Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

DYNAMIC CALIBRATION FRAMEWORK FOR DIGITAL TWINS USING ACTIVE LEARNING AND CONFORMAL PREDICTION

Özge Sürer[1], Xi Chen[2], and Sara Shashaani[3]

[1]Department of Information Systems and Analytics, Miami University, Oxford, OH, USA
[2]Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA, USA
[3]Fitts Dept. of Industrial and Systems Eng., North Carolina State University, Raleigh, NC, USA

ABSTRACT

This work presents an adaptive framework for dynamically calibrating digital twins (DTs) in response to evolving real-world (RW) conditions. Traditional simulation-based models often rely on fixed parameter estimates, limiting their adaptability over time. To address this, our approach integrates active learning (AL) with a dynamic calibration mechanism that keeps the DT aligned with RW observations. At each time step, a new data batch is received, and a conformal prediction-based monitoring system assesses whether recalibration is needed. When a change in the RW system state is detected, DT parameters are updated using an efficient AL strategy. The framework reduces computational overhead by avoiding unnecessary DT evaluations while maintaining accurate system representation. We demonstrate the effectiveness of the proposed approach in achieving adaptive, cost-efficient DT calibration over time.

1 INTRODUCTION

Digital twins (DTs) are powerful tools that create virtual representations of real-world (RW) systems, enabling users to monitor, analyze, and optimize system performance. Unlike traditional simulation models, DTs require continuous updates to remain aligned with RW observations. For instance, in epidemiological modeling, DT parameters such as transmission or contact rates often evolve with the RW system state. For example, a rise in infection counts may trigger behavioral changes or policy interventions (e.g., increased social distancing), which in turn alter the effective reproduction number. As such, the DT must recalibrate its parameters dynamically to maintain alignment with the evolving epidemic trajectory. Conventional calibration approaches typically infer unknown parameters to match model outputs with RW data (Sung and Tuo 2024), but they are often applied as a one-time process and do not support ongoing updates. This work proposes an adaptive framework that continuously calibrates a simulation-based DT, allowing it to remain responsive to evolving RW conditions.

Running a DT model is often computationally expensive due to its reliance on high-fidelity physical simulations. To reduce this cost, traditional one-time calibration methods often use emulators—statistical or machine learning (ML) surrogates that approximate DT behavior with lower computational overhead. In continuous calibration, where the DT must regularly adapt to RW changes, minimizing DT evaluations becomes even more critical. Efficient data collection is thus vital, especially in real-time settings. We use Gaussian processes (GPs; Rasmussen and Williams 2005)—a popular class of emulators—to approximate DT outputs. Trained on simulation results at selected inputs, GPs provide predictive distributions at new inputs. We propose an active learning (AL) strategy that adaptively selects inputs for evaluating the DT over time, improving calibration accuracy while reducing the number of DT evaluations.

Over the past decade, conformal prediction (CP) has gained substantial traction within the ML community due to its ability to provide distribution-free uncertainty quantification (UQ) for both classification and regression tasks (Angelopoulos and Bates 2023). CP enables the construction of prediction intervals that, with a user-specified probability guarantee, contain the true output at an unseen input—regardless of the

underlying ML model. In this work, we use CP to generate prediction intervals for RW observations based on a GP emulator. In the context of DTs, which require dynamic calibration to remain aligned with the evolving RW system, CP intervals serve a dual purpose. They not only quantify prediction uncertainty but also function as a diagnostic tool for detecting when the current DT parameters may no longer be valid. To this end, we propose a novel monitoring system that combines CP with a cumulative sum (CUSUM) control chart (Page 1954) to track deviations between observed RW data and predicted values. When systematic deviations exceed a predefined threshold, an alarm is triggered, indicating the need for DT recalibration. This mechanism enables timely, data-driven updates, ensuring the DT continues to accurately represent the RW system dynamics over time.

The remainder of this article is organized as follows. Section 2 formulates the problem and provides an overview of the dynamic calibration framework. Section 3 introduces the AL strategy used to collect DT data and estimate the calibration parameters at each time step. Section 4 details the CP approach for constructing prediction intervals and detecting changes in the state of the RW system. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

2 OVERVIEW

2.1 Problem Formulation

Let $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^q$ denote the design input and $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$ the generic calibration parameter of the DT model. The DT is represented as a deterministic function $\eta(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}$, mapping each input pair $(\mathbf{x}, \boldsymbol{\theta})$ to a scalar output. Denote the true but unknown calibration parameter by $\boldsymbol{\theta}^* \in \Theta$, which best aligns the DT output with RW observations. The design inputs at which the RW observations are collected are denoted by $\mathbf{x}^r \in \mathcal{X}$. Following the conventional simulation model calibration literature, we assume that the observed RW system responses are generated according to the following mechanism:

$$y\left(\mathbf{x}^r\right) = \eta\left(\mathbf{x}^r, \boldsymbol{\theta}^*\right) + \varepsilon, \quad \text{and} \quad \varepsilon \sim \mathcal{N}\left(0, \sigma^2\left(\mathbf{x}^r\right)\right),$$

where the term ε represents observational noise, assumed to follow a normal distribution with mean zero and variance $\sigma^2(\mathbf{x}^r)$.

A key distinction between a DT and a traditional simulation model is that, unlike standard practices, the physical system continuously provides RW data. As a result, the DT must be continuously updated to reflect RW changes, enabling adaptive predictions and decision-making. To capture this evolution, we introduce a time index t to denote when new RW data becomes available. The calibration parameter $\boldsymbol{\theta}_t^* \in \Theta$ then represents the unknown parameter vector that best characterizes the RW system at time t; that is, it encapsulates the state of the RW at time t. Let $\hat{\boldsymbol{\theta}}_t^*$ denote the estimated calibration parameter at time t based on available data. To incorporate this temporal dependency, we refine the RW data generation mechanism as follows:

$$y_t\left(\mathbf{x}_i^r\right) = \eta\left(\mathbf{x}_i^r, \boldsymbol{\theta}_t^*\right) + \varepsilon_{i,t}, \quad \text{and} \quad \varepsilon_{i,t} \sim \mathcal{N}\left(0, \sigma_t^2\left(\mathbf{x}_i^r\right)\right) \ , \ i = 1, 2, \ldots, b^r \tag{1}$$

where we assume that at each time step t, RW system responses are observed at b^r different design points, denoted as $y\left(\mathbf{x}_1^r\right), y\left(\mathbf{x}_2^r\right), \ldots, y\left(\mathbf{x}_{b^r}^r\right)$. We also assume that, given the true system state $\boldsymbol{\theta}_t^*$, the observed pairs $\left(\mathbf{x}_i^r, y_t\left(\mathbf{x}_i^r\right)\right)$ are independent, implying that the noise terms $\varepsilon_{i,t}$ are independently distributed across both time steps and design points. The formulation in (1) explicitly captures temporal variations in both the calibration parameter vector and the observation noise, distinguishing our approach from static calibration methods and more accurately reflecting the dynamic nature of the RW system.

