters. Following our implementation, each residual component was modeled as a univariate Gaussian conditional on the state (equivalently, a multivariate Gaussian with diagonal covariance) which

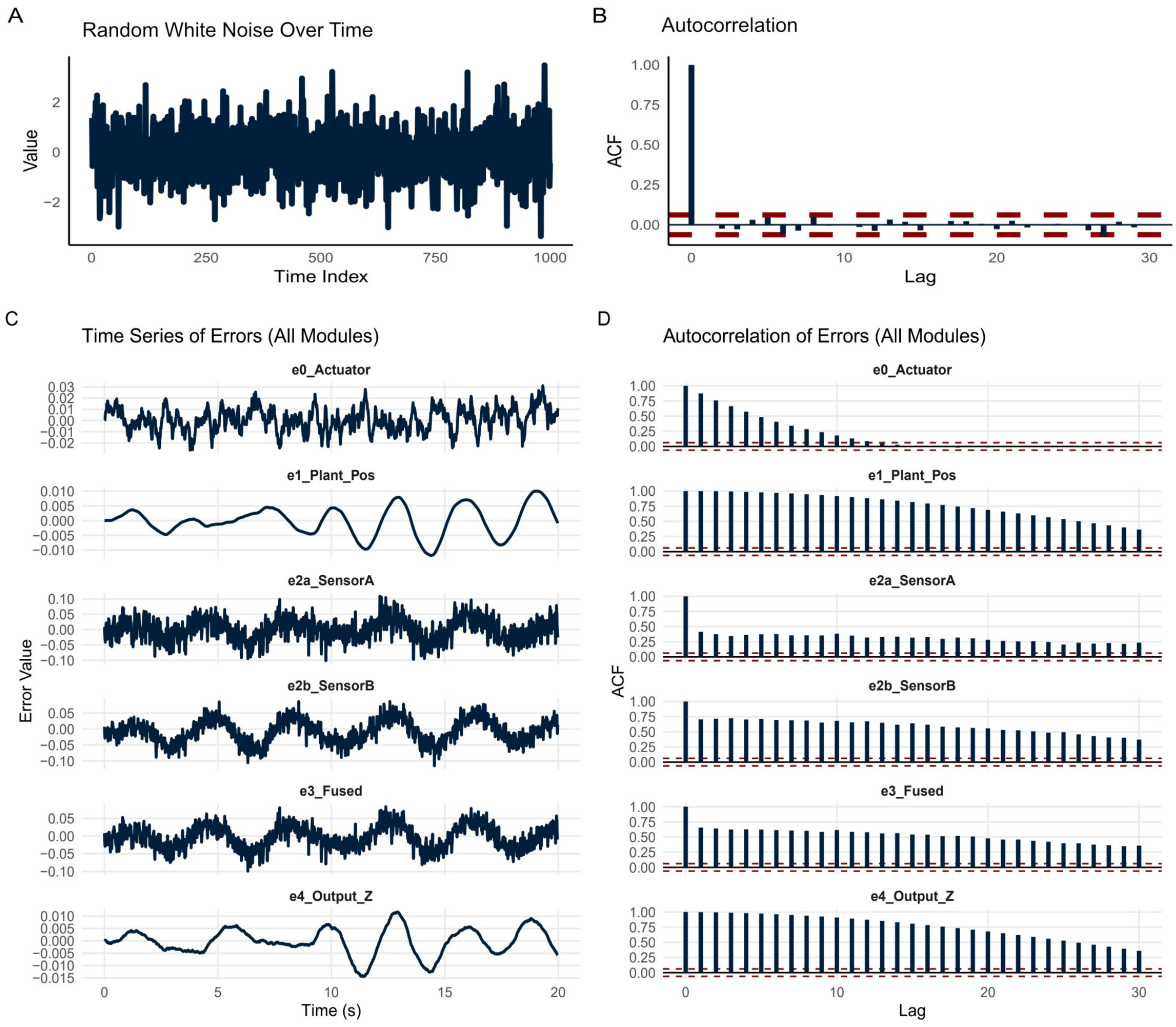

Figure 3: A shows random white noise values over time. B shows the autocorrelation function values for the generated random noise. C depicts module residuals and D depicts ACF plots for the six module residuals confirming that the noise is structured.

resulted in state-specific mean/variance fingerprints. Figure 4 shows the resulting smoothed regime posteriors over time.

We note that for the demonstration in this section, the HMM latent states are unlabeled and are therefore inherently arbitrary. The state index $k \in \{1, \ldots, K\}$ has no intrinsic semantic meaning, so the plotted posteriors indicate when the residuals are best explained by different learned regimes, but do not by themselves identify which regime corresponds to "Nominal," "Sensor noisy," "Dynamics-off," or "Drift" without an additional post-hoc state-labeling step based on the fitted emission parameters.

4.3. Targeted Interventions

Next in our analysis, we devised a targeted intervention strategy to post-process the surrogate ARX predictions by subtracting a learned and state-dependent bias in order to mitigate error propagation. Towards this goal, we first estimated per-state residual means and then computed a time-varying bias estimate for