2.2 Framework for Dynamic Calibration of Digital Twins

At each time step t for $t = 1, 2, \ldots, T$, a batch of $b^r \geq 1$ RW data points is received. Let $\mathbf{x}_{t,i}^r$, for $i = 1, 2, \ldots, b^r$, denote the design input points at which RW data are observed at time t. While our

formulation allows $\mathbf{x}^r_{t,i}$ to vary within a time step, it also accommodates repeated observations at fixed design inputs. Denote the streaming dataset at time t as $\mathcal{D}_t = \{(\mathbf{x}^r_{t,i}, y_t(\mathbf{x}^r_{t,i})) : i = 1, 2, \ldots, b^r\}$. A key challenge is that as new data arrive, the state of the system that the DT intends to predict may shift. This potential change requires updating the estimated calibration parameter vector over time. Algorithm 1 outlines the proposed framework for dynamically calibrating the DT.

Algorithm 1: Dynamic calibration framework.

1 **Input:** The initial batch of RW data, $\mathcal{D}_0 = \{(\mathbf{x}^r_{0,i}, y_0(\mathbf{x}^r_{0,i})) : i = 1, 2, \ldots, b^r\}$
2 *Obtain* the parameter estimate $\hat{\boldsymbol{\theta}}^*_0$ and the emulator $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_0)$ via Algorithm 2
3 **for** $t = 1, 2, \ldots, T$ **do**
4 *Receive* $\mathcal{D}_t = \{(\mathbf{x}^r_{t,i}, y_t(\mathbf{x}^r_{t,i})) : i = 1, 2, \ldots, b^r\}$
5 **if** `stateChanged` *(detected via Algorithm 3)* **then**
6 *Obtain* the parameter estimate $\hat{\boldsymbol{\theta}}^*_t$ and the emulator $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_t)$ via Algorithm 2
7 **else**
8 *Set* $\hat{\boldsymbol{\theta}}^*_t \leftarrow \hat{\boldsymbol{\theta}}^*_{t-1}$, $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_t) \leftarrow (\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_{t-1})$
9 **Output:** $\hat{\boldsymbol{\theta}}^*_t$ and $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_t)$ for $t = 1, 2, \ldots, T$

Our framework consists of two main components that enable dynamic information exchange between the DT and the RW system. The first component updates the estimate of the calibration parameter vector using observed RW data, while the second monitors potential changes in the RW system state by comparing observations against the calibrated DT. This structure allows the DT to remain aligned with evolving RW system behaviors. While methods such as Bayesian optimization (BO) can also be used to estimate calibration parameters (Jeon and Shashaani 2024), they typically treat calibration and monitoring as separate objectives. For example, BO would require one GP emulator to identify optimal parameters and a separate GP to support monitoring, leading to a split treatment of estimation and change detection. In contrast, our framework uses a single GP to support both tasks, enabling the two components to share information and provide mutual feedback throughout the sequential process.

We begin with the initial batch of RW data, $\mathcal{D}_0 = \{(\mathbf{x}^r_{0,i}, y_0(\mathbf{x}^r_{0,i})) : i = 1, \ldots, b^r\}$ (see line 1 in Algorithm 1). An initial parameter estimate $\hat{\boldsymbol{\theta}}^*_0$ is obtained using $\mathcal{D}_0$ and the DT via Algorithm 2 (see Section 3). Since DT evaluations are typically grounded in complex physics and can be computationally expensive, it is essential to minimize the number of such evaluations required during parameter estimation. To this end, we employ an AL strategy that efficiently selects informative design and calibration parameter input points to run the DT. We use subscripted $\mathbf{x}$ and $\boldsymbol{\theta}$ (e.g., $\mathbf{x}_j$, $\boldsymbol{\theta}_j$) to denote inputs selected by the AL procedure. This procedure adaptively queries the DT to generate a dataset of size $n_1 + b^s$, that is, $\mathcal{S}_0 = \{((\mathbf{x}_j, \boldsymbol{\theta}_j), \eta(\mathbf{x}_j, \boldsymbol{\theta}_j)) : j = 1, 2, \ldots, n_1 + b^s\}$, where n_1 is the size of the initial sample and b^s denotes the number of adaptively acquired samples. The output of Algorithm 2 includes not only the parameter estimate $\hat{\boldsymbol{\theta}}^*_0$ but also a GP emulator, $\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_0$, constructed through targeted sampling. This surrogate model enables fast and accurate predictions of the DT output, acting as a proxy for real-time RW inference and the detection of system state changes.

At each time step $t \geq 1$, we receive a batch of b^r new RW data points, denoted by $\mathcal{D}_t$ (line 4 of Algorithm 1). While new data arrive continuously, the RW system state does not necessarily change meaningfully at every step. Therefore, recalibrating the DT at each time point would be inefficient and often unnecessary. As a result, we incorporate a monitoring mechanism to evaluate the current DT model's adequacy over time. Specifically, we employ CP (described in Section 4) to construct prediction intervals for RW observations and use a CUSUM-based method to detect potential shifts in the RW system state. This combination enables us to quantify the uncertainty in predictions and promptly detect when the DT no longer aligns with reality to the extent that recalibration is needed. If the CP approach detects a meaningful

state change (line 5 of Algorithm 1), recalibration of the DT is triggered. This involves obtaining an updated parameter vector estimate $\hat{\boldsymbol{\theta}}_t^*$ and constructing a new emulator $\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_t$ using the DT dataset, $\mathcal{S}_t$, collected through the proposed AL strategy (line 6 of Algorithm 1). Otherwise, the previously estimated parameter vector is retained, and the current emulator remains unchanged (line 8 of Algorithm 1), thereby avoiding unnecessary DT evaluations and reducing computational costs.

3 ADAPTIVE DIGITAL TWIN DATA ACQUISITION VIA ACTIVE LEARNING

Suppose that we receive the RW dataset, $\mathcal{D}_t$, at time t, and our monitoring tool (discussed in Section 4) indicates that the DT, using the previously estimated parameter vector $\hat{\boldsymbol{\theta}}_{t-1}^*$, no longer adequately represents the current RW system. In this case, we initiate a recalibration process using AL to efficiently update the calibration parameter estimate at time t by sequentially selecting the most informative input points for evaluating the DT.

AL leverages acquisition functions to quantify the utility of unseen input locations, prioritizing regions in the input space that are expected to maximize the information gain. While AL has been successfully applied to construct globally accurate emulators for simulation models (Gramacy (2020), Chapter 6), these strategies are not directly applicable to calibration tasks. Specifically, evaluating the simulation model in regions far from the calibration-relevant space can result in unnecessary computational costs. Furthermore, as the input-space dimensionality increases, the calibration region becomes proportionally smaller, making it even less likely that traditional AL strategies will effectively target the area of interest.

Recent work has demonstrated the potential of tailored AL strategies for calibration. For instance, Sürer et al. (2024), Sürer (2025), and Koermer et al. (2024) propose approaches that specifically target parameter estimation and RW system response prediction. While Sürer et al. (2024) and Sürer (2025) develop acquisition functions aimed at learning the posterior distribution of the calibration parameter vector, Koermer et al. (2024) focus on improving RW response predictions within the Kennedy and O'Hagan (KOH) framework (Kennedy and O'Hagan 2001). However, these approaches are designed for calibration with a fixed RW dataset and typically treat the calibration process as a one-time, offline task.

Algorithm 2 outlines the AL procedure, which serves two primary objectives: (1) estimating the calibration parameter vector $\boldsymbol{\theta}_t^*$, and (2) constructing an emulator that accurately predicts the mean of the RW response $y_t(\mathbf{x}^r)$ for any design input $\mathbf{x}^r \in \mathcal{X}$ at time t. This emulator plays a central role in the monitoring mechanism by facilitating the detection of system-state changes and enabling timely recalibration and adaptation—topics that will be discussed in Section 4.

Algorithm 2: Active learning for digital twins (at time t).

1 **Input:** The initial batch of DT data, $\mathcal{S}_{t,1} = \{((\mathbf{x}_i, \boldsymbol{\theta}_i), \eta(\mathbf{x}_i, \boldsymbol{\theta}_i)) : i = 1, 2, \ldots, n_1\}$ for $n_1 \in \mathbb{Z}_+$
2 *Build* an emulator $(\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_{t,1})$
3 **for** $\ell = 1, 2, \ldots, b^s$ **do**
4 *Find* $\hat{\boldsymbol{\theta}}_{t,\ell}^*$ via (4)
5 *Acquire* $(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell}) \leftarrow \underset{(\mathbf{x},\boldsymbol{\theta}) \in \mathcal{X} \times \Theta}{\arg\min}\ \mathrm{IMSE}_\ell(\mathbf{x}, \boldsymbol{\theta})$, where $\mathrm{IMSE}_\ell(\cdot, \cdot)$ is defined in (5)
6 *Evaluate* the digital twin at $(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell})$ to obtain $\eta(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell})$
7 *Set* $\mathcal{S}_{t,\ell+1} \leftarrow \mathcal{S}_{t,\ell} \cup ((\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell}), \eta(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell}))$
8 *Build* an emulator $\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_{t,\ell+1}$
9 $\hat{\boldsymbol{\theta}}_t^* \leftarrow \hat{\boldsymbol{\theta}}_{t,b^s}^*$, $\mathcal{S}_t \leftarrow \mathcal{S}_{t,b^s+1}$
10 **Output:** Emulator $(\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_t)$ and $\hat{\boldsymbol{\theta}}_t^*$

At each time step t, we implement Algorithm 2, which comprises b^s stages indexed by ℓ. In each stage, a new input vector for running the DT is acquired, and its corresponding output is obtained. The procedure begins

by generating an initial DT simulation dataset of size n_1 to construct a preliminary emulator (lines 1–2 of Algorithm 2). The initial sample can be generated using space-filling design strategies—such as Latin hypercube sampling—or by uniformly drawing points from the input space. These techniques promote broad coverage of the input space and are well-established in the design of computer experiments; see Santner et al. (2018) for a comprehensive overview. Let $\mathcal{S}_{t,\ell} = \{((\mathbf{x}_i, \boldsymbol{\theta}_i), \eta(\mathbf{x}_i, \boldsymbol{\theta}_i)) : i = 1, 2, \ldots, n_\ell\}$ denote the DT simulation dataset collected up to stage ℓ of Algorithm 2 during time step t. The full set of observed DT outputs up to stage ℓ is denoted by $\boldsymbol{\eta}_\ell = (\eta(\mathbf{x}_1, \boldsymbol{\theta}_1), \eta(\mathbf{x}_2, \boldsymbol{\theta}_2), \ldots, \eta(\mathbf{x}_{n_\ell}, \boldsymbol{\theta}_{n_\ell}))^\top$. Under the GP prior, the joint distribution of the observed DT outputs in $\boldsymbol{\eta}_\ell$ and the unknown DT output $\eta(\mathbf{x}, \boldsymbol{\theta})$ at an unseen input point $(\mathbf{x}, \boldsymbol{\theta})$ follows a multivariate normal (MVN) distribution:

$$\begin{bmatrix} \boldsymbol{\eta}_\ell \\ \eta(\mathbf{x}, \boldsymbol{\theta}) \end{bmatrix} \sim \mathcal{MVN}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_\ell & \mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta}) \\ \mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta})^\top & k_\ell((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}, \boldsymbol{\theta})) \end{bmatrix}\right), \tag{2}$$

where $\mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta}) := \left(k_\ell\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}_1, \boldsymbol{\theta}_1)\right), \ldots, k_t\left((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}_{n_\ell}, \boldsymbol{\theta}_{n_\ell})\right)\right)^\top$ represents the $n_\ell \times 1$ vector of cross-kernel evaluations between the input point $(\mathbf{x}, \boldsymbol{\theta})$ and each observed training point $(\mathbf{x}_i, \boldsymbol{\theta}_i)$, $i = 1, 2, \ldots, n_\ell$. While we adopt a Gaussian kernel in our implementation, the approach is kernel-agnostic and can accommodate alternative choices. The matrix $\mathbf{K}_\ell$ has dimensions $n_\ell \times n_\ell$, with entries given by $k_\ell((\mathbf{x}_i, \boldsymbol{\theta}_i), (\mathbf{x}_j, \boldsymbol{\theta}_j)) + \upsilon\delta_{ij}$, where $\upsilon > 0$ is the nugget parameter, and δ_{ij} is the Kronecker delta function, which equals 1 if $i = j$ and 0 otherwise. The nugget parameter υ helps ensure numerical stability by enforcing positive definiteness. Conditioning the joint GP prior (2) on $\mathcal{S}_{t,\ell}$ yields the predictive distribution $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_{t,\ell})$, which is normal with mean $m_\ell(\mathbf{x}, \boldsymbol{\theta})$ and variance $s^2_\ell(\mathbf{x}, \boldsymbol{\theta})$, i.e., $(\eta(\mathbf{x}, \boldsymbol{\theta})|\mathcal{S}_{t,\ell}) \sim \mathcal{N}\left(m_\ell(\mathbf{x}, \boldsymbol{\theta}), s^2_\ell(\mathbf{x}, \boldsymbol{\theta})\right)$, where

$$m_\ell(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta})^\top \mathbf{K}_\ell^{-1} \boldsymbol{\eta}_\ell \text{ and } s^2_\ell(\mathbf{x}, \boldsymbol{\theta}) = k_\ell((\mathbf{x}, \boldsymbol{\theta}), (\mathbf{x}, \boldsymbol{\theta})) - \mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta})^\top \mathbf{K}_\ell^{-1} \mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta}) \,. \tag{3}$$