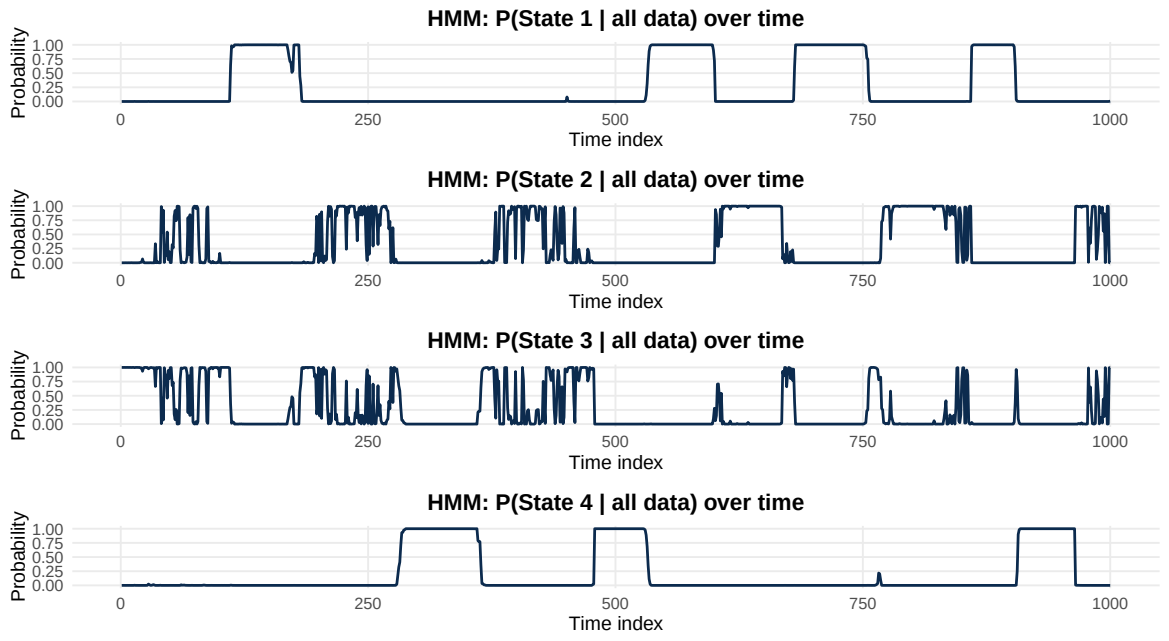

Figure 4: Conditional probabilities of the system residing in state $k \in \{1, 2, 3, 4\}$ given the residual vector.

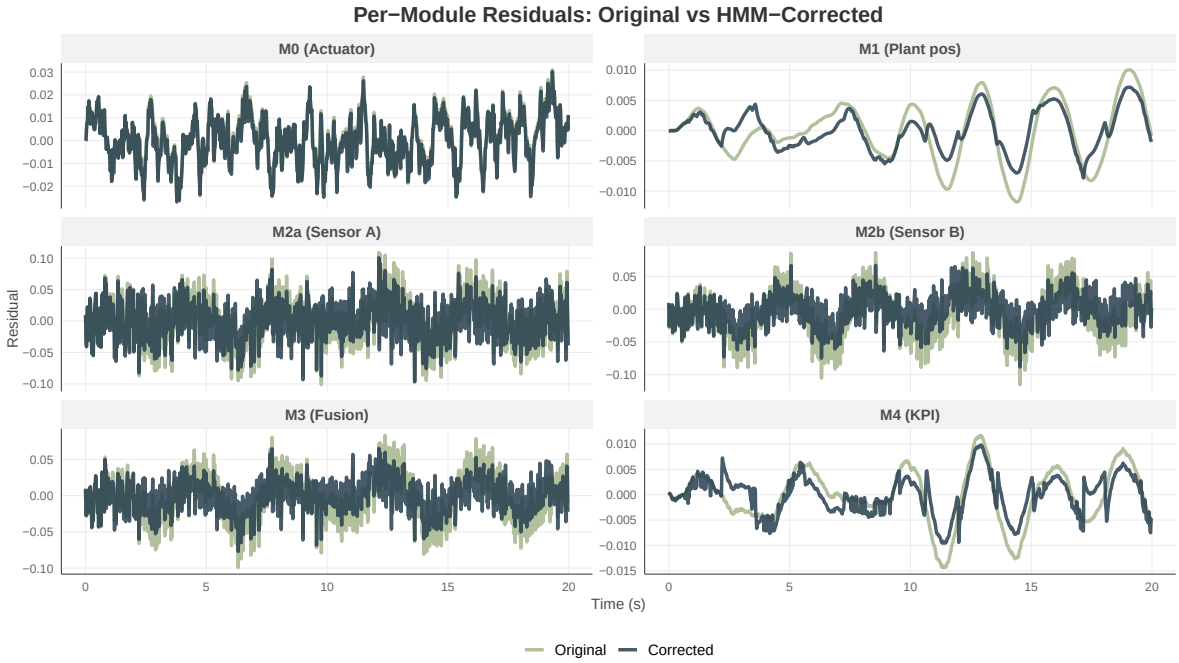

Figure 5: Per-module residual values over time before and after HMM-based bias correction.