At each stage ℓ, we refine the parameter vector estimate $\hat{\boldsymbol{\theta}}^*_{t,\ell}$ to align the emulator's predictions with the RW system (line 4 of Algorithm 2). By iteratively updating $\hat{\boldsymbol{\theta}}^*_{t,\ell}$ through targeted DT data collection, we enhance the emulator's accuracy for RW response predictions at time t. This is achieved by minimizing the sum of squared differences between the observed RW responses and the emulator's predictive mean:

$$\hat{\boldsymbol{\theta}}^*_{t,\ell} = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \sum_{j=1}^{\tau_t} \sum_{i=1}^{b^r} \left(y_{t-j+1}\left(\mathbf{x}^r_{t-j+1,i}\right) - m_l\left(\mathbf{x}^r_{t-j+1,i}, \boldsymbol{\theta}\right)\right)^2, \tag{4}$$

where τ_t denotes the detection delay—the time elapsed between the actual change in the RW system's state and the moment the monitoring mechanism raises an alarm. The method for identifying τ_t is described in Section 4. We solve the nonlinear optimization program in (4) using standard off-the-shelf optimizers. At stage ℓ, the emulator and the parameter vector estimate $\hat{\boldsymbol{\theta}}^*_{t,\ell}$ are used to provide a predictive distribution for the RW response $y_t(\mathbf{x}^r)$ at any design input $\mathbf{x}^r \in \mathcal{X}$. Due to the data generation model in (1), the predictive distribution of the RW observation at design point $\mathbf{x}^r$ is normal with mean and variance respectively given by $\mathbb{E}\left[y_t(\mathbf{x}^r)\right] = m_\ell\left(\mathbf{x}^r, \hat{\boldsymbol{\theta}}^*_{t,\ell}\right)$ and $\mathbb{V}\left[y_t(\mathbf{x}^r)\right] = s^2_\ell\left(\mathbf{x}^r, \hat{\boldsymbol{\theta}}^*_{t,\ell}\right) + \sigma^2_t(\mathbf{x}^r)$, that is, $y_t(\mathbf{x}^r) \sim \mathcal{N}\left(m_\ell\left(\mathbf{x}^r, \hat{\boldsymbol{\theta}}^*_{t,\ell}\right), s^2_\ell\left(\mathbf{x}^r, \hat{\boldsymbol{\theta}}^*_{t,\ell}\right) + \sigma^2_t(\mathbf{x}^r)\right)$.

After updating the parameter vector estimate $\hat{\boldsymbol{\theta}}^*_{t,\ell}$, we select a new input $(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell})$ to evaluate DT and obtain the corresponding output $\eta(\mathbf{x}_{n_1+\ell}, \boldsymbol{\theta}_{n_1+\ell})$. This new data point is then added to the current DT dataset $\mathcal{S}_{t,\ell}$ (lines 6–7 of Algorithm 2). To guide this selection, we use the integrated mean squared error (IMSE) of RW predictions as our acquisition function (line 5 of Algorithm 2). At each stage ℓ, we identify the input that, when added to the emulator's training set, most effectively reduces the overall prediction uncertainty. Specifically, the IMSE for a candidate input point $(\mathbf{x}, \boldsymbol{\theta})$ is defined as:

$$\mathrm{IMSE}_\ell(\mathbf{x}, \boldsymbol{\theta}) = \int_{\mathbf{x}' \in \mathcal{X}} s^2_{\ell,(\mathbf{x},\boldsymbol{\theta})}\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right) d\mathbf{x}' \,, \tag{5}$$

where $s^2_{\ell,(\mathbf{x},\boldsymbol{\theta})}\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right)$ denotes the predictive variance of the emulator at $\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right)$ after adding the candidate input point $(\mathbf{x}, \boldsymbol{\theta})$ and its unknown output $\eta(\mathbf{x}, \boldsymbol{\theta})$ to the current DT dataset. With fixed GP hyperparameters, the covariance matrix $\mathbf{K}_\ell$ is updated by adding a new row and column associated with the candidate input point $(\mathbf{x}, \boldsymbol{\theta})$, and its inverse can be efficiently computed via a rank-one update. Substituting the matrix inverse into the expression for the emulator's predictive variance in (3) yields

$$s^2_{\ell,(\mathbf{x},\boldsymbol{\theta})}\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right) = s^2_\ell\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right) + \frac{\mathrm{cov}_\ell\left((\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}), (\mathbf{x}, \boldsymbol{\theta})\right)^2}{s^2_\ell(\mathbf{x}, \boldsymbol{\theta}) + \upsilon},$$

where $\mathrm{cov}_\ell\left((\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}), (\mathbf{x}, \boldsymbol{\theta})\right) = k_\ell\left((\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}), (\mathbf{x}, \boldsymbol{\theta})\right) - \mathbf{k}_\ell\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right)\mathbf{K}_\ell^{-1}\mathbf{k}_\ell(\mathbf{x}, \boldsymbol{\theta})$. Notice that the predictive variance $s^2_{\ell,(\mathbf{x},\boldsymbol{\theta})}\left(\mathbf{x}', \hat{\boldsymbol{\theta}}^*_{t,\ell}\right)$ remains unaffected by the hypothetical DT output $\eta(\mathbf{x}, \boldsymbol{\theta})$.

In summary, at each time step t, Algorithm 2 is executed to construct the DT dataset $\mathcal{S}_{t,\ell}$ from scratch. After acquiring a sample of $n_1 + b^s$ DT data points, the parameter vector estimate and the emulator at the final stage are returned (lines 9–10 of Algorithm 2). Rather than generating a new space-filling design at each time step t, we reuse the same initial sample across all time steps; that is, $\mathcal{S}_{1,1} = \mathcal{S}_{2,1} = \cdots = \mathcal{S}_{T,1}$.

4 DETECTION OF REAL-WORLD SYSTEM STATE CHANGES VIA CONFORMAL PREDICTION

Algorithm 3: Detecting the need for DT recalibration using conformal prediction (at time t).

1 **Input:** The past τ_{t-1} RW datasets $\bigcup_{t'=t-\tau_{t-1}}^{t-1} \mathcal{D}_{t'}$ (training), the new RW dataset $\mathcal{D}_t$ (validation), the emulator $\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_{t-1}$, the parameter vector estimate $\hat{\boldsymbol{\theta}}^*_{t-1}$, the miscoverage level α_t, $\alpha \in (0, 1)$, and $\alpha_{t,1} \leftarrow \alpha_t$
2 *Obtain* conformal interval $\hat{C}_t(\mathbf{x}^r_{t,i})$ via (6), and update $\alpha_{t,i}$ via (7), for $i = 1, 2, \ldots, b^r$
3 *Compute* the cumulative sum $CS_{t'}$, for $t' = 1, 2, \ldots, t$, iteratively via (8)
4 **if** $CS_t > \bar{c} + 2s_{\bar{c}}$ *(Computed via* (9)*)* **then**
5 `stateChanged` $\leftarrow$ **true**
6 *Set* $\alpha_{t+1} \leftarrow \alpha$; $\tau_t \leftarrow$ length of the most recent upward excursion of $CS_{t'}$
7 **else**
8 `stateChanged` $\leftarrow$ **false**
9 *Update* $\alpha_{t+1} \leftarrow \alpha_{t,b^r+1}$; $\tau_t \leftarrow \tau_{t-1} + 1$
10 **Output:** `stateChanged`, α_{t+1}, and τ_t

We use CP to construct high-probability, distribution-free prediction intervals for the true RW system response at a new design input point. CP guarantees marginal coverage under the assumption of data exchangeability, whereas GPs yield predictive uncertainty estimates contingent on strong modeling assumptions. To enhance robustness, CP has recently been integrated with GP modeling (Papadopoulos 2024; Jaber et al. 2025). However, ensuring data exchangeability becomes challenging in streaming settings, where data arrive sequentially and the underlying distribution may evolve. Recent advances in adaptive conformal inference address this issue by accommodating temporal distribution shifts (Gibbs and Candès 2021). In parallel, conformalized GP methods that account for such distribution shifts have been investigated by Xu et al. (2025).

Full conformal and split CP are two foundational methods for constructing prediction intervals (Angelopoulos and Bates 2023). While split CP offers greater computational efficiency by requiring only a single model fit, it typically sacrifices statistical efficiency relative to full CP approaches. In contrast, full

CP achieves higher statistical efficiency by leveraging the available data more effectively, but it incurs substantially higher computational costs due to repeated model training. Between these extremes, methods such as cross-CP (Vovk 2015) and CV+/Jackknife+ (Barber et al. 2021) provide a practical trade-off, minimizing the number of model fits while using the entire dataset for both training and calibration. In this work, we integrate Jackknife+ into our dynamic calibration framework to construct CP intervals.

At time step t, we receive a new RW dataset, $\mathcal{D}_t = \left\{ \left(\mathbf{x}^r_{t,i}, y_t(\mathbf{x}^r_{t,i}) \right), i = 1, 2, \ldots, b^r \right\}$. We aim to assess whether the RW system's state has changed enough to necessitate recalibration of the DT. To do so, we use the most recent RW observations in $\mathcal{D}_t$ as well as a window of historical data spanning the previous τ_{t-1} time steps, where τ_{t-1} denotes the detection delay. To assess consistency with the past behavior, we construct a prediction set $\hat{C}_t(\mathbf{x}^r_{t,i})$ for each RW response $y_t(\mathbf{x}^r_{t,i})$, for $i = 1, 2, \ldots, b^r$, using the accumulated dataset $\bigcup_{t'=t-\tau_{t-1}}^{t-1} \mathcal{D}_{t'}$, where $\mathcal{D}_{t'} = \left\{ \left(\mathbf{x}^r_{t',i}, y_{t'}(\mathbf{x}^r_{t',i}) \right), i = 1, 2, \ldots, b^r \right\}$, and the new design inputs $\mathbf{x}^r_{t,i}$ for $i = 1, 2, \ldots, b^r$. Given a target miscoverage level $\alpha \in (0, 1)$, we aim to ensure that the true RW system responses lie within their corresponding prediction sets with probability at least $1 - \alpha$. If a significant portion of the new RW system observations fall outside their prediction sets, it may signal a change in the underlying RW system, warranting DT recalibration. Algorithm 3 outlines the full procedure for detecting such changes.

We begin by defining some notation. For any value v_i indexed by $i = 1, 2, \ldots, n$, define $\hat{q}^+_{n,\alpha} :=$ the $\lceil (1-\alpha)(n+1) \rceil$-th smallest value of $v_1, v_2, \ldots, v_n$ as the $(1-\alpha)$-quantile of the empirical distribution of the v_i's. Similarly, let $\hat{q}^-_{n,\alpha}$ denote the α-quantile of the empirical distribution, i.e., $\hat{q}^-_{n,\alpha} :=$ the $\lfloor \alpha(n+1) \rfloor$-th smallest value of $v_1, v_2, \ldots, v_n$. CP relies on the concept of a non-conformity score, which quantifies the discrepancy between the prediction made by a baseline model and the true value $y_t(\mathbf{x}^r_{t,i})$. A higher score indicates a greater discrepancy between the predicted and true values, implying lower conformity. In this work, we use a normalized non-conformity score based on the leave-one-out (LOO) residuals—see, e.g., (Jaber et al. 2025; Papadopoulos 2024) for similar use in GP settings—defined as follows:

$$R^{LOO}_{t',i} := \frac{\left| y_{t'}\left(\mathbf{x}^r_{t',i}\right) - m_{t,-(t',i)}\left(\mathbf{x}^r_{t',i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right) \right|}{\sqrt{\sigma^2_{t'}\left(\mathbf{x}^r_{t',i}\right) + s^2_{t,-(t',i)}\left(\mathbf{x}^r_{t',i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right)}}, \quad t' = t - \tau_{t-1}, \ldots, t-1 \text{ and } i = 1, 2, \ldots, b^r,$$

where $m_{t,-(t',i)}(\cdot,\cdot)$ and $s^2_{t,-(t',i)}(\cdot,\cdot)$ denote the emulator's predictive mean and variance at time t, computed by excluding the training point $\left(\mathbf{x}^r_{t',i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right)$. The term $\sigma^2_{t'}(\mathbf{x}^r_{t',i})$ accounts for the observation noise. To compute residuals, we use the emulator's mean and variance at $\left(\mathbf{x}^r_{t',i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right)$ to predict the RW system response at design input $\mathbf{x}^r_{t',i}$. Since the emulator may not have been trained on this exact point—due to the DT data collection mechanism in Algorithm 2—we identify the nearest training input in $\mathcal{S}_{t-1}$ and exclude it when computing the LOO prediction.

Computing LOO residuals typically requires retraining the emulator multiple times, once for each excluded input point. However, we circumvent this computational burden by using efficient rank-one update formulas for GP modeling, allowing fast LOO residual computation without retraining the GP model. The Jackknife+ prediction interval (Barber et al. 2021) $\hat{C}_t(\mathbf{x}^r_{t,i})$ for the RW system response at design input $\mathbf{x}^r_{t,i}$ is given by

$$\left[\hat{q}^{\pm}_{\tau_{t-1} \times b^r, \alpha} \left\{ m_{t,-(t',i)}\left(\mathbf{x}^r_{t,i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right) \pm R^{LOO}_{t',i} \times \sqrt{\sigma^2_t\left(\mathbf{x}^r_{t,i}\right) + s^2_{t,-(t',i)}\left(\mathbf{x}^r_{t,i}, \hat{\boldsymbol{\theta}}^*_{t-1}\right)} \right\}_{t'=t-\tau_{t-1}, i=1}^{t-1, b^r} \right]. \quad (6)$$

When the RW data generation mechanism evolves over time, conformal inference must be adapted to account for these distributional shifts. To this end, we introduce a time-varying miscoverage sequence α_t,

updated online. This adaptive mechanism monitors the empirical miscoverage frequency of past prediction sets and adjusts α_t accordingly: decreasing the level when the prediction sets under-cover and increasing it when they over-cover. Let α_1 denote the initial miscoverage estimate (we set $\alpha_1 = \alpha$ in our experiments). The update rule at each time step t for each i is given by:

$$\alpha_{t,i+1} = \alpha_{t,i} + \gamma \left(\alpha - \mathbb{1} \left(y_t(\mathbf{x}^r_{t,i}) \notin \hat{C}_t(\mathbf{x}^r_{t,i}) \right) \right) \ , \tag{7}$$

where $\gamma > 0$ is a fixed step size parameter, and $\mathbb{1}(\mathcal{A})$ is the indicator function, returning 1 if event $\mathcal{A}$ is true and 0 otherwise. Intuitively, if the prediction set fails to cover the RW response, $\alpha_{t,i}$ is reduced, widening prediction intervals in subsequent steps.