each residual channel as a posterior-weighted mixture,

$$\hat{b}_j(t) = \sum_{k=1}^{K} \gamma_t(k)\, \mu_{k,j}, \tag{25}$$

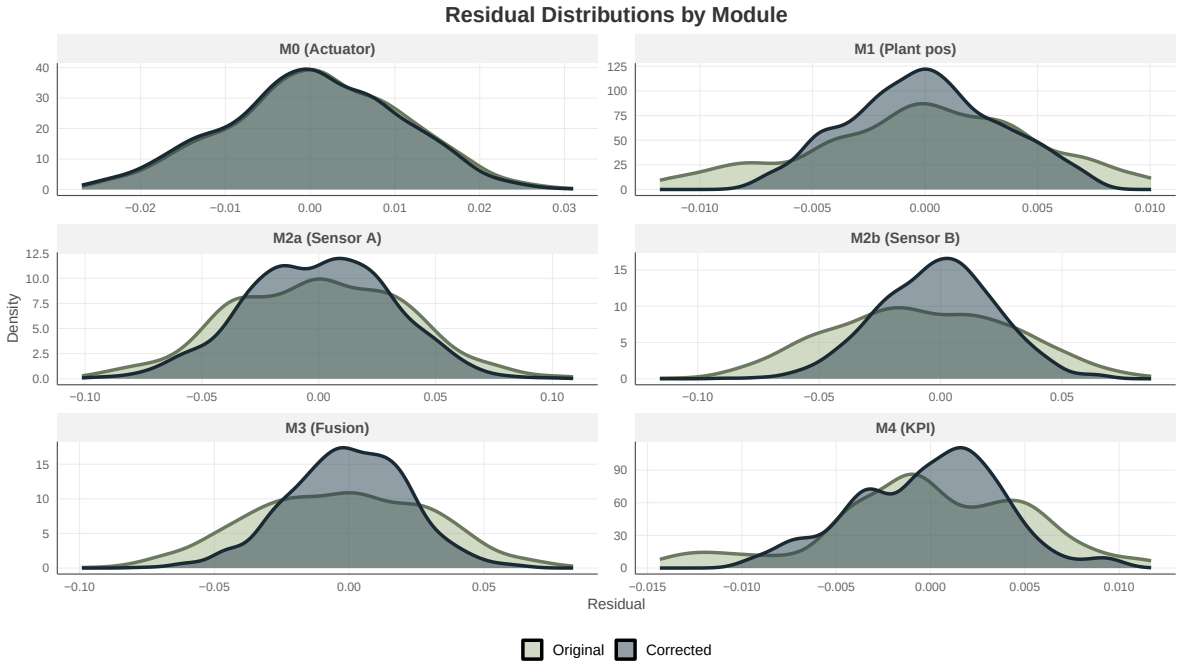

Figure 6: Per-module residual distributions before and after HMM-based bias correction.

where $\gamma_t(k) = P(S_t = k \mid e_{1:T})$ are the smoothed HMM posteriors and $\mu_{k,j}$ is the mean of residual channel j in state k. We scaled $\hat{b}_j(t)$ by a gain factor ($\kappa = 0.8$) and applied an exponential moving average ($\alpha = 0.15$) followed by a rate limiter (maximum step size $r_{\max} = 0.05$) to ensure a smooth, bounded correction signal. The corrected surrogate outputs were then obtained by subtracting the resulting bias estimate from the corresponding module outputs (e.g., $u_{\text{act}}, x, y_A, y_B$), after which downstream fused signals and KPIs were recomputed.

Figure 5 compares the original and corrected residual time series which demonstrates that the residuals have visibly been lowered after correction (shown with blue curves) at least in the $M1$, $M2b$, $M3$, and $M4$ modules. To better visualize the residual values, we plotted the distributions of residuals for each module and have shown the density plots in Figure 6. As shown in the figure the residual distributions become more concentrated around zero with reduced variance after correction for all modules except $M0$.

4.4. Deliberate Fault Injection for Testing

In the previous sections, we introduced our model and performed simulations to show how our HMM model would allow us to devise targeted intervention strategies to mitigate the propagation of error through the digital twin system. A natural question that arises is how accurate our model is in identifying different types of errors. To address this question, we introduced specific fault regimes into the simulation according to a predefined schedule: nominal (0-5s), sensor-noisy (5-10s), dynamics-off (10-14.5s), and drift (14.5-20). We injected regime-dependent perturbations into the surrogate/observed streams to induce controlled discrepancies relative to the nominal physics baseline. Below, we explain the method of error injection for each module:

1. For the “Sensor-Noisy” regime, we injected additional Gaussian noise into the sensor channels.

2. For “Dynamics-Off,” we mathematically attenuated the position response to simulate a loss of stiff-

ness or mechanical degradation.

3. For "Drift," we added a cumulative linear bias to Sensor A.

We then calculated per-module residuals $e_0, e_1, e_{2a}, e_{2b}, e_3, e_4$ at each time step by subtracting the nominal physics baseline from the fault-injected observed values, and then formed the HMM observation vector by augmenting these residual channels with four residual-derived diagnostics as defined below:

- Sum of absolute magnitudes: $e_{\text{abs\_sum}}(t) = \sum_{i\in\mathcal{I}} |e_i(t)|$
- Root-mean-square magnitude: $e_{\text{rms}}(t) = \sqrt{\frac{1}{6}\sum_{i\in\mathcal{I}} e_i(t)^2}$
- First differences ($k \in \{1, 3\}$): $\Delta e_k(t) = e_k(t) - e_k(t-1)$, with $\Delta e_k(1) = 0$

where $\mathcal{I} = \{0, 1, 2a, 2b, 3, 4\}$. All channels/features were z-scored across time before fitting the Gaussian-emission HMM (we note that we include first differences only for e_1 (plant-position residual) and e_3 (fusion residual) because they provide sensitive indications of regime changes through capturing changes in residual dynamics rather than only magnitude). Because HMM state labels are arbitrary, we evaluated identification accuracy by mapping inferred states to ground-truth regimes via a maximum-overlap assignment (Hungarian matching) on the Viterbi path. We first computed the most likely discrete state sequence via the Viterbi algorithm and formed a state-regime contingency table by counting, for each inferred state, how often it occurred during each scheduled fault interval. We then solved a one-to-one assignment problem that maximized the total agreement between inferred states and ground-truth regimes thereby extracting a permutation that relabeled the HMM states as nominal, sensor-Noisy, dynamics-off, and drift for evaluation purposes. After applying this relabeling, we visualized the smoothed posterior probabilities over time with the scheduled regime boundaries overlaid in Figure 7. As shown in the figure, the framework was able to accurately identify regimes from data with some difficulties in identifying nominal from sensor noisy regimes (We solved the maximum-overlap one-to-one assignment using the Hungarian algorithm via `clue` package in R (Hornik, 2005)). In Figure 8 we show for each residual-derived feature, how its z-scored values distribute across the four ground-truth regimes (nominal, sensor-noisy, dynamics-off, drift). Features that separate regimes well show clear shifts in median and/or spread between distributions. For example, the drift regime stands out with large positive shifts in the aggregate error features (e_abs_sum, e_rms) and downstream residuals such as e_3, while dynamics-off produces a pronounced change in the plant-related channel (e_1) and propagates to e_4. In contrast, the sensor-noisy regime is characterized more by wider distributions (higher variance) rather than a large mean shift, especially in sensor-related features.

5. Discussion

Here, we developed a stochastic, module-level framework for understanding and mitigating error propagation in digital twins by decomposing a simple six-module pipeline consisting of an actuator LPF, mass-spring-damper plant, two sensors, linear fusion module, and a KPI module into a physics ground truth path and a lightweight ARX surrogate for each module. We defined a per-module residual by measuring the discrepancy between the ARX prediction and the physics output, and stacking them into a multivariate residual vector. We performed autocorrelation analysis that showed key residual channels especially plant/output are structured and temporally correlated which then justified the utilization of a regime-based model. We therefore fit a Gaussian HMM (via EM) to z-scored residuals to infer latent error regimes and computed smoothed posteriors. Next, we implemented a targeted intervention strategy that estimated a posterior-weighted and state-dependent bias using per-state residual means, transformed it back to physical units,

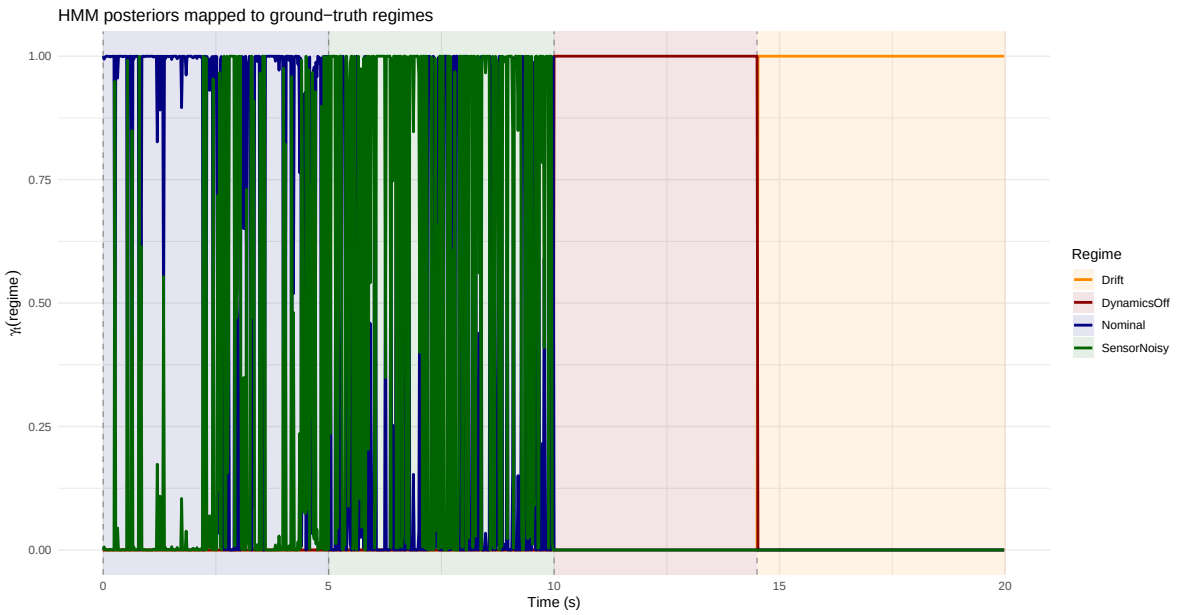

Figure 7: HMM regime identification under deliberate fault injection: smoothed posterior probabilities (Hungarian-mapped to ground-truth regimes) over time, with transitions aligning to the scheduled Nominal, Sensor-Noisy, Dynamics-Off, and Drift intervals.

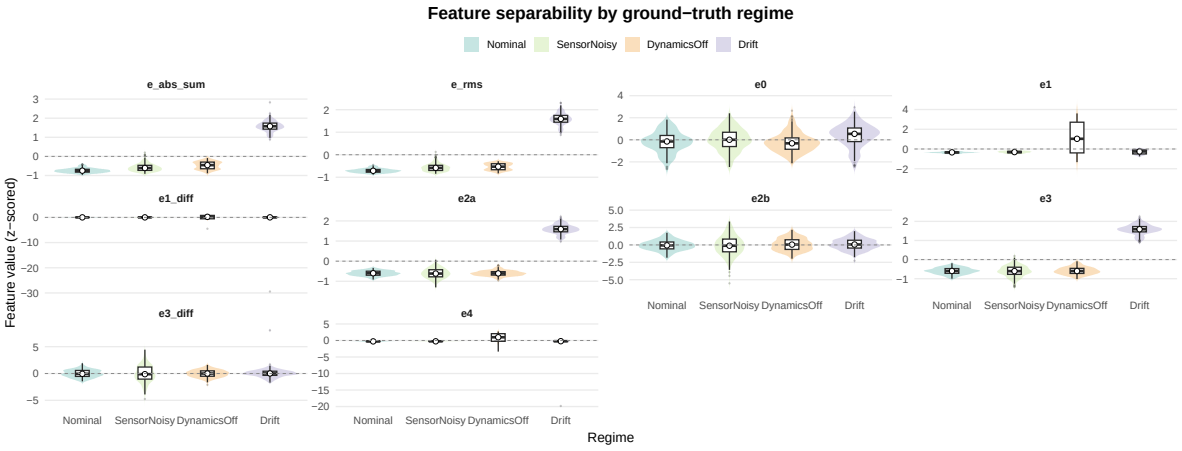

Figure 8: Feature separability by ground-truth regime under deliberate fault injection. Each panel shows boxplots of a z-scored residual-derived feature.

and applied smoothing/rate limiting before subtracting it from the surrogate outputs and recomputing downstream signals. This strategy reduced residual magnitudes over time and resulted in tighter residual distributions (closer to zero) and a lower aggregate residual norm across modules. Finally, we validated identification performance using a deliberate fault-injection schedule that created regime-specific residual signatures; because HMM state labels are arbitrary, we evaluated segmentation by mapping inferred states to ground-truth regimes using Viterbi decoding and Hungarian maximum-overlap matching which showed strong alignment with the scheduled regimes and highlighted where separability is weaker.