To detect change points, we monitor prediction interval widths. As the RW system evolves, these intervals widen, signaling increased uncertainty in the predictions. For completeness, suppose that the underlying RW system remains unchanged from time step 1 to $t-1$. To detect an upward shift in the CP interval widths, we use the CUSUM control chart. At time step t, the cumulative sum (CS) is recursively computed as

$$CS_{t'} = \max \left\{ 0, \bar{C}_{t'} - (\bar{c} + s_{\bar{c}}) + CS_{t'-1} \right\} \ , \tag{8}$$

where $CS_0 = 0$, and $t' = 1, 2, \ldots, t$. The term $\bar{C}_{t'}$ denotes the average CP interval width at time t', defined as $\bar{C}_{t'} = (b^r)^{-1} \sum_{i=1}^{b^r} \left| \hat{C}_{t'}(\mathbf{x}^r_{t',i}) \right|$, and the average and standard deviation of widths for all CP intervals constructed up to time t are given by

$$\bar{c} = t^{-1} \sum_{t'=1}^{t} \bar{C}_{t'} \quad \text{and} \quad s_{\bar{c}} = \sqrt{(t-1)^{-1} \sum_{t'=1}^{t} (\bar{C}_{t'} - \bar{c})^2} \ . \tag{9}$$

An alarm is triggered when CS_t exceeds the threshold $\bar{c} + 2s_{\bar{c}}$ (line 4 of Algorithm 3), signaling a significant deviation from historical system behavior. When this occurs, the flag `stateChanged` is set to **true**, and τ_t is reset to the number of consecutive time points with positive CS values (lines 5–6 of Algorithm 3), i.e., the length of the most recent upward excursion of CS_t. If no change is detected, then τ_t is incremented by one, extending the delay window (lines 7–9 of Algorithm 3).

Thus far, we have assumed the observational noise variance $\sigma^2_t(\cdot)$ is known. In practice, it is typically unknown and must be estimated from data. We approximate $\sigma^2_t(\cdot)$ using the sample variance of residuals between observed RW responses and their corresponding emulator predictions. After obtaining the final estimate $\hat{\boldsymbol{\theta}}^*_t$ and the emulator $\eta(\mathbf{x}, \boldsymbol{\theta}) \mid \mathcal{S}_t$ (the last step of Algorithm 2), we compute the residuals $y_{t'}\left(\mathbf{x}^r_{t',i}\right) - m'_t\left(\mathbf{x}^r_{t',i}, \hat{\boldsymbol{\theta}}^*_t\right)$ for $i = 1, 2, \ldots, b^r$ and $t' = t - \tau_t + 1, \ldots, t$. Their sample variance estimates $\sigma^2_t(\cdot)$. While our mathematical formulation in (1) allows the observational noise variance $\sigma^2_t(\cdot)$ to depend on $\mathbf{x}^r$, our current inference assumes that the noise is homoscedastic, i.e., constant across all design inputs at a given time t. This simplifying assumption may be restrictive in scenarios where the observational noise is input-dependent. Inaccurate variance estimates can lead to CP intervals that are either too narrow or too wide, which may misrepresent uncertainty and hinder accurate detection of state changes. Incorporating heteroscedastic noise estimation into our framework is a direction that we are actively pursuing for future work.

5 EXPERIMENTS

This section demonstrates the proposed dynamic calibration framework using a synthetic example. Specifically, we consider the DT model $\eta(x, \theta) = \sin(10x - 5\theta)$, with scalar-valued design input $x \in \mathcal{X} = [0, 1]$ and parameter input $\theta \in \Theta = [0, 1]$. The RW system responses $y_t(x^r_i)$ are generated according to (1). The

RW system undergoes changes in its underlying state at four unknown time points: $t = 15$, 35, 60, and 80. Specifically, the RW system is observed over $T = 100$ time steps, with observations given by:

$$y_t(x^r) = \begin{cases} \eta(x^r, \theta_t^* = 0.1) + \varepsilon, & \varepsilon \sim \mathcal{N}\left(0, 0.2^2\right) \quad \text{if } 0 \le t \le 14 \\ \eta(x^r, \theta_t^* = 0.3) + \varepsilon, & \varepsilon \sim \mathcal{N}\left(0, 0.1^2\right) \quad \text{if } 15 \le t \le 34 \\ \eta(x^r, \theta_t^* = 0.8) + \varepsilon, & \varepsilon \sim \mathcal{N}\left(0, 0.2^2\right) \quad \text{if } 35 \le t \le 59 \\ \eta(x^r, \theta_t^* = 0.5) + \varepsilon, & \varepsilon \sim \mathcal{N}\left(0, 0.1^2\right) \quad \text{if } 60 \le t \le 79 \\ \eta(x^r, \theta_t^* = 0.7) + \varepsilon, & \varepsilon \sim \mathcal{N}\left(0, 0.2^2\right) \quad \text{if } 80 \le t \le 100\ . \end{cases} \tag{10}$$

Algorithm 1 begins with an initial RW sample of size 10, observed at uniformly distributed design points in $\mathcal{X}$ (i.e., $|\mathcal{D}_0| = 10$). The DT data are then collected according to Algorithm 2 to estimate the initial unknown calibration parameter, θ_0^*, based on $\mathcal{D}_0$. At each time step, we receive a batch of $b^r = 3$ RW data points, generated via (10) at design points uniformly sampled from $\mathcal{X}$. When Algorithm 3 detects a state change, we collect $b^s = 50$ new DT data points and estimate the corresponding calibration parameter using Algorithm 2. We use a fixed initial DT sample $\mathcal{S}_{t,0}$ of size $n_0 = 10$, with the input points uniformly drawn from the input space, in Algorithm 2.

Figure 1 shows the CP intervals over time steps $t = 1, 2, \ldots, 38$. From $t = 1$ to $t = 14$, and again from $t = 20$ to $t = 34$, the RW system remains unchanged, during which most RW data points lie within the CP intervals. At $t = 15$ and $t = 35$, the RW system undergoes a change, leading to many RW data points falling outside the CP intervals. As residuals increase after these changes, the CP interval widths also widen in subsequent time steps.

Figure 2(a) shows the CP interval widths over time, corresponding to each batch of $b^r = 3$ new RW data points. Because the interval widths for the three new points at each time step t are similar, the CP intervals appear indistinguishable. Figure 2(b) displays the number of RW data points not covered by the CP interval at each time step. Since we receive $b^r = 3$ new points at each step, the number of uncovered points can be at most three. When the RW system changes but the method has not detected it yet, the number of uncovered points increases, causing wider CP intervals. Once the interval width surpasses the CUSUM threshold, a new state is declared.

Figure 3 illustrates the $b^s = 50$ DT data points collected through our AL procedure, along with the estimated calibration parameters. When a meaningful RW state change is detected, DT data are densely collected around the parameter region of interest, ensuring detailed exploration of the most relevant areas. Simultaneously, the design space is fully covered to ensure accurate estimation of the RW system behavior at any given design input x using the emulator. Although initial parameter estimates deviate from the true region of interest, the estimated parameters $\hat{\theta}_{t,\ell}^*$ converge closely to the true values in the later stages of the AL procedure.

Finally, Figure 4 shows the coverage rate of RW data points over time for different RW batch sizes, $b^r \in \{1, 2, 4, 8\}$. In our framework, larger batch sizes typically result in more robust CP intervals over time, as the RW data points at each time step are used to construct intervals in subsequent steps based on their residuals. Consequently, increasing b^r tends to improve the reliability of the coverage. An exception occurs with $b^r = 1$, where the coverage rate improves after approximately $t = 60$. However, this improvement is misleading: because change points are not accurately detected using this batch size, larger residuals lead to wider CP intervals and artificially high coverage.