One of the key assumptions of our framework was that we considered the number of HMM latent states as $K = 4$. To assess the sensitivity of our conclusions to this choice, we performed a model-order selection

analysis on the residuals produced by the six-module digital twin pipeline. We computed the per-module residual vector $e_t = [e_t^{(0)}, e_t^{(1)}, e_t^{(2a)}, e_t^{(2b)}, e_t^{(3)}, e_t^{(4)}]^\top$ as the difference between the ARX surrogate predictions (trained on the first 50% of the data) and the physics-based ground truth, and standardized each residual channel (z-scored) prior to HMM fitting to place all channels on a comparable scale. We then fitted Gaussian-emission HMMs via EM for $K \in \{2, \ldots, 9\}$ and compared models using the Bayesian Information Criterion (BIC):

$$\text{BIC} = -2\ln(\hat{L}) + p\ln(n) \tag{26}$$

where $\hat{L}$ represents the maximized likelihood, p denotes the number of free parameters, and n is the number of time points. The BIC imposes a penalty for model complexity, thereby favoring models that explain the data structure most efficiently. Our results showed (as illustrated in Figure 9) that the BIC decreased over this range and attained its minimum at $K = 8$ with very close value at $K = 7$ (meaning our assumed $K = 4$ is not optimal for this example) thereby suggesting that the residual process exhibits richer multi-regime structure than can be captured by a small number of modes. In our future work, we will explore more adaptive and non-parametric models such as Hierarchical Dirichlet Process (HDP) HMM models where the number of latent states are automatically learned from the data and not hard-coded to increase the accuracy of our model (Teh et al., 2006).

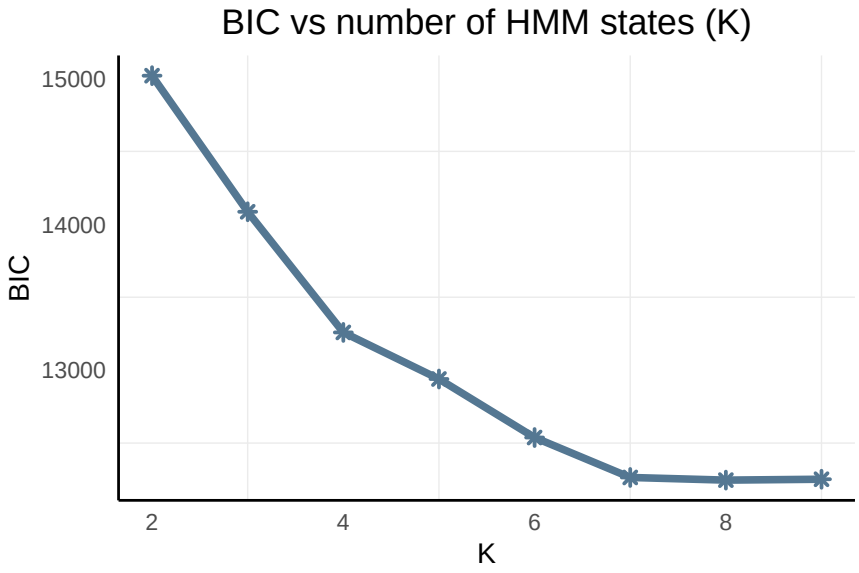

Figure 9: Model selection analysis showing Bayesian Information Criterion (BIC) as a function of the number of HMM states K. The plot shows a monotonic decrease up to $K = 8$ followed by plateauing which suggests the $K = 4$ scenario used in the previous sections is not the optimal number of states.

In this article, we performed targeted interventions by utilizing the smoothed posterior probabilities and applying a continuous bias correction derived from state-dependent residual means. However, we note that there may be other factors that may affect the choice of the error mitigation strategy such as costs. Consequently, for a complementary decision-oriented formulation that selects among discrete mitigation options, we implemented a Multi Criteria Decision Analysis (MCDA)-inspired action-selection layer in R using the TOPSIS (Hwang and Yoon, 1981) implementation in the `RMCDA` package (Najafi and Mirzaei, 2025). At each time step, we first constructed residual-derived features from the same multivariate residual vector, including instantaneous magnitudes, a rolling variance proxy for sensor noise computed via a sliding-window standard deviation, and a rolling mean proxy for drift computed via sliding-window averages. These features

were normalized (MAD-based scaling) to place criteria on comparable scales, and combined with regime relevance through the smoothed posteriors (when available) to form time-varying "suitability" scores for candidate actions (down-weight Sensor A, down-weight Sensor B, increase filtering, re-identify the plant surrogate, or apply bias correction). We then ranked actions using TOPSIS (Hwang and Yoon, 1981). Finally, to avoid chattering, we included a conservative gating rule that admitted a `NoAction` choice only under quiet, confidently nominal conditions (low residual magnitudes and high nominal posterior probability), and we applied a minimum dwell-time and score-margin hysteresis before allowing action switches. We have shown the optimal action chosen through MCDA over time in Figure 10.

Table 1: TOPSIS criteria used in the MCDA action-selection layer. All criteria are oriented so that larger values are preferred; cost is converted to a benefit criterion by negation prior to TOPSIS. Weights are normalized internally (they sum to 1 in our configuration).

Criterion	Weight	Definition / construction (from code)
KPI_gain	0.35	Posterior-weighted expected improvement in KPI consistency. Computed per action as a product of regime relevance (e.g., $p_{\text{SensorNoisy}}, p_{\text{DynamicsOff}}, p_{\text{Drift}}$), normalized KPI severity $kpi_n \propto \lvert e^{(4)}\rvert$, and action-relevant need indicators (e.g., $sensor_n$, $noise_n$, $plant_n$, $drift_n$), with action-specific modifiers (including dominant-sensor logic).
Module_gain	0.25	Posterior-weighted expected improvement at the affected module level (sensor/plant). Computed similarly to KPI_gain but excluding the explicit kpi_n factor, using normalized module severity indicators (e.g., $sensor_n \propto \lvert e^{(2a)}\rvert + \lvert e^{(2b)}\rvert$, $plant_n \propto \lvert e^{(1)}\rvert$) and regime posteriors.
Drift_gain	0.15	Posterior-weighted expected reduction in drift-related error. Uses normalized drift magnitude $drift_n$ derived from rolling means of $e^{(2a)}$, $e^{(0)}$, and $e^{(1)}$ (aggregated as $\lvert\overline{e^{(2a)}}\rvert + \lvert\overline{e^{(0)}}\rvert + \lvert\overline{e^{(1)}}\rvert$ over a sliding window), combined with regime posteriors and action modifiers.
Stability	0.10	Operational stability preference, implemented as a benefit criterion via a negative penalty: $Stability = -StabilityPenalty(action)$, where StabilityPenalty is a fixed, action-dependent constant (larger penalty implies less stability).
Cost	0.15	Operational cost preference. Cost is specified as a fixed, action-dependent constant and converted to benefit form by negation prior to TOPSIS (i.e., lower cost is preferred by maximizing $-Cost$).

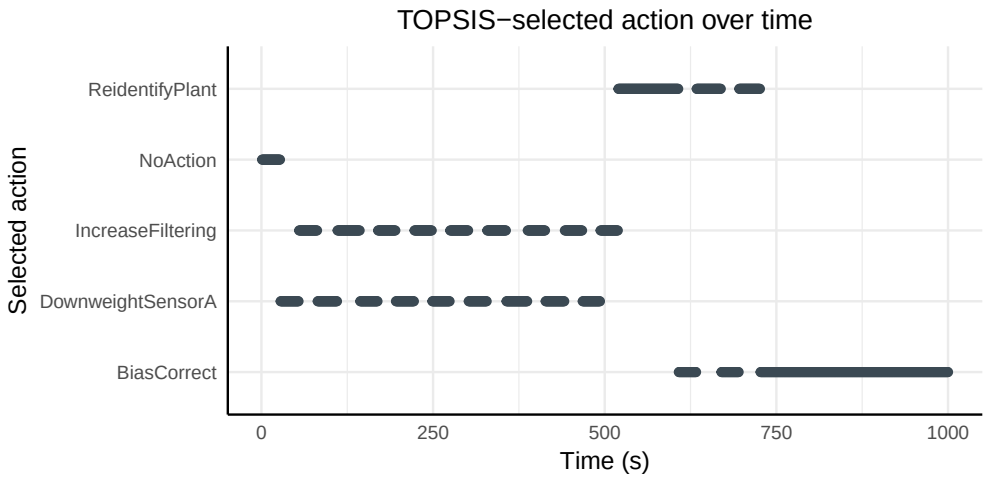

Figure 10: MCDA guided optimal actions over time.

We note that the MCDA approach has several limitations that motivate future refinement. First, the action suitability functions are heuristic and include fixed coefficients that are not estimated from data. Consequently, decision outcomes may be sensitive to these hyperparameters and should be subject to further scrutiny in the future (we chose a hypothetical example, and we fixed the coefficients in this model for demonstration purposes only). Second, some criteria used in TOPSIS are structurally correlated, which may unintentionally overweight particular residual channels or regime posteriors within the aggregation step. Third, the current scoring logic permits limited cross-regime coupling (e.g., small posterior contributions from non-target regimes) that may be reasonable in practice but requires explicit justification to avoid regime-action leakage. Fourth, the NoAction gate is based on distribution-relative thresholds (quantiles) rather than absolute performance tolerances, so quiet/nominal conditions are defined relative to the dataset rather than specification-based error bounds. We believe that incorporating absolute thresholds or task-level tolerances would make the gating policy more robust across operating conditions. Finally, because the TOPSIS matrix entries are products of posteriors and severity proxies, some timesteps can result in near-degenerate criterion columns (e.g., near-zero norms), which may lead to numerical instability and ad hoc fallback behavior. We suggest that future implementations include explicit degeneracy checks, principled fallbacks (e.g., rule-based baselines), and reporting of decision confidence.

To understand how robust the HMM-guided targeted intervention is, we performed a simulation-based sensitivity analysis in which we generated data from the six-module digital twin pipeline, trained ARX surrogates on a nominal slice, inferred latent error regimes with a Gaussian HMM, and lastly applied a posterior-weighted, time-varying bias correction to the ARX plant output. We then quantified how much this correction reduced the plant-position residual error and how that reduction changed as we varied sensing/fusion conditions.

For the sensing/fusion robustness analysis, we performed a factorial sweep over the fusion weight $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ and independent sensor-noise multipliers $(m_A, m_B) \in \{0.5, 1.0, 1.5\} \times \{0.5, 1.0, 1.5\}$. In each configuration, we reran the physics simulation, retrained the ARX surrogates on the first half of the run, refit the Gaussian-emission HMM to the z-scored residual channels, and then recomputed the correction signal by forming a posterior-weighted expected plant bias from the smoothed state posteriors and Viterbi-state plant residual means. The estimated bias was then converted back to physical units and conservatively shaped via exponential smoothing and a rate limiter (using fixed shaping parameters across the sweep) before being subtracted from the ARX plant position. Performance was summarized as the percent

change in plant residual RMSE:

$$\Delta\text{RMSE}_{e1}(\%) = 100 \cdot \frac{\text{RMSE}_{\text{pre}} - \text{RMSE}_{\text{post}}}{\text{RMSE}_{\text{pre}}}. \tag{27}$$

We show the results in Figure 11 as a faceted plot of ΔRMSE_{e1} versus α, with facet rows/columns corresponding to the Sensor A and Sensor B noise multipliers, respectively.