6 CONCLUSION

This work introduces a dynamic calibration framework for DTs and outlines several directions for future research. While the data generation model in (1) assumes normally distributed errors, the framework itself is distribution-free and broadly applicable. Future work will explore alternative generation processes, methods for estimating input-dependent noise variance, and applications to real-world case studies. An

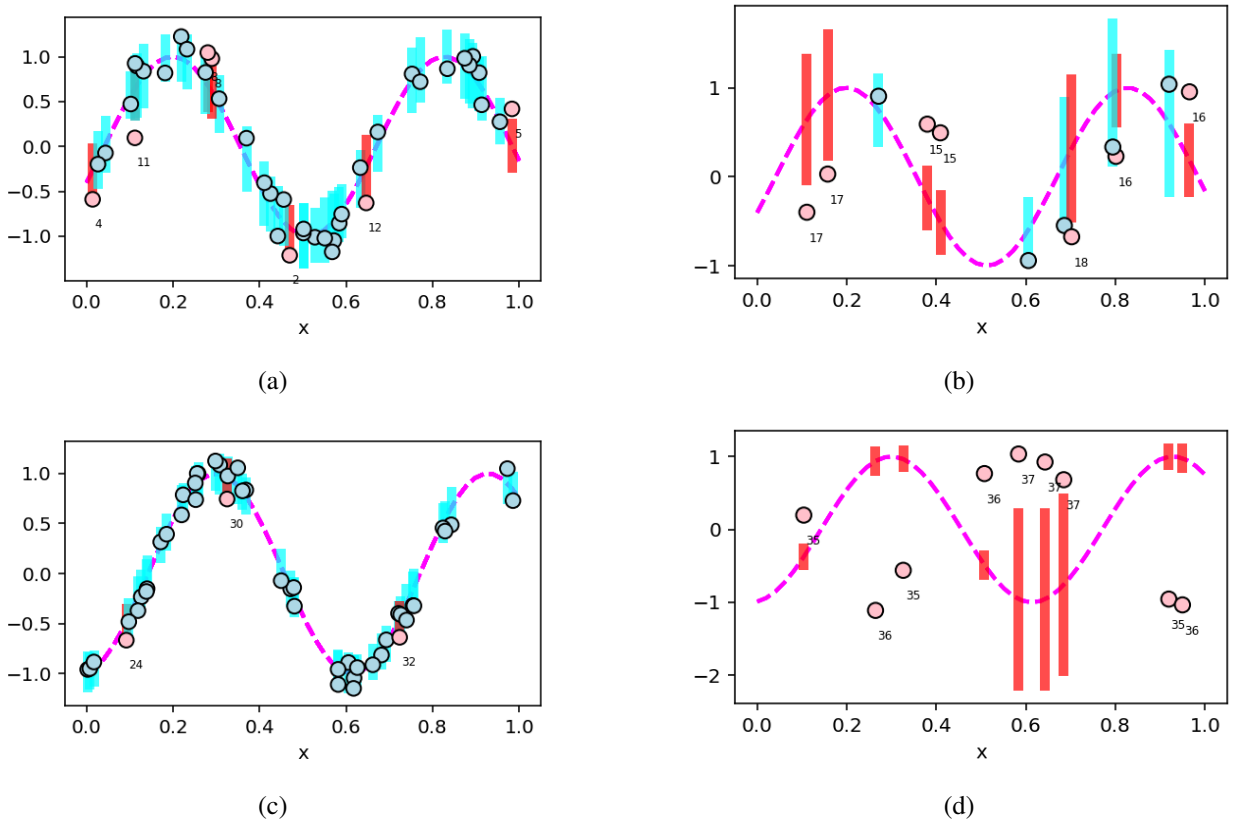


Figure 1: CP intervals obtained at time steps $t = 1, 2, \ldots, 14$ (a), $t = 15, 16, \ldots, 19$ (b), $t = 20, 21, \ldots, 34$ (c), and $t = 35, 36, \ldots, 38$ (d), with each time step receiving $b^r = 3$ RW observations. Circle markers represent the RW data points received, and vertical lines indicate the CP intervals associated with each point at its corresponding time step. Cyan and red indicate whether the RW observation is covered or uncovered, respectively. The dashed magenta line shows the expected RW response value $\mathbb{E}[y_t(x^r)]$ for $0 \leq t \leq 14$ and $15 \leq t \leq 34$ in (a)–(d).

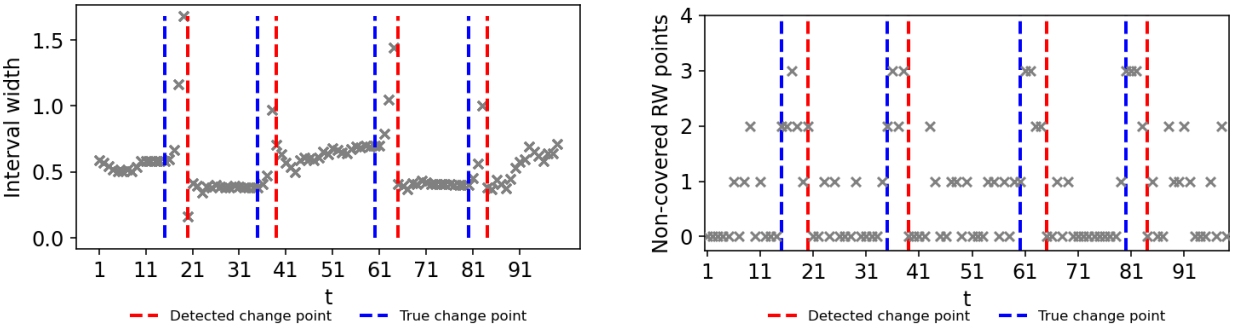
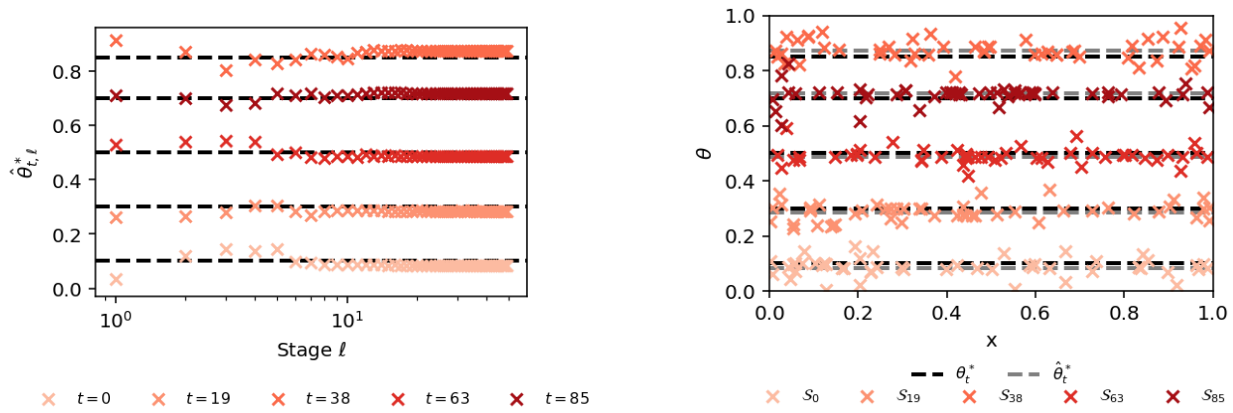
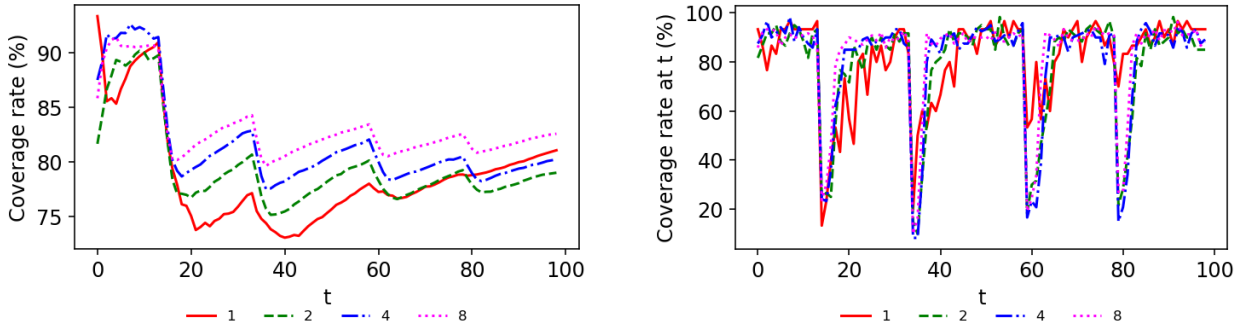