Across all 36 fusion/noise configurations, the intervention produced consistently positive error reductions, with ΔRMSE_{e1} remaining in the $\sim$ 23%-40% range which indicates that the HMM-guided correction is robust to substantial changes in measurement quality and fusion weighting. Sensitivity to α depended on the relative signal-to-noise ratio of the two sensors: when Sensor B was degraded, improvements generally increased with α (for example when placing greater weight on Sensor A in the fusion stage), whereas when Sensor B was cleaner, the dependence on α was weaker and in some cases slightly decreased with increasing α. The largest gains occurred in strongly asymmetric noise conditions (e.g., high Sensor A noise with low Sensor B noise), where ΔRMSE_{e1} reached the upper end of the observed band for moderate-to-large α. In more balanced cases (e.g., $m_A \approx m_B$), the improvement remained positive but clustered around the mid-20% to high-20% range, suggesting that the correction remains beneficial even when neither sensor provides a clearly superior measurement stream.

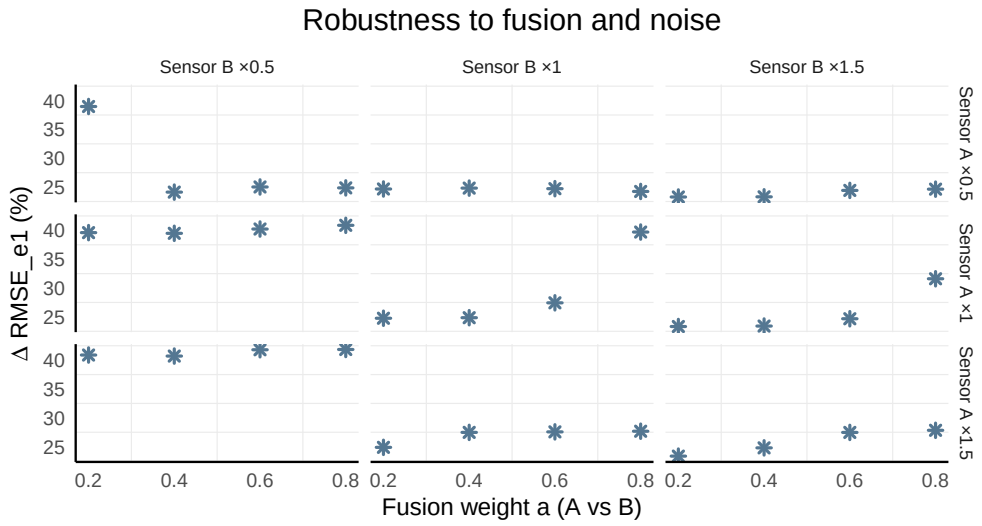

Figure 11: Sensitivity of plant residual improvement to fusion and measurement noise.

We note that these trends should be interpreted in the context of the full residual-driven inference loop. Varying α and the sensor noise levels modifies the multivariate residual structure presented to the HMM (including downstream fusion and KPI residual channels), which in turn can shift the inferred regime posteriors and the resulting bias estimate. Thus, the observed fusion-SNR alignment is indicative of measurement quality in the fusion stage, and also how changes in sensing and fusion influence regime inference and the posterior-weighted correction signal.

Lastly, in Figure 3 we showed that several residual channels are not white (i.e., they exhibit non-negligible autocorrelation), which as we mentioned motivated a regime-based model rather than purely instantaneous thresholding. This observation does not, however, preclude the use of a Gaussian-emission HMM. We note that "Gaussian" refers to the marginal distribution of residuals within a regime, whereas "white" refers to temporal independence. In fact, an HMM with conditionally independent Gaussian emissions can still

generate substantial autocorrelation in the observed residual sequence through persistent latent-state dynamics and state-dependent means/variances, result in piecewise-stationary segments and slow ACF decay. We therefore adopt the Gaussian-emission HMM as a baseline that provides stable smoothed posteriors for downstream mitigation. Nevertheless, if residuals retain strong temporal structure within inferred regimes (e.g., when computing state-conditional ACFs using Viterbi segments or posterior-weighted assignments), then the i.i.d. emission assumption is misspecified and a richer model would be warranted. A natural extension is to replace memoryless Gaussian emissions with state-dependent autoregressive dynamics (e.g., switching AR/AR-HMM or related SLDS formulations), which can explicitly capture within-regime oscillations and correlated noise while preserving the regime-inference and posterior-weighting framework used for intervention.

6. Conclusions

Here, we introduced a stochastic, module-level framework to understand and mitigate error propagation in digital twins by decomposing a six-module pipeline into a physics ground-truth path and lightweight ARX surrogates. By defining module residuals as surrogate-physics discrepancies and stacking them into a multivariate residual vector, we obtained a signature of where and how error propagates through the digital twin system. Through performing autocorrelation analysis of the residual channels we uncovered structured temporal behavior particularly in the plant and KPI residuals which justified the use of a regime-based stochastic model rather than static thresholds or purely instantaneous detectors.

We then devised and fit a Gaussian-emission Hidden Markov Model to standardized residual vectors to infer latent error regimes and compute smoothed posterior probabilities that quantify regime membership over time. Using these posteriors, we implemented a targeted intervention strategy that estimates a state-dependent bias (via per-state residual means), transforms it back to physical units, and shapes it using smoothing and rate limiting before subtracting it from the surrogate outputs and recomputing downstream quantities. Across our experiments, this posterior-weighted correction reduced residual magnitudes, tightened per-module residual distributions toward zero, and lowered the aggregate residual norm which in turn demonstrated that stochastic regime inference can be directly used for practical mitigation as opposed to only post hoc labeling.

Finally, we validated regime identification through a deliberate fault-injection schedule that produced distinct residual signatures for nominal, sensor-noisy, dynamics-off, and drift conditions. Despite the inherent label ambiguity of HMM states, Hungarian maximum-overlap matching of the Viterbi path to ground-truth regimes showed strong alignment with the injected schedule and clarified where separability is weaker. We also performed robustness analysis which further indicated that the HMM-guided correction maintains positive error reductions across variations in sensing quality and fusion weighting which in turn reinforced the applicability of the approach under heterogeneous operating conditions. Together, these results suggest that combining modular residual modeling with probabilistic regime inference would allow us to prevent error propagation through digital twin models.

Code and Data Availability

The scripts used to generate the results in this paper are available on Github: https://github.com/AnniceNajafi/ARX_HMM. All analyses were performed in R version 4.5.1. Figure 1 was created with Microsoft PowerPoint.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant Number 2514616. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Mahmoud Abdelrahman, Edgardo Macatulad, Binyu Lei, Matias Quintana, Clayton Miller, and Filip Biljecki. What is a digital twin anyway? deriving the definition for the built environment from over 15,000 scientific publications. *Building and Environment*, 274:112748, 2025.

Mohsen Attaran and Bilge Gokhan Celik. Digital twin: Benefits, use cases, challenges, and opportunities. *Decision Analytics Journal*, 6:100165, 2023.

Peng Cao, Jian Liu, Honglin Wang, Yu Wang, Xue Liu, and Dong Wang. Uncertainty analysis of digital twin model of mine ventilation system. *Scientific Reports*, 14(1):26558, 2024.

Mario Reyes Garcia, Jesus Castillo, and Afroza Shirin. Test and evaluation of ai/ml enhanced digital twin. *Systems*, 13(8):656, 2025.

Taylor R Grimm, Amos Branch, Kyle A Thompson, Andrew Salveson, John Zhao, Darrell Johnson, Amanda S Hering, and Kathryn B Newhart. Long-term statistical process monitoring of an ultrafiltration water treatment process. *ACS Es&t Engineering*, 4(6):1492–1506, 2024.

A Hartwell, F Montana, W Jacobs, V Kadirkamanathan, N Ameri, and AR Mills. Distributed digital twins for health monitoring: resource constrained aero-engine fleet management. *The Aeronautical Journal*, 128 (1325):1556–1575, 2024.

Matthias Haslbeck and Thomas Braml. The role of uncertainty propagation for digital twins. In *International Probabilistic Workshop*, pages 303–312. Springer, 2024.

Deqiang He, Jiayang Zhao, Zhenzhen Jin, Chenggeng Huang, Fan Zhang, and Jinxin Wu. Prediction of bearing remaining useful life based on a two-stage updated digital twin. *Advanced Engineering Informatics*, 65: 103123, 2025.

Martin Hiller, Arshad Jhumka, and Neeraj Suri. An approach for analysing the propagation of data errors in software. In *2001 International Conference on Dependable Systems and Networks*, pages 161–170. IEEE, 2001.

Kurt Hornik. A clue for cluster ensembles. *Journal of Statistical Software*, 14:1–25, 2005.

Ching-Lai Hwang and Kwangsun Yoon. *Multiple Attribute Decision Making: Methods and Applications: A State-of-the-Art Survey*, volume 186 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1981. doi: 10.1007/978-3-642-48318-9.

Zhongcheng Lei, Hong Zhou, Xiaoran Dai, Wenshan Hu, and Guo-Ping Liu. Digital twin based monitoring and control for dc-dc converters. *Nature Communications*, 14(1):5604, 2023.

Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

Annice Najafi and Shokoufeh Mirzaei. Rmcda: The comprehensive r library for applying multi-criteria decision analysis methods. *Software Impacts*, page 100762, 2025.

Annice Najafi, Shokoufeh Mirzaei, and Zahra Sotoudeh. Stochastic framework for analyzing error propagation in digital twins: a two-link robotic arm case. In *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2025. in press.

Y. J. Park, S. K. S. Fan, and C. Y. Hsu. A review on fault detection and process diagnostics in industrial processes. *Processes*, 8(9):1123, 2020. doi: 10.3390/pr8091123.

Mahsa Salehi and Lida Rashidi. A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]. *ACM SIGKDD Explorations Newsletter*, 20(1):13–23, 2018.

Ali Sayghe. Digital twin-driven intrusion detection for industrial scada: A cyber-physical case study. *Sensors*, 25(16):4963, 2025.

Chanan Singh and Joydeep Mitra. Monte carlo simulation for reliability analysis of emergency and standby power systems. In *IAS'95. Conference Record of the 1995 IEEE Industry Applications Conference Thirtieth IAS Annual Meeting*, volume 3, pages 2290–2295. IEEE, 1995.

Maulshree Singh, Evert Fuenmayor, Eoin P Hinchy, Yuansong Qiao, Niall Murray, and Declan Devine. Digital twin: Origin to future. *Applied System Innovation*, 4(2):36, 2021.

Yifan Tang, Mostafa Rahmani Dehaghani, and G Gary Wang. Capturing lifecycle system degradation in digital twin model updating. *arXiv preprint arXiv:2503.08953*, 2025.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006.

Trung-Thanh Thai, Phuc Do, Benoit Iung, and Paolo Gardoni. Impact of sensor degradation on measurement uncertainty in prognostics and maintenance decision-making: State of the art and challenges. *International Journal of Prognostics and Health Management*, 16(4), 2025.

Ingmar Visser and Maarten Speekenbrink. depmixs4: an r package for hidden markov models. *Journal of statistical Software*, 36:1–21, 2010.