Figure 2: The CP interval widths (a) and the number of RW data points not covered by their respective CP intervals (b).



(a) (b)

Figure 3: The trajectory of the estimated calibration parameter at each stage of the AL procedure (a) and the design and parameter input points selected by the AL procedure (b).



(a) (b)

Figure 4: Illustration of the RW cumulative (a) and per-time-step (b) coverage rates for different batch sizes $b^r \in \{1, 2, 4, 8\}$ across 30 independent macro-replications of the experiment.

important extension is the incorporation of model discrepancy. The KOH framework models the RW system as the sum of the simulation output and a discrepancy term, capturing systematic differences between the model and reality. Despite known identifiability challenges (Plumlee 2017), incorporating a discrepancy component would enable a more comprehensive treatment of uncertainty in simulation-based calibration. Finally, the current DT data collection strategy via AL does not reuse evaluations from previous time steps. Retaining informative data points across time could reduce DT evaluation costs and improve overall efficiency—another promising direction for future work.

ACKNOWLEDGMENTS

The authors acknowledge support by the National Science Foundation under grants OAC-2004601, PHY-2402275, CMMI-1846663, and CMMI-2226347, respectively.

REFERENCES

Angelopoulos, A. N., and S. Bates. 2023, March. "Conformal Prediction: A Gentle Introduction". *Foundations and Trends in Machine Learning* 16(4):494–591 https://doi.org/10.1561/2200000101.

Barber, R. F., E. J. Candès, A. Ramdas, and R. J. Tibshirani. 2021. "Predictive Inference with the Jackknife+". *The Annals of Statistics* 49(1):486–507 https://doi.org/DOI:10.1214/20-AOS1965.

Gibbs, I., and E. Candès. 2021. "Adaptive Conformal Inference Under Distribution Shift". In *Advances in Neural Information Processing Systems*, edited by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, NIPS '21, 1660–1672. Red Hook, NY, USA: Curran Associates Inc.

Gramacy, R. B. 2020. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. New York, NY: CRC Press ; Taylor & Francis Group.

Jaber, E., V. Blot, N. Brunel, V. Chabridon, E. Remy, B. Iooss *et al*. 2025. "Conformal Approach To Gaussian Process Surrogate Evaluation With Marginal Coverage Guarantees". *Journal of Machine Learning for Modeling and Computing* 6(3):37–68.

Jeon, Y., and S. Shashaani. 2024. "Calibrating Digital Twins via Bayesian Optimization with a Root Finding Strategy". In *2024 Winter Simulation Conference (WSC)*, 335–346 https://doi.org/10.1109/WSC63780.2024.10838781.

Kennedy, M. C., and A. O'Hagan. 2001. "Bayesian Calibration of Computer Models". *Journal of the Royal Statistical Society. Series B* 63(3):425–464 https://doi.org/10.1080/00401706.2018.1552203.

Koermer, S., J. Loda, A. Noble, and R. B. Gramacy. 2024. "Augmenting a Simulation Campaign for Hybrid Computer Model and Field Data Experiments". *Technometrics* 66(4):638–650 https://doi.org/10.1080/00401706.2024.2345139.

Page, E. S. 1954, 06. "Continuous Inspection Schemes". *Biometrika* 41(1-2):100–115 https://doi.org/10.1093/biomet/41.1-2.100.

Papadopoulos, H. 2024. "Guaranteed Coverage Prediction Intervals With Gaussian Process Regression". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46(12):9072–9083 https://doi.org/10.1109/TPAMI.2024.3418214.

Plumlee, M. 2017. "Bayesian Calibration of Inexact Computer Models". *Journal of the American Statistical Association* 112(519):1274–1285 https://doi.org/10.1080/01621459.2016.1211016.

Rasmussen, C. E., and C. K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA: The MIT Press.

Santner, T. J., B. J. Williams, and W. I. Notz. 2018. *The Design and Analysis of Computer Experiments*. 2nd ed. 2018 ed. Springer Series in Statistics. New York, NY: Springer New York : Imprint: Springer.

Sung, C.-L., and R. Tuo. 2024. "A Review on Computer Model Calibration". *WIREs Computational Statistics* 16(1):e1645 https://doi.org/https://doi.org/10.1002/wics.1645.

Sürer, O. 2025. "Simulation Experiment Design for Calibration via Active Learning". *Journal of Quality Technology* 57(1):16–34 https://doi.org/10.1080/00224065.2024.2391780.

Sürer, O., M. Plumlee, and S. M. Wild. 2024. "Sequential Bayesian Experimental Design for Calibration of Expensive Simulation Models". *Technometrics* 66(2):157–171 https://doi.org/10.1080/00401706.2023.2246157.

Vovk, V. 2015. "Cross-conformal Predictors". *Annals of Mathematics and Artificial Intelligence* 74:9–28 https://doi.org/10.1007/s10472-013-9368-4.

Xu, J., Q. Lu, and G. B. Giannakis. 2025. "Online Scalable Gaussian Processes with Conformal Prediction for Guaranteed Coverage". In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. April 6th-11th, Hyderabad, India, 1-5. https://doi.org/10.1109/ICASSP49660.2025.10889532.

AUTHOR BIOGRAPHIES

ÖZGE SÜRER is an Assistant Professor of Business Analytics at Miami University. Her research interests include uncertainty quantification, the design and analysis of physical and computer experiments, and computational statistics. Her email address is surero@miamioh.edu and her website is https://ozgesurer.github.io/.

XI CHEN is an Associate Professor in the Grado Department of Industrial and Systems Engineering at Virginia Tech. Her research interests include simulation modeling and analysis, applied probability and statistics, computer experiment design and analysis, and simulation optimization. Her email address is xchen.ise@vt.edu and her webpage is https://sites.google.com/vt.edu/xi-chen-ise/home.

SARA SHASHAANI is an Assistant Professor and Bowman Faculty Scholar in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her research interests include simulation optimization and uncertainty quantification in data-driven modeling and prediction. She is a co-creator of SimOpt. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.